Building an Open Source J2EE Weblogger

David M. Johnson

April 17, 2002

Originally published here: http://onjava.com/onjava/2002/04/17/wblogosj2ee.html

As a Java developer, you should be aware of the tremendous wealth of open source development software that is available for your use -- even if you have no desire to release any of your own software as open source. In this article, I will introduce you to some of the most useful open source Java development tools by showing you how I used these tools to develop a complete database-driven Web application called Roller.

Roller fits into the relatively new category of software called webloggers: applications that make it easy for you to maintain a weblog, also known as a *blog* -- a public diary where you link to recent reading on the Web and comment on items of interest to you.

The Roller Web application allows you to maintain a Web site that consists of a weblog, an organized collection of favorite Web bookmarks, and a collection of favorite news feeds. You can define Web pages to display your weblog, bookmarks, and news feeds. By editing the HTML templates that define these pages, you have almost total control over the layout and appearance of these pages. Most importantly, you can do all of this without leaving the Roller Web application -- no programming is required.

I used over a dozen open source development tools to develop Roller, the most useful of which are listed in Table 1; however, this article focuses on just four tools: the XDoclet code generator, the Castor persistence framework, the Struts Servlet/JSP framework, and the Velocity code-generation engine. In this article I will describe the Roller application, its architecture, and specifically how I used XDoclet, Castor, Struts, and Velocity in its development.

Table 1: Open source tools used in Roller Development

Name	**Description**	**Developer**	**Type of License***
Castor	Persistence framework	Exolab	Similar to BSD license
HSQL	Small but powerful Java database	Thomas Meuller	Similar to BSD license
Jakarta Ant	XML-driven Java build system	Apache	Apache Public License
Jakarta Commons	Collections, utilities	Apache	Apache Public License

Jakarta Struts	Servlet/JSP framework	Apache	Apache Public License
Jakarta Tomcat	Servlet/JSP Server	Apache	Apache Public License
Jakarta Velocity	Template-driven code generator	Apache	Apache Public License
Netbeans	Integrated Dev. Environment		Sun Public License
Xerces	XML parser	Apache	Apache Public License
XDoclet	Code generator	Dreambean	Similar to MIT License

* For more information on open source licenses see *opensource.org*

The Roller Application

Roller does not support all of the features of commercial weblogging software (such as Userland's Radio or Pyra Labs' Blogger products), but Roller does support what I consider the essential weblogging features. With Roller you can:

- **Maintain a weblog, with user-defined categories.** You can write new weblog entries and edit entries that have already been posted. You can define a set of weblog categories and can assign weblog entries to different categories. This allows you to maintain several different weblogs, each covering a different topic.
- **Publish your weblog as an RSS news feed.** Roller makes your weblog available as a standard Rich Site Summary (RSS) news feed so that readers can subscribe to and read your weblog without visiting your Roller site.
- **Maintain a collection of favorite bookmarks, organized by bookmark folders.** You can define new bookmark folders and can add, delete, and edit the bookmarks within these folders. You can then display these bookmarks on one or more of your Roller site's pages. This allows you to do *blogrolling* -- displaying links to your favorite weblogs.
- **Maintain a collection of favorite RSS news feeds.** This allows you to display headlines with links to news stories from your favorite news sources or weblogs.
- **Define a set of Web pages to display your weblog, bookmarks, and news feeds.** Pages are defined using HTML templates with embedded macros for each type of data. For example, there is a `$Bookmarks` macro that will draw a portion of your bookmark collection on a Web page and a `$WeblogCalendar` macro that will draw a calendar view of your past weblog entries. These templates allow you almost complete control over the layout and look-and-feel of your Web pages.

There are two types of Roller users: readers and editors. Readers are simply anonymous visitors to the Roller Web site. Editors have user accounts and must log in by providing a user name and password. Editors have the ability to edit their weblog entries, bookmarks, newsfeeds, and page templates.

Figure 1 illustrates the Roller application by showing the Roller Web page navigation tree. The

boxes represent Web pages and the arrows represent links between pages. The gray pages are the public pages that any visitor may access, the yellow pages are the login pages, and the red pages are the pages that only editors can access.

