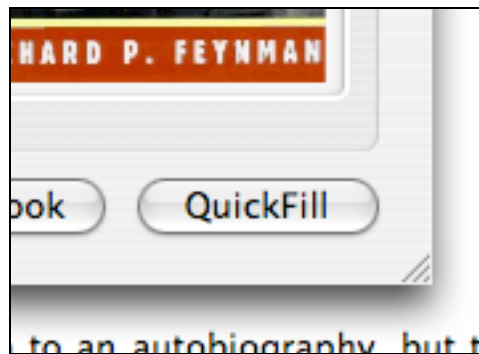


Writing Books 3.0 Quickfill Plug-Ins

Version 1.0
February 9, 2006

Introduction

Quickfill plug-ins are small bits of computer code that allow Books to fetch a detailed set of information for a book record using a limited amount of information such as a title or ISBN number. Quickfill plug-ins are invoked from the book details window:



There are several reasons to write a Quickfill plug-in. If you're a user with books in languages other than the ones the default set of plug-ins support, you can write a plug-in that supports particular locales. If you're a book data provider, giving Books users a plug-in that supports your service can advertise your service and gives you the opportunity to push information down to users, such as links where a given book may be found (and maybe purchased) within your service.

Creating a new Quickfill plug-in is a relatively simple exercise, but it also requires that you be knowledgeable in a few areas as a developer:

1. Quickfill plug-ins contain specialized scripts and applications that communicate with the target data sources. You should be comfortable writing small Unix command-line programs.
2. Communication between the plug-in and the main application is established using XML. You should be comfortable parsing and manipulating XML in your development language.
3. Quickfill plug-ins assume a Unix-like environment. You should be comfortable working with a Unix-like file system and you should be familiar with concepts like pipes and output streams.

4. Quickfill plug-ins almost always requires the use of a network. You should be comfortable with network programming, or comfortable using a software library that handles this for you.

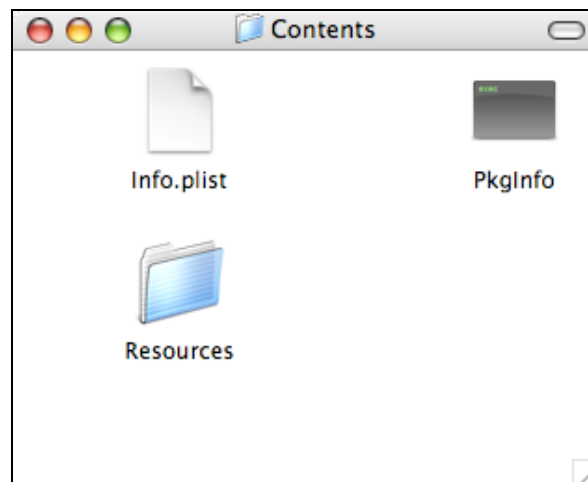
If you have these skills mastered, writing a Quickfill plug-in will be a quick and simple exercise. Note that no particular program languages are required. A Quickfill plug-in can be written in any language that can be used to compose a command-line Unix program. This includes scripting languages like Python, Perl, and Ruby in addition to more traditional languages like C and Java.

Note: While it's perfectly acceptable to write a Quickfill plug-in in a compiled language, you should be aware of the fact that Books is a Universal application, and you may need to include architecture-specific binaries for compiled programs. For this reason, it's recommended that you use an interpreted or scripting language when possible in order to minimize platform-specific headaches and maintenance.

The Quickfill plug-in structure

Quickfill plug-ins are distributed and packaged as bundles. Bundles are folders that appear as single files from the perspective of the user. Bundles are common on MacOS X for things like documents (Keynote) and applications (Books). While Import and Export plug-ins are full application bundles, Quickfill plug-ins are simpler.

At the root of a Quick-fill plug-in is a solitary Contents folder. This folder in turn contains a Resources folder, a PkgInfo text file, and the Info.plist property list. The Resources folder contains the program you write and any supporting materials. The Info.plist file contains plug-in details such as the program name.



