BibleTools User Documentation

by Jesse Jacobsen

$Id : bibletools.tex, v1.72002/02/2105 : 32 : 56jmatjacoExp$

BibleTools was written originally in Python 1.5, but is being updated to run under 2.x. The syntax should still work properly, but some incompatibilities were introduced in the Python libraries that BibleTools makes use of. Please be patient while the kinks are worked out. If it proves too cumbersome to continue supporting Python 1.5, then that support will be dropped in favor of Python 2.x.

1 Installation

1.1 Information Common to All Platforms

At this point it is recommended that you use the "source" distributions as found on the SourceForge BibleTools download page. Just download the latest release. For Unix (e.g. Linux), run `tar -xzf FILENAME` on it to unpack it in the current directory. For other OS's, use your favorite multi-archive unpacking tool (e.g. WinZip, etc.).

Once you have the source directory in front of you, your next step depends a little on certain preferences you may have, and a little on the platform you're running. The main element is that you will run the program **setup.py**. *How* you run it and what you do afterward may vary.

Note that **setup.py** will check to make sure you have certain add-on Python libraries installed. If it does not detect them, then you should install them or make sure **setup.py** is running in the environment you think it is.

The add-on libraries are found in the mxExtension series by Marc Andre Lemburg. At the time of this writing, three of them are used in BibleTools: mxDateTime, mxTools, and mxTextTools.

If you are running a flavor of Linux or another Unix that uses package files customized for your system, then check to see if these are available for your system.

On Debian GNU/Linux, there are several of these mxExtension packages available, and you must choose the ones that go with the version of Python you use.

If you are running Windows or Macintosh, I recommend that you head straight to the mxExtension home page at:

http://www.lemburg.com/files/python/eGenix-mx-Extensions.html

Follow the instructions there to download and install the needed extensions for your system and Python version. (In rare cases, you may need to compile them too, unfortunately.)

Once these extensions are installed, you can test that they are available by running an interactive Python session (on a command line, type `python` and press Enter) and typing the commands:

```
>>> from mx import TextTools
>>> from mx import DateTime
>>> from mx import NewBuiltins
```

If the above commands produce an error, then either the extension was not installed at all, or it was not installed correctly. You can quit your interactive session by pressing Control–D or Control–Z, depending on the system.

Note: Older versions of these extensions imported without the words `from mx`, in other words, they just used the keyword `import` and the module name used above. If that's what works on your system, then you will have to upgrade these extensions to a newer version.

BibleTools uses a configuration file that allows you to customize its behavior in many ways. There is a sample file in the BibleTools distribution called **bibletools.cfg**. You should copy it to the place where system-wide configuration files live. (For example, /etc/bibletools.cfg or C:\windows\bibletools.ini. If your platform is different, please let me know where you put it so I can support that configuration.) You can edit the configuration file to change the settings. Also, individual users can make a copy of the main configuration file to override the default behavior. That copy is located where the configuration variable `userconfig` specifies. This might only be useful on Unix-like systems.

1.2 Unix and Linux

I run Debian GNU/Linux, which strictly observes the rule that all software not originating with Debian should be installed under /usr/local/. You may want to install BibleTools there too, for a number of possible reasons. If so, then you might use the same command I do (as root):

```
# ./setup.py install --prefix=/usr/local
```

If that doesn't work, it may be that the file **setup.py** is not executable. Or, maybe your Unix (for whatever strange reason) does not support the #! interpreter convention. Or maybe (gasp!) Python is not installed correctly. Or, if Python is installed, maybe you have version 1.5.x and have not installed Distutils. If that is the problem, then fix it either by installing the Distutils package for your OS or downloading it straight from www.python.org. (Distutils comes with Python 2.x automatically.)

If Python and Distutils are both installed properly, then you can probably run the same command as above like this instead:

```
# python setup.py install --prefix=/usr/local
```

Note: If your Linux distribution or your Unix setup does not anticipate the installation of Python packages under /usr/local, you have two choices. Either install BibleTools in your main Python tree under /usr/lib/pythonX.X, or add a file named "local.pth" under /usr/lib/pythonX.X/site-packages. It should contain at least one line that reads "/usr/local/lib/pythonX.X/site-packages". You may want another line that contains everything up to the last slash. That should add the "local" Python tree to the search path every time you run any Python program. Of course, for this discussion, "X.X" should be replaced with the version number of your Python, for example 1.5 or 2.1.

Once that's done, you can move on to using BibleTools if you wish. But if you still have patience left, you could expend it by getting your system ready to run the BibleTools server every time it starts up, like a regular system daemon. The server always just runs in the background, ready to service any client requests — much like a web server — so this kind of arrangement makes sense. The way to go about it, though, varies from system to system and therefore is beyond the scope of this document. Hint: I'd start by looking in your /etc/init.d/ directory, if you have one. There is a sample daemon control script in the BibleTools distribution at platforms/unix/sysvinit.sh. This file contains some additional documentation inside it. If you write a daemon control script for any Unix other than Debian GNU/Linux, please send it to me so I can include that one too.

1.3 Microsoft Platforms

BibleTools should run on Windows, though I don't try it very often, and then only briefly to see if something works. The Windows filesystem is much less structured by default than in Unix, so I don't know if it would be practical to attempt enforcing

a standard layout for add-on software. Therefore, I recommend running **setup.py** like this:

```
C:> python setup.py install
```

This will cause Distutils to use defaults for all locations, installing BibleTools into your main Python directory tree. Once **setup.py** runs successfully, you might want to automate starting the **btsrv.py** daemon. You can probably do this by putting the command in a shell script and then putting the shell script in a directory like C:\WINDOWS\STARTUP\. The command you use will depend upon your preferences. I suggest you take a look at the sample **bibletools.cfg** file for some information and run the command (at the DOS prompt) `python btsrv.py --help` (assuming the file **btsrv.py** is in your current directory).

1.4 Macintosh

I don't have a Mac to try anything with, unfortunately. At this point, I suggest you run BibleTools under OS X, where you can follow the instructions above for Unix machines. If you use BibleTools successfully on the Mac, please let me know so I can write something useful here.

2 Usage

Write Me!

3 Development

Write Me!