Basic eLearning ToolSet (BELTS)

Getting Started with BELTS Development

Kevin O'Neill

Basic eLearning ToolSet (BELTS): Getting Started with BELTS Development

by Kevin O'Neill

Published 14 April 2005

Disclaimer of Warranties and Limitations of Liabilities

Table of Contents

List of Figures

Chapter 1. Introduction to BELTS

1.1. About BELTS

The Basic E-Learning Tool Set (BELTS) has been developed by The Le@rning Federation (TLF) [http://www.thelearningfederation.edu.au] to demonstrate the distribution, management and use of online curriculum content and to aid investigation of requirements for e-learning environments by Australian and New Zealand school jurisdictions.

BELTS currently provides a limited set of tools, including:

- A content repository;
- Basic activity creation, using lessons;
- Basic group management, using classes;
- Content to curriculum outcomes matching (the curriculum organiser);
- Downloading of content from The Le@rning Federation's Exchange repository of online curriculum content;
- Content replication from one BELTS to another, and
- System administration.

Note

BELTS has currently not been developed as a fully featured learning management system. BELTS is, however, an open source project that can be further developed. The Le@rning Federation encourages Australian and New Zealand education jurisdictions, and others, to consider options for collaborating and contributing to the evolution of BELTS. For more information about the project and how you can participate visit the BELTS project web site [http://belts.sourceforge.net]

1.2. About The Le@rning Federation

The Le@rning Federation [http://www.thelearningfederation.edu.au], is an initiative delivered on behalf of the Australian Education Systems Officials Committee (AESOC) by a joint venture of education.au limited [http://www.educationau.edu.au] and Curriculum Corporation [http://www.curriculum.edu.au]

In January 2001, as part of the Backing Australia's Ability: Innovation Action Plan [http://backingaus.innovation.gov.au] the Prime Minister announced funding of $34.1 million over 5 years to support the Initiative to:

- Develop a body of high-quality curriculum content, suitable to each State and Territory;
- Develop a framework which supports distributed access;
- In the long term, use the framework and content to stimulate further contribution to the pool of material.

In July 2001, all Australian States and Territories agreed to match the Commonwealth funds. Following this, New Zealand joined in the Initiative.

1.3. Who is This Guide For?

This guide is aimed at primarily at two types of people. *Stylists* who want to change the look of a BELTS installation, and *Developers* who want to modify or extend belts functionality.

1.4. Comments and Feedback

If you have any comments, corrections or feedback on this guide please feel free to contact the BELTS developers using the mailing lists at the BELTS SourceForge Site [http://belts.sourceforge.net].

Chapter 2. Stylist

2.1. What is a Stylist

A Stylist is someone who wants to change the look of a BELTS installation.

The structure of BELTS makes it possible to change most look-and-feel elements using CSS (Cascading Style Sheets). Simple changes to the supplied CSS files will allow you to change things like:

- font styles
- background colours
- roll-over images

More complex changes can be made by editing the *XSLT* files that are used to generate the final *HTML* pages.

2.2. What a Stylist needs to know

Before you start changing the look and feel of BELTS, you will need to be familiar with a number of technologies and products.

CSS is used to manage all of the style elements in BELTS. A solid understanding of the box model [http://www.w3.org/TR/CSS2/box.html] (used to determine element position) is particularly important. Although some CSS level 2 constructs have been used, the majority of BELTS uses level 1 to help with Internet Explorer™ compatibility.

You'll need a working knowledge of *XSLT* if you want to change the structure of pages.

All the pages generated by BELTS conform to the *XHTML* specification and there are a number of subtle differences from *HTML* 4.0.1 with which you need to be familiarise yourself.

You may also want to learn how to read a *Cocoon* site map. These control what steps are taken to generate a page. Being able to read these files will help you when you want to know which XSLT to change for a specific screen.

