WebMan: A Basic RefEntry Servlet

Mark Craig

WebMan: A Basic RefEntry Servlet
by Mark Craig

ACCIS (http://www.accis.edu/) offers CS445, Independent Programming Project, a four-hour elective course for students completing the undergraduate Computer Science program. The project documented here involves designing, building, and testing an application using the Java™ programming language to handle, format, and deliver reference manual entries marked up in in DocBook-descendant XML.

The application, a basic RefEntry servlet, performs a function similar to the **man** command on UNIX® systems. Instead of handling **roff** format files and posting formatted results to a terminal, however, the RefEntry servlet handles XML files and posts results in XHTML that can be viewed in most web browsers.

Mark Craig, 370-68-2121

Table of Contents

Preface

You are reading the documentation for a CS445 Independent Programming Project. The project involves building a basic servlet for reference manual entries.

Background

A key element of product documentation for software written on UNIX systems, online **man** pages have been around for a long time. Dennis Ritchie originally wrote the **man** command to format reference documentation for display on a terminal. The command has a few helpful but aging features for searching and displaying reference documentation. Although more recent forms of online help have surpassed its user-friendliness, many UNIX developers use **man** more than any other documentation tool.

The **man** command implementations available today remain based on **roff** formatting tools, so documenters either format online manual pages directly with **roff** macros, or they mark up the manual pages in another language, which the command then translates to **roff** markup for formatting with the old tools.

Recently, XML languages and tools have begun to appear. Applications have started to offer support for languages such as XSLT and XPath to format and search XML documents. Today applications can do more, more simply, with reference manual entries edited using XML markup than used to be feasible with **roff** markup. XML makes it easier for example to write a reference manual browser that beginners find intuitive.

XML also implies semantic rather than presentational markup. This means XML RefEntry documents need no rewriting when tools change, only when content changes. Documenters can mark up given content once and then web developers can reformat them as needed using languages such as XSLT. Furthermore, as XSLT engines are available on all platforms, developers no longer need lots of extra software or a UNIX system to read RefEntry documents. This could encourage documenters to write reference manual entries with XML markup for cross-platform portability.

Project Content

Like every other Independent Programming Project, this one grew out of what the author has learned while studying at ACCIS, and even before starting my studies. I rely for example on what I learned in CS260, Concepts of Java, for Java language basics. My interest in hacking on software like this, however, comes mainly from the need to make the job of maintaining documentation and rendering it usable easier. We have not left the dark ages of technical writing, yet.

Project texts are specified in the bibliography of this document. I also used the on-line documentation delivered with all test frameworks, applications and Java classes used for this project. As I did not always manage to find answers to all my questions in books and web pages, some answers about Java and XSLT were offered by colleagues at Sun Microsystems, especially Guillaume Jacob, who patiently explained a thing or two about default node processing in XSLT that I probably would not have understood otherwise.

The project includes four progressively complete deliveries. Each delivery includes, in addition to the deliverables specified in the following table, release notes, and corrected content from the previous delivery.

Table 1. Project Milestones, By Delivery

Deliverable	First	Second	Third	Final
Project documentation	X	X	X	X
Software design specifications	X	X	X	X
RefEntry document type definition	X	X	X	X
Test plans	X	X	X	X
Software components		X	X	X
Stylesheets		X	X	X
Functional tests		X	X	X
User documentation			X	X
Integrated software components			X	X
System tests			X	X
Installation documentation, Index				X
Debugged, installable software				X
Installation tests				X

This represents a volume of work greater than that required for other four-hour courses taken previously.

In this Document

In this document you find the information about this Independent Programming Project. This document encompasses everything from design specifications to testing, to user documentation, to reference documentation. In addition to this preface, this document includes the following chapters.

- *Installing the Application* details how to install the project software both on Linux and on Windows systems.
- *Using the RefEntry Servlet and Client* provides procedures and examples showing how to use the servlet and client interface.
- *RefEntry Client Design* covers the client architecture and describes the user interface for the browser based client.
- *RefEntry Servlet Design* covers the servlet architecture and specifies the application interface.
- *Test Plan and Results* lays out the testing strategy, specifies the specific tests used to qualify the application, and includes the resulting test report.

This document also includes the following appendices.

- *RefEntry Elements* describes the XML document type definition (DTD) elements and attribute lists to which supported RefEntry documents conform. It specifies formatting expectations for each element.
- *Code Listings* includes listings of the code written for the project.
- *Bibliography* cites the books used fairly intensively for this project.

An index figures at the end of this document.

Chapter 1. Installing the Application

This chapter describes how to install the project software both on Linux and on Windows systems.

Installing on Linux

This seems to work on Solaris systems and possibly other UNIX systems as well. The examples in the following steps work if you are running Bash, the default shell on Linux. To run Bash in a terminal window, enter `bash` at the shell prompt.

1. Make sure you have a Java Development Kit 1.4 or later installed. If you do not have a recent JDK, download one from http://java.sun.com. If you do not know which version you have, then check. For example, if you have installed 1.4.1:

   ```
   bash$ java -version
   java version "1.4.1-rc"
   Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1-rc-b19)
   Java HotSpot(TM) Client VM (build 1.4.1-rc-b19, mixed mode)
   ```

2. Do not begin the installation before reading this:

   The only pack you *must* install is the servlet `.war` file. This is the case so you do not have to reinstall Ant, HttpUnit, or Tomcat if you have it already.

   If you want to use the servlet `.war` file, you must have a servlet container such as Tomcat. So if you are missing a servlet container, install the Tomcat pack at least.

   If you want to do any development, such as write your own RefEntry documents and build a servlet, then you need Ant and HttpUnit. Unless you have Ant and HttpUnit already installed, take these packs as well.

   If you are reading this file, you may have already decided to install the documentation. Good.

3. Run the self-extracting installer, `refentry-installer.jar`.

   ```
   bash$ java -jar install-dir/refentry-installer.jar
   ```

   Follow the instructions based on what you decided to install.

   If you run this in a terminal without starting X, the installer generates an unexpected exception on startup and exits, displaying the stack trace. This particular installer does not work without a GUI, so start X before you run it. You can go back to your terminal window after you install it.

4. Set JAVA_HOME to point to the JDK if it is not set already. For example:

   ```
   bash$ echo $JAVA_HOME

   $ which java
   path-to-java/bin/java
   $ export JAVA_HOME=path-to-java
   ```

   If you are on a network where multiple Java SDK versions are maintained side by side, finding the JAVA_HOME might be much less straightforward. At work, one of my install testers had to follow three symbolic links before she finally got to the directory where Java was actually physically installed.

5. Source the `refentry.profile` file in the directory where you installed the bits. For example:

   ```
   bash$ source install-dir/refentry.profile
   ```

   The `refentry.profile` file is written for Bash. Hack the file yourself if you want to use a different shell.

6. Copy the servlet `.war` file to the Tomcat `webapps` directory.

   ```
   bash$ cp install-dir/refentry.war $CATALINA_HOME/webapps/
   ```

   CATALINA_HOME designates the Tomcat servlet container installation directory.

7. Start the Tomcat servlet container.

   ```
   bash$ $CATALINA_HOME/bin/startup.sh
   ```

   This starts the Tomcat servlet container in the background.

8. Browse http://localhost:8080/refentry and you should see the default page of the RefEntry servlet.

If everything went well and you want to develop your own RefEntry servlet with your own documentation, you might want to add two lines in your `.bash_profile` to set JAVA_HOME and to source `refentry.profile` at startup.

If you want to stop Tomcat, the command is: `$CATALINA_HOME/bin/shutdown.sh`

Installing on Windows

1. Make sure you have a Java Development Kit 1.4 or later installed. If you do not have a recent JDK, download one from http://java.sun.com. If you do not know which version you have, then check. For example, if you installed 1.4.1:

   ```
   C:\Windows>java -version
   java version "1.4.1_01"
   Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_01-b01)
   Java HotSpot(TM) Client VM (build 1.4.1_01-b01, mixed mode)
   ```

2. Do not begin the installation before reading this:

   Avoid installing in a folder with a name longer than 8 characters. This software relies on environment variables rather than the Windows registry. If you use long folder names, you have to figure out how to translate them when setting the environment variables. For example, `C:\Program Files\RefEntry` may become `C:\Progra~1\RefEntry`.

   The only pack you *must* install is the servlet `.war` file. This is the case so you do not have to reinstall Ant, HttpUnit, or Tomcat if you have it already.

   If you want to use the servlet `.war` file, you must have a servlet container such as Tomcat. So if you are missing a servlet container, install the Tomcat pack at least.

   If you want to do any development, such as write your own RefEntry documents and build a servlet, then you need Ant and HttpUnit. Unless you have Ant and HttpUnit already installed, take these packs as well.

   If you are reading this file, you may have already decided to install the documentation. Fine.

3. Double-click the self-extracting installer, `refentry-installer.jar`.

Follow the instructions based on what you decided to install.

4. If you are using Windows 95 or 98, copy the *install-dir*\ant folder to a location with a very short path name such as C:\ant.

   Ant uses a startup script that suffers from DOS-related limitations.

5. Adjust and set the environment variables from *install-dir*\refentry.bat. How you do this depends on your version of Windows.

   Windows command.com shell cannot handle long file and folder names. You must therefore declare environment variables using the 8+3 MS-DOS names. In other words, if you install the software in C:\Program Files\RefEntry, then you must translate this into something like C:\Progra~1\RefEntry in refentry.bat and in autoexec.bat.

   When using Windows 2000, you can set environment variables for your account, and Windows puts them in the registry. Set the environment variables you need from Control Panel > System > Advanced > Environment Variables, based on the packs you installed.

   When using Windows 95 or 98, you can call a script from C:\autoexec.bat which sets the environment at startup. You can also add the environment variables you need, based on the pack you installed, directly to autoexec.bat. If you use ANT_HOME, note that it must be a very short path name on Windows 95 or 98.

6. At this point, if you are using Windows 95 or 98, you must reboot.

7. Copy *install-dir*\refentry.war file to the *install-dir*\tomcat\webapps folder.

8. Start Tomcat.

   With Windows 95 and 98, you must increase the memory available in the command shell to be able to start Tomcat. To increase the memory available, click the Properties icon, then in the MS-DOS Prompt Properties window, select the Memory tab and set Initial environment to 4096. Next, click OK, and type exit to leave the MS-DOS Prompt command shell. Restart the MS-DOS Prompt command shell for the change to take effect.

   ```
   C:\Windows>%CATALINA_HOME%\bin\startup
   ```

   This starts the Tomcat servlet container in the background.

9. Browse http://localhost:8080/refentry and you should see the default page of the RefEntry servlet.

If you want to stop Tomcat, the command is: %CATALINA_HOME%\bin\shutdown

Uninstalling

1. If you are running the version of Apache Tomcat installed with this software, then stop the Tomcat server.

   ```
   (Linux) $ $CATALINA_HOME/bin/shutdown.sh
   (Windows) C:\Windows>%CATALINA_HOME%\bin\shutdown
   ```

2. If you added any environment variables to your environment manually, then retrace your steps to remove them manually.

3. If you are using Windows 95 or 98 and you copied Ant to some short path like `C:\ant`, then delete the folder.
4. Run the self-extracting uninstaller, `uninstaller.jar`.

   ```
   (Linux) $ java -jar install-dir/Uninstaller/uninstaller.jar
   (Windows) Double-click install-dir/Uninstaller/uninstaller.jar.
   ```

You may delete any remaining files at this point.

Chapter 2. Using the RefEntry Servlet and Client

This chapter provides procedures and examples showing how to use the servlet and client interface.

Using the Client

Once the servlet has been installed, you may be able to browse the content without instructions, entering keywords and clicking `Search`, as long as you are familiar with a web browser. Simply point the browser to the `http://`*`Container URL`*`/refentry`, such as `http://localhost:8080/refentry`, and begin searching.

You may consider as drawbacks of this version of the servlet that it supports only keyword searches, and that it returns every page it finds without ranking the pages. When refining a search, keep in mind that all words shorter than 3 letters, such as `to`, `an`, `ID` and so forth, are ignored. Furthermore, a configurable set of default words are also ignored. By default this includes `the`, `and`, `you`, `that`, `for`, `was`, `are`, `with`, `his`, `they`, `this`, `from`, `have`, `one`, `had`, `not`, `but`, `what`, `all`, `were`, `when`, `there`, `can`, `your`, and `which`.

For example, if the URL to the servlet is `http://localhost:8080/refentry`, you enter this in your browser address bar. To find all documents containing the word `refentry`, you type `refentry` into the search bar and click `Search`. It does not help to type `the refentry for you` in the default servlet, as the first, third, and fourth words in the search string are ignored. If you want to extend the results to include all documents containing `refentry` or `element`, notice that after you search for `refentry`, the servlet returns a page with `refentry` in the search bar. You do not have to type the word again.

Creating RefEntry Documents

For this section it is assumed you have some experience editing XML documents, that you know how to handle DTDs in your environment, and that you know enough about DocBook reference entries and **man** pages to write documents against the RefEntry DTD without breaking much of a sweat. If you do not feel comfortable editing SGML/XML documents in your environment, consider reading Bob DuCharme's book, *SGML CD*. Bob shows you how to get started with free software tools, so you do not have to pour money in before you even get started. (In my experience, the free tools are easier to use than the expensive ones, anyway.) If you do not feel comfortable writing **man** pages in DocBook, consider Norman Walsh's *DocBook: The Definitive Guide*. You can get a preview from http://docbook.org/. This servlet comes with a number of examples, too. My markup is less informed than Norman's, but then my DTD is a lot simpler than DocBook.

In addition to prerequisite knowledge, you'll also need an XML parser, such as Xerces, which you may have installed with the other software as part of the HttpUnit pack. You cannot use the servlet **format** tool without `org.apache.xerces.parsers.SAXParser` in your CLASSPATH. You may decide that you want an alternative XSLT processor as well for general purposes, in which case a useful free tool is Xalan for Java from the Apache Software Foundation, somewhere under http://xml.apache.org/. If you have the Xalan `*.jar` files in your CLASSPATH, you also have the Xerces `SAXParser`.

To Get Started

Follow these steps:

1. Unpack the servlet content into a working directory using the **jar** utility.

   For example:

   ```
   $ mkdir -p working-dir
   $ cd working-dir
   $ jar -xvf install-dir/refentry.war
   ```

2. Set your CLASSPATH to include at least the Xerces SAX Parser, the current directory, and the `WEB-INF/classes` directory where you unpacked the servlet content.

   For example:

   ```
   $ export CLASSPATH=.:working-dir/WEB-INF/classes:$CLASSPATH
   $ export CLASSPATH=$CLASSPATH:/path/to/xercesImpl.jar
   ```

   If you installed the HttpUnit package and set your CLASSPATH as described in the installation procedure, you need only set the current directory and the `WEB-INF/classes` directory. HttpUnit comes with a Xerces parser implementation.

3. Set up your system to use the `refentry.dtd` when editing.

   This step depends on your system, and you may have to hack a little to get this to work. On Red Hat Linux 7.2, I had some difficulty when using both DocBook (to write this document) and the RefEntry DTD (to write RefEntry documents) at the same time. Maybe Emacs in PSGML mode had some trouble with both a DocBook RefEntry document type and a system RefEntry document type.

To Create Your First RefEntry

Follow these steps:

1. Create a new file, `test.xml`, with the following content:

```
<?xml version="1.0"?>
<!DOCTYPE refentry SYSTEM "working-dir/src/refentry.dtd">
<refentry>
  <refmeta>
    <refentrytitle>test</refentrytitle>
    <manvolnum>test</manvolnum>
  </refmeta>
  <refnamediv>
    <refname>test</refname>
    <refpurpose>test RefEntry servlet code</refpurpose>
  </refnamediv>
  <refsynopsisdiv>
    <title>SYNOPSIS</title>
    <synopsis>A synopsis.</synopsis>
  </refsynopsisdiv>
  <refsect1>
    <title>REFSECT1</title>
    <para>A reference section.</para>
  </refsect1>
</refentry>
```

Depending on how you set up you set up your environment to edit the document, you may want to change the document type declaration line.

2. Convince yourself that you can edit this document as you intend.

   In particular, if you expect your editor to be aware of the DTD and to facilitate your work, verify that the editor actually does help you insert the elements and attributes as you edit.

   Experience has shown me that using an XML-aware editor is a good way to get to know a DTD.

To Create Check RefEntry Validity

Follow these steps:

1. Ensure you have set your CLASSPATH appropriately as described in Procedure 2.1, *To Get Started*.
2. Change to the directory where you unpacked the servlet content.
3. Run the **format** command on the file.

   For example:

```
$ java format -infile working-dir/src/test/test.xml
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>test(test) - test RefEntry servlet code</title>
</head>
<body bgcolor="white">
<form>
<input name="SrchString" size="60" value="" />
<input type="submit" value="Search" />
</form>
<hr />
<div>
<table width="100%" border="0">
<tr>
<td align="left">test(test)</td><td align="center">Section unknown (see refentry.x
</tr>
</table>
<h1>NAME</h1>
<blockquote>
<tt>test</tt> - test RefEntry servlet code</blockquote>
<h1>SYNOPSIS</h1>
<blockquote>
<p>
<tt>A synopsis.</tt>
</p>
</blockquote>
<h1>REFSECT1</h1>
<blockquote>
<p>A reference section.</p>
</blockquote>
</div>
<table border="0" width="100%">
<tr>
<td align="left">Generated [Sun Feb 23 07:22:52 CET 2003]</td>
```

```
<td align="right">Server [CLI format]</td>
</tr>
</table>
</body>
</html>
```

At this point, if your XML is not valid, the format command tells you why.

For example:

```
$ java format -infile src/test/invalid.xml
==>Fatal parse error<==
  Line:    21
  URI:     null
  Message: XML document structures must start and end within the same entity.
org.xml.sax.SAXException: Fatal parse error encountered
Input file is not valid.
```

Building A Servlet of Your Own

Once you have a bunch of RefEntry documents you are happy with, you can build your own servlet containing them. The servlet content comes with an Ant build file for generating your own servlet.

Before generating the servlet, you must install both Ant, which can be installed with the project software, and JUnit, which is installed when you install the HttpUnit pack with the project software. Ant is indispensable; JUnit is strongly recommended. If you choose not to install the HttpUnit pack, which contains the JUnit framework and other software used for functional unit testing, resist the temptation to hack at any of the Java code, and edit the build file, `build.xml`, to remove dependencies on the `accept` target.

To Adjust The List of RefEntry Documents

Follow these steps:

1. Change to the `src/conf/` directory where you unpacked the servlet content.
2. Edit the `refentrys.props` Java properties file in which each line is of the form:

   `refentryID=/path/to/refentry.xml`

   Here, *`refentryID`* must be equivalent to *`refentrytitle-manvolnum`* for the servlet to work.

(Optional) To Change The List of Words to Ignore

Follow these steps:

1. Change to the `src/conf/` directory where you unpacked the servlet content.
2. Edit the `cw.props` Java properties file such that the `CommonWords` property includes a list of the words to ignore when indexing RefEntry documents in the servlet and when handling searches from client browsers.

(Optional) To Modify Stylesheet Formatting

Follow these steps:

1. Change to the `src/xslt/` directory where you unpacked the servlet content.
2. Edit only `refentry.xslt`.

   `refentry.xslt` changes the layout of the resulting XHTML document, but does not affect how the servlet itself functions. Other XSLT files in the directory are used by the servlet, so modify them at your peril.

   For example, you may add a ManVolNum section title, by adding and xsl:when element similar to the existing elements under the xsl:variable element near the top of the file.

To Build Your Servlet

Follow these steps:

1. Ensure Ant and JUnit are installed.
2. Change to the directory where you unpacked the servlet content.
3. (Optional) If you did not install JUnit, edit `build.xml` to remove `accept` from the list of dependencies for the `war` target.
4. Run **ant**.

   For example:

```
$ ant
Buildfile: build.xml

init:
    [mkdir] Created dir: /home/mcraig/refentry/build
    [mkdir] Created dir: /home/mcraig/refentry/build/build
    [mkdir] Created dir: /home/mcraig/refentry/build/classes
    [mkdir] Created dir: /home/mcraig/refentry/build/copy
    [mkdir] Created dir: /home/mcraig/refentry/build/docs
    [mkdir] Created dir: /home/mcraig/refentry/build/docs/api
    [mkdir] Created dir: /home/mcraig/refentry/build/serial

compile:
    [javac] Compiling 20 source files to /home/mcraig/refentry/build/classes

accept:
     [java] ......
     [java] Time: 9.064

     [java] OK (6 tests)


servletunit:
    [javac] Compiling 4 source files

tool:
    [javac] Compiling 1 source file
```

```
copy:
     [copy] Copying 1 file to /home/mcraig/refentry/build/build
     [copy] Copying 1 file to /home/mcraig/refentry/build/build
     [copy] Copying 1 file to /home/mcraig/refentry/build/build
     [copy] Copying 1 file to /home/mcraig/refentry/build/build
     [copy] Copying 1 file to /home/mcraig/refentry/build/build
     [copy] Copying 1 file to /home/mcraig/refentry/build/build
     [copy] Copying 1 file to /home/mcraig/refentry/build/build
     [copy] Copying 1 file to /home/mcraig/refentry/build/build
     [copy] Copying 1 file to /home/mcraig/refentry/build/build
     [copy] Copying 1 file to /home/mcraig/refentry/build/build
     [copy] Copying 1 file to /home/mcraig/refentry/build/build
     [copy] Copying 1 file to /home/mcraig/refentry/build/build
     [copy] Copying 1 file to /home/mcraig/refentry/build/build
     [copy] Copying 142 files to /home/mcraig/refentry/build/copy

javadoc:
  [javadoc] Generating Javadoc
  [javadoc] Javadoc execution
  [javadoc] Loading source files for package org.mcraig.cs445.refentry...
  [javadoc] Constructing Javadoc information...
  [javadoc] Standard Doclet version 1.4.1

  [javadoc] Building tree for all the packages and classes...
  [javadoc] Building index for all the packages and classes...
  [javadoc] Building index for all classes...

serialize:

war:
      [war] Building war: /home/mcraig/refentry/dist/refentry.war

BUILD SUCCESSFUL
Total time: 1 minute 41 seconds
```

This step may take a couple of minutes while the RefEntry documents are serialized.

5. (Optional) Use the `clean` target to delete generated files.

```
$ ant clean
```

For a full list of targets, read the `build.xml` file.

Chapter 3. RefEntry Client Design

This chapter covers client architecture and describes the user interface both for the browser based client and for the documentation tools.

WebMan Client Viewer

On the client side, only a browser capable of reading XHTML is required. The XHTML used must fit as much as possible the behavior of existing browsers. Empty tags such as <element /> must include the space for this to work well with older browsers.

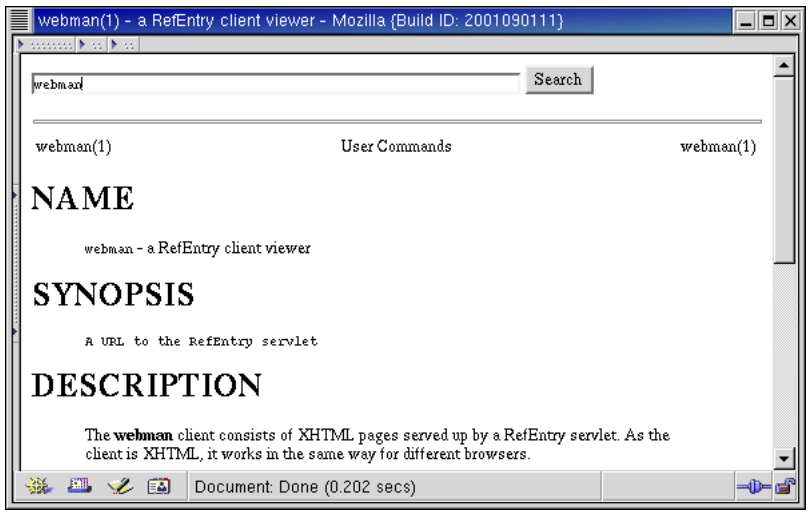

Figure 3-1. Default Client Page

The default client page is the RefEntry for the client browser itself. At the top of each **webman** page, a search bar allows the reader to enter a search string used by the servlet to find corresponding RefEntry pages. The servlet displays different **webman** pages, depending on the results of a search. This interface should be usable both in graphical user interface browsers, and in command-line user interface browsers. The command-line version should enable accessibility technologies such as text-to-speech or braille interfaces.

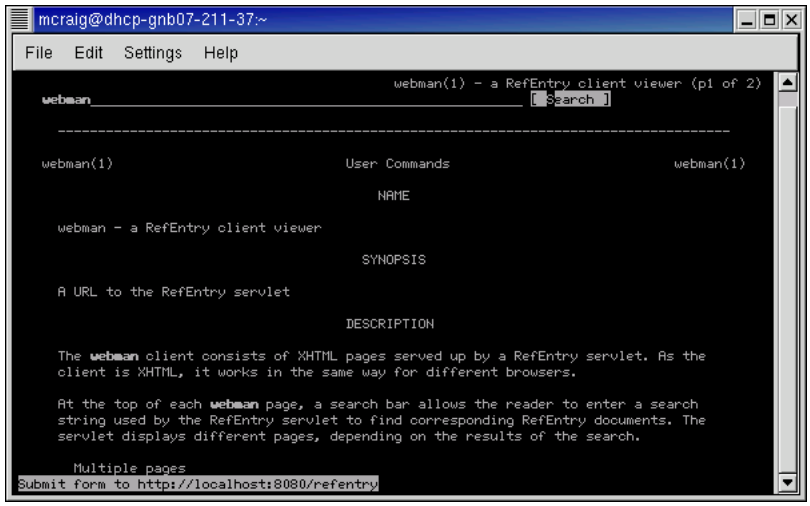

Figure 3-2. Default Client Page, CLI Browser

The XHTML itself should be accessible, such that it is easy to read without a graphical user interface.

The search interface works only for keyword searches, returning a union of everything found. This results in more search hits than desired in many cases, meaning the algorithm is not very useful for a large number of RefEntry documents. Implementing a more effective algorithm is something to be done in a subsequent version of the project.

When a search turns up more than one result, the client shows the RefName and RefPurpose for each result, allowing the user to select a further page.

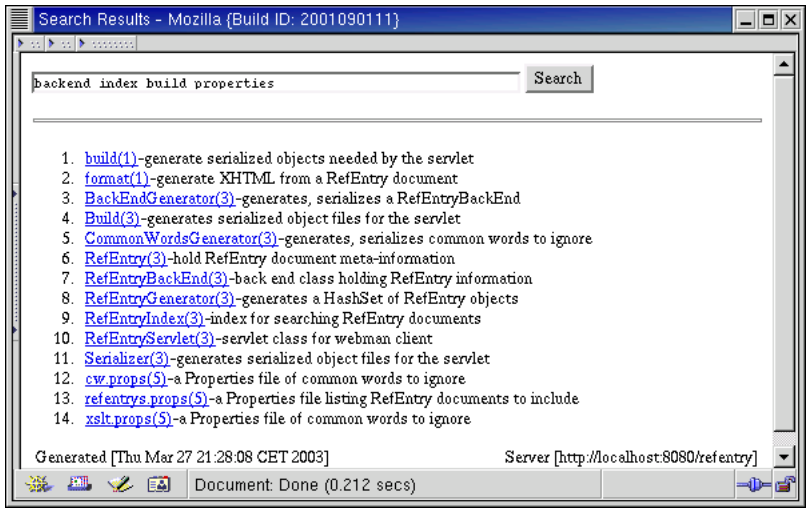

Figure 3-3. Multiple Results

The search results are always returned in the search area of the client page, so the user can easily refine a search based on what has already been typed.

Figure 3-4. No Results

If a particular search turns up no results at all, the page in the preceding figure is returned.

RefEntry Development

The documenter creating content marks up RefEntry documents using a text editor or specialized XML editor. Non-conforming documents are rejected by the formatter with detailed information as to where the XML document fails to conform either to the DTD or to the XML standard so the documenter can fix the problem.

The documenter works on an unpacked copy of the original RefEntry servlet. RefEntry documents to be included in the new servlet are listed in a configuration file. This allows the writer to locate the documentation wherever is convenient, but maintain total control over the documents handled by the servlet.

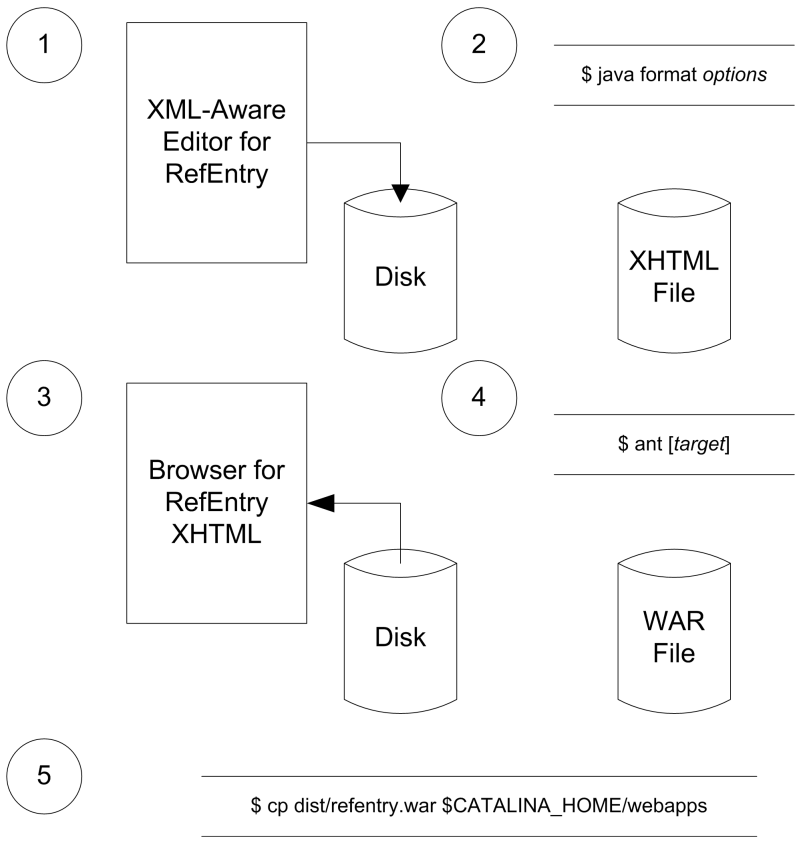

Figure 3-5. Content Development Cycle

After creating a RefEntry document, the documenter runs the **format** utility on the document to obtain an XHTML version. This allows the documenter to verify that the formatting appears as desired. Inter-document links may not work during the initial file by file formatting, but otherwise the **format** utility shows essentially the same final result as the servlet would. The documenter alternately browses and fixes bugs until satisfied with the output.