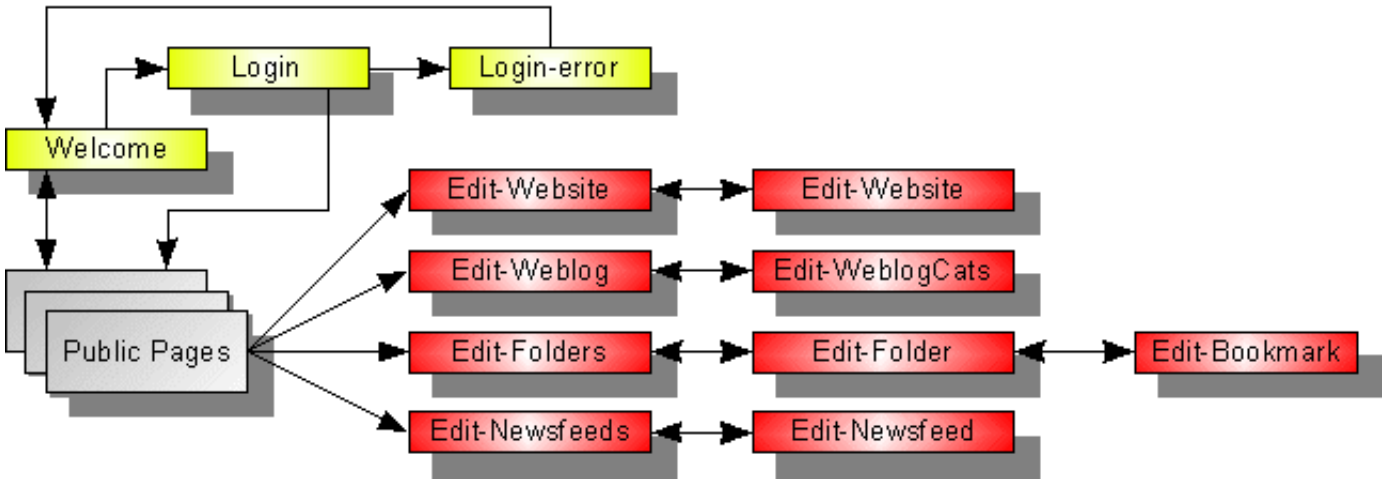

Figure 1: Roller Web Pages

Roller Architecture

Internally, Roller is divided into a presentation tier and a business tier, as recommended in Sun's J2EE Pattern Catalog. The presentation tier is responsible for Roller's user interface, and the business tier is responsible for Roller's application logic and the persistence of application data. Figure 2 provides an overview of the Roller architecture.

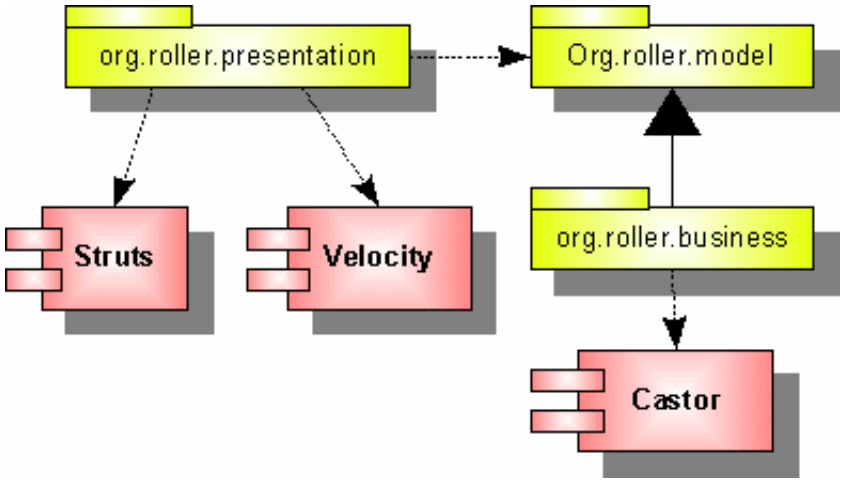

Figure 2: Roller Architecture

The presentation tier is implemented using the Model-View-Controller (MVC) pattern and the Struts MVC framework. The Model is an abstraction of the application logic and application data and is represented by a set of interfaces defined in the `org.roller.model` package. The View is implemented using Servlets, JSP pages, and Velocity page templates. The Controller is Struts, which is responsible for receiving incoming requests and dispatching them to the View. The implementation of the presentation tier is further discussed in the sections on Struts and Velocity.

The business tier implements the interfaces in the `org.roller.model` package, using the Castor

JDO persistence framework. The business tier exchanges data with the presentation tier in the form of simple, lightweight JavaBeans known as Value Objects. Value Objects are yet another of the Sun J2EE patterns. Each Value Object maps to a table in the Roller database.

Figure 3 shows the Roller Value Objects, their properties, and the relationships between them. Each editor is represented by a User object. Each User has a Website object, which represents the editor's Web site and which has weblog entries, bookmark folders, newsfeeds, and page templates. The Website object also specifies the default page template of the Web site and which page template is used for rendering a day of weblog entries.

