Implementing SRU in Perl

As a part of a sponsored National Science Foundation (NSF) grant called Ockham, the University Libraries of Notre Dame implemented a set of SRU modules and scripts written in Perl. This text describes this process in more detail.

Ockham

Ockham is a sponsored NSF Digital Library grant with co-PI's at Emory University, Virginia Tech, Oregon State University, and the University of Notre Dame. The primary purpose of the grant is/was to explore and implement programatic methods of better integrating NSF digital library content into traditional library settings through the use of "light-weight" protocols. In general, light-weight protocols were characterized as non-proprietary, modular in design, and Web Services-based methods for data exchange and display. We used the name Ockham to denote our desire to be not overly complicated. Our example implementations included, among other things, a registry of digital library collections and services, a Find More Like This One service, and an Alerting service. For more information about Ockham see http://ockkham.org/.

Alerting service

Notre Dame was charged with implementing the Alerting service. This service, analogous to a current awareness service, is intended to provide the means of learning "What's new?" from the NSF Digital Library. In a nutshell, this is how it works:

1. The last thirty days of metadata content available from the NSF OAI Repository is harvested and saved locally.

2. The content is indexed.

3. Searches against the index are returned as HTML pages, email messages, or RSS feeds.

4. Everyday content is harvested from the Repository that is one day old.

5. Everyday content older than thirty days is deleted from the local cache.

6. Go to Step #2.

Through such an algorithm, the user is expected to articulate one or more searches against the index and then save useful queries as RSS feeds in their RSS news reader. Using this approach the user should be able to read their news feeds on a daily basis as view an ever-changing set of results.

Implementation

With the help of a very able and expert Perl programmer, the Alerting service was implemented through a set of object oriented Perl modules and accompanying scripts:

- Ockham::Alert supports the harvesting/caching/indexing process. Given a set of one or more OAI URL's and associated dates, this module allows the

developer to harvest OAI content, save it to a local cache (relational database), and dump the data from the cache to an indexer. Since traditional library content also manifests itself as MARC data, the module supports the incorporation of this data into the cache as well.

- SRU::Request and SRU::Response facilitate the implementation of an SRU server. The Request module is used to read the SRU operation parameter and create an associated Request object. Based on the type of the Request object, the SRU::Response module initializes and builds Response objects. The result of this build process is an XML stream in compliance with the SRU schema.

- CQL-Parser provides the ability to read CQL statements and convert them into queries supported by an underlying indexer. The indexer used in this implementation is swish-e. CQL-Parser is essentially a port of Mike Taylor and IndexData's cql-java package, and we are appreciative of their support.

Links to source code and our implementation are available at http://alert.ockham.org/.

Discussion

The implementation more or less does what it was designed to do. For example, through a cron job the content of the cache is successfully updated and indexed on a daily basis. Freetext, rudimentary Boolean, and fielded queries are accurately supported by the SRU client and server. Since search results are returned to the client as XML, it is easy to transform the results into HTML, email messages, or RSS news feeds.

The downside includes problems with the indexer. Swish-e only supports 7-bit characters and consequently non-ASCII characters are indexed and displayed poorly. Just as much of a problem is the regular harvesting of the data. While harvests are completed smoothly, new items to an OAI repository were not necessarily written recently. Instead, new items in an OAI repository are defined as items recently added. Things written in 1996 are not necessarily new, but they are returned in harvests because they are new to the repository. This can be confusing and frustrating to users.

Thus, we consider our implementation a qualified success. It is light-weight, standards-compliant, and non-proprietary. Libraries or other content providers could take the tools we have created and apply them to their own settings for their own purposes. A different indexer could be used, and an institution could make an effort to only add truly new items to their repository.

Eric Lease Morgan
University Libraries of Notre Dame

June 14, 2005