Once all pages have been developed and the documenter is satisfied with the results, the documenter edits a properties file specifying the content of the new servlet. The documenter may also edit a properties file specifying the words to ignore, and if changing the stylesheets, can edit the properties file specifying the XSLT stylesheets to use. The documenter then builds a new servlet using the Ant build file, `build.xml`. The new servlet can then be installed in a servlet container and used to serve RefEntry documents.

Development Tools

The following are available to documenters:

format

Name

`format` — generate XHTML from a RefEntry document

Synopsis

java format [-infile *`inputFile`*] [-outfile *`outputFile`*]

Description

The **format** command takes as input a RefEntry document conforming to the RefEntry DTD and uses the default stylesheet associated with the RefEntry servlet to transform the input into XHTML output.

Arguments

The **format** command takes the following optional arguments.

-infile *`inputFile`*

The XML RefEntry document to transform into XHTML. If not provided, the **format** uses standard input.

-outfile *`outputFile`*

The XML RefEntry document generated from the XML. If not provided, the **format** uses standard output.

Notes

The **format** command requires a Properties file, `src/conf/xslt.props`. Refer to xslt.props(5) for details.

ENVIRONMENT

The **format** command depends on XSLT tranformer packages, `javax.xml.transform` and `javax.xml.transform.stream`. It also requires that you include the `WEB-INF/classes` and current directory, `.`, in your `CLASSPATH`.

cw.props

Name

`cw.props` — a Properties file of common words to ignore

Synopsis

```
src/conf/cw.props
```

Description

The `cw.props` Java Properties file holds a single property whose value is a whitespace-separated list of common words the RefEntry servlet ignores both when servicing a search and also when indexing RefEntry documents. In other words, when a word is included in the list of common words to ignore, a search on that word returns no results. The default common words file appears as follows:

```
# Words to ignore when indexing and searching
# From the American Heritage Word Frequency Book, ISBN-0-395-13570-2.
CommonWords = the and you that for was are with his they \
this from have one had not but what all were when there \
can your which
```

Note that you do not need to include words of less than three characters in the list. The servlet automatically ignores words of less than three characters.

refentrys.props

Name

`refentrys.props` — a Properties file listing RefEntry documents to include

Synopsis

```
src/conf/refentrys.props
```

Description

The `refentrys.props` Java Properties file holds a property for each RefEntry source to format and include in the servlet. Each line takes the following format:

`refentry-id=relative/path/to/source`

Here, *`refentry-id`* is a unique identifier for the reference entry, defined from the source file name as *`basename-manvolnum`*, and *`relative/path/to/source`* is something like `src/xml/refentrys.props.5`.

An excerpt of the default file appears as follows:

```
# Individual RefEntry Documents
build-1=src/xml/build.1
Build-3=src/xml/Build.3
refentrys.props-5=src/xml/refentrys.props.5
programlisting-7=src/xml/programlisting.7
```

xslt.props

Name

`xslt.props` — a Properties file of common words to ignore

Synopsis

```
src/conf/xslt.props
```

Description

The `xslt.props` Java Properties file holds the list of XSLT stylesheets used to generate formatted RefEntry documents. The only stylesheet you can modify without also modifying the servlet source code is `refentry.xslt`. If you do opt to change the `refentry.xslt` stylesheet, ensure it continues to work with `refentry.dtd`. The default file appears as follows:

```
# Stylesheets used for transformations. You may modify RefEntry
# formatting by changing RefEntryXSLT. Changing other stylesheets may
# break the application.
ManVolNumXSLT  = src/xslt/manvolnum.xslt
RefEntryXSLT   = src/xslt/refentry.xslt
RefNameXSLT    = src/xslt/refname.xslt
RefPurposeXSLT = src/xslt/refpurpose.xslt
StripXMLXSLT   = src/xslt/stripxml.xslt
```

Chapter 4. RefEntry Servlet Design

This chapter covers the servlet architecture and specifies interfaces. The servlet contains essentially everything necessary to serve up RefEntry documents inside a single WAR file. A first goal for a future version could be to solve the scalability problem presented when everything is included in the same WAR file, perhaps using an external data repository such as a database or a directory.

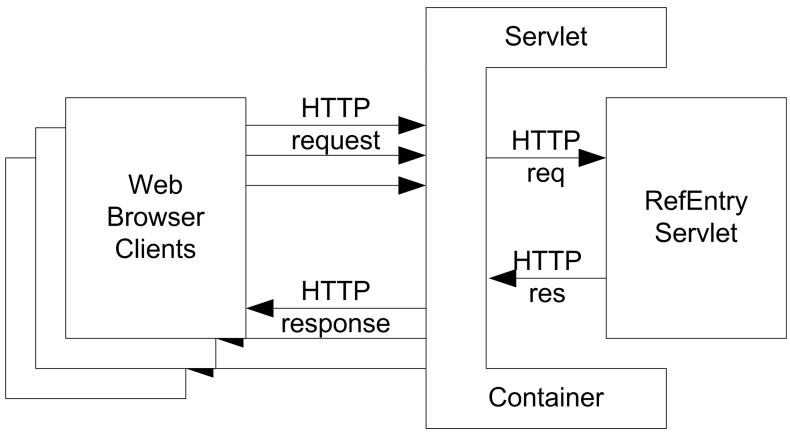

Figure 4-1. RefEntry Client and Servlet Together

The **webman** client viewer accesses the servlet through the servlet container. The servlet container takes responsibility for translating client browser requests into `HttpServletRequest` objects to the servlet, and for translating `HttpServletResponse` objects from the servlet into responses for the client browser. The servlet container also takes responsibility for managing several instances of the servlet to handle multiple threads.

Servlet Classes

The following classes make up the servlet.

RefEntry

Name

`RefEntry` — hold RefEntry document meta-information

Synopsis

```
class RefEntry {
```

```
  RefEntry(File infile, File outfile, Properties xslt);
  int compareTo(Object other);
  String getID();
  HashSet getKeys(HashSet ignore);
  String getManVolNum();
  String getRefName();
  String getRefPurpose();
  String getXHTML();
}
```

Description

`RefEntry` classes hold meta-information about RefEntry documents, enabling access by other classes.

Parameters

The `RefEntry` constructor takes an *`infile`* XML RefEntry input file, an *`outfile`* filename in which to store the XHTML output, and a Properties object specifying where the stylesheets for transformation are stored.

Methods

`RefEntry` class "friendly" (package private) methods provide access to various data used by the servlet, and makes it possible to sort groups of `RefEntry` objects.

compareTo

Allows sorting of lists of `RefEntry` objects.

getID

Returns a string of the form *`refname-manvolnum`*.

getKeys

Returns a set of keys on which to index the `RefEntry`.

getManVolNum

Returns the section name for the document.

getRefName

Returns the name of the item referenced.

getRefPurpose

Returns the purpose of the item referenced.

getXHTML

Returns the page as an XHTML document division. Note that this is not an entire XHTML document, but only the core of the page.

RefEntryBackEnd

Name

`RefEntryBackEnd` — back end class holding RefEntry information

Synopsis

```
 class RefEntryBackEnd {
  RefEntryBackEnd(HashSet cw);
  boolean add(RefEntry page);
  HashSet getEntry(String id);
  HashSet getValues(HashSet keys);
}
```

Description

`RefEntryBackEnd` holds the information required to be able to search the servlet RefEntry documents. Its `add` method adds a RefEntry to the backend index. Its `getValues` method returns a set of RefEntry documents for the search keys.

Parameters

The `RefEntryBackEnd` constructor takes a set of common words *`cw`* to ignore when indexing RefEntry documents.

Methods

`RefEntryBackEnd` class "friendly" (package private) methods make it possible for the servlet to handle `RefEntry` objects.

add

: Add a RefEntry to the back end.

getEntry

: Retrieve a RefEntry through the back end, given the RefEntry identifier.

getValues

: Retrieve a set of RefEntry documents through the back end, given a set of keys on which the documents are indexed.

RefEntryIndex

Name

`RefEntryIndex` — index for searching RefEntry documents

Synopsis

```
class RefEntryIndex {
 RefEntryIndex();
 boolean add(String key, RefEntry value);
 HashSet match(String key);
}
```

Description

`RefEntryIndex` holds the index enabling full text searches. Its `add` method adds a RefEntry to the index for the word in `key`. Its `match` method retrieves a set of RefEntry objects for a `key` word.

Parameters

The `RefEntryIndex` constructor takes no parameters.

Methods

`RefEntryIndex` class "friendly" (package private) methods provide access to RefEntry documents used by the servlet when adding and searching.

add

: Adds an existing RefEntry to the index for a particular string. The string is folded to lower case.

match

: Returns a set of RefEntry documents matching a particular string. The string is folded to lower case.

RefEntryServlet

Name

`RefEntryServlet` — servlet class for webman client

Synopsis

```
 public class RefEntryServlet {
  RefEntryServlet();
  public void destroy();
  public String getServletInfo();
  public void init(ServletConfig config) throws ServletException;
  public void doGet(HttpServletRequest req, HttpServletResponse res);
  public void doPost(HttpServletRequest req, HttpServletResponse res);
}
```

Description

`RefEntryServlet` responds to the client searches for RefEntry documents. Its `init` method loads the `RefEntryBackEnd` instance for the servlet so that searches can be handled. Its `doGet` method handles client requests. The `doPost` method calls `doGet`.

Parameters

The `RefEntryServlet` constructor takes no parameters.

Methods

`RefEntryServlet` class "friendly" (package private) methods provide the standard interface.

destroy

Does nothing.

getServletInfo

Return a String of basic information about the servlet.

init

Reads the serialized backend from the disk and readies tools to handle the searches.

doGet

Handles client requests coming in over HTTP.

doPost

Calls `doGet`.

RefEntryTransformer

Name

`RefEntryTransformer` — transform RefEntry documents to XHTML

Synopsis

```
 class RefEntryTransformer {
  RefEntryTransformer(String path);
  String transform(HashSet refentries, String search);
  String transform(RefEntry source, String search);
  String transform(String search);
}
```

Description

`RefEntryTransformer` transforms zero or more RefEntry sources into XHTML using its `transform` methods.

Parameters

The `RefEntryTransformer` constructor takes as its only parameter the URL to the base of the servlet as a String.

Methods

The `RefEntryTransformer` `transform` methods take various parameters. When in doubt, include `RefEntry` objects in a `HashSet` and pass them to the method that takes a `HashSet` as a parameter. The `search` parameter is just the original search as a `String` as it would appear in the browser. That parameter may be `null`.

Servlet Packaging

The servlet is built using Ant. Source files are formatted during the build using the stylesheets provided. The build process serializes the `RefEntryBackEnd` for the servlet so the servlet is completely self contained for initialization. Everything is in the servlet WAR file. Because it is self-contained, the servlet can be rebuilt from its own contents. Because it loads everything into memory at initialization time, the servlet is also less susceptible to file system corruption while it runs, though memory corruption could conceivably ruin it easily, and servlets containing many, many documents could use significant amounts of memory.

The servlet WAR file includes the following content.

Table 4-1. Servlet Content

File Element	Description
`build.xml`	Ant build file
`docs/`	Project documentation source and Javadoc
`docs/README.txt`	Read this first
`docs/relnotes.txt`	Release notes for this release
`format.java`	Wrapper for **format** utility
`META-INF/`	Servlet manifest files
`src/`	Project source files
`src/conf/`	Servlet configuration files
`src/org/mcraig/cs445/refentry`	Java source files for the servlet
`src/refentry.dtd`	DocBook-based DTD for RefEntry documents
`src/test/`	Test RefEntry documents
`src/web.xml`	Web application descriptor
`src/xml/`	RefEntry XML sources
`src/xslt/`	XSLT stylesheets
`Test*.java`	System tests for finished servlet
`WEB-INF/`	Servlet generated files

All content, except the documentation graphics and documentation build files, can be retrieved by unpacking the archive.

Chapter 5. Test Plan and Results

This chapter lays out the testing strategy, specifies the specific tests used to qualify the application, and includes the resulting test report.

Test Plans

The cover the project documentation, the **webman** client, the **Format** utilities, the build process, and the RefEntry servlet itself.

The tests verify that the documentation:

- Reflects completely and accurately the architecture and implementation of the project
- Covers all procedures for using and installing the software
- Explains how to use all public features of the clients, including RefEntry development and stylesheet extension

The tests verify that the **webman** client:

- Provides readable output for a variety of browsers, including at least Mozilla, Netscape, Internet Explorer, and links (or lynx)
- Allows use of the search functionality on each of the browsers indicated
- Includes the search string in the output

The tests verify that the **Format** utility:

- Handles garbage input gracefully
- Allows use of customized stylesheets
- Produces useful, intelligible diagnostics when encountering problems

The tests verify that the build process:

- Produces a working servlet WAR file if at all possible, including all files specified for the WAR file architecture
- Provides diagnostics for missing files, corrupt configuration files, and other similar configuration problems
- Allows use of customized stylesheets

The tests verify that the RefEntry servlet:

- Permits multiple concurrent searches
- Handles garbage requests appropriately
- Returns simple search results for a single word within 2 seconds
- Is straightforward to install and use
- Complies with servlet standards for portability

In order to verify each of these points, functional, system, and installation tests are run for the second, third, and fourth deliveries respectively. Tests continue to be developed as the software and documentation take shape for the delivery. Before each delivery including software tests, a test result report is completed, and a list of bugs compiled. Bugs that are not to be fixed are identified as such in the release notes for the delivery.

The test results are cumulative. Also, once tests are developed, they are run regularly to check for regressions in the software.

Functional Tests

Functional tests are developed during implementation. Their overall behavior is complete when the interface definition is complete. They verify the following.

- All public and friendly methods respond as documented.
- All elements are formatted as described in the documentation.

Functional tests may be installed alongside the software and may be run through the JUnit test framework.

System Tests

System tests are defined and developed during components integration. Their overall behavior is complete when the components are integrated together. They verify the following.

- Configuration file content has the documented effect on the build results.
- Concurrent searches may be run against the servlet. The servlet does not corrupt requests or results for concurrent searches. Servlet performance does not degrade steeply for reasonable numbers of clients (given machine power and hardware).
- The servlet handles garbage client requests gracefully.

System tests run alongside the servlet and clients. Some of the system tests (comparing documents) may be at least partially manual.

Installation Tests

Installation tests are defined and developed as the installation procedures are being defined. They verify the following.

- Installing over an existing copy does not overwrite what has been changed by the user.
- Improper installation configuration causes the servlet to provide useful, intelligible diagnostic messages.
- Installation procedures are clear and easy to follow. Given that all the prerequisite software is correctly installed, installation of the default servlet (without added RefEntry documents) should take no more than 10-15 minutes for a novice.

- Development and installation of modified servlets works as documented.

Test Results

This section covers test results for the second, third, and fourth deliveries.

Results for the Second Delivery

These results were gathered on 21 December 2002. They reflect the state of the tests and code at that time. As mentioned in the release notes (`relnotes.txt`), three bugs remain following testing for this delivery. Many other bugs were fixed during formal and informal unit testing.

The tests at the time of these results are mainly functional, but do not fully cover the all the features of the current code. The primary functional unit testing is performed using the JUnit test framework for Java. JUnit testing is lightweight and mostly easy to write, so it has been used for *acceptance* testing. Acceptance testing in this context means running a suite of lightweight non-regression tests during development every time code is compiled. The theory is that with acceptance testing, you catch many bugs right as they are coded. While the acceptance tests used are not perfect, they have helped prevent slipping back behind where the code is now. JUnit tests are relatively easy to manage, so it was also easy to comment out tests when locating bugs.

The results of the acceptance tests are not very visibly thrilling when things go well. For example, the code currently passes the acceptance tests.

```
[mcraig@localhost java]$ java org.mcraig.cs445.refentry.AcceptanceTest
......
Time: 9.725

OK (6 tests)

[mcraig@localhost java]$
```

The acceptance tests check most of the exposed methods in the `org.mcraig.cs445.refentry` package. The release notes explain why some methods are not checked. If you are wondering why there are only six tests in the acceptance suite, it is due to how they are packaged. Refer to the code listings in the appendix for details.

In addition to the acceptance tests, a shell script has been used to test the **Format** tool.

```
[mcraig@localhost test]$ ./Format.sh
~/cvs/cs445/java ~/cvs/cs445/test
************************************************************
Testing simple but valid input and output...
The input is test.xml:
<?xml version="1.0"?>
<!DOCTYPE refentry SYSTEM "../refentry.dtd">
<refentry>
  <refmeta>
    <refentrytitle>test</refentrytitle>
    <manvolnum>test</manvolnum>
  </refmeta>
  <refnamediv>
    <refname>test</refname>
    <refpurpose>test RefEntry servlet code</refpurpose>
  </refnamediv>
  <refsynopsisdiv>
    <title>SYNOPSIS</title>
```

```
    <synopsis>A synopsis.</synopsis>
  </refsynopsisdiv>
  <refsect1>
    <title>REFSECT1</title>
    <para>A reference section.</para>
  </refsect1>
</refentry>

The output is as follows:
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>test(test) - test RefEntry servlet code</title>
</head>
<body bgcolor="white">
<form>
<input name="SrchString" size="60" value="" />
<input type="submit" value="Search" />
</form>
<hr />
<div>
<table width="100%" border="0">
<tr>
<td align="left">test(test)</td>
<td align="center">Section unknown (see refentry.xslt)</td>
<td align="right">test(test)</td>
</tr>
</table>
<h1>NAME</h1>
<blockquote>
<tt>test</tt> - test RefEntry servlet code</blockquote>
<h1>SYNOPSIS</h1>
<blockquote>
<p>
<tt>A synopsis.</tt>
</p>
</blockquote>
<h1>REFSECT1</h1>
<blockquote>
<p>A reference section.</p>
</blockquote>
</div>
<table border="0" width="100%">
<tr>
<td align="left">Generated [Sat Dec 21 21:53:17 CET 2002]</td>
<td align="right">Server [CLI format]</td>
</tr>
</table>
</body>
</html>
*****************************************************************


This delivery does not support validation inside Format or Build.
You must perform validation yourself using software such as the
Sun Multi-Schema XML Validator written by Kohsuke Kawaguchi.

*****************************************************************
Testing some garbage input...
The input is this test file...
#!/bin/bash
JAVADIR='../java'
COMMAND='java org.mcraig.cs445.refentry.Format'
TESTDIR='pwd'

pushd $JAVADIR
echo "*****************************************************************"
echo "Testing simple but valid input and output..."
echo "The input is test.xml:"
cat $TESTDIR/test.xml
echo
echo "The output is as follows:"
$COMMAND -infile $TESTDIR/test.xml
echo
echo "*****************************************************************"
echo
echo
echo This delivery does not support validation inside Format or Build.
echo You must perform validation yourself using software such as the
echo Sun Multi-Schema XML Validator written by Kohsuke Kawaguchi.
echo
```

```
echo "****************************************************************"
echo "Testing some garbage input..."
echo "The input is this test file..."
cat $TESTDIR/Format.sh
echo
echo "The output is as follows:"
$COMMAND -infile $TESTDIR/Format.sh
echo
echo "****************************************************************"


The output is as follows:
javax.xml.transform.TransformerException: org.xml.sax.SAXParseException:\
 Content is not allowed in prolog.
java.lang.NullPointerException
javax.xml.transform.TransformerException: org.xml.sax.SAXParseException:\
 Content is not allowed in prolog.
java.lang.NullPointerException
javax.xml.transform.TransformerException: org.xml.sax.SAXParseException:\
 Content is not allowed in prolog.
java.lang.NullPointerException
javax.xml.transform.TransformerException: org.xml.sax.SAXParseException:\
 Content is not allowed in prolog.
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>null(null) - null</title>
</head>
<body bgcolor="white">
<form>
<input name="SrchString" size="60" value="" />
<input type="submit" value="Search" />
</form>
<hr />
<table border="0" width="100%">
<tr>
<td align="left">Generated [Sat Dec 21 21:53:21 CET 2002]</td>
<td align="right">Server [CLI format]</td>
</tr>
</table>
</body>
</html>
****************************************************************
[mcraig@localhost test]$
```

As mentioned in the output, the **Format** tool does not at this time perform validation on the input, so the tool does not handle garbage gracefully. Instead, users must currently find their own validator, and ensure their files comply with the DTD before sending them through **Format**. The parser exception text is not easy to decipher either for a user unfamiliar with the XML specification.

Part of the testing done now is checking the documentation against the code. In other words, checking essentially that the RefEntry documents in section 3 match the **javadoc** results. Such testing was done using recent **javadoc** output, and comparing with the contents of this documentation. Bugs were fixed during the testing process.

The final testing done at this time concerns the appearance of the output in various browsers. This testing amounted to inspecting the results in various browsers. The output file used was `bigtest.xml`, a compendium of nearly everything supported by `refentry.dtd`.

The following snapshots show some results.

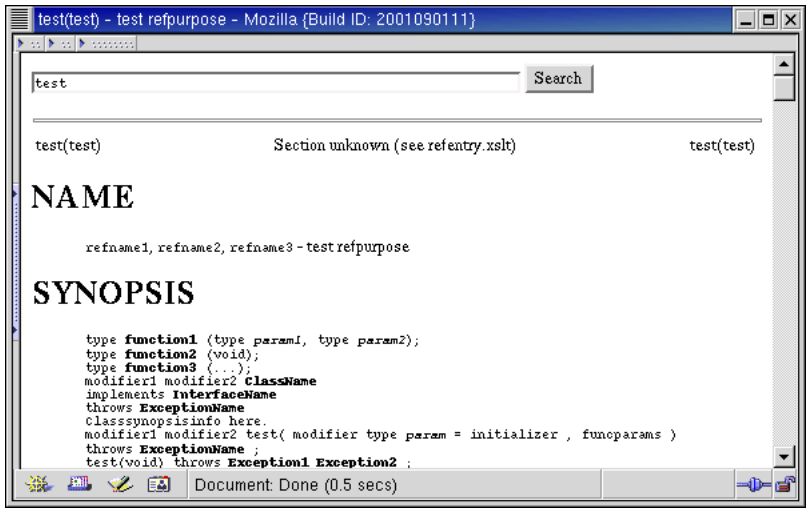

Figure 5-1. `bigtest.html` in Mozilla

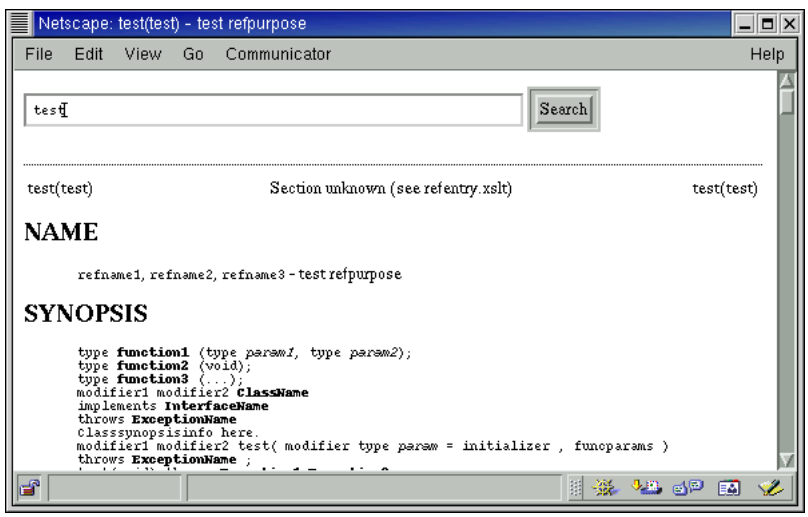

Figure 5-2. `bigtest.html` in Netscape 4

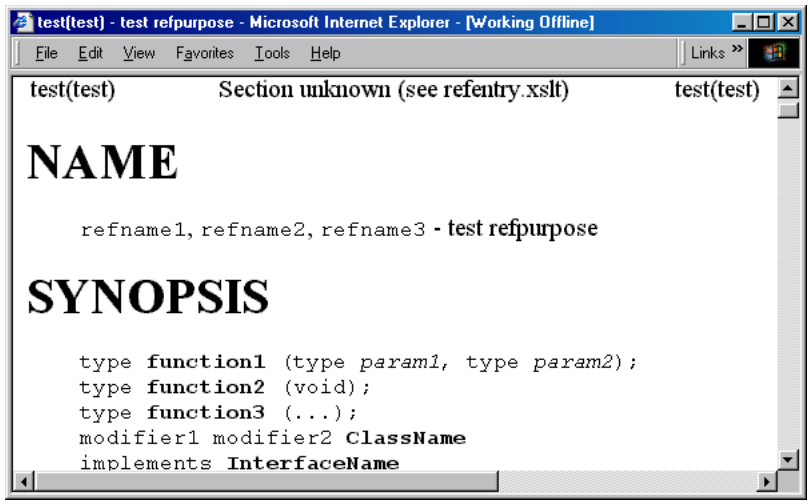

Figure 5-3. `bigtest.html` in Internet Explorer

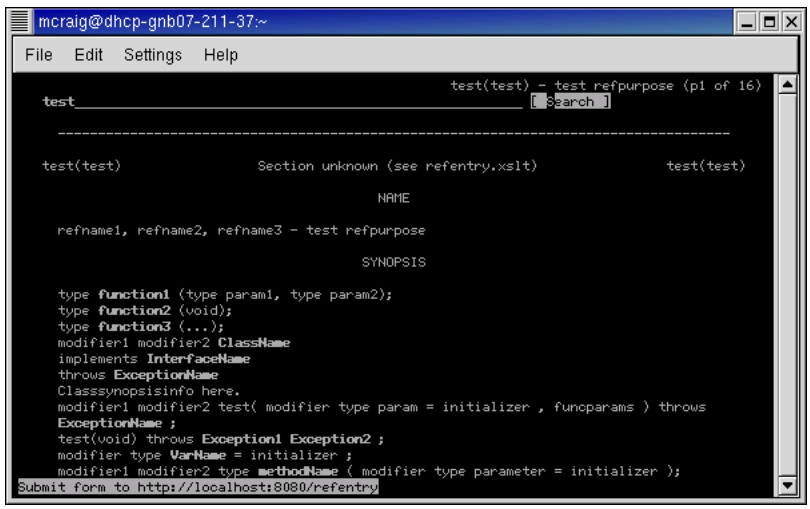

Figure 5-4. `bigtest.html` in Links

Clearly, the text to be generated in the XRef links did not work at the time of these tests. That should be fixed in a subsequent delivery.

Results for the Third Delivery

These results were gathered on 24 February 2003.

The servlet architecture was refined from delivery two to delivery three. The acceptance tests, functional unit tests that work in the JUnit framework, now run as a build task every time the servlet is built.

The file layout also changed considerably after I read suggestions in the documentation for Apache Cactus (although I ended up not using Cactus). I now deliver all the sources inside the servlet `.war` file, and the layout reproduces itself that way. The current layout is supposed to reflect a more "standard" layout for web application projects.

This round of system tests and documentation centers on five items:

1. Documentation accurately and completely explains what you need to do to use the servlet (but not to install it, yet).
2. The **format** utility handles garbage gracefully and generates useful diagnostics when problems are encountered.
3. The build process generates usable diagnostics when things go awry.
4. The servlet works for a variety of browsers, including the search string in the output.
5. The servlet handles garbage requests gracefully, complies with servlet standards for portability, and allows multiple, concurrent searches without performance degradation.

The following section explain what was done for each of these items.

Testing Documentation

I wrote the first design documentation at the beginning of the project. Since then, I have refined that design documentation several times to reflect changes made to work around some issue I was not aware of at the outset, or to implement some improvement learned about along the way. The user documentation I wrote at the end, however, working with the servlet in its current state. I have gone back and checked this documentation to verify that the procedures do work.

One step depends mightily on the user, however. At one point, I leave it up to the user to handle making the DTD available to the user's editor, if necessary. I'm not sure how else to explain that, other than to restrict the user to a single editor such as Emacs with PSGML. Users on Windows systems may be comfortable with other XML editors, such as XML Spy from Altera, however. I do not know whether Altera products support use of an SGML_CATALOG_FILES variable or if they handle DTDs some other way.

Testing the format Utility

For delivery two, the **format** utility was not ready for testing with both valid and invalid documents. For this release it can handle both, so I wrote an acceptance test to check. An example of how it handles both is shown in the user documentation.

It seems to me unnecessary to test this command heavily with a series of separate tests as most of the work is done with the `RefEntryTransformer` it uses, and additionally by the `SAXParser`, which throws the errors and warnings needed to debug RefEntry documents. Instead, I run the acceptance test to verify nothing is seriously broken at build time.

Testing the Build Process

In the original design and first draft implementation, I wrote Java code directly to build the servlet. Then I discovered Ant. Ant is a sort of reimplementation of **make** that looks easier to use and is native to Java. Had I known about it at the outset, I would not have tried to write my own servlet builder. I was trying to avoid **make** because you often cannot find it on Windows systems.