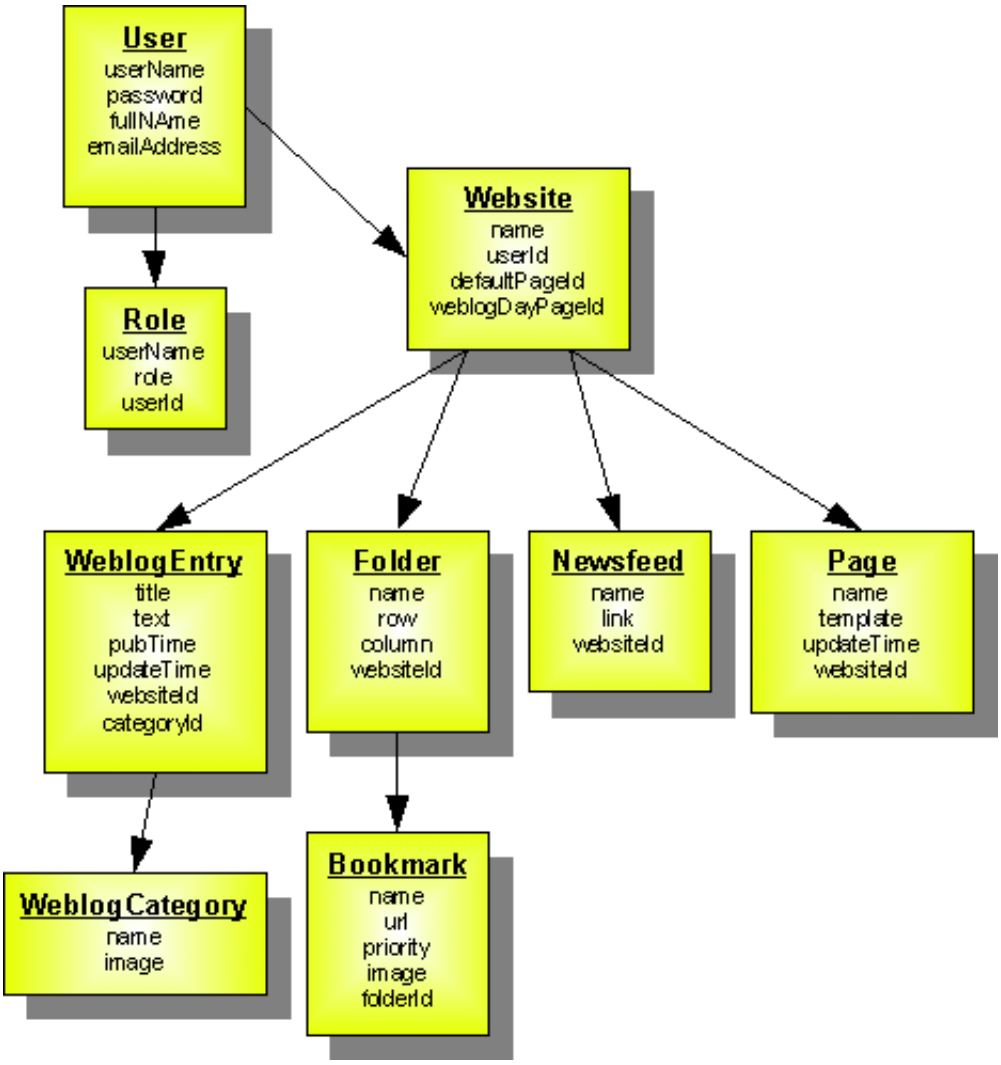

Figure 3: Roller Value Objects

The business tier uses Castor JDO to store and retrieve Value Objects to and from a JDBC-accessible database. Castor JDO is part of the larger Castor data-binding framework, which according to the Castor Web site is "the shortest path between Java objects, XML documents, SQL databases, and LDAP."

As a persistence framework, Castor JDO is similar to commercial object-relational mappers such as TopLink and Cocobase. Castor JDO fulfills a role similar to that of Sun's Java Data Objects, but Castor JDO is not an implementation of Sun's JDO specification (JSR-000012). Castor JDO allows you to define a mapping between Java classes and tables in a relational

database. You can then issue queries using Castor's own Object Query Language (OQL) and receive the results as collections of Java objects.

Before you can use Castor JDO, you must provide a mapping file -- an XML file that maps each class to a database table and each class property to a field within a database table. Below is a portion of Roller's mapping file.

```
<mapping>
<class name=org.roller.model.BookmarkData" identity="id"
    access="shared" key-generator="UUID" auto-complete="false">
    <map-to table="bookmark"/>
    <cache-type type="count-limited"/>
    <field name="folderId" type="java.lang.String"></field>
    <field name="id" type="java.lang.String"></field>
    <field name="image" type="java.lang.String"></field>
    <field name="name" type="java.lang.String"></field>
    <field name="priority" type="java.lang.Integer"></field>
    <field name="url" type="java.lang.String"></field>
</class>
...
</mapping>
```

