Fac-Back-OPAC: An Open Source Interface to Your Library System

By Mike Beccaria and Dan Scott

Overview

Fac-Back-OPAC, the faceted backup OPAC, is an advanced catalog offering features that compare favourably with the traditional catalogs available for today's library systems. Fac-Back-OPAC represents the convergence of two prominent trends in library tools: the decoupling of discovery tools from the traditional integrated library system; and the use of readily available open source components to rapidly produce leading edge technology to meet patron and library needs. Building on code originally developed by Casey Durfee in February 2007, Fac-Back-OPAC is available at no cost under an open source license to any library that wants to offer an advanced search interface or a backup catalog for their patrons from http://code.google.com/p/fac-back-opac/.

Dan Scott Traces the History of Fac-Back-OPAC

I joined Laurentian University as a systems librarian in early 2006, and it quickly became apparent to me that the proprietary library system catalog for which I had assumed responsibility suffered from a stability problem. We needed a backup catalog that was independent of our proprietary library system. Unfortunately, in mid-2006 a good alternative catalog that met my requirements--free, easy to deploy and customize, and able to run on a basic desktop machine--was hard to find. The best apparent option at the time, Koha, turned out to not be up to the task of handling our 750,000 bibliographic records (a limitation that will reportedly be lifted with the integration of the Zebra indexing engine in Koha 3.0). So my search for a backup catalog continued.

Fortunately, the open source search server, Solr, was the dominant theme of the code4lib 2007 conference. An all-day Solr preconference by Erik Hatcher kicked off the conference, and a demonstration of Andrew Nagy's MyResource portal proved Solr's potential. Casey Durfee's presentation, Open Source Endeca in 250 Lines or Less , gently mocked search technology requiring license costs in the six-digit price range by offering a fully open source alternative build on Solr. Casey's presentation held my attention because I was familiar with all of the building blocks of his solution and it promised to meet Laurentian's needs for a backup catalog.

As the conference drew to a close, I asked Casey about his intentions for the source code for his technology demo. He considered his work to be throw-away code but gave me permission to release it under an open source license as Fac-Back-OPAC - the faceted backup OPAC - and I invited others to join me in the project. Today, Fac-Back-OPAC is fulfilling Laurentian's needs for a backup catalog, but it has been greatly enriched by the contributions of others to the project.

Mike Beccaria Explains How He Got Involved:

I started working at Paul Smith's College in fall, 2005 as a Systems Librarian. Paul Smith's is a small college (~850 fte) with a small collection (~40,000 items) and, consequently, a small budget. Our library runs a SirsiDynix ILS using Webcat as an OPAC, and while SirsiDynix has several upgrades available to our OPAC, we couldn't justify the expense of providing a new skin and features that didn't address what we considered the core issues found in many traditional OPACs, namely the lack of customization, poor relevancy ranking algorithms (if any), and the lack of findability and browseability of collections on par with user expectations. The unveiling of NCSU's Endeca catalog in January 2006 set the standard for what I wanted for my library, and since then I had been trying to find a way to offer something similar to our patrons. It couldn't be difficult to create or implement because I'm not a professional programmer, and it had to be free because I didn't have much money to spend. This was a tall order to fill at the time, which was why I was so excited to hear about the presentations being given at Code4lib 2007.

At Code4lib 2007 I attended Erik Hatchers's Solr pre-conference and watched as Casey Durfee, Andrew Nagy and Erik Hatcher showed examples of what was possible even to the non-professional programmer. Except for a couple of books on Ruby on Rails and Lucene that I purchased and skimmed prior to the conference, I was not very familiar with the technologies presented there. The reality was that I had no idea whether a project like this was actually feasible for someone with my skill set. However, that concern was quickly resolved. The conference was held in late February and within a week of getting back I had a fully customizable OPAC with faceted navigation and advanced search features working on my desktop. By mid-March I had learned of Dan's Fac-Back-OPAC Google Code project and quickly signed up and integrated some enhancements I had made. After showing the rest of the library staff what was possible, I got approval to purchase a new server for the library to host, among other things, our new OPAC, which can be accessed at http://library.paulsmiths.edu/catalog.

