CrownPeak Java Web Hosting

Version 0.20

Document History

Author/Editor	Date	Reason for Change	Version
Edwin Shelby	01/27/2014	Draft	0.10
Andrey Karpik	5/21/2014	Preview changes	0.20

Table of Contents

CrownPeak Java Web Hosting 1
Document History 2
CrownPeak Java Web Hosting Overview 4
Java Web Framework Overview 4
Java Web Framework Benefits 5
Integrating Java Web Framework in CrownPeak 5
CrownPeak Integration 7
Mechanisms for reusing JSP content in a JSP page. 7
Deploying Java Web Framework with the CrownPeak CMS 9
Implementing the Java Web Framework to CrownPeak CMS 9

CrownPeak Java Web Hosting Overview

Java Web applications are built from object oriented programs written in the Java language, originally developed by Sun Microsystems, and now owned by Oracle Corporation.

Java source files, having a .Java filename extension, are compiled into executable programs with a .class extension that can be deployed as a single library files (.JAR, .WAR or .EAR).

Java objects are executed by the Java Runtime Environment (JRE), which runs in the widest array of operating systems, from Windows to Mac OS X, and most Linux varieties. It continues the C language tradition of “write once, run everywhere”.

Java EE is a Web development platform that provides all functionality needed for complex internet applications. A client who has built a Web application based on Java Enterprise Edition (Java EE) can integrate the Java design with the CrownPeak CMS, and thus take full advantage of the CMS content authoring features.

At the minimum, a Java EE application will contain a servlet class (controller) and Java Server Pages (JSP). This is in fact the design used in the Moog corporate Websites and BNY Mellon marketing site. The JSPs are made up of HTML, Javascript,Java tags and inline Java code, but there are no Java business objects on those implementations, and data persistence is limited to XML files, Web service calls, and emails.

At the most complex, the Java platform will include Enterprise Java Beans (EJB), a complex container that manages objects in multi-server environments. It will likely also include data persistence capability (JDBC or JPA - Java object/relational mapping), as well as remote method calls (RMI), emails, messaging, Web services, connections, and other APIs. The goal of any Web development platform is to allow programmers to focus on business logic, reducing the need to manage the infrastructure components. To this end Java Application Server manages security, concurrency, transaction processing, and component deployment.

A Java Web page, like a .NET page, will combine a variety of different language elements to define the final page rendering. There will always be elements of HTML and CSS, combined with Javascript, JSON, JSP Standard Tag Lib (JSTL) tags, inline Java code fragments, XLST, and so on.

Java Web Framework Overview

With the wide implementation of Java EE technology, a number of Web design patterns have been implemented. The most common pattern is the model-view-controller pattern: model representing business data and logic, the view being the display of model data, and controller selecting the views and responding to user requests.

The earliest implementation of MVC in Java was Struts, based on XML tables for validation, navigation, and localization. It is implemented with an ActionServlet and an XML configuration file (controller layer), JSPs with Struts XML tags (view layer), and Java ActionForm classes (model layer).

This was followed by the Spring MVC framework from JBoss, which expands on it inversion-of-control technology to create Web applications without the complexity of the EJB environment. It was the first Web technology to use Java annotations, and it has the largest installed base of Java Web frameworks.

There are a number of less commonly used Java based Web frameworks, such as Google Web Toolkit (GWT), Play, Grails, Vaadin, etc.

The most recent Java Web framework is Java Server Faces (JSF), which has wide acceptance as the new standard for Web technology. It introduces the Facelet view definition, incorporating HTML tag extensions, templating, and view composites. The standard JSF 2 framework is further extended by a number of open source projects, such as RichFaces, PrimeFaces, MyFaces, etc.

Java Web Framework Benefits

Corporate Web sites often incorporate a large number of pages, containing a huge number of components. Managing such projects successfully demands a structured approach. Use of the MVC framework decouples the presentation, control, and data persistence functions. This provides a number of benefits:

- Application flexibility: changes to any component in one layer have a minimal impact on other components.
- Reduced complexity: a clear structure with well-defined responsibilities will minimize code duplication.
- Promotes division of labor: collaboration of developers with different skill sets results in a quicker and more stable development flow.

Integrating Java Web Framework in CrownPeak

Java Web applications are quite analogous in overall structure to ASP.NET Web forms or ASP.NET MVC. Both are composed in an integrated development environment (IDE), compiled, and exported to a well-defined file structure. In the case of Java, the application is deployed in a single library file, called a WAR or EAR file, which contains the defined folder structure.

When it comes to CrownPeak integration, Java applications will be treated in the same way as .NET applications:

- All server side programs for the model layer are exported from the IDE as executables, which then are imported into the CrownPeak CMS.
- The view layer is converted into CMS templates and models. Decisions about creating templates dynamic JSP pages, where variable output is involved, will depend on cost and benefits of converting an existing data source to CMS widgets.

Note: In the past, some JSP pages have been imported as a binary file and deployed without a template. This is not a best practice, since the only way to view the contents is in an IDE outside of CMS.

Capability	Web Forms	Java EE
Presentation	Web pages deployed as files with an .aspx extension.	Web pages are deployed with either a .jsp or an .xhtml extension (for JSF).

Server-side Logic	Code behind developed as supporting files with the same name as the pages they support with a .cs extension (for C#), and are deployed in a complied DLL.	Controllers and Models provide the server-side logic. The files are named with a .Java extension, and are compiled and deployed as .class files in the /bin folder.
		
