Catalogueing your books (and other media)

1 The Problem

I am an avid reader. In fact, most of the space in my shelves is used by books. Books that cover many different topics: Maths, Physics, Programming, Fantasy, Sci-Fi, Dramas/Plays, Poems, Classics…

Recently, I became aware of the fact that this collection had to be sorted and catalogued, somehow. This would enable me to sort out the duplicate books[1] and to prevent getting duplicates. Moreover, I said to myself:

> If I catalogue *all* books that are stored somewhere in our house, I could easily spot the literary gems that are present in some dark box, unbeknownst to me and the others.

To solve this problem, several subproblems had to be taken care of: **What** book data do I need? **How** do I get this data? **Where** do I store the data? Thus, the problem was well-defined and I could begin looking for solutions.

2 A Solution

In Germany, most of the books that are published are assigned a unique number, the ISBN[2]. On newer books, this number is printed on the back of the book as a barcode, allowing booksellers to get information (price etc.) about the book as quickly as possible.

For many years, the ISBN was a 10-digit number: 9 digits for the code itself and one for the checksum. However, since the beginning of 2007 (European) books are required to carry a 13-digit ISBN on their backs: The EAN[3]. EAN is a much more general code that is, as the name implies, used for other product categories, such as DVDs, CDs or food.

Luckily (for me), the 13-digit ISBNs are a subset of the 13-digit EANs[4], so there would be no conversion necessary between these two codes.

2.1 Getting data

Quite naturally, the thought of how to exploit this fact came almost immediately to me. Given a list of EANs[5], I could get all the data I need if I had a source that allowed me to search for EANs and return information about the item.

[1] A general problem when one has "some" books: One is given books by other people as a present, but most of them are already in the personal collection.

[2] International Standard Book Number

[3] European Article Number

[4] Actually, the EAN comes in two flavours: 13 digit or 8 digits. The EAN-8 is commonly used for "smaller" items and since we are talking about catalogueing books and not about running a grocery store, I think we can gracefully ignore it.

[5] I will now use this word regardless of whether I am writing about ISBNs or EANs.

Since I have not much money to spend on this project, I decided to query the most comprehensive (and free!) database of books I knew: The online bookstore “amazon.com”.

This task was easier than I expected it to be: amazon.com provides developers with (free) access to the “Amazon E-Commerce Service” (ECS). Being an amazon.com customer anyway, after reading the documentation, I decided to sign up for it and give it a try.

2.2 Getting EANs

Before doing anything else, I first had to find a way of how to read EANs. I was not willing to enter *all* EANs manually. But what was good for booksellers should work for me as well. So I bought an `OVOX CCD-800 USB Barcode Scanner`.

This device is truly perfectly suited for my needs: It reads EAN and UPC barcodes in all flavours (and a whole lot more barcodes which I will probably never encounter in my life) fast and reliably (if the barcode can be read, it will be read properly without any errors), it can be configured nicely (by scanning barcodes, of course[6]) and it identifies itself as a normal USB keyboard, hence allowing me to use it with FreeBSD (or Windows).

I configured the scanner to read the barcode, print it with alphanumeric characters and add a newline afterwards. This meant I could use `vim`, the best text editor, and scan every book I encountered. The EANs would be added to a list, which I would process later on.

3 Hyppolyta, the Amazon queen

After I knew what had to be done, I could begin implementing a program that would to the following:

1. Accept a list of EANs as an input file.
2. Request further information (Title, Author(s)...) via Amazon’s ECS.
3. Save information about the items in a sensible and practical format.

The name “Hyppolyta” seemed very apt for this program. Forgive me my bad taste. Besides wondering how anyone could name a program “Hyppolyta”, you may have noticed that I wrote ‘items’ instead of ‘books’. This was a perfectly sane decision: Since amazon.com offers a lot more products besides books, I wanted my program to be as general as possible in order to get information about all sorts of products that can be identified by EANs. This resulted in me catalogueing my audio CDs and DVDs, too (but this is an entirely different story. Let’s focus here.).