2.3. Styling BELTS

2.3.1. Setting up

Stylists will normally work on a prebuilt version of BELTS. If you want to build your own version of BELTS from the source you should consult Chapter 3, *Developer* [4].

First thing you'll need to do is establish `$BELTS_HOME`. To do this follow the instructions laid out in Installing BELTS [http://belts.sourceforge.net/installation/index.html] to install a local working copy of BELTS. Once installed you should ensure the system is functioning correctly by browsing to the home page. If you can browse to the local home page, you're ready to edit BELTS.

2.3.2. Editing

All the files you might like to modify are in `$BELTS_HOME/server/default/deploy/belts.ear/belts-web.war/`, which is the *WAR* directory where you'll make all changes. The listing below

```
[kevin@titus belts-web.war]$ ll
total 144K
-rw-rw-r--    1 kevin    kevin        107K Aug 15 10:09 belts.xmap
drwxrwxr-x    2 kevin    kevin        4.0K Aug 15 12:49 META-INF
drwxrwxr-x    2 kevin    kevin        4.0K Jun 12 15:03 sample-data
```

```
-rw-rw-r--    1 kevin    kevin        9.1K Aug  4 13:47 sitemap.xmap
drwxrwxr-x    3 kevin    kevin        4.0K Aug 11 14:23 static
drwxrwxr-x    5 kevin    kevin        4.0K Aug 15 12:49 WEB-INF
drwxrwxr-x   13 kevin    kevin        4.0K Aug 11 12:10 xsl
```

The directory of most interest to the Stylist is `static`. In here, you'll find all the CSS and image files used in the interface. The most important CSS file is `common.css`, which sets the style for most pages. The other CSS files (like `nude-common.css`) are used in specialised cases. Normally, you will not need to make too many changes to these.

If you want to change the structure of the HTML generated, `xsl` is the directory for you. The XSLT style sheets often interact with each other, so it's important you take your time and check your work as you go. Lots of small changes are better than a simple large-scale change. If something goes wrong, it's easy to back your last change out if you've made small changes only.

Although we've tried to make the file names self-documenting sometimes it might not be clear which XSLT file needs editing. In these cases, armed with your *Cocoon* knowledge, you'll need to dive into what is probably the biggest single file in BELTS, `belts.xmap`. You'll normally want to open this *read-only*, as an incorrect change in this file can stop the web interface functioning correctly. A short time following the `match` statements and you should know which XSLT you'll need to edit.

BELTS will detect any changes you make to these files and reload them as necessary. This means you can edit them while BELTS is running and see you're changes immediately.

Chapter 3. Developer

3.1. What is a Developer

A Developer is someone who wants to enhance or extend BELTS.

BELTS provides a framework for a learning delivery but there are many new features that could be added:

- tracking and managing assessment
- integrating assessment results with other administrative and assessment systems
- monitoring students lesson progress
- collaboration tools such as discussion lists, mail, chat and announcements
- storing, managing and distributing user created content

3.2. What a Developer needs to know

Before you can start adding new features to BELTS you will need to be familiar with a number of technologies and products, many of which are described in this chapter. This list is not exhaustive, however; BELTS uses many open source libraries not specifically mentioned here. We assume you will read the *Javadoc* for these libraries as you encounter them.

The primary technology used to build BELTS is *Java™*, so it stands to reason that you'll need to be proficient with it. As BELTS uses the *J2EE™ APIs* for things such as data persistence and transaction management, you'll also need to be familiar with these. In any server environment, multi-threading plays a key role, so understanding Java™ concurrency is mandatory. To help you when things go wrong you should become familiar with Java™ remote debugging.

Much of the data and the entire presentation system uses *XML* in one way or another so you should be comfortable working with XML and understand the associated Java™ APIs. Additionally, you should understand *XML Namespaces* and namespace prefix handling.

The *services* that make up BELTS are exposed as *JMX* management beans. If you're adding new services or extending current ones (say by adding a new content provider), you'll need to understand their creation and configuration.

