**Blacklight: Leveraging a Next Generation Discovery**

**Application as a Flexible, Extensible, Repository Front End**

**PAPER PROPOSAL for Open Repositories 2010**

Blacklight is an open source, next generation discovery application. Originally developed to serve as an overarching “discovery layer” for libraries, its design and engineering give it the necessary feature set and flexibility to also serve as a repository interface, capable of fronting content of any kind, local or remote. With a rich set of search, browse and view functions, Blacklight’s look, features and behaviors can be readily configured to meet local needs “out of the box.” As an application with a modular architecture, it provides a framework capable of supporting additional libraries and widgets that extend Blacklight’s capabilities beyond resource discovery. And as a vibrant open source project integrating enhancements and development from more than a dozen institutions, Blacklight is becoming a proven platform for content discovery and access, agnostic of underlying systems or repositories.

**Blacklight Features and Functions**

Blacklight has the basic, desired functions one expects in a modern search tool: relevance ranked search results, presented in a faceted browser, all in a streamlined and intuitive interface. The power of faceted browsers is well known; they not only expose the distribution of a collection or any result set along pre-defined axes (format, collection, location, author, subject, etc.), but the facets serve as live links, enabling searchers to refine their search and create complex queries with a simple series of mouse clicks.

Blacklight also includes a number of elegant enhancements, such as facet-aware advanced search, easily manipulated results sets (sorting by relevance, date, alphabetically, etc.), ability to export results in a number of ways, including RSS feeds, citation export, email and SMS. It also supports personalization, enabling users to save searches, bookmarks, and tags.

While these features are relatively standard for any type of academic or library search interface, whether for a catalog or a repository, Blacklight has four key functions that make it particularly useful as a repository front-end:

* *More than MARC.* While the Blacklight interface fulfills the same functions as a library catalog, and fully exploits the richness of library metadata records, it is not constrained to supporting search across MARC records alone. This is particularly useful for repositories, where few if any digital objects will have a MARC record, unless it is for the analog form of a digitized item.
* *Object-specific behaviors.* Blacklight code supports the use of different views for different types of content. Thus, for example, the detailed record view (or splash page) for a document can be quite different from the splash page for a still image, and different still from a media asset. Blacklight chooses the correct view based on object type (or any other criteria), to display that object in its appropriate context (metadata fields, thumbnail image, links, embedded players, etc.)
* *Tailored views.* Blacklight can simultaneously present customized search and browse interfaces onto the same repository. This can be used to scope search results to present tailored application functionality (e.g., different facets or search features than in the default view, such as showing performer and instruments for a music view, or school, department and program for theses and dissertations); to limit searches to a pre-defined subset of items (e.g., a single collection in a repository); to present different branding options (e.g., a header crediting the source in an aggregated collection); or all of the above. Critically, these tailored views can all be provided through the same instance of Blacklight, and are determined simply based on URL.
* *Rich Defaults and Over-rides.* A central premise of Blacklight’s engineering has been to provide rich, default behavior, while supporting options for local customization without extensive reengineering. At the simplest level, this means that, with an underlying Solr index in place, installing a fully functional instance of Blacklight (including even customizing its appearance with a cascading style sheet) is an afternoon’s worth of work. At a more sophisticated level, local sites can turn off or replace the application’s default behaviors with their own code with a simple one line over-ride command.

**Blacklight Users and Uses**

Led by the University of Virginia and Stanford University, large academic research libraries form the seed of Blacklight’s community of users and contributors; over the last year, there has also been expansive adoption in the repository sphere. The project is an example of successful, community supported open source software, with dozens of adopting institutions, and an active committers group of roughly ten programmers from a half dozen different sites.

