

Leveraging eXist-db for Efficient TEI Document Management

Kyle Morgan
knmorgan@vt.edu

Kyle Schutt
kschutt@vt.edu

Final project report for CS 5604, Fall 2012 (Project Byron)

In collaboration with Professor David Radcliffe,
Purdum Lindblad, and Keith Battleson

Department of Computer Science
Virginia Tech
11 December 2012

Contents

| | | |
|---|--|----|
| 1 | Project Details | 1 |
| | 1.1 Overview | 1 |
| | 1.2 Management | 1 |
| | 1.3 Challenges | 2 |
| | 1.4 Solutions | 2 |
| | 1.5 Performance Metrics | 2 |
| | 1.6 Development | 3 |
| | 1.7 Future Deliverables | 3 |
| 2 | User's Manual | 4 |
| | 2.1 Overview | 4 |
| | 2.2 Screenshots | 4 |
| 3 | Developer's Manual | 6 |
| | 3.1 LBT Overview | 6 |
| | 3.2 TEI | 6 |
| | 3.3 XSL and XSLT | 7 |
| | 3.4 Current Architecture | 7 |
| | 3.5 Proposed Architecture | 8 |
| 4 | Technologies | 9 |
| | 4.1 eXist-db with Lucene | 9 |
| | 4.2 YAHOO! Connection Manager and jQuery | 10 |
| | 4.3 Client-side Templating (HTML/CSS) | 10 |
| | 4.4 XQuery | 11 |
| | 4.5 Example Migration | 11 |
| 5 | Installation and Documentation | 13 |
| | 5.1 eXist-db Installation | 13 |
| | 5.2 eXist-db Notes | 14 |
| 6 | VTechWorks Inventory | 14 |
| 7 | Contacts | 15 |

Abstract

Professor David Radcliffe has created Lord Byron and his Times (LBT), a large digital archive of works surrounding Lord Byron and his contemporaries. The original website was unusable slow due to the expensive XSLT transformations and XPath queries being applied to TEI documents. By removing the reliance on XSL and using eXist-db and XQuery, while relying on ubiquitous and well-documented CSS for client-side styling, we have been able to increase performance dramatically without sacrificing features or functionality. In this paper, we go over an overview of the project, including challenges and potential solutions, and performance metrics of the original system in section 1. Section 2 contains a user's manual detailing difference between the old and proposed systems. Section 3 contains a developer's manual, which contains overviews of various technologies that were being used in the system designed by Professor Radcliffe. The fourth section describes technologies relevant to the proposed system. Finally, documentation and installation instructions are given in the fifth section. The rest of the paper contains a VTechWorks inventory, contacts for everyone involved with the LBT project, and references.

1 Project Details

This section outlines the details of the project including a brief description of the project's contract between the stakeholder and the developers. This section aims to provide high-level managerial information on the project's scope, challenges, solutions, and future deliverables.

1.1 Overview

The project proposed by Professor David Radcliffe was to create a new information server for the current Lord Byron and His Times (LBT)^[18] system, with a focus on an architectural design that can accommodate an increasing number of TEI-XML^[10] documents in the repository. The new system must be capable of indexing TEI-XML documents for quick and effective retrieval of interconnected documents and persons. The system must be scalable and flexible to accommodate new indexes or documents, and updates to existing documents.

The main goals of the project are broken down into two categories: performance metrics and development. In order to determine a proper architecture recommendation, a performance analysis of the former system was necessary to identify any weaknesses. The second goal was development, specifically the development of a new XML-based database system with indexing, full-text searching modules, and proper templating in the browser. Later in this document, each goal will be discussed in more detail.

The overall architectural design envisioned by Radcliffe was one focused on increased performance, scalability, and flexibility. By focusing on the goals mentioned above, this project addresses each of these concerns in turn to provide an enhanced experience for Radcliffe's clients. The remainder of this document aims to detail these goals, the methods used, and future deliverables for the project.

1.2 Management

The management team involves three types of personnel: stakeholders, software engineers, and collaborators. The primary and solitary stakeholder of the project is Professor David Radcliffe, curator of LBT under the Center for Applied Technologies in the Humanities (CATH)^[17] at the Virginia Polytechnic Institute and State University (VT). The software engineers are Kyle Morgan and Kyle Schutt, both members of Dr. Edward Fox's CS5604: Information Storage and Retrieval class in the Fall of 2012. Dr. Fox, Purdom Lindblad, and Keith Battleson are the main collaborators who provided guidance in the areas of information design, architectural design, and encouragement.

