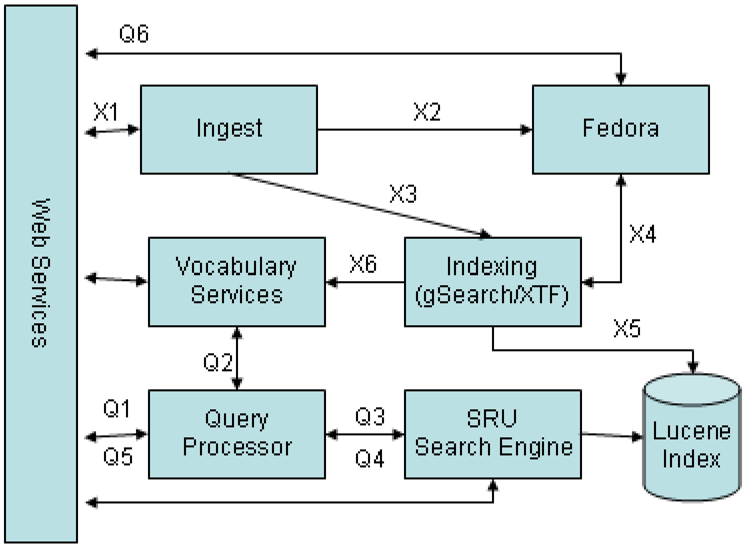
Search

We are beginning to develop a system for searching through content in the repository. This system is based on a combination of Fedora gSearch, Lucene, XTF, and the Java Query Analyzer from the Cushman collection searching system.

See also:

- Searching System Requirements
- Libraries Search System
- Federated Search
- Search System Indexing
- SRU Server
- Making a Collection Searchable

Basic search architecture



Pieces:

- **Ingest** - Our Ingest Tool.
- **Fedora** - The core Fedora repository.
- **Vocabulary Service** - A general-purpose utility for managing controlled vocabularies, some of which will have "full" thesaurus relationships. We need both a place to store the full vocabularies (e.g., all of LCSH), and a place to note which terms are actually used in each collection, to facilitate faceted browsing. Must be Z39.19 compliant. Our initial implementation will be based on Oracle's Thesaurus services.
- **Indexing** - A utility that transform a Fedora object into a set of fields to be indexed by the search engine. It is likely that the indexing system will be based on some combination of Feodora gSearch and XTF, though it is still unclear whether these will be two completely separate indexing processes, or whether they will be combined in some way.
- **Lucene Index** - A set of index files in Lucene format. More information about the index.
- **Query** - Java Query Analyzer. A pre-processor that translates queries from our "user input" format into SRU queries.
- **Search Engine** - A search engine toolkit, based on the OCLC SRW server, using the Lucene indices created by gSearch and XTF.

Indexing (X):

1. An external tool/user sends an object to the ingestion system.
2. The ingestion system submits the object to Fedora (or modifies an existing object).
3. The ingestion system notifies the indexer that the new object needs to be indexed.
4. The indexer retrieves the object's metadata and/or textual content from Fedora.
5. The indexer parses the data and submits fielded entries to the search engine.
6. The indexer submits updated information to the vocabulary service (but it may have to request data from Fedora or the search engine to accomplish this).

Querying (Q):

1. The user (or external webapp) submits a query, in "IU form". We would like to eventually allow external users to submit SRU queries.
2. The query processor optionally uses a thesaurus from the vocabulary service to expand the query.
3. The query processor translates the query into CQL format, and executes an SRW search request that includes this query.
4. The search engine returns a list of results (and a continuation token).
    a. The search engine optionally uses a thesaurus from the vocabulary service to expand the query.
5. The query processor returns the results to the user/webapp. Note that if external programs want to execute SRW queries, they can speak to the search engine directly.
6. The user/webapp either displays the result page, or asks Fedora for the relevant objects to perform any needed rendering.

Browsing

When a document is indexed, browsing information needs to be pre-computed, including:

- Number of items that are accessible by selecting any level in the browse hierarchy.
- The list of available subject headings (or other controlled vocabulary terms) for use in faceted searches needs to be updated. (This list may or may not be used as the basis for a browse hierarchy.)

How does XTF fit in?

Peter Murray has written a series of blog articles on combining XTF with Fedora.

