Servlets - Annotations

So far, you have learnt how Servlet uses the deployment descriptor (web.xml file) for deploying your application into a web server. Servlet API 3.0 has introduced a new package called javax.servlet.annotation. It provides annotation types which can be used for annotating a servlet class. If you use annotation, then the deployment descriptor (web.xml) is not required. But you should use tomcat7 or any later version of tomcat.

Annotations can replace equivalent XML configuration in the web deployment descriptor file (web.xml) such as servlet declaration and servlet mapping. Servlet containers will process the annotated classes at deployment time.

The annotation types introduced in Servlet 3.0 are −

|  |  |
| --- | --- |
| **Sr.No.** | **Annotation & Description** |
| 1 | **@WebServlet**  To declare a servlet. |
| 2 | **@WebInitParam**  To specify an initialization parameter. |
| 3 | **@WebFilter**  To declare a servlet filter. |
| 4 | **@WebListener**  To declare a WebListener |
| 5 | **@HandlesTypes**  To declare the class types that a ServletContainerInitializer can handle. |
| 6 | **@HttpConstraint**  This annotation is used within the ServletSecurity annotation to represent the security constraints to be applied to all HTTP protocol methods for which a corresponding HttpMethodConstraint element does NOT occur within the ServletSecurity annotation. |
| 7 | **@HttpMethodConstraint**  This annotation is used within the ServletSecurity annotation to represent security constraints on specific HTTP protocol messages. |
| 8 | **@MultipartConfig**  Annotation that may be specified on a Servlet class, indicating that instances of the Servlet expect requests that conform to the multipart/form-data MIME type. |
| 9 | **@ServletSecurity**  This annotation is used on a Servlet implementation class to specify security constraints to be enforced by a Servlet container on HTTP protocol messages. |

Here we have discussed some of the Annotations in detail.

@WebServlet

The @WebServlet is used to declare the configuration of a Servlet with a container. The following table contains the list of attributes used for WebServlet annotation.

|  |  |
| --- | --- |
| **Sr.No.** | **Attribute & Description** |
| 1 | **String name**  Name of the Servlet |
| 2 | **String[] value**  Array of URL patterns |
| 3 | **String[] urlPatterns**  Array of URL patterns to which this Filter applies |
| 4 | **Int loadOnStartup**  The integer value gives you the startup ordering hint |
| 5 | **WebInitParam[] initParams**  Array of initialization parameters for this Servlet |
| 6 | **Boolean asyncSupported**  Asynchronous operation supported by this Servlet |
| 7 | **String smallIcon**  Small icon for this Servlet, if present |
| 8 | **String largeIcon**  Large icon for this Servlet, if present |
| 9 | **String description**  Description of this Servlet, if present |
| 10 | **String displayName**  Display name of this Servlet, if present |

At least one URL pattern MUST be declared in either the **value** or **urlPattern** attribute of the annotation, but not both.

The **value** attribute is recommended for use when the URL pattern is the only attribute being set, otherwise the **urlPattern** attribute should be used.

Example

The following example describes how to use @WebServlet annotation. It is a simple servlet that displays the text **Hello Servlet**.

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebInitParam;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet(value = "/Simple")

public class Simple extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.print("<html><body>");

out.print("<h3>Hello Servlet</h3>");

out.print("</body></html>");

}

}

Compile **Simple.java** in the usual way and put your class file in <Tomcat-installationdirectory>/webapps/ROOT/WEB-INF/classes.

Now try to call any servlet by just running *http://localhost:8080/Simple*. You will see the following output on the web page.