As noted above BELTS is built to run in a J2EE™ environment. *JBoss* is an application server that implements the J2EE™ standard. You'll need to be able to configure and run JBoss, as well as understand where it differs from other J2EE™ servers[1], in order to work effectively.

The presentation framework is powered by *Cocoon*, a component based web development framework with powerful *XML* processing capabilities. As all page processing is done via Cocoon, and we've added a command processing framework to extend its capabilities (see Section 3.4, “Command Processing” []). You'll need to understand the Cocoon site map and configuration files, as well as its page processing model.

Information such as content meta data and curriculum outcomes are stored as XML. To store and manage this BELTS uses the *eXist* native XML database. If you want to store or query XML data, you need to get to know eXist and the *XML:DB* API.

Although eXist handles XML searching, general searching is handled using *Lucene*. If you need to index data that's not XML based, you'll need to understand Lucene.

The build process for BELTS uses *Ant*. If you intend to add new modules, you'll want a clear understanding of ant. For more information on building BELTS, see Section 3.5, “Building BELTS” [].

J2EE™ development involves creating lots of configuration and derivative files. *XDoclet* takes the pain out of this by generating these files using special Javadoc tags. If you're creating *entity* or 'm' beans, XDoclet is your friend.

1 Although the BELTS is a J2EE™ application, it can't be run inside of J2EE™ servers other than JBoss.

If you'll be adding pages or changing the look and feel of BELTS, then you should also be familiar with the technologies in Section 2.1, “What is a Stylist” [2].

3.3. The BELTS architecture

When looked at as a whole BELTS is a large and complex system with hundreds of components. Like most complex things, though, BELTS can be broken down into component parts, which themselves can often be broken down. Once you have a basic understanding of how the system breaks down, working with the individual parts is far less daunting than trying to work with the system as a whole.