We would like to thank all of those involved for their support throughout this project.

1.3 Challenges

The two main challenges of our project was both understanding and quantifying the relationships between the current indexes and their corresponding documents, and properly indexing existing documents such that the efficiency of Lucene and eXist-db^[1] is maximized. Some of the current indexes are redundant and bloated with unnecessary text; these indexes needed to be identified and minimized before being re-indexed into eXist. This re-indexing and reclassification of existing indexes also required changes in the underlying document retrieval system that uses XQuery^[9] instead of XPath.

1.4 Solutions

In order to address these challenges, we focused on maintaining a working knowledge of the current indexes and focused on modifying these indexes prior to implementing the retrieval system. This reduced the maintenance and development costs associated with refactoring existing code. Indexing in Lucene and eXist is fairly consistent for the majority of time, but it is possible to cause a massive increase in the index size when the definition is incorrect. The failure to index properly caused a large amount of downstream performance degradation when fetching documents. Therefore, the index definition currently used has been tuned over several iterations to the point where we feel that it is quite stable.

1.5 Performance Metrics

Prior to analysis, there was much speculation over which parts of the system were causing performance degradation, such as, the TEI-XML documents, XML indexes, or the XSLT Processor. After performing analyses and profiling on a subset of large TEI-XML documents and XSL documents, it was determined that the major bottleneck for processing time was within the XSL documents and, therefore, the XSLT Processor. Additionally, XSLT processing is a synchronous, blocking function that causes the browser to wait until it is complete. This function provides little information to the user about its current action or when it will be completed. This gives the impression to the user that the website itself is slow, and may be slow to the point that it is unusable (even if it is working properly). In Figure 4, we can see an example of a long-running function of fetching data from a single document. In this case, we are fetching data from a very large XML document in a very large XSL document which has already been tuned. In the modern era of high-speed Internet connections, a response of more than ten seconds from a web-server is unacceptable. With this in mind, we aimed to tackle this problem on two fronts: minimize the time it takes to display some content on the web-page, and minimize the time it takes to load all the data asynchronously.

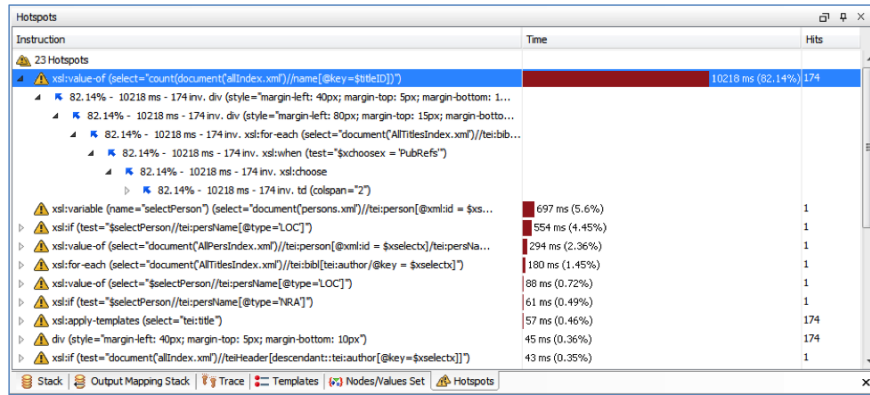


Figure 1: Profiling example of an XSL document using oXygen^[7]

1.6 Development

With the results of the performance metrics in mind, we were able to focus on specific areas that required tuning. Re-indexing and substituting a flat-file system of an XML database system was not going to solve all the issues in performance. While an XML database system provides certain improvements in terms of caching and in-memory fetching for common documents, it does not address the problem of transferring large amount of data to a client over the web. Therefore, we implemented an asynchronous batch-loading system that stores data within an in-memory session variable on the server. Additionally, the client-side template system fetches the data asynchronously from the server. As the client receives data, it is automatically displayed on the client's browser such that they can begin exploring the site without having to wait for the entire page to load. While they are browsing, the system will continue to load new data that it has fetched.

1.7 Future Deliverables

This section aims to detail future deliverables for the LBT project with respect to migration, maintenance, and future development. The major deliverables are listed below, and further discussion of each deliverable is available in Developer's Manual.

