

calibre User Manual

Release 0.9.22

Kovid Goyal

March 08, 2013

Contents

1 Sections **3**

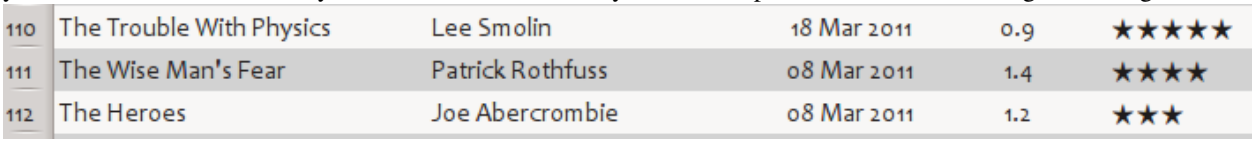
1.1 The Graphical User Interface . . . 3
1.2 Adding your favorite news website . . . 17
1.3 The Ebook Viewer . . . 44
1.4 Ebook Conversion . . . 47
1.5 Editing Ebook Metadata . . . 60
1.6 Frequently Asked Questions . . . 63
1.7 Tutorials . . . 83
1.8 Customizing calibre . . . 182
1.9 Command Line Interface . . . 213
1.10 Setting up a calibre development environment . . . 254
1.11 Glossary . . . 260
1.12 The main calibre user interface . . . 261
1.13 Adding your favorite news website to calibre . . . 275
1.14 The calibre ebook viewer . . . 302
1.15 Customizing calibre‘s ebook conversion . . . 305
1.16 Editing ebook metadata . . . 318
1.17 Frequently Asked Questions . . . 321
1.18 Tutorials . . . 341
1.19 Customizing calibre . . . 440
1.20 The Command Line Interface . . . 472
1.21 Setting up a calibre development environment . . . 511

Python Module Index **517**

Index **519**

calibre is an ebook library manager. It can view, convert and catalog ebooks in most of the major ebook formats. It can also talk to many ebook reader devices. It can go out to the Internet and fetch metadata for your books. It can download newspapers and convert them into ebooks for convenient reading. It is cross platform, running on Linux, Windows and OS X.

You've just started calibre. What do you do now? Before calibre can do anything with your ebooks, it first has to know about them. Drag and drop a few ebook files into calibre, or click the "Add books" button and browse for the ebooks you want to work with. Once you've added the books, they will show up in the main view looking something like this:



110	The Trouble With Physics	Lee Smolin	18 Mar 2011	0.9	★★★★★
111	The Wise Man's Fear	Patrick Rothfuss	08 Mar 2011	1.4	★★★★
112	The Heroes	Joe Abercrombie	08 Mar 2011	1.2	★★★

Once you've admired the list of books you just added to your heart's content, you'll probably want to read one. In order to do that you'll have to convert the book to a format your reader understands. When first running calibre, the Welcome Wizard starts and will set up calibre for your reader device. Conversion is a breeze. Just select the book you want to convert then click the "Convert books" button. Ignore all the options for now and click "OK". The little icon in the bottom right corner will start spinning. Once it's finished spinning, your converted book is ready. Click the "View" button to read the book.

If you want to read the book on your reader, connect it to the computer, wait till calibre detects it (10-20 seconds) and then click the "Send to device" button. Once the icon stops spinning again, disconnect your reader and read away! If you didn't convert the book in the previous step, calibre will auto convert it to the format your reader device understands.

To get started with more advanced usage, you should read about the *Graphical User Interface* (page 261). For even more power and versatility, learn the *Command Line Interface* (page 472). You will find the list of *Frequently Asked Questions* (page 321) useful as well.

CHAPTER 1

Sections

1.1 The Graphical User Interface

The Graphical User Interface *(GUI)* provides access to all library management and ebook format conversion features. The basic workflow for using calibre is to first add books to the library from your hard disk. calibre will automatically try to read metadata from the books and add them to its internal database. Once they are in the database, you can perform various *Actions* (page 261) on them that include conversion from one format to another, transfer to the reading device, viewing on your computer, and editing metadata. The latter includes modifying the cover, description, and tags among other details. Note that calibre creates copies of the files you add to it. Your original files are left untouched.

The interface is divided into various sections:

- Actions (page 261)
- Preferences (page 267)
- Catalogs (page 268)
- Search & Sort (page 268)
- The Search Interface (page 269)
- Saving searches (page 271)
- Book Details (page 272)
- Tag Browser (page 273)
- Jobs (page 274)
- Keyboard Shortcuts (page 274)

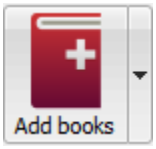
1.1.1 Actions



The actions toolbar provides convenient shortcuts to commonly used actions. If you right-click the buttons, you can perform variations on the default action. Please note that the actions toolbar will look slightly different depending on whether you have an ebook reader attached to your computer.

- Add books (page 262)
- Edit metadata (page 263)
- Convert books (page 263)
- View (page 264)
- Send to device (page 264)
- Fetch news (page 264)
- Library (page 265)
- Device (page 265)
- Save to disk (page 266)
- Connect/Share (page 266)
- Remove books (page 267)

Add books



The *Add books* action has six variations accessed by doing a right-click on the button.

1. **Add books from a single directory**: Opens a file chooser dialog and allows you to specify which books in a directory should be added. This action is *context sensitive*, i.e. it depends on which *catalog* (page 268) you have selected. If you have selected the *Library*, books will be added to the library. If you have selected the ebook reader device, the books will be uploaded to the device, and so on.
2. **Add books from directories, including sub-directories (One book per directory, assumes every ebook file is the same book in a different format)**: Allows you to choose a directory. The directory and all its sub-directories are scanned recursively, and any ebooks found are added to the library. calibre assumes that each directory contains a single book. All ebook files in a directory are assumed to be the same book in different formats. This action is the inverse of the *Save to disk* (page 266) action, i.e. you can *Save to disk*, delete the books and re-add them with no lost information except for the date (this assumes you have not changed any of the setting for the Save to disk action).
3. **Add books from directories, including sub-directories (Multiple books per directory, assumes every ebook file is a different book)**: Allows you to choose a directory. The directory and all its sub-directories are scanned recursively and any ebooks found are added to the library. calibre assumes that each directory contains many books. All ebook files with the same name in a directory are assumed to be the same book in different formats. Ebooks with different names are added as different books.
4. **Add empty book. (Book Entry with no formats)**: Allows you to create a blank book record. This can be used to then manually fill out the information about a book that you may not have yet in your collection.
5. **Add from ISBN**: Allows you to add one or more books by entering their ISBNs.
6. **Add files to selected book records**: Allows you to add or update the files associated with an existing book in your library.

The *Add books* action can read metadata from a wide variety of ebook formats. In addition, it tries to guess metadata from the filename. See the *Guessing metadata from file names* (page 271) section, to learn how to configure this.

To add an additional format for an existing book use the *Edit metadata* (page 263) action.

Edit metadata



The *Edit metadata* action has four variations which can be accessed by doing a right-click on the button.

1. **Edit metadata individually**: Allows you to edit the metadata of books one-by-one with the option of fetching metadata, including covers, from the Internet. It also allows you to add or remove particular ebook formats from a book.
2. **Edit metadata in bulk**: Allows you to edit common metadata fields for large numbers of books simulataneously. It operates on all the books you have selected in the *Library view* (page 268).
3. **Download metadata and covers**: Downloads metadata and covers (if available) for the books that are selected in the book list.
4. **Merge book records**: Gives you the capability of merging the metadata and formats of two or more book records. You can choose to either delete or keep the records that were not clicked first.

For more details see *Editing Ebook Metadata* (page 318).

Convert books



Ebooks can be converted from a number of formats into whatever format your ebook reader prefers. Many ebooks available for purchase will be protected by Digital Rights Management[1] *(DRM)* technology. calibre will not convert these ebooks. It is easy to remove the DRM from many formats, but as this may be illegal, you will have to find tools to liberate your books yourself and then use calibre to convert them.

For most people, conversion should be a simple one-click affair. If you want to learn more about the conversion process, see *Ebook Conversion* (page 305).

The *Convert books* action has three variations, accessed by doing a right-click on the button.

1. **Convert individually**: Allows you to specify conversion options to customize the conversion of each selected ebook.
2. **Bulk convert**: Allows you to specify options only once to convert a number of ebooks in bulk.
3. **Create a catalog of the books in your calibre library**: Allows you to generate a complete listing of the books in your library, including all metadata, in several formats such as XML, CSV, BiBTeX, EPUB and MOBI. The catalog will contain all the books currently showing in the library view. This allows you to use the search features to limit the books to be catalogued. In addition, if you select multiple books using the mouse, only those books will be added to the catalog. If you generate the catalog in an ebook format such as EPUB, MOBI or AZW3, the next time you connect your ebook reader the catalog will be automatically sent to the device. For more information on how catalogs work, read the *Creating AZW3 • EPUB • MOBI Catalogs* (page 436).

[1] http://drmfree.calibre-ebook.com/about#drm

View



The *View* action displays the book in an ebook viewer program. calibre has a built-in viewer for many ebook formats. For other formats it uses the default operating system application. You can configure which formats should open with the internal viewer via Preferences->Behavior. If a book has more than one format, you can view a particular format by doing a right-click on the button.

Send to device



The *Send to device* action has eight variations, accessed by doing a right-click on the button.

1. **Send to main memory**: The selected books are transferred to the main memory of the ebook reader.
2. **Send to card (A)**: The selected books are transferred to the storage card (A) on the ebook reader.
3. **Send to card (B)**: The selected books are transferred to the storage card (B) on the ebook reader.
4. **Send specific format to**: The selected books are transferred to the selected storage location on the device, in the format that you specify.
5. **Eject device**: Detaches the device from calibre.
6. **Set default send to device action**: Allows you to specify which of the options, 1 through 5 above or 7 below, will be the default action when you click the main button.
7. **Send and delete from library**: The selected books are transferred to the selected storage location on the device and then **deleted** from the Library.
8. **Fetch Annotations (experimental)**: Transfers annotations you may have made on an ebook on your device to the comments metadata of the book in the calibre library.

You can control the file name and folder structure of files sent to the device by setting up a template in *Preferences->Import/Export->Sending books to devices*. Also see *The calibre template language* (page 375).

Fetch news



The *Fetch news* action downloads news from various websites and converts it into an ebook that can be read on your ebook reader. Normally, the newly created ebook is added to your ebook library, but if an ebook reader is connected at the time the download finishes, the news is also uploaded to the reader automatically.

The *Fetch news* action uses simple recipes (10-15 lines of code) for each news site. To learn how to create recipes for your own news sources, see *Adding your favorite news website* (page 341).

The *Fetch news* action has three variations, accessed by doing a right-click on the button.

1. **Schedule news download**: Allows you to schedule the download of of your selected news sources from a list of hundreds available. Scheduling can be set individually for each news source you select and the scheduling is flexible allowing you to select specific days of the week or a frequency of days between downloads.
2. **Add a custom news source**: Allows you to create a simple recipe for downloading news from a custom news site that you wish to access. Creating the recipe can be as simple as specifying an RSS news feed URL, or you can be more prescriptive by creating Python-based code for the task. For more information see *Adding your favorite news website* (page 341).
3. **Download all scheduled news sources**: Causes calibre to immediately begin downloading all news sources that you have scheduled.

Library



The *Library* action allows you to create, switch between, rename or remove a Library. calibre allows you to create as many libraries as you wish. You could, for instance, create a fiction library, a non-fiction library, a foreign language library, a project library, or any structure that suits your needs. Libraries are the highest organizational structure within calibre. Each library has its own set of books, tags, categories and base storage location.

1. **Switch/create library...**: Allows you to; a) connect to a pre-existing calibre library at another location, b) create an empty library at a new location or, c) move the current library to a newly specified location.
2. **Quick switch**: Allows you to switch between libraries that have been registered or created within calibre.
3. **Rename library**: Allows you to rename a Library.
4. **Delete library**: Allows you to unregister a library from calibre.
5. **<library name>**: Actions 5, 6 etc... give you immediate switch access between multiple libraries that you have created or attached to. This list contains only the 5 most frequently used libraries. For the complete list, use the Quick Switch menu.
6. **Library maintenance**: Allows you to check the current library for data consistency issues and restore the current library's database from backups.

Note: Metadata about your ebooks, e.g. title, author, and tags, is stored in a single file in your calibre library folder called metadata.db. If this file gets corrupted (a very rare event), you can lose the metadata. Fortunately, calibre automatically backs up the metadata for every individual book in the book's folder as an OPF file. By using the Restore Library action under Library Maintenance described above, you can have calibre rebuild the metadata.db file from the individual OPF files for you.

You can copy or move books between different libraries (once you have more than one library setup) by right clicking on the book and selecting the action *Copy to library*.

Device



The *Device* action allows you to view the books in the main memory or storage cards of your device, or to eject the device (detach it from calibre). This icon shows up automatically on the main calibre toolbar when you

connect a supported device. You can click on it to see the books on your device. You can also drag and drop books from your calibre library onto the icon to transfer them to your device. Conversely, you can drag and drop books from your device onto the library icon on the toolbar to transfer books from your device to the calibre library.

Save to disk



The *Save to disk* action has five variations, accessed by doing a right-click on the button.

1. **Save to disk**: Saves the selected books to disk organized in directories. The directory structure looks like:

```
Author_(sort)
    Title
        Book Files
```

You can control the file name and folder structure of files saved to disk by setting up a template in *Preferences->Import/Export->Saving books to disk*. Also see *The calibre template language* (page 375).

2. **Save to disk in a single directory**: Saves the selected books to disk in a single directory.

For 1. and 2., all available formats, as well as metadata, are stored to disk for each selected book. Metadata is stored in an OPF file. Saved books can be re-imported to the library without any loss of information by using the *Add books* (page 262) action.

3. **Save only *<your preferred>* format to disk**: Saves the selected books to disk in the directory structure as shown in (1.) but only in your preferred ebook format. You can set your preferred format in *Preferences->Behaviour->Preferred output format*

4. **Save only *<your preferred>* format to disk in a single directory**: Saves the selected books to disk in a single directory but only in your preferred ebook format. You can set your preferred format in *Preferences->Behaviour->Preferred output format*

5. **Save single format to disk...**: Saves the selected books to disk in the directory structure as shown in (1.) but only in the format you select from the pop-out list.

Connect/Share



The *Connect/Share* action allows you to manually connect to a device or folder on your computer. It also allows you to set up you calibre library for access via a web browser or email.

The *Connect/Share* action has four variations, accessed by doing a right-click on the button.

1. **Connect to folder**: Allows you to connect to any folder on your computer as though it were a device and use all the facilities calibre has for devices with that folder. Useful if your device cannot be supported by calibre but is available as a USB disk.

2. **Connect to iTunes**: Allows you to connect to your iTunes books database as though it were a device. Once the books are sent to iTunes, you can use iTunes to make them available to your various iDevices.

3. **Start Content Server**: Starts calibre's built-in web server. When started, your calibre library will be accessible via a web browser from the Internet (if you choose). You can configure how the web server is accessed by setting preferences at *Preferences->Sharing->Sharing over the net*
4. **Setup email based sharing of books**: Allows sharing of books and news feeds by email. After setting up email addresses for this option, calibre will send news updates and book updates to the entered email addresses. You can configure how calibre sends email by setting preferences at *Preferences->Sharing->Sharing books by email*. Once you have set up one or more email addresses, this menu entry will be replaced by menu entries to send books to the configured email addresses.

Remove books



The *Remove books* action **deletes books permanently**, so use it with care. It is *context sensitive*, i.e. it depends on which *catalog* (page 268) you have selected. If you have selected the *Library*, books will be removed from the library. If you have selected the ebook reader device, books will be removed from the device. To remove only a particular format for a given book use the *Edit metadata* (page 263) action. Remove books also has five variations which can be accessed by doing a right-click on the button.

1. **Remove selected books**: Allows you to **permanently** remove all books that are selected in the book list.
2. **Remove files of a specific format from selected books...**: Allows you to **permanently** remove ebook files of a specified format from books that are selected in the book list.
3. **Remove all formats from selected books, except...**: Allows you to **permanently** remove ebook files of any format except a specified format from books that are selected in the book list.
3. **Remove all formats from selected books**: Allows you to **permanently** remove all ebook files from books that are selected in the book list. Only the metadata will remain.
4. **Remove covers from selected books**: Allows you to **permanently** remove cover image files from books that are selected in the book list.
5. **Remove matching books from device**: Allows you to remove ebook files from a connected device that match the books that are selected in the book list.

Note: Note that when you use Remove books to delete books from your calibre library, the book record is permanently deleted, but on Windows and OS X the files are placed into the recycle bin. This allows you to recover them if you change your mind.

1.1.2 Preferences

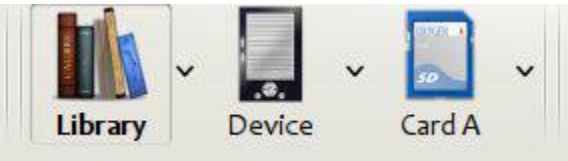


The *Preferences* action allows you to change the way various aspects of calibre work. It has four variations, accessed by doing a right-click on the button.

1. **Preferences**: Allows you to change the way various aspects of calibre work. Clicking the button also performs this action.

2. **Run welcome wizard**: Allows you to start the Welcome Wizard which appeared the first time you started calibre.
3. **Get plugins to enhance |app|**: Opens a new windows that shows plugins for calibre. These plugins are developed by third parties to extend calibre's functionality.
4. **Restart in debug mode**: Allows you to enable a debugging mode that can assist the calibre developers in solving problems you encounter with the program. For most users this should remain disabled unless instructed by a developer to enable it.

1.1.3 Catalogs



A *catalog* is a collection of books. calibre can manage two types of different catalogs:

1. **Library**: This is a collection of books stored in your calibre library on your computer.
2. **Device**: This is a collection of books stored in your ebook reader. It will be available when you connect the reader to your computer.

Many operations, such as adding books, deleting, viewing, etc., are context sensitive. So, for example, if you click the View button when you have the **Device** catalog selected, calibre will open the files on the device to view. If you have the **Library** catalog selected, files in your calibre library will be opened instead.

1.1.4 Search & Sort



The Search & Sort section allows you to perform several powerful actions on your book collections.

- You can sort them by title, author, date, rating, etc. by clicking on the column titles. You can also sub-sort, i.e. sort on multiple columns. For example, if you click on the title column and then the author column, the book will be sorted by author and then all the entries for the same author will be sorted by title.

- You can search for a particular book or set of books using the search bar. More on that below.
- You can quickly and conveniently edit metadata by double-clicking the entry you want changed in the list.
- You can perform *Actions* (page 261) on sets to books. To select multiple books you can either:
  - Keep the `Ctrl` key pressed and click on the books you want selected.
  - Keep the `Shift` key pressed and click on the starting and ending book of a range of books you want selected.
- You can configure which fields you want displayed by using the *Preferences* (page 267) dialog.

1.1.5 The Search Interface

You can search all the metadata by entering search terms in the search bar. Searches are case insensitive. For example:

```
Asimov Foundation format:lrf
```

This will match all books in your library that have `Asimov` and `Foundation` in their metadata and are available in the LRF format. Some more examples:

```
author:Asimov and not series:Foundation
title:"The Ring" or "This book is about a ring"
format:epub publisher:feedbooks.com
```

Searches are by default 'contains'. An item matches if the search string appears anywhere in the indicated metadata. Two other kinds of searches are available: equality search and search using regular expressions[2].

Equality searches are indicated by prefixing the search string with an equals sign (=). For example, the query `tag:"=science"` will match "science", but not "science fiction" or "hard science". Regular expression searches are indicated by prefixing the search string with a tilde (~). Any python-compatible regular expression[3] can be used. Note that backslashes used to escape special characters in requlars expressions must be doubled because single backslashes will be removed during query parsing. For example, to match a literal parenthesis you must enter `\\(`. Regular expression searches are 'contains' searches unless the expression contains anchors.

Should you need to search for a string with a leading equals or tilde, prefix the string with a backslash.

Enclose search strings with quotes (") if the string contains parenthesis or spaces. For example, to search for the tag `Science Fiction` you would need to search for `tag:"=science fiction"`. If you search for `tag:=science fiction` you will find all books with the tag 'science' and containing the word 'fiction' in any metadata.

You can build advanced search queries easily using the *Advanced Search Dialog* accessed by clicking the button .

Available fields for searching are: `tag`, `title`, `author`, `publisher`, `series`, `series_index`, `rating`, `cover`, `comments`, `format`, `identifiers`, `date`, `pubdate`, `search`, `size` and custom columns. If a device is plugged in, the `ondevice` field becomes available, when searching the calibre library view. To find the search name (actually called the *lookup name*) for a custom column, hover your mouse over the column header in the library view.

The syntax for searching for dates is:

```
pubdate:>2000-1 Will find all books published after Jan, 2000
date:<=2000-1-3 Will find all books added to calibre before 3 Jan, 2000
pubdate:=2009 Will find all books published in 2009
```

[2] http://en.wikipedia.org/wiki/Regular_expression
[3] http://docs.python.org/library/re.html

If the date is ambiguous, the current locale is used for date comparison. For example, in an mm/dd/yyyy locale 2/1/2009 is interpreted as 1 Feb 2009. In a dd/mm/yyyy locale it is interpreted as 2 Jan 2009. Some special date strings are available. The string `today` translates to today's date, whatever it is. The strings `yesterday` and `thismonth` (or the translated equivalent in the current language) also work. In addition, the string `daysago` (also translated) can be used to compare to a date some number of days ago. For example:

```
date:>10daysago
date:<=45daysago
```

You can search for books that have a format of a certain size like this:

```
size:>1.1M Will find books with a format larger than 1.1MB
size:<=1K  Will find books with a format smaller than 1KB
```

Dates and numeric fields support the relational operators = (equals), > (greater than), >= (greater than or equal to), < (less than), <= (less than or equal to), and != (not equal to). Rating fields are considered to be numeric. For example, the search `rating:>=3` will find all books rated 3 or higher.

You can search for the number of items in multiple-valued fields such as tags. These searches begin with the character `#`, then use the same syntax as numeric fields. For example, to find all books with more than 4 tags use `tags:#>4`. To find all books with exactly 10 tags use `tags:#=10`.

Series indices are searchable. For the standard series, the search name is 'series_index'. For custom series columns, use the column search name followed by _index. For example, to search the indices for a custom series column named `#my_series`, you would use the search name `#my_series_index`. Series indices are numbers, so you can use the relational operators described above.

The special field `search` is used for saved searches. So if you save a search with the name "My spouse's books" you can enter `search:"My spouse's books"` in the search bar to reuse the saved search. More about saving searches below.

You can search for the absence or presence of a field using the special "true" and "false" values. For example:

```
cover:false will give you all books without a cover
series:true will give you all books that belong to a series
comments:false will give you all books with an empty comment
format:false will give you all books with no actual files (empty records)
```

Yes/no custom columns are searchable. Searching for `false`, `empty`, or `blank` will find all books with undefined values in the column. Searching for `true` will find all books that do not have undefined values in the column. Searching for `yes` or `checked` will find all books with `Yes` in the column. Searching for `no` or `unchecked` will find all books with `No` in the column. Note that the words `yes`, `no`, `blank`, `empty`, `checked` and `unchecked` are translated; you must use the current language's equivalent word. The words `true` and `false` and the special values `_yes` and `_no` are not translated.

Hierarchical items (e.g. A.B.C) use an extended syntax to match initial parts of the hierarchy. This is done by adding a period between the exact match indicator (=) and the text. For example, the query `tags:=.A` will find the tags *A* and *A.B*, but will not find the tags *AA* or *AA.B*. The query `tags:=.A.B` will find the tags *A.B* and *A.B.C*, but not the tag *A*.

Identifiers (e.g., isbn, doi, lccn etc) also use an extended syntax. First, note that an identifier has the form `type:value`, as in `isbn:123456789`. The extended syntax permits you to specify independently which type and value to search for. Both the type and the value parts of the query can use *equality*, *contains*, or *regular expression* matches. Examples:

- `identifiers:true` will find books with any identifier.
- `identifiers:false` will find books with no identifier.
- `identifiers:123` will search for books with any type having a value containing *123*.

- `identifiers:=123456789` will search for books with any type having a value equal to *123456789*.
- `identifiers:=isbn:` and `identifiers:isbn:true` will find books with a type equal to isbn having any value
- `identifiers:=isbn:false` will find books with no type equal to isbn.
- `identifiers:=isbn:123` will find books with a type equal to isbn having a value containing *123*.
- `identifiers:=isbn:=123456789` will find books with a type equal to isbn having a value equal to *123456789*.
- `identifiers:i:1` will find books with a type containing an *i* having a value containing a *1*.

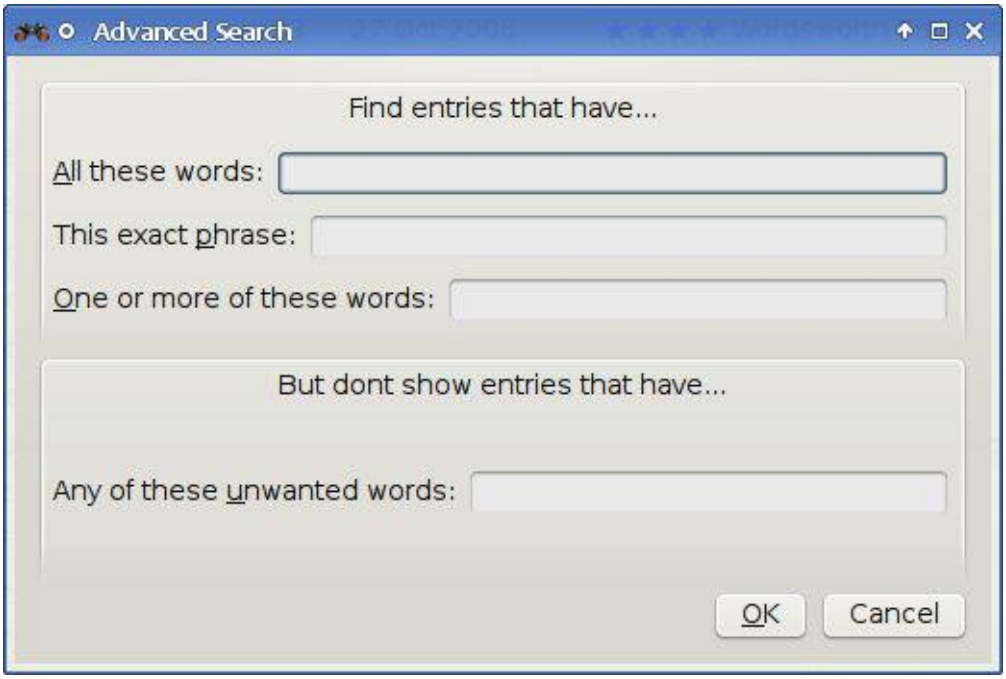


Figure 1.1: *Advanced Search Dialog*

1.1.6 Saving searches

calibre allows you to save a frequently used search under a special name and then reuse that search with a single click. To do this, create your search either by typing it in the search bar or using the Tag Browser. Then type the name you would like to give to the search in the Saved Searches box next to the search bar. Click the plus icon next to the saved searches box to save the search.

Now you can access your saved search in the Tag Browser under "Searches". A single click will allow you to reuse any arbitrarily complex search easily, without needing to re-create it.

Guessing metadata from file names

In the *Add/Save* section of the configuration dialog, you can specify a regular expression that calibre will use to try and guess metadata from the names of ebook files that you add to the library. The default regular expression is:

```
title - author
```

i.e., it assumes that all characters up to the first – are the title of the book and subsequent characters are the author of the book. For example, the filename:

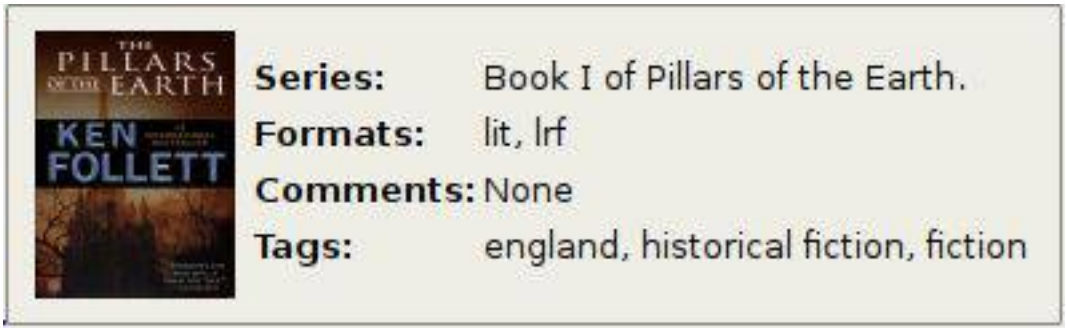
```
Foundation and Earth - Isaac Asimov.txt
```

will be interpreted to have the title: Foundation and Earth and author: Isaac Asimov

---

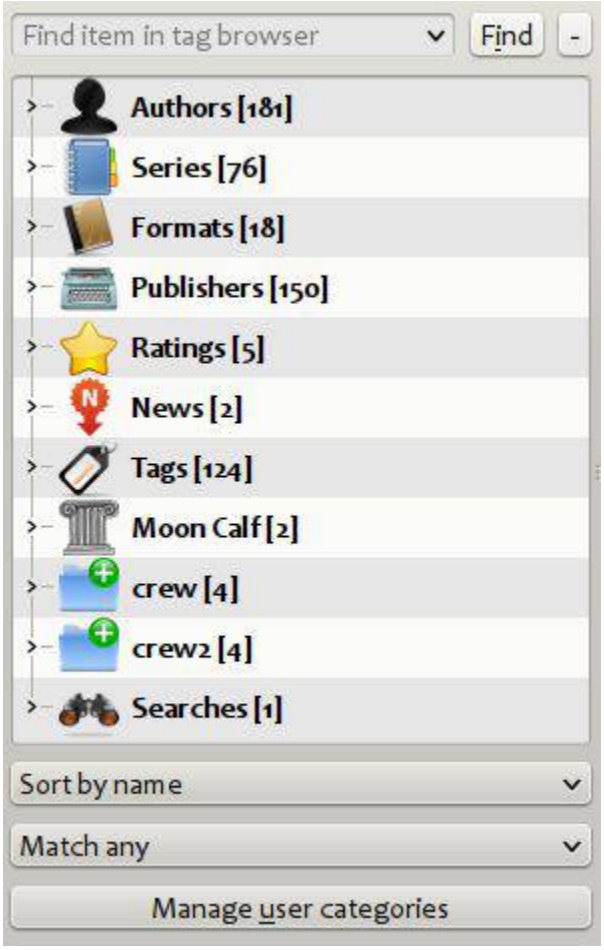
Tip: If the filename does not contain the hyphen, the regular expression will fail.

1.1.7 Book Details



The Book Details display shows extra information and the cover for the currently selected book.

1.1.8 Tag Browser



The Tag Browser allows you to easily browse your collection by Author/Tags/Series/etc. If you click on any item in the Tag Browser, for example the author name Isaac Asimov, then the list of books to the right is restricted to showing books by that author. You can click on category names as well. For example, clicking on "Series" will show you all books in any series.

The first click on an item will restrict the list of books to those that contain or match the item. Continuing the above example, clicking on Isaac Asimov will show books by that author. Clicking again on the item will change what is shown, depending on whether the item has children (see sub-categories and hierarchical items below). Continuing the Isaac Asimov example, clicking again on Isaac Asimov will restrict the list of books to those not by Isaac Asimov. A third click will remove the restriction, showing all books. If you hold down the Ctrl or Shift keys and click on multiple items, then restrictions based on multiple items are created. For example you could hold Ctrl and click on the tags History and Europe for finding books on European history. The Tag Browser works by constructing search expressions that are automatically entered into the Search bar. Looking at what the Tag Browser generates is a good way to learn how to construct basic search expressions.

Items in the Tag browser have their icons partially colored. The amount of color depends on the average rating of the books in that category. So for example if the books by Isaac Asimov have an average of four stars, the icon for Isaac Asimov in the Tag Browser will be 4/5th colored. You can hover your mouse over the icon to see the average rating.

The outer-level items in the tag browser, such as Authors and Series, are called categories. You can create your own categories, called User Categories, which are useful for organizing items. For example, you can use the User Categories Editor (click the Manage User Categories button) to create a user category called Favorite Authors, then put the items for your favorites into the category. User categories can have sub-categories. For example, the user category

Favorites.Authors is a sub-category of Favorites. You might also have Favorites.Series, in which case there will be two sub-categories under Favorites. Sub-categories can be created by right-clicking on a user category, choosing "Add sub-category to ...", and entering the sub-category name; or by using the User Categories Editor by entering names like the Favorites example above.

You can search user categories in the same way as built-in categories, by clicking on them. There are four different searches cyc

1. "everything matching an item in the category" indicated by a single green plus sign.
2. "everything matching an item in the category or its sub-categories" indicated by two green plus signs.
3. "everything not matching an item in the category" shown by a single red minus sign.
4. "everything not matching an item in the category or its sub-categories" shown by two red minus signs.

It is also possible to create hierarchies inside some of the text categories such as tags, series, and custom columns. These hierarchies show with the small triangle, permitting the sub-items to be hidden. To use hierarchies of items in a category, you must first go to Preferences->Look & Feel and enter the category name(s) into the "Categories with hierarchical items" box. Once this is done, items in that category that contain periods will be shown using the small triangle. For example, assume you create a custom column called "Genre" and indicate that it contains hierarchical items. Once done, items such as Mystery.Thriller and Mystery.English will display as Mystery with the small triangle next to it. Clicking on the triangle will show Thriller and English as sub-items. See *Managing subgroups of books, for example "genre"* (page 368) for more information.

Hierarchical items (items with children) use the same four 'click-on' searches as user categories. Items that do not have children use two of the searches: "everything matching" and "everything not matching".

You can drag and drop items in the Tag browser onto user categories to add them to that category. If the source is a user category, holding the shift key while dragging will move the item to the new category. You can also drag and drop books from the book list onto items in the Tag Browser; dropping a book on an item causes that item to be automatically applied to the dropped books. For example, dragging a book onto Isaac Asimov will set the author of that book to Isaac Asimov. Dropping it onto the tag History will add the tag History to the book's tags.

There is a search bar at the top of the Tag Browser that allows you to easily find any item in the Tag Browser. In addition, you can right click on any item and choose one of several operations. Some examples are to hide the it, rename it, or open a "Manage x" dialog that allows you to manage items of that kind. For example, the "Manage Authors" dialog allows you to rename authors and control how their names are sorted.

You can control how items are sorted in the Tag browser via the box at the bottom of the Tag Browser. You can choose to sort by name, average rating or popularity (popularity is the number of books with an item in your library; for example, the popularity of Isaac Asimov is the number of books in your library by Isaac Asimov).

1.1.9 Jobs



The Jobs panel shows the number of currently running jobs. Jobs are tasks that run in a separate process. They include converting ebooks and talking to your reader device. You can click on the jobs panel to access the list of jobs. Once a job has completed you can see a detailed log from that job by double-clicking it in the list. This is useful to debug jobs that may not have completed successfully.

1.1.10 Keyboard Shortcuts

Calibre has several keyboard shortcuts to save you time and mouse movement. These shortcuts are active in the book list view (when you're not editing the details of a particular book), and most of them affect the title you have selected.

The calibre ebook viewer has its own shortcuts which can be customised by clicking the Preferences button in the viewer.

Note: Note: The Calibre keyboard shortcuts do not require a modifier key (Command, Option, Control, etc.), unless specifically noted. You only need to press the letter key, e.g. E to edit.

Table 1.1: Keyboard Shortcuts

Keyboard Shortcut	Action
F2 (Enter in OS X)	Edit the metadata of the currently selected field in the book list.
A	Add Books
Shift+A	Add Formats to the selected books
C	Convert selected Books
D	Send to device
Del	Remove selected Books
E	Edit metadata of selected books
G	Get Books
I	Show book details
M	Merge selected records
Alt+M	Merge selected records, keeping originals
O	Open containing folder
P	Polish books
S	Save to Disk
V	View
Alt+V/Cmd+V in OS X	View specific format
Alt+Shift+J	Toggle jobs list
Alt+Shift+B	Toggle Cover Browser
Alt+Shift+D	Toggle Book Details panel
Alt+Shift+T	Toggle Tag Browser
Alt+A	Show books by the same author as the current book
Alt+T	Show books with the same tags as current book
Alt+P	Show books by the same publisher as current book
Alt+Shift+S	Show books in the same series as current book
/, Ctrl+F	Focus the search bar
Shift+Ctrl+F	Open the advanced search dialog
Esc	Clear the current search
N or F3	Find the next book that matches the current search (only works if the highlight checkbox next to the se
Shift+N or Shift+F3	Find the next book that matches the current search (only works if the highlight checkbox next to the se
Ctrl+D	Download metadata and shortcuts
Ctrl+R	Restart calibre
Ctrl+Shift+R	Restart calibre in debug mode
Shift+Ctrl+E	Add empty books to calibre
Ctrl+Q	Quit calibre

1.2 Adding your favorite news website

calibre has a powerful, flexible and easy-to-use framework for downloading news from the Internet and converting it into an ebook. The following will show you, by means of examples, how to get news from various websites.

To gain an understanding of how to use the framework, follow the examples in the order listed below:

- Completely automatic fetching (page 342)
  - portfolio.com (page 342)
  - bbc.co.uk (page 344)
- Customizing the fetch process (page 344)
  - Using the print version of bbc.co.uk (page 344)
  - Replacing article styles (page 345)
  - Slicing and dicing (page 346)
  - Real life example (page 357)
- Tips for developing new recipes (page 359)
- Further reading (page 360)
- API documentation (page 360)

1.2.1 Completely automatic fetching

If your news source is simple enough, calibre may well be able to fetch it completely automatically, all you need to do is provide the URL. calibre gathers all the information needed to download a news source into a *recipe*. In order to tell calibre about a news source, you have to create a *recipe* for it. Let's see some examples:

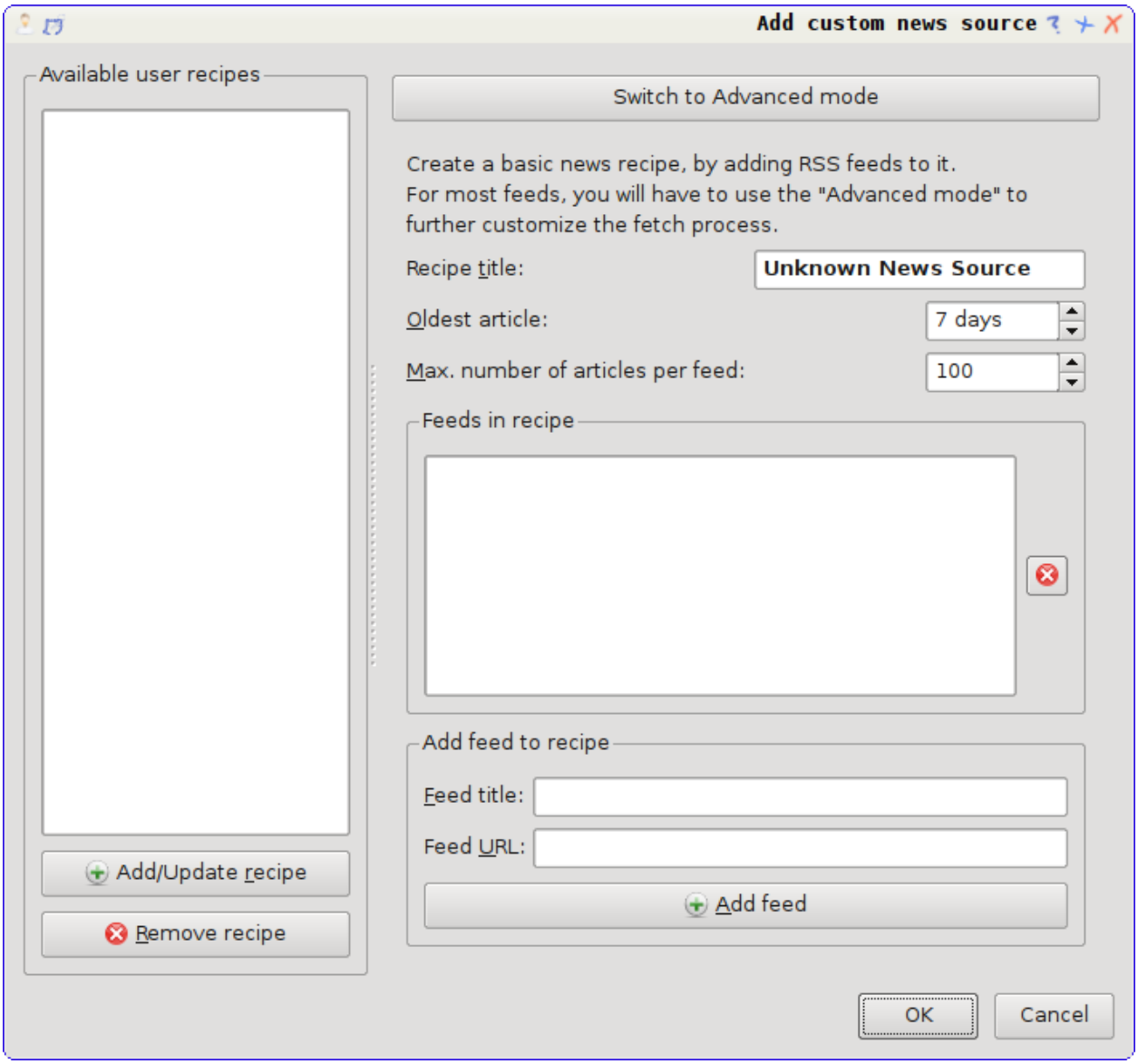
portfolio.com

portfolio.com is the website for *Condé Nast Portfolio*, a business related magazine. In order to download articles from the magazine and convert them to ebooks, we rely on the *RSS* feeds of portfolio.com. A list of such feeds is available at http://www.portfolio.com/rss/.

Lets pick a couple of feeds that look interesting:

1. Business Travel: http://feeds.portfolio.com/portfolio/businesstravel
2. Tech Observer: http://feeds.portfolio.com/portfolio/thetechobserver

I got the URLs by clicking the little orange RSS icon next to each feed name. To make calibre download the feeds and convert them into an ebook, you should right click the *Fetch news* button and then the *Add a custom news source* menu item. A dialog similar to that shown below should open up.



First enter `Portfolio` into the *Recipe title* field. This will be the title of the ebook that will be created from the articles in the above feeds.

The next two fields (*Oldest article* and *Max. number of articles*) allow you some control over how many articles should be downloaded from each feed, and they are pretty self explanatory.

To add the feeds to the recipe, enter the feed title and the feed URL and click the *Add feed* button. Once you have added both feeds, simply click the *Add/update recipe* button and you're done! Close the dialog.

To test your new *recipe*, click the *Fetch news* button and in the *Custom news sources* sub-menu click *Portfolio*. After a couple of minutes, the newly downloaded Portfolio ebook will appear in the main library view (if you have your reader connected, it will be put onto the reader instead of into the library). Select it and hit the *View* button to read!

The reason this worked so well, with so little effort is that *portfolio.com* provides *full-content RSS* feeds, i.e., the article content is embedded in the feed itself. For most news sources that provide news in this fashion, with *full-content* feeds, you don't need any more effort to convert them to ebooks. Now we will look at a news source that does not provide full content feeds. In such feeds, the full article is a webpage and the feed only contains a link to the webpage with a short summary of the article.

bbc.co.uk

Lets try the following two feeds from *The BBC*:

1. News Front Page: http://newsrss.bbc.co.uk/rss/newsonline_world_edition/front_page/rss.xml
2. Science/Nature: http://newsrss.bbc.co.uk/rss/newsonline_world_edition/science/nature/rss.xml

Follow the procedure outlined in *portfolio.com* (page 342) to create a recipe for *The BBC* (using the feeds above). Looking at the downloaded ebook, we see that calibre has done a creditable job of extracting only the content you care about from each article's webpage. However, the extraction process is not perfect. Sometimes it leaves in undesirable content like menus and navigation aids or it removes content that should have been left alone, like article headings. In order, to have perfect content extraction, we will need to customize the fetch process, as described in the next section.

1.2.2 Customizing the fetch process

When you want to perfect the download process, or download content from a particularly complex website, you can avail yourself of all the power and flexibility of the *recipe* framework. In order to do that, in the *Add custom news sources* dialog, simply click the *Switch to Advanced mode* button.

The easiest and often most productive customization is to use the print version of the online articles. The print version typically has much less cruft and translates much more smoothly to an ebook. Let's try to use the print version of the articles from *The BBC*.

Using the print version of bbc.co.uk

The first step is to look at the ebook we downloaded previously from *bbc.co.uk* (page 344). At the end of each article, in the ebook is a little blurb telling you where the article was downloaded from. Copy and paste that URL into a browser. Now on the article webpage look for a link that points to the "Printable version". Click it to see the print version of the article. It looks much neater! Now compare the two URLs. For me they were:

Article URL http://news.bbc.co.uk/2/hi/science/nature/7312016.stm

Print version URL http://newsvote.bbc.co.uk/mpapps/pagetools/print/news.bbc.co.uk/2/hi/science/nature/7312016.stm

So it looks like to get the print version, we need to prefix every article URL with:

newsvote.bbc.co.uk/mpapps/pagetools/print/

Now in the *Advanced Mode* of the Custom news sources dialog, you should see something like (remember to select *The BBC* recipe before switching to advanced mode):



You can see that the fields from the *Basic mode* have been translated to python code in a straightforward manner. We need to add instructions to this recipe to use the print version of the articles. All that's needed is to add the following two lines:

```
def print_version(self, url):
    return url.replace('http://', 'http://newsvote.bbc.co.uk/mpapps/pagetools/print/')
```

This is python, so indentation is important. After you've added the lines, it should look like:

Recipe source code (python)

```
class AdvancedUserRecipe1206418393(BasicNewsRecipe):
    title          = u'The BBC'
    oldest_article = 7
    max_articles_per_feed = 100

    feeds          = [(u'News Front Page', u'http://newsrss.bbc.co.uk/rss/newsonlir

    def print_version(self, url):
        return url.replace('http://', 'http://newsvote.bbc.co.uk/mpapps/pagetools/p
```

In the above, `def print_version(self, url)` defines a *method* that is called by calibre for every article. `url` is the URL of the original article. What `print_version` does is take that url and replace it with the new URL that points to the print version of the article. To learn about python[4] see the tutorial[5].

Now, click the *Add/update recipe* button and your changes will be saved. Re-download the ebook. You should have a much improved ebook. One of the problems with the new version is that the fonts on the print version webpage are too small. This is automatically fixed when converting to an ebook, but even after the fixing process, the font size of the menus and navigation bar to become too large relative to the article text. To fix this, we will do some more customization, in the next section.

Replacing article styles

In the previous section, we saw that the font size for articles from the print version of *The BBC* was too small. In most websites, *The BBC* included, this font size is set by means of *CSS* stylesheets. We can disable the fetching of such stylesheets by adding the line:

```
no_stylesheets = True
```

The recipe now looks like:

Recipe source code (python)

```
class AdvancedUserRecipe1206419520(BasicNewsRecipe):
    title          = u'The BBC'
    oldest_article = 7
    max_articles_per_feed = 100
    no_stylesheets = True

    feeds          = [(u'News Front Page', u'http://newsrss.bbc.co.uk/rss/news

    def print_version(self, url):
        return url.replace('http://', 'http://newsvote.bbc.co.uk/mpapps/pageto
```

[4] http://www.python.org
[5] http://docs.python.org/tut/

The new version looks pretty good. If you're a perfectionist, you'll want to read the next section, which deals with actually modifying the downloaded content.

Slicing and dicing

calibre contains very powerful and flexible abilities when it comes to manipulating downloaded content. To show off a couple of these, let's look at our old friend the *The BBC* (page 345) recipe again. Looking at the source code (*HTML*) of a couple of articles (print version), we see that they have a footer that contains no useful information, contained in

```
<div class="footer">
...
</div>
```

This can be removed by adding:

```
remove_tags     = [dict(name='div', attrs={'class':'footer'})]
```

to the recipe. Finally, lets replace some of the *CSS* that we disabled earlier, with our own *CSS* that is suitable for conversion to an ebook:

```
extra_css       = '.headline {font-size: x-large;} \n .fact { padding-top: 10pt  }'
```

With these additions, our recipe has become "production quality", indeed it is very close to the actual recipe used by calibre for the *BBC*, shown below:

```
##
## Title:          BBC News, Sport, and Blog Calibre Recipe
## Contact:        mattst - jmstanfield@gmail.com
##
## License:        GNU General Public License v3 - http://www.gnu.org/copyleft/gpl.html
## Copyright:      mattst - jmstanfield@gmail.com
##
## Written:        November 2011
## Last Edited:    2011-11-19
##

__license__     = 'GNU General Public License v3 - http://www.gnu.org/copyleft/gpl.html'
__copyright__   = 'mattst - jmstanfield@gmail.com'


'''
BBC News, Sport, and Blog Calibre Recipe
'''

# Import the regular expressions module.
import re

# Import the BasicNewsRecipe class which this class extends.
from calibre.web.feeds.recipes import BasicNewsRecipe

class BBCNewsSportBlog(BasicNewsRecipe):

    #
    #    **** IMPORTANT USERS READ ME ****
    #
    #  First select the feeds you want then scroll down below the feeds list
    #  and select the values you want for the other user preferences, like
    #  oldest_article and such like.
```

```
#
#
#  Select the BBC rss feeds which you want in your ebook.
#  Selected feed have NO '#' at their start, de-selected feeds begin with a '#'.
#
#  Eg.  ("News Home", "http://feeds.bbci.co.uk/... - include feed.
#  Eg. #("News Home", "http://feeds.bbci.co.uk/... - do not include feed.
#
# There are 68 feeds below which constitute the bulk of the available rss
# feeds on the BBC web site. These include 5 blogs by editors and
# correspondants, 16 sports feeds, 15 'sub' regional feeds (Eg. North West
# Wales, Scotland Business), and 7 Welsh language feeds.
#
# Some of the feeds are low volume (Eg. blogs), or very low volume (Eg. Click)
# so if "oldest_article = 1.5" (only articles published in the last 36 hours)
# you may get some 'empty feeds' which will not then be included in the ebook.
#
# The 15 feeds currently selected below are simply my default ones.
#
# Note: With all 68 feeds selected, oldest_article set to 2,
# max_articles_per_feed set to 100, and simultaneous_downloads set to 10,
# the ebook creation took 29 minutes on my speedy 100 mbps net connection,
# fairly high-end desktop PC running Linux (Ubuntu Lucid-Lynx).
# More realistically with 15 feeds selected, oldest_article set to 1.5,
# max_articles_per_feed set to 100, and simultaneous_downloads set to 20,
# it took 6 minutes. If that's too slow increase 'simultaneous_downloads'.
#
# Select / de-select the feeds you want in your ebook.
#
feeds = [
          ("News Home", "http://feeds.bbci.co.uk/news/rss.xml"),
          ("UK", "http://feeds.bbci.co.uk/news/uk/rss.xml"),
          ("World", "http://feeds.bbci.co.uk/news/world/rss.xml"),
          #("England", "http://feeds.bbci.co.uk/news/england/rss.xml"),
          #("Scotland", "http://feeds.bbci.co.uk/news/scotland/rss.xml"),
          #("Wales", "http://feeds.bbci.co.uk/news/wales/rss.xml"),
          #("N. Ireland", "http://feeds.bbci.co.uk/news/northern_ireland/rss.xml"),
          #("Africa", "http://feeds.bbci.co.uk/news/world/africa/rss.xml"),
          #("Asia", "http://feeds.bbci.co.uk/news/world/asia/rss.xml"),
          #("Europe", "http://feeds.bbci.co.uk/news/world/europe/rss.xml"),
          #("Latin America", "http://feeds.bbci.co.uk/news/world/latin_america/rss.xml"),
          #("Middle East", "http://feeds.bbci.co.uk/news/world/middle_east/rss.xml"),
          ("US & Canada", "http://feeds.bbci.co.uk/news/world/us_and_canada/rss.xml"),
          ("Politics", "http://feeds.bbci.co.uk/news/politics/rss.xml"),
          ("Science/Environment", "http://feeds.bbci.co.uk/news/science_and_environment/rss.xml"
          ("Technology", "http://feeds.bbci.co.uk/news/technology/rss.xml"),
          ("Magazine", "http://feeds.bbci.co.uk/news/magazine/rss.xml"),
          ("Entertainment/Arts", "http://feeds.bbci.co.uk/news/entertainment_and_arts/rss.xml"),
          #("Health", "http://feeds.bbci.co.uk/news/health/rss.xml"),
          #("Education/Family", "http://feeds.bbci.co.uk/news/education/rss.xml"),
          ("Business", "http://feeds.bbci.co.uk/news/business/rss.xml"),
          ("Special Reports", "http://feeds.bbci.co.uk/news/special_reports/rss.xml"),
          ("Also in the News", "http://feeds.bbci.co.uk/news/also_in_the_news/rss.xml"),
          #("Newsbeat", "http://www.bbc.co.uk/newsbeat/rss.xml"),
          #("Click", "http://newsrss.bbc.co.uk/rss/newsonline_uk_edition/programmes/click_online/
          ("Blog: Nick Robinson (Political Editor)", "http://feeds.bbci.co.uk/news/correspondents
          #("Blog: Mark D'Arcy (Parliamentary Correspondent)", "http://feeds.bbci.co.uk/news/cor
          #("Blog: Robert Peston (Business Editor)", "http://feeds.bbci.co.uk/news/correspondents
```

```
            #("Blog: Stephanie Flanders (Economics Editor)", "http://feeds.bbci.co.uk/news/correspo
            ("Blog: Rory Cellan-Jones (Technology correspondent)", "http://feeds.bbci.co.uk/news/co
            ("Sport Front Page", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/front_page/rs
            #("Football", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/football/rss.xml"),
            #("Cricket", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/cricket/rss.xml"),
            #("Rugby Union", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/rugby_union/rss.x
            #("Rugby League", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/rugby_league/rss
            #("Tennis", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/tennis/rss.xml"),
            #("Golf", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/golf/rss.xml"),
            #("Motorsport", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/motorsport/rss.xml
            #("Boxing", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/boxing/rss.xml"),
            #("Athletics", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/athletics/rss.xml",
            #("Snooker", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/snooker/
            #("Horse Racing", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/hor
            #("Cycling", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/cycling/
            #("Disability Sport", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports
            #("Other Sport", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/rss
            #("Olympics 2012", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/ol
            #("N. Ireland Politics", "http://feeds.bbci.co.uk/news/northern_ireland/northern_irelan
            #("Scotland Politics", "http://feeds.bbci.co.uk/news/scotland/scotland_politics/rss.xml
            #("Scotland Business", "http://feeds.bbci.co.uk/news/scotland/scotland_business/rss.xml
            #("E. Scotland, Edinburgh & Fife", "http://feeds.bbci.co.uk/news/scotland/edinburgh_eas
            #("W. Scotland & Glasgow", "http://feeds.bbci.co.uk/news/scotland/glasgow_and_west/rss
            #("Highlands & Islands", "http://feeds.bbci.co.uk/news/scotland/highlands_and_islands/r
            #("NE. Scotland, Orkney & Shetland", "http://feeds.bbci.co.uk/news/scotland/north_east_
            #("South Scotland", "http://feeds.bbci.co.uk/news/scotland/south_scotland/rss.xml"),
            #("Central Scotland & Tayside", "http://feeds.bbci.co.uk/news/scotland/tayside_and_cent
            #("Wales Politics", "http://feeds.bbci.co.uk/news/wales/wales_politics/rss.xml"),
            #("NW. Wales", "http://feeds.bbci.co.uk/news/wales/north_west_wales/rss.xml"),
            #("NE. Wales", "http://feeds.bbci.co.uk/news/wales/north_east_wales/rss.xml"),
            #("Mid. Wales", "http://feeds.bbci.co.uk/news/wales/mid_wales/rss.xml"),
            #("SW. Wales", "http://feeds.bbci.co.uk/news/wales/south_west_wales/rss.xml"),
            #("SE. Wales", "http://feeds.bbci.co.uk/news/wales/south_east_wales/rss.xml"),
            #("Newyddion - News in Welsh", "http://feeds.bbci.co.uk/newyddion/rss.xml"),
            #("Gwleidyddiaeth", "http://feeds.bbci.co.uk/newyddion/gwleidyddiaeth/rss.xml"),
            #("Gogledd-Ddwyrain", "http://feeds.bbci.co.uk/newyddion/gogledd-ddwyrain/rss.xml"),
            #("Gogledd-Orllewin", "http://feeds.bbci.co.uk/newyddion/gogledd-orllewin/rss.xml"),
            #("Canolbarth", "http://feeds.bbci.co.uk/newyddion/canolbarth/rss.xml"),
            #("De-Ddwyrain", "http://feeds.bbci.co.uk/newyddion/de-ddwyrain/rss.xml"),
            #("De-Orllewin", "http://feeds.bbci.co.uk/newyddion/de-orllewin/rss.xml"),
        ]


#    **** SELECT YOUR USER PREFERENCES ****

# Title to use for the ebook.
#
title = 'BBC News'

# A brief description for the ebook.
#
description = u'BBC web site ebook created using rss feeds.'

# The max number of articles which may be downloaded from each feed.
# I've never seen more than about 70 articles in a single feed in the
# BBC feeds.
#
max_articles_per_feed = 100
```

```
    # The max age of articles which may be downloaded from each feed. This is
    # specified in days - note fractions of days are allowed, Eg. 2.5 (2 and a
    # half days). My default of 1.5 days is the last 36 hours, the point at
    # which I've decided 'news' becomes 'old news', but be warned this is not
    # so good for the blogs, technology, magazine, etc., and sports feeds.
    # You may wish to extend this to 2-5 but watch out ebook creation time will
    # increase as well. Setting this to 30 will get everything (AFAICT) as long
    # as max_articles_per_feed remains set high (except for 'Click' which is
    # v. low volume and its currently oldest article is 4th Feb 2011).
    #
    oldest_article = 1.5

    # Number of simultaneous downloads. 20 is consistantly working fine on the
    # BBC News feeds with no problems. Speeds things up from the defualt of 5.
    # If you have a lot of feeds and/or have increased oldest_article above 2
    # then you may wish to try increasing simultaneous_downloads to 25-30,
    # Or, of course, if you are in a hurry. [I've not tried beyond 20.]
    #
    simultaneous_downloads = 20

    # Timeout for fetching files from the server in seconds. The default of
    # 120 seconds, seems somewhat excessive.
    #
    timeout = 30

    # The format string for the date shown on the ebook's first page.
    # List of all values: http://docs.python.org/library/time.html
    # Default in news.py has a leading space so that's mirrored here.
    # As with 'feeds' select/de-select by adding/removing the initial '#',
    # only one timefmt should be selected, here's a few to choose from.
    #
    timefmt = ' [%a, %d %b %Y]'              # [Fri, 14 Nov 2011] (Calibre default)
    #timefmt = ' [%a, %d %b %Y %H:%M]'       # [Fri, 14 Nov 2011 18:30]
    #timefmt = ' [%a, %d %b %Y %I:%M %p]'    # [Fri, 14 Nov 2011 06:30 PM]
    #timefmt = ' [%d %b %Y]'                 # [14 Nov 2011]
    #timefmt = ' [%d %b %Y %H:%M]'           # [14 Nov 2011 18.30]
    #timefmt = ' [%Y-%m-%d]'                 # [2011-11-14]
    #timefmt = ' [%Y-%m-%d-%H-%M]'           # [2011-11-14-18-30]



    #
    #    **** IMPORTANT ****
    #
    #    DO NOT EDIT BELOW HERE UNLESS YOU KNOW WHAT YOU ARE DOING.
    #
    #    DO NOT EDIT BELOW HERE UNLESS YOU KNOW WHAT YOU ARE DOING.
    #
    #    I MEAN IT, YES I DO, ABSOLUTELY, AT YOU OWN RISK. :)
    #
    #    **** IMPORTANT ****
    #



    # Author of this recipe.
    __author__ = 'mattst'
```

```
    # Specify English as the language of the RSS feeds (ISO-639 code).
    language = 'en_GB'

    # Set tags.
    tags = 'news, sport, blog'

    # Set publisher and publication type.
    publisher = 'BBC'
    publication_type = 'newspaper'

    # Disable stylesheets from site.
    no_stylesheets = True

    # Specifies an override encoding for sites that have an incorrect charset
    # specified. Default of 'None' says to auto-detect. Some other BBC recipes
    # use 'utf8', which works fine (so use that if necessary) but auto-detecting
    # with None is working fine, so stick with that for robustness.
    encoding = None

    # Sets whether a feed has full articles embedded in it. The BBC feeds do not.
    use_embedded_content = False

    # Removes empty feeds - why keep them!?
    remove_empty_feeds = True

    # Create a custom title which fits nicely in the Kindle title list.
    # Requires "import time" above class declaration, and replacing
    # title with custom_title in conversion_options (right column only).
    # Example of string below: "BBC News - 14 Nov 2011"
    #
    # custom_title = "BBC News - " + time.strftime('%d %b %Y')

    '''
    # Conversion options for advanced users, but don't forget to comment out the
    # current conversion_options below. Avoid setting 'linearize_tables' as that
    # plays havoc with the 'old style' table based pages.
    #
    conversion_options = { 'title'       : title,
                           'comments'    : description,
                           'tags'        : tags,
                           'language'    : language,
                           'publisher'   : publisher,
                           'authors'     : publisher,
                           'smarten_punctuation' : True
                         }
    '''

    conversion_options = { 'smarten_punctuation' : True }

    # Specify extra CSS - overrides ALL other CSS (IE. Added last).
    extra_css = 'body { font-family: verdana, helvetica, sans-serif; } \
                 .introduction, .first { font-weight: bold; } \
                 .cross-head { font-weight: bold; font-size: 125%; } \
                 .cap, .caption { display: block; font-size: 80%; font-style: italic; } \
                 .cap, .caption, .caption img, .caption span { display: block; text-align: center; ma
                 .byl, .byd, .byline img, .byline-name, .byline-title, .author-name, .author-position
                    .correspondent-portrait img, .byline-lead-in, .name, .bbc-role { display: block;
                    text-align: center; font-size: 80%; font-style: italic; margin: 1px auto; } \
```

```
                .story-date, .published { font-size: 80%; } \
                table { width: 100%; } \
                td img { display: block; margin: 5px auto; } \
                ul { padding-top: 10px; } \
                ol { padding-top: 10px; } \
                li { padding-top: 5px; padding-bottom: 5px; } \
                h1 { text-align: center; font-size: 175%; font-weight: bold; } \
                h2 { text-align: center; font-size: 150%; font-weight: bold; } \
                h3 { text-align: center; font-size: 125%; font-weight: bold; } \
                h4, h5, h6 { text-align: center; font-size: 100%; font-weight: bold; }'

    # Remove various tag attributes to improve the look of the ebook pages.
    remove_attributes = [ 'border', 'cellspacing', 'align', 'cellpadding', 'colspan',
                          'valign', 'vspace', 'hspace', 'alt', 'width', 'height' ]

    # Remove the (admittedly rarely used) line breaks, "<br />", which sometimes
    # cause a section of the ebook to start in an unsightly fashion or, more
    # frequently, a "<br />" will muck up the formatting of a correspondant's byline.
    # "<br />" and "<br clear/>" are far more frequently used on the table formatted
    # style of pages, and really spoil the look of the ebook pages.
    preprocess_regexps     = [(re.compile(r'<br[ ]*/>', re.IGNORECASE), lambda m: ''),
                              (re.compile(r'<br[ ]*clear.*/>', re.IGNORECASE), lambda m: '')]


    # Create regular expressions for tag keeping and removal to make the matches more
    # robust against minor changes and errors in the HTML, Eg. double spaces, leading
    # and trailing spaces, missing hyphens, and such like.
    # Python regular expression ('re' class) page: http://docs.python.org/library/re.html

    # ***************************************
    # Regular expressions for keep_only_tags:
    # ***************************************

    # The BBC News HTML pages use variants of 'storybody' to denote the section of a HTML
    # page which contains the main text of the article. Match storybody variants: 'storybody',
    # 'story-body', 'story body','storybody ', etc.
    storybody_reg_exp = '^.*story[_ -]*body.*$'

    # The BBC sport and 'newsbeat' (features) HTML pages use 'blq_content' to hold the title
    # and published date. This is one level above the usual news pages which have the title
    # and date within 'story-body'. This is annoying since 'blq_content' must also be kept,
    # resulting in a lot of extra things to be removed by remove_tags.
    blq_content_reg_exp = '^.*blq[_ -]*content.*$'

    # The BBC has an alternative page design structure, which I suspect is an out-of-date
    # design but which is still used in some articles, Eg. 'Click' (technology), 'FastTrack'
    # (travel), and in some sport pages. These alternative pages are table based (which is
    # why I think they are an out-of-date design) and account for -I'm guesstimaking- less
    # than 1% of all articles. They use a table class 'storycontent' to hold the article
    # and like blq_content (above) have required lots of extra removal by remove_tags.
    story_content_reg_exp = '^.*story[_ -]*content.*$'

    # Keep the sections of the HTML which match the list below. The HTML page created by
    # Calibre will fill <body> with those sections which are matched. Note that the
    # blq_content_reg_exp must be listed before storybody_reg_exp in keep_only_tags due to
    # it being the parent of storybody_reg_exp, that is to say the div class/id 'story-body'
    # will be inside div class/id 'blq_content' in the HTML (if 'blq_content' is there at
    # all). If they are the other way around in keep_only_tags then blq_content_reg_exp
```

```
    # will end up being discarded.
    keep_only_tags = [ dict(name='table', attrs={'class':re.compile(story_content_reg_exp, re.IGNOREC
                       dict(name='div',   attrs={'class':re.compile(blq_content_reg_exp, re.IGNORECAS
                       dict(name='div',   attrs={'id':re.compile(blq_content_reg_exp, re.IGNORECASE)
                       dict(name='div',   attrs={'class':re.compile(storybody_reg_exp, re.IGNORECASE)
                       dict(name='div',   attrs={'id':re.compile(storybody_reg_exp, re.IGNORECASE)})

    # ***********************************
    # Regular expressions for remove_tags:
    # ***********************************

    # Regular expression to remove share-help and variant tags. The share-help class
    # is used by the site for a variety of 'sharing' type links, Eg. Facebook, delicious,
    # twitter, email. Removed to avoid page clutter.
    share_help_reg_exp = '^.*share[_ -]*help.*$'

    # Regular expression to remove embedded-hyper and variant tags. This class is used to
    # display links to other BBC News articles on the same/similar subject.
    embedded_hyper_reg_exp = '^.*embed*ed[_ -]*hyper.*$'

    # Regular expression to remove hypertabs and variant tags. This class is used to
    # display a tab bar at the top of an article which allows the user to switch to
    # an article (viewed on the same page) providing further info., 'in depth' analysis,
    # an editorial, a correspondant's blog entry, and such like. The ability to handle
    # a tab bar of this nature is currently beyond the scope of this recipe and
    # possibly of Calibre itself (not sure about that - TO DO - check!).
    hypertabs_reg_exp = '^.*hyper[_ -]*tabs.*$'

    # Regular expression to remove story-feature and variant tags. Eg. 'story-feature',
    # 'story-feature related narrow', 'story-feature wide', 'story-feature narrow'.
    # This class is used to add additional info. boxes, or small lists, outside of
    # the main story. TO DO: Work out a way to incorporate these neatly.
    story_feature_reg_exp = '^.*story[_ -]*feature.*$'

    # Regular expression to remove video and variant tags, Eg. 'videoInStoryB',
    # 'videoInStoryC'. This class is used to embed video.
    video_reg_exp = '^.*video.*$'

    # Regular expression to remove audio and variant tags, Eg. 'audioInStoryD'.
    # This class is used to embed audio.
    audio_reg_exp = '^.*audio.*$'

    # Regular expression to remove pictureGallery and variant tags, Eg. 'pictureGallery'.
    # This class is used to embed a photo slideshow. See also 'slideshow' below.
    picture_gallery_reg_exp = '^.*picture.*$'

    # Regular expression to remove slideshow and variant tags, Eg. 'dslideshow-enclosure'.
    # This class is used to embed a slideshow (not necessarily photo) but both
    # 'slideshow' and 'pictureGallery' are used for slideshows.
    slideshow_reg_exp = '^.*slide[_ -]*show.*$'

    # Regular expression to remove social-links and variant tags. This class is used to
    # display links to a BBC bloggers main page, used in various columnist's blogs
    # (Eg. Nick Robinson, Robert Preston).
    social_links_reg_exp = '^.*social[_ -]*links.*$'

    # Regular expression to remove quote and (multi) variant tags, Eg. 'quote',
    # 'endquote', 'quote-credit', 'quote-credit-title', etc. These are usually
```

```
    # removed by 'story-feature' removal (as they are usually within them), but
    # not always. The quotation removed is always (AFAICT) in the article text
    # as well but a 2nd copy is placed in a quote tag to draw attention to it.
    # The quote class tags may or may not appear in div's.
    quote_reg_exp = '^.*quote.*$'

    # Regular expression to remove hidden and variant tags, Eg. 'hidden'.
    # The purpose of these is unclear, they seem to be an internal link to a
    # section within the article, but the text of the link (Eg. 'Continue reading
    # the main story') never seems to be displayed anyway. Removed to avoid clutter.
    # The hidden class tags may or may not appear in div's.
    hidden_reg_exp = '^.*hidden.*$'

    # Regular expression to remove comment and variant tags, Eg. 'comment-introduction'.
    # Used on the site to display text about registered users entering comments.
    comment_reg_exp = '^.*comment.*$'

    # Regular expression to remove form and variant tags, Eg. 'comment-form'.
    # Used on the site to allow registered BBC users to fill in forms, typically
    # for entering comments about an article.
    form_reg_exp = '^.*form.*$'

    # Extra things to remove due to the addition of 'blq_content' in keep_only_tags.

    #<div class="story-actions"> Used on sports pages for 'email' and 'print'.
    story_actions_reg_exp = '^.*story[_ -]*actions.*$'

    #<div class="bookmark-list"> Used on sports pages instead of 'share-help' (for
    # social networking links).
    bookmark_list_reg_exp = '^.*bookmark[_ -]*list.*$'

    #<div id="secondary-content" class="content-group">
    # NOTE: Don't remove class="content-group" that is needed.
    # Used on sports pages to link to 'similar stories'.
    secondary_content_reg_exp = '^.*secondary[_ -]*content.*$'

    #<div id="featured-content" class="content-group">
    # NOTE: Don't remove class="content-group" that is needed.
    # Used on sports pages to link to pages like 'tables', 'fixtures', etc.
    featured_content_reg_exp = '^.*featured[_ -]*content.*$'

    #<div id="navigation">
    # Used on sports pages to link to pages like 'tables', 'fixtures', etc.
    # Used sometimes instead of "featured-content" above.
    navigation_reg_exp = '^.*navigation.*$'

    #<a class="skip" href="#blq-container-inner">Skip to top</a>
    # Used on sports pages to link to the top of the page.
    skip_reg_exp = '^.*skip.*$'

    # Extra things to remove due to the addition of 'storycontent' in keep_only_tags,
    # which are the alterative table design based pages. The purpose of some of these
    # is not entirely clear from the pages (which are a total mess!).

    # Remove mapping based tags, Eg. <map id="world_map">
    # The dynamic maps don't seem to work during ebook creation. TO DO: Investigate.
    map_reg_exp = '^.*map.*$'
```

```
# Remove social bookmarking variation, called 'socialBookMarks'.
social_bookmarks_reg_exp = '^.*social[_ -]*bookmarks.*$'

# Remove page navigation tools, like 'search', 'email', 'print', called 'blq-mast'.
blq_mast_reg_exp = '^.*blq[_ -]*mast.*$'

# Remove 'sharesb', I think this is a generic 'sharing' class. It seems to appear
# alongside 'socialBookMarks' whenever that appears. I am removing it as well
# under the assumption that it can appear alone as well.
sharesb_reg_exp = '^.*sharesb.*$'

# Remove class 'o'. The worst named user created css class of all time. The creator
# should immediately be fired. I've seen it used to hold nothing at all but with
# 20 or so empty lines in it. Also to hold a single link to another article.
# Whatever it was designed to do it is not wanted by this recipe. Exact match only.
o_reg_exp = '^o$'

# Remove 'promotopbg' and 'promobottombg', link lists. Have decided to
# use two reg expressions to make removing this (and variants) robust.
promo_top_reg_exp = '^.*promotopbg.*$'
promo_bottom_reg_exp = '^.*promobottombg.*$'

# Remove 'nlp', provides heading for link lists. Requires an exact match due to
# risk of matching those letters in something needed, unless I see a variation
# of 'nlp' used at a later date.
nlp_reg_exp = '^nlp$'

# Remove 'mva', provides embedded floating content of various types. Variant 'mvb'
# has also now been seen. Requires an exact match of 'mva' or 'mvb' due to risk of
# matching those letters in something needed.
mva_or_mvb_reg_exp = '^mv[ab]$'

# Remove 'mvtb', seems to be page navigation tools, like 'blq-mast'.
mvtb_reg_exp = '^mvtb$'

# Remove 'blq-toplink', class to provide a link to the top of the page.
blq_toplink_reg_exp = '^.*blq[_ -]*top[_ -]*link.*$'

# Remove 'products and services' links, Eg. desktop tools, alerts, and so on.
# Eg. Class="servicev4 ukfs_services" - what a mess of a name. Have decided to
# use two reg expressions to make removing this (and variants) robust.
prods_services_01_reg_exp = '^.*servicev4.*$'
prods_services_02_reg_exp = '^.*ukfs[_ -]*services.*$'

# Remove -what I think is- some kind of navigation tools helper class, though I am
# not sure, it's called: 'blq-rst blq-new-nav'. What I do know is it pops up
# frequently and it is not wanted. Have decided to use two reg expressions to make
# removing this (and variants) robust.
blq_misc_01_reg_exp = '^.*blq[_ -]*rst.*$'
blq_misc_02_reg_exp = '^.*blq[_ -]*new[_ -]*nav.*$'

# Remove 'puffbox' - this may only appear inside 'storyextra', so it may not
# need removing - I have no clue what it does other than it contains links.
# Whatever it is - it is not part of the article and is not wanted.
puffbox_reg_exp = '^.*puffbox.*$'

# Remove 'sibtbg' and 'sibtbgf' - some kind of table formatting classes.
sibtbg_reg_exp = '^.*sibtbg.*$'
```

```
    # Remove 'storyextra' - links to relevant articles and external sites.
    storyextra_reg_exp = '^.*story[_ -]*extra.*$'


    remove_tags = [ dict(name='div',  attrs={'class':re.compile(story_feature_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'class':re.compile(share_help_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(embedded_hyper_reg_exp, re.IGNORECASE
                    dict(name='div',  attrs={'class':re.compile(hypertabs_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(video_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(audio_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(picture_gallery_reg_exp, re.IGNORECAS
                    dict(name='div',  attrs={'class':re.compile(slideshow_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(quote_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(hidden_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(comment_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(story_actions_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'class':re.compile(bookmark_list_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'id':re.compile(secondary_content_reg_exp, re.IGNORECASE
                    dict(name='div',  attrs={'id':re.compile(featured_content_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'id':re.compile(navigation_reg_exp, re.IGNORECASE)}),
                    dict(name='form', attrs={'id':re.compile(form_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(quote_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(hidden_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(social_links_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(comment_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(skip_reg_exp, re.IGNORECASE)}),
                    dict(name='map', attrs={'id':re.compile(map_reg_exp, re.IGNORECASE)}),
                    dict(name='map', attrs={'name':re.compile(map_reg_exp, re.IGNORECASE)}),
                    dict(name='div', attrs={'id':re.compile(social_bookmarks_reg_exp, re.IGNORECASE)
                    dict(name='div', attrs={'id':re.compile(blq_mast_reg_exp, re.IGNORECASE)}),
                    dict(name='div', attrs={'class':re.compile(sharesb_reg_exp, re.IGNORECASE)}),
                    dict(name='div', attrs={'class':re.compile(o_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(promo_top_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(promo_bottom_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'class':re.compile(nlp_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(mva_or_mvb_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(mvtb_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(blq_toplink_reg_exp, re.IGNORECASE)}
                    dict(name='div',  attrs={'class':re.compile(prods_services_01_reg_exp, re.IGNOREC
                    dict(name='div',  attrs={'class':re.compile(prods_services_02_reg_exp, re.IGNOREC
                    dict(name='div',  attrs={'class':re.compile(blq_misc_01_reg_exp, re.IGNORECASE)}
                    dict(name='div',  attrs={'class':re.compile(blq_misc_02_reg_exp, re.IGNORECASE)}
                    dict(name='div',  attrs={'class':re.compile(puffbox_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(sibtbg_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(storyextra_reg_exp, re.IGNORECASE)})
                  ]

    # Uses url to create and return the 'printer friendly' version of the url.
    # In other words the 'print this page' address of the page.
    #
    # There are 3 types of urls used in the BBC site's rss feeds. There is just
    # 1 type for the standard news while there are 2 used for sports feed urls.
    # Note: Sports urls are linked from regular news feeds (Eg. 'News Home') when
    # there is a major story of interest to 'everyone'. So even if no BBC sports
    # feeds are added to 'feeds' the logic of this method is still needed to avoid
    # blank / missing / empty articles which have an index title and then no body.
    def print_version(self, url):
```

```
        # Handle sports page urls type 01:
        if (url.find("go/rss/-/sport1/") != -1):
            temp_url = url.replace("go/rss/-/", "")

        # Handle sports page urls type 02:
        elif (url.find("go/rss/int/news/-/sport1/") != -1):
            temp_url = url.replace("go/rss/int/news/-/", "")

        # Handle regular news page urls:
        else:
            temp_url = url.replace("go/rss/int/news/-/", "")

        # Always add "?print=true" to the end of the url.
        print_url = temp_url + "?print=true"

        return print_url


    # Remove articles in feeds based on a string in the article title or url.
    #
    # Code logic written by: Starson17 - posted in: "Recipes - Re-usable code"
    # thread, in post with title: "Remove articles from feed", see url:
    # http://www.mobileread.com/forums/showpost.php?p=1165462&postcount=6
    # Many thanks and all credit to Starson17.
    #
    # Starson17's code has obviously been altered to suite my requirements.
    def parse_feeds(self):

        # Call parent's method.
        feeds = BasicNewsRecipe.parse_feeds(self)

        # Loop through all feeds.
        for feed in feeds:

            # Loop through all articles in feed.
            for article in feed.articles[:]:

                # Match key words and remove article if there's a match.

                # Most BBC rss feed video only 'articles' use upper case 'VIDEO'
                # as a title prefix. Just match upper case 'VIDEO', so that
                # articles like 'Video game banned' won't be matched and removed.
                if 'VIDEO' in article.title:
                    feed.articles.remove(article)

                # Most BBC rss feed audio only 'articles' use upper case 'AUDIO'
                # as a title prefix. Just match upper case 'AUDIO', so that
                # articles like 'Hi-Def audio...' won't be matched and removed.
                elif 'AUDIO' in article.title:
                    feed.articles.remove(article)

                # Most BBC rss feed photo slideshow 'articles' use 'In Pictures',
                # 'In pictures', and 'in pictures', somewhere in their title.
                # Match any case of that phrase.
                elif 'IN PICTURES' in article.title.upper():
                    feed.articles.remove(article)

                # As above, but user contributed pictures. Match any case.
```

```
                elif 'YOUR PICTURES' in article.title.upper():
                    feed.articles.remove(article)

                # 'Sportsday Live' are articles which contain a constantly and
                # dynamically updated 'running commentary' during a live sporting
                # event. Match any case.
                elif 'SPORTSDAY LIVE' in article.title.upper():
                    feed.articles.remove(article)

                # Sometimes 'Sportsday Live' (above) becomes 'Live - Sport Name'.
                # These are being matched below using 'Live - ' because removing all
                # articles with 'live' in their titles would remove some articles
                # that are in fact not live sports pages. Match any case.
                elif 'LIVE - ' in article.title.upper():
                    feed.articles.remove(article)

                # 'Quiz of the week' is a Flash player weekly news quiz. Match only
                # the 'Quiz of the' part in anticipation of monthly and yearly
                # variants. Match any case.
                elif 'QUIZ OF THE' in article.title.upper():
                    feed.articles.remove(article)

                # Remove articles with 'scorecards' in the url. These are BBC sports
                # pages which just display a cricket scorecard. The pages have a mass
                # of table and css entries to display the scorecards nicely. Probably
                # could make them work with this recipe, but might take a whole day
                # of work to sort out all the css - basically a formatting nightmare.
                elif 'scorecards' in article.url:
                    feed.articles.remove(article)

        return feeds

# End of class and file.
```

This *recipe* explores only the tip of the iceberg when it comes to the power of calibre. To explore more of the abilities of calibre we'll examine a more complex real life example in the next section.

Real life example

A reasonably complex real life example that exposes more of the *API* of `BasicNewsRecipe` is the *recipe* for *The New York Times*

```
import string, re
from calibre import strftime
from calibre.web.feeds.recipes import BasicNewsRecipe
from calibre.ebooks.BeautifulSoup import BeautifulSoup

class NYTimes(BasicNewsRecipe):

    title       = 'The New York Times'
    __author__  = 'Kovid Goyal'
    description = 'Daily news from the New York Times'
    timefmt = ' [%a, %d %b, %Y]'
    needs_subscription = True
    remove_tags_before = dict(id='article')
    remove_tags_after  = dict(id='article')
    remove_tags = [dict(attrs={'class':['articleTools', 'post-tools', 'side_tool', 'nextArticleLink c
```

```
            dict(id=['footer', 'toolsRight', 'articleInline', 'navigation', 'archive', 'side_sear
            dict(name=['script', 'noscript', 'style'])]
    encoding = 'cp1252'
    no_stylesheets = True
    extra_css = 'h1 {font: sans-serif large;}\n.byline {font:monospace;}'

    def get_browser(self):
        br = BasicNewsRecipe.get_browser()
        if self.username is not None and self.password is not None:
            br.open('http://www.nytimes.com/auth/login')
            br.select_form(name='login')
            br['USERID']   = self.username
            br['PASSWORD'] = self.password
            br.submit()
        return br

    def parse_index(self):
        soup = self.index_to_soup('http://www.nytimes.com/pages/todayspaper/index.html')

        def feed_title(div):
            return ''.join(div.findAll(text=True, recursive=False)).strip()

        articles = {}
        key = None
        ans = []
        for div in soup.findAll(True,
             attrs={'class':['section-headline', 'story', 'story headline']}):

            if div['class'] == 'section-headline':
                key = string.capwords(feed_title(div))
                articles[key] = []
                ans.append(key)

            elif div['class'] in ['story', 'story headline']:
                a = div.find('a', href=True)
                if not a:
                    continue
                url = re.sub(r'\?.*', '', a['href'])
                url += '?pagewanted=all'
                title = self.tag_to_string(a, use_alt=True).strip()
                description = ''
                pubdate = strftime('%a, %d %b')
                summary = div.find(True, attrs={'class':'summary'})
                if summary:
                    description = self.tag_to_string(summary, use_alt=False)

                feed = key if key is not None else 'Uncategorized'
                if not articles.has_key(feed):
                    articles[feed] = []
                if not 'podcasts' in url:
                    articles[feed].append(
                              dict(title=title, url=url, date=pubdate,
                                   description=description,
                                   content=''))
        ans = self.sort_index_by(ans, {'The Front Page':-1, 'Dining In, Dining Out':1, 'Obituaries':2
        ans = [(key, articles[key]) for key in ans if articles.has_key(key)]
        return ans
```

```
    def preprocess_html(self, soup):
        refresh = soup.find('meta', {'http-equiv':'refresh'})
        if refresh is None:
            return soup
        content = refresh.get('content').partition('=')[2]
        raw = self.browser.open('http://www.nytimes.com'+content).read()
        return BeautifulSoup(raw.decode('cp1252', 'replace'))
```

We see several new features in this *recipe*. First, we have:

```
timefmt = ' [%a, %d %b, %Y]'
```

This sets the displayed time on the front page of the created ebook to be in the format, `Day, Day_Number Month, Year`. See `timefmt` (page 368).

Then we see a group of directives to cleanup the downloaded *HTML*:

```
remove_tags_before = dict(name='h1')
remove_tags_after  = dict(id='footer')
remove_tags = ...
```

These remove everything before the first `<h1>` tag and everything after the first tag whose id is `footer`. See `remove_tags` (page 367), `remove_tags_before` (page 367), `remove_tags_after` (page 367).

The next interesting feature is:

```
needs_subscription = True
...
def get_browser(self):
    ...
```

`needs_subscription = True` tells calibre that this recipe needs a username and password in order to access the content. This causes, calibre to ask for a username and password whenever you try to use this recipe. The code in `calibre.web.feeds.news.BasicNewsRecipe.get_browser()` (page 361) actually does the login into the NYT website. Once logged in, calibre will use the same, logged in, browser instance to fetch all content. See mechanize[6] to understand the code in `get_browser`.

The next new feature is the `calibre.web.feeds.news.BasicNewsRecipe.parse_index()` (page 362) method. Its job is to go to http://www.nytimes.com/pages/todayspaper/index.html and fetch the list of articles that appear in *todays* paper. While more complex than simply using *RSS*, the recipe creates an ebook that corresponds very closely to the days paper. `parse_index` makes heavy use of BeautifulSoup[7] to parse the daily paper webpage.

The final new feature is the `calibre.web.feeds.news.BasicNewsRecipe.preprocess_html()` (page 363) method. It can be used to perform arbitrary transformations on every downloaded HTML page. Here it is used to bypass the ads that the nytimes shows you before each article.

1.2.3 Tips for developing new recipes

The best way to develop new recipes is to use the command line interface. Create the recipe using your favorite python editor and save it to a file say `myrecipe.recipe`. The *.recipe* extension is required. You can download content using this recipe with the command:

```
ebook-convert myrecipe.recipe .epub --test -vv --debug-pipeline debug
```

The command **ebook-convert** will download all the webpages and save them to the EPUB file `myrecipe.epub`. The `-vv` makes ebook-convert spit out a lot of information about what it is doing. The `--test` makes it download

[6] http://wwwsearch.sourceforge.net/mechanize/
[7] http://www.crummy.com/software/BeautifulSoup/documentation.html

only a couple of articles from at most two feeds. In addition, ebook-convert will put the downloaded HTML into the `debug/input` directory, where `debug` is the directory you specified in the *`--debug-pipeline`* option.

Once the download is complete, you can look at the downloaded *HTML* by opening the file `debug/input/index.html` in a browser. Once you're satisfied that the download and preprocessing is happening correctly, you can generate ebooks in different formats as shown below:

```
ebook-convert myrecipe.recipe myrecipe.epub
ebook-convert myrecipe.recipe myrecipe.mobi
...
```

If you're satisfied with your recipe, and you feel there is enough demand to justify its inclusion into the set of built-in recipes, post your recipe in the calibre recipes forum[8] to share it with other calibre users.

Note: On OS X, the ebook-convert command will not be available by default. Go to Preferences->Miscellaneous and click the install command line tools button to make it available.

See Also:

ebook-convert **(page 486)** The command line interface for all ebook conversion.

1.2.4 Further reading

To learn more about writing advanced recipes using some of the facilities, available in `BasicNewsRecipe` you should consult the following sources:

API Documentation **(page 360)** Documentation of the `BasicNewsRecipe` class and all its important methods and fields.

BasicNewsRecipe[9] The source code of `BasicNewsRecipe`

Built-in recipes[10] The source code for the built-in recipes that come with calibre

The calibre recipes forum[11] Lots of knowledgeable calibre recipe writers hang out here.

1.2.5 API documentation

API Documentation for recipes

The API for writing recipes is defined by the `BasicNewsRecipe` (page 360)

class `calibre.web.feeds.news.`**`BasicNewsRecipe`**(*options*, *log*, *progress_reporter*)

Base class that contains logic needed in all recipes. By overriding progressively more of the functionality in this class, you can make progressively more customized/powerful recipes. For a tutorial introduction to creating recipes, see *Adding your favorite news website* (page 341).

`abort_recipe_processing`(*msg*)

Causes the recipe download system to abort the download of this recipe, displaying a simple feedback message to the user.

`add_toc_thumbnail`(*article*, *src*)

Call this from populate_article_metadata with the src attribute of an <img> tag from the article that is appropriate for use as the thumbnail representing the article in the Table of Contents. Whether the thumbnail is actually used is device dependent (currently only used by the Kindles). Note that the referenced image must be one that was successfully downloaded, otherwise it will be ignored.

[8] http://www.mobileread.com/forums/forumdisplay.php?f=228

classmethod adeify_images(*soup*)
If your recipe when converted to EPUB has problems with images when viewed in Adobe Digital Editions, call this method from within postprocess_html() (page 363).

cleanup()
Called after all articles have been download. Use it to do any cleanup like logging out of subscription sites, etc.

clone_browser(*br*)
Clone the browser br. Cloned browsers are used for multi-threaded downloads, since mechanize is not thread safe. The default cloning routines should capture most browser customization, but if you do something exotic in your recipe, you should override this method in your recipe and clone manually.

Cloned browser instances use the same, thread-safe CookieJar by default, unless you have customized cookie handling.

default_cover(*cover_file*)
Create a generic cover for recipes that dont have a cover

download()
Download and pre-process all articles from the feeds in this recipe. This method should be called only once on a particular Recipe instance. Calling it more than once will lead to undefined behavior. :return: Path to index.html

extract_readable_article(*html*, *url*)
Extracts main article content from 'html', cleans up and returns as a (article_html, extracted_title) tuple. Based on the original readability algorithm by Arc90.

get_article_url(*article*)
Override in a subclass to customize extraction of the *URL* that points to the content for each article. Return the article URL. It is called with *article*, an object representing a parsed article from a feed. See feedparser[12]. By default it looks for the original link (for feeds syndicated via a service like feedburner or pheedo) and if found, returns that or else returns article.link[13].

get_browser(**args*, ***kwargs*)
Return a browser instance used to fetch documents from the web. By default it returns a mechanize[14] browser instance that supports cookies, ignores robots.txt, handles refreshes and has a mozilla firefox user agent.

If your recipe requires that you login first, override this method in your subclass. For example, the following code is used in the New York Times recipe to login for full access:

```
def get_browser(self):
    br = BasicNewsRecipe.get_browser(self)
    if self.username is not None and self.password is not None:
        br.open('http://www.nytimes.com/auth/login')
        br.select_form(name='login')
        br['USERID']   = self.username
        br['PASSWORD'] = self.password
        br.submit()
    return br
```

get_cover_url()
Return a *URL* to the cover image for this issue or *None*. By default it returns the value of the member *self.cover_url* which is normally *None*. If you want your recipe to download a cover for the e-book override this method in your subclass, or set the member variable *self.cover_url* before this method is called.

[12] http://packages.python.org/feedparser/
[13] http://packages.python.org/feedparser/reference-entry-link.html
[14] http://wwwsearch.sourceforge.net/mechanize/

get_feeds()
Return a list of *RSS* feeds to fetch for this profile. Each element of the list must be a 2-element tuple of the form (title, url). If title is None or an empty string, the title from the feed is used. This method is useful if your recipe needs to do some processing to figure out the list of feeds to download. If so, override in your subclass.

get_masthead_title()
Override in subclass to use something other than the recipe title

get_masthead_url()
Return a *URL* to the masthead image for this issue or *None*. By default it returns the value of the member *self.masthead_url* which is normally *None*. If you want your recipe to download a masthead for the e-book override this method in your subclass, or set the member variable *self.masthead_url* before this method is called. Masthead images are used in Kindle MOBI files.

get_obfuscated_article(*url*)
If you set *articles_are_obfuscated* this method is called with every article URL. It should return the path to a file on the filesystem that contains the article HTML. That file is processed by the recursive HTML fetching engine, so it can contain links to pages/images on the web.

This method is typically useful for sites that try to make it difficult to access article content automatically.

classmethod image_url_processor(*baseurl*, *url*)
Perform some processing on image urls (perhaps removing size restrictions for dynamically generated images, etc.) and return the precessed URL.

index_to_soup(*url_or_raw*, *raw=False*)
Convenience method that takes an URL to the index page and returns a BeautifulSoup[15] of it.

url_or_raw: Either a URL or the downloaded index page as a string

is_link_wanted(*url*, *tag*)
Return True if the link should be followed or False otherwise. By default, raises NotImplementedError which causes the downloader to ignore it.

Parameters

- **url** – The URL to be followed
- **tag** – The Tag from which the URL was derived

javascript_login(*browser*, *username*, *password*)
This method is used to login to a website that uses javascript for its login form. After the login is complete, the cookies returned from the website are copied to a normal (non-javascript) browser and the download proceeds using those cookies.

An example implementation:

```
def javascript_login(self, browser, username, password):
    browser.visit('http://some-page-that-has-a-login')
    form = browser.select_form(nr=0) # Select the first form on the page
    form['username'] = username
    form['password'] = password
    browser.submit(timeout=120) # Submit the form and wait at most two minutes for loading t
```

Note that you can also select forms with CSS2 selectors, like this:

```
browser.select_form('form#login_form')
browser.select_from('form[name="someform"]')
```

[15] http://www.crummy.com/software/BeautifulSoup/documentation.html

parse_feeds()
Create a list of articles from the list of feeds returned by BasicNewsRecipe.get_feeds() (page 361). Return a list of Feed objects.

parse_index()
This method should be implemented in recipes that parse a website instead of feeds to generate a list of articles. Typical uses are for news sources that have a "Print Edition" webpage that lists all the articles in the current print edition. If this function is implemented, it will be used in preference to BasicNewsRecipe.parse_feeds() (page 362).

It must return a list. Each element of the list must be a 2-element tuple of the form ('feed title', list of articles).

Each list of articles must contain dictionaries of the form:

```
{
'title'       : article title,
'url'         : URL of print version,
'date'        : The publication date of the article as a string,
'description' : A summary of the article
'content'     : The full article (can be an empty string). Obsolete
                do not use, instead save the content to a temporary
                file and pass a file:///path/to/temp/file.html as
                the URL.
}
```

For an example, see the recipe for downloading *The Atlantic*. In addition, you can add 'author' for the author of the article.

If you want to abort processing for some reason and have calibre show the user a simple message instead of an error, call abort_recipe_processing() (page 360).

populate_article_metadata(*article*, *soup*, *first*)
Called when each HTML page belonging to article is downloaded. Intended to be used to get article metadata like author/summary/etc. from the parsed HTML (soup). :param article: A object of class calibre.web.feeds.Article. If you change the summary, remember to also change the text_summary :param soup: Parsed HTML belonging to this article :param first: True iff the parsed HTML is the first page of the article.

postprocess_book(*oeb*, *opts*, *log*)
Run any needed post processing on the parsed downloaded e-book.

Parameters

- **oeb** – An OEBBook object
- **opts** – Conversion options

postprocess_html(*soup*, *first_fetch*)
This method is called with the source of each downloaded *HTML* file, after it is parsed for links and images. It can be used to do arbitrarily powerful post-processing on the *HTML*. It should return *soup* after processing it.

Parameters

- **soup** – A BeautifulSoup[16] instance containing the downloaded *HTML*.
- **first_fetch** – True if this is the first page of an article.

preprocess_html(*soup*)
This method is called with the source of each downloaded *HTML* file, before it is parsed for links and

[16] http://www.crummy.com/software/BeautifulSoup/documentation.html

images. It is called after the cleanup as specified by remove_tags etc. It can be used to do arbitrarily powerful pre-processing on the *HTML*. It should return *soup* after processing it.

soup: A BeautifulSoup[17] instance containing the downloaded *HTML*.

preprocess_raw_html(*raw_html*, *url*)
This method is called with the source of each downloaded *HTML* file, before it is parsed into an object tree. raw_html is a unicode string representing the raw HTML downloaded from the web. url is the URL from which the HTML was downloaded.

Note that this method acts *before* preprocess_regexps.

This method must return the processed raw_html as a unicode object.

classmethod print_version(*url*)
Take a *url* pointing to the webpage with article content and return the *URL* pointing to the print version of the article. By default does nothing. For example:

```
def print_version(self, url):
    return url + '?&pagewanted=print'
```

skip_ad_pages(*soup*)
This method is called with the source of each downloaded *HTML* file, before any of the cleanup attributes like remove_tags, keep_only_tags are applied. Note that preprocess_regexps will have already been applied. It is meant to allow the recipe to skip ad pages. If the soup represents an ad page, return the HTML of the real page. Otherwise return None.

soup: A BeautifulSoup[18] instance containing the downloaded *HTML*.

sort_index_by(*index*, *weights*)
Convenience method to sort the titles in *index* according to *weights*. *index* is sorted in place. Returns *index*.

index: A list of titles.

weights: A dictionary that maps weights to titles. If any titles in index are not in weights, they are assumed to have a weight of 0.

classmethod tag_to_string(*tag*, *use_alt=True*, *normalize_whitespace=True*)
Convenience method to take a BeautifulSoup[19] *Tag* and extract the text from it recursively, including any CDATA sections and alt tag attributes. Return a possibly empty unicode string.

use_alt: If *True* try to use the alt attribute for tags that don't have any textual content

tag: BeautifulSoup[20] *Tag*

articles_are_obfuscated = False
Set to True and implement `get_obfuscated_article()` (page 362) to handle websites that try to make it difficult to scrape content.

auto_cleanup = False
Automatically extract all the text from downloaded article pages. Uses the algorithms from the readability project. Setting this to True, means that you do not have to worry about cleaning up the downloaded HTML manually (though manual cleanup will always be superior).

auto_cleanup_keep = None
Specify elements that the auto cleanup algorithm should never remove The syntax is a XPath expression. For example:

[17] http://www.crummy.com/software/BeautifulSoup/documentation.html
[18] http://www.crummy.com/software/BeautifulSoup/documentation.html
[19] http://www.crummy.com/software/BeautifulSoup/documentation.html
[20] http://www.crummy.com/software/BeautifulSoup/documentation.html

```
auto_cleanup_keep = '//div[@id="article-image"]' will keep all divs with
                                                 id="article-image"
auto_cleanup_keep = '//*[@class="important"]' will keep all elements
                                               with class="important"
auto_cleanup_keep = '//div[@id="article-image"]|//span[@class="important"]'
                  will keep all divs with id="article-image" and spans
                  with class="important"
```

center_navbar = True
If True the navigation bar is center aligned, otherwise it is left aligned

conversion_options = {}
Recipe specific options to control the conversion of the downloaded content into an e-book. These will override any user or plugin specified values, so only use if absolutely necessary. For example:

```
conversion_options = {
  'base_font_size'   : 16,
  'tags'             : 'mytag1,mytag2',
  'title'            : 'My Title',
  'linearize_tables' : True,
}
```

cover_margins = (0, 0, '#ffffff')
By default, the cover image returned by get_cover_url() will be used as the cover for the periodical. Overriding this in your recipe instructs calibre to render the downloaded cover into a frame whose width and height are expressed as a percentage of the downloaded cover. cover_margins = (10, 15, '#ffffff') pads the cover with a white margin 10px on the left and right, 15px on the top and bottom. Color names defined at http://www.imagemagick.org/script/color.php Note that for some reason, white does not always work on windows. Use #ffffff instead

delay = 0
Delay between consecutive downloads in seconds. The argument may be a floating point number to indicate a more precise time.

description = u''
A couple of lines that describe the content this recipe downloads. This will be used primarily in a GUI that presents a list of recipes.

encoding = None
Specify an override encoding for sites that have an incorrect charset specification. The most common being specifying `latin1` and using `cp1252`. If None, try to detect the encoding. If it is a callable, the callable is called with two arguments: The recipe object and the source to be decoded. It must return the decoded source.

extra_css = None
Specify any extra *CSS* that should be added to downloaded *HTML* files It will be inserted into *<style>* tags, just before the closing *</head>* tag thereby overriding all *CSS* except that which is declared using the style attribute on individual *HTML* tags. For example:

```
extra_css = '.heading { font: serif x-large }'
```

feeds = None
List of feeds to download Can be either `[url1, url2, ...]` or `[('title1', url1), ('title2', url2),...]`

filter_regexps = []
List of regular expressions that determines which links to ignore If empty it is ignored. Used only if is_link_wanted is not implemented. For example:

```
filter_regexps = [r'ads\.doubleclick\.net']
```

will remove all URLs that have *ads.doubleclick.net* in them.

Only one of BasicNewsRecipe.match_regexps (page 366) or BasicNewsRecipe.filter_regexps (page 365) should be defined.

ignore_duplicate_articles = None
Ignore duplicates of articles that are present in more than one section. A duplicate article is an article that has the same title and/or URL. To ignore articles with the same title, set this to: ignore_duplicate_articles = {'title'} To use URLs instead, set it to: ignore_duplicate_articles = {'url'} To match on title or URL, set it to: ignore_duplicate_articles = {'title', 'url'}

keep_only_tags = []
Keep only the specified tags and their children. For the format for specifying a tag see BasicNewsRecipe.remove_tags (page 367). If this list is not empty, then the <*body*> tag will be emptied and re-filled with the tags that match the entries in this list. For example:

```
keep_only_tags = [dict(id=['content', 'heading'])]
```

will keep only tags that have an *id* attribute of *"content"* or *"heading"*.

language = 'und'
The language that the news is in. Must be an ISO-639 code either two or three characters long

masthead_url = None
By default, calibre will use a default image for the masthead (Kindle only). Override this in your recipe to provide a url to use as a masthead.

match_regexps = []
List of regular expressions that determines which links to follow If empty, it is ignored. Used only if is_link_wanted is not implemented. For example:

```
match_regexps = [r'page=[0-9]+']
```

will match all URLs that have *page=some number* in them.

Only one of BasicNewsRecipe.match_regexps (page 366) or BasicNewsRecipe.filter_regexps (page 365) should be defined.

max_articles_per_feed = 100
Maximum number of articles to download from each feed. This is primarily useful for feeds that don't have article dates. For most feeds, you should use BasicNewsRecipe.oldest_article (page 366)

needs_subscription = False
If True the GUI will ask the user for a username and password to use while downloading If set to "optional" the use of a username and password becomes optional

no_stylesheets = False
Convenient flag to disable loading of stylesheets for websites that have overly complex stylesheets unsuitable for conversion to ebooks formats If True stylesheets are not downloaded and processed

oldest_article = 7.0
Oldest article to download from this news source. In days.

preprocess_regexps = []
List of *regexp* substitution rules to run on the downloaded *HTML*. Each element of the list should be a two element tuple. The first element of the tuple should be a compiled regular expression and the second a callable that takes a single match object and returns a string to replace the match. For example:

```
preprocess_regexps = [
   (re.compile(r'<!--Article ends here-->.*</body>', re.DOTALL|re.IGNORECASE),
    lambda match: '</body>'),
]
```

will remove everythong from *<!–Article ends here–>* to *</body>*.

publication_type = 'unknown'
Publication type Set to newspaper, magazine or blog. If set to None, no publication type metadata will be written to the opf file.

recipe_disabled = None
Set to a non empty string to disable this recipe The string will be used as the disabled message

recursions = 0
Number of levels of links to follow on article webpages

remove_attributes = []
List of attributes to remove from all tags For example:

```
remove_attributes = ['style', 'font']
```

remove_empty_feeds = False
If True empty feeds are removed from the output. This option has no effect if parse_index is overridden in the sub class. It is meant only for recipes that return a list of feeds using *feeds* or `get_feeds()` (page 361). It is also used if you use the ignore_duplicate_articles option.

remove_javascript = True
Convenient flag to strip all javascript tags from the downloaded HTML

remove_tags = []
List of tags to be removed. Specified tags are removed from downloaded HTML. A tag is specified as a dictionary of the form:

```
{
 name      : 'tag name',   #e.g. 'div'
 attrs     : a dictionary, #e.g. {class: 'advertisment'}
}
```

All keys are optional. For a full explanantion of the search criteria, see Beautiful Soup[21] A common example:

```
remove_tags = [dict(name='div', attrs={'class':'advert'})]
```

This will remove all * * tags and all their children from the downloaded *HTML*.

remove_tags_after = None
Remove all tags that occur after the specified tag. For the format for specifying a tag see `BasicNewsRecipe.remove_tags` (page 367). For example:

```
remove_tags_after = [dict(id='content')]
```

will remove all tags after the first element with *id="content"*.

remove_tags_before = None
Remove all tags that occur before the specified tag. For the format for specifying a tag see `BasicNewsRecipe.remove_tags` (page 367). For example:

[21] http://www.crummy.com/software/BeautifulSoup/documentation.html#Thebasicfindmethod:findAll(name,attrs,recursive,text,limit,**kwargs)

```
remove_tags_before = dict(id='content')
```

will remove all tags before the first element with *id="content"*.

requires_version = (0, 6, 0)
Minimum calibre version needed to use this recipe

reverse_article_order = False
Reverse the order of articles in each feed

simultaneous_downloads = 5
Number of simultaneous downloads. Set to 1 if the server is picky. Automatically reduced to 1 if `BasicNewsRecipe.delay` (page 365) > 0

summary_length = 500
Max number of characters in the short description

template_css = u'\n .article_date {\n color: gray; font-family: monospace;\n }\n\n .article_description {\n text-indent:
The CSS that is used to style the templates, i.e., the navigation bars and the Tables of Contents. Rather than overriding this variable, you should use *extra_css* in your recipe to customize look and feel.

timefmt = ' [%a, %d %b %Y]'
The format string for the date shown on the first page. By default: Day_Name, Day_Number Month_Name Year

timeout = 120.0
Timeout for fetching files from server in seconds

title = u'Unknown News Source'
The title to use for the ebook

use_embedded_content = None
Normally we try to guess if a feed has full articles embedded in it based on the length of the embedded content. If *None*, then the default guessing is used. If *True* then the we always assume the feeds has embedded content and if *False* we always assume the feed does not have embedded content.

use_javascript_to_login = False
If you set this True, then calibre will use javascript to login to the website. This is needed for some websites that require the use of javascript to login. If you set this to True you must implement the `javascript_login()` (page 362) method, to do the actual logging in.

1.3 The Ebook Viewer

calibre includes a built-in ebook viewer that can view all the major ebook formats. The viewer is highly customizable and has many advanced features.

- Starting the viewer (page 303)
- Navigating around an ebook (page 303)
- Customizing the look and feel of your reading experience (page 304)
- Dictionary lookup (page 305)
- Copying text and images (page 305)

1.3.1 Starting the viewer

You can view any of the books in your calibre library by selecting the book and pressing the View button. This will open up the book in the ebook viewer. You can also launch the viewer by itself from the Start menu in Windows or using the command **ebook-viewer** in Linux and OS X (you have to install the command line tools on OS X first by going to *Preferences->Advanced->Miscellaneous*).

1.3.2 Navigating around an ebook

You can "turn pages" in a book by using the *Page Next* and *Page Previous* buttons , or by pressing the Page Down/Page Up keys. Unlike most ebook viewers, calibre does not force you to view books in paged mode. You can scroll by amounts less than a page by using the scroll bar or various customizable keyboard shortcuts.

Bookmarks

When you are in the middle of a book and close the viewer, it will remember where you stopped reading and return there the next time you open the book. You can also set bookmarks in the book by using the Bookmark button . When viewing EPUB format books, these bookmarks are actually saved in the EPUB file itself. You can add bookmarks, then send the file to a friend. When they open the file, they will be able to see your bookmarks.

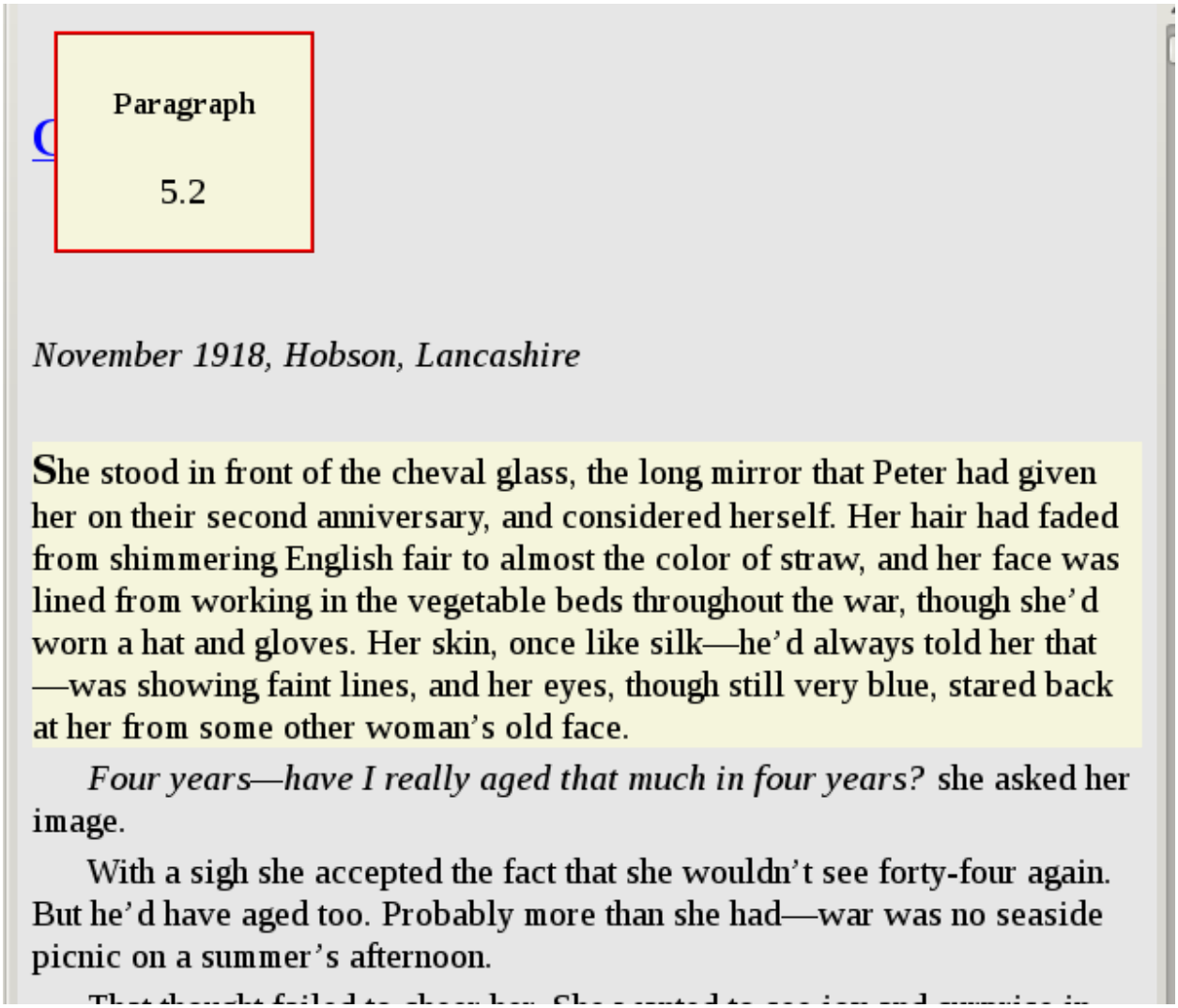
Table of Contents

If the book you are reading defines a Table of Contents, you can access it by pressing the Table of Contents button . This will bring up a list of sections in the book. You can click on any of them to jump to that portion of the book.

Navigating by location

Ebooks, unlike paper books, have no concept of pages. Instead, as you read through the book, you will notice that your position in the book is displayed in the upper left corner in a box like this . This is both your current position and the total length of the book. These numbers are independent of the screen size and font size you are viewing the book at, and they play a similar role to page numbers in paper books. You can enter any number you like to go to the corresponding location in the book.

calibre also has a very handy reference mode. You can turn it on by clicking the Reference Mode button . Once you do this, every time you move your mouse over a paragraph, calibre will display a unique number made up of the section and paragraph numbers.



You can use this number to unambiguously refer to parts of the books when discussing it with friends or referring to it in other works. You can enter these numbers in the box marked Go to at the top of the window to go to a particular reference location.

If you click on links inside the ebook to take you to different parts of the book, such as an endnote, you can use the back and forward buttons in the top left corner to return to where you were. These buttons behave just like those in a web browser.

1.3.3 Customizing the look and feel of your reading experience

You can change font sizes on the fly by using the font size buttons . You can also make the viewer full screen by pressing the Full Screen button . By clicking the Preferences button , you can change the default fonts used by the viewer to ones you like as well as the default font size when the viewer starts up.

More advanced customization can be achieved by the User Stylesheet setting. This is a stylesheet you can set that will be applied to every book. Using it you can do things like have white text on a black background, change paragraph styles, text justification, etc. For examples of custom stylesheets used by calibre's users, see the forums[22].

1.3.4 Dictionary lookup

You can look up the meaning of words in the current book by right clicking on a word. calibre uses the publicly available dictionary server at `dict.org` to look up words. The definition is displayed in a small box at the bottom of the screen.

1.3.5 Copying text and images

You can select text and images by dragging the content with your mouse and then right clicking to copy to the clipboard. The copied material can be pasted into another application as plain text and images.

1.4 Ebook Conversion

calibre has a conversion system that is designed to be very easy to use. Normally, you just add a book to calibre, click convert and calibre will try hard to generate output that is as close as possible to the input. However, calibre accepts a very large number of input formats, not all of which are as suitable as others for conversion to ebooks. In the case of such input formats, or if you just want greater control over the conversion system, calibre has a lot of options to fine tune the conversion process. Note however that calibre's conversion system is not a substitute for a full blown ebook editor. To edit ebooks, I would recommend first converting them to EPUB using calibre and then using a dedicated EPUB editor, like Sigil[23] to get the book into perfect shape. You can then use the edited EPUB as input for conversion into other formats in calibre.

This document will refer mainly to the conversion settings as found in the conversion dialog, pictured below. All these settings are also available via command line interface to conversion, documented at *ebook-convert* (page 486). In calibre, you can obtain help on any individual setting by holding your mouse over it, a tooltip will appear describing the setting.



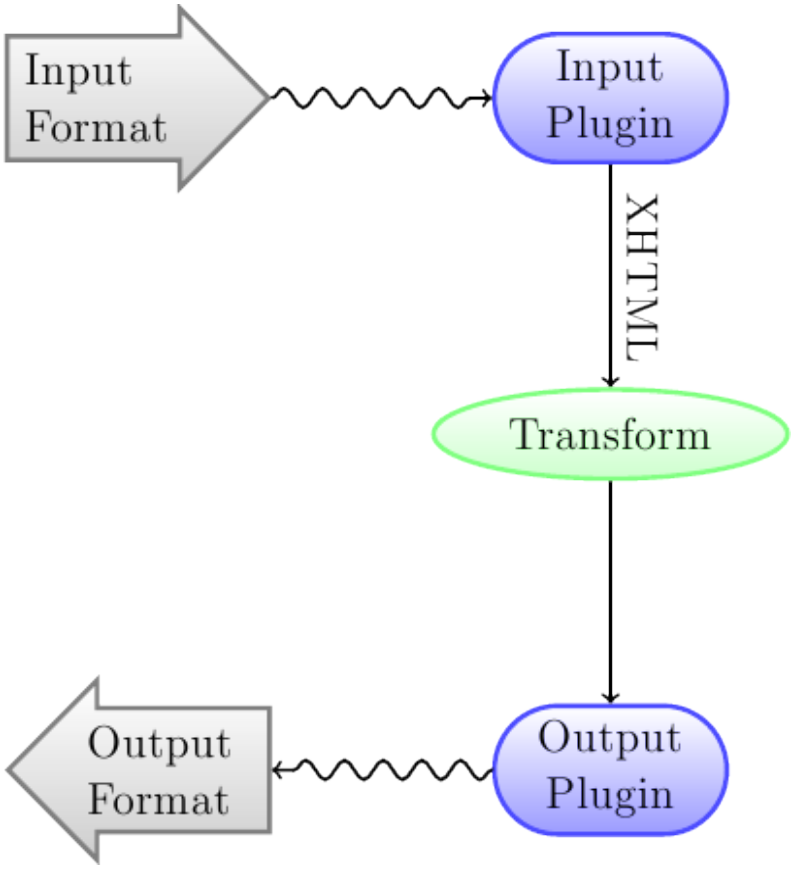
[22] http://www.mobileread.com/forums/showthread.php?t=51500
[23] http://code.google.com/p/sigil/

Contents

- Introduction (page 306)
- Look & Feel (page 307)
- Page Setup (page 309)
- Heuristic Processing (page 310)
- Search & Replace (page 311)
- Structure Detection (page 311)
- Table of Contents (page 312)
- Using images as chapter titles when converting HTML input documents (page 314)
- How options are set/saved for Conversion (page 314)
- Format specific tips (page 314)

1.4.1 Introduction

The first thing to understand about the conversion system is that it is designed as a pipeline. Schematically, it looks like this:



The input format is first converted to XHTML by the appropriate *Input Plugin*. This HTML is then *transformed*. In the last step, the processed XHTML is converted to the specified output format by the appropriate *Output Plugin*. The results of the conversion can vary greatly, based on the input format. Some formats convert much better than others. A list of the best source formats for conversion is available *here* (page 322).

The transforms that act on the XHTML output are where all the work happens. There are various transforms, for example, to insert book metadata as a page at the start of the book, to detect chapter headings and automatically create

a Table of Contents, to proportionally adjust font sizes, et cetera. It is important to remeber that all the transforms act on the XHTML output by the *Input Plugin*, not on the input file itself. So, for example, if you ask calibre to convert an RTF file to EPUB, it will first be converted to XHTML internally, the various transforms will be applied to the XHTML and then the *Output Plugin* will create the EPUB file, automatically generating all metadata, Table of Contents, et cetera.

You can see this process in action by using the debug option  . Just specify the path to a directory for the debug output. During conversion, calibre will place the XHTML generated by the various stages of the conversion pipeline in different sub-directories. The four sub-directories are:

Table 1.2: Stages of the conversion pipeline

Directory	Description
input	This contains the HTML output by the Input Plugin. Use this to debug the Input Plugin.
parsed	The result of pre-processing and converting to XHTML the output from the Input Plugin. Use to debug structure detection.
structure	Post structure detection, but before CSS flattening and font size conversion. Use to debug font size conversion and CSS transforms.
processed	Just before the ebook is passed to the output plugin. Use to debug the Output Plugin.

If you want to edit the input document a little before having calibre convert it, the best thing to do is edit the files in the `input` sub-directory, then zip it up, and use the zip file as the input format for subsequent conversions. To do this use the *Edit meta information* dialog to add the zip file as a format for the book and then, in the top left corner of the conversion dialog, select ZIP as the input format.

This document will deal mainly with the various transforms that operate on the intermediate XHTML and how to control them. At the end are some tips specific to each Input/Output format.

1.4.2 Look & Feel

Contents

- Font size rescaling (page 307)
- Paragraph spacing (page 308)
- Extra CSS (page 309)
- Miscellaneous (page 309)

This group of options controls various aspects of the look and feel of the converted ebook.

Font size rescaling

One of the nicest features of the e-reading experience is the ability to easily adjust font sizes to suit individual needs and lighting conditions. calibre has sophisticated algorithms to ensure that all the books it outputs have a consistent font sizes, no matter what font sizes are specified in the input document.

The base font size of a document is the most common font size in that document, i.e., the size of the bulk of text in that document. When you specify a *Base font size*, calibre automatically rescales all font sizes in the document

proportionately, so that the most common font size becomes the specified base font size and other font sizes are rescaled appropriately. By choosing a larger base font size, you can make the fonts in the document larger and vice versa. When you set the base font size, for best results, you should also set the font size key.

Normally, calibre will automatically choose a base font size appropriate to the Output Profile you have chosen (see *Page Setup* (page 309)). However, you can override this here in case the default is not suitable for you.

The *Font size key* option lets you control how non-base font sizes are rescaled. The font rescaling algorithm works using a font size key, which is simply a comma-separated list of font sizes. The font size key tells calibre how many "steps" bigger or smaller a given font size should be compared to the base font size. The idea is that there should be a limited number of font sizes in a document. For example, one size for the body text, a couple of sizes for different levels of headings and a couple of sizes for super/sub scripts and footnotes. The font size key allows calibre to compartmentalize the font sizes in the input documents into separate "bins" corresponding to the different logical font sizes.

Let's illustrate with an example. Suppose the source document we are converting was produced by someone with excellent eyesight and has a base font size of 8pt. That means the bulk of the text in the document is sized at 8pts, while headings are somewhat larger (say 10 and 12pt) and footnotes somewhat smaller at 6pt. Now if we use the following settings:

```
Base font size : 12pt
Font size key  : 7, 8, 10, 12, 14, 16, 18, 20
```

The output document will have a base font size of 12pt, headings of 14 and 16pt and footnotes of 8pt. Now suppose we want to make the largest heading size stand out more and make the footnotes a little larger as well. To achieve this, the font key should be changed to:

```
New font size key : 7, 9, 12, 14, 18, 20, 22
```

The largest headings will now become 18pt, while the footnotes will become 9pt. You can play with these settings to try and figure out what would be optimum for you by using the font rescaling wizard, which can be accessed by clicking the little button next to the *Font size key* setting.

All the font size rescaling in the conversion can also be disabled here, if you would like to preserve the font sizes in the input document.

A related setting is *Line height*. Line height controls the vertical height of lines. By default, (a line height of 0), no manipulation of line heights is performed. If you specify a non-default value, line heights will be set in all locations that don't specify their own line heights. However, this is something of a blunt weapon and should be used sparingly. If you want to adjust the line heights for some section of the input, it's better to use the *Extra CSS* (page 309).

Paragraph spacing

Normally, paragraphs in XHTML are rendered with a blank line between them and no leading text indent. calibre has a couple of options to control this. *Remove spacing between paragraphs* forcefully ensure that all paragraphs have no inter paragraph spacing. It also sets the text indent to 1.5em (can be changed) to mark the start of every paragraph. *Insert blank line* does the opposite, guaranteeing that there is exactly one blank line between each pair of paragraphs. Both these options are very comprehensive, removing spacing, or inserting it for *all* paragraphs (technically <p> and tags). This is so that you can just set the option and be sure that it performs as advertised, irrespective of how messy the input file is. The one exception is when the input file uses hard line breaks to implement inter-paragraph spacing.

If you want to remove the spacing between all paragraphs, except a select few, don't use these options. Instead add the following CSS code to *Extra CSS* (page 309):

```
p, div { margin: 0pt; border: 0pt; text-indent: 1.5em }
.spacious { margin-bottom: 1em; text-indent: 0pt; }
```

Then, in your source document, mark the paragraphs that need spacing with *class="spacious"*. If your input document is not in HTML, use the Debug option, described in the Introduction to get HTML (use the `input` sub-directory).

Extra CSS

This option allows you to specify arbitrary CSS that will be applied to all HTML files in the input. This CSS is applied with very high priority and so should override most CSS present in the **input document** itself. You can use this setting to fine tune the presentation/layout of your document. For example, if you want all paragraphs of class *endnote* to be right aligned, just add:

```
.endnote { text-align: right }
```

or if you want to change the indentation of all paragraphs:

```
p { text-indent: 5mm; }
```

Extra CSS is a very powerful option, but you do need an understanding of how CSS works to use it to its full potential. You can use the debug pipeline option described above to see what CSS is present in your input document.

Miscellaneous

There are a few more options in this section.

No text justification Normally, if the output format supports it, calibre will force the output ebook to have *justified* text (i.e., a smooth right margin). This option will turn off this behavior, in which case whatever justification is specified in the input document will be used instead.

Linearize tables Some badly designed documents use tables to control the layout of text on the page. When converted these documents often have text that runs off the page and other artifacts. This option will extract the content from the tables and present it in a linear fashion. Note that this option linearizes *all* tables, so only use it if you are sure the input document does not use tables for legitimate purposes, like presenting tabular information.

Transliterate unicode characters Transliterate unicode characters to an ASCII representation. Use with care because this will replace unicode characters with ASCII. For instance it will replace "Михаил Горбачёв" with "Mikhail Gorbachiov". Also, note that in cases where there are multiple representations of a character (characters shared by Chinese and Japanese for instance) the representation used by the largest number of people will be used (Chinese in the previous example). This option is mainly useful if you are going to view the ebook on a device that does not have support for unicode.

Input character encoding Older documents sometimes don't specify their character encoding. When converted, this can result in non-English characters or special characters like smart quotes being corrupted. calibre tries to auto-detect the character encoding of the source document, but it does not' always succeed. You can force it to assume a particular character encoding by using this setting. *cp1252* is a common encoding for documents produced using windows software. You should also read *How do I convert my file containing non-English characters, or smart quotes?* (page 322) for more on encoding issues.

1.4.3 Page Setup

The Page Setup options are for controlling screen layout, like margins and screen sizes. There are options to setup page margins, which will be used by the Output Plugin, if the selected Output Format supports page margins. In addition, you should choose an Input profile and an Output profile. Both sets of profiles basically deal with how to interpret measurements in the input/output documents, screen sizes and default font rescaling keys.

If you know that the file you are converting was intended to be used on a particular device/software platform, choose the corresponding input profile, otherwise just choose the default input profile. If you know the files you are producing

are meant for a particular device type, choose the corresponding Output profile. In particular, for MOBI Output files, you should choose the Kindle, for LIT the Microsoft Reader and for EPUB the Sony Reader. In the case of EPUB, the Sony Reader profile will result in EPUB files that will work everywhere. However, it has some side effects, like inserting artificial section breaks to keep internal components below the size threshold, needed for SONY devices. In particular for the iPhone/Android phones, choose the SONY output profile. If you know your EPUB files will not be read on a SONY or similar device, use the default output profile. If you are producing MOBI files that are not intended for the Kindle, choose the Mobipocket books output profile.

The Output profile also controls the screen size. This will cause, for example, images to be auto-resized to be fit to the screen in some output formats. So choose a profile of a device that has a screen size similar to your device.

1.4.4 Heuristic Processing

Heuristic Processing provides a variety of functions which can be used to try and detect and correct common problems in poorly formatted input documents. Use these functions if your input document suffers from poor formatting. Because these functions rely on common patterns, be aware that in some cases an option may lead to worse results, so use with care. As an example, several of these options will remove all non-breaking-space entities, or may include false positive matches relating to the function.

Enable heuristic processing This option activates calibre's Heuristic Processing stage of the conversion pipeline. This must be enabled in order for various sub-functions to be applied

Unwrap lines Enabling this option will cause calibre to attempt to detect and correct hard line breaks that exist within a document using punctuation clues and line length. calibre will first attempt to detect whether hard line breaks exist, if they do not appear to exist calibre will not attempt to unwrap lines. The line-unwrap factor can be reduced if you want to 'force' calibre to unwrap lines.

Line-unwrap factor This option controls the algorithm calibre uses to remove hard line breaks. For example, if the value of this option is 0.4, that means calibre will remove hard line breaks from the end of lines whose lengths are less than the length of 40% of all lines in the document. If your document only has a few line breaks which need correction, then this value should be reduced to somewhere between 0.1 and 0.2.

Detect and markup unformatted chapter headings and sub headings If your document does not have chapter headings and titles formatted differently from the rest of the text, calibre can use this option to attempt detection them and surround them with heading tags. <h2> tags are used for chapter headings; <h3> tags are used for any titles that are detected.

This function will not create a TOC, but in many cases it will cause calibre's default chapter detection settings to correctly detect chapters and build a TOC. Adjust the XPath under Structure Detection if a TOC is not automatically created. If there are no other headings used in the document then setting "//h:h2" under Structure Detection would be the easiest way to create a TOC for the document.

The inserted headings are not formatted, to apply formatting use the *Extra CSS* option under the Look and Feel conversion settings. For example, to center heading tags, use the following:

```
h2, h3 { text-align: center }
```

Renumber sequences of <h1> or <h2> tags Some publishers format chapter headings using multiple <h1> or <h2> tags sequentially. calibre's default conversion settings will cause such titles to be split into two pieces. This option will re-number the heading tags to prevent splitting.

Delete blank lines between paragraphs This option will cause calibre to analyze blank lines included within the document. If every paragraph is interleaved with a blank line, then calibre will remove all those blank paragraphs. Sequences of multiple blank lines will be considered scene breaks and retained as a single paragraph. This option differs from the 'Remove Paragraph Spacing' option under 'Look and Feel' in that it actually modifies the HTML content, while the other option modifies the document styles. This option can also remove paragraphs which were inserted using calibre's 'Insert blank line' option.

Ensure scene breaks are consistently formatted With this option calibre will attempt to detect common scene-break markers and ensure that they are center aligned. 'Soft' scene break markers, i.e. scene breaks only defined by extra white space, are styled to ensure that they will not be displayed in conjunction with page breaks.

Replace scene breaks If this option is configured then calibre will replace scene break markers it finds with the replacement text specified by the user. Please note that some ornamental characters may not be supported across all reading devices.

In general you should avoid using html tags, calibre will discard any tags and use pre-defined markup. <hr /> tags, i.e. horizontal rules, and <img> tags are exceptions. Horizontal rules can optionally be specified with styles, if you choose to add your own style be sure to include the 'width' setting, otherwise the style information will be discarded. Image tags can used, but calibre does not provide the ability to add the image during conversion, this must be done after the fact using the 'Tweak Book' feature, or Sigil.

Example image tag (place the image within an 'Images' folder inside the epub after conversion):

Example horizontal rule with styles: <hr style="width:20%;padding-top: 1px;border-top: 2px ridge black;border-bottom: 2px groove black;"/>

Remove unnecessary hyphens calibre will analyze all hyphenated content in the document when this option is enabled. The document itself is used as a dictionary for analysis. This allows calibre to accurately remove hyphens for any words in the document in any language, along with made-up and obscure scientific words. The primary drawback is words appearing only a single time in the document will not be changed. Analysis happens in two passes, the first pass analyzes line endings. Lines are only unwrapped if the word exists with or without a hyphen in the document. The second pass analyzes all hyphenated words throughout the document, hyphens are removed if the word exists elsewhere in the document without a match.

Italicize common words and patterns When enabled, calibre will look for common words and patterns that denote italics and italicize them. Examples are common text conventions such as ~word~ or phrases that should generally be italicized, e.g. latin phrases like 'etc.' or 'et cetera'.

Replace entity indents with CSS indents Some documents use a convention of defining text indents using non-breaking space entities. When this option is enabled calibre will attempt to detect this sort of formatting and convert them to a 3% text indent using css.

1.4.5 Search & Replace

These options are useful primarily for conversion of PDF documents or OCR conversions, though they can also be used to fix many document specific problems. As an example, some conversions can leaves behind page headers and footers in the text. These options use regular expressions to try and detect headers, footers, or other arbitrary text and remove or replace them. Remember that they operate on the intermediate XHTML produced by the conversion pipeline. There is a wizard to help you customize the regular expressions for your document. Click the magic wand beside the expression box, and click the 'Test' button after composing your search expression. Successful matches will be highlighted in Yellow.

The search works by using a python regular expression. All matched text is simply removed from the document or replaced using the replacement pattern. The replacement pattern is optional, if left blank then text matching the search pattern will be deleted from the document. You can learn more about regular expressions and their syntax at *All about using regular expressions in calibre* (page 415).

1.4.6 Structure Detection

Structure detection involves calibre trying its best to detect structural elements in the input document, when they are not properly specified. For example, chapters, page breaks, headers, footers, etc. As you can imagine, this process

varies widely from book to book. Fortunately, calibre has very powerful options to control this. With power comes complexity, but if once you take the time to learn the complexity, you will find it well worth the effort.

Chapters and page breaks

calibre has two sets of options for *chapter detection* and *inserting page breaks*. This can sometimes be slightly confusing, as by default, calibre will insert page breaks before detected chapters as well as the locations detected by the page breaks option. The reason for this is that there are often location where page breaks should be inserted that are not chapter boundaries. Also, detected chapters can be optionally inserted into the auto generated Table of Contents.

calibre uses *XPath*, a powerful language to allow the user to specify chapter boundaries/page breaks. XPath can seem a little daunting to use at first, fortunately, there is a *XPath tutorial* (page 373) in the User Manual. Remember that Structure Detection operates on the intermediate XHTML produced by the conversion pipeline. Use the debug option described in the *Introduction* (page 306) to figure out the appropriate settings for your book. There is also a button for a XPath wizard to help with the generation of simple XPath expressions.

By default, calibre uses the following expression for chapter detection:

```
//*[((name()='h1' or name()='h2') and re:test(., 'chapter|book|section|part\s+', 'i')) or @class = 'c
```

This expression is rather complex, because it tries to handle a number of common cases simulataneously. What it means is that calibre will assume chapters start at either <*h1*> or <*h2*> tags that have any of the words *(chapter, book, section or part)* in them or that have the *class="chapter"* attribute.

A related option is *Chapter mark*, which allows you to control what calibre does when it detects a chapter. By default, it will insert a page break before the chapter. You can have it insert a ruled line instead of, or in addition to the page break. You can also have it do nothing.

The default setting for detecting page breaks is:

```
//*[name()='h1' or name()='h2']
```

which means that calibre will insert page breaks before every <*h1*> and <*h2*> tag by default.

Note: The default expressions may change depending on the input format you are converting.

Miscellaneous

There are a few more options in this section.

Insert metadata as page at start of book One of the great things about calibre is that it allows you to maintain very complete metadata about all of your books, for example, a rating, tags, comments, etc. This option will create a single page with all this metadata and insert it into the converted ebook, typically just after the cover. Think of it as a way to create your own customised book jacket.

Remove first image Sometimes, the source document you are converting includes the cover as part of the book, instead of as a separate cover. If you also specify a cover in calibre, then the converted book will have two covers. This option will simply remove the first image from the source document, thereby ensuring that the converted book has only one cover, the one specified in calibre.

1.4.7 Table of Contents

When the input document has a Table of Contents in its metadata, calibre will just use that. However, a number of older formats either do not support a metadata based Table of Contents, or individual documents do not have one. In

these cases, the options in this section can help you automatically generate a Table of Contents in the converted ebook, based on the actual content in the input document.

The first option is *Force use of auto-generated Table of Contents*. By checking this option you can have calibre override any Table of Contents found in the metadata of the input document with the auto generated one.

The default way that the creation of the auto generated Table of Contents works is that, calibre will first try to add any detected chapters to the generated table of contents. You can learn how to customize the detection of chapters in the *Structure Detection* (page 311) section above. If you do not want to include detected chapters in the generated table of contents, check the *Do not add detected chapters* option.

If less than the *Chapter threshold* number of chapters were detected, calibre will then add any hyperlinks it finds in the input document to the Table of Contents. This often works well many input documents include a hyperlinked Table of Contents right at the start. The *Number of links* option can be used to control this behavior. If set to zero, no links are added. If set to a number greater than zero, at most that number of links is added.

calibre will automatically filter duplicates from the generated Table of Contents. However, if there are some additional undesirable entries, you can filter them using the *TOC Filter* option. This is a regular expression that will match the title of entries in the generated table of contents. Whenever a match is found, it will be removed. For example, to remove all entries titles "Next" or "Previous" use:

```
Next|Previous
```

Finally, the *Level 1,2,3 TOC* options allow you to create a sophisticated multi-level Table of Contents. They are XPath expressions that match tags in the intermediate XHTML produced by the conversion pipeline. See the *Introduction* (page 306) for how to get access to this XHTML. Also read the *XPath Tutorial* (page 373), to learn how to construct XPath expressions. Next to each option is a button that launches a wizard to help with the creation of basic XPath expressions. The following simple example illustrates how to use these options.

Suppose you have an input document that results in XHTML that look like this:

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Sample document</title>
    </head>
    <body>
        <h1>Chapter 1</h1>
        ...
        <h2>Section 1.1</h2>
        ...
        <h2>Section 1.2</h2>
        ...
        <h1>Chapter 2</h1>
        ...
        <h2>Section 2.1</h2>
        ...
    </body>
</html>
```

Then, we set the options as:

```
Level 1 TOC : //h:h1
Level 2 TOC : //h:h2
```

This will result in an automatically generated two level Table of Contents that looks like:

```
Chapter 1
    Section 1.1
    Section 1.2
```

```
Chapter 2
    Section 2.1
```

Warning: Not all output formats support a multi level Table of Contents. You should first try with EPUB Output. If that works, then try your format of choice.

1.4.8 Using images as chapter titles when converting HTML input documents

Suppose you want to use an image as your chapter title, but still want calibre to be able to automatically generate a Table of Contents for you from the chapter titles. Use the following HTML markup to achieve this

```
<html>
    <body>
        <h2>Chapter 1</h2>
        <p>chapter 1 text...</p>
        <h2 title="Chapter 2"><img src="chapter2.jpg" /></h2>
        <p>chapter 2 text...</p>
    </body>
</html>
```

Set the *Level 1 TOC* setting to `//h:h2`. Then, for chapter two, calibre will take the title from the value of the `title` attribute on the `<h2>` tag, since the tag has no text.

1.4.9 How options are set/saved for Conversion

There are two places where conversion options can be set in calibre. The first is in Preferences->Conversion. These settings are the defaults for the conversion options. Whenever you try to convert a new book, the settings set here will be used by default.

You can also change settings in the conversion dialog for each book conversion. When you convert a book, calibre remembers the settings you used for that book, so that if you convert it again, the saved settings for the individual book will take precedence over the defaults set in Preferences. You can restore the individual settings to defaults by using the Restore to defaults button in the individual book conversion dialog.

When you Bulk Convert a set of books, settings are taken in the following order:

- From the defaults set in Preferences->Conversion
- From the saved conversion settings for each book being converted (if any). This can be turned off by the option in the top left corner of the Bulk Conversion dialog.
- From the settings set in the Bulk conversion dialog

Note that the final settings for each book in a Bulk Conversion will be saved and re-used if the book is converted again. Since the highest priority in Bulk Conversion is given to the settings in the Bulk Conversion dialog, these will override any book specific settings. So you should only bulk convert books together that need similar settings. The exceptions are metadata and input format specific settings. Since the Bulk Conversion dialog does not have settings for these two categories, they will be taken from book specific settings (if any) or the defaults.

Note: You can see the actual settings used during any conversion by clicking the rotating icon in the lower right corner and then double clicking the individual conversion job. This will bring up a conversion log that will contain the actual settings used, near the top.

1.4.10 Format specific tips

Here you will find tips specific to the conversion of particular formats. Options specific to particular format, whether input or output are available in the conversion dialog under their own section, for example *TXT Input* or *EPUB Output*.

Convert Microsoft Word documents

calibre does not directly convert .doc/.docx files from Microsoft Word. However, in Word, you can save the document as HTML and then convert the resulting HTML file with calibre. When saving as HTML, be sure to use the "Save as Web Page, Filtered" option as this will produce clean HTML that will convert well. Note that Word produces really messy HTML, converting it can take a long time, so be patient. Another alternative is to use the free OpenOffice. Open your .doc file in OpenOffice and save it in OpenOffice's format .odt. calibre can directly convert .odt files.

There is a Word macro package that can automate the conversion of Word documents using calibre. It also makes generating the Table of Contents much simpler. It is called BookCreator and is available for free at mobileread[24].

An easy way to generate a Table of Contents when converting a Word document is:

1. Mark your Chapters and sub-Chapters in the doc file with one of the MS built-in styles called 'Heading 1', 'Heading 2', ..., 'Heading 6'. 'Heading 1' equates to the HTML tag <h1>, 'Heading 2' to <h2> etc
2. Save the doc as Webpage-filtered (rather than Webpage) and import the html file into calibre
3. When you convert in calibre you use what you did in step 1 to set the box called 'Detect chapters at' on the Convert - Structure Detection page. For example:

- If you mark Chapters with style 'Heading 2' then set the 'Detect chapters at' box to //h:h2 This will give you a proper external metadata TOC in the converted epub.
- A slightly more complex example...if your book has Sections and Chapters and you want a 2-level nested metadata TOC. Mark the doc Sections with style 'Heading 2' and the Chapters with style 'Heading 3'. When you convert set the 'Detect chapters at' box to //h:h2|//h:h3. On the Convert - TOC page set the 'Level 1 TOC' box to //h:h2 and the 'Level 2 TOC' box to //h:h3.

Convert TXT documents

TXT documents have no well defined way to specify formatting like bold, italics, etc, or document structure like paragraphs, headings, sections and so on, but there are a variety of conventions commonly used. By default calibre attempts automatic detection of the correct formatting and markup based on those conventions.

TXT input supports a number of options to differentiate how paragraphs are detected.

Paragraph Style: Auto Analyzes the text file and attempts to automatically determine how paragraphs are defined. This option will generally work fine, if you achieve undesirable results try one of the manual options.

Paragraph Style: Block Assumes one or more blank lines are a paragraph boundary:

```
This is the first.

This is the
second paragraph.
```

Paragraph Style: Single Assumes that every line is a paragraph:

[24] http://www.mobileread.com/forums/showthread.php?t=28313

```
This is the first.
This is the second.
This is the third.
```

Paragraph Style: Print Assumes that every paragraph starts with an indent (either a tab or 2+ spaces). Paragraphs end when the next line that starts with an indent is reached:

```
  This is the
first.
  This is the second.

