Providing Library Reserves to Sakai using RSS

by

Bill Dueber, dueberb@umich.edu

Library Web Services Programmer, University Library, University of Michigan

and

Susan Hollar, shollar@umich.edu

Curriculum Integration Coordinator, University Library, University Michigan

October 30, 2006

Table of Contents

Introduction

As more and more data and metadata at libraries becomes available in electronic form, we are presented opportunities to expand not only the types of services we offer, but the contexts within which we frame those services. This brief paper provides an overview of one such reframing: the automated publishing of course reserves data to the course management system Sakai. We briefly look at the electronic infrastructure on college and university campuses that allows this sort of expansion of library services, and then discuss in more detail the technical underpinnings of this particular service.

Contextualizing library information

While many campus environments have been making extensive use of computer systems for years, it is only fairly recently that data within these systems has become available for use *across* technical and bureaucratic boundaries. The electronic plumbing provided by the use of standard protocols (HTTP and SOAP) and syntax (XML) offers new opportunities to link data from disparate systems and offer users views of these data centered more on their common usage and less on the organizational structures behind them.

While most library systems exploit email to expand service, “mashing up” arbitrary information from the library has been considerably more difficult due to our inability to work with library data from outside the system. This limits the library’s ability to deliver its services to online spaces such as web sites and, perhaps more importantly, campus course management systems such as Sakai and Blackboard where the possibilities for high-impact library interventions are enormous.

In the last few years, however, library systems such as the online public access catalog (OPAC) and metasearch engines have begun to appear with useful programmatic interfaces, enabling more complex use of library data from outside the hosting systems. This opens up a wide range of opportunities to take library information directly to the users.

Divorcing *The Library* from the library

The cliché of the library — a vault-like building protecting physical books and papers — is less accurate than it ever has been. Vast swaths of a library’s information are now available in electronic formats that need not be tied to the space that houses the library’s print collections. The ability to divorce the abstract concept of *The Library* (the information and services that come together to benefit patrons) from “the library” (the physical building) conjures up a new image of the library as not a point-of-service, but as a place where services are organized before being placed into the most appropriate context for the user.

The rest of this paper explores a simple instantiation of this principle: feeding course reserves information directly into a course management system. The lessons

learned — about loosely-coupled services, cooperation across university systems, and the importance of programmatic access and standard formats — are hopefully applicable to the larger issues surrounding the reframing of library resources and services in other contexts.

Overview of distributing course reserves information

Course reserves are an excellent starting point for the reframing of library resources in other contexts. They represent a set of data whose creation, maintenance, and dissemination clearly cross the boundaries between the library and the classroom. While librarians are the obvious choice to organize, maintain, and control dissemination of course reserves, these resources are provided in service of a specific learning context *outside* of the library.

Goals when distributing library data

Put succinctly: course reserves clearly *belong to* the course, and to ask users to think about them in any other way invites confusion. By placing reserves information (but not management of that information) in the course management tool, librarians maintain control over the dissemination of the materials required by licensing agreements and logistics using they systems with which they are familiar, and students know that the course management tool will provide the highly-contextualized, course-specific data it is designed to organized.

When distributing library data to other systems, a few goals are obvious:

- **Automation**. Once the initial investment in infrastructure is made, the process must not require a lot of work for either the librarians or the instructors.
- **Use of standard data formats**. Creating a service that produces a data format unique to a single consuming application restricts possible serendipitous uses of that data in the future by other, perhaps currently unknown systems.
- **Live updates**. A simple “export” of a course reserves list (or any other data) severely restricts its utility. For example, a reserves list that doesn’t accurately reflect the underlying live data at the library — addition of new items, current availability of items, etc. — is not a trustworthy source and will soon be ignored by the users.

Infrastructure needed to distribute course reserves data

The generic goals listed above point to a few requirements in the technical infrastructure and procedures of an institution that wishes to distribute, in our case, course reserves information.