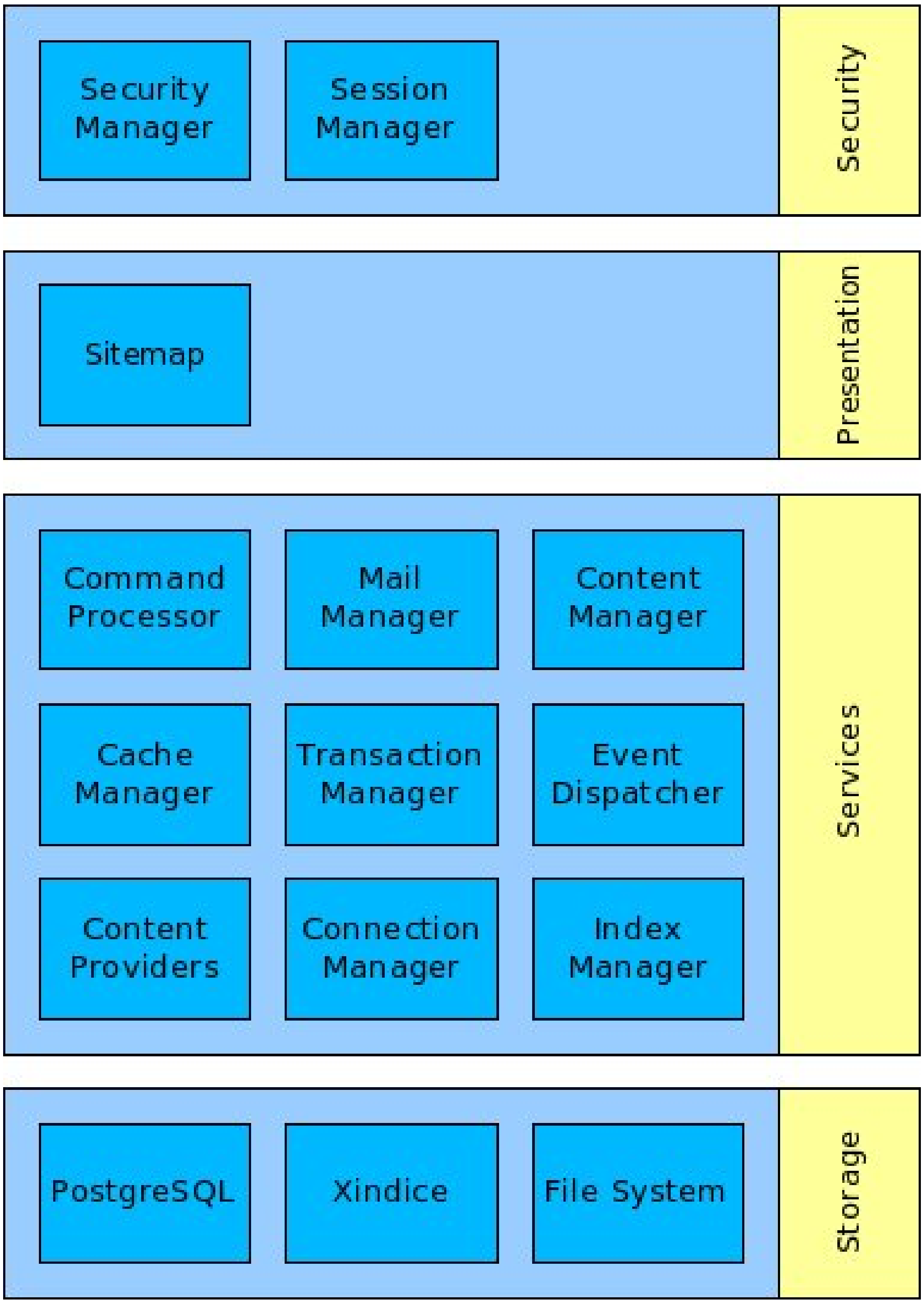

Figure 3.1. Architectural Overview

Figure 3.1 [5] provides a general breakdown of BELTS. BELTS is first broken down into four layers: security, presentation, service and storage. These layers each contain a number of components. The function of each of these layers and an overview of what each layer contains is provided below.

3.3.1. Security

Before any user-initiated action occurs the security layer is invoked. There are two primary components to this layer: the security manager and the session manager. The security manager is responsible for checking credentials against principals, and also ensures that the security context is propagated to the current thread of execution. The session manager maintains user information between *HTTP* requests.

3.3.2. Presentation

The presentation layer accepts the HTTP request, decides what tasks must be executed in order to fill it, and then initiates their execution. In most cases, this involves invoking the command processor in the service layer and then transforming the result via a series of XSLT style sheets. The site map (which is a part of Cocoon) maintains the mapping of request parameters to task pipelines.

3.3.3. Service

By far the biggest layer in BELTS, the service layer contains the components that make up the core of the system. It contains services for managing transactions, commands, content, indexes, caches, mail, caches, events, content and external connections. The most significant of these is the command processor. For more information on command processing see Section 3.4, “Command Processing” [].

3.3.4. Storage

The storage layer is responsible for ensuring that data is available between system executions. BELTS stores its data in three places. The file system might seem a little obvious but it's an important storage location for BELTS for things like content downloaded from upstream systems. *PostgreSQL* is used to store relational data. It is accessed almost exclusively via enterprise beans. The native XML database eXist is used to store the XML data produced by BELTS.

3.4. Command Processing

All actions in BELTS are mapped through its command processing system. The command processor takes care of all the administrative issues in executing a web command. It sets the command parameters, validates them, and then executes the command in the transaction environment (see Figure 3.2 [6]). The command processing system is designed to make writing commands simple.