Fac-Back-Opac Features

Fac-Back-Opac, though developed recently, already boasts a rich feature set absent in many proprietary ILS OPACs. It is one of the canonical examples of the next generation of OPACs coming to the market, providing librarians and patrons with advanced functionality that previously had only been available to institutions capable of footing a large bill and providing a team of programmers to implement. Using a car as a metaphor, let's take a look at some of the features that are offered, starting off under the hood to examine some functionality that librarians may be looking for and ending with the patron's experience of taking the OPAC for a test drive.

The Materials

Every piece of software is built on some sort of programming framework or architecture. Consider this the materials with which the OPAC was made. Under the hood, Fac-Back-OPAC has been developed using enterprise class open source software (Solr and Django). This combination delivers speed,

reliability, and efficiency, and is customizable by librarians and programmers willing to take the time to learn the technology from readily available documentation and examples. This capability simply isn't possible in the current field of proprietary ILS products. It also means that once a user understands the less than 2,000 lines of code that glues these components together, additional features can be added with relative ease.

Under the Hood

The engine in Fac-Back-OPAC is made up of two components: the indexer and the search engine. At its most basic level, Fac-Back-OPAC is designed to gather data from MARC records that are extracted from a library's ILS and is thus completely independent from the ILS itself. The indexer is very powerful, and is able to analyze and extract data at the most granular level of MARC fields and subfields using simple configuration files. More advanced users can create custom filters to tailor their data and index it in any way they wish. For example, with the indexer, a library can write a custom filter to change values in MARC records on the fly so the data can be displayed in a more user friendly way to the patron.

Complementing the indexer is the search engine. While the technical details will be discussed further in the next section, Fac-Back-OPAC uses Solr, an open source enterprise class search engine platform that is capable of advanced search functionality that rivals many proprietary systems. With the ability to customize fields and relevancy algorithms, Solr puts the power in the hands of the systems administrator and allows the librarians to determine what data is most important in order to specify which search results float to the top. Additionally, from the end-user perspective, Solr's query syntax includes many advanced features which are inherited when searching a Fac-Back-OPAC catalog. Specifically, these include field boosting, simple field searching (keyword, title, author, etc.), advanced search and Boolean operators (single and multiple character wildcards, stemming, grouping, phrase search, AND, +, OR, NOT, -), and sorting (relevance, publication date, descending, and ascending).

The Ride

When test driving a car you want it to be comfortable, easy to drive, and have some cool extras. Fac-Back-OPAC delivers on these same aspects. Fac-Back-OPAC gets patrons to their search destination with powerful resource discovery and sharing features. The catalog leverages descriptive metadata and controlled vocabulary by providing patrons with customizable faceted browsing. Additionally, because the catalog performs searches by reading information from the URL, all of the links in the catalog are shareable. Fac-Back-OPAC provides RSS and Atom feeds for every search, enabling patrons to keep track of new materials that are found for arbitrary searches. Combined with the option of a customizable item level display or linking to your existing catalog, as well as full internationalization (currently offering French and English interfaces), and Fac-back-OPAC demonstrates what is possible with the next generation of OPAC software. The dream of integrating your catalog into your website without having to hack the ILS while offering next generation search functionality is now a real possibility.

Technology Building Blocks

Most of Fac-Back-OPAC's sucess can be attributed directly to the excellent open source components on which it is built: Solr, marc4j, and Django. Let's follow the life cycle of a MARC record as it goes through the process of becoming a searchable document in Fac-Back-OPAC.

From the ILS to MARC

Following the same approach used by NCSU and other institutions to build a catalog outside of the bounds of the traditional integrated library system, we begin by exporting all of the MARC records and holdings on a nightly basis. After the initial export, you can use whatever tools your ILS offers to determine which records are new or changed, and export only those changed records to update your Fac-BACK-OPAC instance. We use the word instance here because multiple copies, or instances, of Fac-BACK-OPAC can run simultaneously on the same server.