[6]Which seems to be a mild case of bootstrapping...

Hyppolyta is written in Perl[7]. I am by no means a Perl wizard. In fact, it was my first project that was done in Perl. I am sure that the code could be implemented much more smoothly, therefore I encourage you to rewrite it if you consider it too offensive.

The current version of Hyppolyta is able to generate CSV output. This simple format should allow you to use your data in almost any program. Just make sure that the program you are going to use for imports is able to understand CSV data that contains the newline character in fields! Every field is escaped by default via `"`[8].

3.1 Installation

Untar everything. Install `wget` and place it in the path of your Perl interpreter. Furthermore, install the following Perl modules, if not already present:

- `LWP::UserAgent`
- `MIME::Base64`
- `XML::XPath`
- `Date::Format`
- `Text::CSV_XS`
- `Getopt::Long`

As a last step, open `hyppolyta.pl`, look for the line:

```
my $req_key = "[YOUR KEY]";
```

Add your AWS access key. If you do not have one, please register with Amazon (http://aws.amazon.com).

3.2 First steps

The steps of a general session with Hyppolyta could be something like this:

1. Go through the shelves/boxes one by one. Scan all books you encounter. I suggest sorting them before starting to scan them. For example, I tend to group them in two categories, “technical literature” versus “fiction” and scan each category in a separate text file.

2. You should now have a list of EANs/ISBNs. Each line does not contain any whitespaces and ends with a newline character. Start Hyppolyta from the command-line via:

[7] With lots of modules and its general flexibility, this language seemed most suited for my task. Not to mention the fact that it is one of the languages that is offically supported by the ECS API.

[8] If you are unsure about which program to use, just try Perl. . .

```
./hyppolyta.pl -i <input file> -o <output file>.csv
```

If you do not want Hyppolyta to download images for each item, specifiy the `-n` switch, too. Otherwise, all available images are stored in a subdirectory depending on the search index: If the search index is `DVD`, images will be stored in the folder `./dvd/`. The URI of the image is set accordingly, thus allowing you to simply upload the appropriate folder to your server in order to display all DVDs. By default, `wget` is used to download the images.

3. Hyppolyta will now download all information about the items and store them in a CSV file.

4. Import the file and have fun. As always, this step is left out as an exercice to the reader.

3.3 More complex usages

To understand Hyppolyta's configuration options you have to learn the basics about the "Amazon E-Commerce Service" (ECS). To get data from Amazon, an `ItemLookup` request is used. This request has several important parameters, which are preset by default when no other options are present:

- The `ReqEnd` which specifies the locale. Set to `de` by default. You can change this via the `-l` or `-locale` switch:

  ```
  ./hyppolyta.pl -i <input file> --locale co.uk
  ```

  Valid locales are (among others): `com`, `co.uk`, `de`. Please note that not all locales share the same database. It is advisable to change the locale if an insufficient number of items has been retrieved.

- The `IDType` that specifies the type of the search parameters, for example ISBN (default), UPC[9] or EAN. Actually, more options are possible, but these are not covered here.

- The `ItemID`, a list of up to 10 items of the type specified by `IDType`. These are, in fact, the codes you scanned earlier. The ID type can be changed using the `-t` or the `--idtype` switch:

  ```
  ./hyppolyta.pl -i upc.txt -t UPC
  ```

  Not all locales support all ID types. Read the fine manual for reference purposes.