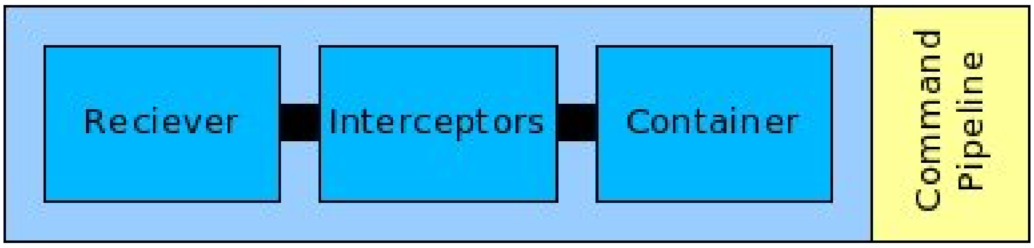

Figure 3.2. Command Processing Pipeline

3.4.1. Receiver

A request is received via the cocoon pipeline. The receiver maps the request to an action and establishes the interceptor chain.

3.4.2. Interceptors

Interceptors are the heart of the command processing system. When a command is executed, the receiver determines the list of interceptors to be run on the command before it's executed. Interceptors are responsible for things such as setting the properties of the command by extracting information from the request and ensuring that the requester has the correct system role.

Each interceptor in the chain has the opportunity to abort the command. For example, the validation interceptor will do this if the parameters required to execute the command haven't been supplied.

3.4.3. Container

Once the command has passed though the interceptor chain, it's handed to the container. The container establishes the transaction and executes the command. If the command fails for any reason, the container ensures that the transaction is rolled back. The container also makes a number of services available to the command.

3.4.4. Sample Command

Because all of the parameter extraction and transaction setup is done by the command framework, commands themselves can be very simple. The listing below shows the command to update a content provider in BELTS.

```
/* Copyright 2002-2005, education.au limited and
                Curriculum Corporation, All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 modification,
 * are permitted provided that the following conditions are met:
 *
 * Redistributions of source code must retain the above copyright notice, this
 list of
 * conditions and the following disclaimer.
 *
 * Redistributions in binary form must reproduce the above copyright notice, this
 list
 * of conditions and the following disclaimer in the documentation and/or other
 materials
 * provided with the distribution.
 *
 * Neither the names education.au limited, Curriculum Corporation nor The
 Learning
 * Federation nor the names of its contributors may be used to endorse or promote
 products
 * derived from this software without specific prior written permission.
 *
 * To the extent permitted by law, the copyright owners of this software and its
 contributors:
 *
 * (i) exclude all warranties in relation to the software; and
 * (ii) exclude liability for all losses, costs, expenses and damages arising in
 any way
 * from the use of the software whether arising from or in relation to breach of
 contract,
 * negligence or any other tort, in equity or otherwise. If the software is in
 breach of a
 * warranty which is implied by law, the copyright owners and contributors
 liability is
 * limited to the replacement of the software.
 *
 * Created: 8/07/2003
 *
 * $Id: developer.xml,v 1.5 2005/01/25 10:59:59 gregj_jacus Exp $
 */
```

```
package au.edu.educationau.belts.command.content.action;

import java.util.Date;

import org.apache.commons.lang.StringUtils;

import au.edu.educationau.belts.command.CommandResult;
import au.edu.educationau.belts.command.container.AbstractBeltsContainerCommand;
import au.edu.educationau.belts.content.Update;

/**
 * UpdateProviderAction causes a update to be run on the provider service named.
 *
 * @author <a href="mailto:kevin@rocketred.com.au">Kevin O'Neill</a>
 * @version $Revision: 1.5 $ - $Date: 2005/01/25 10:59:59 $
 */
public class UpdateProviderAction extends AbstractBeltsContainerCommand
{
        private String _provider;

        public void execute()
        {
                _outcome = CommandResult.OUTCOME_ERROR;

                new Update(_provider).update(new Date(0L));

                _outcome = CommandResult.OUTCOME_OK;
        }

        /**
         * @param name - provider name to update
         */
        public void setProvider(String name)
        {
                _provider = name;
        }

        /* (non-Javadoc)
         * @see au.edu.educationau.belts.command.Command#validate()
         */
        public boolean validate()
        {
                if (StringUtils.isEmpty(_provider))
                {
                        addFieldValidationError("provider");
                }

                return super.validate();
        }
}
```