Once you provide Castor with a mapping file, retrieving a collection of objects from the database can be as simple as the code snippet shown below:

```
// Construct a new query and bind its parameters
String query = "SELECT p FROM BookmarkData p WHERE websiteId=$";
OQLQuery oql = db.getOQLQuery( query );
oql.bind( websiteId );

// Retrieve results and print each one
QueryResults results = oql.execute();
while ( results.hasMore() ) {
   BookmarkData bookmark = (BookmarkData)results.next();
   System.out.println( bookmark.toString() );
}
```

XDoclet

XDoclet is a code generator that is implemented as a Javadoc extension, a *Doclet*. To use XDoclet, you place special Javadoc tags in your Java source code. Based on these tags, XDoclet can generate additional Java code that supports your classes, mapping files that map your classes to database tables, and deployment descriptors that assist in deploying your classes.

XDoclet started out its life as EJBDoclet, a tool that allows you to implement an Enterprise JavaBean by writing just one source code file. Now, the XDoclet product includes two Doclets: EJBDoclet and WebDoclet. EJBDoclet is for generating EJB classes, value objects, and database mappings. WebDoclet is for generating all sorts of Servlet Web Application deployment descriptors, including `web.xml` files, Tag Library Descriptors, and Struts configuration files.

The Roller build process uses both EJBDoclet and WebDoclet, as shown in Figure 4. In Step 1, EJBDoclet is used to process a set of abstract classes of type `javax.ejb.EntityBean` -- one for each one of the Roller Value Objects. From these classes, EJBDoclet generates a Castor mapping file, the Roller Value Object classes, and a set of corresponding Struts form classes. In Step 2, WebDoclet is used to process a source directory that contains JSP tags, Servlet classes, and Struts classes. The output of the WebDoclet is the complete set of Roller Web Application deployment descriptors.