		JavaServer Pages Standard Tag Library (JSTL) can be used for the server side manipulations.
		
		It encapsulates as simple tags the core functionality common to many Web applications. JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags. It also provides a framework for integrating existing custom tags with JSTL tags.
		
		`<c:forEach var="item" items="${sessionScope.cart.items}">`
		`...`
		`</c:forEach>`

Shared Global Content (Header/Footer)	Masterpages can be deployed containing all global content (header, footer), and deployed with a .master extension. The masterpages are referenced in the .aspx pages.	Layout files, which are Views, can be deployed with a .jsp or an .xhtml extension, and referenced by any view. `<jsp:include page="<%= assetHeader.GetLink().ToLower()%>"></jsp:include>` See below other include mechanism possible in JSP
Includes	User controls provide include capabilities, and are files deployed as pages with an .ascx exntesion, and referenced by any .aspx page.	Any view can be included within another view as a “Partial View”. As such, includes are deployed as standard .jsp or .xhtml files.
In-line code	C# code can be inserted in-line in any aspx page.	Java code can be inserted in a view page, Java Expression Language (EL) construction also can be used. `${header["host"]}`

CrownPeak Integration

Capability	Web Forms	Java EE
Output Template	Web Pages containing html, css, js, and any server-side C# code are implemented in output templates.	Views containing html, css, js, and any server-side Java code are implemented in output templates.
Input Template	Content placeholders are exposed as editable input form fields.	Content placeholders are exposed as editable input form fields.

Mechanisms for reusing JSP content in a JSP page.

http://stackoverflow.com/questions/14580120/whats-the-difference-between-including-files-with-jsp-include-directive-jsp-in

There are several mechanisms for reusing JSP content in a JSP page.

The following four mechanisms to include content in JSP can be categorized as direct reuse:

The include directive:

```
<%@ include file="header.html" %>
```

Static: adds the content from the value of the file attribute to the current page at translation time. The directive was originally intended for static layout templates, like HTML headers.

The <jsp:include> standard action

```
<jsp:include page="header.jsp" />
```

Dynamic: adds the content from the value of the page attribute to the current page at request time. Was intended more for dynamic content coming from JSPs.

The <c:import> JSTL tag:

```
<c:import url="http://www.example.com/foo/bar.html" />
```

Dynamic: adds the content from the value of the URL attribute to the current page, at request time. It works a lot like `<jsp:include>`, but it's more powerful and flexible: unlike the other two includes, the `<c:import>` url can be from outside the Web Container!

Preludes and codas

Another way to do a static include. Preludes and codas can be applied only to the beginnings and ends of pages. See the example here: **Defining implicit includes**

A tag file is an indirect method of content reuse, the way of encapsulating reusable content. A tag file is a source file that contains a fragment of JSP code that is reusable as a custom tag.

The purpose of includes and tag files is different.

A tag file (a concept introduced with JSP 2.0) is one of the options for creating custom tags. It's a faster and easier way to build custom tags. Custom tags, also known as tag extensions, are JSP elements that allow custom logic and output provided by other Java components to be inserted into JSP pages. The logic provided through a custom tag is implemented by a Java object known as a tag handler.

Some examples of tasks that can be performed by custom tags include operating on implicit objects, processing forms, accessing databases and other enterprise services such as email and directories, and implementing flow control.

Custom tags have a rich set of features. They can:

- Be customized by means of attributes passed from the calling page.
- Pass variables back to the calling page.
- Access all the objects available to JSP pages.
- Communicate with each other. You can create and initialize a JavaBeans component, create a public EL variable that refers to that bean in one tag, and then use the bean in another tag.
- Be nested within one another and communicate by means of private variables.

Deploying Java Web Framework with the CrownPeak CMS

Before creating the Java Web framework project CrownPeak Java Web hosting will be configured in the following CMS creation process:

- Create ticket for IT and send it to support@crownpeak.com providing the following information:
  Instance Name: Name of the CMS instance
  Hosting: Java Runtime Environment (JRE) for both Stage and Live
- Application server version: currently Tomcat V7 is supported; some Java EE implementations will require other app servers, such as JBoss or WebSphere (IBM).
- Request JSTL (JavaServer Pages Standard Tag Library) installation
- After internal approval process has completed, IT will create the JRE and will setup the application server.
- Request JSTL installation

Implementing the Java Web Framework to CrownPeak CMS

Integrating a Java Web application with CrownPeak CMS is accomplished in the multiple development environments:

- The Java Web application objects will be developed and maintained in the client's existing IDE, such as Eclipse, Netbeans, IntelliJ.
- The CrownPeak CMS, is used to manage all phases of the integrated Web application, from importing Java objects and global assets through to deployment.
- The CrownPeak Desktop Connection, along with Microsoft Visual Studio or Visual Web Developer 2010 Express, will be used develop and maintain all CMS templates, models and widgets.

A Java Web application will be implemented in the CrownPeak CMS in the following steps:

1. Build (compile) the Java EE application project in the Java IDE, and export it to a library file (JAR/WAR/EAR). Unzip the library file into the defined file folders.
2. Import to CMS all configuration files - Web.xml, struts.config, etc.
3. Import to CMS all compiled .class objects to the /bin folder in CMS.
4. Import to CMS all global assets - .css, .js, images, documents, etc.
5. Create any widgets required.
6. Create templates for all the view files (*.jsp) in CMS.
7. Create models for all templates available for authoring.
8. Publish the files and assets from the CMS.