1. Monograph migration – migrate the display of documents to the new system.
2. Styling simplification – eliminated hardcoded styles in the TEI-XML documents by using the defined CSS^[16] classes.
3. Index consolidation – eliminate redundant and unused index by creating a single index per category; i.e., a person and document index.

2 User's Manual

This user's manual aims to provide an overview of the current LBT website.

2.1 Overview

The former LBT system is fairly straight-forward website with a navigation bar on the left-side that is context sensitive, and a content pane that provides details of the currently selected navigation item. The current LBT system aims to mimic the look-and-feel of the former to provide a consistent and unnoticeable transition for users. There are minor changes in the display, especially for batch-loading large documents.

2.2 Screenshots

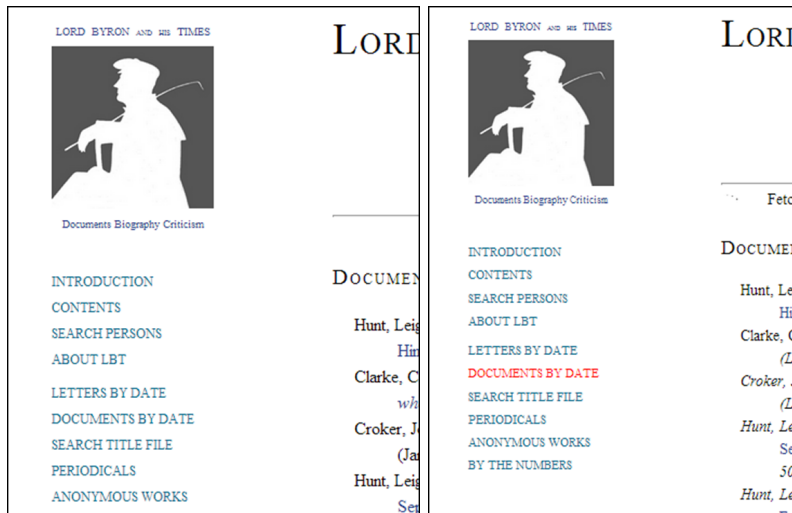



Figure 2: Before and after, respectively, of the Project Byron website. Note the similarity in styling.

LORD BYRON AND HIS TIMES



Documents Biography Criticism

SIR WALTER SCOTT, BARONET

(1771-1832)

Fetching page 9 of 29

NAME AUTHORITIES:
LOC: Scott, Walter, Sir, 1771-1832
B/BAP: 1771-08-15
NRA: Scott, Sir Walter (1771-1832), 1st Baronet poet and novelist
DNB: Scott, Sir Walter (1771-1832), 1st Baronet poet and novelist
DIED: 1832-09-21
LBT ID: WaScott **VIAF ID:** 95207079 **LOC ID:** n78095541
SOURCES: DNB; France, Companion to Charles Lamb (1983); Virtual International Authority File; thePeerage.com; LOC Name Authority File

Scottish poet, novelist, antiquary, biographer, editor, and sheriff.

CORRESPONDENCE OF SIR WALTER SCOTT, BARONET:
 Walter Scott to William Clerk, [1788 c.?] in *Memoirs of the Life of Sir Walter Scott, Bart. 7 vols. (Robert Cadell, 1827)*

CC BY NC ND
 Licensed under a Creative Commons Attribution

Figure 3: Batch loading the Correspondence to and from Sir Walter Scott.

LORD BYRON AND HIS TIMES

Byron

SUBMIT

Select from the alphabetical list above, or perform a string search (whole or partial words): forename and surname; forename, forename, surname; or forename, surname, and title, etc. Searches are cAsE Sensitive.

RESULTS: 38

1. Bettesworth, George Edmund Byron (1780-1808). Captain Bettesworth of the Royal Navy married Lady Hannah Althea Whitbread, daughter of General Charles Grey, first Earl Grey. He was killed at Bergen 25 May 1808 while in command of the Tartar. He was a descendant of the Travonian family (Marchand). Sophia Trevanion had married Admiral John Byron.
2. King [née Byron], Ada, countess of Lovelace (1815-1852). Byron's daughter by Lady Byron; in 1835 she married William King, eighth Baron King of Ockham (1805-1893),

INTRODUCTION
 CONTENTS
 SEARCH PERSONS
 ABOUT LBT
 LETTERS BY DATE
 DOCUMENTS BY DATE

Figure 4: Using Lucene to search and score documents for the term “Byron” in all indexed documents.