[9] Universal Product Code; used in US and UK

- The `SearchIndex` that tells Amazon which part of the catalogue is to be searched. Valid values include: Books, ForeignBooks, Music, DVD. The default value is "Books". However this implies that *only* books of the request's locale are queried! That means: If the locale is set to `de` (by default), books in other languages will be available *only* (!) by using the "ForeignBooks" search index. Specifying the search index is possible via the `-s` or `--search-index` switch:

  ```
  ./hyppolyta.pl -e -i classical_music.txt -s Music
  ```

- The `ResponseGroup`, which used by Amazon to decide what information about an items is sent back. For example: The response group `ItemAttributes` returns a lot of information about each item, including package dimensions, whereas `EditorialReviews` adds the "official" review of the item to the response (if available). More information is available in the ECS documentation.

  This parameter should *not* be changed by the end user. Keep your hands off it unless you know what you are doing.

If available, as much as 10 items are used in one request. After the request has been assembled, it is sent to Amazon for processing. The response will be an XML file which is parsed by the `XML::XPath` module. This is where the heart of the program lies.

3.4 Tweaking Hyppolyta

To keep the program as general as possible, hashes are used everywhere in the program. If you look at the code, you will find two important variables: `@attributes_general` and `%attributes_specific`.

I will now explain to you how to use these wisely.

3.4.1 `@attributes_general`

This is a list of attributes that have to be included in every request. Each item has two subitems: First, the title/name of the attribute is specified. Then, a relative path that tells Hyppolyta where the attribute can be found in the response[10].

The name of the item is used when the CSV header is written. Although it is possible to use titles such as `Item1`, `Item2` etc., it is not prudent as it makes parsing and understanding a CSV file much harder. You can change the names to your leisure.

The `NodeIterate` value tells Hyppolyta wich nodes are to be used to iterate over multiple items. In general, this is just the node of the attribute as described in the ECS documentation. It is also possible to add new attributes. Just make sure that the path to `NodeIterate` is correct.

[10]Just think of a tree.

The value in `NodeValue` tell Hyppolyta where it should look for the actual value you want to receive. This may be different from the `NodeIterate` setting.

Furthermore, specify the flag `Output => 1` if you want Hyppolyta to print these values every time a new item has been retrieved from a request.

Example: Suppose we have a node called ‘Person’. This node has a child called ‘Name’. It it stored by ECS in the following way:

```
Person -- Name, Person -- Name, Person -- Name
```

In this case, we would set `NodeIterate` to ‘Person’ (actually, the whole path has to be used. But this is an example, after all) and `NodeValue` to ‘Name’.

3.4.2 `@attributes_specific`

An associative array for attributes that are only requested if a certain search index is used. The array’s key is the search index (keep in mind that ECS is indeed case-sensitive), the value is another hash of attributes that are requested for every item. The syntax is the same as above (`Name`, `NodeIterate` etc.).

While this may seem clumsy, it is, in fact, *not*, as I dare to say: People who want to organize their music collection need more information about artists etc., whereas people who want to sort their books are almost always happy to know about the author and the publisher. . .

If you want to add attributes, please read the ECS documentation for more information.

3.5 Use case: My collection

Let’s suppose you have your well-ordered list of ISBNs. Before doing anything else, these should be sorted according to their origin: In 13-digit ISBNs, the 4th digit will specify the origin: 4 is for Germany (or German-speaking countries), anything else for, well, another country. In 10-digit ISBNs, the origin is coded in the first number. Same game.

Use whatever you want to divide the list in two new lists (one with the “foreign” books and one with the local ones): Write a script, do it manually, use `awk` and `sed` etc.

At last, do two runs of Hyppolyta:

```
./hyppolyta.pl -i books_german.txt -o books_german.csv
./hyppolyta.pl -i books_foreign.txt -s ForeignBooks -o books_foreign.csv
```

The .csv files now contain data about your books. Furthermore, images for every item available should have been downloaded.

4 Updates and Feedback

This document will be updated as soon as I add new features or consider the documentation to be outdated. If you like the program, please drop me a line. My e-mail address is `canmore[AT]annwfn[THIS_IS_A_DOT]net`. New versions of Hyppolyta are available on my personal homepage (http://canmore.annwfn.net).