The Info.plist file from the Amazon Quickfill plug-ins that ships with Books contains the following contents:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>BooksPluginName</key>
    <string>Amazon (US)</string>
    <key>BooksPluginType</key>
    <string>Quickfill</string>
    <key>BooksScriptName</key>
    <string>amazonScript.py</string>
    <key>CFBundleIdentifier</key>
    <string>net.aetherial.books.importer.amazonquickfill</string>
    <key>CFBundleInfoDictionaryVersion</key>
    <string>6.0</string>
    <key>CFBundleName</key>
    <string>Amazon (US)</string>
    <key>CFBundlePackageType</key>
    <string>APPL</string>
    <key>CFBundleSignature</key>
    <string>????</string>
    <key>CFBundleVersion</key>
    <string>1.0</string>
</dict>
</plist>

```

The name of the plug-in is `Amazon (US)`, the plug-in type is `Quickfill`, and the script used to communicate with Amazon is called `amazonScript.py`. The custom bundle identifier is completely optional.

If you look in the Resources folder of the Amazon plug-in, you'll find two files (three if the Python code has created compiled byte-code). The `amazonScript.py` is the program referenced by `Info.plist`, and the `amazon.py` is an auxiliary library that the main program uses.

Writing a custom Quickfill plug-in

The recommended way of writing a Quickfill plug-in is to take an existing plug-in and modify it by replacing the relevant values in `Info.plist` and replacing the program with your own. Plug-ins can be found within the Resources folder in the main `Books.app` bundle or online along with this guide.

After you locate a suitable plug-in to use as a template, you can begin writing your own custom program that will serve as the core of the new plug-in. This is how Books discovers Quickfill plug-ins:

1. On boot-up, Books looks within its own Resource folder in addition to
`~/Library/Application Support/Books/Plugins`
`/Library/Application Support/Books/Plugins`
for files ending in a `.plugin` extension.
2. Books checks the `Info.plist` file to verify that the value corresponding to `BooksPluginType` is `Quickfill`.
3. Once this process is finished, users can select this plug-in from the preferences window.

When the user hits the Quickfill button from the book details window, the following process is executed:

1. Books writes an XML file with the available data for the current book record to
`/tmp/books-quickfill.xml`
2. It looks for the name of the plugin program to use from the plug-in's `Info.plist` file.
3. Books executes the plug-in program, attaching a listener to the program's standard output stream.
4. The plug-in program reads the `/tmp/books-quickfill.xml` file and parses it for the values that it needs to obtain information from the remote source. If an ISBN number is sufficient, then only that need be read. Likewise, if the source needs an author and title, those values can also be used.

An example of a sample XML file passed to the plug-in program:

```
<Book>
  <field name="isbn">192975213X</field>
  <field name="authors">John F. X. Sundman</field>
  <field name="title">Acts of the Apostles</field>
</Book>
```

5. Using the information gathered, the program communicates with the data source to obtain a fuller record.
6. The program creates an XML document and plugs in the values received from the data source into the XML file. The returned XML contains a document element named `importedData` containing a single `Book` element that mirrors the format the program parses:

```

<importedData>
  <Book>
    <field name="publisher">Rosalita Associates</field>
    <field name="publishDate">
      1999-11-17 12:00:00 -0600
    </field>
    <field name="isbn">192975213X</field>
    <field name="authors">John F. X. Sundman</field>
    <field name="title">Acts of the Apostles</field>
    <field name="format">Paperback</field>
    <field name="genre">Science Fiction</field>
    <field name="customField">customData</field>
  </Book>
</importedData>

```

7. When the XML document is complete, the program outputs it using the UTF-8 character set to the standard output stream and exits. Books reads this data, parses it, and adds any new fields to the current book record. (Books will not overwrite a field that already contains a value.)

While you're developing the plug-in, it's often useful to run the plug-in program from the Terminal and verify the data output to standard out. When you're confident that the program works correctly, you should place the program in the Resources folder and update Info.plist to reflect the name of the plug-in program.

When this is complete, place your plug-in in

```
~/Library/Application Support/Books/Plugins
```

and launch Books. Your plug-in should appear in the menu in the preferences window. Select it and give it a spin.

Distributing your plug-in

When you're confident that your plug-in is well-tested and robust, you may wish to distribute it to other Books users. There's a couple of ways to get the word out:

1. Send an e-mail to books@aetherial.net. This is the best way to get it listed on the Books Plug-In weblog at

```
http://www.aetherial.net/booksplugins/
```

2. Send a note to the Books Users e-mail list. Details about the list are available at

```
http://groups.yahoo.com/group/books\_users/
```

If the plug-in would be useful to a large enough number of Books users, it may be included in the main distribution.

Frequently asked questions

Q: Where are the frequently asked questions?

A: This is a 1.0 document. No one has had the opportunity to ask any questions, much less ask any questions frequently.

Feedback

If you have any questions, comments, or suggestions, please send them to

`books@aetherial.net`

Document changes

February 9, 2006 (1.0): Initial version