3 Developer’s Manual

3.1 LBT Overview

Professor Radcliffe has been maintaining Lord Byron and his Times (LBT) in conjunction with the Center for Applied Technology in the Humanities (CATH) for many years. The LBT website contains a large digital archive of various works predominantly written during the 18th and 19th centuries. The website is a great tool for illustrating connections between people. When viewing a letter of correspondence between two people, LBT generates footnotes for references to other people or works from within the document. A user may click on a person’s name, for instance, to view a footnote containing more information about that person. One may search for references to a certain person in other documents, or find references to works authored by that person.

The LBT website as it originally existed was slow for “popular” people. That is, people with a large number of references or who have authored a large number of works. Finding references to Sir Walter Scott, for instance, takes roughly 15 seconds on the original LBT site. This is simply too slow, especially when considering scalability.

3.2 TEI

The Text Encoding Initiative (TEI) is a standard for maintaining digital archives of text^[10]. TEI is largely used in the humanities for maintenance of digital archives, and is used extensively throughout the LBT project. TEI is an XML format containing several hundred tags that can be used for marking up text with varying levels of granularity. It should be noted that TEI *is not* a presentational markup language. That is, it is not sufficient markup for presentation in, say, a web browser. Some computation is required to translate a TEI XML format into XHTML which will be displayed within a browser. An example of TEI markup used by Professor Radcliffe is below.

```
Moreover, his reasoning as to
<persName>Mr. Moore’s</persName>
conduct with regard to
<persName>Lord Byron’s</persName>
<name type="title" key="LdByron.Memoir">Memoirs</name>
, seems to us to be at once vague and inapplicable.
```

In the archives maintained by Professor Radcliffe, much of the TEI markup is references to a person’s name as well as references to written works and correspondence. In the above example, you may notice that Lord Byron’s name is placed in a TEI tag, as well as a tag for work a work entitled *Memoir*. When the TEI is processed, the first two `persName` tags will not be processed into some sort of reference as they contain no key. However, the `name` tag will be converted into a footnote as it contains a key referencing that particular document. In the existing system, XPath^[8] is used to generate these footnotes. In the proposed

LORD BYRON AND HIS CONTEMPORARIES.

Lord Byron and some of his Contemporaries, with Recollections of the Author's Life, and of his Visit to Italy. By LEIGH HUNT. 1 vol. 4to., pp. 513. London, 1828. Coulburn.

'It is for slaves to lie, and for freemen to speak truth.'

Montaigne.

ABOUT as much of this work as is devoted to Lord Byron, is assigned to all the other literary men together who are noticed by the author. Of these papers the most important relate to Mr. Moore, Mr. Shelley, Mr. Keats, Mr. Charles Lamb, and Mr. Coleridge. Of Mr. Moore there is a very lively, pleasant, and characteristic description. Mr. Hunt's anecdotes about the writer of 'Lalla Rookh'

Figure 5: An example of what a marked up document looks like.

system, XQuery will be used instead. Ultimately, the TEI will be processed and displayed in a web browser in a pleasing format, which will be seen in the next section.

3.3 XSL and XSLT

Technically, XSL is a language for expressing stylesheets that consists of XPath and XSLT as subcomponents^[8,11,12]. XPath is a query language used to access certain parts of XML documents, and is used extensively throughout the LBT project to generate references between works and people. XSLT (or XSL Transformations) is a language used to transform XML documents into other XML documents. XSLT is used to transform TEI documents (which are XML documents as discussed previously) into XHTML documents to be displayed in a web browser. Figure 5 is an example of an XHTML document generated using TEI, XSLT, and XPath.

3.4 Current Architecture

In this section, we will describe the architecture designed in implemented by Professor Radcliffe, which can be viewed in figure 6. In this architecture, a web browser requests a page from an Apache web server. A PHP page then fetches the relevant XSL stylesheets, XPath queries, and TEI documents, and the XSLT processor (SAXON, in this case^[13]) then generates an XHTML document. This document is sent back to the web browser for viewing.