Hello servlet

@WebInitParam

The @WebInitParam annotation is used for specifying an initialization parameter for a Servlet or a Filter. It is used within a WebFilter or WebSevlet annotations. The following table contains the list of attributes used for WebInitParam annotation.

|  |  |
| --- | --- |
| **Sr.No.** | **Attribute & Description** |
| 1 | **String name**  Name of the initialization parameter |
| 2 | **String value**  Value of the initialization parameter |
| 3 | **String description**  Description of the initialization parameter |

Example

The following example describes how to use @WeInitParam annotation along with @WebServlet annotation. It is a simple servlet that displays the text **Hello Servlet** and the string value **Hello World!** which are taken from the **init** parameters.

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebInitParam;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet(value = "/Simple", initParams = {

@WebInitParam(name = "foo", value = "Hello "),

@WebInitParam(name = "bar", value = " World!")

})

public class Simple extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.print("<html><body>");

out.print("<h3>Hello Servlet</h3>");

out.println(getInitParameter("foo"));

out.println(getInitParameter("bar"));

out.print("</body></html>");

}

}

Compile **Simple.java** in the usual way and put your class file in <Tomcat-installationdirectory>;/webapps/ROOT/WEB-INF/classes.

Now try to call any servlet by just running *http://localhost:8080/Simple*. You will see the following output on the web page.

Hello Servlet

Hello World!

@Webfilter

This is the annotation used to declare a servlet filter. It is processed by the container at deployment time, and the corresponding filter applied to the specified URL patterns, servlets, and dispatcher types.

The **@WebFilter** annotation defines a filter in a web application. This annotation is specified on a class and contains metadata about the filter being declared. The annotated filter must specify at least one URL pattern. The following table lists the attributes used for WebFilter annotation.

|  |  |
| --- | --- |
| **Sr.No.** | **Attribute & Description** |
| 1 | **String filterName**  Name of the filter |
| 2 | **String[] urlPatterns**  Provides array of values or urlPatterns to which the filter applies |
| 3 | **DispatcherType[] dispatcherTypes**  Specifies the types of dispatcher (Request/Response) to which the filter applies |
| 4 | **String[] servletNames**  Provides an array of servlet names |
| 5 | **String displayName**  Name of the filter |
| 6 | **String description**  Description of the filter |
| 7 | **WebInitParam[] initParams**  Array of initialization parameters for this filter |
| 8 | **Boolean asyncSupported**  Asynchronous operation supported by this filter |
| 9 | **String smallIcon**  Small icon for this filter, if present |
| 10 | **String largeIcon**  Large icon for this filter, if present |

Example

The following example describes how to use @WebFilter annotation. It is a simple LogFilter that displays the value of Init-param **test-param** and the current time timestamp on the console. That means, the filter works like an interface layer between the request and the response. Here we use "/\*" for urlPattern. It means, this filter is applicable for all the servlets.

import java.io.IOException;

import javax.servlet.annotation.WebFilter;

import javax.servlet.annotation.WebInitParam;

import javax.servlet.\*;

import java.util.\*;

// Implements Filter class

@WebFilter(urlPatterns = {"/\*"}, initParams = {

@WebInitParam(name = "test-param", value = "Initialization Paramter")})

public class LogFilter implements Filter {

public void init(FilterConfig config) throws ServletException {

// Get init parameter

String testParam = config.getInitParameter("test-param");

//Print the init parameter

System.out.println("Test Param: " + testParam);

}

public void doFilter(ServletRequest request, ServletResponse response,

FilterChain chain) throws IOException, ServletException {

// Log the current timestamp.

System.out.println("Time " + new Date().toString());

// Pass request back down the filter chain

chain.doFilter(request,response);

}

public void destroy( ) {

/\* Called before the Filter instance is removed

from service by the web container\*/

}

}

Compile **Simple.java** in the usual way and put your class file in <Tomcat-installationdirectory>/webapps/ROOT/WEB-INF/classes.

Now try to call any servlet by just running *http://localhost:8080/Simple*. You will see the following output on the web page.

Hello Servlet

Hello World!

Now, open the servlet console. There, you will find the value of the **init** parameter **testparam** and the **current timestamp** along with servlet notification messages.