Ant takes care of the build error messages and many other issues in building the servlet. Once I learned some Ant, it was much easier to break my build process down into bite-sized pieces and to fix small issues I found in the layout. I was also able to integrate the acceptance tests into the build process to run them iteratively as a target, `accept`, on which the build depends.

```
accept:
     [java] ......
     [java] Time: 9.064

     [java] OK (6 tests)
```

After building the servlet, I install it in the servlet container. I have used Tomcat 4.1.18. Next, I run a few manual tests on the servlet to check that backend, common words, and links work. For example, open the servlet to the default page using the URL, `http://localhost:8080/refentry/`, then I search for `build`. This results in a page of links to other entries. Searching for `the build` does not change that list. From the list, I click the link for Build(3), and check that the link inside that page to build(1) works correctly.

The last of these manual tests caught a bug that I had in my `refentry.xslt` stylesheet. It turns out I was creating full, rather than relative links. In other words, I was generating static links to `http://mcraig.org/refentry/` rather than to pages inside the servlet. Easy to fix, but I might not have seen that without relatively systematic testing.

Finally, I also run post-integration tests to check that the servlet handles both garbage and a couple of acceptable requests without difficulty. These tests are written using the JUnit and HttpUnit test frameworks.

```
$ java TestGarbageRequests
.
Time: 4.221

OK (1 test)

$ java TestOkayRequests
.
Time: 3.106

OK (1 test)
```

```
$
```

If all goes well with those tests, I feel satisfied that the servlet I built basically works.

Testing a Variety of Browsers

To set this up, I installed the servlet on my workstation at work and ran the basic manual tests on Internet Explorer, Netscape Navigator, Mozilla, links, and Galeon. (Galeon is a GNOME browser based on Mozilla.)

Everything seems to function, although I admit my XHTML is ugly. Maybe I need to learn something about attractive page layout. I solemnly swear to do that as soon as I have learned to stop taking my T-shirts of the top of the pile.

Testing Performance

To simulate multiple, simultaneous client access, I put the servlet on my workstation and ran the `TestClient.java` thread 100 times, more or less in parallel, using `TestThreads.java` test case from a 10-processor server in our QA lab. I had to modify `TestThreads.java` slightly to change `localhost` strings to `celadon.france.sun.com`, but otherwise I ran the test as is.

The results show that response time degrades as the load increases. I examined the test output by first using UNIX tools to massage it into comma-separated variable format, then using a spreadsheet program to analyze the results. I found the following times:

Table 5-1. Response Times For 100 "Simultaneous" Threads

Value	Number of Seconds
Average	21.58
Median	23.72
Maximum	26.21
Minimum	0.34

In other words, with 100 "simultaneous" users, the servlet response is more than 10 times too slow. I am not sure what can be done about this, but it is definitely not the required behavior. For now, this problem is being logged as a bug to fix in the release notes. Perhaps it would suffice to change the way Tomcat is configured to handle the servlet. Perhaps implementing `SingleThreadModel` simply does not scale for many client requests. In that case, improving performance might involve moving the locking to a smaller section of the servlet code.

Results for the Fourth Delivery

The test campaign for this delivery aims to validate as much of the planned functionality as possible, including that checked by functional, system, and installation tests specified above. No new automated test cases were written for this campaign;

instead, existing tests were reused, and other conditions checked by hand.

Possibly the most serious failure of the project with respect to initial expectations is failure of the "installation procedures [to be] clear and easy to follow." On Windows systems, users must set environment variables, for example. To my surprise, I found that even some colleagues working day in and day out on Solaris felt uncomfortable setting environment variables on UNIX systems, not to mention Windows systems.

Additionally, auxiliary configuration that could almost be considered installation, such as configuring an XML-aware text editor to use a DTD -- a task I have not described in detail -- proved to cause confusion even for highly qualified software architects, which perhaps explains why so few people edit documents directly in SGML and XML.

Despite those failings, the project software essentially fulfills initial expectations, as documented in the list below.

Final Test Results

Servlet public and friendly methods respond as documented.

> These methods are documented using Javadoc, but also as reference entries. The Javadoc and RefEntry documentation have therefore been compared, and the result has been compared with the design documentation. Where necessary, the documentation was aligned with the implementation, which is different in some aspects from the original specifications (provided with the first delivery).
>
> The methods are unit tested as part of the build process.

RefEntry document elements are formatted as described in the documentation.

> Verifications are indicated in the following table.

Table 5-2. RefEntry Element Formatting

Element	Verified using `bigtest.html`?
arg	Yes, in the `SYNOPSIS` section
attribution	Yes, as in the quote attributed to Lao Tsu and the other attributed to Lyman Bryson
blockquote	Yes, as in the two quotes mentioned for the attribution element
citerefentry	Yes, as shown at the outset of the `Paras and Lists` section; no check is made to ensure that the target document exists in the servlet, however
citetitle	Yes, as shown at the outset of the `Paras and Lists` section
classname	Yes, as shown at the outset of the `Paras and Lists` section

Element	Verified using `bigtest.html`?
classsynopsis	Yes, as appears in the `SYNOPSIS` section of the page
classsynopsisinfo	Yes, also shown in the `SYNOPSIS` section
cmdsynopsis	Yes, also shown in the `SYNOPSIS` section
command	Yes, as shown at the outset of the `Paras and Lists` section
computeroutput	Yes, as shown in the first example of the `Example` section; come to think of it, this element is probably useless
constructorsynopsis	Yes, as shown in the `SYNOPSIS` section
destructorsynopsis	Yes, also shown in the `SYNOPSIS` section
emphasis	Yes, as shown at the outset of the `Paras and Lists` section
entry	Yes, as shown in the various tables in the `Tables` section
errorcode	Yes, as shown at the outset of the `Paras and Lists` section
errorname	Yes, as shown at the outset of the `Paras and Lists` section
errortype	Yes, as shown at the outset of the `Paras and Lists` section
example	Yes, as shown in the `Example` section
exceptionname	Yes, as appears in the `SYNOPSIS` section
fieldsynopsis	Yes, also in the `SYNOPSIS` section
funcdef	Yes, also in the `SYNOPSIS` section
funcparams	Yes, also in the `SYNOPSIS` section
funcprototype	Yes, also in the `SYNOPSIS` section
funcsynopsis	Yes, also in the `SYNOPSIS` section
funcsynopsisinfo	Do not know; have not used this one yet even in `bigtest.xml`. Formatting as surrounding element is, however, the default behavior during the transformation so this should not be broken.
function	Yes, as shown in the `Paras and Lists` section
group	Yes, as shown in the `SYNOPSIS` section

Element	**Verified using `bigtest.html`?**
initializer	Yes, as shown in the `SYNOPSIS` section
interfacename	Yes, also shown in the `SYNOPSIS` section
itemizedlist	Yes, as shown in the `Paras and Lists` section
listitem	Yes, as shown in the `Paras and Lists` section
literal	Yes, as shown in the first paragraph of the `Paras and Lists` section
manvolnum	Yes, as shown in the `Paras and Lists` section
methodname	Yes, as shown in the `SYNOPSIS` section
methodparam	Yes, also shown in the `SYNOPSIS` section
methodsynopsis	Yes, also shown in the `SYNOPSIS` section
modifier	Yes, also shown in the `SYNOPSIS` section
msg	Yes, as shown in the `MsgSet` section
msgentry	Yes, also shown in the `MsgSet` section
msgexplan	Yes, also shown in the `MsgSet` section
msginfo	Yes, also shown in the `MsgSet` section
msglevel	Yes, also shown in the `MsgSet` section
msgmain	Yes, also shown in the `MsgSet` section
msgorig	Yes, also shown in the `MsgSet` section
msgrel	Yes, also shown in the `MsgSet` section
msgset	Yes, also shown in the `MsgSet` section
msgsub	Yes, also shown in the `MsgSet` section
msgtext	Yes, also shown in the `MsgSet` section
ooclass	Yes, as shown in the `SYNOPSIS` section
ooexception	Yes, also shown in the `SYNOPSIS` section
oointerface	Yes, also shown in the `SYNOPSIS` section
orderedlist	Yes, as shown in the `Paras and Lists` section
para	Yes, as shown all over the place
paramdef	Yes, as shown in the `SYNOPSIS` section

Element	Verified using `bigtest.html`?
parameter	Yes, also shown in the `SYNOPSIS` section
programlisting	Yes, as shown in the `Example` section
quote	Yes, as shown in the `SYNOPSIS` section
refentry	Yes, the whole thing is a RefEntry
refentrytitle	Yes, as shown in all the section titles
refmeta	Yes, as shown at the outset of the document
refmiscinfo	That's right, it does not appear
refname	Yes, as shown at the outset of the document
refnamediv	Yes, as shown at the outset of the document
refpurpose	Yes, as shown at the outset of the document
refsect1	Yes, as shown for each of the top-level sections
refsect2	Yes, as shown in the `MsgSet` and `Tables` sections
refsynopsisdiv	Yes, as shown at the top of the document
replaceable	Yes, as shown in the first paragraph of the `Paras and Lists` section
row	Yes, as demonstrated in the `Tables` sections
sbr	Yes, as shown in the command synopsis in the `SYNOPSIS` section
screen	Yes, as shown in the `Example` section
synopsis	Yes, as shown in the `SYNOPSIS` section, though the handling of whitespace at line ends has not been checked
table	Yes, as shown in the `Tables` sections
tbody	Yes, also shown in the `Tables` sections
term	Yes, as shown in the `Paras and Lists` section
tgroup	Yes, as shown in the `Tables` sections
thead	Yes, also shown in the `Tables` sections
title	Yes, as shown throughout the document
trademark	Yes, as shown in the first paragraph of the `Paras and Lists` section

Element	**Verified using `bigtest.html`?**
type	Yes, as shown in the `SYNOPSIS` section
ulink	Yes, as shown in the first paragraph of the `Paras and Lists` section
userinput	Yes, as shown in the `Example` section
varargs	Yes, as shown in the `SYNOPSIS` section
variablelist	Yes, as shown in the `Paras and Lists` section
varlistentry	Yes, also shown in the `Paras and Lists` section
varname	Yes, as shown in the `SYNOPSIS` section
void	Yes, as shown in the `SYNOPSIS` section
xref	Yes, as shown in the first paragraph of the `Paras and Lists` section

Documentation covers all public features of the clients including RefEntry document development and stylesheet extension.

Originally, the design called for a build client. That functionality has been delegated to Ant, which comes with its own extensive documentation.

The servlet output **webman** client is documented above, as is the **format** command. Both are relatively simple, so their documentation is short.

Configuration file content has the documented effect on build results.

This requirement was originally intended to refer to the Build configuration file. Since the switch to Ant, this requirement has certainly been met for the build (configuration) file, `build.xml`. Other configuration files, `cw.props`, `refentrys.props`, and `xslt.props`, were documented retrospectively according to how they impact the servlet.

Build process results in a working servlet `.war` file if at all possible, otherwise providing good diagnostics for configuration problems.

Before learning about Ant, my intention was to write a Java class by hand to perform the build. The project now depends on Ant for builds. Ant provides clear messages on its own. For example:

```
$ mv README.txt mvREADME.txt
$ ant
Buildfile: build.xml

init:
    [mkdir] Created dir: /test/tomcat/webapps/refentry/build
    [mkdir] Created dir: /test/tomcat/webapps/refentry/build/build
    [mkdir] Created dir: /test/tomcat/webapps/refentry/build/classes
    [mkdir] Created dir: /test/tomcat/webapps/refentry/build/copy
    [mkdir] Created dir: /test/tomcat/webapps/refentry/build/docs
    [mkdir] Created dir: /test/tomcat/webapps/refentry/build/docs/api
```

```
    [mkdir] Created dir: /test/tomcat/webapps/refentry/build/serial
    [mkdir] Created dir: /test/tomcat/webapps/refentry/dist

compile:
    [javac] Compiling 20 source files to
            /test/tomcat/webapps/refentry/build/classes

accept:
     [java] ......
     [java] Time: 9.991

     [java] OK (6 tests)


servletunit:

tool:

copy:
     [copy] Copying 1 file to /test/tomcat/webapps/refentry/build/build

BUILD FAILED
file:/test/tomcat/webapps/refentry/build.xml:64:
Warning: Could not find file /test/tomcat/webapps/refentry/README.txt to copy.

Total time: 27 seconds
```

Concurrent searches does not corrupt requests or results, and the servlet returns simple search results withing 2 seconds given a reasonable number of concurrent clients.

The bug for delivery three has been fixed. It turns out that the servlet need not implement `SingleThreadModel`, because the servlet only reads, never writes global data when handling a client request. My earlier understanding was flawed even after having taking the ACCIS course on programming languages that included explanations of how memory is typically managed, and how it is managed for implementations of the Java language. There are probably still many things about threading I do not understand, but at least now I understand that when a thread executes the code for a method, the thread has memory specifically for local objects created in that code. Those objects are not at risk of being modified by another thread calling the same method code.

The new results for handling client requests using the same performance test as for delivery three:

Table 5-3. Response Times For 100 “Simultaneous” Threads

Value	Number of Seconds
Average	0.40
Median	0.15
Maximum	1.90
Minimum	0.01

Servlet complies with standards for portability.

> The web application descriptor complies with the 2.3 version of the DTD, which you can download from Sun Microsystems, Inc. at http://java.sun.com/dtd/web-app_2_3.dtd. In principle, this means none of the exposed functionality is non-standard. In practice, the servlet functionality is so basic that it hardly stretches the limits of that allowed by the 2.3 standard.

Servlet handles garbage requests gracefully.

> This is shown by the `TestGarbageRequests.java` test.

```
$ java TestGarbageRequests
.
Time: 3.744

OK (1 test)
```

The browser-based servlet output (**webman** client) allows input of search strings, echoes search strings in output, and remains readable with a variety of different browsers.

> This has been shown in snapshots with the tests for delivery two. I have validated it by hand for the updated servlet.

Installation procedures are clear and easy to follow. Installation, up to installation of the default servlet, takes no more than 10-15 minutes for a novice.

> Catherine Hislam, one of my colleagues, installed the project software on 15 March 2003. It took Catherine 14 minutes from the time she started downloading the installer `.jar` file, to the time she had the default servlet running on her workstation. Catherine made a number of remarks about the installation, comments I have integrated into the project documentation. For example, she found references to CATALINA_HOME and Tomcat did not originally indicate clearly enough that Tomcat is a servlet container, and that CATALINA_HOME designates the Tomcat installation directory.
>
> Gordon Stretch, another colleague, installed the project software on 25 March 2003, in 13 minutes. Gordon had one remark: that I should replace the example JAVA_HOME path with a variable to prevent the temptation to copy and paste an incorrect path. I fixed this.
>
> Stuart Clements, also a colleague, installed the software on 25 March 2003 as well. It took Stu 18 minutes, but he was interrupted in the middle of the process to look at a bug. Stu gave me feedback on the order of the steps in the installation procedure, and I have fixed that based on his feedback.

Development and installation of modified servlets works as documented.

> This was documented retrospectively, and I used the servlet software itself to generate the listings present in the documentation.

Installing over an existing copy does not overwrite what has been changed by the user.

The installer configuration file is set to build an installer that overwrites only the top level text files, `install.txt`, `license.txt`, `README.txt`, and `relnotes.txt`.

This was verified by installing once in an `Original` directory, once in a `Modified` directory. After installation, the `refentry.profile` file was changed in the `Modified` directory. Then the servlet `refentry.war` file was copied to the Tomcat `webapps` directory, Tomcat started, and http://localhost:8080/refentry read to unpack the `refentry.war` file. Subsequently, a new `refentry.war` servlet was built after `bigtest.xml` was added to the list of RefEntry documents. This simulated a user creating a modified servlet. At this point, the differences were captured:

```
$ diff -r Original Modified > before.diff
```

Next, the project software was installed atop the files in Modified, the message stating that files might be overwritten was ignored. At this point, the differences were again captured:

```
$ diff -r Original Modified > after.diff
```

To demonstrate that the modified files had not been changed upon re-installation, the two difference files were checked for differences:

```
$ diff before.diff after.diff
$
```

The installer therefore did not overwrite modified files.

Customized servlets may use customized stylesheets.

The servlet code remains independent of particular stylesheets. For example, all default stylesheets take the extension `.xslt`:

```
$ find . -name "*.xslt"
./src/xslt/manvolnum.xslt
./src/xslt/stripxml.xslt
./src/xslt/refentry.xslt
./src/xslt/refname.xslt
./src/xslt/refpurpose.xslt
```

None of the code contains references to such files, however:

```
$ find . -name "*.java" -exec egrep -e '\.xslt' {} \;
$
```

As documented, the user may change `refentry.xslt` to support different formatting; such changes are handled transparently by the servlet.

Improper installation configuration results in useful, intelligible diagnostic messages.

Improper configuration results in diagnostic messages. Whether the messages are useful and intelligible depends on how much the user knows about the software installed. The messages about unset JAVA_HOME and Tomcat installation location environment variables seem relatively clear:

```
$ unset JAVA_HOME
```

```
$ unset CATALINA_HOME
$ mv tomcat timcat
$ . refentry.profile
Java Development Kit:
    JAVA_HOME does not appear to be set.
    Set JAVA_HOME and run this script again.
    For example, under bash:
    export JAVA_HOME=/usr/java/j2sdk1.4.1
Apache Tomcat:
    Apache Tomcat does not appear to be installed.
    This project is designed to use Tomcat as the
    servlet container. If you have an existing
    installation of Tomcat you want to use, set the
    CATALINA_HOME environment variable as described
    in the Tomcat doc and source this file again.
```

Should the installer skip the step of copying the `refentry.war` servlet file to the proper location, however, the user sees a 404 status message when browsing the servlet location: "The requested resource (/refentry) is not available." This is a Tomcat message, however. Messages from Ant and HttpUnit may be confusing as well to the inexperienced user.

Test result data was gathered through the month of March, 2003.

Appendix A. RefEntry Elements

This appendix describes the XML document type definition (DTD) elements and attribute lists to which supported RefEntry documents conform and specifies formatting expectations for the translation to XHMTL. It is presented as a series of reference entries in alphabetical order.

> **Note:** Other sections in this documentation use camel backed capitalization such as RefEntry for all element and attribute identifiers. Camel backed capitalization makes identifiers easier to read. Technically, however, all such identifiers occur in lower case in the document type definition. XML is case sensitive, so for exactness this section discusses refentry and manvolnum rather than RefEntry and ManVolNum.

arg

Name

`arg` — command argument element

Content Model

```
arg ::= (#PCDATA | replaceable)+
```

Description

`arg` elements hold command line arguments in command synopses of `refentry` documents. The parent node is `cmdsynopsis` or `group`.

Attributes

`arg` elements have `choice` and `rep` attributes.

`choice` attributes take one of the following values.

opt
: Optional argument

plain
: Plain argument

required
: Required argument

`choice` values are implied if not given.

`rep` attributes take the following values.

norepeat

Argument may not be repeated

repeat

Argument may be repeated

`rep` values are implied if not given.

Formatting Expectations

When the `choice` attribute is `optional`, surround with brackets (`[ arg ]`). When the `choice` element is `required`, surround with braces (`{ arg }`). When the `rep` element is `repeat`, follow with an ellipsis, (`arg ...`).

attribution

Name

`attribution` — whom to attribute a quote

Content Model

```
attribution ::= (#PCDATA)+
```

Description

`attribution` elements hold the name of the person to whom a quote is attributed. The parent node is `blockquote`.

Attributes

`attribution` elements have no attributes.

Formatting Expectations

Format as a paragraph following the body of the quote, aligned right, preceded by a dash and a space.

blockquote

Name

`blockquote` — lengthy quote in the midst of other text

Content Model

```
blockquote ::= attribution, para+
```

Description

`blockquote` elements set a quote off from the main body of text, typically because the quote is lengthy.

The parent node is `msgexplan`, `msgtext`, `refsect1` or `refsect2`.

Attributes

`blockquote` elements have no attributes.

Formatting Expectations

Format as an inset block element with wider margins than the surrounding block elements.

citerefentry

Name

`citerefentry` — reference to another RefEntry document

Content Model

```
citerefentry ::= refentrytitle, manvolnum
```

Description

`citerefentry` references another RefEntry document. The parent node is `para`.

Attributes

`citerefentry` elements have no attributes.

Formatting Expectations

Format as a link to the corresponding document, if that document is available.

citetitle

Name

`citetitle` — mention title of another document

Content Model

```
citetitle ::= (#PCDATA)+
```

Description

`citetitle` elements indicate the title of another document. The parent node is `para`.

Attributes

`citetitle` elements have no attributes.

Formatting Expectations

Format in italics.

classname

Name

`classname` — object class name

Content Model

```
classname ::= (#PCDATA)+
```

Description

`classname` elements hold the name of an object class from an object oriented programming language. The parent node is `classsynopsis`.

Attributes

`classname` elements have no attributes.

Formatting Expectations

Format in bold.

classsynopsis

Name

`classsynopsis` — syonpsis for an object class

Content Model

```
classsynopsis ::= (ooclass | oointerface | ooexception)+,
(classsynopsisinfo | constructorsynopsis | destructorysynopsis |
fieldsynopsis | methodsynopsis)*
```

Description

`classsynopsis` elements hold the synopsis of an object class from an object oriented programming language. The parent node is `refsynopsisdiv`.

Attributes

`classsynopsis` elements have no attributes.

Formatting Expectations

Format in fixed width font.

classsynopsisinfo

Name

`classsynopsisinfo` — information about an object class

Content Model

```
classsynopsisinfo ::= (#PCDATA)+
```

Description

`classsynopsisinfo` elements hold information about an object class from an object oriented programming language. The parent node is `classsynopsis`.

Attributes

`classsynopsisinfo` elements have no attributes.

Formatting Expectations

Format as surrounding element.

cmdsynopsis

Name

`cmdsynopsis` — object class name

Content Model

```
cmdsynopsis ::= (arg | command | group | sbr)+
```

Description

`cmdsynopsis` elements hold the synopsis of a command line tool. The parent node is `refsynopsisdiv`.

Attributes

`cmdsynopsis` elements have no attributes.

Formatting Expectations

Format in fixed width font.

command

Name

`command` — command line tool name

Content Model

```
command ::= (#PCDATA)+
```

Description

`command` elements hold the identifier of a command line tool. The parent nodes is `cmdsynopsis`, or `para`.

Attributes

`command` elements have no attributes.

Formatting Expectations

Format in bold.

computeroutput

Name

`computeroutput` — output on terminal screen

Content Model

```
computeroutput ::= (#PCDATA | replaceable)+
```

Description

`computeroutput` elements identify output displayed on a computer terminal screen. The parent node is `screen`.

Attributes

`computeroutput` elements have no attributes.

Formatting Expectations

Format as surrounding element.

constructorsynopsis

Name

`constructorsynopsis` — synopsis for an object constructor method

Content Model

```
constructorsynopsis := modifier*, methodname?, (methodparam+ | void),
exceptionname*
```

Description

`constructorsynopsis` elements hold synopses for object constructor methods. The parent node is `classsynopsis`, or `refsynopsisdiv`.

Attributes

`constructorsynopsis` elements have no attributes.

Formatting Expectations

Format in fixed width font. Surround the parameter list with parentheses, separating parameters with commas, finishing with a semicolon.

destructorsynopsis

Name

`destructorsynopsis` — synopsis for an object destructor method

Content Model

```
destructorsynopsis := modifier*, methodname?, (methodparam+ | void),
exceptionname*
```

Description

`destructorsynopsis` elements hold synopses for object destructor methods. The parent node is `classsynopsis`, or `refsynopsisdiv`.

Attributes

`destructorsynopsis` elements have no attributes.

Formatting Expectations

Format in fixed width font. Surround the parameter list with parentheses, separating parameters with commas, finishing with a semicolon.

emphasis

Name

`emphasis` — emphasized word or phrase

Content Model

```
emphasis ::= (#PCDATA)+
```

Description

`emphasis` elements mark an emphasized word or phrase. The parent node is `para`.

Attributes

`emphasis` elements have no attributes.

Formatting Expectations

Format in italic.

emphasis

entry

Name

`entry` — a table cell

Content Model

```
entry ::= para+
```

Description

`entry` elements hold the content of a table cell. The parent node is `row`.

Attributes

`entry` elements have `align`, `morerows`, and `valign` attributes. `align` attributes take one of the following values.

center
: Align the entry to the center of the table cell.

left
: Align the entry to the left of the table cell.

right
: Align the entry to the right of the table cell.

`morerows` attributes take a number value, indicating the number of additional columns rows spanned by the table cell. For example, a value of `2` indicates that the table cell spans three rows.

`valign` attributes take one of the following values.

bottom
: Align the entry to the bottom of the table cell.

middle
: Align the entry to the middle of the table cell.

top
: Align the entry to the top of the table cell.

Formatting Expectations

Format adjusting alignment according to alignment attributes. If `morerows` is set to a number, format the content to span that number of rows.

errorcode

Name

`errorcode` — code identifying an error

Content Model

```
errorcode ::= (#PCDATA)+
```

Description

`errorcode` elements mark an error code such as a unique identifier for an error, especially as part of a list of errors. The parent node is `para`.

Attributes

`errorcode` elements have no attributes.

Formatting Expectations

Format in fixed width font.

errorname

Name

`errorname` — name identifying an error

Content Model

```
errorname ::= (#PCDATA)+
```

Description

`errorname` elements mark an error name, especially as part of a list of errors. The parent node is `para`.

Attributes

`errorname` elements have no attributes.

Formatting Expectations

Format in fixed width font.

errortype

Name

`errortype` — type of error

Content Model

```
errortype ::= (#PCDATA)+
```

Description

`errortype` elements mark a type of error, especially as part of a list of errors. The parent node is `para`.

Attributes

`errortype` elements have no attributes.

Formatting Expectations

Format in fixed width font.

example

Name

`example` — sample use of the reference subject

Content Model

```
example ::= title, (para | programlisting | screen)+
```

Description

`example` elements hold an example of how to use the subject of the reference. The parent node is `refsect1`.

Attributes

`example` elements may take an `id` attribute. `id` attributes take name tokens, used for building links from elsewhere in the same `refentry` document.

Formatting Expectations

Format as a block element with the title followed by the rest of the content. If the `id` attribute is set, mark the location for cross referencing using the value of the attribute.

exceptionname

Name

`exceptionname` — name of an object exception

Content Model

```
exceptionname ::= (#PCDATA)+
```

Description

`exceptionname` elements mark the name of an exception in an object oriented programming language. The parent node is `constructorsynopsis`, `destructorsynopsis`, `methodsynopsis`, or `ooexception`.

Attributes

`exceptionname` elements have no attributes.

Formatting Expectations

Format in bold.

fieldsynopsis

Name

`fieldsynopsis` — synopsis for an object field

Content Model

```
fieldsynopsis ::= modifier*, type?, varname, initializer?
```

Description

`fieldsynopsis` elements hold a synopsis for an object field in an object oriented programming language. The parent node is `classsynopsis`, or `refsynopsisdiv`.

Attributes

`fieldsynopsis` elements have no attributes.

Formatting Expectations

Format in fixed width font.

funcdef

Name

`funcdef` — definition of a function

Content Model

```
funcdef ::= (#PCDATA | function)+
```

Description

`funcdef` elements hold a function definition. The parent node is `funcprototype`.

Attributes

`funcdef` elements have no attributes.

Formatting Expectations

Format as surrounding elements.

funcparams

Name

`funcparams` — function referenced through a function pointer

Content Model

```
funcparams ::= (#PCDATA)+
```

Description

`funcparams` elements hold a function referenced through a function pointer. The parent node is `paramdef`.

Attributes

`funcparams` elements have no attributes.

Formatting Expectations

Format as surrounding elements.

funcprototype

Name

`funcprototype` — prototype for a function

Content Model

```
funcprototype ::= funcdef, (varargs | void | paramdef+)
```

Description

`funcprototype` elements hold a function prototype. The parent node is `funcsynopsis`.

Attributes

`funcprototype` elements have no attributes.

Formatting Expectations

Format `funcdef` followed by arguments that are separated by commas and surrounded by parentheses. End with a semicolon.

funcsynopsis

Name

`funcsynopsis` — synopsis for a function

Content Model

```
funcsynopsis := funcsynopsisinfo?, funcprototype+
```

Description

`funcsynopsis` elements hold synopses for functions. The parent node is `refsynopsisdiv`.

Attributes

`funcsynopsis` elements have no attributes.

Formatting Expectations

Format in fixed width font.

funcsynopsisinfo

Name

`funcsynopsisinfo` — information needed to use a function

Content Model

```
funcsynopsisinfo ::= (#PCDATA | replaceable)+
```

Description

`funcsynopsisinfo` elements hold information needed to use a function, such as a header file declaration. The parent node is `funcsynopsis`.

Attributes

`funcsynopsisinfo` elements have no attributes.

Formatting Expectations

Format in as surrounding element.

function

Name

`function` — name of a function

Content Model

```
function ::= (#PCDATA)+
```

Description

`function` elements mark a function name. The parent node is `funcdef`.

Attributes

`function` elements have no attributes.

Formatting Expectations

Format in bold.

group

Name

`group` — group of command arguments

Content Model

```
group ::= (arg | sbr)+
```

Description

`group` elements hold groups of command line arguments in command synopses. The parent node is `cmdsynopsis`.

Attributes

`group` elements have `choice` and `rep` attributes.

`choice` attributes take one of the following values.

opt
: Optional group

plain
: Plain group

required
: Required group

`choice` values are implied if not given.

`rep` attributes take the following values.

norepeat
: Group may not be repeated

repeat
: Group may be repeated

`rep` values are implied if not given.