Exemplar Blacklight sites include those using it as…

* a next generation library catalog (Virginia’s VirgoBeta and Stanford’s SearchWorks). VirgoBeta includes both Virginia’s full set of MARC, plus a number of digital manuscript and image collections being served from a Fedora repository.
* an institutional repository front-end (University of Hull). The IR at the University of Hull has been available now for almost two years running a Fedora repository underneath a Muradora user interface. A new interface is being launched for the coming academic year which uses Blacklight for repository discovery. Further, Hull is investigating combining its Solr repository index with a Solr index of its libraries’ MARC records potentially to provide a single, Blacklight-driven point of discovery.
* a union catalog (University of Wisconsin’s Forward). With more than 8 million records, UW is running the largest known Blacklight instance, with tailored views for each its fourteen individual campus collections.
* a search and display interface for a digital archive (Stanford’s SALT). SALT (the Self-Archiving Legacy Toolkit) hosts digital objects from the papers of notable scientists, intellectual and artists; a modified version of Blacklight provides an EAD-like view onto these collections. Fedora serves as SALT’s back-end repository.
* an index of digitized medieval manuscripts (Stanford’s DMS Index). To facilitate across and within digitized medieval manuscripts, Stanford has created a Blacklight instance of these resources, pulling records from a half-dozen different institutions, each with their own (or no) local repository.
* a repository administrative reporting front-end. Several institutions (Stanford, Hull, Virginia) are experimenting with using Blacklight as an administrative interface to their Fedora repositories; with its high-performance solr index and faceted browsing features, it promises to provide a rich, user-friendly reporting and administrative search interface.
* An aggregated discovery interface for disparate digital collections (Northwest Digital Archives). The Northwest Digital Archives uses Blacklight to provide a unified discovery front end for heterogeneous collections stored in ContentDM, Archivists’ Toolkit, and a variety of legacy formats.

**Blacklight Technical Architecture**

Blacklight has two main components: an underlying Solr-Lucene index, which serves as its fast-access data store, and a Ruby-on-Rails (RoR) application, which provides the system logic and user interface. Ruby on Rails is a language and framework that supports rapid development of web-based applications. It has a number of features that make it well suited as a language choice for Blacklight. It is quick to code in, and quick to deploy; its separation of logic from data from presentation makes it supportable over the longterm; it is easy to learn; and it has an excellent set of accompanying testing tools. More than 90% of Blacklight’s code base is covered by automated tests, providing a high level of assurance in the project’s changing code base both for continuous integration testing and regression testing.

Ruby-on-Rails’ modular architecture also makes it straightforward to add new functions to Blacklight by including new libraries (or “Ruby gems”) and code in any deployed application. This capability, and the Ruby community’s large library of gems, effectively makes Blacklight an application platform, into which additional modules can be integrated. These modules can extend its functionality beyond its core discovery features, to include such services as djatoka image streaming, TEI viewers, OAI-PMH data providers, or third party authorization modules.

The Solr index is in many ways the key to the system; not only does it serve as the data store, but also the logic and effort of mapping data sources into the appropriate Solr fields enables the Blacklight application to search and facet the data with straightforward (and common) algorithms. Solr brings a number of technical advantages as the data store—it is highly scalable, high performance, and easily distributed across multiple instances for load-balancing and fault-tolerance. Solr also serves as an abstraction layer between the search application (Blacklight) and the underlying repository(ies) and data structures; this isolates the search application from changes in the repository technology. It allows Solr to function as a normalization source and integration point for data from different systems with different data models. Once the data from these different schemas are mapped into a common Solr schema, they become visible and accessible in a single application—this is the case with Stanford’s DMS Index, where each contributing repository has its own unique data model. Finally, the latest versions of Solr also support searching across indexes with different schemas, facilitating integration of different data sources (e.g., federated searching of two different repositories’ Solr index in real time, or a simultaneous search of a metadata and a full text index at the same time).

**Summary**

This presentation will demonstrate the broad-based utility of Blacklight, including its key features, its use in different contexts, and how it integrates with different repositories to provide a rich and ready-made discovery application.