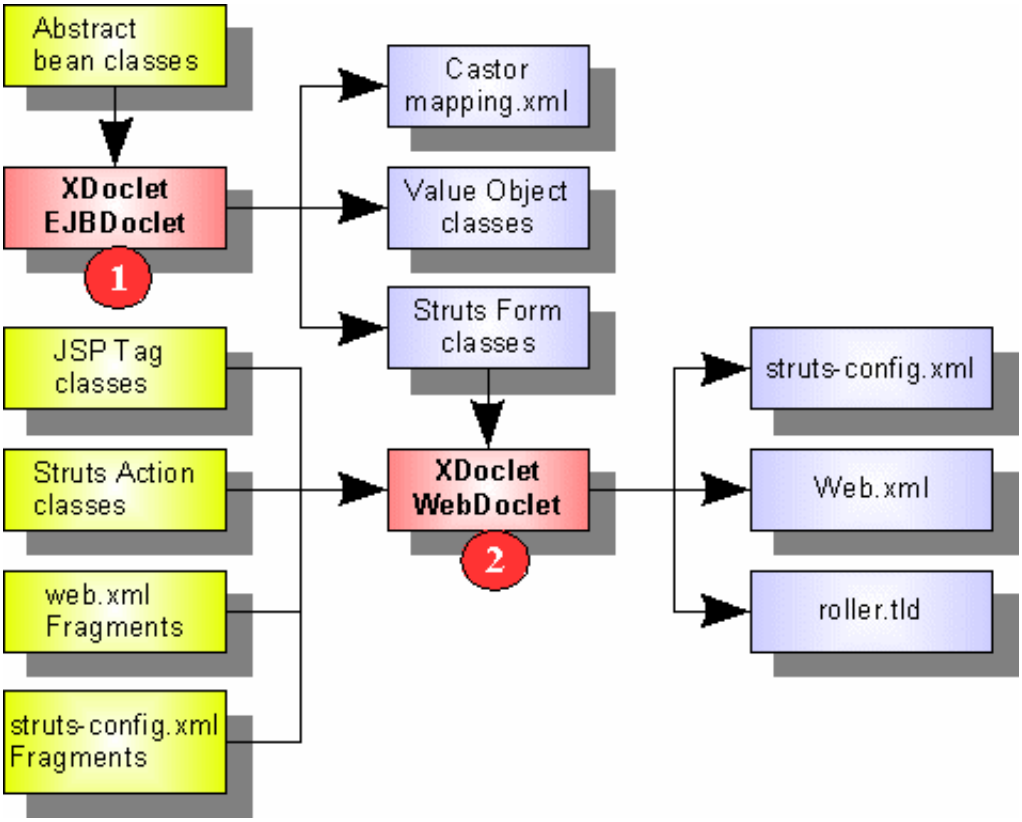

Figure 4: XDoclet and the Roller Build Process

Below is a simple example bean that shows the EJBDoclet tags necessary to create a Value Object. The `@castor` tags provide the information needed to generate the Castor mapping entries for the bean. The `@ejb` tags provide the information needed to generate the Value Object and a complete EJB entity bean (which Roller does not use).

```
/**
 * Represents a single URL in a user's favorite web-bookmarks collection.
 * @ejb:bean name="Bookmark" type="CMP" jndi-name="roller/Bookmark"
 * @ejb:data-object extends="org.roller.model.ValueObject"
 * @struts:form
 * @castor:class name="bookmark" table="bookmark" xml="bookmark"
 *               id="id" key-generator="UUID"
 */

public abstract class BookmarkBean implements EntityBean
{

  /** @ejb:interface-method
   *  @ejb:transaction type="Required" */
   public abstract void setData(org.roller.model.BookmarkData dataHolder);
```

```
  /** @ejb:interface-method */
   public abstract org.roller.model.BookmarkData getData();

  /** @castor:field set-method="setId"
   *  @castor:field-xml node="attribute"
   *  @castor:field-sql name="id" sql-dirty="check" dirty="true"
   *  @ejb:interface-method
   *  @ejb:pk-field
   *  @ejb:persistent-field */
    public abstract String getId();

  /** @ejb:pk-field
   *  @ejb:persistent-field */
    public abstract void setId( String value );
    ...
}
```

Struts

The Roller presentation tier is implemented using Struts and Velocity. Struts is a Servlet application framework that is based on the MVC pattern. In a typical Struts application, the Model is a set of JavaBeans that hold the data to be presented in the View; the View is a set of JSP pages that render HTML; and the Controller is a Servlet and set of action classes that are registered to handle incoming requests.

Roller's Edit-Bookmark form provides a nice, simple example of how Struts works. There are four parts to the Edit-Bookmark form implementation: the `edit-bookmark.jsp` page, the `BookmarkForm` JavaBean class, the `BookmarkFormAction` action handler, and some entries in Roller's `struts-config.xml` file that tie the first three items together. So, let's introduce the players:

- The `edit-bookmark.jsp` page looks just like an HTML page, except that it uses the Struts HTML form tags instead of standard HTML form tags. The Struts HTML form tags know how to find the `BookmarkForm` JavaBean and how to use its properties to populate the form with data.

- The `BookmarkForm` class is a dumb JavaBean that just holds data -- it has the exact same properties as the Bookmark Value Object. As you may recall, the `BookmarkForm` class and all of its sibling form classes are generated by XDoclet. In Struts, form classes must extend `org.apache.struts.action.ActionForm`.

- The `BookmarkFormAction` is essentially an action handler. It is registered (in the `struts-config.xml` file) to handle incoming requests that include the pattern `/bookmark.do`. In Struts, action classes must extend `org.apache.struts.action.Action`.

Figure 5 shows the sequence of events that occurs when a request for the Edit-Bookmark form comes into the system. Roller needs to respond to this request by creating an HTML form populated with data for the bookmark that is to be edited.

