T.A.SANDALI SEWMINI-27745

Tutorial 01

Answer all questions.

1)What is a Servlet in Java?

A servlet in Java is a server-side component that extends the capabilities of servers to generate dynamic web content. It handles requests, processes them, and delivers responses back to the client. Servlets are platform-independent Java classes used to create web application.

2) What is meant by static and dynamic websites. List down their differences.

- A static webpage remains the same or fixed, in terms of the content it displays.
- A dynamic webpage is the opposite, its content changes according to the location of the user, or based on actions a user has made on the page before.

STATIC	DYNAMIC
It is faster to load as compared to dynamic websites.	It is slower than a static websites.
The content of web pages can not be changed at runtime.	The content of web pages can be changed.
Cheaper Development costs.	More Development costs.
No interaction with the database is possible.	Interaction with the database is possible.
The same content is delivered ever time the page is loaded.	Content may change every time the page is loaded.
HTML,CSS,Javascript is used for developing the website.	Server-side languages such as PHP,and Node.js are used.

3) What are the differences between the Get and Post requests?

GET requests: Used to request data from a specified resource. Parameters are passed in the URL, visible to users. Suitable for retrieving data, but less secure for sensitive information.

POST requests: Used to submit data to be processed to a specified resource. Parameters are sent in the request body, not visible in the URL. Suitable for sending sensitive data like passwords.

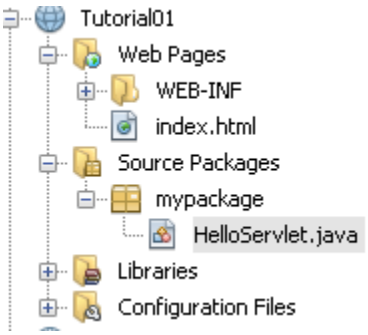
Feature	GET Method	POST Method
Operation	Used to retrieve information from the server.	Used to send data to the server to create/update a resource.
Data Location	Appends data to the URL, visible to all.	Includes data in the request body, not displayed in the URL.
Idempotency	idempotent; the same request can be repeated with no further changes.	Non-idempotent; repeating the same request can lead to different results.
Data Size	Limited by the URL length; less data can be sent.	No limitations on data size; suitable for large amounts of data.
Caching	Can be cached.	Not cached by default.
Security	Less secure as data is exposed in the URL	More secure; data is concealed within the request body.
Use Case	Ideal for searching and retrieving data.	Ideal for transactions and updating data

4) Create a new Java web project in NetBeans.

2.2 Add the necessary servlet dependencies to your project.

2.3 Define a simple servlet class named HelloServlet that extends HttpServlet

Project



HelloServlet.java

```
@WebServlet(name = "HelloServlet", urlPatterns = {"/HelloServlet"})
public class HelloServlet extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
```

5) What is the purpose of the web.xml file in a servlet-based web application?

3.2 Configure the HelloServlet in the web.xml file with a servlet mapping.

The web.xml file is a deployment descriptor for a Java web application. It contains configuration information that the servlet container uses to deploy and run the application.

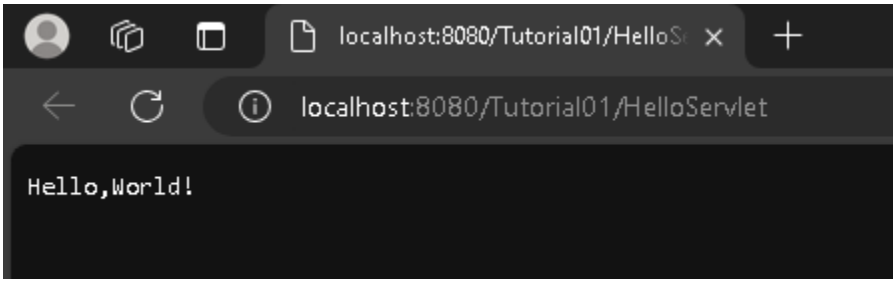
```
<servlet>
   <servlet-name>HelloServlet</servlet-name>
   <servlet-class>HelloServlet</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>HelloServlet</servlet-name>
   <url-pattern>/hello</url-pattern>
</servlet-mapping>
```

6) Override the doGet method in the HelloServlet to handle HTTP GET requests.

4.2 Write code inside the doGet method to send a simple "Hello, World!" response to the client.

HelloServlet.java

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        PrintWriter out = response.getWriter();
    out.println("Hello,World!");
       // processRequest(request, response);
    }
```



7) Override the doPost method in the HelloServlet to handle HTTP POST requests. 5.2 Write code inside the doPost method to send a "Hello, POST!" response to the client.

HelloServlet.java

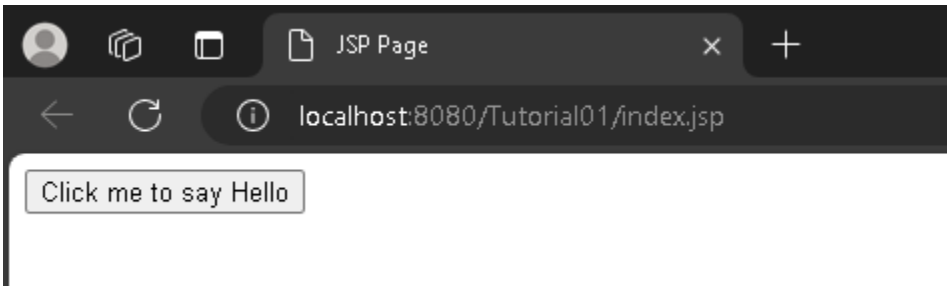
```
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("Hello, POST!");
       // processRequest(request, response);
    }
```

8) Configure the Web Pages Folder and delete index.html

6.2 create a new index.jsp and add a click button to redirect to your servlet.

Index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <form action="HelloServlet" method="get">
        <input type="submit" value="Click me to say Hello">
    </form>
    </body>
</html>
```



9) Deploy your servlet project to a local servlet container (e.g., GlassFish Server).

Right-click on your project.

Select Run.

Choose your local servlet container (e.g., GlassFish Server) and click OK.

Open a web browser and navigate to the URL where your servlet is mapped. Verify that you can see the "Hello, World!" or "Hello, POST!" message:

Open a web browser.

Navigate to http://localhost:8080/Tutorial01/index.jsp.

Click on the button to trigger the servlet.

You should see the "Hello, World!" or "Hello, POST!" message displayed.