Formatting Expectations

When the `choice` attribute is `optional`, surround with brackets (`[ group ]`). When the `choice` element is `required`, surround with braces (`{ group }`). When the `rep` element is `repeat`, follow with an ellipsis, (`group ...`).

initializer

Name

`initializer` — initializer for a method parameter

Content Model

```
initializer ::= (#PCDATA)+
```

Description

`initializer` elements mark the name of an initializer for a method parameter in an object oriented programming language. The parent node is `fieldsynopsis`, or `methodparam`.

Attributes

`initializer` elements have no attributes.

Formatting Expectations

Format as surrounding element preceded by =.

interfacename

Name

`interfacename` — name of an object oriented interface

Content Model

```
interfacename ::= (#PCDATA)+
```

Description

`interfacename` elements mark the name of an interface for an object oriented programming language. The parent node is `oointerface`.

Attributes

`interfacename` elements have no attributes.

Formatting Expectations

Format in bold.

itemizedlist

Name

`itemizedlist` — list without a mandatory order

Content Model

```
itemizedlist ::= listitem+
```

Description

`itemizedlist` elements hold a series of items having no mandatory order. The parent node is `msgexplan`, `msgtext`, `refsect1`, or `refsect2`.

Attributes

`itemizedlist` elements have no attributes.

Formatting Expectations

Format as a block element with `listitem` elements bulleted.

listitem

Name

`listitem` — item in a simple list

Content Model

```
listitem ::= para, (itemizedlist | orderedlist | para | variablelist)*
```

Description

`listitem` elements hold an item in a simple list. The parent node is `itemizedlist`, `orderedlist`, or `varlistentry`.

Attributes

`listitem` elements have no attributes.

Formatting Expectations

Format as a block element.

literal

Name

`literal` — inline literal word or phrase

Content Model

```
literal ::= (#PCDATA)+
```

Description

`literal` elements hold a literal word or phrase. The parent node is `para`.

Attributes

`literal` elements have no attributes.

Formatting Expectations

Format in fixed width font.

manvolnum

Name

`manvolnum` — volume of RefEntry XML document

Content Model

```
manvolnum ::= (#PCDATA)+
```

Description

`manvolnum` elements hold section identifiers for `refentry` documents. The parent node is `refmeta`. `manvolnum` identifiers typically follow the **man** section conventions, and are generally short.

Attributes

`manvolnum` elements have no attributes.

Formatting Expectations

Format surrounded by parentheses, `(manvolnum)`.

methodname

Name

`methodname` — name of an object method

Content Model

```
methodname ::= (#PCDATA)+
```

Description

`methodname` elements hold the name of an object method in an object oriented programming language. The parent node is `constructorsynopsis`, `destructorsynopsis`, or `methodsynopsis`.

Attributes

`methodname` elements have no attributes.

Formatting Expectations

Format in bold.

methodparam

Name

`methodparam` — parameter for an object method

Content Model

```
methodparam ::= modifier*, type?, ((parameter, initializer?) |
funcparams), modifier*
```

Description

`methodparam` elements hold parameters for methods in an object oriented language. The parent node is `constructorsynopsis`, `destructorsynopsis`, or `methodsynopsis`.

Attributes

`methodparam` elements have `choice` and `rep` attributes.

`choice` attributes take one of the following values.

opt
: Optional parameter

plain
: Plain parameter

required
: Required parameter

`choice` values are implied if not given.

`rep` attributes take the following values.

norepeat
: Parameter may not be repeated

repeat
: Parameter may be repeated

`rep` values are implied if not given.

Formatting Expectations

When the `rep` element is `repeat`, follow with an ellipsis, (`methodparam ...`).

methodsynopsis

Name

`methodsynopsis` — synopsis for an object method

Content Model

```
methodsynopsis := modifier*, (type | void), methodname,
(methodparam+ | void), exceptionname*
```

Description

`methodsynopsis` elements hold synopses for object methods in an object oriented programming language. The parent node is `classsynopsis`, or `refsynopsisdiv`.

Attributes

`methodsynopsis` elements have no attributes.

Formatting Expectations

Format in fixed width font. Surround the parameter list with parentheses, separating parameters with commas, finishing with a semicolon.

modifier

Name

`modifier` — method or method parameter modifier

Content Model

```
modifier ::= (#PCDATA)+
```

Description

`modifier` elements hold the identifier for an object method or method parameter modifier in an object oriented programming language. The parent node is `constructorsynopsis`, `destructorsynopsis`, `fieldsynopsis`, `methodparam`, `methodsynopsis`, or `ooexception`.

Attributes

`modifier` elements have no attributes.

Formatting Expectations

Format as surrounding element.

msg

Name

`msg` — a single error message

Content Model

```
msg ::= msgmain, (msgrel | msgsub)*
```

Description

`msg` elements hold one error message in a set of error messages. The parent node is `msgentry`.

Attributes

`msg` elements have no attributes.

Formatting Expectations

Format as block element.

msgentry

Name

`msgentry` — entry in a set of error messages

Content Model

```
msgentry ::= msg+, msginfo, msgexplan
```

Description

`msgentry` elements hold an entry in a list of error messages. The parent node is `msgset`.

Attributes

`msgentry` elements have no attributes.

Formatting Expectations

Format as block element.

msgexplan

Name

`msgexplan` — explanation of an error message

Content Model

```
msgexplan ::= para, (blockquote | itemizedlist | orderedlist |
para | programlisting | screen | variablelist)*
```

Description

`msgexplan` elements hold the explanation for an error message. The parent node is `msgentry`.

Attributes

`msgexplan` elements have no attributes.

Formatting Expectations

Format as block element.

msginfo

Name

`msginfo` — information about an error message

Content Model

```
msginfo ::= msglevel | msgorig
```

Description

`msginfo` elements hold information about an error message. The parent node is `msgentry`.

Attributes

`msginfo` elements have no attributes.

Formatting Expectations

Format as block element.

msglevel

Name

`msglevel` — information about the level of an error message

Content Model

```
msglevel ::= (#PCDATA)+
```

Description

`msglevel` elements hold information about the level of an error message. The parent node is `msginfo`.

Attributes

`msglevel` elements have no attributes.

Formatting Expectations

Format as block element prefixed by the label *Level*.

msgmain

Name

`msgmain` — main component of an error message

Content Model

```
msgmain ::= msgtext
```

Description

`msgmain` elements hold the main component of an error message. The parent node is `msg`.

Attributes

`msgmain` elements have no attributes.

Formatting Expectations

Format as block element.

msgorig

Name

`msgorig` — information about the origin of an error message

Content Model

```
msgorig ::= (#PCDATA)+
```

Description

`msgorig` elements hold information about the origin of an error message. The parent node is `msginfo`.

Attributes

`msgorig` elements have no attributes.

Formatting Expectations

Format as block element prefixed by the label *Origin*.

msgrel

Name

`msgrel` — related component of an error message

Content Model

```
msgrel ::= msgtext
```

Description

`msgrel` elements hold a related component of an error message. The parent node is `msg`.

Attributes

`msgrel` elements have no attributes.

Formatting Expectations

Format as block element.

msgset

Name

`msgset` — list of error messages

Content Model

```
msgset ::= msgentry+
```

Description

`msgset` elements hold list of error messages. The parent node is `refsect1`.

Attributes

`msgset` elements have no attributes.

Formatting Expectations

Format as a list of related block elements.

msgsub

Name

`msgsub` — subcomponent of an error message

Content Model

```
msgsub ::= msgtext
```

Description

`msgsub` elements hold a subcomponent of an error message. The parent node is `msg`.

Attributes

`msgsub` elements have no attributes.

Formatting Expectations

Format as block element.

msgtext

Name

`msgtext` — text of an error message

Content Model

```
msgtext ::= para, (blockquote | itemizedlist | orderedlist |
para | programlisting | screen | variablelist)*
```

Description

`msgtext` elements hold the text of an error message. The parent node is `msgmain`, `msgrel`, or `msgsub`.

Attributes

`msgtext` elements have no attributes.

Formatting Expectations

Format as block element.

ooclass

Name

`ooclass` — object class

Content Model

```
ooclass ::= modifier*, classname
```

Description

`ooclass` elements hold an object class from an object oriented programming language. The parent node is `classsynopsis`.

Attributes

`ooclass` elements have no attributes.

Formatting Expectations

Format as surrounding element.

ooexception

Name

`ooexception` — object exception

Content Model

```
ooexception ::= modifier*, exceptionname
```

Description

`ooexception` elements hold an exception from an object oriented programming language. The parent node is `classsynopsis`.

Attributes

`ooexception` elements have no attributes.

Formatting Expectations

Format as surrounding element.

oointerface

Name

`oointerface` — object interface

Content Model

```
oointerface ::= modifier*, interfacename
```

Description

`oointerface` elements hold an interface from an object oriented programming language. The parent node is `classsynopsis`.

Attributes

`oointerface` elements have no attributes.

Formatting Expectations

Format as surrounding element.

orderedlist

Name

`orderedlist` — list having an explicit order

Content Model

```
orderedlist ::= listitem+
```

Description

`orderedlist` elements hold a series of items having an explicit order. The parent node is `msgexplan`, `msgtext`, `refsect1`, or `refsect2`.

Attributes

`orderedlist` elements have no attributes.

Formatting Expectations

Format as a block element with `listitem` elements numbered.

para

Name

`para` — text of an error message

Content Model

```
para ::= (#PCDATA | citerefentry | citetitle | classname |
command | emphasis | errorcode | errorname | errortype | function |
literal | quote | replaceable | trademark | ulink | xref)+
```

Description

`para` elements hold a paragraph. The parent node is `blockquote`, `entry`, `example`, `listitem`, `msgexplan`, `msgtext`, `refsect1`, or `refsect2`.

Attributes

`para` elements have no attributes.

Formatting Expectations

Format as block element.

para

paramdef

Name

`paramdef` — definition of a function parameter

Content Model

```
paramdef ::= (#PCDATA | funcparams | parameter)+
```

Description

`paramdef` elements hold a function parameter definition. The parent node is `funcprototype`.

Attributes

`paramdef` elements have no attributes.

Formatting Expectations

Format as surrounding element.

parameter

Name

`parameter` — function parameter name

Content Model

```
parameter ::= (#PCDATA)+
```

Description

`parameter` elements hold a function parameter name. The parent node is `method-param`, or `paramdef`.

Attributes

`parameter` elements have no attributes.

Formatting Expectations

Format in italics.

programlisting

Name

`programlisting` — code listing

Content Model

```
programlisting ::= (#PCDATA | classname | command | function |
replaceable)+
```

Description

`programlisting` elements hold a code listing. The parent node is `example`, `msgexplan`, `msgtext`, `refsect1` or `refsect2`.

Attributes

`programlisting` elements have no attributes.

Formatting Expectations

Format as preformatted text, respecting whitespace.

quote

Name

`quote` — inline quote

Content Model

```
quote ::= (#PCDATA)+
```

Description

`quote` elements hold an inline quote. The parent node is `para`.

Attributes

`quote` elements have no attributes.

Formatting Expectations

Format surrounded by double quotes (`"quote"`).

refentry

Name

`refentry` — root element for reference entry XML document

Content Model

```
refentry ::= refmeta, refnamediv, refsynopsisdiv, refsect1+
```

Description

`refentry` elements are root nodes of the document type of the same name. They do not have parent nodes when used with RefEntry servlets.

Attributes

`refentry` elements may take an `id` attribute. `id` attributes take name tokens, used for building links from other documents.

Formatting Expectations

Format as an individual page. If the `id` attribute is set, mark the location for cross referencing using the value of the attribute.

refentrytitle

Name

`refentrytitle` — title of a RefEntry XML document

Content Model

```
refentrytitle ::= (#PCDATA)+
```

Description

`refentrytitle` elements hold titles for `refentry` documents. The parent node is `refmeta`.

Attributes

`refentrytitle` elements have no attributes.

Formatting Expectations

Format in bold.

refmeta

Name

`refmeta` — container for meta-information about a reference entry XML document

Content Model

```
refmeta ::= refentrytitle, manvolnum, refmiscinfo*
```

Description

`refmeta` elements hold meta-information about `refentry` documents. The parent node is `refentry` itself.

Attributes

`refmeta` elements have no attributes.

Formatting Expectations

Format as block element with `refentrytitle`, followed by `manvolnum` in parentheses, at top of page. `refmiscinfo` elements are not rendered in the visible output.

refmiscinfo

Name

`refmiscinfo` — miscellaneous meta-information for a RefEntry document

Content Model

```
refmiscinfo ::= (#PCDATA)+
```

Description

`refmiscinfo` elements hold miscellaneous information about `refentry` documents. The parent node is `refmeta`.

Attributes

`refmiscinfo` elements have `class` attributes that take one of the following required values.

arch

: Machine architecture to which the `refentry` pertains

: Specify generic if the information applies irrespective of architecture.

copyright

: Copyright including at least the holder and date

: The element content may be the entire copyright text, if necessary.

date

: Date the `refentry` was last updated

sectdesc

: Short description of the reference section

software

: Software package to which this `refentry` belongs

Formatting Expectations

`refmiscinfo` elements are not rendered in the visible output.

refname

Name

`refname` — name of the reference subject

Content Model

```
refname ::= (#PCDATA)+
```

Description

`refname` elements hold the name of the subject of the `refentry`. The parent node is `refnamediv`.

Attributes

`refname` elements have no attributes.

Formatting Expectations

Format in fixed width font.

refnamediv

Name

`refnamediv` — name and description of reference subject

Content Model

```
refnamediv ::= refname+, refpurpose
```

Description

`refnamediv` elements hold the names of the subjects of the `refentry` and a quick description for the entry. The parent node is `refentry`.

Attributes

`refnamediv` elements have no attributes.

Formatting Expectations

Format as a block element having title `NAME` followed by a paragraph with the `refname` followed by a dash, followed by the `refpurpose`.

refpurpose

Name

`refpurpose` — short description of the reference subject

Content Model

```
refpurpose ::= (#PCDATA)+
```

Description

`refpurpose` elements hold a short (one line) description of the subject of the `refentry`. The parent node is `refnamediv`.

Attributes

`refpurpose` elements have no attributes.

Formatting Expectations

Format as surrounding element.

refsect1

Name

`refsect1` — generic reference entry section

Content Model

```
refsect1 ::= title, para, (blockquote | example | itemizedlist |
msgset | orderedlist | para | programlisting | screen | table |
variablelist)*, refsect2*
```

Description

`refsect1` elements hold generic `refentry` sections. The parent node is `refentry`.

Attributes

`refsect1` elements may take an `id` attribute. `id` attributes take name tokens, used for building links for internal cross references.

Formatting Expectations

Format as a block element with the title followed by the rest of the content. If the `id` attribute is set, mark the location for cross referencing using the value of the attribute.

refsect2

Name

`refsect2` — generic reference entry subsection

Content Model

```
refsect2 ::= title, para, (blockquote | itemizedlist | orderedlist |
para | programlisting | screen | variablelist)*
```

Description

`refsect2` elements hold generic `refentry` subsections. The parent node is `refsect1`.

Attributes

`refsect2` elements may take an `id` attribute. `id` attributes take name tokens, used for building links for internal cross references.

Formatting Expectations

Format as a block element with the title followed by the rest of the content. If the `id` attribute is set, mark the location for cross referencing using the value of the attribute.

refsynopsisdiv

Name

`refsynopsisdiv` — reference entry section for a synopsis

Content Model

```
refsynopsisdiv ::= title, (classsynopsis | cmdsynopsis |
constructorsynopsis | destructorsynopsis | fieldsynopsis |
funcsynopsis | methodsynopsis | synopsis)+
```

Description

`refsynopsisdiv` elements hold `refentry` sections for synopses. The parent node is `refentry`.

Attributes

`refsynopsisdiv` elements may take an `id` attribute. `id` attributes take name tokens, used for building links for internal cross references.

Formatting Expectations

Format as a block element with the title followed by the rest of the content. If the `id` attribute is set, mark the location for cross-referencing using the value of the attribute.

replaceable

Name

`replaceable` — item to be replaced with a literal string

Content Model

```
replaceable ::= (#PCDATA)+
```

Description

`replaceable` elements hold items intended to be replaced with a literal string such as *`filename`* to be replaced with the actual name of a file. The parent node is `arg`, `computeroutput`, `funcsynopsisinfo`, `para`, `programlisting`, `screen`, `synopsis`, or `term`.

Attributes

`replaceable` elements have no attributes.

Formatting Expectations

Format as italic.

row

Name

`row` — row in a table

Content Model

```
row ::= entry+
```

Description

`row` elements hold a row in a table. The parent node is `tbody`.

Attributes

`row` elements have `valign` attributes. `valign` attributes can take the following values.

bottom
: Align with the bottom edge of the row

middle
: Align with the middle of the row

top
: Align with the top edge of the row

Formatting Expectations

Format as a single table row, aligned as specified by the value of the `valign` attribute.

sbr

Name

`sbr` — line break in command synopsis

Content Model

```
sbr ::= EMPTY
```

Description

`sbr` elements indicate a forced line break in a command synopsis. The parent node is `cmdsynopsis`, or `group`.

Attributes

`sbr` elements have no attributes.

Formatting Expectations

Format as a line break.

screen

Name

`screen` — terminal as seen by user

Content Model

```
screen ::= (#PCDATA | computeroutput | replaceable | userinput)+
```

Description

`screen` elements hold a terminal screen as seen by user. The parent node is `example`, `msgexplan`, `msgtext`, `refsect1`, or `refsect2`.

Attributes

`screen` elements have no attributes.

Formatting Expectations

Format as preformatted text, respecting whitespace, surrounded by a thin box.

synopsis

Name

`synopsis` — generic synopsis

Content Model

```
synopsis ::= (#PCDATA | replaceable)+
```

Description

`synopsis` elements hold a synopsis for which there is no specific synopsis type in DocBook. The parent node is `refsynopsisdiv`.

Attributes

`synopsis` elements have no attributes.

Formatting Expectations

Format in fixed width font respecting use of whitespace.

table

Name

`table` — information presented in tabular form

Content Model

```
table ::= (title, tgroup)
```

Description

`table` elements hold information in tabular form. The parent node is `refsect1`.

Attributes

`table` elements have `colsep`, `frame`, and `rowsep` attributes.

`colsep` attributes, if present, take the value `0`, or `1`.

`frame` attributes take the following values.

all
: Borders on all cells

bottom
: Border at the bottom of the table

none
: No border

sides
: Border on outside edges of the table

top
: Border at the top of the table

topbot
: Border at the top and the bottom of the table

`frame` values are implied if not given.

`rowsep` attributes, if present, take the value `0`, or `1`.

Formatting Expectations

Format as a table where `colsep` and `rowsep` elements determine whether columns and rows are separated by borders. The `frame` element takes precedence if border style is specified. Default border style is as for `frame` with value `topbot`.

tbody

Name

`tbody` — body of a table

Content Model

```
tbody ::= row+
```

Description

`tbody` elements hold the body of a table. The parent node is `tgroup`.

Attributes

`tbody` elements have `valign` attributes. `valign` attributes can take the following values.

bottom
: Align with the bottom edge of the table body

middle
: Align with the middle of the table body

top
: Align with the top edge of the table body

Formatting Expectations

Format according to the value of the `valign` attribute. Cells in rows are formatted normally.

term

Name

`term` — element of a list of terms and definitions

Content Model

```
term ::= (#PCDATA | replaceable)+
```

Description

`term` elements hold a term in a list of terms and their associated definitions. The parent node is `variablelist`.

Attributes

`term` elements have no attributes.

Formatting Expectations

Format as a definition term in a definition list element.

tgroup

Name

`tgroup` — group of table block elements

Content Model

```
tgroup ::= thead?, tbody
```

Description

`tgroup` elements hold a group of table block elements. The parent node is `tgroup`.

Attributes

`tgroup` elements have no attributes.

Formatting Expectations

Format as surrounding element.

thead

Name

`thead` — header of a table

Content Model

```
thead ::= row+
```

Description

`thead` elements hold the header of a table. The parent node is `tgroup`.

Attributes

`thead` elements have `valign` attributes. `valign` attributes can take the following values.

bottom
: Align with the bottom edge of the table header

middle
: Align with the middle of the table header

top
: Align with the top edge of the table header

Formatting Expectations

Format according to the value of the `valign` attribute. Cells in rows are formatted in bold.

title

Name

`title` — section, example, or table title

Content Model

```
title ::= (#PCDATA)+
```

Description

`title` elements hold titles for sections, examples, and tables in a `refentry` document. The parent node is `example`, `refsect1`, `refsect2`, `refsynopsisdiv`, or `table`.

Attributes

`title` elements have no attributes.

Formatting Expectations

Format as block element in header format.

trademark

Name

`trademark` — trademarked term

Content Model

```
trademark ::= (#PCDATA)+
```

Description

`trademark` elements hold a trademarked term. The parent node is `para`.

Attributes

`trademark` elements have `class` attributes. `class` attributes take one of the following values.

copyright
: Copyright (C)

registered
: Registered trademark (R)

service
: Service mark (SM)

trade
: Trademark (TM)

`class` values are implied if not given.

Formatting Expectations

Format followed by appropriate trademark character, depending on the value of class. Default is `trademark`.

type

Name

`type` — object data type

Content Model

```
type ::= (#PCDATA)+
```

Description

`type` elements hold the type of a data element in an object oriented language. The parent node is `fieldsynopsis`, `methodparam`, or `methodsynopsis`.

Attributes

`type` elements have no attributes.

Formatting Expectations

Format as surrounding element.

ulink

Name

`ulink` — URL link

Content Model

```
ulink ::= (#PCDATA)+
```

Description

`ulink` elements hold URL links to other resources. The parent node is `para`.

Attributes

`ulink` elements take a `url` attribute. `url` attribute values must be absolute, not relative, URLs.

Formatting Expectations

Format as surrounding element and a hyperlink to the value of the `url` attribute.

userinput

Name

`userinput` — text as typed by a user

Content Model

```
userinput ::= (#PCDATA)+
```

Description

`userinput` elements hold text as typed by a user. The parent node is `screen`.

Attributes

`userinput` elements have no attributes.

Formatting Expectations

Format in bold.

varargs

Name

`varargs` — repeated variable arguments in a function synopsis

Content Model

```
varargs ::= EMPTY
```

Description

`varargs` elements indicate that a variable argument in a function synopsis repeats. The parent node is `funcprototype`.

Attributes

`varargs` elements have no attributes.

Formatting Expectations

Format as an ellipsis (. . .).

variablelist

Name

`variablelist` — list of terms and associated explanations

Content Model

```
variablelist ::= varlistentry+
```

Description

`variablelist` elements hold a list of terms and their associated explanations. The parent node is `listitem`, `msgexplan`, `msgtext`, `refsect1`, or `refsect2`.

Attributes

`variablelist` elements have no attributes.

Formatting Expectations

Format as a block element with `variablelistentry` elements set as definitions having terms and explanations.

varlistentry

Name

`varlistentry` — item in list of terms and associated explanations

Content Model

```
varlistentry ::= term+, listitem
```

Description

`varlistentry` elements hold an item in list of terms and their associated explanations. The parent node is `variablelist`.

Attributes

`varlistentry` elements have no attributes.

Formatting Expectations

Format as a definition explanation in a definition list element.

varname

Name

`varname` — object variable name

Content Model

```
varname ::= (#PCDATA)+
```

Description

`varname` elements hold the name of a variable in an object oriented programming language. The parent node is `fieldsynopsis`.

Attributes

`varname` elements have no attributes.

Formatting Expectations

Format in bold.

void

Name

`void` — void type in a synopsis

Content Model

```
void ::= EMPTY
```

Description

`void` elements indicate a void type in a synopsis. The parent node is `constructorsynopsis`, `destructorsynopsis`, `funcsynopsis`, or `methodsynopsis`.

Attributes

`void` elements have no attributes.

Formatting Expectations

Format as `void`.

xref

Name

`xref` — cross-reference link

Content Model

```
xref ::= EMPTY
```

Description

`xref` elements hold pointers to cross-reference targets. The parent node is `para`.

Attributes

`xref` elements take a `linkend` attribute. `linkend` attribute values must correspond to the `id` value of the cross-reference target.

Formatting Expectations

Format as a hyperlink to the value of the `linkend` attribute, containing the value of the title of the cross-reference target.

Appendix B. Code Listings

This appendix lists code written for the project.

Ant Build File

```
<?xml version="1.0"?>
<!-- Ant build.xml file for CS445 project                        -->
<!--                                                             -->
<!-- Copyright 2003, Mark Craig                                  -->
<!--                                                             -->
<project name="Basic RefEntry Servlet" default="war" basedir=".">
<!-- Generic project properties                                  -->
  <property name="project.fullname" value="Basic RefEntry Servlet"/>
  <property name="project.version"  value="1.0"/>
  <property name="project.name"     value="refentry"/>

<!-- Source locations for the build                              -->
  <property name="bin.dir" value="bin"/>
  <property name="src.dir" value="src"/>

<!-- Destination locations for the build                         -->
  <property name="out.dir"    value="."/>
  <property name="build.dir"  value="${out.dir}/build"/>
  <property name="class.dir"  value="${build.dir}/classes"/>
  <property name="copy.dir"   value="${build.dir}/copy"/>
  <property name="doc.dir"    value="${build.dir}/docs"/>
  <property name="serial.dir" value="${build.dir}/serial"/>
  <property name="dist.dir"   value="${out.dir}/dist"/>

<!-- Initialization target                                       -->
  <target name="init">
    <tstamp/>
    <mkdir dir="${build.dir}"/>
    <mkdir dir="${build.dir}/build"/> <!-- For build.xml file, etc.-->
    <mkdir dir="${class.dir}"/>
    <mkdir dir="${copy.dir}"/>
    <mkdir dir="${doc.dir}"/>
    <mkdir dir="${doc.dir}/api"/>
    <mkdir dir="${serial.dir}"/>
    <mkdir dir="${dist.dir}"/>
  </target>

<!-- Compile the Java code from ${src.dir} into ${class.dir}     -->
  <target name="compile" depends="init">
    <javac srcdir="${src.dir}" destdir="${class.dir}"/>
  </target>

<!-- Compile the Java format wrapper tool                        -->
  <target name="tool" depends="init,compile">
    <javac srcdir="${out.dir}" includes="format.java">
      <classpath>
        <pathelement path="${class.dir}"/>
      </classpath>
    </javac>
  </target>

<!-- Compile the ServletUnit based tests for the finished servlet  -->
  <target name="servletunit" depends="init">
    <javac srcdir="${out.dir}" includes="Test*.java">
      <classpath>
        <pathelement path="${class.dir}"/>
      </classpath>
    </javac>
  </target>

<!-- Copy the javadoc, sources, and format wrapper               -->
  <target name="copy" depends="init,servletunit,tool">
    <copy file="${out.dir}/docs/install.txt"  todir="${doc.dir}"/>
    <copy file="${out.dir}/docs/README.txt"   todir="${doc.dir}"/>
    <copy file="${out.dir}/docs/relnotes.txt" todir="${doc.dir}"/>
    <copy file="${out.dir}/docs/project.sgml" todir="${doc.dir}"/>
    <copy file="${out.dir}/build.xml"         todir="${build.dir}/build"/>
    <copy file="${out.dir}/format.class"      todir="${build.dir}/build"/>
    <copy file="${out.dir}/format.java"       todir="${build.dir}/build"/>
```

```
    <copy file="${out.dir}/TestClient.class"  todir="${build.dir}/build"/>
    <copy file="${out.dir}/TestClient.java"   todir="${build.dir}/build"/>
    <copy file="${out.dir}/TestGarbageRequests.class"
      todir="${build.dir}/build"/>
    <copy file="${out.dir}/TestGarbageRequests.java"
      todir="${build.dir}/build"/>
    <copy file="${out.dir}/TestOkayRequests.class"
      todir="${build.dir}/build"/>
    <copy file="${out.dir}/TestOkayRequests.java"
      todir="${build.dir}/build"/>
    <copy file="${out.dir}/TestThreads.class" todir="${build.dir}/build"/>
    <copy file="${out.dir}/TestThreads.java"  todir="${build.dir}/build"/>
    <copy todir="${copy.dir}">
      <fileset dir="${src.dir}"/>
    </copy>
  </target>

<!-- Serialize the back end and common words                        -->
  <property name="src"        value="${src.dir}/conf/refentrys.props"/>
  <property name="xsl"        value="${src.dir}/conf/xslt.props"/>
  <property name="cw"         value="${src.dir}/conf/cw.props"/>
  <property name="scw"        value="${serial.dir}/commonWords"/>
  <property name="sbe"        value="${serial.dir}/backend"/>
  <property name="class.base" value="org.mcraig.cs445.refentry"/>
  <target name="serialize" depends="compile">
    <java classname="${class.base}.Build" fork="yes" failonerror="true">
      <arg
        line="-src ${src} -xsl ${xsl} -cw ${cw} -cwOut ${scw} -beOut ${sbe}"
      />
      <classpath>
        <pathelement path="${class.dir}"/>
        <pathelement path="${java.class.path}"/>
      </classpath>
    </java>
  </target>

<!-- Generate the javadoc                                           -->
  <target name="javadoc" depends="init">
    <javadoc
      access="package"
      author="true"
      destdir="${doc.dir}/api"
      sourcepath="${src.dir}"
      use="true"
      windowtitle="Basic RefEntry Servlet API">
      <packageset dir="${src.dir}">
        <include name="org/mcraig/cs445/refentry/**"/>
      </packageset>
      <doctitle><![CDATA[<h1>Basic RefEntry Servlet API</h1>]]></doctitle>
      <bottom><![CDATA[<i>Copyright © 2002-2003 Mark Craig</i>]]></bottom>
    </javadoc>
  </target>

<!-- Run the acceptance tests                                       -->
  <target name="accept" depends="compile">
    <java classname="${class.base}.AcceptanceTest"
      failonerror="true" fork="yes">
      <classpath>
        <pathelement location="${class.dir}"/>
        <pathelement path="${java.class.path}"/>
      </classpath>
    </java>
  </target>

<!-- Generate the war file                                          -->
  <target name="war" depends="accept,copy,javadoc,serialize">
    <war
      destfile="${dist.dir}/${project.name}.war"
      webxml="${src.dir}/web.xml">
      <webinf     dir="${serial.dir}"/>
      <classes    dir="${class.dir}"/>
      <zipfileset dir="${doc.dir}" prefix="docs"/>
      <zipfileset dir="${copy.dir}" prefix="src"/>
      <zipfileset dir="${build.dir}/build"/>
    </war>
  </target>

<!-- Clean up the build directory                                   -->
  <target name="clean">
    <delete dir="${build.dir}"/>
```

```
    <delete><fileset dir="${out.dir}" includes="*.class"/></delete>
  </target>

<!-- Clean everything                                                -->
  <target name="cleandist" depends="clean">
    <delete dir="${dist.dir}"/>
  </target>

</project>
```