From MARC to MARCXML: marc4j and Jython

We begin our journey with traditional MARC records. MARC, although a wonderful transmission and encoding format, is not well-suited to indexing by traditional search engines. So we use the excellent marc4j library to convert each MARC record from MARC to the more easily handled MARCXML format. As Solr only understands UTF-8 (an 8-bit Unicode character encoding format), we also convert any records encoded in MARC-8 to UTF-8 if required. In addition, we use the Jython programming language (a Python interpreter running in a Java Virtual Machine) to control marc4j and use the simpler Python syntax to control the rest of the indexing process.

Indexing records: Solr

Solr is a search solution that builds upon the enterprise-quality Lucene search engine by offering Web services that simplify indexing documents and add additional features such as faceted search results. Each Solr instance can be configured with a schema that describes the fields of a document and the characteristics of those fields (such as text, date, or integer types). Fac-Back-OPAC uses an instance of Solr with a bibliographic schema that runs inside the Jetty application server. We then take each MARCXML record, extract the fields of interest for faceting and indexing purposes, and generate an XML document as a string to send to the Solr instance for indexing via an HTTP POST method.

Search interface: Django

Django is a popular Web application framework implementing the model-view-controller (MVC) pattern written in Python that also offers support for features including caching search results for performance, translation into different languages, and RSS feeds. The robustness and performance of the framework has been amply demonstrated by its use at the Washington Post by the original developer of Django, Adrian Holovaty. To customize the three Django templates that constitute the initial search screen, the search results page, and the item detail page, you simply edit the HTML-based

templates found in the catalog/catalog/templates/ subdirectory of Fac-Back-OPAC.

A free ride? Or a sports car at public transit prices?

Fac-Bac-OPAC's code is freely available under an open source license, but as with any technology there are other costs that must be considered. These costs include obtaining the technical expertise to provide implementation and support, hardware costs and the possible short-term sacrifice must have functionality that some proprietary ILS vendors offer.

First, you should ensure that you have personnel with experience in the technologies being used, or someone who has the time and willingness to learn. Unfortunately, Fac-Back-OPAC is not yet as simple as installing Itunes in Windows or filling out a form on a website. Installation may require someone who is comfortable performing tasks done by a systems administrator. Fac-Back-OPAC requires the systems administrator to be familiar with the following components: Python, Django, Solr, Java JDK 1.5, and a Subversion Client. The good news is that help is available. Details for the installation can be found on the Google Code project page wiki and documentation for these components is found online for free. Another great resource for us has been the Code4lib community and online forums. They are very responsive to any questions that people might need help with.

In addition to personnel expertise, Fac-Back-OPAC has computer hardware requirements. Surprisingly, for such a powerful tool, Fac-Back-OPAC isn't a resource hog. So far it has been tested on Windows and Linux, although theoretically it can be run on any operating system capable of running a recent version of the core components. The requirements for a production site will vary largely depending on several factors including the size of your collection, how much traffic your Web server will get, and what other applications your server will be running. For comparison purposes, Mike has a test install of 40,000 records running on his Windows XP workstation with 1GB RAM and a Pentium 4 2.8Ghz processor. Dan has a test install of 400,000 records running on a Linux workstation with 1.5GB of RAM and a Pentium 4 2.8Ghz, but less than half of that RAM is allocated to Fac-Back-OPAC. The caching combination of Solr and Django consistently delivers sub-second results even for large collections hosted on mid-level hardware.

One of the strengths of Fac-Back-OPAC is that it is completely independent of the ILS. While this allows for greater flexibility, it comes with a price. Because Fac-Back-OPAC only imports item records, the catalog has no link to user information, which means any related functionality that may have been provided by your ILS such as holds, payments, account status, and in some cases, item status will have to be written into the application separately. For many, including large consortia libraries that rely on such functionality, this may be a deal breaker.