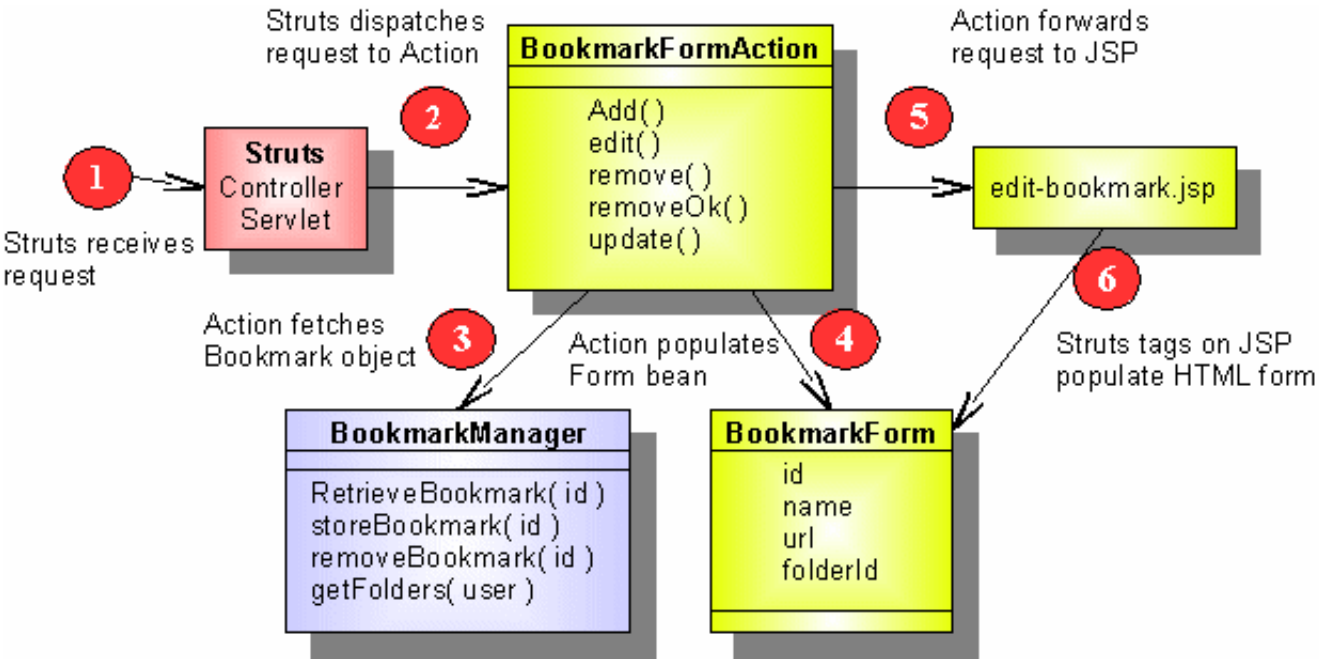

Figure 5: Incoming request for Edit-Bookmark page

Here are the steps in processing an incoming request for the Edit-Bookmark page:

1. The Struts Controller Servlet receives a request for the Edit-Bookmark action. The Controller uses the URI of the request to look up the `FormAction` that should handle the request.

2. The Struts Controller Servlet dispatches the request to the `BookmarkFormAction.edit()` method. Knowing that the user has requested the Edit-Bookmark page, the `BookmarkFormAction` looks for a request parameter that specifies the bookmark that is to be edited.

3. The `BookmarkFormAction` calls the `BookmarkManager` to retrieve the bookmark information that is to be edited.

4. The `BookmarkFormAction` creates the `BookmarkForm` bean and adds that bean to the request's attributes so that it can be accessed by the JSP page.

5. The `BookmarkFormAction` finally forwards the request to `edit-bookmark.jsp` so that the page may be rendered.

6. The Struts form tags on the `edit-bookmark.jsp` page reads data from the `BookmarkForm` bean and uses that data to populate the Edit-Bookmark form. After that, the HTML page is returned to the user's browser for display.

Figure 6 shows the sequence of events that occurs when the request that contains posted data from the Edit-Bookmark page comes into the system. Roller needs to take the incoming form data and use it to update the bookmark that is stored in the data store managed by the business tier.