The current vision

- Indexing: All data that is created/updated in Fedora will be processed by the Ingest tool. The Ingest tool will forward all objects to the gSearch indexer for basic bibliographic indexing. Objects that have textual content will also be sent through the XTF indexer. While it may be possible in the future to combine the XTF index with the bibliographic index, initially they will be stored separately.
- Searching: When a query addresses textual content, the SRW search engine will search the bibliographic index, the XTF index, or both as necessary. We have two options for querying XTF, and we will need to evaluate which is the best:
    - Have the SRU server generate a Lucene query that goes directly against the XTF Lucene index.
    - Have the SRU server generate an (XML) XTF query, which is sent to the XTF query engine.
- Search results: We must determine the most appropriate results format. Will the default XTF return format work?
- Display (images): We will use MetsNav for paged images.
- Display (text): For display of text, we will write Fedora disseminators that invoke XTF's dynaXML. We may need a wrapper around dynaXML if we want additional features, such as links that properly return to search results. For each page of text (`<pb/>`), the rendered version will include an HTML anchor that can be used to reference this point, as well as an HTML link to the image version of the page, if an image version exists.

One possible problem with this approach is combining searches across the XTF and non-XTF indices. Since they're both based on Lucene, this should be possible, but it's not clear how much work this would take.

We cannot always break TEI into "pages", so we will need to maintain separate page structure in the METS document, coordinating it with the TEI as much as possible.

Notes about XTF

- XTF is optimized for processing textual content, rather than fielded metadata.
- We can use XTF for the textual content, and tie directly into Lucene for more precise control of metadata indexing.
- Lazy Trees can provide a speedup for retrieving/rendering portions of a text document.
- We will probably need to modify many pieces of XTF to work on URLs instead of a filesystem.

The XTF **textIndexer** is a command-line tool that initiates Lucene indexing of files in a given directory. We may need to modify it to use URLs instead of files, although the ingest tool could write a temporary file for indexing purposes. It is possible to use the textIndexer without using the query processor.

The XTF query processor, **crossQuery**:

- isn't incredibly powerful. If we use it, we will need to expand it.
- has its own query language, but it also supports SRU. It appears the SRU portion needs some debugging, or reconfiguration, since the sample query doesn't return correct results. The documentation states that the SRU interface will **only** return DC records at this time, so that portion would need some extra work.
- It is unclear how easy it would be to attach a thesaurus to crossQuery.

The XTF rendering system, **dynaXML**, is useful for rendering paged text. We will use it along with METSNav for paged images. We want to retain the dynaXML feature that allows clicks from search results to go directly to the position of the hit in a result page. To do this, dynaXML will need to have direct access to the XTF Lucene index.

Possible Thesaurus tools

- Eviada thesaurus tool
- Oracle
- Protege
- THE
- Lucene WordNet extension (sandbox version)
- Other software
- Our SRU server, modified to support Zthes

Current work

We have an SRU front end to the Lucene database created by gSearch. It is based on the OCLC SRU implementation.

For more information, see SRU Server.

Adding a new collection to the search system

Follow these instructions.

Documents that may affect design of the searching system

- Research Information Network report
- Melvyl project report

Open questions

1. Is it possible to build a thesaurus into Lucene, or will we need an external database/tool to handle this? Regardless, we want query expansion to happen as close to Lucene as possible, to prevent problems caused by sending expanded queries through URLs.
2. How should we structure the indexes to make re-indexing less painful? (Currently, when something changes, the entire repository must be re-indexed.) Should we store each collection in a separate index? Or keep a single master index and develop tools for selectively updating content?
3. How do we implement a search across multiple finding aids?
4. How can we combine searches across XTF indexes and bibliographic indexes? Is it possible to use the a single Lucene query and a MultiSearcher? Or will we have to manually combine results from searches that have both text and bibliographic features?
5. Should we store the transformed XTF-form TEI documents in Fedora, or only store the original documents in Fedora and provide a dissemination for exporting them to the XTF directory?
6. How best to combine MetsNav and dynaXML? Can we write a single webapp that wraps both of them and coordinates their interaction?
7. FedoraGSearch can support multiple indexes. Will this make our searching system easier to implement?
8. FedoraGSearch supports highlighting of matched terms in result sets. Is this functionality better than the native Lucene support? Do we need result highlighting in non-textual (non-XTF) searches?