The Future of Fac-Back-OPAC

With Fac-Back-OPAC the sky is the limit. While currently in its infancy Fac-Back-OPAC has a lot of room to grow. At Paul Smith's College the intention is to monitor user behaviour this fall and continue

to develop the catalog based on patron needs and search behavior. Additional features such as patron book lists, Dewey/LC subject browsing, and full-MARC display or Web 2.0 tools like tagging and patron reviews would be great enhancements. If you're interested in working on making Fac-Back-OPAC better, sign up at the Google Code project site.

SIDEBAR: code4lib spells relief

code4lib is the name of the informal community of library technology specialists who cling together for moral and technical support. While the bulk of the community's interactions are held over IRC, Email or blogs, in the spring of the last two years the code4lib community has gathered at an eponymous conference to exchange experiences and knowledge in person. If you're reading this article, you should definitely check out http://planet.code4lib.org/, http://code4lib.org/irc/, and http://code4lib.org/conference/.

SIDEBAR: The Buzz About Endeca

In January 2006, North Carolina State University (NCSU) officially unveiled their new catalog. The new catalog represented several departures from standard library practice:

- it replaced the catalog that was bundled as part of the integrated library system, but rather than being a simple off-the-shelf purchase, the catalog was developed from the ground up at NCSU
- it used search technology licensed from Endeca, a software developer and consulting firm with experience in the commercial sector but no previous experience in the library sector
- it introduced the use of facets to enable users to refine search results by narrowing search by specific subject headings, genres, authors, call number ranges, etc.

The Endeca-based catalog at NCSU (http://www.lib.ncsu.edu/endeca/) reinvigorated debate about the features that library discovery tools should offer. It also legitimized Endeca as a purveyor of search technology for libraries, and spurred open source enthusiasts to develop alternatives to the costly proprietary options.

SIDEBAR: Casey Durfee on open source catalogs

Helios and other open source OPACs represent an effort to bring the core values of librarianship to the library software world. To many patrons, our Web presence is the library. They spend far more time with our Web sites and search interfaces than they do with librarians. So it is imperative that the catalog be much better than most systems in place today if we want to save the time of the reader and make our libraries easier to use.

Open source software also embodies the simple but revolutionary idea behind libraries: that information should be open and free for anyone to use. The power of open source makes it possible to create free systems that are as good or better than the commercial products out there now, systems that can be easily modified and extended by anyone, and allows all libraries, not just the ones with enough money, to have the best possible software.

SIDEBAR: More about the Software and Websites Mentioned in this Article

- Fac-BACK-OPAC Code: http://code.google.com/p/fac-back-opac/
- Fac-BACK-OPAC at Paulsmith's College: http://library.paulsmiths.edu/catalog
- Code4lib: http://www.code4lib.org/
- NCSU's Endeca Based OPAC: http://www.lib.ncsu.edu/endeca/
- Helios OPAC: http://catalog.spl.org/catalog/about/
- Solr: http://lucene.apache.org/solr/
- Django: http://www.djangoproject.com/
- Marc4j: http://marc4j.tigris.org/
- Jython: http://jython.org/

About the authors

Dan Scott is the systems librarian for Laurentian University in Sudbury, Ontario. In addition to Fac-Back-OPAC, he is the creator and maintainer of the File_MARC PHP library, a regular contributor to the Evergreen Open-ILS project, and has contributed patches and documentation to the LibX browser extension. He is the co-author of one book (Apache Derby: Off to the Races) and occasionally writes about coffee, code, and other good things at http://coffeecode.net/. Dan holds a BA in English and Philosophy from Laurentian University and a Masters of Information Studies from the University of Toronto. He can be reached at dscott@laurentian.ca.

Mike Beccaria is the Systems Librarian for Paul Smith's College located in northern NY. He received a BA in History from S.U.N.Y Geneseo and a MLS from Southern Connecticut State University. He can be reached at mbeccaria@paulsmiths.edu.