First, and most obviously, *reserves data must be kept in a system that can be usefully programmed against*. While a large number of library systems don’t allow such programmatic access, more and more vendors are offering such interfaces, either as part of the product or as an optional, cost-added component. Then it’s a matter of determining whether the interface provides the needed functionality, and whether the necessary queries can be done quickly and cheaply enough that the level of service for all users is acceptable.

Second, *courses must be uniquely identifiable* either through the use of a single unique id or through some combination of course data (e.g., year / semester / department / course / days / time).

Third, *both the course management system and the reserves system must have data that can be resolved to a course's unique id.* Both systems may actually use the unique id directly, or one or both may need to coerce internal data into the unique id (possibly using yet a third system).

Finally, *the course management system must be able to consume data from an outside source*, or the whole exercise is pointless.

Distributing course reserves via RSS at the University of Michigan

The University Library at the University of Michigan set out to distribute course reserves information to course sites in CTools, the local implementation of the Sakai course management tool, with the necessary infrastructure *almost* already in place.

Existing infrastructure

Much of the technological and information infrastructure necessary for this project was already in place and ready to be combined into the new service:

- Course reserves information is stored in the library catalog, an implementation of Ex Libris' *Aleph* system. Aleph exposes an XML-based programmatic interface called the *X-Server* that accepts queries in URLs or as XML and returns records in the Library of Congress' standard MARC-XML format.
- CTools (a locally-branded implementation of Sakai), our course management system, which is automatically populated with information such as the year, semester, department, section, etc. of each course when the course sites are created.
- A Registrar Service that tracks (among other things) which students are in which courses, stores the unique ids of those courses and provides an HTTP-based translation service from basic course information (like that available in Sakai) to these unique ids.

It's easy to see that we have almost everything we need: a reserves system that exposes a programmer's API and returns data in a neutral, well-specified format; a course management system that can uniquely identify a course, and a service that serves to provide unique course identifiers based on other, more easily-available information.

The missing pieces

There are only two parts of the puzzle we still need:

1. A program to glue all these disparate services together, doing data translation as needed
2. A way to display these data once they get to CTools

The former is a relatively simple matter of creating a program that can talk to both the X-Server and the Registrar service and can deal with the XML formats being used (MARC-XML and RSS). The University of Michigan implementation was written in PHP, but any general-purpose programming language with support for XML would have worked just as well.

Oddly enough, it is the latter issue that causes the most trouble. Writing a "tool" for CTools (Sakai) is a relatively time-consuming process. After some investigation, it was decided that a modified version of the Sakai "News" tool — an RSS display engine — could be developed in a fraction of the time needed to build a new display plugin. Not only is the existing tool optimized to display potentially-changing data, but RSS is a well-understood (if underspecified) open standard, with potentially wide application beyond this single use in CTools. Only slight modifications were necessary, to override terminology (e.g., "Read full story") that would be confusing or irrelevant when dealing with course reserves instead of syndicated news stories. For a description of specific changes made to the News tool, see Appendix 1.

Based on these needs, the University produced a modified RSS tool called the *Library Reserves Tool*, and a program to broker all the transactions called the *RSS Builder*.