Format Tool

```
package org.mcraig.cs445.refentry;
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.util.HashSet;
import java.util.Properties;
import org.xml.sax.InputSource;

/*
 * Utility for formatting RefEntry documents.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
public class Format {
    /** Filename of XSLT Properties File. */
    private final static String XSLTProps = "src/conf/xslt.props";

    /** USAGE message. */
    private final static String usage =
    "Usage:\n" +
    "\tjava format [-infile <inputFile>][-outfile <outputFile>]\n" +
    "\tWhere <inputFile> is an XML RefEntry and <outputFile> is\n" +
    "\tan XHTML representation of the RefEntry.\n" +
    "\tIf <inputFile> is not provided, standard input is used.\n" +
    "\tIf <outputFile> is not provided, standard output is used.\n";
    /** Print a usage message to System.err and exit. */
    private static void usage() {
        System.err.println(usage);
        System.exit(1);
    }

    /** Filename of XML RefEntry source. */
    private static String infile;
    /** Filename of XHTML output. */
    private static String outfile;

    /**
     * Format a single RefEntry document.
     * @param args Expects arguments as shown:<p>
     * <tt>java format
     * [-infile <i>inputFile</i>][-outfile <i>outputFile</i>]</tt>
     * <p>If the <i>inputFile</i> is not provided,
     * input is taken from System.in.
     * <p>If the <i>outputFile</i> is not provided,
     * input is sent to System.out.
     */
    public static void main(String[] args) {
        handleInput(args);

        Properties xslt = new Properties();
        try {
            xslt.load(new FileInputStream(XSLTProps));

            File inf;
            if (infile.equals("")) {
                inf = File.createTempFile("format", "");
```

```
            BufferedReader br =
                new BufferedReader(new InputStreamReader(System.in));
            PrintStream ps =
                new PrintStream(
                    new BufferedOutputStream(
                        new FileOutputStream(inf)));
            String str;
            while ((str = br.readLine()) != null) ps.println(str);
            br.close();
            ps.close();
        } else {
            inf = new File(infile);
        }

        InputSource input = new InputSource(new FileReader(inf));
        if (!RefEntryValidator.isValid(input)) {
            System.err.println("Input file is not valid.");
            if (infile.equals("")) inf.delete();
            System.exit(1);
        }

        File outf = File.createTempFile("format", "");
        RefEntry re = new RefEntry(inf, outf, xslt);
        RefEntryTransformer tr =
            new RefEntryTransformer("CLI format");
        String outstr = new String(tr.transform(re, ""));

        if (outfile.equals("")) {
            System.out.print(outstr);
        } else {
            PrintWriter pw =
                new PrintWriter(
                    new BufferedWriter(
                        new FileWriter(outfile)));
            pw.write(outstr);
            pw.close();
        }

        if (infile.equals("")) inf.delete();
        outf.delete();
    } catch (Exception e) {
        System.err.println(e.toString());
        System.exit(1);
    }
}

/** Handle command line input. */
private static void handleInput(String[] args) {
    if (args.length == 0) {
        infile  = "";
        outfile = "";
    } else if (args.length == 2) {
        if (args[0].equals("-infile")) {
            infile  = args[1];
            outfile = "";
        } else if (args[0].equals("-outfile")) {
            infile  = "";
            outfile = args[1];
        } else {
            usage();
        }
    } else if (args.length == 4) {
        if (args[0].equals("-infile") && args[2].equals("-outfile")) {
            infile  = args[1];
            outfile = args[3];
        } else {
            usage();
        }
    } else {
        usage();
    }

    if (infile.length() != 0) {
        File test = new File(infile);
        if (!test.canRead()) {
            System.err.println("ERROR: Cannot read " + infile);
            usage();
        }
        if (!test.isFile()) {
            System.err.println("ERROR: " + infile + " is not a file.");
```

```
                usage();
            }
        }

        if (outfile.length() != 0) {
            File test = new File(outfile);
            if (test.isDirectory() && test.canWrite()) {
                outfile += "OUTFILE.HTM";
            } else if (test.isFile()) {
                System.err.println("ERROR: " + outfile + " already exists.");
                usage();
            }
        }
    }
}
```

Format Wrapper

```
import org.mcraig.cs445.refentry.Format;

/**
 * Format an individual refentry document.
 */
public class format {
    /**
     * Call the RefEntry formatter.
     */
    public static void main(String[] args) {
        Format.main(args);
    }
}
```

Common Words Configuration File

```
# Words to ignore when indexing and searching
# From the American Heritage Word Frequency Book, ISBN-0-395-13570-2.
CommonWords = the and you that for was are with his they this \
from have one had not but what all were when there can your which
```

RefEntry Documents Configuration File

```
# Individual RefEntry Documents
build-1=src/xml/build.1
format-1=src/xml/format.1
webman-1=src/xml/webman.1
BackEndGenerator-3=src/xml/BackEndGenerator.3
Build-3=src/xml/Build.3
CommonWordsGenerator-3=src/xml/CommonWordsGenerator.3
Format-3=src/xml/Format.3
RefEntry-3=src/xml/RefEntry.3
RefEntryBackEnd-3=src/xml/RefEntryBackEnd.3
RefEntryErrorHandler-3=src/xml/RefEntryErrorHandler.3
RefEntryGenerator-3=src/xml/RefEntryGenerator.3
RefEntryIndex-3=src/xml/RefEntryIndex.3
RefEntryServlet-3=src/xml/RefEntryServlet.3
RefEntryTransformer-3=src/xml/RefEntryTransformer.3
RefEntryValidator-3=src/xml/RefEntryValidator.3
Serializer-3=src/xml/Serializer.3
cw.props-5=src/xml/cw.props.5
refentrys.props-5=src/xml/refentrys.props.5
xslt.props-5=src/xml/xslt.props.5
arg-7=src/xml/arg.7
attribution-7=src/xml/attribution.7
blockquote-7=src/xml/blockquote.7
citerefentry-7=src/xml/citerefentry.7
citetitle-7=src/xml/citetitle.7
```

```
classname-7=src/xml/classname.7
classsynopsis-7=src/xml/classsynopsis.7
classsynopsisinfo-7=src/xml/classsynopsisinfo.7
cmdsynopsis-7=src/xml/cmdsynopsis.7
command-7=src/xml/command.7
computeroutput-7=src/xml/computeroutput.7
constructorsynopsis-7=src/xml/constructorsynopsis.7
destructorsynopsis-7=src/xml/destructorsynopsis.7
emphasis-7=src/xml/emphasis.7
entry-7=src/xml/entry.7
errorcode-7=src/xml/errorcode.7
errorname-7=src/xml/errorname.7
errortype-7=src/xml/errortype.7
example-7=src/xml/example.7
exceptionname-7=src/xml/exceptionname.7
fieldsynopsis-7=src/xml/fieldsynopsis.7
funcdef-7=src/xml/funcdef.7
funcparams-7=src/xml/funcparams.7
funcprototype-7=src/xml/funcprototype.7
funcsynopsis-7=src/xml/funcsynopsis.7
funcsynopsisinfo-7=src/xml/funcsynopsisinfo.7
function-7=src/xml/function.7
group-7=src/xml/group.7
initializer-7=src/xml/initializer.7
interfacename-7=src/xml/interfacename.7
itemizedlist-7=src/xml/itemizedlist.7
listitem-7=src/xml/listitem.7
literal-7=src/xml/literal.7
manvolnum-7=src/xml/manvolnum.7
methodname-7=src/xml/methodname.7
methodparam-7=src/xml/methodparam.7
methodsynopsis-7=src/xml/methodsynopsis.7
modifier-7=src/xml/modifier.7
msg-7=src/xml/msg.7
msgentry-7=src/xml/msgentry.7
msgexplan-7=src/xml/msgexplan.7
msginfo-7=src/xml/msginfo.7
msglevel-7=src/xml/msglevel.7
msgmain-7=src/xml/msgmain.7
msgorig-7=src/xml/msgorig.7
msgrel-7=src/xml/msgrel.7
msgset-7=src/xml/msgset.7
msgsub-7=src/xml/msgsub.7
msgtext-7=src/xml/msgtext.7
ooclass-7=src/xml/ooclass.7
ooexception-7=src/xml/ooexception.7
oointerface-7=src/xml/oointerface.7
orderedlist-7=src/xml/orderedlist.7
para-7=src/xml/para.7
paramdef-7=src/xml/paramdef.7
parameter-7=src/xml/parameter.7
programlisting-7=src/xml/programlisting.7
quote-7=src/xml/quote.7
refentry-7=src/xml/refentry.7
refentrytitle-7=src/xml/refentrytitle.7
refmeta-7=src/xml/refmeta.7
refmiscinfo-7=src/xml/refmiscinfo.7
refname-7=src/xml/refname.7
refnamediv-7=src/xml/refnamediv.7
refpurpose-7=src/xml/refpurpose.7
refsect1-7=src/xml/refsect1.7
refsect2-7=src/xml/refsect2.7
refsynopsisdiv-7=src/xml/refsynopsisdiv.7
replaceable-7=src/xml/replaceable.7
row-7=src/xml/row.7
sbr-7=src/xml/sbr.7
screen-7=src/xml/screen.7
synopsis-7=src/xml/synopsis.7
table-7=src/xml/table.7
tbody-7=src/xml/tbody.7
term-7=src/xml/term.7
tgroup-7=src/xml/tgroup.7
thead-7=src/xml/thead.7
title-7=src/xml/title.7
trademark-7=src/xml/trademark.7
type-7=src/xml/type.7
ulink-7=src/xml/ulink.7
userinput-7=src/xml/userinput.7
varargs-7=src/xml/varargs.7
variablelist-7=src/xml/variablelist.7
```

```
varlistentry-7=src/xml/varlistentry.7
varname-7=src/xml/varname.7
void-7=src/xml/void.7
xref-7=src/xml/xref.7
test-test=src/test/bigtest.xml
```

XSLT Stylesheets Configuration File

```
# Stylesheets used for transformations. You may modify RefEntry
# formatting by changing RefEntryXSLT. Changing other stylesheets may
# break the application.
ManVolNumXSLT  = src/xslt/manvolnum.xslt
RefEntryXSLT   = src/xslt/refentry.xslt
RefNameXSLT    = src/xslt/refname.xslt
RefPurposeXSLT = src/xslt/refpurpose.xslt
StripXMLXSLT   = src/xslt/stripxml.xslt
```

BackEndGenerator Class

```
package org.mcraig.cs445.refentry;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Properties;

/**
 * Generates and serializes a RefEntryBackEnd referencing a number
 * of RefEntry documents.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
class BackEndGenerator {
    /** Default constructor. */
    public BackEndGenerator() { CommonWords = new HashSet(); }
    /**
     * Generate a serialized RefEntryBackEnd.
     * @param sources HashSet of RefEntry documents to add.
     * @param serialized Path to serialized output BackEnd.
     */
    public void generate(HashSet sources, String serialized) {
        RefEntryBackEnd backend = new RefEntryBackEnd(CommonWords);

        Iterator iter = sources.iterator();
        while (iter.hasNext()) backend.add((RefEntry)iter.next());

        try {
            ObjectOutputStream os =
                new ObjectOutputStream(
                    new BufferedOutputStream(
                        new FileOutputStream(serialized)));
            os.writeObject(backend);
            os.close();
        } catch (Exception e) {
            System.err.println("In BackEndGenerator::generate...");
            System.err.println("Failed to serialze BackEnd: " + serialized);
            System.err.println(e.toString());
        }
    }
    /**
     * Mutator for set of common words to ignore.
     * @param words Path to serialized HashSet of common words.
     */
    public void setCommonWords(String words) {
        try {
            ObjectInputStream os =
                new ObjectInputStream(
                    new BufferedInputStream(
```

```
                        new FileInputStream(words)));
            CommonWords = (HashSet)os.readObject();
            os.close();
        } catch (Exception e) {
            System.err.println("In BackEndGenerator::setCommonWords...");
            System.err.println("Failed to open common words: " + words);
            System.err.println(e.toString());
        }
    }
    /** Set of CommonWords to ignore. */
    private HashSet CommonWords;
}
```

Build Class

```
package org.mcraig.cs445.refentry;

/**
 * Generate RefEntry servlet serialized object files.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
class Build {
    /** Filename of the sources Properties file. */
    private static String SrcProps;
    /** Filename of XSLT Properties file. */
    private static String XSLTProps;
    /** Filename of the common words Properties file. */
    private static String CWProps;
    /** Filename of the serialized common words object file. */
    private static String CWFile;
    /** Filename of the serialized RefEntryBackEnd object file. */
    private static String BEFile;

    /** USAGE message. */
    private final static String usage =
    "Usage:\n" +
    "\tbuild -src <PropertiesFile> -xsl <PropertiesFile>\n" +
    "\t      -cw <PropertiesFile> -cwOut <File> -beOut <File>\n" +
    "\tWhere <PropertiesFile> options are source (src), XSLT\n" +
    "\tstylesheets (xsl), and common words (cw) Java properties\n" +
    "\tfiles; the cwOut <File> is the name of the serialized\n" +
    "\tobject file containing common words; the beOut <File>\n" +
    "\tis the name of the serialized RefEntryBackEnd object file.";

    /** Print a usage message to System.err and exit. */
    private static void usage() {
        System.err.println(usage);
        System.exit(1);
    }

    /**
      * Build a RefEntry servlet.
      * @param args Arguments expected as per the Usage message.
      */
    public static void main(String[] args) {
        handleInput(args);

        Serializer s = new Serializer();
        s.build(SrcProps, XSLTProps, CWProps, CWFile, BEFile);
    }

    /** Handle command line input. */
    static void handleInput(String[] args) {
        if (args.length != 10) usage();
        if (args[0].equals("-src"))   SrcProps = args[1];
        else {
            System.err.println(args[0] + " not recognized.");
            usage();
        }
        if (args[2].equals("-xsl"))   XSLTProps = args[3];
        else {
            System.err.println(args[2] + " not recognized.");
            usage();
        }
        if (args[4].equals("-cw"))    CWProps = args[5];
```

```
        else {
            System.err.println(args[4] + " not recognized.");
            usage();
        }
        if (args[6].equals("-cwOut")) CWFile = args[7];
        else {
            System.err.println(args[6] + " not recognized.");
            usage();
        }
        if (args[8].equals("-beOut")) BEFile = args[9];
        else {
            System.err.println(args[8] + " not recognized.");
            usage();
        }
    }
}
```

CommonWordsGenerator Class

```
package org.mcraig.cs445.refentry;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Properties;
import java.util.StringTokenizer;

/**
 * Serializes a HashSet of common words to ignore when indexing and
 * searching.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
class CommonWordsGenerator {
    /** Default constructor. */
    public CommonWordsGenerator() { ; }
    /**
     * Generate a serialized HashSet of common words.
     * @param properties Path to properties file containing
     * <tt>CommonWords</tt> property having as its value the list
     * of common words to ignore.
     * @param serialized Path to serialized output HashSet.
     */
    public void generate(String properties, String serialized) {
        Properties props = new Properties();
        try {
            FileInputStream fip = new FileInputStream(properties);
            props.load(new BufferedInputStream(fip));
            fip.close();
        } catch (Exception e) {
            System.err.println("In CommonWordsGenerator::generate...");
            System.err.println("Failed to load properties: " + properties);
            System.err.println(e.toString());
        }

        HashSet         words = new HashSet();
        StringTokenizer in    = new StringTokenizer(
            props.getProperty("CommonWords"));
        String tok = new String();
        while (in.hasMoreTokens()) {
            tok = in.nextToken();
            if (tok.length() >= 3) words.add(tok.toLowerCase());
        }

        try {
            ObjectOutputStream os =
                new ObjectOutputStream(
                    new BufferedOutputStream(
                        new FileOutputStream(serialized)));
            os.writeObject(words);
            os.close();
        } catch (Exception e) {
            System.err.println("In CommonWordsGenerator::generate...");
```

```
            System.err.println("Failed to serialze HashSet: " + serialized);
            System.err.println(e.toString());
        }
    }
}
```

RefEntryBackEnd Class

```
package org.mcraig.cs445.refentry;
import java.io.Serializable;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;

/**
 * Backend keeping track of RefEntry documents for the servlet.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
class RefEntryBackEnd implements Serializable {
    /**
     * Default constructor.
     * @param cw Common words to ignore when indexing.
     */
    RefEntryBackEnd(HashSet cw) {
        Index       = new RefEntryIndex();
        Entries     = new HashMap();
        CommonWords = cw;
    }
    /**
     * Add a unique, formatted RefEntry document to those tracked.
     * @param page RefEntry document to add.
     */
    boolean add(RefEntry page) {
        // Add the page to the set of all known documents.
        if (Entries.containsKey(page.getID())) return false;
        Entries.put(page.getID(), page);

        // Add the page to the index.
        HashSet  keys  = page.getKeys(CommonWords);
        Iterator iter  = keys.iterator();
        while (iter.hasNext()) {
            Index.add((String)iter.next(), page);
        }
        return true;
    }
    /**
     * Return a set of RefEntry documents.
     * @param keys Strings to match in RefEntry documents.
     */
    HashSet getValues(HashSet keys) {
        HashSet values = new HashSet();
        Iterator iter  = keys.iterator();
        while (iter.hasNext()) {
            values.addAll(Index.match((String)iter.next()));
        }
        return values;
    }
    /**
     * Return a single RefEntry as a set.
     * @param id RefEntry identifier.
     */
    HashSet getEntry(String id) {
        HashSet set = new HashSet();
        if (Entries.containsKey(id) && Entries.get(id) != null)
            set.add(Entries.get(id));
        return set;
    }
    /** Set common words to ignore when indexing. */
    private HashSet CommonWords;
    /** Index holding tokens corresponding to RefEntry documents. */
    private RefEntryIndex Index;
    /** Set of all known RefEntry documents. */
    private HashMap Entries;
}
```

RefEntryErrorHandler Class

```
package org.mcraig.cs445.refentry;
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

/**
 * Error handler for SAX parse errors, useful during validation.
 * <p>The methods here do throw exceptions when triggered, so
 * if you do not catch them in the parser, they may stop your
 * progress, even for non-fatal errors and warnings.
 * They are intended to provide useful messages to human beings
 * checking their RefEntry documents for validity.
 * @author <a href=mailto:mark@mcraig.org>Mark Craig</a>
 */
public class RefEntryErrorHandler implements ErrorHandler {
    /**
     * Log a diagnostic message for a non-fatal error from the SAX parser.
     * @param exception The exception from the parser.
     * @throws SAXException every time this method is called.
     */
    public void error(SAXParseException exception)
        throws SAXException {
        log("==>Parse error<==", exception);
        throw new SAXException("Parse error encountered");
    }
    /**
     * Log a diagnostic message for a fatal error from the SAX parser.
     * @param exception The exception from the parser.
     * @throws SAXException every time this method is called.
     */
    public void fatalError(SAXParseException exception)
        throws SAXException {
        log("==>Fatal parse error<==", exception);
        throw new SAXException("Fatal parse error encountered");
    }
    /**
     * Log a diagnostic message for a warning from the SAX parser.
     * @param exception The exception from the parser.
     * @throws SAXException every time this method is called.
     */
    public void warning(SAXParseException exception)
        throws SAXException {
        log("==>Parse warning<==", exception);
        throw new SAXException("Parse warning encountered");
    }
    /**
     * Utility to format a diagnostic message on System.out.
     * @param message Header to identify the message.
     * @param spe Exception containing the diagnostic information.
     */
    private void log(String message, SAXParseException spe) {
        System.out.println(
            message + "\n" +
            "  Line:    " + spe.getLineNumber() + "\n" +
            "  URI:     " + spe.getSystemId() + "\n" +
            "  Message: " + spe.getMessage()
        );
    }
}
```

RefEntryGenerator Class

```
package org.mcraig.cs445.refentry;
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.util.Enumeration;
import java.util.HashSet;
import java.util.Properties;

/**
 * Generates a HashSet of RefEntry objects.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
```

```
 */
class RefEntryGenerator {
    /** Default constructor. */
    public RefEntryGenerator() {
        Stylesheets = new Properties();
        Sources     = new Properties();
    }
    /** XSLT stylesheets. */
    private Properties Stylesheets;
    /** RefEntry source files. */
    private Properties Sources;
    /**
     * Mutator for set of XSLT stylesheets.
     * @param stylesheets Path to Properties file indicating XSLT files.
     */
    public void setStyleSheets(String stylesheets) {
        try {
            FileInputStream fip = new FileInputStream(stylesheets);
            Stylesheets.load(new BufferedInputStream(fip));
            fip.close();
        } catch (Exception e) {
            System.err.println("In RefEntryGenerator::setStyleSheets...");
            System.err.println("Failed to load properties: " + stylesheets);
            System.err.println(e.toString());
        }
    }
    /**
     * Mutator for set of RefEntry sources.
     * @param sources Path to Properties file indicating RefEntry sources.
     */
    public void setSources(String sources) {
        try {
            FileInputStream fip = new FileInputStream(sources);
            Sources.load(new BufferedInputStream(fip));
            fip.close();
        } catch (Exception e) {
            System.err.println("In RefEntryGenerator::setSources...");
            System.err.println("Failed to load properties: " + sources);
            System.err.println(e.toString());
        }
    }
    /** Generate a HashSet of RefEntry documents. */
    public HashSet generate() {
        HashSet refentrys = new HashSet();
        try {
            Enumeration e = Sources.propertyNames();
            String      element;
            while (e.hasMoreElements()) {
                element  = (String)e.nextElement();
                File src = new File(Sources.getProperty(element));
                File htm = File.createTempFile(element, ".html");

                RefEntry re = new RefEntry(src, htm, Stylesheets);

                refentrys.add(re);
            }
        } catch (Exception e) {
            System.err.println("In RefEntryGenerator::generate...");
            System.err.println("Failed to : ");
            System.err.println(e.toString());
        }
        return refentrys;
    }
}
```

RefEntryIndex Class

```
package org.mcraig.cs445.refentry;
import java.io.Serializable;
import java.util.HashMap;
import java.util.HashSet;

/**
 * Hash of RefEntry documents grouped according to search strings.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
class RefEntryIndex implements Serializable {
    /** Default constructor. */
    RefEntryIndex() {
        Index = new HashMap();
    }
    /**
     * Add an existing RefEntry document for a particular
     * string. The string is folded to lower case.
     * @param key String to match later.
     * @param value The RefEntry to add.
     */
    boolean add(String key, RefEntry value) {
        if (key == null || value == null) return false;

        key = key.toLowerCase();
        HashSet values = new HashSet();
        if (!Index.isEmpty() && Index.get(key) != null)
            values.addAll((HashSet)Index.get(key));
        if (!values.add(value)) return false;
        Index.put(key, values);
        return true;
    }
    /**
     * Get the set of RefEntry documents matching a string.
     * The string is folded to lower case.
     * @param key String to match.
     */
    HashSet match(String key) {
        if (key == null) return (new HashSet());
        key = key.toLowerCase();
 if (Index.get(key) == null) return (new HashSet());
        HashSet matches = new HashSet((HashSet)Index.get(key));
        return matches;
    }
    /** Structure containing the indexed RefEntry documents. */
    private HashMap Index;
}
```

RefEntry Class

```
package org.mcraig.cs445.refentry;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.Serializable;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Properties;
import java.util.StringTokenizer;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;

/**
 * A single RefEntry document.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
class RefEntry implements Comparable, Serializable {

    /** XML source of the document. */
    private File InFile;
    /** XHTML div of the document. */
    private File OutFile;
```

```
/** XSLT stylesheets. */
private Properties StyleSheets;

/** Default constructor.
  * @param infile XML source.
  * @param outifile XHTML format output.
  * @param xslt Set of XSLT stylesheets.
  */
RefEntry(File infile, File outfile, Properties xslt) {
    if (infile.canRead()) InFile = infile;
    OutFile = new File(outfile.getPath());
    if (!xslt.isEmpty()) StyleSheets = xslt;

    setRefName();
    setManVolNum();
    setRefPurpose();
    setKeys();
    format();

    setSource();
    setXHTML();
}

/**
 * Apply a stylesheet to the source.
 * @param xml Source to transform.
 * @param xslt Stylesheet to apply.
 * @param out Output file.
 */
private File applyTransform(File xml, File xslt, File out) {
    try {
        BufferedReader rin  = new BufferedReader(new FileReader(xml));
        BufferedReader rxsl = new BufferedReader(new FileReader(xslt));
        BufferedWriter wout = new BufferedWriter(new FileWriter(out));

        StreamSource src = new StreamSource(rin);
        StreamSource xsl = new StreamSource(rxsl);
        StreamResult res = new StreamResult(wout);

        TransformerFactory tf = TransformerFactory.newInstance();
        Transformer        tr = tf.newTransformer(xsl);
        tr.transform(src, res);

        rin.close(); rxsl.close(); wout.close();
    } catch (Exception e) {
        System.err.println(e.toString());
    }
    return out;
}

/**
 * Comparison between RefEntries.
 * @param other Object to compare this one to.
 */
public int compareTo(Object other) {
    if (this == other) return 0;
    int bySect = ManVolNum.compareTo(((RefEntry)other).getManVolNum());
    if (bySect != 0) return bySect;
    int byName = RefName.compareTo(((RefEntry)other).getRefName());
    if (byName != 0) return byName;
    return RefPurpose.compareTo(((RefEntry)other).getRefPurpose());
}

/** Format the document as an XHTML div. */
private void format() {
    try {
        File xslt = new File(StyleSheets.getProperty("RefEntryXSLT"));
        OutFile   = applyTransform(InFile, xslt, OutFile);
    } catch (Exception e) {
        System.err.println(e.toString());
    }
}

/** Return an ID for the page. */
String getID() {
    return (new String(getRefName() + "-" + getManVolNum()));
}

/**
 * Get strings from the document content to use
```

```
 * when indexing. No strings shorter than three
 * characters are returned.
 * @param ignore Strings not to return in the results.
 */
HashSet getKeys(HashSet ignore) {
    HashSet keys = new HashSet(Keys);
    keys.removeAll(ignore);
    return keys;
}
/** Strings to use when indexing and searching. */
HashSet Keys;
/** Build a set of Strings to use when indexing and searching. */
private void setKeys() {
    Keys       = new HashSet();
    String str = new String();
    try {
        File xsl = new File(StyleSheets.getProperty("StripXMLXSLT"));
        File out = File.createTempFile("keys-", ".out");
        out      = applyTransform(InFile, xsl, out);

        BufferedReader br = new BufferedReader(new FileReader(out));
        String         tmp;
        while ((tmp = br.readLine()) != null) str = str.concat(tmp + " ");

        br.close();
        out.delete();
    } catch (Exception e) {
        System.err.println(e.toString());
    }

    String TokenDelimiters =
        new String(" \t\n\f\r!\"#$%&'()*+,./:;<=>?@[\\]^_`{|}~");
    StringTokenizer in = new StringTokenizer(str, TokenDelimiters);
    while (in.hasMoreTokens()) {
        str = in.nextToken();
        if (str.matches("\\w+") && (str.length() >= 3))
            Keys.add(str.toLowerCase());
    }
}

/** Return XML document as a String. */
String getSource() { return Source; }
/** XML document content as a String. */
private String Source;
/** Build a String holding the XML document. */
private void setSource() { Source = fileToString(InFile); }

/** Return XHTML as a string. */
String getXHTML() { return XHTML; }
/** Formatted document content as a String. */
private String XHTML;
/** Build a String holding the formatted document content. */
private void setXHTML() { XHTML = fileToString(OutFile); }

/**
 * Helper method to manage translations.
 * @param file File to convert to a string.
 */
private String fileToString(File file) {
    String str = new String();
    try {
        BufferedReader br = new BufferedReader(new FileReader(file));
        String tmp;
        while ((tmp = br.readLine()) != null)
            str = str.concat(tmp + "\n");
        br.close();
    } catch (Exception e) {
        System.err.println(e.toString());
    }
    return str;
}

/** Content of the manvolnum element. */
private String ManVolNum;
/** Return the content of the manvolnum element. */
String getManVolNum() { return ManVolNum; }
/** Set content of manvolnum element. */
private void setManVolNum() {
    try {
        File xslt = new File(StyleSheets.getProperty("ManVolNumXSLT"));
```

```
            File out  = File.createTempFile("mvn-", ".out");
            BufferedReader br =
                new BufferedReader(
                    new FileReader(applyTransform(InFile, xslt, out)));
            ManVolNum = br.readLine().trim();
            br.close();
            out.delete();
        } catch (Exception e) {
            System.err.println(e.toString());
        }
    }

    /** Content of the refname element. */
    private String RefName;
    /** Return the content of the refname element. */
    String getRefName() { return RefName; }
    /** Set content of refname element. */
    private void setRefName() {
        try {
            File xslt = new File(StyleSheets.getProperty("RefNameXSLT"));
            File out  = File.createTempFile("rn-", ".out");
            BufferedReader br =
                new BufferedReader(
                    new FileReader(applyTransform(InFile, xslt, out)));
            RefName   = br.readLine().trim();
            br.close();
            out.delete();
        } catch (Exception e) {
            System.err.println(e.toString());
        }
    }

    /** Content of the refpurpose element. */
    private String RefPurpose;
    /** Return the content of the refpurpose element. */
    String getRefPurpose () { return RefPurpose;}
    /** Set content of refpurpose element. */
    private void setRefPurpose() {
        try {
            File xslt  = new File(StyleSheets.getProperty("RefPurposeXSLT"));
            File out   = File.createTempFile("rp-", ".out");
            BufferedReader br =
                new BufferedReader(
                    new FileReader(applyTransform(InFile, xslt, out)));
            RefPurpose = br.readLine().trim();
            br.close();
            out.delete();
        } catch (Exception e) {
            System.err.println(e.toString());
        }
    }
}
```

