Servlets:

Annotations in servlet3

1. **WebServlet** – We can use this annotation with Servlet classes to define init parameters, loadOnStartup value, description and url patterns etc. At least one URL pattern MUST be declared in either the value or urlPattern attribute of the annotation, but not both. The class on which this annotation is declared MUST extend HttpServlet.
2. **WebInitParam** – This annotation is used to define init parameters for servlet or filter, it contains name, value pair and we can provide description also. This annotation can be used within a WebFilter or WebServlet annotation.
3. **WebFilter** – This annotation is used to declare a servlet filter. This annotation is processed by the container during deployment, the Filter class in which it is found will be created as per the configuration and applied to the URL patterns, Servlets and DispatcherTypes. The annotated class MUST implement javax.servlet.Filter interface.
4. **WebListener** – The annotation used to declare a listener for various types of event, in a given web application context.
5. **@Override** – When we want to override a method of Superclass, we should use this annotation to inform compiler that we are overriding a method. So when superclass method is removed or changed, compiler will show error message. Learn why we should always use [java override annotation](http://www.journaldev.com/817/overriding-methods-in-java-always-use-override-annotation) while overriding a method.
6. **@Deprecated** – when we want the compiler to know that a method is deprecated, we should use this annotation. Java recommends that in javadoc, we should provide information for why this method is deprecated and what is the alternative to use.
7. **@SuppressWarnings** – This is just to tell compiler to ignore specific warnings they produce, for example using raw types in [java generics](http://www.journaldev.com/1663/java-generics-example-method-class-interface). It’s retention policy is SOURCE and it gets discarded by compiler.
8. package com.journaldev.servlet;
9. import java.io.IOException;
10. import java.io.PrintWriter;
11. import javax.servlet.RequestDispatcher;
12. import javax.servlet.ServletException;
13. import javax.servlet.annotation.WebInitParam;
14. import javax.servlet.annotation.WebServlet;
15. import javax.servlet.http.HttpServlet;
16. import javax.servlet.http.HttpServletRequest;
17. import javax.servlet.http.HttpServletResponse;
18. /\*\*
19. \* Servlet Tutorial - Servlet Example
20. \*/
21. @WebServlet(
22. description = "Login Servlet",
23. urlPatterns = { "/LoginServlet" },
24. initParams = {
25. @WebInitParam(name = "user", value = "Pankaj"),
26. @WebInitParam(name = "password", value = "journaldev")
27. })
28. public class LoginServlet extends HttpServlet {
29. private static final long serialVersionUID = 1L;

32. public void init() throws ServletException {
33. //we can create DB connection resource here and set it to Servlet context
34. if(getServletContext().getInitParameter("dbURL").equals("jdbc:mysql://localhost/mysql\_db") &&
35. getServletContext().getInitParameter("dbUser").equals("mysql\_user") &&
36. getServletContext().getInitParameter("dbUserPwd").equals("mysql\_pwd"))
37. getServletContext().setAttribute("DB\_Success", "True");
38. else throw new ServletException("DB Connection error");
39. }
41. protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
42. //get request parameters for userID and password
43. String user = request.getParameter("user");
44. String pwd = request.getParameter("pwd");
46. //get servlet config init params
47. String userID = getServletConfig().getInitParameter("user");
48. String password = getServletConfig().getInitParameter("password");
49. //logging example
50. log("User="+user+"::password="+pwd);
52. if(userID.equals(user) && password.equals(pwd)){
53. response.sendRedirect("LoginSuccess.jsp");
54. }else{
55. RequestDispatcher rd = getServletContext().getRequestDispatcher("/login.html");
56. PrintWriter out= response.getWriter();
57. out.println("<font color=red>Either user name or password is wrong.</font>");
58. rd.include(request, response);
60. }
62. }
63. }

Methods of RequestDispatcher interface

The RequestDispatcher interface provides two methods. They are:

1. **public void forward(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
2. **public void include(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Includes the content of a resource (servlet, JSP page, or HTML file) in the response.

**Life Cycle:**

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.

|  |  |
| --- | --- |
| **Get** | **Post** |
| 1) Limited amount of data can be sent because data is sent in header. | Large amount of data can be sent because data is sent in body. |
| 2) Not Secured because data is exposed in URL bar. | Secured because data is not exposed in URL bar. |
| 3) Can be bookmarked | Cannot be bookmarked |
| 4) Idempotent | Non-Idempotent |
| 5) It is more efficient and used than Post | It is less efficient and used |

PrintWriter is a character-stream class where as ServletOutputStream is a byte-stream class. The PrintWriter class can be used to write only character-based information whereas ServletOutputStream class can be used to write primitive values as well as character-based information.

|  |  |
| --- | --- |
| **forward() method** | **sendRedirect() method** |
| 1) forward() sends the same request to another resource. | 1) sendRedirect() method sends new request always because it uses the URL bar of the browser. |
| 2) forward() method works at server side. | 2) sendRedirect() method works at client side. |
| 3) forward() method works within the server only. | 3) sendRedirect() method works within and outside the server. |

Filter

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.\*;

**public** **class** MyFilter **implements** Filter{

**public** **void** init(FilterConfig arg0) **throws** ServletException {}

**public** **void** doFilter(ServletRequest req, ServletResponse resp,

    FilterChain chain) **throws** IOException, ServletException {

    PrintWriter out=resp.getWriter();

    out.print("filter is invoked before");

    chain.doFilter(req, resp);//sends request to next resource

    out.print("filter is invoked after");

    }

**public** **void** destroy() {}

}