Controlling Web Application Behavior with web.xml

Prepared By :Vaishali Bharvada

Purpose of the Deployment Descriptor

- The deployment descriptor, web.xml, is used to control many facets of a Web application.
- Using web.xml, you can assign custom URLs for invoking servlets, specify initialization parameters for the entire application as well as for specific servlets, control session timeouts, declare filters, declare security roles, restrict access to Web resources based on declared security roles, and so on.
- The deployment descriptor is not part of the Java compilation process. Therefore, changes in web.xml don’t force you to recompile your code.
- In addition, the separation between the configuration mechanism, web.xml, and the Java code allows for the division between the development and deployment roles within the development process.

Elements of Web.xml file

- **icon**
- **display-name**
- **description**
- **distributable**
- **context-param**
- **filter**
- **filter-mapping**
- **listener**
- **servlet**
- **servlet-mapping**

Continue…….

- session-config
- mime-mapping
- welcome-file-list
- error-page
- taglib
- resource-env-ref
- resource-ref
- security-constraint
- login-config
- security-role
- env-entry
- ejb-ref
- ejb-local-ref

Assigning Names and Custom URLs

- **Assigning Names :**
- You assign a name by means of the servlet element.
- <servlet>
- <servlet-name>Test</servlet-name>
- <servlet-class>
- coreservlets.TestServlet
- </servlet-class>
- </servlet>

Continue…

- This means that the servlet at WEB-INF/classes/coreservlets/TestServlet is now known by the registered name Test.
- Giving a servlet a name has the following major implications:
- Initialization parameters, custom URL patterns, and other customizations refer to the servlet by the registered name, not by the class name.

Defining Custom URLs

- To assign a custom URL, you use the servlet-mapping element along with its servlet-name and url-pattern subelements.
- *The value of url-pattern must begin with either / or *..*
- Exact-Match Patterns:
- **<servlet-mapping>**
- **<servlet-name>Test</servlet-name>**
- **<url-pattern>/UrlTest</url-pattern>**
- **</servlet-mapping>**

Continue….

- Multimapping Patterns :
- By giving a url-pattern of /directoryName/*, you can specify that all URLs of the form http://host/webAppPrefix/directoryName/blah are handled by the designated servlet.
- By giving a url-pattern of *.jsp, you can specify that all URLs of the form http://host/webAppPrefix/.../blah.jsp are handled by the designated servlet.

Matching Overlapping Patterns

- When mapping a servlet to a URL, the specification does not allow the same value of url-pattern to appear twice within the same web.xml file.
- Thus, there can never be an overlap between two patterns that map exact matches.
- However, if one or more servlet mappings use "*", an overlap could occur.
- Compliant servers are required to use the following rules to resolve these overlaps.

Continue……

- **Exact matches are handled first.**
- Thus, if /Servlet/DS and /Servlet/* were both url-pattern entries, the first would take precedence for a request URL of http://host/webAppPrefix/Servlet/DS.
- **Directory mappings are preferred over extension mappings.**
- Thus, if /servlet/* and *.html were both url-pattern entries, the first would take precedence for a request URL of http://host/webAppPrefix/servlet/first.html.
- **For overlapping directory mappings, the longest path is preferred.**
- Thus, if /Servlet/DS/* and /servlet/* were both url-pattern entries, the first would take precedence for a request URL of http://host/webAppPrefix/Servlet/DS/first.html.

Naming JSP Pages

- Because JSP pages get translated into servlets, it is natural to expect that you can name JSP pages just as you can name servlets.
- After all, JSP pages might benefit from initialization parameters, security settings, or custom URLs, just as regular servlets do.
- it is true that JSP pages are really servlets behind the scenes.
- You don’t know the actual class name of JSP pages (because the system picks the name). So, to name JSP pages, you substitute the jsp-file element for the servlet-class element, as follows:

Continue…….