  This is the
third.
```

Paragraph Style: Unformatted Assumes that the document has no formatting, but does use hard line breaks. Punctuation and median line length are used to attempt to re-create paragraphs.

Formatting Style: Auto Attemtps to detect the type of formatting markup being used. If no markup is used then heuristic formatting will be applied.

Formatting Style: Heuristic Analyzes the document for common chapter headings, scene breaks, and italicized words and applies the appropriate html markup during conversion.

Formatting Style: Markdown calibre also supports running TXT input though a transformation preprocessor known as markdown. Markdown allows for basic formatting to be added to TXT documents, such as bold, italics, section headings, tables, lists, a Table of Contents, etc. Marking chapter headings with a leading # and setting the chapter XPath detection expression to "//h:h1" is the easiest way to have a proper table of contents generated from a TXT document. You can learn more about the markdown syntax at daringfireball[25].

Formatting Style: None Applies no special formatting to the text, the document is converted to html with no other changes.

Convert PDF documents

PDF documents are one of the worst formats to convert from. They are a fixed page size and text placement format. Meaning, it is very difficult to determine where one paragraph ends and another begins. calibre will try to unwrap paragraphs using a configurable, *Line Un-Wrapping Factor*. This is a scale used to determine the length at which a line should be unwrapped. Valid values are a decimal between 0 and 1. The default is 0.45, just under the median line length. Lower this value to include more text in the unwrapping. Increase to include less. You can adjust this value in the conversion settings under *PDF Input*.

Also, they often have headers and footers as part of the document that will become included with the text. Use the Search and Replace panel to remove headers and footers to mitigate this issue. If the headers and footers are not removed from the text it can throw off the paragraph unwrapping. To learn how to use the header and footer removal options, read *All about using regular expressions in calibre* (page 415).

Some limitations of PDF input are:

- Complex, multi-column, and image based documents are not supported.
- Extraction of vector images and tables from within the document is also not supported.
- Some PDFs use special glyphs to represent ll or ff or fi, etc. Conversion of these may or may not work depending on just how they are represented internally in the PDF.
- Links and Tables of Contents are not supported

[25] http://daringfireball.net/projects/markdown/syntax

- PDFs that use embedded non-unicode fonts to represent non-English characters will result in garbled output for those characters
- Some PDFs are made up of photographs of the page with OCRed text behind them. In such cases calibre uses the OCRed text, which can be very different from what you see when you view the PDF file
- PDFs that are used to display complex text, like right to left languages and math typesetting will not convert correctly

To re-iterate **PDF is a really, really bad** format to use as input. If you absolutely must use PDF, then be prepared for an output ranging anywhere from decent to unusable, depending on the input PDF.

Comic Book Collections

A comic book collection is a .cbc file. A .cbc file is a zip file that contains other CBZ/CBR files. In addition the .cbc file must contain a simple text file called comics.txt, encoded in UTF-8. The comics.txt file must contain a list of the comics files inside the .cbc file, in the form filename:title, as shown below:

```
one.cbz:Chapter One
two.cbz:Chapter Two
three.cbz:Chapter Three
```

The .cbc file will then contain:

```
comics.txt
one.cbz
two.cbz
three.cbz
```

calibre will automatically convert this .cbc file into a ebook with a Table of Contents pointing to each entry in comics.txt.

EPUB advanced formatting demo

Various advanced formatting for EPUB files is demonstrated in this demo file[26]. The file was created from hand coded HTML using calibre and is meant to be used as a template for your own EPUB creation efforts.

The source HTML it was created from is available demo.zip[27]. The settings used to create the EPUB from the ZIP file are:

```
ebook-convert demo.zip .epub -vv --authors "Kovid Goyal" --language en --level1-toc '//*[@class="titl
```

Note that because this file explores the potential of EPUB, most of the advanced formatting is not going to work on readers less capable than calibre's built-in EPUB viewer.

Convert ODT documents

calibre can directly convert ODT (OpenDocument Text) files. You should use styles to format your document and minimize the use of direct formatting. When inserting images into your document you need to anchor them to the paragraph, images anchored to a page will all end up in the front of the conversion.

To enable automatic detection of chapters, you need to mark them with the build-in styles called 'Heading 1', 'Heading 2', ..., 'Heading 6' ('Heading 1' equates to the HTML tag <h1>, 'Heading 2' to <h2> etc). When you convert in calibre you can enter which style you used into the 'Detect chapters at' box. Example:

[26] http://calibre-ebook.com/downloads/demos/demo.epub
[27] http://calibre-ebook.com/downloads/demos/demo.zip

- If you mark Chapters with style 'Heading 2', you have to set the 'Detect chapters at' box to `//h:h2`
- For a nested TOC with Sections marked with 'Heading 2' and the Chapters marked with 'Heading 3' you need to enter `//h:h2|//h:h3`. On the Convert - TOC page set the 'Level 1 TOC' box to `//h:h2` and the 'Level 2 TOC' box to `//h:h3`.

Well-known document properties (Title, Keywords, Description, Creator) are recognized and calibre will use the first image (not to small, and with good aspect-ratio) as the cover image.

There is also an advanced property conversion mode, which is activated by setting the custom property `opf.metadata` ('Yes or No' type) to Yes in your ODT document (File->Properties->Custom Properties). If this property is detected by calibre, the following custom properties are recognized (`opf.authors` overrides document creator):

```
opf.titlesort
opf.authors
opf.authorsort
opf.publisher
opf.pubdate
opf.isbn
opf.language
opf.series
opf.seriesindex
```

In addition to this, you can specify the picture to use as the cover by naming it `opf.cover` (right click, Picture->Options->Name) in the ODT. If no picture with this name is found, the 'smart' method is used. As the cover detection might result in double covers in certain output formats, the process will remove the paragraph (only if the only content is the cover!) from the document. But this works only with the named picture!

To disable cover detection you can set the custom property `opf.nocover` ('Yes or No' type) to Yes in advanced mode.

1.5 Editing Ebook Metadata

Contents

- Editing the metadata of one book at a time (page 318)
  - Downloading metadata (page 319)
  - Managing book formats (page 319)
  - All about covers (page 319)
- Editing the metadata of many books at a time (page 319)
  - Search and replace (page 320)
  - Bulk downloading of metadata (page 321)

Ebooks come in all shapes and sizes and more often than not, their metadata (things like title/author/series/publisher) is incomplete or incorrect. The simplest way to change metadata in calibre is to simply double click on an entry and type in the correct replacement. For more sophisticated, "power editing" use the edit metadata tools discussed below.

1.5.1 Editing the metadata of one book at a time

Click the book you want to edit and then click the *Edit metadata* button or press the `E` key. A dialog opens that allows you to edit all aspects of the metadata. It has various features to make editing faster and more efficient. A list of the commonly used tips:

- You can click the button in between title and authors to swap them automatically.
- You can click the button next to author sort to have calibre automatically fill it in using the sort values stored with each author. Use the *Manage authors* dialog to see and change the authors' sort values. This dialog can be opened by clicking and holding the button next to author sort.
- You can click the button next to tags to use the Tag Editor to manage the tags associated with the book.
- The ISBN box will have a red background if you enter an invalid ISBN. It will be green for valid ISBNs
- The author sort box will be red if the author sort value differs from what calibre thinks it should be.

Downloading metadata

The nicest feature of the edit metadata dialog is its ability to automatically fill in many metadata fields by getting metadata from various websites. Currently, calibre uses isbndb.com, Google Books, Amazon and Library Thing. The metadata download can fill in Title, author, series, tags, rating, description and ISBN for you.

To use the download, fill in the title and author fields and click the *Fetch metadata* button. calibre will present you with a list of books that most closely match the title and author. If you fill in the ISBN field first, it will be used in preference to the title and author. If no matches are found, try making your search a little less specific by including only some key words in the title and only the author last name.

Managing book formats

In calibre, a single book entry can have many different *formats* associated with it. For example you may have obtained the Complete Works of Shakespeare in EPUB format and later converted it to MOBI to read on your Kindle. calibre automatically manages multiple formats for you. In the *Available formats* section of the Edit metadata dialog, you can manage these formats. You can add a new format, delete an existing format and also ask calibre to set the metadata and cover for the book entry from the metadata in one of the formats.

All about covers

You can ask calibre to download book covers for you, provided the book has a known ISBN. Alternatively you can specify a file on your computer to use as the cover. calibre can even generate a default cover with basic metadata on it for you. You can drag and drop images onto the cover to change it and also right click to copy/paste cover images.

In addition, there is a button to automatically trim borders from the cover, in case your cover image has an ugly border.

1.5.2 Editing the metadata of many books at a time

First select the books you want to edit by holding Ctrl or Shift and clicking on them. If you select more than one book, clicking the *Edit metadata* button will cause a new *Bulk* metadata edit dialog to open. Using this dialog, you can quickly set the author/publisher/rating/tags/series etc of a bunch of books to the same value. This is particularly useful if you have just imported a number of books that have some metadata in common. This dialog is very powerful, for example, it has a Search and Replace tab that you can use to perform bulk operations on metadata and even copy metadata from one column to another.

The normal edit metadata dialog also has Next and Previous buttons that you can use to edit the metadata of several books one after the other.

Search and replace

The Bulk metadata edit dialog allows you to perform arbitrarily powerful search and replace operations on the selected books. By default it uses a simple text search and replace, but it also support *regular expressions*. For more on regular expressions, see *All about using regular expressions in calibre* (page 415).

As noted above, there are two search and replace modes: character match and regular expression. Character match will look in the *Search field* you choose for the characters you type in the *search for* box and replace those characters with what you type in the *replace with* box. Each occurance of the search characters in the field will be replaced. For example, assume the field being searched contains *a bad cat*. if you search for *a* to be replaced with *HELLO*, then the result will be *HELLO bHELLOd cHELLOt*.

If the field you are searching on is a *multiple* field like tags, then each tag is treated separately. For example, if your tags contain *Horror, Scary*, the search expression *r,* will not match anything because the expression will first be applied to *Horror* and then to *Scary*.

If you want the search to ignore upper/lowercase differences, uncheck the *Case sensitive* box.

You can have calibre change the case of the result (information after the replace has happened) by choosing one of the functions from the *Apply function after replace* box. The operations available are:

- *Lower case* – change all the characters in the field to lower case
- *Upper case* – change all the characters in the field to upper case
- *Title case* – capitalize each word in the result.

The *Your test* box is provided for you to enter text to check that search/replace is doing what you want. In the majority of cases the book test boxes will be sufficient, but it is possible that there is a case you want to check that isn't shown in these boxes. Enter that case into *Your test*.

Regular expression mode has some differences from character mode, beyond (of course) using regular expressions. The first is that functions are applied to the parts of the string matched by the search string, not the entire field. The second is that functions apply to the replacement string, not to the entire field.

The third and most important is that the replace string can make reference to parts of the search string by using backreferences. A backreference is `\\n` where n is an integer that refers to the n'th parenthesized group in the search expression. For example, given the same example as above, *a bad cat*, a search expression *a (...) (...)*, and a replace expression *a \2 \1*, the result will be *a cat bad*. Please see the *All about using regular expressions in calibre* (page 415) for more information on backreferences.

One useful pattern: assume you want to change the case of an entire field. The easiest way to do this is to use character mode, but lets further assume you want to use regular expression mode. The search expression should be *(.*)* the replace expression should be *\1*, and the desired case change function should be selected.

Finally, in regular expression mode you can copy values from one field to another. Simply make the source and destination field different. The copy can replace the destination field, prepend to the field (add to the front), or append to the field (add at the end). The 'use comma' checkbox tells calibre to (or not to) add a comma between the text and the destination field in prepend and append modes. If the destination is multiple (e.g., tags), then you cannot uncheck this box.

Search and replace is done after all the other metadata changes in the other tabs are applied. This can lead to some confusion, because the test boxes will show the information before the other changes, but the operation will be applied after the other changes. If you have any doubts about what is going to happen, do not mix search/replace with other changes.

Bulk downloading of metadata

If you want to download the metadata for multiple books at once, right-click the *Edit metadata* button and select *Download metadata*. You can choose to download only metadata, only covers, or both.

1.6 Frequently Asked Questions

Contents

- Ebook Format Conversion (page 321)
- Device Integration (page 324)
- Library Management (page 331)
- Content From The Web (page 335)
- Miscellaneous (page 335)

1.6.1 Ebook Format Conversion

Contents

- What formats does calibre support conversion to/from? (page 321)
- What are the best source formats to convert? (page 322)
- I converted a PDF file, but the result has various problems? (page 322)
- How do I convert my file containing non-English characters, or smart quotes? (page 322)
- What's the deal with Table of Contents in MOBI files? (page 322)
- The covers for my MOBI files have stopped showing up in Kindle for PC/Kindle for Android/iPad etc. (page 323)
- How do I convert a collection of HTML files in a specific order? (page 323)
- The EPUB I produced with calibre is not valid? (page 324)
- How do I use some of the advanced features of the conversion tools? (page 324)

What formats does calibre support conversion to/from?

calibre supports the conversion of many input formats to many output formats. It can convert every input format in the following list, to every output format.

Input Formats: CBZ, CBR, CBC, CHM, DJVU, EPUB, FB2, HTML, HTMLZ, LIT, LRF, MOBI, ODT, PDF, PRC, PDB, PML, RB, RTF, SNB, TCR, TXT, TXTZ

Output Formats: AZW3, EPUB, FB2, OEB, LIT, LRF, MOBI, HTMLZ, PDB, PML, RB, PDF, RTF, SNB, TCR, TXT, TXTZ

Note: PRC is a generic format, calibre supports PRC files with TextRead and MOBIBook headers. PDB is also a generic format. calibre supports eReder, Plucker, PML and zTxt PDB files. DJVU support is only for converting DJVU files that contain embedded text. These are typically generated by OCR software. MOBI books can be of two types Mobi6 and KF8. calibre fully supports both. MOBI files often have .azw or .azw3 file extensions

---

What are the best source formats to convert?

In order of decreasing preference: LIT, MOBI, AZW, EPUB, AZW3, FB2, HTML, PRC, RTF, PDB, TXT, PDF

I converted a PDF file, but the result has various problems?

PDF is a terrible format to convert from. For a list of the various issues you will encounter when converting PDF, see: *Convert PDF documents* (page 316).

How do I convert my file containing non-English characters, or smart quotes?

There are two aspects to this problem:

1. Knowing the encoding of the source file: calibre tries to guess what character encoding your source files use, but often, this is impossible, so you need to tell it what encoding to use. This can be done in the GUI via the *Input character encoding* field in the *Look & Feel* section. The command-line tools all have an `--input-encoding` option.
2. When adding HTML files to calibre, you may need to tell calibre what encoding the files are in. To do this go to *Preferences->Advanced->Plugins->File Type plugins* and customize the HTML2Zip plugin, telling it what encoding your HTML files are in. Now when you add HTML files to calibre they will be correctly processed. HTML files from different sources often have different encodings, so you may have to change this setting repeatedly. A common encoding for many files from the web is `cp1252` and I would suggest you try that first. Note that when converting HTML files, leave the input encoding setting mentioned above blank. This is because the HTML2ZIP plugin automatically converts the HTML files to a standard encoding (utf-8).
3. Embedding fonts: If you are generating an LRF file to read on your SONY Reader, you are limited by the fact that the Reader only supports a few non-English characters in the fonts it comes pre-loaded with. You can work around this problem by embedding a unicode-aware font that supports the character set your file uses into the LRF file. You should embed atleast a serif and a sans-serif font. Be aware that embedding fonts significantly slows down page-turn speed on the reader.

What's the deal with Table of Contents in MOBI files?

The first thing to realize is that most ebooks have two tables of contents. One is the traditional Table of Contents, like the TOC you find in paper books. This Table of Contents is part of the main document flow and can be styled however you like. This TOC is called the *content TOC*.

Then there is the *metadata TOC*. A metadata TOC is a TOC that is not part of the book text and is typically accessed by some special button on a reader. For example, in the calibre viewer, you use the Show Table of Contents button to see this TOC. This TOC cannot be styled by the book creator. How it is represented is up to the viewer program.

In the MOBI format, the situation is a little confused. This is because the MOBI format, alone amongst mainstream ebook formats, *does not* have decent support for a metadata TOC. A MOBI book simulates the presence of a metadata TOC by putting an *extra* content TOC at the end of the book. When you click Goto Table of Contents on your Kindle, it is to this extra content TOC that the Kindle takes you.

Now it might well seem to you that the MOBI book has two identical TOCs. Remember that one is semantically a content TOC and the other is a metadata TOC, even though both might have exactly the same entries and look the same. One can be accessed directly from the Kindle's menus, the other cannot.

When converting to MOBI, calibre detects the *metadata TOC* in the input document and generates an end-of-file TOC in the output MOBI file. You can turn this off by an option in the MOBI Output settings. You can also tell calibre whether to put it and the start or the end of the book via an option in the MOBI Output settings. Remember this TOC is semantically a *metadata TOC*, in any format other than MOBI it *cannot not be part of the text*. The fact that it is part of the text in MOBI is an accident caused by the limitations of MOBI. If you want a TOC at a particular location in your document text, create one by hand. So we strongly recommend that you leave the default as it is, i.e. with the metadata TOC at the end of the book.

If you have a hand edited TOC in the input document, you can use the TOC detection options in calibre to automatically generate the metadata TOC from it. See the conversion section of the User Manual for more details on how to use these options.

Finally, I encourage you to ditch the content TOC and only have a metadata TOC in your ebooks. Metadata TOCs will give the people reading your ebooks a much superior navigation experience (except on the Kindle, where they are essentially the same as a content TOC).

The covers for my MOBI files have stopped showing up in Kindle for PC/Kindle for Android/iPad etc.

This is caused by a bug in the Amazon software. You can work around it by going to Preferences->Output Options->MOBI output and setting the "Enable sharing of book content" option. If you are reconverting a previously converted book, you will also have to enable the option in the conversion dialog for that individual book (as per book conversion settings are saved and take precedence).

Note that doing this will mean that the generated MOBI will show up under personal documents instead of Books on the Kindle Fire and Amazon whispersync will not work, but the covers will. It's your choice which functionality is more important to you. I encourage you to contact Amazon and ask them to fix this bug.

How do I convert a collection of HTML files in a specific order?

In order to convert a collection of HTML files in a specific oder, you have to create a table of contents file. That is, another HTML file that contains links to all the other files in the desired order. Such a file looks like:

```
<html>
   <body>
     <h1>Table of Contents</h1>
     <p style="text-indent:0pt">
        <a href="file1.html">First File</a><br/>
        <a href="file2.html">Second File</a><br/>
        .
        .
        .
     </p>
   </body>
</html>
```

Then just add this HTML file to the GUI and use the convert button to create your ebook.

Note: By default, when adding HTML files, calibre follows links in the files in *depth first* order. This means that if file A.html links to B.html and C.html and D.html, but B.html also links to D.html, then the files will be in the order A.html, B.html, D.html, C.html. If instead you want the order to be A.html, B.html, C.html, D.html then you must tell calibre to add your files in *breadth first* order. Do this by going to Preferences->Plugins and customizing the HTML to ZIP plugin.

The EPUB I produced with calibre is not valid?

calibre does not guarantee that an EPUB produced by it is valid. The only guarantee it makes is that if you feed it valid XHTML 1.1 + CSS 2.1 it will output a valid EPUB. calibre is designed for ebook consumers, not producers. It tries hard to ensure that EPUBs it produces actually work as intended on a wide variety of devices, a goal that is incompatible with producing valid EPUBs, and one that is far more important to the vast majority of its users. If you need a tool that always produces valid EPUBs, calibre is not for you.

How do I use some of the advanced features of the conversion tools?

You can get help on any individual feature of the converters by mousing over it in the GUI or running `ebook-convert`

- html-demo.zip[28]

1.6.2 Device Integration

Contents

- What devices does calibre support? (page 324)
- How can I help get my device supported in calibre? (page 324)
- My device is not being detected by calibre? (page 325)
- My device is non-standard or unusual. What can I do to connect to it? (page 325)
- How does calibre manage collections on my SONY reader? (page 325)
- Can I use both calibre and the SONY software to manage my reader? (page 326)
- How do I use calibre with my iPad/iPhone/iPod touch? (page 327)
- How do I use calibre with my Android phone/tablet or Kindle Fire HD? (page 328)
- Can I access my calibre books using the web browser in my Kindle or other reading device? (page 329)
- I get the error message "Failed to start content server: Port 8080 not free on '0.0.0.0"'? (page 329)
- I cannot send emails using calibre? (page 329)
- Why is my device not detected in linux? (page 329)
- My device is getting mounted read-only in linux, so calibre cannot connect to it? (page 330)
- Why does calibre not support collections on the Kindle or shelves on the Nook? (page 330)
- I am getting an error when I try to use calibre with my Kobo Touch/Glo/etc.? (page 330)

What devices does calibre support?

calibre can directly connect to all the major (and most of the minor) ebook reading devices, smarthphones, tablets, etc. In addition, using the *Connect to folder* function you can use it with any ebook reader that exports itself as a USB disk. You can even connect to Apple devices (via iTunes), using the *Connect to iTunes* function.

How can I help get my device supported in calibre?

If your device appears as a USB disk to the operating system, adding support for it to calibre is very easy. We just need some information from you:

- Complete list of ebook formats that your device supports.
- Is there a special directory on the device in which all ebook files should be placed? Also does the device detect files placed in sub directories?
- We also need information about your device that calibre will collect automatically. First, if your device supports SD cards, insert them. Then connect your device to the computer. In calibre go to *Preferences->Advanced->Miscellaneous* and click the "Debug device detection" button. This will create some debug output. Copy it to a file and repeat the process, this time with your device disconnected from your computer.
- Send both the above outputs to us with the other information and we will write a device driver for your device.

[28] http://calibre-ebook.com/downloads/html-demo.zip

Once you send us the output for a particular operating system, support for the device in that operating system will appear in the next release of calibre. To send us the output, open a bug report and attach the output to it. See calibre bugs[29].

My device is not being detected by calibre?

Follow these steps to find the problem:

- Make sure that you are connecting only a single device to your computer at a time. Do not have another calibre supported device like an iPhone/iPad etc. at the same time.
- If you are connecting an Apple iDevice (iPad, iPod Touch, iPhone), use the 'Connect to iTunes' method in the 'Getting started' instructions in Calibre + Apple iDevices: Start here[30].
- Make sure you are running the latest version of calibre. The latest version can always be downloaded from the calibre website[31]. You can tell what version of calibre you are currently running by looking at the bottom line of the main calibre window.
- Ensure your operating system is seeing the device. That is, the device should show up in Windows Explorer (in Windows) or Finder (in OS X).
- In calibre, go to Preferences->Ignored Devices and check that your device is not being ignored
- If all the above steps fail, go to Preferences->Miscellaneous and click debug device detection with your device attached and post the output as a ticket on the calibre bug tracker[32].

My device is non-standard or unusual. What can I do to connect to it?

In addition to the *Connect to Folder* function found under the Connect/Share button, calibre provides a `User Defined` device plugin that can be used to connect to any USB device that shows up as a disk drive in your operating system. Note: on windows, the device must have a drive letter for calibre to use it. See the device plugin `Preferences -> Plugins -> Device Plugins -> User Defined` and `Preferences -> Miscellaneous -> Get information to setup the user defined device` for more information. Note that if you are using the user defined plugin for a device normally detected by a builtin calibre plugin, you must disable the builtin plugin first, so that your user defined plugin is used instead.

How does calibre manage collections on my SONY reader?

When calibre connects with the reader, it retrieves all collections for the books on the reader. The collections of which books are members are shown on the device view.

When you send a book to the reader, calibre will add the book to collections based on the metadata for that book. By default, collections are created from tags and series. You can control what metadata is used by going to *Preferences->Advanced->Plugins->Device Interface plugins* and customizing the SONY device interface plugin. If you remove all values, calibre will not add the book to any collection.

Collection management is largely controlled by the 'Metadata management' option found at *Preferences->Import/Export->Sending books to devices*. If set to 'Manual' (the default), managing collections is left to the user; calibre will not delete already existing collections for a book on your reader when you resend the book to the reader, but calibre will add the book to collections if necessary. To ensure that the collections for a book are based only on current calibre metadata, first delete the books from the reader, then resend the books. You can edit collections directly on the device view by double-clicking or right-clicking in the collections column.

[29] http://calibre-ebook.com/bugs
[30] http://www.mobileread.com/forums/showthread.php?t=118559
[31] http://calibre-ebook.com/download
[32] http://bugs.calibre-ebook.com

If 'Metadata management' is set to 'Only on send', then calibre will manage collections more aggressively. Collections will be built using calibre metadata exclusively. Sending a book to the reader will correct the collections for that book so its collections exactly match the book's metadata, adding and deleting collections as necessary. Editing collections on the device view is not permitted, because collections not in the metadata will be removed automatically.

If 'Metadata management' is set to 'Automatic management', then calibre will update metadata and collections both when the reader is connected and when books are sent. When calibre detects the reader and generates the list of books on the reader, it will send metadata from the library to the reader for all books on the reader that are in the library (On device is True), adding and removing books from collections as indicated by the metadata and device customization. When a book is sent, calibre corrects the metadata for that book, adding and deleting collections. Manual editing of metadata on the device view is not allowed. Note that this option specifies sending metadata, not books. The book files on the reader are not changed.

In summary, choose 'manual management' if you want to manage collections yourself. Collections for a book will never be removed by calibre, but can be removed by you by editing on the device view. Choose 'Only on send' if you want calibre to manage collections when you send a book, adding books to and removing books from collections as needed. Choose 'Automatic management' if you want calibre to keep collections up to date whenever the reader is connected.

If you use multiple installations of calibre to manage your reader, then option 'Automatic management' may not be what you want. Connecting the reader to one library will reset the metadata to what is in that library. Connecting to the other library will reset the metadata to what is in that other library. Metadata in books found in both libraries will be flopped back and forth.

Can I use both calibre and the SONY software to manage my reader?

Yes, you can use both, provided you do not run them at the same time. That is, you should use the following sequence: Connect reader->Use one of the programs->Disconnect reader. Reconnect reader->Use the other program->disconnect reader.

The underlying reason is that the Reader uses a single file to keep track of 'meta' information, such as collections, and this is written to by both calibre and the Sony software when either updates something on the Reader. The file will be saved when the Reader is (safely) disconnected, so using one or the other is safe if there's a disconnection between them, but if you're not the type to remember this, then the simple answer is to stick to one or the other for the transfer and just export/import from/to the other via the computers hard disk.

If you do need to reset your metadata due to problems caused by using both at the same time, then just delete the media.xml file on the Reader using your PC's file explorer and it will be recreated after disconnection.

With recent reader iterations, SONY, in all its wisdom has decided to try to force you to use their software. If you install it, it auto-launches whenever you connect the reader. If you don't want to uninstall it altogether, there are a couple of tricks you can use. The simplest is to simply re-name the executable file that launches the library program. More detail in the forums[33].

How do I use calibre with my iPad/iPhone/iPod touch?

Over the air

The easiest way to browse your calibre collection on your Apple device (iPad/iPhone/iPod) is by using the calibre content server, which makes your collection available over the net. First perform the following steps in calibre

- Set the Preferred Output Format in calibre to EPUB (The output format can be set under *Preferences->Interface->Behavior*)

[33] http://www.mobileread.com/forums/showthread.php?t=65809

- Set the output profile to iPad (this will work for iPhone/iPods as well), under *Preferences->Conversion->Common Options->Page Setup*
- Convert the books you want to read on your iDevice to EPUB format by selecting them and clicking the Convert button.
- Turn on the Content Server by clicking the *Connect/Share* button and leave calibre running. You can also tell calibre to automatically start the content server via *Preferences->Sharing over the net*.

There are many apps for your iDevice that can connect to calibre. Here we describe using two of them, iBooks and Stanza.

Using Stanza You should be able to access your books on your iPhone by opening Stanza. Go to "Get Books" and then click the "Shared" tab. Under Shared you will see an entry "Books in calibre". If you don't, make sure your iPad/iPhone is connected using the WiFi network in your house, not 3G. If the calibre catalog is still not detected in Stanza, you can add it manually in Stanza. To do this, click the "Shared" tab, then click the "Edit" button and then click "Add book source" to add a new book source. In the Add Book Source screen enter whatever name you like and in the URL field, enter the following:

```
http://192.168.1.2:8080/
```

Replace `192.168.1.2` with the local IP address of the computer running calibre. If you have changed the port the calibre content server is running on, you will have to change `8080` as well to the new port. The local IP address is the IP address you computer is assigned on your home network. A quick Google search will tell you how to find out your local IP address. Now click "Save" and you are done.

If you get timeout errors while browsing the calibre catalog in Stanza, try increasing the connection timeout value in the stanza settings. Go to Info->Settings and increase the value of Download Timeout.

Using iBooks Start the Safari browser and type in the IP address and port of the computer running the calibre server, like this:

```
http://192.168.1.2:8080/
```

Replace `192.168.1.2` with the local IP address of the computer running calibre. If you have changed the port the calibre content server is running on, you will have to change `8080` as well to the new port. The local IP address is the IP address you computer is assigned on your home network. A quick Google search will tell you how to find out your local IP address.

You will see a list of books in Safari, just click on the epub link for whichever book you want to read, Safari will then prompt you to open it with iBooks.

With the USB cable + iTunes

Use the 'Connect to iTunes' method in the 'Getting started' instructions in Calibre + Apple iDevices: Start here[34].

This method only works on Windows XP and higher, and OS X 10.5 and higher. Linux is not supported (iTunes is not available in linux) and OS X 10.4 is not supported.

How do I use calibre with my Android phone/tablet or Kindle Fire HD?

There are two ways that you can connect your Android device to calibre. Using a USB cable – or wirelessly, over the air. The first step to using an Android device is installing an ebook reading application on it. There are many free

[34] http://www.mobileread.com/forums/showthread.php?t=118559

and paid ebook reading applications for Android: Some examples (in no particular order): FBReader[35], Moon+[36], Mantano[37], Aldiko[38], Kindle[39].

Using a USB cable

Simply plug your device into the computer with a USB cable. calibre should automatically detect the device and then you can transfer books to it by clicking the Send to Device button. calibre does not have support for every single android device out there, so if your device is not automatically detected, follow the instructions at *How can I help get my device supported in calibre?* (page 324) to get your device supported in calibre.

Note: With newer Android devices, the USB connection is only supported on Windows Vista and newer and Linux. If you are on Windows XP or OS X, you should use one of the wireless connection methods.

Over the air

The easiest way to transfer books wirelessly to your Android device is to use the Calibre Companion[40] Android app. This app is maintained by a core calibre developer and allows calibre to connect to your Android device wirelessly, just as though you plugged in the device with a USB cable. You can browse files on the device in calibre and use the *Send to device* button to transfer files to your device wirelessly.

calibre also has a builtin web server, the *Content Server*. You can browse your calibre collection on your Android device is by using the calibre content server, which makes your collection available over the net. First perform the following steps in calibre

- Set the *Preferred Output Format* in calibre to EPUB for normal Android devices or MOBI for Kindles (The output format can be set under *Preferences->Interface->Behavior*)
- Convert the books you want to read on your device to EPUB/MOBI format by selecting them and clicking the Convert button.
- Turn on the Content Server in calibre's preferences and leave calibre running.

Now on your Android device, open the browser and browse to

http://192.168.1.2:8080/

Replace `192.168.1.2` with the local IP address of the computer running calibre. If your local network supports the use of computer names, you can replace the IP address with the network name of the computer. If you have changed the port the calibre content server is running on, you will have to change `8080` as well to the new port.

The local IP address is the IP address you computer is assigned on your home network. A quick Google search will tell you how to find out your local IP address. You can now browse your book collection and download books from calibre to your device to open with whatever ebook reading software you have on your android device.

Some reading programs support browsing the Calibre library directly. For example, in Aldiko, click My Catalogs, then + to add a catalog, then give the catalog a title such as "Calibre" and provide the URL listed above. You can now browse the Calibre library and download directly into the reading software.

[35] https://play.google.com/store/apps/details?id=org.geometerplus.zlibrary.ui.android&hl=en
[36] https://play.google.com/store/apps/details?id=com.flyersoft.moonreader&hl=en
[37] https://play.google.com/store/apps/details?id=com.mantano.reader.android.lite&hl=en
[38] https://play.google.com/store/apps/details?id=com.aldiko.android&hl=en
[39] https://play.google.com/store/apps/details?id=com.amazon.kindle&feature=related_apps
[40] http://www.multipie.co.uk/calibre-companion/

Can I access my calibre books using the web browser in my Kindle or other reading device?

calibre has a *Content Server* that exports the books in calibre as a web page. You can turn it on under *Preferences->Network->Sharing over the net*. Then just point the web browser on your device to the computer running the Content Server and you will be able to browse your book collection. For example, if the computer running the server has IP address 63.45.128.5, in the browser, you would type:

```
http://63.45.128.5:8080
```

Some devices, like the Kindle (1/2/DX), do not allow you to access port 8080 (the default port on which the content server runs. In that case, change the port in the calibre Preferences to 80. (On some operating systems, you may not be able to run the server on a port number less than 1024 because of security settings. In this case the simplest solution is to adjust your router to forward requests on port 80 to port 8080).

I get the error message "Failed to start content server: Port 8080 not free on '0.0.0.0'"?

The most likely cause of this is your antivirus program. Try temporarily disabling it and see if it does the trick.

I cannot send emails using calibre?

Because of the large amount of spam in email, sending email can be tricky, as different mail servers use different strategies to block email. The most common problem is if you are sending email directly (without a mail relay) in calibre. Many servers (for example, Amazon) block email that does not come from a well known relay. The most robust way to setup email sending in calibre is to do the following:

- Create a free GMail account at Google[41].
- Goto Preferences->Email in calibre and click the "Use Gmail" button and fill in the information asked for.
- calibre will then use GMail to send the mail.
- If you are sending to your Kindle, remember to update the email preferences on your Amazon Kindle page to allow email sent from your GMail email address.

Even after doing this, you may have problems. One common source of problems is that some poorly designed antivirus programs block calibre from opening a connection to send email. Try adding an exclusion for calibre in your antivirus program.

Note: Google can disable your account if you use it to send large amounts of email. So, when using GMail to send mail calibre automatically restricts itself to sending one book every five minutes. If you don't mind risking your account being blocked you can reduce this wait interval by going to Preferences->Tweaks in calibre.

Why is my device not detected in linux?

calibre needs your linux kernel to have been setup correctly to detect devices. If your devices are not detected, perform the following tests:

```
grep SYSFS_DEPRECATED /boot/config-`uname -r`
```

You should see something like `CONFIG_SYSFS_DEPRECATED_V2 is not set`. Also,

```
grep CONFIG_SCSI_MULTI_LUN /boot/config-`uname -r`
```

[41] http://www.gmail.com

must return `CONFIG_SCSI_MULTI_LUN=y`. If you don't see either, you have to recompile your kernel with the correct settings.

My device is getting mounted read-only in linux, so calibre cannot connect to it?

Linux kernels mount devices read-only when their filesystems have errors. You can repair the filesystem with:

```
sudo fsck.vfat -y /dev/sdc
```

Replace /dev/sdc with the path to the device node of your device. You can find the device node of your device, which will always be under /dev by examining the output of:

```
mount
```

Why does calibre not support collections on the Kindle or shelves on the Nook?

Neither the Kindle nor the Nook provide any way to manipulate collections over a USB connection. If you really care about using collections, I would urge you to sell your Kindle/Nook and get a SONY. Only SONY seems to understand that life is too short to be entering collections one by one on an e-ink screen :)

Note that in the case of the Kindle, there is a way to manipulate collections via USB, but it requires that the Kindle be rebooted *every time* it is disconnected from the computer, for the changes to the collections to be recognized. As such, it is unlikely that any calibre developers will ever feel motivated enough to support it. There is however, a calibre plugin that allows you to create collections on your Kindle from the calibre metadata. It is available from here[42].

Note: Amazon have removed the ability to manipulate collections completely in their newer models, like the Kindle Touch and Kindle Fire, making even the above plugin useless. If you really want the ability to manage collections on your Kindle via a USB connection, we encourage you to complain to Amazon about it, or get a reader where this is supported, like the SONY or Kobo Readers.

I am getting an error when I try to use calibre with my Kobo Touch/Glo/etc.?

The Kobo has very buggy firmware. Connecting to it has been known to fail at random. Certain combinations of motherboard, USB ports/cables/hubs can exacerbate this tendency to fail. If you are getting an error when connecting to your touch with calibre try the following, each of which has solved the problem for *some* calibre users.

- Connect the Kobo directly to your computer, not via USB Hub
- Try a different USB cable and a different USB port on your computer
- Try a different computer (preferably an older model)
- Try upgrading the firmware on your Kobo Touch to the latest
- Try resetting the Kobo (sometimes this cures the problem for a little while, but then it re-appears, in which case you have to reset again and again)
- Try only putting one or two books onto the Kobo at a time and do not keep large collections on the Kobo

[42] http://www.mobileread.com/forums/showthread.php?t=118635

1.6.3 Library Management

Contents

- What formats does calibre read metadata from? (page 331)
- Where are the book files stored? (page 331)
- How does calibre manage author names and sorting? (page 331)
- Why doesn't calibre let me store books in my own directory structure? (page 332)
- Why doesn't calibre have a column for foo? (page 333)
- Can I have a column showing the formats or the ISBN? (page 333)
- How do I move my calibre library from one computer to another? (page 333)
- The list of books in calibre is blank! (page 334)
- I am getting errors with my calibre library on a networked drive/NAS? (page 334)

What formats does calibre read metadata from?

calibre reads metadata from the following formats: CHM, LRF, PDF, LIT, RTF, OPF, MOBI, PRC, EPUB, FB2, IMP, RB, HTML. In addition it can write metadata to: LRF, RTF, OPF, EPUB, PDF, MOBI

Where are the book files stored?

When you first run calibre, it will ask you for a folder in which to store your books. Whenever you add a book to calibre, it will copy the book into that folder. Books in the folder are nicely arranged into sub-folders by Author and Title. Note that the contents of this folder are automatically managed by calibre, **do not** add any files/folders manually to this folder, as they may be automatically deleted. If you want to add a file associated to a particular book, use the top right area of *Edit metadata* dialog to do so. Then, calibre will automatically put that file into the correct folder and move it around when the title/author changes.

Metadata about the books is stored in the file `metadata.db` at the top level of the library folder This file is is a sqlite database. When backing up your library make sure you copy the entire folder and all its sub-folders.

The library folder and all it's contents make up what is called a calibre library. You can have multiple such libraries. To manage the libraries, click the calibre icon on the toolbar. You can create new libraries, remove/rename existing ones and switch between libraries easily.

You can copy or move books between different libraries (once you have more than one library setup) by right clicking on a book and selecting the *Copy to library* action.

How does calibre manage author names and sorting?

Author names are complex, especially across cultures. calibre has a very flexible strategy for managing author names. The first thing to understand is that books and authors are separate entities in calibre. A book can have more than one author, and an author can have more than one book. You can manage the authors of a book by the edit metadata dialog. You can manage individual authors by right clicking on the author in the Tag Browser on the left of the main calibre screen and selecting *Manage authors*. Using this dialog you can change the name of an author and also how that name is sorted. This will automatically change the name of the author in all the books of that author. When a book has multiple authors, separate their names using the & character.

Now coming to author name sorting:

- When a new author is added to calibre (this happens whenever a book by a new author is added), calibre automatically computes a sort string for both the book and the author.

- Authors in the Tag Browser are sorted by the sort value for the **authors**. Remember that this is different from the Author sort field for a book.
- By default, this sort algorithm assumes that the author name is in `First name Last name` format and generates a `Last name, First name` sort value.
- You can change this algorithm by going to Preferences->Tweaks and setting the *author_sort_copy_method* tweak.
- You can force calibre to recalculate the author sort values for every author by right clicking on any author and selecting *Manage authors*, then pushing the *Recalculate all author sort values* button. Do this after you have set the author_sort_copy_method tweak to what you want.
- You can force calibre to recalculate the author sort values for all books by using the bulk metadata edit dialog (select all books and click edit metadata, check the *Automatically set author sort* checkbox, then press OK.)
- When recalculating the author sort values for books, calibre uses the author sort values for each individual author. Therefore, ensure that the individual author sort values are correct before recalculating the books' author sort values.
- You can control whether the Tag Browser display authors using their names or their sort values by setting the *categories_use_field_for_author_name* tweak in Preferences->Tweaks

Note that you can set an individual author's sort value to whatever you want using *Manage authors*. This is useful when dealing with names that calibre will not get right, such as complex multi-part names like Miguel de Cervantes Saavedra or when dealing with Asian names like Sun Tzu.

With all this flexibility, it is possible to have calibre manage your author names however you like. For example, one common req

- Set the `author_sort_copy_method` tweak to `copy` as described above.
- Restart calibre. Do not change any book metadata before doing the remaining steps.
- Change all author names to LN, FN using the Manage authors dialog.
- After you have changed all the authors, press the *Recalculate all author sort values* button.
- Press OK, at which point calibre will change the authors in all your books. This can take a while.

Note:

When changing from FN LN to LN, FN, it is often the case that the values in author_sort are already in LN, FN format. If this i

- set the `author_sort_copy_method` tweak to `copy` as described above.
- restart calibre. Do not change any book metadata before doing the remaining steps.
- open the Manage authors dialog. Press the `copy all author sort values to author` button.
- Check through the authors to be sure you are happy. You can still press Cancel to abandon the changes. Once you press OK, there is no undo.
- Press OK, at which point calibre will change the authors in all your books. This can take a while.

Why doesn't calibre let me store books in my own directory structure?

The whole point of calibre's library management features is that they provide a search and sort based interface for locating books that is *much* more efficient than any possible directory scheme you could come up with for your collection. Indeed, once you become comfortable using calibre's interface to find, sort and browse your collection, you wont ever feel the need to hunt through the files on your disk to find a book again. By managing books in

its own directory structure of Author -> Title -> Book files, calibre is able to achieve a high level of reliability and standardization. To illustrate why a search/tagging based interface is superior to folders, consider the following. Suppose your book collection is nicely sorted into folders with the following scheme:

```
Genre -> Author -> Series -> ReadStatus
```

Now this makes it very easy to find for example all science fiction books by Isaac Asimov in the Foundation series. But suppose you want to find all unread science fiction books. There's no easy way to do this with this folder scheme, you would instead need a folder scheme that looks like:

```
ReadStatus -> Genre -> Author -> Series
```

In calibre, you would instead use tags to mark genre and read status and then just use a simple search query like `tag:scifi and not tag:read`. calibre even has a nice graphical interface, so you don't need to learn its search language instead you can just click on tags to include or exclude them from the search.

To those of you that claim that you need access to the filesystem to so that you can have access to your books over the network, calibre has an excellent content server that gives you access to your calibre library over the net.

If you are worried that someday calibre will cease to be developed, leaving all your books marooned in its folder structure, explore the powerful "Save to Disk" feature in calibre that lets you export all your files into a folder structure of arbitrary complexity based on their metadata.

Finally, the reason there are numbers at the end of every title folder, is for *robustness*. That number is the id number of the book record in the calibre database. The presence of the number allows you to have multiple records with the same title and author names. It is also part of what allows calibre to magically regenerate the database with all metadata if the database file gets corrupted. Given that calibre's mission is to get you to stop storing metadata in filenames and stop using the filesystem to find things, the increased robustness afforded by the id numbers is well worth the uglier folder names.

If you are still not convinced, then I'm afraid calibre is not for you. Look elsewhere for your book cataloguing needs. Just so we're clear, **this is not going to change**. Kindly do not contact us in an attempt to get us to change this.

Why doesn't calibre have a column for foo?

calibre is designed to have columns for the most frequently and widely used fields. In addition, you can add any columns you like. Columns can be added via *Preferences->Interface->Add your own columns*. Watch the tutorial UI Power tips[43] to learn how to create your own columns.

You can also create "virtual columns" that contain combinations of the metadata from other columns. In the add column dialog use the *Quick create* links to easily create columns to show the book ISBN, formats or the time the book was last modified. For more details, see *The calibre template language* (page 375).

Can I have a column showing the formats or the ISBN?

Yes, you can. Follow the instructions in the answer above for adding custom columns.

How do I move my calibre library from one computer to another?

Simply copy the calibre library folder from the old to the new computer. You can find out what the library folder is by clicking the calibre icon in the toolbar. The very first item is the path to the library folder. Now on the new computer, start calibre for the first time. It will run the Welcome Wizard asking you for the location of the calibre library. Point it to the previously copied folder. If the computer you are transferring to already has a calibre installation, then the Welcome wizard wont run. In that case, right-click the calibre icon in the tooolbar and point it to the newly copied

[43] http://calibre-ebook.com/demo#tutorials

directory. You will now have two calibre libraries on your computer and you can switch between them by clicking the calibre icon on the toolbar. Transferring your library in this manner preserver all your metadata, tags, custom columns, etc.

Note that if you are transferring between different types of computers (for example Windows to OS X) then after doing the above you should also right-click the calibre icon on the tool bar, select Library Maintenance and run the Check Library action. It will warn you about any problems in your library, which you should fix by hand.

Note: A calibre library is just a folder which contains all the book files and their metadata. All the metadata is stored in a single file called metadata.db, in the top level folder. If this file gets corrupted, you may see an empty list of books in calibre. In this case you can ask calibre to restore your books by doing a right-click on the calibre icon in the toolbar and selecting Library Maintenance->Restore Library.

The list of books in calibre is blank!

In order to understand why that happened, you have to understand what a calibre library is. At the most basic level, a calibre library is just a folder. Whenever you add a book to calibre, that book's files are copied into this folder (arranged into sub folders by author and title). Inside the calibre library folder, at the top level, you will see a file called metadata.db. This file is where calibre stores the metadata like title/author/rating/tags etc. for *every* book in your calibre library. The list of books that calibre displays is created by reading the contents of this metadata.db file.

There can be two reasons why calibre is showing a empty list of books:

- Your calibre library folder changed its location. This can happen if it was on an external disk and the drive letter for that disk changed. Or if you accidentally moved the folder. In this case, calibre cannot find its library and so starts up with an empty library instead. To remedy this, do a right-click on the calibre icon in the calibre toolbar (it will say 0 books underneath it) and select Switch/create library. Click the little blue icon to select the new location of your calibre library and click OK.
- Your metadata.db file was deleted/corrupted. In this case, you can ask calibre to rebuild the metadata.db from its backups. Right click the calibre icon in the calibre toolbar (it will say 0 books underneath it) and select Library maintenance->Restore database. calibre will automatically rebuild metadata.db.

I am getting errors with my calibre library on a networked drive/NAS?

Do not put your calibre library on a networked drive.

A filesystem is a complex beast. Most network filesystems lack various filesystem features that calibre uses. Some dont support file locking, some dont support hardlinking, some are just flaky. Additionally, calibre is a single user application, if you accidentally run two copies of calibre on the same networked library, bad things will happen. Finally, different OSes impose different limitations on filesystems, so if you share your networked drive across OSes, once again, bad things *will happen*.

Consider using the calibre Content Server to make your books available on other computers. Run calibre on a single computer and access it via the Content Server or a Remote Desktop solution.

If you must share the actual library, use a file syncing tool like DropBox or rsync or Microsoft SkyDrive instead of a networked drive. Even with these tools there is danger of data corruption/loss, so only do this if you are willing to live with that risk. In particular, be aware that **Google Drive** is incompatible with calibre, if you put your calibre library in Google Drive, you *will* suffer data loss. See this thread[44] for details.

[44] http://www.mobileread.com/forums/showthread.php?t=205581

1.6.4 Content From The Web

Contents

- I obtained a recipe for a news site as a .py file from somewhere, how do I use it? (page 335)
- I want calibre to download news from my favorite news website. (page 335)
- Can I use web2disk to download an arbitrary website? (page 335)

I obtained a recipe for a news site as a .py file from somewhere, how do I use it?

Start the *Add custom news sources* dialog (from the *Fetch news* menu) and click the *Switch to advanced mode* button. Delete everything in the box with the recipe source code and copy paste the contents of your .py file into the box. Click *Add/update recipe*.

I want calibre to download news from my favorite news website.

If you are reasonably proficient with computers, you can teach calibre to download news from any website of your choosing. To learn how to do this see *Adding your favorite news website* (page 341).

Otherwise, you can request a particular news site by posting in the calibre Recipes forum[45].

Can I use web2disk to download an arbitrary website?

```
web2disk http://mywebsite.com
```

1.6.5 Miscellaneous

Contents

- Why the name calibre? (page 336)
- Why does calibre show only some of my fonts on OS X? (page 336)
- calibre is not starting on Windows? (page 336)
- calibre freezes/crashes occasionally? (page 337)
- calibre is not starting on OS X? (page 337)
- I downloaded the installer, but it is not working? (page 338)
- My antivirus program claims calibre is a virus/trojan? (page 338)
- How do I backup calibre? (page 338)
- How do I use purchased EPUB books with calibre (or what do I do with .acsm files)? (page 338)
- I am getting a "Permission Denied" error? (page 339)
- Can I have the comment metadata show up on my reader? (page 339)
- How do I get calibre to use my HTTP proxy? (page 339)
- I want some feature added to calibre. What can I do? (page 339)
- Why doesn't calibre have an automatic update? (page 340)
- How is calibre licensed? (page 340)
- How do I run calibre from my USB stick? (page 340)
- How do I run parts of calibre like news download and the content server on my own linux server? (page 340)

[45] http://www.mobileread.com/forums/forumdisplay.php?f=228

Why the name calibre?

Take your pick:

- Convertor And LIBRary for Ebooks
- A high *calibre* product
- A tribute to the SONY Librie which was the first e-ink based ebook reader
- My wife chose it ;-)

calibre is pronounced as cal-i-ber *not* ca-li-bre. If you're wondering, calibre is the British/commonwealth spelling for caliber. Being Indian, that's the natural spelling for me.

Why does calibre show only some of my fonts on OS X?

calibre embeds fonts in ebook files it creates. Ebook files support embedding only TrueType (.ttf) fonts. Most fonts on OS X systems are in .dfont format, thus they cannot be embedded. calibre shows only TrueType fonts found on your system. You can obtain many TrueType fonts on the web. Simply download the .ttf files and add them to the Library/Fonts directory in your home directory.

calibre is not starting on Windows?

There can be several causes for this:

- If you get an error about calibre not being able to open a file because it is in use by another program, do the following:
  - Uninstall calibre
  - Reboot your computer
  - Re-install calibre. But do not start calibre from the installation wizard.
  - Temporarily disable your antivirus program (disconnect from the Internet before doing so, to be safe)
  - Look inside the folder you chose for your calibre library. If you see a file named metadata.db, delete it.
  - Start calibre
  - From now on you should be able to start calibre normally.
- If you get an error about a Python function terminating unexpectedly after upgrading calibre, first uninstall calibre, then delete the folders (if they exists) `C:\Program Files\Calibre` and `C:\Program Files\Calibre2`. Now re-install and you should be fine.
- If you get an error in the welcome wizard on an initial run of calibre, try choosing a folder like `C:\library` as the calibre library (calibre sometimes has trouble with library locations if the path contains non-English characters, or only numbers, etc.)
- Try running it as Administrator (Right click on the icon and select "Run as Administrator")
- **Windows Vista**: If the folder `C:\Users\Your User Name\AppData\Local\VirtualStore\Program Files\calibre` exists, delete it. Uninstall calibre. Reboot. Re-install.
- **Any windows version**: Try disabling any antivirus program you have running and see if that fixes it. Also try disabling any firewall software that prevents connections to the local computer.

If it still wont launch, start a command prompt (press the windows key and R; then type **cmd.exe** in the Run dialog that appears). At the command prompt type the following command and press Enter:

```
calibre-debug -g
```

Post any output you see in a help message on the Forum[46].

calibre freezes/crashes occasionally?

There are several possible things I know of, that can cause this:

- You recently connected an external monitor or TV to your computer. In this case, whenever calibre opens a new window like the edit metadata window or the conversion dialog, it appears on the second monitor where you dont notice it and so you think calibre has frozen. Disconnect your second monitor and restart calibre.
- If you use RoboForm, it is known to cause calibre to crash. Add calibre to the blacklist of programs inside RoboForm to fix this. Or uninstall RoboForm.
- The Logitech SetPoint Settings application causes random crashes in calibre when it is open. Close it before starting calibre.
- Constant Guard Protection by Xfinity causes crashes in calibre. You have to manually allow calibre in it or uninstall Constant Guard Protection.
- Spybot - Search & Destroy blocks calibre from accessing its temporary files breaking viewing and converting of books.
- You are using a Wacom branded USB mouse. There is an incompatibility between Wacom mice and the graphics toolkit calibre uses. Try using a non-Wacom mouse.
- On some 64 bit versions of Windows there are security software/settings that prevent 64-bit calibre from working properly. If you are using the 64-bit version of calibre try switching to the 32-bit version.

If none of the above apply to you, then there is some other program on your computer that is interfering with calibre. First reboot your computer in safe mode, to have as few running programs as possible, and see if the crashes still happen. If they do not, then you know it is some program causing the problem. The most likely such culprit is a program that modifies other programs' behavior, such as an antivirus, a device driver, something like RoboForm (an automatic form filling app) or an assistive technology like Voice Control or a Screen Reader.

The only way to find the culprit is to eliminate the programs one by one and see which one is causing the issue. Basically, stop a program, run calibre, check for crashes. If they still happen, stop another program and repeat.

calibre is not starting on OS X?

One common cause of failures on OS X is the use of accessibility technologies that are incompatible with the graphics toolkit calibre uses. Try turning off VoiceOver if you have it on. Also go to System Preferences->System->Universal Access and turn off the setting for enabling access for assistive devices in all the tabs.

You can obtain debug output about why calibre is not starting by running *Console.app*. Debug output will be printed to it. If the debug output contains a line that looks like:

```
Qt: internal: -108: Error ATSUMeasureTextImage text/qfontengine_mac.mm
```

then the problem is probably a corrupted font cache. You can clear the cache by following these instructions[47]. If that doesn't solve it, look for a corrupted font file on your system, in ~/Library/Fonts or the like. An easy way to check for corrupted fonts in OS X is to start the "Font Book" application, select all fonts and then in the File menu, choose "Validate fonts".

[46] http://www.mobileread.com/forums/forumdisplay.php?f=166
[47] http://www.macworld.com/article/139383/2009/03/fontcacheclear.html

I downloaded the installer, but it is not working?

Downloading from the Internet can sometimes result in a corrupted download. If the calibre installer you downloaded is not opening, try downloading it again. If re-downloading it does not work, download it from an alternate location[48]. If the installer still doesn't work, then something on your computer is preventing it from running.

- Try temporarily disabling your antivirus program (Microsoft Security Essentials, or Kaspersky or Norton or McAfee or whatever). This is most likely the culprit if the upgrade process is hanging in the middle.
- Try rebooting your computer and running a registry cleaner like Wise registry cleaner[49].
- Try downloading the installer with an alternate browser. For example if you are using Internet Explorer, try using Firefox or Chrome instead.

If you still cannot get the installer to work and you are on windows, you can use the calibre portable install[50], which does not need an installer (it is just a zip file).

My antivirus program claims calibre is a virus/trojan?

The first thing to check is that you are downloading calibre from the official website: http://calibre-ebook.com/download. calibre is a very popular program and unscrupulous people try to setup websites offering it for download to fool the unwary.

If you have the official download and your antivirus program is still claiming calibre is a virus, then, your antivirus program is wrong. Antivirus programs use heuristics, patterns of code that "look suspicious" to detect viruses. It's rather like racial profiling. calibre is a completely open source product. You can actually browse the source code yourself (or hire someone to do it for you) to verify that it is not a virus. Please report the false identification to whatever company you buy your antivirus software from. If the antivirus program is preventing you from downloading/installing calibre, disable it temporarily, install calibre and then re-enable it.

How do I backup calibre?

The most important thing to backup is the calibre library folder, that contains all your books and metadata. This is the folder you chose for your calibre library when you ran calibre for the first time. You can get the path to the library folder by clicking the calibre icon on the main toolbar. You must backup this complete folder with all its files and sub-folders.

You can switch calibre to using a backed up library folder by simply clicking the calibre icon on the toolbar and choosing your backup library folder. A backed up library folder backs up your custom columns and saved searches as well as all your books and metadata.

If you want to backup the calibre configuration/plugins, you have to backup the config directory. You can find this config directory via *Preferences->Miscellaneous*. Note that restoring configuration directories is not officially supported, but should work in most cases. Just copy the contents of the backup directory into the current configuration directory to restore.

How do I use purchased EPUB books with calibre (or what do I do with .acsm files)?

Most purchased EPUB books have DRM[51]. This prevents calibre from opening them. You can still use calibre to store and transfer them to your ebook reader. First, you must authorize your reader on a windows machine with Adobe Digital Editions. Once this is done, EPUB books transferred with calibre will work fine on your reader. When you

[48] http://sourceforge.net/projects/calibre/files/
[49] http://www.wisecleaner.com
[50] http://calibre-ebook.com/download_portable
[51] http://drmfree.calibre-ebook.com/about#drm

purchase an epub book from a website, you will get an ".acsm" file. This file should be opened with Adobe Digital Editions, which will then download the actual ".epub" ebook. The ebook file will be stored in the folder "My Digital Editions", from where you can add it to calibre.

I am getting a "Permission Denied" error?

A permission denied error can occur because of many possible reasons, none of them having anything to do with calibre.

- You can get permission denied errors if you are using an SD card with write protect enabled.
- If you, or some program you used changed the file permissions of the files in question to read only.
- If there is a filesystem error on the device which caused your operating system to mount the filesystem in read only mode or mark a particular file as read only pending recovery.
- If the files have their owner set to a user other than you.
- If your file is open in another program.
- If the file resides on a device, you may have reached the limit of a maximum of 256 files in the root of the device. In this case you need to reformat the device/sd card referered to in the error message with a FAT32 filesystem, or delete some files from the SD card/device memory.

You will need to fix the underlying cause of the permissions error before resuming to use calibre. Read the error message carefully, see what file it points to and fix the permissions on that file.

Can I have the comment metadata show up on my reader?

Most readers do not support this. You should complain to the manufacturer about it and hopefully if enough people complain, things will change. In the meantime, you can insert the metadata, including comments into a "Jacket page" at the start of the ebook, by using the option to "Insert metadata as page at start of book" during conversion. The option is found in the *Structure Detection* section of the conversion settings. Note that for this to have effect you have to *convert* the book. If your book is already in a format that does not need conversion, you can convert from that format to the same format.

Another alternative is to create a catalog in ebook form containing a listing of all the books in your calibre library, with their metadata. Click-and-hold the convert button to access the catalog creation tool. And before you ask, no you cannot have the catalog "link directly to" books on your reader.

How do I get calibre to use my HTTP proxy?

By default, calibre uses whatever proxy settings are set in your OS. Sometimes these are incorrect, for example, on windows if you don't use Internet Explorer then the proxy settings may not be up to date. You can tell calibre to use a particular proxy server by setting the http_proxy environment variable. The format of the variable is: http://username:password@servername you should ask your network admin to give you the correct value for this variable. Note that calibre only supports HTTP proxies not SOCKS proxies. You can see the current proxies used by calibre in Preferences->Miscellaneous.

I want some feature added to calibre. What can I do?

You have two choices:

1. Create a patch by hacking on calibre and send it to me for review and inclusion. See Development[52].

[52] http://calibre-ebook.com/get-involved

2. Open a bug requesting the feature[53] . Remember that while you may think your feature request is extremely important/essential, calibre developers might not agree. Fortunately, calibre is open source, which means you always have the option of implementing your feature yourself, or hiring someone to do it for you. Furthermore, calibre has a comprehensive plugin architecture, so you might be able to develop your feature as a plugin, see *Writing your own plugins to extend calibre's functionality* (page 423).

Why doesn't calibre have an automatic update?

For many reasons:

- *There is no need to update every week*. If you are happy with how calibre works turn off the update notification and be on your merry way. Check back to see if you want to update once a year or so.
- Pre downloading the updates for all users in the background would require about 80TB of bandwidth *every week*. That costs thousands of dollars a month. And calibre is currently growing at 300,000 new users every month.
- If I implement a dialog that downloads the update and launches it, instead of going to the website as it does now, that would save the most ardent calibre updater, *at most five clicks a week*. There are far higher priority things to do in calibre development.
- If you really, really hate downloading calibre every week but still want to be up to the latest, I encourage you to run from source, which makes updating trivial. Instructions are *available here* (page 511).

How is calibre licensed?

calibre is licensed under the GNU General Public License v3 (an open source license). This means that you are free to redistribute calibre as long as you make the source code available. So if you want to put calibre on a CD with your product, you must also put the calibre source code on the CD. The source code is available for download from googlecode[54]. You are free to use the results of conversions from calibre however you want. You cannot use code, libraries from calibre in your software without making your software open source. For details, see The GNU GPL v3[55].

How do I run calibre from my USB stick?

A portable version of calibre is available here[56].

How do I run parts of calibre like news download and the content server on my own linux server?

First, you must install calibre onto your linux server. If your server is using a modern linux distro, you should have no problems installing calibre onto it.

Note: If you bought into the notion that a real server must run a decade old version of Debian, then you will have to jump through a few hoops. First, compile a newer version of glibc (>= 2.10) on your server from source. Then get the calibre linux binary tarball from the calibre google code page for your server architecture. Extract it into `/opt/calibre`. Put your previously compiled glibc into `/opt/calibre` as `libc.so.6`. You can now run the calibre binaries from `/opt/calibre`.

You can run the calibre server via the command:

[53] http://calibre-ebook.com/bugs
[54] http://code.google.com/p/calibre-ebook/downloads/list
[55] http://www.gnu.org/licenses/gpl.html
[56] http://calibre-ebook.com/download_portable

```
/opt/calibre/calibre-server --with-library /path/to/the/library/you/want/to/share
```

You can download news and convert it into an ebook with the command:

```
/opt/calibre/ebook-convert "Title of news source.recipe" outputfile.epub
```

If you want to generate MOBI, use outputfile.mobi instead and use `--output-profile kindle`.

You can email downloaded news with the command:

```
/opt/calibre/calibre-smtp
```

I leave figuring out the exact command line as an exercise for the reader.

Finally, you can add downloaded news to the calibre library with:

```
/opt/calibre/calibredb add --with-library /path/to/library outfile.epub
```

Remember to read the command line documentation section of the calibre User Manual to learn more about these, and other commands.

Note: Some parts of calibre require a X server. If you're lucky, nothing you do will fall into this category, if not, you will have to look into using xvfb.

1.7 Tutorials

Here you will find tutorials to get you started using calibre's more advanced features, such as XPath and templates.

1.7.1 Adding your favorite news website

calibre has a powerful, flexible and easy-to-use framework for downloading news from the Internet and converting it into an ebook. The following will show you, by means of examples, how to get news from various websites.

To gain an understanding of how to use the framework, follow the examples in the order listed below:

- Completely automatic fetching (page 342)
  - portfolio.com (page 342)
  - bbc.co.uk (page 344)
- Customizing the fetch process (page 344)
  - Using the print version of bbc.co.uk (page 344)
  - Replacing article styles (page 345)
  - Slicing and dicing (page 346)
  - Real life example (page 357)
- Tips for developing new recipes (page 359)
- Further reading (page 360)
- API documentation (page 360)

Completely automatic fetching

If your news source is simple enough, calibre may well be able to fetch it completely automatically, all you need to do is provide the URL. calibre gathers all the information needed to download a news source into a *recipe*. In order to

tell calibre about a news source, you have to create a *recipe* for it. Let's see some examples:

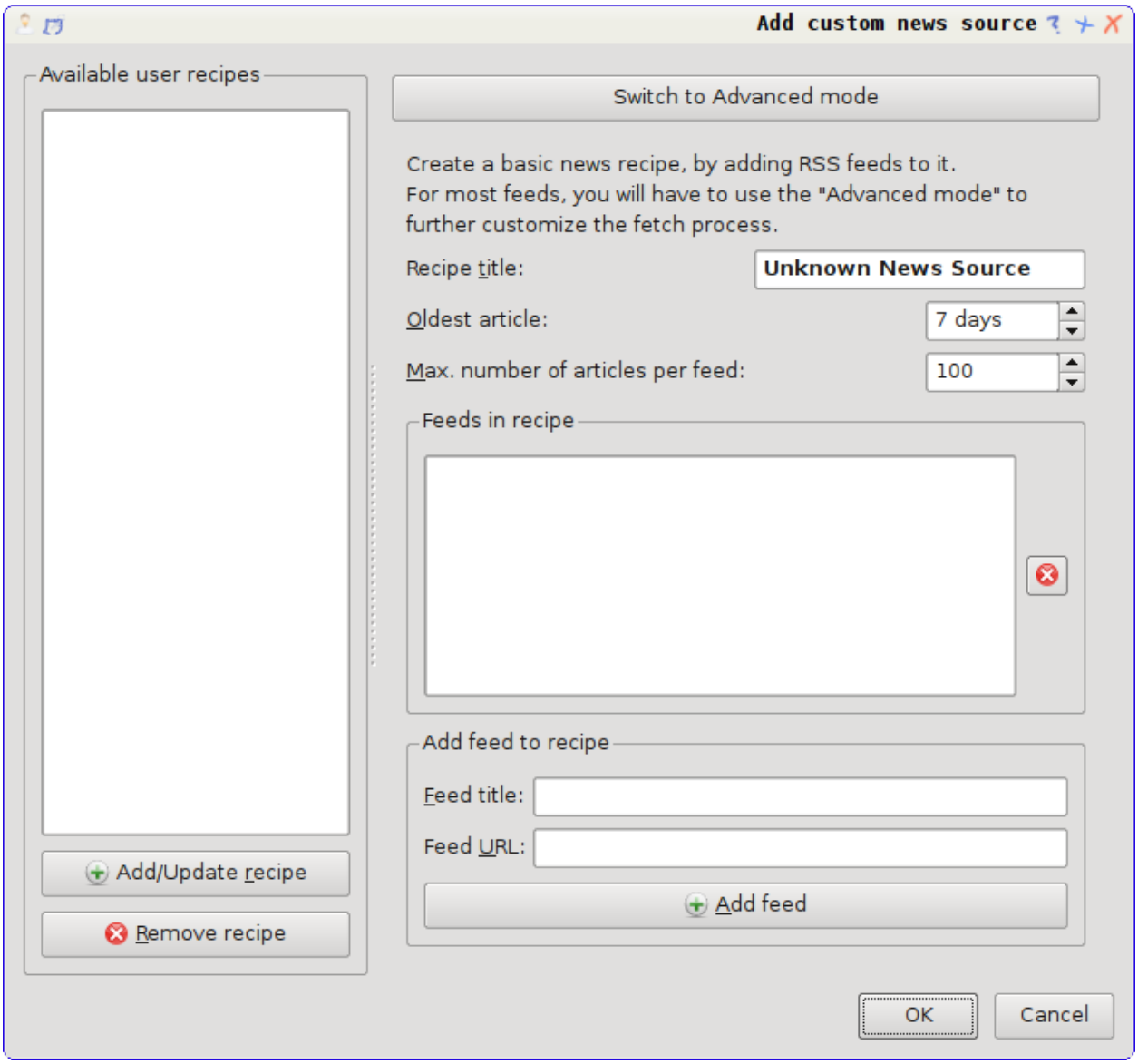
portfolio.com

portfolio.com is the website for *Condé Nast Portfolio*, a business related magazine. In order to download articles from the magazine and convert them to ebooks, we rely on the *RSS* feeds of portfolio.com. A list of such feeds is available at http://www.portfolio.com/rss/.

Lets pick a couple of feeds that look interesting:

1. Business Travel: http://feeds.portfolio.com/portfolio/businesstravel
2. Tech Observer: http://feeds.portfolio.com/portfolio/thetechobserver

I got the URLs by clicking the little orange RSS icon next to each feed name. To make calibre download the feeds and convert them into an ebook, you should right click the *Fetch news* button and then the *Add a custom news source* menu item. A dialog similar to that shown below should open up.



First enter `Portfolio` into the *Recipe title* field. This will be the title of the ebook that will be created from the articles in the above feeds.

The next two fields (*Oldest article* and *Max. number of articles*) allow you some control over how many articles should be downloaded from each feed, and they are pretty self explanatory.

To add the feeds to the recipe, enter the feed title and the feed URL and click the *Add feed* button. Once you have added both feeds, simply click the *Add/update recipe* button and you're done! Close the dialog.

To test your new *recipe*, click the *Fetch news* button and in the *Custom news sources* sub-menu click *Portfolio*. After a couple of minutes, the newly downloaded Portfolio ebook will appear in the main library view (if you have your reader connected, it will be put onto the reader instead of into the library). Select it and hit the *View* button to read!

The reason this worked so well, with so little effort is that *portfolio.com* provides *full-content RSS* feeds, i.e., the article content is embedded in the feed itself. For most news sources that provide news in this fashion, with *full-content* feeds, you don't need any more effort to convert them to ebooks. Now we will look at a news source that does not provide full content feeds. In such feeds, the full article is a webpage and the feed only contains a link to the webpage with a short summary of the article.

bbc.co.uk

Lets try the following two feeds from *The BBC*:

1. News Front Page: http://newsrss.bbc.co.uk/rss/newsonline_world_edition/front_page/rss.xml
2. Science/Nature: http://newsrss.bbc.co.uk/rss/newsonline_world_edition/science/nature/rss.xml

Follow the procedure outlined in *portfolio.com* (page 342) to create a recipe for *The BBC* (using the feeds above). Looking at the downloaded ebook, we see that calibre has done a creditable job of extracting only the content you care about from each article's webpage. However, the extraction process is not perfect. Sometimes it leaves in undesirable content like menus and navigation aids or it removes content that should have been left alone, like article headings. In order, to have perfect content extraction, we will need to customize the fetch process, as described in the next section.

Customizing the fetch process

When you want to perfect the download process, or download content from a particularly complex website, you can avail yourself of all the power and flexibility of the *recipe* framework. In order to do that, in the *Add custom news sources* dialog, simply click the *Switch to Advanced mode* button.

The easiest and often most productive customization is to use the print version of the online articles. The print version typically has much less cruft and translates much more smoothly to an ebook. Let's try to use the print version of the articles from *The BBC*.

Using the print version of bbc.co.uk

The first step is to look at the ebook we downloaded previously from *bbc.co.uk* (page 344). At the end of each article, in the ebook is a little blurb telling you where the article was downloaded from. Copy and paste that URL into a browser. Now on the article webpage look for a link that points to the "Printable version". Click it to see the print version of the article. It looks much neater! Now compare the two URLs. For me they were:

Article URL http://news.bbc.co.uk/2/hi/science/nature/7312016.stm

Print version URL http://newsvote.bbc.co.uk/mpapps/pagetools/print/news.bbc.co.uk/2/hi/science/nature/7312016.stm

So it looks like to get the print version, we need to prefix every article URL with:

newsvote.bbc.co.uk/mpapps/pagetools/print/

Now in the *Advanced Mode* of the Custom news sources dialog, you should see something like (remember to select *The BBC* recipe before switching to advanced mode):



You can see that the fields from the *Basic mode* have been translated to python code in a straightforward manner. We need to add instructions to this recipe to use the print version of the articles. All that's needed is to add the following two lines:

```
def print_version(self, url):
    return url.replace('http://', 'http://newsvote.bbc.co.uk/mpapps/pagetools/print/')
```

This is python, so indentation is important. After you've added the lines, it should look like:



In the above, `def print_version(self, url)` defines a *method* that is called by calibre for every article. `url` is the URL of the original article. What `print_version` does is take that url and replace it with the new URL that points to the print version of the article. To learn about python[57] see the tutorial[58].

Now, click the *Add/update recipe* button and your changes will be saved. Re-download the ebook. You should have a much improved ebook. One of the problems with the new version is that the fonts on the print version webpage are too small. This is automatically fixed when converting to an ebook, but even after the fixing process, the font size of the menus and navigation bar to become too large relative to the article text. To fix this, we will do some more customization, in the next section.

Replacing article styles

In the previous section, we saw that the font size for articles from the print version of *The BBC* was too small. In most websites, *The BBC* included, this font size is set by means of *CSS* stylesheets. We can disable the fetching of such stylesheets by adding the line:

[57] http://www.python.org
[58] http://docs.python.org/tut/

```
no_stylesheets = True
```

The recipe now looks like:

Recipe source code (python)

```
class AdvancedUserRecipe1206419520(BasicNewsRecipe):
    title          = u'The BBC'
    oldest_article = 7
    max_articles_per_feed = 100
    no_stylesheets = True

    feeds          = [(u'News Front Page', u'http://newsrss.bbc.co.uk/rss/news

    def print_version(self, url):
        return url.replace('http://', 'http://newsvote.bbc.co.uk/mpapps/pageto
```

The new version looks pretty good. If you're a perfectionist, you'll want to read the next section, which deals with actually modifying the downloaded content.

Slicing and dicing

calibre contains very powerful and flexible abilities when it comes to manipulating downloaded content. To show off a couple of these, let's look at our old friend the *The BBC* (page 345) recipe again. Looking at the source code (*HTML*) of a couple of articles (print version), we see that they have a footer that contains no useful information, contained in

```
<div class="footer">
...
</div>
```

This can be removed by adding:

```
remove_tags    = [dict(name='div', attrs={'class':'footer'})]
```

to the recipe. Finally, lets replace some of the *CSS* that we disabled earlier, with our own *CSS* that is suitable for conversion to an ebook:

```
extra_css      = '.headline {font-size: x-large;} \n .fact { padding-top: 10pt  }'
```

With these additions, our recipe has become "production quality", indeed it is very close to the actual recipe used by calibre for the *BBC*, shown below:

```
##
## Title:         BBC News, Sport, and Blog Calibre Recipe
## Contact:       mattst - jmstanfield@gmail.com
##
## License:       GNU General Public License v3 - http://www.gnu.org/copyleft/gpl.html
## Copyright:     mattst - jmstanfield@gmail.com
##
## Written:       November 2011
## Last Edited:   2011-11-19
##

__license__     = 'GNU General Public License v3 - http://www.gnu.org/copyleft/gpl.html'
__copyright__   = 'mattst - jmstanfield@gmail.com'
```

```
'''
BBC News, Sport, and Blog Calibre Recipe
'''

# Import the regular expressions module.
import re

# Import the BasicNewsRecipe class which this class extends.
from calibre.web.feeds.recipes import BasicNewsRecipe

class BBCNewsSportBlog(BasicNewsRecipe):

    #
    #    **** IMPORTANT USERS READ ME ****
    #
    #  First select the feeds you want then scroll down below the feeds list
    #  and select the values you want for the other user preferences, like
    #  oldest_article and such like.
    #
    #
    #  Select the BBC rss feeds which you want in your ebook.
    #  Selected feed have NO '#' at their start, de-selected feeds begin with a '#'.
    #
    #  Eg.  ("News Home", "http://feeds.bbci.co.uk/... - include feed.
    #  Eg. #("News Home", "http://feeds.bbci.co.uk/... - do not include feed.
    #
    # There are 68 feeds below which constitute the bulk of the available rss
    # feeds on the BBC web site. These include 5 blogs by editors and
    # correspondants, 16 sports feeds, 15 'sub' regional feeds (Eg. North West
    # Wales, Scotland Business), and 7 Welsh language feeds.
    #
    # Some of the feeds are low volume (Eg. blogs), or very low volume (Eg. Click)
    # so if "oldest_article = 1.5" (only articles published in the last 36 hours)
    # you may get some 'empty feeds' which will not then be included in the ebook.
    #
    # The 15 feeds currently selected below are simply my default ones.
    #
    # Note: With all 68 feeds selected, oldest_article set to 2,
    # max_articles_per_feed set to 100, and simultaneous_downloads set to 10,
    # the ebook creation took 29 minutes on my speedy 100 mbps net connection,
    # fairly high-end desktop PC running Linux (Ubuntu Lucid-Lynx).
    # More realistically with 15 feeds selected, oldest_article set to 1.5,
    # max_articles_per_feed set to 100, and simultaneous_downloads set to 20,
    # it took 6 minutes. If that's too slow increase 'simultaneous_downloads'.
    #
    # Select / de-select the feeds you want in your ebook.
    #
    feeds = [
              ("News Home", "http://feeds.bbci.co.uk/news/rss.xml"),
              ("UK", "http://feeds.bbci.co.uk/news/uk/rss.xml"),
              ("World", "http://feeds.bbci.co.uk/news/world/rss.xml"),
              #("England", "http://feeds.bbci.co.uk/news/england/rss.xml"),
              #("Scotland", "http://feeds.bbci.co.uk/news/scotland/rss.xml"),
              #("Wales", "http://feeds.bbci.co.uk/news/wales/rss.xml"),
              #("N. Ireland", "http://feeds.bbci.co.uk/news/northern_ireland/rss.xml"),
              #("Africa", "http://feeds.bbci.co.uk/news/world/africa/rss.xml"),
              #("Asia", "http://feeds.bbci.co.uk/news/world/asia/rss.xml"),
              #("Europe", "http://feeds.bbci.co.uk/news/world/europe/rss.xml"),
```

```
#("Latin America", "http://feeds.bbci.co.uk/news/world/latin_america/rss.xml"),
#("Middle East", "http://feeds.bbci.co.uk/news/world/middle_east/rss.xml"),
("US & Canada", "http://feeds.bbci.co.uk/news/world/us_and_canada/rss.xml"),
("Politics", "http://feeds.bbci.co.uk/news/politics/rss.xml"),
("Science/Environment", "http://feeds.bbci.co.uk/news/science_and_environment/rss.xml"
("Technology", "http://feeds.bbci.co.uk/news/technology/rss.xml"),
("Magazine", "http://feeds.bbci.co.uk/news/magazine/rss.xml"),
("Entertainment/Arts", "http://feeds.bbci.co.uk/news/entertainment_and_arts/rss.xml"),
#("Health", "http://feeds.bbci.co.uk/news/health/rss.xml"),
#("Education/Family", "http://feeds.bbci.co.uk/news/education/rss.xml"),
("Business", "http://feeds.bbci.co.uk/news/business/rss.xml"),
("Special Reports", "http://feeds.bbci.co.uk/news/special_reports/rss.xml"),
("Also in the News", "http://feeds.bbci.co.uk/news/also_in_the_news/rss.xml"),
#("Newsbeat", "http://www.bbc.co.uk/newsbeat/rss.xml"),
#("Click", "http://newsrss.bbc.co.uk/rss/newsonline_uk_edition/programmes/click_online/
("Blog: Nick Robinson (Political Editor)", "http://feeds.bbci.co.uk/news/correspondents
#("Blog: Mark D'Arcy (Parliamentary Correspondent)", "http://feeds.bbci.co.uk/news/cor
#("Blog: Robert Peston (Business Editor)", "http://feeds.bbci.co.uk/news/correspondents
#("Blog: Stephanie Flanders (Economics Editor)", "http://feeds.bbci.co.uk/news/corresp
("Blog: Rory Cellan-Jones (Technology correspondent)", "http://feeds.bbci.co.uk/news/c
("Sport Front Page", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/front_page/rs
#("Football", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/football/rss.xml"),
#("Cricket", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/cricket/rss.xml"),
#("Rugby Union", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/rugby_union/rss.
#("Rugby League", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/rugby_league/rss
#("Tennis", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/tennis/rss.xml"),
#("Golf", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/golf/rss.xml"),
#("Motorsport", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/motorsport/rss.xml
#("Boxing", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/boxing/rss.xml"),
#("Athletics", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/athletics/rss.xml",
#("Snooker", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/snooker/
#("Horse Racing", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/hor
#("Cycling", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/cycling/
#("Disability Sport", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports
#("Other Sport", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/rss
#("Olympics 2012", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/o
#("N. Ireland Politics", "http://feeds.bbci.co.uk/news/northern_ireland/northern_irelan
#("Scotland Politics", "http://feeds.bbci.co.uk/news/scotland/scotland_politics/rss.xml
#("Scotland Business", "http://feeds.bbci.co.uk/news/scotland/scotland_business/rss.xml
#("E. Scotland, Edinburgh & Fife", "http://feeds.bbci.co.uk/news/scotland/edinburgh_eas
#("W. Scotland & Glasgow", "http://feeds.bbci.co.uk/news/scotland/glasgow_and_west/rss
#("Highlands & Islands", "http://feeds.bbci.co.uk/news/scotland/highlands_and_islands/r
#("NE. Scotland, Orkney & Shetland", "http://feeds.bbci.co.uk/news/scotland/north_east_
#("South Scotland", "http://feeds.bbci.co.uk/news/scotland/south_scotland/rss.xml"),
#("Central Scotland & Tayside", "http://feeds.bbci.co.uk/news/scotland/tayside_and_cent
#("Wales Politics", "http://feeds.bbci.co.uk/news/wales/wales_politics/rss.xml"),
#("NW. Wales", "http://feeds.bbci.co.uk/news/wales/north_west_wales/rss.xml"),
#("NE. Wales", "http://feeds.bbci.co.uk/news/wales/north_east_wales/rss.xml"),
#("Mid. Wales", "http://feeds.bbci.co.uk/news/wales/mid_wales/rss.xml"),
#("SW. Wales", "http://feeds.bbci.co.uk/news/wales/south_west_wales/rss.xml"),
#("SE. Wales", "http://feeds.bbci.co.uk/news/wales/south_east_wales/rss.xml"),
#("Newyddion - News in Welsh", "http://feeds.bbci.co.uk/newyddion/rss.xml"),
#("Gwleidyddiaeth", "http://feeds.bbci.co.uk/newyddion/gwleidyddiaeth/rss.xml"),
#("Gogledd-Ddwyrain", "http://feeds.bbci.co.uk/newyddion/gogledd-ddwyrain/rss.xml"),
#("Gogledd-Orllewin", "http://feeds.bbci.co.uk/newyddion/gogledd-orllewin/rss.xml"),
#("Canolbarth", "http://feeds.bbci.co.uk/newyddion/canolbarth/rss.xml"),
#("De-Ddwyrain", "http://feeds.bbci.co.uk/newyddion/de-ddwyrain/rss.xml"),
#("De-Orllewin", "http://feeds.bbci.co.uk/newyddion/de-orllewin/rss.xml"),
```

```
        ]


#    **** SELECT YOUR USER PREFERENCES ****

# Title to use for the ebook.
#
title = 'BBC News'

# A brief description for the ebook.
#
description = u'BBC web site ebook created using rss feeds.'

# The max number of articles which may be downloaded from each feed.
# I've never seen more than about 70 articles in a single feed in the
# BBC feeds.
#
max_articles_per_feed = 100

# The max age of articles which may be downloaded from each feed. This is
# specified in days - note fractions of days are allowed, Eg. 2.5 (2 and a
# half days). My default of 1.5 days is the last 36 hours, the point at
# which I've decided 'news' becomes 'old news', but be warned this is not
# so good for the blogs, technology, magazine, etc., and sports feeds.
# You may wish to extend this to 2-5 but watch out ebook creation time will
# increase as well. Setting this to 30 will get everything (AFAICT) as long
# as max_articles_per_feed remains set high (except for 'Click' which is
# v. low volume and its currently oldest article is 4th Feb 2011).
#
oldest_article = 1.5

# Number of simultaneous downloads. 20 is consistantly working fine on the
# BBC News feeds with no problems. Speeds things up from the defualt of 5.
# If you have a lot of feeds and/or have increased oldest_article above 2
# then you may wish to try increasing simultaneous_downloads to 25-30,
# Or, of course, if you are in a hurry. [I've not tried beyond 20.]
#
simultaneous_downloads = 20

# Timeout for fetching files from the server in seconds. The default of
# 120 seconds, seems somewhat excessive.
#
timeout = 30

# The format string for the date shown on the ebook's first page.
# List of all values: http://docs.python.org/library/time.html
# Default in news.py has a leading space so that's mirrored here.
# As with 'feeds' select/de-select by adding/removing the initial '#',
# only one timefmt should be selected, here's a few to choose from.
#
timefmt = ' [%a, %d %b %Y]'             # [Fri, 14 Nov 2011] (Calibre default)
#timefmt = ' [%a, %d %b %Y %H:%M]'      # [Fri, 14 Nov 2011 18:30]
#timefmt = ' [%a, %d %b %Y %I:%M %p]'   # [Fri, 14 Nov 2011 06:30 PM]
#timefmt = ' [%d %b %Y]'                # [14 Nov 2011]
#timefmt = ' [%d %b %Y %H:%M]'          # [14 Nov 2011 18.30]
#timefmt = ' [%Y-%m-%d]'                # [2011-11-14]
#timefmt = ' [%Y-%m-%d-%H-%M]'          # [2011-11-14-18-30]
```

```
#
#     **** IMPORTANT ****
#
#     DO NOT EDIT BELOW HERE UNLESS YOU KNOW WHAT YOU ARE DOING.
#
#     DO NOT EDIT BELOW HERE UNLESS YOU KNOW WHAT YOU ARE DOING.
#
#     I MEAN IT, YES I DO, ABSOLUTELY, AT YOU OWN RISK. :)
#
#     **** IMPORTANT ****
#



# Author of this recipe.
__author__ = 'mattst'

# Specify English as the language of the RSS feeds (ISO-639 code).
language = 'en_GB'

# Set tags.
tags = 'news, sport, blog'

# Set publisher and publication type.
publisher = 'BBC'
publication_type = 'newspaper'

# Disable stylesheets from site.
no_stylesheets = True

# Specifies an override encoding for sites that have an incorrect charset
# specified. Default of 'None' says to auto-detect. Some other BBC recipes
# use 'utf8', which works fine (so use that if necessary) but auto-detecting
# with None is working fine, so stick with that for robustness.
encoding = None

# Sets whether a feed has full articles embedded in it. The BBC feeds do not.
use_embedded_content = False

# Removes empty feeds - why keep them!?
remove_empty_feeds = True

# Create a custom title which fits nicely in the Kindle title list.
# Requires "import time" above class declaration, and replacing
# title with custom_title in conversion_options (right column only).
# Example of string below: "BBC News - 14 Nov 2011"
#
# custom_title = "BBC News - " + time.strftime('%d %b %Y')

'''
# Conversion options for advanced users, but don't forget to comment out the
# current conversion_options below. Avoid setting 'linearize_tables' as that
# plays havoc with the 'old style' table based pages.
#
conversion_options = { 'title'       : title,
                       'comments'    : description,
                       'tags'        : tags,
```

```
                        'language'    : language,
                        'publisher'   : publisher,
                        'authors'     : publisher,
                        'smarten_punctuation' : True
                      }
'''

conversion_options = { 'smarten_punctuation' : True }

# Specify extra CSS - overrides ALL other CSS (IE. Added last).
extra_css = 'body { font-family: verdana, helvetica, sans-serif; } \
             .introduction, .first { font-weight: bold; } \
             .cross-head { font-weight: bold; font-size: 125%; } \
             .cap, .caption { display: block; font-size: 80%; font-style: italic; } \
             .cap, .caption, .caption img, .caption span { display: block; text-align: center; ma
             .byl, .byd, .byline img, .byline-name, .byline-title, .author-name, .author-positio
                .correspondent-portrait img, .byline-lead-in, .name, .bbc-role { display: block;
                text-align: center; font-size: 80%; font-style: italic; margin: 1px auto; } \
             .story-date, .published { font-size: 80%; } \
             table { width: 100%; } \
             td img { display: block; margin: 5px auto; } \
             ul { padding-top: 10px; } \
             ol { padding-top: 10px; } \
             li { padding-top: 5px; padding-bottom: 5px; } \
             h1 { text-align: center; font-size: 175%; font-weight: bold; } \
             h2 { text-align: center; font-size: 150%; font-weight: bold; } \
             h3 { text-align: center; font-size: 125%; font-weight: bold; } \
             h4, h5, h6 { text-align: center; font-size: 100%; font-weight: bold; }'

# Remove various tag attributes to improve the look of the ebook pages.
remove_attributes = [ 'border', 'cellspacing', 'align', 'cellpadding', 'colspan',
                      'valign', 'vspace', 'hspace', 'alt', 'width', 'height' ]

# Remove the (admittedly rarely used) line breaks, "<br />", which sometimes
# cause a section of the ebook to start in an unsightly fashion or, more
# frequently, a "<br />" will muck up the formatting of a correspondant's byline.
# "<br />" and "<br clear/>" are far more frequently used on the table formatted
# style of pages, and really spoil the look of the ebook pages.
preprocess_regexps     = [(re.compile(r'<br[ ]*/>', re.IGNORECASE), lambda m: ''),
                          (re.compile(r'<br[ ]*clear.*/>', re.IGNORECASE), lambda m: '')]


# Create regular expressions for tag keeping and removal to make the matches more
# robust against minor changes and errors in the HTML, Eg. double spaces, leading
# and trailing spaces, missing hyphens, and such like.
# Python regular expression ('re' class) page: http://docs.python.org/library/re.html

# ***************************************
# Regular expressions for keep_only_tags:
# ***************************************

# The BBC News HTML pages use variants of 'storybody' to denote the section of a HTML
# page which contains the main text of the article. Match storybody variants: 'storybody',
# 'story-body', 'story body','storybody ', etc.
storybody_reg_exp = '^.*story[_ -]*body.*$'

# The BBC sport and 'newsbeat' (features) HTML pages use 'blq_content' to hold the title
# and published date. This is one level above the usual news pages which have the title
```

```
    # and date within 'story-body'. This is annoying since 'blq_content' must also be kept,
    # resulting in a lot of extra things to be removed by remove_tags.
    blq_content_reg_exp = '^.*blq[_ -]*content.*$'

    # The BBC has an alternative page design structure, which I suspect is an out-of-date
    # design but which is still used in some articles, Eg. 'Click' (technology), 'FastTrack'
    # (travel), and in some sport pages. These alternative pages are table based (which is
    # why I think they are an out-of-date design) and account for -I'm guesstimaking- less
    # than 1% of all articles. They use a table class 'storycontent' to hold the article
    # and like blq_content (above) have required lots of extra removal by remove_tags.
    story_content_reg_exp = '^.*story[_ -]*content.*$'

    # Keep the sections of the HTML which match the list below. The HTML page created by
    # Calibre will fill <body> with those sections which are matched. Note that the
    # blq_content_reg_exp must be listed before storybody_reg_exp in keep_only_tags due to
    # it being the parent of storybody_reg_exp, that is to say the div class/id 'story-body'
    # will be inside div class/id 'blq_content' in the HTML (if 'blq_content' is there at
    # all). If they are the other way around in keep_only_tags then blq_content_reg_exp
    # will end up being discarded.
    keep_only_tags = [ dict(name='table', attrs={'class':re.compile(story_content_reg_exp, re.IGNOREC
                       dict(name='div',   attrs={'class':re.compile(blq_content_reg_exp, re.IGNORECAS
                       dict(name='div',   attrs={'id':re.compile(blq_content_reg_exp, re.IGNORECASE)
                       dict(name='div',   attrs={'class':re.compile(storybody_reg_exp, re.IGNORECASE)
                       dict(name='div',   attrs={'id':re.compile(storybody_reg_exp, re.IGNORECASE)})

    # ***********************************
    # Regular expressions for remove_tags:
    # ***********************************

    # Regular expression to remove share-help and variant tags. The share-help class
    # is used by the site for a variety of 'sharing' type links, Eg. Facebook, delicious,
    # twitter, email. Removed to avoid page clutter.
    share_help_reg_exp = '^.*share[_ -]*help.*$'

    # Regular expression to remove embedded-hyper and variant tags. This class is used to
    # display links to other BBC News articles on the same/similar subject.
    embedded_hyper_reg_exp = '^.*embed*ed[_ -]*hyper.*$'

    # Regular expression to remove hypertabs and variant tags. This class is used to
    # display a tab bar at the top of an article which allows the user to switch to
    # an article (viewed on the same page) providing further info., 'in depth' analysis,
    # an editorial, a correspondant's blog entry, and such like. The ability to handle
    # a tab bar of this nature is currently beyond the scope of this recipe and
    # possibly of Calibre itself (not sure about that - TO DO - check!).
    hypertabs_reg_exp = '^.*hyper[_ -]*tabs.*$'

    # Regular expression to remove story-feature and variant tags. Eg. 'story-feature',
    # 'story-feature related narrow', 'story-feature wide', 'story-feature narrow'.
    # This class is used to add additional info. boxes, or small lists, outside of
    # the main story. TO DO: Work out a way to incorporate these neatly.
    story_feature_reg_exp = '^.*story[_ -]*feature.*$'

    # Regular expression to remove video and variant tags, Eg. 'videoInStoryB',
    # 'videoInStoryC'. This class is used to embed video.
    video_reg_exp = '^.*video.*$'

    # Regular expression to remove audio and variant tags, Eg. 'audioInStoryD'.
    # This class is used to embed audio.
```

```
    audio_reg_exp = '^.*audio.*$'

    # Regular expression to remove pictureGallery and variant tags, Eg. 'pictureGallery'.
    # This class is used to embed a photo slideshow. See also 'slideshow' below.
    picture_gallery_reg_exp = '^.*picture.*$'

    # Regular expression to remove slideshow and variant tags, Eg. 'dslideshow-enclosure'.
    # This class is used to embed a slideshow (not necessarily photo) but both
    # 'slideshow' and 'pictureGallery' are used for slideshows.
    slideshow_reg_exp = '^.*slide[_ -]*show.*$'

    # Regular expression to remove social-links and variant tags. This class is used to
    # display links to a BBC bloggers main page, used in various columnist's blogs
    # (Eg. Nick Robinson, Robert Preston).
    social_links_reg_exp = '^.*social[_ -]*links.*$'

    # Regular expression to remove quote and (multi) variant tags, Eg. 'quote',
    # 'endquote', 'quote-credit', 'quote-credit-title', etc. These are usually
    # removed by 'story-feature' removal (as they are usually within them), but
    # not always. The quotation removed is always (AFAICT) in the article text
    # as well but a 2nd copy is placed in a quote tag to draw attention to it.
    # The quote class tags may or may not appear in div's.
    quote_reg_exp = '^.*quote.*$'

    # Regular expression to remove hidden and variant tags, Eg. 'hidden'.
    # The purpose of these is unclear, they seem to be an internal link to a
    # section within the article, but the text of the link (Eg. 'Continue reading
    # the main story') never seems to be displayed anyway. Removed to avoid clutter.
    # The hidden class tags may or may not appear in div's.
    hidden_reg_exp = '^.*hidden.*$'

    # Regular expression to remove comment and variant tags, Eg. 'comment-introduction'.
    # Used on the site to display text about registered users entering comments.
    comment_reg_exp = '^.*comment.*$'

    # Regular expression to remove form and variant tags, Eg. 'comment-form'.
    # Used on the site to allow registered BBC users to fill in forms, typically
    # for entering comments about an article.
    form_reg_exp = '^.*form.*$'

    # Extra things to remove due to the addition of 'blq_content' in keep_only_tags.

    #<div class="story-actions"> Used on sports pages for 'email' and 'print'.
    story_actions_reg_exp = '^.*story[_ -]*actions.*$'

    #<div class="bookmark-list"> Used on sports pages instead of 'share-help' (for
    # social networking links).
    bookmark_list_reg_exp = '^.*bookmark[_ -]*list.*$'

    #<div id="secondary-content" class="content-group">
    # NOTE: Don't remove class="content-group" that is needed.
    # Used on sports pages to link to 'similar stories'.
    secondary_content_reg_exp = '^.*secondary[_ -]*content.*$'

    #<div id="featured-content" class="content-group">
    # NOTE: Don't remove class="content-group" that is needed.
    # Used on sports pages to link to pages like 'tables', 'fixtures', etc.
    featured_content_reg_exp = '^.*featured[_ -]*content.*$'
```

```
#<div id="navigation">
# Used on sports pages to link to pages like 'tables', 'fixtures', etc.
# Used sometimes instead of "featured-content" above.
navigation_reg_exp = '^.*navigation.*$'

#<a class="skip" href="#blq-container-inner">Skip to top</a>
# Used on sports pages to link to the top of the page.
skip_reg_exp = '^.*skip.*$'

# Extra things to remove due to the addition of 'storycontent' in keep_only_tags,
# which are the alterative table design based pages. The purpose of some of these
# is not entirely clear from the pages (which are a total mess!).

# Remove mapping based tags, Eg. <map id="world_map">
# The dynamic maps don't seem to work during ebook creation. TO DO: Investigate.
map_reg_exp = '^.*map.*$'

# Remove social bookmarking variation, called 'socialBookMarks'.
social_bookmarks_reg_exp = '^.*social[_ -]*bookmarks.*$'

# Remove page navigation tools, like 'search', 'email', 'print', called 'blq-mast'.
blq_mast_reg_exp = '^.*blq[_ -]*mast.*$'

# Remove 'sharesb', I think this is a generic 'sharing' class. It seems to appear
# alongside 'socialBookMarks' whenever that appears. I am removing it as well
# under the assumption that it can appear alone as well.
sharesb_reg_exp = '^.*sharesb.*$'

# Remove class 'o'. The worst named user created css class of all time. The creator
# should immediately be fired. I've seen it used to hold nothing at all but with
# 20 or so empty lines in it. Also to hold a single link to another article.
# Whatever it was designed to do it is not wanted by this recipe. Exact match only.
o_reg_exp = '^o$'

# Remove 'promotopbg' and 'promobottombg', link lists. Have decided to
# use two reg expressions to make removing this (and variants) robust.
promo_top_reg_exp = '^.*promotopbg.*$'
promo_bottom_reg_exp = '^.*promobottombg.*$'

# Remove 'nlp', provides heading for link lists. Requires an exact match due to
# risk of matching those letters in something needed, unless I see a variation
# of 'nlp' used at a later date.
nlp_reg_exp = '^nlp$'

# Remove 'mva', provides embedded floating content of various types. Variant 'mvb'
# has also now been seen. Requires an exact match of 'mva' or 'mvb' due to risk of
# matching those letters in something needed.
mva_or_mvb_reg_exp = '^mv[ab]$'

# Remove 'mvtb', seems to be page navigation tools, like 'blq-mast'.
mvtb_reg_exp = '^mvtb$'

# Remove 'blq-toplink', class to provide a link to the top of the page.
blq_toplink_reg_exp = '^.*blq[_ -]*top[_ -]*link.*$'

# Remove 'products and services' links, Eg. desktop tools, alerts, and so on.
# Eg. Class="servicev4 ukfs_services" - what a mess of a name. Have decided to
# use two reg expressions to make removing this (and variants) robust.
```

```
    prods_services_01_reg_exp = '^.*servicev4.*$'
    prods_services_02_reg_exp = '^.*ukfs[_ -]*services.*$'

    # Remove -what I think is- some kind of navigation tools helper class, though I am
    # not sure, it's called: 'blq-rst blq-new-nav'. What I do know is it pops up
    # frequently and it is not wanted. Have decided to use two reg expressions to make
    # removing this (and variants) robust.
    blq_misc_01_reg_exp = '^.*blq[_ -]*rst.*$'
    blq_misc_02_reg_exp = '^.*blq[_ -]*new[_ -]*nav.*$'

    # Remove 'puffbox' - this may only appear inside 'storyextra', so it may not
    # need removing - I have no clue what it does other than it contains links.
    # Whatever it is - it is not part of the article and is not wanted.
    puffbox_reg_exp = '^.*puffbox.*$'

    # Remove 'sibtbg' and 'sibtbgf' - some kind of table formatting classes.
    sibtbg_reg_exp = '^.*sibtbg.*$'

    # Remove 'storyextra' - links to relevant articles and external sites.
    storyextra_reg_exp = '^.*story[_ -]*extra.*$'


    remove_tags = [ dict(name='div',  attrs={'class':re.compile(story_feature_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'class':re.compile(share_help_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(embedded_hyper_reg_exp, re.IGNORECASE
                    dict(name='div',  attrs={'class':re.compile(hypertabs_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(video_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(audio_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(picture_gallery_reg_exp, re.IGNORECAS
                    dict(name='div',  attrs={'class':re.compile(slideshow_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(quote_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(hidden_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(comment_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(story_actions_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'class':re.compile(bookmark_list_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'id':re.compile(secondary_content_reg_exp, re.IGNORECASE
                    dict(name='div',  attrs={'id':re.compile(featured_content_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'id':re.compile(navigation_reg_exp, re.IGNORECASE)}),
                    dict(name='form', attrs={'id':re.compile(form_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(quote_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(hidden_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(social_links_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(comment_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(skip_reg_exp, re.IGNORECASE)}),
                    dict(name='map', attrs={'id':re.compile(map_reg_exp, re.IGNORECASE)}),
                    dict(name='map', attrs={'name':re.compile(map_reg_exp, re.IGNORECASE)}),
                    dict(name='div', attrs={'id':re.compile(social_bookmarks_reg_exp, re.IGNORECASE)
                    dict(name='div', attrs={'id':re.compile(blq_mast_reg_exp, re.IGNORECASE)}),
                    dict(name='div', attrs={'class':re.compile(sharesb_reg_exp, re.IGNORECASE)}),
                    dict(name='div', attrs={'class':re.compile(o_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(promo_top_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(promo_bottom_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'class':re.compile(nlp_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(mva_or_mvb_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(mvtb_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(blq_toplink_reg_exp, re.IGNORECASE)}
                    dict(name='div',  attrs={'class':re.compile(prods_services_01_reg_exp, re.IGNOREC
                    dict(name='div',  attrs={'class':re.compile(prods_services_02_reg_exp, re.IGNOREC
```

```
                dict(name='div',  attrs={'class':re.compile(blq_misc_01_reg_exp, re.IGNORECASE)}),
                dict(name='div',  attrs={'class':re.compile(blq_misc_02_reg_exp, re.IGNORECASE)}),
                dict(name='div',  attrs={'class':re.compile(puffbox_reg_exp, re.IGNORECASE)}),
                dict(attrs={'class':re.compile(sibtbg_reg_exp, re.IGNORECASE)}),
                dict(attrs={'class':re.compile(storyextra_reg_exp, re.IGNORECASE)})
            ]

    # Uses url to create and return the 'printer friendly' version of the url.
    # In other words the 'print this page' address of the page.
    #
    # There are 3 types of urls used in the BBC site's rss feeds. There is just
    # 1 type for the standard news while there are 2 used for sports feed urls.
    # Note: Sports urls are linked from regular news feeds (Eg. 'News Home') when
    # there is a major story of interest to 'everyone'. So even if no BBC sports
    # feeds are added to 'feeds' the logic of this method is still needed to avoid
    # blank / missing / empty articles which have an index title and then no body.
    def print_version(self, url):

        # Handle sports page urls type 01:
        if (url.find("go/rss/-/sport1/") != -1):
            temp_url = url.replace("go/rss/-/", "")

        # Handle sports page urls type 02:
        elif (url.find("go/rss/int/news/-/sport1/") != -1):
            temp_url = url.replace("go/rss/int/news/-/", "")

        # Handle regular news page urls:
        else:
            temp_url = url.replace("go/rss/int/news/-/", "")

        # Always add "?print=true" to the end of the url.
        print_url = temp_url + "?print=true"

        return print_url


    # Remove articles in feeds based on a string in the article title or url.
    #
    # Code logic written by: Starson17 - posted in: "Recipes - Re-usable code"
    # thread, in post with title: "Remove articles from feed", see url:
    # http://www.mobileread.com/forums/showpost.php?p=1165462&postcount=6
    # Many thanks and all credit to Starson17.
    #
    # Starson17's code has obviously been altered to suite my requirements.
    def parse_feeds(self):

        # Call parent's method.
        feeds = BasicNewsRecipe.parse_feeds(self)

        # Loop through all feeds.
        for feed in feeds:

            # Loop through all articles in feed.
            for article in feed.articles[:]:

                # Match key words and remove article if there's a match.

                # Most BBC rss feed video only 'articles' use upper case 'VIDEO'
```

```
                # as a title prefix. Just match upper case 'VIDEO', so that
                # articles like 'Video game banned' won't be matched and removed.
                if 'VIDEO' in article.title:
                    feed.articles.remove(article)

                # Most BBC rss feed audio only 'articles' use upper case 'AUDIO'
                # as a title prefix. Just match upper case 'AUDIO', so that
                # articles like 'Hi-Def audio...' won't be matched and removed.
                elif 'AUDIO' in article.title:
                    feed.articles.remove(article)

                # Most BBC rss feed photo slideshow 'articles' use 'In Pictures',
                # 'In pictures', and 'in pictures', somewhere in their title.
                # Match any case of that phrase.
                elif 'IN PICTURES' in article.title.upper():
                    feed.articles.remove(article)

                # As above, but user contributed pictures. Match any case.
                elif 'YOUR PICTURES' in article.title.upper():
                    feed.articles.remove(article)

                # 'Sportsday Live' are articles which contain a constantly and
                # dynamically updated 'running commentary' during a live sporting
                # event. Match any case.
                elif 'SPORTSDAY LIVE' in article.title.upper():
                    feed.articles.remove(article)

                # Sometimes 'Sportsday Live' (above) becomes 'Live - Sport Name'.
                # These are being matched below using 'Live - ' because removing all
                # articles with 'live' in their titles would remove some articles
                # that are in fact not live sports pages. Match any case.
                elif 'LIVE - ' in article.title.upper():
                    feed.articles.remove(article)

                # 'Quiz of the week' is a Flash player weekly news quiz. Match only
                # the 'Quiz of the' part in anticipation of monthly and yearly
                # variants. Match any case.
                elif 'QUIZ OF THE' in article.title.upper():
                    feed.articles.remove(article)

                # Remove articles with 'scorecards' in the url. These are BBC sports
                # pages which just display a cricket scorecard. The pages have a mass
                # of table and css entries to display the scorecards nicely. Probably
                # could make them work with this recipe, but might take a whole day
                # of work to sort out all the css - basically a formatting nightmare.
                elif 'scorecards' in article.url:
                    feed.articles.remove(article)

        return feeds

# End of class and file.
```

This *recipe* explores only the tip of the iceberg when it comes to the power of calibre. To explore more of the abilities of calibre we'll examine a more complex real life example in the next section.

Real life example

A reasonably complex real life example that exposes more of the *API* of `BasicNewsRecipe` is the *recipe* for *The New York Times*

```
import string, re
from calibre import strftime
from calibre.web.feeds.recipes import BasicNewsRecipe
from calibre.ebooks.BeautifulSoup import BeautifulSoup

class NYTimes(BasicNewsRecipe):

    title       = 'The New York Times'
    __author__  = 'Kovid Goyal'
    description = 'Daily news from the New York Times'
    timefmt = ' [%a, %d %b, %Y]'
    needs_subscription = True
    remove_tags_before = dict(id='article')
    remove_tags_after  = dict(id='article')
    remove_tags = [dict(attrs={'class':['articleTools', 'post-tools', 'side_tool', 'nextArticleLink c
                dict(id=['footer', 'toolsRight', 'articleInline', 'navigation', 'archive', 'side_sear
                dict(name=['script', 'noscript', 'style'])]
    encoding = 'cp1252'
    no_stylesheets = True
    extra_css = 'h1 {font: sans-serif large;}\n.byline {font:monospace;}'

    def get_browser(self):
        br = BasicNewsRecipe.get_browser()
        if self.username is not None and self.password is not None:
            br.open('http://www.nytimes.com/auth/login')
            br.select_form(name='login')
            br['USERID']   = self.username
            br['PASSWORD'] = self.password
            br.submit()
        return br

    def parse_index(self):
        soup = self.index_to_soup('http://www.nytimes.com/pages/todayspaper/index.html')

        def feed_title(div):
            return ''.join(div.findAll(text=True, recursive=False)).strip()

        articles = {}
        key = None
        ans = []
        for div in soup.findAll(True,
             attrs={'class':['section-headline', 'story', 'story headline']}):

             if div['class'] == 'section-headline':
                 key = string.capwords(feed_title(div))
                 articles[key] = []
                 ans.append(key)

             elif div['class'] in ['story', 'story headline']:
                 a = div.find('a', href=True)
                 if not a:
                     continue
                 url = re.sub(r'\?.*', '', a['href'])
```

```
                url += '?pagewanted=all'
                title = self.tag_to_string(a, use_alt=True).strip()
                description = ''
                pubdate = strftime('%a, %d %b')
                summary = div.find(True, attrs={'class':'summary'})
                if summary:
                    description = self.tag_to_string(summary, use_alt=False)

                feed = key if key is not None else 'Uncategorized'
                if not articles.has_key(feed):
                    articles[feed] = []
                if not 'podcasts' in url:
                    articles[feed].append(
                              dict(title=title, url=url, date=pubdate,
                                   description=description,
                                   content=''))
        ans = self.sort_index_by(ans, {'The Front Page':-1, 'Dining In, Dining Out':1, 'Obituaries':2
        ans = [(key, articles[key]) for key in ans if articles.has_key(key)]
        return ans

    def preprocess_html(self, soup):
        refresh = soup.find('meta', {'http-equiv':'refresh'})
        if refresh is None:
            return soup
        content = refresh.get('content').partition('=')[2]
        raw = self.browser.open('http://www.nytimes.com'+content).read()
        return BeautifulSoup(raw.decode('cp1252', 'replace'))
```

We see several new features in this *recipe*. First, we have:

```
timefmt = ' [%a, %d %b, %Y]'
```

This sets the displayed time on the front page of the created ebook to be in the format, `Day, Day_Number Month, Year`. See `timefmt` (page 368).

Then we see a group of directives to cleanup the downloaded *HTML*:

```
remove_tags_before = dict(name='h1')
remove_tags_after  = dict(id='footer')
remove_tags = ...
```

These remove everything before the first `<h1>` tag and everything after the first tag whose id is `footer`. See `remove_tags` (page 367), `remove_tags_before` (page 367), `remove_tags_after` (page 367).

The next interesting feature is:

```
needs_subscription = True
...
def get_browser(self):
    ...
```

`needs_subscription = True` tells calibre that this recipe needs a username and password in order to access the content. This causes, calibre to ask for a username and password whenever you try to use this recipe. The code in `calibre.web.feeds.news.BasicNewsRecipe.get_browser()` (page 361) actually does the login into the NYT website. Once logged in, calibre will use the same, logged in, browser instance to fetch all content. See mechanize[59] to understand the code in `get_browser`.

The next new feature is the `calibre.web.feeds.news.BasicNewsRecipe.parse_index()` (page 362) method. Its job is to go to http://www.nytimes.com/pages/todayspaper/index.html and fetch the list of articles that

[59] http://wwwsearch.sourceforge.net/mechanize/

appear in *todays* paper. While more complex than simply using *RSS*, the recipe creates an ebook that corresponds very closely to the days paper. `parse_index` makes heavy use of BeautifulSoup[60] to parse the daily paper webpage.

The final new feature is the `calibre.web.feeds.news.BasicNewsRecipe.preprocess_html()` (page 363) method. It can be used to perform arbitrary transformations on every downloaded HTML page. Here it is used to bypass the ads that the nytimes shows you before each article.

Tips for developing new recipes

The best way to develop new recipes is to use the command line interface. Create the recipe using your favorite python editor and save it to a file say `myrecipe.recipe`. The *.recipe* extension is required. You can download content using this recipe with the command:

```
ebook-convert myrecipe.recipe .epub --test -vv --debug-pipeline debug
```

The command **ebook-convert** will download all the webpages and save them to the EPUB file `myrecipe.epub`. The `-vv` makes ebook-convert spit out a lot of information about what it is doing. The `--test` makes it download only a couple of articles from at most two feeds. In addition, ebook-convert will put the downloaded HTML into the `debug/input` directory, where `debug` is the directory you specified in the `--debug-pipeline` option.

Once the download is complete, you can look at the downloaded *HTML* by opening the file `debug/input/index.html` in a browser. Once you're satisfied that the download and preprocessing is happening correctly, you can generate ebooks in different formats as shown below:

```
ebook-convert myrecipe.recipe myrecipe.epub
ebook-convert myrecipe.recipe myrecipe.mobi
...
```

If you're satisfied with your recipe, and you feel there is enough demand to justify its inclusion into the set of built-in recipes, post your recipe in the calibre recipes forum[61] to share it with other calibre users.

Note: On OS X, the ebook-convert command will not be available by default. Go to Preferences->Miscellaneous and click the install command line tools button to make it available.

See Also:

ebook-convert **(page 486)** The command line interface for all ebook conversion.

Further reading

To learn more about writing advanced recipes using some of the facilities, available in `BasicNewsRecipe` you should consult the following sources:

API Documentation **(page 360)** Documentation of the `BasicNewsRecipe` class and all its important methods and fields.

BasicNewsRecipe[62] The source code of `BasicNewsRecipe`

Built-in recipes[63] The source code for the built-in recipes that come with calibre

The calibre recipes forum[64] Lots of knowledgeable calibre recipe writers hang out here.

[60] http://www.crummy.com/software/BeautifulSoup/documentation.html
[61] http://www.mobileread.com/forums/forumdisplay.php?f=228

API documentation

API Documentation for recipes

The API for writing recipes is defined by the `BasicNewsRecipe` (page 360)

class `calibre.web.feeds.news.`**BasicNewsRecipe**(*options*, *log*, *progress_reporter*)

Base class that contains logic needed in all recipes. By overriding progressively more of the functionality in this class, you can make progressively more customized/powerful recipes. For a tutorial introduction to creating recipes, see *Adding your favorite news website* (page 341).

abort_recipe_processing(*msg*)

Causes the recipe download system to abort the download of this recipe, displaying a simple feedback message to the user.

add_toc_thumbnail(*article*, *src*)

Call this from populate_article_metadata with the src attribute of an <img> tag from the article that is appropriate for use as the thumbnail representing the article in the Table of Contents. Whether the thumbnail is actually used is device dependent (currently only used by the Kindles). Note that the referenced image must be one that was successfully downloaded, otherwise it will be ignored.

classmethod adeify_images(*soup*)

If your recipe when converted to EPUB has problems with images when viewed in Adobe Digital Editions, call this method from within `postprocess_html()` (page 363).

cleanup()

Called after all articles have been download. Use it to do any cleanup like logging out of subscription sites, etc.

clone_browser(*br*)

Clone the browser br. Cloned browsers are used for multi-threaded downloads, since mechanize is not thread safe. The default cloning routines should capture most browser customization, but if you do something exotic in your recipe, you should override this method in your recipe and clone manually.

Cloned browser instances use the same, thread-safe CookieJar by default, unless you have customized cookie handling.

default_cover(*cover_file*)

Create a generic cover for recipes that dont have a cover

download()

Download and pre-process all articles from the feeds in this recipe. This method should be called only once on a particular Recipe instance. Calling it more than once will lead to undefined behavior. :return: Path to index.html

extract_readable_article(*html*, *url*)

Extracts main article content from 'html', cleans up and returns as a (article_html, extracted_title) tuple. Based on the original readability algorithm by Arc90.

get_article_url(*article*)

Override in a subclass to customize extraction of the *URL* that points to the content for each article. Return the article URL. It is called with *article*, an object representing a parsed article from a feed. See feedparser[65]. By default it looks for the original link (for feeds syndicated via a service like feedburner or pheedo) and if found, returns that or else returns article.link[66].

[65] http://packages.python.org/feedparser/
[66] http://packages.python.org/feedparser/reference-entry-link.html

get_browser(**args*, ***kwargs*)

Return a browser instance used to fetch documents from the web. By default it returns a mechanize[67] browser instance that supports cookies, ignores robots.txt, handles refreshes and has a mozilla firefox user agent.

If your recipe requires that you login first, override this method in your subclass. For example, the following code is used in the New York Times recipe to login for full access:

```
def get_browser(self):
    br = BasicNewsRecipe.get_browser(self)
    if self.username is not None and self.password is not None:
        br.open('http://www.nytimes.com/auth/login')
        br.select_form(name='login')
        br['USERID']   = self.username
        br['PASSWORD'] = self.password
        br.submit()
    return br
```

get_cover_url()

Return a *URL* to the cover image for this issue or *None*. By default it returns the value of the member *self.cover_url* which is normally *None*. If you want your recipe to download a cover for the e-book override this method in your subclass, or set the member variable *self.cover_url* before this method is called.

get_feeds()

Return a list of *RSS* feeds to fetch for this profile. Each element of the list must be a 2-element tuple of the form (title, url). If title is None or an empty string, the title from the feed is used. This method is useful if your recipe needs to do some processing to figure out the list of feeds to download. If so, override in your subclass.

get_masthead_title()

Override in subclass to use something other than the recipe title

get_masthead_url()

Return a *URL* to the masthead image for this issue or *None*. By default it returns the value of the member *self.masthead_url* which is normally *None*. If you want your recipe to download a masthead for the e-book override this method in your subclass, or set the member variable *self.masthead_url* before this method is called. Masthead images are used in Kindle MOBI files.

get_obfuscated_article(*url*)

If you set *articles_are_obfuscated* this method is called with every article URL. It should return the path to a file on the filesystem that contains the article HTML. That file is processed by the recursive HTML fetching engine, so it can contain links to pages/images on the web.

This method is typically useful for sites that try to make it difficult to access article content automatically.

classmethod image_url_processor(*baseurl*, *url*)

Perform some processing on image urls (perhaps removing size restrictions for dynamically generated images, etc.) and return the precessed URL.

index_to_soup(*url_or_raw*, *raw=False*)

Convenience method that takes an URL to the index page and returns a BeautifulSoup[68] of it.

url_or_raw: Either a URL or the downloaded index page as a string

is_link_wanted(*url*, *tag*)

Return True if the link should be followed or False otherwise. By default, raises NotImplementedError which causes the downloader to ignore it.

Parameters

[67] http://wwwsearch.sourceforge.net/mechanize/

[68] http://www.crummy.com/software/BeautifulSoup/documentation.html

- **url** – The URL to be followed
- **tag** – The Tag from which the URL was derived

javascript_login(*browser*, *username*, *password*)
This method is used to login to a website that uses javascript for its login form. After the login is complete, the cookies returned from the website are copied to a normal (non-javascript) browser and the download proceeds using those cookies.

An example implementation:

```
def javascript_login(self, browser, username, password):
    browser.visit('http://some-page-that-has-a-login')
    form = browser.select_form(nr=0) # Select the first form on the page
    form['username'] = username
    form['password'] = password
    browser.submit(timeout=120) # Submit the form and wait at most two minutes for loading t
```

Note that you can also select forms with CSS2 selectors, like this:

```
browser.select_form('form#login_form')
browser.select_from('form[name="someform"]')
```

parse_feeds()
Create a list of articles from the list of feeds returned by `BasicNewsRecipe.get_feeds()` (page 361). Return a list of `Feed` objects.

parse_index()
This method should be implemented in recipes that parse a website instead of feeds to generate a list of articles. Typical uses are for news sources that have a "Print Edition" webpage that lists all the articles in the current print edition. If this function is implemented, it will be used in preference to `BasicNewsRecipe.parse_feeds()` (page 362).

It must return a list. Each element of the list must be a 2-element tuple of the form `('feed title', list of articles)`.

Each list of articles must contain dictionaries of the form:

```
{
'title'       : article title,
'url'         : URL of print version,
'date'        : The publication date of the article as a string,
'description' : A summary of the article
'content'     : The full article (can be an empty string). Obsolete
                do not use, instead save the content to a temporary
                file and pass a file:///path/to/temp/file.html as
                the URL.
}
```

For an example, see the recipe for downloading *The Atlantic*. In addition, you can add 'author' for the author of the article.

If you want to abort processing for some reason and have calibre show the user a simple message instead of an error, call `abort_recipe_processing()` (page 360).

populate_article_metadata(*article*, *soup*, *first*)
Called when each HTML page belonging to article is downloaded. Intended to be used to get article metadata like author/summary/etc. from the parsed HTML (soup). :param article: A object of class `calibre.web.feeds.Article`. If you change the summary, remember to also change the text_summary :param soup: Parsed HTML belonging to this article :param first: True iff the parsed HTML is the first page of the article.

postprocess_book(*oeb*, *opts*, *log*)
Run any needed post processing on the parsed downloaded e-book.

Parameters

- **oeb** – An OEBBook object
- **opts** – Conversion options

postprocess_html(*soup*, *first_fetch*)
This method is called with the source of each downloaded *HTML* file, after it is parsed for links and images. It can be used to do arbitrarily powerful post-processing on the *HTML*. It should return *soup* after processing it.

Parameters

- **soup** – A BeautifulSoup[69] instance containing the downloaded *HTML*.
- **first_fetch** – True if this is the first page of an article.

preprocess_html(*soup*)
This method is called with the source of each downloaded *HTML* file, before it is parsed for links and images. It is called after the cleanup as specified by remove_tags etc. It can be used to do arbitrarily powerful pre-processing on the *HTML*. It should return *soup* after processing it.

soup: A BeautifulSoup[70] instance containing the downloaded *HTML*.

preprocess_raw_html(*raw_html*, *url*)
This method is called with the source of each downloaded *HTML* file, before it is parsed into an object tree. raw_html is a unicode string representing the raw HTML downloaded from the web. url is the URL from which the HTML was downloaded.

Note that this method acts *before* preprocess_regexps.

This method must return the processed raw_html as a unicode object.

classmethod print_version(*url*)
Take a *url* pointing to the webpage with article content and return the *URL* pointing to the print version of the article. By default does nothing. For example:

```
def print_version(self, url):
    return url + '?&pagewanted=print'
```

skip_ad_pages(*soup*)
This method is called with the source of each downloaded *HTML* file, before any of the cleanup attributes like remove_tags, keep_only_tags are applied. Note that preprocess_regexps will have already been applied. It is meant to allow the recipe to skip ad pages. If the soup represents an ad page, return the HTML of the real page. Otherwise return None.

soup: A BeautifulSoup[71] instance containing the downloaded *HTML*.

sort_index_by(*index*, *weights*)
Convenience method to sort the titles in *index* according to *weights*. *index* is sorted in place. Returns *index*.

index: A list of titles.

weights: A dictionary that maps weights to titles. If any titles in index are not in weights, they are assumed to have a weight of 0.

[69] http://www.crummy.com/software/BeautifulSoup/documentation.html
[70] http://www.crummy.com/software/BeautifulSoup/documentation.html
[71] http://www.crummy.com/software/BeautifulSoup/documentation.html

classmethod tag_to_string(*tag*, *use_alt=True*, *normalize_whitespace=True*)
Convenience method to take a BeautifulSoup[72] *Tag* and extract the text from it recursively, including any CDATA sections and alt tag attributes. Return a possibly empty unicode string.

use_alt: If *True* try to use the alt attribute for tags that don't have any textual content

tag: BeautifulSoup[73] *Tag*

articles_are_obfuscated = False
Set to True and implement `get_obfuscated_article()` (page 362) to handle websites that try to make it difficult to scrape content.

auto_cleanup = False
Automatically extract all the text from downloaded article pages. Uses the algorithms from the readability project. Setting this to True, means that you do not have to worry about cleaning up the downloaded HTML manually (though manual cleanup will always be superior).

auto_cleanup_keep = None
Specify elements that the auto cleanup algorithm should never remove The syntax is a XPath expression. For example:

```
auto_cleanup_keep = '//div[@id="article-image"]' will keep all divs with
                                                 id="article-image"
auto_cleanup_keep = '//*[@class="important"]' will keep all elements
                                              with class="important"
auto_cleanup_keep = '//div[@id="article-image"]|//span[@class="important"]'
                  will keep all divs with id="article-image" and spans
                  with class="important"
```

center_navbar = True
If True the navigation bar is center aligned, otherwise it is left aligned

conversion_options = {}
Recipe specific options to control the conversion of the downloaded content into an e-book. These will override any user or plugin specified values, so only use if absolutely necessary. For example:

```
conversion_options = {
  'base_font_size'   : 16,
  'tags'             : 'mytag1,mytag2',
  'title'            : 'My Title',
  'linearize_tables' : True,
}
```

cover_margins = (0, 0, '#ffffff')
By default, the cover image returned by get_cover_url() will be used as the cover for the periodical. Overriding this in your recipe instructs calibre to render the downloaded cover into a frame whose width and height are expressed as a percentage of the downloaded cover. cover_margins = (10, 15, '#ffffff') pads the cover with a white margin 10px on the left and right, 15px on the top and bottom. Color names defined at http://www.imagemagick.org/script/color.php Note that for some reason, white does not always work on windows. Use #ffffff instead

delay = 0
Delay between consecutive downloads in seconds. The argument may be a floating point number to indicate a more precise time.

description = u''
A couple of lines that describe the content this recipe downloads. This will be used primarily in a GUI that presents a list of recipes.

[72] http://www.crummy.com/software/BeautifulSoup/documentation.html
[73] http://www.crummy.com/software/BeautifulSoup/documentation.html

encoding = None
Specify an override encoding for sites that have an incorrect charset specification. The most common being specifying `latin1` and using `cp1252`. If None, try to detect the encoding. If it is a callable, the callable is called with two arguments: The recipe object and the source to be decoded. It must return the decoded source.

extra_css = None
Specify any extra *CSS* that should be added to downloaded *HTML* files It will be inserted into *<style>* tags, just before the closing *</head>* tag thereby overriding all *CSS* except that which is declared using the style attribute on individual *HTML* tags. For example:

```
extra_css = '.heading { font: serif x-large }'
```

feeds = None
List of feeds to download Can be either `[url1, url2, ...]` or `[('title1', url1), ('title2', url2),...]`

filter_regexps = []
List of regular expressions that determines which links to ignore If empty it is ignored. Used only if is_link_wanted is not implemented. For example:

```
filter_regexps = [r'ads\.doubleclick\.net']
```

will remove all URLs that have *ads.doubleclick.net* in them.

Only one of `BasicNewsRecipe.match_regexps` (page 366) or `BasicNewsRecipe.filter_regexps` (page 365) should be defined.

ignore_duplicate_articles = None
Ignore duplicates of articles that are present in more than one section. A duplicate article is an article that has the same title and/or URL. To ignore articles with the same title, set this to: ignore_duplicate_articles = {'title'} To use URLs instead, set it to: ignore_duplicate_articles = {'url'} To match on title or URL, set it to: ignore_duplicate_articles = {'title', 'url'}

keep_only_tags = []
Keep only the specified tags and their children. For the format for specifying a tag see `BasicNewsRecipe.remove_tags` (page 367). If this list is not empty, then the *<body>* tag will be emptied and re-filled with the tags that match the entries in this list. For example:

```
keep_only_tags = [dict(id=['content', 'heading'])]
```

will keep only tags that have an *id* attribute of *"content"* or *"heading"*.

language = 'und'
The language that the news is in. Must be an ISO-639 code either two or three characters long

masthead_url = None
By default, calibre will use a default image for the masthead (Kindle only). Override this in your recipe to provide a url to use as a masthead.

match_regexps = []
List of regular expressions that determines which links to follow If empty, it is ignored. Used only if is_link_wanted is not implemented. For example:

```
match_regexps = [r'page=[0-9]+']
```

will match all URLs that have *page=some number* in them.

Only one of `BasicNewsRecipe.match_regexps` (page 366) or `BasicNewsRecipe.filter_regexps` (page 365) should be defined.

max_articles_per_feed = 100
Maximum number of articles to download from each feed. This is primarily useful for feeds that don't have article dates. For most feeds, you should use `BasicNewsRecipe.oldest_article` (page 366)

needs_subscription = False
If True the GUI will ask the user for a username and password to use while downloading If set to "optional" the use of a username and password becomes optional

no_stylesheets = False
Convenient flag to disable loading of stylesheets for websites that have overly complex stylesheets unsuitable for conversion to ebooks formats If True stylesheets are not downloaded and processed

oldest_article = 7.0
Oldest article to download from this news source. In days.

preprocess_regexps = []
List of *regexp* substitution rules to run on the downloaded *HTML*. Each element of the list should be a two element tuple. The first element of the tuple should be a compiled regular expression and the second a callable that takes a single match object and returns a string to replace the match. For example:

```
preprocess_regexps = [
   (re.compile(r'<!--Article ends here-->.*</body>', re.DOTALL|re.IGNORECASE),
    lambda match: '</body>'),
]
```

will remove everythong from *<!–Article ends here–>* to *</body>*.

publication_type = 'unknown'
Publication type Set to newspaper, magazine or blog. If set to None, no publication type metadata will be written to the opf file.

recipe_disabled = None
Set to a non empty string to disable this recipe The string will be used as the disabled message

recursions = 0
Number of levels of links to follow on article webpages

remove_attributes = []
List of attributes to remove from all tags For example:

```
remove_attributes = ['style', 'font']
```

remove_empty_feeds = False
If True empty feeds are removed from the output. This option has no effect if parse_index is overridden in the sub class. It is meant only for recipes that return a list of feeds using *feeds* or `get_feeds()` (page 361). It is also used if you use the ignore_duplicate_articles option.

remove_javascript = True
Convenient flag to strip all javascript tags from the downloaded HTML

remove_tags = []
List of tags to be removed. Specified tags are removed from downloaded HTML. A tag is specified as a dictionary of the form:

```
{
 name       : 'tag name',   #e.g. 'div'
 attrs      : a dictionary, #e.g. {class: 'advertisment'}
}
```

All keys are optional. For a full explanantion of the search criteria, see Beautiful Soup[74] A common example:

```
remove_tags = [dict(name='div', attrs={'class':'advert'})]
```

This will remove all <*div class="advert"*> tags and all their children from the downloaded *HTML*.

remove_tags_after = None
Remove all tags that occur after the specified tag. For the format for specifying a tag see BasicNewsRecipe.remove_tags (page 367). For example:

```
remove_tags_after = [dict(id='content')]
```

will remove all tags after the first element with *id="content"*.

remove_tags_before = None
Remove all tags that occur before the specified tag. For the format for specifying a tag see BasicNewsRecipe.remove_tags (page 367). For example:

```
remove_tags_before = dict(id='content')
```

will remove all tags before the first element with *id="content"*.

requires_version = (0, 6, 0)
Minimum calibre version needed to use this recipe

reverse_article_order = False
Reverse the order of articles in each feed

simultaneous_downloads = 5
Number of simultaneous downloads. Set to 1 if the server is picky. Automatically reduced to 1 if BasicNewsRecipe.delay (page 365) > 0

summary_length = 500
Max number of characters in the short description

template_css = u'\n .article_date {\n color: gray; font-family: monospace;\n }\n\n .article_description {\n text-indent:
The CSS that is used to style the templates, i.e., the navigation bars and the Tables of Contents. Rather than overriding this variable, you should use *extra_css* in your recipe to customize look and feel.

timefmt = ' [%a, %d %b %Y]'
The format string for the date shown on the first page. By default: Day_Name, Day_Number Month_Name Year

timeout = 120.0
Timeout for fetching files from server in seconds

title = u'Unknown News Source'
The title to use for the ebook

use_embedded_content = None
Normally we try to guess if a feed has full articles embedded in it based on the length of the embedded content. If *None*, then the default guessing is used. If *True* then the we always assume the feeds has embedded content and if *False* we always assume the feed does not have embedded content.

use_javascript_to_login = False
If you set this True, then calibre will use javascript to login to the website. This is needed for some websites that require the use of javascript to login. If you set this to True you must implement the javascript_login() (page 362) method, to do the actual logging in.

[74] http://www.crummy.com/software/BeautifulSoup/documentation.html#Thebasicfindmethod:findAll(name,attrs,recursive,text,limit,**kwargs)

1.7.2 Managing subgroups of books, for example "genre"

Some people wish to organize the books in their library into subgroups, similar to subfolders. The most commonly provided reason is to create genre hierarchies, but there are many others. One user asked for a way to organize textbooks by subject and course number. Another wanted to keep track of gifts by subject and recipient. This tutorial will use the genre example for the rest of this post.

Before going on, please note that we are not talking about folders on the hard disk. Subgroups are not file folders. Books will not be copied anywhere. Calibre's library file structure is not affected. Instead, we are presenting a way to organize and display subgroups of books within a calibre library.

- Setup (page 370)
- Searching (page 372)
- Restrictions (page 372)
- Useful Template Functions (page 373)

The commonly-provided requirements for subgroups such as genres are:

- A subgroup (e.g., a genre) must contain (point to) books, not categories of books. This is what distinguishes subgroups from calibre user categories.
- A book can be in multiple subgroups (genres). This distinguishes subgroups from physical file folders.
- Subgroups (genres) must form a hierarchy; subgroups can contain subgroups.

Tags give you the first two. If you tag a book with the genre then you can use the tag browser (or search) for find the books with that genre, giving you the first. Many books can have the same tag(s), giving you the second. The problem is that tags don't satisfy the third requirement. They don't provide a hierarchy.



Calibre's hierarchy feature gives you the third, the ability to see the genres in a 'tree' and the ability to easily search for books in genre or sub-genre. For example, assume that your genre structure is similar to the following:

```
Genre
    . History
    .. Japanese
    .. Military
    .. Roman
    . Mysteries
    .. English
    .. Vampire
    . Science Fiction
    .. Alternate History
    .. Military
    .. Space Opera
    . Thrillers
    .. Crime
    .. Horror
    etc.
```

By using the hierarchy feature, you can see these genres in the tag browser in tree form, as shown in the screen image. In this example the outermost level (Genre) is a custom column that contains the genres. Genres containing sub-genres appear with a small triangle next to them. Clicking on that triangle will open the item and show the sub-genres, as you can see with History and Science Fiction.

Clicking on a genre can search for all books with that genre or children of that genre. For example, clicking on Science Fiction can give all three of the child genres, Alternate History, Military, and Space Opera. Clicking on Alternate History will give books in that genre, ignoring those in Military and Space Opera. Of course, a book can have multiple genres. If a book has both Space Opera and Military genres, then you will see that book if you click on either genre. Searching is discussed in more detail below.

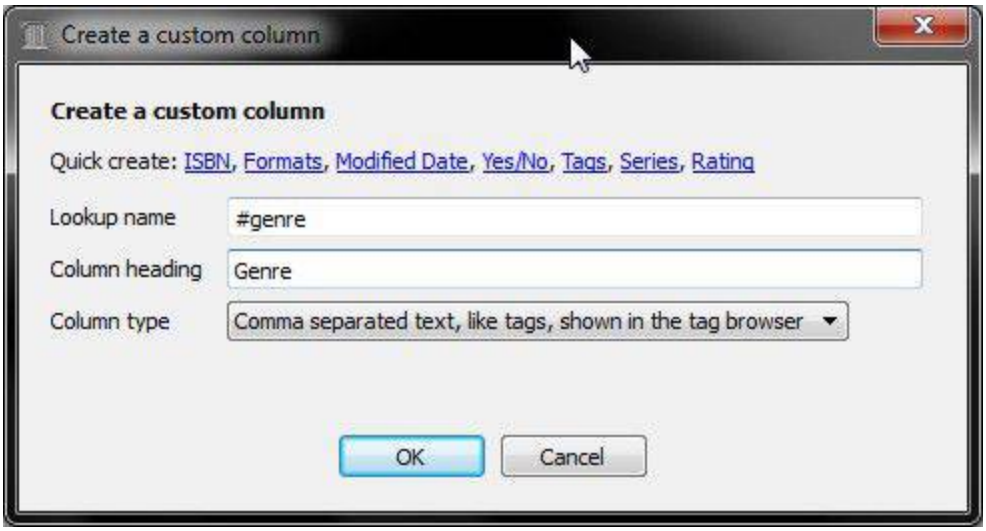
Another thing you can see from the image is that the genre Military appears twice, once under History and once under Science Fiction. Because the genres are in a hierarchy, these are two separate genres. A book can be in one, the other, or (doubtfully in this case) both. For example, the books in Winston Churchill's "The Second World War" could be in "History.Military". David Weber's Honor Harrington books could be in "Science Fiction.Military", and for that matter also in "Science Fiction.Space Opera."

Once a genre exists, that is at least one book has that genre, you can easily apply it to other books by dragging the books from the library view onto the genre you want the books to have. You can also apply genres in the metadata editors; more on this below.

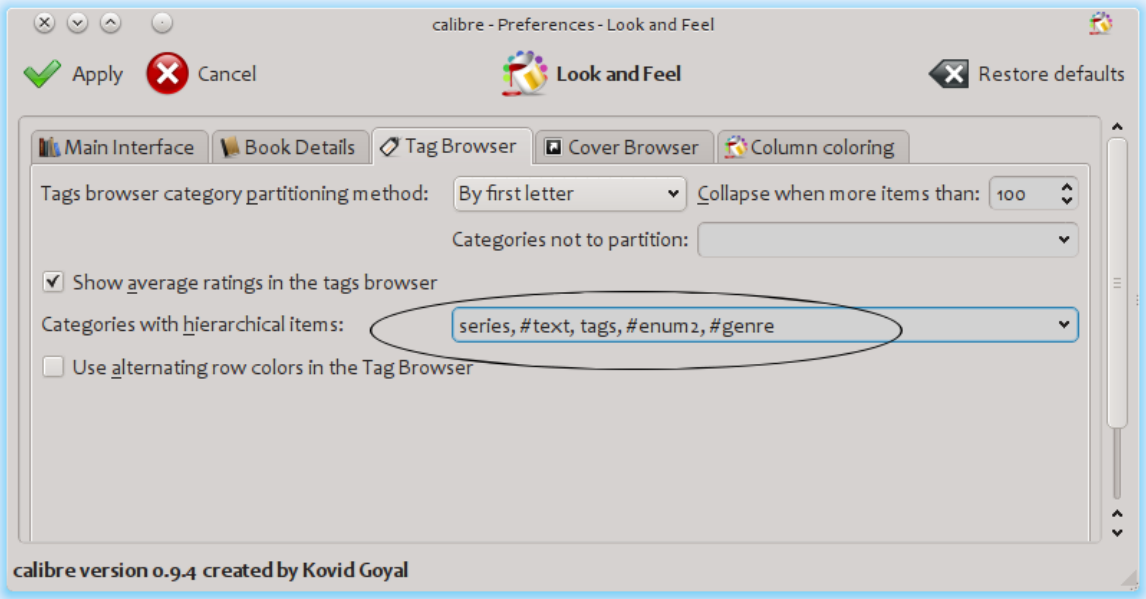
Setup

By now, your question might be "How was all of this up?" There are three steps: 1) create the custom column, 2) tell calibre that the new column is to be treated as a hierarchy, and 3) add genres.

You create the custom column in the usual way, using Preferences -> Add your own columns. This example uses "#genre" as the lookup name and "Genre" as the column heading. The column type is "Comma-separated text, like tags, shown in the tag browser."

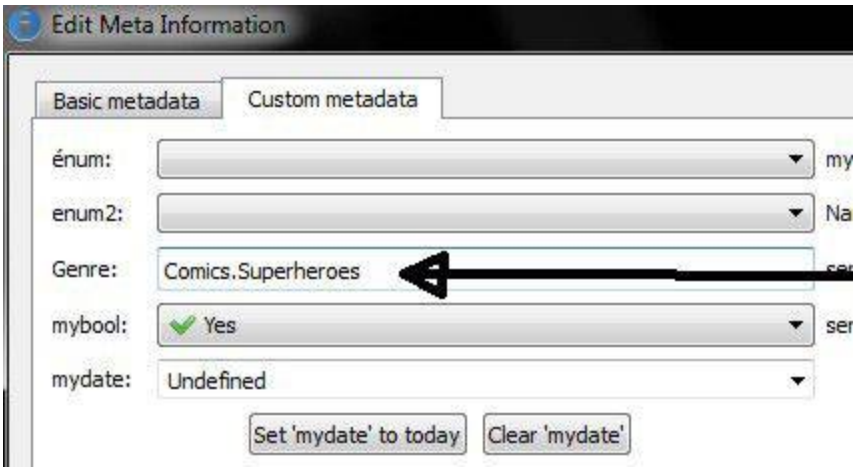


Then after restarting calibre, you must tell calibre that the column is to be treated as a hierarchy. Go to Preferences -> Look and Feel -> Tag Browser and enter the lookup name "#genre" into the "Categories with hierarchical items" box. Press Apply, and you are done with setting up.



At the point there are no genres in the column. We are left with the last step: how to apply a genre to a book. A genre does not exist in calibre until it appears on at least one book. To learn how to apply a genre for the first time, we must go into some detail about what a genre looks like in the metadata for a book.

A hierarchy of 'things' is built by creating an item consisting of phrases separated by periods. Continuing the genre example, these items would "History.Military", "Mysteries.Vampire", "Science Fiction.Space Opera", etc. Thus to create a new genre, you pick a book that should have that genre, edit its metadata, and enter the new genre into the column you created. Continuing our example, if you want to assign a new genre "Comics" with a sub-genre "Superheroes" to a book, you would 'edit metadata' for that (comic) book, choose the Custom metadata tab, and then enter "Comics.Superheroes" as shown in the following (ignore the other custom columns):



After doing the above, you see in the tag browser:



From here on, to apply this new genre to a book (a comic book, presumably), you can either drag the book onto the genre, or add it to the book using edit metadata in exactly the same way as done above.

Searching



The easiest way to search for genres is using the tag browser, clicking on the genre you wish to see. Clicking on a genre with children will show you books with that genre and all child genres. However, this might bring up a question.

Just because a genre has children doesn't mean that it isn't a genre in its own right. For example, a book can have the genre "History" but not "History.Military". How do you search for books with only "History"?

The tag browser search mechanism knows if an item has children. If it does, clicking on the item cycles through 5 searches instead of the normal three. The first is the normal green plus, which shows you books with that genre only (e.g., History). The second is a doubled plus (shown above), which shows you books with that genre and all sub-genres (e.g., History and History.Military). The third is the normal red minus, which shows you books without that exact genre. The fourth is a doubled minus, which shows you books without that genre or sub-genres. The fifth is back to the beginning, no mark, meaning no search.

Restrictions

If you search for a genre then create a saved search for it, you can use the 'restrict to' box to create a virtual library of books with that genre. This is useful if you want to do other searches within the genre or to manage/update metadata for books in the genre. Continuing our example, you can create a saved search named 'History.Japanese' by first clicking on the genre Japanese in the tag browser to get a search into the search box, entering History.Japanese into the saved search box, then pushing the "save search" button (the green box with the white plus, on the right-hand side).



After creating the saved search, you can use it as a restriction.



Useful Template Functions

You might want to use the genre information in a template, such as with save to disk or send to device. The question might then be "How do I get the outermost genre name or names?" A calibre template function, subitems, is provided to make doing this easier.

For example, assume you want to add the outermost genre level to the save-to-disk template to make genre folders, as in "History/The Gathering Storm - Churchill, Winston". To do this, you must extract the first level of the hierarchy and add it to the front along with a slash to indicate that it should make a folder. The template below accomplishes this:

```
{#genre:subitems(0,1)||/}{title} - {authors}
```

See *The template language* (page 375) for more information templates and the `subitems()` function.

1.7.3 XPath Tutorial

In this tutorial, you will be given a gentle introduction to XPath[75], a query language that can be used to select arbitrary parts of HTML[76] documents in calibre. XPath is a widely used standard, and googling it will yield a ton of information. This tutorial, however, focuses on using XPath for ebook related tasks like finding chapter headings in an unstructured HTML document.

Contents

- Selecting by tagname (page 374)
- Selecting by attributes (page 374)
- Selecting by tag content (page 375)
- Sample ebook (page 375)
- XPath built-in functions (page 375)

Selecting by tagname

The simplest form of selection is to select tags by name. For example, suppose you want to select all the `<h2>` tags in a document. The XPath query for this is simply:

```
//h:h2        (Selects all <h2> tags)
```

The prefix *//* means *search at any level of the document*. Now suppose you want to search for `<span>` tags that are inside `<a>` tags. That can be achieved with:

```
//h:a/h:span    (Selects <span> tags inside <a> tags)
```

If you want to search for tags at a particular level in the document, change the prefix:

```
/h:body/h:div/h:p (Selects <p> tags that are children of <div> tags that are
             children of the <body> tag)
```

This will match only `<p>A very short ebook to demonstrate the use of XPath.</p>` in the Sample ebook (page 375) but not any of the other `<p>` tags. The `h:` prefix in the above examples is needed to match XHTML tags. This is because internally, calibre represents all content as XHTML. In XHTML tags have a *namespace*, and `h:` is the namespace prefix for HTML tags.

Now suppose you want to select both `<h1>` and `<h2>` tags. To do that, we need a XPath construct called *predicate*. A *predicate* is simply a test that is used to select tags. Tests can be arbitrarily powerful and as this tutorial progresses, you will see more powerful examples. A predicate is created by enclosing the test expression in square brackets:

```
//*[name()='h1' or name()='h2']
```

There are several new features in this XPath expression. The first is the use of the wildcard `*`. It means *match any tag*. Now look at the test expression `name()='h1' or name()='h2'`. *name()* is an example of a *built-in function*. It simply evaluates to the name of the tag. So by using it, we can select tags whose names are either *h1* or *h2*. Note that the *name()* function ignores namespaces so that there is no need for the `h:` prefix. XPath has several useful built-in functions. A few more will be introduced in this tutorial.

[75] http://en.wikipedia.org/wiki/XPath
[76] http://en.wikipedia.org/wiki/HTML

Selecting by attributes

To select tags based on their attributes, the use of predicates is required:

```
//*[@style]              (Select all tags that have a style attribute)
//*[@class="chapter"]    (Select all tags that have class="chapter")
//h:h1[@class="bookTitle"] (Select all h1 tags that have class="bookTitle")
```

Here, the `@` operator refers to the attributes of the tag. You can use some of the XPath built-in functions (page 375) to perform more sophisticated matching on attribute values.

Selecting by tag content

Using XPath, you can even select tags based on the text they contain. The best way to do this is to use the power of *regular expressions* via the built-in function *re:test()*:

```
//h:h2[re:test(., 'chapter|section', 'i')] (Selects <h2> tags that contain the words chapter or
                                           section)
```

Here the `.` operator refers to the contents of the tag, just as the `@` operator referred to its attributes.

Sample ebook

```
<html>
    <head>
        <title>A very short ebook</title>
        <meta name="charset" value="utf-8" />
    </head>
    <body>
        <h1 class="bookTitle">A very short ebook</h1>
        <p style="text-align:right">Written by Kovid Goyal</p>
        <div class="introduction">
            <p>A very short ebook to demonstrate the use of XPath.</p>
        </div>

        <h2 class="chapter">Chapter One</h2>
        <p>This is a truly fascinating chapter.</p>

        <h2 class="chapter">Chapter Two</h2>
        <p>A worthy continuation of a fine tradition.</p>
    </body>
</html>
```

XPath built-in functions

name() The name of the current tag.

contains() `contains(s1, s2)` returns *true* if s1 contains s2.

re:test() `re:test(src, pattern, flags)` returns *true* if the string *src* matches the regular expression *pattern*. A particularly useful flag is `i`, it makes matching case insensitive. A good primer on the syntax for regular expressions can be found at regexp syntax[77]

[77] http://docs.python.org/lib/re-syntax.html

1.7.4 The calibre template language

The calibre template language is used in various places. It is used to control the folder structure and file name when saving files from the calibre library to the disk or eBook reader. It is also used to define "virtual" columns that contain data from other columns and so on.

The basic template language is very simple, but has very powerful advanced features. The basic idea is that a template consists of text and names in curly brackets that are then replaced by the corresponding metadata from the book being processed. So, for example, the default template used for saving books to device in calibre is:

```
{author_sort}/{title}/{title} - {authors}
```

For the book "The Foundation" by "Isaac Asimov" it will become:

```
Asimov, Isaac/The Foundation/The Foundation - Isaac Asimov
```

The slashes are text, which is put into the template where it appears. For example, if your template is:

```
{author_sort} Some Important Text {title}/{title} - {authors}
```

For the book "The Foundation" by "Isaac Asimov" it will become:

```
Asimov, Isaac Some Important Text The Foundation/The Foundation - Isaac Asimov
```

You can use all the various metadata fields available in calibre in a template, including any custom columns you have created yourself. To find out the template name for a column simply hover your mouse over the column header. Names for custom fields (columns you have created yourself) always have a # as the first character. For series type custom fields, there is always an additional field named `#seriesname_index` that becomes the series index for that series. So if you have a custom series field named `#myseries`, there will also be a field named `#myseries_index`.

In addition to the column based fields, you also can use:

```
{formats} - A list of formats available in the calibre library for a book
{identifiers:select(isbn)} - The ISBN number of the book
```

If a particular book does not have a particular piece of metadata, the field in the template is automatically removed for that book. Consider, for example:

```
{author_sort}/{series}/{title} {series_index}
```

If a book has a series, the template will produce:

```
Asimov, Isaac/Foundation/Second Foundation 3
```

and if a book does not have a series:

```
Asimov, Isaac/Second Foundation
```

(calibre automatically removes multiple slashes and leading or trailing spaces).

Advanced formatting

You can do more than just simple substitution with the templates. You can also conditionally include text and control how the substituted data is formatted.

First, conditionally including text. There are cases where you might want to have text appear in the output only if a field is not empty. A common case is `series` and `series_index`, where you want either nothing or the two values with a hyphen between them. Calibre handles this case using a special field syntax.

For example, assume you want to use the template:

```
{series} - {series_index} - {title}
```

If the book has no series, the answer will be `- - title`. Many people would rather the result be simply `title`, without the hyphens. To do this, use the extended syntax `{field:|prefix_text|suffix_text}`. When you use this syntax, if field has the value SERIES then the result will be `prefix_textSERIESsuffix_text`. If field has no value, then the result will be the empty string (nothing); the prefix and suffix are ignored. The prefix and suffix can contain blanks. **Do not use subtemplates ('{ ... }') or functions (see below) as the prefix or the suffix.**

Using this syntax, we can solve the above series problem with the template:

```
{series}{series_index:| - | - }{title}
```

The hyphens will be included only if the book has a series index, which it will have only if it has a series.

Notes: you must include the : character if you want to use a prefix or a suffix. You must either use no | characters or both of them; using one, as in `{field:| - }`, is not allowed. It is OK not to provide any text for one side or the other, such as in `{series:|| - }`. Using `{title:||}` is the same as using `{title}`.

Second: formatting. Suppose you wanted to ensure that the series_index is always formatted as three digits with leading zeros. This would do the trick:

```
{series_index:0>3s} - Three digits with leading zeros
```

If instead of leading zeros you want leading spaces, use:

```
{series_index:>3s} - Three digits with leading spaces
```

For trailing zeros, use:

```
{series_index:0<3s} - Three digits with trailing zeros
```

If you use series indices with sub values (e.g., 1.1), you might want to ensure that the decimal points line up. For example, you might want the indices 1 and 2.5 to appear as 01.00 and 02.50 so that they will sort correctly. To do this, use:

```
{series_index:0>5.2f} - Five characters, consisting of two digits with leading zeros, a decimal point
```

If you want only the first two letters of the data, use:

```
{author_sort:.2} - Only the first two letter of the author sort name
```

The calibre template language comes from python and for more details on the syntax of these advanced formatting operations, look at the Python documentation[78].

Advanced features

Using templates in custom columns

There are sometimes cases where you want to display metadata that calibre does not normally display, or to display data in a way different from how calibre normally does. For example, you might want to display the ISBN, a field that calibre does not display. You can use custom columns for this by creating a column with the type 'column built from other columns' (hereafter called composite columns), and entering a template. Result: calibre will display a column showing the result of evaluating that template. To display the ISBN, create the column and enter `{identifiers:select(isbn)}` into the template box. To display a column containing the values of two series custom columns separated by a comma, use `{#series1:||,}{#series2}`.

Composite columns can use any template option, including formatting.

[78] http://docs.python.org/library/string.html#format-string-syntax

You cannot change the data contained in a composite column. If you edit a composite column by double-clicking on any item, you will open the template for editing, not the underlying data. Editing the template on the GUI is a quick way of testing and changing composite columns.

Using functions in templates - single-function mode

Suppose you want to display the value of a field in upper case, when that field is normally in title case. You can do this (and many more things) using the functions available for templates. For example, to display the title in upper case, use `{title:uppercase()}`. To display it in title case, use `{title:titlecase()}`.

Function references appear in the format part, going after the `:` and before the first `|` or the closing `}`. If you have both a format and a function reference, the function comes after another `:`. Functions must always end with `()`. Some functions take extra values (arguments), and these go inside the `()`.

Functions are always applied before format specifications. See further down for an example of using both a format and a function, where this order is demonstrated.

The syntax for using functions is `{field:function(arguments)}`, or `{field:function(arguments)|prefix|suffix}`. Arguments are separated by commas. Commas inside arguments must be preceeded by a backslash ('\'). The last (or only) argument cannot contain a closing parenthesis (')'). Functions return the value of the field used in the template, suitably modified.

Important: If you have programming experience, please note that the syntax in this mode (single function) is not what you might expect. Strings are not quoted. Spaces are significant. All arguments must be constants; there is no sub-evaluation. **Do not use subtemplates ('{ ... }') as function arguments.** Instead, use *template program mode* (page 380) and *general program mode* (page 398).

Many functions use regular expressions. In all cases, regular expression matching is case-insensitive.

The functions available are listed below. Note that the definitive documentation for functions is available in the section *Function classification* (page 385):

- `lowercase()` – return value of the field in lower case.
- `uppercase()` – return the value of the field in upper case.
- `titlecase()` – return the value of the field in title case.
- `capitalize()` – return the value with the first letter upper case and the rest lower case.
- `contains(pattern, text if match, text if not match)` – checks if field contains matches for the regular expression *pattern*. Returns *text if match* if matches are found, otherwise it returns *text if no match*.
- `count(separator)` – interprets the value as a list of items separated by *separator*, returning the number of items in the list. Most lists use a comma as the separator, but authors uses an ampersand. Examples: *{tags:count(,)}*, *{authors:count(&)}*
- `format_number(template)` – interprets the value as a number and format that number using a python formatting template such as "{0:5.2f}" or "{0:,d}" or "${0:5,.2f}". The field_name part of the template must be a 0 (zero) (the "{0:" in the above examples). See the template language and python documentation for more examples. Returns the empty string if formatting fails.
- `human_readable()` – expects the value to be a number and returns a string representing that number in KB, MB, GB, etc.
- `ifempty(text)` – if the field is not empty, return the value of the field. Otherwise return *text*.
- `in_list(separator, pattern, found_val, not_found_val)` – interpret the field as a list of items separated by *separator*, comparing the *pattern* against each value in the list. If the pattern matches a value, return *found_val*, otherwise return *not_found_val*.

- `language_codes(lang_strings)` – return the language codes for the strings passed in *lang_strings*. The strings must be in the language of the current locale. *Lang_strings* is a comma-separated list.
- `language_strings(lang_codes, localize)` – return the strings for the language codes passed in *lang_codes*. If *localize* is zero, return the strings in English. If localize is not zero, return the strings in the language of the current locale. *Lang_codes* is a comma-separated list.
- `list_item(index, separator)` – interpret the field as a list of items separated by *separator*, returning the *index'th item. The first item is number zero. The last item can be returned using 'list_item(-1,separator)*. If the item is not in the list, then the empty value is returned. The separator has the same meaning as in the *count* function.
- `re(pattern, replacement)` – return the field after applying the regular expression. All instances of *pattern* are replaced with *replacement*. As in all of calibre, these are python-compatible regular expressions.
- `shorten(left chars, middle text, right chars)` – Return a shortened version of the field, consisting of *left chars* characters from the beginning of the field, followed by *middle text*, followed by *right chars* characters from the end of the string. *Left chars* and *right chars* must be integers. For example, assume the title of the book is *Ancient English Laws in the Times of Ivanhoe*, and you want it to fit in a space of at most 15 characters. If you use `{title:shorten(9,-,5)}`, the result will be *Ancient E-nhoe*. If the field's length is less than `left chars` + `right chars` + the length of `middle text`, then the field will be used intact. For example, the title *The Dome* would not be changed.
- `swap_around_comma(val)` `` -- given a value of the form ``B, A``, return `A B`. This is most useful for converting names in LN, FN format to FN LN. If there is no comma, the function returns val unchanged.
- `switch(pattern, value, pattern, value, ..., else_value)` – for each `pattern, value` pair, checks if the field matches the regular expression `pattern` and if so, returns that `value`. If no `pattern` matches, then `else_value` is returned. You can have as many `pattern, value` pairs as you want.
- `lookup(pattern, field, pattern, field, ..., else_field)` – like switch, except the arguments are field (metadata) names, not text. The value of the appropriate field will be fetched and used. Note that because composite columns are fields, you can use this function in one composite field to use the value of some other composite field. This is extremely useful when constructing variable save paths (more later).
- `select(key)` – interpret the field as a comma-separated list of items, with the items being of the form "id:value". Find the pair with the id equal to key, and return the corresponding value. This function is particularly useful for extracting a value such as an isbn from the set of identifiers for a book.
- `str_in_list(val, separator, string, found_val, not_found_val)` – treat val as a list of items separated by separator, comparing the string against each value in the list. If the string matches a value, return found_val, otherwise return not_found_val. If the string contains separators, then it is also treated as a list and each value is checked.
- `subitems(val, start_index, end_index)` – This function is used to break apart lists of tag-like hierarchical items such as genres. It interprets the value as a comma-separated list of tag-like items, where each item is a period-separated list. Returns a new list made by first finding all the period-separated tag-like items, then for each such item extracting the components from *start_index* to *end_index*, then combining the results back together. The first component in a period-separated list has an index of zero. If an index is negative, then it counts from the end of the list. As a special case, an end_index of zero is assumed to be the length of the list. Examples:

```
Assuming a #genre column containing "A.B.C":
    {#genre:subitems(0,1)} returns "A"
    {#genre:subitems(0,2)} returns "A.B"
    {#genre:subitems(1,0)} returns "B.C"
Assuming a #genre column containing "A.B.C, D.E":
```

```
{#genre:subitems(0,1)} returns "A, D"
{#genre:subitems(0,2)} returns "A.B, D.E"
```

- `sublist(val, start_index, end_index, separator)` – interpret the value as a list of items separated by *separator*, returning a new list made from the items from *start_index'to 'end_index*. The first item is number zero. If an index is negative, then it counts from the end of the list. As a special case, an end_index of zero is assumed to be the length of the list. Examples assuming that the tags column (which is comma-separated) contains "A, B ,C":

  ```
  {tags:sublist(0,1,\,)} returns "A"
  {tags:sublist(-1,0,\,)} returns "C"
  {tags:sublist(0,-1,\,)} returns "A, B"
  ```

- `test(text if not empty, text if empty)` – return *text if not empty* if the field is not empty, otherwise return *text if empty*.

Now, what about using functions and formatting in the same field. Suppose you have an integer custom column called `#myint` that you want to see with leading zeros, as in `003`. To do this, you would use a format of `0>3s`. However, by default, if a number (integer or float) equals zero then the field produces the empty value, so zero values will produce nothing, not `000`. If you really want to see `000` values, then you use both the format string and the `ifempty` function to change the empty value back to a zero. The field reference would be:

```
{#myint:0>3s:ifempty(0)}
```

Note that you can use the prefix and suffix as well. If you want the number to appear as `[003]` or `[000]`, then use the field:

```
{#myint:0>3s:ifempty(0)|[|]}
```

Using functions in templates - template program mode

The template language program mode differs from single-function mode in that it permits you to write template expressions that refer to other metadata fields, modify values, and do arithmetic. It is a reasonably complete programming language.

You can use the functions documented above in template program mode. See below for details.

Beginning with an example, assume that you want your template to show the series for a book if it has one, otherwise show the value of a custom field #genre. You cannot do this in the basic language because you cannot make reference to another metadata field within a template expression. In program mode, you can. The following expression works:

```
{#series:'ifempty($, field('#genre'))'}
```

The example shows several things:

- program mode is used if the expression begins with `:'` and ends with `'`. Anything else is assumed to be single-function.
- the variable `$` stands for the field the expression is operating upon, `#series` in this case.
- functions must be given all their arguments. There is no default value. For example, the standard built-in functions must be given an additional initial parameter indicating the source field, which is a significant difference from single-function mode.
- white space is ignored and can be used anywhere within the expression.
- constant strings are enclosed in matching quotes, either `'` or `"`.

The language is similar to `functional` languages in that it is built almost entirely from functions. A statement is a function. An expression is a function. Constants and identifiers can be thought of as functions returning the value indicated by the constant or stored in the identifier.

The syntax of the language is shown by the following grammar:

```
constant   ::= " string " | ' string ' | number
identifier ::= sequence of letters or ``_`` characters
function   ::= identifier ( statement [ , statement ]* )
expression ::= identifier | constant | function | assignment
assignment ::= identifier '=' expression
statement  ::= expression [ ; expression ]*
program    ::= statement
```

Comments are lines with a '#' character at the beginning of the line.

An `expression` always has a value, either the value of the constant, the value contained in the identifier, or the value returned by a function. The value of a `statement` is the value of the last expression in the sequence of statements. As such, the value of the program (statement):

```
1; 2; 'foobar'; 3
```

is 3.

Another example of a complex but rather silly program might help make things clearer:

```
{series_index:'
    substr(
        strcat($, '->',
            cmp(divide($, 2), 1,
                assign(c, 1); substr('lt123', c, 0),
                'eq', 'gt')),
        0, 6)
   '| prefix | suffix}
```

This program does the following:

- specify that the field being looked at is series_index. This sets the value of the variable `$`.
- calls the `substr` function, which takes 3 parameters `(str, start, end)`. It returns a string formed by extracting the start through end characters from string, zero-based (the first character is character zero). In this case the string will be computed by the `strcat` function, the start is 0, and the end is 6. In this case it will return the first 6 characters of the string returned by `strcat`, which must be evaluated before substr can return.
- calls the `strcat` (string concatenation) function. Strcat accepts 1 or more arguments, and returns a string formed by concatenating all the values. In this case there are three arguments. The first parameter is the value in `$`, which here is the value of `series_index`. The second paremeter is the constant string `'->'`. The third parameter is the value returned by the `cmp` function, which must be fully evaluated before `strcat` can return.
- The `cmp` function takes 5 arguments `(x, y, lt, eq, gt)`. It compares x and y and returns the third argument `lt` if x < y, the fourth argument `eq` if x == y, and the fifth argument `gt` if x > y. As with all functions, all of the parameters can be statements. In this case the first parameter (the value for `x`) is the result of dividing the series_index by 2. The second parameter `y` is the constant `1`. The third parameter `lt` is a statement (more later). The fourth parameter `eq` is the constant string `'eq'`. The fifth parameter is the constant string `'gt'`.
- The third parameter (the one for `lt`) is a statement, or a sequence of expressions. Remember that a statement (a sequence of semicolon-separated expressions) is also an expression, returning the value of the last expression in the list. In this case, the program first assigns the value `1` to a local variable `c`, then returns a substring made by extracting the c'th character to the end. Since c always contains the constant `1`, the substring will return the second through end'th characters, or `'t123'`.

- Once the statement providing the value to the third parameter is executed, `cmp` can return a value. At that point, `strcat` can return a value, then ``substr`` can return a value. The program then terminates.

For various values of series_index, the program returns:

- series_index == undefined, result = `prefix ->t123 suffix`
- series_index == 0.5, result = `prefix 0.50-> suffix`
- series_index == 1, result = `prefix 1->t12 suffix`
- series_index == 2, result = `prefix 2->eq suffix`
- series_index == 3, result = `prefix 3->gt suffix`

All the functions listed under single-function mode can be used in program mode. To do so, you must supply the value that the function is to act upon as the first parameter, in addition to the parameters documented above. For example, in program mode the parameters of the *test* function are `test(x, text_if_not_empty, text_if_empty)`. The *x* parameter, which is the value to be tested, will almost always be a variable or a function call, often *field()*.

The following functions are available in addition to those described in single-function mode. Remember from the example above that the single-function mode functions require an additional first parameter specifying the field to operate on. With the exception of the `id` parameter of assign, all parameters can be statements (sequences of expressions). Note that the definitive documentation for functions is available in the section *Function classification* (page 385):

- `and(value, value, ...)` – returns the string "1" if all values are not empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.
- `add(x, y)` – returns x + y. Throws an exception if either x or y are not numbers.
- `assign(id, val)` – assigns val to id, then returns val. id must be an identifier, not an expression
- `approximate_formats()` – return a comma-separated list of formats that at one point were associated with the book. There is no guarantee that the list is correct, although it probably is. This function can be called in template program mode using the template `{:'approximate_formats()'}`. Note that format names are always uppercase, as in EPUB.
- `booksize()` – returns the value of the calibre 'size' field. Returns '' if there are no formats.
- `cmp(x, y, lt, eq, gt)` – compares x and y after converting both to numbers. Returns `lt` if x < y. Returns `eq` if x == y. Otherwise returns `gt`.
- `current_library_name() -- `` return the last name on the path to the current calibre library. This function can be called in template program mode using the template ``{:'current_library_name()'}`.
- `current_library_path() -- `` eturn the path to the current calibre library. This function can be called in template program mode using the template ``{:'current_library_path()'}`..
- `days_between(date1, date2)` – return the number of days between `date1` and `date2`. The number is positive if `date1` is greater than `date2`, otherwise negative. If either `date1` or `date2` are not dates, the function returns the empty string.
- `divide(x, y)` – returns x / y. Throws an exception if either x or y are not numbers.
- `eval(string)` – evaluates the string as a program, passing the local variables (those `assign`ed to). This permits using the template processor to construct complex results from local variables. Because the *{* and *}* characters are special, you must use *[[* for the *{* character and *]]* for the '}'

character; they are converted automatically. Note also that prefixes and suffixes (the *|prefix|suffix* syntax) cannot be used in the argument to this function when using template program mode.

- `field(name)` – returns the metadata field named by `name`.
- `first_non_empty(value, value, ...)` – returns the first value that is not empty. If all values are empty, then the empty value is returned. You can have as many values as you want.
- `format_date(x, date_format)` – format_date(val, format_string) – format the value, which must be a date field, using the format_string, returning a string. The formatting codes are:

```
d    : the day as number without a leading zero (1 to 31)
dd   : the day as number with a leading zero (01 to 31)
ddd  : the abbreviated localized day name (e.g. "Mon" to "Sun").
dddd : the long localized day name (e.g. "Monday" to "Sunday").
M    : the month as number without a leading zero (1 to 12).
MM   : the month as number with a leading zero (01 to 12)
MMM  : the abbreviated localized month name (e.g. "Jan" to "Dec").
MMMM : the long localized month name (e.g. "January" to "December").
yy   : the year as two digit number (00 to 99).
yyyy : the year as four digit number.
h    : the hours without a leading 0 (0 to 11 or 0 to 23, depending on am/pm)
hh   : the hours with a leading 0 (00 to 11 or 00 to 23, depending on am/pm)
m    : the minutes without a leading 0 (0 to 59)
mm   : the minutes with a leading 0 (00 to 59)
s    : the seconds without a leading 0 (0 to 59)
ss   : the seconds with a leading 0 (00 to 59)
ap   : use a 12-hour clock instead of a 24-hour clock, with 'ap' replaced by the localized s
AP   : use a 12-hour clock instead of a 24-hour clock, with 'AP' replaced by the localized s
iso  : the date with time and timezone. Must be the only format present.
```

You might get unexpected results if the date you are formatting contains localized month names, which can happen if you changed the format tweaks to contain MMMM. In this case, instead of using something like `{pubdate:format_date(yyyy)}`, write the template using template program mode as in `{:'format_date(raw_field('pubdate'),'yyyy')'}`.

- finish_formatting(val, fmt, prefix, suffix) – apply the format, prefix, and suffix to a value in the same way as done in a template like `{series_index:05.2f| - |- }`. This function is provided to ease conversion of complex single-function- or template-program-mode templates to *general program mode* (page 398) (see below) to take advantage of GPM template compilation. For example, the following program produces the same output as the above template:

```
program: finish_formatting(field("series_index"), "05.2f", " - ", " - ")
```

Another example: for the template `{series:re(([^\s])[^\s]+(\s|$),\1)}{series_index:0>2s| - | - }{title}` use:

```
program:
    strcat(
        re(field('series'), '([^\s])[^\s]+(\s|$)', '\1'),
        finish_formatting(field('series_index'), '0>2s', ' - ', ' - '),
        field('title')
    )
```

- `formats_modtimes(date_format)` – return a comma-separated list of colon_separated items representing modification times for the formats of a book. The date_format parameter specifies how the date is to be formatted. See the date_format function for details. You can use the select function to get the mod time for a specific format. Note that format names are always uppercase, as in EPUB.

- `formats_paths()` – return a comma-separated list of colon_separated items representing full path to the formats of a book. You can use the select function to get the path for a specific format. Note that format names are always uppercase, as in EPUB.
- `formats_sizes()` – return a comma-separated list of colon_separated items representing sizes in bytes of the formats of a book. You can use the select function to get the size for a specific format. Note that format names are always uppercase, as in EPUB.
- `has_cover()` – return `Yes` if the book has a cover, otherwise return the empty string
- `not(value)` – returns the string "1" if the value is empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.
- `list_difference(list1, list2, separator)` – return a list made by removing from *list1* any item found in *list2*, using a case-insensitive compare. The items in *list1* and *list2* are separated by separator, as are the items in the returned list.
- `list_equals(list1, sep1, list2, sep2, yes_val, no_val)` – return *yes_val* if *list1* and *list2* contain the same items, otherwise return *no_val*. The items are determined by splitting each list using the appropriate separator character (*sep1* or *sep2*). The order of items in the lists is not relevant. The compare is case insensitive.
- `list_intersection(list1, list2, separator)` – return a list made by removing from *list1* any item not found in *list2*, using a case-insensitive compare. The items in *list1* and *list2* are separated by separator, as are the items in the returned list.
- `list_re(src_list, separator, search_re, opt_replace)` – Construct a list by first separating *src_list* into items using the *separator* character. For each item in the list, check if it matches *search_re*. If it does, then add it to the list to be returned. If *opt_replace* is not the empty string, then apply the replacement before adding the item to the returned list.
- `list_sort(list, direction, separator)` – return list sorted using a case-insensitive sort. If *direction* is zero, the list is sorted ascending, otherwise descending. The list items are separated by separator, as are the items in the returned list.
- `list_union(list1, list2, separator)` – return a list made by merging the items in list1 and list2, removing duplicate items using a case-insensitive compare. If items differ in case, the one in list1 is used. The items in list1 and list2 are separated by separator, as are the items in the returned list.
- `multiply(x, y)` – returns x * y. Throws an exception if either x or y are not numbers.
- `ondevice()` – return the string "Yes" if ondevice is set, otherwise return the empty string
- `or(value, value, ...)` – returns the string "1" if any value is not empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.
- `print(a, b, ...)` – prints the arguments to standard output. Unless you start calibre from the command line (`calibre-debug -g`), the output will go to a black hole.
- `raw_field(name)` – returns the metadata field named by name without applying any formatting.
- `series_sort()` – returns the series sort value.
- `strcat(a, b, ...)` – can take any number of arguments. Returns a string formed by concatenating all the arguments.
- `strcat_max(max, string1, prefix2, string2, ...)` – Returns a string formed by concatenating the arguments. The returned value is initialized to string1. *Prefix, string* pairs are added to the end of the value as long as the resulting string length is less than *max*. String1 is returned even if string1 is longer than max. You can pass as many *prefix, string* pairs as you wish.

- `strcmp(x, y, lt, eq, gt)` – does a case-insensitive comparison x and y as strings. Returns `lt` if x < y. Returns `eq` if x == y. Otherwise returns `gt`.
- `strlen(a)` – Returns the length of the string passed as the argument.
- `substr(str, start, end)` – returns the `start`'th through the `end`'th characters of `str`. The first character in `str` is the zero'th character. If end is negative, then it indicates that many characters counting from the right. If end is zero, then it indicates the last character. For example, `substr('12345', 1, 0)` returns `'2345'`, and `substr('12345', 1, -1)` returns `'234'`.
- `subtract(x, y)` – returns x - y. Throws an exception if either x or y are not numbers.
- `today()` – return a date string for today. This value is designed for use in format_date or days_between, but can be manipulated like any other string. The date is in ISO format.
- `template(x)` – evaluates x as a template. The evaluation is done in its own context, meaning that variables are not shared between the caller and the template evaluation. Because the *{* and *}* characters are special, you must use *[[* for the *{* character and *]]* for the '}' character; they are converted automatically. For example, `template('[[title_sort]]') will evaluate the template ``{title_sort}` and return its value. Note also that prefixes and suffixes (the *|prefix|suffix* syntax) cannot be used in the argument to this function when using template program mode.

Function classification

Reference for all built-in template language functions

Here, we document all the built-in functions available in the calibre template language. Every function is implemented as a class in python and you can click the source links to see the source code, in case the documentation is insufficient. The functions are arranged in logical groups by type.

- Arithmetic (page 404)
  - add(x, y) (page 404)
  - divide(x, y) (page 404)
  - multiply(x, y) (page 404)
  - subtract(x, y) (page 404)
- Boolean (page 404)
  - and(value, value, ...) (page 404)
  - not(value) (page 404)
  - or(value, value, ...) (page 404)
- Date functions (page 404)
  - days_between(date1, date2) (page 404)
  - today() (page 404)
- Formatting values (page 405)
  - finish_formatting(val, fmt, prefix, suffix) (page 405)
  - format_date(val, format_string) (page 405)
  - format_number(v, template) (page 405)
  - human_readable(v) (page 405)
- Get values from metadata (page 405)
  - approximate_formats() (page 405)
  - booksize() (page 405)
  - current_library_name() (page 406)
  - current_library_path() (page 406)
  - field(name) (page 406)
  - formats_modtimes(date_format) (page 406)
  - formats_paths() (page 406)
  - formats_sizes() (page 406)
  - has_cover() (page 406)
  - language_codes(lang_strings) (page 406)
  - language_strings(lang_codes, localize) (page 406)
  - ondevice() (page 407)
  - raw_field(name) (page 407)
  - series_sort() (page 407)
- If-then-else (page 407)
  - contains(val, pattern, text if match, text if not match) (page 407)
  - ifempty(val, text if empty) (page 407)
  - test(val, text if not empty, text if empty) (page 407)
- Iterating over values (page 407)
  - first_non_empty(value, value, ...) (page 407)
  - lookup(val, pattern, field, pattern, field, ..., else_field) (page 407)
  - switch(val, pattern, value, pattern, value, ..., else_value) (page 407)
- List lookup (page 408)
  - identifier_in_list(val, id, found_val, not_found_val) (page 408)
  - in_list(val, separator, pattern, found_val, not_found_val) (page 408)
  - list_item(val, index, separator) (page 408)
  - select(val, key) (page 408)
  - str_in_list(val, separator, string, found_val, not_found_val) (page 408)
- List manipulation (page 408)
  - count(val, separator) (page 408)
  - list_difference(list1, list2, separator) (page 408)
  - list_equals(list1, sep1, list2, sep2, yes_val, no_val) (page 409)
  - list_intersection(list1, list2, separator) (page 409)
  - list_re(src_list, separator, search_re, opt_replace) (page 409)
  - list_sort(list, direction, separator) (page 409)
  - list_union(list1, list2, separator) (page 409)
  - subitems(val, start_index, end_index) (page 409)
  - sublist(val, start_index, end_index, separator) (page 409)
- Other (page 410)
  - assign(id, val) (page 410)
  - print(a, b, ...) (page 410)
- Recursion (page 410)

Arithmetic

add(x, y)
class calibre.utils.formatter_functions.**BuiltinAdd**
add(x, y) – returns x + y. Throws an exception if either x or y are not numbers.

divide(x, y)
class calibre.utils.formatter_functions.**BuiltinDivide**
divide(x, y) – returns x / y. Throws an exception if either x or y are not numbers.

multiply(x, y)
class calibre.utils.formatter_functions.**BuiltinMultiply**
multiply(x, y) – returns x * y. Throws an exception if either x or y are not numbers.

subtract(x, y)
class calibre.utils.formatter_functions.**BuiltinSubtract**
subtract(x, y) – returns x - y. Throws an exception if either x or y are not numbers.

Boolean

and(value, value, ...)
class calibre.utils.formatter_functions.**BuiltinAnd**
and(value, value, ...) – returns the string "1" if all values are not empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.

not(value)
class calibre.utils.formatter_functions.**BuiltinNot**
not(value) – returns the string "1" if the value is empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.

or(value, value, ...)
class calibre.utils.formatter_functions.**BuiltinOr**
or(value, value, ...) – returns the string "1" if any value is not empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.

Date functions

days_between(date1, date2)
class calibre.utils.formatter_functions.**BuiltinDaysBetween**
days_between(date1, date2) – return the number of days between date1 and date2. The number is positive if date1 is greater than date2, otherwise negative. If either date1 or date2 are not dates, the function returns the empty string.

today()
class calibre.utils.formatter_functions.**BuiltinToday**
today() – return a date string for today. This value is designed for use in format_date or days_between, but can be manipulated like any other string. The date is in ISO format.

Formatting values

finish_formatting(val, fmt, prefix, suffix)

class calibre.utils.formatter_functions.**BuiltinFinishFormatting**

finish_formatting(val, fmt, prefix, suffix) – apply the format, prefix, and suffix to a value in the same way as done in a template like *{series_index:05.2f| - |- }*. For example, the following program produces the same output as the above template: program: finish_formatting(field("series_index"), "05.2f", " - ", " - ")

format_date(val, format_string)

class calibre.utils.formatter_functions.**BuiltinFormatDate**

format_date(val, format_string) – format the value, which must be a date, using the format_string, returning a string. The formatting codes are: d : the day as number without a leading zero (1 to 31) dd : the day as number with a leading zero (01 to 31) ddd : the abbreviated localized day name (e.g. "Mon" to "Sun"). dddd : the long localized day name (e.g. "Monday" to "Sunday"). M : the month as number without a leading zero (1 to 12). MM : the month as number with a leading zero (01 to 12) MMM : the abbreviated localized month name (e.g. "Jan" to "Dec"). MMMM : the long localized month name (e.g. "January" to "December"). yy : the year as two digit number (00 to 99). yyyy : the year as four digit number. h : the hours without a leading 0 (0 to 11 or 0 to 23, depending on am/pm) hh : the hours with a leading 0 (00 to 11 or 00 to 23, depending on am/pm) m : the minutes without a leading 0 (0 to 59) mm : the minutes with a leading 0 (00 to 59) s : the seconds without a leading 0 (0 to 59) ss : the seconds with a leading 0 (00 to 59) ap : use a 12-hour clock instead of a 24-hour clock, with "ap" replaced by the localized string for am or pm AP : use a 12-hour clock instead of a 24-hour clock, with "AP" replaced by the localized string for AM or PM iso : the date with time and timezone. Must be the only format present

format_number(v, template)

class calibre.utils.formatter_functions.**BuiltinFormatNumber**

format_number(v, template) – format the number v using a python formatting template such as "{0:5.2f}" or "{0:,d}" or "${0:5,.2f}". The field_name part of the template must be a 0 (zero) (the "{0:" in the above examples). See the template language and python documentation for more examples. Returns the empty string if formatting fails.

human_readable(v)

class calibre.utils.formatter_functions.**BuiltinHumanReadable**

human_readable(v) – return a string representing the number v in KB, MB, GB, etc.

Get values from metadata

approximate_formats()

class calibre.utils.formatter_functions.**BuiltinApproximateFormats**

approximate_formats() – return a comma-separated list of formats that at one point were associated with the book. There is no guarantee that this list is correct, although it probably is. This function can be called in template program mode using the template "{:'approximate_formats()'}". Note that format names are always uppercase, as in EPUB.

booksize()

class calibre.utils.formatter_functions.**BuiltinBooksize**

booksize() – return value of the size field

current_library_name()

class calibre.utils.formatter_functions.**BuiltinCurrentLibraryName**

current_library_name() – return the last name on the path to the current calibre library. This function can be called in template program mode using the template "{:'current_library_name()'}".

current_library_path()

class calibre.utils.formatter_functions.**BuiltinCurrentLibraryPath**

current_library_path() – return the path to the current calibre library. This function can be called in template program mode using the template "{:'current_library_path()'}".

field(name)

class calibre.utils.formatter_functions.**BuiltinField**

field(name) – returns the metadata field named by name

formats_modtimes(date_format)

class calibre.utils.formatter_functions.**BuiltinFormatsModtimes**

formats_modtimes(date_format) – return a comma-separated list of colon_separated items representing modification times for the formats of a book. The date_format parameter specifies how the date is to be formatted. See the date_format function for details. You can use the select function to get the mod time for a specific format. Note that format names are always uppercase, as in EPUB.

formats_paths()

class calibre.utils.formatter_functions.**BuiltinFormatsPaths**

formats_paths() – return a comma-separated list of colon_separated items representing full path to the formats of a book. You can use the select function to get the path for a specific format. Note that format names are always uppercase, as in EPUB.

formats_sizes()

class calibre.utils.formatter_functions.**BuiltinFormatsSizes**

formats_sizes() – return a comma-separated list of colon_separated items representing sizes in bytes of the formats of a book. You can use the select function to get the size for a specific format. Note that format names are always uppercase, as in EPUB.

has_cover()

class calibre.utils.formatter_functions.**BuiltinHasCover**

has_cover() – return Yes if the book has a cover, otherwise return the empty string

language_codes(lang_strings)

class calibre.utils.formatter_functions.**BuiltinLanguageCodes**

language_codes(lang_strings) – return the language codes for the strings passed in lang_strings. The strings must be in the language of the current locale. Lang_strings is a comma-separated list.

language_strings(lang_codes, localize)

class calibre.utils.formatter_functions.**BuiltinLanguageStrings**

language_strings(lang_codes, localize) – return the strings for the language codes passed in lang_codes. If localize is zero, return the strings in English. If localize is not zero, return the strings in the language of the current locale. Lang_codes is a comma-separated list.

ondevice()
class calibre.utils.formatter_functions.**BuiltinOndevice**
ondevice() – return Yes if ondevice is set, otherwise return the empty string

raw_field(name)
class calibre.utils.formatter_functions.**BuiltinRawField**
raw_field(name) – returns the metadata field named by name without applying any formatting.

series_sort()
class calibre.utils.formatter_functions.**BuiltinSeriesSort**
series_sort() – return the series sort value

If-then-else

contains(val, pattern, text if match, text if not match)
class calibre.utils.formatter_functions.**BuiltinContains**
contains(val, pattern, text if match, text if not match) – checks if field contains matches for the regular expression *pattern*. Returns *text if match* if matches are found, otherwise it returns *text if no match*

ifempty(val, text if empty)
class calibre.utils.formatter_functions.**BuiltinIfempty**
ifempty(val, text if empty) – return val if val is not empty, otherwise return *text if empty*

test(val, text if not empty, text if empty)
class calibre.utils.formatter_functions.**BuiltinTest**
test(val, text if not empty, text if empty) – return *text if not empty* if the field is not empty, otherwise return *text if empty*

Iterating over values

first_non_empty(value, value, ...)
class calibre.utils.formatter_functions.**BuiltinFirstNonEmpty**
first_non_empty(value, value, ...) – returns the first value that is not empty. If all values are empty, then the empty value is returned. You can have as many values as you want.

lookup(val, pattern, field, pattern, field, ..., else_field)
class calibre.utils.formatter_functions.**BuiltinLookup**
lookup(val, pattern, field, pattern, field, ..., else_field) – like switch, except the arguments are field (metadata) names, not text. The value of the appropriate field will be fetched and used. Note that because composite columns are fields, you can use this function in one composite field to use the value of some other composite field. This is extremely useful when constructing variable save paths

switch(val, pattern, value, pattern, value, ..., else_value)
class calibre.utils.formatter_functions.**BuiltinSwitch**
switch(val, pattern, value, pattern, value, ..., else_value) – for each *pattern, value* pair, checks if the field matches the regular expression *pattern* and if so, returns that *value*. If no pattern matches, then else_value is returned. You can have as many *pattern, value* pairs as you want

List lookup

identifier_in_list(val, id, found_val, not_found_val)

class calibre.utils.formatter_functions.**BuiltinIdentifierInList**

identifier_in_list(val, id, found_val, not_found_val) – treat val as a list of identifiers separated by commas, comparing the string against each value in the list. An identifier has the format "identifier:value". The id parameter should be either "id" or "id:regexp". The first case matches if there is any identifier with that id. The second case matches if the regexp matches the identifier's value. If there is a match, return found_val, otherwise return not_found_val.

in_list(val, separator, pattern, found_val, not_found_val)

class calibre.utils.formatter_functions.**BuiltinInList**

in_list(val, separator, pattern, found_val, not_found_val) – treat val as a list of items separated by separator, comparing the pattern against each value in the list. If the pattern matches a value, return found_val, otherwise return not_found_val.

list_item(val, index, separator)

class calibre.utils.formatter_functions.**BuiltinListitem**

list_item(val, index, separator) – interpret the value as a list of items separated by *separator*, returning the *index'th item. The first item is number zero. The last item can be returned using 'list_item(-1,separator)*. If the item is not in the list, then the empty value is returned. The separator has the same meaning as in the count function.

select(val, key)

class calibre.utils.formatter_functions.**BuiltinSelect**

select(val, key) – interpret the value as a comma-separated list of items, with the items being "id:value". Find the pair with the id equal to key, and return the corresponding value.

str_in_list(val, separator, string, found_val, not_found_val)

class calibre.utils.formatter_functions.**BuiltinStrInList**

str_in_list(val, separator, string, found_val, not_found_val) – treat val as a list of items separated by separator, comparing the string against each value in the list. If the string matches a value, return found_val, otherwise return not_found_val. If the string contains separators, then it is also treated as a list and each value is checked.

List manipulation

count(val, separator)

class calibre.utils.formatter_functions.**BuiltinCount**

count(val, separator) – interprets the value as a list of items separated by *separator*, returning the number of items in the list. Most lists use a comma as the separator, but authors uses an ampersand. Examples: {tags:count(,)}, {authors:count(&)}

list_difference(list1, list2, separator)

class calibre.utils.formatter_functions.**BuiltinListDifference**

list_difference(list1, list2, separator) – return a list made by removing from list1 any item found in list2, using a case-insensitive compare. The items in list1 and list2 are separated by separator, as are the items in the returned list.

list_equals(list1, sep1, list2, sep2, yes_val, no_val)

class calibre.utils.formatter_functions.**BuiltinListEquals**

list_equals(list1, sep1, list2, sep2, yes_val, no_val) – return yes_val if list1 and list2 contain the same items, otherwise return no_val. The items are determined by splitting each list using the appropriate separator character (sep1 or sep2). The order of items in the lists is not relevant. The compare is case insensitive.

list_intersection(list1, list2, separator)

class calibre.utils.formatter_functions.**BuiltinListIntersection**

list_intersection(list1, list2, separator) – return a list made by removing from list1 any item not found in list2, using a case-insensitive compare. The items in list1 and list2 are separated by separator, as are the items in the returned list.

list_re(src_list, separator, search_re, opt_replace)

class calibre.utils.formatter_functions.**BuiltinListRe**

list_re(src_list, separator, search_re, opt_replace) – Construct a list by first separating src_list into items using the separator character. For each item in the list, check if it matches search_re. If it does, then add it to the list to be returned. If opt_replace is not the empty string, then apply the replacement before adding the item to the returned list.

list_sort(list, direction, separator)

class calibre.utils.formatter_functions.**BuiltinListSort**

list_sort(list, direction, separator) – return list sorted using a case-insensitive sort. If direction is zero, the list is sorted ascending, otherwise descending. The list items are separated by separator, as are the items in the returned list.

list_union(list1, list2, separator)

class calibre.utils.formatter_functions.**BuiltinListUnion**

list_union(list1, list2, separator) – return a list made by merging the items in list1 and list2, removing duplicate items using a case-insensitive compare. If items differ in case, the one in list1 is used. The items in list1 and list2 are separated by separator, as are the items in the returned list.

subitems(val, start_index, end_index)

class calibre.utils.formatter_functions.**BuiltinSubitems**

subitems(val, start_index, end_index) – This function is used to break apart lists of items such as genres. It interprets the value as a comma-separated list of items, where each item is a period-separated list. Returns a new list made by first finding all the period-separated items, then for each such item extracting the *start_index* to the *end_index* components, then combining the results back together. The first component in a period-separated list has an index of zero. If an index is negative, then it counts from the end of the list. As a special case, an end_index of zero is assumed to be the length of the list. Example using basic template mode and assuming a #genre value of "A.B.C": {#genre:subitems(0,1)} returns "A". {#genre:subitems(0,2)} returns "A.B". {#genre:subitems(1,0)} returns "B.C". Assuming a #genre value of "A.B.C, D.E.F", {#genre:subitems(0,1)} returns "A, D". {#genre:subitems(0,2)} returns "A.B, D.E"

sublist(val, start_index, end_index, separator)

class calibre.utils.formatter_functions.**BuiltinSublist**

sublist(val, start_index, end_index, separator) – interpret the value as a list of items separated by *separator*, returning a new list made from the *start_index* to the *end_index* item. The first item is number zero. If an index is negative, then it counts from the end of the list. As a special case, an end_index of zero is assumed to be the length of the list. Examples using basic template mode and assuming that the tags column

(which is comma-separated) contains "A, B, C": {tags:sublist(0,1,,)} returns "A". {tags:sublist(-1,0,,)} returns "C". {tags:sublist(0,-1,,)} returns "A, B".

Other

assign(id, val)
class calibre.utils.formatter_functions.**BuiltinAssign**
assign(id, val) – assigns val to id, then returns val. id must be an identifier, not an expression

print(a, b, ...)
class calibre.utils.formatter_functions.**BuiltinPrint**
print(a, b, ...) – prints the arguments to standard output. Unless you start calibre from the command line (calibre-debug -g), the output will go to a black hole.

Recursion

eval(template)
class calibre.utils.formatter_functions.**BuiltinEval**
eval(template) – evaluates the template, passing the local variables (those 'assign'ed to) instead of the book metadata. This permits using the template processor to construct complex results from local variables. Because the { and } characters are special, you must use [[for the { character and]] for the } character; they are converted automatically. Note also that prefixes and suffixes (the *|prefix|suffix* syntax) cannot be used in the argument to this function when using template program mode.

template(x)
class calibre.utils.formatter_functions.**BuiltinTemplate**
template(x) – evaluates x as a template. The evaluation is done in its own context, meaning that variables are not shared between the caller and the template evaluation. Because the { and } characters are special, you must use [[for the { character and]] for the } character; they are converted automatically. For example, template('[[title_sort]]') will evaluate the template {title_sort} and return its value. Note also that prefixes and suffixes (the *|prefix|suffix* syntax) cannot be used in the argument to this function when using template program mode.

Relational

cmp(x, y, lt, eq, gt)
class calibre.utils.formatter_functions.**BuiltinCmp**
cmp(x, y, lt, eq, gt) – compares x and y after converting both to numbers. Returns lt if x < y. Returns eq if x == y. Otherwise returns gt.

strcmp(x, y, lt, eq, gt)
class calibre.utils.formatter_functions.**BuiltinStrcmp**
strcmp(x, y, lt, eq, gt) – does a case-insensitive comparison of x and y as strings. Returns lt if x < y. Returns eq if x == y. Otherwise returns gt.

String case changes

capitalize(val)
class calibre.utils.formatter_functions.**BuiltinCapitalize**
capitalize(val) – return value of the field capitalized

lowercase(val)
class calibre.utils.formatter_functions.**BuiltinLowercase**
lowercase(val) – return value of the field in lower case

titlecase(val)
class calibre.utils.formatter_functions.**BuiltinTitlecase**
titlecase(val) – return value of the field in title case

uppercase(val)
class calibre.utils.formatter_functions.**BuiltinUppercase**
uppercase(val) – return value of the field in upper case

String manipulation

re(val, pattern, replacement)
class calibre.utils.formatter_functions.**BuiltinRe**
re(val, pattern, replacement) – return the field after applying the regular expression. All instances of *pattern* are replaced with *replacement*. As in all of calibre, these are python-compatible regular expressions

shorten(val, left chars, middle text, right chars)
class calibre.utils.formatter_functions.**BuiltinShorten**
shorten(val, left chars, middle text, right chars) – Return a shortened version of the field, consisting of *left chars* characters from the beginning of the field, followed by *middle text*, followed by *right chars* characters from the end of the string. *Left chars* and *right chars* must be integers. For example, assume the title of the book is *Ancient English Laws in the Times of Ivanhoe*, and you want it to fit in a space of at most 15 characters. If you use {title:shorten(9,-,5)}, the result will be *Ancient E-nhoe*. If the field's length is less than left chars + right chars + the length of *middle text*, then the field will be used intact. For example, the title *The Dome* would not be changed.

strcat(a, b, ...)
class calibre.utils.formatter_functions.**BuiltinStrcat**
strcat(a, b, ...) – can take any number of arguments. Returns a string formed by concatenating all the arguments

strcat_max(max, string1, prefix2, string2, ...)
class calibre.utils.formatter_functions.**BuiltinStrcatMax**
strcat_max(max, string1, prefix2, string2, ...) – Returns a string formed by concatenating the arguments. The returned value is initialized to string1. *Prefix, string* pairs are added to the end of the value as long as the resulting string length is less than *max*. String1 is returned even if string1 is longer than max. You can pass as many *prefix, string* pairs as you wish.

strlen(a)
class calibre.utils.formatter_functions.**BuiltinStrlen**
strlen(a) – Returns the length of the string passed as the argument

substr(str, start, end)

class calibre.utils.formatter_functions.**BuiltinSubstr**

substr(str, start, end) – returns the start'th through the end'th characters of str. The first character in str is the zero'th character. If end is negative, then it indicates that many characters counting from the right. If end is zero, then it indicates the last character. For example, substr('12345', 1, 0) returns '2345', and substr('12345', 1, -1) returns '234'.

swap_around_comma(val)

class calibre.utils.formatter_functions.**BuiltinSwapAroundComma**

swap_around_comma(val) – given a value of the form "B, A", return "A B". This is most useful for converting names in LN, FN format to FN LN. If there is no comma, the function returns val unchanged

API of the Metadata objects The python implementation of the template functions is passed in a Metadata object. Knowing it's API is useful if you want to define your own template functions.

class calibre.ebooks.metadata.book.base.**Metadata**(*title*, *authors=(u'Unknown',)*, *other=None*, *template_cache=None*)

A class representing all the metadata for a book. The various standard metadata fields are available as attributes of this object. You can also stick arbitrary attributes onto this object.

Metadata from custom columns should be accessed via the get() method, passing in the lookup name for the column, for example: "#mytags".

Use the is_null() (page 412) method to test if a field is null.

This object also has functions to format fields into strings.

The list of standard metadata fields grows with time is in STANDARD_METADATA_FIELDS (page 413).

Please keep the method based API of this class to a minimum. Every method becomes a reserved field name.

is_null(*field*)

Return True if the value of field is null in this object. 'null' means it is unknown or evaluates to False. So a title of _('Unknown') is null or a language of 'und' is null.

Be careful with numeric fields since this will return True for zero as well as None.

Also returns True if the field does not exist.

get_identifiers()

Return a copy of the identifiers dictionary. The dict is small, and the penalty for using a reference where a copy is needed is large. Also, we don't want any manipulations of the returned dict to show up in the book.

set_identifiers(*identifiers*)

Set all identifiers. Note that if you previously set ISBN, calling this method will delete it.

set_identifier(*typ*, *val*)

If val is empty, deletes identifier of type typ

standard_field_keys()

return a list of all possible keys, even if this book doesn't have them

custom_field_keys()

return a list of the custom fields in this book

all_field_keys()

All field keys known by this instance, even if their value is None

metadata_for_field(*key*)

return metadata describing a standard or custom field.

all_non_none_fields()
Return a dictionary containing all non-None metadata fields, including the custom ones.

get_standard_metadata(*field*, *make_copy*)
return field metadata from the field if it is there. Otherwise return None. field is the key name, not the label. Return a copy if requested, just in case the user wants to change values in the dict.

get_all_standard_metadata(*make_copy*)
return a dict containing all the standard field metadata associated with the book.

get_all_user_metadata(*make_copy*)
return a dict containing all the custom field metadata associated with the book.

get_user_metadata(*field*, *make_copy*)
return field metadata from the object if it is there. Otherwise return None. field is the key name, not the label. Return a copy if requested, just in case the user wants to change values in the dict.

set_all_user_metadata(*metadata*)
store custom field metadata into the object. Field is the key name not the label

set_user_metadata(*field*, *metadata*)
store custom field metadata for one column into the object. Field is the key name not the label

template_to_attribute(*other*, *ops*)
Takes a list [(src,dest), (src,dest)], evaluates the template in the context of other, then copies the result to self[dest]. This is on a best-efforts basis. Some assignments can make no sense.

smart_update(*other*, *replace_metadata=False*)
Merge the information in *other* into self. In case of conflicts, the information in *other* takes precedence, unless the information in *other* is NULL.

format_field(*key*, *series_with_index=True*)
Returns the tuple (display_name, formatted_value)

to_html()
A HTML representation of this object.

calibre.ebooks.metadata.book.base.**STANDARD_METADATA_FIELDS**
The set of standard metadata fields.

```
'''
All fields must have a NULL value represented as None for simple types,
an empty list/dictionary for complex types and (None, None) for cover_data
'''

SOCIAL_METADATA_FIELDS = frozenset([
    'tags',             # Ordered list
    'rating',           # A floating point number between 0 and 10
    'comments',         # A simple HTML enabled string
    'series',           # A simple string
    'series_index',     # A floating point number
    # Of the form { scheme1:value1, scheme2:value2}
    # For example: {'isbn':'123456789', 'doi':'xxxx', ... }
    'identifiers',
])

'''
The list of names that convert to identifiers when in get and set.
'''

TOP_LEVEL_IDENTIFIERS = frozenset([
```

```
    'isbn',
])

PUBLICATION_METADATA_FIELDS = frozenset([
    'title',            # title must never be None. Should be _('Unknown')
    # Pseudo field that can be set, but if not set is auto generated
    # from title and languages
    'title_sort',
    'authors',          # Ordered list. Must never be None, can be [_('Unknown')]
    'author_sort_map',  # Map of sort strings for each author
    # Pseudo field that can be set, but if not set is auto generated
    # from authors and languages
    'author_sort',
    'book_producer',
    'timestamp',        # Dates and times must be timezone aware
    'pubdate',
    'last_modified',
    'rights',
    # So far only known publication type is periodical:calibre
    # If None, means book
    'publication_type',
    'uuid',             # A UUID usually of type 4
    'languages',        # ordered list of languages in this publication
    'publisher',        # Simple string, no special semantics
    # Absolute path to image file encoded in filesystem_encoding
    'cover',
    # Of the form (format, data) where format is, for e.g. 'jpeg', 'png', 'gif'...
    'cover_data',
    # Either thumbnail data, or an object with the attribute
    # image_path which is the path to an image file, encoded
    # in filesystem_encoding
    'thumbnail',
    ])

BOOK_STRUCTURE_FIELDS = frozenset([
    # These are used by code, Null values are None.
    'toc', 'spine', 'guide', 'manifest',
    ])

USER_METADATA_FIELDS = frozenset([
    # A dict of dicts similar to field_metadata. Each field description dict
    # also contains a value field with the key #value#.
    'user_metadata',
])

DEVICE_METADATA_FIELDS = frozenset([
    'device_collections',   # Ordered list of strings
    'lpath',                # Unicode, / separated
    'size',                 # In bytes
    'mime',                 # Mimetype of the book file being represented

])

CALIBRE_METADATA_FIELDS = frozenset([
    'application_id',   # An application id, currently set to the db_id.
    'db_id',            # the calibre primary key of the item.
    'formats',          # list of formats (extensions) for this book
    # a dict of user category names, where the value is a list of item names
```

```
    # from the book that are in that category
    'user_categories',
    # a dict of author to an associated hyperlink
    'author_link_map',

    ]
)

ALL_METADATA_FIELDS =       SOCIAL_METADATA_FIELDS.union(
                            PUBLICATION_METADATA_FIELDS).union(
                            BOOK_STRUCTURE_FIELDS).union(
                            USER_METADATA_FIELDS).union(
                            DEVICE_METADATA_FIELDS).union(
                            CALIBRE_METADATA_FIELDS)

# All fields except custom fields
STANDARD_METADATA_FIELDS = SOCIAL_METADATA_FIELDS.union(
                            PUBLICATION_METADATA_FIELDS).union(
                            BOOK_STRUCTURE_FIELDS).union(
                            DEVICE_METADATA_FIELDS).union(
                            CALIBRE_METADATA_FIELDS)

# Metadata fields that smart update must do special processing to copy.
SC_FIELDS_NOT_COPIED =      frozenset(['title', 'title_sort', 'authors',
                                       'author_sort', 'author_sort_map',
                                       'cover_data', 'tags', 'languages',
                                       'identifiers'])

# Metadata fields that smart update should copy only if the source is not None
SC_FIELDS_COPY_NOT_NULL =  frozenset(['lpath', 'size', 'comments', 'thumbnail'])

# Metadata fields that smart update should copy without special handling
SC_COPYABLE_FIELDS =        SOCIAL_METADATA_FIELDS.union(
                            PUBLICATION_METADATA_FIELDS).union(
                            BOOK_STRUCTURE_FIELDS).union(
                            DEVICE_METADATA_FIELDS).union(
                            CALIBRE_METADATA_FIELDS) - \
                            SC_FIELDS_NOT_COPIED.union(
                            SC_FIELDS_COPY_NOT_NULL)

SERIALIZABLE_FIELDS =       SOCIAL_METADATA_FIELDS.union(
                            USER_METADATA_FIELDS).union(
                            PUBLICATION_METADATA_FIELDS).union(
                            CALIBRE_METADATA_FIELDS).union(
                            DEVICE_METADATA_FIELDS) - \
                            frozenset(['device_collections', 'formats',
                                'cover_data'])
                            # these are rebuilt when needed
```

Using general program mode

For more complicated template programs, it is sometimes easier to avoid template syntax (all the *{* and *}* characters), instead writing a more classical-looking program. You can do this in calibre by beginning the template with *program:*. In this case, no template processing is done. The special variable *$* is not set. It is up to your program to produce the correct results.

One advantage of *program:* mode is that the brackets are no longer special. For example, it is not necessary to use *[[*

and *]]* when using the *template()* function. Another advantage is that program mode templates are compiled to Python and can run much faster than templates in the other two modes. Speed improvement depends on the complexity of the templates; the more complicated the template the more the improvement. Compilation is turned off or on using the tweak `compile_gpm_templates` (Compile General Program Mode templates to Python). The main reason to turn off compilation is if a compiled template does not work, in which case please file a bug report.

The following example is a *program:* mode implementation of a recipe on the MobileRead forum: "Put series into the title, using either initials or a shortened form. Strip leading articles from the series name (any)." For example, for the book The Two Towers in the Lord of the Rings series, the recipe gives *LotR [02] The Two Towers*. Using standard templates, the recipe requires three custom columns and a plugboard, as explained in the following:

The solution requires creating three composite columns. The first column is used to remove the leading articles. The second is used to compute the 'shorten' form. The third is to compute the 'initials' form. Once you have these columns, the plugboard selects between them. You can hide any or all of the three columns on the library view.

First column: Name: #stripped_series. Template: {series:re(^(A|The|An)s+,)||}

Second column (the shortened form): Name: #shortened. Template: {#stripped_series:shorten(4,-,4)}

Third column (the initials form): Name: #initials. Template: {#stripped_series:re(([^s])[^s]+(s|$),1)}

Plugboard expression: Template:{#stripped_series:lookup(.s,#initials,.,#shortened,series)}{series_index:0>2.0f|[|] }{title} Destination field: title

This set of fields and plugboard produces: Series: The Lord of the Rings Series index: 2 Title: The Two Towers Output: LotR [02] The Two Towers

Series: Dahak Series index: 1 Title: Mutineers Moon Output: Dahak [01] Mutineers Moon

Series: Berserkers Series Index: 4 Title: Berserker Throne Output: Bers-kers [04] Berserker Throne

Series: Meg Langslow Mysteries Series Index: 3 Title: Revenge of the Wrought-Iron Flamingos Output: MLM [03] Revenge of the Wrought-Iron Flamingos

The following program produces the same results as the original recipe, using only one custom column to hold the results of a program that computes the special title value:

```
Custom column:
Name: #special_title
Template: (the following with all leading spaces removed)
    program:
    #       compute the equivalent of the composite fields and store them in local variables
        stripped = re(field('series'), '^(A|The|An)\s+', '');
        shortened = shorten(stripped, 4, '-' ,4);
        initials = re(stripped, '[^\w]*(\w?)[^\s]+(\s|$)', '\1');

    #       Format the series index. Ends up as empty if there is no series index.
    #       Note that leading and trailing spaces will be removed by the formatter,
    #       so we cannot add them here. We will do that in the strcat below.
    #       Also note that because we are in 'program' mode, we can freely use
    #       curly brackets in strings, something we cannot do in template mode.
        s_index = template('{series_index:0>2.0f}');

    #       print(stripped, shortened, initials, s_index);

    #       Now concatenate all the bits together. The switch picks between
    #       initials and shortened, depending on whether there is a space
    #       in stripped. We then add the brackets around s_index if it is
    #       not empty. Finally, add the title. As this is the last function in
    #       the program, its value will be returned.
        strcat(
            switch( stripped,
```

```
                '.\s', initials,
                '.', shortened,
                field('series')),
            test(s_index, strcat(' [', s_index, '] '), ''),
            field('title'));

Plugboard expression:
Template:{#special_title}
Destination field: title
```

It would be possible to do the above with no custom columns by putting the program into the template box of the plugboard. However, to do so, all comments must be removed because the plugboard text box does not support multi-line editing. It is debatable whether the gain of not having the custom column is worth the vast increase in difficulty caused by the program being one giant line.

User-defined Template Functions

You can add your own functions to the template processor. Such functions are written in python, and can be used in any of the three template programming modes. The functions are added by going to Preferences -> Advanced -> Template Functions. Instructions are shown in that dialog.

Special notes for save/send templates

Special processing is applied when a template is used in a *save to disk* or *send to device* template. The values of the fields are cleaned, replacing characters that are special to file systems with underscores, including slashes. This means that field text cannot be used to create folders. However, slashes are not changed in prefix or suffix strings, so slashes in these strings will cause folders to be created. Because of this, you can create variable-depth folder structure.

For example, assume we want the folder structure *series/series_index - title*, with the caveat that if series does not exist, then the title should be in the top folder. The template to do this is:

```
{series:||/}{series_index:|| - }{title}
```

The slash and the hyphen appear only if series is not empty.

The lookup function lets us do even fancier processing. For example, assume that if a book has a series, then we want the folder structure *series/series index - title.fmt*. If the book does not have a series, then we want the folder structure *genre/author_sort/title.fmt*. If the book has no genre, we want to use 'Unknown'. We want two completely different paths, depending on the value of series.

To accomplish this, we:

1. Create a composite field (call it AA) containing `{series}/{series_index} - {title'}`. If the series is not empty, then this template will produce *series/series_index - title*.
2. Create a composite field (call it BB) containing `{#genre:ifempty(Unknown)}/{author_sort}/{title}`. This template produces *genre/author_sort/title*, where an empty genre is replaced wuth *Unknown*.
3. Set the save template to `{series:lookup(.,AA,BB)}`. This template chooses composite field AA if series is not empty, and composite field BB if series is empty. We therefore have two completely different save paths, depending on whether or not *series* is empty.

Templates and Plugboards

Plugboards are used for changing the metadata written into books during send-to-device and save-to-disk operations. A plugboard permits you to specify a template to provide the data to write into the book's metadata. You can use

plugboards to modify the following fields: authors, author_sort, language, publisher, tags, title, title_sort. This feature helps people who want to use different metadata in books on devices to solve sorting or display issues.

When you create a plugboard, you specify the format and device for which the plugboard is to be used. A special device is provided, save_to_disk, that is used when saving formats (as opposed to sending them to a device). Once you have chosen the format and device, you choose the metadata fields to change, providing templates to supply the new values. These templates are *connected* to their destination fields, hence the name *plugboards*. You can, of course, use composite columns in these templates.

When a plugboard might apply (content server, save to disk, or send to device), calibre searches the defined plugboards to choose the correct one for the given format and device. For example, to find the appropriate plugboard for an EPUB book being sent to an ANDROID device, calibre searches the plugboards using the following search order:

- a plugboard with an exact match on format and device, e.g., `EPUB` and `ANDROID`
- a plugboard with an exact match on format and the special `any device` choice, e.g., `EPUB` and `any device`
- a plugboard with the special `any format` choice and an exact match on device, e.g., `any format` and `ANDROID`
- a plugboard with `any format` and `any device`

The tags and authors fields have special treatment, because both of these fields can hold more than one item. A book can have many tags and many authors. When you specify that one of these two fields is to be changed, the template's result is examined to see if more than one item is there. For tags, the result is cut apart wherever calibre finds a comma. For example, if the template produces the value `Thriller, Horror`, then the result will be two tags, `Thriller` and `Horror`. There is no way to put a comma in the middle of a tag.

The same thing happens for authors, but using a different character for the cut, a *&* (ampersand) instead of a comma. For example, if the template produces the value `Blogs, Joe&Posts, Susan`, then the book will end up with two authors, `Blogs, Joe` and `Posts, Susan`. If the template produces the value `Blogs, Joe;Posts, Susan`, then the book will have one author with a rather strange name.

Plugboards affect the metadata written into the book when it is saved to disk or written to the device. Plugboards do not affect the metadata used by `save to disk` and `send to device` to create the file names. Instead, file names are constructed using the templates entered on the appropriate preferences window.

Helpful Tips

You might find the following tips useful.

- Create a custom composite column to test templates. Once you have the column, you can change its template simply by double-clicking on the column. Hide the column when you are not testing.
- Templates can use other templates by referencing a composite custom column.
- In a plugboard, you can set a field to empty (or whatever is equivalent to empty) by using the special template `{}`. This template will always evaluate to an empty string.
- The technique described above to show numbers even if they have a zero value works with the standard field series_index.

Reference for all built-in template language functions

Here, we document all the built-in functions available in the calibre template language. Every function is implemented as a class in python and you can click the source links to see the source code, in case the documentation is insufficient. The functions are arranged in logical groups by type.

- Arithmetic (page 404)
  - add(x, y) (page 404)
  - divide(x, y) (page 404)
  - multiply(x, y) (page 404)
  - subtract(x, y) (page 404)
- Boolean (page 404)
  - and(value, value, ...) (page 404)
  - not(value) (page 404)
  - or(value, value, ...) (page 404)
- Date functions (page 404)
  - days_between(date1, date2) (page 404)
  - today() (page 404)
- Formatting values (page 405)
  - finish_formatting(val, fmt, prefix, suffix) (page 405)
  - format_date(val, format_string) (page 405)
  - format_number(v, template) (page 405)
  - human_readable(v) (page 405)
- Get values from metadata (page 405)
  - approximate_formats() (page 405)
  - booksize() (page 405)
  - current_library_name() (page 406)
  - current_library_path() (page 406)
  - field(name) (page 406)
  - formats_modtimes(date_format) (page 406)
  - formats_paths() (page 406)
  - formats_sizes() (page 406)
  - has_cover() (page 406)
  - language_codes(lang_strings) (page 406)
  - language_strings(lang_codes, localize) (page 406)
  - ondevice() (page 407)
  - raw_field(name) (page 407)
  - series_sort() (page 407)
- If-then-else (page 407)
  - contains(val, pattern, text if match, text if not match) (page 407)
  - ifempty(val, text if empty) (page 407)
  - test(val, text if not empty, text if empty) (page 407)
- Iterating over values (page 407)
  - first_non_empty(value, value, ...) (page 407)
  - lookup(val, pattern, field, pattern, field, ..., else_field) (page 407)
  - switch(val, pattern, value, pattern, value, ..., else_value) (page 407)
- List lookup (page 408)
  - identifier_in_list(val, id, found_val, not_found_val) (page 408)
  - in_list(val, separator, pattern, found_val, not_found_val) (page 408)
  - list_item(val, index, separator) (page 408)
  - select(val, key) (page 408)
  - str_in_list(val, separator, string, found_val, not_found_val) (page 408)
- List manipulation (page 408)
  - count(val, separator) (page 408)
  - list_difference(list1, list2, separator) (page 408)
  - list_equals(list1, sep1, list2, sep2, yes_val, no_val) (page 409)
  - list_intersection(list1, list2, separator) (page 409)
  - list_re(src_list, separator, search_re, opt_replace) (page 409)
  - list_sort(list, direction, separator) (page 409)
  - list_union(list1, list2, separator) (page 409)
  - subitems(val, start_index, end_index) (page 409)
  - sublist(val, start_index, end_index, separator) (page 409)
- Other (page 410)
  - assign(id, val) (page 410)
  - print(a, b, ...) (page 410)
- Recursion (page 410)

Arithmetic

add(x, y)
class calibre.utils.formatter_functions.**BuiltinAdd**
add(x, y) – returns x + y. Throws an exception if either x or y are not numbers.

divide(x, y)
class calibre.utils.formatter_functions.**BuiltinDivide**
divide(x, y) – returns x / y. Throws an exception if either x or y are not numbers.

multiply(x, y)
class calibre.utils.formatter_functions.**BuiltinMultiply**
multiply(x, y) – returns x * y. Throws an exception if either x or y are not numbers.

subtract(x, y)
class calibre.utils.formatter_functions.**BuiltinSubtract**
subtract(x, y) – returns x - y. Throws an exception if either x or y are not numbers.

Boolean

and(value, value, ...)
class calibre.utils.formatter_functions.**BuiltinAnd**
and(value, value, ...) – returns the string "1" if all values are not empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.

not(value)
class calibre.utils.formatter_functions.**BuiltinNot**
not(value) – returns the string "1" if the value is empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.

or(value, value, ...)
class calibre.utils.formatter_functions.**BuiltinOr**
or(value, value, ...) – returns the string "1" if any value is not empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.

Date functions

days_between(date1, date2)
class calibre.utils.formatter_functions.**BuiltinDaysBetween**
days_between(date1, date2) – return the number of days between date1 and date2. The number is positive if date1 is greater than date2, otherwise negative. If either date1 or date2 are not dates, the function returns the empty string.

today()
class calibre.utils.formatter_functions.**BuiltinToday**
today() – return a date string for today. This value is designed for use in format_date or days_between, but can be manipulated like any other string. The date is in ISO format.

Formatting values

finish_formatting(val, fmt, prefix, suffix)

class calibre.utils.formatter_functions.**BuiltinFinishFormatting**

finish_formatting(val, fmt, prefix, suffix) – apply the format, prefix, and suffix to a value in the same way as done in a template like *{series_index:05.2f| - |- }*. For example, the following program produces the same output as the above template: program: finish_formatting(field("series_index"), "05.2f", " - ", " - ")

format_date(val, format_string)

class calibre.utils.formatter_functions.**BuiltinFormatDate**

format_date(val, format_string) – format the value, which must be a date, using the format_string, returning a string. The formatting codes are: d : the day as number without a leading zero (1 to 31) dd : the day as number with a leading zero (01 to 31) ddd : the abbreviated localized day name (e.g. "Mon" to "Sun"). dddd : the long localized day name (e.g. "Monday" to "Sunday"). M : the month as number without a leading zero (1 to 12). MM : the month as number with a leading zero (01 to 12) MMM : the abbreviated localized month name (e.g. "Jan" to "Dec"). MMMM : the long localized month name (e.g. "January" to "December"). yy : the year as two digit number (00 to 99). yyyy : the year as four digit number. h : the hours without a leading 0 (0 to 11 or 0 to 23, depending on am/pm) hh : the hours with a leading 0 (00 to 11 or 00 to 23, depending on am/pm) m : the minutes without a leading 0 (0 to 59) mm : the minutes with a leading 0 (00 to 59) s : the seconds without a leading 0 (0 to 59) ss : the seconds with a leading 0 (00 to 59) ap : use a 12-hour clock instead of a 24-hour clock, with "ap" replaced by the localized string for am or pm AP : use a 12-hour clock instead of a 24-hour clock, with "AP" replaced by the localized string for AM or PM iso : the date with time and timezone. Must be the only format present

format_number(v, template)

class calibre.utils.formatter_functions.**BuiltinFormatNumber**

format_number(v, template) – format the number v using a python formatting template such as "{0:5.2f}" or "{0:,d}" or "${0:5,.2f}". The field_name part of the template must be a 0 (zero) (the "{0:" in the above examples). See the template language and python documentation for more examples. Returns the empty string if formatting fails.

human_readable(v)

class calibre.utils.formatter_functions.**BuiltinHumanReadable**

human_readable(v) – return a string representing the number v in KB, MB, GB, etc.

Get values from metadata

approximate_formats()

class calibre.utils.formatter_functions.**BuiltinApproximateFormats**

approximate_formats() – return a comma-separated list of formats that at one point were associated with the book. There is no guarantee that this list is correct, although it probably is. This function can be called in template program mode using the template "{:'approximate_formats()'}". Note that format names are always uppercase, as in EPUB.

booksize()

class calibre.utils.formatter_functions.**BuiltinBooksize**

booksize() – return value of the size field

current_library_name()

class calibre.utils.formatter_functions.**BuiltinCurrentLibraryName**

current_library_name() – return the last name on the path to the current calibre library. This function can be called in template program mode using the template "{:'current_library_name()'}".

current_library_path()

class calibre.utils.formatter_functions.**BuiltinCurrentLibraryPath**

current_library_path() – return the path to the current calibre library. This function can be called in template program mode using the template "{:'current_library_path()'}".

field(name)

class calibre.utils.formatter_functions.**BuiltinField**

field(name) – returns the metadata field named by name

formats_modtimes(date_format)

class calibre.utils.formatter_functions.**BuiltinFormatsModtimes**

formats_modtimes(date_format) – return a comma-separated list of colon_separated items representing modification times for the formats of a book. The date_format parameter specifies how the date is to be formatted. See the date_format function for details. You can use the select function to get the mod time for a specific format. Note that format names are always uppercase, as in EPUB.

formats_paths()

class calibre.utils.formatter_functions.**BuiltinFormatsPaths**

formats_paths() – return a comma-separated list of colon_separated items representing full path to the formats of a book. You can use the select function to get the path for a specific format. Note that format names are always uppercase, as in EPUB.

formats_sizes()

class calibre.utils.formatter_functions.**BuiltinFormatsSizes**

formats_sizes() – return a comma-separated list of colon_separated items representing sizes in bytes of the formats of a book. You can use the select function to get the size for a specific format. Note that format names are always uppercase, as in EPUB.

has_cover()

class calibre.utils.formatter_functions.**BuiltinHasCover**

has_cover() – return Yes if the book has a cover, otherwise return the empty string

language_codes(lang_strings)

class calibre.utils.formatter_functions.**BuiltinLanguageCodes**

language_codes(lang_strings) – return the language codes for the strings passed in lang_strings. The strings must be in the language of the current locale. Lang_strings is a comma-separated list.

language_strings(lang_codes, localize)

class calibre.utils.formatter_functions.**BuiltinLanguageStrings**

language_strings(lang_codes, localize) – return the strings for the language codes passed in lang_codes. If localize is zero, return the strings in English. If localize is not zero, return the strings in the language of the current locale. Lang_codes is a comma-separated list.

ondevice()
class calibre.utils.formatter_functions.**BuiltinOndevice**
ondevice() – return Yes if ondevice is set, otherwise return the empty string

raw_field(name)
class calibre.utils.formatter_functions.**BuiltinRawField**
raw_field(name) – returns the metadata field named by name without applying any formatting.

series_sort()
class calibre.utils.formatter_functions.**BuiltinSeriesSort**
series_sort() – return the series sort value

If-then-else

contains(val, pattern, text if match, text if not match)
class calibre.utils.formatter_functions.**BuiltinContains**
contains(val, pattern, text if match, text if not match) – checks if field contains matches for the regular expression *pattern*. Returns *text if match* if matches are found, otherwise it returns *text if no match*

ifempty(val, text if empty)
class calibre.utils.formatter_functions.**BuiltinIfempty**
ifempty(val, text if empty) – return val if val is not empty, otherwise return *text if empty*

test(val, text if not empty, text if empty)
class calibre.utils.formatter_functions.**BuiltinTest**
test(val, text if not empty, text if empty) – return *text if not empty* if the field is not empty, otherwise return *text if empty*

Iterating over values

first_non_empty(value, value, ...)
class calibre.utils.formatter_functions.**BuiltinFirstNonEmpty**
first_non_empty(value, value, ...) – returns the first value that is not empty. If all values are empty, then the empty value is returned. You can have as many values as you want.

lookup(val, pattern, field, pattern, field, ..., else_field)
class calibre.utils.formatter_functions.**BuiltinLookup**
lookup(val, pattern, field, pattern, field, ..., else_field) – like switch, except the arguments are field (metadata) names, not text. The value of the appropriate field will be fetched and used. Note that because composite columns are fields, you can use this function in one composite field to use the value of some other composite field. This is extremely useful when constructing variable save paths

switch(val, pattern, value, pattern, value, ..., else_value)
class calibre.utils.formatter_functions.**BuiltinSwitch**
switch(val, pattern, value, pattern, value, ..., else_value) – for each *pattern, value* pair, checks if the field matches the regular expression *pattern* and if so, returns that *value*. If no pattern matches, then else_value is returned. You can have as many *pattern, value* pairs as you want

List lookup

identifier_in_list(val, id, found_val, not_found_val)

class calibre.utils.formatter_functions.**BuiltinIdentifierInList**

identifier_in_list(val, id, found_val, not_found_val) – treat val as a list of identifiers separated by commas, comparing the string against each value in the list. An identifier has the format "identifier:value". The id parameter should be either "id" or "id:regexp". The first case matches if there is any identifier with that id. The second case matches if the regexp matches the identifier's value. If there is a match, return found_val, otherwise return not_found_val.

in_list(val, separator, pattern, found_val, not_found_val)

class calibre.utils.formatter_functions.**BuiltinInList**

in_list(val, separator, pattern, found_val, not_found_val) – treat val as a list of items separated by separator, comparing the pattern against each value in the list. If the pattern matches a value, return found_val, otherwise return not_found_val.

list_item(val, index, separator)

class calibre.utils.formatter_functions.**BuiltinListitem**

list_item(val, index, separator) – interpret the value as a list of items separated by *separator*, returning the *index'th item. The first item is number zero. The last item can be returned using 'list_item(-1,separator)*. If the item is not in the list, then the empty value is returned. The separator has the same meaning as in the count function.

select(val, key)

class calibre.utils.formatter_functions.**BuiltinSelect**

select(val, key) – interpret the value as a comma-separated list of items, with the items being "id:value". Find the pair with the id equal to key, and return the corresponding value.

str_in_list(val, separator, string, found_val, not_found_val)

class calibre.utils.formatter_functions.**BuiltinStrInList**

str_in_list(val, separator, string, found_val, not_found_val) – treat val as a list of items separated by separator, comparing the string against each value in the list. If the string matches a value, return found_val, otherwise return not_found_val. If the string contains separators, then it is also treated as a list and each value is checked.

List manipulation

count(val, separator)

class calibre.utils.formatter_functions.**BuiltinCount**

count(val, separator) – interprets the value as a list of items separated by *separator*, returning the number of items in the list. Most lists use a comma as the separator, but authors uses an ampersand. Examples: {tags:count(,)}, {authors:count(&)}

list_difference(list1, list2, separator)

class calibre.utils.formatter_functions.**BuiltinListDifference**

list_difference(list1, list2, separator) – return a list made by removing from list1 any item found in list2, using a case-insensitive compare. The items in list1 and list2 are separated by separator, as are the items in the returned list.

list_equals(list1, sep1, list2, sep2, yes_val, no_val)

class calibre.utils.formatter_functions.**BuiltinListEquals**

list_equals(list1, sep1, list2, sep2, yes_val, no_val) – return yes_val if list1 and list2 contain the same items, otherwise return no_val. The items are determined by splitting each list using the appropriate separator character (sep1 or sep2). The order of items in the lists is not relevant. The compare is case insensitive.

list_intersection(list1, list2, separator)

class calibre.utils.formatter_functions.**BuiltinListIntersection**

list_intersection(list1, list2, separator) – return a list made by removing from list1 any item not found in list2, using a case-insensitive compare. The items in list1 and list2 are separated by separator, as are the items in the returned list.

list_re(src_list, separator, search_re, opt_replace)

class calibre.utils.formatter_functions.**BuiltinListRe**

list_re(src_list, separator, search_re, opt_replace) – Construct a list by first separating src_list into items using the separator character. For each item in the list, check if it matches search_re. If it does, then add it to the list to be returned. If opt_replace is not the empty string, then apply the replacement before adding the item to the returned list.

list_sort(list, direction, separator)

class calibre.utils.formatter_functions.**BuiltinListSort**

list_sort(list, direction, separator) – return list sorted using a case-insensitive sort. If direction is zero, the list is sorted ascending, otherwise descending. The list items are separated by separator, as are the items in the returned list.

list_union(list1, list2, separator)

class calibre.utils.formatter_functions.**BuiltinListUnion**

list_union(list1, list2, separator) – return a list made by merging the items in list1 and list2, removing duplicate items using a case-insensitive compare. If items differ in case, the one in list1 is used. The items in list1 and list2 are separated by separator, as are the items in the returned list.

subitems(val, start_index, end_index)

class calibre.utils.formatter_functions.**BuiltinSubitems**

subitems(val, start_index, end_index) – This function is used to break apart lists of items such as genres. It interprets the value as a comma-separated list of items, where each item is a period-separated list. Returns a new list made by first finding all the period-separated items, then for each such item extracting the *start_index* to the *end_index* components, then combining the results back together. The first component in a period-separated list has an index of zero. If an index is negative, then it counts from the end of the list. As a special case, an end_index of zero is assumed to be the length of the list. Example using basic template mode and assuming a #genre value of "A.B.C": {#genre:subitems(0,1)} returns "A". {#genre:subitems(0,2)} returns "A.B". {#genre:subitems(1,0)} returns "B.C". Assuming a #genre value of "A.B.C, D.E.F", {#genre:subitems(0,1)} returns "A, D". {#genre:subitems(0,2)} returns "A.B, D.E"

sublist(val, start_index, end_index, separator)

class calibre.utils.formatter_functions.**BuiltinSublist**

sublist(val, start_index, end_index, separator) – interpret the value as a list of items separated by *separator*, returning a new list made from the *start_index* to the *end_index* item. The first item is number zero. If an index is negative, then it counts from the end of the list. As a special case, an end_index of zero is assumed to be the length of the list. Examples using basic template mode and assuming that the tags column

(which is comma-separated) contains "A, B, C": {tags:sublist(0,1,,)} returns "A". {tags:sublist(-1,0,,)} returns "C". {tags:sublist(0,-1,,)} returns "A, B".

Other

assign(id, val)
class calibre.utils.formatter_functions.**BuiltinAssign**
assign(id, val) – assigns val to id, then returns val. id must be an identifier, not an expression

print(a, b, ...)
class calibre.utils.formatter_functions.**BuiltinPrint**
print(a, b, ...) – prints the arguments to standard output. Unless you start calibre from the command line (calibre-debug -g), the output will go to a black hole.

Recursion

eval(template)
class calibre.utils.formatter_functions.**BuiltinEval**
eval(template) – evaluates the template, passing the local variables (those 'assign'ed to) instead of the book metadata. This permits using the template processor to construct complex results from local variables. Because the { and } characters are special, you must use [[for the { character and]] for the } character; they are converted automatically. Note also that prefixes and suffixes (the *|prefix|suffix* syntax) cannot be used in the argument to this function when using template program mode.

template(x)
class calibre.utils.formatter_functions.**BuiltinTemplate**
template(x) – evaluates x as a template. The evaluation is done in its own context, meaning that variables are not shared between the caller and the template evaluation. Because the { and } characters are special, you must use [[for the { character and]] for the } character; they are converted automatically. For example, template('[[title_sort]]') will evaluate the template {title_sort} and return its value. Note also that prefixes and suffixes (the *|prefix|suffix* syntax) cannot be used in the argument to this function when using template program mode.

Relational

cmp(x, y, lt, eq, gt)
class calibre.utils.formatter_functions.**BuiltinCmp**
cmp(x, y, lt, eq, gt) – compares x and y after converting both to numbers. Returns lt if x < y. Returns eq if x == y. Otherwise returns gt.

strcmp(x, y, lt, eq, gt)
class calibre.utils.formatter_functions.**BuiltinStrcmp**
strcmp(x, y, lt, eq, gt) – does a case-insensitive comparison of x and y as strings. Returns lt if x < y. Returns eq if x == y. Otherwise returns gt.

String case changes

capitalize(val)
class calibre.utils.formatter_functions.**BuiltinCapitalize**
capitalize(val) – return value of the field capitalized

lowercase(val)
class calibre.utils.formatter_functions.**BuiltinLowercase**
lowercase(val) – return value of the field in lower case

titlecase(val)
class calibre.utils.formatter_functions.**BuiltinTitlecase**
titlecase(val) – return value of the field in title case

uppercase(val)
class calibre.utils.formatter_functions.**BuiltinUppercase**
uppercase(val) – return value of the field in upper case

String manipulation

re(val, pattern, replacement)
class calibre.utils.formatter_functions.**BuiltinRe**
re(val, pattern, replacement) – return the field after applying the regular expression. All instances of *pattern* are replaced with *replacement*. As in all of calibre, these are python-compatible regular expressions

shorten(val, left chars, middle text, right chars)
class calibre.utils.formatter_functions.**BuiltinShorten**
shorten(val, left chars, middle text, right chars) – Return a shortened version of the field, consisting of *left chars* characters from the beginning of the field, followed by *middle text*, followed by *right chars* characters from the end of the string. *Left chars* and *right chars* must be integers. For example, assume the title of the book is *Ancient English Laws in the Times of Ivanhoe*, and you want it to fit in a space of at most 15 characters. If you use {title:shorten(9,-,5)}, the result will be *Ancient E-nhoe*. If the field's length is less than left chars + right chars + the length of *middle text*, then the field will be used intact. For example, the title *The Dome* would not be changed.

strcat(a, b, ...)
class calibre.utils.formatter_functions.**BuiltinStrcat**
strcat(a, b, ...) – can take any number of arguments. Returns a string formed by concatenating all the arguments

strcat_max(max, string1, prefix2, string2, ...)
class calibre.utils.formatter_functions.**BuiltinStrcatMax**
strcat_max(max, string1, prefix2, string2, ...) – Returns a string formed by concatenating the arguments. The returned value is initialized to string1. *Prefix, string* pairs are added to the end of the value as long as the resulting string length is less than *max*. String1 is returned even if string1 is longer than max. You can pass as many *prefix, string* pairs as you wish.

strlen(a)
class calibre.utils.formatter_functions.**BuiltinStrlen**
strlen(a) – Returns the length of the string passed as the argument

substr(str, start, end)

class calibre.utils.formatter_functions.**BuiltinSubstr**

substr(str, start, end) – returns the start'th through the end'th characters of str. The first character in str is the zero'th character. If end is negative, then it indicates that many characters counting from the right. If end is zero, then it indicates the last character. For example, substr('12345', 1, 0) returns '2345', and substr('12345', 1, -1) returns '234'.

swap_around_comma(val)

class calibre.utils.formatter_functions.**BuiltinSwapAroundComma**

swap_around_comma(val) – given a value of the form "B, A", return "A B". This is most useful for converting names in LN, FN format to FN LN. If there is no comma, the function returns val unchanged

API of the Metadata objects The python implementation of the template functions is passed in a Metadata object. Knowing it's API is useful if you want to define your own template functions.

class calibre.ebooks.metadata.book.base.**Metadata**(*title*, *authors=(u'Unknown',)*, *other=None*, *template_cache=None*)

A class representing all the metadata for a book. The various standard metadata fields are available as attributes of this object. You can also stick arbitrary attributes onto this object.

Metadata from custom columns should be accessed via the get() method, passing in the lookup name for the column, for example: "#mytags".

Use the is_null() (page 412) method to test if a field is null.

This object also has functions to format fields into strings.

The list of standard metadata fields grows with time is in STANDARD_METADATA_FIELDS (page 413).

Please keep the method based API of this class to a minimum. Every method becomes a reserved field name.

is_null(*field*)

Return True if the value of field is null in this object. 'null' means it is unknown or evaluates to False. So a title of _('Unknown') is null or a language of 'und' is null.

Be careful with numeric fields since this will return True for zero as well as None.

Also returns True if the field does not exist.

get_identifiers()

Return a copy of the identifiers dictionary. The dict is small, and the penalty for using a reference where a copy is needed is large. Also, we don't want any manipulations of the returned dict to show up in the book.

set_identifiers(*identifiers*)

Set all identifiers. Note that if you previously set ISBN, calling this method will delete it.

set_identifier(*typ*, *val*)

If val is empty, deletes identifier of type typ

standard_field_keys()

return a list of all possible keys, even if this book doesn't have them

custom_field_keys()

return a list of the custom fields in this book

all_field_keys()

All field keys known by this instance, even if their value is None

metadata_for_field(*key*)

return metadata describing a standard or custom field.

all_non_none_fields()
Return a dictionary containing all non-None metadata fields, including the custom ones.

get_standard_metadata(*field*, *make_copy*)
return field metadata from the field if it is there. Otherwise return None. field is the key name, not the label. Return a copy if requested, just in case the user wants to change values in the dict.

get_all_standard_metadata(*make_copy*)
return a dict containing all the standard field metadata associated with the book.

get_all_user_metadata(*make_copy*)
return a dict containing all the custom field metadata associated with the book.

get_user_metadata(*field*, *make_copy*)
return field metadata from the object if it is there. Otherwise return None. field is the key name, not the label. Return a copy if requested, just in case the user wants to change values in the dict.

set_all_user_metadata(*metadata*)
store custom field metadata into the object. Field is the key name not the label

set_user_metadata(*field*, *metadata*)
store custom field metadata for one column into the object. Field is the key name not the label

template_to_attribute(*other*, *ops*)
Takes a list [(src,dest), (src,dest)], evaluates the template in the context of other, then copies the result to self[dest]. This is on a best-efforts basis. Some assignments can make no sense.

smart_update(*other*, *replace_metadata=False*)
Merge the information in *other* into self. In case of conflicts, the information in *other* takes precedence, unless the information in *other* is NULL.

format_field(*key*, *series_with_index=True*)
Returns the tuple (display_name, formatted_value)

to_html()
A HTML representation of this object.

calibre.ebooks.metadata.book.base.**STANDARD_METADATA_FIELDS**
The set of standard metadata fields.

```
'''
All fields must have a NULL value represented as None for simple types,
an empty list/dictionary for complex types and (None, None) for cover_data
'''

SOCIAL_METADATA_FIELDS = frozenset([
    'tags',             # Ordered list
    'rating',           # A floating point number between 0 and 10
    'comments',         # A simple HTML enabled string
    'series',           # A simple string
    'series_index',     # A floating point number
    # Of the form { scheme1:value1, scheme2:value2}
    # For example: {'isbn':'123456789', 'doi':'xxxx', ... }
    'identifiers',
])

'''
The list of names that convert to identifiers when in get and set.
'''

TOP_LEVEL_IDENTIFIERS = frozenset([
```

```
    'isbn',
])

PUBLICATION_METADATA_FIELDS = frozenset([
    'title',            # title must never be None. Should be _('Unknown')
    # Pseudo field that can be set, but if not set is auto generated
    # from title and languages
    'title_sort',
    'authors',          # Ordered list. Must never be None, can be [_('Unknown')]
    'author_sort_map',  # Map of sort strings for each author
    # Pseudo field that can be set, but if not set is auto generated
    # from authors and languages
    'author_sort',
    'book_producer',
    'timestamp',        # Dates and times must be timezone aware
    'pubdate',
    'last_modified',
    'rights',
    # So far only known publication type is periodical:calibre
    # If None, means book
    'publication_type',
    'uuid',             # A UUID usually of type 4
    'languages',        # ordered list of languages in this publication
    'publisher',        # Simple string, no special semantics
    # Absolute path to image file encoded in filesystem_encoding
    'cover',
    # Of the form (format, data) where format is, for e.g. 'jpeg', 'png', 'gif'...
    'cover_data',
    # Either thumbnail data, or an object with the attribute
    # image_path which is the path to an image file, encoded
    # in filesystem_encoding
    'thumbnail',
    ])

BOOK_STRUCTURE_FIELDS = frozenset([
    # These are used by code, Null values are None.
    'toc', 'spine', 'guide', 'manifest',
    ])

USER_METADATA_FIELDS = frozenset([
    # A dict of dicts similar to field_metadata. Each field description dict
    # also contains a value field with the key #value#.
    'user_metadata',
])

DEVICE_METADATA_FIELDS = frozenset([
    'device_collections',   # Ordered list of strings
    'lpath',                # Unicode, / separated
    'size',                 # In bytes
    'mime',                 # Mimetype of the book file being represented

])

CALIBRE_METADATA_FIELDS = frozenset([
    'application_id',   # An application id, currently set to the db_id.
    'db_id',            # the calibre primary key of the item.
    'formats',          # list of formats (extensions) for this book
    # a dict of user category names, where the value is a list of item names
```

```
    # from the book that are in that category
    'user_categories',
    # a dict of author to an associated hyperlink
    'author_link_map',

    ]
)

ALL_METADATA_FIELDS =       SOCIAL_METADATA_FIELDS.union(
                            PUBLICATION_METADATA_FIELDS).union(
                            BOOK_STRUCTURE_FIELDS).union(
                            USER_METADATA_FIELDS).union(
                            DEVICE_METADATA_FIELDS).union(
                            CALIBRE_METADATA_FIELDS)

# All fields except custom fields
STANDARD_METADATA_FIELDS = SOCIAL_METADATA_FIELDS.union(
                            PUBLICATION_METADATA_FIELDS).union(
                            BOOK_STRUCTURE_FIELDS).union(
                            DEVICE_METADATA_FIELDS).union(
                            CALIBRE_METADATA_FIELDS)

# Metadata fields that smart update must do special processing to copy.
SC_FIELDS_NOT_COPIED =      frozenset(['title', 'title_sort', 'authors',
                                       'author_sort', 'author_sort_map',
                                       'cover_data', 'tags', 'languages',
                                       'identifiers'])

# Metadata fields that smart update should copy only if the source is not None
SC_FIELDS_COPY_NOT_NULL =  frozenset(['lpath', 'size', 'comments', 'thumbnail'])

# Metadata fields that smart update should copy without special handling
SC_COPYABLE_FIELDS =        SOCIAL_METADATA_FIELDS.union(
                            PUBLICATION_METADATA_FIELDS).union(
                            BOOK_STRUCTURE_FIELDS).union(
                            DEVICE_METADATA_FIELDS).union(
                            CALIBRE_METADATA_FIELDS) - \
                            SC_FIELDS_NOT_COPIED.union(
                            SC_FIELDS_COPY_NOT_NULL)

SERIALIZABLE_FIELDS =       SOCIAL_METADATA_FIELDS.union(
                            USER_METADATA_FIELDS).union(
                            PUBLICATION_METADATA_FIELDS).union(
                            CALIBRE_METADATA_FIELDS).union(
                            DEVICE_METADATA_FIELDS) - \
                            frozenset(['device_collections', 'formats',
                                'cover_data'])
                            # these are rebuilt when needed
```

1.7.5 All about using regular expressions in calibre

Regular expressions are features used in many places in calibre to perform sophisticated manipulation of ebook content and metadata. This tutorial is a gentle introduction to getting you started with using regular expressions in calibre.

Contents

- First, a word of warning and a word of courage (page 416)
- Where in calibre can you use regular expressions? (page 416)
- What on earth *is* a regular expression? (page 416)
- Care to explain? (page 417)
- That doesn't sound too bad. What's next? (page 417)
- Hey, neat! This is starting to make sense! (page 417)
- Well, these special characters are very neat and all, but what if I wanted to match a dot or a question mark? (page 418)
- So, what are the most useful sets? (page 418)
- But if I had a few varying strings I wanted to match, things get complicated? (page 418)
- You missed... (page 419)
- In the beginning, you said there was a way to make a regular expression case insensitive? (page 419)
- I think I'm beginning to understand these regular expressions now... how do I use them in calibre? (page 419)
  - Conversions (page 419)
  - Adding books (page 420)
  - Bulk editing metadata (page 420)
- Credits (page 420)

First, a word of warning and a word of courage

This is, inevitably, going to be somewhat technical- after all, regular expressions are a technical tool for doing technical stuff. I'm going to have to use some jargon and concepts that may seem complicated or convoluted. I'm going to try to explain those concepts as clearly as I can, but really can't do without using them at all. That being said, don't be discouraged by any jargon, as I've tried to explain everything new. And while regular expressions themselves may seem like an arcane, black magic (or, to be more prosaic, a random string of mumbo-jumbo letters and signs), I promise that they are not all that complicated. Even those who understand regular expressions really well have trouble reading the more complex ones, but writing them isn't as difficult- you construct the expression step by step. So, take a step and follow me into the rabbit hole.

Where in calibre can you use regular expressions?

There are a few places calibre uses regular expressions. There's the Search & Replace in conversion options, metadata detection from filenames in the import settings and Search & Replace when editing the metadata of books in bulk.

What on earth *is* a regular expression?

A regular expression is a way to describe sets of strings. A single regular expression cat *match* a number of different strings. This is what makes regular expression so powerful – they are a concise way of describing a potentially large number of variations.

Note: I'm using string here in the sense it is used in programming languages: a string of one or more characters, characters including actual characters, numbers, punctuation and so-called whitespace (linebreaks, tabulators etc.). Please note that generally, uppercase and lowercase characters are not considered the same, thus "a" being a different character from "A" and so forth. In calibre, regular expressions are case insensitive in the search bar, but not in the conversion options. There's a way to make every regular expression case insensitive, but we'll discuss that later. It gets complicated because regular expressions allow for variations in the strings it matches, so one expression can match multiple strings, which is why people bother using them at all. More on that in a bit.

Care to explain?

Well, that's why we're here. First, this is the most important concept in regular expressions: *A string by itself is a regular expression that matches itself*. That is to say, if I wanted to match the string `"Hello, World!"` using a regular expression, the regular expression to use would be `Hello, World!`. And yes, it really is that simple. You'll notice, though, that this *only* matches the exact string `"Hello, World!"`, not e.g. `"Hello, wOrld!"` or `"hello, world!"` or any other such variation.

That doesn't sound too bad. What's next?

Next is the beginning of the really good stuff. Remember where I said that regular expressions can match multiple strings? This is were it gets a little more complicated. Say, as a somewhat more practical exercise, the ebook you wanted to convert had a nasty footer counting the pages, like "Page 5 of 423". Obviously the page number would rise from 1 to 423, thus you'd have to match 423 different strings, right? Wrong, actually: regular expressions allow you to define sets of characters that are matched: To define a set, you put all the characters you want to be in the set into square brackets. So, for example, the set `[abc]` would match either the character "a", "b" or "c". *Sets will always only match one of the characters in the set*. They "understand" character ranges, that is, if you wanted to match all the lower case characters, you'd use the set `[a-z]` for lower- and uppercase characters you'd use `[a-zA-Z]` and so on. Got the idea? So, obviously, using the expression `Page [0-9] of 423` you'd be able to match the first 9 pages, thus reducing the expressions needed to three: The second expression `Page [0-9][0-9] of 423` would match all two-digit page numbers, and I'm sure you can guess what the third expression would look like. Yes, go ahead. Write it down.

Hey, neat! This is starting to make sense!

I was hoping you'd say that. But brace yourself, now it gets even better! We just saw that using sets, we could match one of several characters at once. But you can even repeat a character or set, reducing the number of expressions needed to handle the above page number example to one. Yes, ONE! Excited? You should be! It works like this: Some so-called special characters, "+", "?" and "*", *repeat the single element preceding them.* (Element means either a single character, a character set, an escape sequence or a group (we'll learn about those last two later)- in short, any single entity in a regular expression.) These characters are called wildcards or quantifiers. To be more precise, "?" matches *0 or 1* of the preceding element, "*" matches *0 or more* of the preceding element and "+" matches *1 or more* of the preceding element. A few examples: The expression `a?` would match either "" (which is the empty string, not strictly useful in this case) or "a", the expression `a*` would match "", "a", "aa" or any number of a's in a row, and, finally, the expression `a+` would match "a", "aa" or any number of a's in a row (Note: it wouldn't match the empty string!). Same deal for sets: The expression `[0-9]+` would match *every integer number there is*! I know what you're thinking, and you're right: If you use that in the above case of matching page numbers, wouldn't that be the single one expression to match all the page numbers? Yes, the expression `Page [0-9]+ of 423` would match every page number in that book!

Note: A note on these quantifiers: They generally try to match as much text as possible, so be careful when using them. This is called "greedy behaviour"- I'm sure you get why. It gets problematic when you, say, try to match a tag. Consider, for example, the string `"<p class="calibre2">Title here</p>"` and let's say you'd want to match the opening tag (the part between the first pair of angle brackets, a little more on tags later). You'd think that the expression `<p.*>` would match that tag, but actually, it matches the whole string! (The character "." is another special character. It matches anything *except* linebreaks, so, basically, the expression `.*` would match any single line you can think of.) Instead, try using `<p.*?>` which makes the quantifier `"*"` non-greedy. That expression would only match the first opening tag, as intended. There's actually another way to accomplish this: The expression `<p[^>]*>` will match that same opening tag- you'll see why after the next section. Just note that there quite frequently is more than one way to write a regular expression.

Well, these special characters are very neat and all, but what if I wanted to match a dot or a question mark?

You can of course do that: Just put a backslash in front of any special character and it is interpreted as the literal character, without any special meaning. This pair of a backslash followed by a single character is called an escape sequence, and the act of putting a backslash in front of a special character is called escaping that character. An escape sequence is interpreted as a single element. There are of course escape sequences that do more than just escaping special characters, for example `"\t"` means a tabulator. We'll get to some of the escape sequences later. Oh, and by the way, concerning those special characters: Consider any character we discuss in this introduction as having some function to be special and thus needing to be escaped if you want the literal character.

So, what are the most useful sets?

Knew you'd ask. Some useful sets are `[0-9]` matching a single number, `[a-z]` matching a single lowercase letter, `[A-Z]` matching a single uppercase letter, `[a-zA-Z]` matching a single letter and `[a-zA-Z0-9]` matching a single letter or number. You can also use an escape sequence as shorthand:

```
\d is equivalent to [0-9]
\w is equivalent to [a-zA-Z0-9_]
\s is equivalent to any whitespace
```

Note: "Whitespace" is a term for anything that won't be printed. These characters include space, tabulator, line feed, form feed and carriage return.

As a last note on sets, you can also define a set as any character *but* those in the set. You do that by including the character `"^"` as the *very first character in the set*. Thus, `[^a]` would match any character excluding "a". That's called complementing the set. Those escape sequence shorthands we saw earlier can also be complemented: `"\D"` means any non-number character, thus being equivalent to `[^0-9]`. The other shorthands can be complemented by, you guessed it, using the respective uppercase letter instead of the lowercase one. So, going back to the example `<p[^>]*>` from the previous section, now you can see that the character set it's using tries to match any character except for a closing angle bracket.

But if I had a few varying strings I wanted to match, things get complicated?

Fear not, life still is good and easy. Consider this example: The book you're converting has "Title" written on every odd page and "Author" written on every even page. Looks great in print, right? But in ebooks, it's annoying. You can group whole expressions in normal parentheses, and the character `"|"` will let you match *either* the expression to its right *or* the one to its left. Combine those and you're done. Too fast for you? Okay, first off, we group the expressions for odd and even pages, thus getting `(Title)(Author)` as our two needed expressions. Now we make things simpler by using the vertical bar (`"|"` is called the vertical bar character): If you use the expression `(Title|Author)` you'll either get a match for "Title" (on the odd pages) or you'd match "Author" (on the even pages). Well, wasn't that easy?

You can, of course, use the vertical bar without using grouping parentheses, as well. Remember when I said that quantifiers repeat the element preceding them? Well, the vertical bar works a little differently: The expression "Title|Author" will also match either the string "Title" or the string "Author", just as the above example using grouping. *The vertical bar selects between the entire expression preceding and following it*. So, if you wanted to match the strings "Calibre" and "calibre" and wanted to select only between the upper- and lowercase "c", you'd have to use the expression `(c|C)alibre`, where the grouping ensures that only the "c" will be selected. If you were to use `c|Calibre`, you'd get a match on the string "c" or on the string "Calibre", which isn't what we wanted. In short: If in doubt, use grouping together with the vertical bar.

You missed...

... wait just a minute, there's one last, really neat thing you can do with groups. If you have a group that you previously matched, you can use references to that group later in the expression: Groups are numbered starting with 1, and you reference them by escaping the number of the group you want to reference, thus, the fifth group would be referenced as `\5`. So, if you searched for `([^ ]+) \1` in the string "Test Test", you'd match the whole string!

In the beginning, you said there was a way to make a regular expression case insensitive?

Yes, I did, thanks for paying attention and reminding me. You can tell calibre how you want certain things handled by using something called flags. You include flags in your expression by using the special construct `(?flags go here)` where, obviously, you'd replace "flags go here" with the specific flags you want. For ignoring case, the flag is `i`, thus you include `(?i)` in your expression. Thus, `test(?i)` would match "Test", "tEst", "TEst" and any case variation you could think of.

Another useful flag lets the dot match any character at all, *including* the newline, the flag `s`. If you want to use multiple flags in an expression, just put them in the same statement: `(?is)` would ignore case and make the dot match all. It doesn't matter which flag you state first, `(?si)` would be equivalent to the above. By the way, good places for putting flags in your expression would be either the very beginning or the very end. That way, they don't get mixed up with anything else.

I think I'm beginning to understand these regular expressions now... how do I use them in calibre?

Conversions

Let's begin with the conversion settings, which is really neat. In the Search and Replace part, you can input a regexp (short for regular expression) that describes the string that will be replaced during the conversion. The neat part is the wizard. Click on the wizard staff and you get a preview of what calibre "sees" during the conversion process. Scroll down to the string you want to remove, select and copy it, paste it into the regexp field on top of the window. If there are variable parts, like page numbers or so, use sets and quantifiers to cover those, and while you're at it, remember to escape special characters, if there are some. Hit the button labeled *Test* and calibre highlights the parts it would replace were you to use the regexp. Once you're satisfied, hit OK and convert. Be careful if your conversion source has tags like this example:

```
Maybe, but the cops feel like you do, Anita. What's one more dead vampire?
New laws don't change that. </p>
<p class="calibre4"> <b class="calibre2">Generated by ABC Amber LIT Conv
<a href="http://www.processtext.com/abclit.html" class="calibre3">erter,
http://www.processtext.com/abclit.html</a></b></p>
<p class="calibre4"> It had only been two years since Addison v. Clark.
The court case gave us a revised version of what life was
```

(shamelessly ripped out of this thread[79]). You'd have to remove some of the tags as well. In this example, I'd recommend beginning with the tag `<b class="calibre2">`, now you have to end with the corresponding closing tag (opening tags are `<tag>`, closing tags are `</tag>`), which is simply the next `</b>` in this case. (Refer to a good HTML manual or ask in the forum if you are unclear on this point.) The opening tag can be described using `<b.*?>`, the closing tag using `</b>`, thus we could remove everything between those tags using `<b.*?>.*?</b>`. But using this expression would be a bad idea, because it removes everything enclosed by <b>-tags (which, by the way, render the enclosed text in bold print), and it's a fair bet that we'll remove portions of the book in this way. Instead, include the beginning of the enclosed string as well, making the regular expression `<b.*?>\s*Generated\s+by\s+ABC\s+Amber\s+LIT.*?</b>` The `\s` with quantifiers are included here

[79] http://www.mobileread.com/forums/showthread.php?t=75594"

instead of explicitly using the spaces as seen in the string to catch any variations of the string that might occur. Remember to check what calibre will remove to make sure you don't remove any portions you want to keep if you test a new expression. If you only check one occurrence, you might miss a mismatch somewhere else in the text. Also note that should you accidentally remove more or fewer tags than you actually wanted to, calibre tries to repair the damaged code after doing the removal.

Adding books

Another thing you can use regular expressions for is to extract metadata from filenames. You can find this feature in the "Adding books" part of the settings. There's a special feature here: You can use field names for metadata fields, for example `(?P<title>)` would indicate that calibre uses this part of the string as book title. The allowed field names are listed in the windows, together with another nice test field. An example: Say you want to import a whole bunch of files named like `Classical Texts: The Divine Comedy by Dante Alighieri.mobi`. (Obviously, this is already in your library, since we all love classical italian poetry) or `Science Fiction epics: The Foundation Trilogy by Isaac Asimov.epub`. This is obviously a naming scheme that calibre won't extract any meaningful data out of - its standard expression for extracting metadata is `(?P<title>.+) - (?P<author>[^_]+)`. A regular expression that works here would be `[a-zA-Z]+: (?P<title>.+) by (?P<author>.+)`. Please note that, inside the group for the metadata field, you need to use expressions to describe what the field actually matches. And also note that, when using the test field calibre provides, you need to add the file extension to your testing filename, otherwise you won't get any matches at all, despite using a working expression.

Bulk editing metadata

The last part is regular expression search and replace in metadata fields. You can access this by selecting multiple books in the library and using bulk metadata edit. Be very careful when using this last feature, as it can do **Very Bad Things** to your library! Doublecheck that your expressions do what you want them to using the test fields, and only mark the books you really want to change! In the regular expression search mode, you can search in one field, replace the text with something and even write the result into another field. A practical example: Say your library contained the books of Frank Herbert's Dune series, named after the fashion `Dune 1 - Dune`, `Dune 2 - Dune Messiah` and so on. Now you want to get `Dune` into the series field. You can do that by searching for `(.*?) \d+ - .*` in the title field and replacing it with `\1` in the series field. See what I did there? That's a reference to the first group you're replacing the series field with. Now that you have the series all set, you only need to do another search for `.*? -` in the title field and replace it with `""` (an empty string), again in the title field, and your metadata is all neat and tidy. Isn't that great? By the way, instead of replacing the entire field, you can also append or prepend to the field, so, if you *wanted* the book title to be prepended with series info, you could do that as well. As you by now have undoubtedly noticed, there's a checkbox labeled *Case sensitive*, so you won't have to use flags to select behaviour here.

Well, that just about concludes the very short introduction to regular expressions. Hopefully I'll have shown you enough to at least get you started and to enable you to continue learning by yourself- a good starting point would be the Python documentation for regexps[80].

One last word of warning, though: Regexps are powerful, but also really easy to get wrong. calibre provides really great testing possibilities to see if your expressions behave as you expect them to. Use them. Try not to shoot yourself in the foot. (God, I love that expression...) But should you, despite the warning, injure your foot (or any other body parts), try to learn from it.

Credits

Thanks for helping with tips, corrections and such:

[80] http://docs.python.org/library/re.html

- ldolse
- kovidgoyal
- chaley
- dwanthny
- kacir
- Starson17

For more about regexps see The Python User Manual[81].

1.7.6 Integrating the calibre content server into other servers

Here, we will show you how to integrate the calibre content server into another server. The most common reason for this is to make use of SSL or more sophisticated authentication. There are two main techniques: Running the calibre content server as a standalone process and using a reverse proxy to connect it with your main server or running the content server in process in your main server with WSGI. The examples below are all for Apache 2.x on linux, but should be easily adaptable to other platforms.

Contents

- Using a reverse proxy (page 421)
- In process (page 422)

Note: This only applies to calibre releases >= 0.7.25

Using a reverse proxy

A reverse proxy is when your normal server accepts incoming requests and passes them onto the calibre server. It then reads the response from the calibre server and forwards it to the client. This means that you can simply run the calibre server as normal without trying to integrate it closely with your main server, and you can take advantage of whatever authentication systems you main server has in place. This is the simplest approach as it allows you to use the binary calibre install with no external dependencies/system integration requirements. Below, is an example of how to achieve this with Apache as your main server, but it will work with any server that supports Reverse Proxies.

First start the calibre content server as shown below:

```
calibre-server --url-prefix /calibre --port 8080
```

The key parameter here is `--url-prefix /calibre`. This causes the content server to serve all URLs prefixed by calibre. To see this in action, visit `http://localhost:8080/calibre` in your browser. You should see the normal content server website, but now it will run under /calibre.

Now suppose you are using Apache as your main server. First enable the proxy modules in apache, by adding the following to `httpd.conf`:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

[81] http://docs.python.org/library/re.html

The exact technique for enabling the proxy modules will vary depending on your Apache installation. Once you have the proxy modules enabled, add the following rules to httpd.conf (or if you are using virtual hosts to the conf file for the virtual host in question:

```
RewriteEngine on
RewriteRule ^/calibre/(.*) http://localhost:8080/calibre/$1 [proxy]
RewriteRule ^/calibre http://localhost:8080 [proxy]
SetEnv force-proxy-request-1.0 1
SetEnv proxy-nokeepalive 1
```

That's all, you will now be able to access the calibre Content Server under the /calibre URL in your apache server. The above rules pass all requests under /calibre to the calibre server running on port 8080 and thanks to the –url-prefix option above, the calibre server handles them transparently.

Note: If you are willing to devote an entire VirtualHost to the content server, then there is no need to use –url-prefix and RewriteRule, instead just use the ProxyPass directive.

Note: The server engine calibre uses, CherryPy, can have trouble with proxying and KeepAlive requests, so turn them off in Apache, with the SetEnv directives shown above.

In process

The calibre content server can be run directly, in process, inside a host server like Apache using the WSGI framework.

Note: For this to work, all the dependencies needed by calibre must be installed on your system. On linux, this can be achieved fairly easily by installing the distribution provided calibre package (provided it is up to date).

First, we have to create a WSGI *adapter* for the calibre content server. Here is a template you can use for the purpose. Replace the paths as directed in the comments

```
# WSGI script file to run calibre content server as a WSGI app

import sys, os


# You can get the paths referenced here by running
# calibre-debug --paths
# on your server

# The first entry from CALIBRE_PYTHON_PATH
sys.path.insert(0, '/home/kovid/work/calibre/src')

# CALIBRE_RESOURCES_PATH
sys.resources_location = '/home/kovid/work/calibre/resources'

# CALIBRE_EXTENSIONS_PATH
sys.extensions_location = '/home/kovid/work/calibre/src/calibre/plugins'

# Path to directory containing calibre executables
sys.executables_location = '/usr/bin'

# Path to a directory for which the server has read/write permissions
# calibre config will be stored here
```

```
os.environ['CALIBRE_CONFIG_DIRECTORY'] = '/var/www/localhost/calibre-config'

del sys
del os

from calibre.library.server.main import create_wsgi_app
application = create_wsgi_app(
        # The mount point of this WSGI application (i.e. the first argument to
        # the WSGIScriptAlias directive). Set to empty string is mounted at /
        prefix='/calibre',

        # Path to the calibre library to be served
        # The server process must have write permission for all files/dirs
        # in this directory or BAD things will happen
        path_to_library='/home/kovid/documents/demo library'
)

del create_wsgi_app
```

Save this adapter as `calibre-wsgi-adpater.py` somewhere your server will have access to it.

Let's suppose that we want to use WSGI in Apache. First enable WSGI in Apache by adding the following to `httpd.conf`:

```
LoadModule proxy_module modules/mod_wsgi.so
```

The exact technique for enabling the wsgi module will vary depending on your Apache installation. Once you have the proxy modules enabled, add the following rules to httpd.conf (or if you are using virtual hosts to the conf file for the virtual host in question:

```
WSGIScriptAlias /calibre /var/www/localhost/cgi-bin/calibre-wsgi-adapter.py
```

Change the path to `calibre-wsgi-adapter.py` to wherever you saved it previously (make sure Apache has access to it).

That's all, you will now be able to access the calibre Content Server under the /calibre URL in your apache server.

Note: For more help with using mod_wsgi in Apache, see mod_wsgi[82].

1.7.7 Writing your own plugins to extend calibre's functionality

calibre has a very modular design. Almost all functionality in calibre comes in the form of plugins. Plugins are used for conversion, for downloading news (though these are called recipes), for various components of the user interface, to connect to different devices, to process files when adding them to calibre and so on. You can get a complete list of all the built-in plugins in calibre by going to *Preferences->Plugins*.

Here, we will teach you how to create your own plugins to add new features to calibre.

[82] http://code.google.com/p/modwsgi/wiki/WhereToGetHelp

Contents

- Anatomy of a calibre plugin (page 424)
- A User Interface plugin (page 425)
  - __init__.py (page 425)
  - ui.py (page 427)
  - main.py (page 428)
  - Getting resources from the plugin zip file (page 430)
  - Enabling user configuration of your plugin (page 431)
- The plugin API (page 432)
- Debugging plugins (page 433)
- More plugin examples (page 433)
- Sharing your plugins with others (page 433)

Note: This only applies to calibre releases >= 0.8.60

Anatomy of a calibre plugin

A calibre plugin is very simple, it's just a zip file that contains some python code and any other resources like image files needed by the plugin. Without further ado, let's see a basic example.

Suppose you have an installation of calibre that you are using to self publish various e-documents in EPUB and MOBI formats. You would like all files generated by calibre to have their publisher set as "Hello world", here's how to do it. Create a file named `__init__.py` (this is a special name and must always be used for the main file of your plugin) and enter the following Python code into it:

```
import os
from calibre.customize import FileTypePlugin

class HelloWorld(FileTypePlugin):

    name                = 'Hello World Plugin' # Name of the plugin
    description         = 'Set the publisher to Hello World for all new conversions'
    supported_platforms = ['windows', 'osx', 'linux'] # Platforms this plugin will run on
    author              = 'Acme Inc.' # The author of this plugin
    version             = (1, 0, 0)   # The version number of this plugin
    file_types          = set(['epub', 'mobi']) # The file types that this plugin will be applied to
    on_postprocess      = True # Run this plugin after conversion is complete
    minimum_calibre_version = (0, 7, 53)

    def run(self, path_to_ebook):
        from calibre.ebooks.metadata.meta import get_metadata, set_metadata
        file = open(path_to_ebook, 'r+b')
        ext  = os.path.splitext(path_to_ebook)[-1][1:].lower()
        mi = get_metadata(file, ext)
        mi.publisher = 'Hello World'
        set_metadata(file, mi, ext)
        return path_to_ebook
```

That's all. To add this code to calibre as a plugin, simply run the following in the directory in which you created `__init__.py`:

```
calibre-customize -b .
```

Note: On OS X you have to first install the calibre command line tools, by going to *Preferences->Miscellaneous* and clicking the *Install command line tools* button.

You can download the Hello World plugin from helloworld_plugin.zip[83].

Every time you use calibre to convert a book, the plugin's `run()` method will be called and the converted book will have its publisher set to "Hello World". This is a trivial plugin, lets move on to a more complex example that actually adds a component to the user interface.

A User Interface plugin

This plugin will be spread over a few files (to keep the code clean). It will show you how to get resources (images or data files) from the plugin zip file, allow users to configure your plugin, how to create elements in the calibre user interface and how to access and query the books database in calibre.

You can download this plugin from interface_demo_plugin.zip[84]

The first thing to note is that this zip file has a lot more files in it, explained below, pay particular attention to `plugin-import-name-interface_demo.txt`.

plugin-import-name-interface_demo.txt An empty text file used to enable the multi-file plugin magic. This file must be present in all plugins that use more than one .py file. It should be empty and its filename must be of the form: plugin-import-name-**some_name**.txt The presence of this file allows you to import code from the .py files present inside the zip file, using a statement like:

```
from calibre_plugins.some_name.some_module import some_object
```

The prefix `calibre_plugins` must always be present. `some_name` comes from the filename of the empty text file. `some_module` refers to `some_module.py` file inside the zip file. Note that this importing is just as powerful as regular python imports. You can create packages and subpackages of .py modules inside the zip file, just like you would normally (by defining __init__.py in each sub directory), and everything should Just Work.

The name you use for `some_name` enters a global namespace shared by all plugins, **so make it as unique as possible**. But remember that it must be a valid python identifier (only alphabets, numbers and the underscore).

__init__.py As before, the file that defines the plugin class

main.py This file contains the actual code that does something useful

ui.py This file defines the interface part of the plugin

images/icon.png The icon for this plugin

about.txt A text file with information about the plugin

Now let's look at the code.

__init__.py

First, the obligatory `__init__.py` to define the plugin metadata:

```
# The class that all Interface Action plugin wrappers must inherit from
from calibre.customize import InterfaceActionBase
```

[83] http://calibre-ebook.com/downloads/helloworld_plugin.zip
[84] http://calibre-ebook.com/downloads/interface_demo_plugin.zip

```
class InterfacePluginDemo(InterfaceActionBase):
    '''
    This class is a simple wrapper that provides information about the actual
    plugin class. The actual interface plugin class is called InterfacePlugin
    and is defined in the ui.py file, as specified in the actual_plugin field
    below.

    The reason for having two classes is that it allows the command line
    calibre utilities to run without needing to load the GUI libraries.
    '''
    name                = 'Interface Plugin Demo'
    description         = 'An advanced plugin demo'
    supported_platforms = ['windows', 'osx', 'linux']
    author              = 'Kovid Goyal'
    version             = (1, 0, 0)
    minimum_calibre_version = (0, 7, 53)

    #: This field defines the GUI plugin class that contains all the code
    #: that actually does something. Its format is module_path:class_name
    #: The specified class must be defined in the specified module.
    actual_plugin       = 'calibre_plugins.interface_demo.ui:InterfacePlugin'

    def is_customizable(self):
        '''
        This method must return True to enable customization via
        Preferences->Plugins
        '''
        return True

    def config_widget(self):
        '''
        Implement this method and :meth:`save_settings` in your plugin to
        use a custom configuration dialog.

        This method, if implemented, must return a QWidget. The widget can have
        an optional method validate() that takes no arguments and is called
        immediately after the user clicks OK. Changes are applied if and only
        if the method returns True.

        If for some reason you cannot perform the configuration at this time,
        return a tuple of two strings (message, details), these will be
        displayed as a warning dialog to the user and the process will be
        aborted.

        The base class implementation of this method raises NotImplementedError
        so by default no user configuration is possible.
        '''
        # It is important to put this import statement here rather than at the
        # top of the module as importing the config class will also cause the
        # GUI libraries to be loaded, which we do not want when using calibre
        # from the command line
        from calibre_plugins.interface_demo.config import ConfigWidget
        return ConfigWidget()

    def save_settings(self, config_widget):
        '''
        Save the settings specified by the user with config_widget.
```

```
        :param config_widget: The widget returned by :meth:'config_widget'.
        '''
        config_widget.save_settings()

        # Apply the changes
        ac = self.actual_plugin_
        if ac is not None:
            ac.apply_settings()
```

The only noteworthy feature is the field `actual_plugin`. Since calibre has both command line and GUI interfaces, GUI plugins like this one should not load any GUI libraries in __init__.py. The actual_plugin field does this for you, by telling calibre that the actual plugin is to be found in another file inside your zip archive, which will only be loaded in a GUI context.

Remember that for this to work, you must have a plugin-import-name-some_name.txt file in your plugin zip file, as discussed above.

Also there are a couple of methods for enabling user configuration of the plugin. These are discussed below.

ui.py

Now let's look at ui.py which defines the actual GUI plugin. The source code is heavily commented and should be self explanatory:

```
# The class that all interface action plugins must inherit from
from calibre.gui2.actions import InterfaceAction
from calibre_plugins.interface_demo.main import DemoDialog

class InterfacePlugin(InterfaceAction):

    name = 'Interface Plugin Demo'

    # Declare the main action associated with this plugin
    # The keyboard shortcut can be None if you dont want to use a keyboard
    # shortcut. Remember that currently calibre has no central management for
    # keyboard shortcuts, so try to use an unusual/unused shortcut.
    action_spec = ('Interface Plugin Demo', None,
            'Run the Interface Plugin Demo', 'Ctrl+Shift+F1')

    def genesis(self):
        # This method is called once per plugin, do initial setup here

        # Set the icon for this interface action
        # The get_icons function is a builtin function defined for all your
        # plugin code. It loads icons from the plugin zip file. It returns
        # QIcon objects, if you want the actual data, use the analogous
        # get_resources builtin function.
        #
        # Note that if you are loading more than one icon, for performance, you
        # should pass a list of names to get_icons. In this case, get_icons
        # will return a dictionary mapping names to QIcons. Names that
        # are not found in the zip file will result in null QIcons.
        icon = get_icons('images/icon.png')

        # The qaction is automatically created from the action_spec defined
        # above
        self.qaction.setIcon(icon)
```

```
        self.qaction.triggered.connect(self.show_dialog)

    def show_dialog(self):
        # The base plugin object defined in __init__.py
        base_plugin_object = self.interface_action_base_plugin
        # Show the config dialog
        # The config dialog can also be shown from within
        # Preferences->Plugins, which is why the do_user_config
        # method is defined on the base plugin class
        do_user_config = base_plugin_object.do_user_config

        # self.gui is the main calibre GUI. It acts as the gateway to access
        # all the elements of the calibre user interface, it should also be the
        # parent of the dialog
        d = DemoDialog(self.gui, self.qaction.icon(), do_user_config)
        d.show()

    def apply_settings(self):
        from calibre_plugins.interface_demo.config import prefs
        # In an actual non trivial plugin, you would probably need to
        # do something based on the settings in prefs
        prefs
```

main.py

The actual logic to implement the Interface Plugin Demo dialog.

```
from PyQt4.Qt import QDialog, QVBoxLayout, QPushButton, QMessageBox, QLabel

from calibre_plugins.interface_demo.config import prefs

class DemoDialog(QDialog):

    def __init__(self, gui, icon, do_user_config):
        QDialog.__init__(self, gui)
        self.gui = gui
        self.do_user_config = do_user_config

        # The current database shown in the GUI
        # db is an instance of the class LibraryDatabase2 from database.py
        # This class has many, many methods that allow you to do a lot of
        # things.
        self.db = gui.current_db

        self.l = QVBoxLayout()
        self.setLayout(self.l)

        self.label = QLabel(prefs['hello_world_msg'])
        self.l.addWidget(self.label)

        self.setWindowTitle('Interface Plugin Demo')
        self.setWindowIcon(icon)

        self.about_button = QPushButton('About', self)
        self.about_button.clicked.connect(self.about)
        self.l.addWidget(self.about_button)
```

```
        self.marked_button = QPushButton(
            'Show books with only one format in the calibre GUI', self)
        self.marked_button.clicked.connect(self.marked)
        self.l.addWidget(self.marked_button)

        self.view_button = QPushButton(
            'View the most recently added book', self)
        self.view_button.clicked.connect(self.view)
        self.l.addWidget(self.view_button)

        self.update_metadata_button = QPushButton(
            'Update metadata in a book\'s files', self)
        self.update_metadata_button.clicked.connect(self.update_metadata)
        self.l.addWidget(self.update_metadata_button)

        self.conf_button = QPushButton(
                'Configure this plugin', self)
        self.conf_button.clicked.connect(self.config)
        self.l.addWidget(self.conf_button)

        self.resize(self.sizeHint())

    def about(self):
        # Get the about text from a file inside the plugin zip file
        # The get_resources function is a builtin function defined for all your
        # plugin code. It loads files from the plugin zip file. It returns
        # the bytes from the specified file.
        #
        # Note that if you are loading more than one file, for performance, you
        # should pass a list of names to get_resources. In this case,
        # get_resources will return a dictionary mapping names to bytes. Names that
        # are not found in the zip file will not be in the returned dictionary.
        text = get_resources('about.txt')
        QMessageBox.about(self, 'About the Interface Plugin Demo',
                text.decode('utf-8'))

    def marked(self):
        ''' Show books with only one format '''
        fmt_idx = self.db.FIELD_MAP['formats']
        matched_ids = set()
        for record in self.db.data.iterall():
            # Iterate over all records
            fmts = record[fmt_idx]
            # fmts is either None or a comma separated list of formats
            if fmts and ',' not in fmts:
                matched_ids.add(record[0])
        # Mark the records with the matching ids
        self.db.set_marked_ids(matched_ids)

        # Tell the GUI to search for all marked records
        self.gui.search.setEditText('marked:true')
        self.gui.search.do_search()

    def view(self):
        ''' View the most recently added book '''
        most_recent = most_recent_id = None
        timestamp_idx = self.db.FIELD_MAP['timestamp']
```

```
        for record in self.db.data:
            # Iterate over all currently showing records
            timestamp = record[timestamp_idx]
            if most_recent is None or timestamp > most_recent:
                most_recent = timestamp
                most_recent_id = record[0]

        if most_recent_id is not None:
            # Get the row number of the id as shown in the GUI
            row_number = self.db.row(most_recent_id)
            # Get a reference to the View plugin
            view_plugin = self.gui.iactions['View']
            # Ask the view plugin to launch the viewer for row_number
            view_plugin._view_books([row_number])

    def update_metadata(self):
        '''
        Set the metadata in the files in the selected book's record to
        match the current metadata in the database.
        '''
        from calibre.ebooks.metadata.meta import set_metadata
        from calibre.gui2 import error_dialog, info_dialog

        # Get currently selected books
        rows = self.gui.library_view.selectionModel().selectedRows()
        if not rows or len(rows) == 0:
            return error_dialog(self.gui, 'Cannot update metadata',
                                'No books selected', show=True)
        # Map the rows to book ids
        ids = list(map(self.gui.library_view.model().id, rows))
        for book_id in ids:
            # Get the current metadata for this book from the db
            mi = self.db.get_metadata(book_id, index_is_id=True,
                    get_cover=True, cover_as_data=True)
            fmts = self.db.formats(book_id, index_is_id=True)
            if not fmts: continue
            for fmt in fmts.split(','):
                fmt = fmt.lower()
                # Get a python file object for the format. This will be either
                # an in memory file or a temporary on disk file
                ffile = self.db.format(book_id, fmt, index_is_id=True,
                        as_file=True)
                # Set metadata in the format
                set_metadata(ffile, mi, fmt)
                ffile.seek(0)
                # Now replace the file in the calibre library with the updated
                # file. We dont use add_format_with_hooks as the hooks were
                # already run when the file was first added to calibre.
                ffile.name = 'xxx'  # add_format() will not work if the file
                                    # path of the file being added is the same
                                    # as the path of the file being replaced
                self.db.add_format(book_id, fmt, ffile, index_is_id=True)

        info_dialog(self, 'Updated files',
                'Updated the metadata in the files of %d book(s)'%len(ids),
                show=True)

    def config(self):
```

```
        self.do_user_config(parent=self)
        # Apply the changes
        self.label.setText(prefs['hello_world_msg'])
```

Getting resources from the plugin zip file

calibre's plugin loading system defines a couple of built-in functions that allow you to conveniently get files from the plugin zip file.

get_resources(name_or_list_of_names) This function should be called with a list of paths to files inside the zip file. For example to access the file icon.png in the directory images in the zip file, you would use: `images/icon.png`. Always use a forward slash as the path separator, even on windows. When you pass in a single name, the function will return the raw bytes of that file or None if the name was not found in the zip file. If you pass in more than one name then it returns a dict mapping the names to bytes. If a name is not found, it will not be present in the returned dict.

get_icons(name_or_list_of_names) A convenience wrapper for get_resources() that creates QIcon objects from the raw bytes returned by get_resources. If a name is not found in the zip file the corresponding QIcon will be null.

Enabling user configuration of your plugin

To allow users to configure your plugin, you must define three methods in your base plugin class, '**is_customizable**, **config_widget** and **save_settings** as shown below:

```
    def is_customizable(self):
        '''
        This method must return True to enable customization via
        Preferences->Plugins
        '''
        return True

    def config_widget(self):
        '''
        Implement this method and :meth:`save_settings` in your plugin to
        use a custom configuration dialog.

        This method, if implemented, must return a QWidget. The widget can have
        an optional method validate() that takes no arguments and is called
        immediately after the user clicks OK. Changes are applied if and only
        if the method returns True.

        If for some reason you cannot perform the configuration at this time,
        return a tuple of two strings (message, details), these will be
        displayed as a warning dialog to the user and the process will be
        aborted.

        The base class implementation of this method raises NotImplementedError
        so by default no user configuration is possible.
        '''
        # It is important to put this import statement here rather than at the
        # top of the module as importing the config class will also cause the
        # GUI libraries to be loaded, which we do not want when using calibre
        # from the command line
        from calibre_plugins.interface_demo.config import ConfigWidget
        return ConfigWidget()
```

```
    def save_settings(self, config_widget):
        '''
        Save the settings specified by the user with config_widget.

        :param config_widget: The widget returned by :meth:'config_widget'.
        '''
        config_widget.save_settings()

        # Apply the changes
        ac = self.actual_plugin_
        if ac is not None:
            ac.apply_settings()
```

calibre has many different ways to store configuration data (a legacy of its long history). The recommended way is to use the **JSONConfig** class, which stores your configuration information in a .json file.

The code to manage configuration data in the demo plugin is in config.py:

```
from PyQt4.Qt import QWidget, QHBoxLayout, QLabel, QLineEdit

from calibre.utils.config import JSONConfig

# This is where all preferences for this plugin will be stored
# Remember that this name (i.e. plugins/interface_demo) is also
# in a global namespace, so make it as unique as possible.
# You should always prefix your config file name with plugins/,
# so as to ensure you dont accidentally clobber a calibre config file
prefs = JSONConfig('plugins/interface_demo')

# Set defaults
prefs.defaults['hello_world_msg'] = 'Hello, World!'

class ConfigWidget(QWidget):

    def __init__(self):
        QWidget.__init__(self)
        self.l = QHBoxLayout()
        self.setLayout(self.l)

        self.label = QLabel('Hello world &message:')
        self.l.addWidget(self.label)

        self.msg = QLineEdit(self)
        self.msg.setText(prefs['hello_world_msg'])
        self.l.addWidget(self.msg)
        self.label.setBuddy(self.msg)

    def save_settings(self):
        prefs['hello_world_msg'] = unicode(self.msg.text())
```

The `prefs` object is now available throughout the plugin code by a simple:

```
from calibre_plugins.interface_demo.config import prefs
```

You can see the `prefs` object being used in main.py:

```
    def config(self):
        self.do_user_config(parent=self)
        # Apply the changes
        self.label.setText(prefs['hello_world_msg'])
```

The plugin API

As you may have noticed above, a plugin in calibre is a class. There are different classes for the different types of plugins in calibre. Details on each class, including the base class of all plugins can be found in *API Documentation for plugins* (page 441).

Your plugin is almost certainly going to use code from calibre. To learn how to find various bits of functionality in the calibre code base, read the section on the calibre *Code layout* (page 512).

Debugging plugins

The first, most important step is to run calibre in debug mode. You can do this from the command line with:

```
calibre-debug -g
```

Or from within calibre by right-clicking the preferences button or using the *Ctrl+Shift+R* keyboard shortcut.

When running from the command line, debug output will be printed to the console, when running from within calibre the output will go to a txt file.

You can insert print statements anywhere in your plugin code, they will be output in debug mode. Remember, this is python, you really shouldn't need anything more than print statements to debug ;) I developed all of calibre using just this debugging technique.

You can quickly test changes to your plugin by using the following command line:

```
calibre-debug -s; calibre-customize -b /path/to/your/plugin/directory; calibre
```

This will shutdown a running calibre, wait for the shutdown to complete, then update your plugin in calibre and relaunch calibre.

More plugin examples

You can find a list of many, sophisticated calibre plugins here[85].

Sharing your plugins with others

If you would like to share the plugins you have created with other users of calibre, post your plugin in a new thread in the calibre plugins forum[86].

1.7.8 Typesetting Math in ebooks

The calibre ebook viewer has the ability to display math embedded in ebooks (ePub and HTML files). You can typeset the math directly with TeX or MathML or AsciiMath. The calibre viewer uses the excellent MathJax[87] library to do this. This is a brief tutorial on creating ebooks with math in them that work well with the calibre viewer.

[85] http://www.mobileread.com/forums/showthread.php?t=118764
[86] http://www.mobileread.com/forums/forumdisplay.php?f=237
[87] http://www.mathjax.org

Note: This only applies to calibre version 0.8.66 and newer

A simple HTML file with mathematics

You can write mathematics inline inside a simple HTML file and the calibre viewer will render it into properly typeset mathematics. In the example below, we use TeX notation for mathematics. You will see that you can use normal TeX commands, with the small caveat that ampersands and less than and greater than signs have to be written as & < and > respectively.

The first step is to tell calibre that this will contains maths. You do this by adding the following snippet of code to the <head> section of the HTML file:

```
<script type="text/x-mathjax-config"></script>
```

That's it, now you can type mathematics just as you would in a .tex file. For example, here are Lorentz's equations:

```
<h2>The Lorenz Equations</h2>

<p>
\begin{align}
\dot{x} &amp;= \sigma(y-x) \\
\dot{y} &amp;= \rho x - y - xz \\
\dot{z} &amp;= -\beta z + xy
\end{align}
</p>
```

This snippet looks like the following screen shot in the calibre viewer.

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

Figure 1.2: *The Lorenz Equations*

The complete HTML file, with more equations and inline mathematics is reproduced below. You can convert this HTML file to EPUB in calibre to end up with an ebook you can distribute easily to other people.

```
<!DOCTYPE html>
<html>
<!-- Copyright (c) 2012 Design Science, Inc. -->
<head>
<title>Math Test Page</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />

<!-- This script tag is needed to make calibre's ebook-viewer recpgnize that this file needs math typ
<script type="text/x-mathjax-config">
    // This line adds numbers to all equations automatically, unless explicitly suppressed.
    MathJax.Hub.Config({ TeX: { equationNumbers: {autoNumber: "all"} } });
</script>

<style>
```

```
h1 {text-align:center}
h2 {
  font-weight: bold;
  background-color: #DDDDDD;
  padding: .2em .5em;
  margin-top: 1.5em;
  border-top: 3px solid #666666;
  border-bottom: 2px solid #999999;
}
</style>
</head>
<body>

<h1>Sample Equations</h1>

<h2>The Lorenz Equations</h2>

<p>
\begin{align}
\dot{x} &amp; = \sigma(y-x) \label{lorenz}\\
\dot{y} &amp; = \rho x - y - xz \\
\dot{z} &amp; = -\beta z + xy
\end{align}
</p>

<h2>The Cauchy-Schwarz Inequality</h2>

<p>\[
\left( \sum_{k=1}^n a_k b_k \right)^{\!\!2} \leq
 \left( \sum_{k=1}^n a_k^2 \right) \left( \sum_{k=1}^n b_k^2 \right)
\]</p>

<h2>A Cross Product Formula</h2>

<p>\[
  \mathbf{V}_1 \times \mathbf{V}_2 =
   \begin{vmatrix}
    \mathbf{i} &amp; \mathbf{j} &amp; \mathbf{k} \\
    \frac{\partial X}{\partial u} &amp; \frac{\partial Y}{\partial u} &amp; 0 \\
    \frac{\partial X}{\partial v} &amp; \frac{\partial Y}{\partial v} &amp; 0 \\
   \end{vmatrix}
\]</p>

<h2>The probability of getting \(k\) heads when flipping \(n\) coins is:</h2>

<p>\[P(E) = {n \choose k} p^k (1-p)^{ n-k} \]</p>

<h2>An Identity of Ramanujan</h2>

<p>\[
   \frac{1}{(\sqrt{\phi \sqrt{5}}-\phi) e^{\frac25 \pi}} =
     1+\frac{e^{-2\pi}} {1+\frac{e^{-4\pi}} {1+\frac{e^{-6\pi}}
      {1+\frac{e^{-8\pi}} {1+\ldots} } } }
\]</p>

<h2>A Rogers-Ramanujan Identity</h2>

<p>\[
```

```
  1 +  \frac{q^2}{(1-q)}+\frac{q^6}{(1-q)(1-q^2)}+\cdots =
    \prod_{j=0}^{\infty}\frac{1}{(1-q^{5j+2})(1-q^{5j+3})},
     \quad\quad \text{for $|q|&lt;1$}.
\]</p>

<h2>Maxwell's Equations</h2>

<p>
\begin{align}
  \nabla \times \vec{\mathbf{B}} -\, \frac1c\, \frac{\partial\vec{\mathbf{E}}}{\partial t} &amp; = \
  \nabla \cdot \vec{\mathbf{E}} &amp; = 4 \pi \rho \\
  \nabla \times \vec{\mathbf{E}}\, +\, \frac1c\, \frac{\partial\vec{\mathbf{B}}}{\partial t} &amp; =
  \nabla \cdot \vec{\mathbf{B}} &amp; = 0
\end{align}
</p>

<h2>In-line Mathematics</h2>

<p>While display equations look good for a page of samples, the
ability to mix math and text in a paragraph is also important.  This
expression \(\sqrt{3x-1}+(1+x)^2\) is an example of an inline equation.  As
you see, equations can be used this way as well, without unduly
disturbing the spacing between lines.</p>

<h2>References to equations</h2>

<p>Here is a reference to the Lorenz Equations (\ref{lorenz}). Clicking on the equation number will t

</body>
</html>
```

More information

Since the calibre viewer uses the MathJax library to render mathematics, the best place to find out more about math in ebooks and get help is the MathJax website[88].

1.7.9 Creating AZW3 • EPUB • MOBI Catalogs

calibre's Create catalog feature enables you to create a catalog of your library in a variety of formats. This help file describes cataloging options when generating a catalog in AZW3, EPUB and MOBI formats.

- Selecting books to catalog (page 436)
- Included sections (page 437)
- Prefixes (page 438)
- Excluded books (page 438)
- Excluded genres (page 439)
- Other options (page 439)
- Custom catalog covers (page 440)
- Additional help resources (page 440)

[88] http://www.mathjax.org

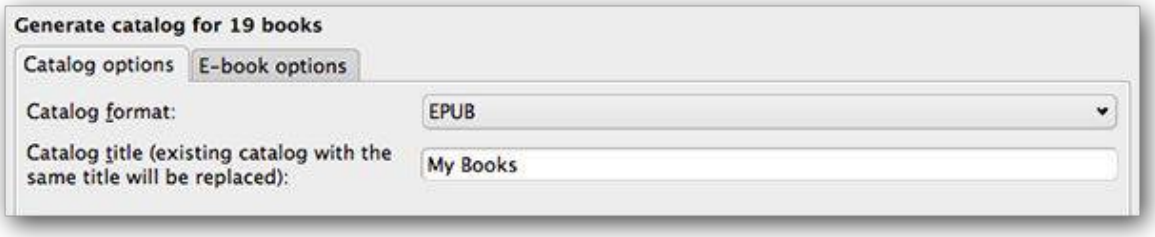
Selecting books to catalog

If you want *all* of your library cataloged, remove any search or filtering criteria in the main window. With a single book selected, all books in your library will be candidates for inclusion in the generated catalog. Individual books may be excluded by various criteria; see the *Excluded genres* (page 439) section below for more information.

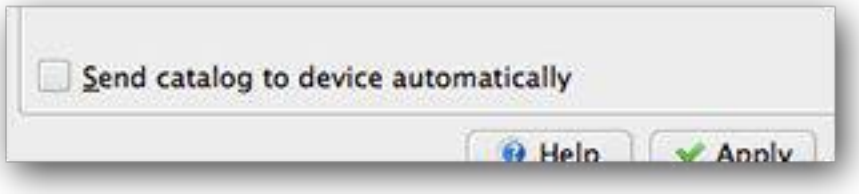
If you want only *some* of your library cataloged, you have two options:

- Create a multiple selection of the books you want cataloged. With more than one book selected in calibre's main window, only the selected books will be cataloged.
- Use the Search field or the Tag Browser to filter the displayed books. Only the displayed books will be cataloged.

To begin catalog generation, select the menu item *Convert books > Create a catalog of the books in your calibre library*. You may also add a *Create Catalog* button to a toolbar in *Preferences > Interface > Toolbars* for easier access to the Generate catalog dialog.

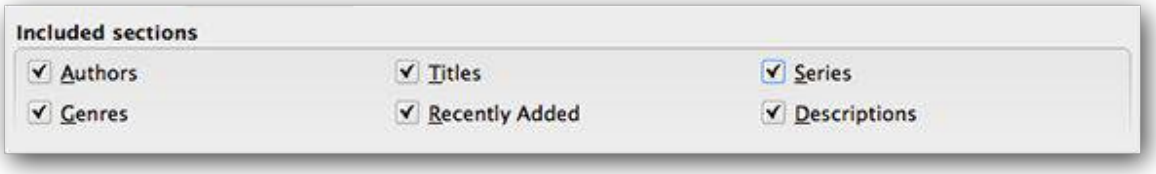


In *Catalog options*, select **AZW3, EPUB or MOBI** as the Catalog format. In the *Catalog title* field, provide a name that will be used for the generated catalog. If a catalog of the same name and format already exists, it will be replaced with the newly-generated catalog.



Enabling *Send catalog to device automatically* will download the generated catalog to a connected device upon completion.

Included sections

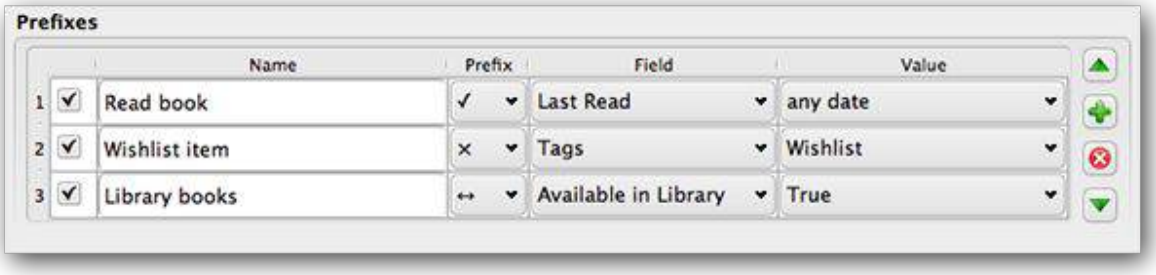


Sections enabled by a checkmark will be included in the generated catalog:

- *Authors* - all books, sorted by author, presented in a list format. Non-series books are listed before series books.
- *Titles* - all books, sorted by title, presented in a list format.
- *Series* - all books that are part of a series, sorted by series, presented in a list format.

- *Genres* - individual genres presented in a list, sorted by Author and Series.
- *Recently* Added - all books, sorted in reverse chronological order. List includes books added in the last 30 days, then a month-by-month listing of added books.
- *Descriptions* - detailed description page for each book, including a cover thumbnail and comments. Sorted by author, with non-series books listed before series books.

Prefixes



Prefix rules allow you to add a prefix to book listings when certain criteria are met. For example, you might want to mark books you've read with a checkmark, or books on your wishlist with an X.

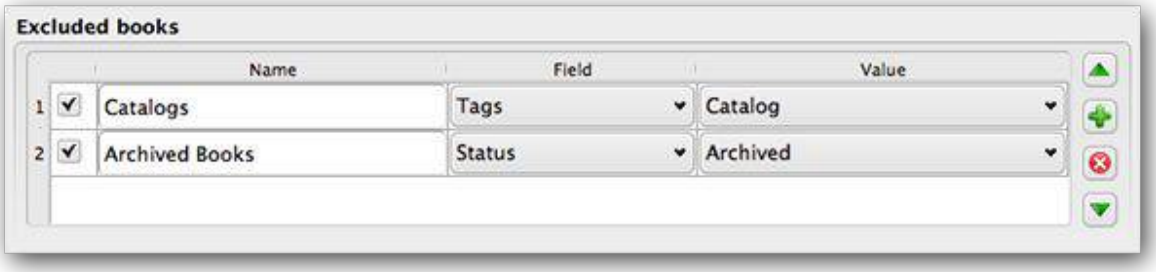
The checkbox in the first column enables the rule. *Name* is a rule name that you provide. *Field* is either *Tags* or a custom column from your library. *Value* is the content of *Field* to match. When a prefix rule is satisfied, the book will be marked with the selected *Prefix*.

Three prefix rules have been specified in the example above:

1. *Read book* specifies that a book with any date in a custom column named *Last read* will be prefixed with a checkmark symbol.
2. *Wishlist* item specifies that any book with a *Wishlist* tag will be prefixed with an X symbol.
3. *Library* books specifies that any book with a value of True (or Yes) in a custom column *Available in Library* will be prefixed with a double arrow symbol.

The first matching rule supplies the prefix. Disabled or incomplete rules are ignored.

Excluded books



Exclusion rules allow you to specify books that will not be cataloged.

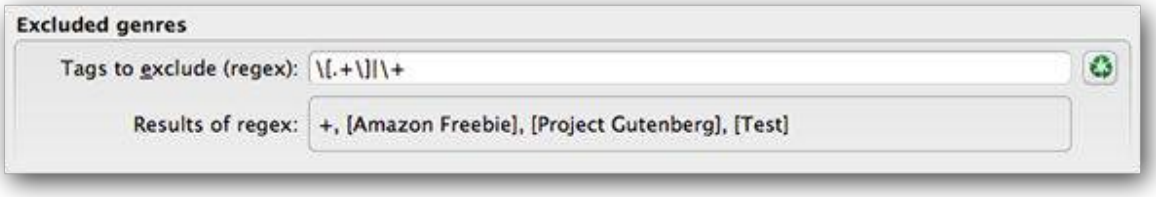
The checkbox in the first column enables the rule. *Name* is a rule name that you provide. *Field* is either *Tags* or a custom column in your library. *Value* is the content of *Field* to match. When an exclusion rule is satisfied, the book will be excluded from the generated catalog.

Two exclusion rules have been specified in the example above:

1. The *Catalogs* rule specifies that any book with a *Catalog* tag will be excluded from the generated catalog.
2. The *Archived* Books rule specifies that any book with a value of *Archived* in the custom column *Status* will be excluded from the generated catalog.

All rules are evaluated for every book. Disabled or incomplete rules are ignored.

Excluded genres



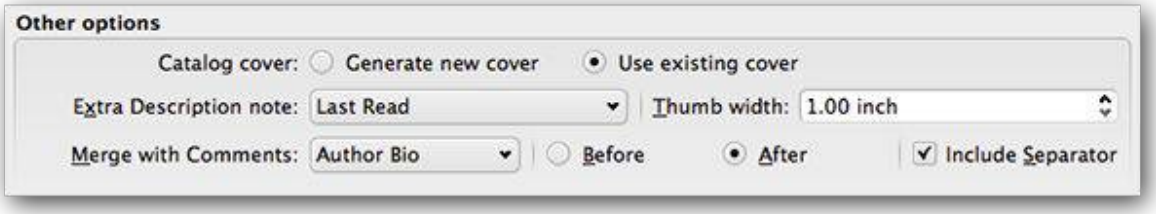
When the catalog is generated, tags in your database are used as genres. For example, you may use the tags `Fiction` and `Nonfiction`. These tags become genres in the generated catalog, with books listed under their respective genre lists based on their assigned tags. A book will be listed in every genre section for which it has a corresponding tag.

You may be using certain tags for other purposes, perhaps a + to indicate a read book, or a bracketed tag like `[Amazon Freebie]` to indicate a book's source. The *Excluded genres* regex allows you to specify tags that you don't want used as genres in the generated catalog. The default exclusion regex pattern `\[.+\]\+` excludes any tags of the form `[tag]`, as well as excluding +, the default tag for read books, from being used as genres in the generated catalog.

You can also use an exact tag name in a regex. For example, `[Amazon Freebie]` or `[Project Gutenberg]`. If you want to list multiple exact tags for exclusion, put a pipe (vertical bar) character between them: `[Amazon Freebie]|[Project Gutenberg]`.

Results of regex shows you which tags will be excluded when the catalog is built, based on the tags in your database and the regex pattern you enter. The results are updated as you modify the regex pattern.

Other options



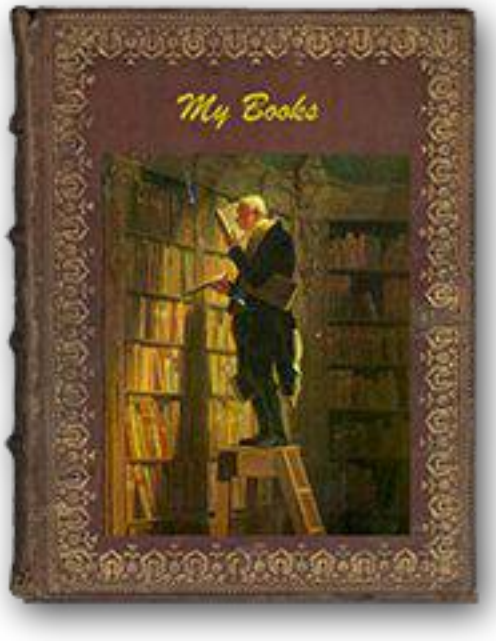
Catalog cover specifies whether to generate a new cover or use an existing cover. It is possible to create a custom cover for your catalogs - see *Custom catalog covers* (page 440) for more information. If you have created a custom cover that you want to reuse, select *Use existing cover*. Otherwise, select *Generate new cover*.

Extra Description note specifies a custom column's contents to be inserted into the Description page, next to the cover thumbnail. For example, you might want to display the date you last read a book using a *Last Read* custom column. For advanced use of the Description note feature, see this post in the calibre forum[89].

Thumb width specifies a width preference for cover thumbnails included with Descriptions pages. Thumbnails are cached to improve performance.To experiment with different widths, try generating a catalog with just a few books until you've determined your preferred width, then generate your full catalog. The first time a catalog is generated with a new thumbnail width, performance will be slower, but subsequent builds of the catalog will take advantage of the thumbnail cache.

Merge with Comments specifies a custom column whose content will be non-destructively merged with the Comments metadata during catalog generation. For example, you might have a custom column *Author Bio* that you'd like to append to the Comments metadata. You can choose to insert the custom column contents *before or after* the Comments section, and optionally separate the appended content with a horizontal rule separator. Eligible custom column types include `text, comments, and composite`.

Custom catalog covers



With the Generate Cover plugin[90] installed, you can create custom covers for your catalog. To install the plugin, go to *Preferences* > *Advanced* > *Plugins* > *Get new plugins*.

Additional help resources

For more information on calibre's Catalog feature, see the MobileRead forum sticky Creating Catalogs - Start here[91], where you can find information on how to customize the catalog templates, and how to submit a bug report.

To ask questions or discuss calibre's Catalog feature with other users, visit the MobileRead forum Calibre Catalogs[92].

[89] http://www.mobileread.com/forums/showpost.php?p=1335767&postcount=395
[90] http://www.mobileread.com/forums/showthread.php?t=124219
[91] http://www.mobileread.com/forums/showthread.php?t=118556
[92] http://www.mobileread.com/forums/forumdisplay.php?f=238

1.8 Customizing calibre

calibre has a highly modular design. Various parts of it can be customized. You can learn how to create *recipes* to add new sources of online content to calibre in the Section *Adding your favorite news website* (page 341). Here, you will learn, first, how to use environment variables and *tweaks* to customize calibre's behavior, and then how to specify your own static resources like icons and templates to override the defaults and finally how to use *plugins* to add functionality to calibre.

- Environment variables (page 462)
- Tweaks (page 462)
- Overriding icons, templates, et cetera (page 472)
- Customizing calibre with plugins (page 472)

1.8.1 API Documentation for plugins

Defines various abstract base classes that can be subclassed to create powerful plugins. The useful classes are:

- Plugin (page 441)
- FileTypePlugin (page 443)
- Metadata plugins (page 444)
- Catalog plugins (page 444)
- Metadata download plugins (page 445)
- Conversion plugins (page 447)
- Device Drivers (page 449)
- User Interface Actions (page 458)
- Preferences Plugins (page 460)

Plugin

class `calibre.customize.`**`Plugin`**(*plugin_path*)

A calibre plugin. Useful members include:

- **`self.plugin_path`: Stores path to the zip file that contains** this plugin or None if it is a builtin plugin
- **`self.site_customization`: Stores a customization string entered** by the user.

Methods that should be overridden in sub classes:

- `initialize()` (page 442)
- `customization_help()` (page 442)

Useful methods:

- `temporary_file()` (page 443)

`supported_platforms` = []

List of platforms this plugin works on For example: `['windows', 'osx', 'linux']`

`name` = 'Trivial Plugin'

The name of this plugin. You must set it something other than Trivial Plugin for it to work.

version = (1, 0, 0)
The version of this plugin as a 3-tuple (major, minor, revision)

description = u'Does absolutely nothing'
A short string describing what this plugin does

author = u'Unknown'
The author of this plugin

priority = 1
When more than one plugin exists for a filetype, the plugins are run in order of decreasing priority i.e. plugins with higher priority will be run first. The highest possible priority is `sys.maxint`. Default priority is 1.

minimum_calibre_version = (0, 4, 118)
The earliest version of calibre this plugin requires

can_be_disabled = True
If False, the user will not be able to disable this plugin. Use with care.

type = u'Base'
The type of this plugin. Used for categorizing plugins in the GUI

initialize()
Called once when calibre plugins are initialized. Plugins are re-initialized every time a new plugin is added.

Perform any plugin specific initialization here, such as extracting resources from the plugin zip file. The path to the zip file is available as `self.plugin_path`.

Note that `self.site_customization` is **not** available at this point.

config_widget()
Implement this method and `save_settings()` (page 442) in your plugin to use a custom configuration dialog, rather then relying on the simple string based default customization.

This method, if implemented, must return a QWidget. The widget can have an optional method validate() that takes no arguments and is called immediately after the user clicks OK. Changes are applied if and only if the method returns True.

If for some reason you cannot perform the configuration at this time, return a tuple of two strings (message, details), these will be displayed as a warning dialog to the user and the process will be aborted.

save_settings(*config_widget*)
Save the settings specified by the user with config_widget.

Parameters **config_widget** – The widget returned by `config_widget()` (page 442).

do_user_config(*parent=None*)
This method shows a configuration dialog for this plugin. It returns True if the user clicks OK, False otherwise. The changes are automatically applied.

load_resources(*names*)
If this plugin comes in a ZIP file (user added plugin), this method will allow you to load resources from the ZIP file.

For example to load an image:

```
pixmap = QPixmap()
pixmap.loadFromData(self.load_resources(['images/icon.png']).itervalues().next())
icon = QIcon(pixmap)
```

Parameters **names** – List of paths to resources in the zip file using / as separator

Returns A dictionary of the form `{name : file_contents}`. Any names that were not found in the zip file will not be present in the dictionary.

customization_help(*gui=False*)
Return a string giving help on how to customize this plugin. By default raise a `NotImplementedError`, which indicates that the plugin does not require customization.

If you re-implement this method in your subclass, the user will be asked to enter a string as customization for this plugin. The customization string will be available as `self.site_customization`.

Site customization could be anything, for example, the path to a needed binary on the user's computer.

Parameters **gui** – If True return HTML help, otherwise return plain text help.

temporary_file(*suffix*)
Return a file-like object that is a temporary file on the file system. This file will remain available even after being closed and will only be removed on interpreter shutdown. Use the `name` member of the returned object to access the full path to the created temporary file.

Parameters **suffix** – The suffix that the temporary file will have.

FileTypePlugin

class `calibre.customize.`**FileTypePlugin**(*plugin_path*)
Bases: `calibre.customize.Plugin` (page 441)

A plugin that is associated with a particular set of file types.

file_types = set([])
Set of file types for which this plugin should be run For example: `set(['lit', 'mobi', 'prc'])`

on_import = False
If True, this plugin is run when books are added to the database

on_postimport = False
If True, this plugin is run after books are added to the database

on_preprocess = False
If True, this plugin is run just before a conversion

on_postprocess = False
If True, this plugin is run after conversion on the final file produced by the conversion output plugin.

run(*path_to_ebook*)
Run the plugin. Must be implemented in subclasses. It should perform whatever modifications are required on the ebook and return the absolute path to the modified ebook. If no modifications are needed, it should return the path to the original ebook. If an error is encountered it should raise an Exception. The default implementation simply return the path to the original ebook.

The modified ebook file should be created with the `temporary_file()` method.

Parameters **path_to_ebook** – Absolute path to the ebook.

Returns Absolute path to the modified ebook.

postimport(*book_id*, *book_format*, *db*)
Called post import, i.e., after the book file has been added to the database.

Parameters

- **book_id** – Database id of the added book.
- **book_format** – The file type of the book that was added. :param db: Library database.

Metadata plugins

class calibre.customize.**MetadataReaderPlugin**(**args*, ***kwargs*)
Bases: calibre.customize.Plugin (page 441)

A plugin that implements reading metadata from a set of file types.

file_types = set([])
Set of file types for which this plugin should be run For example: `set(['lit', 'mobi', 'prc'])`

get_metadata(*stream*, *type*)
Return metadata for the file represented by stream (a file like object that supports reading). Raise an exception when there is an error with the input data. :param type: The type of file. Guaranteed to be one of the entries in file_types (page 444). :return: A `calibre.ebooks.metadata.book.Metadata` object

class calibre.customize.**MetadataWriterPlugin**(**args*, ***kwargs*)
Bases: calibre.customize.Plugin (page 441)

A plugin that implements reading metadata from a set of file types.

file_types = set([])
Set of file types for which this plugin should be run For example: `set(['lit', 'mobi', 'prc'])`

set_metadata(*stream*, *mi*, *type*)
Set metadata for the file represented by stream (a file like object that supports reading). Raise an exception when there is an error with the input data. :param type: The type of file. Guaranteed to be one of the entries in file_types (page 444). :param mi: A `calibre.ebooks.metadata.book.Metadata` object

Catalog plugins

class calibre.customize.**CatalogPlugin**(*plugin_path*)
Bases: calibre.customize.Plugin (page 441)

A plugin that implements a catalog generator.

file_types = set([])
Output file type for which this plugin should be run For example: 'epub' or 'xml'

cli_options = []
CLI parser options specific to this plugin, declared as namedtuple Option:

```
from collections import namedtuple
Option = namedtuple('Option', 'option, default, dest, help')
cli_options = [Option('--catalog-title',
                      default = 'My Catalog',
                      dest = 'catalog_title',
                      help = (_('Title of generated catalog. \nDefault:') + " '" +
                      '%default' + "'"))]
cli_options parsed in library.cli:catalog_option_parser()
```

initialize()
If plugin is not a built-in, copy the plugin's .ui and .py files from the zip file to $TMPDIR. Tab will be dynamically generated and added to the Catalog Options dialog in calibre.gui2.dialogs.catalog.py:Catalog

run(*path_to_output*, *opts*, *db*, *ids*, *notification=None*)
Run the plugin. Must be implemented in subclasses. It should generate the catalog in the format specified in file_types, returning the absolute path to the generated catalog file. If an error is encountered it should raise an Exception.

The generated catalog file should be created with the `temporary_file()` method.

Parameters

- **path_to_output** – Absolute path to the generated catalog file.
- **opts** – A dictionary of keyword arguments
- **db** – A LibraryDatabase2 object

Metadata download plugins

class `calibre.ebooks.metadata.sources.base.`**Source**(**args*, ***kwargs*)
Bases: `calibre.customize.Plugin` (page 441)

capabilities = frozenset([])
Set of capabilities supported by this plugin. Useful capabilities are: 'identify', 'cover'

touched_fields = frozenset([])
List of metadata fields that can potentially be download by this plugin during the identify phase

has_html_comments = False
Set this to True if your plugin returns HTML formatted comments

supports_gzip_transfer_encoding = False
Setting this to True means that the browser object will add Accept-Encoding: gzip to all requests. This can speedup downloads but make sure that the source actually supports gzip transfer encoding correctly first

cached_cover_url_is_reliable = True
Cached cover URLs can sometimes be unreliable (i.e. the download could fail or the returned image could be bogus. If that is often the case with this source set to False

options = ()
A list of `Option` objects. They will be used to automatically construct the configuration widget for this plugin

config_help_message = None
A string that is displayed at the top of the config widget for this plugin

is_configured()
Return False if your plugin needs to be configured before it can be used. For example, it might need a username/password/API key.

get_author_tokens(*authors*, *only_first_author=True*)
Take a list of authors and return a list of tokens useful for an AND search query. This function tries to return tokens in first name middle names last name order, by assuming that if a comma is in the author name, the name is in lastname, other names form.

get_title_tokens(*title*, *strip_joiners=True*, *strip_subtitle=False*)
Take a title and return a list of tokens useful for an AND search query. Excludes connectives(optionally) and punctuation.

split_jobs(*jobs*, *num*)
Split a list of jobs into at most num groups, as evenly as possible

test_fields(*mi*)
Return the first field from self.touched_fields that is null on the mi object

clean_downloaded_metadata(*mi*)
Call this method in your plugin's identify method to normalize metadata before putting the Metadata object into result_queue. You can of course, use a custom algorithm suited to your metadata source.

get_book_url(*identifiers*)
Return a 3-tuple or None. The 3-tuple is of the form: (identifier_type, identifier_value, URL). The URL is the URL for the book identified by identifiers at this source. identifier_type, identifier_value specify the identifier corresponding to the URL. This URL must be browseable to by a human using a browser. It is meant to provide a clickable link for the user to easily visit the books page at this source. If no URL is found, return None. This method must be quick, and consistent, so only implement it if it is possible to construct the URL from a known scheme given identifiers.

get_book_url_name(*idtype*, *idval*, *url*)
Return a human readable name from the return value of get_book_url().

get_cached_cover_url(*identifiers*)
Return cached cover URL for the book identified by the identifiers dict or None if no such URL exists.

Note that this method must only return validated URLs, i.e. not URLS that could result in a generic cover image or a not found error.

identify_results_keygen(*title=None*, *authors=None*, *identifiers={}*)
Return a function that is used to generate a key that can sort Metadata objects by their relevance given a search query (title, authors, identifiers).

These keys are used to sort the results of a call to `identify()` (page 446).

For details on the default algorithm see `InternalMetadataCompareKeyGen` (page 447). Re-implement this function in your plugin if the default algorithm is not suitable.

identify(*log*, *result_queue*, *abort*, *title=None*, *authors=None*, *identifiers={}*, *timeout=30*)
Identify a book by its title/author/isbn/etc.

If identifiers(s) are specified and no match is found and this metadata source does not store all related identifiers (for example, all ISBNs of a book), this method should retry with just the title and author (assuming they were specified).

If this metadata source also provides covers, the URL to the cover should be cached so that a subsequent call to the get covers API with the same ISBN/special identifier does not need to get the cover URL again. Use the caching API for this.

Every Metadata object put into result_queue by this method must have a *source_relevance* attribute that is an integer indicating the order in which the results were returned by the metadata source for this query. This integer will be used by `compare_identify_results()`. If the order is unimportant, set it to zero for every result.

Make sure that any cover/isbn mapping information is cached before the Metadata object is put into result_queue.

Parameters

- **log** – A log object, use it to output debugging information/errors
- **result_queue** – A result Queue, results should be put into it. Each result is a Metadata object
- **abort** – If abort.is_set() returns True, abort further processing and return as soon as possible
- **title** – The title of the book, can be None
- **authors** – A list of authors of the book, can be None
- **identifiers** – A dictionary of other identifiers, most commonly {'isbn':'1234...'}
- **timeout** – Timeout in seconds, no network request should hang for longer than timeout.

Returns None if no errors occurred, otherwise a unicode representation of the error suitable for showing to the user

download_cover(*log*, *result_queue*, *abort*, *title=None*, *authors=None*, *identifiers={}*, *timeout=30*)
Download a cover and put it into result_queue. The parameters all have the same meaning as for identify() (page 446). Put (self, cover_data) into result_queue.

This method should use cached cover URLs for efficiency whenever possible. When cached data is not present, most plugins simply call identify and use its results.

class calibre.ebooks.metadata.sources.base.**InternalMetadataCompareKeyGen**(*mi*, *source_plugin*, *title*, *authors*, *identifiers*)
Generate a sort key for comparison of the relevance of Metadata objects, given a search query. This is used only to compare results from the same metadata source, not across different sources.

The sort key ensures that an ascending order sort is a sort by order of decreasing relevance.

The algorithm is:

- Prefer results that have the same ISBN as specified in the query
- Prefer results with a cached cover URL
- Prefer results with all available fields filled in
- Prefer results that are an exact title match to the query
- Prefer results with longer comments (greater than 10% longer)
- **Use the relevance of the result as reported by the metadata source's search** engine

Conversion plugins

class calibre.customize.conversion.**InputFormatPlugin**(**args*)
Bases: calibre.customize.Plugin (page 441)

InputFormatPlugins are responsible for converting a document into HTML+OPF+CSS+etc. The results of the conversion *must* be encoded in UTF-8. The main action happens in convert() (page 448).

file_types = set([])
Set of file types for which this plugin should be run For example: `set(['azw', 'mobi', 'prc'])`

is_image_collection = False
If True, this input plugin generates a collection of images, one per HTML file. You can obtain access to the images via convenience method, `get_image_collection()`.

core_usage = 1
Number of CPU cores used by this plugin A value of -1 means that it uses all available cores

for_viewer = False
If set to True, the input plugin will perform special processing to make its output suitable for viewing

output_encoding = 'utf-8'
The encoding that this input plugin creates files in. A value of None means that the encoding is undefined and must be detected individually

common_options = set([<calibre.customize.conversion.OptionRecommendation object at 0x305ba90>])
Options shared by all Input format plugins. Do not override in sub-classes. Use options (page 448) instead. Every option must be an instance of OptionRecommendation.

options = set([])
Options to customize the behavior of this plugin. Every option must be an instance of OptionRecommendation.

recommendations = set([])
A set of 3-tuples of the form (option_name, recommended_value, recommendation_level)

get_images()
Return a list of absolute paths to the images, if this input plugin represents an image collection. The list of images is in the same order as the spine and the TOC.

convert(*stream*, *options*, *file_ext*, *log*, *accelerators*)
This method must be implemented in sub-classes. It must return the path to the created OPF file or an OEBBook instance. All output should be contained in the current directory. If this plugin creates files outside the current directory they must be deleted/marked for deletion before this method returns.

Parameters

- **stream** – A file like object that contains the input file.
- **options** – Options to customize the conversion process. Guaranteed to have attributes corresponding to all the options declared by this plugin. In addition, it will have a verbose attribute that takes integral values from zero upwards. Higher numbers mean be more verbose. Another useful attribute is input_profile that is an instance of calibre.customize.profiles.InputProfile.
- **file_ext** – The extension (without the .) of the input file. It is guaranteed to be one of the *file_types* supported by this plugin.
- **log** – A calibre.utils.logging.Log object. All output should use this object.
- **accelarators** – A dictionary of various information that the input plugin can get easily that would speed up the subsequent stages of the conversion.

postprocess_book(*oeb*, *opts*, *log*)
Called to allow the input plugin to perform postprocessing after the book has been parsed.

specialize(*oeb*, *opts*, *log*, *output_fmt*)
Called to allow the input plugin to specialize the parsed book for a particular output format. Called after postprocess_book and before any transforms are performed on the parsed book.

gui_configuration_widget(*parent*, *get_option_by_name*, *get_option_help*, *db*, *book_id=None*)
Called to create the widget used for configuring this plugin in the calibre GUI. The widget must be an instance of the PluginWidget class. See the builting input plugins for examples.

class calibre.customize.conversion.**OutputFormatPlugin**(**args*)
Bases: calibre.customize.Plugin (page 441)

OutputFormatPlugins are responsible for converting an OEB document (OPF+HTML) into an output ebook.

The OEB document can be assumed to be encoded in UTF-8. The main action happens in convert() (page 449).

file_type = None
The file type (extension without leading period) that this plugin outputs

common_options = set([<calibre.customize.conversion.OptionRecommendation object at 0x305bbd0>])
Options shared by all Input format plugins. Do not override in sub-classes. Use options (page 448) instead. Every option must be an instance of OptionRecommendation.

options = set([])
Options to customize the behavior of this plugin. Every option must be an instance of `OptionRecommendation`.

recommendations = set([])
A set of 3-tuples of the form (option_name, recommended_value, recommendation_level)

convert(*oeb_book*, *output*, *input_plugin*, *opts*, *log*)
Render the contents of *oeb_book* (which is an instance of `calibre.ebooks.oeb.OEBBook` to the file specified by output.

Parameters

- **output** – Either a file like object or a string. If it is a string it is the path to a directory that may or may not exist. The output plugin should write its output into that directory. If it is a file like object, the output plugin should write its output into the file.
- **input_plugin** – The input plugin that was used at the beginning of the conversion pipeline.
- **opts** – Conversion options. Guaranteed to have attributes corresponding to the OptionRecommendations of this plugin.
- **log** – The logger. Print debug/info messages etc. using this.

specialize_css_for_output(*log*, *opts*, *item*, *stylizer*)
Can be used to make changes to the css during the CSS flattening process.

Parameters

- **item** – The item (HTML file) being processed
- **stylizer** – A Stylizer object containing the flattened styles for item. You can get the style for any element by stylizer.style(element).

gui_configuration_widget(*parent*, *get_option_by_name*, *get_option_help*, *db*, *book_id=None*)
Called to create the widget used for configuring this plugin in the calibre GUI. The widget must be an instance of the PluginWidget class. See the builtin output plugins for examples.

Device Drivers

The base class for all device drivers is `DevicePlugin` (page 449). However, if your device exposes itself as a USBMS drive to the operating system, you should use the USBMS class instead as it implements all the logic needed to support these kinds of devices.

class `calibre.devices.interface.`**DevicePlugin**(*plugin_path*)
Bases: `calibre.customize.Plugin` (page 441)

Defines the interface that should be implemented by backends that communicate with an ebook reader.

FORMATS = ['lrf', 'rtf', 'pdf', 'txt']
Ordered list of supported formats

VENDOR_ID = 0
VENDOR_ID can be either an integer, a list of integers or a dictionary If it is a dictionary, it must be a dictionary of dictionaries, of the form:

```
{
 integer_vendor_id : { product_id : [list of BCDs], ... },
 ...
}
```

PRODUCT_ID = 0
An integer or a list of integers

BCD = None
BCD can be either None to not distinguish between devices based on BCD, or it can be a list of the BCD numbers of all devices supported by this driver.

THUMBNAIL_HEIGHT = 68
Height for thumbnails on the device

WANTS_UPDATED_THUMBNAILS = False
Width for thumbnails on the device. Setting this will force thumbnails to this size, not preserving aspect ratio. If it is not set, then the aspect ratio will be preserved and the thumbnail will be no higher than THUMBNAIL_HEIGHT Set this to True if the device supports updating cover thumbnails during sync_booklists. Setting it to true will ask device.py to refresh the cover thumbnails during book matching

CAN_SET_METADATA = ['title', 'authors', 'collections']
Whether the metadata on books can be set via the GUI.

CAN_DO_DEVICE_DB_PLUGBOARD = False
Whether the device can handle device_db metadata plugboards

path_sep = '/'
Path separator for paths to books on device

icon = '/home/kovid/work/calibre/resources/images/reader.png'
Icon for this device

UserAnnotation
alias of `Annotation`

OPEN_FEEDBACK_MESSAGE = None
GUI displays this as a message if not None. Useful if opening can take a long time

VIRTUAL_BOOK_EXTENSIONS = frozenset([])
Set of extensions that are "virtual books" on the device and therefore cannot be viewed/saved/added to library For example: `frozenset(['kobo'])`

NUKE_COMMENTS = None
Whether to nuke comments in the copy of the book sent to the device. If not None this should be short string that the comments will be replaced by.

MANAGES_DEVICE_PRESENCE = False
If True indicates that this driver completely manages device detection, ejecting and so forth. If you set this to True, you *must* implement the detect_managed_devices and debug_managed_device_detection methods. A driver with this set to true is responsible for detection of devices, managing a blacklist of devices, a list of ejected devices and so forth. calibre will periodically call the detect_managed_devices() method and is it returns a detected device, calibre will call open(). open() will be called every time a device is returned even is previous calls to open() failed, therefore the driver must maintain its own blacklist of failed devices. Similarly, when ejecting, calibre will call eject() and then assuming the next call to detect_managed_devices() returns None, it will call post_yank_cleanup().

SLOW_DRIVEINFO = False
If set the True, calibre will call the `get_driveinfo()` (page 452) method after the books lists have been loaded to get the driveinfo.

ASK_TO_ALLOW_CONNECT = False
If set to True, calibre will ask the user if they want to manage the device with calibre, the first time it is detected. If you set this to True you must implement `get_device_uid()` (page 454) and `ignore_connected_device()` (page 454) and `get_user_blacklisted_devices()` (page 455) and `set_user_blacklisted_devices()` (page 455)

is_usb_connected(*devices_on_system*, *debug=False*, *only_presence=False*)
Return True, device_info if a device handled by this plugin is currently connected.

Parameters **devices_on_system** – List of devices currently connected

detect_managed_devices(*devices_on_system*, *force_refresh=False*)
Called only if MANAGES_DEVICE_PRESENCE is True.

Scan for devices that this driver can handle. Should return a device object if a device is found. This object will be passed to the open() method as the connected_device. If no device is found, return None. The returned object can be anything, calibre does not use it, it is only passed to open().

This method is called periodically by the GUI, so make sure it is not too resource intensive. Use a cache to avoid repeatedly scanning the system.

Parameters

- **devices_on_system** – Set of USB devices found on the system.
- **force_refresh** – If True and the driver uses a cache to prevent repeated scanning, the cache must be flushed.

debug_managed_device_detection(*devices_on_system*, *output*)
Called only if MANAGES_DEVICE_PRESENCE is True.

Should write information about the devices detected on the system to output, which is a file like object.

Should return True if a device was detected and successfully opened, otherwise False.

reset(*key='-1'*, *log_packets=False*, *report_progress=None*, *detected_device=None*)

Parameters

- **key** – The key to unlock the device
- **log_packets** – If true the packet stream to/from the device is logged
- **report_progress** – Function that is called with a % progress (number between 0 and 100) for various tasks If it is called with -1 that means that the task does not have any progress information
- **detected_device** – Device information from the device scanner

can_handle_windows(*device_id*, *debug=False*)
Optional method to perform further checks on a device to see if this driver is capable of handling it. If it is not it should return False. This method is only called after the vendor, product ids and the bcd have matched, so it can do some relatively time intensive checks. The default implementation returns True. This method is called only on windows. See also `can_handle()` (page 451).

Parameters **device_info** – On windows a device ID string. On Unix a tuple of `(vendor_id, product_id, bcd)`.

can_handle(*device_info*, *debug=False*)
Unix version of `can_handle_windows()` (page 451)

Parameters **device_info** – Is a tuple of (vid, pid, bcd, manufacturer, product, serial number)

open(*connected_device*, *library_uuid*)
Perform any device specific initialization. Called after the device is detected but before any other functions that communicate with the device. For example: For devices that present themselves as USB Mass storage devices, this method would be responsible for mounting the device or if the device has been automounted, for finding out where it has been mounted. The method `calibre.devices.usbms.device.Device.open()` has an implementation of this function that should serve as a good example for USB Mass storage devices.

This method can raise an OpenFeedback exception to display a message to the user.

Parameters

- **connected_device** – The device that we are trying to open. It is a tuple of (vendor id, product id, bcd, manufacturer name, product name, device serial number). However, some devices have no serial number and on windows only the first three fields are present, the rest are None.
- **library_uuid** – The UUID of the current calibre library. Can be None if there is no library (for example when used from the command line).

eject()
Un-mount / eject the device from the OS. This does not check if there are pending GUI jobs that need to communicate with the device.

NOTE: That this method may not be called on the same thread as the rest of the device methods.

post_yank_cleanup()
Called if the user yanks the device without ejecting it first.

set_progress_reporter(*report_progress*)
Set a function to report progress information.

Parameters **report_progress** – Function that is called with a % progress (number between 0 and 100) for various tasks If it is called with -1 that means that the task does not have any progress information

get_device_information(*end_session=True*)
Ask device for device information. See L{DeviceInfoQuery}.

Returns (device name, device version, software version on device, mime type) The tuple can optionally have a fifth element, which is a drive information dictionary. See usbms.driver for an example.

get_driveinfo()
Return the driveinfo dictionary. Usually called from get_device_information(), but if loading the driveinfo is slow for this driver, then it should set SLOW_DRIVEINFO. In this case, this method will be called by calibre after the book lists have been loaded. Note that it is not called on the device thread, so the driver should cache the drive info in the books() method and this function should return the cached data.

card_prefix(*end_session=True*)
Return a 2 element list of the prefix to paths on the cards. If no card is present None is set for the card's prefix. E.G. ('/place', '/place2') (None, 'place2') ('place', None) (None, None)

total_space(*end_session=True*)

Get total space available on the mountpoints:

1. Main memory
2. Memory Card A
3. Memory Card B

Returns A 3 element list with total space in bytes of (1, 2, 3). If a particular device doesn't have any of these locations it should return 0.

free_space(*end_session=True*)

Get free space available on the mountpoints:

1. Main memory

2. Card A
3. Card B

Returns A 3 element list with free space in bytes of (1, 2, 3). If a particular device doesn't have any of these locations it should return -1.

books (*oncard=None*, *end_session=True*)
Return a list of ebooks on the device.

Parameters **oncard** – If 'carda' or 'cardb' return a list of ebooks on the specific storage card, otherwise return list of ebooks in main memory of device. If a card is specified and no books are on the card return empty list.

Returns A BookList.

upload_books (*files*, *names*, *on_card=None*, *end_session=True*, *metadata=None*)
Upload a list of books to the device. If a file already exists on the device, it should be replaced. This method should raise a `FreeSpaceError` if there is not enough free space on the device. The text of the FreeSpaceError must contain the word "card" if `on_card` is not None otherwise it must contain the word "memory".

Parameters

- **files** – A list of paths
- **names** – A list of file names that the books should have once uploaded to the device. len(names) == len(files)
- **metadata** – If not None, it is a list of `Metadata` objects. The idea is to use the metadata to determine where on the device to put the book. len(metadata) == len(files). Apart from the regular cover (path to cover), there may also be a thumbnail attribute, which should be used in preference. The thumbnail attribute is of the form (width, height, cover_data as jpeg).

Returns A list of 3-element tuples. The list is meant to be passed to `add_books_to_metadata()` (page 453).

classmethod **add_books_to_metadata** (*locations*, *metadata*, *booklists*)
Add locations to the booklists. This function must not communicate with the device.

Parameters

- **locations** – Result of a call to L{upload_books}
- **metadata** – List of `Metadata` objects, same as for `upload_books()` (page 453).
- **booklists** – A tuple containing the result of calls to (`books(oncard=None)()`, `books(oncard='carda')()`, :meth'books(oncard='cardb')').

delete_books (*paths*, *end_session=True*)
Delete books at paths on device.

classmethod **remove_books_from_metadata** (*paths*, *booklists*)
Remove books from the metadata list. This function must not communicate with the device.

Parameters

- **paths** – paths to books on the device.
- **booklists** – A tuple containing the result of calls to (`books(oncard=None)()`, `books(oncard='carda')()`, :meth'books(oncard='cardb')').

sync_booklists(*booklists*, *end_session=True*)
Update metadata on device.

Parameters **booklists** – A tuple containing the result of calls to (`books(oncard=None)()`, `books(oncard='carda')()`, :meth'books(oncard='cardb')').

get_file(*path*, *outfile*, *end_session=True*)
Read the file at `path` on the device and write it to outfile.

Parameters **outfile** – file object like `sys.stdout` or the result of an `open()` (page 451) call.

classmethod config_widget()
Should return a QWidget. The QWidget contains the settings for the device interface

classmethod save_settings(*settings_widget*)
Should save settings to disk. Takes the widget created in `config_widget()` (page 454) and saves all settings to disk.

classmethod settings()
Should return an opts object. The opts object should have at least one attribute *format_map* which is an ordered list of formats for the device.

set_plugboards(*plugboards*, *pb_func*)
provide the driver the current set of plugboards and a function to select a specific plugboard. This method is called immediately before add_books and sync_booklists.

pb_func is a callable with the following signature:: def pb_func(device_name, format, plugboards)

You give it the current device name (either the class name or DEVICE_PLUGBOARD_NAME), the format you are interested in (a 'real' format or 'device_db'), and the plugboards (you were given those by set_plugboards, the same place you got this method).

Returns None or a single plugboard instance.

set_driveinfo_name(*location_code*, *name*)
Set the device name in the driveinfo file to 'name'. This setting will persist until the file is re-created or the name is changed again.

Non-disk devices should implement this method based on the location codes returned by the get_device_information() method.

prepare_addable_books(*paths*)
Given a list of paths, returns another list of paths. These paths point to addable versions of the books.

If there is an error preparing a book, then instead of a path, the position in the returned list for that book should be a three tuple: (original_path, the exception instance, traceback)

startup()
Called when calibre is is starting the device. Do any initialization required. Note that multiple instances of the class can be instantiated, and thus __init__ can be called multiple times, but only one instance will have this method called. This method is called on the device thread, not the GUI thread.

shutdown()
Called when calibre is shutting down, either for good or in preparation to restart. Do any cleanup required. This method is called on the device thread, not the GUI thread.

get_device_uid()
Must return a unique id for the currently connected device (this is called immediately after a successful call to open()). You must implement this method if you set ASK_TO_ALLOW_CONNECT = True

ignore_connected_device(*uid*)
Should ignore the device identified by uid (the result of a call to get_device_uid()) in the future. You must

implement this method if you set ASK_TO_ALLOW_CONNECT = True. Note that this function is called immediately after open(), so if open() caches some state, the driver should reset that state.

get_user_blacklisted_devices()
Return map of device uid to friendly name for all devices that the user has asked to be ignored.

set_user_blacklisted_devices(*devices*)
Set the list of device uids that should be ignored by this driver.

specialize_global_preferences(*device_prefs*)
Implement this method if your device wants to override a particular preference. You must ensure that all call sites that want a preference that can be overridden use device_prefs['something'] instead of prefs['something']. Your method should call device_prefs.set_overrides(pref=val, pref=val, ...). Currently used for: metadata management (prefs['manage_device_metadata'])

is_dynamically_controllable()
Called by the device manager when starting plugins. If this method returns a string, then a) it supports the device manager's dynamic control interface, and b) that name is to be used when talking to the plugin.

This method can be called on the GUI thread. A driver that implements this method must be thread safe.

start_plugin()
This method is called to start the plugin. The plugin should begin to accept device connections however it does that. If the plugin is already accepting connections, then do nothing.

This method can be called on the GUI thread. A driver that implements this method must be thread safe.

stop_plugin()
This method is called to stop the plugin. The plugin should no longer accept connections, and should cleanup behind itself. It is likely that this method should call shutdown. If the plugin is already not accepting connections, then do nothing.

This method can be called on the GUI thread. A driver that implements this method must be thread safe.

get_option(*opt_string*, *default=None*)
Return the value of the option indicated by opt_string. This method can be called when the plugin is not started. Return None if the option does not exist.

This method can be called on the GUI thread. A driver that implements this method must be thread safe.

set_option(*opt_string*, *opt_value*)
Set the value of the option indicated by opt_string. This method can be called when the plugin is not started.

This method can be called on the GUI thread. A driver that implements this method must be thread safe.

is_running()
Return True if the plugin is started, otherwise false

This method can be called on the GUI thread. A driver that implements this method must be thread safe.

class calibre.devices.interface.**BookList**(*oncard*, *prefix*, *settings*)
Bases: list

A list of books. Each Book object must have the fields

1.title

2.authors

3.size (file size of the book)

4.datetime (a UTC time tuple)

5.path (path on the device to the book)

6. thumbnail (can be None) thumbnail is either a str/bytes object with the image data or it should have an attribute image_path that stores an absolute (platform native) path to the image

7. tags (a list of strings, can be empty).

supports_collections()
Return True if the device supports collections for this book list.

add_book(*book*, *replace_metadata*)
Add the book to the booklist. Intent is to maintain any device-internal metadata. Return True if booklists must be sync'ed

remove_book(*book*)
Remove a book from the booklist. Correct any device metadata at the same time

get_collections(*collection_attributes*)
Return a dictionary of collections created from collection_attributes. Each entry in the dictionary is of the form collection name:[list of books]

The list of books is sorted by book title, except for collections created from series, in which case series_index is used.

Parameters **collection_attributes** – A list of attributes of the Book object

USB Mass Storage based devices

The base class for such devices is calibre.devices.usbms.driver.USBMS (page 457). This class in turn inherits some of its functionality from its bases, documented below. A typical basic USBMS based driver looks like this:

```
from calibre.devices.usbms.driver import USBMS

class PDNOVEL(USBMS):
    name = 'Pandigital Novel device interface'
    gui_name = 'PD Novel'
    description = _('Communicate with the Pandigital Novel')
    author = 'Kovid Goyal'
    supported_platforms = ['windows', 'linux', 'osx']
    FORMATS = ['epub', 'pdf']

    VENDOR_ID   = [0x18d1]
    PRODUCT_ID  = [0xb004]
    BCD         = [0x224]

    VENDOR_NAME = 'ANDROID'
    WINDOWS_MAIN_MEM = WINDOWS_CARD_A_MEM = '__UMS_COMPOSITE'
    THUMBNAIL_HEIGHT = 144

    EBOOK_DIR_MAIN = 'eBooks'
    SUPPORTS_SUB_DIRS = False

    def upload_cover(self, path, filename, metadata):
        coverdata = getattr(metadata, 'thumbnail', None)
        if coverdata and coverdata[2]:
            with open('%s.jpg' % os.path.join(path, filename), 'wb') as coverfile:
                coverfile.write(coverdata[2])
```

class calibre.devices.usbms.device.**Device**(*plugin_path*)
Bases: calibre.devices.usbms.deviceconfig.DeviceConfig,

calibre.devices.interface.DevicePlugin (page 449)

This class provides logic common to all drivers for devices that export themselves as USB Mass Storage devices. Provides implementations for mounting/ejecting of USBMS devices on all platforms.

WINDOWS_MAIN_MEM = None
String identifying the main memory of the device in the windows PnP id strings This can be None, string, list of strings or compiled regex

WINDOWS_CARD_A_MEM = None
String identifying the first card of the device in the windows PnP id strings This can be None, string, list of strings or compiled regex

WINDOWS_CARD_B_MEM = None
String identifying the second card of the device in the windows PnP id strings This can be None, string, list of strings or compiled regex

OSX_MAIN_MEM_VOL_PAT = None
Used by the new driver detection to disambiguate main memory from storage cards. Should be a regular expression that matches the main memory mount point assigned by OS X

MAX_PATH_LEN = 250
The maximum length of paths created on the device

NEWS_IN_FOLDER = True
Put news in its own folder

windows_sort_drives(*drives*)
Called to disambiguate main memory and storage card for devices that do not distinguish between them on the basis of *WINDOWS_CARD_NAME*. For e.g.: The EB600

filename_callback(*default*, *mi*)
Callback to allow drivers to change the default file name set by `create_upload_path()`.

sanitize_path_components(*components*)
Perform any device specific sanitization on the path components for files to be uploaded to the device

get_annotations(*path_map*)
Resolve path_map to annotation_map of files found on the device

add_annotation_to_library(*db*, *db_id*, *annotation*)
Add an annotation to the calibre library

class calibre.devices.usbms.cli.**CLI**

class calibre.devices.usbms.driver.**USBMS**(*plugin_path*)
Bases: calibre.devices.usbms.cli.CLI (page 457), calibre.devices.usbms.device.Device (page 456)

The base class for all USBMS devices. Implements the logic for sending/getting/updating metadata/caching metadata/etc.

upload_cover(*path*, *filename*, *metadata*, *filepath*)
Upload book cover to the device. Default implementation does nothing.

Parameters

- **path** – The full path to the directory where the associated book is located.
- **filename** – The name of the book file without the extension.
- **metadata** – metadata belonging to the book. Use metadata.thumbnail for cover
- **filepath** – The full path to the ebook file

classmethod normalize_path(*path*)
Return path with platform native path separators

User Interface Actions

If you are adding your own plugin in a zip file, you should subclass both InterfaceActionBase and InterfaceAction. The `load_actual_plugin()` method of you InterfaceActionBase subclass must return an instantiated object of your InterfaceBase subclass.

class `calibre.gui2.actions.`**InterfaceAction**(*parent*, *site_customization*)
Bases: `PyQt4.QtCore.QObject`

A plugin representing an "action" that can be taken in the graphical user interface. All the items in the toolbar and context menus are implemented by these plugins.

Note that this class is the base class for these plugins, however, to integrate the plugin with calibre's plugin system, you have to make a wrapper class that references the actual plugin. See the `calibre.customize.builtins` module for examples.

If two `InterfaceAction` objects have the same name, the one with higher priority takes precedence.

Sub-classes should implement the `genesis()`, `library_changed()`, `location_selected()` `shutting_down()` and `initialization_complete()` methods.

Once initialized, this plugin has access to the main calibre GUI via the `gui` member. You can access other plugins by name, for example:

```
self.gui.iactions['Save To Disk']
```

To access the actual plugin, use the `interface_action_base_plugin` attribute, this attribute only becomes available after the plugin has been initialized. Useful if you want to use methods from the plugin class like do_user_config().

The QAction specified by `action_spec` is automatically create and made available as `self.qaction`.

name = 'Implement me'
The plugin name. If two plugins with the same name are present, the one with higher priority takes precedence.

priority = 1
The plugin priority. If two plugins with the same name are present, the one with higher priority takes precedence.

popup_type = 1
The menu popup type for when this plugin is added to a toolbar

auto_repeat = False
Whether this action should be auto repeated when its shortcut key is held down.

action_spec = ('text', 'icon', None, None)
Of the form: (text, icon_path, tooltip, keyboard shortcut) icon, tooltip and keyboard shortcut can be None shortcut must be a string, None or tuple of shortcuts. If None, a keyboard shortcut corresponding to the action is not registered. If you pass an empty tuple, then the shortcut is registered with no default key binding.

action_add_menu = False
If True, a menu is automatically created and added to self.qaction

action_menu_clone_qaction = False
If True, a clone of self.qaction is added to the menu of self.qaction If you want the text of this action to be different from that of self.qaction, set this variable to the new text

dont_add_to = frozenset([])
Set of locations to which this action must not be added. See `all_locations` for a list of possible locations

dont_remove_from = frozenset([])
Set of locations from which this action must not be removed. See `all_locations` for a list of possible locations

action_type = 'global'
Type of action 'current' means acts on the current view 'global' means an action that does not act on the current view, but rather on calibre as a whole

accepts_drops = False
If True, then this InterfaceAction will have the opportunity to interact with drag and drop events. See the methods, `accept_enter_event()`, :meth':accept_drag_move_event', `drop_event()` for details.

accept_enter_event(*event*, *mime_data*)
This method should return True iff this interface action is capable of handling the drag event. Do not call accept/ignore on the event, that will be taken care of by the calibre UI.

accept_drag_move_event(*event*, *mime_data*)
This method should return True iff this interface action is capable of handling the drag event. Do not call accept/ignore on the event, that will be taken care of by the calibre UI.

drop_event(*event*, *mime_data*)
This method should perform some useful action and return True iff this interface action is capable of handling the drop event. Do not call accept/ignore on the event, that will be taken care of by the calibre UI. You should not perform blocking/long operations in this function. Instead emit a signal or use QTimer.singleShot and return quickly. See the builtin actions for examples.

create_menu_action(*menu*, *unique_name*, *text*, *icon=None*, *shortcut=None*, *description=None*, *triggered=None*, *shortcut_name=None*)
Convenience method to easily add actions to a QMenu. Returns the created QAction, This action has one extra attribute calibre_shortcut_unique_name which if not None refers to the unique name under which this action is registered with the keyboard manager.

Parameters

- **menu** – The QMenu the newly created action will be added to
- **unique_name** – A unique name for this action, this must be globally unique, so make it as descriptive as possible. If in doubt add a uuid to it.
- **text** – The text of the action.
- **icon** – Either a QIcon or a file name. The file name is passed to the I() builtin, so you do not need to pass the full path to the images directory.
- **shortcut** – A string, a list of strings, None or False. If False, no keyboard shortcut is registered for this action. If None, a keyboard shortcut with no default keybinding is registered. String and list of strings register a shortcut with default keybinding as specified.
- **description** – A description for this action. Used to set tooltips.
- **triggered** – A callable which is connected to the triggered signal of the created action.
- **shortcut_name** – The test displayed to the user when customizing the keyboard shortcuts for this action. By default it is set to the value of `text`.

load_resources(*names*)
If this plugin comes in a ZIP file (user added plugin), this method will allow you to load resources from the ZIP file.

For example to load an image:

```
pixmap = QPixmap()
pixmap.loadFromData(self.load_resources(['images/icon.png']).itervalues().next())
icon = QIcon(pixmap)
```

Parameters **names** – List of paths to resources in the zip file using / as separator

Returns A dictionary of the form `{name : file_contents}`. Any names that were not found in the zip file will not be present in the dictionary.

genesis()
Setup this plugin. Only called once during initialization. self.gui is available. The action specified by `action_spec` is available as `self.qaction`.

location_selected(*loc*)
Called whenever the book list being displayed in calibre changes. Currently values for loc are: `library, main, card and cardb`.

This method should enable/disable this action and its sub actions as appropriate for the location.

library_changed(*db*)
Called whenever the current library is changed.

Parameters **db** – The LibraryDatabase corresponding to the current library.

gui_layout_complete()
Called once per action when the layout of the main GUI is completed. If your action needs to make changes to the layout, they should be done here, rather than in `initialization_complete()`.

initialization_complete()
Called once per action when the initialization of the main GUI is completed.

shutting_down()
Called once per plugin when the main GUI is in the process of shutting down. Release any used resources, but try not to block the shutdown for long periods of time.

Returns False to halt the shutdown. You are responsible for telling the user why the shutdown was halted.

class calibre.customize.**InterfaceActionBase**(**args*, ***kwargs*)
Bases: calibre.customize.Plugin (page 441)

load_actual_plugin(*gui*)
This method must return the actual interface action plugin object.

Preferences Plugins

class calibre.customize.**PreferencesPlugin**(*plugin_path*)
Bases: calibre.customize.Plugin (page 441)

A plugin representing a widget displayed in the Preferences dialog.

This plugin has only one important method `create_widget()`. The various fields of the plugin control how it is categorized in the UI.

config_widget = None
Import path to module that contains a class named ConfigWidget which implements the ConfigWidget-Interface. Used by `create_widget()`.

category_order = 100
Where in the list of categories the `category` of this plugin should be.

name_order = 100
Where in the list of names in a category, the `gui_name` of this plugin should be

category = None
The category this plugin should be in

gui_category = None
The category name displayed to the user for this plugin

gui_name = None
The name displayed to the user for this plugin

icon = None
The icon for this plugin, should be an absolute path

description = None
The description used for tooltips and the like

create_widget(*parent=None*)
Create and return the actual Qt widget used for setting this group of preferences. The widget must implement the `calibre.gui2.preferences.ConfigWidgetInterface` (page 461).

The default implementation uses `config_widget` to instantiate the widget.

class `calibre.gui2.preferences.`**ConfigWidgetInterface**
This class defines the interface that all widgets displayed in the Preferences dialog must implement. See `ConfigWidgetBase` for a base class that implements this interface and defines various convenience methods as well.

changed_signal = None
This signal must be emitted whenever the user changes a value in this widget

supports_restoring_to_defaults = True
Set to True iff the `restore_to_defaults()` method is implemented.

restore_defaults_desc = u'Restore settings to default values. You have to click Apply to actually save the default se
The tooltip for the Restore to defaults button

restart_critical = False
If True the Preferences dialog will not allow the user to set any more preferences. Only has effect if `commit()` returns True.

genesis(*gui*)
Called once before the widget is displayed, should perform any necessary setup.

Parameters **gui** – The main calibre graphical user interface

initialize()
Should set all config values to their initial values (the values stored in the config files).

restore_defaults()
Should set all config values to their defaults.

commit()
Save any changed settings. Return True if the changes require a restart, False otherwise. Raise an `AbortCommit` exception to indicate that an error occurred. You are responsible for giving the user feedback about what the error is and how to correct it.

refresh_gui(*gui*)
Called once after this widget is committed. Responsible for causing the gui to reread any changed settings.

Note that by default the GUI re-initializes various elements anyway, so most widgets won't need to use this method.

class `calibre.gui2.preferences.`**`ConfigWidgetBase`**(*parent=None*)

Base class that contains code to easily add standard config widgets like checkboxes, combo boxes, text fields and so on. See the `register()` method.

This class automatically handles change notification, resetting to default, translation between gui objects and config objects, etc. for registered settings.

If your config widget inherits from this class but includes setting that are not registered, you should override the `ConfigWidgetInterface` methods and call the base class methods inside the overrides.

`register`(*name*, *config_obj*, *gui_name=None*, *choices=None*, *restart_required=False*, *empty_string_is_None=True*, *setting=<class 'calibre.gui2.preferences.Setting'>*)

Register a setting.

Parameters

- **name** – The setting name
- **config** – The config object that reads/writes the setting
- **gui_name** – The name of the GUI object that presents an interface to change the setting. By default it is assumed to be `'opt_' + name`.
- **choices** – If this setting is a multiple choice (combobox) based setting, the list of choices. The list is a list of two element tuples of the form: `[(gui name, value), ...]`
- **setting** – The class responsible for managing this setting. The default class handles almost all cases, so this param is rarely used.

1.8.2 Environment variables

- `CALIBRE_CONFIG_DIRECTORY` - sets the directory where configuration files are stored/read.
- `CALIBRE_TEMP_DIR` - sets the temporary directory used by calibre
- `CALIBRE_OVERRIDE_DATABASE_PATH` - allows you to specify the full path to metadata.db. Using this variable you can have metadata.db be in a location other than the library folder. Useful if your library folder is on a networked drive that does not support file locking.
- `CALIBRE_DEVELOP_FROM` - Used to run from a calibre development environment. See *Setting up a calibre development environment* (page 511).
- `CALIBRE_OVERRIDE_LANG` - Used to force the language used by the interface (ISO 639 language code)
- `CALIBRE_NO_NATIVE_FILEDIALOGS` - Causes calibre to not use native file dialogs for selecting files/directories.
- `SYSFS_PATH` - Use if sysfs is mounted somewhere other than /sys
- `http_proxy` - Used on linux to specify an HTTP proxy

1.8.3 Tweaks

Tweaks are small changes that you can specify to control various aspects of calibre's behavior. You can change them by going to Preferences->Advanced->Tweaks. The default values for the tweaks are reproduced below

```
#!/usr/bin/env python
# vim:fileencoding=UTF-8:ts=4:sw=4:sta:et:sts=4:ai
__license__   = 'GPL v3'
__copyright__ = '2010, Kovid Goyal <kovid@kovidgoyal.net>'
__docformat__ = 'restructuredtext en'

'''
Contains various tweaks that affect calibre behavior. Only edit this file if
you know what you are doing. If you delete this file, it will be recreated from
defaults.
'''

#: Auto increment series index
# The algorithm used to assign a book added to an existing series a series number.
# New series numbers assigned using this tweak are always integer values, except
# if a constant non-integer is specified.
# Possible values are:
# next - First available integer larger than the largest existing number
# first_free - First available integer larger than 0
# next_free - First available integer larger than the smallest existing number
# last_free - First available integer smaller than the largest existing number
#             Return largest existing + 1 if no free number is found
# const - Assign the number 1 always
# no_change - Do not change the series index
# a number - Assign that number always. The number is not in quotes. Note that
#            0.0 can be used here.
# Examples:
# series_index_auto_increment = 'next'
# series_index_auto_increment = 'next_free'
# series_index_auto_increment = 16.5
#
# Set the use_series_auto_increment_tweak_when_importing tweak to True to
# use the above values when importing/adding books. If this tweak is set to
# False (the default) then the series number will be set to 1 if it is not
# explicitly set to during the import. If set to True, then the
# series index will be set according to the series_index_auto_increment setting.
# Note that the use_series_auto_increment_tweak_when_importing tweak is used
# only when a value is not provided during import. If the importing regular
# expression produces a value for series_index, or if you are reading metadata
# from books and the import plugin produces a value, than that value will
# be used irrespective of the setting of the tweak.
series_index_auto_increment = 'next'
use_series_auto_increment_tweak_when_importing = False

#: Add separator after completing an author name
# Should the completion separator be append
# to the end of the completed text to
# automatically begin a new completion operation
# for authors.
# Can be either True or False
authors_completer_append_separator = False

#: Author sort name algorithm
# The algorithm used to copy author to author_sort
# Possible values are:
#  invert: use "fn ln" -> "ln, fn"
#  copy  : copy author to author_sort without modification
#  comma : use 'copy' if there is a ',' in the name, otherwise use 'invert'
```

```
#  nocomma : "fn ln" -> "ln fn" (without the comma)
# When this tweak is changed, the author_sort values stored with each author
# must be recomputed by right-clicking on an author in the left-hand tags pane,
# selecting 'manage authors', and pressing 'Recalculate all author sort values'.
# The author name suffixes are words that are ignored when they occur at the
# end of an author name. The case of the suffix is ignored and trailing
# periods are automatically handled. The same is true for prefixes.
# The author name copy words are a set of words which if they occur in an
# author name cause the automatically generated author sort string to be
# identical to the author name. This means that the sort for a string like Acme
# Inc. will be Acme Inc. instead of Inc., Acme
author_sort_copy_method = 'comma'
author_name_suffixes = ('Jr', 'Sr', 'Inc', 'Ph.D', 'Phd',
                        'MD', 'M.D', 'I', 'II', 'III', 'IV',
                        'Junior', 'Senior')
author_name_prefixes = ('Mr', 'Mrs', 'Ms', 'Dr', 'Prof')
author_name_copywords = ('Corporation', 'Company', 'Co.', 'Agency', 'Council',
        'Committee', 'Inc.', 'Institute', 'Society', 'Club', 'Team')

#: Splitting multiple author names
# By default, calibre splits a string containing multiple author names on
# ampersands and the words "and" and "with". You can customize the splitting
# by changing the regular expression below. Strings are split on whatever the
# specified regular expression matches.
# Default: r'(?i),?\s+(and|with)\s+'
authors_split_regex = r'(?i),?\s+(and|with)\s+'

#: Use author sort in Tag Browser
# Set which author field to display in the tags pane (the list of authors,
# series, publishers etc on the left hand side). The choices are author and
# author_sort. This tweak affects only what is displayed under the authors
# category in the tags pane and content server. Please note that if you set this
# to author_sort, it is very possible to see duplicate names in the list because
# although it is guaranteed that author names are unique, there is no such
# guarantee for author_sort values. Showing duplicates won't break anything, but
# it could lead to some confusion. When using 'author_sort', the tooltip will
# show the author's name.
# Examples:
#   categories_use_field_for_author_name = 'author'
#   categories_use_field_for_author_name = 'author_sort'
categories_use_field_for_author_name = 'author'

#: Control partitioning of Tag Browser
# When partitioning the tags browser, the format of the subcategory label is
# controlled by a template: categories_collapsed_name_template if sorting by
# name, categories_collapsed_rating_template if sorting by average rating, and
# categories_collapsed_popularity_template if sorting by popularity. There are
# two variables available to the template: first and last. The variable 'first'
# is the initial item in the subcategory, and the variable 'last' is the final
# item in the subcategory. Both variables are 'objects'; they each have multiple
# values that are obtained by using a suffix. For example, first.name for an
# author category will be the name of the author. The sub-values available are:
#  name: the printable name of the item
#  count: the number of books that references this item
#  avg_rating: the average rating of all the books referencing this item
#  sort: the sort value. For authors, this is the author_sort for that author
#  category: the category (e.g., authors, series) that the item is in.
# Note that the "r'" in front of the { is necessary if there are backslashes
```

```
# (\ characters) in the template. It doesn't hurt anything to leave it there
# even if there aren't any backslashes.
categories_collapsed_name_template = r'{first.sort:shorten(4,,0)} - {last.sort:shorten(4,,0)}'
categories_collapsed_rating_template = r'{first.avg_rating:4.2f:ifempty(0)} - {last.avg_rating:4.2f:i
categories_collapsed_popularity_template = r'{first.count:d} - {last.count:d}'

#: Control order of categories in the tag browser
# Change the following dict to change the order that categories are displayed in
# the tag browser. Items are named using their lookup name, and will be sorted
# using the number supplied. The lookup name '*' stands for all names that
# otherwise do not appear. Two names with the same value will be sorted
# using the default order; the one used when the dict is empty.
# Example: tag_browser_category_order = {'series':1, 'tags':2, '*':3}
# resulting in the order series, tags, then everything else in default order.
tag_browser_category_order = {'*':1}


#: Specify columns to sort the booklist by on startup
# Provide a set of columns to be sorted on when calibre starts
#  The argument is None if saved sort history is to be used
#  otherwise it is a list of column,order pairs. Column is the
#  lookup/search name, found using the tooltip for the column
#  Order is 0 for ascending, 1 for descending
# For example, set it to [('authors',0),('title',0)] to sort by
# title within authors.
sort_columns_at_startup = None

#: Control how dates are displayed
# Format to be used for publication date and the timestamp (date).
#  A string controlling how the publication date is displayed in the GUI
#  d     the day as number without a leading zero (1 to 31)
#  dd    the day as number with a leading zero (01 to 31)
#  ddd   the abbreviated localized day name (e.g. 'Mon' to 'Sun').
#  dddd  the long localized day name (e.g. 'Monday' to 'Qt::Sunday').
#  M     the month as number without a leading zero (1-12)
#  MM    the month as number with a leading zero (01-12)
#  MMM   the abbreviated localized month name (e.g. 'Jan' to 'Dec').
#  MMMM  the long localized month name (e.g. 'January' to 'December').
#  yy    the year as two digit number (00-99)
#  yyyy  the year as four digit number
#  h     the hours without a leading 0 (0 to 11 or 0 to 23, depending on am/pm) '
#  hh    the hours with a leading 0 (00 to 11 or 00 to 23, depending on am/pm) '
#  m     the minutes without a leading 0 (0 to 59) '
#  mm    the minutes with a leading 0 (00 to 59) '
#  s     the seconds without a leading 0 (0 to 59) '
#  ss    the seconds with a leading 0 (00 to 59) '
#  ap    use a 12-hour clock instead of a 24-hour clock, with "ap"
#        replaced by the localized string for am or pm '
#  AP    use a 12-hour clock instead of a 24-hour clock, with "AP"
#        replaced by the localized string for AM or PM '
#  iso   the date with time and timezone. Must be the only format present
#  For example, given the date of 9 Jan 2010, the following formats show
#  MMM yyyy ==> Jan 2010    yyyy ==> 2010       dd MMM yyyy ==> 09 Jan 2010
#  MM/yyyy ==> 01/2010      d/M/yy ==> 9/1/10   yy ==> 10
# publication default if not set: MMM yyyy
# timestamp default if not set: dd MMM yyyy
# last_modified_display_format if not set: dd MMM yyyy
gui_pubdate_display_format = 'MMM yyyy'
```

```
gui_timestamp_display_format = 'dd MMM yyyy'
gui_last_modified_display_format = 'dd MMM yyyy'

#: Control sorting of titles and series in the library display
# Control title and series sorting in the library view. If set to
# 'library_order', the title sort field will be used instead of the title.
# Unless you have manually edited the title sort field, leading articles such as
# The and A will be ignored. If set to 'strictly_alphabetic', the titles will be
# sorted as-is (sort by title instead of title sort). For example, with
# library_order, The Client will sort under 'C'. With strictly_alphabetic, the
# book will sort under 'T'.
# This flag affects Calibre's library display. It has no effect on devices. In
# addition, titles for books added before changing the flag will retain their
# order until the title is edited. Double-clicking on a title and hitting return
# without changing anything is sufficient to change the sort.
title_series_sorting = 'library_order'

#: Control formatting of title and series when used in templates
# Control how title and series names are formatted when saving to disk/sending
# to device. The behavior depends on the field being processed. If processing
# title, then if this tweak is set to 'library_order', the title will be
# replaced with title_sort. If it is set to 'strictly_alphabetic', then the
# title will not be changed. If processing series, then if set to
# 'library_order', articles such as 'The' and 'An' will be moved to the end. If
# set to 'strictly_alphabetic', the series will be sent without change.
# For example, if the tweak is set to library_order, "The Lord of the Rings"
# will become "Lord of the Rings, The". If the tweak is set to
# strictly_alphabetic, it would remain "The Lord of the Rings". Note that the
# formatter function raw_field will return the base value for title and
# series regardless of the setting of this tweak.
save_template_title_series_sorting = 'library_order'

#: Set the list of words considered to be "articles" for sort strings
# Set the list of words that are to be considered 'articles' when computing the
# title sort strings. The articles differ by language. By default, calibre uses
# a combination of articles from English and whatever language the calibre user
# interface is set to. In addition, in some contexts where the book language is
# available, the language of the book is used. You can change the list of
# articles for a given language or add a new language by editing
# per_language_title_sort_articles. To tell calibre to use a language other
# than the user interface language, set, default_language_for_title_sort. For
# example, to use German, set it to 'deu'. A value of None means the user
# interface language is used. The setting title_sort_articles is ignored
# (present only for legacy reasons).
per_language_title_sort_articles = {
        # English
        'eng'  : (r'A\s+', r'The\s+', r'An\s+'),
        # Spanish
        'spa'  : (r'El\s+', r'La\s+', r'Lo\s+', r'Los\s+', r'Las\s+', r'Un\s+',
                  r'Una\s+', r'Unos\s+', r'Unas\s+'),
        # French
        'fra'  : (r'Le\s+', r'La\s+', r"L'", r'Les\s+', r'Un\s+', r'Une\s+',
                  r'Des\s+', r'De\s+La\s+', r'De\s+', r"D'"),
        # Italian
        'ita'  : (r'Lo\s+', r'Il\s+', r"L'", r'La\s+', r'Gli\s+', r'I\s+',
                  r'Le\s+', ),
        # Portuguese
        'por'  : (r'A\s+', r'O\s+', r'Os\s+', r'As\s+', r'Um\s+', r'Uns\s+',
```

```
                r'Uma\s+', r'Umas\s+', ),
        # Romanian
        'ron'  : (r'Un\s+', r'O\s+', r'Nişte\s+', ),
        # German
        'deu'  : (r'Der\s+', r'Die\s+', r'Das\s+', r'Den\s+', r'Ein\s+',
                  r'Eine\s+', r'Einen\s+', r'Dem\s+', r'Des\s+', r'Einem\s+',
                  r'Eines\s+'),
        # Dutch
        'nld'  : (r'De\s+', r'Het\s+', r'Een\s+', r"'n\s+", r"'s\s+", r'Ene\s+',
                  r'Ener\s+', r'Enes\s+', r'Den\s+', r'Der\s+', r'Des\s+',
                  r"'t\s+"),
        # Swedish
        'swe'  : (r'En\s+', r'Ett\s+', r'Det\s+', r'Den\s+', r'De\s+', ),
        # Turkish
        'tur'  : (r'Bir\s+', ),
        # Afrikaans
        'afr'  : (r"'n\s+", r'Die\s+', ),
        # Greek
        'ell'  : (r'O\s+', r'I\s+', r'To\s+', r'Ta\s+', r'Tus\s+', r'Tis\s+',
                  r"'Enas\s+", r"'Mia\s+", r"'Ena\s+", r"'Enan\s+", ),
        # Hungarian
        'hun'  : (r'A\s+', 'Az\s+', 'Egy\s+',),
}
default_language_for_title_sort = None
title_sort_articles=r'^(A|The|An)\s+'

#: Specify a folder calibre should connect to at startup
# Specify a folder that calibre should connect to at startup using
# connect_to_folder. This must be a full path to the folder. If the folder does
# not exist when calibre starts, it is ignored. If there are '\' characters in
# the path (such as in Windows paths), you must double them.
# Examples:
#     auto_connect_to_folder = 'C:\\Users\\someone\\Desktop\\testlib'
#     auto_connect_to_folder = '/home/dropbox/My Dropbox/someone/library'
auto_connect_to_folder = ''

#: Specify renaming rules for SONY collections
# Specify renaming rules for sony collections. This tweak is only applicable if
# metadata management is set to automatic. Collections on Sonys are named
# depending upon whether the field is standard or custom. A collection derived
# from a standard field is named for the value in that field. For example, if
# the standard 'series' column contains the value 'Darkover', then the
# collection name is 'Darkover'. A collection derived from a custom field will
# have the name of the field added to the value. For example, if a custom series
# column named 'My Series' contains the name 'Darkover', then the collection
# will by default be named 'Darkover (My Series)'. For purposes of this
# documentation, 'Darkover' is called the value and 'My Series' is called the
# category. If two books have fields that generate the same collection name,
# then both books will be in that collection.
# This set of tweaks lets you specify for a standard or custom field how
# the collections are to be named. You can use it to add a description to a
# standard field, for example 'Foo (Tag)' instead of the 'Foo'. You can also use
# it to force multiple fields to end up in the same collection. For example, you
# could force the values in 'series', '#my_series_1', and '#my_series_2' to
# appear in collections named 'some_value (Series)', thereby merging all of the
# fields into one set of collections.
# There are two related tweaks. The first determines the category name to use
# for a metadata field.  The second is a template, used to determines how the
```

```
# value and category are combined to create the collection name.
# The syntax of the first tweak, sony_collection_renaming_rules, is:
# {'field_lookup_name':'category_name_to_use', 'lookup_name':'name', ...}
# The second tweak, sony_collection_name_template, is a template. It uses the
# same template language as plugboards and save templates. This tweak controls
# how the value and category are combined together to make the collection name.
# The only two fields available are {category} and {value}. The {value} field is
# never empty. The {category} field can be empty. The default is to put the
# value first, then the category enclosed in parentheses, it isn't empty:
# '{value} {category:|(|)}'
# Examples: The first three examples assume that the second tweak
# has not been changed.
# 1: I want three series columns to be merged into one set of collections. The
# column lookup names are 'series', '#series_1' and '#series_2'. I want nothing
# in the parenthesis. The value to use in the tweak value would be:
#    sony_collection_renaming_rules={'series':'', '#series_1':'', '#series_2':''}
# 2: I want the word '(Series)' to appear on collections made from series, and
# the word '(Tag)' to appear on collections made from tags. Use:
#    sony_collection_renaming_rules={'series':'Series', 'tags':'Tag'}
# 3: I want 'series' and '#myseries' to be merged, and for the collection name
# to have '(Series)' appended. The renaming rule is:
#    sony_collection_renaming_rules={'series':'Series', '#myseries':'Series'}
# 4: Same as example 2, but instead of having the category name in parentheses
# and appended to the value, I want it prepended and separated by a colon, such
# as in Series: Darkover. I must change the template used to format the category name
# The resulting two tweaks are:
#    sony_collection_renaming_rules={'series':'Series', 'tags':'Tag'}
#    sony_collection_name_template='{category:||: }{value}'
sony_collection_renaming_rules={}
sony_collection_name_template='{value}{category:| (|)}'

#: Specify how SONY collections are sorted
# Specify how sony collections are sorted. This tweak is only applicable if
# metadata management is set to automatic. You can indicate which metadata is to
# be used to sort on a collection-by-collection basis. The format of the tweak
# is a list of metadata fields from which collections are made, followed by the
# name of the metadata field containing the sort value.
# Example: The following indicates that collections built from pubdate and tags
# are to be sorted by the value in the custom column '#mydate', that collections
# built from 'series' are to be sorted by 'series_index', and that all other
# collections are to be sorted by title. If a collection metadata field is not
# named, then if it is a series- based collection it is sorted by series order,
# otherwise it is sorted by title order.
# [(['pubdate', 'tags'],'#mydate'), (['series'],'series_index'), (['*'], 'title')]
# Note that the bracketing and parentheses are required. The syntax is
# [ ( [list of fields], sort field ) , ( [ list of fields ] , sort field ) ]
# Default: empty (no rules), so no collection attributes are named.
sony_collection_sorting_rules = []

#: Control how tags are applied when copying books to another library
# Set this to True to ensure that tags in 'Tags to add when adding
# a book' are added when copying books to another library
add_new_book_tags_when_importing_books = False

#: Set the maximum number of tags to show per book in the content server
max_content_server_tags_shown=5

#: Set custom metadata fields that the content server will or will not display.
```

```
# content_server_will_display is a list of custom fields to be displayed.
# content_server_wont_display is a list of custom fields not to be displayed.
# wont_display has priority over will_display.
# The special value '*' means all custom fields. The value [] means no entries.
# Defaults:
#    content_server_will_display = ['*']
#    content_server_wont_display = []
# Examples:
# To display only the custom fields #mytags and #genre:
#   content_server_will_display = ['#mytags', '#genre']
#   content_server_wont_display = []
# To display all fields except #mycomments:
#   content_server_will_display = ['*']
#   content_server_wont_display['#mycomments']
content_server_will_display = ['*']
content_server_wont_display = []

#: Set the maximum number of sort 'levels'
# Set the maximum number of sort 'levels' that calibre will use to resort the
# library after certain operations such as searches or device insertion. Each
# sort level adds a performance penalty. If the database is large (thousands of
# books) the penalty might be noticeable. If you are not concerned about multi-
# level sorts, and if you are seeing a slowdown, reduce the value of this tweak.
maximum_resort_levels = 5

#: Choose whether dates are sorted using visible fields
# Date values contain both a date and a time. When sorted, all the fields are
# used, regardless of what is displayed. Set this tweak to True to use only
# the fields that are being displayed.
sort_dates_using_visible_fields = False

#: Specify which font to use when generating a default cover or masthead
# Absolute path to .ttf font files to use as the fonts for the title, author
# and footer when generating a default cover or masthead image. Useful if the
# default font (Liberation Serif) does not contain glyphs for the language of
# the books in your library.
generate_cover_title_font = None
generate_cover_foot_font = None

#: Control behavior of the book list
# You can control the behavior of doubleclicks on the books list.
# Choices: open_viewer, do_nothing,
# edit_cell, edit_metadata. Selecting edit_metadata has the side effect of
# disabling editing a field using a single click.
# Default: open_viewer.
# Example: doubleclick_on_library_view = 'do_nothing'
# You can also control whether the book list scrolls horizontal per column or
# per pixel. Default is per column.
doubleclick_on_library_view = 'open_viewer'
horizontal_scrolling_per_column = True

#: Language to use when sorting.
# Setting this tweak will force sorting to use the
# collating order for the specified language. This might be useful if you run
# calibre in English but want sorting to work in the language where you live.
# Set the tweak to the desired ISO 639-1 language code, in lower case.
# You can find the list of supported locales at
# http://publib.boulder.ibm.com/infocenter/iseries/v5r3/topic/nls/rbagsicusortsequencetables.htm
```

```
# Default: locale_for_sorting = '' -- use the language calibre displays in
# Example: locale_for_sorting = 'fr' -- sort using French rules.
# Example: locale_for_sorting = 'nb' -- sort using Norwegian rules.
locale_for_sorting =  ''

#: Number of columns for custom metadata in the edit metadata dialog
# Set whether to use one or two columns for custom metadata when editing
# metadata  one book at a time. If True, then the fields are laid out using two
# columns. If False, one column is used.
metadata_single_use_2_cols_for_custom_fields = True

#: Order of custom column(s) in edit metadata
# Controls the order that custom columns are listed in edit metadata single
# and bulk. The columns listed in the tweak are displayed first and in the
# order provided. Any columns not listed are dislayed after the listed ones,
# in alphabetical order. Do note that this tweak does not change the size of
# the edit widgets. Putting comments widgets in this list may result in some
# odd widget spacing when using two-column mode.
# Enter a comma-separated list of custom field lookup names, as in
# metadata_edit_custom_column_order = ['#genre', '#mytags', '#etc']
metadata_edit_custom_column_order = []

#: The number of seconds to wait before sending emails
# The number of seconds to wait before sending emails when using a
# public email server like gmail or hotmail. Default is: 5 minutes
# Setting it to lower may cause the server's SPAM controls to kick in,
# making email sending fail. Changes will take effect only after a restart of
# calibre.
public_smtp_relay_delay = 301

#: The maximum width and height for covers saved in the calibre library
# All covers in the calibre library will be resized, preserving aspect ratio,
# to fit within this size. This is to prevent slowdowns caused by extremely
# large covers
maximum_cover_size = (1200, 1600)

#: Where to send downloaded news
# When automatically sending downloaded news to a connected device, calibre
# will by default send it to the main memory. By changing this tweak, you can
# control where it is sent. Valid values are "main", "carda", "cardb". Note
# that if there isn't enough free space available on the location you choose,
# the files will be sent to the location with the most free space.
send_news_to_device_location = "main"

#: What interfaces should the content server listen on
# By default, the calibre content server listens on '0.0.0.0' which means that it
# accepts IPv4 connections on all interfaces. You can change this to, for
# example, '127.0.0.1' to only listen for connections from the local machine, or
# to '::' to listen to all incoming IPv6 and IPv4 connections (this may not
# work on all operating systems)
server_listen_on = '0.0.0.0'

#: Unified toolbar on OS X
# If you enable this option and restart calibre, the toolbar will be 'unified'
# with the titlebar as is normal for OS X applications. However, doing this has
# various bugs, for instance the minimum width of the toolbar becomes twice
# what it should be and it causes other random bugs on some systems, so turn it
# on at your own risk!
```

```
unified_title_toolbar_on_osx = False

#: Save original file when converting/polishing from same format to same format
# When calibre does a conversion from the same format to the same format, for
# example, from EPUB to EPUB, the original file is saved, so that in case the
# conversion is poor, you can tweak the settings and run it again. By setting
# this to False you can prevent calibre from saving the original file.
# Similarly, by setting save_original_format_when_polishing to False you can
# prevent calibre from saving the original file when polishing.
save_original_format = True
save_original_format_when_polishing = True

#: Number of recently viewed books to show
# Right-clicking the View button shows a list of recently viewed books. Control
# how many should be shown, here.
gui_view_history_size = 15

#: Change the font size of book details in the interface
# Change the font size at which book details are rendered in the side panel and
# comments are rendered in the metadata edit dialog. Set it to a positive or
# negative number to increase or decrease the font size.
change_book_details_font_size_by = 0

#: Compile General Program Mode templates to Python
# Compiled general program mode templates are significantly faster than
# interpreted templates. Setting this tweak to True causes calibre to compile
# (in most cases) general program mode templates. Setting it to False causes
# calibre to use the old behavior -- interpreting the templates. Set the tweak
# to False if some compiled templates produce incorrect values.
# Default:    compile_gpm_templates = True
# No compile: compile_gpm_templates = False
compile_gpm_templates = True

#: What format to default to when using the Tweak feature
# The Tweak feature of calibre allows direct editing of a book format.
# If multiple formats are available, calibre will offer you a choice
# of formats, defaulting to your preferred output format if it is available.
# Set this tweak to a specific value of 'EPUB' or 'AZW3' to always default
# to that format rather than your output format preference.
# Set to a value of 'remember' to use whichever format you chose last time you
# used the Tweak feature.
# Examples:
#   default_tweak_format = None       (Use output format)
#   default_tweak_format = 'EPUB'
#   default_tweak_format = 'remember'
default_tweak_format = None

#: Do not preselect a completion when editing authors/tags/series/etc.
# This means that you can make changes and press Enter and your changes will
# not be overwritten by a matching completion. However, if you wish to use the
# completions you will now have to press Tab to select one before pressing
# Enter. Which technique you prefer will depend on the state of metadata in
# your library and your personal editing style.
preselect_first_completion = False

#: Recognize numbers inside text when sorting
# This means that when sorting on text fields like title the text "Book 2"
# will sort before the text "Book 100". If you want this behavior, set
```

```
# numeric_collation = True note that doing so will cause problems with text
# that starts with numbers and is a little slower.
numeric_collation = False

#: Sort the list of libraries alphabetically
# The list of libraries in the Copy to Library and Quick Switch menus are
# normally sorted by most used. However, if there are more than a certain
# number of such libraries, the sorting becomes alphabetic. You can set that
# number here. The default is ten libraries.
many_libraries = 10
```

1.8.4 Overriding icons, templates, et cetera

calibre allows you to override the static resources, like icons, templates, javascript, etc. with customized versions that you like. All static resources are stored in the resources sub-folder of the calibre install location. On Windows, this is usually `C:/Program Files/Calibre2/resources`. On OS X, `/Applications/calibre.app/Contents/Resources/resources/`. On linux, if you are using the binary installer from the calibre website it will be `/opt/calibre/resources`. These paths can change depending on where you choose to install calibre.

You should not change the files in this resources folder, as your changes will get overwritten the next time you update calibre. Instead, go to *Preferences->Advanced->Miscellaneous* and click *Open calibre configuration directory*. In this configuration directory, create a sub-folder called resources and place the files you want to override in it. Place the files in the appropriate sub folders, for example place images in `resources/images`, etc. calibre will automatically use your custom file in preference to the built-in one the next time it is started.

For example, if you wanted to change the icon for the *Remove books* action, you would first look in the built-in resources folder and see that the relevant file is `resources/images/trash.png`. Assuming you have an alternate icon in PNG format called `mytrash.png` you would save it in the configuration directory as `resources/images/trash.png`. All the icons used by the calibre user interface are in `resources/images` and its sub-folders.

1.8.5 Customizing calibre with plugins

calibre has a very modular design. Almost all functionality in calibre comes in the form of plugins. Plugins are used for conversion, for downloading news (though these are called recipes), for various components of the user interface, to connect to different devices, to process files when adding them to calibre and so on. You can get a complete list of all the built-in plugins in calibre by going to *Preferences->Plugins*.

You can write your own plugins to customize and extend the behavior of calibre. The plugin architecture in calibre is very simple, see the tutorial *Writing your own plugins to extend calibre's functionality* (page 423).

1.9 Command Line Interface



On OS X you have to go to Preferences->Advanced->Miscellaneous and click install command line tools to make the command line tools available. On other platforms, just start a terminal and type the command.

1.9.1 Documented Commands

calibre

```
calibre [opts] [path_to_ebook]
```

Launch the main **calibre** Graphical User Interface and optionally add the ebook at path_to_ebook to the database.

Whenever you pass arguments to **calibre** that have spaces in them, enclose the arguments in quotation marks.

[options]

-help, -h
show this help message and exit

-ignore-plugins
Ignore custom plugins, useful if you installed a plugin that is preventing calibre from starting

-no-update-check
Do not check for updates

-shutdown-running-calibre, -s
Cause a running calibre instance, if any, to be shutdown. Note that if there are running jobs, they will be silently aborted, so use with care.

-start-in-tray
Start minimized to system tray.

-verbose, -v
Log debugging information to console

-version
show program's version number and exit

-with-library
Use the library located at the specified path.

calibre-customize

```
calibre-customize options
```

Customize calibre by loading external plugins.

Whenever you pass arguments to **calibre-customize** that have spaces in them, enclose the arguments in quotation marks.

[options]

-add-plugin, -a
Add a plugin by specifying the path to the zip file containing it.

-build-plugin, -b
For plugin developers: Path to the directory where you are developing the plugin. This command will automatically zip up the plugin and update it in calibre.

-customize-plugin
Customize plugin. Specify name of plugin and customization string separated by a comma.

-disable-plugin
Disable the named plugin

-enable-plugin
Enable the named plugin

-help, -h
show this help message and exit

-list-plugins, -l
List all installed plugins

-remove-plugin, -r
Remove a custom plugin by name. Has no effect on builtin plugins

-version
show program's version number and exit

calibre-debug

```
calibre-debug [options]
```

Various command line interfaces useful for debugging calibre. With no options, this command starts an embedded python interpreter. You can also run the main calibre GUI and the calibre viewer in debug mode.

It also contains interfaces to various bits of calibre that do not have dedicated command line tools, such as font subsetting, tweaking ebooks and so on.

You can also use **calibre-debug** to run standalone scripts. To do that use it like this:

> **calibre-debug** mysrcipt.py – –option1 –option2 file1 file2 ...

Everything after the – is passed to the script.

Whenever you pass arguments to **calibre-debug** that have spaces in them, enclose the arguments in quotation marks.

[options]

-add-simple-plugin
Add a simple plugin (i.e. a plugin that consists of only a .py file), by specifying the path to the py file containing the plugin code.

-command, -c
Run python code.

-debug-device-driver, -d
Debug device detection

-exec-file, -e
Run the python code in file.

-gui, -g
Run the GUI with debugging enabled. Debug output is printed to stdout and stderr.

-gui-debug
Run the GUI with a debug console, logging to the specified path. For internal use only, use the -g option to run the GUI in debug mode

-help, -h
show this help message and exit

-inspect-mobi, -m
Inspect the MOBI file(s) at the specified path(s)

-paths
Output the paths necessary to setup the calibre environment

-py-console, -p
Run python console

-reinitialize-db
Re-initialize the sqlite calibre database at the specified path. Useful to recover from db corruption. You can also specify the path to an SQL dump which will be used instead of trying to dump the database. This can be useful when dumping fails, but dumping with sqlite3 works.

-show-gui-debug
Display the specified log file. For internal use only.

-shutdown-running-calibre, -s
Cause a running calibre instance, if any, to be shutdown. Note that if there are running jobs, they will be silently aborted, so use with care.

-subset-font, -f
Subset the specified font

-test-build
Test binary modules in build

-tweak-book, -t
Tweak the book (exports the book as a collection of HTML files and metadata, which you can edit using standard HTML editing tools, and then rebuilds the file from the edited HTML. Makes no additional changes to the HTML, unlike a full calibre conversion).

-version
show program's version number and exit

-viewer, -w
Run the ebook viewer

calibre-server

```
calibre-server [options]
```

Start the calibre content server. The calibre content server exposes your calibre library over the internet. The default interface allows you to browse you calibre library by categories. You can also access an interface optimized for mobile browsers at /mobile and an OPDS based interface for use with reading applications at /opds.

The OPDS interface is advertised via BonJour automatically.

Whenever you pass arguments to **calibre-server** that have spaces in them, enclose the arguments in quotation marks.

[options]

-auto-reload
Auto reload server when source code changes. May not work in all environments.

-daemonize
Run process in background as a daemon. No effect on windows.

-develop
Development mode. Server automatically restarts on file changes and serves code files (html, css, js) from the file system instead of calibre's resource system.

-help, -h
show this help message and exit

-max-cover
The maximum size for displayed covers. Default is '600x800'.

-max-opds-items
The maximum number of matches to return per OPDS query. This affects Stanza, WordPlayer, etc. integration.

-max-opds-ungrouped-items
Group items in categories such as author/tags by first letter when there are more than this number of items. Default: 100. Set to a large number to disable grouping.

-password
Set a password to restrict access. By default access is unrestricted.

-pidfile
Write process PID to the specified file

-port, -p
The port on which to listen. Default is 8080

-restriction
Specifies a restriction to be used for this invocation. This option overrides any per-library settings specified in the GUI

-thread-pool
The max number of worker threads to use. Default is 30

-timeout, -t
The server timeout in seconds. Default is 120

-url-prefix
Prefix to prepend to all URLs. Useful for reverseproxying to this server from Apache/nginx/etc.

-username
Username for access. By default, it is: 'calibre'

-version
show program's version number and exit

-with-library
Path to the library folder to serve with the content server

calibre-smtp

```
calibre-smtp [options] [from to text]
```

Send mail using the SMTP protocol. **calibre-smtp** has two modes of operation. In the compose mode you specify from to and text and these are used to build and send an email message. In the filter mode, **calibre-smtp** reads a complete email message from STDIN and sends it.

text is the body of the email message. If text is not specified, a complete email message is read from STDIN. from is the email address of the sender and to is the email address of the recipient. When a complete email is read from STDIN, from and to are only used in the SMTP negotiation, the message headers are not modified.

Whenever you pass arguments to **calibre-smtp** that have spaces in them, enclose the arguments in quotation marks.

[options]

-fork, -f
Fork and deliver message in background. If you use this option, you should also use –outbox to handle delivery failures.

-help, -h
show this help message and exit

-localhost, -l
Host name of localhost. Used when connecting to SMTP server.

-outbox, -o
Path to maildir folder to store failed email messages in.

-timeout, -t
Timeout for connection

-verbose, -v
Be more verbose

-version
show program's version number and exit

COMPOSE MAIL Options to compose an email. Ignored if text is not specified

-attachment, -a
File to attach to the email

-subject, -s
Subject of the email

SMTP RELAY Options to use an SMTP relay server to send mail. calibre will try to send the email directly unless –relay is specified.

-encryption-method, -e
Encryption method to use when connecting to relay. Choices are TLS, SSL and NONE. Default is TLS. WARNING: Choosing NONE is highly insecure

-password, -p
Password for relay

-port
Port to connect to on relay server. Default is to use 465 if encryption method is SSL and 25 otherwise.

-relay, -r
An SMTP relay server to use to send mail.

-username, -u
Username for relay

calibredb

```
calibredb command [options] [arguments]
```

calibredb is the command line interface to the calibre database. It has several sub-commands, documented below:

- *list* (page 478)

- *add* (page 479)
- *remove* (page 479)
- *add_format* (page 480)
- *remove_format* (page 480)
- *show_metadata* (page 480)
- *set_metadata* (page 481)
- *export* (page 481)
- *catalog* (page 482)
- *saved_searches* (page 482)
- *add_custom_column* (page 483)
- *custom_columns* (page 483)
- *remove_custom_column* (page 483)
- *set_custom* (page 484)
- *restore_database* (page 484)
- *check_library* (page 484)
- *list_categories* (page 485)
- *backup_metadata* (page 485)

Global options

-dont-notify-gui
Do not notify the running calibre GUI (if any) that the database has changed. Use with care, as it can lead to database corruption!

-library-path
Path to the calibre library. Default is to use the path stored in the settings.

list

```
calibredb list [options]
```

List the books available in the calibre database.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-ascending
Sort results in ascending order

-fields, -f
The fields to display when listing books in the database. Should be a comma separated list of fields. Available fields: author_sort,authors,comments,cover,formats,identifiers,isbn,last_modified,pubdate,publisher,rating,series,series_index,size
Default: title,authors. The special field "all" can be used to select all fields. Only has effect in the text output format.

-help, -h
show this help message and exit

-line-width, -w
The maximum width of a single line in the output. Defaults to detecting screen size.

-prefix
The prefix for all file paths. Default is the absolute path to the library folder.

-search, -s
Filter the results by the search query. For the format of the search query, please see the search related documentation in the User Manual. Default is to do no filtering.

-separator
The string used to separate fields. Default is a space.

-sort-by
The field by which to sort the results. Available fields: publisher,series_index,formats,isbn,uuid,pubdate,rating,series,timestamp,author_sort,cover,comments,identifiers,last_modified,author
Default: None

-version
show program's version number and exit

add

```
calibredb add [options] file1 file2 file3 ...
```

Add the specified files as books to the database. You can also specify directories, see the directory related options below.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-authors, -a
Set the authors of the added book(s)

-cover, -c
Path to the cover to use for the added book

-duplicates, -d
Add books to database even if they already exist. Comparison is done based on book titles.

-empty, -e
Add an empty book (a book with no formats)

-help, -h
show this help message and exit

-isbn, -i
Set the ISBN of the added book(s)

-one-book-per-directory, -1
Assume that each directory has only a single logical book and that all files in it are different e-book formats of that book

-recurse, -r
Process directories recursively

-series, -s
Set the series of the added book(s)

-series-index, -S
Set the series number of the added book(s)

-tags, -T
Set the tags of the added book(s)

-title, -t
Set the title of the added book(s)

-version
show program's version number and exit

remove

```
calibredb remove ids
```

Remove the books identified by ids from the database. ids should be a comma separated list of id numbers (you can get id numbers by using the list command). For example, 23,34,57-85 (when specifying a range, the last number in the range is not included).

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-help, -h
show this help message and exit

-version
show program's version number and exit

add_format

```
calibredb add_format [options] id ebook_file
```

Add the ebook in ebook_file to the available formats for the logical book identified by id. You can get id by using the list command. If the format already exists, it is replaced.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-help, -h
show this help message and exit

-version
show program's version number and exit

remove_format

```
calibredb remove_format [options] id fmt
```

Remove the format fmt from the logical book identified by id. You can get id by using the list command. fmt should be a file extension like LRF or TXT or EPUB. If the logical book does not have fmt available, do nothing.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-help, -h
show this help message and exit

-version
show program's version number and exit

show_metadata

```
calibredb show_metadata [options] id
```

Show the metadata stored in the calibre database for the book identified by id. id is an id number from the list command.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-as-opf
Print metadata in OPF form (XML)

-help, -h
show this help message and exit

-version
show program's version number and exit

set_metadata

```
calibredb set_metadata [options] id /path/to/metadata.opf
```

Set the metadata stored in the calibre database for the book identified by id from the OPF file metadata.opf. id is an id number from the list command. You can get a quick feel for the OPF format by using the –as-opf switch to the show_metadata command. You can also set the metadata of individual fields with the –field option.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-help, -h
show this help message and exit

-version
show program's version number and exit

export

```
calibredb export [options] ids
```

Export the books specified by ids (a comma separated list) to the filesystem. The **export** operation saves all formats of the book, its cover and metadata (in an opf file). You can get id numbers from the list command.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-all
Export all books in database, ignoring the list of ids.

-dont-asciiize
Normally, calibre will convert all non English characters into English equivalents for the file names. WARNING: If you turn this off, you may experience errors when saving, depending on how well the filesystem you are saving to supports unicode. Specifying this switch will turn this behavior off.

-dont-save-cover
Normally, calibre will save the cover in a separate file along with the actual e-book file(s). Specifying this switch will turn this behavior off.

-dont-update-metadata
Normally, calibre will update the metadata in the saved files from what is in the calibre library. Makes saving to disk slower. Specifying this switch will turn this behavior off.

–dont-write-opf
Normally, calibre will write the metadata into a separate OPF file along with the actual e-book files. Specifying this switch will turn this behavior off.

–formats
Comma separated list of formats to save for each book. By default all available formats are saved.

–help, -h
show this help message and exit

–replace-whitespace
Replace whitespace with underscores.

–single-dir
Export all books into a single directory

–template
The template to control the filename and directory structure of the saved files. Default is "{author_sort}/{title}/{title} - {authors}" which will save books into a per-author subdirectory with filenames containing title and author. Available controls are: {publisher, series_index, isbn, pubdate, rating, series, author_sort, authors, last_modified, timestamp, title, id, tags}

–timefmt
The format in which to display dates. %d - day, %b - month, %m - month number, %Y - year. Default is: %b, %Y

–to-dir
Export books to the specified directory. Default is .

–to-lowercase
Convert paths to lowercase.

–version
show program's version number and exit

catalog

```
calibredb catalog /path/to/destination.(CSV|EPUB|MOBI|XML ...) [options]
```

Export a **catalog** in format specified by path/to/destination extension. Options control how entries are displayed in the generated **catalog** ouput.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

–fields
The fields to output when cataloging books in the database. Should be a comma-separated list of fields. Available fields: all, title, title_sort, author_sort, authors, comments, cover, formats, id, isbn, library_name, ondevice, pubdate, publisher, rating, series_index, series, size, tags, timestamp, uuid, languages, identifiers, plus user-created custom fields. Example: –fields=title,authors,tags Default: 'all' Applies to: CSV, XML output formats

–help, -h
show this help message and exit

–ids, -i
Comma-separated list of database IDs to catalog. If declared, –search is ignored. Default: all

–search, -s
Filter the results by the search query. For the format of the search query, please see the search-related documentation in the User Manual. Default: no filtering

-sort-by
Output field to sort on. Available fields: author_sort, id, rating, size, timestamp, title_sort Default: 'id' Applies to: CSV, XML output formats

-verbose, -v
Show detailed output information. Useful for debugging

-version
show program's version number and exit

saved_searches

```
calibredb saved_searches [options] list
```

calibredb **saved_searches** add name search calibredb **saved_searches** remove name

Manage the saved searches stored in this database. If you try to add a query with a name that already exists, it will be replaced.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-help, -h
show this help message and exit

-version
show program's version number and exit

add_custom_column

```
calibredb add_custom_column [options] label name datatype
```

Create a custom column. label is the machine friendly name of the column. Should not contain spaces or colons. name is the human friendly name of the column. datatype is one of: rating, int, text, float, comments, datetime, composite, bool, enumeration, series

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-display
A dictionary of options to customize how the data in this column will be interpreted. This is a JSON string. For enumeration columns, use –display="{"enum_values":["val1", "val2"]}" There are many options that can go into the display variable.The options by column type are: composite: composite_template, composite_sort, make_category,contains_html, use_decorations datetime: date_format enumeration: enum_values, enum_colors, use_decorations int, float: number_format text: is_names, use_decorations The best way to find legal combinations is to create a customcolumn of the appropriate type in the GUI then look at thebackup OPF for a book (ensure that a new OPF has been createdsince the column was added). You will see the JSON for the"display" for the new column in the OPF.

-help, -h
show this help message and exit

-is-multiple
This column stores tag like data (i.e. multiple comma separated values). Only applies if datatype is text.

-version
show program's version number and exit

custom_columns

```
calibredb custom_columns [options]
```

List available custom columns. Shows column labels and ids.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-details, -d
Show details for each column.

-help, -h
show this help message and exit

-version
show program's version number and exit

remove_custom_column

```
calibredb remove_custom_column [options] label
```

Remove the custom column identified by label. You can see available columns with the custom_columns command.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-force, -f
Do not ask for confirmation

-help, -h
show this help message and exit

-version
show program's version number and exit

set_custom

```
calibredb set_custom [options] column id value
```

Set the value of a custom column for the book identified by id. You can get a list of ids using the list command. You can get a list of custom column names using the custom_columns command.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-append, -a
If the column stores multiple values, append the specified values to the existing ones, instead of replacing them.

-help, -h
show this help message and exit

-version
show program's version number and exit

restore_database

```
calibredb restore_database [options]
```

Restore this database from the metadata stored in OPF files in each directory of the calibre library. This is useful if your metadata.db file has been corrupted.

WARNING: This command completely regenerates your database. You will lose all saved searches, user categories, plugboards, stored per-book conversion settings, and custom recipes. Restored metadata will only be as accurate as what is found in the OPF files.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-help, -h
show this help message and exit

-really-do-it, -r
Really do the recovery. The command will not run unless this option is specified.

-version
show program's version number and exit

check_library

```
calibredb check_library [options]
```

Perform some checks on the filesystem representing a library. Reports are invalid_titles, extra_titles, invalid_authors, extra_authors, missing_formats, extra_formats, extra_files, missing_covers, extra_covers, failed_folders

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-csv, -c
Output in CSV

-help, -h
show this help message and exit

-ignore_extensions, -e
Comma-separated list of extensions to ignore. Default: all

-ignore_names, -n
Comma-separated list of names to ignore. Default: all

-report, -r
Comma-separated list of reports. Default: all

-version
show program's version number and exit

list_categories

```
calibredb list_categories [options]
```

Produce a report of the category information in the database. The information is the equivalent of what is shown in the tags pane.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-categories, -r
Comma-separated list of category lookup names. Default: all

-csv, -c
Output in CSV

-help, -h
show this help message and exit

-item_count, -i
Output only the number of items in a category instead of the counts per item within the category

-quote, -q
The character to put around the category value in CSV mode. Default is quotes (").

-separator, -s
The string used to separate fields in CSV mode. Default is a comma.

-version
show program's version number and exit

-width, -w
The maximum width of a single line in the output. Defaults to detecting screen size.

backup_metadata

```
calibredb backup_metadata [options]
```

Backup the metadata stored in the database into individual OPF files in each books directory. This normally happens automatically, but you can run this command to force re-generation of the OPF files, with the –all option.

Note that there is normally no need to do this, as the OPF files are backed up automatically, every time metadata is changed.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-all
Normally, this command only operates on books that have out of date OPF files. This option makes it operate on all books.

-help, -h
show this help message and exit

-version
show program's version number and exit

ebook-convert

```
ebook-convert input_file output_file [options]
```

Convert an ebook from one format to another.

input_file is the input and output_file is the output. Both must be specified as the first two arguments to the command.

The output ebook format is guessed from the file extension of output_file. output_file can also be of the special format .EXT where EXT is the output file extension. In this case, the name of the output file is derived the name of the input file. Note that the filenames must not start with a hyphen. Finally, if output_file has no extension, then it is treated as a directory and an "open ebook" (OEB) consisting of HTML files is written to that directory. These files are the files that would normally have been passed to the output plugin.

After specifying the input and output file you can customize the conversion by specifying various options. The available options depend on the input and output file types. To get help on them specify the input and output file and then use the -h option.

For full documentation of the conversion system see *Ebook Conversion* (page 305)

Whenever you pass arguments to **ebook-convert** that have spaces in them, enclose the arguments in quotation marks.

The options and default values for the options change depending on both the input and output formats, so you should always check with:

```
ebook-convert myfile.input_format myfile.output_format -h
```

Below are the options that are common to all conversion, followed by the options specific to every input and output format

- Look And Feel (page 488)
- Heuristic Processing (page 490)
- Search And Replace (page 490)
- Structure Detection (page 491)
- Table Of Contents (page 491)
- Metadata (page 492)
- Debug (page 493)
- AZW4 Input Options (page 493)
- CHM Input Options (page 493)
- Comic Input Options (page 493)
- DJVU Input Options (page 494)
- EPUB Input Options (page 494)
- FB2 Input Options (page 494)
- HTLZ Input Options (page 495)
- HTML Input Options (page 495)
- LIT Input Options (page 495)
- LRF Input Options (page 495)
- MOBI Input Options (page 495)
- ODT Input Options (page 495)
- PDB Input Options (page 496)
- PDF Input Options (page 496)
- PML Input Options (page 496)
- RB Input Options (page 496)
- RTF Input Options (page 496)
- Recipe Input Options (page 496)
- SNB Input Options (page 497)
- TCR Input Options (page 497)
- TXT Input Options (page 497)
- AZW3 Output Options (page 497)
- EPUB Output Options (page 498)
- FB2 Output Options (page 499)
- HTML Output Options (page 499)
- HTMLZ Output Options (page 499)
- LIT Output Options (page 500)
- LRF Output Options (page 500)
- MOBI Output Options (page 500)
- OEB Output Options (page 501)
- PDB Output Options (page 501)
- PDF Output Options (page 501)
- PML Output Options (page 502)
- RB Output Options (page 503)
- RTF Output Options (page 503)
- SNB Output Options (page 503)
- TCR Output Options (page 503)
- TXT Output Options (page 504)
- TXTZ Output Options (page 504)

-help, -h
show this help message and exit

-input-profile
Specify the input profile. The input profile gives the conversion system information on how to interpret various information in the input document. For example resolution dependent lengths (i.e. lengths in pixels). Choices are:cybookg3, cybook_opus, default, hanlinv3, hanlinv5, illiad, irexdr1000, irexdr800, kindle, msreader, mo-

bipocket, nook, sony, sony300, sony900

-list-recipes
List builtin recipe names. You can create an ebook from a builtin recipe like this: ebook-convert "Recipe Name.recipe" output.epub

-output-profile
Specify the output profile. The output profile tells the conversion system how to optimize the created document for the specified device. In some cases, an output profile is required to produce documents that will work on a device. For example EPUB on the SONY reader. Choices are:cybookg3, cybook_opus, default, generic_eink, generic_eink_large, hanlinv3, hanlinv5, illiad, ipad, ipad3, irexdr1000, irexdr800, jetbook5, kindle, kindle_dx, kindle_fire, kindle_pw, kobo, msreader, mobipocket, nook, nook_color, nook_hd_plus, pocketbook_900, pocketbook_pro_912, galaxy, bambook, sony, sony300, sony900, sony-landscape, tablet

-version
show program's version number and exit

Look And Feel

Options to control the look and feel of the output

-asciiize
Transliterate unicode characters to an ASCII representation. Use with care because this will replace unicode characters with ASCII. For instance it will replace "Михаил Горбачёв" with "Mikhail Gorbachiov". Also, note that in cases where there are multiple representations of a character (characters shared by Chinese and Japanese for instance) the representation based on the current calibre interface language will be used.

-base-font-size
The base font size in pts. All font sizes in the produced book will be rescaled based on this size. By choosing a larger size you can make the fonts in the output bigger and vice versa. By default, the base font size is chosen based on the output profile you chose.

-change-justification
Change text justification. A value of "left" converts all justified text in the source to left aligned (i.e. unjustified) text. A value of "justify" converts all unjustified text to justified. A value of "original" (the default) does not change justification in the source file. Note that only some output formats support justification.

-disable-font-rescaling
Disable all rescaling of font sizes.

-embed-font-family
Embed the specified font family into the book. This specifies the "base" font used for the book. If the input document specifies its own fonts, they may override this base font. You can use the filter style information option to remove fonts from the input document. Note that font embedding only works with some output formats, principally EPUB and AZW3.

-extra-css
Either the path to a CSS stylesheet or raw CSS. This CSS will be appended to the style rules from the source file, so it can be used to override those rules.

-filter-css
A comma separated list of CSS properties that will be removed from all CSS style rules. This is useful if the presence of some style information prevents it from being overridden on your device. For example: font-family,color,margin-left,margin-right

-font-size-mapping
Mapping from CSS font names to font sizes in pts. An example setting is 12,12,14,16,18,20,22,24. These are the mappings for the sizes xx-small to xx-large, with the final size being for huge fonts. The font rescaling

algorithm uses these sizes to intelligently rescale fonts. The default is to use a mapping based on the output profile you chose.

-insert-blank-line
Insert a blank line between paragraphs. Will not work if the source file does not use paragraphs (<p> or tags).

-insert-blank-line-size
Set the height of the inserted blank lines (in em). The height of the lines between paragraphs will be twice the value set here.

-keep-ligatures
Preserve ligatures present in the input document. A ligature is a special rendering of a pair of characters like ff, fi, fl et cetera. Most readers do not have support for ligatures in their default fonts, so they are unlikely to render correctly. By default, calibre will turn a ligature into the corresponding pair of normal characters. This option will preserve them instead.

-line-height
The line height in pts. Controls spacing between consecutive lines of text. Only applies to elements that do not define their own line height. In most cases, the minimum line height option is more useful. By default no line height manipulation is performed.

-linearize-tables
Some badly designed documents use tables to control the layout of text on the page. When converted these documents often have text that runs off the page and other artifacts. This option will extract the content from the tables and present it in a linear fashion.

-margin-bottom
Set the bottom margin in pts. Default is 5.0. Setting this to less than zero will cause no margin to be set. Note: 72 pts equals 1 inch

-margin-left
Set the left margin in pts. Default is 5.0. Setting this to less than zero will cause no margin to be set. Note: 72 pts equals 1 inch

-margin-right
Set the right margin in pts. Default is 5.0. Setting this to less than zero will cause no margin to be set. Note: 72 pts equals 1 inch

-margin-top
Set the top margin in pts. Default is 5.0. Setting this to less than zero will cause no margin to be set. Note: 72 pts equals 1 inch

-minimum-line-height
The minimum line height, as a percentage of the element's calculated font size. calibre will ensure that every element has a line height of at least this setting, irrespective of what the input document specifies. Set to zero to disable. Default is 120%. Use this setting in preference to the direct line height specification, unless you know what you are doing. For example, you can achieve "double spaced" text by setting this to 240.

-remove-paragraph-spacing
Remove spacing between paragraphs. Also sets an indent on paragraphs of 1.5em. Spacing removal will not work if the source file does not use paragraphs (<p> or tags).

-remove-paragraph-spacing-indent-size
When calibre removes blank lines between paragraphs, it automatically sets a paragraph indent, to ensure that paragraphs can be easily distinguished. This option controls the width of that indent (in em). If you set this value negative, then the indent specified in the input document is used, that is, calibre does not change the indentation.

-smarten-punctuation
Convert plain quotes, dashes and ellipsis to their typographically correct equivalents. For details, see

http://daringfireball.net/projects/smartypants

-subset-embedded-fonts
Subset all embedded fonts. Every embedded font is reduced to contain only the glyphs used in this document. This decreases the size of the font files. Useful if you are embedding a particularly large font with lots of unused glyphs.

-unsmarten-punctuation
Convert fancy quotes, dashes and ellipsis to their plain equivalents.

Heuristic Processing

Modify the document text and structure using common patterns. Disabled by default. Use –enable-heuristics to enable. Individual actions can be disabled with the –disable-* options.

-disable-dehyphenate
Analyze hyphenated words throughout the document. The document itself is used as a dictionary to determine whether hyphens should be retained or removed.

-disable-delete-blank-paragraphs
Remove empty paragraphs from the document when they exist between every other paragraph

-disable-fix-indents
Turn indentation created from multiple non-breaking space entities into CSS indents.

-disable-format-scene-breaks
Left aligned scene break markers are center aligned. Replace soft scene breaks that use multiple blank lines with horizontal rules.

-disable-italicize-common-cases
Look for common words and patterns that denote italics and italicize them.

-disable-markup-chapter-headings
Detect unformatted chapter headings and sub headings. Change them to h2 and h3 tags. This setting will not create a TOC, but can be used in conjunction with structure detection to create one.

-disable-renumber-headings
Looks for occurrences of sequential <h1> or <h2> tags. The tags are renumbered to prevent splitting in the middle of chapter headings.

-disable-unwrap-lines
Unwrap lines using punctuation and other formatting clues.

-enable-heuristics
Enable heuristic processing. This option must be set for any heuristic processing to take place.

-html-unwrap-factor
Scale used to determine the length at which a line should be unwrapped. Valid values are a decimal between 0 and 1. The default is 0.4, just below the median line length. If only a few lines in the document require unwrapping this value should be reduced

-replace-scene-breaks
Replace scene breaks with the specified text. By default, the text from the input document is used.

Search And Replace

Modify the document text and structure using user defined patterns.

-search-replace
Path to a file containing search and replace regular expressions. The file must contain alternating lines of regular expression followed by replacement pattern (which can be an empty line). The regular expression must be in the python regex syntax and the file must be UTF-8 encoded.

-sr1-replace
Replacement to replace the text found with sr1-search.

-sr1-search
Search pattern (regular expression) to be replaced with sr1-replace.

-sr2-replace
Replacement to replace the text found with sr2-search.

-sr2-search
Search pattern (regular expression) to be replaced with sr2-replace.

-sr3-replace
Replacement to replace the text found with sr3-search.

-sr3-search
Search pattern (regular expression) to be replaced with sr3-replace.

Structure Detection

Control auto-detection of document structure.

-chapter
An XPath expression to detect chapter titles. The default is to consider <h1> or <h2> tags that contain the words “chapter”,”book”,”section”, “prologue”, “epilogue”, or “part” as chapter titles as well as any tags that have class=”chapter”. The expression used must evaluate to a list of elements. To disable chapter detection, use the expression “/”. See the XPath Tutorial in the calibre User Manual for further help on using this feature.

-chapter-mark
Specify how to mark detected chapters. A value of “pagebreak” will insert page breaks before chapters. A value of “rule” will insert a line before chapters. A value of “none” will disable chapter marking and a value of “both” will use both page breaks and lines to mark chapters.

-disable-remove-fake-margins
Some documents specify page margins by specifying a left and right margin on each individual paragraph. calibre will try to detect and remove these margins. Sometimes, this can cause the removal of margins that should not have been removed. In this case you can disable the removal.

-insert-metadata
Insert the book metadata at the start of the book. This is useful if your ebook reader does not support displaying/searching metadata directly.

-page-breaks-before
An XPath expression. Page breaks are inserted before the specified elements. To disable use the expression: /

-prefer-metadata-cover
Use the cover detected from the source file in preference to the specified cover.

-remove-first-image
Remove the first image from the input ebook. Useful if the input document has a cover image that is not identified as a cover. In this case, if you set a cover in calibre, the output document will end up with two cover images if you do not specify this option.

-start-reading-at
An XPath expression to detect the location in the document at which to start reading. Some ebook reading

programs (most prominently the Kindle) use this location as the position at which to open the book. See the XPath tutorial in the calibre User Manual for further help using this feature.

Table Of Contents

Control the automatic generation of a Table of Contents. By default, if the source file has a Table of Contents, it will be used in preference to the automatically generated one.

-duplicate-links-in-toc
When creating a TOC from links in the input document, allow duplicate entries, i.e. allow more than one entry with the same text, provided that they point to a different location.

-level1-toc
XPath expression that specifies all tags that should be added to the Table of Contents at level one. If this is specified, it takes precedence over other forms of auto-detection. See the XPath Tutorial in the calibre User Manual for examples.

-level2-toc
XPath expression that specifies all tags that should be added to the Table of Contents at level two. Each entry is added under the previous level one entry. See the XPath Tutorial in the calibre User Manual for examples.

-level3-toc
XPath expression that specifies all tags that should be added to the Table of Contents at level three. Each entry is added under the previous level two entry. See the XPath Tutorial in the calibre User Manual for examples.

-max-toc-links
Maximum number of links to insert into the TOC. Set to 0 to disable. Default is: 50. Links are only added to the TOC if less than the threshold number of chapters were detected.

-no-chapters-in-toc
Don't add auto-detected chapters to the Table of Contents.

-toc-filter
Remove entries from the Table of Contents whose titles match the specified regular expression. Matching entries and all their children are removed.

-toc-threshold
If fewer than this number of chapters is detected, then links are added to the Table of Contents. Default: 6

-use-auto-toc
Normally, if the source file already has a Table of Contents, it is used in preference to the auto-generated one. With this option, the auto-generated one is always used.

Metadata

Options to set metadata in the output

-author-sort
String to be used when sorting by author.

-authors
Set the authors. Multiple authors should be separated by ampersands.

-book-producer
Set the book producer.

-comments
Set the ebook description.

-cover
Set the cover to the specified file or URL

-isbn
Set the ISBN of the book.

-language
Set the language.

-pubdate
Set the publication date.

-publisher
Set the ebook publisher.

-rating
Set the rating. Should be a number between 1 and 5.

-series
Set the series this ebook belongs to.

-series-index
Set the index of the book in this series.

-tags
Set the tags for the book. Should be a comma separated list.

-timestamp
Set the book timestamp (no longer used anywhere)

-title
Set the title.

-title-sort
The version of the title to be used for sorting.

Debug

Options to help with debugging the conversion

-debug-pipeline, -d
Save the output from different stages of the conversion pipeline to the specified directory. Useful if you are unsure at which stage of the conversion process a bug is occurring.

-verbose, -v
Level of verbosity. Specify multiple times for greater verbosity.

AZW4 Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

CHM Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by

the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

Comic Input Options

-colors
Number of colors for grayscale image conversion. Default: 256. Values of less than 256 may result in blurred text on your device if you are creating your comics in EPUB format.

-comic-image-size
Specify the image size as widthxheight pixels. Normally, an image size is automatically calculated from the output profile, this option overrides it.

-despeckle
Enable Despeckle. Reduces speckle noise. May greatly increase processing time.

-disable-trim
Disable trimming of comic pages. For some comics, trimming might remove content as well as borders.

-dont-add-comic-pages-to-toc
When converting a CBC do not add links to each page to the TOC. Note this only applies if the TOC has more than one section

-dont-grayscale
Do not convert the image to grayscale (black and white)

-dont-normalize
Disable normalize (improve contrast) color range for pictures. Default: False

-dont-sharpen
Disable sharpening.

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

-keep-aspect-ratio
Maintain picture aspect ratio. Default is to fill the screen.

-landscape
Don't split landscape images into two portrait images

-no-process
Apply no processing to the image

-no-sort
Don't sort the files found in the comic alphabetically by name. Instead use the order they were added to the comic.

-output-format
The format that images in the created ebook are converted to. You can experiment to see which format gives you optimal size and look on your device.

-right2left
Used for right-to-left publications like manga. Causes landscape pages to be split into portrait pages from right to left.

-wide
Keep aspect ratio and scale image using screen height as image width for viewing in landscape mode.

DJVU Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

-use-djvutxt
Try to use the djvutxt program and fall back to pure python implementation if it fails or is not available

EPUB Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

FB2 Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

-no-inline-fb2-toc
Do not insert a Table of Contents at the beginning of the book.

HTLZ Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

HTML Input Options

-breadth-first
Traverse links in HTML files breadth first. Normally, they are traversed depth first.

-dont-package
Normally this input plugin re-arranges all the input files into a standard folder hierarchy. Only use this option if you know what you are doing as it can result in various nasty side effects in the rest of the conversion pipeline.

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

-max-levels
Maximum levels of recursion when following links in HTML files. Must be non-negative. 0 implies that no links in the root HTML file are followed. Default is 5.

LIT Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

LRF Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

MOBI Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

ODT Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

PDB Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

PDF Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

-new-pdf-engine
Use the new PDF conversion engine.

-no-images
Do not extract images from the document

-unwrap-factor
Scale used to determine the length at which a line should be unwrapped. Valid values are a decimal between 0 and 1. The default is 0.45, just below the median line length.

PML Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

RB Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

RTF Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

Recipe Input Options

-dont-download-recipe
Do not download latest version of builtin recipes from the calibre server

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

-lrf
Optimize fetching for subsequent conversion to LRF.

-password
Password for sites that require a login to access content.

-test
Useful for recipe development. Forces max_articles_per_feed to 2 and downloads at most 2 feeds.

-username
Username for sites that require a login to access content.

SNB Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

TCR Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

TXT Input Options

-formatting-type
Formatting used within the document.* auto: Automatically decide which formatting processor to use. * plain: Do not process the document formatting. Everything is a paragraph and no styling is applied. * heuristic: Process using heuristics to determine formatting such as chapter headings and italic text. * textile: Processing using textile formatting. * markdown: Processing using markdown formatting. To learn more about markdown see http://daringfireball.net/projects/markdown/

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

-markdown-disable-toc
Do not insert a Table of Contents into the output text.

-paragraph-type
Paragraph structure. choices are ['auto', 'block', 'single', 'print', 'unformatted', 'off'] * auto: Try to auto detect paragraph type. * block: Treat a blank line as a paragraph break. * single: Assume every line is a paragraph. * print: Assume every line starting with 2+ spaces or a tab starts a paragraph. * unformatted: Most lines have hard line breaks, few/no blank lines or indents. Tries to determine structure and reformat the differentiate elements. * off: Don't modify the paragraph structure. This is useful when combined with Markdown or Textile formatting to ensure no formatting is lost.

-preserve-spaces
Normally extra spaces are condensed into a single space. With this option all spaces will be displayed.

-txt-in-remove-indents
Normally extra space at the beginning of lines is retained. With this option they will be removed.

AZW3 Output Options

-dont-compress
Disable compression of the file contents.

-extract-to
Extract the contents of the MOBI file to the specified directory. If the directory already exists, it will be deleted.

-mobi-toc-at-start
When adding the Table of Contents to the book, add it at the start of the book instead of the end. Not recommended.

-no-inline-toc
Don't add Table of Contents to the book. Useful if the book has its own table of contents.

-prefer-author-sort
When present, use author sort field as author.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-share-not-sync
Enable sharing of book content via Facebook etc. on the Kindle. WARNING: Using this feature means that the book will not auto sync its last read position on multiple devices. Complain to Amazon.

-toc-title
Title for any generated in-line table of contents.

EPUB Output Options

-dont-split-on-page-breaks
Turn off splitting at page breaks. Normally, input files are automatically split at every page break into two files. This gives an output ebook that can be parsed faster and with less resources. However, splitting is slow and if your source file contains a very large number of page breaks, you should turn off splitting on page breaks.

-epub-flatten
This option is needed only if you intend to use the EPUB with FBReaderJ. It will flatten the file system inside the EPUB, putting all files into the top level.

-extract-to
Extract the contents of the generated EPUB file to the specified directory. The contents of the directory are first deleted, so be careful.

-flow-size
Split all HTML files larger than this size (in KB). This is necessary as most EPUB readers cannot handle large file sizes. The default of 260KB is the size required for Adobe Digital Editions.

-no-default-epub-cover
Normally, if the input file has no cover and you don't specify one, a default cover is generated with the title, authors, etc. This option disables the generation of this cover.

-no-svg-cover
Do not use SVG for the book cover. Use this option if your EPUB is going to be used on a device that does not support SVG, like the iPhone or the JetBook Lite. Without this option, such devices will display the cover as a blank page.

-preserve-cover-aspect-ratio
When using an SVG cover, this option will cause the cover to scale to cover the available screen area, but still preserve its aspect ratio (ratio of width to height). That means there may be white borders at the sides or top and bottom of the image, but the image will never be distorted. Without this option the image may be slightly distorted, but there will be no borders.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

FB2 Output Options

-fb2-genre
Genre for the book. Choices: sf_history, sf_action, sf_epic, sf_heroic, sf_detective, sf_cyberpunk, sf_space, sf_social, sf_horror, sf_humor, sf_fantasy, sf, det_classic, det_police, det_action, det_irony, det_history, det_espionage, det_crime, det_political, det_maniac, det_hard, thriller, detective, prose_classic, prose_history, prose_contemporary, prose_counter, prose_rus_classic, prose_su_classics, love_contemporary, love_history, love_detective, love_short, love_erotica, adv_western, adv_history, adv_indian, adv_maritime, adv_geo,

adv_animal, adventure, child_tale, child_verse, child_prose, child_sf, child_det, child_adv, child_education, children, poetry, dramaturgy, antique_ant, antique_european, antique_russian, antique_east, antique_myths, antique, sci_history, sci_psychology, sci_culture, sci_religion, sci_philosophy, sci_politics, sci_business, sci_juris, sci_linguistic, sci_medicine, sci_phys, sci_math, sci_chem, sci_biology, sci_tech, science, comp_www, comp_programming, comp_hard, comp_soft, comp_db, comp_osnet, computers, ref_encyc, ref_dict, ref_ref, ref_guide, reference, nonf_biography, nonf_publicism, nonf_criticism, design, nonfiction, religion_rel, religion_esoterics, religion_self, religion, humor_anecdote, humor_prose, humor_verse, humor, home_cooking, home_pets, home_crafts, home_entertain, home_health, home_garden, home_diy, home_sport, home_sex, home See: http://www.fictionbook.org/index.php/Eng:FictionBook_2.1_genres for a complete list with descriptions.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-sectionize
Specify the sectionization of elements. A value of "nothing" turns the book into a single section. A value of "files" turns each file into a separate section; use this if your device is having trouble. A value of "Table of Contents" turns the entries in the Table of Contents into titles and creates sections; if it fails, adjust the "Structure Detection" and/or "Table of Contents" settings (turn on "Force use of auto-generated Table of Contents").

HTML Output Options

-extract-to
Extract the contents of the generated ZIP file to the specified directory. WARNING: The contents of the directory will be deleted.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-template-css
CSS file used for the output instead of the default file

-template-html
Template used for the generation of the html contents of the book instead of the default file

-template-html-index
Template used for generation of the html index file instead of the default file

HTMLZ Output Options

-htmlz-class-style
How to handle the CSS when using css-type = 'class'. Default is external. external: Use an external CSS file that is linked in the document. inline: Place the CSS in the head section of the document.

-htmlz-css-type
Specify the handling of CSS. Default is class. class: Use CSS classes and have elements reference them. inline: Write the CSS as an inline style attribute. tag: Turn as many CSS styles as possible into HTML tags.

-htmlz-title-filename
If set this option causes the file name of the html file inside the htmlz archive to be based on the book title.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

LIT Output Options

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

LRF Output Options

-enable-autorotation
Enable autorotation of images that are wider than the screen width.

-header
Add a header to all the pages with title and author.

-header-format
Set the format of the header. %a is replaced by the author and %t by the title. Default is %t by %a

-header-separation
Add extra spacing below the header. Default is 0 pt.

-minimum-indent
Minimum paragraph indent (the indent of the first line of a paragraph) in pts. Default: 0

-mono-family
The monospace family of fonts to embed

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-render-tables-as-images
Render tables in the HTML as images (useful if the document has large or complex tables)

-sans-family
The sans-serif family of fonts to embed

-serif-family
The serif family of fonts to embed

-text-size-multiplier-for-rendered-tables
Multiply the size of text in rendered tables by this factor. Default is 1.0

-wordspace
Set the space between words in pts. Default is 2.5

MOBI Output Options

-dont-compress
Disable compression of the file contents.

-extract-to
Extract the contents of the MOBI file to the specified directory. If the directory already exists, it will be deleted.

-mobi-file-type
By default calibre generates MOBI files that contain the old MOBI 6 format. This format is compatible with all devices. However, by changing this setting, you can tell calibre to generate MOBI files that contain both MOBI 6 and the new KF8 format, or only the new KF8 format. KF8 has more features than MOBI 6, but only works with newer Kindles.

-mobi-ignore-margins
Ignore margins in the input document. If False, then the MOBI output plugin will try to convert margins specified in the input document, otherwise it will ignore them.

-mobi-keep-original-images
By default calibre converts all images to JPEG format in the output MOBI file. This is for maximum compatibility as some older MOBI viewers have problems with other image formats. This option tells calibre not to do this. Useful if your document contains lots of GIF/PNG images that become very large when converted to JPEG.

-mobi-toc-at-start
When adding the Table of Contents to the book, add it at the start of the book instead of the end. Not recommended.

-no-inline-toc
Don't add Table of Contents to the book. Useful if the book has its own table of contents.

-personal-doc
Tag marking book to be filed with Personal Docs

-prefer-author-sort
When present, use author sort field as author.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-share-not-sync
Enable sharing of book content via Facebook etc. on the Kindle. WARNING: Using this feature means that the book will not auto sync its last read position on multiple devices. Complain to Amazon.

-toc-title
Title for any generated in-line table of contents.

OEB Output Options

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

PDB Output Options

-format, -f
Format to use inside the pdb container. Choices are: ['doc', 'ztxt', 'ereader']

-inline-toc
Add Table of Contents to beginning of the book.

-pdb-output-encoding
Specify the character encoding of the output document. The default is cp1252. Note: This option is not honored by all formats.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

PDF Output Options

-custom-size
Custom size of the document. Use the form widthxheight EG. *123x321* to specify the width and height. This overrides any specified paper-size.

-old-pdf-engine
Use the old, less capable engine to generate the PDF

-override-profile-size
Normally, the PDF page size is set by the output profile chosen under page options. This option will cause the page size settings under PDF Output to override the size specified by the output profile.

-paper-size
The size of the paper. This size will be overridden when a non default output profile is used. Default is letter. Choices are [u'a0', u'a1', u'a2', u'a3', u'a4', u'a5', u'a6', u'b0', u'b1', u'b2', u'b3', u'b4', u'b5', u'b6', u'legal', u'letter']

-pdf-default-font-size
The default font size

-pdf-footer-template
An HTML template used to generate footers on every page. The string _PAGENUM_ will be replaced by the current page number.

-pdf-header-template
An HTML template used to generate headers on every page. The string _PAGENUM_ will be replaced by the current page number.

-pdf-mark-links
Surround all links with a red box, useful for debugging.

-pdf-mono-family
The font family used to render monospaced fonts

-pdf-mono-font-size
The default font size for monospaced text

-pdf-page-numbers
Add page numbers to the bottom of every page in the generated PDF file. If you specify a footer template, it will take precedence over this option.

-pdf-sans-family
The font family used to render sans-serif fonts

-pdf-serif-family
The font family used to render serif fonts

-pdf-standard-font
The font family used to render monospaced fonts

-preserve-cover-aspect-ratio
Preserve the aspect ratio of the cover, instead of stretching it to fill the full first page of the generated pdf.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-uncompressed-pdf
Generate an uncompressed PDF, useful for debugging, only works with the new PDF engine.

-unit, -u
The unit of measure for page sizes. Default is inch. Choices are ['millimeter', 'centimeter', 'point', 'inch', 'pica', 'didot', 'cicero', 'devicepixel'] Note: This does not override the unit for margins!

PML Output Options

-full-image-depth
Do not reduce the size or bit depth of images. Images have their size and depth reduced by default to accommodate applications that can not convert images on their own such as Dropbook.

-inline-toc
Add Table of Contents to beginning of the book.

-pml-output-encoding
Specify the character encoding of the output document. The default is cp1252.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

RB Output Options

-inline-toc
Add Table of Contents to beginning of the book.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

RTF Output Options

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

SNB Output Options

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-snb-dont-indent-first-line
Specify whether or not to insert two space characters to indent the first line of each paragraph.

-snb-full-screen
Resize all the images for full screen view.

-snb-hide-chapter-name
Specify whether or not to hide the chapter title for each chapter. Useful for image-only output (eg. comics).

-snb-insert-empty-line
Specify whether or not to insert an empty line between two paragraphs.

-snb-max-line-length
The maximum number of characters per line. This splits on the first space before the specified value. If no space is found the line will be broken at the space after and will exceed the specified value. Also, there is a minimum of 25 characters. Use 0 to disable line splitting.

-snb-output-encoding
Specify the character encoding of the output document. The default is utf-8.

TCR Output Options

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-tcr-output-encoding
Specify the character encoding of the output document. The default is utf-8.

TXT Output Options

-force-max-line-length
Force splitting on the max-line-length value when no space is present. Also allows max-line-length to be below the minimum

-inline-toc
Add Table of Contents to beginning of the book.

-keep-color
Do not remove font color from output. This is only useful when txt-output-formatting is set to textile. Textile is the only formatting that supports setting font color. If this option is not specified font color will not be set and default to the color displayed by the reader (generally this is black).

-keep-image-references
Do not remove image references within the document. This is only useful when paired with a txt-output-formatting option that is not none because links are always removed with plain text output.

-keep-links
Do not remove links within the document. This is only useful when paired with a txt-output-formatting option that is not none because links are always removed with plain text output.

-max-line-length
The maximum number of characters per line. This splits on the first space before the specified value. If no space is found the line will be broken at the space after and will exceed the specified value. Also, there is a minimum of 25 characters. Use 0 to disable line splitting.

-newline, -n
Type of newline to use. Options are ['old_mac', 'system', 'unix', 'windows']. Default is 'system'. Use 'old_mac' for compatibility with Mac OS 9 and earlier. For Mac OS X use 'unix'. 'system' will default to the newline type used by this OS.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-txt-output-encoding
Specify the character encoding of the output document. The default is utf-8.

-txt-output-formatting
Formatting used within the document. * plain: Produce plain text. * markdown: Produce Markdown formatted text. * textile: Produce Textile formatted text.

TXTZ Output Options

-force-max-line-length
Force splitting on the max-line-length value when no space is present. Also allows max-line-length to be below the minimum

-inline-toc
Add Table of Contents to beginning of the book.

-keep-color
Do not remove font color from output. This is only useful when txt-output-formatting is set to textile. Textile is the only formatting that supports setting font color. If this option is not specified font color will not be set and default to the color displayed by the reader (generally this is black).

-keep-image-references
Do not remove image references within the document. This is only useful when paired with a txt-output-formatting option that is not none because links are always removed with plain text output.

-keep-links
Do not remove links within the document. This is only useful when paired with a txt-output-formatting option that is not none because links are always removed with plain text output.

-max-line-length
The maximum number of characters per line. This splits on the first space before the specified value. If no space is found the line will be broken at the space after and will exceed the specified value. Also, there is a minimum of 25 characters. Use 0 to disable line splitting.

-newline, -n
Type of newline to use. Options are ['old_mac', 'system', 'unix', 'windows']. Default is 'system'. Use 'old_mac' for compatibility with Mac OS 9 and earlier. For Mac OS X use 'unix'. 'system' will default to the newline type used by this OS.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-txt-output-encoding
Specify the character encoding of the output document. The default is utf-8.

-txt-output-formatting
Formatting used within the document. * plain: Produce plain text. * markdown: Produce Markdown formatted text. * textile: Produce Textile formatted text.

ebook-meta

```
ebook-meta ebook_file [options]
```

Read/Write metadata from/to ebook files.

Supported formats for reading metadata: azw, azw1, azw3, azw4, cbr, cbz, chm, docx, epub, fb2, html, htmlz, imp, lit, lrf, lrx, mobi, odt, oebzip, opf, pdb, pdf, pml, pmlz, pobi, prc, rar, rb, rtf, snb, tpz, txt, txtz, updb, zip

Supported formats for writing metadata: azw, azw1, azw3, azw4, epub, fb2, htmlz, lrf, mobi, pdb, pdf, prc, rtf, tpz, txtz

Different file types support different kinds of metadata. If you try to set some metadata on a file type that does not support it, the metadata will be silently ignored.

Whenever you pass arguments to **ebook-meta** that have spaces in them, enclose the arguments in quotation marks.

[options]

--author-sort
String to be used when sorting by author. If unspecified, and the author(s) are specified, it will be auto-generated from the author(s).

--authors, -a
Set the authors. Multiple authors should be separated by the & character. Author names should be in the order Firstname Lastname.

--book-producer, -k
Set the book producer.

--category
Set the book category.

--comments, -c
Set the ebook description.

--cover
Set the cover to the specified file.

--date, -d
Set the published date.

--from-opf
Read metadata from the specified OPF file and use it to set metadata in the ebook. Metadata specified on the command line will override metadata read from the OPF file

--get-cover
Get the cover from the ebook and save it at as the specified file.

--help, -h
show this help message and exit

--index, -i
Set the index of the book in this series.

--isbn
Set the ISBN of the book.

--language, -l
Set the language.

--lrf-bookid
Set the BookID in LRF files

--publisher, -p
Set the ebook publisher.

--rating, -r
Set the rating. Should be a number between 1 and 5.

--series, -s
Set the series this ebook belongs to.

-tags
Set the tags for the book. Should be a comma separated list.

-title, -t
Set the title.

-title-sort
The version of the title to be used for sorting. If unspecified, and the title is specified, it will be auto-generated from the title.

-to-opf
Specify the name of an OPF file. The metadata will be written to the OPF file.

-version
show program's version number and exit

ebook-polish

```
ebook-polish [options] input_file [output_file]
```

Polishing books is all about putting the shine of perfection onto your carefully crafted ebooks.

Polishing tries to minimize the changes to the internal code of your ebook. Unlike conversion, it does not flatten CSS, rename files, change font sizes, adjust margins, etc. Every action performs only the minimum set of changes needed for the desired effect.

You should use this tool as the last step in your ebook creation process.

Note that polishing only works on files in the AZW3 or EPUB formats.

Whenever you pass arguments to **ebook-polish** that have spaces in them, enclose the arguments in quotation marks.

[options]

-cover, -c
Path to a cover image. Changes the cover specified in the ebook. If no cover is present, or the cover is not properly identified, inserts a new cover.

-help, -h
show this help message and exit

-jacket, -j
Insert a "book jacket" page at the start of the book that contains all the book metadata such as title, tags, authors, series, comments, etc.

-opf, -o
Path to an OPF file. The metadata in the book is updated from the OPF file.

-remove-jacket
Remove a previous inserted book jacket page.

-smarten-punctuation, -p
Convert plain text dashes, ellipsis, quotes, multiple hyphens, etc. into their typographically correct equivalents. Note that the algorithm can sometimes generate incorrect results, especially when single quotes at the start of contractions are involved.

-subset-fonts, -f
Subsetting fonts means reducing an embedded font to contain only the characters used from that font in the book. This greatly reduces the size of the font files (halving the font file sizes is common). For example, if the

book uses a specific font for headers, then subsetting will reduce that font to contain only the characters present in the actual headers in the book. Or if the book embeds the bold and italic versions of a font, but bold and italic text is relatively rare, or absent altogether, then the bold and italic fonts can either be reduced to only a few characters or completely removed. The only downside to subsetting fonts is that if, at a later date you decide to add more text to your books, the newly added text might not be covered by the subset font.

-verbose
Produce more verbose output, useful for debugging.

-version
show program's version number and exit

ebook-viewer

```
ebook-viewer [options] file
```

View an ebook.

Whenever you pass arguments to **ebook-viewer** that have spaces in them, enclose the arguments in quotation marks.

[options]

-debug-javascript
Print javascript alert and console messages to the console

-full-screen, -f
If specified, viewer window will try to open full screen when started.

-help, -h
show this help message and exit

-open-at
The position at which to open the specified book. The position is a location as displayed in the top left corner of the viewer.

-raise-window
If specified, viewer window will try to come to the front when started.

-version
show program's version number and exit

fetch-ebook-metadata

```
fetch-ebook-metadata [options]
```

Fetch book metadata from online sources. You must specify at least one of title, authors or ISBN.

Whenever you pass arguments to **fetch-ebook-metadata** that have spaces in them, enclose the arguments in quotation marks.

[options]

-authors, -a
Book author(s)

-cover, -c
Specify a filename. The cover, if available, will be saved to it

-help, -h
show this help message and exit

-isbn, -i
Book ISBN

-opf, -o
Output the metadata in OPF format

-timeout, -d
Timeout in seconds. Default is 30

-title, -t
Book title

-verbose, -v
Print the log to the console (stderr)

-version
show program's version number and exit

lrf2lrs

```
lrf2lrs book.lrf
```

Convert an LRF file into an LRS (XML UTF-8 encoded) file

Whenever you pass arguments to **lrf2lrs** that have spaces in them, enclose the arguments in quotation marks.

[options]

-dont-output-resources
Do not save embedded image and font files to disk

-help, -h
show this help message and exit

-output, -o
Output LRS file

-verbose

-version
show program's version number and exit

lrfviewer

```
lrfviewer [options] book.lrf
```

Read the LRF ebook book.lrf

Whenever you pass arguments to **lrfviewer** that have spaces in them, enclose the arguments in quotation marks.

[options]

-disable-hyphenation
Disable hyphenation. Should significantly speed up rendering.

-help, -h
show this help message and exit

-profile
Profile the LRF renderer

-verbose
Print more information about the rendering process

-version
show program's version number and exit

-visual-debug
Turn on visual aids to debugging the rendering engine

-white-background
By default the background is off white as I find this easier on the eyes. Use this option to make the background pure white.

lrs2lrf

```
lrs2lrf [options] file.lrs
```

Compile an LRS file into an LRF file.

Whenever you pass arguments to **lrs2lrf** that have spaces in them, enclose the arguments in quotation marks.

[options]

-help, -h
show this help message and exit

-lrs
Convert LRS to LRS, useful for debugging.

-output, -o
Path to output file

-verbose
Verbose processing

-version
show program's version number and exit

web2disk

```
web2disk URL
```

Where URL is for example http://google.com

Whenever you pass arguments to **web2disk** that have spaces in them, enclose the arguments in quotation marks.

[options]

-base-dir, -d
Base directory into which URL is saved. Default is .

-delay
Minimum interval in seconds between consecutive fetches. Default is 0 s

-dont-download-stylesheets
Do not download CSS stylesheets.

-encoding
The character encoding for the websites you are trying to download. The default is to try and guess the encoding.

-filter-regexp
Any link that matches this regular expression will be ignored. This option can be specified multiple times, in which case as long as any regexp matches a link, it will be ignored. By default, no links are ignored. If both filter regexp and match regexp are specified, then filter regexp is applied first.

-help, -h
show this help message and exit

-match-regexp
Only links that match this regular expression will be followed. This option can be specified multiple times, in which case as long as a link matches any one regexp, it will be followed. By default all links are followed.

-max-files, -n
The maximum number of files to download. This only applies to files from <a href> tags. Default is 9223372036854775807

-max-recursions, -r
Maximum number of levels to recurse i.e. depth of links to follow. Default 1

-timeout, -t
Timeout in seconds to wait for a response from the server. Default: 10.0 s

-verbose
Show detailed output information. Useful for debugging

-version
show program's version number and exit

1.9.2 Undocumented Commands

- ebook-device
- markdown-calibre

You can see usage for undocumented commands by executing them without arguments in a terminal.

1.10 Setting up a calibre development environment

calibre is completely open source, licensed under the GNU GPL v3[93]. This means that you are free to download and modify the program to your heart's content. In this section, you will learn how to get a calibre development environment set up on the operating system of your choice. calibre is written primarily in Python[94] with some C/C++

[93] http://www.gnu.org/copyleft/gpl.html
[94] http://www.python.org

code for speed and system interfacing. Note that calibre is not compatible with Python 3 and requires at least Python 2.7.

Contents

- Design philosophy (page 511)
  - Code layout (page 512)
- Getting the code (page 512)
  - Submitting your changes to be included (page 513)
- Windows development environment (page 513)
- OS X development environment (page 514)
- Linux development environment (page 514)
- Having separate "normal" and "development" calibre installs on the same computer (page 515)
- Debugging tips (page 515)
  - Using an interactive python interpreter (page 515)
  - Using print statements (page 515)
  - Using the debugger in PyDev (page 516)
  - Executing arbitrary scripts in the calibre python environment (page 516)
- Using calibre in your projects (page 516)
  - Binary install of calibre (page 516)
  - Source install on Linux (page 516)

1.10.1 Design philosophy

calibre has its roots in the Unix world, which means that its design is highly modular. The modules interact with each other via well defined interfaces. This makes adding new features and fixing bugs in calibre very easy, resulting in a frenetic pace of development. Because of its roots, calibre has a comprehensive command line interface for all its functions, documented in *Command Line Interface* (page 472).

The modular design of calibre is expressed via `Plugins`. There is a *tutorial* (page 440) on writing calibre plugins. For example, adding support for a new device to calibre typically involves writing less than a 100 lines of code in the form of a device driver plugin. You can browse the built-in drivers[95]. Similarly, adding support for new conversion formats involves writing input/output format plugins. Another example of the modular design is the *recipe system* (page 341) for fetching news. For more examples of plugins designed to add features to calibre, see the plugin index[96].

Code layout

All the calibre python code is in the `calibre` package. This package contains the following main sub-packages

- devices - All the device drivers. Just look through some of the built-in drivers to get an idea for how they work.
  - For details, see: devices.interface which defines the interface supported by device drivers and `devices.usbms` which defines a generic driver that connects to a USBMS device. All USBMS based drivers in calibre inherit from it.
- ebooks - All the ebook conversion/metadata code. A good starting point is `calibre.ebooks.conversion.cli` which is the module powering the **ebook-convert** command. The conversion process is controlled via `conversion.plumber`. The format independent code is all in `ebooks.oeb` and the format dependent code is in `ebooks.format_name`.
  - Metadata reading, writing, and downloading is all in `ebooks.metadata`

[95] http://bazaar.launchpad.net/%7Ekovid/calibre/trunk/files/head%3A/src/calibre/devices/
[96] http://www.mobileread.com/forums/showthread.php?p=1362767#post1362767

  - Conversion happens in a pipeline, for the structure of the pipeline, see *Introduction* (page 306). The pipeline consists of an input plugin, various transforms and an output plugin. The that code constructs and drives the pipeline is in `plumber.py`. The pipeline works on a representation of an ebook that is like an unzipped epub, with manifest, spine, toc, guide, html content, etc. The class that manages this representation is OEBBook in `ebooks.oeb.base`. The various transformations that are applied to the book during conversions live in `oeb/transforms/*.py`. And the input and output plugins live in `conversion/plugins/*.py`.

- library - The database back-end and the content server. See `library.database2` for the interface to the calibre library. `library.server` is the calibre Content Server.
- gui2 - The Graphical User Interface. GUI initialization happens in `gui2.main` and `gui2.ui`. The ebook-viewer is in `gui2.viewer`.

If you need help understanding the code, post in the development forum[97] and you will most likely get help from one of calibre's many developers.

1.10.2 Getting the code

calibre uses Bazaar[98], a distributed version control system. Bazaar is available on all the platforms calibre supports. After installing Bazaar, you can get the calibre source code with the command:

```
bzr branch lp:calibre
```

On Windows you will need the complete path name, that will be something like `C:\Program Files\Bazaar\bzr.exe`.

calibre is a very large project with a very long source control history, so the above can take a while (10mins to an hour depending on your internet speed).

If you want to get the code faster, the sourcecode for the latest release is always available as an archive[99]. You can also use bzr to just download the source code, without the history, using:

```
bzr branch --stacked lp:calibre
```

To update a branch to the latest code, use the command:

```
bzr merge
```

Submitting your changes to be included

If you only plan to make a few small changes, you can make your changes and create a "merge directive" which you can then attach to a ticket in the calibre bug tracker for consideration. To do this, make your changes, then run:

```
bzr commit -m "Comment describing your changes"
bzr send -o my-changes
```

This will create a `my-changes` file in the current directory, simply attach that to a ticket on the calibre bug tracker[100].

If you plan to do a lot of development on calibre, then the best method is to create a Launchpad[101] account. Once you have an account, you can use it to register your bzr branch created by the *bzr branch* command above. First run the following command to tell bzr about your launchpad account:

[97] http://www.mobileread.com/forums/forumdisplay.php?f=240
[98] http://bazaar-vcs.org/
[99] http://status.calibre-ebook.com/dist/src
[100] https://bugs.launchpad.net/calibre
[101] http://launchpad.net

```
bzr launchpad-login your_launchpad_username
```

Now, you have to setup SSH access to Launchpad. First create an SSH public/private keypair. Then upload the public key to Launchpad by going to your Launchpad account page. Instructions for setting up the private key in bzr are at http://bazaar-vcs.org/Bzr_and_SSH. Now you can upload your branch to the calibre project in Launchpad by following the instructions at https://help.launchpad.net/Code/UploadingABranch. Whenever you commit changes to your branch with the command:

```
bzr commit -m "Comment describing your change"
```

Kovid can merge it directly from your branch into the main calibre source tree. You should also keep an eye on the calibre development forum[102]. Before making major changes, you should discuss them in the forum or contact Kovid directly (his email address is all over the source code).

1.10.3 Windows development environment

Install calibre normally, using the Windows installer. Then open a Command Prompt and change to the previously checked out calibre code directory. For example:

```
cd C:\Users\kovid\work\calibre
```

calibre is the directory that contains the src and resources sub-directories.

The next step is to set the environment variable `CALIBRE_DEVELOP_FROM` to the absolute path of the src directory. So, following the example above, it would be `C:\Users\kovid\work\calibre\src`. Here is a short guide[103] to setting environment variables on Windows.

Once you have set the environment variable, open a new command prompt and check that it was correctly set by using the command:

```
echo %CALIBRE_DEVELOP_FROM%
```

Setting this environment variable means that calibre will now load all its Python code from the specified location.

That's it! You are now ready to start hacking on the calibre code. For example, open the file `src\calibre\__init__.py` in your favorite editor and add the line:

```
print ("Hello, world!")
```

near the top of the file. Now run the command **calibredb**. The very first line of output should be `Hello, world!`.

1.10.4 OS X development environment

Install calibre normally using the provided .dmg. Then open a Terminal and change to the previously checked out calibre code directory, for example:

```
cd /Users/kovid/work/calibre
```

calibre is the directory that contains the src and resources sub-directories. Ensure you have installed the calibre commandline tools via *Preferences->Advanced->Miscellaneous* in the calibre GUI.

The next step is to create a bash script that will set the environment variable `CALIBRE_DEVELOP_FROM` to the absolute path of the src directory when running calibre in debug mode.

Create a plain text file:

[102] http://www.mobileread.com/forums/forumdisplay.php?f=240
[103] http://docs.python.org/using/windows.html#excursus-setting-environment-variables

```
#!/bin/sh
export CALIBRE_DEVELOP_FROM="/Users/kovid/work/calibre/src"
calibre-debug -g
```

Save this file as `/usr/bin/calibre-develop`, then set its permissions so that it can be executed:

```
chmod +x /usr/bin/calibre-develop
```

Once you have done this, run:

```
calibre-develop
```

You should see some diagnostic information in the Terminal window as calibre starts up, and you should see an asterisk after the version number in the GUI window, indicating that you are running from source.

1.10.5 Linux development environment

calibre is primarily developed on Linux. You have two choices in setting up the development environment. You can install the calibre binary as normal and use that as a runtime environment to do your development. This approach is similar to that used in Windows and OS X. Alternatively, you can install calibre from source. Instructions for setting up a development environment from source are in the INSTALL file in the source tree. Here we will address using the binary at runtime, which is the recommended method.

Install the calibre using the binary installer. Then open a terminal and change to the previously checked out calibre code directory, for example:

```
cd /home/kovid/work/calibre
```

calibre is the directory that contains the src and resources sub-directories.

The next step is to set the environment variable `CALIBRE_DEVELOP_FROM` to the absolute path of the src directory. So, following the example above, it would be `/home/kovid/work/calibre/src`. How to set environment variables depends on your Linux distribution and what shell you are using.

Once you have set the environment variable, open a new terminal and check that it was correctly set by using the command:

```
echo $CALIBRE_DEVELOP_FROM
```

Setting this environment variable means that calibre will now load all its Python code from the specified location.

That's it! You are now ready to start hacking on the calibre code. For example, open the file `src/calibre/__init__.py` in your favorite editor and add the line:

```
print ("Hello, world!")
```

near the top of the file. Now run the command **calibredb**. The very first line of output should be `Hello, world!`.

1.10.6 Having separate "normal" and "development" calibre installs on the same computer

The calibre source tree is very stable and rarely breaks, but if you feel the need to run from source on a separate test library and run the released calibre version with your everyday library, you can achieve this easily using .bat files or shell scripts to launch calibre. The example below shows how to do this on Windows using .bat files (the instructions for other platforms are the same, just use a shell script instead of a .bat file)

To launch the release version of calibre with your everyday library:

calibre-normal.bat:

```
calibre.exe "--with-library=C:\path\to\everyday\library folder"
```

calibre-dev.bat:

```
set CALIBRE_DEVELOP_FROM=C:\path\to\calibre\checkout\src
calibre.exe "--with-library=C:\path\to\test\library folder"
```

1.10.7 Debugging tips

Python is a dynamically typed language with excellent facilities for introspection. Kovid wrote the core calibre code without once using a debugger. There are many strategies to debug calibre code:

Using an interactive python interpreter

You can insert the following two lines of code to start an interactive python session at that point:

```
from calibre import ipython
ipython(locals())
```

When running from the command line, this will start an interactive Python interpreter with access to all locally defined variables (variables in the local scope). The interactive prompt even has TAB completion for object properties and you can use the various Python facilities for introspection, such as `dir()`, `type()`, `repr()`, etc.

Using print statements

This is Kovid's favorite way to debug. Simply insert print statements at points of interest and run your program in the terminal. For example, you can start the GUI from the terminal as:

```
calibre-debug -g
```

Similarly, you can start the ebook-viewer as:

```
calibre-debug -w /path/to/file/to/be/viewed
```

Using the debugger in PyDev

It is possible to get the debugger in PyDev working with the calibre development environment, see the forum thread[104].

Executing arbitrary scripts in the calibre python environment

The **calibre-debug** command provides a couple of handy switches to execute your own code, with access to the calibre modules:

```
calibre-debug -c "some python code"
```

is great for testing a little snippet of code on the command line. It works in the same way as the -c switch to the python interpreter:

```
calibre-debug -e myscript.py
```

[104] http://www.mobileread.com/forums/showthread.php?t=143208

can be used to execute your own Python script. It works in the same way as passing the script to the Python interpreter, except that the calibre environment is fully initialized, so you can use all the calibre code in your script.

1.10.8 Using calibre in your projects

It is possible to directly use calibre functions/code in your Python project. Two ways exist to do this:

Binary install of calibre

If you have a binary install of calibre, you can use the Python interpreter bundled with calibre, like this:

```
calibre-debug -e /path/to/your/python/script.py
```

Source install on Linux

In addition to using the above technique, if you do a source install on Linux, you can also directly import calibre, as follows:

```
import init_calibre
import calibre

print calibre.__version__
```

It is essential that you import the init_calibre module before any other calibre modules/packages as it sets up the interpreter to run calibre code.

1.11 Glossary

RSS **RSS** *(Really Simple Syndication)* is a web feed format that is used to publish frequently updated content, like news articles, blog posts, etc. It is a format that is particularly suited to being read by computers, and is therefore the preferred way of getting content from the web into an ebook. There are many other feed formats in use on the Internet, and calibre understands most of them. In particular, it has good support for the *ATOM* format, which is commonly used for blogs.

recipe A recipe is a set of instructions that teach calibre how to convert an online news source, such as a magazine or a blog, into an ebook. A recipe is essentially Python[105] code. As such, it is capable of converting arbitrarily complex news sources into ebooks. At the simplest level, it is just a set of variables, such as URLs, that give calibre enough information to go out onto the Internet and download the news.

HTML **HTML** *(Hyper Text Mark-Up Language)*, a subset of Standard Generalized Mark-Up Language (SGML) for electronic publishing, is the specific standard used for the World Wide Web.

CSS **CSS** *(Cascading Style Sheets)* is a language used to describe how an *HTML* document should be rendered (visual styling).

API **API** *(Application Programming Interface)* is a source code interface that a library provides to support requests for services to be made of it by computer programs.

LRF **LRF** The ebook format that is read by the SONY ebook readers.

URL **URL** *(Uniform Resource Locator)* for example: `http://example.com`

[105] http://www.python.org

regexp **Regular expressions** provide a concise and flexible means for identifying strings of text of interest, such as particular characters, words, or patterns of characters. See regexp syntax[106] for the syntax of regular expressions used in Python.

1.12 The main calibre user interface

1.12.1 The Graphical User Interface

The Graphical User Interface *(GUI)* provides access to all library management and ebook format conversion features. The basic workflow for using calibre is to first add books to the library from your hard disk. calibre will automatically try to read metadata from the books and add them to its internal database. Once they are in the database, you can perform various *Actions* (page 261) on them that include conversion from one format to another, transfer to the reading device, viewing on your computer, and editing metadata. The latter includes modifying the cover, description, and tags among other details. Note that calibre creates copies of the files you add to it. Your original files are left untouched.

The interface is divided into various sections:

- Actions (page 261)
- Preferences (page 267)
- Catalogs (page 268)
- Search & Sort (page 268)
- The Search Interface (page 269)
- Saving searches (page 271)
- Book Details (page 272)
- Tag Browser (page 273)
- Jobs (page 274)
- Keyboard Shortcuts (page 274)

Actions

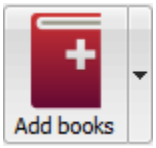


The actions toolbar provides convenient shortcuts to commonly used actions. If you right-click the buttons, you can perform variations on the default action. Please note that the actions toolbar will look slightly different depending on whether you have an ebook reader attached to your computer.

[106] http://docs.python.org/lib/re-syntax.html

- Add books (page 262)
- Edit metadata (page 263)
- Convert books (page 263)
- View (page 264)
- Send to device (page 264)
- Fetch news (page 264)
- Library (page 265)
- Device (page 265)
- Save to disk (page 266)
- Connect/Share (page 266)
- Remove books (page 267)

Add books



The *Add books* action has six variations accessed by doing a right-click on the button.

1. **Add books from a single directory**: Opens a file chooser dialog and allows you to specify which books in a directory should be added. This action is *context sensitive*, i.e. it depends on which *catalog* (page 268) you have selected. If you have selected the *Library*, books will be added to the library. If you have selected the ebook reader device, the books will be uploaded to the device, and so on.
2. **Add books from directories, including sub-directories (One book per directory, assumes every ebook file is the same book in a different format)**: Allows you to choose a directory. The directory and all its sub-directories are scanned recursively, and any ebooks found are added to the library. calibre assumes that each directory contains a single book. All ebook files in a directory are assumed to be the same book in different formats. This action is the inverse of the *Save to disk* (page 266) action, i.e. you can *Save to disk*, delete the books and re-add them with no lost information except for the date (this assumes you have not changed any of the setting for the Save to disk action).
3. **Add books from directories, including sub-directories (Multiple books per directory, assumes every ebook file is a different book)**: Allows you to choose a directory. The directory and all its sub-directories are scanned recursively and any ebooks found are added to the library. calibre assumes that each directory contains many books. All ebook files with the same name in a directory are assumed to be the same book in different formats. Ebooks with different names are added as different books.
4. **Add empty book. (Book Entry with no formats)**: Allows you to create a blank book record. This can be used to then manually fill out the information about a book that you may not have yet in your collection.
5. **Add from ISBN**: Allows you to add one or more books by entering their ISBNs.
6. **Add files to selected book records**: Allows you to add or update the files associated with an existing book in your library.

The *Add books* action can read metadata from a wide variety of ebook formats. In addition, it tries to guess metadata from the filename. See the *Guessing metadata from file names* (page 271) section, to learn how to configure this.

To add an additional format for an existing book use the *Edit metadata* (page 263) action.

Edit metadata



The *Edit metadata* action has four variations which can be accessed by doing a right-click on the button.

1. **Edit metadata individually**: Allows you to edit the metadata of books one-by-one with the option of fetching metadata, including covers, from the Internet. It also allows you to add or remove particular ebook formats from a book.
2. **Edit metadata in bulk**: Allows you to edit common metadata fields for large numbers of books simulataneously. It operates on all the books you have selected in the *Library view* (page 268).
3. **Download metadata and covers**: Downloads metadata and covers (if available) for the books that are selected in the book list.
4. **Merge book records**: Gives you the capability of merging the metadata and formats of two or more book records. You can choose to either delete or keep the records that were not clicked first.

For more details see *Editing Ebook Metadata* (page 318).

Convert books



Ebooks can be converted from a number of formats into whatever format your ebook reader prefers. Many ebooks available for purchase will be protected by Digital Rights Management[107] *(DRM)* technology. calibre will not convert these ebooks. It is easy to remove the DRM from many formats, but as this may be illegal, you will have to find tools to liberate your books yourself and then use calibre to convert them.

For most people, conversion should be a simple one-click affair. If you want to learn more about the conversion process, see *Ebook Conversion* (page 305).

The *Convert books* action has three variations, accessed by doing a right-click on the button.

1. **Convert individually**: Allows you to specify conversion options to customize the conversion of each selected ebook.
2. **Bulk convert**: Allows you to specify options only once to convert a number of ebooks in bulk.
3. **Create a catalog of the books in your calibre library**: Allows you to generate a complete listing of the books in your library, including all metadata, in several formats such as XML, CSV, BiBTeX, EPUB and MOBI. The catalog will contain all the books currently showing in the library view. This allows you to use the search features to limit the books to be catalogued. In addition, if you select multiple books using the mouse, only those books will be added to the catalog. If you generate the catalog in an ebook format such as EPUB, MOBI or AZW3, the next time you connect your ebook reader the catalog will be automatically sent to the device. For more information on how catalogs work, read the *Creating AZW3 • EPUB • MOBI Catalogs* (page 436).

[107] http://drmfree.calibre-ebook.com/about#drm

View



The *View* action displays the book in an ebook viewer program. calibre has a built-in viewer for many ebook formats. For other formats it uses the default operating system application. You can configure which formats should open with the internal viewer via Preferences->Behavior. If a book has more than one format, you can view a particular format by doing a right-click on the button.

Send to device



The *Send to device* action has eight variations, accessed by doing a right-click on the button.

1. **Send to main memory**: The selected books are transferred to the main memory of the ebook reader.
2. **Send to card (A)**: The selected books are transferred to the storage card (A) on the ebook reader.
3. **Send to card (B)**: The selected books are transferred to the storage card (B) on the ebook reader.
4. **Send specific format to**: The selected books are transferred to the selected storage location on the device, in the format that you specify.
5. **Eject device**: Detaches the device from calibre.
6. **Set default send to device action**: Allows you to specify which of the options, 1 through 5 above or 7 below, will be the default action when you click the main button.
7. **Send and delete from library**: The selected books are transferred to the selected storage location on the device and then **deleted** from the Library.
8. **Fetch Annotations (experimental)**: Transfers annotations you may have made on an ebook on your device to the comments metadata of the book in the calibre library.

You can control the file name and folder structure of files sent to the device by setting up a template in *Preferences->Import/Export->Sending books to devices*. Also see *The calibre template language* (page 375).

Fetch news



The *Fetch news* action downloads news from various websites and converts it into an ebook that can be read on your ebook reader. Normally, the newly created ebook is added to your ebook library, but if an ebook reader is connected at the time the download finishes, the news is also uploaded to the reader automatically.

The *Fetch news* action uses simple recipes (10-15 lines of code) for each news site. To learn how to create recipes for your own news sources, see *Adding your favorite news website* (page 341).

The *Fetch news* action has three variations, accessed by doing a right-click on the button.

1. **Schedule news download**: Allows you to schedule the download of of your selected news sources from a list of hundreds available. Scheduling can be set individually for each news source you select and the scheduling is flexible allowing you to select specific days of the week or a frequency of days between downloads.
2. **Add a custom news source**: Allows you to create a simple recipe for downloading news from a custom news site that you wish to access. Creating the recipe can be as simple as specifying an RSS news feed URL, or you can be more prescriptive by creating Python-based code for the task. For more information see *Adding your favorite news website* (page 341).
3. **Download all scheduled news sources**: Causes calibre to immediately begin downloading all news sources that you have scheduled.

Library



The *Library* action allows you to create, switch between, rename or remove a Library. calibre allows you to create as many libraries as you wish. You could, for instance, create a fiction library, a non-fiction library, a foreign language library, a project library, or any structure that suits your needs. Libraries are the highest organizational structure within calibre. Each library has its own set of books, tags, categories and base storage location.

1. **Switch/create library...**: Allows you to; a) connect to a pre-existing calibre library at another location, b) create an empty library at a new location or, c) move the current library to a newly specified location.
2. **Quick switch**: Allows you to switch between libraries that have been registered or created within calibre.
3. **Rename library**: Allows you to rename a Library.
4. **Delete library**: Allows you to unregister a library from calibre.
5. **<library name>**: Actions 5, 6 etc... give you immediate switch access between multiple libraries that you have created or attached to. This list contains only the 5 most frequently used libraries. For the complete list, use the Quick Switch menu.
6. **Library maintenance**: Allows you to check the current library for data consistency issues and restore the current library's database from backups.

Note: Metadata about your ebooks, e.g. title, author, and tags, is stored in a single file in your calibre library folder called metadata.db. If this file gets corrupted (a very rare event), you can lose the metadata. Fortunately, calibre automatically backs up the metadata for every individual book in the book's folder as an OPF file. By using the Restore Library action under Library Maintenance described above, you can have calibre rebuild the metadata.db file from the individual OPF files for you.

You can copy or move books between different libraries (once you have more than one library setup) by right clicking on the book and selecting the action *Copy to library*.

Device



The *Device* action allows you to view the books in the main memory or storage cards of your device, or to eject the device (detach it from calibre). This icon shows up automatically on the main calibre toolbar when you

connect a supported device. You can click on it to see the books on your device. You can also drag and drop books from your calibre library onto the icon to transfer them to your device. Conversely, you can drag and drop books from your device onto the library icon on the toolbar to transfer books from your device to the calibre library.

Save to disk



The *Save to disk* action has five variations, accessed by doing a right-click on the button.

1. **Save to disk**: Saves the selected books to disk organized in directories. The directory structure looks like:

```
Author_(sort)
    Title
        Book Files
```

You can control the file name and folder structure of files saved to disk by setting up a template in *Preferences->Import/Export->Saving books to disk*. Also see *The calibre template language* (page 375).

2. **Save to disk in a single directory**: Saves the selected books to disk in a single directory.

For 1. and 2., all available formats, as well as metadata, are stored to disk for each selected book. Metadata is stored in an OPF file. Saved books can be re-imported to the library without any loss of information by using the *Add books* (page 262) action.

3. **Save only *<your preferred>* format to disk**: Saves the selected books to disk in the directory structure as shown in (1.) but only in your preferred ebook format. You can set your preferred format in *Preferences->Behaviour->Preferred output format*

4. **Save only *<your preferred>* format to disk in a single directory**: Saves the selected books to disk in a single directory but only in your preferred ebook format. You can set your preferred format in *Preferences->Behaviour->Preferred output format*

5. **Save single format to disk...**: Saves the selected books to disk in the directory structure as shown in (1.) but only in the format you select from the pop-out list.

Connect/Share



The *Connect/Share* action allows you to manually connect to a device or folder on your computer. It also allows you to set up you calibre library for access via a web browser or email.

The *Connect/Share* action has four variations, accessed by doing a right-click on the button.

1. **Connect to folder**: Allows you to connect to any folder on your computer as though it were a device and use all the facilities calibre has for devices with that folder. Useful if your device cannot be supported by calibre but is available as a USB disk.

2. **Connect to iTunes**: Allows you to connect to your iTunes books database as though it were a device. Once the books are sent to iTunes, you can use iTunes to make them available to your various iDevices.

3. **Start Content Server**: Starts calibre's built-in web server. When started, your calibre library will be accessible via a web browser from the Internet (if you choose). You can configure how the web server is accessed by setting preferences at *Preferences->Sharing->Sharing over the net*
4. **Setup email based sharing of books**: Allows sharing of books and news feeds by email. After setting up email addresses for this option, calibre will send news updates and book updates to the entered email addresses. You can configure how calibre sends email by setting preferences at *Preferences->Sharing->Sharing books by email*. Once you have set up one or more email addresses, this menu entry will be replaced by menu entries to send books to the configured email addresses.

Remove books



The *Remove books* action **deletes books permanently**, so use it with care. It is *context sensitive*, i.e. it depends on which *catalog* (page 268) you have selected. If you have selected the *Library*, books will be removed from the library. If you have selected the ebook reader device, books will be removed from the device. To remove only a particular format for a given book use the *Edit metadata* (page 263) action. Remove books also has five variations which can be accessed by doing a right-click on the button.

1. **Remove selected books**: Allows you to **permanently** remove all books that are selected in the book list.
2. **Remove files of a specific format from selected books...**: Allows you to **permanently** remove ebook files of a specified format from books that are selected in the book list.
3. **Remove all formats from selected books, except...**: Allows you to **permanently** remove ebook files of any format except a specified format from books that are selected in the book list.
3. **Remove all formats from selected books**: Allows you to **permanently** remove all ebook files from books that are selected in the book list. Only the metadata will remain.
4. **Remove covers from selected books**: Allows you to **permanently** remove cover image files from books that are selected in the book list.
5. **Remove matching books from device**: Allows you to remove ebook files from a connected device that match the books that are selected in the book list.

Note: Note that when you use Remove books to delete books from your calibre library, the book record is permanently deleted, but on Windows and OS X the files are placed into the recycle bin. This allows you to recover them if you change your mind.

Preferences

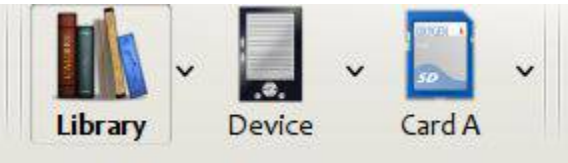


The *Preferences* action allows you to change the way various aspects of calibre work. It has four variations, accessed by doing a right-click on the button.

1. **Preferences**: Allows you to change the way various aspects of calibre work. Clicking the button also performs this action.

2. **Run welcome wizard**: Allows you to start the Welcome Wizard which appeared the first time you started calibre.
3. **Get plugins to enhance |app|**: Opens a new windows that shows plugins for calibre. These plugins are developed by third parties to extend calibre's functionality.
4. **Restart in debug mode**: Allows you to enable a debugging mode that can assist the calibre developers in solving problems you encounter with the program. For most users this should remain disabled unless instructed by a developer to enable it.

Catalogs



A *catalog* is a collection of books. calibre can manage two types of different catalogs:

1. **Library**: This is a collection of books stored in your calibre library on your computer.
2. **Device**: This is a collection of books stored in your ebook reader. It will be available when you connect the reader to your computer.

Many operations, such as adding books, deleting, viewing, etc., are context sensitive. So, for example, if you click the View button when you have the **Device** catalog selected, calibre will open the files on the device to view. If you have the **Library** catalog selected, files in your calibre library will be opened instead.

Search & Sort



The Search & Sort section allows you to perform several powerful actions on your book collections.

- You can sort them by title, author, date, rating, etc. by clicking on the column titles. You can also sub-sort, i.e. sort on multiple columns. For example, if you click on the title column and then the author column, the book will be sorted by author and then all the entries for the same author will be sorted by title.

- You can search for a particular book or set of books using the search bar. More on that below.
- You can quickly and conveniently edit metadata by double-clicking the entry you want changed in the list.
- You can perform *Actions* (page 261) on sets to books. To select multiple books you can either:
  - Keep the `Ctrl` key pressed and click on the books you want selected.
  - Keep the `Shift` key pressed and click on the starting and ending book of a range of books you want selected.
- You can configure which fields you want displayed by using the *Preferences* (page 267) dialog.

The Search Interface

You can search all the metadata by entering search terms in the search bar. Searches are case insensitive. For example:

```
Asimov Foundation format:lrf
```

This will match all books in your library that have `Asimov` and `Foundation` in their metadata and are available in the LRF format. Some more examples:

```
author:Asimov and not series:Foundation
title:"The Ring" or "This book is about a ring"
format:epub publisher:feedbooks.com
```

Searches are by default 'contains'. An item matches if the search string appears anywhere in the indicated metadata. Two other kinds of searches are available: equality search and search using regular expressions[108].

Equality searches are indicated by prefixing the search string with an equals sign (=). For example, the query `tag:"=science"` will match "science", but not "science fiction" or "hard science". Regular expression searches are indicated by prefixing the search string with a tilde (~). Any python-compatible regular expression[109] can be used. Note that backslashes used to escape special characters in requlars expressions must be doubled because single backslashes will be removed during query parsing. For example, to match a literal parenthesis you must enter `\\(`. Regular expression searches are 'contains' searches unless the expression contains anchors.

Should you need to search for a string with a leading equals or tilde, prefix the string with a backslash.

Enclose search strings with quotes (") if the string contains parenthesis or spaces. For example, to search for the tag `Science Fiction` you would need to search for `tag:"=science fiction"`. If you search for `tag:=science fiction` you will find all books with the tag 'science' and containing the word 'fiction' in any metadata.

You can build advanced search queries easily using the *Advanced Search Dialog* accessed by clicking the button .

Available fields for searching are: `tag`, `title`, `author`, `publisher`, `series`, `series_index`, `rating`, `cover`, `comments`, `format`, `identifiers`, `date`, `pubdate`, `search`, `size` and custom columns. If a device is plugged in, the `ondevice` field becomes available, when searching the calibre library view. To find the search name (actually called the *lookup name*) for a custom column, hover your mouse over the column header in the library view.

The syntax for searching for dates is:

```
pubdate:>2000-1 Will find all books published after Jan, 2000
date:<=2000-1-3 Will find all books added to calibre before 3 Jan, 2000
pubdate:=2009 Will find all books published in 2009
```

[108] http://en.wikipedia.org/wiki/Regular_expression
[109] http://docs.python.org/library/re.html

If the date is ambiguous, the current locale is used for date comparison. For example, in an mm/dd/yyyy locale 2/1/2009 is interpreted as 1 Feb 2009. In a dd/mm/yyyy locale it is interpreted as 2 Jan 2009. Some special date strings are available. The string `today` translates to today's date, whatever it is. The strings `yesterday` and `thismonth` (or the translated equivalent in the current language) also work. In addition, the string `daysago` (also translated) can be used to compare to a date some number of days ago. For example:

```
date:>10daysago
date:<=45daysago
```

You can search for books that have a format of a certain size like this:

```
size:>1.1M Will find books with a format larger than 1.1MB
size:<=1K  Will find books with a format smaller than 1KB
```

Dates and numeric fields support the relational operators = (equals), > (greater than), >= (greater than or equal to), < (less than), <= (less than or equal to), and != (not equal to). Rating fields are considered to be numeric. For example, the search `rating:>=3` will find all books rated 3 or higher.

You can search for the number of items in multiple-valued fields such as tags. These searches begin with the character `#`, then use the same syntax as numeric fields. For example, to find all books with more than 4 tags use `tags:#>4`. To find all books with exactly 10 tags use `tags:#=10`.

Series indices are searchable. For the standard series, the search name is 'series_index'. For custom series columns, use the column search name followed by _index. For example, to search the indices for a custom series column named `#my_series`, you would use the search name `#my_series_index`. Series indices are numbers, so you can use the relational operators described above.

The special field `search` is used for saved searches. So if you save a search with the name "My spouse's books" you can enter `search:"My spouse's books"` in the search bar to reuse the saved search. More about saving searches below.

You can search for the absence or presence of a field using the special "true" and "false" values. For example:

```
cover:false will give you all books without a cover
series:true will give you all books that belong to a series
comments:false will give you all books with an empty comment
format:false will give you all books with no actual files (empty records)
```

Yes/no custom columns are searchable. Searching for `false`, `empty`, or `blank` will find all books with undefined values in the column. Searching for `true` will find all books that do not have undefined values in the column. Searching for `yes` or `checked` will find all books with `Yes` in the column. Searching for `no` or `unchecked` will find all books with `No` in the column. Note that the words `yes`, `no`, `blank`, `empty`, `checked` and `unchecked` are translated; you must use the current language's equivalent word. The words `true` and `false` and the special values `_yes` and `_no` are not translated.

Hierarchical items (e.g. A.B.C) use an extended syntax to match initial parts of the hierarchy. This is done by adding a period between the exact match indicator (=) and the text. For example, the query `tags:=.A` will find the tags *A* and *A.B*, but will not find the tags *AA* or *AA.B*. The query `tags:=.A.B` will find the tags *A.B* and *A.B.C*, but not the tag *A*.

Identifiers (e.g., isbn, doi, lccn etc) also use an extended syntax. First, note that an identifier has the form `type:value`, as in `isbn:123456789`. The extended syntax permits you to specify independently which type and value to search for. Both the type and the value parts of the query can use *equality*, *contains*, or *regular expression* matches. Examples:

- `identifiers:true` will find books with any identifier.
- `identifiers:false` will find books with no identifier.
- `identifiers:123` will search for books with any type having a value containing *123*.

- `identifiers:=123456789` will search for books with any type having a value equal to *123456789*.
- `identifiers:=isbn:` and `identifiers:isbn:true` will find books with a type equal to isbn having any value
- `identifiers:=isbn:false` will find books with no type equal to isbn.
- `identifiers:=isbn:123` will find books with a type equal to isbn having a value containing *123*.
- `identifiers:=isbn:=123456789` will find books with a type equal to isbn having a value equal to *123456789*.
- `identifiers:i:1` will find books with a type containing an *i* having a value containing a *1*.

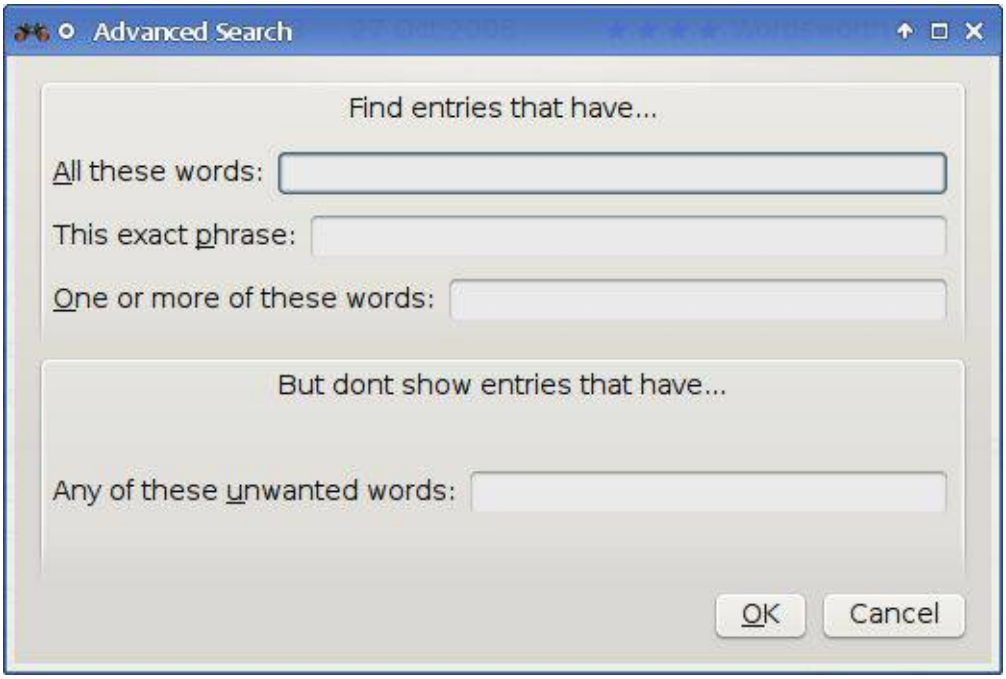


Figure 1.3: *Advanced Search Dialog*

Saving searches

calibre allows you to save a frequently used search under a special name and then reuse that search with a single click. To do this, create your search either by typing it in the search bar or using the Tag Browser. Then type the name you would like to give to the search in the Saved Searches box next to the search bar. Click the plus icon next to the saved searches box to save the search.

Now you can access your saved search in the Tag Browser under "Searches". A single click will allow you to reuse any arbitrarily complex search easily, without needing to re-create it.

Guessing metadata from file names

In the *Add/Save* section of the configuration dialog, you can specify a regular expression that calibre will use to try and guess metadata from the names of ebook files that you add to the library. The default regular expression is:

```
title - author
```

i.e., it assumes that all characters up to the first – are the title of the book and subsequent characters are the author of the book. For example, the filename:

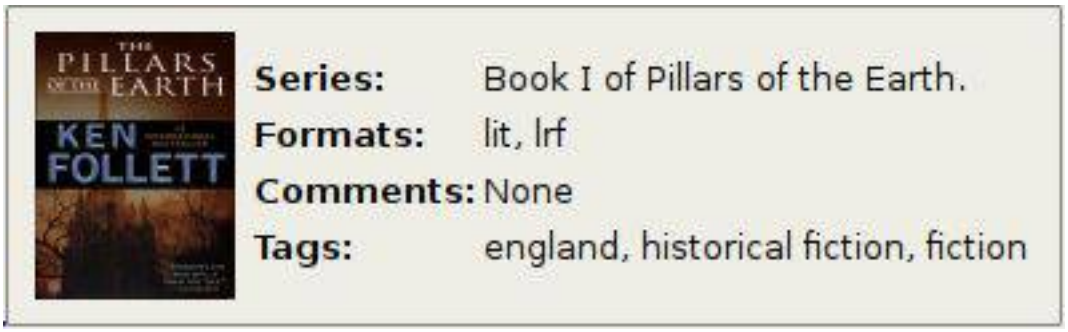
```
Foundation and Earth - Isaac Asimov.txt
```

will be interpreted to have the title: Foundation and Earth and author: Isaac Asimov

---

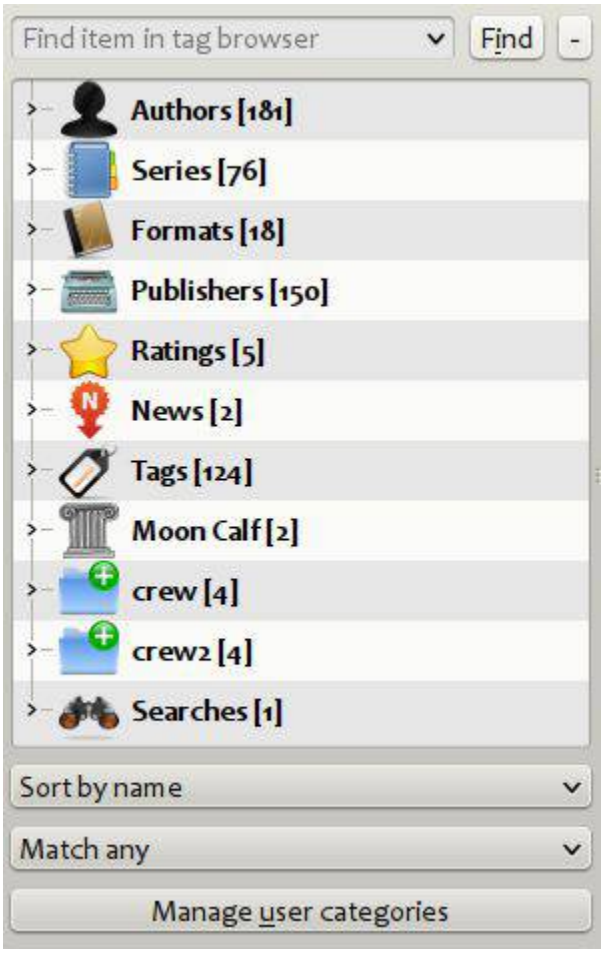
Tip: If the filename does not contain the hyphen, the regular expression will fail.

Book Details



The Book Details display shows extra information and the cover for the currently selected book.

Tag Browser



The Tag Browser allows you to easily browse your collection by Author/Tags/Series/etc. If you click on any item in the Tag Browser, for example the author name Isaac Asimov, then the list of books to the right is restricted to showing books by that author. You can click on category names as well. For example, clicking on "Series" will show you all books in any series.

The first click on an item will restrict the list of books to those that contain or match the item. Continuing the above example, clicking on Isaac Asimov will show books by that author. Clicking again on the item will change what is shown, depending on whether the item has children (see sub-categories and hierarchical items below). Continuing the Isaac Asimov example, clicking again on Isaac Asimov will restrict the list of books to those not by Isaac Asimov. A third click will remove the restriction, showing all books. If you hold down the Ctrl or Shift keys and click on multiple items, then restrictions based on multiple items are created. For example you could hold Ctrl and click on the tags History and Europe for finding books on European history. The Tag Browser works by constructing search expressions that are automatically entered into the Search bar. Looking at what the Tag Browser generates is a good way to learn how to construct basic search expressions.

Items in the Tag browser have their icons partially colored. The amount of color depends on the average rating of the books in that category. So for example if the books by Isaac Asimov have an average of four stars, the icon for Isaac Asimov in the Tag Browser will be 4/5th colored. You can hover your mouse over the icon to see the average rating.

The outer-level items in the tag browser, such as Authors and Series, are called categories. You can create your own categories, called User Categories, which are useful for organizing items. For example, you can use the User Categories Editor (click the Manage User Categories button) to create a user category called Favorite Authors, then put the items for your favorites into the category. User categories can have sub-categories. For example, the user category

Favorites.Authors is a sub-category of Favorites. You might also have Favorites.Series, in which case there will be two sub-categories under Favorites. Sub-categories can be created by right-clicking on a user category, choosing "Add sub-category to ...", and entering the sub-category name; or by using the User Categories Editor by entering names like the Favorites example above.

You can search user categories in the same way as built-in categories, by clicking on them. There are four different searches cyc

1. "everything matching an item in the category" indicated by a single green plus sign.
2. "everything matching an item in the category or its sub-categories" indicated by two green plus signs.
3. "everything not matching an item in the category" shown by a single red minus sign.
4. "everything not matching an item in the category or its sub-categories" shown by two red minus signs.

It is also possible to create hierarchies inside some of the text categories such as tags, series, and custom columns. These hierarchies show with the small triangle, permitting the sub-items to be hidden. To use hierarchies of items in a category, you must first go to Preferences->Look & Feel and enter the category name(s) into the "Categories with hierarchical items" box. Once this is done, items in that category that contain periods will be shown using the small triangle. For example, assume you create a custom column called "Genre" and indicate that it contains hierarchical items. Once done, items such as Mystery.Thriller and Mystery.English will display as Mystery with the small triangle next to it. Clicking on the triangle will show Thriller and English as sub-items. See *Managing subgroups of books, for example "genre"* (page 368) for more information.

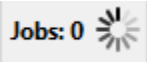
Hierarchical items (items with children) use the same four 'click-on' searches as user categories. Items that do not have children use two of the searches: "everything matching" and "everything not matching".

You can drag and drop items in the Tag browser onto user categories to add them to that category. If the source is a user category, holding the shift key while dragging will move the item to the new category. You can also drag and drop books from the book list onto items in the Tag Browser; dropping a book on an item causes that item to be automatically applied to the dropped books. For example, dragging a book onto Isaac Asimov will set the author of that book to Isaac Asimov. Dropping it onto the tag History will add the tag History to the book's tags.

There is a search bar at the top of the Tag Browser that allows you to easily find any item in the Tag Browser. In addition, you can right click on any item and choose one of several operations. Some examples are to hide the it, rename it, or open a "Manage x" dialog that allows you to manage items of that kind. For example, the "Manage Authors" dialog allows you to rename authors and control how their names are sorted.

You can control how items are sorted in the Tag browser via the box at the bottom of the Tag Browser. You can choose to sort by name, average rating or popularity (popularity is the number of books with an item in your library; for example, the popularity of Isaac Asimov is the number of books in your library by Isaac Asimov).

Jobs



The Jobs panel shows the number of currently running jobs. Jobs are tasks that run in a separate process. They include converting ebooks and talking to your reader device. You can click on the jobs panel to access the list of jobs. Once a job has completed you can see a detailed log from that job by double-clicking it in the list. This is useful to debug jobs that may not have completed successfully.

Keyboard Shortcuts

Calibre has several keyboard shortcuts to save you time and mouse movement. These shortcuts are active in the book list view (when you're not editing the details of a particular book), and most of them affect the title you have selected.

The calibre ebook viewer has its own shortcuts which can be customised by clicking the Preferences button in the viewer.

Note: Note: The Calibre keyboard shortcuts do not require a modifier key (Command, Option, Control, etc.), unless specifically noted. You only need to press the letter key, e.g. E to edit.

Table 1.3: Keyboard Shortcuts

Keyboard Shortcut	Action
F2 (Enter in OS X)	Edit the metadata of the currently selected field in the book list.
A	Add Books
Shift+A	Add Formats to the selected books
C	Convert selected Books
D	Send to device
Del	Remove selected Books
E	Edit metadata of selected books
G	Get Books
I	Show book details
M	Merge selected records
Alt+M	Merge selected records, keeping originals
O	Open containing folder
P	Polish books
S	Save to Disk
V	View
Alt+V/Cmd+V in OS X	View specific format
Alt+Shift+J	Toggle jobs list
Alt+Shift+B	Toggle Cover Browser
Alt+Shift+D	Toggle Book Details panel
Alt+Shift+T	Toggle Tag Browser
Alt+A	Show books by the same author as the current book
Alt+T	Show books with the same tags as current book
Alt+P	Show books by the same publisher as current book
Alt+Shift+S	Show books in the same series as current book
/, Ctrl+F	Focus the search bar
Shift+Ctrl+F	Open the advanced search dialog
Esc	Clear the current search
N or F3	Find the next book that matches the current search (only works if the highlight checkbox next to the se
Shift+N or Shift+F3	Find the next book that matches the current search (only works if the highlight checkbox next to the se
Ctrl+D	Download metadata and shortcuts
Ctrl+R	Restart calibre
Ctrl+Shift+R	Restart calibre in debug mode
Shift+Ctrl+E	Add empty books to calibre
Ctrl+Q	Quit calibre

1.13 Adding your favorite news website to calibre

1.13.1 Adding your favorite news website

calibre has a powerful, flexible and easy-to-use framework for downloading news from the Internet and converting it into an ebook. The following will show you, by means of examples, how to get news from various websites.

To gain an understanding of how to use the framework, follow the examples in the order listed below:

- Completely automatic fetching (page 342)
  - portfolio.com (page 342)
  - bbc.co.uk (page 344)
- Customizing the fetch process (page 344)
  - Using the print version of bbc.co.uk (page 344)
  - Replacing article styles (page 345)
  - Slicing and dicing (page 346)
  - Real life example (page 357)
- Tips for developing new recipes (page 359)
- Further reading (page 360)
- API documentation (page 360)

Completely automatic fetching

If your news source is simple enough, calibre may well be able to fetch it completely automatically, all you need to do is provide the URL. calibre gathers all the information needed to download a news source into a *recipe*. In order to tell calibre about a news source, you have to create a *recipe* for it. Let's see some examples:

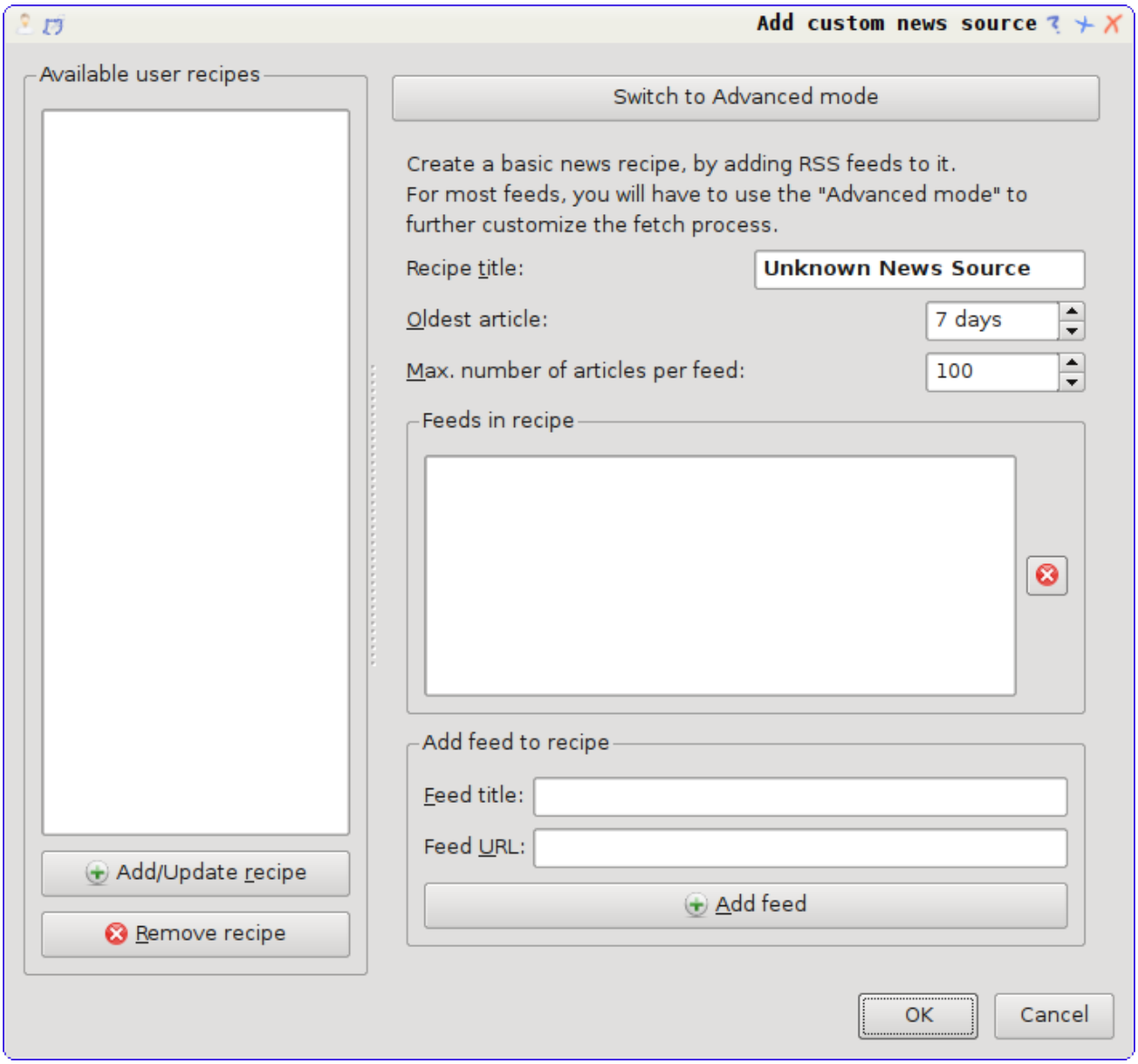
portfolio.com

portfolio.com is the website for *Condé Nast Portfolio*, a business related magazine. In order to download articles from the magazine and convert them to ebooks, we rely on the *RSS* feeds of portfolio.com. A list of such feeds is available at http://www.portfolio.com/rss/.

Lets pick a couple of feeds that look interesting:

1. Business Travel: http://feeds.portfolio.com/portfolio/businesstravel
2. Tech Observer: http://feeds.portfolio.com/portfolio/thetechobserver

I got the URLs by clicking the little orange RSS icon next to each feed name. To make calibre download the feeds and convert them into an ebook, you should right click the *Fetch news* button and then the *Add a custom news source* menu item. A dialog similar to that shown below should open up.



First enter `Portfolio` into the *Recipe title* field. This will be the title of the ebook that will be created from the articles in the above feeds.

The next two fields (*Oldest article* and *Max. number of articles*) allow you some control over how many articles should be downloaded from each feed, and they are pretty self explanatory.

To add the feeds to the recipe, enter the feed title and the feed URL and click the *Add feed* button. Once you have added both feeds, simply click the *Add/update recipe* button and you're done! Close the dialog.

To test your new *recipe*, click the *Fetch news* button and in the *Custom news sources* sub-menu click *Portfolio*. After a couple of minutes, the newly downloaded Portfolio ebook will appear in the main library view (if you have your reader connected, it will be put onto the reader instead of into the library). Select it and hit the *View* button to read!

The reason this worked so well, with so little effort is that *portfolio.com* provides *full-content RSS* feeds, i.e., the article content is embedded in the feed itself. For most news sources that provide news in this fashion, with *full-content* feeds, you don't need any more effort to convert them to ebooks. Now we will look at a news source that does not provide full content feeds. In such feeds, the full article is a webpage and the feed only contains a link to the webpage with a short summary of the article.

bbc.co.uk

Lets try the following two feeds from *The BBC*:

1. News Front Page: http://newsrss.bbc.co.uk/rss/newsonline_world_edition/front_page/rss.xml
2. Science/Nature: http://newsrss.bbc.co.uk/rss/newsonline_world_edition/science/nature/rss.xml

Follow the procedure outlined in *portfolio.com* (page 342) to create a recipe for *The BBC* (using the feeds above). Looking at the downloaded ebook, we see that calibre has done a creditable job of extracting only the content you care about from each article's webpage. However, the extraction process is not perfect. Sometimes it leaves in undesirable content like menus and navigation aids or it removes content that should have been left alone, like article headings. In order, to have perfect content extraction, we will need to customize the fetch process, as described in the next section.

Customizing the fetch process

When you want to perfect the download process, or download content from a particularly complex website, you can avail yourself of all the power and flexibility of the *recipe* framework. In order to do that, in the *Add custom news sources* dialog, simply click the *Switch to Advanced mode* button.

The easiest and often most productive customization is to use the print version of the online articles. The print version typically has much less cruft and translates much more smoothly to an ebook. Let's try to use the print version of the articles from *The BBC*.

Using the print version of bbc.co.uk

The first step is to look at the ebook we downloaded previously from *bbc.co.uk* (page 344). At the end of each article, in the ebook is a little blurb telling you where the article was downloaded from. Copy and paste that URL into a browser. Now on the article webpage look for a link that points to the "Printable version". Click it to see the print version of the article. It looks much neater! Now compare the two URLs. For me they were:

Article URL http://news.bbc.co.uk/2/hi/science/nature/7312016.stm

Print version URL http://newsvote.bbc.co.uk/mpapps/pagetools/print/news.bbc.co.uk/2/hi/science/nature/7312016.stm

So it looks like to get the print version, we need to prefix every article URL with:

newsvote.bbc.co.uk/mpapps/pagetools/print/

Now in the *Advanced Mode* of the Custom news sources dialog, you should see something like (remember to select *The BBC* recipe before switching to advanced mode):



You can see that the fields from the *Basic mode* have been translated to python code in a straightforward manner. We need to add instructions to this recipe to use the print version of the articles. All that's needed is to add the following two lines:

```
def print_version(self, url):
    return url.replace('http://', 'http://newsvote.bbc.co.uk/mpapps/pagetools/print/')
```

This is python, so indentation is important. After you've added the lines, it should look like:

Recipe source code (python)

```
class AdvancedUserRecipe1206418393(BasicNewsRecipe):
    title          = u'The BBC'
    oldest_article = 7
    max_articles_per_feed = 100

    feeds          = [(u'News Front Page', u'http://newsrss.bbc.co.uk/rss/newsonlir

    def print_version(self, url):
        return url.replace('http://', 'http://newsvote.bbc.co.uk/mpapps/pagetools/p
```

In the above, `def print_version(self, url)` defines a *method* that is called by calibre for every article. `url` is the URL of the original article. What `print_version` does is take that url and replace it with the new URL that points to the print version of the article. To learn about python[110] see the tutorial[111].

Now, click the *Add/update recipe* button and your changes will be saved. Re-download the ebook. You should have a much improved ebook. One of the problems with the new version is that the fonts on the print version webpage are too small. This is automatically fixed when converting to an ebook, but even after the fixing process, the font size of the menus and navigation bar to become too large relative to the article text. To fix this, we will do some more customization, in the next section.

Replacing article styles

In the previous section, we saw that the font size for articles from the print version of *The BBC* was too small. In most websites, *The BBC* included, this font size is set by means of *CSS* stylesheets. We can disable the fetching of such stylesheets by adding the line:

```
no_stylesheets = True
```

The recipe now looks like:

Recipe source code (python)

```
class AdvancedUserRecipe1206419520(BasicNewsRecipe):
    title          = u'The BBC'
    oldest_article = 7
    max_articles_per_feed = 100
    no_stylesheets = True

    feeds          = [(u'News Front Page', u'http://newsrss.bbc.co.uk/rss/news

    def print_version(self, url):
        return url.replace('http://', 'http://newsvote.bbc.co.uk/mpapps/pageto
```

[110] http://www.python.org
[111] http://docs.python.org/tut/

The new version looks pretty good. If you're a perfectionist, you'll want to read the next section, which deals with actually modifying the downloaded content.

Slicing and dicing

calibre contains very powerful and flexible abilities when it comes to manipulating downloaded content. To show off a couple of these, let's look at our old friend the *The BBC* (page 345) recipe again. Looking at the source code (*HTML*) of a couple of articles (print version), we see that they have a footer that contains no useful information, contained in

```
<div class="footer">
...
</div>
```

This can be removed by adding:

```
remove_tags    = [dict(name='div', attrs={'class':'footer'})]
```

to the recipe. Finally, lets replace some of the *CSS* that we disabled earlier, with our own *CSS* that is suitable for conversion to an ebook:

```
extra_css      = '.headline {font-size: x-large;} \n .fact { padding-top: 10pt  }'
```

With these additions, our recipe has become "production quality", indeed it is very close to the actual recipe used by calibre for the *BBC*, shown below:

```
##
## Title:         BBC News, Sport, and Blog Calibre Recipe
## Contact:       mattst - jmstanfield@gmail.com
##
## License:       GNU General Public License v3 - http://www.gnu.org/copyleft/gpl.html
## Copyright:     mattst - jmstanfield@gmail.com
##
## Written:       November 2011
## Last Edited:   2011-11-19
##

__license__     = 'GNU General Public License v3 - http://www.gnu.org/copyleft/gpl.html'
__copyright__   = 'mattst - jmstanfield@gmail.com'


'''
BBC News, Sport, and Blog Calibre Recipe
'''

# Import the regular expressions module.
import re

# Import the BasicNewsRecipe class which this class extends.
from calibre.web.feeds.recipes import BasicNewsRecipe

class BBCNewsSportBlog(BasicNewsRecipe):

    #
    #    **** IMPORTANT USERS READ ME ****
    #
    #  First select the feeds you want then scroll down below the feeds list
```

```
#  and select the values you want for the other user preferences, like
#  oldest_article and such like.
#
#
#  Select the BBC rss feeds which you want in your ebook.
#  Selected feed have NO '#' at their start, de-selected feeds begin with a '#'.
#
#  Eg.  ("News Home", "http://feeds.bbci.co.uk/... - include feed.
#  Eg. #("News Home", "http://feeds.bbci.co.uk/... - do not include feed.
#
# There are 68 feeds below which constitute the bulk of the available rss
# feeds on the BBC web site. These include 5 blogs by editors and
# correspondants, 16 sports feeds, 15 'sub' regional feeds (Eg. North West
# Wales, Scotland Business), and 7 Welsh language feeds.
#
# Some of the feeds are low volume (Eg. blogs), or very low volume (Eg. Click)
# so if "oldest_article = 1.5" (only articles published in the last 36 hours)
# you may get some 'empty feeds' which will not then be included in the ebook.
#
# The 15 feeds currently selected below are simply my default ones.
#
# Note: With all 68 feeds selected, oldest_article set to 2,
# max_articles_per_feed set to 100, and simultaneous_downloads set to 10,
# the ebook creation took 29 minutes on my speedy 100 mbps net connection,
# fairly high-end desktop PC running Linux (Ubuntu Lucid-Lynx).
# More realistically with 15 feeds selected, oldest_article set to 1.5,
# max_articles_per_feed set to 100, and simultaneous_downloads set to 20,
# it took 6 minutes. If that's too slow increase 'simultaneous_downloads'.
#
# Select / de-select the feeds you want in your ebook.
#
feeds = [
          ("News Home", "http://feeds.bbci.co.uk/news/rss.xml"),
          ("UK", "http://feeds.bbci.co.uk/news/uk/rss.xml"),
          ("World", "http://feeds.bbci.co.uk/news/world/rss.xml"),
          #("England", "http://feeds.bbci.co.uk/news/england/rss.xml"),
          #("Scotland", "http://feeds.bbci.co.uk/news/scotland/rss.xml"),
          #("Wales", "http://feeds.bbci.co.uk/news/wales/rss.xml"),
          #("N. Ireland", "http://feeds.bbci.co.uk/news/northern_ireland/rss.xml"),
          #("Africa", "http://feeds.bbci.co.uk/news/world/africa/rss.xml"),
          #("Asia", "http://feeds.bbci.co.uk/news/world/asia/rss.xml"),
          #("Europe", "http://feeds.bbci.co.uk/news/world/europe/rss.xml"),
          #("Latin America", "http://feeds.bbci.co.uk/news/world/latin_america/rss.xml"),
          #("Middle East", "http://feeds.bbci.co.uk/news/world/middle_east/rss.xml"),
          ("US & Canada", "http://feeds.bbci.co.uk/news/world/us_and_canada/rss.xml"),
          ("Politics", "http://feeds.bbci.co.uk/news/politics/rss.xml"),
          ("Science/Environment", "http://feeds.bbci.co.uk/news/science_and_environment/rss.xml"
          ("Technology", "http://feeds.bbci.co.uk/news/technology/rss.xml"),
          ("Magazine", "http://feeds.bbci.co.uk/news/magazine/rss.xml"),
          ("Entertainment/Arts", "http://feeds.bbci.co.uk/news/entertainment_and_arts/rss.xml"),
          #("Health", "http://feeds.bbci.co.uk/news/health/rss.xml"),
          #("Education/Family", "http://feeds.bbci.co.uk/news/education/rss.xml"),
          ("Business", "http://feeds.bbci.co.uk/news/business/rss.xml"),
          ("Special Reports", "http://feeds.bbci.co.uk/news/special_reports/rss.xml"),
          ("Also in the News", "http://feeds.bbci.co.uk/news/also_in_the_news/rss.xml"),
          #("Newsbeat", "http://www.bbc.co.uk/newsbeat/rss.xml"),
          #("Click", "http://newsrss.bbc.co.uk/rss/newsonline_uk_edition/programmes/click_online/
          ("Blog: Nick Robinson (Political Editor)", "http://feeds.bbci.co.uk/news/correspondents
```

```
            #("Blog: Mark D'Arcy (Parliamentary Correspondent)", "http://feeds.bbci.co.uk/news/cor
            #("Blog: Robert Peston (Business Editor)", "http://feeds.bbci.co.uk/news/correspondents
            #("Blog: Stephanie Flanders (Economics Editor)", "http://feeds.bbci.co.uk/news/correspo
            ("Blog: Rory Cellan-Jones (Technology correspondent)", "http://feeds.bbci.co.uk/news/co
            ("Sport Front Page", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/front_page/rs
            #("Football", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/football/rss.xml"),
            #("Cricket", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/cricket/rss.xml"),
            #("Rugby Union", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/rugby_union/rss.x
            #("Rugby League", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/rugby_league/rss
            #("Tennis", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/tennis/rss.xml"),
            #("Golf", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/golf/rss.xml"),
            #("Motorsport", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/motorsport/rss.xml
            #("Boxing", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/boxing/rss.xml"),
            #("Athletics", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/athletics/rss.xml",
            #("Snooker", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/snooker/
            #("Horse Racing", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/hor
            #("Cycling", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/cycling/
            #("Disability Sport", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports
            #("Other Sport", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/rss
            #("Olympics 2012", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/o
            #("N. Ireland Politics", "http://feeds.bbci.co.uk/news/northern_ireland/northern_irelan
            #("Scotland Politics", "http://feeds.bbci.co.uk/news/scotland/scotland_politics/rss.xml
            #("Scotland Business", "http://feeds.bbci.co.uk/news/scotland/scotland_business/rss.xml
            #("E. Scotland, Edinburgh & Fife", "http://feeds.bbci.co.uk/news/scotland/edinburgh_eas
            #("W. Scotland & Glasgow", "http://feeds.bbci.co.uk/news/scotland/glasgow_and_west/rss
            #("Highlands & Islands", "http://feeds.bbci.co.uk/news/scotland/highlands_and_islands/r
            #("NE. Scotland, Orkney & Shetland", "http://feeds.bbci.co.uk/news/scotland/north_east_
            #("South Scotland", "http://feeds.bbci.co.uk/news/scotland/south_scotland/rss.xml"),
            #("Central Scotland & Tayside", "http://feeds.bbci.co.uk/news/scotland/tayside_and_cent
            #("Wales Politics", "http://feeds.bbci.co.uk/news/wales/wales_politics/rss.xml"),
            #("NW. Wales", "http://feeds.bbci.co.uk/news/wales/north_west_wales/rss.xml"),
            #("NE. Wales", "http://feeds.bbci.co.uk/news/wales/north_east_wales/rss.xml"),
            #("Mid. Wales", "http://feeds.bbci.co.uk/news/wales/mid_wales/rss.xml"),
            #("SW. Wales", "http://feeds.bbci.co.uk/news/wales/south_west_wales/rss.xml"),
            #("SE. Wales", "http://feeds.bbci.co.uk/news/wales/south_east_wales/rss.xml"),
            #("Newyddion - News in Welsh", "http://feeds.bbci.co.uk/newyddion/rss.xml"),
            #("Gwleidyddiaeth", "http://feeds.bbci.co.uk/newyddion/gwleidyddiaeth/rss.xml"),
            #("Gogledd-Ddwyrain", "http://feeds.bbci.co.uk/newyddion/gogledd-ddwyrain/rss.xml"),
            #("Gogledd-Orllewin", "http://feeds.bbci.co.uk/newyddion/gogledd-orllewin/rss.xml"),
            #("Canolbarth", "http://feeds.bbci.co.uk/newyddion/canolbarth/rss.xml"),
            #("De-Ddwyrain", "http://feeds.bbci.co.uk/newyddion/de-ddwyrain/rss.xml"),
            #("De-Orllewin", "http://feeds.bbci.co.uk/newyddion/de-orllewin/rss.xml"),
        ]


#     **** SELECT YOUR USER PREFERENCES ****

# Title to use for the ebook.
#
title = 'BBC News'

# A brief description for the ebook.
#
description = u'BBC web site ebook created using rss feeds.'

# The max number of articles which may be downloaded from each feed.
# I've never seen more than about 70 articles in a single feed in the
# BBC feeds.
```

```
#
max_articles_per_feed = 100

# The max age of articles which may be downloaded from each feed. This is
# specified in days - note fractions of days are allowed, Eg. 2.5 (2 and a
# half days). My default of 1.5 days is the last 36 hours, the point at
# which I've decided 'news' becomes 'old news', but be warned this is not
# so good for the blogs, technology, magazine, etc., and sports feeds.
# You may wish to extend this to 2-5 but watch out ebook creation time will
# increase as well. Setting this to 30 will get everything (AFAICT) as long
# as max_articles_per_feed remains set high (except for 'Click' which is
# v. low volume and its currently oldest article is 4th Feb 2011).
#
oldest_article = 1.5

# Number of simultaneous downloads. 20 is consistantly working fine on the
# BBC News feeds with no problems. Speeds things up from the defualt of 5.
# If you have a lot of feeds and/or have increased oldest_article above 2
# then you may wish to try increasing simultaneous_downloads to 25-30,
# Or, of course, if you are in a hurry. [I've not tried beyond 20.]
#
simultaneous_downloads = 20

# Timeout for fetching files from the server in seconds. The default of
# 120 seconds, seems somewhat excessive.
#
timeout = 30

# The format string for the date shown on the ebook's first page.
# List of all values: http://docs.python.org/library/time.html
# Default in news.py has a leading space so that's mirrored here.
# As with 'feeds' select/de-select by adding/removing the initial '#',
# only one timefmt should be selected, here's a few to choose from.
#
timefmt = ' [%a, %d %b %Y]'              # [Fri, 14 Nov 2011] (Calibre default)
#timefmt = ' [%a, %d %b %Y %H:%M]'       # [Fri, 14 Nov 2011 18:30]
#timefmt = ' [%a, %d %b %Y %I:%M %p]'    # [Fri, 14 Nov 2011 06:30 PM]
#timefmt = ' [%d %b %Y]'                 # [14 Nov 2011]
#timefmt = ' [%d %b %Y %H:%M]'           # [14 Nov 2011 18.30]
#timefmt = ' [%Y-%m-%d]'                 # [2011-11-14]
#timefmt = ' [%Y-%m-%d-%H-%M]'           # [2011-11-14-18-30]



#
#    **** IMPORTANT ****
#
#    DO NOT EDIT BELOW HERE UNLESS YOU KNOW WHAT YOU ARE DOING.
#
#    DO NOT EDIT BELOW HERE UNLESS YOU KNOW WHAT YOU ARE DOING.
#
#    I MEAN IT, YES I DO, ABSOLUTELY, AT YOU OWN RISK. :)
#
#    **** IMPORTANT ****
#
```

```
    # Author of this recipe.
    __author__ = 'mattst'

    # Specify English as the language of the RSS feeds (ISO-639 code).
    language = 'en_GB'

    # Set tags.
    tags = 'news, sport, blog'

    # Set publisher and publication type.
    publisher = 'BBC'
    publication_type = 'newspaper'

    # Disable stylesheets from site.
    no_stylesheets = True

    # Specifies an override encoding for sites that have an incorrect charset
    # specified. Default of 'None' says to auto-detect. Some other BBC recipes
    # use 'utf8', which works fine (so use that if necessary) but auto-detecting
    # with None is working fine, so stick with that for robustness.
    encoding = None

    # Sets whether a feed has full articles embedded in it. The BBC feeds do not.
    use_embedded_content = False

    # Removes empty feeds - why keep them!?
    remove_empty_feeds = True

    # Create a custom title which fits nicely in the Kindle title list.
    # Requires "import time" above class declaration, and replacing
    # title with custom_title in conversion_options (right column only).
    # Example of string below: "BBC News - 14 Nov 2011"
    #
    # custom_title = "BBC News - " + time.strftime('%d %b %Y')

    '''
    # Conversion options for advanced users, but don't forget to comment out the
    # current conversion_options below. Avoid setting 'linearize_tables' as that
    # plays havoc with the 'old style' table based pages.
    #
    conversion_options = { 'title'       : title,
                           'comments'    : description,
                           'tags'        : tags,
                           'language'    : language,
                           'publisher'   : publisher,
                           'authors'     : publisher,
                           'smarten_punctuation' : True
                         }
    '''

    conversion_options = { 'smarten_punctuation' : True }

    # Specify extra CSS - overrides ALL other CSS (IE. Added last).
    extra_css = 'body { font-family: verdana, helvetica, sans-serif; } \
                 .introduction, .first { font-weight: bold; } \
                 .cross-head { font-weight: bold; font-size: 125%; } \
                 .cap, .caption { display: block; font-size: 80%; font-style: italic; } \
                 .cap, .caption, .caption img, .caption span { display: block; text-align: center; ma
```

```
                .byl, .byd, .byline img, .byline-name, .byline-title, .author-name, .author-positio
                   .correspondent-portrait img, .byline-lead-in, .name, .bbc-role { display: block;
                   text-align: center; font-size: 80%; font-style: italic; margin: 1px auto; } \
                .story-date, .published { font-size: 80%; } \
                table { width: 100%; } \
                td img { display: block; margin: 5px auto; } \
                ul { padding-top: 10px; } \
                ol { padding-top: 10px; } \
                li { padding-top: 5px; padding-bottom: 5px; } \
                h1 { text-align: center; font-size: 175%; font-weight: bold; } \
                h2 { text-align: center; font-size: 150%; font-weight: bold; } \
                h3 { text-align: center; font-size: 125%; font-weight: bold; } \
                h4, h5, h6 { text-align: center; font-size: 100%; font-weight: bold; }'

    # Remove various tag attributes to improve the look of the ebook pages.
    remove_attributes = [ 'border', 'cellspacing', 'align', 'cellpadding', 'colspan',
                          'valign', 'vspace', 'hspace', 'alt', 'width', 'height' ]

    # Remove the (admittedly rarely used) line breaks, "<br />", which sometimes
    # cause a section of the ebook to start in an unsightly fashion or, more
    # frequently, a "<br />" will muck up the formatting of a correspondant's byline.
    # "<br />" and "<br clear/>" are far more frequently used on the table formatted
    # style of pages, and really spoil the look of the ebook pages.
    preprocess_regexps     = [(re.compile(r'<br[ ]*/>', re.IGNORECASE), lambda m: ''),
                              (re.compile(r'<br[ ]*clear.*/>', re.IGNORECASE), lambda m: '')]


    # Create regular expressions for tag keeping and removal to make the matches more
    # robust against minor changes and errors in the HTML, Eg. double spaces, leading
    # and trailing spaces, missing hyphens, and such like.
    # Python regular expression ('re' class) page: http://docs.python.org/library/re.html

    # ***************************************
    # Regular expressions for keep_only_tags:
    # ***************************************

    # The BBC News HTML pages use variants of 'storybody' to denote the section of a HTML
    # page which contains the main text of the article. Match storybody variants: 'storybody',
    # 'story-body', 'story body','storybody ', etc.
    storybody_reg_exp = '^.*story[_ -]*body.*$'

    # The BBC sport and 'newsbeat' (features) HTML pages use 'blq_content' to hold the title
    # and published date. This is one level above the usual news pages which have the title
    # and date within 'story-body'. This is annoying since 'blq_content' must also be kept,
    # resulting in a lot of extra things to be removed by remove_tags.
    blq_content_reg_exp = '^.*blq[_ -]*content.*$'

    # The BBC has an alternative page design structure, which I suspect is an out-of-date
    # design but which is still used in some articles, Eg. 'Click' (technology), 'FastTrack'
    # (travel), and in some sport pages. These alternative pages are table based (which is
    # why I think they are an out-of-date design) and account for -I'm guesstimaking- less
    # than 1% of all articles. They use a table class 'storycontent' to hold the article
    # and like blq_content (above) have required lots of extra removal by remove_tags.
    story_content_reg_exp = '^.*story[_ -]*content.*$'

    # Keep the sections of the HTML which match the list below. The HTML page created by
    # Calibre will fill <body> with those sections which are matched. Note that the
    # blq_content_reg_exp must be listed before storybody_reg_exp in keep_only_tags due to
```

```
    # it being the parent of storybody_reg_exp, that is to say the div class/id 'story-body'
    # will be inside div class/id 'blq_content' in the HTML (if 'blq_content' is there at
    # all). If they are the other way around in keep_only_tags then blq_content_reg_exp
    # will end up being discarded.
    keep_only_tags = [ dict(name='table', attrs={'class':re.compile(story_content_reg_exp, re.IGNOREC
                       dict(name='div',   attrs={'class':re.compile(blq_content_reg_exp, re.IGNORECAS
                       dict(name='div',   attrs={'id':re.compile(blq_content_reg_exp, re.IGNORECASE)
                       dict(name='div',   attrs={'class':re.compile(storybody_reg_exp, re.IGNORECASE)
                       dict(name='div',   attrs={'id':re.compile(storybody_reg_exp, re.IGNORECASE)})

    # ************************************
    # Regular expressions for remove_tags:
    # ************************************

    # Regular expression to remove share-help and variant tags. The share-help class
    # is used by the site for a variety of 'sharing' type links, Eg. Facebook, delicious,
    # twitter, email. Removed to avoid page clutter.
    share_help_reg_exp = '^.*share[_ -]*help.*$'

    # Regular expression to remove embedded-hyper and variant tags. This class is used to
    # display links to other BBC News articles on the same/similar subject.
    embedded_hyper_reg_exp = '^.*embed*ed[_ -]*hyper.*$'

    # Regular expression to remove hypertabs and variant tags. This class is used to
    # display a tab bar at the top of an article which allows the user to switch to
    # an article (viewed on the same page) providing further info., 'in depth' analysis,
    # an editorial, a correspondant's blog entry, and such like. The ability to handle
    # a tab bar of this nature is currently beyond the scope of this recipe and
    # possibly of Calibre itself (not sure about that - TO DO - check!).
    hypertabs_reg_exp = '^.*hyper[_ -]*tabs.*$'

    # Regular expression to remove story-feature and variant tags. Eg. 'story-feature',
    # 'story-feature related narrow', 'story-feature wide', 'story-feature narrow'.
    # This class is used to add additional info. boxes, or small lists, outside of
    # the main story. TO DO: Work out a way to incorporate these neatly.
    story_feature_reg_exp = '^.*story[_ -]*feature.*$'

    # Regular expression to remove video and variant tags, Eg. 'videoInStoryB',
    # 'videoInStoryC'. This class is used to embed video.
    video_reg_exp = '^.*video.*$'

    # Regular expression to remove audio and variant tags, Eg. 'audioInStoryD'.
    # This class is used to embed audio.
    audio_reg_exp = '^.*audio.*$'

    # Regular expression to remove pictureGallery and variant tags, Eg. 'pictureGallery'.
    # This class is used to embed a photo slideshow. See also 'slideshow' below.
    picture_gallery_reg_exp = '^.*picture.*$'

    # Regular expression to remove slideshow and variant tags, Eg. 'dslideshow-enclosure'.
    # This class is used to embed a slideshow (not necessarily photo) but both
    # 'slideshow' and 'pictureGallery' are used for slideshows.
    slideshow_reg_exp = '^.*slide[_ -]*show.*$'

    # Regular expression to remove social-links and variant tags. This class is used to
    # display links to a BBC bloggers main page, used in various columnist's blogs
    # (Eg. Nick Robinson, Robert Preston).
    social_links_reg_exp = '^.*social[_ -]*links.*$'
```

```
    # Regular expression to remove quote and (multi) variant tags, Eg. 'quote',
    # 'endquote', 'quote-credit', 'quote-credit-title', etc. These are usually
    # removed by 'story-feature' removal (as they are usually within them), but
    # not always. The quotation removed is always (AFAICT) in the article text
    # as well but a 2nd copy is placed in a quote tag to draw attention to it.
    # The quote class tags may or may not appear in div's.
    quote_reg_exp = '^.*quote.*$'

    # Regular expression to remove hidden and variant tags, Eg. 'hidden'.
    # The purpose of these is unclear, they seem to be an internal link to a
    # section within the article, but the text of the link (Eg. 'Continue reading
    # the main story') never seems to be displayed anyway. Removed to avoid clutter.
    # The hidden class tags may or may not appear in div's.
    hidden_reg_exp = '^.*hidden.*$'

    # Regular expression to remove comment and variant tags, Eg. 'comment-introduction'.
    # Used on the site to display text about registered users entering comments.
    comment_reg_exp = '^.*comment.*$'

    # Regular expression to remove form and variant tags, Eg. 'comment-form'.
    # Used on the site to allow registered BBC users to fill in forms, typically
    # for entering comments about an article.
    form_reg_exp = '^.*form.*$'

    # Extra things to remove due to the addition of 'blq_content' in keep_only_tags.

    #<div class="story-actions"> Used on sports pages for 'email' and 'print'.
    story_actions_reg_exp = '^.*story[_ -]*actions.*$'

    #<div class="bookmark-list"> Used on sports pages instead of 'share-help' (for
    # social networking links).
    bookmark_list_reg_exp = '^.*bookmark[_ -]*list.*$'

    #<div id="secondary-content" class="content-group">
    # NOTE: Don't remove class="content-group" that is needed.
    # Used on sports pages to link to 'similar stories'.
    secondary_content_reg_exp = '^.*secondary[_ -]*content.*$'

    #<div id="featured-content" class="content-group">
    # NOTE: Don't remove class="content-group" that is needed.
    # Used on sports pages to link to pages like 'tables', 'fixtures', etc.
    featured_content_reg_exp = '^.*featured[_ -]*content.*$'

    #<div id="navigation">
    # Used on sports pages to link to pages like 'tables', 'fixtures', etc.
    # Used sometimes instead of "featured-content" above.
    navigation_reg_exp = '^.*navigation.*$'

    #<a class="skip" href="#blq-container-inner">Skip to top</a>
    # Used on sports pages to link to the top of the page.
    skip_reg_exp = '^.*skip.*$'

    # Extra things to remove due to the addition of 'storycontent' in keep_only_tags,
    # which are the alterative table design based pages. The purpose of some of these
    # is not entirely clear from the pages (which are a total mess!).

    # Remove mapping based tags, Eg. <map id="world_map">
    # The dynamic maps don't seem to work during ebook creation. TO DO: Investigate.
```

```
    map_reg_exp = '^.*map.*$'

    # Remove social bookmarking variation, called 'socialBookMarks'.
    social_bookmarks_reg_exp = '^.*social[_ -]*bookmarks.*$'

    # Remove page navigation tools, like 'search', 'email', 'print', called 'blq-mast'.
    blq_mast_reg_exp = '^.*blq[_ -]*mast.*$'

    # Remove 'sharesb', I think this is a generic 'sharing' class. It seems to appear
    # alongside 'socialBookMarks' whenever that appears. I am removing it as well
    # under the assumption that it can appear alone as well.
    sharesb_reg_exp = '^.*sharesb.*$'

    # Remove class 'o'. The worst named user created css class of all time. The creator
    # should immediately be fired. I've seen it used to hold nothing at all but with
    # 20 or so empty lines in it. Also to hold a single link to another article.
    # Whatever it was designed to do it is not wanted by this recipe. Exact match only.
    o_reg_exp = '^o$'

    # Remove 'promotopbg' and 'promobottombg', link lists. Have decided to
    # use two reg expressions to make removing this (and variants) robust.
    promo_top_reg_exp = '^.*promotopbg.*$'
    promo_bottom_reg_exp = '^.*promobottombg.*$'

    # Remove 'nlp', provides heading for link lists. Requires an exact match due to
    # risk of matching those letters in something needed, unless I see a variation
    # of 'nlp' used at a later date.
    nlp_reg_exp = '^nlp$'

    # Remove 'mva', provides embedded floating content of various types. Variant 'mvb'
    # has also now been seen. Requires an exact match of 'mva' or 'mvb' due to risk of
    # matching those letters in something needed.
    mva_or_mvb_reg_exp = '^mv[ab]$'

    # Remove 'mvtb', seems to be page navigation tools, like 'blq-mast'.
    mvtb_reg_exp = '^mvtb$'

    # Remove 'blq-toplink', class to provide a link to the top of the page.
    blq_toplink_reg_exp = '^.*blq[_ -]*top[_ -]*link.*$'

    # Remove 'products and services' links, Eg. desktop tools, alerts, and so on.
    # Eg. Class="servicev4 ukfs_services" - what a mess of a name. Have decided to
    # use two reg expressions to make removing this (and variants) robust.
    prods_services_01_reg_exp = '^.*servicev4.*$'
    prods_services_02_reg_exp = '^.*ukfs[_ -]*services.*$'

    # Remove -what I think is- some kind of navigation tools helper class, though I am
    # not sure, it's called: 'blq-rst blq-new-nav'. What I do know is it pops up
    # frequently and it is not wanted. Have decided to use two reg expressions to make
    # removing this (and variants) robust.
    blq_misc_01_reg_exp = '^.*blq[_ -]*rst.*$'
    blq_misc_02_reg_exp = '^.*blq[_ -]*new[_ -]*nav.*$'

    # Remove 'puffbox' - this may only appear inside 'storyextra', so it may not
    # need removing - I have no clue what it does other than it contains links.
    # Whatever it is - it is not part of the article and is not wanted.
    puffbox_reg_exp = '^.*puffbox.*$'
```

```
    # Remove 'sibtbg' and 'sibtbgf' - some kind of table formatting classes.
    sibtbg_reg_exp = '^.*sibtbg.*$'

    # Remove 'storyextra' - links to relevant articles and external sites.
    storyextra_reg_exp = '^.*story[_ -]*extra.*$'


    remove_tags = [ dict(name='div',  attrs={'class':re.compile(story_feature_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'class':re.compile(share_help_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(embedded_hyper_reg_exp, re.IGNORECASE
                    dict(name='div',  attrs={'class':re.compile(hypertabs_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(video_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(audio_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(picture_gallery_reg_exp, re.IGNORECAS
                    dict(name='div',  attrs={'class':re.compile(slideshow_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(quote_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(hidden_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(comment_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(story_actions_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'class':re.compile(bookmark_list_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'id':re.compile(secondary_content_reg_exp, re.IGNORECASE
                    dict(name='div',  attrs={'id':re.compile(featured_content_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'id':re.compile(navigation_reg_exp, re.IGNORECASE)}),
                    dict(name='form', attrs={'id':re.compile(form_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(quote_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(hidden_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(social_links_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(comment_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(skip_reg_exp, re.IGNORECASE)}),
                    dict(name='map', attrs={'id':re.compile(map_reg_exp, re.IGNORECASE)}),
                    dict(name='map', attrs={'name':re.compile(map_reg_exp, re.IGNORECASE)}),
                    dict(name='div', attrs={'id':re.compile(social_bookmarks_reg_exp, re.IGNORECASE)
                    dict(name='div', attrs={'id':re.compile(blq_mast_reg_exp, re.IGNORECASE)}),
                    dict(name='div', attrs={'class':re.compile(sharesb_reg_exp, re.IGNORECASE)}),
                    dict(name='div', attrs={'class':re.compile(o_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(promo_top_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(promo_bottom_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'class':re.compile(nlp_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(mva_or_mvb_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(mvtb_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(blq_toplink_reg_exp, re.IGNORECASE)}
                    dict(name='div',  attrs={'class':re.compile(prods_services_01_reg_exp, re.IGNOREC
                    dict(name='div',  attrs={'class':re.compile(prods_services_02_reg_exp, re.IGNOREC
                    dict(name='div',  attrs={'class':re.compile(blq_misc_01_reg_exp, re.IGNORECASE)}
                    dict(name='div',  attrs={'class':re.compile(blq_misc_02_reg_exp, re.IGNORECASE)}
                    dict(name='div',  attrs={'class':re.compile(puffbox_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(sibtbg_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(storyextra_reg_exp, re.IGNORECASE)})
                  ]

    # Uses url to create and return the 'printer friendly' version of the url.
    # In other words the 'print this page' address of the page.
    #
    # There are 3 types of urls used in the BBC site's rss feeds. There is just
    # 1 type for the standard news while there are 2 used for sports feed urls.
    # Note: Sports urls are linked from regular news feeds (Eg. 'News Home') when
    # there is a major story of interest to 'everyone'. So even if no BBC sports
    # feeds are added to 'feeds' the logic of this method is still needed to avoid
```

```
    # blank / missing / empty articles which have an index title and then no body.
    def print_version(self, url):

        # Handle sports page urls type 01:
        if (url.find("go/rss/-/sport1/") != -1):
            temp_url = url.replace("go/rss/-/", "")

        # Handle sports page urls type 02:
        elif (url.find("go/rss/int/news/-/sport1/") != -1):
            temp_url = url.replace("go/rss/int/news/-/", "")

        # Handle regular news page urls:
        else:
            temp_url = url.replace("go/rss/int/news/-/", "")

        # Always add "?print=true" to the end of the url.
        print_url = temp_url + "?print=true"

        return print_url


    # Remove articles in feeds based on a string in the article title or url.
    #
    # Code logic written by: Starson17 - posted in: "Recipes - Re-usable code"
    # thread, in post with title: "Remove articles from feed", see url:
    # http://www.mobileread.com/forums/showpost.php?p=1165462&postcount=6
    # Many thanks and all credit to Starson17.
    #
    # Starson17's code has obviously been altered to suite my requirements.
    def parse_feeds(self):

        # Call parent's method.
        feeds = BasicNewsRecipe.parse_feeds(self)

        # Loop through all feeds.
        for feed in feeds:

            # Loop through all articles in feed.
            for article in feed.articles[:]:

                # Match key words and remove article if there's a match.

                # Most BBC rss feed video only 'articles' use upper case 'VIDEO'
                # as a title prefix. Just match upper case 'VIDEO', so that
                # articles like 'Video game banned' won't be matched and removed.
                if 'VIDEO' in article.title:
                    feed.articles.remove(article)

                # Most BBC rss feed audio only 'articles' use upper case 'AUDIO'
                # as a title prefix. Just match upper case 'AUDIO', so that
                # articles like 'Hi-Def audio...' won't be matched and removed.
                elif 'AUDIO' in article.title:
                    feed.articles.remove(article)

                # Most BBC rss feed photo slideshow 'articles' use 'In Pictures',
                # 'In pictures', and 'in pictures', somewhere in their title.
                # Match any case of that phrase.
                elif 'IN PICTURES' in article.title.upper():
```

```
                    feed.articles.remove(article)

                # As above, but user contributed pictures. Match any case.
                elif 'YOUR PICTURES' in article.title.upper():
                    feed.articles.remove(article)

                # 'Sportsday Live' are articles which contain a constantly and
                # dynamically updated 'running commentary' during a live sporting
                # event. Match any case.
                elif 'SPORTSDAY LIVE' in article.title.upper():
                    feed.articles.remove(article)

                # Sometimes 'Sportsday Live' (above) becomes 'Live - Sport Name'.
                # These are being matched below using 'Live - ' because removing all
                # articles with 'live' in their titles would remove some articles
                # that are in fact not live sports pages. Match any case.
                elif 'LIVE - ' in article.title.upper():
                    feed.articles.remove(article)

                # 'Quiz of the week' is a Flash player weekly news quiz. Match only
                # the 'Quiz of the' part in anticipation of monthly and yearly
                # variants. Match any case.
                elif 'QUIZ OF THE' in article.title.upper():
                    feed.articles.remove(article)

                # Remove articles with 'scorecards' in the url. These are BBC sports
                # pages which just display a cricket scorecard. The pages have a mass
                # of table and css entries to display the scorecards nicely. Probably
                # could make them work with this recipe, but might take a whole day
                # of work to sort out all the css - basically a formatting nightmare.
                elif 'scorecards' in article.url:
                    feed.articles.remove(article)

        return feeds

# End of class and file.
```

This *recipe* explores only the tip of the iceberg when it comes to the power of calibre. To explore more of the abilities of calibre we'll examine a more complex real life example in the next section.

Real life example

A reasonably complex real life example that exposes more of the *API* of `BasicNewsRecipe` is the *recipe* for *The New York Times*

```
import string, re
from calibre import strftime
from calibre.web.feeds.recipes import BasicNewsRecipe
from calibre.ebooks.BeautifulSoup import BeautifulSoup

class NYTimes(BasicNewsRecipe):

    title       = 'The New York Times'
    __author__  = 'Kovid Goyal'
    description = 'Daily news from the New York Times'
    timefmt = ' [%a, %d %b, %Y]'
    needs_subscription = True
```

```
    remove_tags_before = dict(id='article')
    remove_tags_after  = dict(id='article')
    remove_tags = [dict(attrs={'class':['articleTools', 'post-tools', 'side_tool', 'nextArticleLink c
                dict(id=['footer', 'toolsRight', 'articleInline', 'navigation', 'archive', 'side_sear
                dict(name=['script', 'noscript', 'style'])]
    encoding = 'cp1252'
    no_stylesheets = True
    extra_css = 'h1 {font: sans-serif large;}\n.byline {font:monospace;}'

    def get_browser(self):
        br = BasicNewsRecipe.get_browser()
        if self.username is not None and self.password is not None:
            br.open('http://www.nytimes.com/auth/login')
            br.select_form(name='login')
            br['USERID']   = self.username
            br['PASSWORD'] = self.password
            br.submit()
        return br

    def parse_index(self):
        soup = self.index_to_soup('http://www.nytimes.com/pages/todayspaper/index.html')

        def feed_title(div):
            return ''.join(div.findAll(text=True, recursive=False)).strip()

        articles = {}
        key = None
        ans = []
        for div in soup.findAll(True,
             attrs={'class':['section-headline', 'story', 'story headline']}):

             if div['class'] == 'section-headline':
                 key = string.capwords(feed_title(div))
                 articles[key] = []
                 ans.append(key)

             elif div['class'] in ['story', 'story headline']:
                 a = div.find('a', href=True)
                 if not a:
                     continue
                 url = re.sub(r'\?.*', '', a['href'])
                 url += '?pagewanted=all'
                 title = self.tag_to_string(a, use_alt=True).strip()
                 description = ''
                 pubdate = strftime('%a, %d %b')
                 summary = div.find(True, attrs={'class':'summary'})
                 if summary:
                     description = self.tag_to_string(summary, use_alt=False)

                 feed = key if key is not None else 'Uncategorized'
                 if not articles.has_key(feed):
                     articles[feed] = []
                 if not 'podcasts' in url:
                     articles[feed].append(
                               dict(title=title, url=url, date=pubdate,
                                    description=description,
                                    content=''))
        ans = self.sort_index_by(ans, {'The Front Page':-1, 'Dining In, Dining Out':1, 'Obituaries':2
```

```
        ans = [(key, articles[key]) for key in ans if articles.has_key(key)]
        return ans

    def preprocess_html(self, soup):
        refresh = soup.find('meta', {'http-equiv':'refresh'})
        if refresh is None:
            return soup
        content = refresh.get('content').partition('=')[2]
        raw = self.browser.open('http://www.nytimes.com'+content).read()
        return BeautifulSoup(raw.decode('cp1252', 'replace'))
```

We see several new features in this *recipe*. First, we have:

```
timefmt = ' [%a, %d %b, %Y]'
```

This sets the displayed time on the front page of the created ebook to be in the format, `Day, Day_Number Month, Year`. See `timefmt` (page 368).

Then we see a group of directives to cleanup the downloaded *HTML*:

```
remove_tags_before = dict(name='h1')
remove_tags_after  = dict(id='footer')
remove_tags = ...
```

These remove everything before the first `<h1>` tag and everything after the first tag whose id is `footer`. See `remove_tags` (page 367), `remove_tags_before` (page 367), `remove_tags_after` (page 367).

The next interesting feature is:

```
needs_subscription = True
...
def get_browser(self):
    ...
```

`needs_subscription = True` tells calibre that this recipe needs a username and password in order to access the content. This causes, calibre to ask for a username and password whenever you try to use this recipe. The code in `calibre.web.feeds.news.BasicNewsRecipe.get_browser()` (page 361) actually does the login into the NYT website. Once logged in, calibre will use the same, logged in, browser instance to fetch all content. See mechanize[112] to understand the code in `get_browser`.

The next new feature is the `calibre.web.feeds.news.BasicNewsRecipe.parse_index()` (page 362) method. Its job is to go to http://www.nytimes.com/pages/todayspaper/index.html and fetch the list of articles that appear in *todays* paper. While more complex than simply using *RSS*, the recipe creates an ebook that corresponds very closely to the days paper. `parse_index` makes heavy use of BeautifulSoup[113] to parse the daily paper webpage.

The final new feature is the `calibre.web.feeds.news.BasicNewsRecipe.preprocess_html()` (page 363) method. It can be used to perform arbitrary transformations on every downloaded HTML page. Here it is used to bypass the ads that the nytimes shows you before each article.

Tips for developing new recipes

The best way to develop new recipes is to use the command line interface. Create the recipe using your favorite python editor and save it to a file say `myrecipe.recipe`. The *.recipe* extension is required. You can download content using this recipe with the command:

[112] http://wwwsearch.sourceforge.net/mechanize/
[113] http://www.crummy.com/software/BeautifulSoup/documentation.html

```
ebook-convert myrecipe.recipe .epub --test -vv --debug-pipeline debug
```

The command **ebook-convert** will download all the webpages and save them to the EPUB file `myrecipe.epub`. The *-vv* makes ebook-convert spit out a lot of information about what it is doing. The *--test* makes it download only a couple of articles from at most two feeds. In addition, ebook-convert will put the downloaded HTML into the `debug/input` directory, where `debug` is the directory you specified in the *--debug-pipeline* option.

Once the download is complete, you can look at the downloaded *HTML* by opening the file `debug/input/index.html` in a browser. Once you're satisfied that the download and preprocessing is happening correctly, you can generate ebooks in different formats as shown below:

```
ebook-convert myrecipe.recipe myrecipe.epub
ebook-convert myrecipe.recipe myrecipe.mobi
...
```

If you're satisfied with your recipe, and you feel there is enough demand to justify its inclusion into the set of built-in recipes, post your recipe in the calibre recipes forum[114] to share it with other calibre users.

Note: On OS X, the ebook-convert command will not be available by default. Go to Preferences->Miscellaneous and click the install command line tools button to make it available.

See Also:

ebook-convert **(page 486)** The command line interface for all ebook conversion.

Further reading

To learn more about writing advanced recipes using some of the facilities, available in `BasicNewsRecipe` you should consult the following sources:

API Documentation **(page 360)** Documentation of the `BasicNewsRecipe` class and all its important methods and fields.

BasicNewsRecipe[115] The source code of `BasicNewsRecipe`

Built-in recipes[116] The source code for the built-in recipes that come with calibre

The calibre recipes forum[117] Lots of knowledgeable calibre recipe writers hang out here.

API documentation

API Documentation for recipes

The API for writing recipes is defined by the `BasicNewsRecipe` (page 360)

class `calibre.web.feeds.news.`**BasicNewsRecipe**(*options*, *log*, *progress_reporter*)

Base class that contains logic needed in all recipes. By overriding progressively more of the functionality in this class, you can make progressively more customized/powerful recipes. For a tutorial introduction to creating recipes, see *Adding your favorite news website* (page 341).

abort_recipe_processing(*msg*)

Causes the recipe download system to abort the download of this recipe, displaying a simple feedback message to the user.

[114] http://www.mobileread.com/forums/forumdisplay.php?f=228

add_toc_thumbnail (*article*, *src*)
Call this from populate_article_metadata with the src attribute of an <img> tag from the article that is appropriate for use as the thumbnail representing the article in the Table of Contents. Whether the thumbnail is actually used is device dependent (currently only used by the Kindles). Note that the referenced image must be one that was successfully downloaded, otherwise it will be ignored.

classmethod adeify_images (*soup*)
If your recipe when converted to EPUB has problems with images when viewed in Adobe Digital Editions, call this method from within `postprocess_html()` (page 363).

cleanup ()
Called after all articles have been download. Use it to do any cleanup like logging out of subscription sites, etc.

clone_browser (*br*)
Clone the browser br. Cloned browsers are used for multi-threaded downloads, since mechanize is not thread safe. The default cloning routines should capture most browser customization, but if you do something exotic in your recipe, you should override this method in your recipe and clone manually.

Cloned browser instances use the same, thread-safe CookieJar by default, unless you have customized cookie handling.

default_cover (*cover_file*)
Create a generic cover for recipes that dont have a cover

download ()
Download and pre-process all articles from the feeds in this recipe. This method should be called only once on a particular Recipe instance. Calling it more than once will lead to undefined behavior. :return: Path to index.html

extract_readable_article (*html*, *url*)
Extracts main article content from 'html', cleans up and returns as a (article_html, extracted_title) tuple. Based on the original readability algorithm by Arc90.

get_article_url (*article*)
Override in a subclass to customize extraction of the *URL* that points to the content for each article. Return the article URL. It is called with *article*, an object representing a parsed article from a feed. See feedparser[118]. By default it looks for the original link (for feeds syndicated via a service like feedburner or pheedo) and if found, returns that or else returns article.link[119].

get_browser (**args*, ***kwargs*)
Return a browser instance used to fetch documents from the web. By default it returns a mechanize[120] browser instance that supports cookies, ignores robots.txt, handles refreshes and has a mozilla firefox user agent.

If your recipe requires that you login first, override this method in your subclass. For example, the following code is used in the New York Times recipe to login for full access:

```
def get_browser(self):
    br = BasicNewsRecipe.get_browser(self)
    if self.username is not None and self.password is not None:
        br.open('http://www.nytimes.com/auth/login')
        br.select_form(name='login')
        br['USERID']   = self.username
        br['PASSWORD'] = self.password
        br.submit()
    return br
```

[118] http://packages.python.org/feedparser/
[119] http://packages.python.org/feedparser/reference-entry-link.html
[120] http://wwwsearch.sourceforge.net/mechanize/

get_cover_url()
Return a *URL* to the cover image for this issue or *None*. By default it returns the value of the member *self.cover_url* which is normally *None*. If you want your recipe to download a cover for the e-book override this method in your subclass, or set the member variable *self.cover_url* before this method is called.

get_feeds()
Return a list of *RSS* feeds to fetch for this profile. Each element of the list must be a 2-element tuple of the form (title, url). If title is None or an empty string, the title from the feed is used. This method is useful if your recipe needs to do some processing to figure out the list of feeds to download. If so, override in your subclass.

get_masthead_title()
Override in subclass to use something other than the recipe title

get_masthead_url()
Return a *URL* to the masthead image for this issue or *None*. By default it returns the value of the member *self.masthead_url* which is normally *None*. If you want your recipe to download a masthead for the e-book override this method in your subclass, or set the member variable *self.masthead_url* before this method is called. Masthead images are used in Kindle MOBI files.

get_obfuscated_article(*url*)
If you set *articles_are_obfuscated* this method is called with every article URL. It should return the path to a file on the filesystem that contains the article HTML. That file is processed by the recursive HTML fetching engine, so it can contain links to pages/images on the web.

This method is typically useful for sites that try to make it difficult to access article content automatically.

classmethod image_url_processor(*baseurl*, *url*)
Perform some processing on image urls (perhaps removing size restrictions for dynamically generated images, etc.) and return the precessed URL.

index_to_soup(*url_or_raw*, *raw=False*)
Convenience method that takes an URL to the index page and returns a BeautifulSoup[121] of it.

url_or_raw: Either a URL or the downloaded index page as a string

is_link_wanted(*url*, *tag*)
Return True if the link should be followed or False otherwise. By default, raises NotImplementedError which causes the downloader to ignore it.

Parameters

- **url** – The URL to be followed
- **tag** – The Tag from which the URL was derived

javascript_login(*browser*, *username*, *password*)
This method is used to login to a website that uses javascript for its login form. After the login is complete, the cookies returned from the website are copied to a normal (non-javascript) browser and the download proceeds using those cookies.

An example implementation:

```
def javascript_login(self, browser, username, password):
    browser.visit('http://some-page-that-has-a-login')
    form = browser.select_form(nr=0) # Select the first form on the page
    form['username'] = username
    form['password'] = password
    browser.submit(timeout=120) # Submit the form and wait at most two minutes for loading t
```

[121] http://www.crummy.com/software/BeautifulSoup/documentation.html

Note that you can also select forms with CSS2 selectors, like this:

```
browser.select_form('form#login_form')
browser.select_from('form[name="someform"]')
```

parse_feeds()
Create a list of articles from the list of feeds returned by BasicNewsRecipe.get_feeds() (page 361). Return a list of Feed objects.

parse_index()
This method should be implemented in recipes that parse a website instead of feeds to generate a list of articles. Typical uses are for news sources that have a "Print Edition" webpage that lists all the articles in the current print edition. If this function is implemented, it will be used in preference to BasicNewsRecipe.parse_feeds() (page 362).

It must return a list. Each element of the list must be a 2-element tuple of the form ('feed title', list of articles).

Each list of articles must contain dictionaries of the form:

```
{
'title'       : article title,
'url'         : URL of print version,
'date'        : The publication date of the article as a string,
'description' : A summary of the article
'content'     : The full article (can be an empty string). Obsolete
                do not use, instead save the content to a temporary
                file and pass a file:///path/to/temp/file.html as
                the URL.
}
```

For an example, see the recipe for downloading *The Atlantic*. In addition, you can add 'author' for the author of the article.

If you want to abort processing for some reason and have calibre show the user a simple message instead of an error, call abort_recipe_processing() (page 360).

populate_article_metadata(*article*, *soup*, *first*)
Called when each HTML page belonging to article is downloaded. Intended to be used to get article metadata like author/summary/etc. from the parsed HTML (soup). :param article: A object of class calibre.web.feeds.Article. If you change the summary, remember to also change the text_summary :param soup: Parsed HTML belonging to this article :param first: True iff the parsed HTML is the first page of the article.

postprocess_book(*oeb*, *opts*, *log*)
Run any needed post processing on the parsed downloaded e-book.

Parameters

- **oeb** – An OEBBook object
- **opts** – Conversion options

postprocess_html(*soup*, *first_fetch*)
This method is called with the source of each downloaded *HTML* file, after it is parsed for links and images. It can be used to do arbitrarily powerful post-processing on the *HTML*. It should return *soup* after processing it.

Parameters

- **soup** – A BeautifulSoup[122] instance containing the downloaded *HTML*.

[122] http://www.crummy.com/software/BeautifulSoup/documentation.html

- **first_fetch** – True if this is the first page of an article.

preprocess_html(*soup*)
This method is called with the source of each downloaded *HTML* file, before it is parsed for links and images. It is called after the cleanup as specified by remove_tags etc. It can be used to do arbitrarily powerful pre-processing on the *HTML*. It should return *soup* after processing it.

soup: A BeautifulSoup[123] instance containing the downloaded *HTML*.

preprocess_raw_html(*raw_html*, *url*)
This method is called with the source of each downloaded *HTML* file, before it is parsed into an object tree. raw_html is a unicode string representing the raw HTML downloaded from the web. url is the URL from which the HTML was downloaded.

Note that this method acts *before* preprocess_regexps.

This method must return the processed raw_html as a unicode object.

classmethod print_version(*url*)
Take a *url* pointing to the webpage with article content and return the *URL* pointing to the print version of the article. By default does nothing. For example:

```
def print_version(self, url):
    return url + '?&pagewanted=print'
```

skip_ad_pages(*soup*)
This method is called with the source of each downloaded *HTML* file, before any of the cleanup attributes like remove_tags, keep_only_tags are applied. Note that preprocess_regexps will have already been applied. It is meant to allow the recipe to skip ad pages. If the soup represents an ad page, return the HTML of the real page. Otherwise return None.

soup: A BeautifulSoup[124] instance containing the downloaded *HTML*.

sort_index_by(*index*, *weights*)
Convenience method to sort the titles in *index* according to *weights*. *index* is sorted in place. Returns *index*.

index: A list of titles.

weights: A dictionary that maps weights to titles. If any titles in index are not in weights, they are assumed to have a weight of 0.

classmethod tag_to_string(*tag*, *use_alt=True*, *normalize_whitespace=True*)
Convenience method to take a BeautifulSoup[125] *Tag* and extract the text from it recursively, including any CDATA sections and alt tag attributes. Return a possibly empty unicode string.

use_alt: If *True* try to use the alt attribute for tags that don't have any textual content

tag: BeautifulSoup[126] *Tag*

articles_are_obfuscated = False
Set to True and implement `get_obfuscated_article()` (page 362) to handle websites that try to make it difficult to scrape content.

auto_cleanup = False
Automatically extract all the text from downloaded article pages. Uses the algorithms from the readability project. Setting this to True, means that you do not have to worry about cleaning up the downloaded HTML manually (though manual cleanup will always be superior).

[123] http://www.crummy.com/software/BeautifulSoup/documentation.html
[124] http://www.crummy.com/software/BeautifulSoup/documentation.html
[125] http://www.crummy.com/software/BeautifulSoup/documentation.html
[126] http://www.crummy.com/software/BeautifulSoup/documentation.html

auto_cleanup_keep = None

Specify elements that the auto cleanup algorithm should never remove The syntax is a XPath expression. For example:

```
auto_cleanup_keep = '//div[@id="article-image"]' will keep all divs with
                                                 id="article-image"
auto_cleanup_keep = '//*[@class="important"]' will keep all elements
                                               with class="important"
auto_cleanup_keep = '//div[@id="article-image"]|//span[@class="important"]'
                  will keep all divs with id="article-image" and spans
                  with class="important"
```

center_navbar = True

If True the navigation bar is center aligned, otherwise it is left aligned

conversion_options = {}

Recipe specific options to control the conversion of the downloaded content into an e-book. These will override any user or plugin specified values, so only use if absolutely necessary. For example:

```
conversion_options = {
  'base_font_size'   : 16,
  'tags'             : 'mytag1,mytag2',
  'title'            : 'My Title',
  'linearize_tables' : True,
}
```

cover_margins = (0, 0, '#ffffff')

By default, the cover image returned by get_cover_url() will be used as the cover for the periodical. Overriding this in your recipe instructs calibre to render the downloaded cover into a frame whose width and height are expressed as a percentage of the downloaded cover. cover_margins = (10, 15, '#ffffff') pads the cover with a white margin 10px on the left and right, 15px on the top and bottom. Color names defined at http://www.imagemagick.org/script/color.php Note that for some reason, white does not always work on windows. Use #ffffff instead

delay = 0

Delay between consecutive downloads in seconds. The argument may be a floating point number to indicate a more precise time.

description = u''

A couple of lines that describe the content this recipe downloads. This will be used primarily in a GUI that presents a list of recipes.

encoding = None

Specify an override encoding for sites that have an incorrect charset specification. The most common being specifying `latin1` and using `cp1252`. If None, try to detect the encoding. If it is a callable, the callable is called with two arguments: The recipe object and the source to be decoded. It must return the decoded source.

extra_css = None

Specify any extra *CSS* that should be added to downloaded *HTML* files It will be inserted into *<style>* tags, just before the closing *</head>* tag thereby overriding all *CSS* except that which is declared using the style attribute on individual *HTML* tags. For example:

```
extra_css = '.heading { font: serif x-large }'
```

feeds = None

List of feeds to download Can be either `[url1, url2, ...]` or `[('title1', url1), ('title2', url2),...]`

filter_regexps = []
List of regular expressions that determines which links to ignore If empty it is ignored. Used only if is_link_wanted is not implemented. For example:

```
filter_regexps = [r'ads\.doubleclick\.net']
```

will remove all URLs that have *ads.doubleclick.net* in them.

Only one of BasicNewsRecipe.match_regexps (page 366) or BasicNewsRecipe.filter_regexps (page 365) should be defined.

ignore_duplicate_articles = None
Ignore duplicates of articles that are present in more than one section. A duplicate article is an article that has the same title and/or URL. To ignore articles with the same title, set this to: ignore_duplicate_articles = {'title'} To use URLs instead, set it to: ignore_duplicate_articles = {'url'} To match on title or URL, set it to: ignore_duplicate_articles = {'title', 'url'}

keep_only_tags = []
Keep only the specified tags and their children. For the format for specifying a tag see BasicNewsRecipe.remove_tags (page 367). If this list is not empty, then the <*body*> tag will be emptied and re-filled with the tags that match the entries in this list. For example:

```
keep_only_tags = [dict(id=['content', 'heading'])]
```

will keep only tags that have an *id* attribute of *"content"* or *"heading"*.

language = 'und'
The language that the news is in. Must be an ISO-639 code either two or three characters long

masthead_url = None
By default, calibre will use a default image for the masthead (Kindle only). Override this in your recipe to provide a url to use as a masthead.

match_regexps = []
List of regular expressions that determines which links to follow If empty, it is ignored. Used only if is_link_wanted is not implemented. For example:

```
match_regexps = [r'page=[0-9]+']
```

will match all URLs that have *page=some number* in them.

Only one of BasicNewsRecipe.match_regexps (page 366) or BasicNewsRecipe.filter_regexps (page 365) should be defined.

max_articles_per_feed = 100
Maximum number of articles to download from each feed. This is primarily useful for feeds that don't have article dates. For most feeds, you should use BasicNewsRecipe.oldest_article (page 366)

needs_subscription = False
If True the GUI will ask the user for a username and password to use while downloading If set to "optional" the use of a username and password becomes optional

no_stylesheets = False
Convenient flag to disable loading of stylesheets for websites that have overly complex stylesheets unsuitable for conversion to ebooks formats If True stylesheets are not downloaded and processed

oldest_article = 7.0
Oldest article to download from this news source. In days.

preprocess_regexps = []
List of *regexp* substitution rules to run on the downloaded *HTML*. Each element of the list should be a two

element tuple. The first element of the tuple should be a compiled regular expression and the second a callable that takes a single match object and returns a string to replace the match. For example:

```
preprocess_regexps = [
    (re.compile(r'<!--Article ends here-->.*</body>', re.DOTALL|re.IGNORECASE),
     lambda match: '</body>'),
]
```

will remove everythong from *<!–Article ends here–>* to *</body>*.

publication_type = 'unknown'
Publication type Set to newspaper, magazine or blog. If set to None, no publication type metadata will be written to the opf file.

recipe_disabled = None
Set to a non empty string to disable this recipe The string will be used as the disabled message

recursions = 0
Number of levels of links to follow on article webpages

remove_attributes = []
List of attributes to remove from all tags For example:

```
remove_attributes = ['style', 'font']
```

remove_empty_feeds = False
If True empty feeds are removed from the output. This option has no effect if parse_index is overridden in the sub class. It is meant only for recipes that return a list of feeds using *feeds* or `get_feeds()` (page 361). It is also used if you use the ignore_duplicate_articles option.

remove_javascript = True
Convenient flag to strip all javascript tags from the downloaded HTML

remove_tags = []
List of tags to be removed. Specified tags are removed from downloaded HTML. A tag is specified as a dictionary of the form:

```
{
 name       : 'tag name',   #e.g. 'div'
 attrs      : a dictionary, #e.g. {class: 'advertisment'}
}
```

All keys are optional. For a full explanantion of the search criteria, see Beautiful Soup[127] A common example:

```
remove_tags = [dict(name='div', attrs={'class':'advert'})]
```

This will remove all * * tags and all their children from the downloaded *HTML*.

remove_tags_after = None
Remove all tags that occur after the specified tag. For the format for specifying a tag see `BasicNewsRecipe.remove_tags` (page 367). For example:

```
remove_tags_after = [dict(id='content')]
```

will remove all tags after the first element with *id="content"*.

remove_tags_before = None
Remove all tags that occur before the specified tag. For the format for specifying a tag see `BasicNewsRecipe.remove_tags` (page 367). For example:

[127] http://www.crummy.com/software/BeautifulSoup/documentation.html#Thebasicfindmethod:findAll(name,attrs,recursive,text,limit,**kwargs)

```
remove_tags_before = dict(id='content')
```

will remove all tags before the first element with *id="content"*.

requires_version = (0, 6, 0)
Minimum calibre version needed to use this recipe

reverse_article_order = False
Reverse the order of articles in each feed

simultaneous_downloads = 5
Number of simultaneous downloads. Set to 1 if the server is picky. Automatically reduced to 1 if `BasicNewsRecipe.delay` (page 365) > 0

summary_length = 500
Max number of characters in the short description

template_css = u'\n .article_date {\n color: gray; font-family: monospace;\n }\n\n .article_description {\n text-indent:
The CSS that is used to style the templates, i.e., the navigation bars and the Tables of Contents. Rather than overriding this variable, you should use *extra_css* in your recipe to customize look and feel.

timefmt = ' [%a, %d %b %Y]'
The format string for the date shown on the first page. By default: Day_Name, Day_Number Month_Name Year

timeout = 120.0
Timeout for fetching files from server in seconds

title = u'Unknown News Source'
The title to use for the ebook

use_embedded_content = None
Normally we try to guess if a feed has full articles embedded in it based on the length of the embedded content. If *None*, then the default guessing is used. If *True* then the we always assume the feeds has embedded content and if *False* we always assume the feed does not have embedded content.

use_javascript_to_login = False
If you set this True, then calibre will use javascript to login to the website. This is needed for some websites that require the use of javascript to login. If you set this to True you must implement the `javascript_login()` (page 362) method, to do the actual logging in.

1.14 The calibre ebook viewer

1.14.1 The Ebook Viewer

calibre includes a built-in ebook viewer that can view all the major ebook formats. The viewer is highly customizable and has many advanced features.

- Starting the viewer (page 303)
- Navigating around an ebook (page 303)
- Customizing the look and feel of your reading experience (page 304)
- Dictionary lookup (page 305)
- Copying text and images (page 305)

Starting the viewer

You can view any of the books in your calibre library by selecting the book and pressing the View button. This will open up the book in the ebook viewer. You can also launch the viewer by itself from the Start menu in Windows or using the command **ebook-viewer** in Linux and OS X (you have to install the command line tools on OS X first by going to *Preferences->Advanced->Miscellaneous*).

Navigating around an ebook

You can "turn pages" in a book by using the *Page Next* and *Page Previous* buttons , or by pressing the Page Down/Page Up keys. Unlike most ebook viewers, calibre does not force you to view books in paged mode. You can scroll by amounts less than a page by using the scroll bar or various customizable keyboard shortcuts.

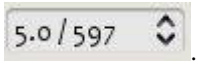
Bookmarks

When you are in the middle of a book and close the viewer, it will remember where you stopped reading and return there the next time you open the book. You can also set bookmarks in the book by using the Bookmark button . When viewing EPUB format books, these bookmarks are actually saved in the EPUB file itself. You can add bookmarks, then send the file to a friend. When they open the file, they will be able to see your bookmarks.

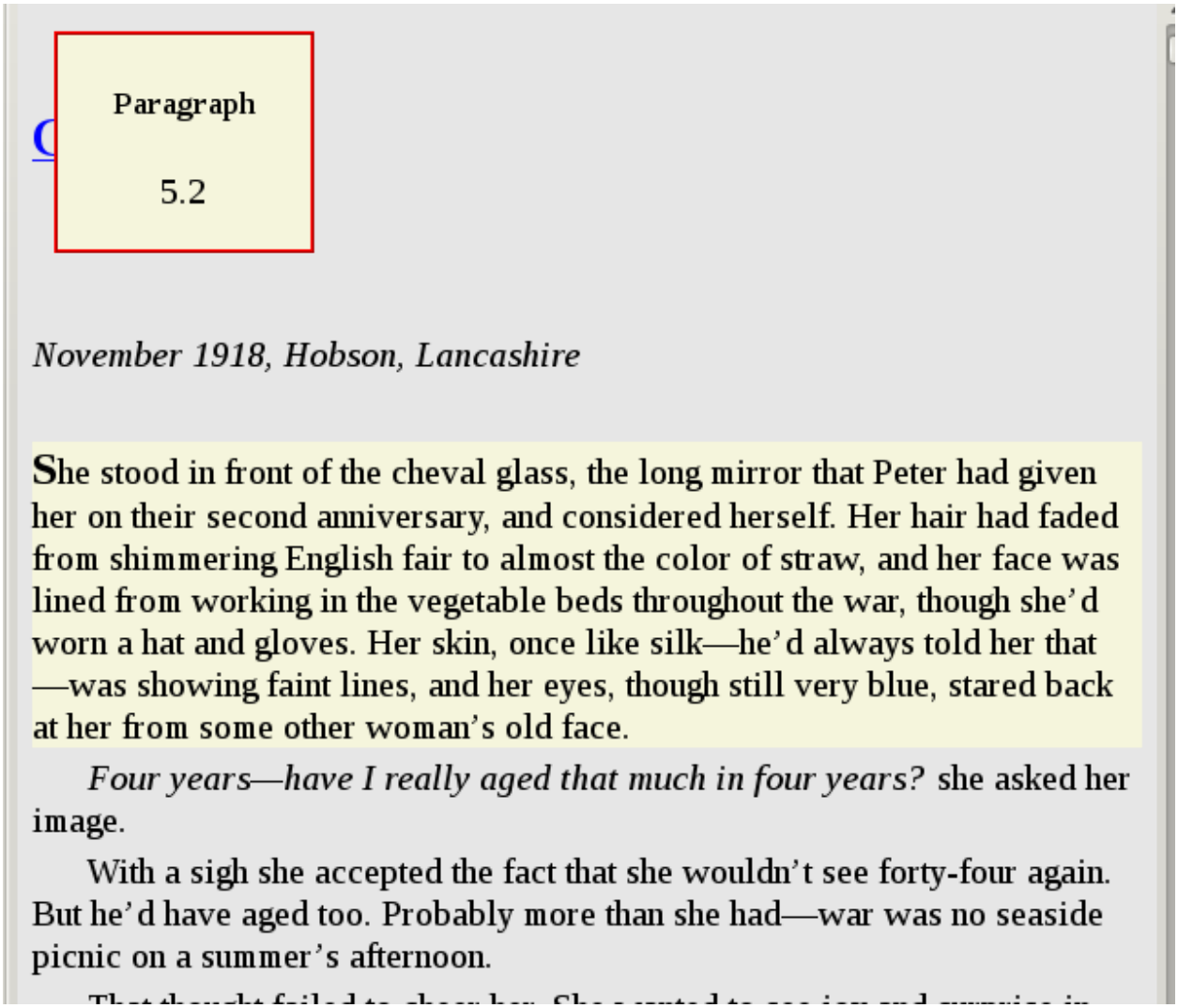
Table of Contents

If the book you are reading defines a Table of Contents, you can access it by pressing the Table of Contents button . This will bring up a list of sections in the book. You can click on any of them to jump to that portion of the book.

Navigating by location

Ebooks, unlike paper books, have no concept of pages. Instead, as you read through the book, you will notice that your position in the book is displayed in the upper left corner in a box like this . This is both your current position and the total length of the book. These numbers are independent of the screen size and font size you are viewing the book at, and they play a similar role to page numbers in paper books. You can enter any number you like to go to the corresponding location in the book.

calibre also has a very handy reference mode. You can turn it on by clicking the Reference Mode button . Once you do this, every time you move your mouse over a paragraph, calibre will display a unique number made up of the section and paragraph numbers.



You can use this number to unambiguously refer to parts of the books when discussing it with friends or referring to it in other works. You can enter these numbers in the box marked Go to at the top of the window to go to a particular reference location.

If you click on links inside the ebook to take you to different parts of the book, such as an endnote, you can use the back and forward buttons in the top left corner to return to where you were. These buttons behave just like those in a web browser.

Customizing the look and feel of your reading experience

You can change font sizes on the fly by using the font size buttons . You can also make the viewer full screen by pressing the Full Screen button . By clicking the Preferences button , you can change the default fonts used by the viewer to ones you like as well as the default font size when the viewer starts up.

More advanced customization can be achieved by the User Stylesheet setting. This is a stylesheet you can set that will be applied to every book. Using it you can do things like have white text on a black background, change paragraph styles, text justification, etc. For examples of custom stylesheets used by calibre's users, see the forums[128].

Dictionary lookup

You can look up the meaning of words in the current book by right clicking on a word. calibre uses the publicly available dictionary server at `dict.org` to look up words. The definition is displayed in a small box at the bottom of the screen.

Copying text and images

You can select text and images by dragging the content with your mouse and then right clicking to copy to the clipboard. The copied material can be pasted into another application as plain text and images.

1.15 Customizing calibre's ebook conversion

1.15.1 Ebook Conversion

calibre has a conversion system that is designed to be very easy to use. Normally, you just add a book to calibre, click convert and calibre will try hard to generate output that is as close as possible to the input. However, calibre accepts a very large number of input formats, not all of which are as suitable as others for conversion to ebooks. In the case of such input formats, or if you just want greater control over the conversion system, calibre has a lot of options to fine tune the conversion process. Note however that calibre's conversion system is not a substitute for a full blown ebook editor. To edit ebooks, I would recommend first converting them to EPUB using calibre and then using a dedicated EPUB editor, like Sigil[129] to get the book into perfect shape. You can then use the edited EPUB as input for conversion into other formats in calibre.

This document will refer mainly to the conversion settings as found in the conversion dialog, pictured below. All these settings are also available via command line interface to conversion, documented at *ebook-convert* (page 486). In calibre, you can obtain help on any individual setting by holding your mouse over it, a tooltip will appear describing the setting.



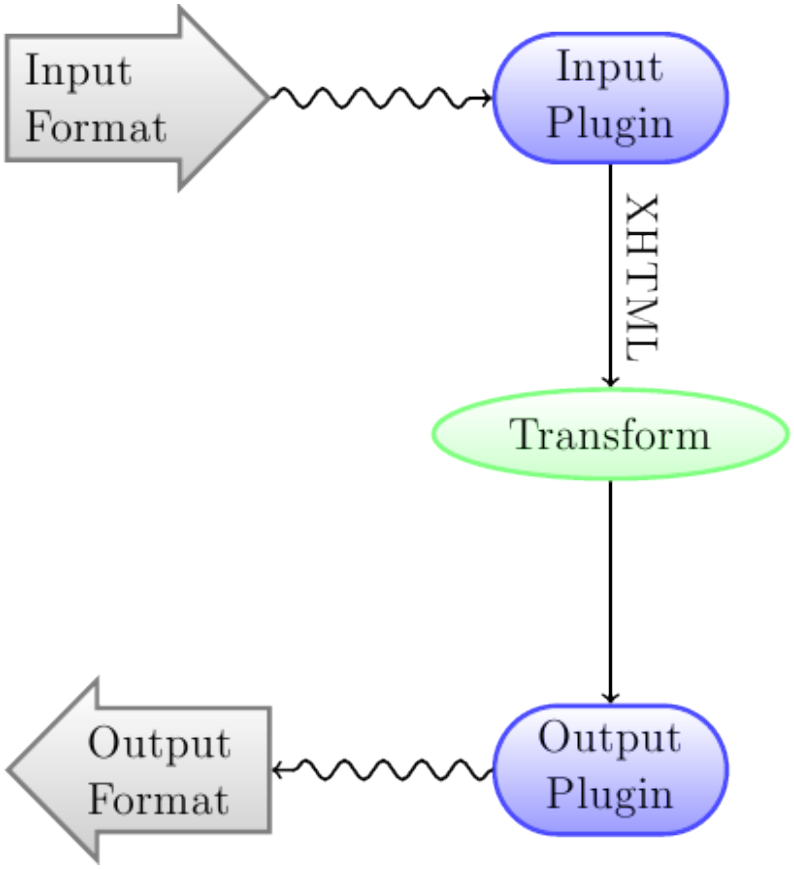
[128] http://www.mobileread.com/forums/showthread.php?t=51500
[129] http://code.google.com/p/sigil/

Contents

- Introduction (page 306)
- Look & Feel (page 307)
- Page Setup (page 309)
- Heuristic Processing (page 310)
- Search & Replace (page 311)
- Structure Detection (page 311)
- Table of Contents (page 312)
- Using images as chapter titles when converting HTML input documents (page 314)
- How options are set/saved for Conversion (page 314)
- Format specific tips (page 314)

Introduction

The first thing to understand about the conversion system is that it is designed as a pipeline. Schematically, it looks like this:



The input format is first converted to XHTML by the appropriate *Input Plugin*. This HTML is then *transformed*. In the last step, the processed XHTML is converted to the specified output format by the appropriate *Output Plugin*. The results of the conversion can vary greatly, based on the input format. Some formats convert much better than others. A list of the best source formats for conversion is available *here* (page 322).

The transforms that act on the XHTML output are where all the work happens. There are various transforms, for example, to insert book metadata as a page at the start of the book, to detect chapter headings and automatically create

a Table of Contents, to proportionally adjust font sizes, et cetera. It is important to remeber that all the transforms act on the XHTML output by the *Input Plugin*, not on the input file itself. So, for example, if you ask calibre to convert an RTF file to EPUB, it will first be converted to XHTML internally, the various transforms will be applied to the XHTML and then the *Output Plugin* will create the EPUB file, automatically generating all metadata, Table of Contents, et cetera.

You can see this process in action by using the debug option  . Just specify the path to a directory for the debug output. During conversion, calibre will place the XHTML generated by the various stages of the conversion pipeline in different sub-directories. The four sub-directories are:

Table 1.4: Stages of the conversion pipeline

Directory	Description
input	This contains the HTML output by the Input Plugin. Use this to debug the Input Plugin.
parsed	The result of pre-processing and converting to XHTML the output from the Input Plugin. Use to debug structure detection.
structure	Post structure detection, but before CSS flattening and font size conversion. Use to debug font size conversion and CSS transforms.
processed	Just before the ebook is passed to the output plugin. Use to debug the Output Plugin.

If you want to edit the input document a little before having calibre convert it, the best thing to do is edit the files in the `input` sub-directory, then zip it up, and use the zip file as the input format for subsequent conversions. To do this use the *Edit meta information* dialog to add the zip file as a format for the book and then, in the top left corner of the conversion dialog, select ZIP as the input format.

This document will deal mainly with the various transforms that operate on the intermediate XHTML and how to control them. At the end are some tips specific to each Input/Output format.

Look & Feel

Contents

- Font size rescaling (page 307)
- Paragraph spacing (page 308)
- Extra CSS (page 309)
- Miscellaneous (page 309)

This group of options controls various aspects of the look and feel of the converted ebook.

Font size rescaling

One of the nicest features of the e-reading experience is the ability to easily adjust font sizes to suit individual needs and lighting conditions. calibre has sophisticated algorithms to ensure that all the books it outputs have a consistent font sizes, no matter what font sizes are specified in the input document.

The base font size of a document is the most common font size in that document, i.e., the size of the bulk of text in that document. When you specify a *Base font size*, calibre automatically rescales all font sizes in the document

proportionately, so that the most common font size becomes the specified base font size and other font sizes are rescaled appropriately. By choosing a larger base font size, you can make the fonts in the document larger and vice versa. When you set the base font size, for best results, you should also set the font size key.

Normally, calibre will automatically choose a base font size appropriate to the Output Profile you have chosen (see *Page Setup* (page 309)). However, you can override this here in case the default is not suitable for you.

The *Font size key* option lets you control how non-base font sizes are rescaled. The font rescaling algorithm works using a font size key, which is simply a comma-separated list of font sizes. The font size key tells calibre how many "steps" bigger or smaller a given font size should be compared to the base font size. The idea is that there should be a limited number of font sizes in a document. For example, one size for the body text, a couple of sizes for different levels of headings and a couple of sizes for super/sub scripts and footnotes. The font size key allows calibre to compartmentalize the font sizes in the input documents into separate "bins" corresponding to the different logical font sizes.

Let's illustrate with an example. Suppose the source document we are converting was produced by someone with excellent eyesight and has a base font size of 8pt. That means the bulk of the text in the document is sized at 8pts, while headings are somewhat larger (say 10 and 12pt) and footnotes somewhat smaller at 6pt. Now if we use the following settings:

```
Base font size : 12pt
Font size key  : 7, 8, 10, 12, 14, 16, 18, 20
```

The output document will have a base font size of 12pt, headings of 14 and 16pt and footnotes of 8pt. Now suppose we want to make the largest heading size stand out more and make the footnotes a little larger as well. To achieve this, the font key should be changed to:

```
New font size key : 7, 9, 12, 14, 18, 20, 22
```

The largest headings will now become 18pt, while the footnotes will become 9pt. You can play with these settings to try and figure out what would be optimum for you by using the font rescaling wizard, which can be accessed by clicking the little button next to the *Font size key* setting.

All the font size rescaling in the conversion can also be disabled here, if you would like to preserve the font sizes in the input document.

A related setting is *Line height*. Line height controls the vertical height of lines. By default, (a line height of 0), no manipulation of line heights is performed. If you specify a non-default value, line heights will be set in all locations that don't specify their own line heights. However, this is something of a blunt weapon and should be used sparingly. If you want to adjust the line heights for some section of the input, it's better to use the *Extra CSS* (page 309).

Paragraph spacing

Normally, paragraphs in XHTML are rendered with a blank line between them and no leading text indent. calibre has a couple of options to control this. *Remove spacing between paragraphs* forcefully ensure that all paragraphs have no inter paragraph spacing. It also sets the text indent to 1.5em (can be changed) to mark the start of every paragraph. *Insert blank line* does the opposite, guaranteeing that there is exactly one blank line between each pair of paragraphs. Both these options are very comprehensive, removing spacing, or inserting it for *all* paragraphs (technically <p> and tags). This is so that you can just set the option and be sure that it performs as advertised, irrespective of how messy the input file is. The one exception is when the input file uses hard line breaks to implement inter-paragraph spacing.

If you want to remove the spacing between all paragraphs, except a select few, don't use these options. Instead add the following CSS code to *Extra CSS* (page 309):

```
p, div { margin: 0pt; border: 0pt; text-indent: 1.5em }
.spacious { margin-bottom: 1em; text-indent: 0pt; }
```

Then, in your source document, mark the paragraphs that need spacing with *class="spacious"*. If your input document is not in HTML, use the Debug option, described in the Introduction to get HTML (use the `input` sub-directory).

Extra CSS

This option allows you to specify arbitrary CSS that will be applied to all HTML files in the input. This CSS is applied with very high priority and so should override most CSS present in the **input document** itself. You can use this setting to fine tune the presentation/layout of your document. For example, if you want all paragraphs of class *endnote* to be right aligned, just add:

```
.endnote { text-align: right }
```

or if you want to change the indentation of all paragraphs:

```
p { text-indent: 5mm; }
```

Extra CSS is a very powerful option, but you do need an understanding of how CSS works to use it to its full potential. You can use the debug pipeline option described above to see what CSS is present in your input document.

Miscellaneous

There are a few more options in this section.

No text justification Normally, if the output format supports it, calibre will force the output ebook to have *justified* text (i.e., a smooth right margin). This option will turn off this behavior, in which case whatever justification is specified in the input document will be used instead.

Linearize tables Some badly designed documents use tables to control the layout of text on the page. When converted these documents often have text that runs off the page and other artifacts. This option will extract the content from the tables and present it in a linear fashion. Note that this option linearizes *all* tables, so only use it if you are sure the input document does not use tables for legitimate purposes, like presenting tabular information.

Transliterate unicode characters Transliterate unicode characters to an ASCII representation. Use with care because this will replace unicode characters with ASCII. For instance it will replace "Михаил Горбачёв" with "Mikhail Gorbachiov". Also, note that in cases where there are multiple representations of a character (characters shared by Chinese and Japanese for instance) the representation used by the largest number of people will be used (Chinese in the previous example). This option is mainly useful if you are going to view the ebook on a device that does not have support for unicode.

Input character encoding Older documents sometimes don't specify their character encoding. When converted, this can result in non-English characters or special characters like smart quotes being corrupted. calibre tries to auto-detect the character encoding of the source document, but it does not' always succeed. You can force it to assume a particular character encoding by using this setting. *cp1252* is a common encoding for documents produced using windows software. You should also read *How do I convert my file containing non-English characters, or smart quotes?* (page 322) for more on encoding issues.

Page Setup

The Page Setup options are for controlling screen layout, like margins and screen sizes. There are options to setup page margins, which will be used by the Output Plugin, if the selected Output Format supports page margins. In addition, you should choose an Input profile and an Output profile. Both sets of profiles basically deal with how to interpret measurements in the input/output documents, screen sizes and default font rescaling keys.

If you know that the file you are converting was intended to be used on a particular device/software platform, choose the corresponding input profile, otherwise just choose the default input profile. If you know the files you are producing

are meant for a particular device type, choose the corresponding Output profile. In particular, for MOBI Output files, you should choose the Kindle, for LIT the Microsoft Reader and for EPUB the Sony Reader. In the case of EPUB, the Sony Reader profile will result in EPUB files that will work everywhere. However, it has some side effects, like inserting artificial section breaks to keep internal components below the size threshold, needed for SONY devices. In particular for the iPhone/Android phones, choose the SONY output profile. If you know your EPUB files will not be read on a SONY or similar device, use the default output profile. If you are producing MOBI files that are not intended for the Kindle, choose the Mobipocket books output profile.

The Output profile also controls the screen size. This will cause, for example, images to be auto-resized to be fit to the screen in some output formats. So choose a profile of a device that has a screen size similar to your device.

Heuristic Processing

Heuristic Processing provides a variety of functions which can be used to try and detect and correct common problems in poorly formatted input documents. Use these functions if your input document suffers from poor formatting. Because these functions rely on common patterns, be aware that in some cases an option may lead to worse results, so use with care. As an example, several of these options will remove all non-breaking-space entities, or may include false positive matches relating to the function.

Enable heuristic processing This option activates calibre's Heuristic Processing stage of the conversion pipeline. This must be enabled in order for various sub-functions to be applied

Unwrap lines Enabling this option will cause calibre to attempt to detect and correct hard line breaks that exist within a document using punctuation clues and line length. calibre will first attempt to detect whether hard line breaks exist, if they do not appear to exist calibre will not attempt to unwrap lines. The line-unwrap factor can be reduced if you want to 'force' calibre to unwrap lines.

Line-unwrap factor This option controls the algorithm calibre uses to remove hard line breaks. For example, if the value of this option is 0.4, that means calibre will remove hard line breaks from the end of lines whose lengths are less than the length of 40% of all lines in the document. If your document only has a few line breaks which need correction, then this value should be reduced to somewhere between 0.1 and 0.2.

Detect and markup unformatted chapter headings and sub headings If your document does not have chapter headings and titles formatted differently from the rest of the text, calibre can use this option to attempt detection them and surround them with heading tags. <h2> tags are used for chapter headings; <h3> tags are used for any titles that are detected.

This function will not create a TOC, but in many cases it will cause calibre's default chapter detection settings to correctly detect chapters and build a TOC. Adjust the XPath under Structure Detection if a TOC is not automatically created. If there are no other headings used in the document then setting "//h:h2" under Structure Detection would be the easiest way to create a TOC for the document.

The inserted headings are not formatted, to apply formatting use the *Extra CSS* option under the Look and Feel conversion settings. For example, to center heading tags, use the following:

```
h2, h3 { text-align: center }
```

Renumber sequences of <h1> or <h2> tags Some publishers format chapter headings using multiple <h1> or <h2> tags sequentially. calibre's default conversion settings will cause such titles to be split into two pieces. This option will re-number the heading tags to prevent splitting.

Delete blank lines between paragraphs This option will cause calibre to analyze blank lines included within the document. If every paragraph is interleaved with a blank line, then calibre will remove all those blank paragraphs. Sequences of multiple blank lines will be considered scene breaks and retained as a single paragraph. This option differs from the 'Remove Paragraph Spacing' option under 'Look and Feel' in that it actually modifies the HTML content, while the other option modifies the document styles. This option can also remove paragraphs which were inserted using calibre's 'Insert blank line' option.

Ensure scene breaks are consistently formatted With this option calibre will attempt to detect common scene-break markers and ensure that they are center aligned. 'Soft' scene break markers, i.e. scene breaks only defined by extra white space, are styled to ensure that they will not be displayed in conjunction with page breaks.

Replace scene breaks If this option is configured then calibre will replace scene break markers it finds with the replacement text specified by the user. Please note that some ornamental characters may not be supported across all reading devices.

In general you should avoid using html tags, calibre will discard any tags and use pre-defined markup. <hr /> tags, i.e. horizontal rules, and <img> tags are exceptions. Horizontal rules can optionally be specified with styles, if you choose to add your own style be sure to include the 'width' setting, otherwise the style information will be discarded. Image tags can used, but calibre does not provide the ability to add the image during conversion, this must be done after the fact using the 'Tweak Book' feature, or Sigil.

Example image tag (place the image within an 'Images' folder inside the epub after conversion):

Example horizontal rule with styles: <hr style="width:20%;padding-top: 1px;border-top: 2px ridge black;border-bottom: 2px groove black;"/>

Remove unnecessary hyphens calibre will analyze all hyphenated content in the document when this option is enabled. The document itself is used as a dictionary for analysis. This allows calibre to accurately remove hyphens for any words in the document in any language, along with made-up and obscure scientific words. The primary drawback is words appearing only a single time in the document will not be changed. Analysis happens in two passes, the first pass analyzes line endings. Lines are only unwrapped if the word exists with or without a hyphen in the document. The second pass analyzes all hyphenated words throughout the document, hyphens are removed if the word exists elsewhere in the document without a match.

Italicize common words and patterns When enabled, calibre will look for common words and patterns that denote italics and italicize them. Examples are common text conventions such as ~word~ or phrases that should generally be italicized, e.g. latin phrases like 'etc.' or 'et cetera'.

Replace entity indents with CSS indents Some documents use a convention of defining text indents using non-breaking space entities. When this option is enabled calibre will attempt to detect this sort of formatting and convert them to a 3% text indent using css.

Search & Replace

These options are useful primarily for conversion of PDF documents or OCR conversions, though they can also be used to fix many document specific problems. As an example, some conversions can leaves behind page headers and footers in the text. These options use regular expressions to try and detect headers, footers, or other arbitrary text and remove or replace them. Remember that they operate on the intermediate XHTML produced by the conversion pipeline. There is a wizard to help you customize the regular expressions for your document. Click the magic wand beside the expression box, and click the 'Test' button after composing your search expression. Successful matches will be highlighted in Yellow.

The search works by using a python regular expression. All matched text is simply removed from the document or replaced using the replacement pattern. The replacement pattern is optional, if left blank then text matching the search pattern will be deleted from the document. You can learn more about regular expressions and their syntax at *All about using regular expressions in calibre* (page 415).

Structure Detection

Structure detection involves calibre trying its best to detect structural elements in the input document, when they are not properly specified. For example, chapters, page breaks, headers, footers, etc. As you can imagine, this process varies widely from book to book. Fortunately, calibre has very powerful options to control this. With power comes complexity, but if once you take the time to learn the complexity, you will find it well worth the effort.

Chapters and page breaks

calibre has two sets of options for *chapter detection* and *inserting page breaks*. This can sometimes be slightly confusing, as by default, calibre will insert page breaks before detected chapters as well as the locations detected by the page breaks option. The reason for this is that there are often location where page breaks should be inserted that are not chapter boundaries. Also, detected chapters can be optionally inserted into the auto generated Table of Contents.

calibre uses *XPath*, a powerful language to allow the user to specify chapter boundaries/page breaks. XPath can seem a little daunting to use at first, fortunately, there is a *XPath tutorial* (page 373) in the User Manual. Remember that Structure Detection operates on the intermediate XHTML produced by the conversion pipeline. Use the debug option described in the *Introduction* (page 306) to figure out the appropriate settings for your book. There is also a button for a XPath wizard to help with the generation of simple XPath expressions.

By default, calibre uses the following expression for chapter detection:

```
//*[((name()='h1' or name()='h2') and re:test(., 'chapter|book|section|part\s+', 'i')) or @class = 'c
```

This expression is rather complex, because it tries to handle a number of common cases simulataneously. What it means is that calibre will assume chapters start at either <*h1*> or <*h2*> tags that have any of the words *(chapter, book, section or part)* in them or that have the *class="chapter"* attribute.

A related option is *Chapter mark*, which allows you to control what calibre does when it detects a chapter. By default, it will insert a page break before the chapter. You can have it insert a ruled line instead of, or in addition to the page break. You can also have it do nothing.

The default setting for detecting page breaks is:

```
//*[name()='h1' or name()='h2']
```

which means that calibre will insert page breaks before every <*h1*> and <*h2*> tag by default.

Note: The default expressions may change depending on the input format you are converting.

Miscellaneous

There are a few more options in this section.

Insert metadata as page at start of book One of the great things about calibre is that it allows you to maintain very complete metadata about all of your books, for example, a rating, tags, comments, etc. This option will create a single page with all this metadata and insert it into the converted ebook, typically just after the cover. Think of it as a way to create your own customised book jacket.

Remove first image Sometimes, the source document you are converting includes the cover as part of the book, instead of as a separate cover. If you also specify a cover in calibre, then the converted book will have two covers. This option will simply remove the first image from the source document, thereby ensuring that the converted book has only one cover, the one specified in calibre.

Table of Contents

When the input document has a Table of Contents in its metadata, calibre will just use that. However, a number of older formats either do not support a metadata based Table of Contents, or individual documents do not have one. In these cases, the options in this section can help you automatically generate a Table of Contents in the converted ebook, based on the actual content in the input document.

The first option is *Force use of auto-generated Table of Contents*. By checking this option you can have calibre override any Table of Contents found in the metadata of the input document with the auto generated one.

The default way that the creation of the auto generated Table of Contents works is that, calibre will first try to add any detected chapters to the generated table of contents. You can learn how to customize the detection of chapters in the *Structure Detection* (page 311) section above. If you do not want to include detected chapters in the generated table of contents, check the *Do not add detected chapters* option.

If less than the *Chapter threshold* number of chapters were detected, calibre will then add any hyperlinks it finds in the input document to the Table of Contents. This often works well many input documents include a hyperlinked Table of Contents right at the start. The *Number of links* option can be used to control this behavior. If set to zero, no links are added. If set to a number greater than zero, at most that number of links is added.

calibre will automatically filter duplicates from the generated Table of Contents. However, if there are some additional undesirable entries, you can filter them using the *TOC Filter* option. This is a regular expression that will match the title of entries in the generated table of contents. Whenever a match is found, it will be removed. For example, to remove all entries titles "Next" or "Previous" use:

```
Next|Previous
```

Finally, the *Level 1,2,3 TOC* options allow you to create a sophisticated multi-level Table of Contents. They are XPath expressions that match tags in the intermediate XHTML produced by the conversion pipeline. See the *Introduction* (page 306) for how to get access to this XHTML. Also read the *XPath Tutorial* (page 373), to learn how to construct XPath expressions. Next to each option is a button that launches a wizard to help with the creation of basic XPath expressions. The following simple example illustrates how to use these options.

Suppose you have an input document that results in XHTML that look like this:

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Sample document</title>
    </head>
    <body>
        <h1>Chapter 1</h1>
        ...
        <h2>Section 1.1</h2>
        ...
        <h2>Section 1.2</h2>
        ...
        <h1>Chapter 2</h1>
        ...
        <h2>Section 2.1</h2>
        ...
    </body>
</html>
```

Then, we set the options as:

```
Level 1 TOC : //h:h1
Level 2 TOC : //h:h2
```

This will result in an automatically generated two level Table of Contents that looks like:

```
Chapter 1
    Section 1.1
    Section 1.2
Chapter 2
    Section 2.1
```

Warning: Not all output formats support a multi level Table of Contents. You should first try with EPUB Output. If that works, then try your format of choice.

Using images as chapter titles when converting HTML input documents

Suppose you want to use an image as your chapter title, but still want calibre to be able to automatically generate a Table of Contents for you from the chapter titles. Use the following HTML markup to achieve this

```
<html>
    <body>
        <h2>Chapter 1</h2>
        <p>chapter 1 text...</p>
        <h2 title="Chapter 2"><img src="chapter2.jpg" /></h2>
        <p>chapter 2 text...</p>
    </body>
</html>
```

Set the *Level 1 TOC* setting to `//h:h2`. Then, for chapter two, calibre will take the title from the value of the `title` attribute on the `<h2>` tag, since the tag has no text.

How options are set/saved for Conversion

There are two places where conversion options can be set in calibre. The first is in Preferences->Conversion. These settings are the defaults for the conversion options. Whenever you try to convert a new book, the settings set here will be used by default.

You can also change settings in the conversion dialog for each book conversion. When you convert a book, calibre remembers the settings you used for that book, so that if you convert it again, the saved settings for the individual book will take precedence over the defaults set in Preferences. You can restore the individual settings to defaults by using the Restore to defaults button in the individual book conversion dialog.

When you Bulk Convert a set of books, settings are taken in the following order:

- From the defaults set in Preferences->Conversion
- From the saved conversion settings for each book being converted (if any). This can be turned off by the option in the top left corner of the Bulk Conversion dialog.
- From the settings set in the Bulk conversion dialog

Note that the final settings for each book in a Bulk Conversion will be saved and re-used if the book is converted again. Since the highest priority in Bulk Conversion is given to the settings in the Bulk Conversion dialog, these will override any book specific settings. So you should only bulk convert books together that need similar settings. The exceptions are metadata and input format specific settings. Since the Bulk Conversion dialog does not have settings for these two categories, they will be taken from book specific settings (if any) or the defaults.

Note: You can see the actual settings used during any conversion by clicking the rotating icon in the lower right corner and then double clicking the individual conversion job. This will bring up a conversion log that will contain the actual settings used, near the top.

Format specific tips

Here you will find tips specific to the conversion of particular formats. Options specific to particular format, whether input or output are available in the conversion dialog under their own section, for example *TXT Input* or *EPUB Output*.

Convert Microsoft Word documents

calibre does not directly convert .doc/.docx files from Microsoft Word. However, in Word, you can save the document as HTML and then convert the resulting HTML file with calibre. When saving as HTML, be sure to use the "Save as Web Page, Filtered" option as this will produce clean HTML that will convert well. Note that Word produces really messy HTML, converting it can take a long time, so be patient. Another alternative is to use the free OpenOffice. Open your .doc file in OpenOffice and save it in OpenOffice's format .odt. calibre can directly convert .odt files.

There is a Word macro package that can automate the conversion of Word documents using calibre. It also makes generating the Table of Contents much simpler. It is called BookCreator and is available for free at mobileread[130].

An easy way to generate a Table of Contents when converting a Word document is:

1. Mark your Chapters and sub-Chapters in the doc file with one of the MS built-in styles called 'Heading 1', 'Heading 2', ..., 'Heading 6'. 'Heading 1' equates to the HTML tag <h1>, 'Heading 2' to <h2> etc
2. Save the doc as Webpage-filtered (rather than Webpage) and import the html file into calibre
3. When you convert in calibre you use what you did in step 1 to set the box called 'Detect chapters at' on the Convert - Structure Detection page. For example:

- If you mark Chapters with style 'Heading 2' then set the 'Detect chapters at' box to //h:h2 This will give you a proper external metadata TOC in the converted epub.
- A slightly more complex example...if your book has Sections and Chapters and you want a 2-level nested metadata TOC. Mark the doc Sections with style 'Heading 2' and the Chapters with style 'Heading 3'. When you convert set the 'Detect chapters at' box to //h:h2|//h:h3. On the Convert - TOC page set the 'Level 1 TOC' box to //h:h2 and the 'Level 2 TOC' box to //h:h3.

Convert TXT documents

TXT documents have no well defined way to specify formatting like bold, italics, etc, or document structure like paragraphs, headings, sections and so on, but there are a variety of conventions commonly used. By default calibre attempts automatic detection of the correct formatting and markup based on those conventions.

TXT input supports a number of options to differentiate how paragraphs are detected.

Paragraph Style: Auto Analyzes the text file and attempts to automatically determine how paragraphs are defined. This option will generally work fine, if you achieve undesirable results try one of the manual options.

Paragraph Style: Block Assumes one or more blank lines are a paragraph boundary:

```
This is the first.

This is the
second paragraph.
```

Paragraph Style: Single Assumes that every line is a paragraph:

```
This is the first.
This is the second.
This is the third.
```

Paragraph Style: Print Assumes that every paragraph starts with an indent (either a tab or 2+ spaces). Paragraphs end when the next line that starts with an indent is reached:

[130] http://www.mobileread.com/forums/showthread.php?t=28313

```
  This is the
first.
  This is the second.

  This is the
third.
```

Paragraph Style: Unformatted Assumes that the document has no formatting, but does use hard line breaks. Punctuation and median line length are used to attempt to re-create paragraphs.

Formatting Style: Auto Attemtps to detect the type of formatting markup being used. If no markup is used then heuristic formatting will be applied.

Formatting Style: Heuristic Analyzes the document for common chapter headings, scene breaks, and italicized words and applies the appropriate html markup during conversion.

Formatting Style: Markdown calibre also supports running TXT input though a transformation preprocessor known as markdown. Markdown allows for basic formatting to be added to TXT documents, such as bold, italics, section headings, tables, lists, a Table of Contents, etc. Marking chapter headings with a leading # and setting the chapter XPath detection expression to "//h:h1" is the easiest way to have a proper table of contents generated from a TXT document. You can learn more about the markdown syntax at daringfireball[131].

Formatting Style: None Applies no special formatting to the text, the document is converted to html with no other changes.

Convert PDF documents

PDF documents are one of the worst formats to convert from. They are a fixed page size and text placement format. Meaning, it is very difficult to determine where one paragraph ends and another begins. calibre will try to unwrap paragraphs using a configurable, *Line Un-Wrapping Factor*. This is a scale used to determine the length at which a line should be unwrapped. Valid values are a decimal between 0 and 1. The default is 0.45, just under the median line length. Lower this value to include more text in the unwrapping. Increase to include less. You can adjust this value in the conversion settings under *PDF Input*.

Also, they often have headers and footers as part of the document that will become included with the text. Use the Search and Replace panel to remove headers and footers to mitigate this issue. If the headers and footers are not removed from the text it can throw off the paragraph unwrapping. To learn how to use the header and footer removal options, read *All about using regular expressions in calibre* (page 415).

Some limitations of PDF input are:

- Complex, multi-column, and image based documents are not supported.
- Extraction of vector images and tables from within the document is also not supported.
- Some PDFs use special glyphs to represent ll or ff or fi, etc. Conversion of these may or may not work depending on just how they are represented internally in the PDF.
- Links and Tables of Contents are not supported
- PDFs that use embedded non-unicode fonts to represent non-English characters will result in garbled output for those characters
- Some PDFs are made up of photographs of the page with OCRed text behind them. In such cases calibre uses the OCRed text, which can be very different from what you see when you view the PDF file

[131] http://daringfireball.net/projects/markdown/syntax

- PDFs that are used to display complex text, like right to left languages and math typesetting will not convert correctly

To re-iterate **PDF is a really, really bad** format to use as input. If you absolutely must use PDF, then be prepared for an output ranging anywhere from decent to unusable, depending on the input PDF.

Comic Book Collections

A comic book collection is a .cbc file. A .cbc file is a zip file that contains other CBZ/CBR files. In addition the .cbc file must contain a simple text file called comics.txt, encoded in UTF-8. The comics.txt file must contain a list of the comics files inside the .cbc file, in the form filename:title, as shown below:

```
one.cbz:Chapter One
two.cbz:Chapter Two
three.cbz:Chapter Three
```

The .cbc file will then contain:

```
comics.txt
one.cbz
two.cbz
three.cbz
```

calibre will automatically convert this .cbc file into a ebook with a Table of Contents pointing to each entry in comics.txt.

EPUB advanced formatting demo

Various advanced formatting for EPUB files is demonstrated in this demo file[132]. The file was created from hand coded HTML using calibre and is meant to be used as a template for your own EPUB creation efforts.

The source HTML it was created from is available demo.zip[133]. The settings used to create the EPUB from the ZIP file are:

```
ebook-convert demo.zip .epub -vv --authors "Kovid Goyal" --language en --level1-toc '//*[@class="titl
```

Note that because this file explores the potential of EPUB, most of the advanced formatting is not going to work on readers less capable than calibre's built-in EPUB viewer.

Convert ODT documents

calibre can directly convert ODT (OpenDocument Text) files. You should use styles to format your document and minimize the use of direct formatting. When inserting images into your document you need to anchor them to the paragraph, images anchored to a page will all end up in the front of the conversion.

To enable automatic detection of chapters, you need to mark them with the build-in styles called 'Heading 1', 'Heading 2', ..., 'Heading 6' ('Heading 1' equates to the HTML tag <h1>, 'Heading 2' to <h2> etc). When you convert in calibre you can enter which style you used into the 'Detect chapters at' box. Example:

- If you mark Chapters with style 'Heading 2', you have to set the 'Detect chapters at' box to `//h:h2`
- For a nested TOC with Sections marked with 'Heading 2' and the Chapters marked with 'Heading 3' you need to enter `//h:h2|//h:h3`. On the Convert - TOC page set the 'Level 1 TOC' box to `//h:h2` and the 'Level 2 TOC' box to `//h:h3`.

[132] http://calibre-ebook.com/downloads/demos/demo.epub
[133] http://calibre-ebook.com/downloads/demos/demo.zip

Well-known document properties (Title, Keywords, Description, Creator) are recognized and calibre will use the first image (not to small, and with good aspect-ratio) as the cover image.

There is also an advanced property conversion mode, which is activated by setting the custom property `opf.metadata` ('Yes or No' type) to Yes in your ODT document (File->Properties->Custom Properties). If this property is detected by calibre, the following custom properties are recognized (`opf.authors` overrides document creator):

```
opf.titlesort
opf.authors
opf.authorsort
opf.publisher
opf.pubdate
opf.isbn
opf.language
opf.series
opf.seriesindex
```

In addition to this, you can specify the picture to use as the cover by naming it `opf.cover` (right click, Picture->Options->Name) in the ODT. If no picture with this name is found, the 'smart' method is used. As the cover detection might result in double covers in certain output formats, the process will remove the paragraph (only if the only content is the cover!) from the document. But this works only with the named picture!

To disable cover detection you can set the custom property `opf.nocover` ('Yes or No' type) to Yes in advanced mode.

1.16 Editing ebook metadata

1.16.1 Editing Ebook Metadata

Contents

- Editing the metadata of one book at a time (page 318)
  - Downloading metadata (page 319)
  - Managing book formats (page 319)
  - All about covers (page 319)
- Editing the metadata of many books at a time (page 319)
  - Search and replace (page 320)
  - Bulk downloading of metadata (page 321)

Ebooks come in all shapes and sizes and more often than not, their metadata (things like title/author/series/publisher) is incomplete or incorrect. The simplest way to change metadata in calibre is to simply double click on an entry and type in the correct replacement. For more sophisticated, "power editing" use the edit metadata tools discussed below.

Editing the metadata of one book at a time

Click the book you want to edit and then click the *Edit metadata* button or press the `E` key. A dialog opens that allows you to edit all aspects of the metadata. It has various features to make editing faster and more efficient. A list of the commonly used tips:

- You can click the button in between title and authors to swap them automatically.

- You can click the button next to author sort to have calibre automatically fill it in using the sort values stored with each author. Use the *Manage authors* dialog to see and change the authors' sort values. This dialog can be opened by clicking and holding the button next to author sort.
- You can click the button next to tags to use the Tag Editor to manage the tags associated with the book.
- The ISBN box will have a red background if you enter an invalid ISBN. It will be green for valid ISBNs
- The author sort box will be red if the author sort value differs from what calibre thinks it should be.

Downloading metadata

The nicest feature of the edit metadata dialog is its ability to automatically fill in many metadata fields by getting metadata from various websites. Currently, calibre uses isbndb.com, Google Books, Amazon and Library Thing. The metadata download can fill in Title, author, series, tags, rating, description and ISBN for you.

To use the download, fill in the title and author fields and click the *Fetch metadata* button. calibre will present you with a list of books that most closely match the title and author. If you fill in the ISBN field first, it will be used in preference to the title and author. If no matches are found, try making your search a little less specific by including only some key words in the title and only the author last name.

Managing book formats

In calibre, a single book entry can have many different *formats* associated with it. For example you may have obtained the Complete Works of Shakespeare in EPUB format and later converted it to MOBI to read on your Kindle. calibre automatically manages multiple formats for you. In the *Available formats* section of the Edit metadata dialog, you can manage these formats. You can add a new format, delete an existing format and also ask calibre to set the metadata and cover for the book entry from the metadata in one of the formats.

All about covers

You can ask calibre to download book covers for you, provided the book has a known ISBN. Alternatively you can specify a file on your computer to use as the cover. calibre can even generate a default cover with basic metadata on it for you. You can drag and drop images onto the cover to change it and also right click to copy/paste cover images.

In addition, there is a button to automatically trim borders from the cover, in case your cover image has an ugly border.

Editing the metadata of many books at a time

First select the books you want to edit by holding Ctrl or Shift and clicking on them. If you select more than one book, clicking the *Edit metadata* button will cause a new *Bulk* metadata edit dialog to open. Using this dialog, you can quickly set the author/publisher/rating/tags/series etc of a bunch of books to the same value. This is particularly useful if you have just imported a number of books that have some metadata in common. This dialog is very powerful, for example, it has a Search and Replace tab that you can use to perform bulk operations on metadata and even copy metadata from one column to another.

The normal edit metadata dialog also has Next and Previous buttons that you can use to edit the metadata of several books one after the other.

Search and replace

The Bulk metadata edit dialog allows you to perform arbitrarily powerful search and replace operations on the selected books. By default it uses a simple text search and replace, but it also support *regular expressions*. For more on regular expressions, see *All about using regular expressions in calibre* (page 415).

As noted above, there are two search and replace modes: character match and regular expression. Character match will look in the *Search field* you choose for the characters you type in the *search for* box and replace those characters with what you type in the *replace with* box. Each occurance of the search characters in the field will be replaced. For example, assume the field being searched contains *a bad cat*. if you search for *a* to be replaced with *HELLO*, then the result will be *HELLO bHELLOd cHELLOt*.

If the field you are searching on is a *multiple* field like tags, then each tag is treated separately. For example, if your tags contain *Horror, Scary*, the search expression *r,* will not match anything because the expression will first be applied to *Horror* and then to *Scary*.

If you want the search to ignore upper/lowercase differences, uncheck the *Case sensitive* box.

You can have calibre change the case of the result (information after the replace has happened) by choosing one of the functions from the *Apply function after replace* box. The operations available are:

- *Lower case* – change all the characters in the field to lower case
- *Upper case* – change all the characters in the field to upper case
- *Title case* – capitalize each word in the result.

The *Your test* box is provided for you to enter text to check that search/replace is doing what you want. In the majority of cases the book test boxes will be sufficient, but it is possible that there is a case you want to check that isn't shown in these boxes. Enter that case into *Your test*.

Regular expression mode has some differences from character mode, beyond (of course) using regular expressions. The first is that functions are applied to the parts of the string matched by the search string, not the entire field. The second is that functions apply to the replacement string, not to the entire field.

The third and most important is that the replace string can make reference to parts of the search string by using backreferences. A backreference is `\\n` where n is an integer that refers to the n'th parenthesized group in the search expression. For example, given the same example as above, *a bad cat*, a search expression *a (...) (...)*, and a replace expression *a \2 \1*, the result will be *a cat bad*. Please see the *All about using regular expressions in calibre* (page 415) for more information on backreferences.

One useful pattern: assume you want to change the case of an entire field. The easiest way to do this is to use character mode, but lets further assume you want to use regular expression mode. The search expression should be *(.*)* the replace expression should be *\1*, and the desired case change function should be selected.

Finally, in regular expression mode you can copy values from one field to another. Simply make the source and destination field different. The copy can replace the destination field, prepend to the field (add to the front), or append to the field (add at the end). The 'use comma' checkbox tells calibre to (or not to) add a comma between the text and the destination field in prepend and append modes. If the destination is multiple (e.g., tags), then you cannot uncheck this box.

Search and replace is done after all the other metadata changes in the other tabs are applied. This can lead to some confusion, because the test boxes will show the information before the other changes, but the operation will be applied after the other changes. If you have any doubts about what is going to happen, do not mix search/replace with other changes.

Bulk downloading of metadata

If you want to download the metadata for multiple books at once, right-click the *Edit metadata* button and select *Download metadata*. You can choose to download only metadata, only covers, or both.

1.17 Frequently Asked Questions

1.17.1 Frequently Asked Questions

Contents

- Ebook Format Conversion (page 321)
- Device Integration (page 324)
- Library Management (page 331)
- Content From The Web (page 335)
- Miscellaneous (page 335)

Ebook Format Conversion

Contents

- What formats does calibre support conversion to/from? (page 321)
- What are the best source formats to convert? (page 322)
- I converted a PDF file, but the result has various problems? (page 322)
- How do I convert my file containing non-English characters, or smart quotes? (page 322)
- What's the deal with Table of Contents in MOBI files? (page 322)
- The covers for my MOBI files have stopped showing up in Kindle for PC/Kindle for Android/iPad etc. (page 323)
- How do I convert a collection of HTML files in a specific order? (page 323)
- The EPUB I produced with calibre is not valid? (page 324)
- How do I use some of the advanced features of the conversion tools? (page 324)

What formats does calibre support conversion to/from?

calibre supports the conversion of many input formats to many output formats. It can convert every input format in the following list, to every output format.

Input Formats: CBZ, CBR, CBC, CHM, DJVU, EPUB, FB2, HTML, HTMLZ, LIT, LRF, MOBI, ODT, PDF, PRC, PDB, PML, RB, RTF, SNB, TCR, TXT, TXTZ

Output Formats: AZW3, EPUB, FB2, OEB, LIT, LRF, MOBI, HTMLZ, PDB, PML, RB, PDF, RTF, SNB, TCR, TXT, TXTZ

Note: PRC is a generic format, calibre supports PRC files with TextRead and MOBIBook headers. PDB is also a generic format. calibre supports eReder, Plucker, PML and zTxt PDB files. DJVU support is only for converting DJVU files that contain embedded text. These are typically generated by OCR software. MOBI books can be of two types Mobi6 and KF8. calibre fully supports both. MOBI files often have .azw or .azw3 file extensions

---

What are the best source formats to convert?

In order of decreasing preference: LIT, MOBI, AZW, EPUB, AZW3, FB2, HTML, PRC, RTF, PDB, TXT, PDF

I converted a PDF file, but the result has various problems?

PDF is a terrible format to convert from. For a list of the various issues you will encounter when converting PDF, see: *Convert PDF documents* (page 316).

How do I convert my file containing non-English characters, or smart quotes?

There are two aspects to this problem:

1. Knowing the encoding of the source file: calibre tries to guess what character encoding your source files use, but often, this is impossible, so you need to tell it what encoding to use. This can be done in the GUI via the *Input character encoding* field in the *Look & Feel* section. The command-line tools all have an `--input-encoding` option.
2. When adding HTML files to calibre, you may need to tell calibre what encoding the files are in. To do this go to *Preferences->Advanced->Plugins->File Type plugins* and customize the HTML2Zip plugin, telling it what encoding your HTML files are in. Now when you add HTML files to calibre they will be correctly processed. HTML files from different sources often have different encodings, so you may have to change this setting repeatedly. A common encoding for many files from the web is `cp1252` and I would suggest you try that first. Note that when converting HTML files, leave the input encoding setting mentioned above blank. This is because the HTML2ZIP plugin automatically converts the HTML files to a standard encoding (utf-8).
3. Embedding fonts: If you are generating an LRF file to read on your SONY Reader, you are limited by the fact that the Reader only supports a few non-English characters in the fonts it comes pre-loaded with. You can work around this problem by embedding a unicode-aware font that supports the character set your file uses into the LRF file. You should embed atleast a serif and a sans-serif font. Be aware that embedding fonts significantly slows down page-turn speed on the reader.

What's the deal with Table of Contents in MOBI files?

The first thing to realize is that most ebooks have two tables of contents. One is the traditional Table of Contents, like the TOC you find in paper books. This Table of Contents is part of the main document flow and can be styled however you like. This TOC is called the *content TOC*.

Then there is the *metadata TOC*. A metadata TOC is a TOC that is not part of the book text and is typically accessed by some special button on a reader. For example, in the calibre viewer, you use the Show Table of Contents button to see this TOC. This TOC cannot be styled by the book creator. How it is represented is up to the viewer program.

In the MOBI format, the situation is a little confused. This is because the MOBI format, alone amongst mainstream ebook formats, *does not* have decent support for a metadata TOC. A MOBI book simulates the presence of a metadata TOC by putting an *extra* content TOC at the end of the book. When you click Goto Table of Contents on your Kindle, it is to this extra content TOC that the Kindle takes you.

Now it might well seem to you that the MOBI book has two identical TOCs. Remember that one is semantically a content TOC and the other is a metadata TOC, even though both might have exactly the same entries and look the same. One can be accessed directly from the Kindle's menus, the other cannot.

When converting to MOBI, calibre detects the *metadata TOC* in the input document and generates an end-of-file TOC in the output MOBI file. You can turn this off by an option in the MOBI Output settings. You can also tell calibre

whether to put it and the start or the end of the book via an option in the MOBI Output settings. Remember this TOC is semantically a *metadata TOC*, in any format other than MOBI it *cannot not be part of the text*. The fact that it is part of the text in MOBI is an accident caused by the limitations of MOBI. If you want a TOC at a particular location in your document text, create one by hand. So we strongly recommend that you leave the default as it is, i.e. with the metadata TOC at the end of the book.

If you have a hand edited TOC in the input document, you can use the TOC detection options in calibre to automatically generate the metadata TOC from it. See the conversion section of the User Manual for more details on how to use these options.

Finally, I encourage you to ditch the content TOC and only have a metadata TOC in your ebooks. Metadata TOCs will give the people reading your ebooks a much superior navigation experience (except on the Kindle, where they are essentially the same as a content TOC).

The covers for my MOBI files have stopped showing up in Kindle for PC/Kindle for Android/iPad etc.

This is caused by a bug in the Amazon software. You can work around it by going to Preferences->Output Options->MOBI output and setting the "Enable sharing of book content" option. If you are reconverting a previously converted book, you will also have to enable the option in the conversion dialog for that individual book (as per book conversion settings are saved and take precedence).

Note that doing this will mean that the generated MOBI will show up under personal documents instead of Books on the Kindle Fire and Amazon whispersync will not work, but the covers will. It's your choice which functionality is more important to you. I encourage you to contact Amazon and ask them to fix this bug.

How do I convert a collection of HTML files in a specific order?

In order to convert a collection of HTML files in a specific oder, you have to create a table of contents file. That is, another HTML file that contains links to all the other files in the desired order. Such a file looks like:

```
<html>
   <body>
     <h1>Table of Contents</h1>
     <p style="text-indent:0pt">
        <a href="file1.html">First File</a><br/>
        <a href="file2.html">Second File</a><br/>
        .
        .
        .
     </p>
   </body>
</html>
```

Then just add this HTML file to the GUI and use the convert button to create your ebook.

Note: By default, when adding HTML files, calibre follows links in the files in *depth first* order. This means that if file A.html links to B.html and C.html and D.html, but B.html also links to D.html, then the files will be in the order A.html, B.html, D.html, C.html. If instead you want the order to be A.html, B.html, C.html, D.html then you must tell calibre to add your files in *breadth first* order. Do this by going to Preferences->Plugins and customizing the HTML to ZIP plugin.

The EPUB I produced with calibre is not valid?

calibre does not guarantee that an EPUB produced by it is valid. The only guarantee it makes is that if you feed it valid XHTML 1.1 + CSS 2.1 it will output a valid EPUB. calibre is designed for ebook consumers, not producers. It tries hard to ensure that EPUBs it produces actually work as intended on a wide variety of devices, a goal that is incompatible with producing valid EPUBs, and one that is far more important to the vast majority of its users. If you need a tool that always produces valid EPUBs, calibre is not for you.

How do I use some of the advanced features of the conversion tools?

You can get help on any individual feature of the converters by mousing over it in the GUI or running `ebook-convert`

- html-demo.zip[134]

Device Integration

Contents

- What devices does calibre support? (page 324)
- How can I help get my device supported in calibre? (page 324)
- My device is not being detected by calibre? (page 325)
- My device is non-standard or unusual. What can I do to connect to it? (page 325)
- How does calibre manage collections on my SONY reader? (page 325)
- Can I use both calibre and the SONY software to manage my reader? (page 326)
- How do I use calibre with my iPad/iPhone/iPod touch? (page 327)
- How do I use calibre with my Android phone/tablet or Kindle Fire HD? (page 328)
- Can I access my calibre books using the web browser in my Kindle or other reading device? (page 329)
- I get the error message "Failed to start content server: Port 8080 not free on '0.0.0.0'"? (page 329)
- I cannot send emails using calibre? (page 329)
- Why is my device not detected in linux? (page 329)
- My device is getting mounted read-only in linux, so calibre cannot connect to it? (page 330)
- Why does calibre not support collections on the Kindle or shelves on the Nook? (page 330)
- I am getting an error when I try to use calibre with my Kobo Touch/Glo/etc.? (page 330)

What devices does calibre support?

calibre can directly connect to all the major (and most of the minor) ebook reading devices, smarthphones, tablets, etc. In addition, using the *Connect to folder* function you can use it with any ebook reader that exports itself as a USB disk. You can even connect to Apple devices (via iTunes), using the *Connect to iTunes* function.

How can I help get my device supported in calibre?

If your device appears as a USB disk to the operating system, adding support for it to calibre is very easy. We just need some information from you:

- Complete list of ebook formats that your device supports.

[134] http://calibre-ebook.com/downloads/html-demo.zip

- Is there a special directory on the device in which all ebook files should be placed? Also does the device detect files placed in sub directories?
- We also need information about your device that calibre will collect automatically. First, if your device supports SD cards, insert them. Then connect your device to the computer. In calibre go to *Preferences->Advanced->Miscellaneous* and click the "Debug device detection" button. This will create some debug output. Copy it to a file and repeat the process, this time with your device disconnected from your computer.
- Send both the above outputs to us with the other information and we will write a device driver for your device.

Once you send us the output for a particular operating system, support for the device in that operating system will appear in the next release of calibre. To send us the output, open a bug report and attach the output to it. See calibre bugs[135].

My device is not being detected by calibre?

Follow these steps to find the problem:

- Make sure that you are connecting only a single device to your computer at a time. Do not have another calibre supported device like an iPhone/iPad etc. at the same time.
- If you are connecting an Apple iDevice (iPad, iPod Touch, iPhone), use the 'Connect to iTunes' method in the 'Getting started' instructions in Calibre + Apple iDevices: Start here[136].
- Make sure you are running the latest version of calibre. The latest version can always be downloaded from the calibre website[137]. You can tell what version of calibre you are currently running by looking at the bottom line of the main calibre window.
- Ensure your operating system is seeing the device. That is, the device should show up in Windows Explorer (in Windows) or Finder (in OS X).
- In calibre, go to Preferences->Ignored Devices and check that your device is not being ignored
- If all the above steps fail, go to Preferences->Miscellaneous and click debug device detection with your device attached and post the output as a ticket on the calibre bug tracker[138].

My device is non-standard or unusual. What can I do to connect to it?

In addition to the *Connect to Folder* function found under the Connect/Share button, calibre provides a `User Defined` device plugin that can be used to connect to any USB device that shows up as a disk drive in your operating system. Note: on windows, the device must have a drive letter for calibre to use it. See the device plugin `Preferences -> Plugins -> Device Plugins -> User Defined` and `Preferences -> Miscellaneous -> Get information to setup the user defined device` for more information. Note that if you are using the user defined plugin for a device normally detected by a builtin calibre plugin, you must disable the builtin plugin first, so that your user defined plugin is used instead.

How does calibre manage collections on my SONY reader?

When calibre connects with the reader, it retrieves all collections for the books on the reader. The collections of which books are members are shown on the device view.

[135] http://calibre-ebook.com/bugs
[136] http://www.mobileread.com/forums/showthread.php?t=118559
[137] http://calibre-ebook.com/download
[138] http://bugs.calibre-ebook.com

When you send a book to the reader, calibre will add the book to collections based on the metadata for that book. By default, collections are created from tags and series. You can control what metadata is used by going to *Preferences->Advanced->Plugins->Device Interface plugins* and customizing the SONY device interface plugin. If you remove all values, calibre will not add the book to any collection.

Collection management is largely controlled by the 'Metadata management' option found at *Preferences->Import/Export->Sending books to devices*. If set to 'Manual' (the default), managing collections is left to the user; calibre will not delete already existing collections for a book on your reader when you resend the book to the reader, but calibre will add the book to collections if necessary. To ensure that the collections for a book are based only on current calibre metadata, first delete the books from the reader, then resend the books. You can edit collections directly on the device view by double-clicking or right-clicking in the collections column.

If 'Metadata management' is set to 'Only on send', then calibre will manage collections more aggressively. Collections will be built using calibre metadata exclusively. Sending a book to the reader will correct the collections for that book so its collections exactly match the book's metadata, adding and deleting collections as necessary. Editing collections on the device view is not permitted, because collections not in the metadata will be removed automatically.

If 'Metadata management' is set to 'Automatic management', then calibre will update metadata and collections both when the reader is connected and when books are sent. When calibre detects the reader and generates the list of books on the reader, it will send metadata from the library to the reader for all books on the reader that are in the library (On device is True), adding and removing books from collections as indicated by the metadata and device customization. When a book is sent, calibre corrects the metadata for that book, adding and deleting collections. Manual editing of metadata on the device view is not allowed. Note that this option specifies sending metadata, not books. The book files on the reader are not changed.

In summary, choose 'manual management' if you want to manage collections yourself. Collections for a book will never be removed by calibre, but can be removed by you by editing on the device view. Choose 'Only on send' if you want calibre to manage collections when you send a book, adding books to and removing books from collections as needed. Choose 'Automatic management' if you want calibre to keep collections up to date whenever the reader is connected.

If you use multiple installations of calibre to manage your reader, then option 'Automatic management' may not be what you want. Connecting the reader to one library will reset the metadata to what is in that library. Connecting to the other library will reset the metadata to what is in that other library. Metadata in books found in both libraries will be flopped back and forth.

Can I use both calibre and the SONY software to manage my reader?

Yes, you can use both, provided you do not run them at the same time. That is, you should use the following sequence: Connect reader->Use one of the programs->Disconnect reader. Reconnect reader->Use the other program->disconnect reader.

The underlying reason is that the Reader uses a single file to keep track of 'meta' information, such as collections, and this is written to by both calibre and the Sony software when either updates something on the Reader. The file will be saved when the Reader is (safely) disconnected, so using one or the other is safe if there's a disconnection between them, but if you're not the type to remember this, then the simple answer is to stick to one or the other for the transfer and just export/import from/to the other via the computers hard disk.

If you do need to reset your metadata due to problems caused by using both at the same time, then just delete the media.xml file on the Reader using your PC's file explorer and it will be recreated after disconnection.

With recent reader iterations, SONY, in all its wisdom has decided to try to force you to use their software. If you install it, it auto-launches whenever you connect the reader. If you don't want to uninstall it altogether, there are a couple of tricks you can use. The simplest is to simply re-name the executable file that launches the library program. More detail in the forums[139].

[139] http://www.mobileread.com/forums/showthread.php?t=65809

How do I use calibre with my iPad/iPhone/iPod touch?

Over the air The easiest way to browse your calibre collection on your Apple device (iPad/iPhone/iPod) is by using the calibre content server, which makes your collection available over the net. First perform the following steps in calibre

- Set the Preferred Output Format in calibre to EPUB (The output format can be set under *Preferences->Interface->Behavior*)
- Set the output profile to iPad (this will work for iPhone/iPods as well), under *Preferences->Conversion->Common Options->Page Setup*
- Convert the books you want to read on your iDevice to EPUB format by selecting them and clicking the Convert button.
- Turn on the Content Server by clicking the *Connect/Share* button and leave calibre running. You can also tell calibre to automatically start the content server via *Preferences->Sharing over the net*.

There are many apps for your iDevice that can connect to calibre. Here we describe using two of them, iBooks and Stanza.

Using Stanza You should be able to access your books on your iPhone by opening Stanza. Go to "Get Books" and then click the "Shared" tab. Under Shared you will see an entry "Books in calibre". If you don't, make sure your iPad/iPhone is connected using the WiFi network in your house, not 3G. If the calibre catalog is still not detected in Stanza, you can add it manually in Stanza. To do this, click the "Shared" tab, then click the "Edit" button and then click "Add book source" to add a new book source. In the Add Book Source screen enter whatever name you like and in the URL field, enter the following:

```
http://192.168.1.2:8080/
```

Replace `192.168.1.2` with the local IP address of the computer running calibre. If you have changed the port the calibre content server is running on, you will have to change `8080` as well to the new port. The local IP address is the IP address you computer is assigned on your home network. A quick Google search will tell you how to find out your local IP address. Now click "Save" and you are done.

If you get timeout errors while browsing the calibre catalog in Stanza, try increasing the connection timeout value in the stanza settings. Go to Info->Settings and increase the value of Download Timeout.

Using iBooks Start the Safari browser and type in the IP address and port of the computer running the calibre server, like this:

```
http://192.168.1.2:8080/
```

Replace `192.168.1.2` with the local IP address of the computer running calibre. If you have changed the port the calibre content server is running on, you will have to change `8080` as well to the new port. The local IP address is the IP address you computer is assigned on your home network. A quick Google search will tell you how to find out your local IP address.

You will see a list of books in Safari, just click on the epub link for whichever book you want to read, Safari will then prompt you to open it with iBooks.

With the USB cable + iTunes Use the 'Connect to iTunes' method in the 'Getting started' instructions in Calibre + Apple iDevices: Start here[140].

This method only works on Windows XP and higher, and OS X 10.5 and higher. Linux is not supported (iTunes is not available in linux) and OS X 10.4 is not supported.

[140] http://www.mobileread.com/forums/showthread.php?t=118559

How do I use calibre with my Android phone/tablet or Kindle Fire HD?

There are two ways that you can connect your Android device to calibre. Using a USB cable – or wirelessly, over the air. The first step to using an Android device is installing an ebook reading application on it. There are many free and paid ebook reading applications for Android: Some examples (in no particular order): FBReader[141], Moon+[142], Mantano[143], Aldiko[144], Kindle[145].

Using a USB cable Simply plug your device into the computer with a USB cable. calibre should automatically detect the device and then you can transfer books to it by clicking the Send to Device button. calibre does not have support for every single android device out there, so if your device is not automatically detected, follow the instructions at *How can I help get my device supported in calibre?* (page 324) to get your device supported in calibre.

Note: With newer Android devices, the USB connection is only supported on Windows Vista and newer and Linux. If you are on Windows XP or OS X, you should use one of the wireless connection methods.

Over the air The easiest way to transfer books wirelessly to your Android device is to use the Calibre Companion[146] Android app. This app is maintained by a core calibre developer and allows calibre to connect to your Android device wirelessly, just as though you plugged in the device with a USB cable. You can browse files on the device in calibre and use the *Send to device* button to transfer files to your device wirelessly.

calibre also has a builtin web server, the *Content Server*. You can browse your calibre collection on your Android device is by using the calibre content server, which makes your collection available over the net. First perform the following steps in calibre

- Set the *Preferred Output Format* in calibre to EPUB for normal Android devices or MOBI for Kindles (The output format can be set under *Preferences->Interface->Behavior*)
- Convert the books you want to read on your device to EPUB/MOBI format by selecting them and clicking the Convert button.
- Turn on the Content Server in calibre's preferences and leave calibre running.

Now on your Android device, open the browser and browse to

> http://192.168.1.2:8080/

Replace `192.168.1.2` with the local IP address of the computer running calibre. If your local network supports the use of computer names, you can replace the IP address with the network name of the computer. If you have changed the port the calibre content server is running on, you will have to change `8080` as well to the new port.

The local IP address is the IP address you computer is assigned on your home network. A quick Google search will tell you how to find out your local IP address. You can now browse your book collection and download books from calibre to your device to open with whatever ebook reading software you have on your android device.

Some reading programs support browsing the Calibre library directly. For example, in Aldiko, click My Catalogs, then + to add a catalog, then give the catalog a title such as "Calibre" and provide the URL listed above. You can now browse the Calibre library and download directly into the reading software.

[141] https://play.google.com/store/apps/details?id=org.geometerplus.zlibrary.ui.android&hl=en
[142] https://play.google.com/store/apps/details?id=com.flyersoft.moonreader&hl=en
[143] https://play.google.com/store/apps/details?id=com.mantano.reader.android.lite&hl=en
[144] https://play.google.com/store/apps/details?id=com.aldiko.android&hl=en
[145] https://play.google.com/store/apps/details?id=com.amazon.kindle&feature=related_apps
[146] http://www.multipie.co.uk/calibre-companion/

Can I access my calibre books using the web browser in my Kindle or other reading device?

calibre has a *Content Server* that exports the books in calibre as a web page. You can turn it on under *Preferences->Network->Sharing over the net*. Then just point the web browser on your device to the computer running the Content Server and you will be able to browse your book collection. For example, if the computer running the server has IP address 63.45.128.5, in the browser, you would type:

```
http://63.45.128.5:8080
```

Some devices, like the Kindle (1/2/DX), do not allow you to access port 8080 (the default port on which the content server runs. In that case, change the port in the calibre Preferences to 80. (On some operating systems, you may not be able to run the server on a port number less than 1024 because of security settings. In this case the simplest solution is to adjust your router to forward requests on port 80 to port 8080).

I get the error message "Failed to start content server: Port 8080 not free on '0.0.0.0"'?

The most likely cause of this is your antivirus program. Try temporarily disabling it and see if it does the trick.

I cannot send emails using calibre?

Because of the large amount of spam in email, sending email can be tricky, as different mail servers use different strategies to block email. The most common problem is if you are sending email directly (without a mail relay) in calibre. Many servers (for example, Amazon) block email that does not come from a well known relay. The most robust way to setup email sending in calibre is to do the following:

- Create a free GMail account at Google[147].
- Goto Preferences->Email in calibre and click the "Use Gmail" button and fill in the information asked for.
- calibre will then use GMail to send the mail.
- If you are sending to your Kindle, remember to update the email preferences on your Amazon Kindle page to allow email sent from your GMail email address.

Even after doing this, you may have problems. One common source of problems is that some poorly designed antivirus programs block calibre from opening a connection to send email. Try adding an exclusion for calibre in your antivirus program.

Note: Google can disable your account if you use it to send large amounts of email. So, when using GMail to send mail calibre automatically restricts itself to sending one book every five minutes. If you don't mind risking your account being blocked you can reduce this wait interval by going to Preferences->Tweaks in calibre.

Why is my device not detected in linux?

calibre needs your linux kernel to have been setup correctly to detect devices. If your devices are not detected, perform the following tests:

```
grep SYSFS_DEPRECATED /boot/config-`uname -r`
```

You should see something like `CONFIG_SYSFS_DEPRECATED_V2 is not set`. Also,

[147] http://www.gmail.com

```
grep CONFIG_SCSI_MULTI_LUN /boot/config-`uname -r`
```

must return `CONFIG_SCSI_MULTI_LUN=y`. If you don't see either, you have to recompile your kernel with the correct settings.

My device is getting mounted read-only in linux, so calibre cannot connect to it?

Linux kernels mount devices read-only when their filesystems have errors. You can repair the filesystem with:

```
sudo fsck.vfat -y /dev/sdc
```

Replace /dev/sdc with the path to the device node of your device. You can find the device node of your device, which will always be under /dev by examining the output of:

```
mount
```

Why does calibre not support collections on the Kindle or shelves on the Nook?

Neither the Kindle nor the Nook provide any way to manipulate collections over a USB connection. If you really care about using collections, I would urge you to sell your Kindle/Nook and get a SONY. Only SONY seems to understand that life is too short to be entering collections one by one on an e-ink screen :)

Note that in the case of the Kindle, there is a way to manipulate collections via USB, but it requires that the Kindle be rebooted *every time* it is disconnected from the computer, for the changes to the collections to be recognized. As such, it is unlikely that any calibre developers will ever feel motivated enough to support it. There is however, a calibre plugin that allows you to create collections on your Kindle from the calibre metadata. It is available from here[148].

Note: Amazon have removed the ability to manipulate collections completely in their newer models, like the Kindle Touch and Kindle Fire, making even the above plugin useless. If you really want the ability to manage collections on your Kindle via a USB connection, we encourage you to complain to Amazon about it, or get a reader where this is supported, like the SONY or Kobo Readers.

I am getting an error when I try to use calibre with my Kobo Touch/Glo/etc.?

The Kobo has very buggy firmware. Connecting to it has been known to fail at random. Certain combinations of motherboard, USB ports/cables/hubs can exacerbate this tendency to fail. If you are getting an error when connecting to your touch with calibre try the following, each of which has solved the problem for *some* calibre users.

- Connect the Kobo directly to your computer, not via USB Hub
- Try a different USB cable and a different USB port on your computer
- Try a different computer (preferably an older model)
- Try upgrading the firmware on your Kobo Touch to the latest
- Try resetting the Kobo (sometimes this cures the problem for a little while, but then it re-appears, in which case you have to reset again and again)
- Try only putting one or two books onto the Kobo at a time and do not keep large collections on the Kobo

[148] http://www.mobileread.com/forums/showthread.php?t=118635

Library Management

Contents

- What formats does calibre read metadata from? (page 331)
- Where are the book files stored? (page 331)
- How does calibre manage author names and sorting? (page 331)
- Why doesn't calibre let me store books in my own directory structure? (page 332)
- Why doesn't calibre have a column for foo? (page 333)
- Can I have a column showing the formats or the ISBN? (page 333)
- How do I move my calibre library from one computer to another? (page 333)
- The list of books in calibre is blank! (page 334)
- I am getting errors with my calibre library on a networked drive/NAS? (page 334)

What formats does calibre read metadata from?

calibre reads metadata from the following formats: CHM, LRF, PDF, LIT, RTF, OPF, MOBI, PRC, EPUB, FB2, IMP, RB, HTML. In addition it can write metadata to: LRF, RTF, OPF, EPUB, PDF, MOBI

Where are the book files stored?

When you first run calibre, it will ask you for a folder in which to store your books. Whenever you add a book to calibre, it will copy the book into that folder. Books in the folder are nicely arranged into sub-folders by Author and Title. Note that the contents of this folder are automatically managed by calibre, **do not** add any files/folders manually to this folder, as they may be automatically deleted. If you want to add a file associated to a particular book, use the top right area of *Edit metadata* dialog to do so. Then, calibre will automatically put that file into the correct folder and move it around when the title/author changes.

Metadata about the books is stored in the file `metadata.db` at the top level of the library folder This file is is a sqlite database. When backing up your library make sure you copy the entire folder and all its sub-folders.

The library folder and all it's contents make up what is called a calibre library. You can have multiple such libraries. To manage the libraries, click the calibre icon on the toolbar. You can create new libraries, remove/rename existing ones and switch between libraries easily.

You can copy or move books between different libraries (once you have more than one library setup) by right clicking on a book and selecting the *Copy to library* action.

How does calibre manage author names and sorting?

Author names are complex, especially across cultures. calibre has a very flexible strategy for managing author names. The first thing to understand is that books and authors are separate entities in calibre. A book can have more than one author, and an author can have more than one book. You can manage the authors of a book by the edit metadata dialog. You can manage individual authors by right clicking on the author in the Tag Browser on the left of the main calibre screen and selecting *Manage authors*. Using this dialog you can change the name of an author and also how that name is sorted. This will automatically change the name of the author in all the books of that author. When a book has multiple authors, separate their names using the & character.

Now coming to author name sorting:

- When a new author is added to calibre (this happens whenever a book by a new author is added), calibre automatically computes a sort string for both the book and the author.

- Authors in the Tag Browser are sorted by the sort value for the **authors**. Remember that this is different from the Author sort field for a book.
- By default, this sort algorithm assumes that the author name is in `First name Last name` format and generates a `Last name, First name` sort value.
- You can change this algorithm by going to Preferences->Tweaks and setting the *author_sort_copy_method* tweak.
- You can force calibre to recalculate the author sort values for every author by right clicking on any author and selecting *Manage authors*, then pushing the *Recalculate all author sort values* button. Do this after you have set the author_sort_copy_method tweak to what you want.
- You can force calibre to recalculate the author sort values for all books by using the bulk metadata edit dialog (select all books and click edit metadata, check the *Automatically set author sort* checkbox, then press OK.)
- When recalculating the author sort values for books, calibre uses the author sort values for each individual author. Therefore, ensure that the individual author sort values are correct before recalculating the books' author sort values.
- You can control whether the Tag Browser display authors using their names or their sort values by setting the *categories_use_field_for_author_name* tweak in Preferences->Tweaks

Note that you can set an individual author's sort value to whatever you want using *Manage authors*. This is useful when dealing with names that calibre will not get right, such as complex multi-part names like Miguel de Cervantes Saavedra or when dealing with Asian names like Sun Tzu.

With all this flexibility, it is possible to have calibre manage your author names however you like. For example, one common req

- Set the `author_sort_copy_method` tweak to `copy` as described above.
- Restart calibre. Do not change any book metadata before doing the remaining steps.
- Change all author names to LN, FN using the Manage authors dialog.
- After you have changed all the authors, press the *Recalculate all author sort values* button.
- Press OK, at which point calibre will change the authors in all your books. This can take a while.

Note:

When changing from FN LN to LN, FN, it is often the case that the values in author_sort are already in LN, FN format. If this i

- set the `author_sort_copy_method` tweak to `copy` as described above.
- restart calibre. Do not change any book metadata before doing the remaining steps.
- open the Manage authors dialog. Press the `copy all author sort values to author` button.
- Check through the authors to be sure you are happy. You can still press Cancel to abandon the changes. Once you press OK, there is no undo.
- Press OK, at which point calibre will change the authors in all your books. This can take a while.

Why doesn't calibre let me store books in my own directory structure?

The whole point of calibre's library management features is that they provide a search and sort based interface for locating books that is *much* more efficient than any possible directory scheme you could come up with for your collection. Indeed, once you become comfortable using calibre's interface to find, sort and browse your collection,

you wont ever feel the need to hunt through the files on your disk to find a book again. By managing books in its own directory structure of Author -> Title -> Book files, calibre is able to achieve a high level of reliability and standardization. To illustrate why a search/tagging based interface is superior to folders, consider the following. Suppose your book collection is nicely sorted into folders with the following scheme:

```
Genre -> Author -> Series -> ReadStatus
```

Now this makes it very easy to find for example all science fiction books by Isaac Asimov in the Foundation series. But suppose you want to find all unread science fiction books. There's no easy way to do this with this folder scheme, you would instead need a folder scheme that looks like:

```
ReadStatus -> Genre -> Author -> Series
```

In calibre, you would instead use tags to mark genre and read status and then just use a simple search query like `tag:scifi and not tag:read`. calibre even has a nice graphical interface, so you don't need to learn its search language instead you can just click on tags to include or exclude them from the search.

To those of you that claim that you need access to the filesystem to so that you can have access to your books over the network, calibre has an excellent content server that gives you access to your calibre library over the net.

If you are worried that someday calibre will cease to be developed, leaving all your books marooned in its folder structure, explore the powerful "Save to Disk" feature in calibre that lets you export all your files into a folder structure of arbitrary complexity based on their metadata.

Finally, the reason there are numbers at the end of every title folder, is for *robustness*. That number is the id number of the book record in the calibre database. The presence of the number allows you to have multiple records with the same title and author names. It is also part of what allows calibre to magically regenerate the database with all metadata if the database file gets corrupted. Given that calibre's mission is to get you to stop storing metadata in filenames and stop using the filesystem to find things, the increased robustness afforded by the id numbers is well worth the uglier folder names.

If you are still not convinced, then I'm afraid calibre is not for you. Look elsewhere for your book cataloguing needs. Just so we're clear, **this is not going to change**. Kindly do not contact us in an attempt to get us to change this.

Why doesn't calibre have a column for foo?

calibre is designed to have columns for the most frequently and widely used fields. In addition, you can add any columns you like. Columns can be added via *Preferences->Interface->Add your own columns*. Watch the tutorial UI Power tips[149] to learn how to create your own columns.

You can also create "virtual columns" that contain combinations of the metadata from other columns. In the add column dialog use the *Quick create* links to easily create columns to show the book ISBN, formats or the time the book was last modified. For more details, see *The calibre template language* (page 375).

Can I have a column showing the formats or the ISBN?

Yes, you can. Follow the instructions in the answer above for adding custom columns.

How do I move my calibre library from one computer to another?

Simply copy the calibre library folder from the old to the new computer. You can find out what the library folder is by clicking the calibre icon in the toolbar. The very first item is the path to the library folder. Now on the new computer, start calibre for the first time. It will run the Welcome Wizard asking you for the location of the calibre library. Point it to the previously copied folder. If the computer you are transferring to already has a calibre installation, then the

[149] http://calibre-ebook.com/demo#tutorials

Welcome wizard wont run. In that case, right-click the calibre icon in the tooolbar and point it to the newly copied directory. You will now have two calibre libraries on your computer and you can switch between them by clicking the calibre icon on the toolbar. Transferring your library in this manner preserver all your metadata, tags, custom columns, etc.

Note that if you are transferring between different types of computers (for example Windows to OS X) then after doing the above you should also right-click the calibre icon on the tool bar, select Library Maintenance and run the Check Library action. It will warn you about any problems in your library, which you should fix by hand.

Note: A calibre library is just a folder which contains all the book files and their metadata. All the metadata is stored in a single file called metadata.db, in the top level folder. If this file gets corrupted, you may see an empty list of books in calibre. In this case you can ask calibre to restore your books by doing a right-click on the calibre icon in the toolbar and selecting Library Maintenance->Restore Library.

The list of books in calibre is blank!

In order to understand why that happened, you have to understand what a calibre library is. At the most basic level, a calibre library is just a folder. Whenever you add a book to calibre, that book's files are copied into this folder (arranged into sub folders by author and title). Inside the calibre library folder, at the top level, you will see a file called metadata.db. This file is where calibre stores the metadata like title/author/rating/tags etc. for *every* book in your calibre library. The list of books that calibre displays is created by reading the contents of this metadata.db file.

There can be two reasons why calibre is showing a empty list of books:

- Your calibre library folder changed its location. This can happen if it was on an external disk and the drive letter for that disk changed. Or if you accidentally moved the folder. In this case, calibre cannot find its library and so starts up with an empty library instead. To remedy this, do a right-click on the calibre icon in the calibre toolbar (it will say 0 books underneath it) and select Switch/create library. Click the little blue icon to select the new location of your calibre library and click OK.
- Your metadata.db file was deleted/corrupted. In this case, you can ask calibre to rebuild the metadata.db from its backups. Right click the calibre icon in the calibre toolbar (it will say 0 books underneath it) and select Library maintenance->Restore database. calibre will automatically rebuild metadata.db.

I am getting errors with my calibre library on a networked drive/NAS?

Do not put your calibre library on a networked drive.

A filesystem is a complex beast. Most network filesystems lack various filesystem features that calibre uses. Some dont support file locking, some dont support hardlinking, some are just flaky. Additionally, calibre is a single user application, if you accidentally run two copies of calibre on the same networked library, bad things will happen. Finally, different OSes impose different limitations on filesystems, so if you share your networked drive across OSes, once again, bad things *will happen*.

Consider using the calibre Content Server to make your books available on other computers. Run calibre on a single computer and access it via the Content Server or a Remote Desktop solution.

If you must share the actual library, use a file syncing tool like DropBox or rsync or Microsoft SkyDrive instead of a networked drive. Even with these tools there is danger of data corruption/loss, so only do this if you are willing to live with that risk. In particular, be aware that **Google Drive** is incompatible with calibre, if you put your calibre library in Google Drive, you *will* suffer data loss. See this thread[150] for details.

[150] http://www.mobileread.com/forums/showthread.php?t=205581

Content From The Web

Contents

- I obtained a recipe for a news site as a .py file from somewhere, how do I use it? (page 335)
- I want calibre to download news from my favorite news website. (page 335)
- Can I use web2disk to download an arbitrary website? (page 335)

I obtained a recipe for a news site as a .py file from somewhere, how do I use it?

Start the *Add custom news sources* dialog (from the *Fetch news* menu) and click the *Switch to advanced mode* button. Delete everything in the box with the recipe source code and copy paste the contents of your .py file into the box. Click *Add/update recipe*.

I want calibre to download news from my favorite news website.

If you are reasonably proficient with computers, you can teach calibre to download news from any website of your choosing. To learn how to do this see *Adding your favorite news website* (page 341).

Otherwise, you can request a particular news site by posting in the calibre Recipes forum[151].

Can I use web2disk to download an arbitrary website?

```
web2disk http://mywebsite.com
```

Miscellaneous

Contents

- Why the name calibre? (page 336)
- Why does calibre show only some of my fonts on OS X? (page 336)
- calibre is not starting on Windows? (page 336)
- calibre freezes/crashes occasionally? (page 337)
- calibre is not starting on OS X? (page 337)
- I downloaded the installer, but it is not working? (page 338)
- My antivirus program claims calibre is a virus/trojan? (page 338)
- How do I backup calibre? (page 338)
- How do I use purchased EPUB books with calibre (or what do I do with .acsm files)? (page 338)
- I am getting a "Permission Denied" error? (page 339)
- Can I have the comment metadata show up on my reader? (page 339)
- How do I get calibre to use my HTTP proxy? (page 339)
- I want some feature added to calibre. What can I do? (page 339)
- Why doesn't calibre have an automatic update? (page 340)
- How is calibre licensed? (page 340)
- How do I run calibre from my USB stick? (page 340)
- How do I run parts of calibre like news download and the content server on my own linux server? (page 340)

[151] http://www.mobileread.com/forums/forumdisplay.php?f=228

Why the name calibre?

Take your pick:

- Convertor And LIBRary for Ebooks
- A high *calibre* product
- A tribute to the SONY Librie which was the first e-ink based ebook reader
- My wife chose it ;-)

calibre is pronounced as cal-i-ber *not* ca-li-bre. If you're wondering, calibre is the British/commonwealth spelling for caliber. Being Indian, that's the natural spelling for me.

Why does calibre show only some of my fonts on OS X?

calibre embeds fonts in ebook files it creates. Ebook files support embedding only TrueType (.ttf) fonts. Most fonts on OS X systems are in .dfont format, thus they cannot be embedded. calibre shows only TrueType fonts found on your system. You can obtain many TrueType fonts on the web. Simply download the .ttf files and add them to the Library/Fonts directory in your home directory.

calibre is not starting on Windows?

There can be several causes for this:

- If you get an error about calibre not being able to open a file because it is in use by another program, do the following:
  - Uninstall calibre
  - Reboot your computer
  - Re-install calibre. But do not start calibre from the installation wizard.
  - Temporarily disable your antivirus program (disconnect from the Internet before doing so, to be safe)
  - Look inside the folder you chose for your calibre library. If you see a file named metadata.db, delete it.
  - Start calibre
  - From now on you should be able to start calibre normally.
- If you get an error about a Python function terminating unexpectedly after upgrading calibre, first uninstall calibre, then delete the folders (if they exists) `C:\Program Files\Calibre` and `C:\Program Files\Calibre2`. Now re-install and you should be fine.
- If you get an error in the welcome wizard on an initial run of calibre, try choosing a folder like `C:\library` as the calibre library (calibre sometimes has trouble with library locations if the path contains non-English characters, or only numbers, etc.)
- Try running it as Administrator (Right click on the icon and select "Run as Administrator")
- **Windows Vista**: If the folder `C:\Users\Your User Name\AppData\Local\VirtualStore\Program Files\calibre` exists, delete it. Uninstall calibre. Reboot. Re-install.
- **Any windows version**: Try disabling any antivirus program you have running and see if that fixes it. Also try disabling any firewall software that prevents connections to the local computer.

If it still wont launch, start a command prompt (press the windows key and R; then type **cmd.exe** in the Run dialog that appears). At the command prompt type the following command and press Enter:

```
calibre-debug -g
```

Post any output you see in a help message on the Forum[152].

calibre freezes/crashes occasionally?

There are several possible things I know of, that can cause this:

- You recently connected an external monitor or TV to your computer. In this case, whenever calibre opens a new window like the edit metadata window or the conversion dialog, it appears on the second monitor where you dont notice it and so you think calibre has frozen. Disconnect your second monitor and restart calibre.
- If you use RoboForm, it is known to cause calibre to crash. Add calibre to the blacklist of programs inside RoboForm to fix this. Or uninstall RoboForm.
- The Logitech SetPoint Settings application causes random crashes in calibre when it is open. Close it before starting calibre.
- Constant Guard Protection by Xfinity causes crashes in calibre. You have to manually allow calibre in it or uninstall Constant Guard Protection.
- Spybot - Search & Destroy blocks calibre from accessing its temporary files breaking viewing and converting of books.
- You are using a Wacom branded USB mouse. There is an incompatibility between Wacom mice and the graphics toolkit calibre uses. Try using a non-Wacom mouse.
- On some 64 bit versions of Windows there are security software/settings that prevent 64-bit calibre from working properly. If you are using the 64-bit version of calibre try switching to the 32-bit version.

If none of the above apply to you, then there is some other program on your computer that is interfering with calibre. First reboot your computer in safe mode, to have as few running programs as possible, and see if the crashes still happen. If they do not, then you know it is some program causing the problem. The most likely such culprit is a program that modifies other programs' behavior, such as an antivirus, a device driver, something like RoboForm (an automatic form filling app) or an assistive technology like Voice Control or a Screen Reader.

The only way to find the culprit is to eliminate the programs one by one and see which one is causing the issue. Basically, stop a program, run calibre, check for crashes. If they still happen, stop another program and repeat.

calibre is not starting on OS X?

One common cause of failures on OS X is the use of accessibility technologies that are incompatible with the graphics toolkit calibre uses. Try turning off VoiceOver if you have it on. Also go to System Preferences->System->Universal Access and turn off the setting for enabling access for assistive devices in all the tabs.

You can obtain debug output about why calibre is not starting by running *Console.app*. Debug output will be printed to it. If the debug output contains a line that looks like:

```
Qt: internal: -108: Error ATSUMeasureTextImage text/qfontengine_mac.mm
```

then the problem is probably a corrupted font cache. You can clear the cache by following these instructions[153]. If that doesn't solve it, look for a corrupted font file on your system, in ~/Library/Fonts or the like. An easy way to check for corrupted fonts in OS X is to start the "Font Book" application, select all fonts and then in the File menu, choose "Validate fonts".

[152] http://www.mobileread.com/forums/forumdisplay.php?f=166
[153] http://www.macworld.com/article/139383/2009/03/fontcacheclear.html

I downloaded the installer, but it is not working?

Downloading from the Internet can sometimes result in a corrupted download. If the calibre installer you downloaded is not opening, try downloading it again. If re-downloading it does not work, download it from an alternate location[154]. If the installer still doesn't work, then something on your computer is preventing it from running.

- Try temporarily disabling your antivirus program (Microsoft Security Essentials, or Kaspersky or Norton or McAfee or whatever). This is most likely the culprit if the upgrade process is hanging in the middle.
- Try rebooting your computer and running a registry cleaner like Wise registry cleaner[155].
- Try downloading the installer with an alternate browser. For example if you are using Internet Explorer, try using Firefox or Chrome instead.

If you still cannot get the installer to work and you are on windows, you can use the calibre portable install[156], which does not need an installer (it is just a zip file).

My antivirus program claims calibre is a virus/trojan?

The first thing to check is that you are downloading calibre from the official website: http://calibre-ebook.com/download. calibre is a very popular program and unscrupulous people try to setup websites offering it for download to fool the unwary.

If you have the official download and your antivirus program is still claiming calibre is a virus, then, your antivirus program is wrong. Antivirus programs use heuristics, patterns of code that "look suspicious" to detect viruses. It's rather like racial profiling. calibre is a completely open source product. You can actually browse the source code yourself (or hire someone to do it for you) to verify that it is not a virus. Please report the false identification to whatever company you buy your antivirus software from. If the antivirus program is preventing you from downloading/installing calibre, disable it temporarily, install calibre and then re-enable it.

