Servlet

Sandeep Kulange, SunBeam

PowerPoint

Introduction

- Servlet is a specification, created by Pavni Diwanji while she worked at Sun Microsystems.
- tomcat/lib contains "servlet-api.jar" file which contains implementation of Servlet specification.
- A servlet is a web component hosted in a servlet container and generates dynamic content.
- If we want to do servlet programming then we should import following packages:
    1. javax.servlet
    2. javax.servlet.http
    3. java.io
- If we want to define servlet then we should extend it from "javax.servlet.http.HttpServlet" class.

Servlet Versions

Sr. No.	Servlet API Version	Platform
1	Servlet 1.0	Not Specified
2	Servlet 2.0	JDK 1.1
3	Servlet 2.1	Not Specified
4	Servlet 2.2	J2EE 1.2
5	Servlet 2.3	J2EE 1.3
6	Servlet 2.4	J2EE 1.4
7	Servlet 2.5	Java EE 5
8	Servlet 3.0	Java EE 6
9	Servlet 3.1	Java EE 7
10	Servlet 4.0	Java EE 8

Why Servlet ?

- Applet is a java class which runs in web browsers memory while Servlet is a java class which runs in web servers memory.
- Job of servlet:
  1. Request processing.
  2. Taking care of business logic(B.L)
  3. Taking care of page navigation
  4. Generating dynamic response
  5. Managing DAO.
- In case of MVC(Model-View-Controller) application, Servlet is designed to use as a controller.

Servlet compilation and execution from terminal

1. create JEE complaint directory structure in tomcat/webapps.
2. Define and save HelloServlet class in WEB-INF/src directory.
3. Configure servlet using either @WebServlet annotation or using web.xml file
4. To compile servlet from WEB-INF directory, Set classpath of "servlet-api.jar”
5. Compile servlet and save .class file inside WEB-INF/classes directory.
6. Start external tomcat server
7. Open web browser and make request using URL

Deployment descriptor

- "web.xml" file is called deployment descriptor.
- It contains deployment instructions for the servlet container.
- There is only one web.xml file per web application.
- Developer is responsible for defining web.xml.
- It is always kept inside WEB-INF directory.
- Web container/Servlet container read "web.xml" file during deployment time and only once.

Servlet Names

- For the purpose of flexibility and improving security, we assign 3 names to servlet.
- A servlet can have three names.
    1. Public URL name - The name client knows
    2. Deployment/internal secret name.
    3. File path name - F.Q. Class Name.
- To configure servlet,either we can use Deployment Descriptor(DD) or WebServlet annotation.
- Using DD to map URLs to servlet:
    1. <servlet>
        - map internal name to F.Q. Class Name
    2. <servlet-mapping>
        - map internal name to URL name

Using Deployment Descriptor to map URLs to servlet

```
<web-app ... >
          <servlet>
                    <servlet-name>HelloServlet</servlet-name>
                    <servlet-class>pages.HelloServlet</servlet-class>
          </servlet>
          <servlet-mapping>
                    <servlet-name>HelloServlet</servlet-name>
                    <url-pattern>/hello</url-pattern>
          </servlet-mapping>
</web-app>
```

- In above code:
    1. /hello : is URL name. It is also called URL pattern.
    2. HelloServlet : is deployment name. It can be any name.
    3. pages.HelloServlet : is F.Q class name.

More About Servlet Names

- Single servlet may have multiple URL patterns but it must have at least one URL pattern.

```
<servlet-mapping>
       <servlet-name>HelloServlet</servlet-name>
       <url-pattern>/hello1</url-pattern>
       <url-pattern>/hello2</url-pattern>
</servlet-mapping>
```

- If we try to deploy and execute servlet without url pattern then we will get "**HTTP Status 404**" error.
- We can not use same url pattern for multiple servlets. In this case WC throws **java.lang.IllegalArgumentException.**

Using annotation to map URLs to servlet

- WebServlet is declared in javax.servlet.annotation package.
- It is used to declare a servlet.
- This annotation is processed by the container at deployment time, and the corresponding servlet made available at the specified URL patterns.
- Consider the following code snippet.

```
@WebServlet("/hello")

public class TestServlet extends HttpServlet {       }
```

- Using urlPatterns attribute, we can specify multiple url patterns for the servlet.

```
@WebServlet(urlPatterns={"/h1","/h2"})

public class TestServlet extends HttpServlet{        }
```

Web Container

1. Servlets do not have main() method. They are under the control of another java application called a container.
2. It is a component of web server that interacts with servlets.
3. It is runtime environment, which is responsible for managing execution of dynamic components such as servlet, jsp, filter etc. In other words It is server side JRE.
4. Tomcat is a web container.
5. Following are the jobs of Web Container
    1. To create, HttpRequest and HttpResponse objects
    2. Mapping URL to a particular servlet and ensuring that the URL requester has the correct access-rights.
    3. To manage the life cycle of Servlet, JSP and Filter.
    4. To manage threads for every client request.
    5. To manage session.
    6. To support database connection pooling.