- <servlet>
- <servlet-name>PageName
- </servlet-name>
- <jsp-file>
- /WEB-INF/jspPages/TestPage.jsp
- </jsp-file>
- </servlet>

Disabling the Invoker Servlet

- One reason for setting up a custom URL for a servlet or JSP page is so that you can register initialization parameters to be read from the init (servlets) or jspInit (JSP pages) methods.
- However, the initialization parameters are available only when the servlet or JSP page is accessed by means of a custom URL pattern, not when it is accessed with the default URL of http://host/webAppPrefix/servlet/package.Servlet-Class.
- Consequently, you might want to turn off the default URL so that nobody accidentally calls the uninitialized servlet.
- This process is sometimes known as disabling *the invoker servlet, because most servers have a standard servlet that is registered* with the default servlet URLs and simply invokes the real servlet.

Initializing and Preloading Servlets and JSP Pages

- **Assigning Servlet Initialization Parameters :**
- There are a few common things that are keeping in mind when dealing with initialization parameters:
- **Return values.**
- The return value of getInitParameter is always a String. So, for integer parameters you might use Integer.parseInt to obtain an int.

Continue……..

- **Nonexistent values.**
- If the key passed into the getInitParameter method does not appear inside the servlet's init-param declarations, null will be returned.
- **Initialization in JSP.**
- JSP pages use jspInit, not init. JSP pages also require use of the jsp-file element in place of servlet-class.
- **Default URLs.**
- Initialization parameters are only available when servlets are accessed through custom URL patterns associated with their registered names.

Assigning JSP Initialization Parameters

- **(1) You use jsp-file instead of servlet-class**

```
<servlet>
<servlet-name>InitPage</servlet-name>
<jsp-file>/InitPage.jsp</jsp-file>
<init-param>
<param-name>...</param-name>
<param-value>...</param-value>
</init-param>
</servlet>
```

- **(2) You should assign the original URL of the JSP page as its custom URL pattern.**

Continue…

- <servlet-mapping>
- <servlet-name>InitPage</servlet-name>
- **<url-pattern>/InitPage.jsp</url-pattern>**
- </servlet-mapping>
- **(3)The JSP page uses jspInit, not init.**
- public void jspInit()
- { }
- Example : http://localhost:9000/chtwo/InitPage

Supplying Application-Wide Initialization Parameters

- In some situations you want to supply system-wide initialization parameters that can be read by any servlet or JSP page by means of the getInitParameter method of **ServletContext.**
- You use the context-param element to declare these system-wide initialization values.
- The context-param element should contain param-name,param-value, and, optionally, description subelements, as shown here.

Continue..

- <context-param>
- <param-name>name</param-name>
- <param-value>abc</param-value>
- </context-param>

- Example: MyServlet.java
- http://localhost:9000/chtwo/index.html

Loading Servlets When the Server Starts

- we use load-on-startup to guarantee that the LoadInit Servlet's init method is run when the Web application is first loaded

```
<servlet>
<servlet-name>LoadInit</servlet-name>
<servlet-class>coreservlets.LoadInitServlet
</servlet-class>
<init-param>
<param-name>companyName</param-name>
<param-value>Atmiya Infotech</param-value>
</init-param>
<load-on-startup>0</load-on-startup>
</servlet>
```

Continue….

- The integer 0 in the load-on-startup's element body tells the server that this servlet should be loaded into memory at server startup before any other servlet or JSP page.