RefEntryServlet Class

```
package org.mcraig.cs445.refentry;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.ObjectInputStream;
import java.net.URL;
import java.net.URLDecoder;
import java.util.HashSet;
import java.util.StringTokenizer;
import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.SingleThreadModel;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Basic RefEntry Server application.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
```

```
public class RefEntryServlet extends HttpServlet {
    /** Default constructor. */
    public RefEntryServlet() { return; }

    /** Log that this method has been called. */
    public void destroy() {
        System.out.println("[RefEntry] destroy");
    }

    /** Get basic information. */
    public String getServletInfo() {
        return "Basic RefEntry Servlet";
    }
    /**
     * Get the backend and tools for transformations.
     * @param config Servlet configuration information.
     * @throws ServletException when configuration information is null.
     */
    public void init(ServletConfig config) throws ServletException {
        try {
            System.out.println("[RefEntry] init starting.");

            // Get the context for real paths
            ServletContext ctx  = config.getServletContext();

            // Read in the RefEntryBackEnd object and the serialized
            // HashSet of common words
            String sbe = config.getInitParameter("serializedBackend");
            if (sbe != null) sbe = "/WEB-INF/" + sbe;
            else throw new ServletException("sbe is null.");
            ObjectInputStream inbe =
                new ObjectInputStream(ctx.getResourceAsStream(sbe));
            BackEnd    = (RefEntryBackEnd)inbe.readObject();
            inbe.close();
            System.out.println("[RefEntry] got backend.");

            String scw  = config.getInitParameter("serializedCommonWords");
            if (scw != null) scw = "/WEB-INF/" + scw;
            else throw new ServletException("scw is null.");
            ObjectInputStream incw =
                new ObjectInputStream(ctx.getResourceAsStream(scw));
            CommonWords = (HashSet)incw.readObject();
            incw.close();
            System.out.println("[RefEntry] got common words.");

            // Get the default page to return when no search keywords
            // are provided and no page is requested.
            DefaultPage = config.getInitParameter("defaultPage");
            if (DefaultPage == null)
                throw new ServletException("DefaultPage is null");
            System.out.println("[RefEntry] got default page.");

            System.out.println("[RefEntry] init succeeded.");
        } catch (Exception e) {
            throw new ServletException(e.toString());
        }
    }
    /**
     * Form an XHTML response given a search request.
     * @param req HTTP request from the servlet container.
     * @param res HTTP response for the client application.
     * @throws ServletException if the servlet cannot respond to the request.
     */
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        // Based on the URL query field, set the RefEntry ID to return
        // or the content of the search request.
        String query = null;
        if (req.getQueryString() != null)
            query = new String(req.getQueryString());

        String  request = null;
        String  entryid = null;
        HashSet search  = new HashSet();
        if (query != null) {
            entryid = req.getParameter("RefEntry");
            request = req.getParameter("SrchString");

            if (request != null) {
                try {
```

```
                    request = URLDecoder.decode(request, "UTF-8");
                } catch (Exception e) {
                    throw new ServletException(e.toString());
                }
                search = new HashSet();
                search = getTokens(request);
            }
        }

        // Construct the response to the client
        res.setContentType("text/html");

        String base = new String((req.getRequestURL()).toString());
        RefEntryTransformer tr = new RefEntryTransformer(base);

        try {
            PrintWriter pw = res.getWriter();
            if (DefaultPage == null) {
                throw new ServletException("[RefEntry] DefaultPage is null.");
            } else if (entryid == null && request == null) {
                // Nothing to look for? Return the default page.
                pw.print(tr.transform(BackEnd.getEntry(DefaultPage), ""));
            } else if (entryid != null) {
                // Link to a RefEntry? Return the document.
                pw.print(tr.transform(BackEnd.getEntry(entryid), ""));
            } else {
                // Search string? Return the results.
                pw.print(tr.transform(BackEnd.getValues(search), request));
            }
            pw.close();
        } catch (Exception e) {
            ServletException se = new ServletException(e.toString());
            res.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
            throw se;
        }
    }
    /** Call doGet if some client tries doPut. */
    public void doPut(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        doGet(req, res);
    }
    /**
     * Tokenize the search request, returning only useful words.
     * @param request String to tokenize.
     */
    HashSet getTokens(String request) {
        StringTokenizer in     = new StringTokenizer(request);
        String          tok    = new String();
        HashSet         tokens = new HashSet();
        while(in.hasMoreTokens()) {
            tok = in.nextToken();
            if (tok.matches("\\w+") &&
                (tok.length() >= 3)) {
                tokens.add(tok);
            }
        }
        if (CommonWords != null) tokens.removeAll(CommonWords);
        return tokens;
    }
    /** Backend used by this servlet. */
    private RefEntryBackEnd BackEnd;
    /** Common words to ignore when searching. */
    private HashSet CommonWords;
    /** ID of default page for this servlet. */
    private String DefaultPage;
}
```

RefEntryTransformer Class

```
package org.mcraig.cs445.refentry;
import java.io.Serializable;
import java.util.Arrays;
import java.util.Date;
import java.util.HashSet;
import java.util.Iterator;

/**
 * Transforms sets of RefEntry documents from XML to XHTML for
 * consumption by the client browser.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
class RefEntryTransformer implements Serializable {
    /**
     * Default constructor.
     * @param path Servlet URL.
     */
    RefEntryTransformer(String path) {
        Server = path;
    }
    /** Servlet URL as a string. */
    private String Server;
    /**
     * Transform a single RefEntry resulting from a search.
     * @param source RefEntry to transform.
     * @param search Literal search string from client request.
     */
    String transform(RefEntry source, String search) {
        return (
                getHead(
                        source.getRefName() +
                        "(" +
                        source.getManVolNum() +
                        ") " +
                        "- " +
                        source.getRefPurpose()
                        ) +
                getSearchBar(search) +
                source.getXHTML() +
                getTail()
                );
    }
    /**
     * Transform a set of RefEntry results from a search.
     * @param refentries Set of documents to transform.
     * @param search Literal search string from client request.
     */
    String transform(HashSet refentries, String search) {
        // Dispatch to other methods unless search has
        // returned more than one entry.
        if (refentries.isEmpty()) return transform(search);
        else if (refentries.size() == 1) {
            Iterator iter  = refentries.iterator();
            RefEntry entry = (RefEntry)iter.next();
            return transform(entry, search);
        }

        return (
                getHead("Search Results") +
                getSearchBar(search) +
                getList(refentries) +
                getTail()
                );
    }
    /** Serve up XHTML for a search that returned nothing.
     * @param search Literal search string from client request.
     */
    String transform(String search) {
        return (
                getHead("Search Results") +
                getSearchBar(search) +
                "<h1 align=\"center\">SEARCH RESULTS</h1>\n" +
                "<p>No results were found at " +
                Server +
                " \n" +
                "   for search string <tt>" +
                search +
```

```
            "</tt>.</p>\n" +
            getTail()
            );
}
/** Generate the head of the page.
 * @param title Title element content for the page.
 */
private String getHead(String title) {
    return (
            "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n" +
            "<head>\n" +
            "<title>" +
            title +
            "</title>\n" +
            "</head>\n" +
            "<body bgcolor=\"white\">\n"
            );
}
/** Generate the search bar form.
 * @param search Literal search string from client request.
 */
private String getSearchBar(String search) {
    return (
            "<form>\n" +
            "<input name=\"SrchString\" size=\"60\" value=\"" +
            search +
            "\" />\n" +
            "<input type=\"submit\" value=\"Search\" />\n" +
            "</form>\n" +
            "<hr />\n"
            );
}
/** Generate a list of search results.
 * @param refentries Set of documents to transform.
 */
private String getList(HashSet refentries) {
    RefEntry[] refs = new RefEntry[refentries.size()];
    Iterator   iter = refentries.iterator();
    for (int i = 0; iter.hasNext(); ++i) {
        refs[i] = (RefEntry)iter.next();
    }
    Arrays.sort(refs);

    String list = new String("<ol>");
    for (int i = 0; i < refentries.size(); ++i) {
        list = list.concat(
                    "<li><a href=\"" +
                    Server +
                    "?RefEntry=" +
                    refs[i].getRefName() +
                    "-" +
                    refs[i].getManVolNum() +
                    "\">" +
                    refs[i].getRefName() +
                    "(" +
                    refs[i].getManVolNum() +
                    ")</a>" +
                    "-" +
                    refs[i].getRefPurpose() +
                    "</li>\n"
                    );
    }
    list = list.concat("</ol>\n");
    return list;
}
/** Generate the tail of the page. */
private String getTail() {
    Date date = new Date();
    return (
            "<table border=\"0\" width=\"100%\">\n" +
            "<tr>\n" +
            "<td align=\"left\">Generated [" +
            date.toString() +
            "]</td>\n" +
            "<td align=\"right\">Server [" +
            Server +
            "]</td>\n" +
            "</tr>\n" +
            "</table>\n" +
            "</body>\n" +
```

```
                "</html>"
                );
    }
}
```

RefEntryValidator Class

```
package org.mcraig.cs445.refentry;
import java.io.FileReader;
import org.xml.sax.InputSource;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.helpers.XMLReaderFactory;
/**
 * Wrapper for a SAX parser that checks XML document validity.
 * @author <a href=mailto:mark@mcraig.org>Mark Craig</a>
 */
public class RefEntryValidator {
    /**
     * Command-line interface for the validator.<p>
     * Usage: java RefEntryValidator <filename>
     */
    public static void main(String[] args) {
        if (args.length != 1) {
            System.err.println("Usage: java RefEntryValidator <filename>");
            System.exit(1);
        }
        try {
            InputSource src = new InputSource(new FileReader(args[0]));
            if (isValid(src)) System.out.println("Valid.");
            else System.out.println("Not valid.");
        } catch (Exception e) {
            System.err.println(e.toString());
        }
    }
    /**
     * Returns true if the input is a valid XML document.
     * @param src The XML document to validate.
     */
    public static boolean isValid(InputSource src) {
        boolean srcIsValid = false;
        try {
            String saxp = new String("org.apache.xerces.parsers.SAXParser");
            XMLReader reader = XMLReaderFactory.createXMLReader(saxp);
            reader.setFeature("http://xml.org/sax/features/validation", true);
            reader.setContentHandler(new DefaultHandler());
            reader.setErrorHandler(new RefEntryErrorHandler());
            reader.parse(src);
            srcIsValid = true;
        } catch (Exception e) {
            System.out.println(e.toString());
        } finally {
            return srcIsValid;
        }
    }
}
```

Serializer Class

```
package org.mcraig.cs445.refentry;
import java.util.HashSet;

/**
 * Build serialized objects for RefEntry servlet.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
class Serializer {
    /** Default constructor. */
    Serializer() { ; }
    /**
     * Construct RefEntryBackEnd and CommonWords serialized objects.
```

```
     * @param srcProps Properties file specifying RefEntry source documents.
     * @param xsltProps Properties file specifying XSLT stylesheets.
     * @param cwProps Properties file specifying common words to ignore.
     * @param cwFile Path to serialized common words object file.
     * @param beFile Path to serialized RefEntryBackEnd object file.
     */
    public void build(String srcProps, String xsltProps,
                      String cwProps, String cwFile, String beFile) {
        RefEntryGenerator reg = new RefEntryGenerator();
        reg.setSources(srcProps);
        reg.setStyleSheets(xsltProps);
        HashSet entries = reg.generate();

        CommonWordsGenerator cwg = new CommonWordsGenerator();
        cwg.generate(cwProps, cwFile);

        BackEndGenerator beg = new BackEndGenerator();
        beg.setCommonWords(cwFile);
        beg.generate(entries, beFile);
    }
}
```

RefEntry DTD

```
<!-- RefEntry DTD                                                    -->
<!-- Subset of DocBook RefEntry for WebMan project                   -->
<!--                                                                 -->
<!-- Copyright 2002 Mark Craig                                       -->
<!--                                                                 -->
<!--                                                                 -->
<!-- A command argument for use in command synopses                  -->
<!ELEMENT arg (#PCDATA | replaceable)*                               >
<!ATTLIST arg
          choice   (opt|plain|required)    #IMPLIED
          rep      (norepeat|repeat)       #IMPLIED                  >
<!--                                                                -->
<!-- Whom to attribute a block quote                                -->
<!ELEMENT attribution (#PCDATA)*                                     >
<!--                                                                -->
<!-- An extended quote, formatted as a block element                -->
<!ELEMENT blockquote (attribution, para+)                            >
<!--                                                                -->
<!-- A reference to another RefEntry document                       -->
<!ELEMENT citerefentry (refentrytitle, manvolnum)                    >
<!--                                                                -->
<!-- A reference to another document                                -->
<!ELEMENT citetitle (#PCDATA)*                                       >
<!--                                                                -->
<!-- An object class name                                           -->
<!ELEMENT classname (#PCDATA)*                                       >
<!--                                                                -->
<!-- A class synopsis for the synopsis section, block element       -->
<!ELEMENT classsynopsis ((ooclass | oointerface | ooexception)+,
                         (classsynopsisinfo | constructorsynopsis |
                          destructorsynopsis | fieldsynopsis |
                          methodsynopsis)*)                          >
<!--                                                                -->
<!-- Information about an object class                              -->
<!ELEMENT classsynopsisinfo (#PCDATA)*                               >
<!--                                                                -->
<!-- An command synopsis for the synopsis section, block element    -->
<!ELEMENT cmdsynopsis (arg | command | group | sbr)+                 >
<!--                                                                -->
<!-- An command either as part of a synopsis or in another section  -->
<!ELEMENT command (#PCDATA)*                                         >
<!--                                                                -->
<!-- Output displayed on a computer terminal screen                 -->
<!ELEMENT computeroutput (#PCDATA | replaceable)*                    >
<!--                                                                -->
<!-- Synopsis for an object constructor                             -->
<!ELEMENT constructorsynopsis (modifier*, methodname?,
                               (methodparam+ | void), exceptionname*)  >
<!--                                                                -->
<!-- Synopsis for an object destructor                              -->
<!ELEMENT destructorsynopsis (modifier*, methodname?,
```

```
                          (methodparam+ | void), exceptionname*)   >
<!--                                                                 -->
<!-- An inline, emphasized word or phrase                           -->
<!ELEMENT emphasis (#PCDATA)*                                        >
<!--                                                                 -->
<!-- A table cell                                                   -->
<!ELEMENT entry (para)+                                              >
<!ATTLIST entry
         align    (center | left | right) #IMPLIED
         morerows NMTOKEN                 #IMPLIED
         valign   (bottom | middle | top) #IMPLIED                    >
<!--                                                                 -->
<!-- An error code especially as part of a list of errors           -->
<!ELEMENT errorcode (#PCDATA)*                                       >
<!--                                                                 -->
<!-- An error name especially as part of a list of errors           -->
<!ELEMENT errorname (#PCDATA)*                                       >
<!--                                                                 -->
<!-- An error type especially as part of a list of errors           -->
<!ELEMENT errortype (#PCDATA)*                                       >
<!--                                                                 -->
<!-- An example use of the function or command, with title          -->
<!ELEMENT example (title, (para | programlisting | screen)+)         >
<!ATTLIST example
         id       ID                      #IMPLIED                   >
<!--                                                                 -->
<!-- Name of an OO exception                                        -->
<!ELEMENT exceptionname (#PCDATA)*                                   >
<!--                                                                 -->
<!-- Synopsis for a class field                                     -->
<!ELEMENT fieldsynopsis (modifier*, type?, varname, initializer?)    >
<!--                                                                 -->
<!-- An function definition for a function synopsis                 -->
<!ELEMENT funcdef (#PCDATA | function)*                              >
<!--                                                                 -->
<!-- An function referenced through a function pointer              -->
<!ELEMENT funcparams (#PCDATA)*                                      >
<!--                                                                 -->
<!-- An function prototype for a function synopsis                  -->
<!ELEMENT funcprototype (funcdef, (varargs | void | paramdef+))      >
<!--                                                                 -->
<!-- An function synopsis for the synposis section                  -->
<!ELEMENT funcsynopsis (funcsynopsisinfo?, funcprototype+)           >
<!--                                                                 -->
<!-- Extra function information, such as a header file line         -->
<!ELEMENT funcsynopsisinfo (#PCDATA | replaceable)*                  >
<!--                                                                 -->
<!-- A function either as part of a synopsis or in another section  -->
<!ELEMENT function (#PCDATA)*                                        >
<!--                                                                 -->
<!-- A group of command arguments for a command synopsis            -->
<!ELEMENT group (arg | sbr)+                                         >
<!ATTLIST group
         choice   (opt | plain | required) #IMPLIED
         rep      (norepeat | repeat)      #IMPLIED                  >
<!--                                                                 -->
<!-- Initializer for a method parameter                             -->
<!ELEMENT initializer (#PCDATA)*                                     >
<!--                                                                 -->
<!-- Name of an OO interface                                        -->
<!ELEMENT interfacename (#PCDATA)*                                   >
<!--                                                                 -->
<!-- An unordered list of items                                     -->
<!ELEMENT itemizedlist (listitem)+                                   >
<!--                                                                 -->
<!-- An single item in a list; may include multiple block elements  -->
<!ELEMENT listitem (para, (itemizedlist | orderedlist | para |
                           variablelist)*)                           >
<!--                                                                 -->
<!-- An inline, literal word or phrase                              -->
<!ELEMENT literal (#PCDATA)*                                         >
<!--                                                                 -->
<!-- The RefEntry volume number used when citing a RefEntry         -->
<!ELEMENT manvolnum (#PCDATA)*                                       >
<!--                                                                 -->
<!-- Name of an object method                                       -->
<!ELEMENT methodname (#PCDATA)*                                      >
<!--                                                                 -->
<!-- Parameter of an object method                                  -->
<!ELEMENT methodparam (modifier*, type?, ((parameter, initializer?) |
```

```
                      funcparams), modifier*)                          >
<!ATTLIST methodparam
         choice   (opt | plain | required) #IMPLIED
         rep      (norepeat | repeat)      #IMPLIED                    >
<!--                                                                   -->
<!-- Synopsis for an object method                                     -->
<!ELEMENT methodsynopsis (modifier*, (type | void), methodname,
                          (methodparam+ | void), exceptionname*)       >
<!--                                                                   -->
<!-- Modifier for an object method or method parameter                 -->
<!ELEMENT modifier (#PCDATA)*                                          >
<!--                                                                   -->
<!-- A single entry in a set of error messages                         -->
<!ELEMENT msgentry (msg+, msginfo, msgexplan)                          >
<!--                                                                   -->
<!-- The explanation of an error message                               -->
<!ELEMENT msgexplan (para, (blockquote | itemizedlist | orderedlist |
                            para | programlisting | screen |
                            variablelist)*)                            >
<!--                                                                   -->
<!-- Information about the level and origin of an error message       -->
<!ELEMENT msginfo (msglevel | msgorig)                                 >
<!--                                                                   -->
<!-- Information about the level of an error message                   -->
<!ELEMENT msglevel (#PCDATA)*                                          >
<!--                                                                   -->
<!-- An error message in a set of error messages                       -->
<!ELEMENT msg (msgmain, (msgrel | msgsub)*)                            >
<!--                                                                   -->
<!-- Main component of an error message                                -->
<!ELEMENT msgmain (msgtext)                                            >
<!--                                                                   -->
<!-- Information about the origin of an error message                  -->
<!ELEMENT msgorig (#PCDATA)*                                           >
<!--                                                                   -->
<!-- Related error messages in a set of error messages                 -->
<!ELEMENT msgrel (msgtext)                                             >
<!--                                                                   -->
<!-- A set of error messages                                           -->
<!ELEMENT msgset (msgentry)+                                           >
<!--                                                                   -->
<!-- A subordiate message of a top level error message                 -->
<!ELEMENT msgsub (msgtext)                                             >
<!--                                                                   -->
<!-- Error message text itself, for use in a set of error messages     -->
<!ELEMENT msgtext (para, (blockquote | itemizedlist | orderedlist |
                          para | programlisting | screen |
                          variablelist)+)                              >
<!--                                                                   -->
<!-- Identifier for an OO class                                        -->
<!ELEMENT ooclass (modifier*, classname)                               >
<!--                                                                   -->
<!-- Prototype for an OO exception                                     -->
<!ELEMENT ooexception (modifier*, exceptionname)                       >
<!--                                                                   -->
<!-- Prototype for an OO interface                                     -->
<!ELEMENT oointerface (modifier*, interfacename)                       >
<!--                                                                   -->
<!-- A list of items having a particular order                         -->
<!ELEMENT orderedlist (listitem)+                                      >
<!--                                                                   -->
<!-- A parameter definition for a function synopsis                    -->
<!ELEMENT paramdef (#PCDATA | funcparams | parameter)*                 >
<!--                                                                   -->
<!-- A parameter in a function synopsis or another section             -->
<!ELEMENT parameter (#PCDATA)*                                         >
<!--                                                                   -->
<!-- A paragraph                                                       -->
<!ELEMENT para  (#PCDATA | citerefentry | citetitle | classname |
                 command | emphasis | errorcode | errorname |
                 errortype | function | literal | quote |
                 replaceable | trademark | ulink | xref)*              >
<!--                                                                   -->
<!-- A program listing, may need to be preprocessed for inclusion      -->
<!ELEMENT programlisting (#PCDATA | classname | command | function |
                          replaceable)*                                >
<!--                                                                   -->
<!-- An inline quote                                                   -->
<!ELEMENT quote (#PCDATA)*                                             >
<!--                                                                   -->
```

```
<!-- A RefEntry document type (the root level element for this DTD)  -->
<!ELEMENT refentry (refmeta, refnamediv, refsynopsisdiv, refsect1+)    >
<!ATTLIST refentry
          id       ID                    #IMPLIED                    >
<!--                                                                  -->
<!-- The title of a RefEntry document, usually the ref. subject      -->
<!ELEMENT refentrytitle (#PCDATA)*                                     >
<!--                                                                  -->
<!-- A section of meta information about the RefEntry document       -->
<!ELEMENT refmeta (refentrytitle, manvolnum, refmiscinfo*)             >
<!--                                                                  -->
<!-- An item of meta information about the RefEntry document         -->
<!ELEMENT refmiscinfo (#PCDATA)*                                       >
<!ATTLIST refmiscinfo
          class    (arch | copyright |
                    date | sectdesc |
                    software)           #REQUIRED                    >
<!--                                                                  -->
<!-- A section concerning the ref. subject and quick description     -->
<!ELEMENT refnamediv (refname+, refpurpose)                            >
<!--                                                                  -->
<!-- A ref. subject                                                  -->
<!ELEMENT refname (#PCDATA)*                                           >
<!--                                                                  -->
<!-- A quick description of the ref. subject(s)                      -->
<!ELEMENT refpurpose (#PCDATA)*                                        >
<!--                                                                  -->
<!-- A generic RefEntry section                                      -->
<!ELEMENT refsect1 (title, para, (blockquote | example | itemizedlist |
                                  msgset | orderedlist | para |
                                  programlisting | screen | table |
                                  variablelist)*, refsect2*)           >
<!ATTLIST refsect1
          id       ID                    #IMPLIED                    >
<!--                                                                  -->
<!-- A generic RefEntry subsection                                   -->
<!ELEMENT refsect2 (title, para, (blockquote | itemizedlist |
                                  orderedlist | para |
                                  programlisting | screen |
                                  variablelist)*)                      >
<!ATTLIST refsect2
          id       ID                    #IMPLIED                    >
<!--                                                                  -->
<!-- A RefEntry synopsis section                                     -->
<!ELEMENT refsynopsisdiv (title, (classsynopsis | cmdsynopsis |
                                  constructorsynopsis | destructorsynopsis |
                                  fieldsynopsis | funcsynopsis |
                                  methodsynopsis | synopsis)+)         >
<!ATTLIST refsynopsisdiv
          id       ID                    #IMPLIED                    >
<!--                                                                  -->
<!-- Inline text that the user must replace with a literal value     -->
<!ELEMENT replaceable (#PCDATA)*                                       >
<!--                                                                  -->
<!-- A row in a table                                                -->
<!ELEMENT row (entry)+                                                 >
<!ATTLIST row
          valign   (bottom | middle | top) #IMPLIED                    >
<!--                                                                  -->
<!-- An indicator for a forced line break in cmdsynopsis formatting  -->
<!ELEMENT sbr EMPTY                                                    >
<!--                                                                  -->
<!-- A (terminal) screen as seen by a user                           -->
<!ELEMENT screen (#PCDATA | computeroutput | replaceable | userinput)* >
<!--                                                                  -->
<!-- A generic synopsis (not for a command or function)              -->
<!ELEMENT synopsis (#PCDATA | replaceable)*                            >
<!--                                                                  -->
<!-- A table having a title and formatted as a table                 -->
<!ELEMENT table (title, tgroup)                                        >
<!ATTLIST table
          colsep   NMTOKEN               #IMPLIED
          frame    (all | bottom | none |
                    sides | top | topbot)  #IMPLIED
          rowsep   NMTOKEN               #IMPLIED                    >
<!--                                                                  -->
<!-- A table body element, as distinguished from a table header      -->
<!ELEMENT tbody (row)+                                                 >
<!ATTLIST tbody
          valign   (bottom | middle | top) #IMPLIED                    >
```

```
<!--                                                                  -->
<!-- A term as the first part of a variable list entry                -->
<!ELEMENT term (#PCDATA | replaceable)*                               >
<!--                                                                  -->
<!-- A group of table elements                                        -->
<!ELEMENT tgroup (thead?, tbody)                                      >
<!--                                                                  -->
<!-- A table header element, as distinguished from a table body       -->
<!ELEMENT thead (row+)                                                >
<!ATTLIST thead
          valign   (bottom | middle | top) #IMPLIED                   >
<!--                                                                  -->
<!-- A section title                                                  -->
<!ELEMENT title (#PCDATA)*                                            >
<!--                                                                  -->
<!-- An inline trademarked term                                       -->
<!ELEMENT trademark (#PCDATA)*                                        >
<!ATTLIST trademark
          class    (copyright | registered |
                    service | trade)       #IMPLIED                  >
<!--                                                                  -->
<!-- Return type of an OO method or type of a class field             -->
<!ELEMENT type (#PCDATA)*                                             >
<!--                                                                  -->
<!-- A URL link                                                       -->
<!ELEMENT ulink (#PCDATA)*                                            >
<!ATTLIST ulink
          url      CDATA                  #REQUIRED                  >
<!--                                                                  -->
<!-- User input as shown on a computer terminal                       -->
<!ELEMENT userinput (#PCDATA)*                                        >
<!--                                                                  -->
<!-- Repeated variable arguments in a function synopsis               -->
<!ELEMENT varargs EMPTY                                               >
<!--                                                                  -->
<!-- A list of terms and their associated explanations                -->
<!ELEMENT variablelist (varlistentry)+                                >
<!--                                                                  -->
<!-- A list item in a variable list                                   -->
<!ELEMENT varlistentry (term+, listitem)                              >
<!--                                                                  -->
<!-- Name of the variable in a class field                            -->
<!ELEMENT varname (#PCDATA)*                                          >
<!--                                                                  -->
<!-- A void argument to a function in a function synopsis             -->
<!ELEMENT void EMPTY                                                  >
<!--                                                                  -->
<!-- A cross reference to another element in the same document        -->
<!ELEMENT xref EMPTY                                                  >
<!ATTLIST xref
          linkend  IDREF                  #REQUIRED                   >
```