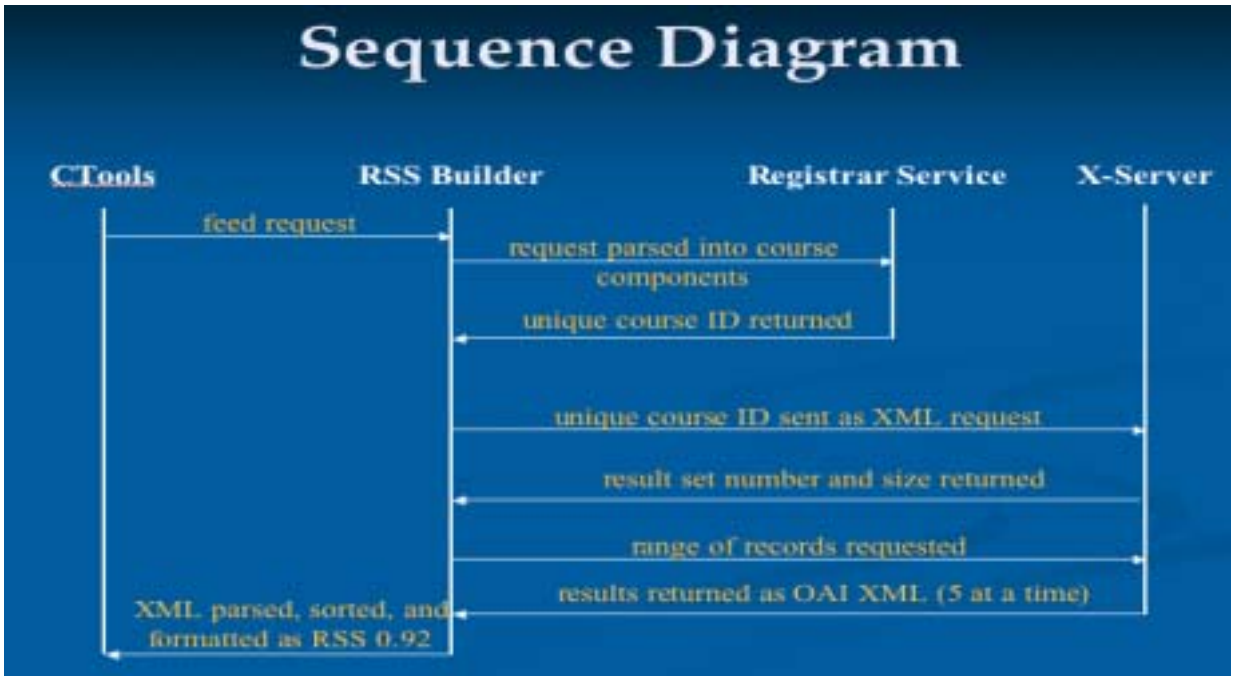

Figure 1: Overview of the dataflow for RSS Reserves

Putting it all together

The data flow based on these systems is fairly straightforward (see Figure 1):

1. Librarians tag all reserves for a course with the appropriate unique course id.
2. CTools makes a *feed request* (via URL) for reserves data, using information about the course it knows.

3. The RSS Builder receives the request, extracts the course information, and asks the Registrar Service for the associated unique course id.
4. Using the unique course id, the RSS Builder queries the Aleph X-Server for the correct reserves.
5. The X-Server performs the query and returns session information that the RSS Builder uses to retrieve results formatted as MARC-XML.
6. The RSS Builder extracts the relevant information from the full MARC-XML record and creates an RSS feed from it.
7. CTools gets the RSS feed in the Library Tool and displays it using the modified News tool.

Benefits and shortcomings

The reframing of reserves lists as course-specific data inside of Sakai fulfills all three of the criteria discussed earlier. The process is automated, relying only on data already available to Sakai and in the library system that tracks reserve items. The data is distributed via RSS, a standard format that could easily be re-used in a different system or transformed using standard tools. And caching is easily implemented by the RSS Builder, allowing the librarians to balance keeping the information up-to-date with real-world concerns about efficiency and performance.

This system is also an excellent example of the benefits of reframing library services instead of exporting those services. Librarians continue to use their specialized reserves software, and authorization/authentication issues (e.g., proxying) are handled by library software external to the Sakai system and require no advanced integration on that front.

The shortcomings of the current implementation are obvious but worth exploring. First, the distributed nature of the process exposes many possible points of failure and several non-overlapping areas of responsibility. If a particular course shows no reserves, is it because Sakai provided the wrong course information? Are the reserve records tagged with the wrong information (or even in the system as of yet)? Is the RSS builder having trouble parsing the file, or is the Sakai Reserves tool having trouble reading what's been produced? Or did the OPAC vendor change the XML export format is a subtle but problematic way? Tracking down errors in this environment is both difficult and tedious and can result in extended games of email- or phone-tag.