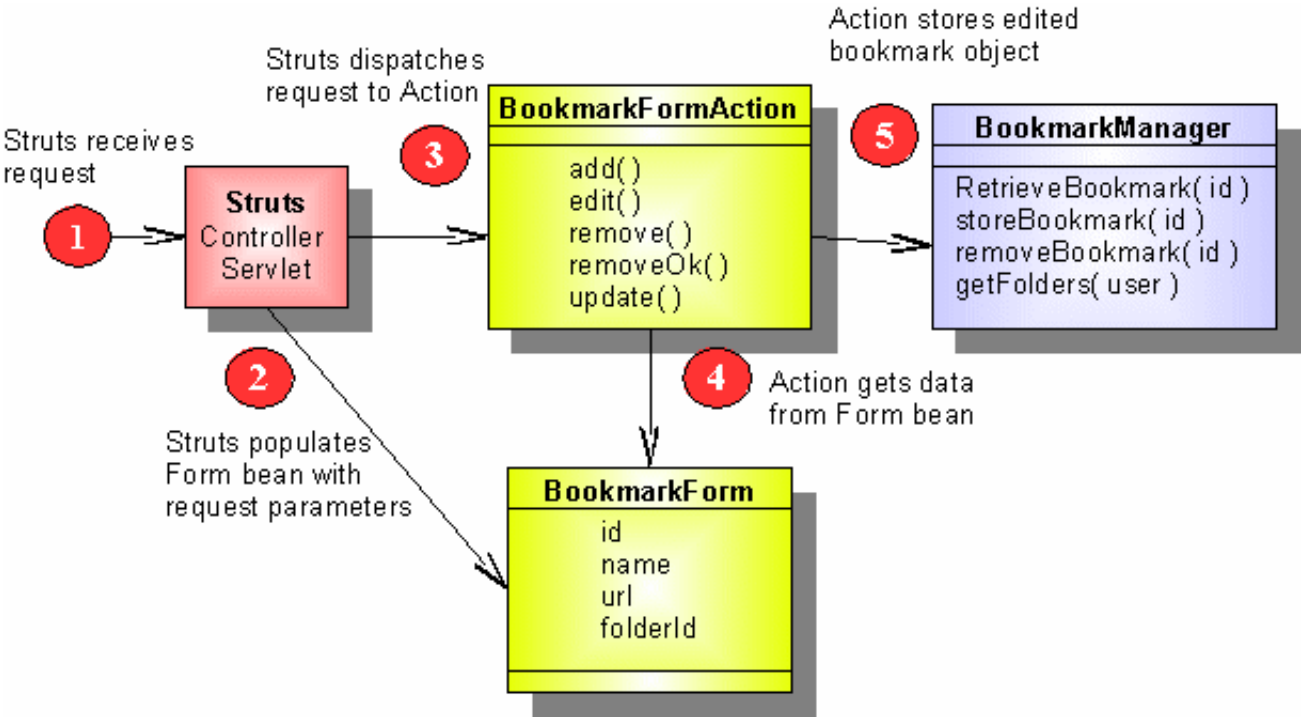

Figure 6: Request with data posted from Edit-Bookmark page

Here are the steps in processing a request with data from a posted Edit-Bookmark page:

1. The Struts Controller Servlet receives a request for the `Update-Bookmark` action. The Struts Controller determines which action should handle the request and which form bean should receive the data from the incoming form post.

2. The Struts Controller Servlet populates the `BookmarkForm` bean with data from the incoming request.

3. The Controller calls the `BookmarkFormAction` and passes in the form bean.

4. The `BookmarkFormAction` retrieves the data from the `BookmarkForm` bean.

5. The action calls upon the `BookmarkManager` to store the updated bookmark information.

Velocity

While JSP pages work well for the Roller editor pages, which rarely change, JSP does not work so well for the user pages. Weblog authors are not programmers, and they cannot be required to learn JSP and Java programming just to customize their weblog and associated Web pages. Furthermore, allowing Roller users to add new JSP pages, and thus new Java code, to the Roller application at runtime is a security risk.