How do I backup calibre?

The most important thing to backup is the calibre library folder, that contains all your books and metadata. This is the folder you chose for your calibre library when you ran calibre for the first time. You can get the path to the library folder by clicking the calibre icon on the main toolbar. You must backup this complete folder with all its files and sub-folders.

You can switch calibre to using a backed up library folder by simply clicking the calibre icon on the toolbar and choosing your backup library folder. A backed up library folder backs up your custom columns and saved searches as well as all your books and metadata.

If you want to backup the calibre configuration/plugins, you have to backup the config directory. You can find this config directory via *Preferences->Miscellaneous*. Note that restoring configuration directories is not officially supported, but should work in most cases. Just copy the contents of the backup directory into the current configuration directory to restore.

How do I use purchased EPUB books with calibre (or what do I do with .acsm files)?

Most purchased EPUB books have DRM[157]. This prevents calibre from opening them. You can still use calibre to store and transfer them to your ebook reader. First, you must authorize your reader on a windows machine with Adobe

[154] http://sourceforge.net/projects/calibre/files/
[155] http://www.wisecleaner.com
[156] http://calibre-ebook.com/download_portable
[157] http://drmfree.calibre-ebook.com/about#drm

Digital Editions. Once this is done, EPUB books transferred with calibre will work fine on your reader. When you purchase an epub book from a website, you will get an ".acsm" file. This file should be opened with Adobe Digital Editions, which will then download the actual ".epub" ebook. The ebook file will be stored in the folder "My Digital Editions", from where you can add it to calibre.

I am getting a "Permission Denied" error?

A permission denied error can occur because of many possible reasons, none of them having anything to do with calibre.

- You can get permission denied errors if you are using an SD card with write protect enabled.
- If you, or some program you used changed the file permissions of the files in question to read only.
- If there is a filesystem error on the device which caused your operating system to mount the filesystem in read only mode or mark a particular file as read only pending recovery.
- If the files have their owner set to a user other than you.
- If your file is open in another program.
- If the file resides on a device, you may have reached the limit of a maximum of 256 files in the root of the device. In this case you need to reformat the device/sd card referered to in the error message with a FAT32 filesystem, or delete some files from the SD card/device memory.

You will need to fix the underlying cause of the permissions error before resuming to use calibre. Read the error message carefully, see what file it points to and fix the permissions on that file.

Can I have the comment metadata show up on my reader?

Most readers do not support this. You should complain to the manufacturer about it and hopefully if enough people complain, things will change. In the meantime, you can insert the metadata, including comments into a "Jacket page" at the start of the ebook, by using the option to "Insert metadata as page at start of book" during conversion. The option is found in the *Structure Detection* section of the conversion settings. Note that for this to have effect you have to *convert* the book. If your book is already in a format that does not need conversion, you can convert from that format to the same format.

Another alternative is to create a catalog in ebook form containing a listing of all the books in your calibre library, with their metadata. Click-and-hold the convert button to access the catalog creation tool. And before you ask, no you cannot have the catalog "link directly to" books on your reader.

How do I get calibre to use my HTTP proxy?

By default, calibre uses whatever proxy settings are set in your OS. Sometimes these are incorrect, for example, on windows if you don't use Internet Explorer then the proxy settings may not be up to date. You can tell calibre to use a particular proxy server by setting the http_proxy environment variable. The format of the variable is: http://username:password@servername you should ask your network admin to give you the correct value for this variable. Note that calibre only supports HTTP proxies not SOCKS proxies. You can see the current proxies used by calibre in Preferences->Miscellaneous.

I want some feature added to calibre. What can I do?

You have two choices:

1. Create a patch by hacking on calibre and send it to me for review and inclusion. See Development[158].
2. Open a bug requesting the feature[159] . Remember that while you may think your feature request is extremely important/essential, calibre developers might not agree. Fortunately, calibre is open source, which means you always have the option of implementing your feature yourself, or hiring someone to do it for you. Furthermore, calibre has a comprehensive plugin architecture, so you might be able to develop your feature as a plugin, see *Writing your own plugins to extend calibre's functionality* (page 423).

Why doesn't calibre have an automatic update?

For many reasons:

- *There is no need to update every week*. If you are happy with how calibre works turn off the update notification and be on your merry way. Check back to see if you want to update once a year or so.
- Pre downloading the updates for all users in the background would require about 80TB of bandwidth *every week*. That costs thousands of dollars a month. And calibre is currently growing at 300,000 new users every month.
- If I implement a dialog that downloads the update and launches it, instead of going to the website as it does now, that would save the most ardent calibre updater, *at most five clicks a week*. There are far higher priority things to do in calibre development.
- If you really, really hate downloading calibre every week but still want to be up to the latest, I encourage you to run from source, which makes updating trivial. Instructions are *available here* (page 511).

How is calibre licensed?

calibre is licensed under the GNU General Public License v3 (an open source license). This means that you are free to redistribute calibre as long as you make the source code available. So if you want to put calibre on a CD with your product, you must also put the calibre source code on the CD. The source code is available for download from googlecode[160]. You are free to use the results of conversions from calibre however you want. You cannot use code, libraries from calibre in your software without making your software open source. For details, see The GNU GPL v3[161].

How do I run calibre from my USB stick?

A portable version of calibre is available here[162].

How do I run parts of calibre like news download and the content server on my own linux server?

First, you must install calibre onto your linux server. If your server is using a modern linux distro, you should have no problems installing calibre onto it.

Note: If you bought into the notion that a real server must run a decade old version of Debian, then you will have to jump through a few hoops. First, compile a newer version of glibc (>= 2.10) on your server from source. Then get the calibre linux binary tarball from the calibre google code page for your server architecture. Extract it into

[158] http://calibre-ebook.com/get-involved
[159] http://calibre-ebook.com/bugs
[160] http://code.google.com/p/calibre-ebook/downloads/list
[161] http://www.gnu.org/licenses/gpl.html
[162] http://calibre-ebook.com/download_portable

`/opt/calibre`. Put your previously compiled glibc into `/opt/calibre` as `libc.so.6`. You can now run the calibre binaries from `/opt/calibre`.

You can run the calibre server via the command:

```
/opt/calibre/calibre-server --with-library /path/to/the/library/you/want/to/share
```

You can download news and convert it into an ebook with the command:

```
/opt/calibre/ebook-convert "Title of news source.recipe" outputfile.epub
```

If you want to generate MOBI, use outputfile.mobi instead and use `--output-profile kindle`.

You can email downloaded news with the command:

```
/opt/calibre/calibre-smtp
```

I leave figuring out the exact command line as an exercise for the reader.

Finally, you can add downloaded news to the calibre library with:

```
/opt/calibre/calibredb add --with-library /path/to/library outfile.epub
```

Remember to read the command line documentation section of the calibre User Manual to learn more about these, and other commands.

Note: Some parts of calibre require a X server. If you're lucky, nothing you do will fall into this category, if not, you will have to look into using xvfb.

1.18 Tutorials

1.18.1 Tutorials

Here you will find tutorials to get you started using calibre's more advanced features, such as XPath and templates.

Adding your favorite news website

calibre has a powerful, flexible and easy-to-use framework for downloading news from the Internet and converting it into an ebook. The following will show you, by means of examples, how to get news from various websites.

To gain an understanding of how to use the framework, follow the examples in the order listed below:

- Completely automatic fetching (page 342)
  - portfolio.com (page 342)
  - bbc.co.uk (page 344)
- Customizing the fetch process (page 344)
  - Using the print version of bbc.co.uk (page 344)
  - Replacing article styles (page 345)
  - Slicing and dicing (page 346)
  - Real life example (page 357)
- Tips for developing new recipes (page 359)
- Further reading (page 360)
- API documentation (page 360)

Completely automatic fetching

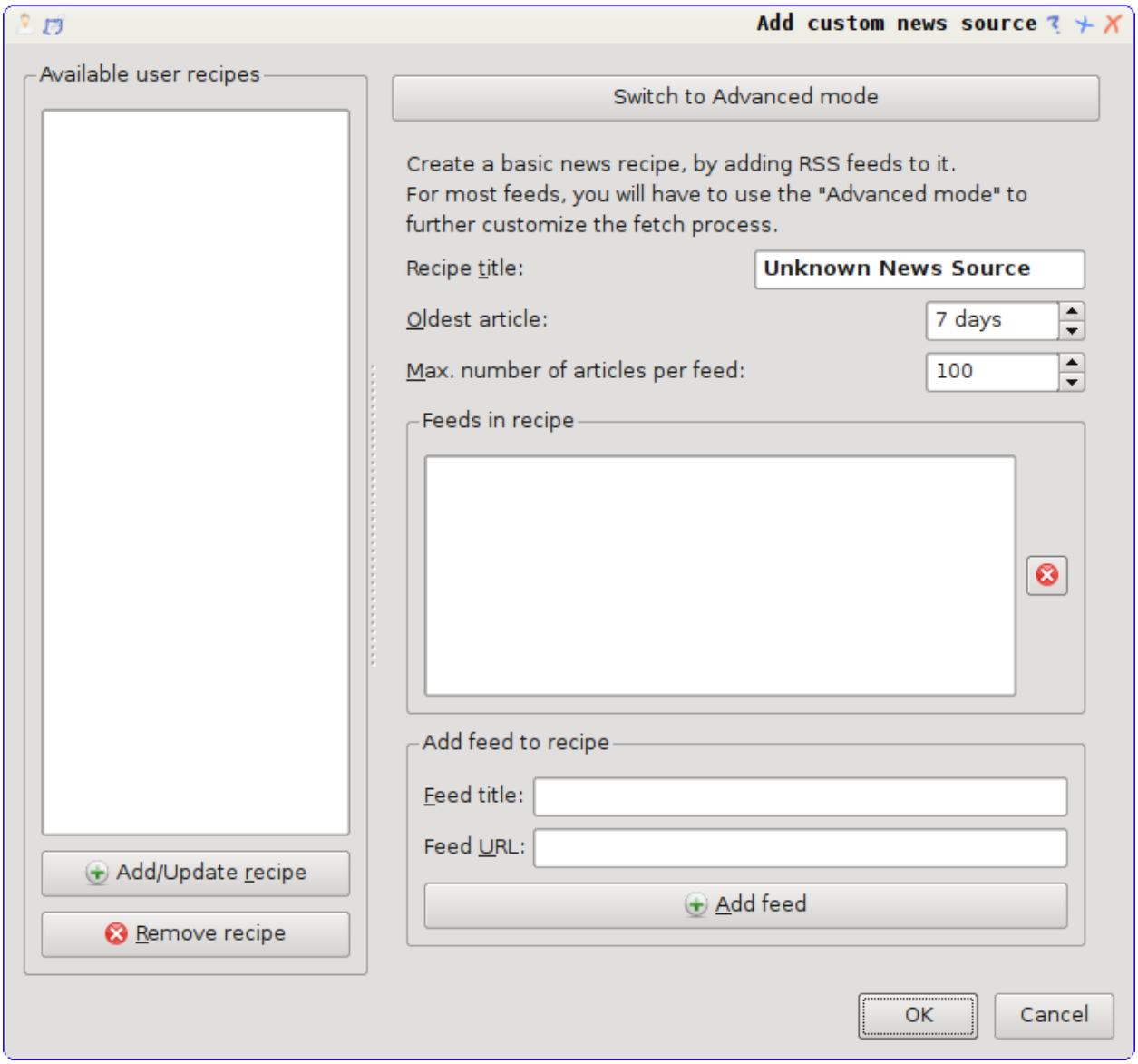
If your news source is simple enough, calibre may well be able to fetch it completely automatically, all you need to do is provide the URL. calibre gathers all the information needed to download a news source into a *recipe*. In order to tell calibre about a news source, you have to create a *recipe* for it. Let's see some examples:

portfolio.com *portfolio.com* is the website for *Condé Nast Portfolio*, a business related magazine. In order to download articles from the magazine and convert them to ebooks, we rely on the *RSS* feeds of portfolio.com. A list of such feeds is available at http://www.portfolio.com/rss/.

Lets pick a couple of feeds that look interesting:

1. Business Travel: http://feeds.portfolio.com/portfolio/businesstravel
2. Tech Observer: http://feeds.portfolio.com/portfolio/thetechobserver

I got the URLs by clicking the little orange RSS icon next to each feed name. To make calibre download the feeds and convert them into an ebook, you should right click the *Fetch news* button and then the *Add a custom news source* menu item. A dialog similar to that shown below should open up.



First enter `Portfolio` into the *Recipe title* field. This will be the title of the ebook that will be created from the articles in the above feeds.

The next two fields (*Oldest article* and *Max. number of articles*) allow you some control over how many articles should be downloaded from each feed, and they are pretty self explanatory.

To add the feeds to the recipe, enter the feed title and the feed URL and click the *Add feed* button. Once you have added both feeds, simply click the *Add/update recipe* button and you're done! Close the dialog.

To test your new *recipe*, click the *Fetch news* button and in the *Custom news sources* sub-menu click *Portfolio*. After a couple of minutes, the newly downloaded Portfolio ebook will appear in the main library view (if you have your reader connected, it will be put onto the reader instead of into the library). Select it and hit the *View* button to read!

The reason this worked so well, with so little effort is that *portfolio.com* provides *full-content RSS* feeds, i.e., the article content is embedded in the feed itself. For most news sources that provide news in this fashion, with *full-content* feeds, you don't need any more effort to convert them to ebooks. Now we will look at a news source that does not provide full content feeds. In such feeds, the full article is a webpage and the feed only contains a link to the webpage with a short summary of the article.

bbc.co.uk Lets try the following two feeds from *The BBC*:

1. News Front Page: http://newsrss.bbc.co.uk/rss/newsonline_world_edition/front_page/rss.xml
2. Science/Nature: http://newsrss.bbc.co.uk/rss/newsonline_world_edition/science/nature/rss.xml

Follow the procedure outlined in *portfolio.com* (page 342) to create a recipe for *The BBC* (using the feeds above). Looking at the downloaded ebook, we see that calibre has done a creditable job of extracting only the content you care about from each article's webpage. However, the extraction process is not perfect. Sometimes it leaves in undesirable content like menus and navigation aids or it removes content that should have been left alone, like article headings. In order, to have perfect content extraction, we will need to customize the fetch process, as described in the next section.

Customizing the fetch process

When you want to perfect the download process, or download content from a particularly complex website, you can avail yourself of all the power and flexibility of the *recipe* framework. In order to do that, in the *Add custom news sources* dialog, simply click the *Switch to Advanced mode* button.

The easiest and often most productive customization is to use the print version of the online articles. The print version typically has much less cruft and translates much more smoothly to an ebook. Let's try to use the print version of the articles from *The BBC*.

Using the print version of bbc.co.uk The first step is to look at the ebook we downloaded previously from *bbc.co.uk* (page 344). At the end of each article, in the ebook is a little blurb telling you where the article was downloaded from. Copy and paste that URL into a browser. Now on the article webpage look for a link that points to the "Printable version". Click it to see the print version of the article. It looks much neater! Now compare the two URLs. For me they were:

> **Article URL** http://news.bbc.co.uk/2/hi/science/nature/7312016.stm
>
> **Print version URL** http://newsvote.bbc.co.uk/mpapps/pagetools/print/news.bbc.co.uk/2/hi/science/nature/7312016.stm

So it looks like to get the print version, we need to prefix every article URL with:

> newsvote.bbc.co.uk/mpapps/pagetools/print/

Now in the *Advanced Mode* of the Custom news sources dialog, you should see something like (remember to select *The BBC* recipe before switching to advanced mode):



You can see that the fields from the *Basic mode* have been translated to python code in a straightforward manner. We need to add instructions to this recipe to use the print version of the articles. All that's needed is to add the following two lines:

```
def print_version(self, url):
    return url.replace('http:///', 'http://newsvote.bbc.co.uk/mpapps/pagetools/print/')
```

This is python, so indentation is important. After you've added the lines, it should look like:

Recipe source code (python)

```
class AdvancedUserRecipe1206418393(BasicNewsRecipe):
    title          = u'The BBC'
    oldest_article = 7
    max_articles_per_feed = 100

    feeds          = [(u'News Front Page', u'http://newsrss.bbc.co.uk/rss/newsonlir

    def print_version(self, url):
        return url.replace('http://', 'http://newsvote.bbc.co.uk/mpapps/pagetools/p
```

In the above, `def print_version(self, url)` defines a *method* that is called by calibre for every article. `url` is the URL of the original article. What `print_version` does is take that url and replace it with the new URL that points to the print version of the article. To learn about python[163] see the tutorial[164].

Now, click the *Add/update recipe* button and your changes will be saved. Re-download the ebook. You should have a much improved ebook. One of the problems with the new version is that the fonts on the print version webpage are too small. This is automatically fixed when converting to an ebook, but even after the fixing process, the font size of the menus and navigation bar to become too large relative to the article text. To fix this, we will do some more customization, in the next section.

Replacing article styles In the previous section, we saw that the font size for articles from the print version of *The BBC* was too small. In most websites, *The BBC* included, this font size is set by means of *CSS* stylesheets. We can disable the fetching of such stylesheets by adding the line:

```
no_stylesheets = True
```

The recipe now looks like:

Recipe source code (python)

```
class AdvancedUserRecipe1206419520(BasicNewsRecipe):
    title          = u'The BBC'
    oldest_article = 7
    max_articles_per_feed = 100
    no_stylesheets = True

    feeds          = [(u'News Front Page', u'http://newsrss.bbc.co.uk/rss/news

    def print_version(self, url):
        return url.replace('http://', 'http://newsvote.bbc.co.uk/mpapps/pageto
```

The new version looks pretty good. If you're a perfectionist, you'll want to read the next section, which deals with actually modifying the downloaded content.

[163] http://www.python.org
[164] http://docs.python.org/tut/

Slicing and dicing calibre contains very powerful and flexible abilities when it comes to manipulating downloaded content. To show off a couple of these, let's look at our old friend the *The BBC* (page 345) recipe again. Looking at the source code (*HTML*) of a couple of articles (print version), we see that they have a footer that contains no useful information, contained in

```
<div class="footer">
...
</div>
```

This can be removed by adding:

```
remove_tags    = [dict(name='div', attrs={'class':'footer'})]
```

to the recipe. Finally, lets replace some of the *CSS* that we disabled earlier, with our own *CSS* that is suitable for conversion to an ebook:

```
extra_css      = '.headline {font-size: x-large;} \n .fact { padding-top: 10pt  }'
```

With these additions, our recipe has become "production quality", indeed it is very close to the actual recipe used by calibre for the *BBC*, shown below:

```
##
## Title:         BBC News, Sport, and Blog Calibre Recipe
## Contact:       mattst - jmstanfield@gmail.com
##
## License:       GNU General Public License v3 - http://www.gnu.org/copyleft/gpl.html
## Copyright:     mattst - jmstanfield@gmail.com
##
## Written:       November 2011
## Last Edited:   2011-11-19
##

__license__     = 'GNU General Public License v3 - http://www.gnu.org/copyleft/gpl.html'
__copyright__   = 'mattst - jmstanfield@gmail.com'


'''
BBC News, Sport, and Blog Calibre Recipe
'''

# Import the regular expressions module.
import re

# Import the BasicNewsRecipe class which this class extends.
from calibre.web.feeds.recipes import BasicNewsRecipe

class BBCNewsSportBlog(BasicNewsRecipe):

    #
    #    **** IMPORTANT USERS READ ME ****
    #
    #  First select the feeds you want then scroll down below the feeds list
    #  and select the values you want for the other user preferences, like
    #  oldest_article and such like.
    #
    #
    #  Select the BBC rss feeds which you want in your ebook.
    #  Selected feed have NO '#' at their start, de-selected feeds begin with a '#'.
    #
```

```
#  Eg.  ("News Home", "http://feeds.bbci.co.uk/... - include feed.
#  Eg. #("News Home", "http://feeds.bbci.co.uk/... - do not include feed.
#
# There are 68 feeds below which constitute the bulk of the available rss
# feeds on the BBC web site. These include 5 blogs by editors and
# correspondants, 16 sports feeds, 15 'sub' regional feeds (Eg. North West
# Wales, Scotland Business), and 7 Welsh language feeds.
#
# Some of the feeds are low volume (Eg. blogs), or very low volume (Eg. Click)
# so if "oldest_article = 1.5" (only articles published in the last 36 hours)
# you may get some 'empty feeds' which will not then be included in the ebook.
#
# The 15 feeds currently selected below are simply my default ones.
#
# Note: With all 68 feeds selected, oldest_article set to 2,
# max_articles_per_feed set to 100, and simultaneous_downloads set to 10,
# the ebook creation took 29 minutes on my speedy 100 mbps net connection,
# fairly high-end desktop PC running Linux (Ubuntu Lucid-Lynx).
# More realistically with 15 feeds selected, oldest_article set to 1.5,
# max_articles_per_feed set to 100, and simultaneous_downloads set to 20,
# it took 6 minutes. If that's too slow increase 'simultaneous_downloads'.
#
# Select / de-select the feeds you want in your ebook.
#
feeds = [
          ("News Home", "http://feeds.bbci.co.uk/news/rss.xml"),
          ("UK", "http://feeds.bbci.co.uk/news/uk/rss.xml"),
          ("World", "http://feeds.bbci.co.uk/news/world/rss.xml"),
          #("England", "http://feeds.bbci.co.uk/news/england/rss.xml"),
          #("Scotland", "http://feeds.bbci.co.uk/news/scotland/rss.xml"),
          #("Wales", "http://feeds.bbci.co.uk/news/wales/rss.xml"),
          #("N. Ireland", "http://feeds.bbci.co.uk/news/northern_ireland/rss.xml"),
          #("Africa", "http://feeds.bbci.co.uk/news/world/africa/rss.xml"),
          #("Asia", "http://feeds.bbci.co.uk/news/world/asia/rss.xml"),
          #("Europe", "http://feeds.bbci.co.uk/news/world/europe/rss.xml"),
          #("Latin America", "http://feeds.bbci.co.uk/news/world/latin_america/rss.xml"),
          #("Middle East", "http://feeds.bbci.co.uk/news/world/middle_east/rss.xml"),
          ("US & Canada", "http://feeds.bbci.co.uk/news/world/us_and_canada/rss.xml"),
          ("Politics", "http://feeds.bbci.co.uk/news/politics/rss.xml"),
          ("Science/Environment", "http://feeds.bbci.co.uk/news/science_and_environment/rss.xml"
          ("Technology", "http://feeds.bbci.co.uk/news/technology/rss.xml"),
          ("Magazine", "http://feeds.bbci.co.uk/news/magazine/rss.xml"),
          ("Entertainment/Arts", "http://feeds.bbci.co.uk/news/entertainment_and_arts/rss.xml"),
          #("Health", "http://feeds.bbci.co.uk/news/health/rss.xml"),
          #("Education/Family", "http://feeds.bbci.co.uk/news/education/rss.xml"),
          ("Business", "http://feeds.bbci.co.uk/news/business/rss.xml"),
          ("Special Reports", "http://feeds.bbci.co.uk/news/special_reports/rss.xml"),
          ("Also in the News", "http://feeds.bbci.co.uk/news/also_in_the_news/rss.xml"),
          #("Newsbeat", "http://www.bbc.co.uk/newsbeat/rss.xml"),
          #("Click", "http://newsrss.bbc.co.uk/rss/newsonline_uk_edition/programmes/click_online/
          ("Blog: Nick Robinson (Political Editor)", "http://feeds.bbci.co.uk/news/correspondents
          #("Blog: Mark D'Arcy (Parliamentary Correspondent)", "http://feeds.bbci.co.uk/news/cor
          #("Blog: Robert Peston (Business Editor)", "http://feeds.bbci.co.uk/news/correspondents
          #("Blog: Stephanie Flanders (Economics Editor)", "http://feeds.bbci.co.uk/news/corresp
          ("Blog: Rory Cellan-Jones (Technology correspondent)", "http://feeds.bbci.co.uk/news/co
          ("Sport Front Page", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/front_page/rs
          #("Football", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/football/rss.xml"),
          #("Cricket", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/cricket/rss.xml"),
```

```
            #("Rugby Union", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/rugby_union/rss.x
            #("Rugby League", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/rugby_league/rss
            #("Tennis", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/tennis/rss.xml"),
            #("Golf", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/golf/rss.xml"),
            #("Motorsport", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/motorsport/rss.xml
            #("Boxing", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/boxing/rss.xml"),
            #("Athletics", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/athletics/rss.xml",
            #("Snooker", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/snooker/
            #("Horse Racing", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/hor
            #("Cycling", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/cycling/
            #("Disability Sport", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports
            #("Other Sport", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/rss
            #("Olympics 2012", "http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/other_sports/ol
            #("N. Ireland Politics", "http://feeds.bbci.co.uk/news/northern_ireland/northern_irelan
            #("Scotland Politics", "http://feeds.bbci.co.uk/news/scotland/scotland_politics/rss.xml
            #("Scotland Business", "http://feeds.bbci.co.uk/news/scotland/scotland_business/rss.xml
            #("E. Scotland, Edinburgh & Fife", "http://feeds.bbci.co.uk/news/scotland/edinburgh_eas
            #("W. Scotland & Glasgow", "http://feeds.bbci.co.uk/news/scotland/glasgow_and_west/rss
            #("Highlands & Islands", "http://feeds.bbci.co.uk/news/scotland/highlands_and_islands/r
            #("NE. Scotland, Orkney & Shetland", "http://feeds.bbci.co.uk/news/scotland/north_east_
            #("South Scotland", "http://feeds.bbci.co.uk/news/scotland/south_scotland/rss.xml"),
            #("Central Scotland & Tayside", "http://feeds.bbci.co.uk/news/scotland/tayside_and_cent
            #("Wales Politics", "http://feeds.bbci.co.uk/news/wales/wales_politics/rss.xml"),
            #("NW. Wales", "http://feeds.bbci.co.uk/news/wales/north_west_wales/rss.xml"),
            #("NE. Wales", "http://feeds.bbci.co.uk/news/wales/north_east_wales/rss.xml"),
            #("Mid. Wales", "http://feeds.bbci.co.uk/news/wales/mid_wales/rss.xml"),
            #("SW. Wales", "http://feeds.bbci.co.uk/news/wales/south_west_wales/rss.xml"),
            #("SE. Wales", "http://feeds.bbci.co.uk/news/wales/south_east_wales/rss.xml"),
            #("Newyddion - News in Welsh", "http://feeds.bbci.co.uk/newyddion/rss.xml"),
            #("Gwleidyddiaeth", "http://feeds.bbci.co.uk/newyddion/gwleidyddiaeth/rss.xml"),
            #("Gogledd-Ddwyrain", "http://feeds.bbci.co.uk/newyddion/gogledd-ddwyrain/rss.xml"),
            #("Gogledd-Orllewin", "http://feeds.bbci.co.uk/newyddion/gogledd-orllewin/rss.xml"),
            #("Canolbarth", "http://feeds.bbci.co.uk/newyddion/canolbarth/rss.xml"),
            #("De-Ddwyrain", "http://feeds.bbci.co.uk/newyddion/de-ddwyrain/rss.xml"),
            #("De-Orllewin", "http://feeds.bbci.co.uk/newyddion/de-orllewin/rss.xml"),
        ]


#    **** SELECT YOUR USER PREFERENCES ****

# Title to use for the ebook.
#
title = 'BBC News'

# A brief description for the ebook.
#
description = u'BBC web site ebook created using rss feeds.'

# The max number of articles which may be downloaded from each feed.
# I've never seen more than about 70 articles in a single feed in the
# BBC feeds.
#
max_articles_per_feed = 100

# The max age of articles which may be downloaded from each feed. This is
# specified in days - note fractions of days are allowed, Eg. 2.5 (2 and a
# half days). My default of 1.5 days is the last 36 hours, the point at
# which I've decided 'news' becomes 'old news', but be warned this is not
```

```
# so good for the blogs, technology, magazine, etc., and sports feeds.
# You may wish to extend this to 2-5 but watch out ebook creation time will
# increase as well. Setting this to 30 will get everything (AFAICT) as long
# as max_articles_per_feed remains set high (except for 'Click' which is
# v. low volume and its currently oldest article is 4th Feb 2011).
#
oldest_article = 1.5

# Number of simultaneous downloads. 20 is consistantly working fine on the
# BBC News feeds with no problems. Speeds things up from the defualt of 5.
# If you have a lot of feeds and/or have increased oldest_article above 2
# then you may wish to try increasing simultaneous_downloads to 25-30,
# Or, of course, if you are in a hurry. [I've not tried beyond 20.]
#
simultaneous_downloads = 20

# Timeout for fetching files from the server in seconds. The default of
# 120 seconds, seems somewhat excessive.
#
timeout = 30

# The format string for the date shown on the ebook's first page.
# List of all values: http://docs.python.org/library/time.html
# Default in news.py has a leading space so that's mirrored here.
# As with 'feeds' select/de-select by adding/removing the initial '#',
# only one timefmt should be selected, here's a few to choose from.
#
timefmt = ' [%a, %d %b %Y]'                # [Fri, 14 Nov 2011] (Calibre default)
#timefmt = ' [%a, %d %b %Y %H:%M]'         # [Fri, 14 Nov 2011 18:30]
#timefmt = ' [%a, %d %b %Y %I:%M %p]'      # [Fri, 14 Nov 2011 06:30 PM]
#timefmt = ' [%d %b %Y]'                   # [14 Nov 2011]
#timefmt = ' [%d %b %Y %H:%M]'             # [14 Nov 2011 18.30]
#timefmt = ' [%Y-%m-%d]'                   # [2011-11-14]
#timefmt = ' [%Y-%m-%d-%H-%M]'             # [2011-11-14-18-30]



#
#    **** IMPORTANT ****
#
#    DO NOT EDIT BELOW HERE UNLESS YOU KNOW WHAT YOU ARE DOING.
#
#    DO NOT EDIT BELOW HERE UNLESS YOU KNOW WHAT YOU ARE DOING.
#
#    I MEAN IT, YES I DO, ABSOLUTELY, AT YOU OWN RISK. :)
#
#    **** IMPORTANT ****
#



# Author of this recipe.
__author__ = 'mattst'

# Specify English as the language of the RSS feeds (ISO-639 code).
language = 'en_GB'

# Set tags.
```

```
    tags = 'news, sport, blog'

    # Set publisher and publication type.
    publisher = 'BBC'
    publication_type = 'newspaper'

    # Disable stylesheets from site.
    no_stylesheets = True

    # Specifies an override encoding for sites that have an incorrect charset
    # specified. Default of 'None' says to auto-detect. Some other BBC recipes
    # use 'utf8', which works fine (so use that if necessary) but auto-detecting
    # with None is working fine, so stick with that for robustness.
    encoding = None

    # Sets whether a feed has full articles embedded in it. The BBC feeds do not.
    use_embedded_content = False

    # Removes empty feeds - why keep them!?
    remove_empty_feeds = True

    # Create a custom title which fits nicely in the Kindle title list.
    # Requires "import time" above class declaration, and replacing
    # title with custom_title in conversion_options (right column only).
    # Example of string below: "BBC News - 14 Nov 2011"
    #
    # custom_title = "BBC News - " + time.strftime('%d %b %Y')

    '''
    # Conversion options for advanced users, but don't forget to comment out the
    # current conversion_options below. Avoid setting 'linearize_tables' as that
    # plays havoc with the 'old style' table based pages.
    #
    conversion_options = { 'title'       : title,
                           'comments'    : description,
                           'tags'        : tags,
                           'language'    : language,
                           'publisher'   : publisher,
                           'authors'     : publisher,
                           'smarten_punctuation' : True
                         }
    '''

    conversion_options = { 'smarten_punctuation' : True }

    # Specify extra CSS - overrides ALL other CSS (IE. Added last).
    extra_css = 'body { font-family: verdana, helvetica, sans-serif; } \
                 .introduction, .first { font-weight: bold; } \
                 .cross-head { font-weight: bold; font-size: 125%; } \
                 .cap, .caption { display: block; font-size: 80%; font-style: italic; } \
                 .cap, .caption, .caption img, .caption span { display: block; text-align: center; ma
                 .byl, .byd, .byline img, .byline-name, .byline-title, .author-name, .author-position
                    .correspondent-portrait img, .byline-lead-in, .name, .bbc-role { display: block;
                    text-align: center; font-size: 80%; font-style: italic; margin: 1px auto; } \
                 .story-date, .published { font-size: 80%; } \
                 table { width: 100%; } \
                 td img { display: block; margin: 5px auto; } \
                 ul { padding-top: 10px; } \
```

```
                ol { padding-top: 10px; } \
                li { padding-top: 5px; padding-bottom: 5px; } \
                h1 { text-align: center; font-size: 175%; font-weight: bold; } \
                h2 { text-align: center; font-size: 150%; font-weight: bold; } \
                h3 { text-align: center; font-size: 125%; font-weight: bold; } \
                h4, h5, h6 { text-align: center; font-size: 100%; font-weight: bold; }'

    # Remove various tag attributes to improve the look of the ebook pages.
    remove_attributes = [ 'border', 'cellspacing', 'align', 'cellpadding', 'colspan',
                          'valign', 'vspace', 'hspace', 'alt', 'width', 'height' ]

    # Remove the (admittedly rarely used) line breaks, "<br />", which sometimes
    # cause a section of the ebook to start in an unsightly fashion or, more
    # frequently, a "<br />" will muck up the formatting of a correspondant's byline.
    # "<br />" and "<br clear/>" are far more frequently used on the table formatted
    # style of pages, and really spoil the look of the ebook pages.
    preprocess_regexps     = [(re.compile(r'<br[ ]*/>', re.IGNORECASE), lambda m: ''),
                              (re.compile(r'<br[ ]*clear.*/>', re.IGNORECASE), lambda m: '')]


    # Create regular expressions for tag keeping and removal to make the matches more
    # robust against minor changes and errors in the HTML, Eg. double spaces, leading
    # and trailing spaces, missing hyphens, and such like.
    # Python regular expression ('re' class) page: http://docs.python.org/library/re.html

    # ***************************************
    # Regular expressions for keep_only_tags:
    # ***************************************

    # The BBC News HTML pages use variants of 'storybody' to denote the section of a HTML
    # page which contains the main text of the article. Match storybody variants: 'storybody',
    # 'story-body', 'story body','storybody ', etc.
    storybody_reg_exp = '^.*story[_ -]*body.*$'

    # The BBC sport and 'newsbeat' (features) HTML pages use 'blq_content' to hold the title
    # and published date. This is one level above the usual news pages which have the title
    # and date within 'story-body'. This is annoying since 'blq_content' must also be kept,
    # resulting in a lot of extra things to be removed by remove_tags.
    blq_content_reg_exp = '^.*blq[_ -]*content.*$'

    # The BBC has an alternative page design structure, which I suspect is an out-of-date
    # design but which is still used in some articles, Eg. 'Click' (technology), 'FastTrack'
    # (travel), and in some sport pages. These alternative pages are table based (which is
    # why I think they are an out-of-date design) and account for -I'm guesstimaking- less
    # than 1% of all articles. They use a table class 'storycontent' to hold the article
    # and like blq_content (above) have required lots of extra removal by remove_tags.
    story_content_reg_exp = '^.*story[_ -]*content.*$'

    # Keep the sections of the HTML which match the list below. The HTML page created by
    # Calibre will fill <body> with those sections which are matched. Note that the
    # blq_content_reg_exp must be listed before storybody_reg_exp in keep_only_tags due to
    # it being the parent of storybody_reg_exp, that is to say the div class/id 'story-body'
    # will be inside div class/id 'blq_content' in the HTML (if 'blq_content' is there at
    # all). If they are the other way around in keep_only_tags then blq_content_reg_exp
    # will end up being discarded.
    keep_only_tags = [ dict(name='table', attrs={'class':re.compile(story_content_reg_exp, re.IGNOREC
                       dict(name='div',   attrs={'class':re.compile(blq_content_reg_exp, re.IGNORECAS
                       dict(name='div',   attrs={'id':re.compile(blq_content_reg_exp, re.IGNORECASE)
```

```
                    dict(name='div',    attrs={'class':re.compile(storybody_reg_exp, re.IGNORECASE)
                    dict(name='div',    attrs={'id':re.compile(storybody_reg_exp, re.IGNORECASE)})

    # **********************************
    # Regular expressions for remove_tags:
    # **********************************

    # Regular expression to remove share-help and variant tags. The share-help class
    # is used by the site for a variety of 'sharing' type links, Eg. Facebook, delicious,
    # twitter, email. Removed to avoid page clutter.
    share_help_reg_exp = '^.*share[_ -]*help.*$'

    # Regular expression to remove embedded-hyper and variant tags. This class is used to
    # display links to other BBC News articles on the same/similar subject.
    embedded_hyper_reg_exp = '^.*embed*ed[_ -]*hyper.*$'

    # Regular expression to remove hypertabs and variant tags. This class is used to
    # display a tab bar at the top of an article which allows the user to switch to
    # an article (viewed on the same page) providing further info., 'in depth' analysis,
    # an editorial, a correspondant's blog entry, and such like. The ability to handle
    # a tab bar of this nature is currently beyond the scope of this recipe and
    # possibly of Calibre itself (not sure about that - TO DO - check!).
    hypertabs_reg_exp = '^.*hyper[_ -]*tabs.*$'

    # Regular expression to remove story-feature and variant tags. Eg. 'story-feature',
    # 'story-feature related narrow', 'story-feature wide', 'story-feature narrow'.
    # This class is used to add additional info. boxes, or small lists, outside of
    # the main story. TO DO: Work out a way to incorporate these neatly.
    story_feature_reg_exp = '^.*story[_ -]*feature.*$'

    # Regular expression to remove video and variant tags, Eg. 'videoInStoryB',
    # 'videoInStoryC'. This class is used to embed video.
    video_reg_exp = '^.*video.*$'

    # Regular expression to remove audio and variant tags, Eg. 'audioInStoryD'.
    # This class is used to embed audio.
    audio_reg_exp = '^.*audio.*$'

    # Regular expression to remove pictureGallery and variant tags, Eg. 'pictureGallery'.
    # This class is used to embed a photo slideshow. See also 'slideshow' below.
    picture_gallery_reg_exp = '^.*picture.*$'

    # Regular expression to remove slideshow and variant tags, Eg. 'dslideshow-enclosure'.
    # This class is used to embed a slideshow (not necessarily photo) but both
    # 'slideshow' and 'pictureGallery' are used for slideshows.
    slideshow_reg_exp = '^.*slide[_ -]*show.*$'

    # Regular expression to remove social-links and variant tags. This class is used to
    # display links to a BBC bloggers main page, used in various columnist's blogs
    # (Eg. Nick Robinson, Robert Preston).
    social_links_reg_exp = '^.*social[_ -]*links.*$'

    # Regular expression to remove quote and (multi) variant tags, Eg. 'quote',
    # 'endquote', 'quote-credit', 'quote-credit-title', etc. These are usually
    # removed by 'story-feature' removal (as they are usually within them), but
    # not always. The quotation removed is always (AFAICT) in the article text
    # as well but a 2nd copy is placed in a quote tag to draw attention to it.
    # The quote class tags may or may not appear in div's.
```

```
    quote_reg_exp = '^.*quote.*$'

    # Regular expression to remove hidden and variant tags, Eg. 'hidden'.
    # The purpose of these is unclear, they seem to be an internal link to a
    # section within the article, but the text of the link (Eg. 'Continue reading
    # the main story') never seems to be displayed anyway. Removed to avoid clutter.
    # The hidden class tags may or may not appear in div's.
    hidden_reg_exp = '^.*hidden.*$'

    # Regular expression to remove comment and variant tags, Eg. 'comment-introduction'.
    # Used on the site to display text about registered users entering comments.
    comment_reg_exp = '^.*comment.*$'

    # Regular expression to remove form and variant tags, Eg. 'comment-form'.
    # Used on the site to allow registered BBC users to fill in forms, typically
    # for entering comments about an article.
    form_reg_exp = '^.*form.*$'

    # Extra things to remove due to the addition of 'blq_content' in keep_only_tags.

    #<div class="story-actions"> Used on sports pages for 'email' and 'print'.
    story_actions_reg_exp = '^.*story[_ -]*actions.*$'

    #<div class="bookmark-list"> Used on sports pages instead of 'share-help' (for
    # social networking links).
    bookmark_list_reg_exp = '^.*bookmark[_ -]*list.*$'

    #<div id="secondary-content" class="content-group">
    # NOTE: Don't remove class="content-group" that is needed.
    # Used on sports pages to link to 'similar stories'.
    secondary_content_reg_exp = '^.*secondary[_ -]*content.*$'

    #<div id="featured-content" class="content-group">
    # NOTE: Don't remove class="content-group" that is needed.
    # Used on sports pages to link to pages like 'tables', 'fixtures', etc.
    featured_content_reg_exp = '^.*featured[_ -]*content.*$'

    #<div id="navigation">
    # Used on sports pages to link to pages like 'tables', 'fixtures', etc.
    # Used sometimes instead of "featured-content" above.
    navigation_reg_exp = '^.*navigation.*$'

    #<a class="skip" href="#blq-container-inner">Skip to top</a>
    # Used on sports pages to link to the top of the page.
    skip_reg_exp = '^.*skip.*$'

    # Extra things to remove due to the addition of 'storycontent' in keep_only_tags,
    # which are the alterative table design based pages. The purpose of some of these
    # is not entirely clear from the pages (which are a total mess!).

    # Remove mapping based tags, Eg. <map id="world_map">
    # The dynamic maps don't seem to work during ebook creation. TO DO: Investigate.
    map_reg_exp = '^.*map.*$'

    # Remove social bookmarking variation, called 'socialBookMarks'.
    social_bookmarks_reg_exp = '^.*social[_ -]*bookmarks.*$'

    # Remove page navigation tools, like 'search', 'email', 'print', called 'blq-mast'.
```

```
blq_mast_reg_exp = '^.*blq[_ -]*mast.*$'

# Remove 'sharesb', I think this is a generic 'sharing' class. It seems to appear
# alongside 'socialBookMarks' whenever that appears. I am removing it as well
# under the assumption that it can appear alone as well.
sharesb_reg_exp = '^.*sharesb.*$'

# Remove class 'o'. The worst named user created css class of all time. The creator
# should immediately be fired. I've seen it used to hold nothing at all but with
# 20 or so empty lines in it. Also to hold a single link to another article.
# Whatever it was designed to do it is not wanted by this recipe. Exact match only.
o_reg_exp = '^o$'

# Remove 'promotopbg' and 'promobottombg', link lists. Have decided to
# use two reg expressions to make removing this (and variants) robust.
promo_top_reg_exp = '^.*promotopbg.*$'
promo_bottom_reg_exp = '^.*promobottombg.*$'

# Remove 'nlp', provides heading for link lists. Requires an exact match due to
# risk of matching those letters in something needed, unless I see a variation
# of 'nlp' used at a later date.
nlp_reg_exp = '^nlp$'

# Remove 'mva', provides embedded floating content of various types. Variant 'mvb'
# has also now been seen. Requires an exact match of 'mva' or 'mvb' due to risk of
# matching those letters in something needed.
mva_or_mvb_reg_exp = '^mv[ab]$'

# Remove 'mvtb', seems to be page navigation tools, like 'blq-mast'.
mvtb_reg_exp = '^mvtb$'

# Remove 'blq-toplink', class to provide a link to the top of the page.
blq_toplink_reg_exp = '^.*blq[_ -]*top[_ -]*link.*$'

# Remove 'products and services' links, Eg. desktop tools, alerts, and so on.
# Eg. Class="servicev4 ukfs_services" - what a mess of a name. Have decided to
# use two reg expressions to make removing this (and variants) robust.
prods_services_01_reg_exp = '^.*servicev4.*$'
prods_services_02_reg_exp = '^.*ukfs[_ -]*services.*$'

# Remove -what I think is- some kind of navigation tools helper class, though I am
# not sure, it's called: 'blq-rst blq-new-nav'. What I do know is it pops up
# frequently and it is not wanted. Have decided to use two reg expressions to make
# removing this (and variants) robust.
blq_misc_01_reg_exp = '^.*blq[_ -]*rst.*$'
blq_misc_02_reg_exp = '^.*blq[_ -]*new[_ -]*nav.*$'

# Remove 'puffbox' - this may only appear inside 'storyextra', so it may not
# need removing - I have no clue what it does other than it contains links.
# Whatever it is - it is not part of the article and is not wanted.
puffbox_reg_exp = '^.*puffbox.*$'

# Remove 'sibtbg' and 'sibtbgf' - some kind of table formatting classes.
sibtbg_reg_exp = '^.*sibtbg.*$'

# Remove 'storyextra' - links to relevant articles and external sites.
storyextra_reg_exp = '^.*story[_ -]*extra.*$'
```

```
    remove_tags = [ dict(name='div',  attrs={'class':re.compile(story_feature_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'class':re.compile(share_help_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(embedded_hyper_reg_exp, re.IGNORECASE
                    dict(name='div',  attrs={'class':re.compile(hypertabs_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(video_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(audio_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(picture_gallery_reg_exp, re.IGNORECAS
                    dict(name='div',  attrs={'class':re.compile(slideshow_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(quote_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(hidden_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(comment_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(story_actions_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'class':re.compile(bookmark_list_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'id':re.compile(secondary_content_reg_exp, re.IGNORECASE
                    dict(name='div',  attrs={'id':re.compile(featured_content_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'id':re.compile(navigation_reg_exp, re.IGNORECASE)}),
                    dict(name='form', attrs={'id':re.compile(form_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(quote_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(hidden_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(social_links_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(comment_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(skip_reg_exp, re.IGNORECASE)}),
                    dict(name='map', attrs={'id':re.compile(map_reg_exp, re.IGNORECASE)}),
                    dict(name='map', attrs={'name':re.compile(map_reg_exp, re.IGNORECASE)}),
                    dict(name='div', attrs={'id':re.compile(social_bookmarks_reg_exp, re.IGNORECASE)
                    dict(name='div', attrs={'id':re.compile(blq_mast_reg_exp, re.IGNORECASE)}),
                    dict(name='div', attrs={'class':re.compile(sharesb_reg_exp, re.IGNORECASE)}),
                    dict(name='div', attrs={'class':re.compile(o_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(promo_top_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(promo_bottom_reg_exp, re.IGNORECASE)
                    dict(name='div',  attrs={'class':re.compile(nlp_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(mva_or_mvb_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(mvtb_reg_exp, re.IGNORECASE)}),
                    dict(name='div',  attrs={'class':re.compile(blq_toplink_reg_exp, re.IGNORECASE)}
                    dict(name='div',  attrs={'class':re.compile(prods_services_01_reg_exp, re.IGNOREC
                    dict(name='div',  attrs={'class':re.compile(prods_services_02_reg_exp, re.IGNOREC
                    dict(name='div',  attrs={'class':re.compile(blq_misc_01_reg_exp, re.IGNORECASE)}
                    dict(name='div',  attrs={'class':re.compile(blq_misc_02_reg_exp, re.IGNORECASE)}
                    dict(name='div',  attrs={'class':re.compile(puffbox_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(sibtbg_reg_exp, re.IGNORECASE)}),
                    dict(attrs={'class':re.compile(storyextra_reg_exp, re.IGNORECASE)})
                  ]

    # Uses url to create and return the 'printer friendly' version of the url.
    # In other words the 'print this page' address of the page.
    #
    # There are 3 types of urls used in the BBC site's rss feeds. There is just
    # 1 type for the standard news while there are 2 used for sports feed urls.
    # Note: Sports urls are linked from regular news feeds (Eg. 'News Home') when
    # there is a major story of interest to 'everyone'. So even if no BBC sports
    # feeds are added to 'feeds' the logic of this method is still needed to avoid
    # blank / missing / empty articles which have an index title and then no body.
    def print_version(self, url):

        # Handle sports page urls type 01:
        if (url.find("go/rss/-/sport1/") != -1):
            temp_url = url.replace("go/rss/-/", "")
```

```
        # Handle sports page urls type 02:
        elif (url.find("go/rss/int/news/-/sport1/") != -1):
            temp_url = url.replace("go/rss/int/news/-/", "")

        # Handle regular news page urls:
        else:
            temp_url = url.replace("go/rss/int/news/-/", "")

        # Always add "?print=true" to the end of the url.
        print_url = temp_url + "?print=true"

        return print_url


    # Remove articles in feeds based on a string in the article title or url.
    #
    # Code logic written by: Starson17 - posted in: "Recipes - Re-usable code"
    # thread, in post with title: "Remove articles from feed", see url:
    # http://www.mobileread.com/forums/showpost.php?p=1165462&postcount=6
    # Many thanks and all credit to Starson17.
    #
    # Starson17's code has obviously been altered to suite my requirements.
    def parse_feeds(self):

        # Call parent's method.
        feeds = BasicNewsRecipe.parse_feeds(self)

        # Loop through all feeds.
        for feed in feeds:

            # Loop through all articles in feed.
            for article in feed.articles[:]:

                # Match key words and remove article if there's a match.

                # Most BBC rss feed video only 'articles' use upper case 'VIDEO'
                # as a title prefix. Just match upper case 'VIDEO', so that
                # articles like 'Video game banned' won't be matched and removed.
                if 'VIDEO' in article.title:
                    feed.articles.remove(article)

                # Most BBC rss feed audio only 'articles' use upper case 'AUDIO'
                # as a title prefix. Just match upper case 'AUDIO', so that
                # articles like 'Hi-Def audio...' won't be matched and removed.
                elif 'AUDIO' in article.title:
                    feed.articles.remove(article)

                # Most BBC rss feed photo slideshow 'articles' use 'In Pictures',
                # 'In pictures', and 'in pictures', somewhere in their title.
                # Match any case of that phrase.
                elif 'IN PICTURES' in article.title.upper():
                    feed.articles.remove(article)

                # As above, but user contributed pictures. Match any case.
                elif 'YOUR PICTURES' in article.title.upper():
                    feed.articles.remove(article)

                # 'Sportsday Live' are articles which contain a constantly and
```

```
                # dynamically updated 'running commentary' during a live sporting
                # event. Match any case.
                elif 'SPORTSDAY LIVE' in article.title.upper():
                    feed.articles.remove(article)

                # Sometimes 'Sportsday Live' (above) becomes 'Live - Sport Name'.
                # These are being matched below using 'Live - ' because removing all
                # articles with 'live' in their titles would remove some articles
                # that are in fact not live sports pages. Match any case.
                elif 'LIVE - ' in article.title.upper():
                    feed.articles.remove(article)

                # 'Quiz of the week' is a Flash player weekly news quiz. Match only
                # the 'Quiz of the' part in anticipation of monthly and yearly
                # variants. Match any case.
                elif 'QUIZ OF THE' in article.title.upper():
                    feed.articles.remove(article)

                # Remove articles with 'scorecards' in the url. These are BBC sports
                # pages which just display a cricket scorecard. The pages have a mass
                # of table and css entries to display the scorecards nicely. Probably
                # could make them work with this recipe, but might take a whole day
                # of work to sort out all the css - basically a formatting nightmare.
                elif 'scorecards' in article.url:
                    feed.articles.remove(article)

        return feeds

# End of class and file.
```

This *recipe* explores only the tip of the iceberg when it comes to the power of calibre. To explore more of the abilities of calibre we'll examine a more complex real life example in the next section.

Real life example A reasonably complex real life example that exposes more of the *API* of `BasicNewsRecipe` is the *recipe* for *The New York Times*

```
import string, re
from calibre import strftime
from calibre.web.feeds.recipes import BasicNewsRecipe
from calibre.ebooks.BeautifulSoup import BeautifulSoup

class NYTimes(BasicNewsRecipe):

    title       = 'The New York Times'
    __author__  = 'Kovid Goyal'
    description = 'Daily news from the New York Times'
    timefmt = ' [%a, %d %b, %Y]'
    needs_subscription = True
    remove_tags_before = dict(id='article')
    remove_tags_after  = dict(id='article')
    remove_tags = [dict(attrs={'class':['articleTools', 'post-tools', 'side_tool', 'nextArticleLink c
                dict(id=['footer', 'toolsRight', 'articleInline', 'navigation', 'archive', 'side_sear
                dict(name=['script', 'noscript', 'style'])]
    encoding = 'cp1252'
    no_stylesheets = True
    extra_css = 'h1 {font: sans-serif large;}\n.byline {font:monospace;}'

    def get_browser(self):
```

```
        br = BasicNewsRecipe.get_browser()
        if self.username is not None and self.password is not None:
            br.open('http://www.nytimes.com/auth/login')
            br.select_form(name='login')
            br['USERID']   = self.username
            br['PASSWORD'] = self.password
            br.submit()
        return br

    def parse_index(self):
        soup = self.index_to_soup('http://www.nytimes.com/pages/todayspaper/index.html')

        def feed_title(div):
            return ''.join(div.findAll(text=True, recursive=False)).strip()

        articles = {}
        key = None
        ans = []
        for div in soup.findAll(True,
             attrs={'class':['section-headline', 'story', 'story headline']}):

             if div['class'] == 'section-headline':
                 key = string.capwords(feed_title(div))
                 articles[key] = []
                 ans.append(key)

             elif div['class'] in ['story', 'story headline']:
                 a = div.find('a', href=True)
                 if not a:
                     continue
                 url = re.sub(r'\?.*', '', a['href'])
                 url += '?pagewanted=all'
                 title = self.tag_to_string(a, use_alt=True).strip()
                 description = ''
                 pubdate = strftime('%a, %d %b')
                 summary = div.find(True, attrs={'class':'summary'})
                 if summary:
                     description = self.tag_to_string(summary, use_alt=False)

                 feed = key if key is not None else 'Uncategorized'
                 if not articles.has_key(feed):
                     articles[feed] = []
                 if not 'podcasts' in url:
                     articles[feed].append(
                               dict(title=title, url=url, date=pubdate,
                                    description=description,
                                    content=''))
        ans = self.sort_index_by(ans, {'The Front Page':-1, 'Dining In, Dining Out':1, 'Obituaries':2
        ans = [(key, articles[key]) for key in ans if articles.has_key(key)]
        return ans

    def preprocess_html(self, soup):
        refresh = soup.find('meta', {'http-equiv':'refresh'})
        if refresh is None:
            return soup
        content = refresh.get('content').partition('=')[2]
        raw = self.browser.open('http://www.nytimes.com'+content).read()
        return BeautifulSoup(raw.decode('cp1252', 'replace'))
```

We see several new features in this *recipe*. First, we have:

```
timefmt = ' [%a, %d %b, %Y]'
```

This sets the displayed time on the front page of the created ebook to be in the format, `Day, Day_Number Month, Year`. See `timefmt` (page 368).

Then we see a group of directives to cleanup the downloaded *HTML*:

```
remove_tags_before = dict(name='h1')
remove_tags_after  = dict(id='footer')
remove_tags = ...
```

These remove everything before the first `<h1>` tag and everything after the first tag whose id is `footer`. See `remove_tags` (page 367), `remove_tags_before` (page 367), `remove_tags_after` (page 367).

The next interesting feature is:

```
needs_subscription = True
...
def get_browser(self):
    ...
```

`needs_subscription = True` tells calibre that this recipe needs a username and password in order to access the content. This causes, calibre to ask for a username and password whenever you try to use this recipe. The code in `calibre.web.feeds.news.BasicNewsRecipe.get_browser()` (page 361) actually does the login into the NYT website. Once logged in, calibre will use the same, logged in, browser instance to fetch all content. See mechanize[165] to understand the code in `get_browser`.

The next new feature is the `calibre.web.feeds.news.BasicNewsRecipe.parse_index()` (page 362) method. Its job is to go to http://www.nytimes.com/pages/todayspaper/index.html and fetch the list of articles that appear in *todays* paper. While more complex than simply using *RSS*, the recipe creates an ebook that corresponds very closely to the days paper. `parse_index` makes heavy use of BeautifulSoup[166] to parse the daily paper webpage.

The final new feature is the `calibre.web.feeds.news.BasicNewsRecipe.preprocess_html()` (page 363) method. It can be used to perform arbitrary transformations on every downloaded HTML page. Here it is used to bypass the ads that the nytimes shows you before each article.

Tips for developing new recipes

The best way to develop new recipes is to use the command line interface. Create the recipe using your favorite python editor and save it to a file say `myrecipe.recipe`. The *.recipe* extension is required. You can download content using this recipe with the command:

```
ebook-convert myrecipe.recipe .epub --test -vv --debug-pipeline debug
```

The command **ebook-convert** will download all the webpages and save them to the EPUB file `myrecipe.epub`. The *-vv* makes ebook-convert spit out a lot of information about what it is doing. The *--test* makes it download only a couple of articles from at most two feeds. In addition, ebook-convert will put the downloaded HTML into the `debug/input` directory, where `debug` is the directory you specified in the *--debug-pipeline* option.

Once the download is complete, you can look at the downloaded *HTML* by opening the file `debug/input/index.html` in a browser. Once you're satisfied that the download and preprocessing is happening correctly, you can generate ebooks in different formats as shown below:

[165] http://wwwsearch.sourceforge.net/mechanize/
[166] http://www.crummy.com/software/BeautifulSoup/documentation.html

```
ebook-convert myrecipe.recipe myrecipe.epub
ebook-convert myrecipe.recipe myrecipe.mobi
...
```

If you're satisfied with your recipe, and you feel there is enough demand to justify its inclusion into the set of built-in recipes, post your recipe in the calibre recipes forum[167] to share it with other calibre users.

Note: On OS X, the ebook-convert command will not be available by default. Go to Preferences->Miscellaneous and click the install command line tools button to make it available.

See Also:

ebook-convert **(page 486)** The command line interface for all ebook conversion.

Further reading

To learn more about writing advanced recipes using some of the facilities, available in `BasicNewsRecipe` you should consult the following sources:

API Documentation **(page 360)** Documentation of the `BasicNewsRecipe` class and all its important methods and fields.

BasicNewsRecipe[168] The source code of `BasicNewsRecipe`

Built-in recipes[169] The source code for the built-in recipes that come with calibre

The calibre recipes forum[170] Lots of knowledgeable calibre recipe writers hang out here.

API documentation

API Documentation for recipes The API for writing recipes is defined by the `BasicNewsRecipe` (page 360)

class `calibre.web.feeds.news.`**BasicNewsRecipe**(*options*, *log*, *progress_reporter*)

Base class that contains logic needed in all recipes. By overriding progressively more of the functionality in this class, you can make progressively more customized/powerful recipes. For a tutorial introduction to creating recipes, see *Adding your favorite news website* (page 341).

abort_recipe_processing(*msg*)

Causes the recipe download system to abort the download of this recipe, displaying a simple feedback message to the user.

add_toc_thumbnail(*article*, *src*)

Call this from populate_article_metadata with the src attribute of an <img> tag from the article that is appropriate for use as the thumbnail representing the article in the Table of Contents. Whether the thumbnail is actually used is device dependent (currently only used by the Kindles). Note that the referenced image must be one that was successfully downloaded, otherwise it will be ignored.

classmethod adeify_images(*soup*)

If your recipe when converted to EPUB has problems with images when viewed in Adobe Digital Editions, call this method from within `postprocess_html()` (page 363).

cleanup()

Called after all articles have been download. Use it to do any cleanup like logging out of subscription sites, etc.

[167] http://www.mobileread.com/forums/forumdisplay.php?f=228

clone_browser(*br*)
Clone the browser br. Cloned browsers are used for multi-threaded downloads, since mechanize is not thread safe. The default cloning routines should capture most browser customization, but if you do something exotic in your recipe, you should override this method in your recipe and clone manually.

Cloned browser instances use the same, thread-safe CookieJar by default, unless you have customized cookie handling.

default_cover(*cover_file*)
Create a generic cover for recipes that dont have a cover

download()
Download and pre-process all articles from the feeds in this recipe. This method should be called only once on a particular Recipe instance. Calling it more than once will lead to undefined behavior. :return: Path to index.html

extract_readable_article(*html*, *url*)
Extracts main article content from 'html', cleans up and returns as a (article_html, extracted_title) tuple. Based on the original readability algorithm by Arc90.

get_article_url(*article*)
Override in a subclass to customize extraction of the *URL* that points to the content for each article. Return the article URL. It is called with *article*, an object representing a parsed article from a feed. See feedparser[171]. By default it looks for the original link (for feeds syndicated via a service like feedburner or pheedo) and if found, returns that or else returns article.link[172].

get_browser(**args*, ***kwargs*)
Return a browser instance used to fetch documents from the web. By default it returns a mechanize[173] browser instance that supports cookies, ignores robots.txt, handles refreshes and has a mozilla firefox user agent.

If your recipe requires that you login first, override this method in your subclass. For example, the following code is used in the New York Times recipe to login for full access:

```
def get_browser(self):
    br = BasicNewsRecipe.get_browser(self)
    if self.username is not None and self.password is not None:
        br.open('http://www.nytimes.com/auth/login')
        br.select_form(name='login')
        br['USERID']   = self.username
        br['PASSWORD'] = self.password
        br.submit()
    return br
```

get_cover_url()
Return a *URL* to the cover image for this issue or *None*. By default it returns the value of the member *self.cover_url* which is normally *None*. If you want your recipe to download a cover for the e-book override this method in your subclass, or set the member variable *self.cover_url* before this method is called.

get_feeds()
Return a list of *RSS* feeds to fetch for this profile. Each element of the list must be a 2-element tuple of the form (title, url). If title is None or an empty string, the title from the feed is used. This method is useful if your recipe needs to do some processing to figure out the list of feeds to download. If so, override in your subclass.

get_masthead_title()
Override in subclass to use something other than the recipe title

[171] http://packages.python.org/feedparser/
[172] http://packages.python.org/feedparser/reference-entry-link.html
[173] http://wwwsearch.sourceforge.net/mechanize/

get_masthead_url()
Return a *URL* to the masthead image for this issue or *None*. By default it returns the value of the member *self.masthead_url* which is normally *None*. If you want your recipe to download a masthead for the e-book override this method in your subclass, or set the member variable *self.masthead_url* before this method is called. Masthead images are used in Kindle MOBI files.

get_obfuscated_article(*url*)
If you set *articles_are_obfuscated* this method is called with every article URL. It should return the path to a file on the filesystem that contains the article HTML. That file is processed by the recursive HTML fetching engine, so it can contain links to pages/images on the web.

This method is typically useful for sites that try to make it difficult to access article content automatically.

classmethod image_url_processor(*baseurl*, *url*)
Perform some processing on image urls (perhaps removing size restrictions for dynamically generated images, etc.) and return the precessed URL.

index_to_soup(*url_or_raw*, *raw=False*)
Convenience method that takes an URL to the index page and returns a BeautifulSoup[174] of it.

url_or_raw: Either a URL or the downloaded index page as a string

is_link_wanted(*url*, *tag*)
Return True if the link should be followed or False otherwise. By default, raises NotImplementedError which causes the downloader to ignore it.

Parameters

- **url** – The URL to be followed
- **tag** – The Tag from which the URL was derived

javascript_login(*browser*, *username*, *password*)
This method is used to login to a website that uses javascript for its login form. After the login is complete, the cookies returned from the website are copied to a normal (non-javascript) browser and the download proceeds using those cookies.

An example implementation:

```
def javascript_login(self, browser, username, password):
    browser.visit('http://some-page-that-has-a-login')
    form = browser.select_form(nr=0) # Select the first form on the page
    form['username'] = username
    form['password'] = password
    browser.submit(timeout=120) # Submit the form and wait at most two minutes for loading t
```

Note that you can also select forms with CSS2 selectors, like this:

```
browser.select_form('form#login_form')
browser.select_from('form[name="someform"]')
```

parse_feeds()
Create a list of articles from the list of feeds returned by `BasicNewsRecipe.get_feeds()` (page 361). Return a list of `Feed` objects.

parse_index()
This method should be implemented in recipes that parse a website instead of feeds to generate a list of articles. Typical uses are for news sources that have a "Print Edition" webpage that lists all the articles in the current print edition. If this function is implemented, it will be used in preference to `BasicNewsRecipe.parse_feeds()` (page 362).

[174] http://www.crummy.com/software/BeautifulSoup/documentation.html

It must return a list. Each element of the list must be a 2-element tuple of the form `('feed title', list of articles)`.

Each list of articles must contain dictionaries of the form:

```
{
'title'       : article title,
'url'         : URL of print version,
'date'        : The publication date of the article as a string,
'description' : A summary of the article
'content'     : The full article (can be an empty string). Obsolete
                do not use, instead save the content to a temporary
                file and pass a file:///path/to/temp/file.html as
                the URL.
}
```

For an example, see the recipe for downloading *The Atlantic*. In addition, you can add 'author' for the author of the article.

If you want to abort processing for some reason and have calibre show the user a simple message instead of an error, call `abort_recipe_processing()` (page 360).

populate_article_metadata(*article*, *soup*, *first*)

Called when each HTML page belonging to article is downloaded. Intended to be used to get article metadata like author/summary/etc. from the parsed HTML (soup). :param article: A object of class `calibre.web.feeds.Article`. If you change the summary, remember to also change the text_summary :param soup: Parsed HTML belonging to this article :param first: True iff the parsed HTML is the first page of the article.

postprocess_book(*oeb*, *opts*, *log*)

Run any needed post processing on the parsed downloaded e-book.

Parameters

- **oeb** – An OEBBook object
- **opts** – Conversion options

postprocess_html(*soup*, *first_fetch*)

This method is called with the source of each downloaded *HTML* file, after it is parsed for links and images. It can be used to do arbitrarily powerful post-processing on the *HTML*. It should return *soup* after processing it.

Parameters

- **soup** – A BeautifulSoup[175] instance containing the downloaded *HTML*.
- **first_fetch** – True if this is the first page of an article.

preprocess_html(*soup*)

This method is called with the source of each downloaded *HTML* file, before it is parsed for links and images. It is called after the cleanup as specified by remove_tags etc. It can be used to do arbitrarily powerful pre-processing on the *HTML*. It should return *soup* after processing it.

soup: A BeautifulSoup[176] instance containing the downloaded *HTML*.

preprocess_raw_html(*raw_html*, *url*)

This method is called with the source of each downloaded *HTML* file, before it is parsed into an object tree. raw_html is a unicode string representing the raw HTML downloaded from the web. url is the URL from which the HTML was downloaded.

[175] http://www.crummy.com/software/BeautifulSoup/documentation.html
[176] http://www.crummy.com/software/BeautifulSoup/documentation.html

Note that this method acts *before* preprocess_regexps.

This method must return the processed raw_html as a unicode object.

classmethod `print_version` (*url*)
Take a *url* pointing to the webpage with article content and return the *URL* pointing to the print version of the article. By default does nothing. For example:

```
def print_version(self, url):
    return url + '?&pagewanted=print'
```

`skip_ad_pages` (*soup*)
This method is called with the source of each downloaded *HTML* file, before any of the cleanup attributes like remove_tags, keep_only_tags are applied. Note that preprocess_regexps will have already been applied. It is meant to allow the recipe to skip ad pages. If the soup represents an ad page, return the HTML of the real page. Otherwise return None.

soup: A BeautifulSoup[177] instance containing the downloaded *HTML*.

`sort_index_by` (*index*, *weights*)
Convenience method to sort the titles in *index* according to *weights*. *index* is sorted in place. Returns *index*.

index: A list of titles.

weights: A dictionary that maps weights to titles. If any titles in index are not in weights, they are assumed to have a weight of 0.

classmethod `tag_to_string` (*tag*, *use_alt=True*, *normalize_whitespace=True*)
Convenience method to take a BeautifulSoup[178] *Tag* and extract the text from it recursively, including any CDATA sections and alt tag attributes. Return a possibly empty unicode string.

use_alt: If *True* try to use the alt attribute for tags that don't have any textual content

tag: BeautifulSoup[179] *Tag*

`articles_are_obfuscated` = False
Set to True and implement `get_obfuscated_article()` (page 362) to handle websites that try to make it difficult to scrape content.

`auto_cleanup` = False
Automatically extract all the text from downloaded article pages. Uses the algorithms from the readability project. Setting this to True, means that you do not have to worry about cleaning up the downloaded HTML manually (though manual cleanup will always be superior).

`auto_cleanup_keep` = None
Specify elements that the auto cleanup algorithm should never remove The syntax is a XPath expression. For example:

```
auto_cleanup_keep = '//div[@id="article-image"]' will keep all divs with
                                                 id="article-image"
auto_cleanup_keep = '//*[@class="important"]' will keep all elements
                                               with class="important"
auto_cleanup_keep = '//div[@id="article-image"]|//span[@class="important"]'
                  will keep all divs with id="article-image" and spans
                  with class="important"
```

`center_navbar` = True
If True the navigation bar is center aligned, otherwise it is left aligned

[177] http://www.crummy.com/software/BeautifulSoup/documentation.html
[178] http://www.crummy.com/software/BeautifulSoup/documentation.html
[179] http://www.crummy.com/software/BeautifulSoup/documentation.html

conversion_options = {}

Recipe specific options to control the conversion of the downloaded content into an e-book. These will override any user or plugin specified values, so only use if absolutely necessary. For example:

```
conversion_options = {
  'base_font_size'   : 16,
  'tags'             : 'mytag1,mytag2',
  'title'            : 'My Title',
  'linearize_tables' : True,
}
```

cover_margins = (0, 0, '#ffffff')

By default, the cover image returned by get_cover_url() will be used as the cover for the periodical. Overriding this in your recipe instructs calibre to render the downloaded cover into a frame whose width and height are expressed as a percentage of the downloaded cover. cover_margins = (10, 15, '#ffffff') pads the cover with a white margin 10px on the left and right, 15px on the top and bottom. Color names defined at http://www.imagemagick.org/script/color.php Note that for some reason, white does not always work on windows. Use #ffffff instead

delay = 0

Delay between consecutive downloads in seconds. The argument may be a floating point number to indicate a more precise time.

description = u''

A couple of lines that describe the content this recipe downloads. This will be used primarily in a GUI that presents a list of recipes.

encoding = None

Specify an override encoding for sites that have an incorrect charset specification. The most common being specifying `latin1` and using `cp1252`. If None, try to detect the encoding. If it is a callable, the callable is called with two arguments: The recipe object and the source to be decoded. It must return the decoded source.

extra_css = None

Specify any extra *CSS* that should be added to downloaded *HTML* files It will be inserted into *<style>* tags, just before the closing *</head>* tag thereby overriding all *CSS* except that which is declared using the style attribute on individual *HTML* tags. For example:

```
extra_css = '.heading { font: serif x-large }'
```

feeds = None

List of feeds to download Can be either `[url1, url2, ...]` or `[('title1', url1), ('title2', url2),...]`

filter_regexps = []

List of regular expressions that determines which links to ignore If empty it is ignored. Used only if is_link_wanted is not implemented. For example:

```
filter_regexps = [r'ads\.doubleclick\.net']
```

will remove all URLs that have *ads.doubleclick.net* in them.

Only one of `BasicNewsRecipe.match_regexps` (page 366) or `BasicNewsRecipe.filter_regexps` (page 365) should be defined.

ignore_duplicate_articles = None

Ignore duplicates of articles that are present in more than one section. A duplicate article is an article that has the same title and/or URL. To ignore articles with the same title, set this to: ignore_duplicate_articles = {'title'} To use URLs instead, set it to: ignore_duplicate_articles = {'url'} To match on title or URL, set it to: ignore_duplicate_articles = {'title', 'url'}

keep_only_tags = []
Keep only the specified tags and their children. For the format for specifying a tag see BasicNewsRecipe.remove_tags (page 367). If this list is not empty, then the <*body*> tag will be emptied and re-filled with the tags that match the entries in this list. For example:

```
keep_only_tags = [dict(id=['content', 'heading'])]
```

will keep only tags that have an *id* attribute of *"content"* or *"heading"*.

language = 'und'
The language that the news is in. Must be an ISO-639 code either two or three characters long

masthead_url = None
By default, calibre will use a default image for the masthead (Kindle only). Override this in your recipe to provide a url to use as a masthead.

match_regexps = []
List of regular expressions that determines which links to follow If empty, it is ignored. Used only if is_link_wanted is not implemented. For example:

```
match_regexps = [r'page=[0-9]+']
```

will match all URLs that have *page=some number* in them.

Only one of BasicNewsRecipe.match_regexps (page 366) or BasicNewsRecipe.filter_regexps (page 365) should be defined.

max_articles_per_feed = 100
Maximum number of articles to download from each feed. This is primarily useful for feeds that don't have article dates. For most feeds, you should use BasicNewsRecipe.oldest_article (page 366)

needs_subscription = False
If True the GUI will ask the user for a username and password to use while downloading If set to "optional" the use of a username and password becomes optional

no_stylesheets = False
Convenient flag to disable loading of stylesheets for websites that have overly complex stylesheets unsuitable for conversion to ebooks formats If True stylesheets are not downloaded and processed

oldest_article = 7.0
Oldest article to download from this news source. In days.

preprocess_regexps = []
List of *regexp* substitution rules to run on the downloaded *HTML*. Each element of the list should be a two element tuple. The first element of the tuple should be a compiled regular expression and the second a callable that takes a single match object and returns a string to replace the match. For example:

```
preprocess_regexps = [
    (re.compile(r'<!--Article ends here-->.*</body>', re.DOTALL|re.IGNORECASE),
     lambda match: '</body>'),
]
```

will remove everythong from <*!–Article ends here–>* to <*/body*>.

publication_type = 'unknown'
Publication type Set to newspaper, magazine or blog. If set to None, no publication type metadata will be written to the opf file.

recipe_disabled = None
Set to a non empty string to disable this recipe The string will be used as the disabled message

recursions = 0
Number of levels of links to follow on article webpages

remove_attributes = []
List of attributes to remove from all tags For example:

```
remove_attributes = ['style', 'font']
```

remove_empty_feeds = False
If True empty feeds are removed from the output. This option has no effect if parse_index is overridden in the sub class. It is meant only for recipes that return a list of feeds using *feeds* or `get_feeds()` (page 361). It is also used if you use the ignore_duplicate_articles option.

remove_javascript = True
Convenient flag to strip all javascript tags from the downloaded HTML

remove_tags = []
List of tags to be removed. Specified tags are removed from downloaded HTML. A tag is specified as a dictionary of the form:

```
{
 name       : 'tag name',   #e.g. 'div'
 attrs      : a dictionary, #e.g. {class: 'advertisment'}
}
```

All keys are optional. For a full explanantion of the search criteria, see Beautiful Soup[180] A common example:

```
remove_tags = [dict(name='div', attrs={'class':'advert'})]
```

This will remove all <*div class="advert"*> tags and all their children from the downloaded *HTML*.

remove_tags_after = None
Remove all tags that occur after the specified tag. For the format for specifying a tag see `BasicNewsRecipe.remove_tags` (page 367). For example:

```
remove_tags_after = [dict(id='content')]
```

will remove all tags after the first element with *id="content"*.

remove_tags_before = None
Remove all tags that occur before the specified tag. For the format for specifying a tag see `BasicNewsRecipe.remove_tags` (page 367). For example:

```
remove_tags_before = dict(id='content')
```

will remove all tags before the first element with *id="content"*.

requires_version = (0, 6, 0)
Minimum calibre version needed to use this recipe

reverse_article_order = False
Reverse the order of articles in each feed

simultaneous_downloads = 5
Number of simultaneous downloads. Set to 1 if the server is picky. Automatically reduced to 1 if `BasicNewsRecipe.delay` (page 365) > 0

summary_length = 500
Max number of characters in the short description

[180] http://www.crummy.com/software/BeautifulSoup/documentation.html#Thebasicfindmethod:findAll(name,attrs,recursive,text,limit,**kwargs)

template_css = u'\n .article_date {\n color: gray; font-family: monospace;\n }\n\n .article_description {\n text-indent:
The CSS that is used to style the templates, i.e., the navigation bars and the Tables of Contents. Rather than overriding this variable, you should use *extra_css* in your recipe to customize look and feel.

timefmt = ' [%a, %d %b %Y]'
The format string for the date shown on the first page. By default: Day_Name, Day_Number Month_Name Year

timeout = 120.0
Timeout for fetching files from server in seconds

title = u'Unknown News Source'
The title to use for the ebook

use_embedded_content = None
Normally we try to guess if a feed has full articles embedded in it based on the length of the embedded content. If *None*, then the default guessing is used. If *True* then the we always assume the feeds has embedded content and if *False* we always assume the feed does not have embedded content.

use_javascript_to_login = False
If you set this True, then calibre will use javascript to login to the website. This is needed for some websites that require the use of javascript to login. If you set this to True you must implement the `javascript_login()` (page 362) method, to do the actual logging in.

Managing subgroups of books, for example "genre"

Some people wish to organize the books in their library into subgroups, similar to subfolders. The most commonly provided reason is to create genre hierarchies, but there are many others. One user asked for a way to organize textbooks by subject and course number. Another wanted to keep track of gifts by subject and recipient. This tutorial will use the genre example for the rest of this post.

Before going on, please note that we are not talking about folders on the hard disk. Subgroups are not file folders. Books will not be copied anywhere. Calibre's library file structure is not affected. Instead, we are presenting a way to organize and display subgroups of books within a calibre library.

- Setup (page 370)
- Searching (page 372)
- Restrictions (page 372)
- Useful Template Functions (page 373)

The commonly-provided requirements for subgroups such as genres are:

- A subgroup (e.g., a genre) must contain (point to) books, not categories of books. This is what distinguishes subgroups from calibre user categories.
- A book can be in multiple subgroups (genres). This distinguishes subgroups from physical file folders.
- Subgroups (genres) must form a hierarchy; subgroups can contain subgroups.

Tags give you the first two. If you tag a book with the genre then you can use the tag browser (or search) for find the books with that genre, giving you the first. Many books can have the same tag(s), giving you the second. The problem is that tags don't satisfy the third requirement. They don't provide a hierarchy.



Calibre's hierarchy feature gives you the third, the ability to see the genres in a 'tree' and the ability to easily search for books in genre or sub-genre. For example, assume that your genre structure is similar to the following:

```
Genre
    . History
    .. Japanese
    .. Military
    .. Roman
    . Mysteries
    .. English
    .. Vampire
    . Science Fiction
    .. Alternate History
    .. Military
    .. Space Opera
    . Thrillers
    .. Crime
    .. Horror
    etc.
```

By using the hierarchy feature, you can see these genres in the tag browser in tree form, as shown in the screen image. In this example the outermost level (Genre) is a custom column that contains the genres. Genres containing sub-genres appear with a small triangle next to them. Clicking on that triangle will open the item and show the sub-genres, as you can see with History and Science Fiction.

Clicking on a genre can search for all books with that genre or children of that genre. For example, clicking on Science Fiction can give all three of the child genres, Alternate History, Military, and Space Opera. Clicking on Alternate History will give books in that genre, ignoring those in Military and Space Opera. Of course, a book can have multiple genres. If a book has both Space Opera and Military genres, then you will see that book if you click on either genre. Searching is discussed in more detail below.

Another thing you can see from the image is that the genre Military appears twice, once under History and once under Science Fiction. Because the genres are in a hierarchy, these are two separate genres. A book can be in one, the other, or (doubtfully in this case) both. For example, the books in Winston Churchill's "The Second World War" could be in

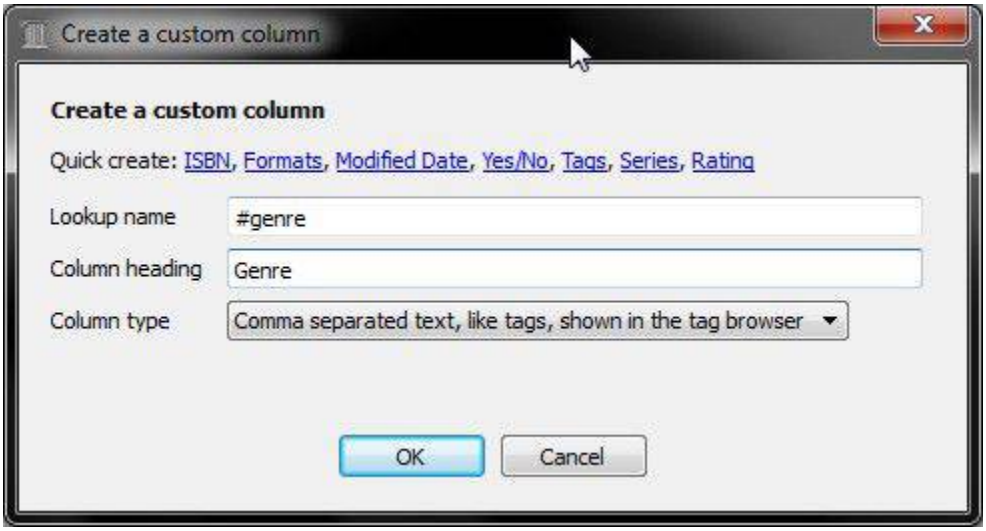
"History.Military". David Weber's Honor Harrington books could be in "Science Fiction.Military", and for that matter also in "Science Fiction.Space Opera."

Once a genre exists, that is at least one book has that genre, you can easily apply it to other books by dragging the books from the library view onto the genre you want the books to have. You can also apply genres in the metadata editors; more on this below.

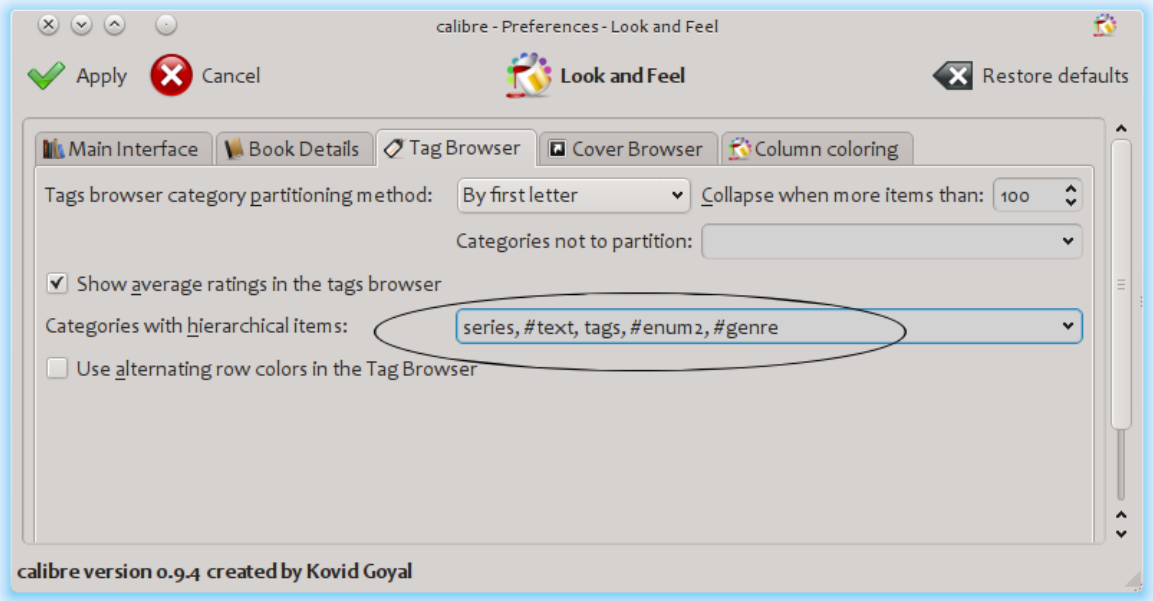
Setup

By now, your question might be "How was all of this up?" There are three steps: 1) create the custom column, 2) tell calibre that the new column is to be treated as a hierarchy, and 3) add genres.

You create the custom column in the usual way, using Preferences -> Add your own columns. This example uses "#genre" as the lookup name and "Genre" as the column heading. The column type is "Comma-separated text, like tags, shown in the tag browser."

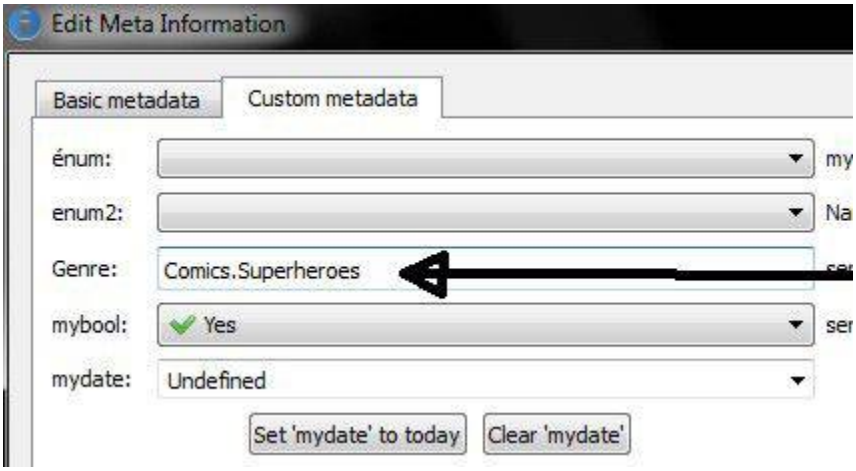


Then after restarting calibre, you must tell calibre that the column is to be treated as a hierarchy. Go to Preferences -> Look and Feel -> Tag Browser and enter the lookup name "#genre" into the "Categories with hierarchical items" box. Press Apply, and you are done with setting up.



At the point there are no genres in the column. We are left with the last step: how to apply a genre to a book. A genre does not exist in calibre until it appears on at least one book. To learn how to apply a genre for the first time, we must go into some detail about what a genre looks like in the metadata for a book.

A hierarchy of 'things' is built by creating an item consisting of phrases separated by periods. Continuing the genre example, these items would "History.Military", "Mysteries.Vampire", "Science Fiction.Space Opera", etc. Thus to create a new genre, you pick a book that should have that genre, edit its metadata, and enter the new genre into the column you created. Continuing our example, if you want to assign a new genre "Comics" with a sub-genre "Superheroes" to a book, you would 'edit metadata' for that (comic) book, choose the Custom metadata tab, and then enter "Comics.Superheroes" as shown in the following (ignore the other custom columns):



After doing the above, you see in the tag browser:



From here on, to apply this new genre to a book (a comic book, presumably), you can either drag the book onto the genre, or add it to the book using edit metadata in exactly the same way as done above.

Searching



The easiest way to search for genres is using the tag browser, clicking on the genre you wish to see. Clicking on a genre with children will show you books with that genre and all child genres. However, this might bring up a question. Just because a genre has children doesn't mean that it isn't a genre in its own right. For example, a book can have the genre "History" but not "History.Military". How do you search for books with only "History"?

The tag browser search mechanism knows if an item has children. If it does, clicking on the item cycles through 5 searches instead of the normal three. The first is the normal green plus, which shows you books with that genre only (e.g., History). The second is a doubled plus (shown above), which shows you books with that genre and all sub-genres (e.g., History and History.Military). The third is the normal red minus, which shows you books without that exact genre. The fourth is a doubled minus, which shows you books without that genre or sub-genres. The fifth is back to the beginning, no mark, meaning no search.

Restrictions

If you search for a genre then create a saved search for it, you can use the 'restrict to' box to create a virtual library of books with that genre. This is useful if you want to do other searches within the genre or to manage/update metadata for books in the genre. Continuing our example, you can create a saved search named 'History.Japanese' by first clicking on the genre Japanese in the tag browser to get a search into the search box, entering History.Japanese into the saved search box, then pushing the "save search" button (the green box with the white plus, on the right-hand side).



After creating the saved search, you can use it as a restriction.



Useful Template Functions

You might want to use the genre information in a template, such as with save to disk or send to device. The question might then be "How do I get the outermost genre name or names?" A calibre template function, subitems, is provided to make doing this easier.

For example, assume you want to add the outermost genre level to the save-to-disk template to make genre folders, as in "History/The Gathering Storm - Churchill, Winston". To do this, you must extract the first level of the hierarchy and add it to the front along with a slash to indicate that it should make a folder. The template below accomplishes this:

```
{#genre:subitems(0,1)||/}{title} - {authors}
```

See *The template language* (page 375) for more information templates and the `subitems()` function.

XPath Tutorial

In this tutorial, you will be given a gentle introduction to XPath[181], a query language that can be used to select arbitrary parts of HTML[182] documents in calibre. XPath is a widely used standard, and googling it will yield a ton of information. This tutorial, however, focuses on using XPath for ebook related tasks like finding chapter headings in an unstructured HTML document.

[181] http://en.wikipedia.org/wiki/XPath
[182] http://en.wikipedia.org/wiki/HTML

Contents

- Selecting by tagname (page 374)
- Selecting by attributes (page 374)
- Selecting by tag content (page 375)
- Sample ebook (page 375)
- XPath built-in functions (page 375)

Selecting by tagname

The simplest form of selection is to select tags by name. For example, suppose you want to select all the `<h2>` tags in a document. The XPath query for this is simply:

```
//h:h2        (Selects all <h2> tags)
```

The prefix *//* means *search at any level of the document*. Now suppose you want to search for `<span>` tags that are inside `<a>` tags. That can be achieved with:

```
//h:a/h:span    (Selects <span> tags inside <a> tags)
```

If you want to search for tags at a particular level in the document, change the prefix:

```
/h:body/h:div/h:p (Selects <p> tags that are children of <div> tags that are
             children of the <body> tag)
```

This will match only `<p>A very short ebook to demonstrate the use of XPath.</p>` in the Sample ebook (page 375) but not any of the other `<p>` tags. The `h:` prefix in the above examples is needed to match XHTML tags. This is because internally, calibre represents all content as XHTML. In XHTML tags have a *namespace*, and `h:` is the namespace prefix for HTML tags.

Now suppose you want to select both `<h1>` and `<h2>` tags. To do that, we need a XPath construct called *predicate*. A *predicate* is simply a test that is used to select tags. Tests can be arbitrarily powerful and as this tutorial progresses, you will see more powerful examples. A predicate is created by enclosing the test expression in square brackets:

```
//*[name()='h1' or name()='h2']
```

There are several new features in this XPath expression. The first is the use of the wildcard `*`. It means *match any tag*. Now look at the test expression `name()='h1' or name()='h2'`. *name()* is an example of a *built-in function*. It simply evaluates to the name of the tag. So by using it, we can select tags whose names are either *h1* or *h2*. Note that the *name()* function ignores namespaces so that there is no need for the `h:` prefix. XPath has several useful built-in functions. A few more will be introduced in this tutorial.

Selecting by attributes

To select tags based on their attributes, the use of predicates is required:

```
//*[@style]              (Select all tags that have a style attribute)
//*[@class="chapter"]    (Select all tags that have class="chapter")
//h:h1[@class="bookTitle"] (Select all h1 tags that have class="bookTitle")
```

Here, the `@` operator refers to the attributes of the tag. You can use some of the XPath built-in functions (page 375) to perform more sophisticated matching on attribute values.

Selecting by tag content

Using XPath, you can even select tags based on the text they contain. The best way to do this is to use the power of *regular expressions* via the built-in function *re:test()*:

```
//h:h2[re:test(., 'chapter|section', 'i')] (Selects <h2> tags that contain the words chapter or
                                          section)
```

Here the `.` operator refers to the contents of the tag, just as the `@` operator referred to its attributes.

Sample ebook

```
<html>
    <head>
        <title>A very short ebook</title>
        <meta name="charset" value="utf-8" />
    </head>
    <body>
        <h1 class="bookTitle">A very short ebook</h1>
        <p style="text-align:right">Written by Kovid Goyal</p>
        <div class="introduction">
            <p>A very short ebook to demonstrate the use of XPath.</p>
        </div>

        <h2 class="chapter">Chapter One</h2>
        <p>This is a truly fascinating chapter.</p>

        <h2 class="chapter">Chapter Two</h2>
        <p>A worthy continuation of a fine tradition.</p>
    </body>
</html>
```

XPath built-in functions

name() The name of the current tag.

contains() `contains(s1, s2)` returns *true* if s1 contains s2.

re:test() `re:test(src, pattern, flags)` returns *true* if the string *src* matches the regular expression *pattern*. A particularly useful flag is `i`, it makes matching case insensitive. A good primer on the syntax for regular expressions can be found at regexp syntax[183]

The calibre template language

The calibre template language is used in various places. It is used to control the folder structure and file name when saving files from the calibre library to the disk or eBook reader. It is also used to define "virtual" columns that contain data from other columns and so on.

The basic template language is very simple, but has very powerful advanced features. The basic idea is that a template consists of text and names in curly brackets that are then replaced by the corresponding metadata from the book being processed. So, for example, the default template used for saving books to device in calibre is:

[183] http://docs.python.org/lib/re-syntax.html

```
{author_sort}/{title}/{title} - {authors}
```

For the book "The Foundation" by "Isaac Asimov" it will become:

```
Asimov, Isaac/The Foundation/The Foundation - Isaac Asimov
```

The slashes are text, which is put into the template where it appears. For example, if your template is:

```
{author_sort} Some Important Text {title}/{title} - {authors}
```

For the book "The Foundation" by "Isaac Asimov" it will become:

```
Asimov, Isaac Some Important Text The Foundation/The Foundation - Isaac Asimov
```

You can use all the various metadata fields available in calibre in a template, including any custom columns you have created yourself. To find out the template name for a column simply hover your mouse over the column header. Names for custom fields (columns you have created yourself) always have a # as the first character. For series type custom fields, there is always an additional field named `#seriesname_index` that becomes the series index for that series. So if you have a custom series field named `#myseries`, there will also be a field named `#myseries_index`.

In addition to the column based fields, you also can use:

```
{formats} - A list of formats available in the calibre library for a book
{identifiers:select(isbn)} - The ISBN number of the book
```

If a particular book does not have a particular piece of metadata, the field in the template is automatically removed for that book. Consider, for example:

```
{author_sort}/{series}/{title} {series_index}
```

If a book has a series, the template will produce:

```
Asimov, Isaac/Foundation/Second Foundation 3
```

and if a book does not have a series:

```
Asimov, Isaac/Second Foundation
```

(calibre automatically removes multiple slashes and leading or trailing spaces).

Advanced formatting

You can do more than just simple substitution with the templates. You can also conditionally include text and control how the substituted data is formatted.

First, conditionally including text. There are cases where you might want to have text appear in the output only if a field is not empty. A common case is `series` and `series_index`, where you want either nothing or the two values with a hyphen between them. Calibre handles this case using a special field syntax.

For example, assume you want to use the template:

```
{series} - {series_index} - {title}
```

If the book has no series, the answer will be `- - title`. Many people would rather the result be simply `title`, without the hyphens. To do this, use the extended syntax `{field:|prefix_text|suffix_text}`. When you use this syntax, if field has the value SERIES then the result will be `prefix_textSERIESsuffix_text`. If field has no value, then the result will be the empty string (nothing); the prefix and suffix are ignored. The prefix and suffix can contain blanks. **Do not use subtemplates ('{ ... }') or functions (see below) as the prefix or the suffix.**

Using this syntax, we can solve the above series problem with the template:

```
{series}{series_index:| - | - }{title}
```

The hyphens will be included only if the book has a series index, which it will have only if it has a series.

Notes: you must include the : character if you want to use a prefix or a suffix. You must either use no | characters or both of them; using one, as in `{field:| - }`, is not allowed. It is OK not to provide any text for one side or the other, such as in `{series:|| - }`. Using `{title:|||}` is the same as using `{title}`.

Second: formatting. Suppose you wanted to ensure that the series_index is always formatted as three digits with leading zeros. This would do the trick:

```
{series_index:0>3s} - Three digits with leading zeros
```

If instead of leading zeros you want leading spaces, use:

```
{series_index:>3s} - Three digits with leading spaces
```

For trailing zeros, use:

```
{series_index:0<3s} - Three digits with trailing zeros
```

If you use series indices with sub values (e.g., 1.1), you might want to ensure that the decimal points line up. For example, you might want the indices 1 and 2.5 to appear as 01.00 and 02.50 so that they will sort correctly. To do this, use:

```
{series_index:0>5.2f} - Five characters, consisting of two digits with leading zeros, a decimal point
```

If you want only the first two letters of the data, use:

```
{author_sort:.2} - Only the first two letter of the author sort name
```

The calibre template language comes from python and for more details on the syntax of these advanced formatting operations, look at the Python documentation[184].

Advanced features

Using templates in custom columns

There are sometimes cases where you want to display metadata that calibre does not normally display, or to display data in a way different from how calibre normally does. For example, you might want to display the ISBN, a field that calibre does not display. You can use custom columns for this by creating a column with the type 'column built from other columns' (hereafter called composite columns), and entering a template. Result: calibre will display a column showing the result of evaluating that template. To display the ISBN, create the column and enter `{identifiers:select(isbn)}` into the template box. To display a column containing the values of two series custom columns separated by a comma, use `{#series1:||,}{#series2}`.

Composite columns can use any template option, including formatting.

You cannot change the data contained in a composite column. If you edit a composite column by double-clicking on any item, you will open the template for editing, not the underlying data. Editing the template on the GUI is a quick way of testing and changing composite columns.

[184] http://docs.python.org/library/string.html#format-string-syntax

Using functions in templates - single-function mode

Suppose you want to display the value of a field in upper case, when that field is normally in title case. You can do this (and many more things) using the functions available for templates. For example, to display the title in upper case, use `{title:uppercase()}`. To display it in title case, use `{title:titlecase()}`.

Function references appear in the format part, going after the `:` and before the first `|` or the closing `}`. If you have both a format and a function reference, the function comes after another `:`. Functions must always end with `()`. Some functions take extra values (arguments), and these go inside the `()`.

Functions are always applied before format specifications. See further down for an example of using both a format and a function, where this order is demonstrated.

The syntax for using functions is `{field:function(arguments)}`, or `{field:function(arguments)|prefix|suffix}`. Arguments are separated by commas. Commas inside arguments must be preceeded by a backslash ('\'). The last (or only) argument cannot contain a closing parenthesis (')'). Functions return the value of the field used in the template, suitably modified.

Important: If you have programming experience, please note that the syntax in this mode (single function) is not what you might expect. Strings are not quoted. Spaces are significant. All arguments must be constants; there is no sub-evaluation. **Do not use subtemplates ('{ ... }') as function arguments.** Instead, use *template program mode* (page 380) and *general program mode* (page 398).

Many functions use regular expressions. In all cases, regular expression matching is case-insensitive.

The functions available are listed below. Note that the definitive documentation for functions is available in the section *Function classification* (page 385):

- `lowercase()` – return value of the field in lower case.
- `uppercase()` – return the value of the field in upper case.
- `titlecase()` – return the value of the field in title case.
- `capitalize()` – return the value with the first letter upper case and the rest lower case.
- `contains(pattern, text if match, text if not match)` – checks if field contains matches for the regular expression *pattern*. Returns *text if match* if matches are found, otherwise it returns *text if no match*.
- `count(separator)` – interprets the value as a list of items separated by *separator*, returning the number of items in the list. Most lists use a comma as the separator, but authors uses an ampersand. Examples: *{tags:count(,)}*, *{authors:count(&)}*
- `format_number(template)` – interprets the value as a number and format that number using a python formatting template such as "{0:5.2f}" or "{0:,d}" or "${0:5,.2f}". The field_name part of the template must be a 0 (zero) (the "{0:" in the above examples). See the template language and python documentation for more examples. Returns the empty string if formatting fails.
- `human_readable()` – expects the value to be a number and returns a string representing that number in KB, MB, GB, etc.
- `ifempty(text)` – if the field is not empty, return the value of the field. Otherwise return *text*.
- `in_list(separator, pattern, found_val, not_found_val)` – interpret the field as a list of items separated by *separator*, comparing the *pattern* against each value in the list. If the pattern matches a value, return *found_val*, otherwise return *not_found_val*.
- `language_codes(lang_strings)` – return the language codes for the strings passed in *lang_strings*. The strings must be in the language of the current locale. *Lang_strings* is a comma-separated list.

- `language_strings(lang_codes, localize)` – return the strings for the language codes passed in *lang_codes*. If *localize* is zero, return the strings in English. If localize is not zero, return the strings in the language of the current locale. *Lang_codes* is a comma-separated list.
- `list_item(index, separator)` – interpret the field as a list of items separated by *separator*, returning the *index'th item. The first item is number zero. The last item can be returned using 'list_item(-1,separator)*. If the item is not in the list, then the empty value is returned. The separator has the same meaning as in the *count* function.
- `re(pattern, replacement)` – return the field after applying the regular expression. All instances of *pattern* are replaced with *replacement*. As in all of calibre, these are python-compatible regular expressions.
- `shorten(left chars, middle text, right chars)` – Return a shortened version of the field, consisting of *left chars* characters from the beginning of the field, followed by *middle text*, followed by *right chars* characters from the end of the string. *Left chars* and *right chars* must be integers. For example, assume the title of the book is *Ancient English Laws in the Times of Ivanhoe*, and you want it to fit in a space of at most 15 characters. If you use `{title:shorten(9,-,5)}`, the result will be *Ancient E-nhoe*. If the field's length is less than `left chars` + `right chars` + the length of `middle text`, then the field will be used intact. For example, the title *The Dome* would not be changed.
- `swap_around_comma(val)` `` -- given a value of the form ``B, A``, return `A B`. This is most useful for converting names in LN, FN format to FN LN. If there is no comma, the function returns val unchanged.
- `switch(pattern, value, pattern, value, ..., else_value)` – for each `pattern, value` pair, checks if the field matches the regular expression `pattern` and if so, returns that `value`. If no `pattern` matches, then `else_value` is returned. You can have as many `pattern, value` pairs as you want.
- `lookup(pattern, field, pattern, field, ..., else_field)` – like switch, except the arguments are field (metadata) names, not text. The value of the appropriate field will be fetched and used. Note that because composite columns are fields, you can use this function in one composite field to use the value of some other composite field. This is extremely useful when constructing variable save paths (more later).
- `select(key)` – interpret the field as a comma-separated list of items, with the items being of the form "id:value". Find the pair with the id equal to key, and return the corresponding value. This function is particularly useful for extracting a value such as an isbn from the set of identifiers for a book.
- `str_in_list(val, separator, string, found_val, not_found_val)` – treat val as a list of items separated by separator, comparing the string against each value in the list. If the string matches a value, return found_val, otherwise return not_found_val. If the string contains separators, then it is also treated as a list and each value is checked.
- `subitems(val, start_index, end_index)` – This function is used to break apart lists of tag-like hierarchical items such as genres. It interprets the value as a comma-separated list of tag-like items, where each item is a period-separated list. Returns a new list made by first finding all the period-separated tag-like items, then for each such item extracting the components from *start_index* to *end_index*, then combining the results back together. The first component in a period-separated list has an index of zero. If an index is negative, then it counts from the end of the list. As a special case, an end_index of zero is assumed to be the length of the list. Examples:

```
Assuming a #genre column containing "A.B.C":
    {#genre:subitems(0,1)} returns "A"
    {#genre:subitems(0,2)} returns "A.B"
    {#genre:subitems(1,0)} returns "B.C"
Assuming a #genre column containing "A.B.C, D.E":
    {#genre:subitems(0,1)} returns "A, D"
    {#genre:subitems(0,2)} returns "A.B, D.E"
```

- `sublist(val, start_index, end_index, separator)` – interpret the value as a list of items separated by *separator*, returning a new list made from the items from *start_index'to 'end_index*. The first item is number zero. If an index is negative, then it counts from the end of the list. As a special case, an end_index of zero is assumed to be the length of the list. Examples assuming that the tags column (which is comma-separated) contains "A, B ,C":

```
{tags:sublist(0,1,\,)} returns "A"
{tags:sublist(-1,0,\,)} returns "C"
{tags:sublist(0,-1,\,)} returns "A, B"
```

- `test(text if not empty, text if empty)` – return *text if not empty* if the field is not empty, otherwise return *text if empty*.

Now, what about using functions and formatting in the same field. Suppose you have an integer custom column called `#myint` that you want to see with leading zeros, as in `003`. To do this, you would use a format of `0>3s`. However, by default, if a number (integer or float) equals zero then the field produces the empty value, so zero values will produce nothing, not `000`. If you really want to see `000` values, then you use both the format string and the `ifempty` function to change the empty value back to a zero. The field reference would be:

```
{#myint:0>3s:ifempty(0)}
```

Note that you can use the prefix and suffix as well. If you want the number to appear as `[003]` or `[000]`, then use the field:

```
{#myint:0>3s:ifempty(0)|[|]}
```

Using functions in templates - template program mode

The template language program mode differs from single-function mode in that it permits you to write template expressions that refer to other metadata fields, modify values, and do arithmetic. It is a reasonably complete programming language.

You can use the functions documented above in template program mode. See below for details.

Beginning with an example, assume that you want your template to show the series for a book if it has one, otherwise show the value of a custom field #genre. You cannot do this in the basic language because you cannot make reference to another metadata field within a template expression. In program mode, you can. The following expression works:

```
{#series:'ifempty($, field('#genre'))'}
```

The example shows several things:

- program mode is used if the expression begins with `:'` and ends with `'`. Anything else is assumed to be single-function.
- the variable `$` stands for the field the expression is operating upon, `#series` in this case.
- functions must be given all their arguments. There is no default value. For example, the standard built-in functions must be given an additional initial parameter indicating the source field, which is a significant difference from single-function mode.
- white space is ignored and can be used anywhere within the expression.
- constant strings are enclosed in matching quotes, either `'` or `"`.

The language is similar to `functional` languages in that it is built almost entirely from functions. A statement is a function. An expression is a function. Constants and identifiers can be thought of as functions returning the value indicated by the constant or stored in the identifier.

The syntax of the language is shown by the following grammar:

```
constant   ::= " string " | ' string ' | number
identifier ::= sequence of letters or ``_`` characters
function   ::= identifier ( statement [ , statement ]* )
expression ::= identifier | constant | function | assignment
assignment ::= identifier '=' expression
statement  ::= expression [ ; expression ]*
program    ::= statement
```

Comments are lines with a '#' character at the beginning of the line.

An `expression` always has a value, either the value of the constant, the value contained in the identifier, or the value returned by a function. The value of a `statement` is the value of the last expression in the sequence of statements. As such, the value of the program (statement):

```
1; 2; 'foobar'; 3
```

is 3.

Another example of a complex but rather silly program might help make things clearer:

```
{series_index:'
    substr(
        strcat($, '->',
            cmp(divide($, 2), 1,
                assign(c, 1); substr('lt123', c, 0),
                'eq', 'gt')),
        0, 6)
   '| prefix | suffix}
```

This program does the following:

- specify that the field being looked at is series_index. This sets the value of the variable `$`.
- calls the `substr` function, which takes 3 parameters `(str, start, end)`. It returns a string formed by extracting the start through end characters from string, zero-based (the first character is character zero). In this case the string will be computed by the `strcat` function, the start is 0, and the end is 6. In this case it will return the first 6 characters of the string returned by `strcat`, which must be evaluated before substr can return.
- calls the `strcat` (string concatenation) function. Strcat accepts 1 or more arguments, and returns a string formed by concatenating all the values. In this case there are three arguments. The first parameter is the value in `$`, which here is the value of `series_index`. The second paremeter is the constant string `'->'`. The third parameter is the value returned by the `cmp` function, which must be fully evaluated before `strcat` can return.
- The `cmp` function takes 5 arguments `(x, y, lt, eq, gt)`. It compares x and y and returns the third argument `lt` if x < y, the fourth argument `eq` if x == y, and the fifth argument `gt` if x > y. As with all functions, all of the parameters can be statements. In this case the first parameter (the value for `x`) is the result of dividing the series_index by 2. The second parameter `y` is the constant `1`. The third parameter `lt` is a statement (more later). The fourth parameter `eq` is the constant string `'eq'`. The fifth parameter is the constant string `'gt'`.
- The third parameter (the one for `lt`) is a statement, or a sequence of expressions. Remember that a statement (a sequence of semicolon-separated expressions) is also an expression, returning the value of the last expression in the list. In this case, the program first assigns the value `1` to a local variable `c`, then returns a substring made by extracting the c'th character to the end. Since c always contains the constant `1`, the substring will return the second through end'th characters, or `'t123'`.
- Once the statement providing the value to the third parameter is executed, `cmp` can return a value. At that point, `strcat` can return a value, then ``substr`` can return a value. The program then terminates.

For various values of series_index, the program returns:

- series_index == undefined, result = `prefix ->t123 suffix`

- series_index == 0.5, result = `prefix 0.50-> suffix`
- series_index == 1, result = `prefix 1->t12 suffix`
- series_index == 2, result = `prefix 2->eq suffix`
- series_index == 3, result = `prefix 3->gt suffix`

All the functions listed under single-function mode can be used in program mode. To do so, you must supply the value that the function is to act upon as the first parameter, in addition to the parameters documented above. For example, in program mode the parameters of the *test* function are `test(x, text_if_not_empty, text_if_empty)`. The *x* parameter, which is the value to be tested, will almost always be a variable or a function call, often *field()*.

The following functions are available in addition to those described in single-function mode. Remember from the example above that the single-function mode functions require an additional first parameter specifying the field to operate on. With the exception of the `id` parameter of assign, all parameters can be statements (sequences of expressions). Note that the definitive documentation for functions is available in the section *Function classification* (page 385):

- `and(value, value, ...)` – returns the string "1" if all values are not empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.
- `add(x, y)` – returns x + y. Throws an exception if either x or y are not numbers.
- `assign(id, val)` – assigns val to id, then returns val. id must be an identifier, not an expression
- `approximate_formats()` – return a comma-separated list of formats that at one point were associated with the book. There is no guarantee that the list is correct, although it probably is. This function can be called in template program mode using the template `{:'approximate_formats()'}`. Note that format names are always uppercase, as in EPUB.
- `booksize()` – returns the value of the calibre 'size' field. Returns '' if there are no formats.
- `cmp(x, y, lt, eq, gt)` – compares x and y after converting both to numbers. Returns `lt` if x < y. Returns `eq` if x == y. Otherwise returns `gt`.
- `current_library_name() -- `` return the last name on the path to the current calibre library. This function can be called in template program mode using the template ``{:'current_library_name()'}`.
- `current_library_path() -- `` eturn the path to the current calibre library. This function can be called in template program mode using the template ``{:'current_library_path()'}`..
- `days_between(date1, date2)` – return the number of days between `date1` and `date2`. The number is positive if `date1` is greater than `date2`, otherwise negative. If either `date1` or `date2` are not dates, the function returns the empty string.
- `divide(x, y)` – returns x / y. Throws an exception if either x or y are not numbers.
- `eval(string)` – evaluates the string as a program, passing the local variables (those `assign`ed to). This permits using the template processor to construct complex results from local variables. Because the *{* and *}* characters are special, you must use *[[* for the *{* character and *]]* for the '}' character; they are converted automatically. Note also that prefixes and suffixes (the *|prefix|suffix* syntax) cannot be used in the argument to this function when using template program mode.
- `field(name)` – returns the metadata field named by `name`.
- `first_non_empty(value, value, ...)` – returns the first value that is not empty. If all values are empty, then the empty value is returned. You can have as many values as you want.

- `format_date(x, date_format)` – format_date(val, format_string) – format the value, which must be a date field, using the format_string, returning a string. The formatting codes are:

```
d    : the day as number without a leading zero (1 to 31)
dd   : the day as number with a leading zero (01 to 31)
ddd  : the abbreviated localized day name (e.g. "Mon" to "Sun").
dddd : the long localized day name (e.g. "Monday" to "Sunday").
M    : the month as number without a leading zero (1 to 12).
MM   : the month as number with a leading zero (01 to 12)
MMM  : the abbreviated localized month name (e.g. "Jan" to "Dec").
MMMM : the long localized month name (e.g. "January" to "December").
yy   : the year as two digit number (00 to 99).
yyyy : the year as four digit number.
h    : the hours without a leading 0 (0 to 11 or 0 to 23, depending on am/pm)
hh   : the hours with a leading 0 (00 to 11 or 00 to 23, depending on am/pm)
m    : the minutes without a leading 0 (0 to 59)
mm   : the minutes with a leading 0 (00 to 59)
s    : the seconds without a leading 0 (0 to 59)
ss   : the seconds with a leading 0 (00 to 59)
ap   : use a 12-hour clock instead of a 24-hour clock, with 'ap' replaced by the localized s
AP   : use a 12-hour clock instead of a 24-hour clock, with 'AP' replaced by the localized s
iso  : the date with time and timezone. Must be the only format present.
```

You might get unexpected results if the date you are formatting contains localized month names, which can happen if you changed the format tweaks to contain MMMM. In this case, instead of using something like `{pubdate:format_date(yyyy)}`, write the template using template program mode as in `{:'format_date(raw_field('pubdate'),'yyyy')'}`.

- finish_formatting(val, fmt, prefix, suffix) – apply the format, prefix, and suffix to a value in the same way as done in a template like `{series_index:05.2f| - |- }`. This function is provided to ease conversion of complex single-function- or template-program-mode templates to *general program mode* (page 398) (see below) to take advantage of GPM template compilation. For example, the following program produces the same output as the above template:

```
program: finish_formatting(field("series_index"), "05.2f", " - ", " - ")
```

Another example: for the template `{series:re(([^\s])[^\s]+(\s|$),\1)}{series_index:0>2s| - | - }{title}` use:

```
program:
    strcat(
        re(field('series'), '([^\s])[^\s]+(\s|$)', '\1'),
        finish_formatting(field('series_index'), '0>2s', ' - ', ' - '),
        field('title')
    )
```

- `formats_modtimes(date_format)` – return a comma-separated list of colon_separated items representing modification times for the formats of a book. The date_format parameter specifies how the date is to be formatted. See the date_format function for details. You can use the select function to get the mod time for a specific format. Note that format names are always uppercase, as in EPUB.
- `formats_paths()` – return a comma-separated list of colon_separated items representing full path to the formats of a book. You can use the select function to get the path for a specific format. Note that format names are always uppercase, as in EPUB.
- `formats_sizes()` – return a comma-separated list of colon_separated items representing sizes in bytes of the formats of a book. You can use the select function to get the size for a specific format. Note that format names are always uppercase, as in EPUB.

- `has_cover()` – return `Yes` if the book has a cover, otherwise return the empty string
- `not(value)` – returns the string "1" if the value is empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.
- `list_difference(list1, list2, separator)` – return a list made by removing from *list1* any item found in *list2*, using a case-insensitive compare. The items in *list1* and *list2* are separated by separator, as are the items in the returned list.
- `list_equals(list1, sep1, list2, sep2, yes_val, no_val)` – return *yes_val* if *list1* and *list2* contain the same items, otherwise return *no_val*. The items are determined by splitting each list using the appropriate separator character (*sep1* or *sep2*). The order of items in the lists is not relevant. The compare is case insensitive.
- `list_intersection(list1, list2, separator)` – return a list made by removing from *list1* any item not found in *list2*, using a case-insensitive compare. The items in *list1* and *list2* are separated by separator, as are the items in the returned list.
- `list_re(src_list, separator, search_re, opt_replace)` – Construct a list by first separating *src_list* into items using the *separator* character. For each item in the list, check if it matches *search_re*. If it does, then add it to the list to be returned. If *opt_replace* is not the empty string, then apply the replacement before adding the item to the returned list.
- `list_sort(list, direction, separator)` – return list sorted using a case-insensitive sort. If *direction* is zero, the list is sorted ascending, otherwise descending. The list items are separated by separator, as are the items in the returned list.
- `list_union(list1, list2, separator)` – return a list made by merging the items in list1 and list2, removing duplicate items using a case-insensitive compare. If items differ in case, the one in list1 is used. The items in list1 and list2 are separated by separator, as are the items in the returned list.
- `multiply(x, y)` – returns x * y. Throws an exception if either x or y are not numbers.
- `ondevice()` – return the string "Yes" if ondevice is set, otherwise return the empty string
- `or(value, value, ...)` – returns the string "1" if any value is not empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.
- `print(a, b, ...)` – prints the arguments to standard output. Unless you start calibre from the command line (`calibre-debug -g`), the output will go to a black hole.
- `raw_field(name)` – returns the metadata field named by name without applying any formatting.
- `series_sort()` – returns the series sort value.
- `strcat(a, b, ...)` – can take any number of arguments. Returns a string formed by concatenating all the arguments.
- `strcat_max(max, string1, prefix2, string2, ...)` – Returns a string formed by concatenating the arguments. The returned value is initialized to string1. *Prefix, string* pairs are added to the end of the value as long as the resulting string length is less than *max*. String1 is returned even if string1 is longer than max. You can pass as many *prefix, string* pairs as you wish.
- `strcmp(x, y, lt, eq, gt)` – does a case-insensitive comparison x and y as strings. Returns `lt` if x < y. Returns `eq` if x == y. Otherwise returns `gt`.
- `strlen(a)` – Returns the length of the string passed as the argument.
- `substr(str, start, end)` – returns the `start`'th through the `end`'th characters of `str`. The first character in `str` is the zero'th character. If end is negative, then it indicates that many

characters counting from the right. If end is zero, then it indicates the last character. For example, `substr('12345', 1, 0)` returns `'2345'`, and `substr('12345', 1, -1)` returns `'234'`.

- `subtract(x, y)` – returns x - y. Throws an exception if either x or y are not numbers.
- `today()` – return a date string for today. This value is designed for use in format_date or days_between, but can be manipulated like any other string. The date is in ISO format.
- `template(x)` – evaluates x as a template. The evaluation is done in its own context, meaning that variables are not shared between the caller and the template evaluation. Because the *{* and *}* characters are special, you must use *[[* for the *{* character and *]]* for the '}' character; they are converted automatically. For example, `template('[[title_sort]]') will evaluate the template ``{title_sort}` and return its value. Note also that prefixes and suffixes (the |*prefix*|*suffix* syntax) cannot be used in the argument to this function when using template program mode.

Function classification

Reference for all built-in template language functions Here, we document all the built-in functions available in the calibre template language. Every function is implemented as a class in python and you can click the source links to see the source code, in case the documentation is insufficient. The functions are arranged in logical groups by type.

- Arithmetic (page 404)
  - add(x, y) (page 404)
  - divide(x, y) (page 404)
  - multiply(x, y) (page 404)
  - subtract(x, y) (page 404)
- Boolean (page 404)
  - and(value, value, ...) (page 404)
  - not(value) (page 404)
  - or(value, value, ...) (page 404)
- Date functions (page 404)
  - days_between(date1, date2) (page 404)
  - today() (page 404)
- Formatting values (page 405)
  - finish_formatting(val, fmt, prefix, suffix) (page 405)
  - format_date(val, format_string) (page 405)
  - format_number(v, template) (page 405)
  - human_readable(v) (page 405)
- Get values from metadata (page 405)
  - approximate_formats() (page 405)
  - booksize() (page 405)
  - current_library_name() (page 406)
  - current_library_path() (page 406)
  - field(name) (page 406)
  - formats_modtimes(date_format) (page 406)
  - formats_paths() (page 406)
  - formats_sizes() (page 406)
  - has_cover() (page 406)
  - language_codes(lang_strings) (page 406)
  - language_strings(lang_codes, localize) (page 406)
  - ondevice() (page 407)
  - raw_field(name) (page 407)
  - series_sort() (page 407)
- If-then-else (page 407)
  - contains(val, pattern, text if match, text if not match) (page 407)
  - ifempty(val, text if empty) (page 407)
  - test(val, text if not empty, text if empty) (page 407)
- Iterating over values (page 407)
  - first_non_empty(value, value, ...) (page 407)
  - lookup(val, pattern, field, pattern, field, ..., else_field) (page 407)
  - switch(val, pattern, value, pattern, value, ..., else_value) (page 407)
- List lookup (page 408)
  - identifier_in_list(val, id, found_val, not_found_val) (page 408)
  - in_list(val, separator, pattern, found_val, not_found_val) (page 408)
  - list_item(val, index, separator) (page 408)
  - select(val, key) (page 408)
  - str_in_list(val, separator, string, found_val, not_found_val) (page 408)
- List manipulation (page 408)
  - count(val, separator) (page 408)
  - list_difference(list1, list2, separator) (page 408)
  - list_equals(list1, sep1, list2, sep2, yes_val, no_val) (page 409)
  - list_intersection(list1, list2, separator) (page 409)
  - list_re(src_list, separator, search_re, opt_replace) (page 409)
  - list_sort(list, direction, separator) (page 409)
  - list_union(list1, list2, separator) (page 409)
  - subitems(val, start_index, end_index) (page 409)
  - sublist(val, start_index, end_index, separator) (page 409)
- Other (page 410)
  - assign(id, val) (page 410)
  - print(a, b, ...) (page 410)
- Recursion (page 410)

Arithmetic

add(x, y)
class calibre.utils.formatter_functions.**BuiltinAdd**
add(x, y) – returns x + y. Throws an exception if either x or y are not numbers.

divide(x, y)
class calibre.utils.formatter_functions.**BuiltinDivide**
divide(x, y) – returns x / y. Throws an exception if either x or y are not numbers.

multiply(x, y)
class calibre.utils.formatter_functions.**BuiltinMultiply**
multiply(x, y) – returns x * y. Throws an exception if either x or y are not numbers.

subtract(x, y)
class calibre.utils.formatter_functions.**BuiltinSubtract**
subtract(x, y) – returns x - y. Throws an exception if either x or y are not numbers.

Boolean

and(value, value, ...)
class calibre.utils.formatter_functions.**BuiltinAnd**
and(value, value, ...) – returns the string "1" if all values are not empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.

not(value)
class calibre.utils.formatter_functions.**BuiltinNot**
not(value) – returns the string "1" if the value is empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.

or(value, value, ...)
class calibre.utils.formatter_functions.**BuiltinOr**
or(value, value, ...) – returns the string "1" if any value is not empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.

Date functions

days_between(date1, date2)
class calibre.utils.formatter_functions.**BuiltinDaysBetween**
days_between(date1, date2) – return the number of days between date1 and date2. The number is positive if date1 is greater than date2, otherwise negative. If either date1 or date2 are not dates, the function returns the empty string.

today()
class calibre.utils.formatter_functions.**BuiltinToday**
today() – return a date string for today. This value is designed for use in format_date or days_between, but can be manipulated like any other string. The date is in ISO format.

Formatting values

finish_formatting(val, fmt, prefix, suffix)

class calibre.utils.formatter_functions.**BuiltinFinishFormatting**

finish_formatting(val, fmt, prefix, suffix) – apply the format, prefix, and suffix to a value in the same way as done in a template like *{series_index:05.2f| - |- }*. For example, the following program produces the same output as the above template: program: finish_formatting(field("series_index"), "05.2f", " - ", " - ")

format_date(val, format_string)

class calibre.utils.formatter_functions.**BuiltinFormatDate**

format_date(val, format_string) – format the value, which must be a date, using the format_string, returning a string. The formatting codes are: d : the day as number without a leading zero (1 to 31) dd : the day as number with a leading zero (01 to 31) ddd : the abbreviated localized day name (e.g. "Mon" to "Sun"). dddd : the long localized day name (e.g. "Monday" to "Sunday"). M : the month as number without a leading zero (1 to 12). MM : the month as number with a leading zero (01 to 12) MMM : the abbreviated localized month name (e.g. "Jan" to "Dec"). MMMM : the long localized month name (e.g. "January" to "December"). yy : the year as two digit number (00 to 99). yyyy : the year as four digit number. h : the hours without a leading 0 (0 to 11 or 0 to 23, depending on am/pm) hh : the hours with a leading 0 (00 to 11 or 00 to 23, depending on am/pm) m : the minutes without a leading 0 (0 to 59) mm : the minutes with a leading 0 (00 to 59) s : the seconds without a leading 0 (0 to 59) ss : the seconds with a leading 0 (00 to 59) ap : use a 12-hour clock instead of a 24-hour clock, with "ap" replaced by the localized string for am or pm AP : use a 12-hour clock instead of a 24-hour clock, with "AP" replaced by the localized string for AM or PM iso : the date with time and timezone. Must be the only format present

format_number(v, template)

class calibre.utils.formatter_functions.**BuiltinFormatNumber**

format_number(v, template) – format the number v using a python formatting template such as "{0:5.2f}" or "{0:,d}" or "${0:5,.2f}". The field_name part of the template must be a 0 (zero) (the "{0:" in the above examples). See the template language and python documentation for more examples. Returns the empty string if formatting fails.

human_readable(v)

class calibre.utils.formatter_functions.**BuiltinHumanReadable**

human_readable(v) – return a string representing the number v in KB, MB, GB, etc.

Get values from metadata

approximate_formats()

class calibre.utils.formatter_functions.**BuiltinApproximateFormats**

approximate_formats() – return a comma-separated list of formats that at one point were associated with the book. There is no guarantee that this list is correct, although it probably is. This function can be called in template program mode using the template "{:'approximate_formats()'}". Note that format names are always uppercase, as in EPUB.

booksize()

class calibre.utils.formatter_functions.**BuiltinBooksize**

booksize() – return value of the size field

current_library_name()

class calibre.utils.formatter_functions.**BuiltinCurrentLibraryName**

current_library_name() – return the last name on the path to the current calibre library. This function can be called in template program mode using the template "{:'current_library_name()'}".

current_library_path()

class calibre.utils.formatter_functions.**BuiltinCurrentLibraryPath**

current_library_path() – return the path to the current calibre library. This function can be called in template program mode using the template "{:'current_library_path()'}".

field(name)

class calibre.utils.formatter_functions.**BuiltinField**

field(name) – returns the metadata field named by name

formats_modtimes(date_format)

class calibre.utils.formatter_functions.**BuiltinFormatsModtimes**

formats_modtimes(date_format) – return a comma-separated list of colon_separated items representing modification times for the formats of a book. The date_format parameter specifies how the date is to be formatted. See the date_format function for details. You can use the select function to get the mod time for a specific format. Note that format names are always uppercase, as in EPUB.

formats_paths()

class calibre.utils.formatter_functions.**BuiltinFormatsPaths**

formats_paths() – return a comma-separated list of colon_separated items representing full path to the formats of a book. You can use the select function to get the path for a specific format. Note that format names are always uppercase, as in EPUB.

formats_sizes()

class calibre.utils.formatter_functions.**BuiltinFormatsSizes**

formats_sizes() – return a comma-separated list of colon_separated items representing sizes in bytes of the formats of a book. You can use the select function to get the size for a specific format. Note that format names are always uppercase, as in EPUB.

has_cover()

class calibre.utils.formatter_functions.**BuiltinHasCover**

has_cover() – return Yes if the book has a cover, otherwise return the empty string

language_codes(lang_strings)

class calibre.utils.formatter_functions.**BuiltinLanguageCodes**

language_codes(lang_strings) – return the language codes for the strings passed in lang_strings. The strings must be in the language of the current locale. Lang_strings is a comma-separated list.

language_strings(lang_codes, localize)

class calibre.utils.formatter_functions.**BuiltinLanguageStrings**

language_strings(lang_codes, localize) – return the strings for the language codes passed in lang_codes. If localize is zero, return the strings in English. If localize is not zero, return the strings in the language of the current locale. Lang_codes is a comma-separated list.

ondevice()
class calibre.utils.formatter_functions.**BuiltinOndevice**
ondevice() – return Yes if ondevice is set, otherwise return the empty string

raw_field(name)
class calibre.utils.formatter_functions.**BuiltinRawField**
raw_field(name) – returns the metadata field named by name without applying any formatting.

series_sort()
class calibre.utils.formatter_functions.**BuiltinSeriesSort**
series_sort() – return the series sort value

If-then-else

contains(val, pattern, text if match, text if not match)
class calibre.utils.formatter_functions.**BuiltinContains**
contains(val, pattern, text if match, text if not match) – checks if field contains matches for the regular expression *pattern*. Returns *text if match* if matches are found, otherwise it returns *text if no match*

ifempty(val, text if empty)
class calibre.utils.formatter_functions.**BuiltinIfempty**
ifempty(val, text if empty) – return val if val is not empty, otherwise return *text if empty*

test(val, text if not empty, text if empty)
class calibre.utils.formatter_functions.**BuiltinTest**
test(val, text if not empty, text if empty) – return *text if not empty* if the field is not empty, otherwise return *text if empty*

Iterating over values

first_non_empty(value, value, ...)
class calibre.utils.formatter_functions.**BuiltinFirstNonEmpty**
first_non_empty(value, value, ...) – returns the first value that is not empty. If all values are empty, then the empty value is returned. You can have as many values as you want.

lookup(val, pattern, field, pattern, field, ..., else_field)
class calibre.utils.formatter_functions.**BuiltinLookup**
lookup(val, pattern, field, pattern, field, ..., else_field) – like switch, except the arguments are field (metadata) names, not text. The value of the appropriate field will be fetched and used. Note that because composite columns are fields, you can use this function in one composite field to use the value of some other composite field. This is extremely useful when constructing variable save paths

switch(val, pattern, value, pattern, value, ..., else_value)
class calibre.utils.formatter_functions.**BuiltinSwitch**
switch(val, pattern, value, pattern, value, ..., else_value) – for each *pattern, value* pair, checks if the field matches the regular expression *pattern* and if so, returns that *value*. If no pattern matches, then else_value is returned. You can have as many *pattern, value* pairs as you want

List lookup

identifier_in_list(val, id, found_val, not_found_val)

class calibre.utils.formatter_functions.**BuiltinIdentifierInList**

identifier_in_list(val, id, found_val, not_found_val) – treat val as a list of identifiers separated by commas, comparing the string against each value in the list. An identifier has the format "identifier:value". The id parameter should be either "id" or "id:regexp". The first case matches if there is any identifier with that id. The second case matches if the regexp matches the identifier's value. If there is a match, return found_val, otherwise return not_found_val.

in_list(val, separator, pattern, found_val, not_found_val)

class calibre.utils.formatter_functions.**BuiltinInList**

in_list(val, separator, pattern, found_val, not_found_val) – treat val as a list of items separated by separator, comparing the pattern against each value in the list. If the pattern matches a value, return found_val, otherwise return not_found_val.

list_item(val, index, separator)

class calibre.utils.formatter_functions.**BuiltinListitem**

list_item(val, index, separator) – interpret the value as a list of items separated by *separator*, returning the *index'th item. The first item is number zero. The last item can be returned using 'list_item(-1,separator)*. If the item is not in the list, then the empty value is returned. The separator has the same meaning as in the count function.

select(val, key)

class calibre.utils.formatter_functions.**BuiltinSelect**

select(val, key) – interpret the value as a comma-separated list of items, with the items being "id:value". Find the pair with the id equal to key, and return the corresponding value.

str_in_list(val, separator, string, found_val, not_found_val)

class calibre.utils.formatter_functions.**BuiltinStrInList**

str_in_list(val, separator, string, found_val, not_found_val) – treat val as a list of items separated by separator, comparing the string against each value in the list. If the string matches a value, return found_val, otherwise return not_found_val. If the string contains separators, then it is also treated as a list and each value is checked.

List manipulation

count(val, separator)

class calibre.utils.formatter_functions.**BuiltinCount**

count(val, separator) – interprets the value as a list of items separated by *separator*, returning the number of items in the list. Most lists use a comma as the separator, but authors uses an ampersand. Examples: {tags:count(,)}, {authors:count(&)}

list_difference(list1, list2, separator)

class calibre.utils.formatter_functions.**BuiltinListDifference**

list_difference(list1, list2, separator) – return a list made by removing from list1 any item found in list2, using a case-insensitive compare. The items in list1 and list2 are separated by separator, as are the items in the returned list.

list_equals(list1, sep1, list2, sep2, yes_val, no_val)

class calibre.utils.formatter_functions.**BuiltinListEquals**

list_equals(list1, sep1, list2, sep2, yes_val, no_val) – return yes_val if list1 and list2 contain the same items, otherwise return no_val. The items are determined by splitting each list using the appropriate separator character (sep1 or sep2). The order of items in the lists is not relevant. The compare is case insensitive.

list_intersection(list1, list2, separator)

class calibre.utils.formatter_functions.**BuiltinListIntersection**

list_intersection(list1, list2, separator) – return a list made by removing from list1 any item not found in list2, using a case-insensitive compare. The items in list1 and list2 are separated by separator, as are the items in the returned list.

list_re(src_list, separator, search_re, opt_replace)

class calibre.utils.formatter_functions.**BuiltinListRe**

list_re(src_list, separator, search_re, opt_replace) – Construct a list by first separating src_list into items using the separator character. For each item in the list, check if it matches search_re. If it does, then add it to the list to be returned. If opt_replace is not the empty string, then apply the replacement before adding the item to the returned list.

list_sort(list, direction, separator)

class calibre.utils.formatter_functions.**BuiltinListSort**

list_sort(list, direction, separator) – return list sorted using a case-insensitive sort. If direction is zero, the list is sorted ascending, otherwise descending. The list items are separated by separator, as are the items in the returned list.

list_union(list1, list2, separator)

class calibre.utils.formatter_functions.**BuiltinListUnion**

list_union(list1, list2, separator) – return a list made by merging the items in list1 and list2, removing duplicate items using a case-insensitive compare. If items differ in case, the one in list1 is used. The items in list1 and list2 are separated by separator, as are the items in the returned list.

subitems(val, start_index, end_index)

class calibre.utils.formatter_functions.**BuiltinSubitems**

subitems(val, start_index, end_index) – This function is used to break apart lists of items such as genres. It interprets the value as a comma-separated list of items, where each item is a period-separated list. Returns a new list made by first finding all the period-separated items, then for each such item extracting the *start_index* to the *end_index* components, then combining the results back together. The first component in a period-separated list has an index of zero. If an index is negative, then it counts from the end of the list. As a special case, an end_index of zero is assumed to be the length of the list. Example using basic template mode and assuming a #genre value of "A.B.C": {#genre:subitems(0,1)} returns "A". {#genre:subitems(0,2)} returns "A.B". {#genre:subitems(1,0)} returns "B.C". Assuming a #genre value of "A.B.C, D.E.F", {#genre:subitems(0,1)} returns "A, D". {#genre:subitems(0,2)} returns "A.B, D.E"

sublist(val, start_index, end_index, separator)

class calibre.utils.formatter_functions.**BuiltinSublist**

sublist(val, start_index, end_index, separator) – interpret the value as a list of items separated by *separator*, returning a new list made from the *start_index* to the *end_index* item. The first item is number zero. If an index is negative, then it counts from the end of the list. As a special case, an end_index of zero is assumed to be the length of the list. Examples using basic template mode and assuming that the tags column

(which is comma-separated) contains "A, B, C": {tags:sublist(0,1,,)} returns "A". {tags:sublist(-1,0,,)} returns "C". {tags:sublist(0,-1,,)} returns "A, B".

Other

assign(id, val)
class calibre.utils.formatter_functions.**BuiltinAssign**
assign(id, val) – assigns val to id, then returns val. id must be an identifier, not an expression

print(a, b, ...)
class calibre.utils.formatter_functions.**BuiltinPrint**
print(a, b, ...) – prints the arguments to standard output. Unless you start calibre from the command line (calibre-debug -g), the output will go to a black hole.

Recursion

eval(template)
class calibre.utils.formatter_functions.**BuiltinEval**
eval(template) – evaluates the template, passing the local variables (those 'assign'ed to) instead of the book metadata. This permits using the template processor to construct complex results from local variables. Because the { and } characters are special, you must use [[for the { character and]] for the } character; they are converted automatically. Note also that prefixes and suffixes (the *|prefix|suffix* syntax) cannot be used in the argument to this function when using template program mode.

template(x)
class calibre.utils.formatter_functions.**BuiltinTemplate**
template(x) – evaluates x as a template. The evaluation is done in its own context, meaning that variables are not shared between the caller and the template evaluation. Because the { and } characters are special, you must use [[for the { character and]] for the } character; they are converted automatically. For example, template('[[title_sort]]') will evaluate the template {title_sort} and return its value. Note also that prefixes and suffixes (the *|prefix|suffix* syntax) cannot be used in the argument to this function when using template program mode.

Relational

cmp(x, y, lt, eq, gt)
class calibre.utils.formatter_functions.**BuiltinCmp**
cmp(x, y, lt, eq, gt) – compares x and y after converting both to numbers. Returns lt if x < y. Returns eq if x == y. Otherwise returns gt.

strcmp(x, y, lt, eq, gt)
class calibre.utils.formatter_functions.**BuiltinStrcmp**
strcmp(x, y, lt, eq, gt) – does a case-insensitive comparison of x and y as strings. Returns lt if x < y. Returns eq if x == y. Otherwise returns gt.

String case changes

capitalize(val)
class calibre.utils.formatter_functions.**BuiltinCapitalize**
capitalize(val) – return value of the field capitalized

lowercase(val)
class calibre.utils.formatter_functions.**BuiltinLowercase**
lowercase(val) – return value of the field in lower case

titlecase(val)
class calibre.utils.formatter_functions.**BuiltinTitlecase**
titlecase(val) – return value of the field in title case

uppercase(val)
class calibre.utils.formatter_functions.**BuiltinUppercase**
uppercase(val) – return value of the field in upper case

String manipulation

re(val, pattern, replacement)
class calibre.utils.formatter_functions.**BuiltinRe**
re(val, pattern, replacement) – return the field after applying the regular expression. All instances of *pattern* are replaced with *replacement*. As in all of calibre, these are python-compatible regular expressions

shorten(val, left chars, middle text, right chars)
class calibre.utils.formatter_functions.**BuiltinShorten**
shorten(val, left chars, middle text, right chars) – Return a shortened version of the field, consisting of *left chars* characters from the beginning of the field, followed by *middle text*, followed by *right chars* characters from the end of the string. *Left chars* and *right chars* must be integers. For example, assume the title of the book is *Ancient English Laws in the Times of Ivanhoe*, and you want it to fit in a space of at most 15 characters. If you use {title:shorten(9,-,5)}, the result will be *Ancient E-nhoe*. If the field's length is less than left chars + right chars + the length of *middle text*, then the field will be used intact. For example, the title *The Dome* would not be changed.

strcat(a, b, ...)
class calibre.utils.formatter_functions.**BuiltinStrcat**
strcat(a, b, ...) – can take any number of arguments. Returns a string formed by concatenating all the arguments

strcat_max(max, string1, prefix2, string2, ...)
class calibre.utils.formatter_functions.**BuiltinStrcatMax**
strcat_max(max, string1, prefix2, string2, ...) – Returns a string formed by concatenating the arguments. The returned value is initialized to string1. *Prefix, string* pairs are added to the end of the value as long as the resulting string length is less than *max*. String1 is returned even if string1 is longer than max. You can pass as many *prefix, string* pairs as you wish.

strlen(a)
class calibre.utils.formatter_functions.**BuiltinStrlen**
strlen(a) – Returns the length of the string passed as the argument

substr(str, start, end)

class calibre.utils.formatter_functions.**BuiltinSubstr**

substr(str, start, end) – returns the start'th through the end'th characters of str. The first character in str is the zero'th character. If end is negative, then it indicates that many characters counting from the right. If end is zero, then it indicates the last character. For example, substr('12345', 1, 0) returns '2345', and substr('12345', 1, -1) returns '234'.

swap_around_comma(val)

class calibre.utils.formatter_functions.**BuiltinSwapAroundComma**

swap_around_comma(val) – given a value of the form "B, A", return "A B". This is most useful for converting names in LN, FN format to FN LN. If there is no comma, the function returns val unchanged

API of the Metadata objects The python implementation of the template functions is passed in a Metadata object. Knowing it's API is useful if you want to define your own template functions.

class calibre.ebooks.metadata.book.base.**Metadata**(*title*, *authors=(u'Unknown'*, *)*, *other=None*, *template_cache=None*)

A class representing all the metadata for a book. The various standard metadata fields are available as attributes of this object. You can also stick arbitrary attributes onto this object.

Metadata from custom columns should be accessed via the get() method, passing in the lookup name for the column, for example: "#mytags".

Use the is_null() (page 412) method to test if a field is null.

This object also has functions to format fields into strings.

The list of standard metadata fields grows with time is in STANDARD_METADATA_FIELDS (page 413).

Please keep the method based API of this class to a minimum. Every method becomes a reserved field name.

is_null(*field*)

Return True if the value of field is null in this object. 'null' means it is unknown or evaluates to False. So a title of _('Unknown') is null or a language of 'und' is null.

Be careful with numeric fields since this will return True for zero as well as None.

Also returns True if the field does not exist.

get_identifiers()

Return a copy of the identifiers dictionary. The dict is small, and the penalty for using a reference where a copy is needed is large. Also, we don't want any manipulations of the returned dict to show up in the book.

set_identifiers(*identifiers*)

Set all identifiers. Note that if you previously set ISBN, calling this method will delete it.

set_identifier(*typ*, *val*)

If val is empty, deletes identifier of type typ

standard_field_keys()

return a list of all possible keys, even if this book doesn't have them

custom_field_keys()

return a list of the custom fields in this book

all_field_keys()

All field keys known by this instance, even if their value is None

metadata_for_field(*key*)

return metadata describing a standard or custom field.

all_non_none_fields()
Return a dictionary containing all non-None metadata fields, including the custom ones.

get_standard_metadata(*field*, *make_copy*)
return field metadata from the field if it is there. Otherwise return None. field is the key name, not the label. Return a copy if requested, just in case the user wants to change values in the dict.

get_all_standard_metadata(*make_copy*)
return a dict containing all the standard field metadata associated with the book.

get_all_user_metadata(*make_copy*)
return a dict containing all the custom field metadata associated with the book.

get_user_metadata(*field*, *make_copy*)
return field metadata from the object if it is there. Otherwise return None. field is the key name, not the label. Return a copy if requested, just in case the user wants to change values in the dict.

set_all_user_metadata(*metadata*)
store custom field metadata into the object. Field is the key name not the label

set_user_metadata(*field*, *metadata*)
store custom field metadata for one column into the object. Field is the key name not the label

template_to_attribute(*other*, *ops*)
Takes a list [(src,dest), (src,dest)], evaluates the template in the context of other, then copies the result to self[dest]. This is on a best-efforts basis. Some assignments can make no sense.

smart_update(*other*, *replace_metadata=False*)
Merge the information in *other* into self. In case of conflicts, the information in *other* takes precedence, unless the information in *other* is NULL.

format_field(*key*, *series_with_index=True*)
Returns the tuple (display_name, formatted_value)

to_html()
A HTML representation of this object.

calibre.ebooks.metadata.book.base.**STANDARD_METADATA_FIELDS**
The set of standard metadata fields.

```
'''
All fields must have a NULL value represented as None for simple types,
an empty list/dictionary for complex types and (None, None) for cover_data
'''

SOCIAL_METADATA_FIELDS = frozenset([
    'tags',             # Ordered list
    'rating',           # A floating point number between 0 and 10
    'comments',         # A simple HTML enabled string
    'series',           # A simple string
    'series_index',     # A floating point number
    # Of the form { scheme1:value1, scheme2:value2}
    # For example: {'isbn':'123456789', 'doi':'xxxx', ... }
    'identifiers',
])

'''
The list of names that convert to identifiers when in get and set.
'''

TOP_LEVEL_IDENTIFIERS = frozenset([
```

```
    'isbn',
])

PUBLICATION_METADATA_FIELDS = frozenset([
    'title',            # title must never be None. Should be _('Unknown')
    # Pseudo field that can be set, but if not set is auto generated
    # from title and languages
    'title_sort',
    'authors',          # Ordered list. Must never be None, can be [_('Unknown')]
    'author_sort_map',  # Map of sort strings for each author
    # Pseudo field that can be set, but if not set is auto generated
    # from authors and languages
    'author_sort',
    'book_producer',
    'timestamp',        # Dates and times must be timezone aware
    'pubdate',
    'last_modified',
    'rights',
    # So far only known publication type is periodical:calibre
    # If None, means book
    'publication_type',
    'uuid',             # A UUID usually of type 4
    'languages',        # ordered list of languages in this publication
    'publisher',        # Simple string, no special semantics
    # Absolute path to image file encoded in filesystem_encoding
    'cover',
    # Of the form (format, data) where format is, for e.g. 'jpeg', 'png', 'gif'...
    'cover_data',
    # Either thumbnail data, or an object with the attribute
    # image_path which is the path to an image file, encoded
    # in filesystem_encoding
    'thumbnail',
    ])

BOOK_STRUCTURE_FIELDS = frozenset([
    # These are used by code, Null values are None.
    'toc', 'spine', 'guide', 'manifest',
    ])

USER_METADATA_FIELDS = frozenset([
    # A dict of dicts similar to field_metadata. Each field description dict
    # also contains a value field with the key #value#.
    'user_metadata',
])

DEVICE_METADATA_FIELDS = frozenset([
    'device_collections',   # Ordered list of strings
    'lpath',                # Unicode, / separated
    'size',                 # In bytes
    'mime',                 # Mimetype of the book file being represented

])

CALIBRE_METADATA_FIELDS = frozenset([
    'application_id',   # An application id, currently set to the db_id.
    'db_id',            # the calibre primary key of the item.
    'formats',          # list of formats (extensions) for this book
    # a dict of user category names, where the value is a list of item names
```

```
    # from the book that are in that category
    'user_categories',
    # a dict of author to an associated hyperlink
    'author_link_map',

    ]
)

ALL_METADATA_FIELDS =       SOCIAL_METADATA_FIELDS.union(
                            PUBLICATION_METADATA_FIELDS).union(
                            BOOK_STRUCTURE_FIELDS).union(
                            USER_METADATA_FIELDS).union(
                            DEVICE_METADATA_FIELDS).union(
                            CALIBRE_METADATA_FIELDS)

# All fields except custom fields
STANDARD_METADATA_FIELDS = SOCIAL_METADATA_FIELDS.union(
                            PUBLICATION_METADATA_FIELDS).union(
                            BOOK_STRUCTURE_FIELDS).union(
                            DEVICE_METADATA_FIELDS).union(
                            CALIBRE_METADATA_FIELDS)

# Metadata fields that smart update must do special processing to copy.
SC_FIELDS_NOT_COPIED =      frozenset(['title', 'title_sort', 'authors',
                                       'author_sort', 'author_sort_map',
                                       'cover_data', 'tags', 'languages',
                                       'identifiers'])

# Metadata fields that smart update should copy only if the source is not None
SC_FIELDS_COPY_NOT_NULL =  frozenset(['lpath', 'size', 'comments', 'thumbnail'])

# Metadata fields that smart update should copy without special handling
SC_COPYABLE_FIELDS =        SOCIAL_METADATA_FIELDS.union(
                            PUBLICATION_METADATA_FIELDS).union(
                            BOOK_STRUCTURE_FIELDS).union(
                            DEVICE_METADATA_FIELDS).union(
                            CALIBRE_METADATA_FIELDS) - \
                            SC_FIELDS_NOT_COPIED.union(
                            SC_FIELDS_COPY_NOT_NULL)

SERIALIZABLE_FIELDS =       SOCIAL_METADATA_FIELDS.union(
                            USER_METADATA_FIELDS).union(
                            PUBLICATION_METADATA_FIELDS).union(
                            CALIBRE_METADATA_FIELDS).union(
                            DEVICE_METADATA_FIELDS) - \
                            frozenset(['device_collections', 'formats',
                                'cover_data'])
                            # these are rebuilt when needed
```

Using general program mode

For more complicated template programs, it is sometimes easier to avoid template syntax (all the *{* and *}* characters), instead writing a more classical-looking program. You can do this in calibre by beginning the template with *program:*. In this case, no template processing is done. The special variable *$* is not set. It is up to your program to produce the correct results.

One advantage of *program:* mode is that the brackets are no longer special. For example, it is not necessary to use *[[*

and *]]* when using the *template()* function. Another advantage is that program mode templates are compiled to Python and can run much faster than templates in the other two modes. Speed improvement depends on the complexity of the templates; the more complicated the template the more the improvement. Compilation is turned off or on using the tweak `compile_gpm_templates` (Compile General Program Mode templates to Python). The main reason to turn off compilation is if a compiled template does not work, in which case please file a bug report.

The following example is a *program:* mode implementation of a recipe on the MobileRead forum: "Put series into the title, using either initials or a shortened form. Strip leading articles from the series name (any)." For example, for the book The Two Towers in the Lord of the Rings series, the recipe gives *LotR [02] The Two Towers*. Using standard templates, the recipe requires three custom columns and a plugboard, as explained in the following:

The solution requires creating three composite columns. The first column is used to remove the leading articles. The second is used to compute the 'shorten' form. The third is to compute the 'initials' form. Once you have these columns, the plugboard selects between them. You can hide any or all of the three columns on the library view.

First column: Name: #stripped_series. Template: {series:re(^(A|The|An)s+,)||}

Second column (the shortened form): Name: #shortened. Template: {#stripped_series:shorten(4,-,4)}

Third column (the initials form): Name: #initials. Template: {#stripped_series:re(([^s])[^s]+(s|$),1)}

Plugboard expression: Template:{#stripped_series:lookup(.s,#initials,.,#shortened,series)}{series_index:0>2.0f|[|] }{title} Destination field: title

This set of fields and plugboard produces: Series: The Lord of the Rings Series index: 2 Title: The Two Towers Output: LotR [02] The Two Towers

Series: Dahak Series index: 1 Title: Mutineers Moon Output: Dahak [01] Mutineers Moon

Series: Berserkers Series Index: 4 Title: Berserker Throne Output: Bers-kers [04] Berserker Throne

Series: Meg Langslow Mysteries Series Index: 3 Title: Revenge of the Wrought-Iron Flamingos Output: MLM [03] Revenge of the Wrought-Iron Flamingos

The following program produces the same results as the original recipe, using only one custom column to hold the results of a program that computes the special title value:

```
Custom column:
Name: #special_title
Template: (the following with all leading spaces removed)
    program:
    #       compute the equivalent of the composite fields and store them in local variables
        stripped = re(field('series'), '^(A|The|An)\s+', '');
        shortened = shorten(stripped, 4, '-' ,4);
        initials = re(stripped, '[^\w]*(\w?)[^\s]+(\s|$)', '\1');

    #       Format the series index. Ends up as empty if there is no series index.
    #       Note that leading and trailing spaces will be removed by the formatter,
    #       so we cannot add them here. We will do that in the strcat below.
    #       Also note that because we are in 'program' mode, we can freely use
    #       curly brackets in strings, something we cannot do in template mode.
        s_index = template('{series_index:0>2.0f}');

    #       print(stripped, shortened, initials, s_index);

    #       Now concatenate all the bits together. The switch picks between
    #       initials and shortened, depending on whether there is a space
    #       in stripped. We then add the brackets around s_index if it is
    #       not empty. Finally, add the title. As this is the last function in
    #       the program, its value will be returned.
        strcat(
            switch( stripped,
```

```
                '.\s', initials,
                '.', shortened,
                field('series')),
            test(s_index, strcat(' [', s_index, '] '), ''),
            field('title'));

Plugboard expression:
Template:{#special_title}
Destination field: title
```

It would be possible to do the above with no custom columns by putting the program into the template box of the plugboard. However, to do so, all comments must be removed because the plugboard text box does not support multi-line editing. It is debatable whether the gain of not having the custom column is worth the vast increase in difficulty caused by the program being one giant line.

User-defined Template Functions

You can add your own functions to the template processor. Such functions are written in python, and can be used in any of the three template programming modes. The functions are added by going to Preferences -> Advanced -> Template Functions. Instructions are shown in that dialog.

Special notes for save/send templates

Special processing is applied when a template is used in a *save to disk* or *send to device* template. The values of the fields are cleaned, replacing characters that are special to file systems with underscores, including slashes. This means that field text cannot be used to create folders. However, slashes are not changed in prefix or suffix strings, so slashes in these strings will cause folders to be created. Because of this, you can create variable-depth folder structure.

For example, assume we want the folder structure *series/series_index - title*, with the caveat that if series does not exist, then the title should be in the top folder. The template to do this is:

```
{series:||/}{series_index:|| - }{title}
```

The slash and the hyphen appear only if series is not empty.

The lookup function lets us do even fancier processing. For example, assume that if a book has a series, then we want the folder structure *series/series index - title.fmt*. If the book does not have a series, then we want the folder structure *genre/author_sort/title.fmt*. If the book has no genre, we want to use 'Unknown'. We want two completely different paths, depending on the value of series.

To accomplish this, we:

1. Create a composite field (call it AA) containing `{series}/{series_index} - {title'}`. If the series is not empty, then this template will produce *series/series_index - title*.
2. Create a composite field (call it BB) containing `{#genre:ifempty(Unknown)}/{author_sort}/{title}`. This template produces *genre/author_sort/title*, where an empty genre is replaced wuth *Unknown*.
3. Set the save template to `{series:lookup(.,AA,BB)}`. This template chooses composite field AA if series is not empty, and composite field BB if series is empty. We therefore have two completely different save paths, depending on whether or not *series* is empty.

Templates and Plugboards

Plugboards are used for changing the metadata written into books during send-to-device and save-to-disk operations. A plugboard permits you to specify a template to provide the data to write into the book's metadata. You can use plugboards to modify the following fields: authors, author_sort, language, publisher, tags, title, title_sort. This feature helps people who want to use different metadata in books on devices to solve sorting or display issues.

When you create a plugboard, you specify the format and device for which the plugboard is to be used. A special device is provided, save_to_disk, that is used when saving formats (as opposed to sending them to a device). Once you have chosen the format and device, you choose the metadata fields to change, providing templates to supply the new values. These templates are *connected* to their destination fields, hence the name *plugboards*. You can, of course, use composite columns in these templates.

When a plugboard might apply (content server, save to disk, or send to device), calibre searches the defined plugboards to choose the correct one for the given format and device. For example, to find the appropriate plugboard for an EPUB book being sent to an ANDROID device, calibre searches the plugboards using the following search order:

- a plugboard with an exact match on format and device, e.g., `EPUB` and `ANDROID`
- a plugboard with an exact match on format and the special `any device` choice, e.g., `EPUB` and `any device`
- a plugboard with the special `any format` choice and an exact match on device, e.g., `any format` and `ANDROID`
- a plugboard with `any format` and `any device`

The tags and authors fields have special treatment, because both of these fields can hold more than one item. A book can have many tags and many authors. When you specify that one of these two fields is to be changed, the template's result is examined to see if more than one item is there. For tags, the result is cut apart wherever calibre finds a comma. For example, if the template produces the value `Thriller, Horror`, then the result will be two tags, `Thriller` and `Horror`. There is no way to put a comma in the middle of a tag.

The same thing happens for authors, but using a different character for the cut, a *&* (ampersand) instead of a comma. For example, if the template produces the value `Blogs, Joe&Posts, Susan`, then the book will end up with two authors, `Blogs, Joe` and `Posts, Susan`. If the template produces the value `Blogs, Joe;Posts, Susan`, then the book will have one author with a rather strange name.

Plugboards affect the metadata written into the book when it is saved to disk or written to the device. Plugboards do not affect the metadata used by `save to disk` and `send to device` to create the file names. Instead, file names are constructed using the templates entered on the appropriate preferences window.

Helpful Tips

You might find the following tips useful.

- Create a custom composite column to test templates. Once you have the column, you can change its template simply by double-clicking on the column. Hide the column when you are not testing.
- Templates can use other templates by referencing a composite custom column.
- In a plugboard, you can set a field to empty (or whatever is equivalent to empty) by using the special template `{}`. This template will always evaluate to an empty string.
- The technique described above to show numbers even if they have a zero value works with the standard field series_index.

Reference for all built-in template language functions Here, we document all the built-in functions available in the calibre template language. Every function is implemented as a class in python and you can click the source links to see the source code, in case the documentation is insufficient. The functions are arranged in logical groups by type.