Second, this process represents a very low level of integration with existing Sakai tools, most notably the Resources tool. Having some readings/resources stored as first-class objects within the Sakai environment while others are hosted by the library and only linked to via the Library Reserves Tool is potentially confusing and certainly not optimal.

Third, RSS is a relatively sparse data format, with generic RSS consumers unlikely to support the extended metadata libraries are able to provide and equally unlikely to provide a default presentation optimized for this sort of data. It may be ultimately more useful to provide a simple pre-formatted HTML file (for inclusion inline or in an iframe), or (as Yale has done) to produce valid XML and produce the desired results with an XSLT transformation.

Finally, as hinted at above, this only scratches the surface of potential integration with a hosting environment like Sakai. The connection here is simple and one-way, the data static once it reaches Sakai and unavailable for further processing/annotating by the user. Research exploring a closer integration between library resources and course management tools in general (and Sakai in particular) is ongoing under the auspices of the Sakaibrary project, referenced below.

For more information

Sakai

Sakai is a free and open source online collaboration and learning environment. Many users of Sakai deploy it to support teaching and learning, ad hoc group collaboration, support for portfolios and research collaboration. CTools is a University of Michigan-branded installation of Sakai.

`http://sakaiproject.org/`

The Sakaibrary Project

The Sakaibrary project is a collaboration between the University of Michigan and Indiana University to develop open source software tools to integrate access to library licensed digital content within the Sakai collaboration and learning environment.

- Main Sakaibrary page:
  `http://www.dlib.indiana.edu/projects/sakai/`
- Sakaibrary Project wiki:
  `http://bugs.sakaiproject.org/confluence/display/SLIB/Home`

Standards referenced

This project is built using standard protocols and formats, most notably:

- **RSS** (Rich Site Summary), an XML dialect primarily used to syndicate news content, repurposed here to deliver reserves data.
  `http://en.wikipedia.org/wiki/RSS_(file_format)`
- **MARC-XML** is a Library of Congress standard that describes an XML rendering of the MARC standard.
  `http://www.loc.gov/standards/marcxml/`

Appendix 1: Modifications to the Sakai News Tool

Text provided by Richard Ellis, CTools Developer

1. Determined how a URL sent to Library Reserves should be formatted (e.g., questions of cross-listed courses, multiple sections, etc.).

2. Registered a new tool for CTools course sites called Library Reserves.

3. Registered a default URL for the tool:
http://www.lib.umich.edu/r/rss-reserves/rss.php?

4. Inherited the functionality of the CTools News tool by creating a new Java class, ReservesAction.java, that extends NewsAction.java.

5. Set the default Library Reserves URL to the URL in #3.

6. If the CTools site has a course provider id, parsed the course provider id, form a Course Reservation URL and make that the effective URL.

```
  {
   //e.g., http://www.lib.umich.edu/r/rss-reserves/rss.php?

catalog_nbr=468&section=001&semester=2&subject=JUDAIC&year=2004
    url = courseReserveUrl + "catalog_nbr=" + firstCatalogNbr +
     "&section=" + firstSection + "&semester=" + firstSemester +
     "&subject=" + firstSubject + "&year=" + firstYear;
  }
  else
  {
   // splash page for Course Reserve
   url = courseReserveUrl +
"catalog_nbr=&section=&semester=&subject=&year=";
  }
```

6. Customized the Web page language by setting properties of the Java ResourceBundle used to fill in variables in the Web page templates, e.g. cus.cus = News Options changed to cus.cus = Library Reserves Options.

7. Debugged the Library Reserves RSS feed and its parsing by the CTools News RSS feed parser.

8. Debugged the RSS feed and its parsing when our locally-written CTools News parser was replaced by the open source Rome parser.

9. When there are new CTools releases, test the tool to make sure it's functioning properly.