In the above pipeline, XPath is used to grab references and generate footnotes. Professor Radcliffe had hand-built a number of XML indexes for the

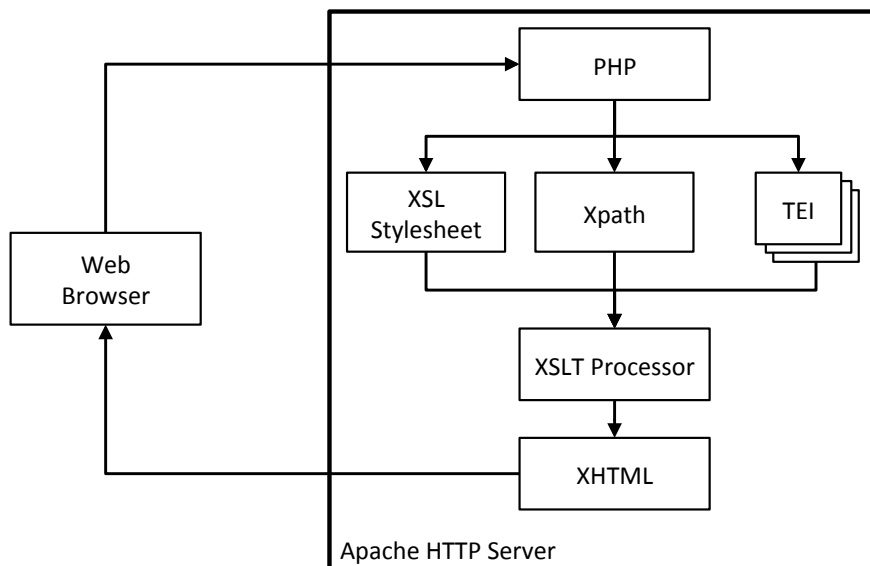


Figure 6: Existing architecture built by Professor Radcliffe.

purpose of looking up someone’s full name, the works associated with them, their birth and death date, etc.

As discussed previously, we found the XSLT processing to be far too slow. Moreover, we felt that maintaining a number of hand-built indexes would be far too cumbersome and scales poorly once more data is accumulated. Our goal was thus to propose a new system that removes the XSLT processor from the pipeline and attempt to streamline it with XQuery. We also wanted to use XQuery, along with eXist, to reduce the reliance on multiple indexes.

3.5 Proposed Architecture

In this section, we discuss the proposed architecture which was implemented as part of the final project. As previously stated, we have removed the XSLT processor and moved all templating and styling to the client side. Figure 7 shows how styling is done primarily on the client-side with CSS. Specific CSS classes still need to be attributed to specific XML elements in eXist when a desired effect is required. The web-pages are displayed using HTML and data is fetched from the eXist-db using asynchronous JavaScript HTTP calls using YAHOO!’s Connection Manager.

On the server side, eXist-db provides several points of access through its web portal and WebDav endpoints. Additionally, it provides a powerful XQuery processing engine that allows us to obtain data through the eXist-db and Lucene indexes. Lucene indexes the documents using the same collection definition that we used to index in eXist-db. Lucene provides us with a robust search

engine that allows us to parse and index each individual TEI document based on specific parameters. The new architecture allows us to perform operations asynchronously and in parallel to help speed up computation and retrieval of documents.

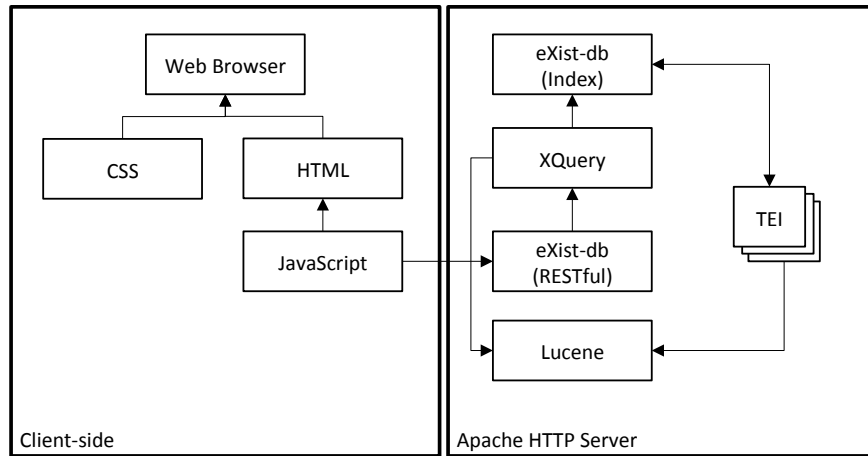


Figure 7: Proposed architecture, minimizing reliance on XSLT and XPath.

