Java Servlet

Overview

- A servlet is a small Java program that runs in a web server (servlet container) such as Apache Tomcat.
- Provides dynamic, user-oriented content in web applications using a request-response programming model.
- Can respond to any type of request but is commonly used to handle HTTP requests.
- Low-level API; other technologies use the Servlet API in some way (e.g. JSP, Spring MVC, Tapestry, etc.).
- Represented by the javax.servlet.Servlet interface.
- Rather than implementing the interface, the usual approach is to extend from the javax.servlet.GenericServlet or most often from javax.servlet.http.HttpServlet.

Maven dependency

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
    <scope>provided</scope>
</dependency>
```

Servlet lifecycle

The lifecycle of a servlet is controlled by a container in which the server has been deployed (e.g. Tomcat). It's the container that creates a new instance of a Servlet class.

1. If an instance of a servlet does not exist, the web container:
    - Loads the servlet class
    - Creates an instance of the servlet class
    - Initializes the servlet class by calling the Servlet.init() method.
2. The container invokes the Servlet.service() method, passing request and response objects.
3. If the container is to be shut down (the Tomcat server is being stopped), the container finalizes the servlet by calling the servlet's destroy() method.

Servlet.init() method

- Designed to be called only once when the servlet is first created.
- Used for one-time initializations.
- Signature:

```
public void init() throws ServletException {
  // Initialization code
}
```

Servlet.service() method

- The main method that performs the actual task of handling the request.
- The servlet container (i.e. the web server) spawns a new thread and invokes the service() method each time a new request is received.
- The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet(), doPost(), doPut(), doDelete(), etc. accordingly.
- Signature:

```
public void service(ServletRequest request,  ServletResponse response)
    throws ServletException, IOException {
}
```

HttpServlet.doGet() method

- Used to process HTTP GET request for e.g. a URL or from a HTML form that has been submitted with GET method type.
- Signature:

```
public void doGet(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {
  // Servlet code
}
```

HttpServlet.doPost() method

- Used to process HTTP POST request which originate from HTML forms that use the POST method (which is the default one).
- Signature:

```
public void doPost(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
  // Servlet code
}
```

@WebServlet

- Used to define a servlet component in a web application.
- The URL pattern to which the servlet will respond needs to be specified.
- Example:

```
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;

@WebServlet("/report")
public class MoodServlet extends HttpServlet {…}
```

Dealing with request data

Passing data with a GET method

- The GET method sends the encoded user data appended to the page request **as part of the URL**. The page and the encoded data are separated by the “?” character.
- Example:
  - http://www.test.com/hello?key1=value1&key2=value2
- Should never be used to pass on sensitive data such as passwords.
- A servlet handles GET methods using the **doGet()** method.

Passing data with a POST method

- A generally more reliable method for passing data to a backend program.
- Instead of using the URL, the data gets encoded in the **request body**.
- Example:

```
<html>
  <form action="http://localhost:8080/hello" method="POST">
    <input type="text" name="name" />
    <input type="submit" value="Send name" />
  </form>
</html>
```

- A servlet handles POST method using the **doPost()** method.

Accessing request data in a servlet

The HttpServletRequest attribute provides several methods for accessing the data which has been sent by a client:

- **getParameter()**: Call this method to get the value of a form parameter.
- **getParameterValues()**: Call this method if the parameter appears more than once and returns multiple values, for example checkbox.
- **getParameterNames()**: Call this method if you want a complete list of all parameters in the current request.

Running a Java web application

Maven war plugin configuration

- The absence of the web.xml deployment descriptor which is optional for Servlet 3.0 and onwards must be explicitly declared.
- Maven configuration:

```
<plugin>
    <artifactId>maven-war-plugin</artifactId>
    <version>2.3</version>
    <configuration>
        <failOnMissingWebXml>false</failOnMissingWebXml>
    </configuration>
</plugin>
```

Tomcat7 Maven plugin

- Useful for debugging purposes during development.
- Maven configuration:

```
<plugin>
    <groupId>org.apache.tomcat.maven</groupId>
    <artifactId>tomcat7-maven-plugin</artifactId>
    <version>2.2</version>
    <configuration>
        <path>/</path>
    </configuration>
</plugin>
```

- Run using “mvn tomcat7:run”

Example: HelloWebWorld

Exercises

Exercise: WebCalculator

- Create a simple web calculator that supports the operations of adding and subtracting two integer numbers.
- Enable the calculator to work through both GET and POST methods.