Declaring Filters

- <filter>
- <filter-name>Reporter</filter-name>
- <filter-class>coreservlets.ReportFilter</filter-class>
- </filter>
- <filter-mapping>
- <filter-name>Reporter</filter-name>
- <servlet-name>SomeServletName</servlet-name>
- </filter-mapping>
- <filter-mapping>
- <filter-name>Reporter</filter-name>
- <url-pattern>/*</url-pattern>
- </filter-mapping>

Specifying Welcome Pages

- <welcome-file-list>
- <welcome-file>index.jsp</welcome-file>
- <welcome-file>index.html</welcome-file>
- </welcome-file-list>

Designating Pages to Handle Errors

- <error-page>
- <error-code>404</error-code>
- <location>/NotFound.jsp</location>
- </error-page>
- <error-page>
- <exception-type>
- package.ClassName
- </exception-type>
- <location>/SomeURL</location>
- </error-page>

Example

- <error-page>
  <exception-type>
- java.lang.Exception
- </exception-type>
  <location>/error.jsp</location>
  </error-page>
- <error-page>
  <exception-type>
- java.lang.Throwable
- </exception-type>
  <location>/OtherErrors.jsp</location>
  </error-page>

Controlling Session Timeouts

- <session-config>
- <session-timeout>180</session-timeout>
- </session-config>
- the value of the session-timeout sub element is specified in minutes.

Documenting Web Applications

- A number of the web.xml elements are designed not for the server, but for the visual development environment.
- These include icon, display-name, and description.

icon

- The icon element specifies the location within the Web Application for a small and large image used to represent the Web Application in a GUI tool.
- <icon>
- <small-icon>/small-book.gif</small-icon>
- <large-icon>/tome.jpg</large-icon>
- </icon>
- A 16 × 16 GIF or JPEG image can be specified with the small-icon element,
- and a 32 × 32 image can be specified with large-icon.

display-name

- The display-name element provides a name that the GUI tools might use to label this particular Web application.
- Here is an example:
- <display-name>Rare Books</display-name>

description

- The description element provides explanatory text, as shown here:
- <description>
- This Web application represents the store developed for rare-books.com, an online bookstore specializing in rare and limited-edition books.
- </description>

Associating Files with MIME Types

- Servers typically have a way for Webmasters to associate file extensions with media types. So, for example, a file named mom.jpg would automatically be given a MIME type of image/jpeg.
- <mime-mapping>
- <extension>.jpg</extension>
- <mime-type>image/jpeg</mime-type>
- </mime-mapping>

Continue….

- perhaps your Web application wants to override standard mappings.
- For instance, the following would tell the server to designate .ps files as plain text (text/plain) rather than as PostScript application/postscript) when sending them to clients.
- <mime-mapping>
- <extension>ps</extension>
- <mime-type>text/plain</mime-type>
- </mime-mapping>

Configuring JSP Pages

- The jsp-config element is used to provide configuration information for the JSP pages in a Web application.
- It has two subelements, taglib and jsp-property-group.
- Both subelements can appear zero or more times under the jsp-config element, but any of the taglib subelements must appear before any of the jsp-property-group subelements.

Locating Tag Library Descriptors

- <jsp-config>
- <taglib>
- <taglib-uri>/charts</taglib-uri>
- <taglib-location>
- /WEB-INF/charttags.tld
- </taglib-location>
- </taglib>
- </jsp-config>
- USE :
- <% taglib uri="/charts" prefix="somePrefix" %>

Configuring JSP Page Properties

- JSP page properties are configured using one or more jsp-property-group elements.
- (1) **url-pattern:**
- The url-pattern element contains the mapping used to match URLs to JSP pages.
- <jsp-config>
- <jsp-property-group>
- **<url-pattern>**/WEB-INF/myjsps/***</url-pattern>**
- <!-- ... -->
- </jsp-property-group>
- </jsp-config>

Continue..

- **(2)el-ignored**
- The el-ignored element can be set to either true or false (e.g.,<el-ignored>true</el-ignored>).
- When set to true, the affected pages turn off JSP Expression Language (EL) processing and treat JSP EL as regular text.

Continue…

- (3) **scripting-invalid:**
- The scripting-invalid element can be set to either true or false.
- e.g.,
- <scripting-invalid>true</scripting-invalid>.
- When set to true, the server will produce a translation time error if any JSP page in this property group uses scripting declarations, scriptlets, or scripting expressions.