4 Technologies

4.1 eXist-db with Lucene

Our implementation of eXist-db and Lucene is available on VTechWorks (see section below for more details). The version used in our project is eXist-db version 1.4.2 Build 20120416 on an Amazon EC2 Enterprise Red Hat Linux machine (Small Instance)^[5].

eXist-db is an XML-based database management system. The current installation of eXist provides several different technologies straight of the box, including Lucene, WebDav, XQuery processing, WebApp development, collection management, and documentation. The eXist-db and Lucene indexes use the same collection definition at `/system/config/db/LBT/collection.xconf` in the WebDav portal to eXist-db. This example configuration assumes the eXist-db collection is called LBT.

The current system uses Jetty as the web server, but eXist-db can be configured to work with Apache, if needed. However, eXist-db and the website must be hosted together because of cross-domain security concerns. These can be alleviated by leveraging domain-agnostic HTTP request libraries in Ajax or YAHOO!^[4].

The Lucene module provided by the eXist-db installation provides all the power of full-text searching offered by Lucene. This module replaces the legacy

searching module within eXist-db for the release used in our project. The same type of configuration for Lucene in a standalone application can be created within eXist-db, anything from boosting fields to custom analyzers and ranking specific results that were queried from the database. We will see that we can combine our Lucene queries uses XQuery to generate search results that are based on rank, and show custom views and explanations as to why a particular document was chosen by Lucene.

The WebDav and WebApp modules of eXist-db allow us to easily manage documents within a particular database. There is no need to remote into a machine and edit documents on the server. By using the RESTful end-points offered by eXist-db, developers and researchers can access and modify remote documents in a local environment. Additionally, by hosting the website within eXist-db, we can execute XQuery scripts easily while taking advantage of eXist-db's caching structure for commonly viewed pages and queries.

The eXist-db installation can be retrieved from VTechWorks.

4.2 YAHOO! Connection Manager and jQuery

We have discussed a lot about batch-loading and the caching system within eXist-db, but in order to access that system we must perform asynchronous HTTP requests to our XQuery scripts while modifying the HTML^[15] document. The YAHOO! Connection Manager (YCM) provides us the ability to perform asynchronous requests to any endpoint within our domain. The main reasons we chose YCM over Ajax was because it is a smaller library, more efficient with resources, and executed slightly faster than basic Ajax request. Additionally, YCM has greater cross-browser support and cross-domain executions are available (although unnecessary for this project). YCM works within JavaScript, and we use jQuery^[3] to make modification to the HTML DOM object when we receive new data. By performing these modifications on the client-side, we can offload some of the computation and styling from the server-side. The former LBT system performed all modifications on the server-side which tied up resources from performing necessary server-side functions (like processing additional requests). Modern web-browsers have more than enough processing power to modify and apply styling and data to web pages on the fly.

Example scripts and files can be retrieved from VTechWorks. In particular, see "example_byron.js" for scripting information.

4.3 Client-side Templating (HTML/CSS)

We have discussed both modifying the content of the HTML and styling on the client-side, but how exactly are we going to do this? The former LBT system defined styles in-line using the standard HTML styles attribute. However, this method is not the best design. For example, if you reuse a specific style for many different elements such as a navigation item, if you wish to change the color or font-size you have to go through every single entry and change the font-size for every styles attribute. Using CSS classes, we can eliminate this bottleneck by

defining a specific type of styles class, and then use the class attribute to assign a style to a particular element within the HTML document. This decreases the complexity and increases the readability of our code without losing contextual information for a particular element.

Examples scripts and files can be retrieved from VTechWorks. In particular, see “example_byron.css” for styling information.

4.4 XQuery

By migrating to eXist-db from flat-XML files, and discontinuing the use of XSL stylesheets, we are required to use XQuery scripts to process and generate XML data from eXist-db. XQuery, in and of itself, is fairly similar to XPath, which is used within XSL. XQuery takes advantage of the power behind eXist-db by directly accessing the index and retrieving documents as quickly as possible. It also allows us to store variables in-memory during execution, which enables us to perform batch-loading on large documents.

Examples scripts and files can be retrieved from VTechWorks. In particular, see “example_documents.xq” for XQuery information.

4.5 Example Migration

This section aims to provide a real-world migration example by modifying a section of XSL in the former LBT system for the current LBT system. This example focuses on transforming the “Documents by Date” listing into the new architecture design.