- Arithmetic (page 404)
  - add(x, y) (page 404)
  - divide(x, y) (page 404)
  - multiply(x, y) (page 404)
  - subtract(x, y) (page 404)
- Boolean (page 404)
  - and(value, value, ...) (page 404)
  - not(value) (page 404)
  - or(value, value, ...) (page 404)
- Date functions (page 404)
  - days_between(date1, date2) (page 404)
  - today() (page 404)
- Formatting values (page 405)
  - finish_formatting(val, fmt, prefix, suffix) (page 405)
  - format_date(val, format_string) (page 405)
  - format_number(v, template) (page 405)
  - human_readable(v) (page 405)
- Get values from metadata (page 405)
  - approximate_formats() (page 405)
  - booksize() (page 405)
  - current_library_name() (page 406)
  - current_library_path() (page 406)
  - field(name) (page 406)
  - formats_modtimes(date_format) (page 406)
  - formats_paths() (page 406)
  - formats_sizes() (page 406)
  - has_cover() (page 406)
  - language_codes(lang_strings) (page 406)
  - language_strings(lang_codes, localize) (page 406)
  - ondevice() (page 407)
  - raw_field(name) (page 407)
  - series_sort() (page 407)
- If-then-else (page 407)
  - contains(val, pattern, text if match, text if not match) (page 407)
  - ifempty(val, text if empty) (page 407)
  - test(val, text if not empty, text if empty) (page 407)
- Iterating over values (page 407)
  - first_non_empty(value, value, ...) (page 407)
  - lookup(val, pattern, field, pattern, field, ..., else_field) (page 407)
  - switch(val, pattern, value, pattern, value, ..., else_value) (page 407)
- List lookup (page 408)
  - identifier_in_list(val, id, found_val, not_found_val) (page 408)
  - in_list(val, separator, pattern, found_val, not_found_val) (page 408)
  - list_item(val, index, separator) (page 408)
  - select(val, key) (page 408)
  - str_in_list(val, separator, string, found_val, not_found_val) (page 408)
- List manipulation (page 408)
  - count(val, separator) (page 408)
  - list_difference(list1, list2, separator) (page 408)
  - list_equals(list1, sep1, list2, sep2, yes_val, no_val) (page 409)
  - list_intersection(list1, list2, separator) (page 409)
  - list_re(src_list, separator, search_re, opt_replace) (page 409)
  - list_sort(list, direction, separator) (page 409)
  - list_union(list1, list2, separator) (page 409)
  - subitems(val, start_index, end_index) (page 409)
  - sublist(val, start_index, end_index, separator) (page 409)
- Other (page 410)
  - assign(id, val) (page 410)
  - print(a, b, ...) (page 410)
- Recursion (page 410)

Arithmetic

add(x, y)
class calibre.utils.formatter_functions.**BuiltinAdd**
add(x, y) – returns x + y. Throws an exception if either x or y are not numbers.

divide(x, y)
class calibre.utils.formatter_functions.**BuiltinDivide**
divide(x, y) – returns x / y. Throws an exception if either x or y are not numbers.

multiply(x, y)
class calibre.utils.formatter_functions.**BuiltinMultiply**
multiply(x, y) – returns x * y. Throws an exception if either x or y are not numbers.

subtract(x, y)
class calibre.utils.formatter_functions.**BuiltinSubtract**
subtract(x, y) – returns x - y. Throws an exception if either x or y are not numbers.

Boolean

and(value, value, ...)
class calibre.utils.formatter_functions.**BuiltinAnd**
and(value, value, ...) – returns the string "1" if all values are not empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.

not(value)
class calibre.utils.formatter_functions.**BuiltinNot**
not(value) – returns the string "1" if the value is empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.

or(value, value, ...)
class calibre.utils.formatter_functions.**BuiltinOr**
or(value, value, ...) – returns the string "1" if any value is not empty, otherwise returns the empty string. This function works well with test or first_non_empty. You can have as many values as you want.

Date functions

days_between(date1, date2)
class calibre.utils.formatter_functions.**BuiltinDaysBetween**
days_between(date1, date2) – return the number of days between date1 and date2. The number is positive if date1 is greater than date2, otherwise negative. If either date1 or date2 are not dates, the function returns the empty string.

today()
class calibre.utils.formatter_functions.**BuiltinToday**
today() – return a date string for today. This value is designed for use in format_date or days_between, but can be manipulated like any other string. The date is in ISO format.

Formatting values

finish_formatting(val, fmt, prefix, suffix)

class calibre.utils.formatter_functions.**BuiltinFinishFormatting**

finish_formatting(val, fmt, prefix, suffix) – apply the format, prefix, and suffix to a value in the same way as done in a template like *{series_index:05.2f| - |- }*. For example, the following program produces the same output as the above template: program: finish_formatting(field("series_index"), "05.2f", " - ", " - ")

format_date(val, format_string)

class calibre.utils.formatter_functions.**BuiltinFormatDate**

format_date(val, format_string) – format the value, which must be a date, using the format_string, returning a string. The formatting codes are: d : the day as number without a leading zero (1 to 31) dd : the day as number with a leading zero (01 to 31) ddd : the abbreviated localized day name (e.g. "Mon" to "Sun"). dddd : the long localized day name (e.g. "Monday" to "Sunday"). M : the month as number without a leading zero (1 to 12). MM : the month as number with a leading zero (01 to 12) MMM : the abbreviated localized month name (e.g. "Jan" to "Dec"). MMMM : the long localized month name (e.g. "January" to "December"). yy : the year as two digit number (00 to 99). yyyy : the year as four digit number. h : the hours without a leading 0 (0 to 11 or 0 to 23, depending on am/pm) hh : the hours with a leading 0 (00 to 11 or 00 to 23, depending on am/pm) m : the minutes without a leading 0 (0 to 59) mm : the minutes with a leading 0 (00 to 59) s : the seconds without a leading 0 (0 to 59) ss : the seconds with a leading 0 (00 to 59) ap : use a 12-hour clock instead of a 24-hour clock, with "ap" replaced by the localized string for am or pm AP : use a 12-hour clock instead of a 24-hour clock, with "AP" replaced by the localized string for AM or PM iso : the date with time and timezone. Must be the only format present

format_number(v, template)

class calibre.utils.formatter_functions.**BuiltinFormatNumber**

format_number(v, template) – format the number v using a python formatting template such as "{0:5.2f}" or "{0:,d}" or "${0:5,.2f}". The field_name part of the template must be a 0 (zero) (the "{0:" in the above examples). See the template language and python documentation for more examples. Returns the empty string if formatting fails.

human_readable(v)

class calibre.utils.formatter_functions.**BuiltinHumanReadable**

human_readable(v) – return a string representing the number v in KB, MB, GB, etc.

Get values from metadata

approximate_formats()

class calibre.utils.formatter_functions.**BuiltinApproximateFormats**

approximate_formats() – return a comma-separated list of formats that at one point were associated with the book. There is no guarantee that this list is correct, although it probably is. This function can be called in template program mode using the template "{:'approximate_formats()'}". Note that format names are always uppercase, as in EPUB.

booksize()

class calibre.utils.formatter_functions.**BuiltinBooksize**

booksize() – return value of the size field

current_library_name()

class calibre.utils.formatter_functions.**BuiltinCurrentLibraryName**

current_library_name() – return the last name on the path to the current calibre library. This function can be called in template program mode using the template "{:'current_library_name()'}".

current_library_path()

class calibre.utils.formatter_functions.**BuiltinCurrentLibraryPath**

current_library_path() – return the path to the current calibre library. This function can be called in template program mode using the template "{:'current_library_path()'}".

field(name)

class calibre.utils.formatter_functions.**BuiltinField**

field(name) – returns the metadata field named by name

formats_modtimes(date_format)

class calibre.utils.formatter_functions.**BuiltinFormatsModtimes**

formats_modtimes(date_format) – return a comma-separated list of colon_separated items representing modification times for the formats of a book. The date_format parameter specifies how the date is to be formatted. See the date_format function for details. You can use the select function to get the mod time for a specific format. Note that format names are always uppercase, as in EPUB.

formats_paths()

class calibre.utils.formatter_functions.**BuiltinFormatsPaths**

formats_paths() – return a comma-separated list of colon_separated items representing full path to the formats of a book. You can use the select function to get the path for a specific format. Note that format names are always uppercase, as in EPUB.

formats_sizes()

class calibre.utils.formatter_functions.**BuiltinFormatsSizes**

formats_sizes() – return a comma-separated list of colon_separated items representing sizes in bytes of the formats of a book. You can use the select function to get the size for a specific format. Note that format names are always uppercase, as in EPUB.

has_cover()

class calibre.utils.formatter_functions.**BuiltinHasCover**

has_cover() – return Yes if the book has a cover, otherwise return the empty string

language_codes(lang_strings)

class calibre.utils.formatter_functions.**BuiltinLanguageCodes**

language_codes(lang_strings) – return the language codes for the strings passed in lang_strings. The strings must be in the language of the current locale. Lang_strings is a comma-separated list.

language_strings(lang_codes, localize)

class calibre.utils.formatter_functions.**BuiltinLanguageStrings**

language_strings(lang_codes, localize) – return the strings for the language codes passed in lang_codes. If localize is zero, return the strings in English. If localize is not zero, return the strings in the language of the current locale. Lang_codes is a comma-separated list.

ondevice()
class calibre.utils.formatter_functions.**BuiltinOndevice**
ondevice() – return Yes if ondevice is set, otherwise return the empty string

raw_field(name)
class calibre.utils.formatter_functions.**BuiltinRawField**
raw_field(name) – returns the metadata field named by name without applying any formatting.

series_sort()
class calibre.utils.formatter_functions.**BuiltinSeriesSort**
series_sort() – return the series sort value

If-then-else

contains(val, pattern, text if match, text if not match)
class calibre.utils.formatter_functions.**BuiltinContains**
contains(val, pattern, text if match, text if not match) – checks if field contains matches for the regular expression *pattern*. Returns *text if match* if matches are found, otherwise it returns *text if no match*

ifempty(val, text if empty)
class calibre.utils.formatter_functions.**BuiltinIfempty**
ifempty(val, text if empty) – return val if val is not empty, otherwise return *text if empty*

test(val, text if not empty, text if empty)
class calibre.utils.formatter_functions.**BuiltinTest**
test(val, text if not empty, text if empty) – return *text if not empty* if the field is not empty, otherwise return *text if empty*

Iterating over values

first_non_empty(value, value, ...)
class calibre.utils.formatter_functions.**BuiltinFirstNonEmpty**
first_non_empty(value, value, ...) – returns the first value that is not empty. If all values are empty, then the empty value is returned. You can have as many values as you want.

lookup(val, pattern, field, pattern, field, ..., else_field)
class calibre.utils.formatter_functions.**BuiltinLookup**
lookup(val, pattern, field, pattern, field, ..., else_field) – like switch, except the arguments are field (metadata) names, not text. The value of the appropriate field will be fetched and used. Note that because composite columns are fields, you can use this function in one composite field to use the value of some other composite field. This is extremely useful when constructing variable save paths

switch(val, pattern, value, pattern, value, ..., else_value)
class calibre.utils.formatter_functions.**BuiltinSwitch**
switch(val, pattern, value, pattern, value, ..., else_value) – for each *pattern, value* pair, checks if the field matches the regular expression *pattern* and if so, returns that *value*. If no pattern matches, then else_value is returned. You can have as many *pattern, value* pairs as you want

List lookup

identifier_in_list(val, id, found_val, not_found_val)

class calibre.utils.formatter_functions.**BuiltinIdentifierInList**

identifier_in_list(val, id, found_val, not_found_val) – treat val as a list of identifiers separated by commas, comparing the string against each value in the list. An identifier has the format "identifier:value". The id parameter should be either "id" or "id:regexp". The first case matches if there is any identifier with that id. The second case matches if the regexp matches the identifier's value. If there is a match, return found_val, otherwise return not_found_val.

in_list(val, separator, pattern, found_val, not_found_val)

class calibre.utils.formatter_functions.**BuiltinInList**

in_list(val, separator, pattern, found_val, not_found_val) – treat val as a list of items separated by separator, comparing the pattern against each value in the list. If the pattern matches a value, return found_val, otherwise return not_found_val.

list_item(val, index, separator)

class calibre.utils.formatter_functions.**BuiltinListitem**

list_item(val, index, separator) – interpret the value as a list of items separated by *separator*, returning the *index'th item. The first item is number zero. The last item can be returned using 'list_item(-1,separator)*. If the item is not in the list, then the empty value is returned. The separator has the same meaning as in the count function.

select(val, key)

class calibre.utils.formatter_functions.**BuiltinSelect**

select(val, key) – interpret the value as a comma-separated list of items, with the items being "id:value". Find the pair with the id equal to key, and return the corresponding value.

str_in_list(val, separator, string, found_val, not_found_val)

class calibre.utils.formatter_functions.**BuiltinStrInList**

str_in_list(val, separator, string, found_val, not_found_val) – treat val as a list of items separated by separator, comparing the string against each value in the list. If the string matches a value, return found_val, otherwise return not_found_val. If the string contains separators, then it is also treated as a list and each value is checked.

List manipulation

count(val, separator)

class calibre.utils.formatter_functions.**BuiltinCount**

count(val, separator) – interprets the value as a list of items separated by *separator*, returning the number of items in the list. Most lists use a comma as the separator, but authors uses an ampersand. Examples: {tags:count(,)}, {authors:count(&)}

list_difference(list1, list2, separator)

class calibre.utils.formatter_functions.**BuiltinListDifference**

list_difference(list1, list2, separator) – return a list made by removing from list1 any item found in list2, using a case-insensitive compare. The items in list1 and list2 are separated by separator, as are the items in the returned list.

list_equals(list1, sep1, list2, sep2, yes_val, no_val)

class calibre.utils.formatter_functions.**BuiltinListEquals**

list_equals(list1, sep1, list2, sep2, yes_val, no_val) – return yes_val if list1 and list2 contain the same items, otherwise return no_val. The items are determined by splitting each list using the appropriate separator character (sep1 or sep2). The order of items in the lists is not relevant. The compare is case insensitive.

list_intersection(list1, list2, separator)

class calibre.utils.formatter_functions.**BuiltinListIntersection**

list_intersection(list1, list2, separator) – return a list made by removing from list1 any item not found in list2, using a case-insensitive compare. The items in list1 and list2 are separated by separator, as are the items in the returned list.

list_re(src_list, separator, search_re, opt_replace)

class calibre.utils.formatter_functions.**BuiltinListRe**

list_re(src_list, separator, search_re, opt_replace) – Construct a list by first separating src_list into items using the separator character. For each item in the list, check if it matches search_re. If it does, then add it to the list to be returned. If opt_replace is not the empty string, then apply the replacement before adding the item to the returned list.

list_sort(list, direction, separator)

class calibre.utils.formatter_functions.**BuiltinListSort**

list_sort(list, direction, separator) – return list sorted using a case-insensitive sort. If direction is zero, the list is sorted ascending, otherwise descending. The list items are separated by separator, as are the items in the returned list.

list_union(list1, list2, separator)

class calibre.utils.formatter_functions.**BuiltinListUnion**

list_union(list1, list2, separator) – return a list made by merging the items in list1 and list2, removing duplicate items using a case-insensitive compare. If items differ in case, the one in list1 is used. The items in list1 and list2 are separated by separator, as are the items in the returned list.

subitems(val, start_index, end_index)

class calibre.utils.formatter_functions.**BuiltinSubitems**

subitems(val, start_index, end_index) – This function is used to break apart lists of items such as genres. It interprets the value as a comma-separated list of items, where each item is a period-separated list. Returns a new list made by first finding all the period-separated items, then for each such item extracting the *start_index* to the *end_index* components, then combining the results back together. The first component in a period-separated list has an index of zero. If an index is negative, then it counts from the end of the list. As a special case, an end_index of zero is assumed to be the length of the list. Example using basic template mode and assuming a #genre value of "A.B.C": {#genre:subitems(0,1)} returns "A". {#genre:subitems(0,2)} returns "A.B". {#genre:subitems(1,0)} returns "B.C". Assuming a #genre value of "A.B.C, D.E.F", {#genre:subitems(0,1)} returns "A, D". {#genre:subitems(0,2)} returns "A.B, D.E"

sublist(val, start_index, end_index, separator)

class calibre.utils.formatter_functions.**BuiltinSublist**

sublist(val, start_index, end_index, separator) – interpret the value as a list of items separated by *separator*, returning a new list made from the *start_index* to the *end_index* item. The first item is number zero. If an index is negative, then it counts from the end of the list. As a special case, an end_index of zero is assumed to be the length of the list. Examples using basic template mode and assuming that the tags column

(which is comma-separated) contains "A, B, C": {tags:sublist(0,1,,)} returns "A". {tags:sublist(-1,0,,)} returns "C". {tags:sublist(0,-1,,)} returns "A, B".

Other

assign(id, val)
class calibre.utils.formatter_functions.**BuiltinAssign**
assign(id, val) – assigns val to id, then returns val. id must be an identifier, not an expression

print(a, b, ...)
class calibre.utils.formatter_functions.**BuiltinPrint**
print(a, b, ...) – prints the arguments to standard output. Unless you start calibre from the command line (calibre-debug -g), the output will go to a black hole.

Recursion

eval(template)
class calibre.utils.formatter_functions.**BuiltinEval**
eval(template) – evaluates the template, passing the local variables (those 'assign'ed to) instead of the book metadata. This permits using the template processor to construct complex results from local variables. Because the { and } characters are special, you must use [[for the { character and]] for the } character; they are converted automatically. Note also that prefixes and suffixes (the *|prefix|suffix* syntax) cannot be used in the argument to this function when using template program mode.

template(x)
class calibre.utils.formatter_functions.**BuiltinTemplate**
template(x) – evaluates x as a template. The evaluation is done in its own context, meaning that variables are not shared between the caller and the template evaluation. Because the { and } characters are special, you must use [[for the { character and]] for the } character; they are converted automatically. For example, template('[[title_sort]]') will evaluate the template {title_sort} and return its value. Note also that prefixes and suffixes (the *|prefix|suffix* syntax) cannot be used in the argument to this function when using template program mode.

Relational

cmp(x, y, lt, eq, gt)
class calibre.utils.formatter_functions.**BuiltinCmp**
cmp(x, y, lt, eq, gt) – compares x and y after converting both to numbers. Returns lt if x < y. Returns eq if x == y. Otherwise returns gt.

strcmp(x, y, lt, eq, gt)
class calibre.utils.formatter_functions.**BuiltinStrcmp**
strcmp(x, y, lt, eq, gt) – does a case-insensitive comparison of x and y as strings. Returns lt if x < y. Returns eq if x == y. Otherwise returns gt.

String case changes

capitalize(val)
class calibre.utils.formatter_functions.**BuiltinCapitalize**
capitalize(val) – return value of the field capitalized

lowercase(val)
class calibre.utils.formatter_functions.**BuiltinLowercase**
lowercase(val) – return value of the field in lower case

titlecase(val)
class calibre.utils.formatter_functions.**BuiltinTitlecase**
titlecase(val) – return value of the field in title case

uppercase(val)
class calibre.utils.formatter_functions.**BuiltinUppercase**
uppercase(val) – return value of the field in upper case

String manipulation

re(val, pattern, replacement)
class calibre.utils.formatter_functions.**BuiltinRe**
re(val, pattern, replacement) – return the field after applying the regular expression. All instances of *pattern* are replaced with *replacement*. As in all of calibre, these are python-compatible regular expressions

shorten(val, left chars, middle text, right chars)
class calibre.utils.formatter_functions.**BuiltinShorten**
shorten(val, left chars, middle text, right chars) – Return a shortened version of the field, consisting of *left chars* characters from the beginning of the field, followed by *middle text*, followed by *right chars* characters from the end of the string. *Left chars* and *right chars* must be integers. For example, assume the title of the book is *Ancient English Laws in the Times of Ivanhoe*, and you want it to fit in a space of at most 15 characters. If you use {title:shorten(9,-,5)}, the result will be *Ancient E-nhoe*. If the field's length is less than left chars + right chars + the length of *middle text*, then the field will be used intact. For example, the title *The Dome* would not be changed.

strcat(a, b, ...)
class calibre.utils.formatter_functions.**BuiltinStrcat**
strcat(a, b, ...) – can take any number of arguments. Returns a string formed by concatenating all the arguments

strcat_max(max, string1, prefix2, string2, ...)
class calibre.utils.formatter_functions.**BuiltinStrcatMax**
strcat_max(max, string1, prefix2, string2, ...) – Returns a string formed by concatenating the arguments. The returned value is initialized to string1. *Prefix, string* pairs are added to the end of the value as long as the resulting string length is less than *max*. String1 is returned even if string1 is longer than max. You can pass as many *prefix, string* pairs as you wish.

strlen(a)
class calibre.utils.formatter_functions.**BuiltinStrlen**
strlen(a) – Returns the length of the string passed as the argument

substr(str, start, end)

class calibre.utils.formatter_functions.**BuiltinSubstr**

substr(str, start, end) – returns the start'th through the end'th characters of str. The first character in str is the zero'th character. If end is negative, then it indicates that many characters counting from the right. If end is zero, then it indicates the last character. For example, substr('12345', 1, 0) returns '2345', and substr('12345', 1, -1) returns '234'.

swap_around_comma(val)

class calibre.utils.formatter_functions.**BuiltinSwapAroundComma**

swap_around_comma(val) – given a value of the form "B, A", return "A B". This is most useful for converting names in LN, FN format to FN LN. If there is no comma, the function returns val unchanged

API of the Metadata objects The python implementation of the template functions is passed in a Metadata object. Knowing it's API is useful if you want to define your own template functions.

class calibre.ebooks.metadata.book.base.**Metadata**(*title*, *authors=(u'Unknown',)*, *other=None*, *template_cache=None*)

A class representing all the metadata for a book. The various standard metadata fields are available as attributes of this object. You can also stick arbitrary attributes onto this object.

Metadata from custom columns should be accessed via the get() method, passing in the lookup name for the column, for example: "#mytags".

Use the is_null() (page 412) method to test if a field is null.

This object also has functions to format fields into strings.

The list of standard metadata fields grows with time is in STANDARD_METADATA_FIELDS (page 413).

Please keep the method based API of this class to a minimum. Every method becomes a reserved field name.

is_null(*field*)

Return True if the value of field is null in this object. 'null' means it is unknown or evaluates to False. So a title of _('Unknown') is null or a language of 'und' is null.

Be careful with numeric fields since this will return True for zero as well as None.

Also returns True if the field does not exist.

get_identifiers()

Return a copy of the identifiers dictionary. The dict is small, and the penalty for using a reference where a copy is needed is large. Also, we don't want any manipulations of the returned dict to show up in the book.

set_identifiers(*identifiers*)

Set all identifiers. Note that if you previously set ISBN, calling this method will delete it.

set_identifier(*typ*, *val*)

If val is empty, deletes identifier of type typ

standard_field_keys()

return a list of all possible keys, even if this book doesn't have them

custom_field_keys()

return a list of the custom fields in this book

all_field_keys()

All field keys known by this instance, even if their value is None

metadata_for_field(*key*)

return metadata describing a standard or custom field.

all_non_none_fields()
Return a dictionary containing all non-None metadata fields, including the custom ones.

get_standard_metadata(*field*, *make_copy*)
return field metadata from the field if it is there. Otherwise return None. field is the key name, not the label. Return a copy if requested, just in case the user wants to change values in the dict.

get_all_standard_metadata(*make_copy*)
return a dict containing all the standard field metadata associated with the book.

get_all_user_metadata(*make_copy*)
return a dict containing all the custom field metadata associated with the book.

get_user_metadata(*field*, *make_copy*)
return field metadata from the object if it is there. Otherwise return None. field is the key name, not the label. Return a copy if requested, just in case the user wants to change values in the dict.

set_all_user_metadata(*metadata*)
store custom field metadata into the object. Field is the key name not the label

set_user_metadata(*field*, *metadata*)
store custom field metadata for one column into the object. Field is the key name not the label

template_to_attribute(*other*, *ops*)
Takes a list [(src,dest), (src,dest)], evaluates the template in the context of other, then copies the result to self[dest]. This is on a best-efforts basis. Some assignments can make no sense.

smart_update(*other*, *replace_metadata=False*)
Merge the information in *other* into self. In case of conflicts, the information in *other* takes precedence, unless the information in *other* is NULL.

format_field(*key*, *series_with_index=True*)
Returns the tuple (display_name, formatted_value)

to_html()
A HTML representation of this object.

calibre.ebooks.metadata.book.base.**STANDARD_METADATA_FIELDS**
The set of standard metadata fields.

```
'''
All fields must have a NULL value represented as None for simple types,
an empty list/dictionary for complex types and (None, None) for cover_data
'''

SOCIAL_METADATA_FIELDS = frozenset([
    'tags',             # Ordered list
    'rating',           # A floating point number between 0 and 10
    'comments',         # A simple HTML enabled string
    'series',           # A simple string
    'series_index',     # A floating point number
    # Of the form { scheme1:value1, scheme2:value2}
    # For example: {'isbn':'123456789', 'doi':'xxxx', ... }
    'identifiers',
])

'''
The list of names that convert to identifiers when in get and set.
'''

TOP_LEVEL_IDENTIFIERS = frozenset([
```

```
    'isbn',
])

PUBLICATION_METADATA_FIELDS = frozenset([
    'title',            # title must never be None. Should be _('Unknown')
    # Pseudo field that can be set, but if not set is auto generated
    # from title and languages
    'title_sort',
    'authors',          # Ordered list. Must never be None, can be [_('Unknown')]
    'author_sort_map',  # Map of sort strings for each author
    # Pseudo field that can be set, but if not set is auto generated
    # from authors and languages
    'author_sort',
    'book_producer',
    'timestamp',        # Dates and times must be timezone aware
    'pubdate',
    'last_modified',
    'rights',
    # So far only known publication type is periodical:calibre
    # If None, means book
    'publication_type',
    'uuid',             # A UUID usually of type 4
    'languages',        # ordered list of languages in this publication
    'publisher',        # Simple string, no special semantics
    # Absolute path to image file encoded in filesystem_encoding
    'cover',
    # Of the form (format, data) where format is, for e.g. 'jpeg', 'png', 'gif'...
    'cover_data',
    # Either thumbnail data, or an object with the attribute
    # image_path which is the path to an image file, encoded
    # in filesystem_encoding
    'thumbnail',
    ])

BOOK_STRUCTURE_FIELDS = frozenset([
    # These are used by code, Null values are None.
    'toc', 'spine', 'guide', 'manifest',
    ])

USER_METADATA_FIELDS = frozenset([
    # A dict of dicts similar to field_metadata. Each field description dict
    # also contains a value field with the key #value#.
    'user_metadata',
])

DEVICE_METADATA_FIELDS = frozenset([
    'device_collections',   # Ordered list of strings
    'lpath',                # Unicode, / separated
    'size',                 # In bytes
    'mime',                 # Mimetype of the book file being represented

])

CALIBRE_METADATA_FIELDS = frozenset([
    'application_id',   # An application id, currently set to the db_id.
    'db_id',            # the calibre primary key of the item.
    'formats',          # list of formats (extensions) for this book
    # a dict of user category names, where the value is a list of item names
```

```
    # from the book that are in that category
    'user_categories',
    # a dict of author to an associated hyperlink
    'author_link_map',

    ]
)

ALL_METADATA_FIELDS =       SOCIAL_METADATA_FIELDS.union(
                            PUBLICATION_METADATA_FIELDS).union(
                            BOOK_STRUCTURE_FIELDS).union(
                            USER_METADATA_FIELDS).union(
                            DEVICE_METADATA_FIELDS).union(
                            CALIBRE_METADATA_FIELDS)

# All fields except custom fields
STANDARD_METADATA_FIELDS = SOCIAL_METADATA_FIELDS.union(
                            PUBLICATION_METADATA_FIELDS).union(
                            BOOK_STRUCTURE_FIELDS).union(
                            DEVICE_METADATA_FIELDS).union(
                            CALIBRE_METADATA_FIELDS)

# Metadata fields that smart update must do special processing to copy.
SC_FIELDS_NOT_COPIED =      frozenset(['title', 'title_sort', 'authors',
                                       'author_sort', 'author_sort_map',
                                       'cover_data', 'tags', 'languages',
                                       'identifiers'])

# Metadata fields that smart update should copy only if the source is not None
SC_FIELDS_COPY_NOT_NULL =  frozenset(['lpath', 'size', 'comments', 'thumbnail'])

# Metadata fields that smart update should copy without special handling
SC_COPYABLE_FIELDS =        SOCIAL_METADATA_FIELDS.union(
                            PUBLICATION_METADATA_FIELDS).union(
                            BOOK_STRUCTURE_FIELDS).union(
                            DEVICE_METADATA_FIELDS).union(
                            CALIBRE_METADATA_FIELDS) - \
                            SC_FIELDS_NOT_COPIED.union(
                            SC_FIELDS_COPY_NOT_NULL)

SERIALIZABLE_FIELDS =       SOCIAL_METADATA_FIELDS.union(
                            USER_METADATA_FIELDS).union(
                            PUBLICATION_METADATA_FIELDS).union(
                            CALIBRE_METADATA_FIELDS).union(
                            DEVICE_METADATA_FIELDS) - \
                            frozenset(['device_collections', 'formats',
                                'cover_data'])
                            # these are rebuilt when needed
```

All about using regular expressions in calibre

Regular expressions are features used in many places in calibre to perform sophisticated manipulation of ebook content and metadata. This tutorial is a gentle introduction to getting you started with using regular expressions in calibre.

Contents

- First, a word of warning and a word of courage (page 416)
- Where in calibre can you use regular expressions? (page 416)
- What on earth *is* a regular expression? (page 416)
- Care to explain? (page 417)
- That doesn't sound too bad. What's next? (page 417)
- Hey, neat! This is starting to make sense! (page 417)
- Well, these special characters are very neat and all, but what if I wanted to match a dot or a question mark? (page 418)
- So, what are the most useful sets? (page 418)
- But if I had a few varying strings I wanted to match, things get complicated? (page 418)
- You missed... (page 419)
- In the beginning, you said there was a way to make a regular expression case insensitive? (page 419)
- I think I'm beginning to understand these regular expressions now... how do I use them in calibre? (page 419)
  - Conversions (page 419)
  - Adding books (page 420)
  - Bulk editing metadata (page 420)
- Credits (page 420)

First, a word of warning and a word of courage

This is, inevitably, going to be somewhat technical- after all, regular expressions are a technical tool for doing technical stuff. I'm going to have to use some jargon and concepts that may seem complicated or convoluted. I'm going to try to explain those concepts as clearly as I can, but really can't do without using them at all. That being said, don't be discouraged by any jargon, as I've tried to explain everything new. And while regular expressions themselves may seem like an arcane, black magic (or, to be more prosaic, a random string of mumbo-jumbo letters and signs), I promise that they are not all that complicated. Even those who understand regular expressions really well have trouble reading the more complex ones, but writing them isn't as difficult- you construct the expression step by step. So, take a step and follow me into the rabbit hole.

Where in calibre can you use regular expressions?

There are a few places calibre uses regular expressions. There's the Search & Replace in conversion options, metadata detection from filenames in the import settings and Search & Replace when editing the metadata of books in bulk.

What on earth *is* a regular expression?

A regular expression is a way to describe sets of strings. A single regular expression cat *match* a number of different strings. This is what makes regular expression so powerful – they are a concise way of describing a potentially large number of variations.

Note: I'm using string here in the sense it is used in programming languages: a string of one or more characters, characters including actual characters, numbers, punctuation and so-called whitespace (linebreaks, tabulators etc.). Please note that generally, uppercase and lowercase characters are not considered the same, thus "a" being a different character from "A" and so forth. In calibre, regular expressions are case insensitive in the search bar, but not in the conversion options. There's a way to make every regular expression case insensitive, but we'll discuss that later. It gets complicated because regular expressions allow for variations in the strings it matches, so one expression can match multiple strings, which is why people bother using them at all. More on that in a bit.

Care to explain?

Well, that's why we're here. First, this is the most important concept in regular expressions: *A string by itself is a regular expression that matches itself*. That is to say, if I wanted to match the string `"Hello, World!"` using a regular expression, the regular expression to use would be `Hello, World!`. And yes, it really is that simple. You'll notice, though, that this *only* matches the exact string `"Hello, World!"`, not e.g. `"Hello, wOrld!"` or `"hello, world!"` or any other such variation.

That doesn't sound too bad. What's next?

Next is the beginning of the really good stuff. Remember where I said that regular expressions can match multiple strings? This is were it gets a little more complicated. Say, as a somewhat more practical exercise, the ebook you wanted to convert had a nasty footer counting the pages, like "Page 5 of 423". Obviously the page number would rise from 1 to 423, thus you'd have to match 423 different strings, right? Wrong, actually: regular expressions allow you to define sets of characters that are matched: To define a set, you put all the characters you want to be in the set into square brackets. So, for example, the set `[abc]` would match either the character "a", "b" or "c". *Sets will always only match one of the characters in the set*. They "understand" character ranges, that is, if you wanted to match all the lower case characters, you'd use the set `[a-z]` for lower- and uppercase characters you'd use `[a-zA-Z]` and so on. Got the idea? So, obviously, using the expression `Page [0-9] of 423` you'd be able to match the first 9 pages, thus reducing the expressions needed to three: The second expression `Page [0-9][0-9] of 423` would match all two-digit page numbers, and I'm sure you can guess what the third expression would look like. Yes, go ahead. Write it down.

Hey, neat! This is starting to make sense!

I was hoping you'd say that. But brace yourself, now it gets even better! We just saw that using sets, we could match one of several characters at once. But you can even repeat a character or set, reducing the number of expressions needed to handle the above page number example to one. Yes, ONE! Excited? You should be! It works like this: Some so-called special characters, "+", "?" and "*", *repeat the single element preceding them*. (Element means either a single character, a character set, an escape sequence or a group (we'll learn about those last two later)- in short, any single entity in a regular expression.) These characters are called wildcards or quantifiers. To be more precise, "?" matches *0 or 1* of the preceding element, "*" matches *0 or more* of the preceding element and "+" matches *1 or more* of the preceding element. A few examples: The expression `a?` would match either "" (which is the empty string, not strictly useful in this case) or "a", the expression `a*` would match "", "a", "aa" or any number of a's in a row, and, finally, the expression `a+` would match "a", "aa" or any number of a's in a row (Note: it wouldn't match the empty string!). Same deal for sets: The expression `[0-9]+` would match *every integer number there is*! I know what you're thinking, and you're right: If you use that in the above case of matching page numbers, wouldn't that be the single one expression to match all the page numbers? Yes, the expression `Page [0-9]+ of 423` would match every page number in that book!

Note: A note on these quantifiers: They generally try to match as much text as possible, so be careful when using them. This is called "greedy behaviour"- I'm sure you get why. It gets problematic when you, say, try to match a tag. Consider, for example, the string `"<p class="calibre2">Title here</p>"` and let's say you'd want to match the opening tag (the part between the first pair of angle brackets, a little more on tags later). You'd think that the expression `<p.*>` would match that tag, but actually, it matches the whole string! (The character "." is another special character. It matches anything *except* linebreaks, so, basically, the expression `.*` would match any single line you can think of.) Instead, try using `<p.*?>` which makes the quantifier `"*"` non-greedy. That expression would only match the first opening tag, as intended. There's actually another way to accomplish this: The expression `<p[^>]*>` will

match that same opening tag- you'll see why after the next section. Just note that there quite frequently is more than one way to write a regular expression.

Well, these special characters are very neat and all, but what if I wanted to match a dot or a question mark?

You can of course do that: Just put a backslash in front of any special character and it is interpreted as the literal character, without any special meaning. This pair of a backslash followed by a single character is called an escape sequence, and the act of putting a backslash in front of a special character is called escaping that character. An escape sequence is interpreted as a single element. There are of course escape sequences that do more than just escaping special characters, for example `"\t"` means a tabulator. We'll get to some of the escape sequences later. Oh, and by the way, concerning those special characters: Consider any character we discuss in this introduction as having some function to be special and thus needing to be escaped if you want the literal character.

So, what are the most useful sets?

Knew you'd ask. Some useful sets are `[0-9]` matching a single number, `[a-z]` matching a single lowercase letter, `[A-Z]` matching a single uppercase letter, `[a-zA-Z]` matching a single letter and `[a-zA-Z0-9]` matching a single letter or number. You can also use an escape sequence as shorthand:

```
\d is equivalent to [0-9]
\w is equivalent to [a-zA-Z0-9_]
\s is equivalent to any whitespace
```

Note: "Whitespace" is a term for anything that won't be printed. These characters include space, tabulator, line feed, form feed and carriage return.

As a last note on sets, you can also define a set as any character *but* those in the set. You do that by including the character `"^"` as the *very first character in the set*. Thus, `[^a]` would match any character excluding "a". That's called complementing the set. Those escape sequence shorthands we saw earlier can also be complemented: `"\D"` means any non-number character, thus being equivalent to `[^0-9]`. The other shorthands can be complemented by, you guessed it, using the respective uppercase letter instead of the lowercase one. So, going back to the example `<p[^>]*>` from the previous section, now you can see that the character set it's using tries to match any character except for a closing angle bracket.

But if I had a few varying strings I wanted to match, things get complicated?

Fear not, life still is good and easy. Consider this example: The book you're converting has "Title" written on every odd page and "Author" written on every even page. Looks great in print, right? But in ebooks, it's annoying. You can group whole expressions in normal parentheses, and the character `"|"` will let you match *either* the expression to its right *or* the one to its left. Combine those and you're done. Too fast for you? Okay, first off, we group the expressions for odd and even pages, thus getting `(Title)(Author)` as our two needed expressions. Now we make things simpler by using the vertical bar (`"|"` is called the vertical bar character): If you use the expression `(Title|Author)` you'll either get a match for "Title" (on the odd pages) or you'd match "Author" (on the even pages). Well, wasn't that easy?

You can, of course, use the vertical bar without using grouping parentheses, as well. Remember when I said that quantifiers repeat the element preceding them? Well, the vertical bar works a little differently: The expression "Title|Author" will also match either the string "Title" or the string "Author", just as the above example using grouping. *The vertical bar selects between the entire expression preceding and following it*. So, if you wanted to match the strings "Calibre" and "calibre" and wanted to select only between the upper- and lowercase "c", you'd have to use the expression `(c|C)alibre`, where the grouping ensures that only the "c" will be selected. If you were to use

`c|Calibre`, you'd get a match on the string "c" or on the string "Calibre", which isn't what we wanted. In short: If in doubt, use grouping together with the vertical bar.

You missed...

... wait just a minute, there's one last, really neat thing you can do with groups. If you have a group that you previously matched, you can use references to that group later in the expression: Groups are numbered starting with 1, and you reference them by escaping the number of the group you want to reference, thus, the fifth group would be referenced as `\5`. So, if you searched for `([^ ]+) \1` in the string "Test Test", you'd match the whole string!

In the beginning, you said there was a way to make a regular expression case insensitive?

Yes, I did, thanks for paying attention and reminding me. You can tell calibre how you want certain things handled by using something called flags. You include flags in your expression by using the special construct `(?flags go here)` where, obviously, you'd replace "flags go here" with the specific flags you want. For ignoring case, the flag is `i`, thus you include `(?i)` in your expression. Thus, `test(?i)` would match "Test", "tEst", "TEst" and any case variation you could think of.

Another useful flag lets the dot match any character at all, *including* the newline, the flag `s`. If you want to use multiple flags in an expression, just put them in the same statement: `(?is)` would ignore case and make the dot match all. It doesn't matter which flag you state first, `(?si)` would be equivalent to the above. By the way, good places for putting flags in your expression would be either the very beginning or the very end. That way, they don't get mixed up with anything else.

I think I'm beginning to understand these regular expressions now... how do I use them in calibre?

Conversions Let's begin with the conversion settings, which is really neat. In the Search and Replace part, you can input a regexp (short for regular expression) that describes the string that will be replaced during the conversion. The neat part is the wizard. Click on the wizard staff and you get a preview of what calibre "sees" during the conversion process. Scroll down to the string you want to remove, select and copy it, paste it into the regexp field on top of the window. If there are variable parts, like page numbers or so, use sets and quantifiers to cover those, and while you're at it, remember to escape special characters, if there are some. Hit the button labeled *Test* and calibre highlights the parts it would replace were you to use the regexp. Once you're satisfied, hit OK and convert. Be careful if your conversion source has tags like this example:

```
Maybe, but the cops feel like you do, Anita. What's one more dead vampire?
New laws don't change that. </p>
<p class="calibre4"> <b class="calibre2">Generated by ABC Amber LIT Conv
<a href="http://www.processtext.com/abclit.html" class="calibre3">erter,
http://www.processtext.com/abclit.html</a></b></p>
<p class="calibre4"> It had only been two years since Addison v. Clark.
The court case gave us a revised version of what life was
```

(shamelessly ripped out of this thread[185]). You'd have to remove some of the tags as well. In this example, I'd recommend beginning with the tag `<b class="calibre2">`, now you have to end with the corresponding closing tag (opening tags are `<tag>`, closing tags are `</tag>`), which is simply the next `</b>` in this case. (Refer to a good HTML manual or ask in the forum if you are unclear on this point.) The opening tag can be described using `<b.*?>`, the closing tag using `</b>`, thus we could remove everything between those tags using `<b.*?>.*?</b>`. But using this expression would be a bad idea, because it removes everything enclosed by <b>-tags (which, by the way, render the enclosed text in bold print), and it's a fair bet that we'll remove portions of the book in this way. Instead, include the beginning of the enclosed string as well, making the regular expression

[185] http://www.mobileread.com/forums/showthread.php?t=75594"

`<b.*?>\s*Generated\s+by\s+ABC\s+Amber\s+LIT.*?</b>` The `\s` with quantifiers are included here instead of explicitly using the spaces as seen in the string to catch any variations of the string that might occur. Remember to check what calibre will remove to make sure you don't remove any portions you want to keep if you test a new expression. If you only check one occurrence, you might miss a mismatch somewhere else in the text. Also note that should you accidentally remove more or fewer tags than you actually wanted to, calibre tries to repair the damaged code after doing the removal.

Adding books Another thing you can use regular expressions for is to extract metadata from filenames. You can find this feature in the "Adding books" part of the settings. There's a special feature here: You can use field names for metadata fields, for example `(?P<title>)` would indicate that calibre uses this part of the string as book title. The allowed field names are listed in the windows, together with another nice test field. An example: Say you want to import a whole bunch of files named like `Classical Texts: The Divine Comedy by Dante Alighieri.mobi`. (Obviously, this is already in your library, since we all love classical italian poetry) or `Science Fiction epics: The Foundation Trilogy by Isaac Asimov.epub`. This is obviously a naming scheme that calibre won't extract any meaningful data out of - its standard expression for extracting metadata is `(?P<title>.+) - (?P<author>[^_]+)`. A regular expression that works here would be `[a-zA-Z]+: (?P<title>.+) by (?P<author>.+)`. Please note that, inside the group for the metadata field, you need to use expressions to describe what the field actually matches. And also note that, when using the test field calibre provides, you need to add the file extension to your testing filename, otherwise you won't get any matches at all, despite using a working expression.

Bulk editing metadata The last part is regular expression search and replace in metadata fields. You can access this by selecting multiple books in the library and using bulk metadata edit. Be very careful when using this last feature, as it can do **Very Bad Things** to your library! Doublecheck that your expressions do what you want them to using the test fields, and only mark the books you really want to change! In the regular expression search mode, you can search in one field, replace the text with something and even write the result into another field. A practical example: Say your library contained the books of Frank Herbert's Dune series, named after the fashion `Dune 1 - Dune`, `Dune 2 - Dune Messiah` and so on. Now you want to get `Dune` into the series field. You can do that by searching for `(.*?) \d+ - .*` in the title field and replacing it with `\1` in the series field. See what I did there? That's a reference to the first group you're replacing the series field with. Now that you have the series all set, you only need to do another search for `.*? -` in the title field and replace it with `""` (an empty string), again in the title field, and your metadata is all neat and tidy. Isn't that great? By the way, instead of replacing the entire field, you can also append or prepend to the field, so, if you *wanted* the book title to be prepended with series info, you could do that as well. As you by now have undoubtedly noticed, there's a checkbox labeled *Case sensitive*, so you won't have to use flags to select behaviour here.

Well, that just about concludes the very short introduction to regular expressions. Hopefully I'll have shown you enough to at least get you started and to enable you to continue learning by yourself- a good starting point would be the Python documentation for regexps[186].

One last word of warning, though: Regexps are powerful, but also really easy to get wrong. calibre provides really great testing possibilities to see if your expressions behave as you expect them to. Use them. Try not to shoot yourself in the foot. (God, I love that expression...) But should you, despite the warning, injure your foot (or any other body parts), try to learn from it.

Credits

Thanks for helping with tips, corrections and such:

- ldolse
- kovidgoyal

[186] http://docs.python.org/library/re.html

- chaley
- dwanthny
- kacir
- Starson17

For more about regexps see The Python User Manual[187].

Integrating the calibre content server into other servers

Here, we will show you how to integrate the calibre content server into another server. The most common reason for this is to make use of SSL or more sophisticated authentication. There are two main techniques: Running the calibre content server as a standalone process and using a reverse proxy to connect it with your main server or running the content server in process in your main server with WSGI. The examples below are all for Apache 2.x on linux, but should be easily adaptable to other platforms.

Contents

- Using a reverse proxy (page 421)
- In process (page 422)

Note: This only applies to calibre releases >= 0.7.25

Using a reverse proxy

A reverse proxy is when your normal server accepts incoming requests and passes them onto the calibre server. It then reads the response from the calibre server and forwards it to the client. This means that you can simply run the calibre server as normal without trying to integrate it closely with your main server, and you can take advantage of whatever authentication systems you main server has in place. This is the simplest approach as it allows you to use the binary calibre install with no external dependencies/system integration requirements. Below, is an example of how to achieve this with Apache as your main server, but it will work with any server that supports Reverse Proxies.

First start the calibre content server as shown below:

```
calibre-server --url-prefix /calibre --port 8080
```

The key parameter here is `--url-prefix /calibre`. This causes the content server to serve all URLs prefixed by calibre. To see this in action, visit `http://localhost:8080/calibre` in your browser. You should see the normal content server website, but now it will run under /calibre.

Now suppose you are using Apache as your main server. First enable the proxy modules in apache, by adding the following to `httpd.conf`:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

The exact technique for enabling the proxy modules will vary depending on your Apache installation. Once you have the proxy modules enabled, add the following rules to httpd.conf (or if you are using virtual hosts to the conf file for the virtual host in question:

[187] http://docs.python.org/library/re.html

```
RewriteEngine on
RewriteRule ^/calibre/(.*) http://localhost:8080/calibre/$1 [proxy]
RewriteRule ^/calibre http://localhost:8080 [proxy]
SetEnv force-proxy-request-1.0 1
SetEnv proxy-nokeepalive 1
```

That's all, you will now be able to access the calibre Content Server under the /calibre URL in your apache server. The above rules pass all requests under /calibre to the calibre server running on port 8080 and thanks to the –url-prefix option above, the calibre server handles them transparently.

Note: If you are willing to devote an entire VirtualHost to the content server, then there is no need to use –url-prefix and RewriteRule, instead just use the ProxyPass directive.

Note: The server engine calibre uses, CherryPy, can have trouble with proxying and KeepAlive requests, so turn them off in Apache, with the SetEnv directives shown above.

In process

The calibre content server can be run directly, in process, inside a host server like Apache using the WSGI framework.

Note: For this to work, all the dependencies needed by calibre must be installed on your system. On linux, this can be achieved fairly easily by installing the distribution provided calibre package (provided it is up to date).

First, we have to create a WSGI *adapter* for the calibre content server. Here is a template you can use for the purpose. Replace the paths as directed in the comments

```
# WSGI script file to run calibre content server as a WSGI app

import sys, os


# You can get the paths referenced here by running
# calibre-debug --paths
# on your server

# The first entry from CALIBRE_PYTHON_PATH
sys.path.insert(0, '/home/kovid/work/calibre/src')

# CALIBRE_RESOURCES_PATH
sys.resources_location = '/home/kovid/work/calibre/resources'

# CALIBRE_EXTENSIONS_PATH
sys.extensions_location = '/home/kovid/work/calibre/src/calibre/plugins'

# Path to directory containing calibre executables
sys.executables_location = '/usr/bin'

# Path to a directory for which the server has read/write permissions
# calibre config will be stored here
os.environ['CALIBRE_CONFIG_DIRECTORY'] = '/var/www/localhost/calibre-config'

del sys
del os
```

```
from calibre.library.server.main import create_wsgi_app
application = create_wsgi_app(
        # The mount point of this WSGI application (i.e. the first argument to
        # the WSGIScriptAlias directive). Set to empty string is mounted at /
        prefix='/calibre',

        # Path to the calibre library to be served
        # The server process must have write permission for all files/dirs
        # in this directory or BAD things will happen
        path_to_library='/home/kovid/documents/demo library'
)

del create_wsgi_app
```

Save this adapter as `calibre-wsgi-adpater.py` somewhere your server will have access to it.

Let's suppose that we want to use WSGI in Apache. First enable WSGI in Apache by adding the following to `httpd.conf`:

```
LoadModule proxy_module modules/mod_wsgi.so
```

The exact technique for enabling the wsgi module will vary depending on your Apache installation. Once you have the proxy modules enabled, add the following rules to httpd.conf (or if you are using virtual hosts to the conf file for the virtual host in question:

```
WSGIScriptAlias /calibre /var/www/localhost/cgi-bin/calibre-wsgi-adapter.py
```

Change the path to `calibre-wsgi-adapter.py` to wherever you saved it previously (make sure Apache has access to it).

That's all, you will now be able to access the calibre Content Server under the /calibre URL in your apache server.

Note: For more help with using mod_wsgi in Apache, see mod_wsgi[188].

Writing your own plugins to extend calibre's functionality

calibre has a very modular design. Almost all functionality in calibre comes in the form of plugins. Plugins are used for conversion, for downloading news (though these are called recipes), for various components of the user interface, to connect to different devices, to process files when adding them to calibre and so on. You can get a complete list of all the built-in plugins in calibre by going to *Preferences->Plugins*.

Here, we will teach you how to create your own plugins to add new features to calibre.

[188] http://code.google.com/p/modwsgi/wiki/WhereToGetHelp

Contents

- Anatomy of a calibre plugin (page 424)
- A User Interface plugin (page 425)
  - __init__.py (page 425)
  - ui.py (page 427)
  - main.py (page 428)
  - Getting resources from the plugin zip file (page 430)
  - Enabling user configuration of your plugin (page 431)
- The plugin API (page 432)
- Debugging plugins (page 433)
- More plugin examples (page 433)
- Sharing your plugins with others (page 433)

Note: This only applies to calibre releases >= 0.8.60

Anatomy of a calibre plugin

A calibre plugin is very simple, it's just a zip file that contains some python code and any other resources like image files needed by the plugin. Without further ado, let's see a basic example.

Suppose you have an installation of calibre that you are using to self publish various e-documents in EPUB and MOBI formats. You would like all files generated by calibre to have their publisher set as "Hello world", here's how to do it. Create a file named `__init__.py` (this is a special name and must always be used for the main file of your plugin) and enter the following Python code into it:

```
import os
from calibre.customize import FileTypePlugin

class HelloWorld(FileTypePlugin):

    name                = 'Hello World Plugin' # Name of the plugin
    description         = 'Set the publisher to Hello World for all new conversions'
    supported_platforms = ['windows', 'osx', 'linux'] # Platforms this plugin will run on
    author              = 'Acme Inc.' # The author of this plugin
    version             = (1, 0, 0)   # The version number of this plugin
    file_types          = set(['epub', 'mobi']) # The file types that this plugin will be applied to
    on_postprocess      = True # Run this plugin after conversion is complete
    minimum_calibre_version = (0, 7, 53)

    def run(self, path_to_ebook):
        from calibre.ebooks.metadata.meta import get_metadata, set_metadata
        file = open(path_to_ebook, 'r+b')
        ext  = os.path.splitext(path_to_ebook)[-1][1:].lower()
        mi = get_metadata(file, ext)
        mi.publisher = 'Hello World'
        set_metadata(file, mi, ext)
        return path_to_ebook
```

That's all. To add this code to calibre as a plugin, simply run the following in the directory in which you created `__init__.py`:

```
calibre-customize -b .
```

Note: On OS X you have to first install the calibre command line tools, by going to *Preferences->Miscellaneous* and clicking the *Install command line tools* button.

You can download the Hello World plugin from helloworld_plugin.zip[189].

Every time you use calibre to convert a book, the plugin's `run()` method will be called and the converted book will have its publisher set to "Hello World". This is a trivial plugin, lets move on to a more complex example that actually adds a component to the user interface.

A User Interface plugin

This plugin will be spread over a few files (to keep the code clean). It will show you how to get resources (images or data files) from the plugin zip file, allow users to configure your plugin, how to create elements in the calibre user interface and how to access and query the books database in calibre.

You can download this plugin from interface_demo_plugin.zip[190]

The first thing to note is that this zip file has a lot more files in it, explained below, pay particular attention to `plugin-import-name-interface_demo.txt`.

plugin-import-name-interface_demo.txt An empty text file used to enable the multi-file plugin magic. This file must be present in all plugins that use more than one .py file. It should be empty and its filename must be of the form: plugin-import-name-**some_name**.txt The presence of this file allows you to import code from the .py files present inside the zip file, using a statement like:

```
from calibre_plugins.some_name.some_module import some_object
```

The prefix `calibre_plugins` must always be present. `some_name` comes from the filename of the empty text file. `some_module` refers to `some_module.py` file inside the zip file. Note that this importing is just as powerful as regular python imports. You can create packages and subpackages of .py modules inside the zip file, just like you would normally (by defining __init__.py in each sub directory), and everything should Just Work.

The name you use for `some_name` enters a global namespace shared by all plugins, **so make it as unique as possible**. But remember that it must be a valid python identifier (only alphabets, numbers and the underscore).

__init__.py As before, the file that defines the plugin class

main.py This file contains the actual code that does something useful

ui.py This file defines the interface part of the plugin

images/icon.png The icon for this plugin

about.txt A text file with information about the plugin

Now let's look at the code.

__init__.py First, the obligatory `__init__.py` to define the plugin metadata:

```
# The class that all Interface Action plugin wrappers must inherit from
from calibre.customize import InterfaceActionBase

class InterfacePluginDemo(InterfaceActionBase):
    '''
```

[189] http://calibre-ebook.com/downloads/helloworld_plugin.zip
[190] http://calibre-ebook.com/downloads/interface_demo_plugin.zip

```
    This class is a simple wrapper that provides information about the actual
    plugin class. The actual interface plugin class is called InterfacePlugin
    and is defined in the ui.py file, as specified in the actual_plugin field
    below.

    The reason for having two classes is that it allows the command line
    calibre utilities to run without needing to load the GUI libraries.
    '''
    name                = 'Interface Plugin Demo'
    description         = 'An advanced plugin demo'
    supported_platforms = ['windows', 'osx', 'linux']
    author              = 'Kovid Goyal'
    version             = (1, 0, 0)
    minimum_calibre_version = (0, 7, 53)

    #: This field defines the GUI plugin class that contains all the code
    #: that actually does something. Its format is module_path:class_name
    #: The specified class must be defined in the specified module.
    actual_plugin       = 'calibre_plugins.interface_demo.ui:InterfacePlugin'

    def is_customizable(self):
        '''
        This method must return True to enable customization via
        Preferences->Plugins
        '''
        return True

    def config_widget(self):
        '''
        Implement this method and :meth:`save_settings` in your plugin to
        use a custom configuration dialog.

        This method, if implemented, must return a QWidget. The widget can have
        an optional method validate() that takes no arguments and is called
        immediately after the user clicks OK. Changes are applied if and only
        if the method returns True.

        If for some reason you cannot perform the configuration at this time,
        return a tuple of two strings (message, details), these will be
        displayed as a warning dialog to the user and the process will be
        aborted.

        The base class implementation of this method raises NotImplementedError
        so by default no user configuration is possible.
        '''
        # It is important to put this import statement here rather than at the
        # top of the module as importing the config class will also cause the
        # GUI libraries to be loaded, which we do not want when using calibre
        # from the command line
        from calibre_plugins.interface_demo.config import ConfigWidget
        return ConfigWidget()

    def save_settings(self, config_widget):
        '''
        Save the settings specified by the user with config_widget.

        :param config_widget: The widget returned by :meth:`config_widget`.
        '''
```

```
        config_widget.save_settings()

        # Apply the changes
        ac = self.actual_plugin_
        if ac is not None:
            ac.apply_settings()
```

The only noteworthy feature is the field `actual_plugin`. Since calibre has both command line and GUI interfaces, GUI plugins like this one should not load any GUI libraries in __init__.py. The actual_plugin field does this for you, by telling calibre that the actual plugin is to be found in another file inside your zip archive, which will only be loaded in a GUI context.

Remember that for this to work, you must have a plugin-import-name-some_name.txt file in your plugin zip file, as discussed above.

Also there are a couple of methods for enabling user configuration of the plugin. These are discussed below.

ui.py Now let's look at ui.py which defines the actual GUI plugin. The source code is heavily commented and should be self explanatory:

```
# The class that all interface action plugins must inherit from
from calibre.gui2.actions import InterfaceAction
from calibre_plugins.interface_demo.main import DemoDialog

class InterfacePlugin(InterfaceAction):

    name = 'Interface Plugin Demo'

    # Declare the main action associated with this plugin
    # The keyboard shortcut can be None if you dont want to use a keyboard
    # shortcut. Remember that currently calibre has no central management for
    # keyboard shortcuts, so try to use an unusual/unused shortcut.
    action_spec = ('Interface Plugin Demo', None,
            'Run the Interface Plugin Demo', 'Ctrl+Shift+F1')

    def genesis(self):
        # This method is called once per plugin, do initial setup here

        # Set the icon for this interface action
        # The get_icons function is a builtin function defined for all your
        # plugin code. It loads icons from the plugin zip file. It returns
        # QIcon objects, if you want the actual data, use the analogous
        # get_resources builtin function.
        #
        # Note that if you are loading more than one icon, for performance, you
        # should pass a list of names to get_icons. In this case, get_icons
        # will return a dictionary mapping names to QIcons. Names that
        # are not found in the zip file will result in null QIcons.
        icon = get_icons('images/icon.png')

        # The qaction is automatically created from the action_spec defined
        # above
        self.qaction.setIcon(icon)
        self.qaction.triggered.connect(self.show_dialog)

    def show_dialog(self):
        # The base plugin object defined in __init__.py
        base_plugin_object = self.interface_action_base_plugin
```

```
        # Show the config dialog
        # The config dialog can also be shown from within
        # Preferences->Plugins, which is why the do_user_config
        # method is defined on the base plugin class
        do_user_config = base_plugin_object.do_user_config

        # self.gui is the main calibre GUI. It acts as the gateway to access
        # all the elements of the calibre user interface, it should also be the
        # parent of the dialog
        d = DemoDialog(self.gui, self.qaction.icon(), do_user_config)
        d.show()

    def apply_settings(self):
        from calibre_plugins.interface_demo.config import prefs
        # In an actual non trivial plugin, you would probably need to
        # do something based on the settings in prefs
        prefs
```

main.py The actual logic to implement the Interface Plugin Demo dialog.

```
from PyQt4.Qt import QDialog, QVBoxLayout, QPushButton, QMessageBox, QLabel

from calibre_plugins.interface_demo.config import prefs

class DemoDialog(QDialog):

    def __init__(self, gui, icon, do_user_config):
        QDialog.__init__(self, gui)
        self.gui = gui
        self.do_user_config = do_user_config

        # The current database shown in the GUI
        # db is an instance of the class LibraryDatabase2 from database.py
        # This class has many, many methods that allow you to do a lot of
        # things.
        self.db = gui.current_db

        self.l = QVBoxLayout()
        self.setLayout(self.l)

        self.label = QLabel(prefs['hello_world_msg'])
        self.l.addWidget(self.label)

        self.setWindowTitle('Interface Plugin Demo')
        self.setWindowIcon(icon)

        self.about_button = QPushButton('About', self)
        self.about_button.clicked.connect(self.about)
        self.l.addWidget(self.about_button)

        self.marked_button = QPushButton(
            'Show books with only one format in the calibre GUI', self)
        self.marked_button.clicked.connect(self.marked)
        self.l.addWidget(self.marked_button)

        self.view_button = QPushButton(
            'View the most recently added book', self)
```

```
        self.view_button.clicked.connect(self.view)
        self.l.addWidget(self.view_button)

        self.update_metadata_button = QPushButton(
            'Update metadata in a book\'s files', self)
        self.update_metadata_button.clicked.connect(self.update_metadata)
        self.l.addWidget(self.update_metadata_button)

        self.conf_button = QPushButton(
                'Configure this plugin', self)
        self.conf_button.clicked.connect(self.config)
        self.l.addWidget(self.conf_button)

        self.resize(self.sizeHint())

    def about(self):
        # Get the about text from a file inside the plugin zip file
        # The get_resources function is a builtin function defined for all your
        # plugin code. It loads files from the plugin zip file. It returns
        # the bytes from the specified file.
        #
        # Note that if you are loading more than one file, for performance, you
        # should pass a list of names to get_resources. In this case,
        # get_resources will return a dictionary mapping names to bytes. Names that
        # are not found in the zip file will not be in the returned dictionary.
        text = get_resources('about.txt')
        QMessageBox.about(self, 'About the Interface Plugin Demo',
                text.decode('utf-8'))

    def marked(self):
        ''' Show books with only one format '''
        fmt_idx = self.db.FIELD_MAP['formats']
        matched_ids = set()
        for record in self.db.data.iterall():
            # Iterate over all records
            fmts = record[fmt_idx]
            # fmts is either None or a comma separated list of formats
            if fmts and ',' not in fmts:
                matched_ids.add(record[0])
        # Mark the records with the matching ids
        self.db.set_marked_ids(matched_ids)

        # Tell the GUI to search for all marked records
        self.gui.search.setEditText('marked:true')
        self.gui.search.do_search()

    def view(self):
        ''' View the most recently added book '''
        most_recent = most_recent_id = None
        timestamp_idx = self.db.FIELD_MAP['timestamp']

        for record in self.db.data:
            # Iterate over all currently showing records
            timestamp = record[timestamp_idx]
            if most_recent is None or timestamp > most_recent:
                most_recent = timestamp
                most_recent_id = record[0]
```

```
        if most_recent_id is not None:
            # Get the row number of the id as shown in the GUI
            row_number = self.db.row(most_recent_id)
            # Get a reference to the View plugin
            view_plugin = self.gui.iactions['View']
            # Ask the view plugin to launch the viewer for row_number
            view_plugin._view_books([row_number])

    def update_metadata(self):
        '''
        Set the metadata in the files in the selected book's record to
        match the current metadata in the database.
        '''
        from calibre.ebooks.metadata.meta import set_metadata
        from calibre.gui2 import error_dialog, info_dialog

        # Get currently selected books
        rows = self.gui.library_view.selectionModel().selectedRows()
        if not rows or len(rows) == 0:
            return error_dialog(self.gui, 'Cannot update metadata',
                                'No books selected', show=True)
        # Map the rows to book ids
        ids = list(map(self.gui.library_view.model().id, rows))
        for book_id in ids:
            # Get the current metadata for this book from the db
            mi = self.db.get_metadata(book_id, index_is_id=True,
                    get_cover=True, cover_as_data=True)
            fmts = self.db.formats(book_id, index_is_id=True)
            if not fmts: continue
            for fmt in fmts.split(','):
                fmt = fmt.lower()
                # Get a python file object for the format. This will be either
                # an in memory file or a temporary on disk file
                ffile = self.db.format(book_id, fmt, index_is_id=True,
                        as_file=True)
                # Set metadata in the format
                set_metadata(ffile, mi, fmt)
                ffile.seek(0)
                # Now replace the file in the calibre library with the updated
                # file. We dont use add_format_with_hooks as the hooks were
                # already run when the file was first added to calibre.
                ffile.name = 'xxx'  # add_format() will not work if the file
                                    # path of the file being added is the same
                                    # as the path of the file being replaced
                self.db.add_format(book_id, fmt, ffile, index_is_id=True)

        info_dialog(self, 'Updated files',
                'Updated the metadata in the files of %d book(s)'%len(ids),
                show=True)

    def config(self):
        self.do_user_config(parent=self)
        # Apply the changes
        self.label.setText(prefs['hello_world_msg'])
```

Getting resources from the plugin zip file calibre's plugin loading system defines a couple of built-in functions that allow you to conveniently get files from the plugin zip file.

get_resources(name_or_list_of_names) This function should be called with a list of paths to files inside the zip file. For example to access the file icon.png in the directory images in the zip file, you would use: `images/icon.png`. Always use a forward slash as the path separator, even on windows. When you pass in a single name, the function will return the raw bytes of that file or None if the name was not found in the zip file. If you pass in more than one name then it returns a dict mapping the names to bytes. If a name is not found, it will not be present in the returned dict.

get_icons(name_or_list_of_names) A convenience wrapper for get_resources() that creates QIcon objects from the raw bytes returned by get_resources. If a name is not found in the zip file the corresponding QIcon will be null.

Enabling user configuration of your plugin To allow users to configure your plugin, you must define three methods in your base plugin class, '**is_customizable**, **config_widget** and **save_settings** as shown below:

```
def is_customizable(self):
    '''
    This method must return True to enable customization via
    Preferences->Plugins
    '''
    return True

def config_widget(self):
    '''
    Implement this method and :meth:`save_settings` in your plugin to
    use a custom configuration dialog.

    This method, if implemented, must return a QWidget. The widget can have
    an optional method validate() that takes no arguments and is called
    immediately after the user clicks OK. Changes are applied if and only
    if the method returns True.

    If for some reason you cannot perform the configuration at this time,
    return a tuple of two strings (message, details), these will be
    displayed as a warning dialog to the user and the process will be
    aborted.

    The base class implementation of this method raises NotImplementedError
    so by default no user configuration is possible.
    '''
    # It is important to put this import statement here rather than at the
    # top of the module as importing the config class will also cause the
    # GUI libraries to be loaded, which we do not want when using calibre
    # from the command line
    from calibre_plugins.interface_demo.config import ConfigWidget
    return ConfigWidget()

def save_settings(self, config_widget):
    '''
    Save the settings specified by the user with config_widget.

    :param config_widget: The widget returned by :meth:`config_widget`.
    '''
    config_widget.save_settings()

    # Apply the changes
    ac = self.actual_plugin_
    if ac is not None:
        ac.apply_settings()
```

calibre has many different ways to store configuration data (a legacy of its long history). The recommended way is to use the **JSONConfig** class, which stores your configuration information in a .json file.

The code to manage configuration data in the demo plugin is in config.py:

```
from PyQt4.Qt import QWidget, QHBoxLayout, QLabel, QLineEdit

from calibre.utils.config import JSONConfig

# This is where all preferences for this plugin will be stored
# Remember that this name (i.e. plugins/interface_demo) is also
# in a global namespace, so make it as unique as possible.
# You should always prefix your config file name with plugins/,
# so as to ensure you dont accidentally clobber a calibre config file
prefs = JSONConfig('plugins/interface_demo')

# Set defaults
prefs.defaults['hello_world_msg'] = 'Hello, World!'

class ConfigWidget(QWidget):

    def __init__(self):
        QWidget.__init__(self)
        self.l = QHBoxLayout()
        self.setLayout(self.l)

        self.label = QLabel('Hello world &message:')
        self.l.addWidget(self.label)

        self.msg = QLineEdit(self)
        self.msg.setText(prefs['hello_world_msg'])
        self.l.addWidget(self.msg)
        self.label.setBuddy(self.msg)

    def save_settings(self):
        prefs['hello_world_msg'] = unicode(self.msg.text())
```

The `prefs` object is now available throughout the plugin code by a simple:

```
from calibre_plugins.interface_demo.config import prefs
```

You can see the `prefs` object being used in main.py:

```
    def config(self):
        self.do_user_config(parent=self)
        # Apply the changes
        self.label.setText(prefs['hello_world_msg'])
```

The plugin API

As you may have noticed above, a plugin in calibre is a class. There are different classes for the different types of plugins in calibre. Details on each class, including the base class of all plugins can be found in *API Documentation for plugins* (page 441).

Your plugin is almost certainly going to use code from calibre. To learn how to find various bits of functionality in the calibre code base, read the section on the calibre *Code layout* (page 512).

Debugging plugins

The first, most important step is to run calibre in debug mode. You can do this from the command line with:

```
calibre-debug -g
```

Or from within calibre by right-clicking the preferences button or using the *Ctrl+Shift+R* keyboard shortcut.

When running from the command line, debug output will be printed to the console, when running from within calibre the output will go to a txt file.

You can insert print statements anywhere in your plugin code, they will be output in debug mode. Remember, this is python, you really shouldn't need anything more than print statements to debug ;) I developed all of calibre using just this debugging technique.

You can quickly test changes to your plugin by using the following command line:

```
calibre-debug -s; calibre-customize -b /path/to/your/plugin/directory; calibre
```

This will shutdown a running calibre, wait for the shutdown to complete, then update your plugin in calibre and relaunch calibre.

More plugin examples

You can find a list of many, sophisticated calibre plugins here[191].

Sharing your plugins with others

If you would like to share the plugins you have created with other users of calibre, post your plugin in a new thread in the calibre plugins forum[192].

Typesetting Math in ebooks

The calibre ebook viewer has the ability to display math embedded in ebooks (ePub and HTML files). You can typeset the math directly with TeX or MathML or AsciiMath. The calibre viewer uses the excellent MathJax[193] library to do this. This is a brief tutorial on creating ebooks with math in them that work well with the calibre viewer.

Note: This only applies to calibre version 0.8.66 and newer

A simple HTML file with mathematics

You can write mathematics inline inside a simple HTML file and the calibre viewer will render it into properly typeset mathematics. In the example below, we use TeX notation for mathematics. You will see that you can use normal TeX commands, with the small caveat that ampersands and less than and greater than signs have to be written as & < and > respectively.

The first step is to tell calibre that this will contains maths. You do this by adding the following snippet of code to the <head> section of the HTML file:

[191] http://www.mobileread.com/forums/showthread.php?t=118764
[192] http://www.mobileread.com/forums/forumdisplay.php?f=237
[193] http://www.mathjax.org

```
<script type="text/x-mathjax-config"></script>
```

That's it, now you can type mathematics just as you would in a .tex file. For example, here are Lorentz's equations:

```
<h2>The Lorenz Equations</h2>

<p>
\begin{align}
\dot{x} &amp; = \sigma(y-x) \\
\dot{y} &amp; = \rho x - y - xz \\
\dot{z} &amp; = -\beta z + xy
\end{align}
</p>
```

This snippet looks like the following screen shot in the calibre viewer.

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

Figure 1.4: *The Lorenz Equations*

The complete HTML file, with more equations and inline mathematics is reproduced below. You can convert this HTML file to EPUB in calibre to end up with an ebook you can distribute easily to other people.

```
<!DOCTYPE html>
<html>
<!-- Copyright (c) 2012 Design Science, Inc. -->
<head>
<title>Math Test Page</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />

<!-- This script tag is needed to make calibre's ebook-viewer recpgnize that this file needs math typ
<script type="text/x-mathjax-config">
    // This line adds numbers to all equations automatically, unless explicitly suppressed.
    MathJax.Hub.Config({ TeX: { equationNumbers: {autoNumber: "all"} } });
</script>

<style>
h1 {text-align:center}
h2 {
  font-weight: bold;
  background-color: #DDDDDD;
  padding: .2em .5em;
  margin-top: 1.5em;
  border-top: 3px solid #666666;
  border-bottom: 2px solid #999999;
}
</style>
</head>
<body>

<h1>Sample Equations</h1>
```

```
<h2>The Lorenz Equations</h2>

<p>
\begin{align}
\dot{x} &amp; = \sigma(y-x) \label{lorenz}\\
\dot{y} &amp; = \rho x - y - xz \\
\dot{z} &amp; = -\beta z + xy
\end{align}
</p>

<h2>The Cauchy-Schwarz Inequality</h2>

<p>\[
\left( \sum_{k=1}^n a_k b_k \right)^{\!\!2} \leq
 \left( \sum_{k=1}^n a_k^2 \right) \left( \sum_{k=1}^n b_k^2 \right)
\]</p>

<h2>A Cross Product Formula</h2>

<p>\[
  \mathbf{V}_1 \times \mathbf{V}_2 =
   \begin{vmatrix}
    \mathbf{i} &amp; \mathbf{j} &amp; \mathbf{k} \\
    \frac{\partial X}{\partial u} &amp; \frac{\partial Y}{\partial u} &amp; 0 \\
    \frac{\partial X}{\partial v} &amp; \frac{\partial Y}{\partial v} &amp; 0 \\
   \end{vmatrix}
\]</p>

<h2>The probability of getting \(k\) heads when flipping \(n\) coins is:</h2>

<p>\[P(E) = {n \choose k} p^k (1-p)^{ n-k} \]</p>

<h2>An Identity of Ramanujan</h2>

<p>\[
   \frac{1}{(\sqrt{\phi \sqrt{5}}-\phi) e^{\frac25 \pi}} =
     1+\frac{e^{-2\pi}} {1+\frac{e^{-4\pi}} {1+\frac{e^{-6\pi}}
      {1+\frac{e^{-8\pi}} {1+\ldots} } } }
\]</p>

<h2>A Rogers-Ramanujan Identity</h2>

<p>\[
  1 +  \frac{q^2}{(1-q)}+\frac{q^6}{(1-q)(1-q^2)}+\cdots =
    \prod_{j=0}^{\infty}\frac{1}{(1-q^{5j+2})(1-q^{5j+3})},
     \quad\quad \text{for $|q|&lt;1$}.
\]</p>

<h2>Maxwell's Equations</h2>

<p>
\begin{align}
  \nabla \times \vec{\mathbf{B}} -\, \frac1c\, \frac{\partial\vec{\mathbf{E}}}{\partial t} &amp; = \
  \nabla \cdot \vec{\mathbf{E}} &amp; = 4 \pi \rho \\
  \nabla \times \vec{\mathbf{E}}\, +\, \frac1c\, \frac{\partial\vec{\mathbf{B}}}{\partial t} &amp; =
  \nabla \cdot \vec{\mathbf{B}} &amp; = 0
\end{align}
</p>
```

```
<h2>In-line Mathematics</h2>

<p>While display equations look good for a page of samples, the
ability to mix math and text in a paragraph is also important.  This
expression \(\sqrt{3x-1}+(1+x)^2\) is an example of an inline equation.  As
you see, equations can be used this way as well, without unduly
disturbing the spacing between lines.</p>

<h2>References to equations</h2>

<p>Here is a reference to the Lorenz Equations (\ref{lorenz}). Clicking on the equation number will t

</body>
</html>
```

More information

Since the calibre viewer uses the MathJax library to render mathematics, the best place to find out more about math in ebooks and get help is the MathJax website[194].

Creating AZW3 • EPUB • MOBI Catalogs

calibre's Create catalog feature enables you to create a catalog of your library in a variety of formats. This help file describes cataloging options when generating a catalog in AZW3, EPUB and MOBI formats.

- Selecting books to catalog (page 436)
- Included sections (page 437)
- Prefixes (page 438)
- Excluded books (page 438)
- Excluded genres (page 439)
- Other options (page 439)
- Custom catalog covers (page 440)
- Additional help resources (page 440)

Selecting books to catalog

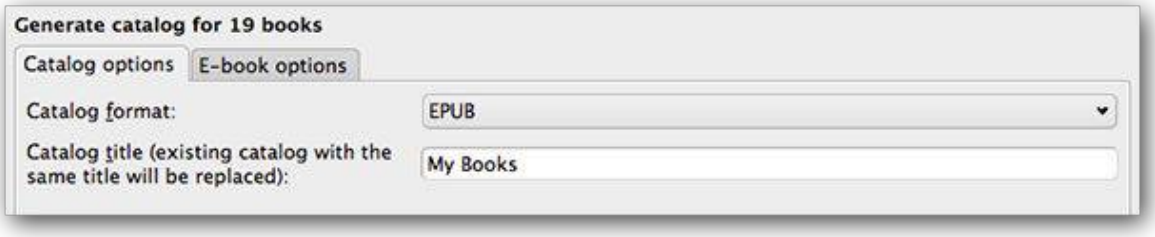
If you want *all* of your library cataloged, remove any search or filtering criteria in the main window. With a single book selected, all books in your library will be candidates for inclusion in the generated catalog. Individual books may be excluded by various criteria; see the *Excluded genres* (page 439) section below for more information.

If you want only *some* of your library cataloged, you have two options:

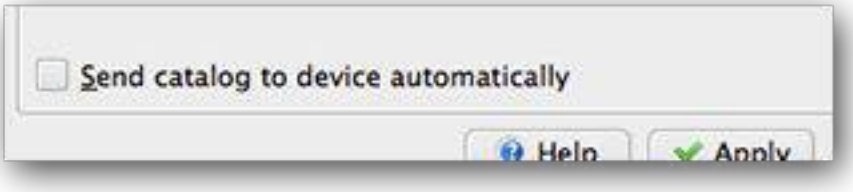
- Create a multiple selection of the books you want cataloged. With more than one book selected in calibre's main window, only the selected books will be cataloged.
- Use the Search field or the Tag Browser to filter the displayed books. Only the displayed books will be cataloged.

To begin catalog generation, select the menu item *Convert books > Create a catalog of the books in your calibre library*. You may also add a *Create Catalog* button to a toolbar in *Preferences > Interface > Toolbars* for easier access to the Generate catalog dialog.

[194] http://www.mathjax.org

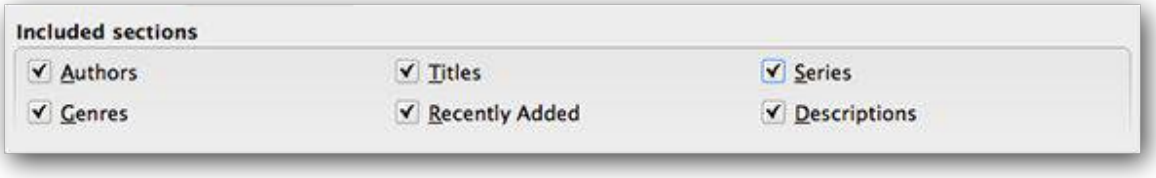


In *Catalog options*, select **AZW3, EPUB or MOBI** as the Catalog format. In the *Catalog title* field, provide a name that will be used for the generated catalog. If a catalog of the same name and format already exists, it will be replaced with the newly-generated catalog.



Enabling *Send catalog to device automatically* will download the generated catalog to a connected device upon completion.

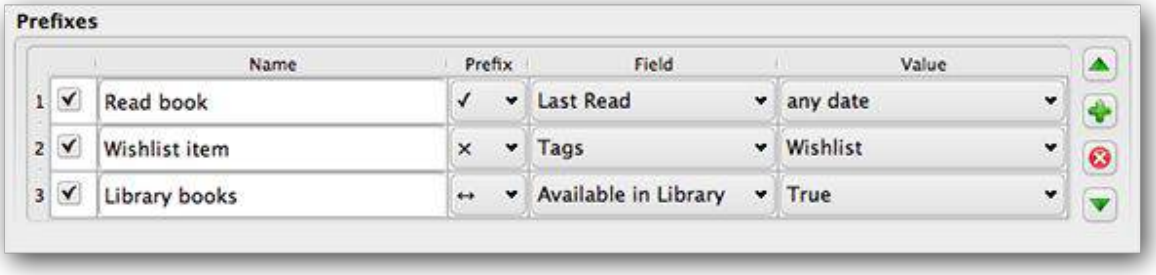
Included sections



Sections enabled by a checkmark will be included in the generated catalog:

- *Authors* - all books, sorted by author, presented in a list format. Non-series books are listed before series books.
- *Titles* - all books, sorted by title, presented in a list format.
- *Series* - all books that are part of a series, sorted by series, presented in a list format.
- *Genres* - individual genres presented in a list, sorted by Author and Series.
- *Recently* Added - all books, sorted in reverse chronological order. List includes books added in the last 30 days, then a month-by-month listing of added books.
- *Descriptions* - detailed description page for each book, including a cover thumbnail and comments. Sorted by author, with non-series books listed before series books.

Prefixes



Prefix rules allow you to add a prefix to book listings when certain criteria are met. For example, you might want to mark books you've read with a checkmark, or books on your wishlist with an X.

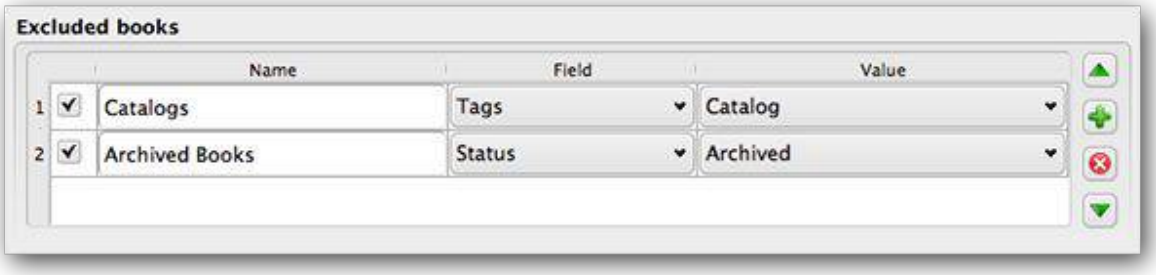
The checkbox in the first column enables the rule. *Name* is a rule name that you provide. *Field* is either *Tags* or a custom column from your library. *Value* is the content of *Field* to match. When a prefix rule is satisfied, the book will be marked with the selected *Prefix*.

Three prefix rules have been specified in the example above:

1. *Read book* specifies that a book with any date in a custom column named *Last read* will be prefixed with a checkmark symbol.
2. *Wishlist* item specifies that any book with a *Wishlist* tag will be prefixed with an X symbol.
3. *Library* books specifies that any book with a value of True (or Yes) in a custom column *Available in Library* will be prefixed with a double arrow symbol.

The first matching rule supplies the prefix. Disabled or incomplete rules are ignored.

Excluded books



Exclusion rules allow you to specify books that will not be cataloged.

The checkbox in the first column enables the rule. *Name* is a rule name that you provide. *Field* is either *Tags* or a custom column in your library. *Value* is the content of *Field* to match. When an exclusion rule is satisfied, the book will be excluded from the generated catalog.

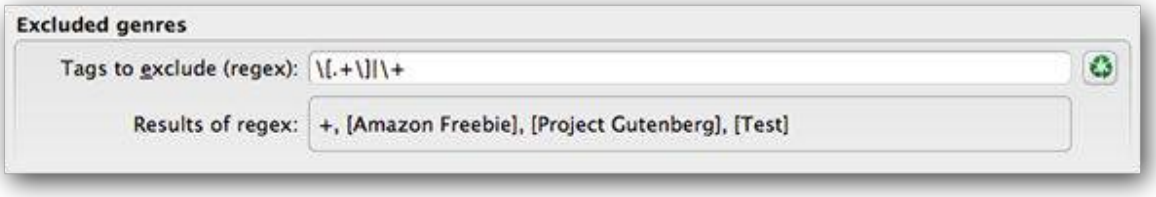
Two exclusion rules have been specified in the example above:

1. The *Catalogs* rule specifies that any book with a *Catalog* tag will be excluded from the generated catalog.

2. The *Archived* Books rule specifies that any book with a value of *Archived* in the custom column *Status* will be excluded from the generated catalog.

All rules are evaluated for every book. Disabled or incomplete rules are ignored.

Excluded genres



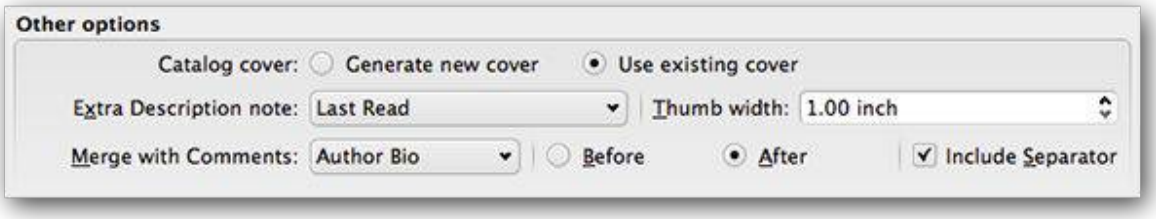
When the catalog is generated, tags in your database are used as genres. For example, you may use the tags `Fiction` and `Nonfiction`. These tags become genres in the generated catalog, with books listed under their respective genre lists based on their assigned tags. A book will be listed in every genre section for which it has a corresponding tag.

You may be using certain tags for other purposes, perhaps a + to indicate a read book, or a bracketed tag like `[Amazon Freebie]` to indicate a book's source. The *Excluded genres* regex allows you to specify tags that you don't want used as genres in the generated catalog. The default exclusion regex pattern `\[.+\]\+` excludes any tags of the form `[tag]`, as well as excluding +, the default tag for read books, from being used as genres in the generated catalog.

You can also use an exact tag name in a regex. For example, `[Amazon Freebie]` or `[Project Gutenberg]`. If you want to list multiple exact tags for exclusion, put a pipe (vertical bar) character between them: `[Amazon Freebie]|[Project Gutenberg]`.

Results of regex shows you which tags will be excluded when the catalog is built, based on the tags in your database and the regex pattern you enter. The results are updated as you modify the regex pattern.

Other options



Catalog cover specifies whether to generate a new cover or use an existing cover. It is possible to create a custom cover for your catalogs - see *Custom catalog covers* (page 440) for more information. If you have created a custom cover that you want to reuse, select *Use existing cover*. Otherwise, select *Generate new cover*.

Extra Description note specifies a custom column's contents to be inserted into the Description page, next to the cover thumbnail. For example, you might want to display the date you last read a book using a *Last Read* custom column. For advanced use of the Description note feature, see this post in the calibre forum[195].

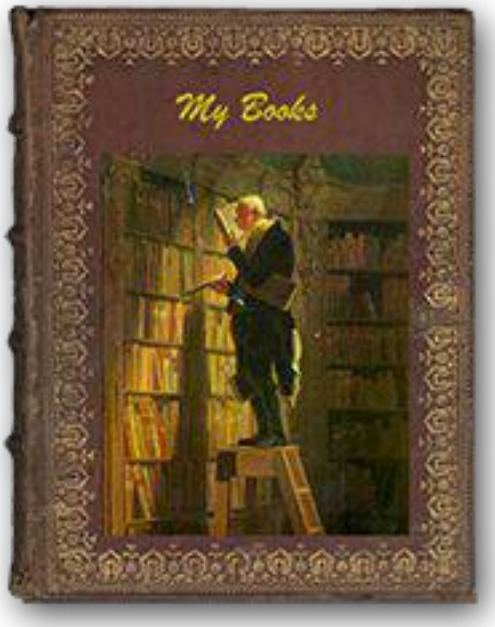
Thumb width specifies a width preference for cover thumbnails included with Descriptions pages. Thumbnails are cached to improve performance.To experiment with different widths, try generating a catalog with just a few books until you've determined your preferred width, then generate your full catalog. The first time a catalog is generated

[195] http://www.mobileread.com/forums/showpost.php?p=1335767&postcount=395

with a new thumbnail width, performance will be slower, but subsequent builds of the catalog will take advantage of the thumbnail cache.

Merge with Comments specifies a custom column whose content will be non-destructively merged with the Comments metadata during catalog generation. For example, you might have a custom column *Author Bio* that you'd like to append to the Comments metadata. You can choose to insert the custom column contents *before or after* the Comments section, and optionally separate the appended content with a horizontal rule separator. Eligible custom column types include `text, comments, and composite`.

Custom catalog covers



With the Generate Cover plugin[196] installed, you can create custom covers for your catalog. To install the plugin, go to *Preferences* > *Advanced* > *Plugins* > *Get new plugins*.

Additional help resources

For more information on calibre's Catalog feature, see the MobileRead forum sticky Creating Catalogs - Start here[197], where you can find information on how to customize the catalog templates, and how to submit a bug report.

To ask questions or discuss calibre's Catalog feature with other users, visit the MobileRead forum Calibre Catalogs[198].

1.19 Customizing calibre

1.19.1 Customizing calibre

calibre has a highly modular design. Various parts of it can be customized. You can learn how to create *recipes* to add new sources of online content to calibre in the Section *Adding your favorite news website* (page 341). Here, you will learn, first, how to use environment variables and *tweaks* to customize calibre's behavior, and then how to specify your

[196] http://www.mobileread.com/forums/showthread.php?t=124219
[197] http://www.mobileread.com/forums/showthread.php?t=118556
[198] http://www.mobileread.com/forums/forumdisplay.php?f=238

own static resources like icons and templates to override the defaults and finally how to use *plugins* to add functionality to calibre.

- Environment variables (page 462)
- Tweaks (page 462)
- Overriding icons, templates, et cetera (page 472)
- Customizing calibre with plugins (page 472)

API Documentation for plugins

Defines various abstract base classes that can be subclassed to create powerful plugins. The useful classes are:

- Plugin (page 441)
- FileTypePlugin (page 443)
- Metadata plugins (page 444)
- Catalog plugins (page 444)
- Metadata download plugins (page 445)
- Conversion plugins (page 447)
- Device Drivers (page 449)
- User Interface Actions (page 458)
- Preferences Plugins (page 460)

Plugin

class `calibre.customize.`**`Plugin`**(*plugin_path*)

A calibre plugin. Useful members include:

- **`self.plugin_path`: Stores path to the zip file that contains** this plugin or None if it is a builtin plugin
- **`self.site_customization`: Stores a customization string entered** by the user.

Methods that should be overridden in sub classes:

- `initialize()` (page 442)
- `customization_help()` (page 442)

Useful methods:

- `temporary_file()` (page 443)

`supported_platforms` = []

List of platforms this plugin works on For example: `['windows', 'osx', 'linux']`

`name` = 'Trivial Plugin'

The name of this plugin. You must set it something other than Trivial Plugin for it to work.

`version` = (1, 0, 0)

The version of this plugin as a 3-tuple (major, minor, revision)

`description` = u'Does absolutely nothing'

A short string describing what this plugin does

author = u'Unknown'
The author of this plugin

priority = 1
When more than one plugin exists for a filetype, the plugins are run in order of decreasing priority i.e. plugins with higher priority will be run first. The highest possible priority is `sys.maxint`. Default priority is 1.

minimum_calibre_version = (0, 4, 118)
The earliest version of calibre this plugin requires

can_be_disabled = True
If False, the user will not be able to disable this plugin. Use with care.

type = u'Base'
The type of this plugin. Used for categorizing plugins in the GUI

initialize()
Called once when calibre plugins are initialized. Plugins are re-initialized every time a new plugin is added.

Perform any plugin specific initialization here, such as extracting resources from the plugin zip file. The path to the zip file is available as `self.plugin_path`.

Note that `self.site_customization` is **not** available at this point.

config_widget()
Implement this method and `save_settings()` (page 442) in your plugin to use a custom configuration dialog, rather then relying on the simple string based default customization.

This method, if implemented, must return a QWidget. The widget can have an optional method validate() that takes no arguments and is called immediately after the user clicks OK. Changes are applied if and only if the method returns True.

If for some reason you cannot perform the configuration at this time, return a tuple of two strings (message, details), these will be displayed as a warning dialog to the user and the process will be aborted.

save_settings(*config_widget*)
Save the settings specified by the user with config_widget.

Parameters **config_widget** – The widget returned by `config_widget()` (page 442).

do_user_config(*parent=None*)
This method shows a configuration dialog for this plugin. It returns True if the user clicks OK, False otherwise. The changes are automatically applied.

load_resources(*names*)
If this plugin comes in a ZIP file (user added plugin), this method will allow you to load resources from the ZIP file.

For example to load an image:

```
pixmap = QPixmap()
pixmap.loadFromData(self.load_resources(['images/icon.png']).itervalues().next())
icon = QIcon(pixmap)
```

Parameters **names** – List of paths to resources in the zip file using / as separator

Returns A dictionary of the form `{name : file_contents}`. Any names that were not found in the zip file will not be present in the dictionary.

customization_help(*gui=False*)
Return a string giving help on how to customize this plugin. By default raise a `NotImplementedError`, which indicates that the plugin does not require customization.

If you re-implement this method in your subclass, the user will be asked to enter a string as customization for this plugin. The customization string will be available as `self.site_customization`.

Site customization could be anything, for example, the path to a needed binary on the user's computer.

Parameters **gui** – If True return HTML help, otherwise return plain text help.

temporary_file(*suffix*)
Return a file-like object that is a temporary file on the file system. This file will remain available even after being closed and will only be removed on interpreter shutdown. Use the `name` member of the returned object to access the full path to the created temporary file.

Parameters **suffix** – The suffix that the temporary file will have.

FileTypePlugin

class calibre.customize.**FileTypePlugin**(*plugin_path*)
Bases: calibre.customize.Plugin (page 441)

A plugin that is associated with a particular set of file types.

file_types = set([])
Set of file types for which this plugin should be run For example: `set(['lit', 'mobi', 'prc'])`

on_import = False
If True, this plugin is run when books are added to the database

on_postimport = False
If True, this plugin is run after books are added to the database

on_preprocess = False
If True, this plugin is run just before a conversion

on_postprocess = False
If True, this plugin is run after conversion on the final file produced by the conversion output plugin.

run(*path_to_ebook*)
Run the plugin. Must be implemented in subclasses. It should perform whatever modifications are required on the ebook and return the absolute path to the modified ebook. If no modifications are needed, it should return the path to the original ebook. If an error is encountered it should raise an Exception. The default implementation simply return the path to the original ebook.

The modified ebook file should be created with the `temporary_file()` method.

Parameters **path_to_ebook** – Absolute path to the ebook.

Returns Absolute path to the modified ebook.

postimport(*book_id*, *book_format*, *db*)
Called post import, i.e., after the book file has been added to the database.

Parameters

- **book_id** – Database id of the added book.
- **book_format** – The file type of the book that was added. :param db: Library database.

Metadata plugins

class calibre.customize.**MetadataReaderPlugin**(**args*, ***kwargs*)
Bases: calibre.customize.Plugin (page 441)

A plugin that implements reading metadata from a set of file types.

file_types = set([])
Set of file types for which this plugin should be run For example: `set(['lit', 'mobi', 'prc'])`

get_metadata(*stream*, *type*)
Return metadata for the file represented by stream (a file like object that supports reading). Raise an exception when there is an error with the input data. :param type: The type of file. Guaranteed to be one of the entries in file_types (page 444). :return: A `calibre.ebooks.metadata.book.Metadata` object

class calibre.customize.**MetadataWriterPlugin**(**args*, ***kwargs*)
Bases: calibre.customize.Plugin (page 441)

A plugin that implements reading metadata from a set of file types.

file_types = set([])
Set of file types for which this plugin should be run For example: `set(['lit', 'mobi', 'prc'])`

set_metadata(*stream*, *mi*, *type*)
Set metadata for the file represented by stream (a file like object that supports reading). Raise an exception when there is an error with the input data. :param type: The type of file. Guaranteed to be one of the entries in file_types (page 444). :param mi: A `calibre.ebooks.metadata.book.Metadata` object

Catalog plugins

class calibre.customize.**CatalogPlugin**(*plugin_path*)
Bases: calibre.customize.Plugin (page 441)

A plugin that implements a catalog generator.

file_types = set([])
Output file type for which this plugin should be run For example: 'epub' or 'xml'

cli_options = []
CLI parser options specific to this plugin, declared as namedtuple Option:

```
from collections import namedtuple
Option = namedtuple('Option', 'option, default, dest, help')
cli_options = [Option('--catalog-title',
                      default = 'My Catalog',
                      dest = 'catalog_title',
                      help = (_('Title of generated catalog. \nDefault:') + " '" +
                      '%default' + "'"))]
cli_options parsed in library.cli:catalog_option_parser()
```

initialize()
If plugin is not a built-in, copy the plugin's .ui and .py files from the zip file to $TMPDIR. Tab will be dynamically generated and added to the Catalog Options dialog in calibre.gui2.dialogs.catalog.py:Catalog

run(*path_to_output*, *opts*, *db*, *ids*, *notification=None*)
Run the plugin. Must be implemented in subclasses. It should generate the catalog in the format specified in file_types, returning the absolute path to the generated catalog file. If an error is encountered it should raise an Exception.

The generated catalog file should be created with the `temporary_file()` method.

Parameters

- **path_to_output** – Absolute path to the generated catalog file.
- **opts** – A dictionary of keyword arguments
- **db** – A LibraryDatabase2 object

Metadata download plugins

class `calibre.ebooks.metadata.sources.base.`**Source**(**args*, ***kwargs*)
Bases: `calibre.customize.Plugin` (page 441)

capabilities = frozenset([])
Set of capabilities supported by this plugin. Useful capabilities are: 'identify', 'cover'

touched_fields = frozenset([])
List of metadata fields that can potentially be download by this plugin during the identify phase

has_html_comments = False
Set this to True if your plugin returns HTML formatted comments

supports_gzip_transfer_encoding = False
Setting this to True means that the browser object will add Accept-Encoding: gzip to all requests. This can speedup downloads but make sure that the source actually supports gzip transfer encoding correctly first

cached_cover_url_is_reliable = True
Cached cover URLs can sometimes be unreliable (i.e. the download could fail or the returned image could be bogus. If that is often the case with this source set to False

options = ()
A list of `Option` objects. They will be used to automatically construct the configuration widget for this plugin

config_help_message = None
A string that is displayed at the top of the config widget for this plugin

is_configured()
Return False if your plugin needs to be configured before it can be used. For example, it might need a username/password/API key.

get_author_tokens(*authors*, *only_first_author=True*)
Take a list of authors and return a list of tokens useful for an AND search query. This function tries to return tokens in first name middle names last name order, by assuming that if a comma is in the author name, the name is in lastname, other names form.

get_title_tokens(*title*, *strip_joiners=True*, *strip_subtitle=False*)
Take a title and return a list of tokens useful for an AND search query. Excludes connectives(optionally) and punctuation.

split_jobs(*jobs*, *num*)
Split a list of jobs into at most num groups, as evenly as possible

test_fields(*mi*)
Return the first field from self.touched_fields that is null on the mi object

clean_downloaded_metadata(*mi*)
Call this method in your plugin's identify method to normalize metadata before putting the Metadata object into result_queue. You can of course, use a custom algorithm suited to your metadata source.

get_book_url(*identifiers*)
Return a 3-tuple or None. The 3-tuple is of the form: (identifier_type, identifier_value, URL). The URL is the URL for the book identified by identifiers at this source. identifier_type, identifier_value specify the identifier corresponding to the URL. This URL must be browseable to by a human using a browser. It is meant to provide a clickable link for the user to easily visit the books page at this source. If no URL is found, return None. This method must be quick, and consistent, so only implement it if it is possible to construct the URL from a known scheme given identifiers.

get_book_url_name(*idtype*, *idval*, *url*)
Return a human readable name from the return value of get_book_url().

get_cached_cover_url(*identifiers*)
Return cached cover URL for the book identified by the identifiers dict or None if no such URL exists.

Note that this method must only return validated URLs, i.e. not URLS that could result in a generic cover image or a not found error.

identify_results_keygen(*title=None*, *authors=None*, *identifiers={}*)
Return a function that is used to generate a key that can sort Metadata objects by their relevance given a search query (title, authors, identifiers).

These keys are used to sort the results of a call to `identify()` (page 446).

For details on the default algorithm see `InternalMetadataCompareKeyGen` (page 447). Re-implement this function in your plugin if the default algorithm is not suitable.

identify(*log*, *result_queue*, *abort*, *title=None*, *authors=None*, *identifiers={}*, *timeout=30*)
Identify a book by its title/author/isbn/etc.

If identifiers(s) are specified and no match is found and this metadata source does not store all related identifiers (for example, all ISBNs of a book), this method should retry with just the title and author (assuming they were specified).

If this metadata source also provides covers, the URL to the cover should be cached so that a subsequent call to the get covers API with the same ISBN/special identifier does not need to get the cover URL again. Use the caching API for this.

Every Metadata object put into result_queue by this method must have a *source_relevance* attribute that is an integer indicating the order in which the results were returned by the metadata source for this query. This integer will be used by `compare_identify_results()`. If the order is unimportant, set it to zero for every result.

Make sure that any cover/isbn mapping information is cached before the Metadata object is put into result_queue.

Parameters

- **log** – A log object, use it to output debugging information/errors
- **result_queue** – A result Queue, results should be put into it. Each result is a Metadata object
- **abort** – If abort.is_set() returns True, abort further processing and return as soon as possible
- **title** – The title of the book, can be None
- **authors** – A list of authors of the book, can be None
- **identifiers** – A dictionary of other identifiers, most commonly {'isbn':'1234...'}
- **timeout** – Timeout in seconds, no network request should hang for longer than timeout.

Returns None if no errors occurred, otherwise a unicode representation of the error suitable for showing to the user

download_cover(*log*, *result_queue*, *abort*, *title=None*, *authors=None*, *identifiers={}*, *timeout=30*)
Download a cover and put it into result_queue. The parameters all have the same meaning as for identify() (page 446). Put (self, cover_data) into result_queue.

This method should use cached cover URLs for efficiency whenever possible. When cached data is not present, most plugins simply call identify and use its results.

class calibre.ebooks.metadata.sources.base.**InternalMetadataCompareKeyGen**(*mi*, *source_plugin*, *title*, *authors*, *identifiers*)
Generate a sort key for comparison of the relevance of Metadata objects, given a search query. This is used only to compare results from the same metadata source, not across different sources.

The sort key ensures that an ascending order sort is a sort by order of decreasing relevance.

The algorithm is:

- Prefer results that have the same ISBN as specified in the query
- Prefer results with a cached cover URL
- Prefer results with all available fields filled in
- Prefer results that are an exact title match to the query
- Prefer results with longer comments (greater than 10% longer)
- **Use the relevance of the result as reported by the metadata source's search** engine

Conversion plugins

class calibre.customize.conversion.**InputFormatPlugin**(**args*)
Bases: calibre.customize.Plugin (page 441)

InputFormatPlugins are responsible for converting a document into HTML+OPF+CSS+etc. The results of the conversion *must* be encoded in UTF-8. The main action happens in convert() (page 448).

file_types = set([])
Set of file types for which this plugin should be run For example: `set(['azw', 'mobi', 'prc'])`

is_image_collection = False
If True, this input plugin generates a collection of images, one per HTML file. You can obtain access to the images via convenience method, `get_image_collection()`.

core_usage = 1
Number of CPU cores used by this plugin A value of -1 means that it uses all available cores

for_viewer = False
If set to True, the input plugin will perform special processing to make its output suitable for viewing

output_encoding = 'utf-8'
The encoding that this input plugin creates files in. A value of None means that the encoding is undefined and must be detected individually

common_options = set([<calibre.customize.conversion.OptionRecommendation object at 0x305ba90>])
Options shared by all Input format plugins. Do not override in sub-classes. Use options (page 448) instead. Every option must be an instance of OptionRecommendation.

options = set([])
Options to customize the behavior of this plugin. Every option must be an instance of OptionRecommendation.

recommendations = set([])
A set of 3-tuples of the form (option_name, recommended_value, recommendation_level)

get_images()
Return a list of absolute paths to the images, if this input plugin represents an image collection. The list of images is in the same order as the spine and the TOC.

convert(*stream*, *options*, *file_ext*, *log*, *accelerators*)
This method must be implemented in sub-classes. It must return the path to the created OPF file or an OEBBook instance. All output should be contained in the current directory. If this plugin creates files outside the current directory they must be deleted/marked for deletion before this method returns.

Parameters

- **stream** – A file like object that contains the input file.
- **options** – Options to customize the conversion process. Guaranteed to have attributes corresponding to all the options declared by this plugin. In addition, it will have a verbose attribute that takes integral values from zero upwards. Higher numbers mean be more verbose. Another useful attribute is input_profile that is an instance of calibre.customize.profiles.InputProfile.
- **file_ext** – The extension (without the .) of the input file. It is guaranteed to be one of the *file_types* supported by this plugin.
- **log** – A calibre.utils.logging.Log object. All output should use this object.
- **accelarators** – A dictionary of various information that the input plugin can get easily that would speed up the subsequent stages of the conversion.

postprocess_book(*oeb*, *opts*, *log*)
Called to allow the input plugin to perform postprocessing after the book has been parsed.

specialize(*oeb*, *opts*, *log*, *output_fmt*)
Called to allow the input plugin to specialize the parsed book for a particular output format. Called after postprocess_book and before any transforms are performed on the parsed book.

gui_configuration_widget(*parent*, *get_option_by_name*, *get_option_help*, *db*, *book_id=None*)
Called to create the widget used for configuring this plugin in the calibre GUI. The widget must be an instance of the PluginWidget class. See the builting input plugins for examples.

class calibre.customize.conversion.**OutputFormatPlugin**(**args*)
Bases: calibre.customize.Plugin (page 441)

OutputFormatPlugins are responsible for converting an OEB document (OPF+HTML) into an output ebook.

The OEB document can be assumed to be encoded in UTF-8. The main action happens in convert() (page 449).

file_type = None
The file type (extension without leading period) that this plugin outputs

common_options = set([<calibre.customize.conversion.OptionRecommendation object at 0x305bbd0>])
Options shared by all Input format plugins. Do not override in sub-classes. Use options (page 448) instead. Every option must be an instance of OptionRecommendation.

options = set([])
Options to customize the behavior of this plugin. Every option must be an instance of `OptionRecommendation`.

recommendations = set([])
A set of 3-tuples of the form (option_name, recommended_value, recommendation_level)

convert(*oeb_book*, *output*, *input_plugin*, *opts*, *log*)
Render the contents of *oeb_book* (which is an instance of `calibre.ebooks.oeb.OEBBook` to the file specified by output.

Parameters

- **output** – Either a file like object or a string. If it is a string it is the path to a directory that may or may not exist. The output plugin should write its output into that directory. If it is a file like object, the output plugin should write its output into the file.
- **input_plugin** – The input plugin that was used at the beginning of the conversion pipeline.
- **opts** – Conversion options. Guaranteed to have attributes corresponding to the OptionRecommendations of this plugin.
- **log** – The logger. Print debug/info messages etc. using this.

specialize_css_for_output(*log*, *opts*, *item*, *stylizer*)
Can be used to make changes to the css during the CSS flattening process.

Parameters

- **item** – The item (HTML file) being processed
- **stylizer** – A Stylizer object containing the flattened styles for item. You can get the style for any element by stylizer.style(element).

gui_configuration_widget(*parent*, *get_option_by_name*, *get_option_help*, *db*, *book_id=None*)
Called to create the widget used for configuring this plugin in the calibre GUI. The widget must be an instance of the PluginWidget class. See the builtin output plugins for examples.

Device Drivers

The base class for all device drivers is `DevicePlugin` (page 449). However, if your device exposes itself as a USBMS drive to the operating system, you should use the USBMS class instead as it implements all the logic needed to support these kinds of devices.

class `calibre.devices.interface.`**DevicePlugin**(*plugin_path*)
Bases: `calibre.customize.Plugin` (page 441)

Defines the interface that should be implemented by backends that communicate with an ebook reader.

FORMATS = ['lrf', 'rtf', 'pdf', 'txt']
Ordered list of supported formats

VENDOR_ID = 0
VENDOR_ID can be either an integer, a list of integers or a dictionary If it is a dictionary, it must be a dictionary of dictionaries, of the form:

```
{
 integer_vendor_id : { product_id : [list of BCDs], ... },
 ...
}
```

PRODUCT_ID = 0
An integer or a list of integers

BCD = None
BCD can be either None to not distinguish between devices based on BCD, or it can be a list of the BCD numbers of all devices supported by this driver.

THUMBNAIL_HEIGHT = 68
Height for thumbnails on the device

WANTS_UPDATED_THUMBNAILS = False
Width for thumbnails on the device. Setting this will force thumbnails to this size, not preserving aspect ratio. If it is not set, then the aspect ratio will be preserved and the thumbnail will be no higher than THUMBNAIL_HEIGHT Set this to True if the device supports updating cover thumbnails during sync_booklists. Setting it to true will ask device.py to refresh the cover thumbnails during book matching

CAN_SET_METADATA = ['title', 'authors', 'collections']
Whether the metadata on books can be set via the GUI.

CAN_DO_DEVICE_DB_PLUGBOARD = False
Whether the device can handle device_db metadata plugboards

path_sep = '/'
Path separator for paths to books on device

icon = '/home/kovid/work/calibre/resources/images/reader.png'
Icon for this device

UserAnnotation
alias of `Annotation`

OPEN_FEEDBACK_MESSAGE = None
GUI displays this as a message if not None. Useful if opening can take a long time

VIRTUAL_BOOK_EXTENSIONS = frozenset([])
Set of extensions that are "virtual books" on the device and therefore cannot be viewed/saved/added to library For example: `frozenset(['kobo'])`

NUKE_COMMENTS = None
Whether to nuke comments in the copy of the book sent to the device. If not None this should be short string that the comments will be replaced by.

MANAGES_DEVICE_PRESENCE = False
If True indicates that this driver completely manages device detection, ejecting and so forth. If you set this to True, you *must* implement the detect_managed_devices and debug_managed_device_detection methods. A driver with this set to true is responsible for detection of devices, managing a blacklist of devices, a list of ejected devices and so forth. calibre will periodically call the detect_managed_devices() method and is it returns a detected device, calibre will call open(). open() will be called every time a device is returned even is previous calls to open() failed, therefore the driver must maintain its own blacklist of failed devices. Similarly, when ejecting, calibre will call eject() and then assuming the next call to detect_managed_devices() returns None, it will call post_yank_cleanup().

SLOW_DRIVEINFO = False
If set the True, calibre will call the `get_driveinfo()` (page 452) method after the books lists have been loaded to get the driveinfo.

ASK_TO_ALLOW_CONNECT = False
If set to True, calibre will ask the user if they want to manage the device with calibre, the first time it is detected. If you set this to True you must implement `get_device_uid()` (page 454) and `ignore_connected_device()` (page 454) and `get_user_blacklisted_devices()` (page 455) and `set_user_blacklisted_devices()` (page 455)

is_usb_connected(*devices_on_system*, *debug=False*, *only_presence=False*)
Return True, device_info if a device handled by this plugin is currently connected.

Parameters **devices_on_system** – List of devices currently connected

detect_managed_devices(*devices_on_system*, *force_refresh=False*)
Called only if MANAGES_DEVICE_PRESENCE is True.

Scan for devices that this driver can handle. Should return a device object if a device is found. This object will be passed to the open() method as the connected_device. If no device is found, return None. The returned object can be anything, calibre does not use it, it is only passed to open().

This method is called periodically by the GUI, so make sure it is not too resource intensive. Use a cache to avoid repeatedly scanning the system.

Parameters

- **devices_on_system** – Set of USB devices found on the system.
- **force_refresh** – If True and the driver uses a cache to prevent repeated scanning, the cache must be flushed.

debug_managed_device_detection(*devices_on_system*, *output*)
Called only if MANAGES_DEVICE_PRESENCE is True.

Should write information about the devices detected on the system to output, which is a file like object.

Should return True if a device was detected and successfully opened, otherwise False.

reset(*key='-1'*, *log_packets=False*, *report_progress=None*, *detected_device=None*)

Parameters

- **key** – The key to unlock the device
- **log_packets** – If true the packet stream to/from the device is logged
- **report_progress** – Function that is called with a % progress (number between 0 and 100) for various tasks If it is called with -1 that means that the task does not have any progress information
- **detected_device** – Device information from the device scanner

can_handle_windows(*device_id*, *debug=False*)
Optional method to perform further checks on a device to see if this driver is capable of handling it. If it is not it should return False. This method is only called after the vendor, product ids and the bcd have matched, so it can do some relatively time intensive checks. The default implementation returns True. This method is called only on windows. See also `can_handle()` (page 451).

Parameters **device_info** – On windows a device ID string. On Unix a tuple of `(vendor_id, product_id, bcd)`.

can_handle(*device_info*, *debug=False*)
Unix version of `can_handle_windows()` (page 451)

Parameters **device_info** – Is a tuple of (vid, pid, bcd, manufacturer, product, serial number)

open(*connected_device*, *library_uuid*)
Perform any device specific initialization. Called after the device is detected but before any other functions that communicate with the device. For example: For devices that present themselves as USB Mass storage devices, this method would be responsible for mounting the device or if the device has been automounted, for finding out where it has been mounted. The method `calibre.devices.usbms.device.Device.open()` has an implementation of this function that should serve as a good example for USB Mass storage devices.

This method can raise an OpenFeedback exception to display a message to the user.

Parameters

- **connected_device** – The device that we are trying to open. It is a tuple of (vendor id, product id, bcd, manufacturer name, product name, device serial number). However, some devices have no serial number and on windows only the first three fields are present, the rest are None.
- **library_uuid** – The UUID of the current calibre library. Can be None if there is no library (for example when used from the command line).

eject()
Un-mount / eject the device from the OS. This does not check if there are pending GUI jobs that need to communicate with the device.

NOTE: That this method may not be called on the same thread as the rest of the device methods.

post_yank_cleanup()
Called if the user yanks the device without ejecting it first.

set_progress_reporter(*report_progress*)
Set a function to report progress information.

Parameters **report_progress** – Function that is called with a % progress (number between 0 and 100) for various tasks If it is called with -1 that means that the task does not have any progress information

get_device_information(*end_session=True*)
Ask device for device information. See L{DeviceInfoQuery}.

Returns (device name, device version, software version on device, mime type) The tuple can optionally have a fifth element, which is a drive information dictionary. See usbms.driver for an example.

get_driveinfo()
Return the driveinfo dictionary. Usually called from get_device_information(), but if loading the driveinfo is slow for this driver, then it should set SLOW_DRIVEINFO. In this case, this method will be called by calibre after the book lists have been loaded. Note that it is not called on the device thread, so the driver should cache the drive info in the books() method and this function should return the cached data.

card_prefix(*end_session=True*)
Return a 2 element list of the prefix to paths on the cards. If no card is present None is set for the card's prefix. E.G. ('/place', '/place2') (None, 'place2') ('place', None) (None, None)

total_space(*end_session=True*)

Get total space available on the mountpoints:

1. Main memory
2. Memory Card A
3. Memory Card B

Returns A 3 element list with total space in bytes of (1, 2, 3). If a particular device doesn't have any of these locations it should return 0.

free_space(*end_session=True*)

Get free space available on the mountpoints:

1. Main memory

2. Card A
3. Card B

Returns A 3 element list with free space in bytes of (1, 2, 3). If a particular device doesn't have any of these locations it should return -1.

books (*oncard=None*, *end_session=True*)
Return a list of ebooks on the device.

Parameters **oncard** – If 'carda' or 'cardb' return a list of ebooks on the specific storage card, otherwise return list of ebooks in main memory of device. If a card is specified and no books are on the card return empty list.

Returns A BookList.

upload_books (*files*, *names*, *on_card=None*, *end_session=True*, *metadata=None*)
Upload a list of books to the device. If a file already exists on the device, it should be replaced. This method should raise a `FreeSpaceError` if there is not enough free space on the device. The text of the FreeSpaceError must contain the word "card" if `on_card` is not None otherwise it must contain the word "memory".

Parameters

- **files** – A list of paths
- **names** – A list of file names that the books should have once uploaded to the device. len(names) == len(files)
- **metadata** – If not None, it is a list of `Metadata` objects. The idea is to use the metadata to determine where on the device to put the book. len(metadata) == len(files). Apart from the regular cover (path to cover), there may also be a thumbnail attribute, which should be used in preference. The thumbnail attribute is of the form (width, height, cover_data as jpeg).

Returns A list of 3-element tuples. The list is meant to be passed to `add_books_to_metadata()` (page 453).

classmethod **add_books_to_metadata** (*locations*, *metadata*, *booklists*)
Add locations to the booklists. This function must not communicate with the device.

Parameters

- **locations** – Result of a call to L{upload_books}
- **metadata** – List of `Metadata` objects, same as for `upload_books()` (page 453).
- **booklists** – A tuple containing the result of calls to (`books(oncard=None)()`, `books(oncard='carda')()`, :meth'books(oncard='cardb')').

delete_books (*paths*, *end_session=True*)
Delete books at paths on device.

classmethod **remove_books_from_metadata** (*paths*, *booklists*)
Remove books from the metadata list. This function must not communicate with the device.

Parameters

- **paths** – paths to books on the device.
- **booklists** – A tuple containing the result of calls to (`books(oncard=None)()`, `books(oncard='carda')()`, :meth'books(oncard='cardb')').

sync_booklists(*booklists*, *end_session=True*)
Update metadata on device.

Parameters **booklists** – A tuple containing the result of calls to (`books(oncard=None)()`, `books(oncard='carda')()`, :meth'books(oncard='cardb')').

get_file(*path*, *outfile*, *end_session=True*)
Read the file at `path` on the device and write it to outfile.

Parameters **outfile** – file object like `sys.stdout` or the result of an `open()` (page 451) call.

classmethod config_widget()
Should return a QWidget. The QWidget contains the settings for the device interface

classmethod save_settings(*settings_widget*)
Should save settings to disk. Takes the widget created in `config_widget()` (page 454) and saves all settings to disk.

classmethod settings()
Should return an opts object. The opts object should have at least one attribute *format_map* which is an ordered list of formats for the device.

set_plugboards(*plugboards*, *pb_func*)
provide the driver the current set of plugboards and a function to select a specific plugboard. This method is called immediately before add_books and sync_booklists.

pb_func is a callable with the following signature:: def pb_func(device_name, format, plugboards)

You give it the current device name (either the class name or DEVICE_PLUGBOARD_NAME), the format you are interested in (a 'real' format or 'device_db'), and the plugboards (you were given those by set_plugboards, the same place you got this method).

Returns None or a single plugboard instance.

set_driveinfo_name(*location_code*, *name*)
Set the device name in the driveinfo file to 'name'. This setting will persist until the file is re-created or the name is changed again.

Non-disk devices should implement this method based on the location codes returned by the get_device_information() method.

prepare_addable_books(*paths*)
Given a list of paths, returns another list of paths. These paths point to addable versions of the books.

If there is an error preparing a book, then instead of a path, the position in the returned list for that book should be a three tuple: (original_path, the exception instance, traceback)

startup()
Called when calibre is is starting the device. Do any initialization required. Note that multiple instances of the class can be instantiated, and thus __init__ can be called multiple times, but only one instance will have this method called. This method is called on the device thread, not the GUI thread.

shutdown()
Called when calibre is shutting down, either for good or in preparation to restart. Do any cleanup required. This method is called on the device thread, not the GUI thread.

get_device_uid()
Must return a unique id for the currently connected device (this is called immediately after a successful call to open()). You must implement this method if you set ASK_TO_ALLOW_CONNECT = True

ignore_connected_device(*uid*)
Should ignore the device identified by uid (the result of a call to get_device_uid()) in the future. You must

implement this method if you set ASK_TO_ALLOW_CONNECT = True. Note that this function is called immediately after open(), so if open() caches some state, the driver should reset that state.

get_user_blacklisted_devices()
Return map of device uid to friendly name for all devices that the user has asked to be ignored.

set_user_blacklisted_devices(*devices*)
Set the list of device uids that should be ignored by this driver.

specialize_global_preferences(*device_prefs*)
Implement this method if your device wants to override a particular preference. You must ensure that all call sites that want a preference that can be overridden use device_prefs['something'] instead of prefs['something']. Your method should call device_prefs.set_overrides(pref=val, pref=val, ...). Currently used for: metadata management (prefs['manage_device_metadata'])

is_dynamically_controllable()
Called by the device manager when starting plugins. If this method returns a string, then a) it supports the device manager's dynamic control interface, and b) that name is to be used when talking to the plugin.

This method can be called on the GUI thread. A driver that implements this method must be thread safe.

start_plugin()
This method is called to start the plugin. The plugin should begin to accept device connections however it does that. If the plugin is already accepting connections, then do nothing.

This method can be called on the GUI thread. A driver that implements this method must be thread safe.

stop_plugin()
This method is called to stop the plugin. The plugin should no longer accept connections, and should cleanup behind itself. It is likely that this method should call shutdown. If the plugin is already not accepting connections, then do nothing.

This method can be called on the GUI thread. A driver that implements this method must be thread safe.

get_option(*opt_string*, *default=None*)
Return the value of the option indicated by opt_string. This method can be called when the plugin is not started. Return None if the option does not exist.

This method can be called on the GUI thread. A driver that implements this method must be thread safe.

set_option(*opt_string*, *opt_value*)
Set the value of the option indicated by opt_string. This method can be called when the plugin is not started.

This method can be called on the GUI thread. A driver that implements this method must be thread safe.

is_running()
Return True if the plugin is started, otherwise false

This method can be called on the GUI thread. A driver that implements this method must be thread safe.

class calibre.devices.interface.**BookList**(*oncard*, *prefix*, *settings*)
Bases: list

A list of books. Each Book object must have the fields

1.title

2.authors

3.size (file size of the book)

4.datetime (a UTC time tuple)

5.path (path on the device to the book)

6. thumbnail (can be None) thumbnail is either a str/bytes object with the image data or it should have an attribute image_path that stores an absolute (platform native) path to the image

7. tags (a list of strings, can be empty).

supports_collections()
Return True if the device supports collections for this book list.

add_book(*book*, *replace_metadata*)
Add the book to the booklist. Intent is to maintain any device-internal metadata. Return True if booklists must be sync'ed

remove_book(*book*)
Remove a book from the booklist. Correct any device metadata at the same time

get_collections(*collection_attributes*)
Return a dictionary of collections created from collection_attributes. Each entry in the dictionary is of the form collection name:[list of books]

The list of books is sorted by book title, except for collections created from series, in which case series_index is used.

Parameters **collection_attributes** – A list of attributes of the Book object

USB Mass Storage based devices The base class for such devices is calibre.devices.usbms.driver.USBMS (page 457). This class in turn inherits some of its functionality from its bases, documented below. A typical basic USBMS based driver looks like this:

```
from calibre.devices.usbms.driver import USBMS

class PDNOVEL(USBMS):
    name = 'Pandigital Novel device interface'
    gui_name = 'PD Novel'
    description = _('Communicate with the Pandigital Novel')
    author = 'Kovid Goyal'
    supported_platforms = ['windows', 'linux', 'osx']
    FORMATS = ['epub', 'pdf']

    VENDOR_ID   = [0x18d1]
    PRODUCT_ID  = [0xb004]
    BCD         = [0x224]

    VENDOR_NAME = 'ANDROID'
    WINDOWS_MAIN_MEM = WINDOWS_CARD_A_MEM = '__UMS_COMPOSITE'
    THUMBNAIL_HEIGHT = 144

    EBOOK_DIR_MAIN = 'eBooks'
    SUPPORTS_SUB_DIRS = False

    def upload_cover(self, path, filename, metadata):
        coverdata = getattr(metadata, 'thumbnail', None)
        if coverdata and coverdata[2]:
            with open('%s.jpg' % os.path.join(path, filename), 'wb') as coverfile:
                coverfile.write(coverdata[2])
```

class calibre.devices.usbms.device.**Device**(*plugin_path*)
Bases: calibre.devices.usbms.deviceconfig.DeviceConfig, calibre.devices.interface.DevicePlugin (page 449)

This class provides logic common to all drivers for devices that export themselves as USB Mass Storage devices. Provides implementations for mounting/ejecting of USBMS devices on all platforms.

WINDOWS_MAIN_MEM = None
String identifying the main memory of the device in the windows PnP id strings This can be None, string, list of strings or compiled regex

WINDOWS_CARD_A_MEM = None
String identifying the first card of the device in the windows PnP id strings This can be None, string, list of strings or compiled regex

WINDOWS_CARD_B_MEM = None
String identifying the second card of the device in the windows PnP id strings This can be None, string, list of strings or compiled regex

OSX_MAIN_MEM_VOL_PAT = None
Used by the new driver detection to disambiguate main memory from storage cards. Should be a regular expression that matches the main memory mount point assigned by OS X

MAX_PATH_LEN = 250
The maximum length of paths created on the device

NEWS_IN_FOLDER = True
Put news in its own folder

windows_sort_drives(*drives*)
Called to disambiguate main memory and storage card for devices that do not distinguish between them on the basis of *WINDOWS_CARD_NAME*. For e.g.: The EB600

filename_callback(*default*, *mi*)
Callback to allow drivers to change the default file name set by `create_upload_path()`.

sanitize_path_components(*components*)
Perform any device specific sanitization on the path components for files to be uploaded to the device

get_annotations(*path_map*)
Resolve path_map to annotation_map of files found on the device

add_annotation_to_library(*db*, *db_id*, *annotation*)
Add an annotation to the calibre library

class `calibre.devices.usbms.cli.`**CLI**

class `calibre.devices.usbms.driver.`**USBMS**(*plugin_path*)
Bases: `calibre.devices.usbms.cli.CLI` (page 457), `calibre.devices.usbms.device.Device` (page 456)

The base class for all USBMS devices. Implements the logic for sending/getting/updating metadata/caching metadata/etc.

upload_cover(*path*, *filename*, *metadata*, *filepath*)
Upload book cover to the device. Default implementation does nothing.

Parameters

- **path** – The full path to the directory where the associated book is located.
- **filename** – The name of the book file without the extension.
- **metadata** – metadata belonging to the book. Use metadata.thumbnail for cover
- **filepath** – The full path to the ebook file

classmethod **normalize_path**(*path*)
Return path with platform native path separators

User Interface Actions

If you are adding your own plugin in a zip file, you should subclass both InterfaceActionBase and InterfaceAction. The `load_actual_plugin()` method of you InterfaceActionBase subclass must return an instantiated object of your InterfaceBase subclass.

class `calibre.gui2.actions.`**InterfaceAction**(*parent*, *site_customization*)
Bases: `PyQt4.QtCore.QObject`

A plugin representing an "action" that can be taken in the graphical user interface. All the items in the toolbar and context menus are implemented by these plugins.

Note that this class is the base class for these plugins, however, to integrate the plugin with calibre's plugin system, you have to make a wrapper class that references the actual plugin. See the `calibre.customize.builtins` module for examples.

If two `InterfaceAction` objects have the same name, the one with higher priority takes precedence.

Sub-classes should implement the `genesis()`, `library_changed()`, `location_selected()` `shutting_down()` and `initialization_complete()` methods.

Once initialized, this plugin has access to the main calibre GUI via the `gui` member. You can access other plugins by name, for example:

```
self.gui.iactions['Save To Disk']
```

To access the actual plugin, use the `interface_action_base_plugin` attribute, this attribute only becomes available after the plugin has been initialized. Useful if you want to use methods from the plugin class like do_user_config().

The QAction specified by `action_spec` is automatically create and made available as `self.qaction`.

name = 'Implement me'
The plugin name. If two plugins with the same name are present, the one with higher priority takes precedence.

priority = 1
The plugin priority. If two plugins with the same name are present, the one with higher priority takes precedence.

popup_type = 1
The menu popup type for when this plugin is added to a toolbar

auto_repeat = False
Whether this action should be auto repeated when its shortcut key is held down.

action_spec = ('text', 'icon', None, None)
Of the form: (text, icon_path, tooltip, keyboard shortcut) icon, tooltip and keyboard shortcut can be None shortcut must be a string, None or tuple of shortcuts. If None, a keyboard shortcut corresponding to the action is not registered. If you pass an empty tuple, then the shortcut is registered with no default key binding.

action_add_menu = False
If True, a menu is automatically created and added to self.qaction

action_menu_clone_qaction = False
If True, a clone of self.qaction is added to the menu of self.qaction If you want the text of this action to be different from that of self.qaction, set this variable to the new text

dont_add_to = frozenset([])
Set of locations to which this action must not be added. See `all_locations` for a list of possible locations

dont_remove_from = frozenset([])
Set of locations from which this action must not be removed. See `all_locations` for a list of possible locations

action_type = 'global'
Type of action 'current' means acts on the current view 'global' means an action that does not act on the current view, but rather on calibre as a whole

accepts_drops = False
If True, then this InterfaceAction will have the opportunity to interact with drag and drop events. See the methods, `accept_enter_event()`, :meth':accept_drag_move_event', `drop_event()` for details.

accept_enter_event(*event*, *mime_data*)
This method should return True iff this interface action is capable of handling the drag event. Do not call accept/ignore on the event, that will be taken care of by the calibre UI.

accept_drag_move_event(*event*, *mime_data*)
This method should return True iff this interface action is capable of handling the drag event. Do not call accept/ignore on the event, that will be taken care of by the calibre UI.

drop_event(*event*, *mime_data*)
This method should perform some useful action and return True iff this interface action is capable of handling the drop event. Do not call accept/ignore on the event, that will be taken care of by the calibre UI. You should not perform blocking/long operations in this function. Instead emit a signal or use QTimer.singleShot and return quickly. See the builtin actions for examples.

create_menu_action(*menu*, *unique_name*, *text*, *icon=None*, *shortcut=None*, *description=None*, *triggered=None*, *shortcut_name=None*)
Convenience method to easily add actions to a QMenu. Returns the created QAction, This action has one extra attribute calibre_shortcut_unique_name which if not None refers to the unique name under which this action is registered with the keyboard manager.

Parameters

- **menu** – The QMenu the newly created action will be added to
- **unique_name** – A unique name for this action, this must be globally unique, so make it as descriptive as possible. If in doubt add a uuid to it.
- **text** – The text of the action.
- **icon** – Either a QIcon or a file name. The file name is passed to the I() builtin, so you do not need to pass the full path to the images directory.
- **shortcut** – A string, a list of strings, None or False. If False, no keyboard shortcut is registered for this action. If None, a keyboard shortcut with no default keybinding is registered. String and list of strings register a shortcut with default keybinding as specified.
- **description** – A description for this action. Used to set tooltips.
- **triggered** – A callable which is connected to the triggered signal of the created action.
- **shortcut_name** – The test displayed to the user when customizing the keyboard shortcuts for this action. By default it is set to the value of `text`.

load_resources(*names*)
If this plugin comes in a ZIP file (user added plugin), this method will allow you to load resources from the ZIP file.

For example to load an image:

```
pixmap = QPixmap()
pixmap.loadFromData(self.load_resources(['images/icon.png']).itervalues().next())
icon = QIcon(pixmap)
```

Parameters **names** – List of paths to resources in the zip file using / as separator

Returns A dictionary of the form `{name : file_contents}`. Any names that were not found in the zip file will not be present in the dictionary.

genesis()
Setup this plugin. Only called once during initialization. self.gui is available. The action specified by `action_spec` is available as `self.qaction`.

location_selected(*loc*)
Called whenever the book list being displayed in calibre changes. Currently values for loc are: `library, main, card and cardb`.

This method should enable/disable this action and its sub actions as appropriate for the location.

library_changed(*db*)
Called whenever the current library is changed.

Parameters **db** – The LibraryDatabase corresponding to the current library.

gui_layout_complete()
Called once per action when the layout of the main GUI is completed. If your action needs to make changes to the layout, they should be done here, rather than in `initialization_complete()`.

initialization_complete()
Called once per action when the initialization of the main GUI is completed.

shutting_down()
Called once per plugin when the main GUI is in the process of shutting down. Release any used resources, but try not to block the shutdown for long periods of time.

Returns False to halt the shutdown. You are responsible for telling the user why the shutdown was halted.

class calibre.customize.**InterfaceActionBase**(**args*, ***kwargs*)
Bases: calibre.customize.Plugin (page 441)

load_actual_plugin(*gui*)
This method must return the actual interface action plugin object.

Preferences Plugins

class calibre.customize.**PreferencesPlugin**(*plugin_path*)
Bases: calibre.customize.Plugin (page 441)

A plugin representing a widget displayed in the Preferences dialog.

This plugin has only one important method `create_widget()`. The various fields of the plugin control how it is categorized in the UI.

config_widget = None
Import path to module that contains a class named ConfigWidget which implements the ConfigWidget-Interface. Used by `create_widget()`.

category_order = 100
Where in the list of categories the `category` of this plugin should be.

name_order = 100
Where in the list of names in a category, the `gui_name` of this plugin should be

category = None
The category this plugin should be in

gui_category = None
The category name displayed to the user for this plugin

gui_name = None
The name displayed to the user for this plugin

icon = None
The icon for this plugin, should be an absolute path

description = None
The description used for tooltips and the like

create_widget(*parent=None*)
Create and return the actual Qt widget used for setting this group of preferences. The widget must implement the `calibre.gui2.preferences.ConfigWidgetInterface` (page 461).

The default implementation uses `config_widget` to instantiate the widget.

class `calibre.gui2.preferences.`**ConfigWidgetInterface**
This class defines the interface that all widgets displayed in the Preferences dialog must implement. See `ConfigWidgetBase` for a base class that implements this interface and defines various convenience methods as well.

changed_signal = None
This signal must be emitted whenever the user changes a value in this widget

supports_restoring_to_defaults = True
Set to True iff the `restore_to_defaults()` method is implemented.

restore_defaults_desc = u'Restore settings to default values. You have to click Apply to actually save the default se
The tooltip for the Restore to defaults button

restart_critical = False
If True the Preferences dialog will not allow the user to set any more preferences. Only has effect if `commit()` returns True.

genesis(*gui*)
Called once before the widget is displayed, should perform any necessary setup.

Parameters **gui** – The main calibre graphical user interface

initialize()
Should set all config values to their initial values (the values stored in the config files).

restore_defaults()
Should set all config values to their defaults.

commit()
Save any changed settings. Return True if the changes require a restart, False otherwise. Raise an `AbortCommit` exception to indicate that an error occurred. You are responsible for giving the user feedback about what the error is and how to correct it.

refresh_gui(*gui*)
Called once after this widget is committed. Responsible for causing the gui to reread any changed settings.

Note that by default the GUI re-initializes various elements anyway, so most widgets won't need to use this method.

class calibre.gui2.preferences.**ConfigWidgetBase**(*parent=None*)

Base class that contains code to easily add standard config widgets like checkboxes, combo boxes, text fields and so on. See the `register()` method.

This class automatically handles change notification, resetting to default, translation between gui objects and config objects, etc. for registered settings.

If your config widget inherits from this class but includes setting that are not registered, you should override the `ConfigWidgetInterface` methods and call the base class methods inside the overrides.

register(*name*, *config_obj*, *gui_name=None*, *choices=None*, *restart_required=False*, *empty_string_is_None=True*, *setting=<class 'calibre.gui2.preferences.Setting'>*)

Register a setting.

Parameters

- **name** – The setting name
- **config** – The config object that reads/writes the setting
- **gui_name** – The name of the GUI object that presents an interface to change the setting. By default it is assumed to be `'opt_' + name`.
- **choices** – If this setting is a multiple choice (combobox) based setting, the list of choices. The list is a list of two element tuples of the form: `[(gui name, value), ...]`
- **setting** – The class responsible for managing this setting. The default class handles almost all cases, so this param is rarely used.

Environment variables

- `CALIBRE_CONFIG_DIRECTORY` - sets the directory where configuration files are stored/read.
- `CALIBRE_TEMP_DIR` - sets the temporary directory used by calibre
- `CALIBRE_OVERRIDE_DATABASE_PATH` - allows you to specify the full path to metadata.db. Using this variable you can have metadata.db be in a location other than the library folder. Useful if your library folder is on a networked drive that does not support file locking.
- `CALIBRE_DEVELOP_FROM` - Used to run from a calibre development environment. See *Setting up a calibre development environment* (page 511).
- `CALIBRE_OVERRIDE_LANG` - Used to force the language used by the interface (ISO 639 language code)
- `CALIBRE_NO_NATIVE_FILEDIALOGS` - Causes calibre to not use native file dialogs for selecting files/directories.
- `SYSFS_PATH` - Use if sysfs is mounted somewhere other than /sys
- `http_proxy` - Used on linux to specify an HTTP proxy

Tweaks

Tweaks are small changes that you can specify to control various aspects of calibre's behavior. You can change them by going to Preferences->Advanced->Tweaks. The default values for the tweaks are reproduced below

```
#!/usr/bin/env python
# vim:fileencoding=UTF-8:ts=4:sw=4:sta:et:sts=4:ai
__license__   = 'GPL v3'
__copyright__ = '2010, Kovid Goyal <kovid@kovidgoyal.net>'
__docformat__ = 'restructuredtext en'

'''
Contains various tweaks that affect calibre behavior. Only edit this file if
you know what you are doing. If you delete this file, it will be recreated from
defaults.
'''

#: Auto increment series index
# The algorithm used to assign a book added to an existing series a series number.
# New series numbers assigned using this tweak are always integer values, except
# if a constant non-integer is specified.
# Possible values are:
# next - First available integer larger than the largest existing number
# first_free - First available integer larger than 0
# next_free - First available integer larger than the smallest existing number
# last_free - First available integer smaller than the largest existing number
#             Return largest existing + 1 if no free number is found
# const - Assign the number 1 always
# no_change - Do not change the series index
# a number - Assign that number always. The number is not in quotes. Note that
#            0.0 can be used here.
# Examples:
# series_index_auto_increment = 'next'
# series_index_auto_increment = 'next_free'
# series_index_auto_increment = 16.5
#
# Set the use_series_auto_increment_tweak_when_importing tweak to True to
# use the above values when importing/adding books. If this tweak is set to
# False (the default) then the series number will be set to 1 if it is not
# explicitly set to during the import. If set to True, then the
# series index will be set according to the series_index_auto_increment setting.
# Note that the use_series_auto_increment_tweak_when_importing tweak is used
# only when a value is not provided during import. If the importing regular
# expression produces a value for series_index, or if you are reading metadata
# from books and the import plugin produces a value, than that value will
# be used irrespective of the setting of the tweak.
series_index_auto_increment = 'next'
use_series_auto_increment_tweak_when_importing = False

#: Add separator after completing an author name
# Should the completion separator be append
# to the end of the completed text to
# automatically begin a new completion operation
# for authors.
# Can be either True or False
authors_completer_append_separator = False

#: Author sort name algorithm
# The algorithm used to copy author to author_sort
# Possible values are:
#  invert: use "fn ln" -> "ln, fn"
#  copy  : copy author to author_sort without modification
#  comma : use 'copy' if there is a ',' in the name, otherwise use 'invert'
```

```
#  nocomma : "fn ln" -> "ln fn" (without the comma)
# When this tweak is changed, the author_sort values stored with each author
# must be recomputed by right-clicking on an author in the left-hand tags pane,
# selecting 'manage authors', and pressing 'Recalculate all author sort values'.
# The author name suffixes are words that are ignored when they occur at the
# end of an author name. The case of the suffix is ignored and trailing
# periods are automatically handled. The same is true for prefixes.
# The author name copy words are a set of words which if they occur in an
# author name cause the automatically generated author sort string to be
# identical to the author name. This means that the sort for a string like Acme
# Inc. will be Acme Inc. instead of Inc., Acme
author_sort_copy_method = 'comma'
author_name_suffixes = ('Jr', 'Sr', 'Inc', 'Ph.D', 'Phd',
                        'MD', 'M.D', 'I', 'II', 'III', 'IV',
                        'Junior', 'Senior')
author_name_prefixes = ('Mr', 'Mrs', 'Ms', 'Dr', 'Prof')
author_name_copywords = ('Corporation', 'Company', 'Co.', 'Agency', 'Council',
        'Committee', 'Inc.', 'Institute', 'Society', 'Club', 'Team')

#: Splitting multiple author names
# By default, calibre splits a string containing multiple author names on
# ampersands and the words "and" and "with". You can customize the splitting
# by changing the regular expression below. Strings are split on whatever the
# specified regular expression matches.
# Default: r'(?i),?\s+(and|with)\s+'
authors_split_regex = r'(?i),?\s+(and|with)\s+'

#: Use author sort in Tag Browser
# Set which author field to display in the tags pane (the list of authors,
# series, publishers etc on the left hand side). The choices are author and
# author_sort. This tweak affects only what is displayed under the authors
# category in the tags pane and content server. Please note that if you set this
# to author_sort, it is very possible to see duplicate names in the list because
# although it is guaranteed that author names are unique, there is no such
# guarantee for author_sort values. Showing duplicates won't break anything, but
# it could lead to some confusion. When using 'author_sort', the tooltip will
# show the author's name.
# Examples:
#   categories_use_field_for_author_name = 'author'
#   categories_use_field_for_author_name = 'author_sort'
categories_use_field_for_author_name = 'author'

#: Control partitioning of Tag Browser
# When partitioning the tags browser, the format of the subcategory label is
# controlled by a template: categories_collapsed_name_template if sorting by
# name, categories_collapsed_rating_template if sorting by average rating, and
# categories_collapsed_popularity_template if sorting by popularity. There are
# two variables available to the template: first and last. The variable 'first'
# is the initial item in the subcategory, and the variable 'last' is the final
# item in the subcategory. Both variables are 'objects'; they each have multiple
# values that are obtained by using a suffix. For example, first.name for an
# author category will be the name of the author. The sub-values available are:
#  name: the printable name of the item
#  count: the number of books that references this item
#  avg_rating: the average rating of all the books referencing this item
#  sort: the sort value. For authors, this is the author_sort for that author
#  category: the category (e.g., authors, series) that the item is in.
# Note that the "r'" in front of the { is necessary if there are backslashes
```

```
# (\ characters) in the template. It doesn't hurt anything to leave it there
# even if there aren't any backslashes.
categories_collapsed_name_template = r'{first.sort:shorten(4,,0)} - {last.sort:shorten(4,,0)}'
categories_collapsed_rating_template = r'{first.avg_rating:4.2f:ifempty(0)} - {last.avg_rating:4.2f:i
categories_collapsed_popularity_template = r'{first.count:d} - {last.count:d}'

#: Control order of categories in the tag browser
# Change the following dict to change the order that categories are displayed in
# the tag browser. Items are named using their lookup name, and will be sorted
# using the number supplied. The lookup name '*' stands for all names that
# otherwise do not appear. Two names with the same value will be sorted
# using the default order; the one used when the dict is empty.
# Example: tag_browser_category_order = {'series':1, 'tags':2, '*':3}
# resulting in the order series, tags, then everything else in default order.
tag_browser_category_order = {'*':1}


#: Specify columns to sort the booklist by on startup
# Provide a set of columns to be sorted on when calibre starts
#  The argument is None if saved sort history is to be used
#  otherwise it is a list of column,order pairs. Column is the
#  lookup/search name, found using the tooltip for the column
#  Order is 0 for ascending, 1 for descending
# For example, set it to [('authors',0),('title',0)] to sort by
# title within authors.
sort_columns_at_startup = None

#: Control how dates are displayed
# Format to be used for publication date and the timestamp (date).
#  A string controlling how the publication date is displayed in the GUI
#  d     the day as number without a leading zero (1 to 31)
#  dd    the day as number with a leading zero (01 to 31)
#  ddd   the abbreviated localized day name (e.g. 'Mon' to 'Sun').
#  dddd  the long localized day name (e.g. 'Monday' to 'Qt::Sunday').
#  M     the month as number without a leading zero (1-12)
#  MM    the month as number with a leading zero (01-12)
#  MMM   the abbreviated localized month name (e.g. 'Jan' to 'Dec').
#  MMMM  the long localized month name (e.g. 'January' to 'December').
#  yy    the year as two digit number (00-99)
#  yyyy  the year as four digit number
#  h     the hours without a leading 0 (0 to 11 or 0 to 23, depending on am/pm) '
#  hh    the hours with a leading 0 (00 to 11 or 00 to 23, depending on am/pm) '
#  m     the minutes without a leading 0 (0 to 59) '
#  mm    the minutes with a leading 0 (00 to 59) '
#  s     the seconds without a leading 0 (0 to 59) '
#  ss    the seconds with a leading 0 (00 to 59) '
#  ap    use a 12-hour clock instead of a 24-hour clock, with "ap"
#        replaced by the localized string for am or pm '
#  AP    use a 12-hour clock instead of a 24-hour clock, with "AP"
#        replaced by the localized string for AM or PM '
#  iso   the date with time and timezone. Must be the only format present
#  For example, given the date of 9 Jan 2010, the following formats show
#  MMM yyyy ==> Jan 2010    yyyy ==> 2010       dd MMM yyyy ==> 09 Jan 2010
#  MM/yyyy ==> 01/2010      d/M/yy ==> 9/1/10   yy ==> 10
# publication default if not set: MMM yyyy
# timestamp default if not set: dd MMM yyyy
# last_modified_display_format if not set: dd MMM yyyy
gui_pubdate_display_format = 'MMM yyyy'
```

```
gui_timestamp_display_format = 'dd MMM yyyy'
gui_last_modified_display_format = 'dd MMM yyyy'

#: Control sorting of titles and series in the library display
# Control title and series sorting in the library view. If set to
# 'library_order', the title sort field will be used instead of the title.
# Unless you have manually edited the title sort field, leading articles such as
# The and A will be ignored. If set to 'strictly_alphabetic', the titles will be
# sorted as-is (sort by title instead of title sort). For example, with
# library_order, The Client will sort under 'C'. With strictly_alphabetic, the
# book will sort under 'T'.
# This flag affects Calibre's library display. It has no effect on devices. In
# addition, titles for books added before changing the flag will retain their
# order until the title is edited. Double-clicking on a title and hitting return
# without changing anything is sufficient to change the sort.
title_series_sorting = 'library_order'

#: Control formatting of title and series when used in templates
# Control how title and series names are formatted when saving to disk/sending
# to device. The behavior depends on the field being processed. If processing
# title, then if this tweak is set to 'library_order', the title will be
# replaced with title_sort. If it is set to 'strictly_alphabetic', then the
# title will not be changed. If processing series, then if set to
# 'library_order', articles such as 'The' and 'An' will be moved to the end. If
# set to 'strictly_alphabetic', the series will be sent without change.
# For example, if the tweak is set to library_order, "The Lord of the Rings"
# will become "Lord of the Rings, The". If the tweak is set to
# strictly_alphabetic, it would remain "The Lord of the Rings". Note that the
# formatter function raw_field will return the base value for title and
# series regardless of the setting of this tweak.
save_template_title_series_sorting = 'library_order'

#: Set the list of words considered to be "articles" for sort strings
# Set the list of words that are to be considered 'articles' when computing the
# title sort strings. The articles differ by language. By default, calibre uses
# a combination of articles from English and whatever language the calibre user
# interface is set to. In addition, in some contexts where the book language is
# available, the language of the book is used. You can change the list of
# articles for a given language or add a new language by editing
# per_language_title_sort_articles. To tell calibre to use a language other
# than the user interface language, set, default_language_for_title_sort. For
# example, to use German, set it to 'deu'. A value of None means the user
# interface language is used. The setting title_sort_articles is ignored
# (present only for legacy reasons).
per_language_title_sort_articles = {
        # English
        'eng'  : (r'A\s+', r'The\s+', r'An\s+'),
        # Spanish
        'spa'  : (r'El\s+', r'La\s+', r'Lo\s+', r'Los\s+', r'Las\s+', r'Un\s+',
                  r'Una\s+', r'Unos\s+', r'Unas\s+'),
        # French
        'fra'  : (r'Le\s+', r'La\s+', r"L'", r'Les\s+', r'Un\s+', r'Une\s+',
                  r'Des\s+', r'De\s+La\s+', r'De\s+', r"D'"),
        # Italian
        'ita'  : (r'Lo\s+', r'Il\s+', r"L'", r'La\s+', r'Gli\s+', r'I\s+',
                  r'Le\s+', ),
        # Portuguese
        'por'  : (r'A\s+', r'O\s+', r'Os\s+', r'As\s+', r'Um\s+', r'Uns\s+',
```

```
                  r'Uma\s+', r'Umas\s+', ),
        # Romanian
        'ron'  : (r'Un\s+', r'O\s+', r'Nişte\s+', ),
        # German
        'deu'  : (r'Der\s+', r'Die\s+', r'Das\s+', r'Den\s+', r'Ein\s+',
                  r'Eine\s+', r'Einen\s+', r'Dem\s+', r'Des\s+', r'Einem\s+',
                  r'Eines\s+'),
        # Dutch
        'nld'  : (r'De\s+', r'Het\s+', r'Een\s+', r"'n\s+", r"'s\s+", r'Ene\s+',
                  r'Ener\s+', r'Enes\s+', r'Den\s+', r'Der\s+', r'Des\s+',
                  r"'t\s+"),
        # Swedish
        'swe'  : (r'En\s+', r'Ett\s+', r'Det\s+', r'Den\s+', r'De\s+', ),
        # Turkish
        'tur'  : (r'Bir\s+', ),
        # Afrikaans
        'afr'  : (r"'n\s+", r'Die\s+', ),
        # Greek
        'ell'  : (r'O\s+', r'I\s+', r'To\s+', r'Ta\s+', r'Tus\s+', r'Tis\s+',
                  r"'Enas\s+", r"'Mia\s+", r"'Ena\s+", r"'Enan\s+", ),
        # Hungarian
        'hun'  : (r'A\s+', 'Az\s+', 'Egy\s+',),
}
default_language_for_title_sort = None
title_sort_articles=r'^(A|The|An)\s+'

#: Specify a folder calibre should connect to at startup
# Specify a folder that calibre should connect to at startup using
# connect_to_folder. This must be a full path to the folder. If the folder does
# not exist when calibre starts, it is ignored. If there are '\' characters in
# the path (such as in Windows paths), you must double them.
# Examples:
#     auto_connect_to_folder = 'C:\\Users\\someone\\Desktop\\testlib'
#     auto_connect_to_folder = '/home/dropbox/My Dropbox/someone/library'
auto_connect_to_folder = ''

#: Specify renaming rules for SONY collections
# Specify renaming rules for sony collections. This tweak is only applicable if
# metadata management is set to automatic. Collections on Sonys are named
# depending upon whether the field is standard or custom. A collection derived
# from a standard field is named for the value in that field. For example, if
# the standard 'series' column contains the value 'Darkover', then the
# collection name is 'Darkover'. A collection derived from a custom field will
# have the name of the field added to the value. For example, if a custom series
# column named 'My Series' contains the name 'Darkover', then the collection
# will by default be named 'Darkover (My Series)'. For purposes of this
# documentation, 'Darkover' is called the value and 'My Series' is called the
# category. If two books have fields that generate the same collection name,
# then both books will be in that collection.
# This set of tweaks lets you specify for a standard or custom field how
# the collections are to be named. You can use it to add a description to a
# standard field, for example 'Foo (Tag)' instead of the 'Foo'. You can also use
# it to force multiple fields to end up in the same collection. For example, you
# could force the values in 'series', '#my_series_1', and '#my_series_2' to
# appear in collections named 'some_value (Series)', thereby merging all of the
# fields into one set of collections.
# There are two related tweaks. The first determines the category name to use
# for a metadata field.  The second is a template, used to determines how the
```

```
# value and category are combined to create the collection name.
# The syntax of the first tweak, sony_collection_renaming_rules, is:
# {'field_lookup_name':'category_name_to_use', 'lookup_name':'name', ...}
# The second tweak, sony_collection_name_template, is a template. It uses the
# same template language as plugboards and save templates. This tweak controls
# how the value and category are combined together to make the collection name.
# The only two fields available are {category} and {value}. The {value} field is
# never empty. The {category} field can be empty. The default is to put the
# value first, then the category enclosed in parentheses, it isn't empty:
# '{value} {category:|(|)}'
# Examples: The first three examples assume that the second tweak
# has not been changed.
# 1: I want three series columns to be merged into one set of collections. The
# column lookup names are 'series', '#series_1' and '#series_2'. I want nothing
# in the parenthesis. The value to use in the tweak value would be:
#    sony_collection_renaming_rules={'series':'', '#series_1':'', '#series_2':''}
# 2: I want the word '(Series)' to appear on collections made from series, and
# the word '(Tag)' to appear on collections made from tags. Use:
#    sony_collection_renaming_rules={'series':'Series', 'tags':'Tag'}
# 3: I want 'series' and '#myseries' to be merged, and for the collection name
# to have '(Series)' appended. The renaming rule is:
#    sony_collection_renaming_rules={'series':'Series', '#myseries':'Series'}
# 4: Same as example 2, but instead of having the category name in parentheses
# and appended to the value, I want it prepended and separated by a colon, such
# as in Series: Darkover. I must change the template used to format the category name
# The resulting two tweaks are:
#    sony_collection_renaming_rules={'series':'Series', 'tags':'Tag'}
#    sony_collection_name_template='{category:||: }{value}'
sony_collection_renaming_rules={}
sony_collection_name_template='{value}{category:| (|)}'

#: Specify how SONY collections are sorted
# Specify how sony collections are sorted. This tweak is only applicable if
# metadata management is set to automatic. You can indicate which metadata is to
# be used to sort on a collection-by-collection basis. The format of the tweak
# is a list of metadata fields from which collections are made, followed by the
# name of the metadata field containing the sort value.
# Example: The following indicates that collections built from pubdate and tags
# are to be sorted by the value in the custom column '#mydate', that collections
# built from 'series' are to be sorted by 'series_index', and that all other
# collections are to be sorted by title. If a collection metadata field is not
# named, then if it is a series- based collection it is sorted by series order,
# otherwise it is sorted by title order.
# [(['pubdate', 'tags'],'#mydate'), (['series'],'series_index'), (['*'], 'title')]
# Note that the bracketing and parentheses are required. The syntax is
# [ ( [list of fields], sort field ) , ( [ list of fields ] , sort field ) ]
# Default: empty (no rules), so no collection attributes are named.
sony_collection_sorting_rules = []

#: Control how tags are applied when copying books to another library
# Set this to True to ensure that tags in 'Tags to add when adding
# a book' are added when copying books to another library
add_new_book_tags_when_importing_books = False

#: Set the maximum number of tags to show per book in the content server
max_content_server_tags_shown=5

#: Set custom metadata fields that the content server will or will not display.
```

```
# content_server_will_display is a list of custom fields to be displayed.
# content_server_wont_display is a list of custom fields not to be displayed.
# wont_display has priority over will_display.
# The special value '*' means all custom fields. The value [] means no entries.
# Defaults:
#    content_server_will_display = ['*']
#    content_server_wont_display = []
# Examples:
# To display only the custom fields #mytags and #genre:
#   content_server_will_display = ['#mytags', '#genre']
#   content_server_wont_display = []
# To display all fields except #mycomments:
#   content_server_will_display = ['*']
#   content_server_wont_display['#mycomments']
content_server_will_display = ['*']
content_server_wont_display = []

#: Set the maximum number of sort 'levels'
# Set the maximum number of sort 'levels' that calibre will use to resort the
# library after certain operations such as searches or device insertion. Each
# sort level adds a performance penalty. If the database is large (thousands of
# books) the penalty might be noticeable. If you are not concerned about multi-
# level sorts, and if you are seeing a slowdown, reduce the value of this tweak.
maximum_resort_levels = 5

#: Choose whether dates are sorted using visible fields
# Date values contain both a date and a time. When sorted, all the fields are
# used, regardless of what is displayed. Set this tweak to True to use only
# the fields that are being displayed.
sort_dates_using_visible_fields = False

#: Specify which font to use when generating a default cover or masthead
# Absolute path to .ttf font files to use as the fonts for the title, author
# and footer when generating a default cover or masthead image. Useful if the
# default font (Liberation Serif) does not contain glyphs for the language of
# the books in your library.
generate_cover_title_font = None
generate_cover_foot_font = None

#: Control behavior of the book list
# You can control the behavior of doubleclicks on the books list.
# Choices: open_viewer, do_nothing,
# edit_cell, edit_metadata. Selecting edit_metadata has the side effect of
# disabling editing a field using a single click.
# Default: open_viewer.
# Example: doubleclick_on_library_view = 'do_nothing'
# You can also control whether the book list scrolls horizontal per column or
# per pixel. Default is per column.
doubleclick_on_library_view = 'open_viewer'
horizontal_scrolling_per_column = True

#: Language to use when sorting.
# Setting this tweak will force sorting to use the
# collating order for the specified language. This might be useful if you run
# calibre in English but want sorting to work in the language where you live.
# Set the tweak to the desired ISO 639-1 language code, in lower case.
# You can find the list of supported locales at
# http://publib.boulder.ibm.com/infocenter/iseries/v5r3/topic/nls/rbagsicusortsequencetables.htm
```

```
# Default: locale_for_sorting = '' -- use the language calibre displays in
# Example: locale_for_sorting = 'fr' -- sort using French rules.
# Example: locale_for_sorting = 'nb' -- sort using Norwegian rules.
locale_for_sorting =  ''

#: Number of columns for custom metadata in the edit metadata dialog
# Set whether to use one or two columns for custom metadata when editing
# metadata  one book at a time. If True, then the fields are laid out using two
# columns. If False, one column is used.
metadata_single_use_2_cols_for_custom_fields = True

#: Order of custom column(s) in edit metadata
# Controls the order that custom columns are listed in edit metadata single
# and bulk. The columns listed in the tweak are displayed first and in the
# order provided. Any columns not listed are dislayed after the listed ones,
# in alphabetical order. Do note that this tweak does not change the size of
# the edit widgets. Putting comments widgets in this list may result in some
# odd widget spacing when using two-column mode.
# Enter a comma-separated list of custom field lookup names, as in
# metadata_edit_custom_column_order = ['#genre', '#mytags', '#etc']
metadata_edit_custom_column_order = []

#: The number of seconds to wait before sending emails
# The number of seconds to wait before sending emails when using a
# public email server like gmail or hotmail. Default is: 5 minutes
# Setting it to lower may cause the server's SPAM controls to kick in,
# making email sending fail. Changes will take effect only after a restart of
# calibre.
public_smtp_relay_delay = 301

#: The maximum width and height for covers saved in the calibre library
# All covers in the calibre library will be resized, preserving aspect ratio,
# to fit within this size. This is to prevent slowdowns caused by extremely
# large covers
maximum_cover_size = (1200, 1600)

#: Where to send downloaded news
# When automatically sending downloaded news to a connected device, calibre
# will by default send it to the main memory. By changing this tweak, you can
# control where it is sent. Valid values are "main", "carda", "cardb". Note
# that if there isn't enough free space available on the location you choose,
# the files will be sent to the location with the most free space.
send_news_to_device_location = "main"

#: What interfaces should the content server listen on
# By default, the calibre content server listens on '0.0.0.0' which means that it
# accepts IPv4 connections on all interfaces. You can change this to, for
# example, '127.0.0.1' to only listen for connections from the local machine, or
# to '::' to listen to all incoming IPv6 and IPv4 connections (this may not
# work on all operating systems)
server_listen_on = '0.0.0.0'

#: Unified toolbar on OS X
# If you enable this option and restart calibre, the toolbar will be 'unified'
# with the titlebar as is normal for OS X applications. However, doing this has
# various bugs, for instance the minimum width of the toolbar becomes twice
# what it should be and it causes other random bugs on some systems, so turn it
# on at your own risk!
```

```
unified_title_toolbar_on_osx = False

#: Save original file when converting/polishing from same format to same format
# When calibre does a conversion from the same format to the same format, for
# example, from EPUB to EPUB, the original file is saved, so that in case the
# conversion is poor, you can tweak the settings and run it again. By setting
# this to False you can prevent calibre from saving the original file.
# Similarly, by setting save_original_format_when_polishing to False you can
# prevent calibre from saving the original file when polishing.
save_original_format = True
save_original_format_when_polishing = True

#: Number of recently viewed books to show
# Right-clicking the View button shows a list of recently viewed books. Control
# how many should be shown, here.
gui_view_history_size = 15

#: Change the font size of book details in the interface
# Change the font size at which book details are rendered in the side panel and
# comments are rendered in the metadata edit dialog. Set it to a positive or
# negative number to increase or decrease the font size.
change_book_details_font_size_by = 0

#: Compile General Program Mode templates to Python
# Compiled general program mode templates are significantly faster than
# interpreted templates. Setting this tweak to True causes calibre to compile
# (in most cases) general program mode templates. Setting it to False causes
# calibre to use the old behavior -- interpreting the templates. Set the tweak
# to False if some compiled templates produce incorrect values.
# Default:    compile_gpm_templates = True
# No compile: compile_gpm_templates = False
compile_gpm_templates = True

#: What format to default to when using the Tweak feature
# The Tweak feature of calibre allows direct editing of a book format.
# If multiple formats are available, calibre will offer you a choice
# of formats, defaulting to your preferred output format if it is available.
# Set this tweak to a specific value of 'EPUB' or 'AZW3' to always default
# to that format rather than your output format preference.
# Set to a value of 'remember' to use whichever format you chose last time you
# used the Tweak feature.
# Examples:
#   default_tweak_format = None       (Use output format)
#   default_tweak_format = 'EPUB'
#   default_tweak_format = 'remember'
default_tweak_format = None

#: Do not preselect a completion when editing authors/tags/series/etc.
# This means that you can make changes and press Enter and your changes will
# not be overwritten by a matching completion. However, if you wish to use the
# completions you will now have to press Tab to select one before pressing
# Enter. Which technique you prefer will depend on the state of metadata in
# your library and your personal editing style.
preselect_first_completion = False

#: Recognize numbers inside text when sorting
# This means that when sorting on text fields like title the text "Book 2"
# will sort before the text "Book 100". If you want this behavior, set
```

```
# numeric_collation = True note that doing so will cause problems with text
# that starts with numbers and is a little slower.
numeric_collation = False

#: Sort the list of libraries alphabetically
# The list of libraries in the Copy to Library and Quick Switch menus are
# normally sorted by most used. However, if there are more than a certain
# number of such libraries, the sorting becomes alphabetic. You can set that
# number here. The default is ten libraries.
many_libraries = 10
```

Overriding icons, templates, et cetera

calibre allows you to override the static resources, like icons, templates, javascript, etc. with customized versions that you like. All static resources are stored in the resources sub-folder of the calibre install location. On Windows, this is usually `C:/Program Files/Calibre2/resources`. On OS X, `/Applications/calibre.app/Contents/Resources/resources/`. On linux, if you are using the binary installer from the calibre website it will be `/opt/calibre/resources`. These paths can change depending on where you choose to install calibre.