ManVolNum XSLT Stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ManVolNum XSLT Stylesheet                                    -->
<!-- Stylesheet for retrieving only ManVolNum for WebMan project  -->
<!--                                                              -->
<!-- Copyright 2002-2003, Mark Craig                              -->
<!--                                                              -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text" indent="no" encoding="UTF-8"/>
  <xsl:template match="/">
    <xsl:value-of select="refentry/refmeta/manvolnum"/>
  </xsl:template>
</xsl:stylesheet>
```

RefEntry XSLT Stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- RefEntry XSLT Stylesheet                                        -->
<!-- Stylesheet for single RefEntry document for WebMan project      -->
<!--                                                                 -->
<!-- Copyright 2002-2003, Mark Craig                                 -->
<!--                                                                 -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" indent="yes" encoding="UTF-8"/>
  <xsl:param name="ServletURL" select="'refentry'"/>
  <!--                                                               -->
  <!-- Section names used for the RefEntry pages.                    -->
  <!--                                                               -->
  <xsl:variable name="manvolnum">
    <xsl:value-of select="/refentry/refmeta/manvolnum"/>
  </xsl:variable>
  <xsl:variable name="section">
    <xsl:choose>
      <xsl:when test="$manvolnum = '1'">User Commands</xsl:when>
      <xsl:when test="$manvolnum = '1M'">Admin Commands</xsl:when>
      <xsl:when test="$manvolnum = '2'">System Calls</xsl:when>
      <xsl:when test="$manvolnum = '3'">Library Functions</xsl:when>
      <xsl:when test="$manvolnum = '4'">Special Files</xsl:when>
      <xsl:when test="$manvolnum = '5'">File Formats</xsl:when>
      <xsl:when test="$manvolnum = '6'">Games</xsl:when>
      <xsl:when test="$manvolnum = '7'">Conventions and Standards</xsl:when>
      <xsl:when test="$manvolnum = '8'">Admin Commands</xsl:when>
      <xsl:otherwise>Section unknown (see refentry.xslt)</xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <!--                                                               -->
  <!-- Output for the RefEntry is a division that belongs together.  -->
  <!--                                                               -->
  <xsl:template match="/">
    <div>
      <xsl:apply-templates select="refentry/*"/>
    </div>
  </xsl:template>
  <!--                                                               -->
  <!-- Output a table with the name of the page at the top.          -->
  <!--                                                               -->
  <xsl:variable name="refentrytitle">
    <xsl:value-of select="/refentry/refmeta/refentrytitle"/>
  </xsl:variable>
  <xsl:template match="refmeta">
    <table border="0" width="100%">
      <tr>
        <td align="left">
          <xsl:value-of select="$refentrytitle"/>(<xsl:value-of
    select="$manvolnum"/>)</td>
        <td align="center">
          <xsl:value-of select="$section"/>
        </td>
        <td align="right">
          <xsl:value-of select="$refentrytitle"/>(<xsl:value-of
   select="$manvolnum"/>)</td>
      </tr>
    </table>
  </xsl:template>
  <!--                                                               -->
  <!-- Handle the RefNameDiv of the document.                        -->
  <!--                                                               -->
  <xsl:template match="refnamediv">
    <h1>NAME</h1>
    <blockquote>
      <xsl:apply-templates select="refname">
        <xsl:sort/>
      </xsl:apply-templates>
      <xsl:text> - </xsl:text>
      <xsl:value-of select="refpurpose"/>
    </blockquote>
  </xsl:template>
  <xsl:template match="refname">
    <tt>
      <xsl:value-of select="."/>
    </tt>
    <xsl:if test="not(position()=last())">, </xsl:if>
  </xsl:template>
```

```
<!--                                                                -->
<!-- Handle the RefSynopsisDiv of the document.                     -->
<!--                                                                -->
<xsl:template match="refsynopsisdiv">
  <xsl:if test="@id">
    <a name="{@id}"/>
  </xsl:if>
  <h1>
    <xsl:value-of select="title"/>
  </h1>
  <blockquote>
    <p>
      <tt>
        <xsl:apply-templates select="classsynopsis | cmdsynopsis |
  constructorsynopsis | destructorsynopsis | fieldsynopsis |
  funcsynopsis | methodsynopsis | synopsis"/>
      </tt>
    </p>
  </blockquote>
</xsl:template>
<xsl:template match="classsynopsis">
  <xsl:apply-templates select="ooclass | ooexception | oointerface"/>
  <xsl:apply-templates select="classsynopsisinfo | constructorsynopsis |
   destructorsynopsis | fieldsynopsis | methodsynopsis"/>
</xsl:template>
<xsl:template match="ooclass | ooexception | oointerface |
 classsynopsisinfo | funcsynopsisinfo">
  <xsl:apply-templates/>
  <br />
</xsl:template>
<xsl:template match="modifier| command | type">
  <xsl:value-of select="concat(., ' ')"/>
</xsl:template>
<xsl:template match="classname | exceptionname | interfacename |
 command | methodname | varname | function">
  <xsl:if test="classname">
    <xsl:text>class </xsl:text>
  </xsl:if>
  <xsl:if test="exceptionname">
    <xsl:text>exception </xsl:text>
  </xsl:if>
  <xsl:if test="interfacename">
    <xsl:text>interface </xsl:text>
  </xsl:if>
  <b>
    <xsl:value-of select="concat(., ' ')"/>
  </b>
</xsl:template>
<xsl:template match="cmdsynopsis | funcsynopsis | funcdef | synopsis">
  <xsl:apply-templates/>
</xsl:template>
<xsl:template match="arg | group">
  <xsl:choose>
    <xsl:when test="@choice = 'opt'">[ <xsl:apply-templates/> ]</xsl:when>
    <xsl:when test="@choice = 'required'">{ <xsl:apply-templates
     /> }</xsl:when>
    <xsl:when test="@rep = 'repeat'">
      <xsl:apply-templates/> ...</xsl:when>
    <xsl:otherwise>
      <xsl:apply-templates/>
      <xsl:text> </xsl:text>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:template match="methodparam">
  <xsl:choose>
    <xsl:when test="not(position()=last())">
      <xsl:apply-templates/>
      <xsl:if test="@rep = 'repeat'">
        <xsl:text>...</xsl:text>
      </xsl:if>
      <xsl:text>, </xsl:text>
    </xsl:when>
    <xsl:otherwise>
      <xsl:apply-templates/>
      <xsl:if test="@rep = 'repeat'">
        <xsl:text>...</xsl:text>
      </xsl:if>
    </xsl:otherwise>
  </xsl:choose>
```

```
</xsl:template>
<xsl:template match="replaceable|parameter|citetitle|emphasis">
  <i>
    <xsl:value-of select="."/>
  </i>
</xsl:template>
<xsl:template match="sbr">
  <br />
</xsl:template>
<xsl:template match="constructorsynopsis | destructorsynopsis">
  <xsl:apply-templates select="modifier"/>
  <xsl:if test="destructorsynopsis">~</xsl:if>
  <xsl:choose>
    <xsl:when test="methodname">
      <b>
        <xsl:value-of select="methodname"/>
      </b>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$refentrytitle"/>
    </xsl:otherwise>
  </xsl:choose>(<xsl:apply-templates select="methodparam"/>
  <xsl:apply-templates select="void"/>)<xsl:if test="exceptionname">
    <xsl:text> throws </xsl:text><xsl:apply-templates
    select="exceptionname"/></xsl:if>;<br />
</xsl:template>
<xsl:template match="void">void</xsl:template>
<xsl:template match="fieldsynopsis">
  <xsl:apply-templates/>;<br />
</xsl:template>
<xsl:template match="initializer">
  <xsl:value-of select="concat(' = ', .)"/>
</xsl:template>
<xsl:template match="funcprototype">
  <xsl:apply-templates select="funcdef"/>(<xsl:apply-templates
   select="varargs | void | paramdef"/>);<br />
</xsl:template>
<xsl:template match="varargs">
  <xsl:text>...</xsl:text>
</xsl:template>
<xsl:template match="paramdef">
  <xsl:apply-templates/>
  <xsl:if test="not(position()=last())">, </xsl:if>
</xsl:template>
<xsl:template match="methodsynopsis">
  <xsl:apply-templates select="modifier"/>
  <xsl:choose>
    <xsl:when test="type">
      <xsl:value-of select="concat(type, ' ')"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:apply-templates select="void"/>
      <xsl:text> </xsl:text>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:apply-templates select="methodname"/>(<xsl:apply-templates
   select="methodparam"/><xsl:apply-templates
   select="void"/>)<xsl:if test="exceptionname"><xsl:text>
   throws </xsl:text><xsl:apply-templates
   select="exceptionname"/></xsl:if>;<br />
</xsl:template>
<!--                                                                -->
<!-- Handle a RefSect1.                                             -->
<!--                                                                -->
<xsl:template match="refsect1">
  <xsl:if test="@id">
    <a name="{@id}"/>
  </xsl:if>
  <h1>
    <xsl:value-of select="title"/>
  </h1>
  <blockquote>
    <xsl:apply-templates select="para | blockquote | example |
     itemizedlist | msgset | orderedlist | programlisting | screen |
     table | variablelist | refsect2"/>
  </blockquote>
</xsl:template>
<xsl:template match="para">
  <p>
    <xsl:apply-templates/>
```

```
  </p>
</xsl:template>
<xsl:template match="citerefentry">
  <a>
    <xsl:attribute name="href"><xsl:value-of
     select="$ServletURL"/>?RefEntry=<xsl:value-of
     select="refentrytitle"/>-<xsl:value-of
     select="manvolnum"/></xsl:attribute>
    <tt>
      <xsl:value-of select="refentrytitle"/>
    </tt>(<xsl:value-of select="manvolnum"/>)
  </a>
</xsl:template>
<xsl:template match="errorcode | errorname | errortype | literal">
  <tt>
    <xsl:value-of select="."/>
  </tt>
</xsl:template>
<xsl:template match="quote">
  <xsl:text>"</xsl:text>
    <xsl:value-of select="."/>
  <xsl:text>"</xsl:text>
</xsl:template>
<xsl:template match="trademark">
  <xsl:value-of select="."/>
  <xsl:choose>
    <xsl:when test="@class = 'copyright'">
      <xsl:text>[C]</xsl:text>
    </xsl:when>
    <xsl:when test="@class = 'registered'">
      <xsl:text>[R]</xsl:text>
    </xsl:when>
    <xsl:when test="@class = 'service'">
      <xsl:text>[sm]</xsl:text>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text>[tm]</xsl:text>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:template match="ulink">
  <a href="{@url}">
    <xsl:value-of select="."/>
  </a>
</xsl:template>
<xsl:template match="xref">
  <a href="#{@linkend}">
    <xsl:value-of select="id(@linkend)/title"/>
  </a>
</xsl:template>
<xsl:template match="blockquote">
  <blockquote>
    <table border="0">
      <tr>
        <td>
          <xsl:apply-templates select="para"/>
        </td>
      </tr>
      <tr>
        <td align="right">
          <xsl:text>- </xsl:text>
          <xsl:value-of select="attribution"/>
        </td>
      </tr>
    </table>
  </blockquote>
</xsl:template>
<!--                                                                  -->
<!-- Handle an Example.                                               -->
<!--                                                                  -->
<xsl:template match="example">
  <xsl:if test="@id">
    <a name="{@id}"/>
  </xsl:if>
  <h2>
    <xsl:value-of select="title"/>
  </h2>
  <xsl:apply-templates select="para | programlisting | screen"/>
</xsl:template>
<xsl:template match="programlisting">
```

```
  <hr noshade="noshade" />
  <pre>
    <xsl:apply-templates/>
  </pre>
  <hr noshade="noshade" />
</xsl:template>
<xsl:template match="screen">
  <table border="1" width="90%">
    <tr>
      <td>
        <pre>
          <xsl:apply-templates/>
        </pre>
      </td>
    </tr>
  </table>
</xsl:template>
<xsl:template match="computeroutput">
  <xsl:apply-templates/>
</xsl:template>
<xsl:template match="userinput">
  <b>
    <xsl:apply-templates/>
  </b>
</xsl:template>
<!--                                                                -->
<!-- Handle lists and tables.                                       -->
<!--                                                                -->
<xsl:template match="itemizedlist">
  <ul>
    <xsl:apply-templates select="listitem"/>
  </ul>
</xsl:template>
<xsl:template match="listitem">
  <xsl:choose>
    <xsl:when test="parent::varlistentry">
      <xsl:apply-templates/>
    </xsl:when>
    <xsl:otherwise>
      <li>
        <xsl:apply-templates select="para | itemizedlist |
  orderedlist | variablelist"/>
      </li>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:template match="msgset">
  <div>
    <xsl:apply-templates select="msgentry"/>
  </div>
</xsl:template>
<xsl:template match="msgentry">
  <xsl:apply-templates select="msg | msginfo | msgexplan"/>
</xsl:template>
<xsl:template match="msg">
  <xsl:apply-templates select="msgmain | msgrel | msgsub"/>
</xsl:template>
<xsl:template match="msgmain | msgrel | msgsub">
  <xsl:apply-templates select="msgtext"/>
</xsl:template>
<xsl:template match="msgtext | msgexplan">
  <xsl:apply-templates select="para | blockquote | itemizedlist |
   orderedlist | programlisting | screen | variablelist"/>
</xsl:template>
<xsl:template match="msginfo">
  <xsl:apply-templates select="msglevel | msgorig"/>
</xsl:template>
<xsl:template match="msglevel">
  <p>
    <b>Level</b>
    <xsl:value-of select="concat(' ',.)"/>
  </p>
</xsl:template>
<xsl:template match="msgorig">
  <p>
    <b>Origin</b>
    <xsl:value-of select="concat(' ',.)"/>
  </p>
</xsl:template>
<xsl:template match="orderedlist">
```

```
  <ol>
    <xsl:apply-templates select="listitem"/>
  </ol>
</xsl:template>
<xsl:template match="table">
  <h2>
    <xsl:value-of select="title"/>
  </h2>
  <table>
    <xsl:choose>
      <xsl:when test="@colsep = 1">
        <xsl:attribute
  name="rules"><xsl:text>cols</xsl:text></xsl:attribute>
      </xsl:when>
      <xsl:when test="@frame">
        <xsl:choose>
          <xsl:when test="@frame = 'bottom'">
            <xsl:attribute
      name="frame"><xsl:text>below</xsl:text></xsl:attribute>
          </xsl:when>
          <xsl:when test="@frame = 'none'">
            <xsl:attribute
      name="frame"><xsl:text>void</xsl:text></xsl:attribute>
          </xsl:when>
          <xsl:when test="@frame = 'sides'">
            <xsl:attribute
      name="frame"><xsl:text>box</xsl:text></xsl:attribute>
          </xsl:when>
          <xsl:when test="@frame = 'top'">
            <xsl:attribute
      name="frame"><xsl:text>above</xsl:text></xsl:attribute>
          </xsl:when>
          <xsl:when test="@frame = 'topbot'">
            <xsl:attribute
      name="frame"><xsl:text>hsides</xsl:text></xsl:attribute>
          </xsl:when>
          <xsl:otherwise>
            <xsl:attribute
      name="border"><xsl:text>1</xsl:text></xsl:attribute>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:when>
      <xsl:when test="@rowsep = 1">
        <xsl:attribute
  name="rules"><xsl:text>rows</xsl:text></xsl:attribute>
      </xsl:when>
    </xsl:choose>
    <xsl:attribute name="width"><xsl:text>90%</xsl:text></xsl:attribute>
    <xsl:apply-templates select="tgroup"/>
  </table>
</xsl:template>
<xsl:template match="tgroup">
  <xsl:apply-templates select="thead | tbody"/>
</xsl:template>
<xsl:template match="thead">
  <thead>
    <xsl:if test="@valign">
      <xsl:attribute name="valign"><xsl:choose><xsl:when
test="@valign = 'bottom'">bottom</xsl:when><xsl:when
test="@valign = 'top'">top</xsl:when><xsl:otherwise
>middle</xsl:otherwise></xsl:choose></xsl:attribute>
    </xsl:if>
    <xsl:apply-templates select="row"/>
  </thead>
</xsl:template>
<xsl:template match="tbody">
  <tbody>
    <xsl:if test="@valign">
      <xsl:attribute name="valign"><xsl:choose><xsl:when
test="@valign = 'bottom'">bottom</xsl:when><xsl:when
test="@valign = 'top'">top</xsl:when><xsl:otherwise
>middle</xsl:otherwise></xsl:choose></xsl:attribute>
    </xsl:if>
    <xsl:apply-templates select="row"/>
  </tbody>
</xsl:template>
<xsl:template match="row">
  <tr>
    <xsl:if test="@valign">
      <xsl:attribute name="valign"><xsl:choose><xsl:when
```

```
 test="@valign = 'bottom'">bottom</xsl:when><xsl:when
 test="@valign = 'top'">top</xsl:when><xsl:otherwise
 >middle</xsl:otherwise></xsl:choose></xsl:attribute>
     </xsl:if>
     <xsl:apply-templates select="entry"/>
   </tr>
 </xsl:template>
 <xsl:template match="entry">
   <td>
     <xsl:if test="@align">
       <xsl:attribute name="align"><xsl:choose><xsl:when
 test="@align = 'left'">left</xsl:when><xsl:when
 test="@align = 'right'">right</xsl:when><xsl:otherwise
 >center</xsl:otherwise></xsl:choose></xsl:attribute>
     </xsl:if>
     <xsl:if test="@morerows">
       <xsl:attribute name="rowspan">
  <xsl:value-of select="number(@morerows + 1)"/>
</xsl:attribute>
     </xsl:if>
     <xsl:if test="@valign">
       <xsl:attribute name="valign"><xsl:choose><xsl:when
 test="@valign = 'bottom'">bottom</xsl:when><xsl:when
 test="@valign = 'top'">top</xsl:when><xsl:otherwise
 >middle</xsl:otherwise></xsl:choose></xsl:attribute>
     </xsl:if>
     <xsl:apply-templates select="para"/>
   </td>
 </xsl:template>
 <xsl:template match="variablelist">
   <blockquote>
     <dl>
       <xsl:apply-templates select="varlistentry"/>
     </dl>
   </blockquote>
 </xsl:template>
 <xsl:template match="varlistentry">
   <xsl:apply-templates select="term"/>
   <dd>
     <xsl:apply-templates select="listitem"/>
   </dd>
 </xsl:template>
 <xsl:template match="term">
   <dt>
     <tt>
       <xsl:value-of select="."/>
     </tt>
   </dt>
 </xsl:template>
 <!--                                                                -->
 <!-- Handle a RefSect2.                                             -->
 <!--                                                                -->
 <xsl:template match="refsect2">
   <xsl:if test="@id">
     <a name="{@id}"/>
   </xsl:if>
   <h2>
     <xsl:value-of select="title"/>
   </h2>
   <blockquote>
     <xsl:apply-templates select="para | blockquote | itemizedlist |
      orderedlist | programlisting | screen | variablelist"/>
   </blockquote>
 </xsl:template>
</xsl:stylesheet>
```

RefName XSLT Stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- RefName XSLT Stylesheet                                        -->
<!-- Stylesheet for retrieving only RefName for WebMan project      -->
<!--                                                                -->
<!-- Copyright 2002-2003, Mark Craig                                -->
<!--                                                                -->
<!-- Use RefEntryTitle only to avoid multiple RefName elements.     -->
<!--                                                                -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text" indent="no" encoding="UTF-8"/>
  <xsl:template match="/">
    <xsl:value-of select="refentry/refmeta/refentrytitle"/>
  </xsl:template>
</xsl:stylesheet>
```

RefPurpose XSLT Stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- RefPurpose XSLT Stylesheet                                     -->
<!-- Stylesheet for retrieving only RefPurpose for WebMan project   -->
<!--                                                                -->
<!-- Copyright 2002-2003, Mark Craig                                -->
<!--                                                                -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text" indent="no" encoding="UTF-8"/>
  <xsl:template match="/">
    <xsl:value-of select="refentry/refnamediv/refpurpose"/>
  </xsl:template>
</xsl:stylesheet>
```

StripXML XSLT Stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- StripXML XSLT Stylesheet                                       -->
<!-- Stylesheet for stripping markup from XML for WebMan project    -->
<!--                                                                -->
<!-- Copyright 2002-2003, Mark Craig                                -->
<!--                                                                -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text"/>
  <xsl:template match="text()">
    <xsl:value-of select="."/>
  </xsl:template>
</xsl:stylesheet>
```

Acceptance Test for Unit Testing

```
package org.mcraig.cs445.refentry;
import junit.framework.TestSuite;

/**
 * Basic functional tests for the RefEntry servlet code that
 * can be performed using the JUnit test framework.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
public class AcceptanceTest {
    public static TestSuite suite() {
        TestSuite suite = new TestSuite();
        suite.addTest(new TestRefEntryBackEnd());
        suite.addTest(new TestRefEntryIndex());
        suite.addTest(new TestRefEntry());
        suite.addTest(new TestRefEntryServlet());
        suite.addTest(new TestRefEntryTransformer());
        suite.addTest(new TestRefEntryValidator());
```

```
        return suite;
    }
    public static void main(String[] args) {
        junit.textui.TestRunner.run(suite());
    }
}
```

Small Entry for Unit Testing

```
<?xml version="1.0"?>
<!DOCTYPE refentry SYSTEM "src/refentry.dtd">
<refentry>
  <refmeta>
    <refentrytitle>test</refentrytitle>
    <manvolnum>test</manvolnum>
  </refmeta>
  <refnamediv>
    <refname>test</refname>
    <refpurpose>test RefEntry servlet code</refpurpose>
  </refnamediv>
  <refsynopsisdiv>
    <title>SYNOPSIS</title>
    <synopsis>A synopsis.</synopsis>
  </refsynopsisdiv>
  <refsect1>
    <title>REFSECT1</title>
    <para>A reference section.</para>
  </refsect1>
</refentry>
```

Small Entry Result for Unit Testing

```
<div>
<table width="100%" border="0">
<tr>
<td align="left">test(test)</td><td align="center">Section unknown (see refentry.xslt)</td><td align="right">tes
</tr>
</table>
<h1>NAME</h1>
<blockquote>
<tt>test</tt> - test RefEntry servlet code</blockquote>
<h1>SYNOPSIS</h1>
<blockquote>
<p>
<tt>A synopsis.</tt>
</p>
</blockquote>
<h1>REFSECT1</h1>
<blockquote>
<p>A reference section.</p>
</blockquote>
</div>
```

Large Entry for Stylesheet Testing

```
<?xml version="1.0"?>
<!DOCTYPE refentry SYSTEM "src/refentry.dtd">
<refentry id="refentry">
  <refmeta>
    <refentrytitle>test</refentrytitle>
    <manvolnum>test</manvolnum>
    <refmiscinfo class="arch">arch refmiscinfo</refmiscinfo>
    <refmiscinfo class="copyright">copyright refmiscinfo</refmiscinfo>
    <refmiscinfo class="date">date refmiscinfo</refmiscinfo>
    <refmiscinfo class="sectdesc">sectdesc refmiscinfo</refmiscinfo>
    <refmiscinfo class="software">software refmiscinfo</refmiscinfo>
```

```
 </refmeta>
 <refnamediv>
   <refname>refname1</refname>
   <refname>refname2</refname>
   <refname>refname3</refname>
   <refpurpose>test refpurpose</refpurpose>
 </refnamediv>
 <refsynopsisdiv id="refsynopsisdiv">
   <title>SYNOPSIS</title>
   <funcsynopsis>
     <funcprototype>
<funcdef>type <function>function1</function></funcdef>
<paramdef>type <parameter>param1</parameter></paramdef>
<paramdef>type <parameter>param2</parameter></paramdef>
     </funcprototype>
     <funcprototype>
<funcdef>type <function>function2</function></funcdef>
<void/>
     </funcprototype>
     <funcprototype>
<funcdef>type <function>function3</function></funcdef>
<varargs/>
     </funcprototype>
   </funcsynopsis>
   <classsynopsis>
     <ooclass>
<modifier>modifier1</modifier>
<modifier>modifier2</modifier>
<classname>ClassName</classname>
     </ooclass>
     <oointerface>
<modifier>implements</modifier>
<interfacename>InterfaceName</interfacename>
     </oointerface>
     <ooexception>
<modifier>throws</modifier>
<exceptionname>ExceptionName</exceptionname>
     </ooexception>
     <classsynopsisinfo>Classsynopsisinfo here.</classsynopsisinfo>
     <constructorsynopsis>
<modifier>modifier1</modifier>
<modifier>modifier2</modifier>
<methodparam>
  <modifier>modifier</modifier>
  <type>type</type>
  <parameter>param</parameter>
  <initializer>initializer</initializer>
</methodparam>
<methodparam>
  <funcparams>funcparams</funcparams>
</methodparam>
<exceptionname>ExceptionName</exceptionname>
     </constructorsynopsis>
     <destructorsynopsis>
<void/>
<exceptionname>Exception1</exceptionname>
<exceptionname>Exception2</exceptionname>
     </destructorsynopsis>
     <fieldsynopsis>
<modifier>modifier</modifier>
<type>type</type>
<varname>VarName</varname>
<initializer>initializer</initializer>
     </fieldsynopsis>
     <methodsynopsis>
<modifier>modifier1</modifier>
<modifier>modifier2</modifier>
<type>type</type>
<methodname>methodName</methodname>
<methodparam>
  <modifier>modifier</modifier>
  <type>type</type>
  <parameter>parameter</parameter>
  <initializer>initializer</initializer>
</methodparam>
     </methodsynopsis>
   </classsynopsis>
   <cmdsynopsis>
     <command>command</command>
     <group choice="opt">
```

```
<arg>-optional <replaceable>arg1-replaceable</replaceable></arg>
<sbr/>
<arg choice="opt" rep="repeat">-optional repeating-literal-arg2</arg>
<sbr/>
<arg choice="required">-required literal-arg3</arg>
     </group>
     <arg rep="norepeat"><replaceable>replaceable</replaceable></arg>
     <sbr/>
     <arg>final-arg</arg>
     <sbr/>
   </cmdsynopsis>
   <synopsis>A plain synopsis.</synopsis>
 </refsynopsisdiv>
 <refsect1 id="para">
   <title>Paras and Lists</title>
   <para>This para includes <citerefentry>
   <refentrytitle>man</refentrytitle> <manvolnum>1</manvolnum>
   </citerefentry>, <citetitle>citetitle</citetitle>,
   <classname>classname</classname>, <command>command</command>,
   <emphasis>emphasis</emphasis>, <errorcode>errorcode</errorcode>,
   <errorname>errorname</errorname>,
   <errortype>errortype</errortype>, <function>function</function>,
   <literal>literal</literal>, <quote>quote</quote>,
   <replaceable>replaceable</replaceable>,
   <trademark>trademark</trademark>, <trademark
   class="copyright">copyright</trademark>, <trademark
   class="registered">registered trademark</trademark>, <trademark
   class="service">service mark</trademark>, a ulink: <ulink
   url="http://mcraig.org/">http://mcraig.org/</ulink>, and an xref
   to <xref linkend="refsynopsisdiv"/>.</para>
   <para>This para is just text. This sentence is false. This
   sentence is false. This sentence is false. This sentence is
   false. This sentence is false. This sentence is false. This
   sentence is false. This sentence is false. This sentence is
   false. This sentence is false. This sentence is false. This
   sentence is false. This sentence is false.</para>
   <itemizedlist>
     <listitem>
<para>item 1</para>
     </listitem>
     <listitem>
<para>item 2</para>
     </listitem>
     <listitem>
<para>item 3</para>
     </listitem>
     <listitem>
<para>item 4</para>
<itemizedlist>
  <listitem>
    <para>item 1</para>
  </listitem>
  <listitem>
    <para>item 2</para>
    <itemizedlist>
      <listitem>
 <para>item 0</para>
      </listitem>
      <listitem>
 <para>item 1</para>
      </listitem>
      <listitem>
 <para>item 2</para>
      </listitem>
      <listitem>
 <para>item 3</para>
      </listitem>
    </itemizedlist>
  </listitem>
  <listitem>
    <para>item 3</para>
  </listitem>
</itemizedlist>
     </listitem>
     <listitem>
<para>item 5</para>
     </listitem>
   </itemizedlist>
   <orderedlist>
     <listitem>
```

```
<para>item 1</para>
     </listitem>
     <listitem>
<para>item 2</para>
     </listitem>
     <listitem>
<para>item 3</para>
     </listitem>
     <listitem>
<para>item 4</para>
<orderedlist>
  <listitem>
    <para>item 1</para>
  </listitem>
  <listitem>
    <para>item 2</para>
    <orderedlist>
      <listitem>
 <para>item 0</para>
      </listitem>
      <listitem>
 <para>item 1</para>
      </listitem>
      <listitem>
 <para>item 2</para>
      </listitem>
      <listitem>
 <para>item 3</para>
      </listitem>
    </orderedlist>
  </listitem>
  <listitem>
    <para>item 3</para>
  </listitem>
</orderedlist>
     </listitem>
     <listitem>
<para>item 5</para>
     </listitem>
   </orderedlist>
   <variablelist>
     <varlistentry>
<term>term1</term>
<listitem>
  <para>Variable list entry 1</para>
</listitem>
     </varlistentry>
     <varlistentry>
<term>term 2</term>
<listitem>
  <para>Variable list entry 2</para>
  <itemizedlist>
    <listitem>
      <para>item 1</para>
      <orderedlist>
 <listitem>
   <para>item 1 a</para>
 </listitem>
 <listitem>
   <para><replaceable>replaceable</replaceable> item 2 a</para>
 </listitem>
      </orderedlist>
    </listitem>
    <listitem>
      <para>item 2, which includes more than item 1. This para
      actually includes a couple of sentences.</para>
    </listitem>
  </itemizedlist>
</listitem>
     </varlistentry>
     <varlistentry>
<term>term 3</term>
<listitem>
  <para>listitem para</para>
</listitem>
     </varlistentry>
   </variablelist>
 </refsect1>
 <refsect1 id="example">
   <title>Example</title>
```