1. The first step is to locate the “index.xml” document and open it in your favorite text editor.
2. Locate the tag where we test if the input is “AllDocs” (approximately line number 709).
 - (a) This is where the “index.xml” generates the output
3. We need to separate all styling from this, so make note the two “style” attributes for the outermost “div” tags
 - (a) There is also another “style” attribute for each of the items in the list.
4. We can add these two styles to our CSS definitions:

```
.document_title {  
    margin-left: 60px;  
    text-indent: -40px;  
}
```

```
.document_list {
```

```

    font-variant: small-caps;
    font-size: 20px;
    letter-spacing: .1em;
    margin-top: 250px;
    margin-bottom: 20px
}

```

5. Next, we must fetch all the data required by the XSL stylesheet by creating an XQuery script. This can include the title, all subtitles, bibliographic information, and publisher information. There are predefined functions that allow you to access certain types of data that are popular to ascertain. Please see below for a full listing.
 - (a) The first function is simply fetching all the data, returning a header, and a “results” element tag, and storing all the items in-memory on the server.
 - (b) The first part of the function does not require a request parameter.
6. Once we have the data, we can generate our header “div” with the class set to “document_list” as above. This header will also contain the number of “hits” or pages that were generated (there are 20 items per page).
7. Secondly, we need to handle when a request comes in for a particular page number.
 - (a) We go out and fetch the page from memory, and generate a list of the items stored for the page based on the details mentioned in Step 5.
 - (b) We apply the “document_title” style class to each item and return the XML document.
8. So far we have only worked on the server-side, let’s take a look at the client-side.
9. We need to execute the JavaScript function that will kick-off our XQuery script.

```

function fetchDocumentsByDate() {
    loadData('/exist/rest/db/LBT/queries/documents.xq',
            true, false);
}

```

10. The JavaScript function ‘loadData’ will take care of iterating over each page found and displaying in contents.
11. That is pretty much all you need to know about migrating from XSL to XQuery and JavaScript.

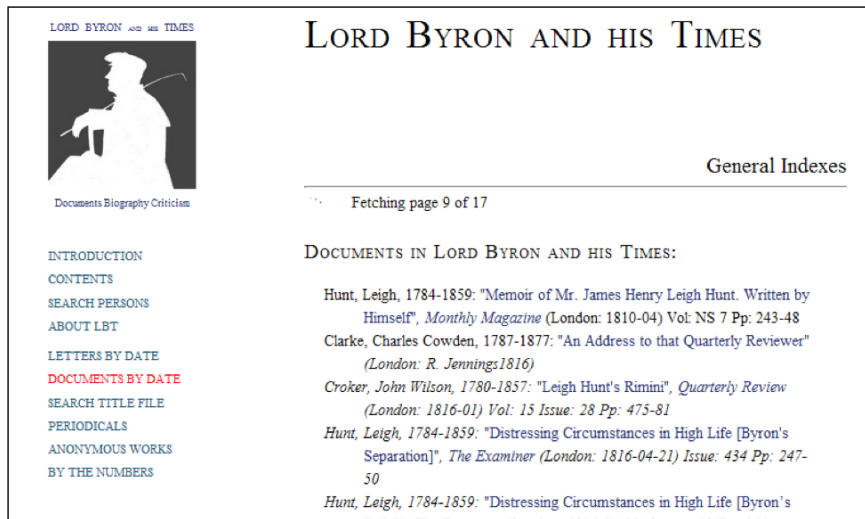


Figure 8: Current LBT System fetching all documents by date using batch-loading.

5 Installation and Documentation

This section aims to provide an overview of installing, operating, and maintain the eXist-db and its associated modules. Additionally, this sections aims to provide a high-level documentation of some of the available functions for XQuery and JavaScript.

5.1 eXist-db Installation

There are two methods for installing eXist-db: custom or prebuilt.

Custom Install

1. Follow the QuickStart guide at <http://exist-db.org/exist/quickstart.xml> for your platform.
2. Connect to the eXist-db WebDav endpoint:
`http://<host>/exist/webdav/db/`
3. Create a new database folder called LBT.
4. Copy LBT XML Files and Queries from VTechWorks into the LBT folder.
5. Create a folder under `/system/config/db/` called LBT.
6. Copy the `collection.xconf` file into this new folder; this sets up the indexing scheme for eXist-db and Lucene. An example is located in VTechWorks under `example_collection.xconf`.