3.5. Building BELTS

To build BELTS you'll need the source. This can be obtained by following the links at the BELTS SourceForge site [http://belts.sourceforge.net/]. Provided you have CVS installed anonymous CVS access is probably the best way to get the source. If you're unsure how to use CVS, there is plenty of documentation on the Sourceforge site.

```
cvs -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/belts login
cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/belts co -d belts belts-all
```

Once you've downloaded or fetched the source, you should have a directory that looks something like this.

```
[kevin@titus belts]$ ll
total 48K
```

```
drwxrwxr-x    6 kevin    kevin         4.0K Aug 15 10:25 build
drwxrwxr-x    6 kevin    kevin         4.0K Aug 15 10:25 components
drwxrwxr-x    6 kevin    kevin         4.0K Aug 15 10:25 ejb
drwxrwxr-x    4 kevin    kevin         4.0K Jul 21 14:02 mock
drwxrwxr-x    6 kevin    kevin         4.0K Aug 15 10:25 model
drwxrwxr-x    6 kevin    kevin         4.0K Aug 15 10:25 store
drwxrwxr-x    4 kevin    kevin         4.0K Jul 24 14:33 test
drwxrwxr-x   17 kevin    kevin         4.0K Aug 15 10:06 thirdparty
drwxrwxr-x    6 kevin    kevin         4.0K Jun 12 11:45 tools
drwxrwxr-x    5 kevin    kevin         4.0K Aug 15 10:26 web
[kevin@titus belts]$
```

You'll also need a local copy of JBoss. This **must** be version 3.2.0. No other version will work. Unpack the distribution and put it in a directory named `jboss`.

Copy `build/etc/local.properties-example` to `build/local.properties`

```
[kevin@titus belts]$ cp build/etc/local.properties-example build/local.properties
```

Edit `build/local.properties` using the text editor of your choice. At the bottom of this file is a property `belts.package-path`. Set this to the directory containing the `jboss` directory created earlier.

```
#
# Root Path for Thirdparty Packages
#
belts.package-path=/home/kevin/tools
```

Now all you have to do it build it using `build/build.sh` on Linux or `build/build.bat` on Windows. The output should start something like this:

```
[kevin@titus belts]$ ./build/build.sh
Searching for build.xml ...
Buildfile: /home/kevin/projects/belts/build/build.xml
```

and end something like this:

```
main:

BUILD SUCCESSFUL
Total time: 39 seconds
[kevin@titus belts]$
```

If you have any problems building BELTS, the best place to go for advice and help is the BELTS SourceForge site [http://belts.sourceforge.net] and the *belts-developers* mailing list.

Appendix A. Products used in BELTS

BELTS is built upon the shoulders of many other open source projects. This appendix provides some information on a few of them.

A.1. JBoss [http://www.jboss.org/]

JBoss was started in March 1999. Now reaching the 3.0 series, JBoss is a mature appserver. It has grown way beyond its original scope. While JBoss started as, and still is, an EJB container, today JBoss does the full J2EE stack.

The 3.x series are fully based on J2EE and, in one easy-to install and easy-to-use package, you will find all of the J2EE stacks and more. From a full fledged HTTP 1.1 web server up to the latest Java Connector Architecture, one of the most ground breaking micro kernel approaches to appserver technology based on Java Management Extensions and an advanced Container Managed Persistence 2.0 framework, JBoss has all the features you want in an application server.

A.2. Cocoon [http://cocoon.apache.org/]

Apache Cocoon is an XML publishing framework that raises the usage of XML and XSLT technologies for server applications to a new level. Designed for performance and scalability around pipelined SAX processing, Cocoon offers a flexible environment based on a separation of concerns between content, logic, and style. To top this all off, Cocoon's centralized configuration system and sophisticated caching help you to create, deploy, and maintain rock-solid XML server applications.