```
    <para>This section contains an example.</para>
    <example id="anExample">
      <title>An Example</title>
      <screen>[mcraig@localhost mcraig]$ <userinput>su</userinput>
Password: <replaceable>password</replaceable>
[root@localhost mcraig]# <userinput>cat /etc/hosts</userinput>
<computeroutput># Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1  localhost.localdomain localhost
192.168.0.99  lethe</computeroutput>
[root@localhost mcraig]# </screen>
      <para>Here's another screen.</para>
      <screen>[mcraig@localhost mcraig]$ <userinput>ls</userinput>
<computeroutput>archive       install.inf  MarkCraig_CS430_3.doc     refentry.dtd  void.html
arg.html      junit        MarkCraig_CS445_book.pdf  testabort     webman.html
cvs           junk         matchingrule.c            testpasswd
debugger.jar  m            plug.tar.gz               test.xml
examples      Mail         public_html</computeroutput>
[mcraig@localhost mcraig]$</screen>
      <para>Here's a programlisting.</para>
      <programlisting>/*
 * redirect is the pseudo-tty that console output
 * is redirected to if asked by TIOCCONS.
 */
struct tty_struct * redirect;

static void initialize_tty_struct(struct tty_struct *tty);

static ssize_t tty_read(struct file *, char *, size_t, loff_t *);
static ssize_t tty_write(struct file *, const char *, size_t, loff_t *);
static unsigned int tty_poll(struct file *, poll_table *);
static int tty_open(struct inode *, struct file *);
static int tty_release(struct inode *, struct file *);
int tty_ioctl(struct inode * inode, struct file * file,
              unsigned int cmd, unsigned long arg);
static int tty_fasync(int fd, struct file * filp, int on);
extern int vme_scc_init (void);
extern long vme_scc_console_init(void);
extern int serial167_init(void);
extern long serial167_console_init(void);
extern void console_8xx_init(void);
extern int rs_8xx_init(void);
extern void mac_scc_console_init(void);
</programlisting>
    </example>
  </refsect1>
  <refsect1 id="msgset">
    <title>MsgSet</title>
    <para>This section contains a message set.</para>
    <msgset>
      <msgentry>
 <msg>
   <msgmain>
     <msgtext>
       <para>msgtext para</para>
       <blockquote>
  <attribution>Lao Tsu</attribution>
  <para>He who knows others is wise.</para>
  <para>He who knows himself is enlightened.</para>
       </blockquote>
       <para>Do you know who you are?</para>
     </msgtext>
   </msgmain>
   <msgsub>
     <msgtext>
       <para>msgsub msgtext para</para>
       <para>another para</para>
     </msgtext>
   </msgsub>
   <msgrel>
     <msgtext>
       <para>msgrel msgtext para</para>
       <para>another para</para>
     </msgtext>
   </msgrel>
 </msg>
 <msginfo>
   <msgorig>msgorig from somewhere in the machine</msgorig>
 </msginfo>
 <msgexplan>
```

```
    <para>This is supposed to explain.</para>
    <blockquote>
      <attribution>Lyman Bryson</attribution>
      <para>The error of youth is to believe that intelligence
      is a substitute for experience, while the error of age is
      to believe experience is a substitute for
      intelligence.</para>
    </blockquote>
    <orderedlist>
      <listitem>
        <para>one</para>
      </listitem>
      <listitem>
        <para>two</para>
      </listitem>
      <listitem>
        <para>three</para>
      </listitem>
    </orderedlist>
    <para>And there you have it.</para>
  </msgexplan>
       </msgentry>
       <msgentry>
  <msg>
    <msgmain>
      <msgtext>
        <para>another</para>
        <variablelist>
   <varlistentry>
     <term>error <replaceable>variable list</replaceable></term>
     <listitem>
       <para>Somewhere here in the machine</para>
     </listitem>
   </varlistentry>
        </variablelist>
      </msgtext>
    </msgmain>
    <msgsub>
      <msgtext>
        <para>msgsub</para>
        <screen>$ <userinput>cat /dev/null</userinput></screen>
      </msgtext>
    </msgsub>
    <msgrel>
      <msgtext>
        <para>msgrel</para>
        <programlisting>#!/bin/sh -x
/bin/ls -l</programlisting>
      </msgtext>
    </msgrel>
  </msg>
  <msginfo>
    <msglevel>very, very, very serious</msglevel>
  </msginfo>
  <msgexplan>
    <para><quote>Meets Tough Quality Standards</quote>: It
    compiles without errors.</para>
  </msgexplan>
       </msgentry>
     </msgset>
     <refsect2 id="refsect2a">
       <title>A refsect2</title>
       <para>Here’s a refsect2.</para>
     </refsect2>
     <refsect2 id="refsect2b">
       <title>A refsect2</title>
       <para>Here’s another refsect2.</para>
     </refsect2>
     <refsect2 id="refsect2c">
       <title>A refsect2</title>
       <para>Here’s yet another refsect2.</para>
     </refsect2>
     <refsect2 id="refsect2d">
       <title>A refsect2</title>
       <para>Here’s a last refsect2.</para>
     </refsect2>
   </refsect1>
   <refsect1 id="table">
     <title>Tables</title>
     <para>This section includes tables.</para>
```

```
   <table frame="topbot">
     <title>A topbot Table</title>
     <tgroup>
<thead valign="bottom">
  <row>
    <entry>
      <para>head1</para>
    </entry>
    <entry>
      <para>head2</para>
    </entry>
    <entry>
      <para>head3</para>
    </entry>
  </row>
</thead>
<tbody valign="top">
  <row valign="top">
    <entry align="left" valign="bottom">
      <para>A para.</para>
    </entry>
    <entry align="center" valign="middle">
      <para>A para.</para>
    </entry>
    <entry align="right" valign="top">
      <para>A para.</para>
    </entry>
  </row>
  <row valign="middle">
    <entry morerows="1">
      <para>morerows is 1</para>
    </entry>
    <entry>
      <para>cell2</para>
    </entry>
    <entry>
      <para>cell3</para>
    </entry>
  </row>
  <row valign="top">
    <entry>
      <para>cell2</para>
    </entry>
    <entry>
      <para>cell3</para>
    </entry>
  </row>
</tbody>
     </tgroup>
   </table>
   <table frame="all">
     <title>An all table</title>
     <tgroup>
<thead>
  <row>
    <entry>
      <para>head1</para>
    </entry>
    <entry>
      <para>head2</para>
    </entry>
  </row>
</thead>
<tbody>
  <row>
    <entry>
      <para>cell1</para>
    </entry>
    <entry>
      <para>cell2</para>
    </entry>
  </row>
  <row>
    <entry>
      <para>cell3</para>
    </entry>
    <entry>
      <para>cell4</para>
    </entry>
  </row>
```

```
</tbody>
     </tgroup>
   </table>
   <table frame="bottom">
     <title>A bottom table</title>
     <tgroup>
<thead>
  <row>
    <entry>
      <para>head1</para>
    </entry>
    <entry>
      <para>head2</para>
    </entry>
  </row>
</thead>
<tbody>
  <row>
    <entry>
      <para>cell1</para>
    </entry>
    <entry>
      <para>cell2</para>
    </entry>
  </row>
  <row>
    <entry>
      <para>cell3</para>
    </entry>
    <entry>
      <para>cell4</para>
    </entry>
  </row>
</tbody>
     </tgroup>
   </table>
   <table frame="none">
     <title>A none table</title>
     <tgroup>
<thead>
  <row>
    <entry>
      <para>head1</para>
    </entry>
    <entry>
      <para>head2</para>
    </entry>
  </row>
</thead>
<tbody>
  <row>
    <entry>
      <para>cell1</para>
    </entry>
    <entry>
      <para>cell2</para>
    </entry>
  </row>
  <row>
    <entry>
      <para>cell3</para>
    </entry>
    <entry>
      <para>cell4</para>
    </entry>
  </row>
</tbody>
     </tgroup>
   </table>
   <table frame="sides">
     <title>A sides table</title>
     <tgroup>
<thead>
  <row>
    <entry>
      <para>head1</para>
    </entry>
    <entry>
      <para>head2</para>
    </entry>
```

```
  </row>
</thead>
<tbody>
  <row>
    <entry>
      <para>cell1</para>
    </entry>
    <entry>
      <para>cell2</para>
    </entry>
  </row>
  <row>
    <entry>
      <para>cell3</para>
    </entry>
    <entry>
      <para>cell4</para>
    </entry>
  </row>
</tbody>
     </tgroup>
   </table>
   <table frame="top">
     <title>A top table</title>
     <tgroup>
<thead>
  <row>
    <entry>
      <para>head1</para>
    </entry>
    <entry>
      <para>head2</para>
    </entry>
  </row>
</thead>
<tbody>
  <row>
    <entry>
      <para>cell1</para>
    </entry>
    <entry>
      <para>cell2</para>
    </entry>
  </row>
  <row>
    <entry>
      <para>cell3</para>
    </entry>
    <entry>
      <para>cell4</para>
    </entry>
  </row>
</tbody>
     </tgroup>
   </table>
   <table colsep="1" rowsep="1">
     <title>A colsep=1, rowsep=1 table</title>
     <tgroup>
<thead>
  <row>
    <entry>
      <para>head1</para>
    </entry>
    <entry>
      <para>head2</para>
    </entry>
  </row>
</thead>
<tbody>
  <row>
    <entry>
      <para>cell1</para>
    </entry>
    <entry>
      <para>cell2</para>
    </entry>
  </row>
  <row>
    <entry>
      <para>cell3</para>
```

```
    </entry>
    <entry>
      <para>cell4</para>
    </entry>
   </row>
 </tbody>
     </tgroup>
   </table>
  </refsect1>
</refentry>
```

Invalid Entry for Stylesheet Testing

```
<?xml version="1.0"?>
<!DOCTYPE refentry SYSTEM "src/refentry.dtd">
<refentry>
  <refmeta>
    <refentrytitle>test</refentrytitle>
    <manvolnum>test</manvolnum>
  </refmeta>
  <refnamediv>
    <refname>test</refname>
    <refpurpose>test RefEntry servlet code</refpurpose>
  </refnamediv>
  <refsynopsisdiv>
    <title>SYNOPSIS</title>
    <synopsis>A synopsis.</synopsis>
  </refsynopsisdiv>
  <refsect1>
    <title>REFSECT1</title>
    <para>A reference section.</para>
  </refsect1>
<!-- </refentry> -->
```

Test for Garbage Requests

```
import com.meterware.httpunit.GetMethodWebRequest;
import com.meterware.httpunit.WebConversation;
import com.meterware.httpunit.WebRequest;
import com.meterware.httpunit.WebResponse;
import junit.framework.TestCase;
import junit.framework.TestSuite;

/**
 * HttpUnit mechanism to test that the servlet manages to handle
 * many kinds of garbage without choking. This mechanism only works
 * when the servlet is running in the container and is reachable
 * under http://localhost:8080/refentry
 * <p>The test is then run by hand. Ideally, it would be integrated
 * into a framework such as Cactus.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
public class TestGarbageRequests extends TestCase {
    public static void main(String[] args) {
        junit.textui.TestRunner.run(suite());
    }
    public static TestSuite suite() {
        return new TestSuite(TestGarbageRequests.class);
    }
    public TestGarbageRequests(String str) {
        super(str);
    }
    public void testGarbageRequests() throws Exception {
        WebConversation wc = new WebConversation();

        WebRequest  request  =
            new GetMethodWebRequest(
                "http://localhost:8080/refentry");
        WebResponse response = wc.getResponse(request);
        String      title    = response.getTitle();
```

```
        // These garbage requests should return the default page.
        WebRequest  req1 =
            new GetMethodWebRequest(
                "http://localhost:8080/refentry?");
        WebResponse res1 = wc.getResponse(req1);
        assertEquals(title, res1.getTitle());

        WebRequest  req2 =
            new GetMethodWebRequest(
                "http://localhost:8080/refentry?garbage");
        WebResponse res2 = wc.getResponse(req2);
        assertEquals(title, res2.getTitle());

        WebRequest  req3 =
            new GetMethodWebRequest(
                "http://localhost:8080/refentry?grbg+in=grbg+out");
        WebResponse res3 = wc.getResponse(req3);
        assertEquals(title, res3.getTitle());

        WebRequest  req4 =
            new GetMethodWebRequest(
                "http://localhost:8080/refentry/path/to/garbage");
        WebResponse res4 = wc.getResponse(req4);
        assertEquals(title, res4.getTitle());

        // These garbage requests should return the search results page.
        WebRequest  req5 =
            new GetMethodWebRequest(
                "http://localhost:8080/refentry?SrchString");
        WebResponse res5 = wc.getResponse(req5);
        assertEquals("Search Results", res5.getTitle());

        WebRequest  req6 =
            new GetMethodWebRequest(
                "http://localhost:8080/refentry?SrchString=");
        WebResponse res6 = wc.getResponse(req6);
        assertEquals("Search Results", res6.getTitle());

        WebRequest  req7 =
            new GetMethodWebRequest(
                "http://localhost:8080/refentry?RefEntry");
        WebResponse res7 = wc.getResponse(req7);
        assertEquals("Search Results", res7.getTitle());

        WebRequest  req8 =
            new GetMethodWebRequest(
                "http://localhost:8080/refentry?RefEntry=");
        WebResponse res8 = wc.getResponse(req8);
        assertEquals("Search Results", res8.getTitle());

        WebRequest  req9 =
            new GetMethodWebRequest(
                "http://localhost:8080/refentry?RefEntry=garbage");
        WebResponse res9 = wc.getResponse(req9);
        assertEquals("Search Results", res9.getTitle());
    }
}
```

Test for Okay Requests

```
import com.meterware.httpunit.GetMethodWebRequest;
import com.meterware.httpunit.WebConversation;
import com.meterware.httpunit.WebRequest;
import com.meterware.httpunit.WebResponse;
import junit.framework.TestCase;
import junit.framework.TestSuite;

/**
 * HttpUnit mechanism to test that the servlet manages to handle
 * a couple of correctly formed requests for the default servlet.
 * This mechanism only works when the servlet is running in the
 * container and is reachable under
 * http://localhost:8080/refentry
 * <p>The test is then run by hand. Ideally, it would be integrated
 * into an automated framework such as Cactus.
```

```
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
public class TestOkayRequests extends TestCase {
    public static void main(String[] args) {
        junit.textui.TestRunner.run(suite());
    }
    public static TestSuite suite() {
        return new TestSuite(TestOkayRequests.class);
    }
    public TestOkayRequests(String str) {
        super(str);
    }
    public void testOkayRequests() throws Exception {
        WebConversation wc = new WebConversation();

        WebRequest  request  =
            new GetMethodWebRequest(
                "http://localhost:8080/refentry");
        WebResponse response = wc.getResponse(request);
        String      title    = response.getTitle();
        assertEquals(title, "webman(1) - a RefEntry client viewer");

        WebRequest  request1 =
            new GetMethodWebRequest(
            "http://localhost:8080/refentry?SrchString=refentry");
        WebResponse respons1 = wc.getResponse(request1);
        assertEquals(respons1.getTitle(), "Search Results");
    }
}
```

Test Client

```
import com.meterware.httpunit.GetMethodWebRequest;
import com.meterware.httpunit.WebConversation;
import com.meterware.httpunit.WebRequest;
import com.meterware.httpunit.WebResponse;
import java.util.Date;

/**
 * Threaded client to test that the servlet manages to handle
 * multiple "simultaneous" requests for the default servlet.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
public class TestClient extends Thread {
    String          URL;
    int             Nbr;
    WebConversation WC;
    WebRequest      Req;
    WebResponse     Res;
    TestClient(String url, int nbr) {
        URL = url;
        Nbr = nbr;
        WC  = new WebConversation();
    }
    public void run() {
        try {
     Date now = new Date();
            System.out.println(Nbr + " started : " + now.getTime());
            Req = new GetMethodWebRequest(URL);
            Res = WC.getResponse(this.Req);
            System.out.println(Nbr + " page    : " + Res.getTitle());
     Date later = new Date();
            System.out.println(Nbr + " finished: " + later.getTime());
            this.finalize();
        } catch (Throwable e) {
            System.out.println(e.toString());
        }
    }
}
```

Test Threader

```
import com.meterware.httpunit.GetMethodWebRequest;
import com.meterware.httpunit.WebConversation;
import com.meterware.httpunit.WebRequest;
import com.meterware.httpunit.WebResponse;
import junit.framework.TestCase;
import junit.framework.TestSuite;

/**
 * HttpUnit mechanism to test that the servlet manages to handle
 * multiple "simultaneous" requests for the default servlet.
 * This mechanism only works when the servlet is running in the
 * container and is reachable under
 * http://localhost:8080/refentry
 * <p>The test is then run by hand. Ideally, it would be integrated
 * into an automated framework such as Cactus.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
public class TestThreads extends TestCase {
    public static void main(String[] args) {
        junit.textui.TestRunner.run(suite());
    }
    public static TestSuite suite() {
        return new TestSuite(TestThreads.class);
    }
    public TestThreads(String str) {
        super(str);
    }
    public void testThreads() throws Exception {
        String URL1 = new String(
            "http://localhost:8080/refentry?RefEntry="
   + "RefEntry-3");
        String URL2 = new String(
            "http://localhost:8080/refentry?SrchString="
          + "parent");
        String URL3 = new String(
            "http://localhost:8080/refentry?SrchString="
          + "garbage+in+garbage+out");

        System.out.println("\nTestThreads timing");
        System.out.println("------------------");
        TestClient[] clients = new TestClient[100];
        for (int i = 0; i < 100; i++) {
            String url;
            if ((i % 3) == 0) url = new String(URL3);
            else if ((i % 2) == 0) url = new String(URL2);
            else url = new String(URL1);
            clients[i] = new TestClient(url, i);

     clients[i].start();
        }
    }
}
```

Unit Test for RefEntryBackEnd

```
package org.mcraig.cs445.refentry;
import java.io.*;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Properties;
import junit.framework.TestCase;

/**
 * Test friendly methods of the RefEntryBackEnd class using the
 * JUnit test framework.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
public class TestRefEntryBackEnd extends TestCase {
    protected RefEntryBackEnd backend;
    protected File            infile;
    protected File            outfile;
    protected Properties      xslt;
    protected RefEntry        refentry;
```

```
    protected void setUp() throws Exception {
        try {
            backend  = new RefEntryBackEnd(new HashSet());
            infile   = new File("src/test/test.xml");
            outfile  = File.createTempFile("test", ".html");
            xslt     = new Properties();
            xslt.load(
                new BufferedInputStream(
                    new FileInputStream("src/conf/xslt.props")));
            refentry = new RefEntry(infile, outfile, xslt);
        } catch (Exception e) {
            throw e;
        }
    }
    protected void tearDown() throws Exception {
        try {
            outfile.delete();
        } catch (Exception e) {
            throw e;
        }
    }
    public void testAdd() {
        boolean result = backend.add(refentry);
        assertTrue(result);
    }
    public void testGetEntry() {
        HashSet  test = backend.getEntry(refentry.getID());
        Iterator iter = test.iterator();
        RefEntry re   = (RefEntry)iter.next();
        assertEquals(re, refentry);
    }
    public void testGetValues() {
        HashSet keys  = new HashSet();
        keys.add("test");
        HashSet vals = backend.getValues(keys);
        assertTrue(vals.size() == 1);
        Iterator iter = vals.iterator();
        RefEntry re   = (RefEntry)iter.next();
        assertEquals(re, refentry);
    }
    public void runTest() {
        testAdd();
        testGetEntry();
        testGetValues();
    }
}
```

Unit Test for RefEntryIndex

```
package org.mcraig.cs445.refentry;
import java.io.*;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Properties;
import junit.framework.TestCase;

/**
 * Test friendly methods of the RefEntryIndex class using the
 * JUnit test framework.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
public class TestRefEntryIndex extends TestCase {
    protected RefEntryIndex index;
    protected File          infile;
    protected File          outfile;
    protected Properties    xslt;
    protected RefEntry      refentry;
    protected void setUp() throws Exception {
        try {
            index    = new RefEntryIndex();
            infile   = new File("src/test/test.xml");
            outfile  = File.createTempFile("test", ".html");
            xslt     = new Properties();
            xslt.load(
                new BufferedInputStream(
```

```
                    new FileInputStream("src/conf/xslt.props")));
            refentry = new RefEntry(infile, outfile, xslt);
        } catch (Exception e) {
            throw e;
        }
    }
    protected void tearDown() throws Exception {
        try {
            outfile.delete();
        } catch (Exception e) {
            throw e;
        }
    }
    public void testAdd() {
        boolean test = index.add("Test", refentry);
        assertTrue(test);
    }
    public void testMatch() {
        HashSet test = index.match("tesT");
        Iterator iter = test.iterator();
        RefEntry re   = (RefEntry)iter.next();
        assertEquals(re, refentry);
    }
    public void runTest() {
        testAdd();
        testMatch();
    }
}
```

Unit Test for RefEntry

```
package org.mcraig.cs445.refentry;
import java.io.*;
import java.util.HashSet;
import java.util.Properties;
import junit.framework.TestCase;

/**
 * Test friendly and public methods of the RefEntry class using the
 * JUnit test framework.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
public class TestRefEntry extends TestCase {
    protected File       infile;
    protected File       outfile;
    protected Properties xslt;
    protected RefEntry   refentry;
    protected void setUp() throws Exception {
        try {
            infile   = new File("src/test/test.xml");
            outfile  = File.createTempFile("test", ".html");
            xslt     = new Properties();
            xslt.load(
                new BufferedInputStream(
                    new FileInputStream("src/conf/xslt.props")));
            refentry = new RefEntry(infile, outfile, xslt);
        } catch (Exception e) {
            throw e;
        }
    }
    protected void tearDown() throws Exception {
        try {
            outfile.delete();
        } catch (Exception e) {
            throw e;
        }
    }
    public void testCompareTo() {
        assertTrue(refentry.compareTo(refentry) == 0);
    }
    public void testGetID() {
        String id = new String(refentry.getID());
        assertEquals("test-test", id);
    }
    public void testGetKeys() {
```

```
        HashSet keys = new HashSet();
        keys.add(new String("test"));
        keys.add(new String("refentry"));
        keys.add(new String("servlet"));
        keys.add(new String("code"));
        keys.add(new String("synopsis"));
        keys.add(new String("refsect1"));
        keys.add(new String("reference"));
        keys.add(new String("section"));

        HashSet gotkeys = refentry.getKeys(new HashSet());
        assertTrue(gotkeys.size() == keys.size());
    }
    public void testGetManVolNum() {
        String mvn = new String(refentry.getManVolNum());
        assertEquals("test", mvn);
    }
    public void testGetRefName() {
        String rn = new String(refentry.getRefName());
        assertEquals("test", rn);
    }
    public void testGetRefPurpose() {
        String rp = new String(refentry.getRefPurpose());
        assertEquals("test RefEntry servlet code", rp);
    }
    public void testGetXHTML() {
        String test;
        try {
            BufferedReader br   =
                new BufferedReader(
                    new FileReader(
                        new File("src/test/test.div")));
            StringWriter   sr   = new StringWriter();
            BufferedWriter bw   = new BufferedWriter(sr);
            String         line = new String("");
            while ((line = br.readLine()) != null) {
                bw.write(line);
                bw.newLine();
            }
            br.close();
            bw.close();
            sr.close();
            test = sr.toString();
            assertEquals(test, refentry.getXHTML());
        } catch (Exception e) {
            fail(e.toString());
        }
    }
    public void runTest() {
        testCompareTo();
        testGetID();
        testGetKeys();
        testGetManVolNum();
        testGetRefName();
        testGetRefPurpose();
        testGetXHTML();
    }
}
```

Unit Test for RefEntryServlet

```
package org.mcraig.cs445.refentry;
import java.io.*;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Properties;
import junit.framework.TestCase;

/**
 * Test public methods of the RefEntryServlet class using the
 * JUnit test framework.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
public class TestRefEntryServlet extends TestCase {
    protected RefEntryServlet servlet = new RefEntryServlet();
```

```
    protected Properties      config;
    protected RefEntryBackEnd backend;
    protected File            infile;
    protected File            outfile;
    protected Properties      xslt;
    protected RefEntry        refentry;
    protected void setUp() throws Exception {
        try {
            backend  = new RefEntryBackEnd(new HashSet());
            infile   = new File("src/test/test.xml");
            outfile  = File.createTempFile("test", ".html");
            xslt     = new Properties();
            xslt.load(new BufferedInputStream(
                new FileInputStream("src/conf/xslt.props")));
            refentry = new RefEntry(infile, outfile, xslt);

            backend.add(refentry);
            ObjectOutputStream os =
                new ObjectOutputStream(
                    new FileOutputStream(
                        new File("be")));
            os.writeObject(backend);
            os.close();

            config   = new Properties();
            config.setProperty("CommonWords", "the and you");
            config.setProperty("DefaultPage", "test-test");
            config.setProperty("test-test", "src/test/test.xml");
        } catch (Exception e) {
            throw e;
        }
    }
    protected void tearDown() throws Exception {
        try {
            outfile.delete();
            File backend = new File("be");
            backend.delete();
        } catch (Exception e) {
            throw e;
        }
    }
    public void testGetServletInfo() {
        assertTrue(servlet.getServletInfo() != null);
    }
    public void testInit() {
        // If the init method breaks, the servlet
 // cannot start, so other tests implicitly
 // get at this one.
        return;
    }
    public void testDoGet() {
        // Test this with ServletUnit.
        return;
    }
    public void testDoPost() {
        // Test this with ServletUnit.
        return;
    }
    public void testGetTokens() {
        // Test this with ServletUnit.
        return;
    }
    public void runTest() {
        testGetServletInfo();
        testInit();
        testDoGet();
        testDoPost();
        testGetTokens();
    }
}
```

Unit Test for RefEntryTransformer

```
package org.mcraig.cs445.refentry;
import java.io.*;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Properties;
import junit.framework.TestCase;

/**
 * Test friendly methods of the RefEntryTransformer class using the
 * JUnit test framework.
 * @author <a href="mailto:mark@mcraig.org">Mark Craig</a>
 */
public class TestRefEntryTransformer extends TestCase {
    protected RefEntryTransformer transformer;
    protected File                infile;
    protected File                outfile;
    protected Properties          xslt;
    protected RefEntry            refentry;
    protected void setUp() throws Exception {
        try {
            transformer = new RefEntryTransformer("TestCase");
            infile      = new File("src/test/test.xml");
            outfile     = File.createTempFile("test", ".html");
            xslt        = new Properties();
            xslt.load(new BufferedInputStream(
                new FileInputStream("src/conf/xslt.props")));
            refentry    = new RefEntry(infile, outfile, xslt);
        } catch (Exception e) {
            throw e;
        }
    }
    protected void tearDown() throws Exception {
        try {
            outfile.delete();
        } catch (Exception e) {
            throw e;
        }
    }
    public void testTransformSet() {
        HashSet set = new HashSet();
        set.add(refentry);
        String output = transformer.transform(set, "test");
        output        = output.concat("\n");
        String str    = new String();
        try {
            BufferedReader br  =
                new BufferedReader(new FileReader("src/test/test.xhtml"));
            String         tmp = new String();
            while ((tmp = br.readLine()) != null)
                str = str.concat(tmp + "\n");
            br.close();

            PrintWriter pr =
                new PrintWriter(
                    new FileWriter(
                        File.createTempFile("out", "xhtml")));
            pr.write(output);
            pr.close();
        } catch (Exception e) {
            fail(e.toString());
        }
        assertTrue(output.equals(str));
    }
    public void testTransformOne() {
        String output = transformer.transform(refentry, "test");
        // TODO: check that it matches certain output
    }
    public void testTransformEmpty() {
        // TODO: check that it matches certain output
    }
    public void runTest() {
        // The problem with testing these is that they contain the
        // date each time a page is generated, so diffs do not work.
        //testTransformSet();
        testTransformOne();
        testTransformEmpty();
    }
```

```
}
```

Unit Test for RefEntryValidator

```
package org.mcraig.cs445.refentry;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.PrintStream;
import org.xml.sax.InputSource;
import junit.framework.TestCase;

/**
 * Test RefEntryValidator using the JUnit test framework.
 * @author <a href=mailto:mark@mcraig.org>Mark Craig</a>
 */
public class TestRefEntryValidator extends TestCase {
    protected InputSource validInput;
    protected InputSource invalidInput;
    protected File        testOut;
    protected void setUp() throws Exception {
        try {
            File validFile   = new File("src/test/test.xml");
            File invalidFile = new File("src/test/invalid.xml");

            validInput   = new InputSource(new FileReader(validFile));
            invalidInput = new InputSource(new FileReader(invalidFile));

            testOut = File.createTempFile("testout", "txt");
        } catch (Exception e) {
            throw e;
        }
    }
    public void tearDown() throws Exception {
        try {
            testOut.delete();
        } catch (Exception e) {
            throw e;
        }
    }
    public void testIsValid() {
        boolean isValid   = RefEntryValidator.isValid(validInput);
        assertTrue(isValid);
    }
    public void testIsInvalid() {
        // Set isValid to true. We expect the invalid input to change
        // it to false upon validation.
        boolean isValid = true;
        try {
            PrintStream out =
                new PrintStream(new FileOutputStream(testOut));
            System.setErr(out);
            System.setOut(out);
            isValid = RefEntryValidator.isValid(invalidInput);
        } catch (Exception se) {
            assertTrue(
                se.getClass().getName().equals("org.xml.sax.SAXException"));
            assertFalse(isValid);
        }
    }
    public void runTest() {
        testIsValid();
        testIsInvalid();
    }
}
```

Web Application Deployment Descriptor

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app
  PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
  "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <display-name>Basic RefEntry Servlet</display-name>
  <description>Serves up RefEntry documents.</description>
  <servlet>
    <servlet-name>RefEntry</servlet-name>
    <servlet-class>org.mcraig.cs445.refentry.RefEntryServlet</servlet-class>
    <init-param>
      <param-name>serializedBackend</param-name>
      <param-value>backend</param-value>
    </init-param>
    <init-param>
      <param-name>serializedCommonWords</param-name>
      <param-value>commonWords</param-value>
    </init-param>
    <init-param>
      <param-name>defaultPage</param-name>
      <param-value>webman-1</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>RefEntry</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
  <mime-mapping>
    <extension>pdf</extension>
    <mime-type>application/pdf</mime-type>
  </mime-mapping>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
</web-app>
```

Bibliography

Books

Bruce Eckel, *Thinking in Java™*, Prentice-Hall, Inc., 2000, Second edition.

Eric M. Burke, *Java™ and XSLT*, O'Reilly & Associates, Inc., 2001, First edition.

Michael Kay, *XSLT Programmer's Reference*, Wrox Press Ltd., 2000, First edition.

Norman Walsh and Leonard Mueller, *DocBook: The Definitive Guide*, O'Reilly & Associates, Inc., 1999, First edition.

Index

S

T

U

W

X

Colophon

The source of this document conforms to SGML DocBook 4.1. Red Hat Linux 7.2 comes with formatting tools for DocBook, enabling generation of target documents in PostScript, PDF, RTF and HTML for example. Many thanks to those who provide DocBook and the tools needed to generate output formats, in particular Norman Walsh.