7. Copy the web site data into the **WebApp** folder of the eXist-db install folder.
8. Go to “Start Jetty”.

Prebuilt Install

1. Download the LBT folder from VTechWorks.
2. Go to “Start Jetty”.

Start Jetty

1. Navigate to the eXist-db install location.
2. Execute the command: `export EXIST_HOME=<install location>`
3. Execute the command: `./bin/startup.sh &`
4. Enjoy!

5.2 eXist-db Notes

Occasionally the environment variable `EXIST_HOME` gets reset, please be mindful that you may need to set `EXIST_HOME`.

6 VTechWorks Inventory

This section outlines the files and inventory loaded on the Byron VTechWorks collection.

1. `ProjByron.zip` - the new website files for Project Byron.
2. `exist.tar` - the full install of eXist-db for Project Byron with all files.
3. `example_byron.css` - CSS stylesheet for Byron items, divs, and elements.
4. `example_byron.js` - JavaScript for accessing XQuery documents on eXist-db and other jQuery functions.
5. `example_collection.xconf` - Collection Index Definition for Lucene and eXist-db indexing.
6. `example_documents.xq` - XQuery for fetching documents by date.
7. `example_index.html` - Generic HTML file for displaying information about Project Byron.
8. `This Document` - Final report for the Project Byron group.

7 Contacts

| Name | Email | Expertise | Role |
|-----------------|-----------------|-------------------------------|------------------|
| David Radcliffe | drad@vt.edu | Lord Byron | Stakeholder |
| Kyle Morgan | knmorgan@vt.edu | Software Engineering | Graduate Student |
| Kyle Schutt | kschutt@vt.edu | Software Engineering | Graduate Student |
| Purdom Lindblad | purdom6@vt.edu | Library Services | Collaborator |
| Keith Battleson | keith12@vt.edu | Information Technology | Collaborator |
| Edward Fox | fox@vt.edu | Information Storage/Retrieval | Collaborator |

References

- [1] eXist-db. (2012, Nov.) eXist-db Open Source Native XML Database. [Online]. <http://exist-db.org/exist/index.xml>
- [2] Michael McCandless, Erik Hatcher, and Otis Gospodnetic, Lucene in Action. Safari Books Online: Manning Publications, 2010.
- [3] The jQuery Foundation. (2012, Nov.) jQuery: The Write Less, Do More, JavaScript Library. [Online]. <http://jquery.com/>
- [4] YAHOO! Developer Network. (2012, Nov.) YUI 2: Connection Manager. [Online]. <http://developer.yahoo.com/yui/connection/>
- [5] Amazon. (2012, Nov.) Amazon Web Services LLC. [Online]. <http://aws.amazon.com/ec2/>
- [6] Mort Bay Consulting. (2012, Nov.) Jetty WebServer. [Online]. <http://jetty.codehaus.org/jetty/>
- [7] SyncRO Soft SRL. (2012, Nov.) XSLT Editor. [Online]. http://oxygenxml.com/xml_editor/xslt_editor.html
- [8] XML Path Language. (2012, Oct.) XPath Documentation. [Online]. <http://www.w3.org/TR/xpath/>
- [9] The eXist Project. (2012, Oct.) XQuery Documentation. [Online]. <http://exist-db.org/exist/xquery.xml>
- [10] TEI: Text Encoding Initiative. (2012, Oct.) TEI: Text Encoding Initiative. [Online]. <http://www.tei-c.org/index.xml>
- [11] w3.org. (2012, Oct.) The Extensible Stylesheet Language Family (XSL). [Online]. <http://www.w3.org/Style/XSL/>
- [12] w3.org. (2012, Oct.) XSL Transformations (XSLT). [Online]. <http://www.w3.org/TR/xslt>
- [13] SAXON. (2012, Dec.) The XSLT and XQuery Processor. [Online]. <http://saxon.sourceforge.net>
- [14] The PHP Group. (2012) PHP: Hypertext Preprocessor. [Online]. <http://php.net/>
- [15] W3C. (2012) W3C HTML. [Online]. <http://www.w3.org/html/>
- [16] W3C. (2012) Cascading Style Sheets. [Online]. <http://www.w3.org/Style/CSS/>
- [17] Center for Applied Technologies in the HUmanities. (2012) Welcome to CATH. [Online]. <http://wiz2.cath.vt.edu>
- [18] David H Radcliffe. (2012) The Life and Times of Lord Byron. [Online]. <http://lordbyron.cath.lib.vt.edu/>