mbib Documentation

mbib is a literature manager, with capabilities similar to JabRef, but intended to better cope with large databases. Key features and limitations are:

- Import from and export to BibTeX
- Import records from DOI and PubMed identifiers
- Push citations to Texmaker/TexStudio
- Push citations to OpenOffice/LibreOffice (collectively referred to as “OOo” below). Formatting of bibliographies in OOo piggybacks on JabRef, that is, you need to have JabRef installed in order to fully use mbib with OOo.
- Written in Python3
- Console-based GUI—basic yet responsive; based on the urwid and urwidtrees Python libraries.
- Data are stored in a SQLite database. Therefore, starting the program does *not* load the entire database into memory. The database has a fairy simple structure that lends itself to direct querying and manipulation with SQL.
- Developed and tested only on Linux, and will likely not work out of the box on other platforms. (Volunteers for porting it to other platforms are welcome.)

1 Preliminary notes

1.1 Motivation

I started writing mbib after my previous literature manager (bibus) broke down because of growing incompatibilities with LibreOffice, wxPython etc. I tried using JabRef for a while, and while it’s really pretty good in many ways, it really slows down once you have several thousand references in your database. (The same goes for Zotero and Mendeley.)

1.2 Status

At present, mbib is alpha software. Incompatible changes might still happen to the code and the database structure. However, in the latter case, I will provide a script for migrating the database to the new format (since I will have to migrate my own data anyway).

The program was written with my own needs in mind; I work in biochemistry and use PubMed as my main online literature database and therefore have implemented import of references via PubMed identifiers. People in other fields might miss tighter integration with other databases. I’m open to adding support for those, but unless prodded, it won’t be a priority. Similarly, some other bits of functionality are tailored to my own personal preferences and may seem a little idiosyncratic to others.

The code violates all manner of software engineering gospel. There are no unit tests, and I don’t plan to add them; the doc strings are a bit spotty and not formatted for automatic conversion into API docs. That said, I will give an overview of the program structure below, which hopefully will help you find your way through the code.

2 Installation

In the following, I am going to describe how things work on *my* system. I am running Debian with a KDE desktop. I don't suppose there will be any major differences with other Linux distros or window managers, but I am not going to verify this by experiment. If you manage to get it to work on other systems and have some specific tricks to share, please let me know, and I will include them here.

2.1 Prerequisites

In order to run `mbib`, you first need to install these programs and libraries:

- Bash
- Python3
- SQLite
- The `urwid` and `urwidtrees` libraries for Python3
- If you intend to use `mbib` with `OOo`, you will also need `JabRef`
- If you want to copy items to the X clipboard, you will need `xclip`
- For viewing or emailing PDF files, `mbib` relies on `xdg-open` and `xdg-email`

On Debian, all of these prerequisites can be installed through the system's package manager. A copy of SQLite already comes as part of the standard library when you install Python3, but you may also want to install the `sqlite3` package, which provides the command line client that lets you run SQL statements on your database.

The `xdg-open` and `xdg-email` utilities are probably installed by default on any graphical Linux desktop; in Debian, they reside in the `xdg-utils` package.

2.2 Installing `mbib`

I guess I need to put it on PyPi somehow, and maybe on GitHub. On second thought, why bother with PyPi? I hate this entire pip and easyinstall setup nonsense, I always find it more trouble than it is worth. Also, the intended audience is not Python programmers. Let's just put it on Github, and that is that.

Well, there is a slight hitch, is there not—pip doesn't really seem to help with installing data files. I guess I will deal with this by creating some bootstrapping code that looks for the directory on start-up and creates it when needed. We can also accept some command line argument with the folder path—yes, I guess I will do that. That means we will have to use argparse or something. As a side effect, this will also allow the user to keep multiple databases in separate directories.

How about this: we stuff everything inside a single folder. All the python code can go into a zip archive, and that can be imported. (We can use pyc, but they have to be renamed according to Python2 conventions.) We can insert the zip into PYTHONPATH in bash, and that is the end of it. Well, it is not even necessary; why not just keep the code inside a subfolder. At the top level, we just have that folder, the config file, and the database skeleton file to which the config file points.

Well, we should make it easy for the user to update the code files only, without clobbering and slobbering the data and config, should we not? How do we do that?

We could define the standard locations for config and database one level above the directory that the `mbib` code resides in. Then, the startup script would check those locations and use them. E.g. ../mbib.cfg and ../mbib.db—let's check for those. If they are not found, the start-up script will ask if they should be generated. The alternative would be to exit the program and let the user copy or move the files from a backup location.

Alternate scenario – save config file in the home directory. If it is found, look for database file in that directory. I guess we should do that. Then, the start-up script can first write the config file. There, it will find the file name of the database. If that exists, then it will be opened; if not, the start up script will propose to either create an empty file or exit.

I guess the second scenario makes more sense. The location of the config file will be hard-coded into the bash script. I guess we will then also have to make sure the same files are read within python; for that, we either need to pass them on the command line or set them into environmental variables with export. I guess I will use the latter. Yes, this all sounds very reasonable.

2.3 Configuration

This will wait until after we have decided on the installation procedure. One point to stress is that we need to put one or the other start-up script into the shell's path; I don't think that pip lets us do this automatically. We will just show an example.

3 Running `mbib`

The program runs inside a console window. Assuming you have put a start-up script into your shell's path, On KDE, you can start

3.1 Starting the program

Probably, the best way is to use `python3 -m`.

First, the standard way, and then maybe the wrapper bash script. Also how to add it to the K Menu and the work-around for konsole - use xterm instead. Customization of xterm via the /.Xresources file and

xrdb -merge /.Xresources

Leave it to others to contribute instructions on how to run it on other desktops.

As noted above, we will need command line argument parsing.

3.2 The user interface

- runs inside a shell
- navigation (mouse, keyboard)
- viewing and editing references
- importing references
- moving things around (using selections)
- searching and filtering
- using the clipboard (requires xclip)
-

Some gotchas to point out

- Item selections are stored in the database. Maybe we should offer a configuration option to deselect everything, just as we do with the "recently added" folder. Yes, we do that now.

- Moving selected items from search results: the search folder is basically a simple folder in the database. So, if items in this folder are selected and moved, the copies in the permanent folder stay where they are.

  We may think about an option to erase all other copies of a reference. Yes, we have that now. Or all selected references – we don't have that yet. If we did have it, I guess we should offer a confirmation dialog that specifies the number of selected references.

- And while we are on it: All the special top-level folders are special just in mbib, because specific operations are available and others excluded; in particular, they cannot be deleted, moved, or renamed. However, no such protective restrictions apply when working with the database in SQL mode.

  Generally speaking, it is a good idea to make a safety backup copy of the database file before performing major surgery in SQL.