Web Component Development with Servlet & JSP Technologies (EE 6)

Module-5: Container Facilities for Servlets and JSPs

Team Emertxe

ORACLE
Certified Expert
Java EE 6 Web
Component Developer

EMERTXE

Objectives

Upon completion of this module, you should be able to:

- Describe the purpose of deployment descriptors
- Determine the context root for a web application
- Create servlet mappings to allow invocation of a servlet
- Create and access context and init parameters
- Use the @WebServlet and @WebInitParam annotations to configure your servlets

Relevance

Discussion – The following questions are relevant to understanding what technologies are available for developing web applications and the limitations of those technologies:

- How can you write an application without knowing everything about the environment in which it will be used?
- How can you configure a web application if you don't have access to the command line that launched it?

Deployment Descriptors

- In versions of Java EE prior to EE 6, deployment configuration was supported using the web.xml file.
- Because of the advent of annotations the web.xml file is optional since Java EE 6.
- In addition to allowing configurationn to be done uisng Annotations, another method of specifying deployemnt information is also supported. This is the use of web-fragment.xml file.

The web-fragment.xml Files

- The web-fragment.xml file provides a mechanism to include deployment information that is specific to a library, without having to edit the main deployment descriptor for the whole web application. This reflects the OO principle of keeping unrelated things apart.

- A web-fragment.xml file, located under the META-INF directory of a JAR file placed within the WEB-INF/lib directory of a WAR archive, will be read and the configuration contained within will be integrated into the configuration of the web application as a whole.

  Note – Not all configurations can be achieved using annotations.

Controlling Configuration

With three elements capable of providing configuration information, web.xml, web-fragment.xml and annotations, it is possible for these to provide conflicting information. To address this, some rules and controles are provided. Three are particularly significant:

- Precedence
- <metadada-complete> Tag
- Ordering
- If desired, you can apply a <name> tag to an XML file, and using that name, specify either an <absolute-ordering> or an <ordering> tag to indicate the order in which configuration elements should be applied.

Dynamic Configuration of Web Applications

Java EE 6 adds features that allow the programmatic configuration of servlets, filters, and related components. This feature is only available as the web application is being loaded, specifically during the time that the servlet context is being initialized. It is perhaps most useful for framework Creators.

Servlet Mapping in web.xml

```
<web-app>

  <servlet>
    <servlet-name>MyServletName</servlet-name>
    <servlet-class>com.emertxe.MyServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>MyServletName</servlet-name>
    <url-pattern>/MyServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

EMERTXE

Multiple and Wildcard URL Patterns

```
<servlet-mapping>
<servlet-name>MyServlet</servlet-name>
<url-pattern>/MyServlet</url-pattern>
<url-pattern>/YourServlet</url-pattern>
<url-pattern>/HisServlet/*</url-pattern>
</servlet-mapping>
```

http://localhost:8080/MyAppRoot/MyServlet

http://localhost:8080/MyAppRoot/YourServlet

http://localhost:8080/MyAppRoot/HisServlet

http://localhost:8080/MyAppRoot/HisServlet/one

http://localhost:8080/MyAppRoot/HisServlet/twenty

Mapping Using Annotations

Since Java EE 6, servlet mappings can be achieved using annotations. A servlet should be given the annotation:

javax.servlet.annotation.WebServlet

This annotation supports various optional elements, but a common form is that which has been used in examples in previous modules:

```
@WebServlet(name="MyServletName",
urlPatterns={"/MyServlet", “/YourServlet”,
“/HisServlet/*”})
```

Invoking a Servlet by Name

A servlet can be invoked directly by its name through a RequestDispatcher, in a forward or include call. The necessary dispatcher is obtained using the servlet context which offers a method getNamedDispatcher(String).

Servlet Context Information

The deployment descriptor can include configuration information for the servlet. Two configuration elements of frequent utility are context parameters and init parameters.

Context Parameters

The term “servlet context” essentially refers to the web application and the container in which the servlet runs.

An interface javax.servlet.ServletContext provides access to several aspects of this environment, but in the context of this discussion, the methods of interest are two methods that allow read-only access to a map of context Parameters. These methods are:

String getInitParameter(String name)

Enumeration<String> getInitParameterNames()

Supplying Context Parameter

Context cannot be supplied using annotations, because they relate to the entire web-application, rather than to a single element of it.

```
<web-fragment [...]>
<context-param>
<param-name>fragmentContext</param-name>
<param-value>fragment Context value</param-value>
</context-param>
</web-fragment>
```

Note: If this is done using a web.xml, then line 1 would read <web..., rather than <web-fragment....

Reading Context Parameters

Context parameters can be read in Java code using statements of the following form:

```
String paramValue =
this.getServletContext().getInitParameter("paramName");
```

Notice that the servlet (this) gives access to an object of type ServletContext, which describes the configuration of the application as a whole, from which the parameter may be read.

Servlet Initialization Parameters

In addition to configuration parameters for the entire application,initialization parameters can be associated with the servlet individually. These are typically known simply as init parameters. This association may be achieved using the deployment descriptor.

Servlet Initialization Parameters

```
<servlet>
<servlet-name>MyServlet</servlet-name>
<servlet-class>SL314.package.MyServlet</servlet-class>
<init-param>
<param-name>name</param-name>
<param-value>Fred Jones</param-value>
</init-param>
<init-param>
</servlet>
```

EMERTXE

Servlet Initialization Using Annotations

Because init parameters are directly associated directly with the servlet, they can be specified using the annotations mechanism. The annotation that would be equivalent to the preceeding declaration would be:

```
WebServlet(
name=”MyServlet”,
urlPatterns=({“/MyServ”}),
initParams =
{@WebInitParam(name="name", value="Fred Jones")}

)
```

Reading Servlet Initialization Parameters

The servlet itself provides direct access to servlet initialization parameters using the method getInitParameter(String), and another method of the same signature is provided in ServletConfig object. So, these two lines of code, executed in the servlet, are equivalent:

```
name = this.getInitParameter("name");
name = getServletConfig().getInitParameter("name");
```

Other Configuration Elements

- The deployment descriptor can contain elements of many other types beyond those discussed in this module. Similarly, several other annotations exist for configuration purposes.

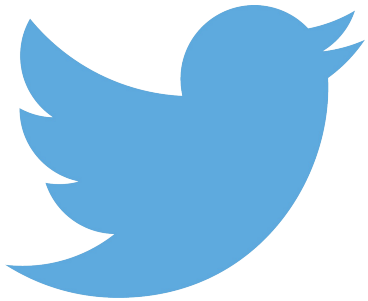
- While these are topics for later modules, you should know that the deployment descriptor is used to configure security, and connections to other aspects of a Java EE system, such as Enterprise JavaBeansTM (EJB),message queues, and databases.

Stay connected

About us: Emertxe is India's one of the top IT finishing schools & self learning kits provider. Our primary focus is on Embedded with diversification focus on Java, Oracle and Android areas

Emertxe Information Technologies,
No-1, 9th Cross, 5th Main,
Jayamahal Extension,
Bangalore, Karnataka 560046
T: +91 80 6562 9666
E: training@emertxe.com

https://www.facebook.com/Emertxe



https://twitter.com/EmertxeTweet

https://www.slideshare.net/EmertxeSlides

EMERTXE

Thank You