Servlet Life Cycle Methods

1. void init(ServletConfig config) throws ServletException
    - Called by the servlet container to indicate to a servlet that the servlet is being placed into service.
    - The servlet container calls the init method exactly once after instantiating the servlet.
2. void service(ServletRequest req, ServletResponse res) throws ServletException, IOException
    - Called by the servlet container to allow the servlet to respond to a request.
    - This method is only called after the servlet's init() method has completed successfully.
    - Servlet Container invoke “service()” method once per client request.
3. void destroy()
    - Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.
    - Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.

How Web Container handles a request?

1. User clicks a link that has a URL to a servlet instead of a static page.
2. The container "sees" that the request is for a servlet, so the container creates two objects:
    - HttpServletRequest
    - HttpServletResponse
3. The container finds the correct servlet based on URL in the request, allocates thread for a request and passes the request and response objects to the servlet thread.
4. The container calls "service()" method. Depending on type of request, the service method calls either the doGet() or doPost() method.
5. The doGet/doPost method generates the dynamic page and stuffs the page into response object.
6. The thread completes, the container response object into an Http response, sends it back to the client, then deletes request and response objects.

ServletRequest

1. It is interface declared in javax.servlet package.
2. The servlet container creates a ServletRequest object and passes it as an argument to the servlet's service() method.
3. To pass client request information to a servlet, container create instance of ServletRequest
4. Methods of ServletRequest Interface
    1. String getParameter(String name)
    2. Enumeration<String> getParameterNames()
    3. String[] getParameterValues(String name)
    4. Object getAttribute(String name)
    5. void setAttribute(String name, Object o)
    6. void removeAttribute(String name)
    7. RequestDispatcher getRequestDispatcher(String path)

HttpServletRequest

1. It is sub interface of ServletRequest interface.
2. It is used to provide request information to the Http Servlets.
3. The servlet container creates an HttpServletRequest object and passes it as an argument to the servlet's service methods (doGet, doPost, etc).
4. Instance of HttpServletRequest encapsulate Http request generated by web browser.
5. Methods of HttpServletRequest interface:
    1. Cookie[] getCookies()
    2. HttpSession getSession()
    3. HttpSession getSession(boolean create)

ServletResponse

1. It is interface declared in javax.servlet package.
2. The servlet container creates a ServletResponse object and passes it as an argument to the servlet's service() method.
3. To send servlet response to a client, container create instance of ServletResponse
4. Methods of ServletResponse Interface:
    1. void flushBuffer()throws IOException
    2. PrintWriter getWriter()throws IOException
    3. void setContentType(String type)
    4. int getBufferSize()
    5. void setBufferSize(int size)

HttpServletResponse

1. It is sub interface of ServletResponse interface.
2. It is used to provide HTTP-specific functionality in sending a response.
3. The servlet container creates an HttpServletResponse object and passes it as an argument to the servlet's service methods (doGet, doPost, etc).
4. Instance of HttpServletResponse encapsulates Http response.
5. Methods of HttpServletResponse:
    1. void addCookie(Cookie cookie)
    2. String encodeURL(String url)
    3. String encodeRedirectURL(String url)
    4. void sendRedirect(String location)throws IOException

PrintWriter

1. It is a sub class of Writer class declared in java.io package.
2. It prints formatted representations of objects to a text-output stream.
3. Methods in this class never throw I/O exceptions.
4. In context of servlet, use PrintWriter object to write HTML text to the response object.
5. Some of the methods are:
    1. public void print(String x)
    2. public void println(String x)
    3. public PrintWriter printf(String format,Object... args)

Linking Html page to Servlet

1. When we are using URL in HTML, If the location is relative without a leading '/' the container interprets it as relative to the current request URI.
    - e.g   <form action="login" method="post">
    - url generated : http://localhost:8080/TestWebApp/login
2. When we are using URL in HTML, If the location is relative with a leading '/' the container interprets it as relative to the servlet container root.
    - e.g   <form action="/login" method="post">
    - url generated : http://localhost:8080/login
3. When we are using URL in HTML, If the location is relative with two leading '/' the container interprets it as a network-path reference
    - e.g   <form action="/login" method="post">
    - url generated : http://login/

Thank you