Cocoon interacts with most data sources, including filesystems, RDBMS, LDAP, native XML databases, and network-based data sources. It adapts content delivery to the capabilities of different devices like HTML, WML, PDF, SVG and RTF to name just a few. You can run Cocoon as a Servlet, as well as through a powerful commandline interface. The deliberate design of its abstract environment gives you the freedom to extend its functionality to meet your special needs in a highly modular fashion.

A.3. Jetty [http://jetty.mortbay.com/]

Jetty is a 100% Java HTTP Server and Servlet Container. This means that you do not need to configure and run a seperate web server (like Apache HTTPd) in order to use java, servlets and JSPs to generate dynamic content.

Jetty is a fully featured web server for static and dynamic content. Unlike separate server/container solutions, this means that your web server and web application run in the same process, without interconnection overheads and complications. Furthermore, as a pure java component, Jetty can be simply included in your application for demonstration, distribution or deployment. Jetty is available on all Java-supported platforms.

A.4. eXist [http://exist.sourceforge.net/]

eXist is an Open Source native XML database featuring efficient, index-based XQuery processing, automatic indexing, extensions for full-text search, XUpdate support and tight integration with existing XML development tools

A.5. Lucene [http://jakarta.apache.org/lucene/]

Apache Lucene is a Java library that adds text indexing and searching capabilities to an application. It offers a simple, yet powerful core API. To start using it, one needs to know only a few Lucene classes and methods. Lucene offers two main services: text indexing and text searching. These two activities are relatively independent of each other, although indexing naturally affects searching.

Glossary

ANT	A tool for building projects created and managed by the Apache Foundation.
API (Application Programming Interface)	An Interface which is used for accessing an application or a service from a program.
Cocoon	A component based web development framework with powerful *XML* processing capabilities.
CSS (Cascading Style Sheets)	A simple mechanism for adding style (e.g. fonts, colors, spacing) to Web documents.
CVS (Concurrent Versions System)	A tool used by developers to manage changes within their source code tree.
Developer	Someone who modifies or enhances (and sometimes breaks) BELTS.
EJB (Enterprise JavaBeans)	A body of code with fields and methods to implement modules of business logic.
eXist	eXist is an Open Source native XML database featuring efficient, index-based XQuery processing, automatic indexing, extensions for full-text search, XUpdate support and tight integration with existing XML development tools.
HTML (HyperText Markup Language)	The *lingua franca* for publishing hypertext on the World Wide Web.
HTTP (Hypertext Transfer Protocol)	The protocol used for information exchange on the world wide web.
J2EE (Java 2 Platform, Enterprise Edition)	A standard for developing multi tier enterprise applications.
Java™	A general-purpose object-oriented programming language.
Javadoc	The tool from Sun Microsystems for generating API documentation from comments in Java™ source code.
JBoss	An open source application server that implements (mostly) the *J2EE™* standard.
Lucene	A high performace text indexing system.
JMX (Java Management Extensions)	The definitive means for implementing management and monitoring in Java™.
PostgreSQL	An advanced relational database management system.

Service	A Service provides a set of features to the BELTS system.
Stylist	Someone responsible for the look and feel of a web system.
WAR (Web Application Archive)	A package for the web components of a *J2EE*™ application.
XDoclet	A code generation engine that enables attribute oriented programming for java.
XHTML (Extensible HyperText Markup Language)	A family of current and future document types and modules that reproduce, subset, and extend *HTML*, reformulated in *XML*.
XML (Extensible Markup Language)	a simple, very flexible text format derived from SGML (ISO 8879 [http://www.iso.ch/cate/d16387.html]).
XML:DB	An API for accessing native XML databases.
XML Namespaces	A simple method for qualifying element and attribute names used in XML documents.
XSL (Extensible Stylesheet Language)	A family of recommendations for defining XML document transformation and presentation.
XSLT (*XSL* Transformations)	A language for transforming XML documents into other XML documents.

Index