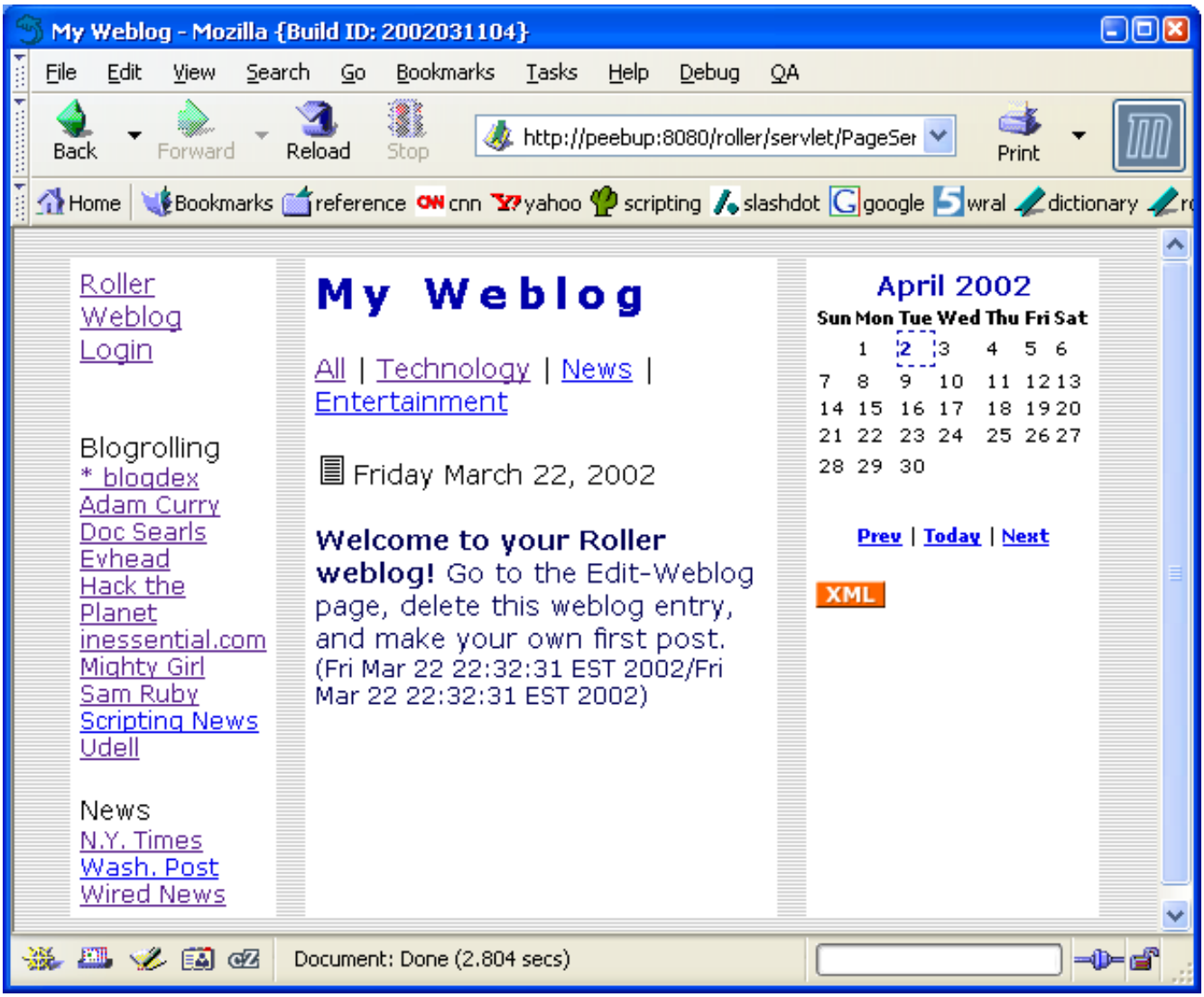

Figure 7: Velocity-generated public page

The best solution to the user pages problem is Velocity. Velocity is a general purpose template-based code-generation engine. That may sound complicated, but from the user's point of view, it is simple and easy-to-use. For example, the weblog page shown in Figure 7 is generated by a simple Velocity template. This template is shown below:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
;<head>
<title>$macros.showWebsiteTitle()</title>
<style type="text/css">$macros.includePage("_css")</style>
</head>
;<body>
<table cellpadding="5" cellspacing="15"
   border="0" align="center" width="95%">
   <tr>
   <td width="20%" valign="top" bgcolor="#ffffff">
      $macros.showNavBar(true)<br>
      $macros.showEditorNavBar(true)<br>
      $macros.showBookmarks("Blogrolling",true)<br>
      $macros.showBookmarks("News",true)
   </td>
   <td width="60%" valign="top" bgcolor="#ffffff">
```

```
      <h2>$macros.showWebsiteTitle()</h2>
      $macros.showWeblogCategoryChooser()<br>
      $macros.showWeblogEntries()
   </td>
   <td valign="top" bgcolor="#ffffff" width="20%">
      $macros.showWeblogCalendar()<br>
      $macros.showRSSBadge()
   </td>
   </tr>
</table>
</body>
</html>
```

The items that start with `$` are Velocity expressions, most of which result in calls to JSP tags that have been specially designed to work with Velocity. For example, the `$macros.showWeblogCategoryChooser()` expression results in the generation of the navigation bar at the top of the page -- the one that reads "All | Technology | News | Entertainment." The navigation bar is implemented in a custom JSP tag class named `org.roller.presentation.tags.NavigationTag`, which is also used in the JSP-based Roller editor pages.

Each user can define any number of pages, and since these pages are simply HTML pages, they can be customized using Front Page or any other HTML editor. The user just has to put the Velocity expressions in the right place. Below is a list of some of the Velocity expressions that are available for use in user-defined Roller Web pages.

Macro	**Emits HTML for:**
$macros.showNavBar()	Navigation bar, with a link to each one of the user's user-defined pages
$macros.showEditorNavBar()	Editor navigation bar, with links to the edit-bookmarks,edit-newsfeeds, edit-weblog, and edit-website pages
$macros.showBookmarks()	Entire bookmark collection in a multi-column table
$macros.showNewsfeeds()	Current headlines and story descriptions for the user's RSS newsfeeds
$macros.showWeblogEntries()	The most recent weblog entries
$macros.showWeblogCalendar()	A weblog calendar, with a link for each day on which there is a weblog entry

Conclusion

In this article, I have described four open source Java development tools and how these tools can be used together to develop a fairly sophisticated Web application. I hope I have given you a good idea of the power and flexibility of these tools.

Although I have not mentioned any problems with the open source tools that I have discussed, I did run into a number of bugs. I was able to find work-arounds and fixes for these bugs, but it was not always easy. I had to spend some time browsing mailing-lists, searching with Google, and, in one case, downloading the latest source for a product and building it myself. Formal technical support is not available for many open source tools, so keep in mind that you may have to solve your own problems.

In closing, I would like to thank the many developers and other contributors that made possible the open source Java development tools that I used in the development of Roller. The tools are great and they just keep getting better.

Resources

Weblogging

- The History of Weblogs
- Userland Radio
- Pyra Labs Blogger

Castor

- The Castor Project Homepage

Struts

- The Jakarta Struts Project
- "Strut your stuff with JSP tags" (JavaWorld article)

Velocity

- The Jakarta Velocity Project Homepage

XDoclet

- The XDoclet Project Homepage
- "Using XDoclet: Developing EJBs with Just the Bean Class" (onJava.com article)
- "Deciding Whether EJB is Appropriate" (OnJava.com article)

Return to ONJava.com.