You should not change the files in this resources folder, as your changes will get overwritten the next time you update calibre. Instead, go to *Preferences->Advanced->Miscellaneous* and click *Open calibre configuration directory*. In this configuration directory, create a sub-folder called resources and place the files you want to override in it. Place the files in the appropriate sub folders, for example place images in `resources/images`, etc. calibre will automatically use your custom file in preference to the built-in one the next time it is started.

For example, if you wanted to change the icon for the *Remove books* action, you would first look in the built-in resources folder and see that the relevant file is `resources/images/trash.png`. Assuming you have an alternate icon in PNG format called `mytrash.png` you would save it in the configuration directory as `resources/images/trash.png`. All the icons used by the calibre user interface are in `resources/images` and its sub-folders.

Customizing calibre with plugins

calibre has a very modular design. Almost all functionality in calibre comes in the form of plugins. Plugins are used for conversion, for downloading news (though these are called recipes), for various components of the user interface, to connect to different devices, to process files when adding them to calibre and so on. You can get a complete list of all the built-in plugins in calibre by going to *Preferences->Plugins*.

You can write your own plugins to customize and extend the behavior of calibre. The plugin architecture in calibre is very simple, see the tutorial *Writing your own plugins to extend calibre's functionality* (page 423).

1.20 The Command Line Interface

1.20.1 Command Line Interface



On OS X you have to go to Preferences->Advanced->Miscellaneous and click install command line tools to make the command line tools available. On other platforms, just start a terminal and type the command.

Documented Commands

calibre

```
calibre [opts] [path_to_ebook]
```

Launch the main **calibre** Graphical User Interface and optionally add the ebook at path_to_ebook to the database.

Whenever you pass arguments to **calibre** that have spaces in them, enclose the arguments in quotation marks.

[options]

-help, -h
show this help message and exit

-ignore-plugins
Ignore custom plugins, useful if you installed a plugin that is preventing calibre from starting

-no-update-check
Do not check for updates

-shutdown-running-calibre, -s
Cause a running calibre instance, if any, to be shutdown. Note that if there are running jobs, they will be silently aborted, so use with care.

-start-in-tray
Start minimized to system tray.

-verbose, -v
Log debugging information to console

-version
show program's version number and exit

-with-library
Use the library located at the specified path.

calibre-customize

```
calibre-customize options
```

Customize calibre by loading external plugins.

Whenever you pass arguments to **calibre-customize** that have spaces in them, enclose the arguments in quotation marks.

[options]

-add-plugin, -a
Add a plugin by specifying the path to the zip file containing it.

-build-plugin, -b
For plugin developers: Path to the directory where you are developing the plugin. This command will automatically zip up the plugin and update it in calibre.

-customize-plugin
Customize plugin. Specify name of plugin and customization string separated by a comma.

-disable-plugin
Disable the named plugin

-enable-plugin
Enable the named plugin

-help, -h
show this help message and exit

-list-plugins, -l
List all installed plugins

-remove-plugin, -r
Remove a custom plugin by name. Has no effect on builtin plugins

-version
show program's version number and exit

calibre-debug

```
calibre-debug [options]
```

Various command line interfaces useful for debugging calibre. With no options, this command starts an embedded python interpreter. You can also run the main calibre GUI and the calibre viewer in debug mode.

It also contains interfaces to various bits of calibre that do not have dedicated command line tools, such as font subsetting, tweaking ebooks and so on.

You can also use **calibre-debug** to run standalone scripts. To do that use it like this:

> **calibre-debug** mysrcipt.py – –option1 –option2 file1 file2 ...

Everything after the – is passed to the script.

Whenever you pass arguments to **calibre-debug** that have spaces in them, enclose the arguments in quotation marks.

[options]
-add-simple-plugin
Add a simple plugin (i.e. a plugin that consists of only a .py file), by specifying the path to the py file containing the plugin code.
-command, -c
Run python code.

-debug-device-driver, -d
Debug device detection

-exec-file, -e
Run the python code in file.

-gui, -g
Run the GUI with debugging enabled. Debug output is printed to stdout and stderr.

-gui-debug
Run the GUI with a debug console, logging to the specified path. For internal use only, use the -g option to run the GUI in debug mode

-help, -h
show this help message and exit

-inspect-mobi, -m
Inspect the MOBI file(s) at the specified path(s)

-paths
Output the paths necessary to setup the calibre environment

-py-console, -p
Run python console

-reinitialize-db
Re-initialize the sqlite calibre database at the specified path. Useful to recover from db corruption. You can also specify the path to an SQL dump which will be used instead of trying to dump the database. This can be useful when dumping fails, but dumping with sqlite3 works.

-show-gui-debug
Display the specified log file. For internal use only.

-shutdown-running-calibre, -s
Cause a running calibre instance, if any, to be shutdown. Note that if there are running jobs, they will be silently aborted, so use with care.

-subset-font, -f
Subset the specified font

-test-build
Test binary modules in build

-tweak-book, -t
Tweak the book (exports the book as a collection of HTML files and metadata, which you can edit using standard HTML editing tools, and then rebuilds the file from the edited HTML. Makes no additional changes to the HTML, unlike a full calibre conversion).

-version
show program's version number and exit

-viewer, -w
Run the ebook viewer

calibre-server

```
calibre-server [options]
```

Start the calibre content server. The calibre content server exposes your calibre library over the internet. The default interface allows you to browse you calibre library by categories. You can also access an interface optimized for mobile browsers at /mobile and an OPDS based interface for use with reading applications at /opds.

The OPDS interface is advertised via BonJour automatically.

Whenever you pass arguments to **calibre-server** that have spaces in them, enclose the arguments in quotation marks.

[options]
-auto-reload
Auto reload server when source code changes. May not work in all environments.
-daemonize
Run process in background as a daemon. No effect on windows.

-develop
Development mode. Server automatically restarts on file changes and serves code files (html, css, js) from the file system instead of calibre's resource system.

-help, -h
show this help message and exit

-max-cover
The maximum size for displayed covers. Default is '600x800'.

-max-opds-items
The maximum number of matches to return per OPDS query. This affects Stanza, WordPlayer, etc. integration.

-max-opds-ungrouped-items
Group items in categories such as author/tags by first letter when there are more than this number of items. Default: 100. Set to a large number to disable grouping.

-password
Set a password to restrict access. By default access is unrestricted.

-pidfile
Write process PID to the specified file

-port, -p
The port on which to listen. Default is 8080

-restriction
Specifies a restriction to be used for this invocation. This option overrides any per-library settings specified in the GUI

-thread-pool
The max number of worker threads to use. Default is 30

-timeout, -t
The server timeout in seconds. Default is 120

-url-prefix
Prefix to prepend to all URLs. Useful for reverseproxying to this server from Apache/nginx/etc.

-username
Username for access. By default, it is: 'calibre'

-version
show program's version number and exit

-with-library
Path to the library folder to serve with the content server

calibre-smtp

```
calibre-smtp [options] [from to text]
```

Send mail using the SMTP protocol. **calibre-smtp** has two modes of operation. In the compose mode you specify from to and text and these are used to build and send an email message. In the filter mode, **calibre-smtp** reads a complete email message from STDIN and sends it.

text is the body of the email message. If text is not specified, a complete email message is read from STDIN. from is the email address of the sender and to is the email address of the recipient. When a complete email is read from STDIN, from and to are only used in the SMTP negotiation, the message headers are not modified.

Whenever you pass arguments to **calibre-smtp** that have spaces in them, enclose the arguments in quotation marks.

[options]

-fork, -f
Fork and deliver message in background. If you use this option, you should also use –outbox to handle delivery failures.

-help, -h
show this help message and exit

-localhost, -l
Host name of localhost. Used when connecting to SMTP server.

-outbox, -o
Path to maildir folder to store failed email messages in.

-timeout, -t
Timeout for connection

-verbose, -v
Be more verbose

-version
show program's version number and exit

COMPOSE MAIL Options to compose an email. Ignored if text is not specified

-attachment, -a
File to attach to the email

-subject, -s
Subject of the email

SMTP RELAY Options to use an SMTP relay server to send mail. calibre will try to send the email directly unless –relay is specified.

-encryption-method, -e
Encryption method to use when connecting to relay. Choices are TLS, SSL and NONE. Default is TLS. WARNING: Choosing NONE is highly insecure

-password, -p
Password for relay

-port
Port to connect to on relay server. Default is to use 465 if encryption method is SSL and 25 otherwise.

-relay, -r
An SMTP relay server to use to send mail.

-username, -u
Username for relay

calibredb

```
calibredb command [options] [arguments]
```

calibredb is the command line interface to the calibre database. It has several sub-commands, documented below:

- *list* (page 478)
- *add* (page 479)
- *remove* (page 479)
- *add_format* (page 480)
- *remove_format* (page 480)
- *show_metadata* (page 480)
- *set_metadata* (page 481)

- *export* (page 481)
- *catalog* (page 482)
- *saved_searches* (page 482)
- *add_custom_column* (page 483)
- *custom_columns* (page 483)
- *remove_custom_column* (page 483)
- *set_custom* (page 484)
- *restore_database* (page 484)
- *check_library* (page 484)
- *list_categories* (page 485)
- *backup_metadata* (page 485)

Global options

-dont-notify-gui
Do not notify the running calibre GUI (if any) that the database has changed. Use with care, as it can lead to database corruption!

-library-path
Path to the calibre library. Default is to use the path stored in the settings.

list

```
calibredb list [options]
```

List the books available in the calibre database.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-ascending
Sort results in ascending order

-fields, -f
The fields to display when listing books in the database. Should be a comma separated list of fields. Available fields: author_sort,authors,comments,cover,formats,identifiers,isbn,last_modified,pubdate,publisher,rating,series,series_index,size
Default: title,authors. The special field "all" can be used to select all fields. Only has effect in the text output format.

-help, -h
show this help message and exit

-line-width, -w
The maximum width of a single line in the output. Defaults to detecting screen size.

-prefix
The prefix for all file paths. Default is the absolute path to the library folder.

-search, -s
Filter the results by the search query. For the format of the search query, please see the search related documentation in the User Manual. Default is to do no filtering.

-separator
The string used to separate fields. Default is a space.

-sort-by
The field by which to sort the results. Available fields: publisher,series_index,formats,isbn,uuid,pubdate,rating,series,timestamp,author_sort,cover,comments,identifiers,last_modified,author
Default: None

-version
show program's version number and exit

add

```
calibredb add [options] file1 file2 file3 ...
```

Add the specified files as books to the database. You can also specify directories, see the directory related options below.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-authors, -a
Set the authors of the added book(s)

-cover, -c
Path to the cover to use for the added book

-duplicates, -d
Add books to database even if they already exist. Comparison is done based on book titles.

-empty, -e
Add an empty book (a book with no formats)

-help, -h
show this help message and exit

-isbn, -i
Set the ISBN of the added book(s)

-one-book-per-directory, -1
Assume that each directory has only a single logical book and that all files in it are different e-book formats of that book

-recurse, -r
Process directories recursively

-series, -s
Set the series of the added book(s)

-series-index, -S
Set the series number of the added book(s)

-tags, -T
Set the tags of the added book(s)

-title, -t
Set the title of the added book(s)

-version
show program's version number and exit

remove

```
calibredb remove ids
```

Remove the books identified by ids from the database. ids should be a comma separated list of id numbers (you can get id numbers by using the list command). For example, 23,34,57-85 (when specifying a range, the last number in the range is not included).

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-help, -h
show this help message and exit

-version
show program's version number and exit

add_format

```
calibredb add_format [options] id ebook_file
```

Add the ebook in ebook_file to the available formats for the logical book identified by id. You can get id by using the list command. If the format already exists, it is replaced.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-help, -h
show this help message and exit

-version
show program's version number and exit

remove_format

```
calibredb remove_format [options] id fmt
```

Remove the format fmt from the logical book identified by id. You can get id by using the list command. fmt should be a file extension like LRF or TXT or EPUB. If the logical book does not have fmt available, do nothing.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-help, -h
show this help message and exit

-version
show program's version number and exit

show_metadata

```
calibredb show_metadata [options] id
```

Show the metadata stored in the calibre database for the book identified by id. id is an id number from the list command.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-as-opf
Print metadata in OPF form (XML)

-help, -h
show this help message and exit

-version
show program's version number and exit

set_metadata

```
calibredb set_metadata [options] id /path/to/metadata.opf
```

Set the metadata stored in the calibre database for the book identified by id from the OPF file metadata.opf. id is an id number from the list command. You can get a quick feel for the OPF format by using the –as-opf switch to the show_metadata command. You can also set the metadata of individual fields with the –field option.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-help, -h
show this help message and exit

-version
show program's version number and exit

export

```
calibredb export [options] ids
```

Export the books specified by ids (a comma separated list) to the filesystem. The **export** operation saves all formats of the book, its cover and metadata (in an opf file). You can get id numbers from the list command.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-all
Export all books in database, ignoring the list of ids.

-dont-asciiize
Normally, calibre will convert all non English characters into English equivalents for the file names. WARNING: If you turn this off, you may experience errors when saving, depending on how well the filesystem you are saving to supports unicode. Specifying this switch will turn this behavior off.

-dont-save-cover
Normally, calibre will save the cover in a separate file along with the actual e-book file(s). Specifying this switch will turn this behavior off.

-dont-update-metadata
Normally, calibre will update the metadata in the saved files from what is in the calibre library. Makes saving to disk slower. Specifying this switch will turn this behavior off.

-dont-write-opf
Normally, calibre will write the metadata into a separate OPF file along with the actual e-book files. Specifying this switch will turn this behavior off.

-formats
Comma separated list of formats to save for each book. By default all available formats are saved.

-help, -h
show this help message and exit

-replace-whitespace
Replace whitespace with underscores.

-single-dir
Export all books into a single directory

-template
The template to control the filename and directory structure of the saved files. Default is "{author_sort}/{title}/{title} - {authors}" which will save books into a per-author subdirectory with filenames containing title and author. Available controls are: {publisher, series_index, isbn, pubdate, rating, series, author_sort, authors, last_modified, timestamp, title, id, tags}

-timefmt
The format in which to display dates. %d - day, %b - month, %m - month number, %Y - year. Default is: %b, %Y

-to-dir
Export books to the specified directory. Default is .

-to-lowercase
Convert paths to lowercase.

-version
show program's version number and exit

catalog

```
calibredb catalog /path/to/destination.(CSV|EPUB|MOBI|XML ...) [options]
```

Export a **catalog** in format specified by path/to/destination extension. Options control how entries are displayed in the generated **catalog** ouput.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-fields
The fields to output when cataloging books in the database. Should be a comma-separated list of fields. Available fields: all, title, title_sort, author_sort, authors, comments, cover, formats, id, isbn, library_name, ondevice, pubdate, publisher, rating, series_index, series, size, tags, timestamp, uuid, languages, identifiers, plus user-created custom fields. Example: –fields=title,authors,tags Default: 'all' Applies to: CSV, XML output formats

-help, -h
show this help message and exit

-ids, -i
Comma-separated list of database IDs to catalog. If declared, –search is ignored. Default: all

-search, -s
Filter the results by the search query. For the format of the search query, please see the search-related documentation in the User Manual. Default: no filtering

-sort-by
Output field to sort on. Available fields: author_sort, id, rating, size, timestamp, title_sort Default: 'id' Applies to: CSV, XML output formats

-verbose, -v
Show detailed output information. Useful for debugging

-version
show program's version number and exit

saved_searches

```
calibredb saved_searches [options] list
```

calibredb **saved_searches** add name search calibredb **saved_searches** remove name

Manage the saved searches stored in this database. If you try to add a query with a name that already exists, it will be replaced.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-help, -h
show this help message and exit

-version
show program's version number and exit

add_custom_column

```
calibredb add_custom_column [options] label name datatype
```

Create a custom column. label is the machine friendly name of the column. Should not contain spaces or colons. name is the human friendly name of the column. datatype is one of: rating, int, text, float, comments, datetime, composite, bool, enumeration, series

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-display
A dictionary of options to customize how the data in this column will be interpreted. This is a JSON string. For enumeration columns, use –display="{"enum_values":["val1", "val2"]}" There are many options that can go into the display variable.The options by column type are: composite: composite_template, composite_sort, make_category,contains_html, use_decorations datetime: date_format enumeration: enum_values, enum_colors, use_decorations int, float: number_format text: is_names, use_decorations The best way to find legal combinations is to create a customcolumn of the appropriate type in the GUI then look at thebackup OPF for a book (ensure that a new OPF has been createdsince the column was added). You will see the JSON for the"display" for the new column in the OPF.

-help, -h
show this help message and exit

-is-multiple
This column stores tag like data (i.e. multiple comma separated values). Only applies if datatype is text.

-version
show program's version number and exit

custom_columns

```
calibredb custom_columns [options]
```

List available custom columns. Shows column labels and ids.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-details, -d
Show details for each column.

-help, -h
show this help message and exit

-version
show program's version number and exit

remove_custom_column

```
calibredb remove_custom_column [options] label
```

Remove the custom column identified by label. You can see available columns with the custom_columns command.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-force, -f
Do not ask for confirmation

-help, -h
show this help message and exit

-version
show program's version number and exit

set_custom

```
calibredb set_custom [options] column id value
```

Set the value of a custom column for the book identified by id. You can get a list of ids using the list command. You can get a list of custom column names using the custom_columns command.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-append, -a
If the column stores multiple values, append the specified values to the existing ones, instead of replacing them.

-help, -h
show this help message and exit

-version
show program's version number and exit

restore_database

```
calibredb restore_database [options]
```

Restore this database from the metadata stored in OPF files in each directory of the calibre library. This is useful if your metadata.db file has been corrupted.

WARNING: This command completely regenerates your database. You will lose all saved searches, user categories, plugboards, stored per-book conversion settings, and custom recipes. Restored metadata will only be as accurate as what is found in the OPF files.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-help, -h
show this help message and exit

-really-do-it, -r
Really do the recovery. The command will not run unless this option is specified.

-version
show program's version number and exit

check_library

```
calibredb check_library [options]
```

Perform some checks on the filesystem representing a library. Reports are invalid_titles, extra_titles, invalid_authors, extra_authors, missing_formats, extra_formats, extra_files, missing_covers, extra_covers, failed_folders

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-csv, -c
Output in CSV

-help, -h
show this help message and exit

-ignore_extensions, -e
Comma-separated list of extensions to ignore. Default: all

-ignore_names, -n
Comma-separated list of names to ignore. Default: all

-report, -r
Comma-separated list of reports. Default: all

-version
show program's version number and exit

list_categories

```
calibredb list_categories [options]
```

Produce a report of the category information in the database. The information is the equivalent of what is shown in the tags pane.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-categories, -r
Comma-separated list of category lookup names. Default: all

-csv, -c
Output in CSV

-help, -h
show this help message and exit

-item_count, -i
Output only the number of items in a category instead of the counts per item within the category

-quote, -q
The character to put around the category value in CSV mode. Default is quotes (").

-separator, -s
The string used to separate fields in CSV mode. Default is a comma.

-version
show program's version number and exit

-width, -w
The maximum width of a single line in the output. Defaults to detecting screen size.

backup_metadata

```
calibredb backup_metadata [options]
```

Backup the metadata stored in the database into individual OPF files in each books directory. This normally happens automatically, but you can run this command to force re-generation of the OPF files, with the –all option.

Note that there is normally no need to do this, as the OPF files are backed up automatically, every time metadata is changed.

Whenever you pass arguments to calibredb that have spaces in them, enclose the arguments in quotation marks.

-all
Normally, this command only operates on books that have out of date OPF files. This option makes it operate on all books.

-help, -h
show this help message and exit

--version
show program's version number and exit

ebook-convert

```
ebook-convert input_file output_file [options]
```

Convert an ebook from one format to another.

input_file is the input and output_file is the output. Both must be specified as the first two arguments to the command.

The output ebook format is guessed from the file extension of output_file. output_file can also be of the special format .EXT where EXT is the output file extension. In this case, the name of the output file is derived the name of the input file. Note that the filenames must not start with a hyphen. Finally, if output_file has no extension, then it is treated as a directory and an "open ebook" (OEB) consisting of HTML files is written to that directory. These files are the files that would normally have been passed to the output plugin.

After specifying the input and output file you can customize the conversion by specifying various options. The available options depend on the input and output file types. To get help on them specify the input and output file and then use the -h option.

For full documentation of the conversion system see *Ebook Conversion* (page 305)

Whenever you pass arguments to **ebook-convert** that have spaces in them, enclose the arguments in quotation marks.

The options and default values for the options change depending on both the input and output formats, so you should always check with:

```
ebook-convert myfile.input_format myfile.output_format -h
```

Below are the options that are common to all conversion, followed by the options specific to every input and output format

- Look And Feel (page 488)
- Heuristic Processing (page 490)
- Search And Replace (page 490)
- Structure Detection (page 491)
- Table Of Contents (page 491)
- Metadata (page 492)
- Debug (page 493)
- AZW4 Input Options (page 493)
- CHM Input Options (page 493)
- Comic Input Options (page 493)
- DJVU Input Options (page 494)
- EPUB Input Options (page 494)
- FB2 Input Options (page 494)
- HTLZ Input Options (page 495)
- HTML Input Options (page 495)
- LIT Input Options (page 495)
- LRF Input Options (page 495)
- MOBI Input Options (page 495)
- ODT Input Options (page 495)
- PDB Input Options (page 496)
- PDF Input Options (page 496)
- PML Input Options (page 496)
- RB Input Options (page 496)
- RTF Input Options (page 496)
- Recipe Input Options (page 496)
- SNB Input Options (page 497)
- TCR Input Options (page 497)
- TXT Input Options (page 497)
- AZW3 Output Options (page 497)
- EPUB Output Options (page 498)
- FB2 Output Options (page 499)
- HTML Output Options (page 499)
- HTMLZ Output Options (page 499)
- LIT Output Options (page 500)
- LRF Output Options (page 500)
- MOBI Output Options (page 500)
- OEB Output Options (page 501)
- PDB Output Options (page 501)
- PDF Output Options (page 501)
- PML Output Options (page 502)
- RB Output Options (page 503)
- RTF Output Options (page 503)
- SNB Output Options (page 503)
- TCR Output Options (page 503)
- TXT Output Options (page 504)
- TXTZ Output Options (page 504)

-help, -h
show this help message and exit

-input-profile
Specify the input profile. The input profile gives the conversion system information on how to interpret various information in the input document. For example resolution dependent lengths (i.e. lengths in pixels). Choices are:cybookg3, cybook_opus, default, hanlinv3, hanlinv5, illiad, irexdr1000, irexdr800, kindle, msreader, mo-

bipocket, nook, sony, sony300, sony900

-list-recipes
List builtin recipe names. You can create an ebook from a builtin recipe like this: ebook-convert "Recipe Name.recipe" output.epub

-output-profile
Specify the output profile. The output profile tells the conversion system how to optimize the created document for the specified device. In some cases, an output profile is required to produce documents that will work on a device. For example EPUB on the SONY reader. Choices are:cybookg3, cybook_opus, default, generic_eink, generic_eink_large, hanlinv3, hanlinv5, illiad, ipad, ipad3, irexdr1000, irexdr800, jetbook5, kindle, kindle_dx, kindle_fire, kindle_pw, kobo, msreader, mobipocket, nook, nook_color, nook_hd_plus, pocketbook_900, pocketbook_pro_912, galaxy, bambook, sony, sony300, sony900, sony-landscape, tablet

-version
show program's version number and exit

Look And Feel Options to control the look and feel of the output

-asciiize
Transliterate unicode characters to an ASCII representation. Use with care because this will replace unicode characters with ASCII. For instance it will replace "Михаил Горбачёв" with "Mikhail Gorbachiov". Also, note that in cases where there are multiple representations of a character (characters shared by Chinese and Japanese for instance) the representation based on the current calibre interface language will be used.

-base-font-size
The base font size in pts. All font sizes in the produced book will be rescaled based on this size. By choosing a larger size you can make the fonts in the output bigger and vice versa. By default, the base font size is chosen based on the output profile you chose.

-change-justification
Change text justification. A value of "left" converts all justified text in the source to left aligned (i.e. unjustified) text. A value of "justify" converts all unjustified text to justified. A value of "original" (the default) does not change justification in the source file. Note that only some output formats support justification.

-disable-font-rescaling
Disable all rescaling of font sizes.

-embed-font-family
Embed the specified font family into the book. This specifies the "base" font used for the book. If the input document specifies its own fonts, they may override this base font. You can use the filter style information option to remove fonts from the input document. Note that font embedding only works with some output formats, principally EPUB and AZW3.

-extra-css
Either the path to a CSS stylesheet or raw CSS. This CSS will be appended to the style rules from the source file, so it can be used to override those rules.

-filter-css
A comma separated list of CSS properties that will be removed from all CSS style rules. This is useful if the presence of some style information prevents it from being overridden on your device. For example: font-family,color,margin-left,margin-right

-font-size-mapping
Mapping from CSS font names to font sizes in pts. An example setting is 12,12,14,16,18,20,22,24. These are the mappings for the sizes xx-small to xx-large, with the final size being for huge fonts. The font rescaling algorithm uses these sizes to intelligently rescale fonts. The default is to use a mapping based on the output profile you chose.

-insert-blank-line
Insert a blank line between paragraphs. Will not work if the source file does not use paragraphs (<p> or tags).

-insert-blank-line-size
Set the height of the inserted blank lines (in em). The height of the lines between paragraphs will be twice the value set here.

-keep-ligatures
Preserve ligatures present in the input document. A ligature is a special rendering of a pair of characters like ff, fi, fl et cetera. Most readers do not have support for ligatures in their default fonts, so they are unlikely to render correctly. By default, calibre will turn a ligature into the corresponding pair of normal characters. This option will preserve them instead.

-line-height
The line height in pts. Controls spacing between consecutive lines of text. Only applies to elements that do not define their own line height. In most cases, the minimum line height option is more useful. By default no line height manipulation is performed.

-linearize-tables
Some badly designed documents use tables to control the layout of text on the page. When converted these documents often have text that runs off the page and other artifacts. This option will extract the content from the tables and present it in a linear fashion.

-margin-bottom
Set the bottom margin in pts. Default is 5.0. Setting this to less than zero will cause no margin to be set. Note: 72 pts equals 1 inch

-margin-left
Set the left margin in pts. Default is 5.0. Setting this to less than zero will cause no margin to be set. Note: 72 pts equals 1 inch

-margin-right
Set the right margin in pts. Default is 5.0. Setting this to less than zero will cause no margin to be set. Note: 72 pts equals 1 inch

-margin-top
Set the top margin in pts. Default is 5.0. Setting this to less than zero will cause no margin to be set. Note: 72 pts equals 1 inch

-minimum-line-height
The minimum line height, as a percentage of the element's calculated font size. calibre will ensure that every element has a line height of at least this setting, irrespective of what the input document specifies. Set to zero to disable. Default is 120%. Use this setting in preference to the direct line height specification, unless you know what you are doing. For example, you can achieve "double spaced" text by setting this to 240.

-remove-paragraph-spacing
Remove spacing between paragraphs. Also sets an indent on paragraphs of 1.5em. Spacing removal will not work if the source file does not use paragraphs (<p> or tags).

-remove-paragraph-spacing-indent-size
When calibre removes blank lines between paragraphs, it automatically sets a paragraph indent, to ensure that paragraphs can be easily distinguished. This option controls the width of that indent (in em). If you set this value negative, then the indent specified in the input document is used, that is, calibre does not change the indentation.

-smarten-punctuation
Convert plain quotes, dashes and ellipsis to their typographically correct equivalents. For details, see http://daringfireball.net/projects/smartypants

-subset-embedded-fonts
Subset all embedded fonts. Every embedded font is reduced to contain only the glyphs used in this document. This decreases the size of the font files. Useful if you are embedding a particularly large font with lots of unused glyphs.

-unsmarten-punctuation
Convert fancy quotes, dashes and ellipsis to their plain equivalents.

Heuristic Processing Modify the document text and structure using common patterns. Disabled by default. Use –enable-heuristics to enable. Individual actions can be disabled with the –disable-* options.

-disable-dehyphenate
Analyze hyphenated words throughout the document. The document itself is used as a dictionary to determine whether hyphens should be retained or removed.

-disable-delete-blank-paragraphs
Remove empty paragraphs from the document when they exist between every other paragraph

-disable-fix-indents
Turn indentation created from multiple non-breaking space entities into CSS indents.

-disable-format-scene-breaks
Left aligned scene break markers are center aligned. Replace soft scene breaks that use multiple blank lines with horizontal rules.

-disable-italicize-common-cases
Look for common words and patterns that denote italics and italicize them.

-disable-markup-chapter-headings
Detect unformatted chapter headings and sub headings. Change them to h2 and h3 tags. This setting will not create a TOC, but can be used in conjunction with structure detection to create one.

-disable-renumber-headings
Looks for occurrences of sequential <h1> or <h2> tags. The tags are renumbered to prevent splitting in the middle of chapter headings.

-disable-unwrap-lines
Unwrap lines using punctuation and other formatting clues.

-enable-heuristics
Enable heuristic processing. This option must be set for any heuristic processing to take place.

-html-unwrap-factor
Scale used to determine the length at which a line should be unwrapped. Valid values are a decimal between 0 and 1. The default is 0.4, just below the median line length. If only a few lines in the document require unwrapping this value should be reduced

-replace-scene-breaks
Replace scene breaks with the specified text. By default, the text from the input document is used.

Search And Replace Modify the document text and structure using user defined patterns.

-search-replace
Path to a file containing search and replace regular expressions. The file must contain alternating lines of regular expression followed by replacement pattern (which can be an empty line). The regular expression must be in the python regex syntax and the file must be UTF-8 encoded.

-sr1-replace
Replacement to replace the text found with sr1-search.

-sr1-search
Search pattern (regular expression) to be replaced with sr1-replace.

-sr2-replace
Replacement to replace the text found with sr2-search.

-sr2-search
Search pattern (regular expression) to be replaced with sr2-replace.

-sr3-replace
Replacement to replace the text found with sr3-search.

-sr3-search
Search pattern (regular expression) to be replaced with sr3-replace.

Structure Detection Control auto-detection of document structure.

-chapter
An XPath expression to detect chapter titles. The default is to consider <h1> or <h2> tags that contain the words “chapter”,”book”,”section”, “prologue”, “epilogue”, or “part” as chapter titles as well as any tags that have class=”chapter”. The expression used must evaluate to a list of elements. To disable chapter detection, use the expression “/”. See the XPath Tutorial in the calibre User Manual for further help on using this feature.

-chapter-mark
Specify how to mark detected chapters. A value of “pagebreak” will insert page breaks before chapters. A value of “rule” will insert a line before chapters. A value of “none” will disable chapter marking and a value of “both” will use both page breaks and lines to mark chapters.

-disable-remove-fake-margins
Some documents specify page margins by specifying a left and right margin on each individual paragraph. calibre will try to detect and remove these margins. Sometimes, this can cause the removal of margins that should not have been removed. In this case you can disable the removal.

-insert-metadata
Insert the book metadata at the start of the book. This is useful if your ebook reader does not support displaying/searching metadata directly.

-page-breaks-before
An XPath expression. Page breaks are inserted before the specified elements. To disable use the expression: /

-prefer-metadata-cover
Use the cover detected from the source file in preference to the specified cover.

-remove-first-image
Remove the first image from the input ebook. Useful if the input document has a cover image that is not identified as a cover. In this case, if you set a cover in calibre, the output document will end up with two cover images if you do not specify this option.

-start-reading-at
An XPath expression to detect the location in the document at which to start reading. Some ebook reading programs (most prominently the Kindle) use this location as the position at which to open the book. See the XPath tutorial in the calibre User Manual for further help using this feature.

Table Of Contents Control the automatic generation of a Table of Contents. By default, if the source file has a Table of Contents, it will be used in preference to the automatically generated one.

-duplicate-links-in-toc
When creating a TOC from links in the input document, allow duplicate entries, i.e. allow more than one entry with the same text, provided that they point to a different location.

-level1-toc
XPath expression that specifies all tags that should be added to the Table of Contents at level one. If this is specified, it takes precedence over other forms of auto-detection. See the XPath Tutorial in the calibre User Manual for examples.

-level2-toc
XPath expression that specifies all tags that should be added to the Table of Contents at level two. Each entry is added under the previous level one entry. See the XPath Tutorial in the calibre User Manual for examples.

-level3-toc
XPath expression that specifies all tags that should be added to the Table of Contents at level three. Each entry is added under the previous level two entry. See the XPath Tutorial in the calibre User Manual for examples.

-max-toc-links
Maximum number of links to insert into the TOC. Set to 0 to disable. Default is: 50. Links are only added to the TOC if less than the threshold number of chapters were detected.

-no-chapters-in-toc
Don't add auto-detected chapters to the Table of Contents.

-toc-filter
Remove entries from the Table of Contents whose titles match the specified regular expression. Matching entries and all their children are removed.

-toc-threshold
If fewer than this number of chapters is detected, then links are added to the Table of Contents. Default: 6

-use-auto-toc
Normally, if the source file already has a Table of Contents, it is used in preference to the auto-generated one. With this option, the auto-generated one is always used.

Metadata Options to set metadata in the output

-author-sort
String to be used when sorting by author.

-authors
Set the authors. Multiple authors should be separated by ampersands.

-book-producer
Set the book producer.

-comments
Set the ebook description.

-cover
Set the cover to the specified file or URL

-isbn
Set the ISBN of the book.

-language
Set the language.

-pubdate
Set the publication date.

-publisher
Set the ebook publisher.

-rating
Set the rating. Should be a number between 1 and 5.

-series
Set the series this ebook belongs to.

-series-index
Set the index of the book in this series.

-tags
Set the tags for the book. Should be a comma separated list.

-timestamp
Set the book timestamp (no longer used anywhere)

-title
Set the title.

-title-sort
The version of the title to be used for sorting.

Debug Options to help with debugging the conversion

-debug-pipeline, -d
Save the output from different stages of the conversion pipeline to the specified directory. Useful if you are unsure at which stage of the conversion process a bug is occurring.

-verbose, -v
Level of verbosity. Specify multiple times for greater verbosity.

AZW4 Input Options
-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

CHM Input Options
-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

Comic Input Options
-colors
Number of colors for grayscale image conversion. Default: 256. Values of less than 256 may result in blurred text on your device if you are creating your comics in EPUB format.
-comic-image-size
Specify the image size as widthxheight pixels. Normally, an image size is automatically calculated from the output profile, this option overrides it.

-despeckle
Enable Despeckle. Reduces speckle noise. May greatly increase processing time.

-disable-trim
Disable trimming of comic pages. For some comics, trimming might remove content as well as borders.

-dont-add-comic-pages-to-toc
When converting a CBC do not add links to each page to the TOC. Note this only applies if the TOC has more than one section

-dont-grayscale
Do not convert the image to grayscale (black and white)

-dont-normalize
Disable normalize (improve contrast) color range for pictures. Default: False

-dont-sharpen
Disable sharpening.

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

-keep-aspect-ratio
Maintain picture aspect ratio. Default is to fill the screen.

-landscape
Don't split landscape images into two portrait images

-no-process
Apply no processing to the image

-no-sort
Don't sort the files found in the comic alphabetically by name. Instead use the order they were added to the comic.

-output-format
The format that images in the created ebook are converted to. You can experiment to see which format gives you optimal size and look on your device.

-right2left
Used for right-to-left publications like manga. Causes landscape pages to be split into portrait pages from right to left.

-wide
Keep aspect ratio and scale image using screen height as image width for viewing in landscape mode.

DJVU Input Options
-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.
-use-djvutxt
Try to use the djvutxt program and fall back to pure python implementation if it fails or is not available

EPUB Input Options
-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

FB2 Input Options
-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

--no-inline-fb2-toc
Do not insert a Table of Contents at the beginning of the book.

HTLZ Input Options
--input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

HTML Input Options
--breadth-first
Traverse links in HTML files breadth first. Normally, they are traversed depth first.
--dont-package
Normally this input plugin re-arranges all the input files into a standard folder hierarchy. Only use this option if you know what you are doing as it can result in various nasty side effects in the rest of the conversion pipeline.

--input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

--max-levels
Maximum levels of recursion when following links in HTML files. Must be non-negative. 0 implies that no links in the root HTML file are followed. Default is 5.

LIT Input Options
--input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

LRF Input Options
--input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

MOBI Input Options
--input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

ODT Input Options
--input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

PDB Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

PDF Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

-new-pdf-engine
Use the new PDF conversion engine.

-no-images
Do not extract images from the document

-unwrap-factor
Scale used to determine the length at which a line should be unwrapped. Valid values are a decimal between 0 and 1. The default is 0.45, just below the median line length.

PML Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

RB Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

RTF Input Options

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

Recipe Input Options

-dont-download-recipe
Do not download latest version of builtin recipes from the calibre server

-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

-lrf
Optimize fetching for subsequent conversion to LRF.

-password
Password for sites that require a login to access content.

-test
Useful for recipe development. Forces max_articles_per_feed to 2 and downloads at most 2 feeds.

-username
Username for sites that require a login to access content.

SNB Input Options
-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

TCR Input Options
-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

TXT Input Options
-formatting-type
Formatting used within the document.* auto: Automatically decide which formatting processor to use. * plain: Do not process the document formatting. Everything is a paragraph and no styling is applied. * heuristic: Process using heuristics to determine formatting such as chapter headings and italic text. * textile: Processing using textile formatting. * markdown: Processing using markdown formatting. To learn more about markdown see http://daringfireball.net/projects/markdown/
-input-encoding
Specify the character encoding of the input document. If set this option will override any encoding declared by the document itself. Particularly useful for documents that do not declare an encoding or that have erroneous encoding declarations.

-markdown-disable-toc
Do not insert a Table of Contents into the output text.

-paragraph-type
Paragraph structure. choices are ['auto', 'block', 'single', 'print', 'unformatted', 'off'] * auto: Try to auto detect paragraph type. * block: Treat a blank line as a paragraph break. * single: Assume every line is a paragraph. * print: Assume every line starting with 2+ spaces or a tab starts a paragraph. * unformatted: Most lines have hard line breaks, few/no blank lines or indents. Tries to determine structure and reformat the differentiate elements. * off: Don't modify the paragraph structure. This is useful when combined with Markdown or Textile formatting to ensure no formatting is lost.

-preserve-spaces
Normally extra spaces are condensed into a single space. With this option all spaces will be displayed.

-txt-in-remove-indents
Normally extra space at the beginning of lines is retained. With this option they will be removed.

AZW3 Output Options
-dont-compress
Disable compression of the file contents.
-extract-to
Extract the contents of the MOBI file to the specified directory. If the directory already exists, it will be deleted.

-mobi-toc-at-start
When adding the Table of Contents to the book, add it at the start of the book instead of the end. Not recommended.

-no-inline-toc
Don't add Table of Contents to the book. Useful if the book has its own table of contents.

-prefer-author-sort
When present, use author sort field as author.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-share-not-sync
Enable sharing of book content via Facebook etc. on the Kindle. WARNING: Using this feature means that the book will not auto sync its last read position on multiple devices. Complain to Amazon.

-toc-title
Title for any generated in-line table of contents.

EPUB Output Options

-dont-split-on-page-breaks
Turn off splitting at page breaks. Normally, input files are automatically split at every page break into two files. This gives an output ebook that can be parsed faster and with less resources. However, splitting is slow and if your source file contains a very large number of page breaks, you should turn off splitting on page breaks.

-epub-flatten
This option is needed only if you intend to use the EPUB with FBReaderJ. It will flatten the file system inside the EPUB, putting all files into the top level.

-extract-to
Extract the contents of the generated EPUB file to the specified directory. The contents of the directory are first deleted, so be careful.

-flow-size
Split all HTML files larger than this size (in KB). This is necessary as most EPUB readers cannot handle large file sizes. The default of 260KB is the size required for Adobe Digital Editions.

-no-default-epub-cover
Normally, if the input file has no cover and you don't specify one, a default cover is generated with the title, authors, etc. This option disables the generation of this cover.

-no-svg-cover
Do not use SVG for the book cover. Use this option if your EPUB is going to be used on a device that does not support SVG, like the iPhone or the JetBook Lite. Without this option, such devices will display the cover as a blank page.

-preserve-cover-aspect-ratio
When using an SVG cover, this option will cause the cover to scale to cover the available screen area, but still preserve its aspect ratio (ratio of width to height). That means there may be white borders at the sides or top and bottom of the image, but the image will never be distorted. Without this option the image may be slightly distorted, but there will be no borders.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

FB2 Output Options

-fb2-genre

Genre for the book. Choices: sf_history, sf_action, sf_epic, sf_heroic, sf_detective, sf_cyberpunk, sf_space, sf_social, sf_horror, sf_humor, sf_fantasy, sf, det_classic, det_police, det_action, det_irony, det_history, det_espionage, det_crime, det_political, det_maniac, det_hard, thriller, detective, prose_classic, prose_history, prose_contemporary, prose_counter, prose_rus_classic, prose_su_classics, love_contemporary, love_history, love_detective, love_short, love_erotica, adv_western, adv_history, adv_indian, adv_maritime, adv_geo, adv_animal, adventure, child_tale, child_verse, child_prose, child_sf, child_det, child_adv, child_education, children, poetry, dramaturgy, antique_ant, antique_european, antique_russian, antique_east, antique_myths, antique, sci_history, sci_psychology, sci_culture, sci_religion, sci_philosophy, sci_politics, sci_business, sci_juris, sci_linguistic, sci_medicine, sci_phys, sci_math, sci_chem, sci_biology, sci_tech, science, comp_www, comp_programming, comp_hard, comp_soft, comp_db, comp_osnet, computers, ref_encyc, ref_dict, ref_ref, ref_guide, reference, nonf_biography, nonf_publicism, nonf_criticism, design, nonfiction, religion_rel, religion_esoterics, religion_self, religion, humor_anecdote, humor_prose, humor_verse, humor, home_cooking, home_pets, home_crafts, home_entertain, home_health, home_garden, home_diy, home_sport, home_sex, home See: http://www.fictionbook.org/index.php/Eng:FictionBook_2.1_genres for a complete list with descriptions.

-pretty-print

If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-sectionize

Specify the sectionization of elements. A value of "nothing" turns the book into a single section. A value of "files" turns each file into a separate section; use this if your device is having trouble. A value of "Table of Contents" turns the entries in the Table of Contents into titles and creates sections; if it fails, adjust the "Structure Detection" and/or "Table of Contents" settings (turn on "Force use of auto-generated Table of Contents").

HTML Output Options

-extract-to

Extract the contents of the generated ZIP file to the specified directory. WARNING: The contents of the directory will be deleted.

-pretty-print

If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-template-css

CSS file used for the output instead of the default file

-template-html

Template used for the generation of the html contents of the book instead of the default file

-template-html-index

Template used for generation of the html index file instead of the default file

HTMLZ Output Options

-htmlz-class-style

How to handle the CSS when using css-type = 'class'. Default is external. external: Use an external CSS file that is linked in the document. inline: Place the CSS in the head section of the document.

-htmlz-css-type

Specify the handling of CSS. Default is class. class: Use CSS classes and have elements reference them. inline: Write the CSS as an inline style attribute. tag: Turn as many CSS styles as possible into HTML tags.

-htmlz-title-filename

If set this option causes the file name of the html file inside the htmlz archive to be based on the book title.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

LIT Output Options
-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

LRF Output Options
-enable-autorotation
Enable autorotation of images that are wider than the screen width.
-header
Add a header to all the pages with title and author.

-header-format
Set the format of the header. %a is replaced by the author and %t by the title. Default is %t by %a

-header-separation
Add extra spacing below the header. Default is 0 pt.

-minimum-indent
Minimum paragraph indent (the indent of the first line of a paragraph) in pts. Default: 0

-mono-family
The monospace family of fonts to embed

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-render-tables-as-images
Render tables in the HTML as images (useful if the document has large or complex tables)

-sans-family
The sans-serif family of fonts to embed

-serif-family
The serif family of fonts to embed

-text-size-multiplier-for-rendered-tables
Multiply the size of text in rendered tables by this factor. Default is 1.0

-wordspace
Set the space between words in pts. Default is 2.5

MOBI Output Options
-dont-compress
Disable compression of the file contents.
-extract-to
Extract the contents of the MOBI file to the specified directory. If the directory already exists, it will be deleted.

-mobi-file-type
By default calibre generates MOBI files that contain the old MOBI 6 format. This format is compatible with all devices. However, by changing this setting, you can tell calibre to generate MOBI files that contain both MOBI 6 and the new KF8 format, or only the new KF8 format. KF8 has more features than MOBI 6, but only works with newer Kindles.

-mobi-ignore-margins
Ignore margins in the input document. If False, then the MOBI output plugin will try to convert margins specified in the input document, otherwise it will ignore them.

-mobi-keep-original-images
By default calibre converts all images to JPEG format in the output MOBI file. This is for maximum compatibility as some older MOBI viewers have problems with other image formats. This option tells calibre not to do this. Useful if your document contains lots of GIF/PNG images that become very large when converted to JPEG.

-mobi-toc-at-start
When adding the Table of Contents to the book, add it at the start of the book instead of the end. Not recommended.

-no-inline-toc
Don't add Table of Contents to the book. Useful if the book has its own table of contents.

-personal-doc
Tag marking book to be filed with Personal Docs

-prefer-author-sort
When present, use author sort field as author.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-share-not-sync
Enable sharing of book content via Facebook etc. on the Kindle. WARNING: Using this feature means that the book will not auto sync its last read position on multiple devices. Complain to Amazon.

-toc-title
Title for any generated in-line table of contents.

OEB Output Options
-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

PDB Output Options
-format, -f
Format to use inside the pdb container. Choices are: ['doc', 'ztxt', 'ereader']
-inline-toc
Add Table of Contents to beginning of the book.

-pdb-output-encoding
Specify the character encoding of the output document. The default is cp1252. Note: This option is not honored by all formats.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

PDF Output Options
-custom-size
Custom size of the document. Use the form widthxheight EG. *123x321* to specify the width and height. This overrides any specified paper-size.

-old-pdf-engine
Use the old, less capable engine to generate the PDF

-override-profile-size
Normally, the PDF page size is set by the output profile chosen under page options. This option will cause the page size settings under PDF Output to override the size specified by the output profile.

-paper-size
The size of the paper. This size will be overridden when a non default output profile is used. Default is letter. Choices are [u'a0', u'a1', u'a2', u'a3', u'a4', u'a5', u'a6', u'b0', u'b1', u'b2', u'b3', u'b4', u'b5', u'b6', u'legal', u'letter']

-pdf-default-font-size
The default font size

-pdf-footer-template
An HTML template used to generate footers on every page. The string _PAGENUM_ will be replaced by the current page number.

-pdf-header-template
An HTML template used to generate headers on every page. The string _PAGENUM_ will be replaced by the current page number.

-pdf-mark-links
Surround all links with a red box, useful for debugging.

-pdf-mono-family
The font family used to render monospaced fonts

-pdf-mono-font-size
The default font size for monospaced text

-pdf-page-numbers
Add page numbers to the bottom of every page in the generated PDF file. If you specify a footer template, it will take precedence over this option.

-pdf-sans-family
The font family used to render sans-serif fonts

-pdf-serif-family
The font family used to render serif fonts

-pdf-standard-font
The font family used to render monospaced fonts

-preserve-cover-aspect-ratio
Preserve the aspect ratio of the cover, instead of stretching it to fill the full first page of the generated pdf.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-uncompressed-pdf
Generate an uncompressed PDF, useful for debugging, only works with the new PDF engine.

-unit, -u
The unit of measure for page sizes. Default is inch. Choices are ['millimeter', 'centimeter', 'point', 'inch', 'pica', 'didot', 'cicero', 'devicepixel'] Note: This does not override the unit for margins!

PML Output Options

-full-image-depth
Do not reduce the size or bit depth of images. Images have their size and depth reduced by default to accommodate applications that can not convert images on their own such as Dropbook.

-inline-toc
Add Table of Contents to beginning of the book.

-pml-output-encoding
Specify the character encoding of the output document. The default is cp1252.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

RB Output Options

-inline-toc
Add Table of Contents to beginning of the book.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

RTF Output Options

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

SNB Output Options

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-snb-dont-indent-first-line
Specify whether or not to insert two space characters to indent the first line of each paragraph.

-snb-full-screen
Resize all the images for full screen view.

-snb-hide-chapter-name
Specify whether or not to hide the chapter title for each chapter. Useful for image-only output (eg. comics).

-snb-insert-empty-line
Specify whether or not to insert an empty line between two paragraphs.

-snb-max-line-length
The maximum number of characters per line. This splits on the first space before the specified value. If no space is found the line will be broken at the space after and will exceed the specified value. Also, there is a minimum of 25 characters. Use 0 to disable line splitting.

-snb-output-encoding
Specify the character encoding of the output document. The default is utf-8.

TCR Output Options

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-tcr-output-encoding
Specify the character encoding of the output document. The default is utf-8.

TXT Output Options

-force-max-line-length
Force splitting on the max-line-length value when no space is present. Also allows max-line-length to be below the minimum

-inline-toc
Add Table of Contents to beginning of the book.

-keep-color
Do not remove font color from output. This is only useful when txt-output-formatting is set to textile. Textile is the only formatting that supports setting font color. If this option is not specified font color will not be set and default to the color displayed by the reader (generally this is black).

-keep-image-references
Do not remove image references within the document. This is only useful when paired with a txt-output-formatting option that is not none because links are always removed with plain text output.

-keep-links
Do not remove links within the document. This is only useful when paired with a txt-output-formatting option that is not none because links are always removed with plain text output.

-max-line-length
The maximum number of characters per line. This splits on the first space before the specified value. If no space is found the line will be broken at the space after and will exceed the specified value. Also, there is a minimum of 25 characters. Use 0 to disable line splitting.

-newline, -n
Type of newline to use. Options are ['old_mac', 'system', 'unix', 'windows']. Default is 'system'. Use 'old_mac' for compatibility with Mac OS 9 and earlier. For Mac OS X use 'unix'. 'system' will default to the newline type used by this OS.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-txt-output-encoding
Specify the character encoding of the output document. The default is utf-8.

-txt-output-formatting
Formatting used within the document. * plain: Produce plain text. * markdown: Produce Markdown formatted text. * textile: Produce Textile formatted text.

TXTZ Output Options

-force-max-line-length
Force splitting on the max-line-length value when no space is present. Also allows max-line-length to be below the minimum

-inline-toc
Add Table of Contents to beginning of the book.

-keep-color
Do not remove font color from output. This is only useful when txt-output-formatting is set to textile. Textile is the only formatting that supports setting font color. If this option is not specified font color will not be set and default to the color displayed by the reader (generally this is black).

-keep-image-references
Do not remove image references within the document. This is only useful when paired with a txt-output-formatting option that is not none because links are always removed with plain text output.

-keep-links
Do not remove links within the document. This is only useful when paired with a txt-output-formatting option that is not none because links are always removed with plain text output.

-max-line-length
The maximum number of characters per line. This splits on the first space before the specified value. If no space is found the line will be broken at the space after and will exceed the specified value. Also, there is a minimum of 25 characters. Use 0 to disable line splitting.

-newline, -n
Type of newline to use. Options are ['old_mac', 'system', 'unix', 'windows']. Default is 'system'. Use 'old_mac' for compatibility with Mac OS 9 and earlier. For Mac OS X use 'unix'. 'system' will default to the newline type used by this OS.

-pretty-print
If specified, the output plugin will try to create output that is as human readable as possible. May not have any effect for some output plugins.

-txt-output-encoding
Specify the character encoding of the output document. The default is utf-8.

-txt-output-formatting
Formatting used within the document. * plain: Produce plain text. * markdown: Produce Markdown formatted text. * textile: Produce Textile formatted text.

ebook-meta

```
ebook-meta ebook_file [options]
```

Read/Write metadata from/to ebook files.

Supported formats for reading metadata: azw, azw1, azw3, azw4, cbr, cbz, chm, docx, epub, fb2, html, htmlz, imp, lit, lrf, lrx, mobi, odt, oebzip, opf, pdb, pdf, pml, pmlz, pobi, prc, rar, rb, rtf, snb, tpz, txt, txtz, updb, zip

Supported formats for writing metadata: azw, azw1, azw3, azw4, epub, fb2, htmlz, lrf, mobi, pdb, pdf, prc, rtf, tpz, txtz

Different file types support different kinds of metadata. If you try to set some metadata on a file type that does not support it, the metadata will be silently ignored.

Whenever you pass arguments to **ebook-meta** that have spaces in them, enclose the arguments in quotation marks.

[options]
-author-sort
String to be used when sorting by author. If unspecified, and the author(s) are specified, it will be auto-generated from the author(s).
-authors, -a
Set the authors. Multiple authors should be separated by the & character. Author names should be in the order Firstname Lastname.

-book-producer, -k
Set the book producer.

-category
Set the book category.

-comments, -c
Set the ebook description.

-cover
Set the cover to the specified file.

-date, -d
Set the published date.

-from-opf
Read metadata from the specified OPF file and use it to set metadata in the ebook. Metadata specified on the command line will override metadata read from the OPF file

-get-cover
Get the cover from the ebook and save it at as the specified file.

-help, -h
show this help message and exit

-index, -i
Set the index of the book in this series.

-isbn
Set the ISBN of the book.

-language, -l
Set the language.

-lrf-bookid
Set the BookID in LRF files

-publisher, -p
Set the ebook publisher.

-rating, -r
Set the rating. Should be a number between 1 and 5.

-series, -s
Set the series this ebook belongs to.

-tags
Set the tags for the book. Should be a comma separated list.

-title, -t
Set the title.

-title-sort
The version of the title to be used for sorting. If unspecified, and the title is specified, it will be auto-generated from the title.

-to-opf
Specify the name of an OPF file. The metadata will be written to the OPF file.

-version
show program's version number and exit

ebook-polish

```
ebook-polish [options] input_file [output_file]
```

Polishing books is all about putting the shine of perfection onto your carefully crafted ebooks.

Polishing tries to minimize the changes to the internal code of your ebook. Unlike conversion, it does not flatten CSS, rename files, change font sizes, adjust margins, etc. Every action performs only the minimum set of changes needed for the desired effect.

You should use this tool as the last step in your ebook creation process.

Note that polishing only works on files in the AZW3 or EPUB formats.

Whenever you pass arguments to **ebook-polish** that have spaces in them, enclose the arguments in quotation marks.

[options]

-cover, -c
Path to a cover image. Changes the cover specified in the ebook. If no cover is present, or the cover is not properly identified, inserts a new cover.

-help, -h
show this help message and exit

-jacket, -j
Insert a "book jacket" page at the start of the book that contains all the book metadata such as title, tags, authors, series, comments, etc.

-opf, -o
Path to an OPF file. The metadata in the book is updated from the OPF file.

-remove-jacket
Remove a previous inserted book jacket page.

-smarten-punctuation, -p
Convert plain text dashes, ellipsis, quotes, multiple hyphens, etc. into their typographically correct equivalents. Note that the algorithm can sometimes generate incorrect results, especially when single quotes at the start of contractions are involved.

-subset-fonts, -f
Subsetting fonts means reducing an embedded font to contain only the characters used from that font in the book. This greatly reduces the size of the font files (halving the font file sizes is common). For example, if the book uses a specific font for headers, then subsetting will reduce that font to contain only the characters present in the actual headers in the book. Or if the book embeds the bold and italic versions of a font, but bold and italic text is relatively rare, or absent altogether, then the bold and italic fonts can either be reduced to only a few characters or completely removed. The only downside to subsetting fonts is that if, at a later date you decide to add more text to your books, the newly added text might not be covered by the subset font.

-verbose
Produce more verbose output, useful for debugging.

-version
show program's version number and exit

ebook-viewer

```
ebook-viewer [options] file
```

View an ebook.

Whenever you pass arguments to **ebook-viewer** that have spaces in them, enclose the arguments in quotation marks.

[options]

-debug-javascript
Print javascript alert and console messages to the console

-full-screen, -f
If specified, viewer window will try to open full screen when started.

-help, -h
show this help message and exit

-open-at
The position at which to open the specified book. The position is a location as displayed in the top left corner of the viewer.

-raise-window
If specified, viewer window will try to come to the front when started.

-version
show program's version number and exit

fetch-ebook-metadata

```
fetch-ebook-metadata [options]
```

Fetch book metadata from online sources. You must specify at least one of title, authors or ISBN.

Whenever you pass arguments to **fetch-ebook-metadata** that have spaces in them, enclose the arguments in quotation marks.

[options]
-authors, -a
Book author(s)
-cover, -c
Specify a filename. The cover, if available, will be saved to it

-help, -h
show this help message and exit

-isbn, -i
Book ISBN

-opf, -o
Output the metadata in OPF format

-timeout, -d
Timeout in seconds. Default is 30

-title, -t
Book title

-verbose, -v
Print the log to the console (stderr)

-version
show program's version number and exit

lrf2lrs

```
lrf2lrs book.lrf
```

Convert an LRF file into an LRS (XML UTF-8 encoded) file

Whenever you pass arguments to **lrf2lrs** that have spaces in them, enclose the arguments in quotation marks.

[options]

--dont-output-resources
Do not save embedded image and font files to disk

-h, --help
show this help message and exit

-o, --output
Output LRS file

--verbose

--version
show program's version number and exit

lrfviewer

```
lrfviewer [options] book.lrf
```

Read the LRF ebook book.lrf

Whenever you pass arguments to **lrfviewer** that have spaces in them, enclose the arguments in quotation marks.

[options]

--disable-hyphenation
Disable hyphenation. Should significantly speed up rendering.

-h, --help
show this help message and exit

--profile
Profile the LRF renderer

--verbose
Print more information about the rendering process

--version
show program's version number and exit

--visual-debug
Turn on visual aids to debugging the rendering engine

--white-background
By default the background is off white as I find this easier on the eyes. Use this option to make the background pure white.

lrs2lrf

```
lrs2lrf [options] file.lrs
```

Compile an LRS file into an LRF file.

Whenever you pass arguments to **lrs2lrf** that have spaces in them, enclose the arguments in quotation marks.

[options]

-h, --help
show this help message and exit

-lrs
Convert LRS to LRS, useful for debugging.

-output, -o
Path to output file

-verbose
Verbose processing

-version
show program's version number and exit

web2disk

```
web2disk URL
```

Where URL is for example http://google.com

Whenever you pass arguments to **web2disk** that have spaces in them, enclose the arguments in quotation marks.

[options]
-base-dir, -d
Base directory into which URL is saved. Default is .
-delay
Minimum interval in seconds between consecutive fetches. Default is 0 s

-dont-download-stylesheets
Do not download CSS stylesheets.

-encoding
The character encoding for the websites you are trying to download. The default is to try and guess the encoding.

-filter-regexp
Any link that matches this regular expression will be ignored. This option can be specified multiple times, in which case as long as any regexp matches a link, it will be ignored. By default, no links are ignored. If both filter regexp and match regexp are specified, then filter regexp is applied first.

-help, -h
show this help message and exit

-match-regexp
Only links that match this regular expression will be followed. This option can be specified multiple times, in which case as long as a link matches any one regexp, it will be followed. By default all links are followed.

-max-files, -n
The maximum number of files to download. This only applies to files from <a href> tags. Default is 9223372036854775807

-max-recursions, -r
Maximum number of levels to recurse i.e. depth of links to follow. Default 1

-timeout, -t
Timeout in seconds to wait for a response from the server. Default: 10.0 s

-verbose
Show detailed output information. Useful for debugging

-version
show program's version number and exit

Undocumented Commands

- ebook-device
- markdown-calibre

You can see usage for undocumented commands by executing them without arguments in a terminal.

1.21 Setting up a calibre development environment

1.21.1 Setting up a calibre development environment

calibre is completely open source, licensed under the GNU GPL v3[199]. This means that you are free to download and modify the program to your heart's content. In this section, you will learn how to get a calibre development environment set up on the operating system of your choice. calibre is written primarily in Python[200] with some C/C++ code for speed and system interfacing. Note that calibre is not compatible with Python 3 and requires at least Python 2.7.

Contents

- Design philosophy (page 511)
  - Code layout (page 512)
- Getting the code (page 512)
  - Submitting your changes to be included (page 513)
- Windows development environment (page 513)
- OS X development environment (page 514)
- Linux development environment (page 514)
- Having separate "normal" and "development" calibre installs on the same computer (page 515)
- Debugging tips (page 515)
  - Using an interactive python interpreter (page 515)
  - Using print statements (page 515)
  - Using the debugger in PyDev (page 516)
  - Executing arbitrary scripts in the calibre python environment (page 516)
- Using calibre in your projects (page 516)
  - Binary install of calibre (page 516)
  - Source install on Linux (page 516)

Design philosophy

calibre has its roots in the Unix world, which means that its design is highly modular. The modules interact with each other via well defined interfaces. This makes adding new features and fixing bugs in calibre very easy, resulting in a frenetic pace of development. Because of its roots, calibre has a comprehensive command line interface for all its functions, documented in *Command Line Interface* (page 472).

The modular design of calibre is expressed via `Plugins`. There is a *tutorial* (page 440) on writing calibre plugins. For example, adding support for a new device to calibre typically involves writing less than a 100 lines of code in the form of a device driver plugin. You can browse the built-in drivers[201]. Similarly, adding support for new conversion formats

[199] http://www.gnu.org/copyleft/gpl.html
[200] http://www.python.org
[201] http://bazaar.launchpad.net/%7Ekovid/calibre/trunk/files/head%3A/src/calibre/devices/

involves writing input/output format plugins. Another example of the modular design is the *recipe system* (page 341) for fetching news. For more examples of plugins designed to add features to calibre, see the plugin index[202].

Code layout

All the calibre python code is in the `calibre` package. This package contains the following main sub-packages

- devices - All the device drivers. Just look through some of the built-in drivers to get an idea for how they work.
  - For details, see: devices.interface which defines the interface supported by device drivers and `devices.usbms` which defines a generic driver that connects to a USBMS device. All USBMS based drivers in calibre inherit from it.
- ebooks - All the ebook conversion/metadata code. A good starting point is `calibre.ebooks.conversion.cli` which is the module powering the **ebook-convert** command. The conversion process is controlled via `conversion.plumber`. The format independent code is all in `ebooks.oeb` and the format dependent code is in `ebooks.format_name`.
  - Metadata reading, writing, and downloading is all in `ebooks.metadata`
  - Conversion happens in a pipeline, for the structure of the pipeline, see *Introduction* (page 306). The pipeline consists of an input plugin, various transforms and an output plugin. The that code constructs and drives the pipeline is in `plumber.py`. The pipeline works on a representation of an ebook that is like an unzipped epub, with manifest, spine, toc, guide, html content, etc. The class that manages this representation is OEBBook in `ebooks.oeb.base`. The various transformations that are applied to the book during conversions live in `oeb/transforms/*.py`. And the input and output plugins live in `conversion/plugins/*.py`.
- library - The database back-end and the content server. See `library.database2` for the interface to the calibre library. `library.server` is the calibre Content Server.
- gui2 - The Graphical User Interface. GUI initialization happens in `gui2.main` and `gui2.ui`. The ebook-viewer is in `gui2.viewer`.

If you need help understanding the code, post in the development forum[203] and you will most likely get help from one of calibre's many developers.

Getting the code

calibre uses Bazaar[204], a distributed version control system. Bazaar is available on all the platforms calibre supports. After installing Bazaar, you can get the calibre source code with the command:

```
bzr branch lp:calibre
```

On Windows you will need the complete path name, that will be something like `C:\Program Files\Bazaar\bzr.exe`.

calibre is a very large project with a very long source control history, so the above can take a while (10mins to an hour depending on your internet speed).

If you want to get the code faster, the sourcecode for the latest release is always available as an archive[205]. You can also use bzr to just download the source code, without the history, using:

[202] http://www.mobileread.com/forums/showthread.php?p=1362767#post1362767
[203] http://www.mobileread.com/forums/forumdisplay.php?f=240
[204] http://bazaar-vcs.org/
[205] http://status.calibre-ebook.com/dist/src

```
bzr branch --stacked lp:calibre
```

To update a branch to the latest code, use the command:

```
bzr merge
```

Submitting your changes to be included

If you only plan to make a few small changes, you can make your changes and create a "merge directive" which you can then attach to a ticket in the calibre bug tracker for consideration. To do this, make your changes, then run:

```
bzr commit -m "Comment describing your changes"
bzr send -o my-changes
```

This will create a `my-changes` file in the current directory, simply attach that to a ticket on the calibre bug tracker[206].

If you plan to do a lot of development on calibre, then the best method is to create a Launchpad[207] account. Once you have an account, you can use it to register your bzr branch created by the *bzr branch* command above. First run the following command to tell bzr about your launchpad account:

```
bzr launchpad-login your_launchpad_username
```

Now, you have to setup SSH access to Launchpad. First create an SSH public/private keypair. Then upload the public key to Launchpad by going to your Launchpad account page. Instructions for setting up the private key in bzr are at http://bazaar-vcs.org/Bzr_and_SSH. Now you can upload your branch to the calibre project in Launchpad by following the instructions at https://help.launchpad.net/Code/UploadingABranch. Whenever you commit changes to your branch with the command:

```
bzr commit -m "Comment describing your change"
```

Kovid can merge it directly from your branch into the main calibre source tree. You should also keep an eye on the calibre development forum[208]. Before making major changes, you should discuss them in the forum or contact Kovid directly (his email address is all over the source code).

Windows development environment

Install calibre normally, using the Windows installer. Then open a Command Prompt and change to the previously checked out calibre code directory. For example:

```
cd C:\Users\kovid\work\calibre
```

calibre is the directory that contains the src and resources sub-directories.

The next step is to set the environment variable `CALIBRE_DEVELOP_FROM` to the absolute path of the src directory. So, following the example above, it would be `C:\Users\kovid\work\calibre\src`. Here is a short guide[209] to setting environment variables on Windows.

Once you have set the environment variable, open a new command prompt and check that it was correctly set by using the command:

```
echo %CALIBRE_DEVELOP_FROM%
```

[206] https://bugs.launchpad.net/calibre
[207] http://launchpad.net
[208] http://www.mobileread.com/forums/forumdisplay.php?f=240
[209] http://docs.python.org/using/windows.html#excursus-setting-environment-variables

Setting this environment variable means that calibre will now load all its Python code from the specified location.

That's it! You are now ready to start hacking on the calibre code. For example, open the file `src\calibre\__init__.py` in your favorite editor and add the line:

```
print ("Hello, world!")
```

near the top of the file. Now run the command **calibredb**. The very first line of output should be `Hello, world!`.

OS X development environment

Install calibre normally using the provided .dmg. Then open a Terminal and change to the previously checked out calibre code directory, for example:

```
cd /Users/kovid/work/calibre
```

calibre is the directory that contains the src and resources sub-directories. Ensure you have installed the calibre commandline tools via *Preferences->Advanced->Miscellaneous* in the calibre GUI.

The next step is to create a bash script that will set the environment variable `CALIBRE_DEVELOP_FROM` to the absolute path of the src directory when running calibre in debug mode.

Create a plain text file:

```
#!/bin/sh
export CALIBRE_DEVELOP_FROM="/Users/kovid/work/calibre/src"
calibre-debug -g
```

Save this file as `/usr/bin/calibre-develop`, then set its permissions so that it can be executed:

```
chmod +x /usr/bin/calibre-develop
```

Once you have done this, run:

```
calibre-develop
```

You should see some diagnostic information in the Terminal window as calibre starts up, and you should see an asterisk after the version number in the GUI window, indicating that you are running from source.

Linux development environment

calibre is primarily developed on Linux. You have two choices in setting up the development environment. You can install the calibre binary as normal and use that as a runtime environment to do your development. This approach is similar to that used in Windows and OS X. Alternatively, you can install calibre from source. Instructions for setting up a development environment from source are in the INSTALL file in the source tree. Here we will address using the binary at runtime, which is the recommended method.

Install the calibre using the binary installer. Then open a terminal and change to the previously checked out calibre code directory, for example:

```
cd /home/kovid/work/calibre
```

calibre is the directory that contains the src and resources sub-directories.

The next step is to set the environment variable `CALIBRE_DEVELOP_FROM` to the absolute path of the src directory. So, following the example above, it would be `/home/kovid/work/calibre/src`. How to set environment variables depends on your Linux distribution and what shell you are using.

Once you have set the environment variable, open a new terminal and check that it was correctly set by using the command:

```
echo $CALIBRE_DEVELOP_FROM
```

Setting this environment variable means that calibre will now load all its Python code from the specified location.

That's it! You are now ready to start hacking on the calibre code. For example, open the file `src/calibre/__init__.py` in your favorite editor and add the line:

```
print ("Hello, world!")
```

near the top of the file. Now run the command **calibredb**. The very first line of output should be `Hello, world!`.

Having separate "normal" and "development" calibre installs on the same computer

The calibre source tree is very stable and rarely breaks, but if you feel the need to run from source on a separate test library and run the released calibre version with your everyday library, you can achieve this easily using .bat files or shell scripts to launch calibre. The example below shows how to do this on Windows using .bat files (the instructions for other platforms are the same, just use a shell script instead of a .bat file)

To launch the release version of calibre with your everyday library:

calibre-normal.bat:

```
calibre.exe "--with-library=C:\path\to\everyday\library folder"
```

calibre-dev.bat:

```
set CALIBRE_DEVELOP_FROM=C:\path\to\calibre\checkout\src
calibre.exe "--with-library=C:\path\to\test\library folder"
```

Debugging tips

Python is a dynamically typed language with excellent facilities for introspection. Kovid wrote the core calibre code without once using a debugger. There are many strategies to debug calibre code:

Using an interactive python interpreter

You can insert the following two lines of code to start an interactive python session at that point:

```
from calibre import ipython
ipython(locals())
```

When running from the command line, this will start an interactive Python interpreter with access to all locally defined variables (variables in the local scope). The interactive prompt even has TAB completion for object properties and you can use the various Python facilities for introspection, such as `dir()`, `type()`, `repr()`, etc.

Using print statements

This is Kovid's favorite way to debug. Simply insert print statements at points of interest and run your program in the terminal. For example, you can start the GUI from the terminal as:

```
calibre-debug -g
```

Similarly, you can start the ebook-viewer as:

```
calibre-debug -w /path/to/file/to/be/viewed
```

Using the debugger in PyDev

It is possible to get the debugger in PyDev working with the calibre development environment, see the forum thread[210].

Executing arbitrary scripts in the calibre python environment

The **calibre-debug** command provides a couple of handy switches to execute your own code, with access to the calibre modules:

```
calibre-debug -c "some python code"
```

is great for testing a little snippet of code on the command line. It works in the same way as the -c switch to the python interpreter:

```
calibre-debug -e myscript.py
```

can be used to execute your own Python script. It works in the same way as passing the script to the Python interpreter, except that the calibre environment is fully initialized, so you can use all the calibre code in your script.

Using calibre in your projects

It is possible to directly use calibre functions/code in your Python project. Two ways exist to do this:

Binary install of calibre

If you have a binary install of calibre, you can use the Python interpreter bundled with calibre, like this:

```
calibre-debug -e /path/to/your/python/script.py
```

Source install on Linux

In addition to using the above technique, if you do a source install on Linux, you can also directly import calibre, as follows:

```
import init_calibre
import calibre

print calibre.__version__
```

It is essential that you import the init_calibre module before any other calibre modules/packages as it sets up the interpreter to run calibre code.

[210] http://www.mobileread.com/forums/showthread.php?t=143208

Python Module Index

C

calibre.customize, 441
calibre.customize.conversion, 447
calibre.devices.interface, 449
calibre.ebooks.metadata.book.base, 412
calibre.ebooks.metadata.sources.base, 445
calibre.utils.formatter_functions, 404
calibre.web.feeds.news, 360

Index

Symbols

–add-plugin, -a
    calibre-customize command line option, 214, 473
–add-simple-plugin
    calibre-debug command line option, 215, 474
–all
    calibredb-backup_metadata command line option, 227, 485
    calibredb-export command line option, 222, 481
–append, -a
    calibredb-set_custom command line option, 225, 484
–as-opf
    calibredb-show_metadata command line option, 222, 480
–ascending
    calibredb-list command line option, 219, 478
–asciiize
    ebook-convert command line option, 230, 488
–attachment, -a
    calibre-smtp command line option, 218, 477
–author-sort
    ebook-convert command line option, 234, 492
    ebook-meta command line option, 249, 505
–authors
    ebook-convert command line option, 234, 492
–authors, -a
    calibredb-add command line option, 220, 479
    ebook-meta command line option, 249, 505
    fetch-ebook-metadata command line option, 251, 508
–auto-reload
    calibre-server command line option, 216, 475
–base-dir, -d
    web2disk command line option, 254, 510
–base-font-size
    ebook-convert command line option, 230, 488
–book-producer
    ebook-convert command line option, 234, 492
–book-producer, -k
    ebook-meta command line option, 249, 505
–breadth-first
    ebook-convert-html-input command line option, 237, 495
–build-plugin, -b
    calibre-customize command line option, 214, 473
–categories, -r
    calibredb-list_categories command line option, 226, 485
–category
    ebook-meta command line option, 249, 505
–change-justification
    ebook-convert command line option, 230, 488
–chapter
    ebook-convert command line option, 233, 491
–chapter-mark
    ebook-convert command line option, 233, 491
–colors
    ebook-convert-comic-input command line option, 236, 493
–comic-image-size
    ebook-convert-comic-input command line option, 236, 493
–command, -c
    calibre-debug command line option, 215, 474
–comments
    ebook-convert command line option, 234, 492
–comments, -c
    ebook-meta command line option, 249, 505
–cover
    ebook-convert command line option, 234, 492
    ebook-meta command line option, 249, 505
–cover, -c
    calibredb-add command line option, 220, 479
    ebook-polish command line option, 250, 507

fetch-ebook-metadata command line option, 251, 508
–csv, -c
calibredb-check_library command line option, 226, 484
calibredb-list_categories command line option, 226, 485
–custom-size
ebook-convert-pdf-output command line option, 245, 501
–customize-plugin
calibre-customize command line option, 214, 473
–daemonize
calibre-server command line option, 216, 475
–date, -d
ebook-meta command line option, 249, 506
–debug-device-driver, -d
calibre-debug command line option, 215, 474
–debug-javascript
ebook-viewer command line option, 251, 507
–debug-pipeline, -d
ebook-convert command line option, 235, 493
–delay
web2disk command line option, 254, 510
–despeckle
ebook-convert-comic-input command line option, 236, 493
–details, -d
calibredb-custom_columns command line option, 225, 483
–develop
calibre-server command line option, 216, 475
–disable-dehyphenate
ebook-convert command line option, 232, 490
–disable-delete-blank-paragraphs
ebook-convert command line option, 232, 490
–disable-fix-indents
ebook-convert command line option, 232, 490
–disable-font-rescaling
ebook-convert command line option, 230, 488
–disable-format-scene-breaks
ebook-convert command line option, 232, 490
–disable-hyphenation
lrfviewer command line option, 253, 509
–disable-italicize-common-cases
ebook-convert command line option, 232, 490
–disable-markup-chapter-headings
ebook-convert command line option, 232, 490
–disable-plugin
calibre-customize command line option, 214, 473
–disable-remove-fake-margins
ebook-convert command line option, 233, 491
–disable-renumber-headings
ebook-convert command line option, 232, 490
–disable-trim
ebook-convert-comic-input command line option, 236, 493
–disable-unwrap-lines
ebook-convert command line option, 232, 490
–display
calibredb-add_custom_column command line option, 224, 483
–dont-add-comic-pages-to-toc
ebook-convert-comic-input command line option, 236, 493
–dont-asciiize
calibredb-export command line option, 222, 481
–dont-compress
ebook-convert-azw3-output command line option, 240, 497
ebook-convert-mobi-output command line option, 243, 500
–dont-download-recipe
ebook-convert-recipe-input command line option, 239, 496
–dont-download-stylesheets
web2disk command line option, 254, 510
–dont-grayscale
ebook-convert-comic-input command line option, 236, 493
–dont-normalize
ebook-convert-comic-input command line option, 236, 494
–dont-notify-gui
command line option, 219, 478
–dont-output-resources
lrf2lrs command line option, 252, 509
–dont-package
ebook-convert-html-input command line option, 237, 495
–dont-save-cover
calibredb-export command line option, 222, 481
–dont-sharpen
ebook-convert-comic-input command line option, 236, 494
–dont-split-on-page-breaks
ebook-convert-epub-output command line option, 241, 498
–dont-update-metadata
calibredb-export command line option, 222, 481
–dont-write-opf
calibredb-export command line option, 222, 481
–duplicate-links-in-toc
ebook-convert command line option, 234, 491
–duplicates, -d
calibredb-add command line option, 220, 479
–embed-font-family
ebook-convert command line option, 230, 488

–empty, -e
calibredb-add command line option, 220, 479
–enable-autorotation
ebook-convert-lrf-output command line option, 243, 500
–enable-heuristics
ebook-convert command line option, 232, 490
–enable-plugin
calibre-customize command line option, 215, 473
–encoding
web2disk command line option, 254, 510
–encryption-method, -e
calibre-smtp command line option, 218, 477
–epub-flatten
ebook-convert-epub-output command line option, 241, 498
–exec-file, -e
calibre-debug command line option, 215, 474
–extra-css
ebook-convert command line option, 230, 488
–extract-to
ebook-convert-azw3-output command line option, 240, 497
ebook-convert-epub-output command line option, 241, 498
ebook-convert-html-output command line option, 242, 499
ebook-convert-mobi-output command line option, 243, 500
–fb2-genre
ebook-convert-fb2-output command line option, 241, 499
–fields
calibredb-catalog command line option, 223, 482
–fields, -f
calibredb-list command line option, 219, 478
–filter-css
ebook-convert command line option, 230, 488
–filter-regexp
web2disk command line option, 254, 510
–flow-size
ebook-convert-epub-output command line option, 241, 498
–font-size-mapping
ebook-convert command line option, 230, 488
–force, -f
calibredb-remove_custom_column command line option, 225, 483
–force-max-line-length
ebook-convert-txt-output command line option, 247, 504
ebook-convert-txtz-output command line option, 248, 504
–fork, -f
calibre-smtp command line option, 218, 476
–format, -f
ebook-convert-pdb-output command line option, 244, 501
–formats
calibredb-export command line option, 223, 481
–formatting-type
ebook-convert-txt-input command line option, 240, 497
–from-opf
ebook-meta command line option, 249, 506
–full-image-depth
ebook-convert-pml-output command line option, 246, 502
–full-screen, -f
ebook-viewer command line option, 251, 507
–get-cover
ebook-meta command line option, 249, 506
–gui, -g
calibre-debug command line option, 215, 474
–gui-debug
calibre-debug command line option, 215, 474
–header
ebook-convert-lrf-output command line option, 243, 500
–header-format
ebook-convert-lrf-output command line option, 243, 500
–header-separation
ebook-convert-lrf-output command line option, 243, 500
–help, -h
calibre command line option, 214, 473
calibre-customize command line option, 215, 474
calibre-debug command line option, 215, 474
calibre-server command line option, 217, 475
calibre-smtp command line option, 218, 476
calibredb-add command line option, 220, 479
calibredb-add_custom_column command line option, 224, 483
calibredb-add_format command line option, 221, 480
calibredb-backup_metadata command line option, 227, 485
calibredb-catalog command line option, 223, 482
calibredb-check_library command line option, 226, 484
calibredb-custom_columns command line option, 225, 483
calibredb-export command line option, 223, 481
calibredb-list command line option, 219, 478
calibredb-list_categories command line option, 226, 485
calibredb-remove command line option, 221, 480

calibredb-remove_custom_column command line option, 225, 483
calibredb-remove_format command line option, 221, 480
calibredb-restore_database command line option, 226, 484
calibredb-saved_searches command line option, 224, 482
calibredb-set_custom command line option, 225, 484
calibredb-set_metadata command line option, 222, 481
calibredb-show_metadata command line option, 222, 480
ebook-convert command line option, 229, 487
ebook-meta command line option, 249, 506
ebook-polish command line option, 250, 507
ebook-viewer command line option, 251, 507
fetch-ebook-metadata command line option, 252, 508
lrf2lrs command line option, 252, 509
lrfviewer command line option, 253, 509
lrs2lrf command line option, 253, 509
web2disk command line option, 254, 510
–html-unwrap-factor
ebook-convert command line option, 232, 490
–htmlz-class-style
ebook-convert-htmlz-output command line option, 242, 499
–htmlz-css-type
ebook-convert-htmlz-output command line option, 242, 499
–htmlz-title-filename
ebook-convert-htmlz-output command line option, 242, 499
–ids, -i
calibredb-catalog command line option, 223, 482
–ignore-plugins
calibre command line option, 214, 473
–ignore_extensions, -e
calibredb-check_library command line option, 226, 484
–ignore_names, -n
calibredb-check_library command line option, 226, 485
–index, -i
ebook-meta command line option, 249, 506
–inline-toc
ebook-convert-pdb-output command line option, 244, 501
ebook-convert-pml-output command line option, 246, 503
ebook-convert-rb-output command line option, 246, 503
ebook-convert-txt-output command line option, 247, 504
ebook-convert-txtz-output command line option, 248, 504
–input-encoding
ebook-convert-azw4-input command line option, 235, 493
ebook-convert-chm-input command line option, 235, 493
ebook-convert-comic-input command line option, 236, 494
ebook-convert-djvu-input command line option, 237, 494
ebook-convert-epub-input command line option, 237, 494
ebook-convert-fb2-input command line option, 237, 494
ebook-convert-htlz-input command line option, 237, 495
ebook-convert-html-input command line option, 237, 495
ebook-convert-lit-input command line option, 238, 495
ebook-convert-lrf-input command line option, 238, 495
ebook-convert-mobi-input command line option, 238, 495
ebook-convert-odt-input command line option, 238, 495
ebook-convert-pdb-input command line option, 238, 496
ebook-convert-pdf-input command line option, 238, 496
ebook-convert-pml-input command line option, 239, 496
ebook-convert-rb-input command line option, 239, 496
ebook-convert-recipe-input command line option, 239, 496
ebook-convert-rtf-input command line option, 239, 496
ebook-convert-snb-input command line option, 239, 497
ebook-convert-tcr-input command line option, 240, 497
ebook-convert-txt-input command line option, 240, 497
–input-profile
ebook-convert command line option, 229, 487
–insert-blank-line
ebook-convert command line option, 231, 488
–insert-blank-line-size
ebook-convert command line option, 231, 489
–insert-metadata

ebook-convert command line option, 233, 491
–inspect-mobi, -m
calibre-debug command line option, 215, 474
–is-multiple
calibredb-add_custom_column command line option, 224, 483
–isbn
ebook-convert command line option, 235, 492
ebook-meta command line option, 249, 506
–isbn, -i
calibredb-add command line option, 220, 479
fetch-ebook-metadata command line option, 252, 508
–item_count, -i
calibredb-list_categories command line option, 227, 485
–jacket, -j
ebook-polish command line option, 250, 507
–keep-aspect-ratio
ebook-convert-comic-input command line option, 236, 494
–keep-color
ebook-convert-txt-output command line option, 247, 504
ebook-convert-txtz-output command line option, 248, 504
–keep-image-references
ebook-convert-txt-output command line option, 247, 504
ebook-convert-txtz-output command line option, 248, 504
–keep-ligatures
ebook-convert command line option, 231, 489
–keep-links
ebook-convert-txt-output command line option, 247, 504
ebook-convert-txtz-output command line option, 248, 504
–landscape
ebook-convert-comic-input command line option, 236, 494
–language
ebook-convert command line option, 235, 492
–language, -l
ebook-meta command line option, 249, 506
–level1-toc
ebook-convert command line option, 234, 491
–level2-toc
ebook-convert command line option, 234, 492
–level3-toc
ebook-convert command line option, 234, 492
–library-path
command line option, 219, 478
–line-height
ebook-convert command line option, 231, 489
–line-width, -w
calibredb-list command line option, 219, 478
–linearize-tables
ebook-convert command line option, 231, 489
–list-plugins, -l
calibre-customize command line option, 215, 474
–list-recipes
ebook-convert command line option, 230, 488
–localhost, -l
calibre-smtp command line option, 218, 476
–lrf
ebook-convert-recipe-input command line option, 239, 496
–lrf-bookid
ebook-meta command line option, 249, 506
–lrs
lrs2lrf command line option, 253, 510
–margin-bottom
ebook-convert command line option, 231, 489
–margin-left
ebook-convert command line option, 231, 489
–margin-right
ebook-convert command line option, 231, 489
–margin-top
ebook-convert command line option, 231, 489
–markdown-disable-toc
ebook-convert-txt-input command line option, 240, 497
–match-regexp
web2disk command line option, 254, 510
–max-cover
calibre-server command line option, 217, 475
–max-files, -n
web2disk command line option, 254, 510
–max-levels
ebook-convert-html-input command line option, 237, 495
–max-line-length
ebook-convert-txt-output command line option, 247, 504
ebook-convert-txtz-output command line option, 248, 505
–max-opds-items
calibre-server command line option, 217, 475
–max-opds-ungrouped-items
calibre-server command line option, 217, 476
–max-recursions, -r
web2disk command line option, 254, 510
–max-toc-links
ebook-convert command line option, 234, 492
–minimum-indent
ebook-convert-lrf-output command line option, 243, 500

–minimum-line-height
ebook-convert command line option, 231, 489
–mobi-file-type
ebook-convert-mobi-output command line option, 243, 500
–mobi-ignore-margins
ebook-convert-mobi-output command line option, 243, 500
–mobi-keep-original-images
ebook-convert-mobi-output command line option, 244, 501
–mobi-toc-at-start
ebook-convert-azw3-output command line option, 240, 497
ebook-convert-mobi-output command line option, 244, 501
–mono-family
ebook-convert-lrf-output command line option, 243, 500
–new-pdf-engine
ebook-convert-pdf-input command line option, 238, 496
–newline, -n
ebook-convert-txt-output command line option, 247, 504
ebook-convert-txtz-output command line option, 248, 505
–no-chapters-in-toc
ebook-convert command line option, 234, 492
–no-default-epub-cover
ebook-convert-epub-output command line option, 241, 498
–no-images
ebook-convert-pdf-input command line option, 238, 496
–no-inline-fb2-toc
ebook-convert-fb2-input command line option, 237, 495
–no-inline-toc
ebook-convert-azw3-output command line option, 240, 498
ebook-convert-mobi-output command line option, 244, 501
–no-process
ebook-convert-comic-input command line option, 236, 494
–no-sort
ebook-convert-comic-input command line option, 236, 494
–no-svg-cover
ebook-convert-epub-output command line option, 241, 498
–no-update-check
calibre command line option, 214, 473
–old-pdf-engine
ebook-convert-pdf-output command line option, 245, 502
–one-book-per-directory, -1
calibredb-add command line option, 220, 479
–open-at
ebook-viewer command line option, 251, 508
–opf, -o
ebook-polish command line option, 250, 507
fetch-ebook-metadata command line option, 252, 508
–outbox, -o
calibre-smtp command line option, 218, 477
–output, -o
lrf2lrs command line option, 252, 509
lrs2lrf command line option, 253, 510
–output-format
ebook-convert-comic-input command line option, 236, 494
–output-profile
ebook-convert command line option, 230, 488
–override-profile-size
ebook-convert-pdf-output command line option, 245, 502
–page-breaks-before
ebook-convert command line option, 233, 491
–paper-size
ebook-convert-pdf-output command line option, 245, 502
–paragraph-type
ebook-convert-txt-input command line option, 240, 497
–password
calibre-server command line option, 217, 476
ebook-convert-recipe-input command line option, 239, 496
–password, -p
calibre-smtp command line option, 218, 477
–paths
calibre-debug command line option, 216, 474
–pdb-output-encoding
ebook-convert-pdb-output command line option, 244, 501
–pdf-default-font-size
ebook-convert-pdf-output command line option, 245, 502
–pdf-footer-template
ebook-convert-pdf-output command line option, 245, 502
–pdf-header-template
ebook-convert-pdf-output command line option, 245, 502
–pdf-mark-links

ebook-convert-pdf-output command line option, 245, 502
–pdf-mono-family
ebook-convert-pdf-output command line option, 245, 502
–pdf-mono-font-size
ebook-convert-pdf-output command line option, 245, 502
–pdf-page-numbers
ebook-convert-pdf-output command line option, 245, 502
–pdf-sans-family
ebook-convert-pdf-output command line option, 245, 502
–pdf-serif-family
ebook-convert-pdf-output command line option, 245, 502
–pdf-standard-font
ebook-convert-pdf-output command line option, 245, 502
–personal-doc
ebook-convert-mobi-output command line option, 244, 501
–pidfile
calibre-server command line option, 217, 476
–pml-output-encoding
ebook-convert-pml-output command line option, 246, 503
–port
calibre-smtp command line option, 218, 477
–port, -p
calibre-server command line option, 217, 476
–prefer-author-sort
ebook-convert-azw3-output command line option, 240, 498
ebook-convert-mobi-output command line option, 244, 501
–prefer-metadata-cover
ebook-convert command line option, 233, 491
–prefix
calibredb-list command line option, 220, 478
–preserve-cover-aspect-ratio
ebook-convert-epub-output command line option, 241, 498
ebook-convert-pdf-output command line option, 245, 502
–preserve-spaces
ebook-convert-txt-input command line option, 240, 497
–pretty-print
ebook-convert-azw3-output command line option, 240, 498
ebook-convert-epub-output command line option, 241, 498
ebook-convert-fb2-output command line option, 242, 499
ebook-convert-html-output command line option, 242, 499
ebook-convert-htmlz-output command line option, 242, 499
ebook-convert-lit-output command line option, 243, 500
ebook-convert-lrf-output command line option, 243, 500
ebook-convert-mobi-output command line option, 244, 501
ebook-convert-oeb-output command line option, 244, 501
ebook-convert-pdb-output command line option, 244, 501
ebook-convert-pdf-output command line option, 245, 502
ebook-convert-pml-output command line option, 246, 503
ebook-convert-rb-output command line option, 246, 503
ebook-convert-rtf-output command line option, 246, 503
ebook-convert-snb-output command line option, 246, 503
ebook-convert-tcr-output command line option, 247, 503
ebook-convert-txt-output command line option, 247, 504
ebook-convert-txtz-output command line option, 248, 505
–profile
lrfviewer command line option, 253, 509
–pubdate
ebook-convert command line option, 235, 492
–publisher
ebook-convert command line option, 235, 492
–publisher, -p
ebook-meta command line option, 249, 506
–py-console, -p
calibre-debug command line option, 216, 475
–quote, -q
calibredb-list_categories command line option, 227, 485
–raise-window
ebook-viewer command line option, 251, 508
–rating
ebook-convert command line option, 235, 492
–rating, -r
ebook-meta command line option, 249, 506
–really-do-it, -r
calibredb-restore_database command line option, 226, 484

–recurse, -r
calibredb-add command line option, 220, 479
–reinitialize-db
calibre-debug command line option, 216, 475
–relay, -r
calibre-smtp command line option, 218, 477
–remove-first-image
ebook-convert command line option, 233, 491
–remove-jacket
ebook-polish command line option, 250, 507
–remove-paragraph-spacing
ebook-convert command line option, 231, 489
–remove-paragraph-spacing-indent-size
ebook-convert command line option, 231, 489
–remove-plugin, -r
calibre-customize command line option, 215, 474
–render-tables-as-images
ebook-convert-lrf-output command line option, 243, 500
–replace-scene-breaks
ebook-convert command line option, 232, 490
–replace-whitespace
calibredb-export command line option, 223, 481
–report, -r
calibredb-check_library command line option, 226, 485
–restriction
calibre-server command line option, 217, 476
–right2left
ebook-convert-comic-input command line option, 236, 494
–sans-family
ebook-convert-lrf-output command line option, 243, 500
–search, -s
calibredb-catalog command line option, 223, 482
calibredb-list command line option, 220, 478
–search-replace
ebook-convert command line option, 232, 490
–sectionize
ebook-convert-fb2-output command line option, 242, 499
–separator
calibredb-list command line option, 220, 478
–separator, -s
calibredb-list_categories command line option, 227, 485
–series
ebook-convert command line option, 235, 492
–series, -s
calibredb-add command line option, 220, 479
ebook-meta command line option, 249, 506
–series-index
ebook-convert command line option, 235, 493
–series-index, -S
calibredb-add command line option, 220, 479
–serif-family
ebook-convert-lrf-output command line option, 243, 500
–share-not-sync
ebook-convert-azw3-output command line option, 241, 498
ebook-convert-mobi-output command line option, 244, 501
–show-gui-debug
calibre-debug command line option, 216, 475
–shutdown-running-calibre, -s
calibre command line option, 214, 473
calibre-debug command line option, 216, 475
–single-dir
calibredb-export command line option, 223, 481
–smarten-punctuation
ebook-convert command line option, 231, 489
–smarten-punctuation, -p
ebook-polish command line option, 250, 507
–snb-dont-indent-first-line
ebook-convert-snb-output command line option, 246, 503
–snb-full-screen
ebook-convert-snb-output command line option, 246, 503
–snb-hide-chapter-name
ebook-convert-snb-output command line option, 246, 503
–snb-insert-empty-line
ebook-convert-snb-output command line option, 246, 503
–snb-max-line-length
ebook-convert-snb-output command line option, 246, 503
–snb-output-encoding
ebook-convert-snb-output command line option, 247, 503
–sort-by
calibredb-catalog command line option, 223, 482
calibredb-list command line option, 220, 478
–sr1-replace
ebook-convert command line option, 233, 490
–sr1-search
ebook-convert command line option, 233, 490
–sr2-replace
ebook-convert command line option, 233, 491
–sr2-search
ebook-convert command line option, 233, 491
–sr3-replace
ebook-convert command line option, 233, 491
–sr3-search
ebook-convert command line option, 233, 491

–start-in-tray
calibre command line option, 214, 473
–start-reading-at
ebook-convert command line option, 233, 491
–subject, -s
calibre-smtp command line option, 218, 477
–subset-embedded-fonts
ebook-convert command line option, 232, 489
–subset-font, -f
calibre-debug command line option, 216, 475
–subset-fonts, -f
ebook-polish command line option, 250, 507
–tags
ebook-convert command line option, 235, 493
ebook-meta command line option, 249, 506
–tags, -T
calibredb-add command line option, 220, 479
–tcr-output-encoding
ebook-convert-tcr-output command line option, 247, 503
–template
calibredb-export command line option, 223, 481
–template-css
ebook-convert-html-output command line option, 242, 499
–template-html
ebook-convert-html-output command line option, 242, 499
–template-html-index
ebook-convert-html-output command line option, 242, 499
–test
ebook-convert-recipe-input command line option, 239, 496
–test-build
calibre-debug command line option, 216, 475
–text-size-multiplier-for-rendered-tables
ebook-convert-lrf-output command line option, 243, 500
–thread-pool
calibre-server command line option, 217, 476
–timefmt
calibredb-export command line option, 223, 481
–timeout, -d
fetch-ebook-metadata command line option, 252, 508
–timeout, -t
calibre-server command line option, 217, 476
calibre-smtp command line option, 218, 477
web2disk command line option, 254, 510
–timestamp
ebook-convert command line option, 235, 493
–title
ebook-convert command line option, 235, 493
–title, -t
calibredb-add command line option, 221, 479
ebook-meta command line option, 250, 506
fetch-ebook-metadata command line option, 252, 508
–title-sort
ebook-convert command line option, 235, 493
ebook-meta command line option, 250, 506
–to-dir
calibredb-export command line option, 223, 482
–to-lowercase
calibredb-export command line option, 223, 482
–to-opf
ebook-meta command line option, 250, 506
–toc-filter
ebook-convert command line option, 234, 492
–toc-threshold
ebook-convert command line option, 234, 492
–toc-title
ebook-convert-azw3-output command line option, 241, 498
ebook-convert-mobi-output command line option, 244, 501
–tweak-book, -t
calibre-debug command line option, 216, 475
–txt-in-remove-indents
ebook-convert-txt-input command line option, 240, 497
–txt-output-encoding
ebook-convert-txt-output command line option, 247, 504
ebook-convert-txtz-output command line option, 248, 505
–txt-output-formatting
ebook-convert-txt-output command line option, 247, 504
ebook-convert-txtz-output command line option, 248, 505
–uncompressed-pdf
ebook-convert-pdf-output command line option, 245, 502
–unit, -u
ebook-convert-pdf-output command line option, 245, 502
–unsmarten-punctuation
ebook-convert command line option, 232, 490
–unwrap-factor
ebook-convert-pdf-input command line option, 238, 496
–url-prefix
calibre-server command line option, 217, 476
–use-auto-toc
ebook-convert command line option, 234, 492
–use-djvutxt

ebook-convert-djvu-input command line option, 237, 494
–username
calibre-server command line option, 217, 476
ebook-convert-recipe-input command line option, 239, 497
–username, -u
calibre-smtp command line option, 218, 477
–verbose
ebook-polish command line option, 251, 507
lrf2lrs command line option, 252, 509
lrfviewer command line option, 253, 509
lrs2lrf command line option, 253, 510
web2disk command line option, 254, 510
–verbose, -v
calibre command line option, 214, 473
calibre-smtp command line option, 218, 477
calibredb-catalog command line option, 224, 482
ebook-convert command line option, 235, 493
fetch-ebook-metadata command line option, 252, 508
–version
calibre command line option, 214, 473
calibre-customize command line option, 215, 474
calibre-debug command line option, 216, 475
calibre-server command line option, 217, 476
calibre-smtp command line option, 218, 477
calibredb-add command line option, 221, 479
calibredb-add_custom_column command line option, 224, 483
calibredb-add_format command line option, 221, 480
calibredb-backup_metadata command line option, 227, 485
calibredb-catalog command line option, 224, 482
calibredb-check_library command line option, 226, 485
calibredb-custom_columns command line option, 225, 483
calibredb-export command line option, 223, 482
calibredb-list command line option, 220, 479
calibredb-list_categories command line option, 227, 485
calibredb-remove command line option, 221, 480
calibredb-remove_custom_column command line option, 225, 484
calibredb-remove_format command line option, 221, 480
calibredb-restore_database command line option, 226, 484
calibredb-saved_searches command line option, 224, 482
calibredb-set_custom command line option, 225, 484
calibredb-set_metadata command line option, 222, 481
calibredb-show_metadata command line option, 222, 480
ebook-convert command line option, 230, 488
ebook-meta command line option, 250, 506
ebook-polish command line option, 251, 507
ebook-viewer command line option, 251, 508
fetch-ebook-metadata command line option, 252, 508
lrf2lrs command line option, 252, 509
lrfviewer command line option, 253, 509
lrs2lrf command line option, 253, 510
web2disk command line option, 254, 510
–viewer, -w
calibre-debug command line option, 216, 475
–visual-debug
lrfviewer command line option, 253, 509
–white-background
lrfviewer command line option, 253, 509
–wide
ebook-convert-comic-input command line option, 236, 494
–width, -w
calibredb-list_categories command line option, 227, 485
–with-library
calibre command line option, 214, 473
calibre-server command line option, 217, 476
–wordspace
ebook-convert-lrf-output command line option, 243, 500

A

abort_recipe_processing() (calibre.web.feeds.news.BasicNewsRecipe method), 36, 102, 294, 360
accept_drag_move_event() (calibre.gui2.actions.InterfaceAction method), 200, 459
accept_enter_event() (calibre.gui2.actions.InterfaceAction method), 200, 459
accepts_drops (calibre.gui2.actions.InterfaceAction attribute), 200, 459
action_add_menu (calibre.gui2.actions.InterfaceAction attribute), 199, 458
action_menu_clone_qaction (calibre.gui2.actions.InterfaceAction attribute), 199, 458
action_spec (calibre.gui2.actions.InterfaceAction attribute), 199, 458
action_type (calibre.gui2.actions.InterfaceAction attribute), 200, 459

add_annotation_to_library() (calibre.devices.usbms.device.Device method), 198, 457
add_book() (calibre.devices.interface.BookList method), 197, 456
add_books_to_metadata() (calibre.devices.interface.DevicePlugin class method), 194, 453
add_toc_thumbnail() (calibre.web.feeds.news.BasicNewsRecipe method), 36, 102, 294, 360
adeify_images() (calibre.web.feeds.news.BasicNewsRecipe class method), 36, 102, 295, 360
all_field_keys() (calibre.ebooks.metadata.book.base.Metadata method), 136, 153, 395, 412
all_non_none_fields() (calibre.ebooks.metadata.book.base.Metadata method), 136, 153, 395, 412
API, **260**
articles_are_obfuscated (calibre.web.feeds.news.BasicNewsRecipe attribute), 40, 106, 298, 364
ASK_TO_ALLOW_CONNECT (calibre.devices.interface.DevicePlugin attribute), 191, 450
author (calibre.customize.Plugin attribute), 183, 441
auto_cleanup (calibre.web.feeds.news.BasicNewsRecipe attribute), 40, 106, 298, 364
auto_cleanup_keep (calibre.web.feeds.news.BasicNewsRecipe attribute), 40, 106, 298, 364
auto_repeat (calibre.gui2.actions.InterfaceAction attribute), 199, 458

B

BasicNewsRecipe (class in calibre.web.feeds.news), 36, 102, 294, 360
BCD (calibre.devices.interface.DevicePlugin attribute), 191, 450
BookList (class in calibre.devices.interface), 196, 455
books() (calibre.devices.interface.DevicePlugin method), 194, 453
BuiltinAdd (class in calibre.utils.formatter_functions), 128, 145, 387, 404
BuiltinAnd (class in calibre.utils.formatter_functions), 128, 145, 387, 404
BuiltinApproximateFormats (class in calibre.utils.formatter_functions), 129, 146, 388, 405
BuiltinAssign (class in calibre.utils.formatter_functions), 134, 151, 393, 410
BuiltinBooksize (class in calibre.utils.formatter_functions), 129, 146, 388, 405
BuiltinCapitalize (class in calibre.utils.formatter_functions), 135, 152, 394, 411
BuiltinCmp (class in calibre.utils.formatter_functions), 134, 151, 393, 410
BuiltinContains (class in calibre.utils.formatter_functions), 131, 148, 390, 407
BuiltinCount (class in calibre.utils.formatter_functions), 132, 149, 391, 408
BuiltinCurrentLibraryName (class in calibre.utils.formatter_functions), 130, 147, 389, 406
BuiltinCurrentLibraryPath (class in calibre.utils.formatter_functions), 130, 147, 389, 406
BuiltinDaysBetween (class in calibre.utils.formatter_functions), 128, 145, 387, 404
BuiltinDivide (class in calibre.utils.formatter_functions), 128, 145, 387, 404
BuiltinEval (class in calibre.utils.formatter_functions), 134, 151, 393, 410
BuiltinField (class in calibre.utils.formatter_functions), 130, 147, 389, 406
BuiltinFinishFormatting (class in calibre.utils.formatter_functions), 129, 146, 388, 405
BuiltinFirstNonEmpty (class in calibre.utils.formatter_functions), 131, 148, 390, 407
BuiltinFormatDate (class in calibre.utils.formatter_functions), 129, 146, 388, 405
BuiltinFormatNumber (class in calibre.utils.formatter_functions), 129, 146, 388, 405
BuiltinFormatsModtimes (class in calibre.utils.formatter_functions), 130, 147, 389, 406
BuiltinFormatsPaths (class in calibre.utils.formatter_functions), 130, 147, 389, 406
BuiltinFormatsSizes (class in calibre.utils.formatter_functions), 130, 147, 389, 406
BuiltinHasCover (class in calibre.utils.formatter_functions), 130, 147, 389, 406
BuiltinHumanReadable (class in calibre.utils.formatter_functions), 129, 146, 388, 405
BuiltinIdentifierInList (class in calibre.utils.formatter_functions), 132, 149,

391, 408
BuiltinIfempty (class in calibre.utils.formatter_functions), 131, 148, 390, 407
BuiltinInList (class in calibre.utils.formatter_functions), 132, 149, 391, 408
BuiltinLanguageCodes (class in calibre.utils.formatter_functions), 130, 147, 389, 406
BuiltinLanguageStrings (class in calibre.utils.formatter_functions), 130, 147, 389, 406
BuiltinListDifference (class in calibre.utils.formatter_functions), 132, 149, 391, 408
BuiltinListEquals (class in calibre.utils.formatter_functions), 133, 150, 392, 409
BuiltinListIntersection (class in calibre.utils.formatter_functions), 133, 150, 392, 409
BuiltinListitem (class in calibre.utils.formatter_functions), 132, 149, 391, 408
BuiltinListRe (class in calibre.utils.formatter_functions), 133, 150, 392, 409
BuiltinListSort (class in calibre.utils.formatter_functions), 133, 150, 392, 409
BuiltinListUnion (class in calibre.utils.formatter_functions), 133, 150, 392, 409
BuiltinLookup (class in calibre.utils.formatter_functions), 131, 148, 390, 407
BuiltinLowercase (class in calibre.utils.formatter_functions), 135, 152, 394, 411
BuiltinMultiply (class in calibre.utils.formatter_functions), 128, 145, 387, 404
BuiltinNot (class in calibre.utils.formatter_functions), 128, 145, 387, 404
BuiltinOndevice (class in calibre.utils.formatter_functions), 131, 148, 390, 407
BuiltinOr (class in calibre.utils.formatter_functions), 128, 145, 387, 404
BuiltinPrint (class in calibre.utils.formatter_functions), 134, 151, 393, 410
BuiltinRawField (class in calibre.utils.formatter_functions), 131, 148, 390, 407
BuiltinRe (class in calibre.utils.formatter_functions), 135, 152, 394, 411
BuiltinSelect (class in calibre.utils.formatter_functions), 132, 149, 391, 408
BuiltinSeriesSort (class in calibre.utils.formatter_functions), 131, 148, 390, 407
BuiltinShorten (class in calibre.utils.formatter_functions), 135, 152, 394, 411
BuiltinStrcat (class in calibre.utils.formatter_functions), 135, 152, 394, 411
BuiltinStrcatMax (class in calibre.utils.formatter_functions), 135, 152, 394, 411
BuiltinStrcmp (class in calibre.utils.formatter_functions), 134, 151, 393, 410
BuiltinStrInList (class in calibre.utils.formatter_functions), 132, 149, 391, 408
BuiltinStrlen (class in calibre.utils.formatter_functions), 135, 152, 394, 411
BuiltinSubitems (class in calibre.utils.formatter_functions), 133, 150, 392, 409
BuiltinSublist (class in calibre.utils.formatter_functions), 133, 150, 392, 409
BuiltinSubstr (class in calibre.utils.formatter_functions), 136, 153, 395, 412
BuiltinSubtract (class in calibre.utils.formatter_functions), 128, 145, 387, 404
BuiltinSwapAroundComma (class in calibre.utils.formatter_functions), 136, 153, 395, 412
BuiltinSwitch (class in calibre.utils.formatter_functions), 131, 148, 390, 407
BuiltinTemplate (class in calibre.utils.formatter_functions), 134, 151, 393, 410
BuiltinTest (class in calibre.utils.formatter_functions), 131, 148, 390, 407
BuiltinTitlecase (class in calibre.utils.formatter_functions), 135, 152, 394, 411
BuiltinToday (class in calibre.utils.formatter_functions), 128, 145, 387, 404
BuiltinUppercase (class in calibre.utils.formatter_functions), 135, 152, 394, 411

C

cached_cover_url_is_reliable (calibre.ebooks.metadata.sources.base.Source attribute), 186, 445
calibre command line option
–help, -h, 214, 473

–ignore-plugins, 214, 473
–no-update-check, 214, 473
–shutdown-running-calibre, -s, 214, 473
–start-in-tray, 214, 473
–verbose, -v, 214, 473
–version, 214, 473
–with-library, 214, 473
calibre-customize command line option
–add-plugin, -a, 214, 473
–build-plugin, -b, 214, 473
–customize-plugin, 214, 473
–disable-plugin, 214, 473
–enable-plugin, 215, 473
–help, -h, 215, 474
–list-plugins, -l, 215, 474
–remove-plugin, -r, 215, 474
–version, 215, 474
calibre-debug command line option
–add-simple-plugin, 215, 474
–command, -c, 215, 474
–debug-device-driver, -d, 215, 474
–exec-file, -e, 215, 474
–gui, -g, 215, 474
–gui-debug, 215, 474
–help, -h, 215, 474
–inspect-mobi, -m, 215, 474
–paths, 216, 474
–py-console, -p, 216, 475
–reinitialize-db, 216, 475
–show-gui-debug, 216, 475
–shutdown-running-calibre, -s, 216, 475
–subset-font, -f, 216, 475
–test-build, 216, 475
–tweak-book, -t, 216, 475
–version, 216, 475
–viewer, -w, 216, 475
calibre-server command line option
–auto-reload, 216, 475
–daemonize, 216, 475
–develop, 216, 475
–help, -h, 217, 475
–max-cover, 217, 475
–max-opds-items, 217, 475
–max-opds-ungrouped-items, 217, 476
–password, 217, 476
–pidfile, 217, 476
–port, -p, 217, 476
–restriction, 217, 476
–thread-pool, 217, 476
–timeout, -t, 217, 476
–url-prefix, 217, 476
–username, 217, 476
–version, 217, 476
–with-library, 217, 476
calibre-smtp command line option
–attachment, -a, 218, 477
–encryption-method, -e, 218, 477
–fork, -f, 218, 476
–help, -h, 218, 476
–localhost, -l, 218, 476
–outbox, -o, 218, 477
–password, -p, 218, 477
–port, 218, 477
–relay, -r, 218, 477
–subject, -s, 218, 477
–timeout, -t, 218, 477
–username, -u, 218, 477
–verbose, -v, 218, 477
–version, 218, 477
calibre.customize (module), 182, 441
calibre.customize.conversion (module), 188, 447
calibre.devices.interface (module), 190, 449
calibre.ebooks.metadata.book.base (module), 136, 153, 395, 412
calibre.ebooks.metadata.sources.base (module), 186, 445
calibre.utils.formatter_functions (module), 128, 145, 387, 404
calibre.web.feeds.news (module), 36, 102, 294, 360
calibredb-add command line option
–authors, -a, 220, 479
–cover, -c, 220, 479
–duplicates, -d, 220, 479
–empty, -e, 220, 479
–help, -h, 220, 479
–isbn, -i, 220, 479
–one-book-per-directory, -1, 220, 479
–recurse, -r, 220, 479
–series, -s, 220, 479
–series-index, -S, 220, 479
–tags, -T, 220, 479
–title, -t, 221, 479
–version, 221, 479
calibredb-add_custom_column command line option
–display, 224, 483
–help, -h, 224, 483
–is-multiple, 224, 483
–version, 224, 483
calibredb-add_format command line option
–help, -h, 221, 480
–version, 221, 480
calibredb-backup_metadata command line option
–all, 227, 485
–help, -h, 227, 485
–version, 227, 485
calibredb-catalog command line option
–fields, 223, 482
–help, -h, 223, 482
–ids, -i, 223, 482

–search, -s, 223, 482
–sort-by, 223, 482
–verbose, -v, 224, 482
–version, 224, 482
calibredb-check_library command line option
–csv, -c, 226, 484
–help, -h, 226, 484
–ignore_extensions, -e, 226, 484
–ignore_names, -n, 226, 485
–report, -r, 226, 485
–version, 226, 485
calibredb-custom_columns command line option
–details, -d, 225, 483
–help, -h, 225, 483
–version, 225, 483
calibredb-export command line option
–all, 222, 481
–dont-asciiize, 222, 481
–dont-save-cover, 222, 481
–dont-update-metadata, 222, 481
–dont-write-opf, 222, 481
–formats, 223, 481
–help, -h, 223, 481
–replace-whitespace, 223, 481
–single-dir, 223, 481
–template, 223, 481
–timefmt, 223, 481
–to-dir, 223, 482
–to-lowercase, 223, 482
–version, 223, 482
calibredb-list command line option
–ascending, 219, 478
–fields, -f, 219, 478
–help, -h, 219, 478
–line-width, -w, 219, 478
–prefix, 220, 478
–search, -s, 220, 478
–separator, 220, 478
–sort-by, 220, 478
–version, 220, 479
calibredb-list_categories command line option
–categories, -r, 226, 485
–csv, -c, 226, 485
–help, -h, 226, 485
–item_count, -i, 227, 485
–quote, -q, 227, 485
–separator, -s, 227, 485
–version, 227, 485
–width, -w, 227, 485
calibredb-remove command line option
–help, -h, 221, 480
–version, 221, 480
calibredb-remove_custom_column command line option
–force, -f, 225, 483
–help, -h, 225, 483
–version, 225, 484
calibredb-remove_format command line option
–help, -h, 221, 480
–version, 221, 480
calibredb-restore_database command line option
–help, -h, 226, 484
–really-do-it, -r, 226, 484
–version, 226, 484
calibredb-saved_searches command line option
–help, -h, 224, 482
–version, 224, 482
calibredb-set_custom command line option
–append, -a, 225, 484
–help, -h, 225, 484
–version, 225, 484
calibredb-set_metadata command line option
–help, -h, 222, 481
–version, 222, 481
calibredb-show_metadata command line option
–as-opf, 222, 480
–help, -h, 222, 480
–version, 222, 480
can_be_disabled (calibre.customize.Plugin attribute), 183, 442
CAN_DO_DEVICE_DB_PLUGBOARD (calibre.devices.interface.DevicePlugin attribute), 191, 450
can_handle() (calibre.devices.interface.DevicePlugin method), 192, 451
can_handle_windows() (calibre.devices.interface.DevicePlugin method), 192, 451
CAN_SET_METADATA (calibre.devices.interface.DevicePlugin attribute), 191, 450
capabilities (calibre.ebooks.metadata.sources.base.Source attribute), 186, 445
card_prefix() (calibre.devices.interface.DevicePlugin method), 193, 452
CatalogPlugin (class in calibre.customize), 185, 444
category (calibre.customize.PreferencesPlugin attribute), 202, 461
category_order (calibre.customize.PreferencesPlugin attribute), 201, 460
center_navbar (calibre.web.feeds.news.BasicNewsRecipe attribute), 41, 106, 299, 364
changed_signal (calibre.gui2.preferences.ConfigWidgetInterface attribute), 202, 461
clean_downloaded_metadata() (calibre.ebooks.metadata.sources.base.Source method), 186, 445
cleanup() (calibre.web.feeds.news.BasicNewsRecipe method), 37, 102, 295, 360

CLI (class in calibre.devices.usbms.cli), 198, 457
cli_options (calibre.customize.CatalogPlugin attribute), 185, 444
clone_browser() (calibre.web.feeds.news.BasicNewsRecipe method), 37, 102, 295, 360
command line option
  –dont-notify-gui, 219, 478
  –library-path, 219, 478
commit() (calibre.gui2.preferences.ConfigWidgetInterface method), 202, 461
common_options (calibre.customize.conversion.InputFormatPlugin attribute), 188, 447
common_options (calibre.customize.conversion.OutputFormatPlugin attribute), 189, 448
config_help_message (calibre.ebooks.metadata.sources.base.Source attribute), 186, 445
config_widget (calibre.customize.PreferencesPlugin attribute), 201, 460
config_widget() (calibre.customize.Plugin method), 183, 442
config_widget() (calibre.devices.interface.DevicePlugin class method), 195, 454
ConfigWidgetBase (class in calibre.gui2.preferences), 203, 462
ConfigWidgetInterface (class in calibre.gui2.preferences), 202, 461
contains(), **116**, **375**
conversion_options (calibre.web.feeds.news.BasicNewsRecipe attribute), 41, 106, 299, 364
convert() (calibre.customize.conversion.InputFormatPlugin method), 189, 448
convert() (calibre.customize.conversion.OutputFormatPlugin method), 190, 449
core_usage (calibre.customize.conversion.InputFormatPlugin attribute), 188, 447
cover_margins (calibre.web.feeds.news.BasicNewsRecipe attribute), 41, 106, 299, 365
create_menu_action() (calibre.gui2.actions.InterfaceAction method), 200, 459
create_widget() (calibre.customize.PreferencesPlugin method), 202, 461
CSS, **260**
custom_field_keys() (calibre.ebooks.metadata.book.base.Metadata method), 136, 153, 395, 412
customization_help() (calibre.customize.Plugin method), 184, 442

D

debug_managed_device_detection() (calibre.devices.interface.DevicePlugin method), 192, 451
default_cover() (calibre.web.feeds.news.BasicNewsRecipe method), 37, 102, 295, 361
delay (calibre.web.feeds.news.BasicNewsRecipe attribute), 41, 106, 299, 365
delete_books() (calibre.devices.interface.DevicePlugin method), 194, 453
description (calibre.customize.Plugin attribute), 183, 441
description (calibre.customize.PreferencesPlugin attribute), 202, 461
description (calibre.web.feeds.news.BasicNewsRecipe attribute), 41, 106, 299, 365
detect_managed_devices() (calibre.devices.interface.DevicePlugin method), 192, 451
Device (class in calibre.devices.usbms.device), 197, 456
DevicePlugin (class in calibre.devices.interface), 190, 449
do_user_config() (calibre.customize.Plugin method), 183, 442
dont_add_to (calibre.gui2.actions.InterfaceAction attribute), 199, 459
dont_remove_from (calibre.gui2.actions.InterfaceAction attribute), 200, 459
download() (calibre.web.feeds.news.BasicNewsRecipe method), 37, 102, 295, 361
download_cover() (calibre.ebooks.metadata.sources.base.Source method), 188, 447
drop_event() (calibre.gui2.actions.InterfaceAction method), 200, 459

E

ebook-convert command line option
  –asciiize, 230, 488
  –author-sort, 234, 492
  –authors, 234, 492
  –base-font-size, 230, 488
  –book-producer, 234, 492
  –change-justification, 230, 488
  –chapter, 233, 491
  –chapter-mark, 233, 491
  –comments, 234, 492
  –cover, 234, 492
  –debug-pipeline, -d, 235, 493
  –disable-dehyphenate, 232, 490
  –disable-delete-blank-paragraphs, 232, 490
  –disable-fix-indents, 232, 490
  –disable-font-rescaling, 230, 488
  –disable-format-scene-breaks, 232, 490
  –disable-italicize-common-cases, 232, 490
  –disable-markup-chapter-headings, 232, 490

–disable-remove-fake-margins, 233, 491
–disable-renumber-headings, 232, 490
–disable-unwrap-lines, 232, 490
–duplicate-links-in-toc, 234, 491
–embed-font-family, 230, 488
–enable-heuristics, 232, 490
–extra-css, 230, 488
–filter-css, 230, 488
–font-size-mapping, 230, 488
–help, -h, 229, 487
–html-unwrap-factor, 232, 490
–input-profile, 229, 487
–insert-blank-line, 231, 488
–insert-blank-line-size, 231, 489
–insert-metadata, 233, 491
–isbn, 235, 492
–keep-ligatures, 231, 489
–language, 235, 492
–level1-toc, 234, 491
–level2-toc, 234, 492
–level3-toc, 234, 492
–line-height, 231, 489
–linearize-tables, 231, 489
–list-recipes, 230, 488
–margin-bottom, 231, 489
–margin-left, 231, 489
–margin-right, 231, 489
–margin-top, 231, 489
–max-toc-links, 234, 492
–minimum-line-height, 231, 489
–no-chapters-in-toc, 234, 492
–output-profile, 230, 488
–page-breaks-before, 233, 491
–prefer-metadata-cover, 233, 491
–pubdate, 235, 492
–publisher, 235, 492
–rating, 235, 492
–remove-first-image, 233, 491
–remove-paragraph-spacing, 231, 489
–remove-paragraph-spacing-indent-size, 231, 489
–replace-scene-breaks, 232, 490
–search-replace, 232, 490
–series, 235, 492
–series-index, 235, 493
–smarten-punctuation, 231, 489
–sr1-replace, 233, 490
–sr1-search, 233, 490
–sr2-replace, 233, 491
–sr2-search, 233, 491
–sr3-replace, 233, 491
–sr3-search, 233, 491
–start-reading-at, 233, 491
–subset-embedded-fonts, 232, 489
–tags, 235, 493
–timestamp, 235, 493
–title, 235, 493
–title-sort, 235, 493
–toc-filter, 234, 492
–toc-threshold, 234, 492
–unsmarten-punctuation, 232, 490
–use-auto-toc, 234, 492
–verbose, -v, 235, 493
–version, 230, 488

ebook-convert-azw3-output command line option
–dont-compress, 240, 497
–extract-to, 240, 497
–mobi-toc-at-start, 240, 497
–no-inline-toc, 240, 498
–prefer-author-sort, 240, 498
–pretty-print, 240, 498
–share-not-sync, 241, 498
–toc-title, 241, 498

ebook-convert-azw4-input command line option
–input-encoding, 235, 493

ebook-convert-chm-input command line option
–input-encoding, 235, 493

ebook-convert-comic-input command line option
–colors, 236, 493
–comic-image-size, 236, 493
–despeckle, 236, 493
–disable-trim, 236, 493
–dont-add-comic-pages-to-toc, 236, 493
–dont-grayscale, 236, 493
–dont-normalize, 236, 494
–dont-sharpen, 236, 494
–input-encoding, 236, 494
–keep-aspect-ratio, 236, 494
–landscape, 236, 494
–no-process, 236, 494
–no-sort, 236, 494
–output-format, 236, 494
–right2left, 236, 494
–wide, 236, 494

ebook-convert-djvu-input command line option
–input-encoding, 237, 494
–use-djvutxt, 237, 494

ebook-convert-epub-input command line option
–input-encoding, 237, 494

ebook-convert-epub-output command line option
–dont-split-on-page-breaks, 241, 498
–epub-flatten, 241, 498
–extract-to, 241, 498
–flow-size, 241, 498
–no-default-epub-cover, 241, 498
–no-svg-cover, 241, 498
–preserve-cover-aspect-ratio, 241, 498
–pretty-print, 241, 498

ebook-convert-fb2-input command line option

–input-encoding, 237, 494
–no-inline-fb2-toc, 237, 495
ebook-convert-fb2-output command line option
–fb2-genre, 241, 499
–pretty-print, 242, 499
–sectionize, 242, 499
ebook-convert-htlz-input command line option
–input-encoding, 237, 495
ebook-convert-html-input command line option
–breadth-first, 237, 495
–dont-package, 237, 495
–input-encoding, 237, 495
–max-levels, 237, 495
ebook-convert-html-output command line option
–extract-to, 242, 499
–pretty-print, 242, 499
–template-css, 242, 499
–template-html, 242, 499
–template-html-index, 242, 499
ebook-convert-htmlz-output command line option
–htmlz-class-style, 242, 499
–htmlz-css-type, 242, 499
–htmlz-title-filename, 242, 499
–pretty-print, 242, 499
ebook-convert-lit-input command line option
–input-encoding, 238, 495
ebook-convert-lit-output command line option
–pretty-print, 243, 500
ebook-convert-lrf-input command line option
–input-encoding, 238, 495
ebook-convert-lrf-output command line option
–enable-autorotation, 243, 500
–header, 243, 500
–header-format, 243, 500
–header-separation, 243, 500
–minimum-indent, 243, 500
–mono-family, 243, 500
–pretty-print, 243, 500
–render-tables-as-images, 243, 500
–sans-family, 243, 500
–serif-family, 243, 500
–text-size-multiplier-for-rendered-tables, 243, 500
–wordspace, 243, 500
ebook-convert-mobi-input command line option
–input-encoding, 238, 495
ebook-convert-mobi-output command line option
–dont-compress, 243, 500
–extract-to, 243, 500
–mobi-file-type, 243, 500
–mobi-ignore-margins, 243, 500
–mobi-keep-original-images, 244, 501
–mobi-toc-at-start, 244, 501
–no-inline-toc, 244, 501
–personal-doc, 244, 501
–prefer-author-sort, 244, 501
–pretty-print, 244, 501
–share-not-sync, 244, 501
–toc-title, 244, 501
ebook-convert-odt-input command line option
–input-encoding, 238, 495
ebook-convert-oeb-output command line option
–pretty-print, 244, 501
ebook-convert-pdb-input command line option
–input-encoding, 238, 496
ebook-convert-pdb-output command line option
–format, -f, 244, 501
–inline-toc, 244, 501
–pdb-output-encoding, 244, 501
–pretty-print, 244, 501
ebook-convert-pdf-input command line option
–input-encoding, 238, 496
–new-pdf-engine, 238, 496
–no-images, 238, 496
–unwrap-factor, 238, 496
ebook-convert-pdf-output command line option
–custom-size, 245, 501
–old-pdf-engine, 245, 502
–override-profile-size, 245, 502
–paper-size, 245, 502
–pdf-default-font-size, 245, 502
–pdf-footer-template, 245, 502
–pdf-header-template, 245, 502
–pdf-mark-links, 245, 502
–pdf-mono-family, 245, 502
–pdf-mono-font-size, 245, 502
–pdf-page-numbers, 245, 502
–pdf-sans-family, 245, 502
–pdf-serif-family, 245, 502
–pdf-standard-font, 245, 502
–preserve-cover-aspect-ratio, 245, 502
–pretty-print, 245, 502
–uncompressed-pdf, 245, 502
–unit, -u, 245, 502
ebook-convert-pml-input command line option
–input-encoding, 239, 496
ebook-convert-pml-output command line option
–full-image-depth, 246, 502
–inline-toc, 246, 503
–pml-output-encoding, 246, 503
–pretty-print, 246, 503
ebook-convert-rb-input command line option
–input-encoding, 239, 496
ebook-convert-rb-output command line option
–inline-toc, 246, 503
–pretty-print, 246, 503
ebook-convert-recipe-input command line option
–dont-download-recipe, 239, 496
–input-encoding, 239, 496

–lrf, 239, 496
–password, 239, 496
–test, 239, 496
–username, 239, 497
ebook-convert-rtf-input command line option
–input-encoding, 239, 496
ebook-convert-rtf-output command line option
–pretty-print, 246, 503
ebook-convert-snb-input command line option
–input-encoding, 239, 497
ebook-convert-snb-output command line option
–pretty-print, 246, 503
–snb-dont-indent-first-line, 246, 503
–snb-full-screen, 246, 503
–snb-hide-chapter-name, 246, 503
–snb-insert-empty-line, 246, 503
–snb-max-line-length, 246, 503
–snb-output-encoding, 247, 503
ebook-convert-tcr-input command line option
–input-encoding, 240, 497
ebook-convert-tcr-output command line option
–pretty-print, 247, 503
–tcr-output-encoding, 247, 503
ebook-convert-txt-input command line option
–formatting-type, 240, 497
–input-encoding, 240, 497
–markdown-disable-toc, 240, 497
–paragraph-type, 240, 497
–preserve-spaces, 240, 497
–txt-in-remove-indents, 240, 497
ebook-convert-txt-output command line option
–force-max-line-length, 247, 504
–inline-toc, 247, 504
–keep-color, 247, 504
–keep-image-references, 247, 504
–keep-links, 247, 504
–max-line-length, 247, 504
–newline, -n, 247, 504
–pretty-print, 247, 504
–txt-output-encoding, 247, 504
–txt-output-formatting, 247, 504
ebook-convert-txtz-output command line option
–force-max-line-length, 248, 504
–inline-toc, 248, 504
–keep-color, 248, 504
–keep-image-references, 248, 504
–keep-links, 248, 504
–max-line-length, 248, 505
–newline, -n, 248, 505
–pretty-print, 248, 505
–txt-output-encoding, 248, 505
–txt-output-formatting, 248, 505
ebook-meta command line option
–author-sort, 249, 505
–authors, -a, 249, 505
–book-producer, -k, 249, 505
–category, 249, 505
–comments, -c, 249, 505
–cover, 249, 505
–date, -d, 249, 506
–from-opf, 249, 506
–get-cover, 249, 506
–help, -h, 249, 506
–index, -i, 249, 506
–isbn, 249, 506
–language, -l, 249, 506
–lrf-bookid, 249, 506
–publisher, -p, 249, 506
–rating, -r, 249, 506
–series, -s, 249, 506
–tags, 249, 506
–title, -t, 250, 506
–title-sort, 250, 506
–to-opf, 250, 506
–version, 250, 506
ebook-polish command line option
–cover, -c, 250, 507
–help, -h, 250, 507
–jacket, -j, 250, 507
–opf, -o, 250, 507
–remove-jacket, 250, 507
–smarten-punctuation, -p, 250, 507
–subset-fonts, -f, 250, 507
–verbose, 251, 507
–version, 251, 507
ebook-viewer command line option
–debug-javascript, 251, 507
–full-screen, -f, 251, 507
–help, -h, 251, 507
–open-at, 251, 508
–raise-window, 251, 508
–version, 251, 508
eject() (calibre.devices.interface.DevicePlugin method), 193, 452
encoding (calibre.web.feeds.news.BasicNewsRecipe attribute), 41, 107, 299, 365
extra_css (calibre.web.feeds.news.BasicNewsRecipe attribute), 41, 107, 299, 365
extract_readable_article() (calibre.web.feeds.news.BasicNewsRecipe method), 37, 102, 295, 361

F

feeds (calibre.web.feeds.news.BasicNewsRecipe attribute), 41, 107, 299, 365
fetch-ebook-metadata command line option
–authors, -a, 251, 508
–cover, -c, 251, 508

–help, -h, 252, 508
–isbn, -i, 252, 508
–opf, -o, 252, 508
–timeout, -d, 252, 508
–title, -t, 252, 508
–verbose, -v, 252, 508
–version, 252, 508
file_type (calibre.customize.conversion.OutputFormatPlugin attribute), 189, 448
file_types (calibre.customize.CatalogPlugin attribute), 185, 444
file_types (calibre.customize.conversion.InputFormatPlugin attribute), 188, 447
file_types (calibre.customize.FileTypePlugin attribute), 184, 443
file_types (calibre.customize.MetadataReaderPlugin attribute), 185, 444
file_types (calibre.customize.MetadataWriterPlugin attribute), 185, 444
filename_callback() (calibre.devices.usbms.device.Device method), 198, 457
FileTypePlugin (class in calibre.customize), 184, 443
filter_regexps (calibre.web.feeds.news.BasicNewsRecipe attribute), 41, 107, 299, 365
for_viewer (calibre.customize.conversion.InputFormatPlugin attribute), 188, 447
format_field() (calibre.ebooks.metadata.book.base.Metadata method), 137, 154, 396, 413
FORMATS (calibre.devices.interface.DevicePlugin attribute), 190, 449
free_space() (calibre.devices.interface.DevicePlugin method), 193, 452

G

genesis() (calibre.gui2.actions.InterfaceAction method), 201, 460
genesis() (calibre.gui2.preferences.ConfigWidgetInterface method), 202, 461
get_all_standard_metadata() (calibre.ebooks.metadata.book.base.Metadata method), 137, 154, 396, 413
get_all_user_metadata() (calibre.ebooks.metadata.book.base.Metadata method), 137, 154, 396, 413
get_annotations() (calibre.devices.usbms.device.Device method), 198, 457
get_article_url() (calibre.web.feeds.news.BasicNewsRecipe method), 37, 102, 295, 361
get_author_tokens() (calibre.ebooks.metadata.sources.base.Source method), 186, 445
get_book_url() (calibre.ebooks.metadata.sources.base.Source method), 186, 445
get_book_url_name() (calibre.ebooks.metadata.sources.base.Source method), 187, 446
get_browser() (calibre.web.feeds.news.BasicNewsRecipe method), 37, 102, 295, 361
get_cached_cover_url() (calibre.ebooks.metadata.sources.base.Source method), 187, 446
get_collections() (calibre.devices.interface.BookList method), 197, 456
get_cover_url() (calibre.web.feeds.news.BasicNewsRecipe method), 37, 103, 295, 361
get_device_information() (calibre.devices.interface.DevicePlugin method), 193, 452
get_device_uid() (calibre.devices.interface.DevicePlugin method), 195, 454
get_driveinfo() (calibre.devices.interface.DevicePlugin method), 193, 452
get_feeds() (calibre.web.feeds.news.BasicNewsRecipe method), 37, 103, 296, 361
get_file() (calibre.devices.interface.DevicePlugin method), 195, 454
get_identifiers() (calibre.ebooks.metadata.book.base.Metadata method), 136, 153, 395, 412
get_images() (calibre.customize.conversion.InputFormatPlugin method), 189, 448
get_masthead_title() (calibre.web.feeds.news.BasicNewsRecipe method), 38, 103, 296, 361
get_masthead_url() (calibre.web.feeds.news.BasicNewsRecipe method), 38, 103, 296, 362
get_metadata() (calibre.customize.MetadataReaderPlugin method), 185, 444
get_obfuscated_article() (calibre.web.feeds.news.BasicNewsRecipe method), 38, 103, 296, 362
get_option() (calibre.devices.interface.DevicePlugin method), 196, 455
get_standard_metadata() (calibre.ebooks.metadata.book.base.Metadata method), 137, 154, 396, 413
get_title_tokens() (calibre.ebooks.metadata.sources.base.Source method), 186, 445
get_user_blacklisted_devices() (calibre.devices.interface.DevicePlugin method), 196, 455
get_user_metadata() (calibre.ebooks.metadata.book.base.Metadata method), 137, 154, 396, 413
gui_category (calibre.customize.PreferencesPlugin attribute), 202, 461

gui_configuration_widget() (calibre.customize.conversion.InputFormatPlugin method), 189, 448
gui_configuration_widget() (calibre.customize.conversion.OutputFormatPlugin method), 190, 449
gui_layout_complete() (calibre.gui2.actions.InterfaceAction method), 201, 460
gui_name (calibre.customize.PreferencesPlugin attribute), 202, 461

H

has_html_comments (calibre.ebooks.metadata.sources.base.Source attribute), 186, 445
HTML, **260**

I

icon (calibre.customize.PreferencesPlugin attribute), 202, 461
icon (calibre.devices.interface.DevicePlugin attribute), 191, 450
identify() (calibre.ebooks.metadata.sources.base.Source method), 187, 446
identify_results_keygen() (calibre.ebooks.metadata.sources.base.Source method), 187, 446
ignore_connected_device() (calibre.devices.interface.DevicePlugin method), 195, 454
ignore_duplicate_articles (calibre.web.feeds.news.BasicNewsRecipe attribute), 42, 107, 300, 365
image_url_processor() (calibre.web.feeds.news.BasicNewsRecipe class method), 38, 103, 296, 362
index_to_soup() (calibre.web.feeds.news.BasicNewsRecipe method), 38, 103, 296, 362
initialization_complete() (calibre.gui2.actions.InterfaceAction method), 201, 460
initialize() (calibre.customize.CatalogPlugin method), 185, 444
initialize() (calibre.customize.Plugin method), 183, 442
initialize() (calibre.gui2.preferences.ConfigWidgetInterface method), 202, 461
InputFormatPlugin (class in calibre.customize.conversion), 188, 447
InterfaceAction (class in calibre.gui2.actions), 199, 458
InterfaceActionBase (class in calibre.customize), 201, 460
InternalMetadataCompareKeyGen (class in calibre.ebooks.metadata.sources.base), 188, 447
is_configured() (calibre.ebooks.metadata.sources.base.Source method), 186, 445
is_dynamically_controllable() (calibre.devices.interface.DevicePlugin method), 196, 455
is_image_collection (calibre.customize.conversion.InputFormatPlugin attribute), 188, 447
is_link_wanted() (calibre.web.feeds.news.BasicNewsRecipe method), 38, 103, 296, 362
is_null() (calibre.ebooks.metadata.book.base.Metadata method), 136, 153, 395, 412
is_running() (calibre.devices.interface.DevicePlugin method), 196, 455
is_usb_connected() (calibre.devices.interface.DevicePlugin method), 191, 450

J

javascript_login() (calibre.web.feeds.news.BasicNewsRecipe method), 38, 104, 296, 362

K

keep_only_tags (calibre.web.feeds.news.BasicNewsRecipe attribute), 42, 107, 300, 366

L

language (calibre.web.feeds.news.BasicNewsRecipe attribute), 42, 107, 300, 366
library_changed() (calibre.gui2.actions.InterfaceAction method), 201, 460
load_actual_plugin() (calibre.customize.InterfaceActionBase method), 201, 460
load_resources() (calibre.customize.Plugin method), 183, 442
load_resources() (calibre.gui2.actions.InterfaceAction method), 200, 459
location_selected() (calibre.gui2.actions.InterfaceAction method), 201, 460
LRF, **260**
lrf2lrs command line option
  –dont-output-resources, 252, 509
  –help, -h, 252, 509
  –output, -o, 252, 509
  –verbose, 252, 509
  –version, 252, 509
lrfviewer command line option
  –disable-hyphenation, 253, 509
  –help, -h, 253, 509
  –profile, 253, 509
  –verbose, 253, 509

–version, 253, 509
–visual-debug, 253, 509
–white-background, 253, 509
lrs2lrf command line option
–help, -h, 253, 509
–lrs, 253, 510
–output, -o, 253, 510
–verbose, 253, 510
–version, 253, 510

M

MANAGES_DEVICE_PRESENCE (calibre.devices.interface.DevicePlugin attribute), 191, 450
masthead_url (calibre.web.feeds.news.BasicNewsRecipe attribute), 42, 107, 300, 366
match_regexps (calibre.web.feeds.news.BasicNewsRecipe attribute), 42, 107, 300, 366
max_articles_per_feed (calibre.web.feeds.news.BasicNewsRecipe attribute), 42, 107, 300, 366
MAX_PATH_LEN (calibre.devices.usbms.device.Device attribute), 198, 457
Metadata (class in calibre.ebooks.metadata.book.base), 136, 153, 395, 412
metadata_for_field() (calibre.ebooks.metadata.book.base.Metadata method), 136, 153, 395, 412
MetadataReaderPlugin (class in calibre.customize), 185, 444
MetadataWriterPlugin (class in calibre.customize), 185, 444
minimum_calibre_version (calibre.customize.Plugin attribute), 183, 442

N

name (calibre.customize.Plugin attribute), 182, 441
name (calibre.gui2.actions.InterfaceAction attribute), 199, 458
name(), **116**, **375**
name_order (calibre.customize.PreferencesPlugin attribute), 202, 461
needs_subscription (calibre.web.feeds.news.BasicNewsRecipe attribute), 42, 108, 300, 366
NEWS_IN_FOLDER (calibre.devices.usbms.device.Device attribute), 198, 457
no_stylesheets (calibre.web.feeds.news.BasicNewsRecipe attribute), 42, 108, 300, 366
normalize_path() (calibre.devices.usbms.driver.USBMS class method), 198, 457
NUKE_COMMENTS (calibre.devices.interface.DevicePlugin attribute), 191, 450

O

oldest_article (calibre.web.feeds.news.BasicNewsRecipe attribute), 42, 108, 300, 366
on_import (calibre.customize.FileTypePlugin attribute), 184, 443
on_postimport (calibre.customize.FileTypePlugin attribute), 184, 443
on_postprocess (calibre.customize.FileTypePlugin attribute), 184, 443
on_preprocess (calibre.customize.FileTypePlugin attribute), 184, 443
open() (calibre.devices.interface.DevicePlugin method), 192, 451
OPEN_FEEDBACK_MESSAGE (calibre.devices.interface.DevicePlugin attribute), 191, 450
options (calibre.customize.conversion.InputFormatPlugin attribute), 189, 448
options (calibre.customize.conversion.OutputFormatPlugin attribute), 189, 448
options (calibre.ebooks.metadata.sources.base.Source attribute), 186, 445
OSX_MAIN_MEM_VOL_PAT (calibre.devices.usbms.device.Device attribute), 198, 457
output_encoding (calibre.customize.conversion.InputFormatPlugin attribute), 188, 447
OutputFormatPlugin (class in calibre.customize.conversion), 189, 448

P

parse_feeds() (calibre.web.feeds.news.BasicNewsRecipe method), 38, 104, 297, 362
parse_index() (calibre.web.feeds.news.BasicNewsRecipe method), 39, 104, 297, 362
path_sep (calibre.devices.interface.DevicePlugin attribute), 191, 450
Plugin (class in calibre.customize), 182, 441
populate_article_metadata() (calibre.web.feeds.news.BasicNewsRecipe method), 39, 104, 297, 363
popup_type (calibre.gui2.actions.InterfaceAction attribute), 199, 458
post_yank_cleanup() (calibre.devices.interface.DevicePlugin method), 193, 452
postimport() (calibre.customize.FileTypePlugin method), 184, 443
postprocess_book() (calibre.customize.conversion.InputFormatPlugin method), 189, 448

postprocess_book() (calibre.web.feeds.news.BasicNewsRecipe method), 39, 104, 297, 363
postprocess_html() (calibre.web.feeds.news.BasicNewsRecipe method), 39, 105, 297, 363
PreferencesPlugin (class in calibre.customize), 201, 460
prepare_addable_books() (calibre.devices.interface.DevicePlugin method), 195, 454
preprocess_html() (calibre.web.feeds.news.BasicNewsRecipe method), 39, 105, 298, 363
preprocess_raw_html() (calibre.web.feeds.news.BasicNewsRecipe method), 40, 105, 298, 363
preprocess_regexps (calibre.web.feeds.news.BasicNewsRecipe attribute), 42, 108, 300, 366
print_version() (calibre.web.feeds.news.BasicNewsRecipe class method), 40, 105, 298, 364
priority (calibre.customize.Plugin attribute), 183, 442
priority (calibre.gui2.actions.InterfaceAction attribute), 199, 458
PRODUCT_ID (calibre.devices.interface.DevicePlugin attribute), 190, 449
publication_type (calibre.web.feeds.news.BasicNewsRecipe attribute), 43, 108, 301, 366

R

re:test(), **116**, **375**
recipe, **260**
recipe_disabled (calibre.web.feeds.news.BasicNewsRecipe attribute), 43, 108, 301, 366
recommendations (calibre.customize.conversion.InputFormatPlugin attribute), 189, 448
recommendations (calibre.customize.conversion.OutputFormatPlugin attribute), 190, 449
recursions (calibre.web.feeds.news.BasicNewsRecipe attribute), 43, 108, 301, 366
refresh_gui() (calibre.gui2.preferences.ConfigWidgetInterface method), 202, 461
regexp, **261**
register() (calibre.gui2.preferences.ConfigWidgetBase method), 203, 462
remove_attributes (calibre.web.feeds.news.BasicNewsRecipe attribute), 43, 108, 301, 367
remove_book() (calibre.devices.interface.BookList method), 197, 456
remove_books_from_metadata() (calibre.devices.interface.DevicePlugin class method), 194, 453
remove_empty_feeds (calibre.web.feeds.news.BasicNewsRecipe attribute), 43, 108, 301, 367
remove_javascript (calibre.web.feeds.news.BasicNewsRecipe attribute), 43, 108, 301, 367
remove_tags (calibre.web.feeds.news.BasicNewsRecipe attribute), 43, 108, 301, 367
remove_tags_after (calibre.web.feeds.news.BasicNewsRecipe attribute), 43, 109, 301, 367
remove_tags_before (calibre.web.feeds.news.BasicNewsRecipe attribute), 43, 109, 301, 367
requires_version (calibre.web.feeds.news.BasicNewsRecipe attribute), 44, 109, 302, 367
reset() (calibre.devices.interface.DevicePlugin method), 192, 451
restart_critical (calibre.gui2.preferences.ConfigWidgetInterface attribute), 202, 461
restore_defaults() (calibre.gui2.preferences.ConfigWidgetInterface method), 202, 461
restore_defaults_desc (calibre.gui2.preferences.ConfigWidgetInterface attribute), 202, 461
reverse_article_order (calibre.web.feeds.news.BasicNewsRecipe attribute), 44, 109, 302, 367
RSS, **260**
run() (calibre.customize.CatalogPlugin method), 185, 444
run() (calibre.customize.FileTypePlugin method), 184, 443

S

sanitize_path_components() (calibre.devices.usbms.device.Device method), 198, 457
save_settings() (calibre.customize.Plugin method), 183, 442
save_settings() (calibre.devices.interface.DevicePlugin class method), 195, 454
set_all_user_metadata() (calibre.ebooks.metadata.book.base.Metadata method), 137, 154, 396, 413
set_driveinfo_name() (calibre.devices.interface.DevicePlugin method), 195, 454
set_identifier() (calibre.ebooks.metadata.book.base.Metadata method), 136, 153, 395, 412
set_identifiers() (calibre.ebooks.metadata.book.base.Metadata method), 136, 153, 395, 412

set_metadata() (calibre.customize.MetadataWriterPlugin method), 185, 444
set_option() (calibre.devices.interface.DevicePlugin method), 196, 455
set_plugboards() (calibre.devices.interface.DevicePlugin method), 195, 454
set_progress_reporter() (calibre.devices.interface.DevicePlugin method), 193, 452
set_user_blacklisted_devices() (calibre.devices.interface.DevicePlugin method), 196, 455
set_user_metadata() (calibre.ebooks.metadata.book.base.Metadata method), 137, 154, 396, 413
settings() (calibre.devices.interface.DevicePlugin class method), 195, 454
shutdown() (calibre.devices.interface.DevicePlugin method), 195, 454
shutting_down() (calibre.gui2.actions.InterfaceAction method), 201, 460
simultaneous_downloads (calibre.web.feeds.news.BasicNewsRecipe attribute), 44, 109, 302, 367
skip_ad_pages() (calibre.web.feeds.news.BasicNewsRecipe method), 40, 105, 298, 364
SLOW_DRIVEINFO (calibre.devices.interface.DevicePlugin attribute), 191, 450
smart_update() (calibre.ebooks.metadata.book.base.Metadata method), 137, 154, 396, 413
sort_index_by() (calibre.web.feeds.news.BasicNewsRecipe method), 40, 105, 298, 364
Source (class in calibre.ebooks.metadata.sources.base), 186, 445
specialize() (calibre.customize.conversion.InputFormatPlugin method), 189, 448
specialize_css_for_output() (calibre.customize.conversion.OutputFormatPlugin method), 190, 449
specialize_global_preferences() (calibre.devices.interface.DevicePlugin method), 196, 455
split_jobs() (calibre.ebooks.metadata.sources.base.Source method), 186, 445
standard_field_keys() (calibre.ebooks.metadata.book.base.Metadata method), 136, 153, 395, 412
STANDARD_METADATA_FIELDS (in module calibre.ebooks.metadata.book.base), 137, 154, 396, 413
start_plugin() (calibre.devices.interface.DevicePlugin method), 196, 455
startup() (calibre.devices.interface.DevicePlugin method), 195, 454
stop_plugin() (calibre.devices.interface.DevicePlugin method), 196, 455
summary_length (calibre.web.feeds.news.BasicNewsRecipe attribute), 44, 109, 302, 367
supported_platforms (calibre.customize.Plugin attribute), 182, 441
supports_collections() (calibre.devices.interface.BookList method), 197, 456
supports_gzip_transfer_encoding (calibre.ebooks.metadata.sources.base.Source attribute), 186, 445
supports_restoring_to_defaults (calibre.gui2.preferences.ConfigWidgetInterface attribute), 202, 461
sync_booklists() (calibre.devices.interface.DevicePlugin method), 194, 453

T

tag_to_string() (calibre.web.feeds.news.BasicNewsRecipe class method), 40, 105, 298, 364
template_css (calibre.web.feeds.news.BasicNewsRecipe attribute), 44, 109, 302, 367
template_to_attribute() (calibre.ebooks.metadata.book.base.Metadata method), 137, 154, 396, 413
temporary_file() (calibre.customize.Plugin method), 184, 443
test_fields() (calibre.ebooks.metadata.sources.base.Source method), 186, 445
THUMBNAIL_HEIGHT (calibre.devices.interface.DevicePlugin attribute), 191, 450
timefmt (calibre.web.feeds.news.BasicNewsRecipe attribute), 44, 109, 302, 368
timeout (calibre.web.feeds.news.BasicNewsRecipe attribute), 44, 109, 302, 368
title (calibre.web.feeds.news.BasicNewsRecipe attribute), 44, 109, 302, 368
to_html() (calibre.ebooks.metadata.book.base.Metadata method), 137, 154, 396, 413
total_space() (calibre.devices.interface.DevicePlugin method), 193, 452
touched_fields (calibre.ebooks.metadata.sources.base.Source attribute), 186, 445
type (calibre.customize.Plugin attribute), 183, 442

U

upload_books() (calibre.devices.interface.DevicePlugin method), 194, 453
upload_cover() (calibre.devices.usbms.driver.USBMS method), 198, 457
URL, **260**

USBMS (class in calibre.devices.usbms.driver), 198, 457
use_embedded_content (calibre.web.feeds.news.BasicNewsRecipe attribute), 44, 109, 302, 368
use_javascript_to_login (calibre.web.feeds.news.BasicNewsRecipe attribute), 44, 109, 302, 368
UserAnnotation (calibre.devices.interface.DevicePlugin attribute), 191, 450

V

VENDOR_ID (calibre.devices.interface.DevicePlugin attribute), 190, 449
version (calibre.customize.Plugin attribute), 182, 441
VIRTUAL_BOOK_EXTENSIONS (calibre.devices.interface.DevicePlugin attribute), 191, 450

W

WANTS_UPDATED_THUMBNAILS (calibre.devices.interface.DevicePlugin attribute), 191, 450
web2disk command line option
–base-dir, -d, 254, 510
–delay, 254, 510
–dont-download-stylesheets, 254, 510
–encoding, 254, 510
–filter-regexp, 254, 510
–help, -h, 254, 510
–match-regexp, 254, 510
–max-files, -n, 254, 510
–max-recursions, -r, 254, 510
–timeout, -t, 254, 510
–verbose, 254, 510
–version, 254, 510
WINDOWS_CARD_A_MEM (calibre.devices.usbms.device.Device attribute), 198, 457
WINDOWS_CARD_B_MEM (calibre.devices.usbms.device.Device attribute), 198, 457
WINDOWS_MAIN_MEM (calibre.devices.usbms.device.Device attribute), 198, 457
windows_sort_drives() (calibre.devices.usbms.device.Device method), 198, 457