Continue…

- (4) **is-xml**
- The is-xml element can be set to either true or false.
- e.g.,
- <is-xml> true</is-xml>
- If set to true, this tells the server that the group of resources that match the URL pattern of this property group are to be treated as JSP Documents.
- A JSP Document is a JSP page that contains only valid XML code.

Continue…

- (5) **include-prelude**
- The include-prelude element contains a context-relative path that must correspond to a resource in the Web application.
- When the resource is present, the given path will be automatically included, using the static include directive, at the beginning of each JSP page in the jsp-property-group.
- This capability can be useful if you want to provide a standard header in multiple pages of your application.

Continue…

- (6) **include-coda**
- The include-coda element contains a context-relative path that must correspond to a resource in the Web application.
- When the resource is present, the given path will be automatically included, using the static include directive,at the end of each JSP page in this jsp-property-group.
- Similar to the include-prelude element, this can be useful if you want to provide a standard footer across multiple pages in your application.

Example :

- **<jsp-config>**
- <jsp-property-group>
- <url-pattern>/WEB-INF/jspPages/ustm/*
- </url-pattern>
- <el-ignored>true</el-ignored>
- <include-prelude>
- /WEB-INF/jspPages/USTMHeader.jsp
- </include-prelude>
- <include-coda>
- /WEB-INF/jspPages/USTMFooter.jsp
- </include-coda>
- </jsp-property-group>
- </jsp-config>

Configuring Character Encoding

- <locale-encoding-mapping-list>
- <locale-encoding-mapping>
- <locale>ja</locale>
- <encoding>Shift_JIS</encoding>
- </locale-encoding-mapping>
- </locale-encoding-mapping-list>

Designating Application Event Listeners

- <listener>
- <listener-class>
- package.ListenerClass
- </listener-class>
- </listener>

Developing for the Clustered Environment

- enterprise-level applications are deployed in a clustered environment.
- A clustered environment usually consists of many machines connected through the local area network (LAN) and sometimes even through the wide area network (WAN).
- a hardware load balancer is placed in front of all of these machines.
- this keeps each machine in the cluster equally loaded, so no single machine's resources are exhausted.

Continue…

- This behavior is achieved by sharing the HttpSession among the machines in the cluster.
- Even though each machine in the cluster has its own Java Virtual Machine (JVM), the HttpSession object gets copied and shared among the cluster.
- Because clustered Web applications run in multiple JVMs, you may not rely on the usual mechanisms of sharing data used in regular Web applications

Continue…

- (1) **Avoid instance variables and static data (such as singletons) for shared data.**
- Each JVM in a cluster will have its own copy of the instance variables and static data. Changes to this data in one JVM will leave all other JVMs unaffected.

(2) **Don't store data in the ServletContext**

- Each JVM in a cluster has its own copy of the ServletContext.
- Therefore, if you store an attribute in the ServletContext, the ServletContext object of other servers (JVMs) will not contain this attribute.

Continue…

- (3) **Objects stored in HttpSession must implement Serializable.**
- The servlet specification requires compliant Web containers to support migration of objects stored in the HttpSession that implement the Serializable interface.
- If the objects stored in the HttpSession do not implement the Serializable interface, the container may fail to migrate the session.

Continue…

- **(4)Only minimal information should be stored in HttpSession.**
- For a clustered environment to function as a single Web application, the data stored in the HttpSession must be kept in sync with the other servers in the cluster.
- This is achieved by sending the data back and forth between the servers.
- Naturally, this consumes a lot of resources.
- Therefore, storing a lot of data in the HttpSession could considerably degrade the performance of your Web application even when the request load is not very high.

Continue…

- In the deployment descriptor, web.xml, the distributable element indicates that the Web application is programmed in such a way that servers that support clustering can safely distribute the Web application across multiple servers.
- The distributable element contains no subelements or data—it is simply a flag (as
- follows):
- <distributable />

THANK YOU