Practical No 7

Aim: Write a Java program to implement servlet

Resources Required:

- · Java Development Kit (JDK)
- · Text Editor (e.g., Notepad++) or IDE (e.g., Eclipse, IntelliJ IDEA, NetBeans)
- · Command-line terminal or Java compiler

Theory:

Java Servlet is a Java program that runs on a Java-enabled web server or application server. It handles client requests, processes them and generates responses dynamically.
Features

- Servlets work on the server side.
- Servlets are capable of handling complex requests obtained from the web server.
- Generate dynamic responses efficiently.

Types of Servlets
1. GenericServlet

- It's an abstract class that implements the Servlet interface.
- Protocol-independent (does not depend on HTTP).
- Provides basic lifecycle methods (init(), destroy()) and the main method service(ServletRequest req, ServletResponse res).

2. HttpServlet

- Extends GenericServlet.
- Specifically designed for handling HTTP requests and responses.

Provides methods for each HTTP method:

1. doGet() for GET requests
2. doPost() for POST requests
3. doPut(), doDelete(), doHead(), etc.

Source Code: Generic Servlet

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author DS
 */
public class MyGenericServlet extends HttpServlet {

    @Override
    public void service(ServletRequest request, ServletResponse response)
          throws ServletException, IOException {

       // Set the content type of the response
       response.setContentType("text/html");

        // Write HTML content to the response
        try ( // Get a PrintWriter to write the response
              PrintWriter out = response.getWriter()) {
           // Write HTML content to the response
           out.println("<html>");
           out.println("<head><title>Generic Servlet Example</title></head>");
           out.println("<body>");
           out.println("<h2>Hello from MyGenericServlet!</h2>");
           out.println("<p>This is a simple generic servlet.</p>");
           out.println("</body>");
           out.println("</html>");

           // Close the PrintWriter
        }
    }
}
```

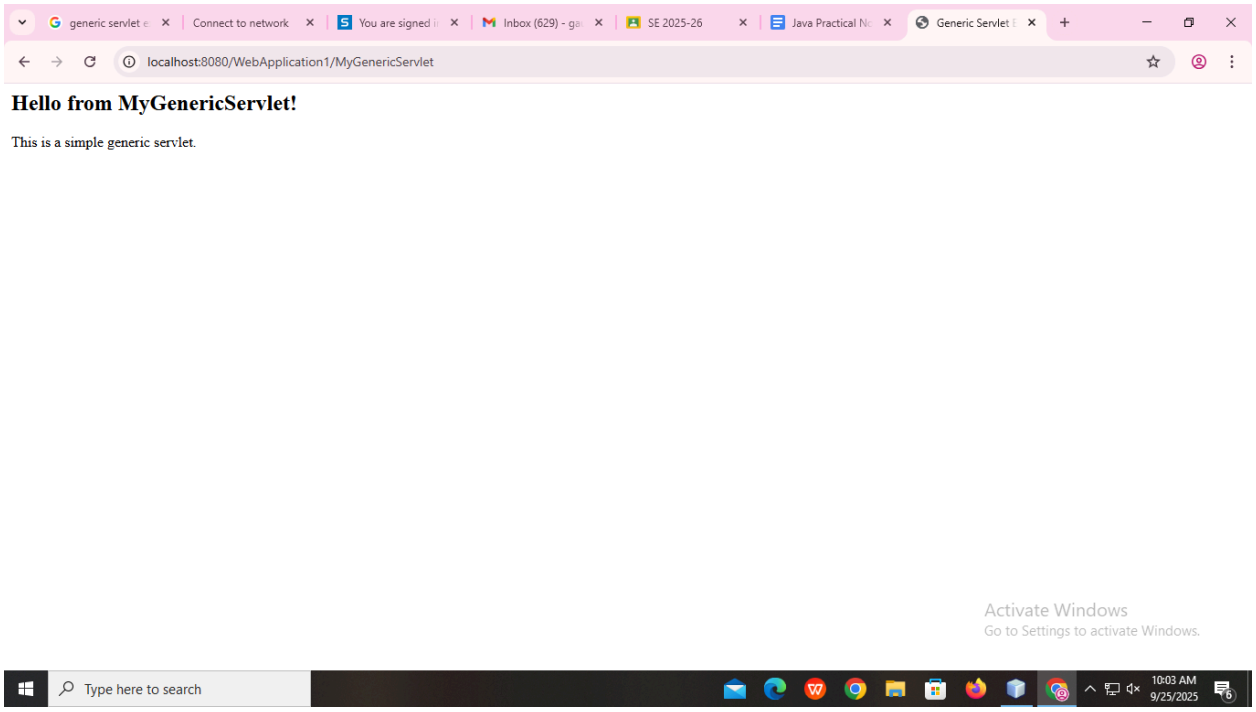
Source Code: Http Servlet

```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class MyHttpServlet extends HttpServlet {
```

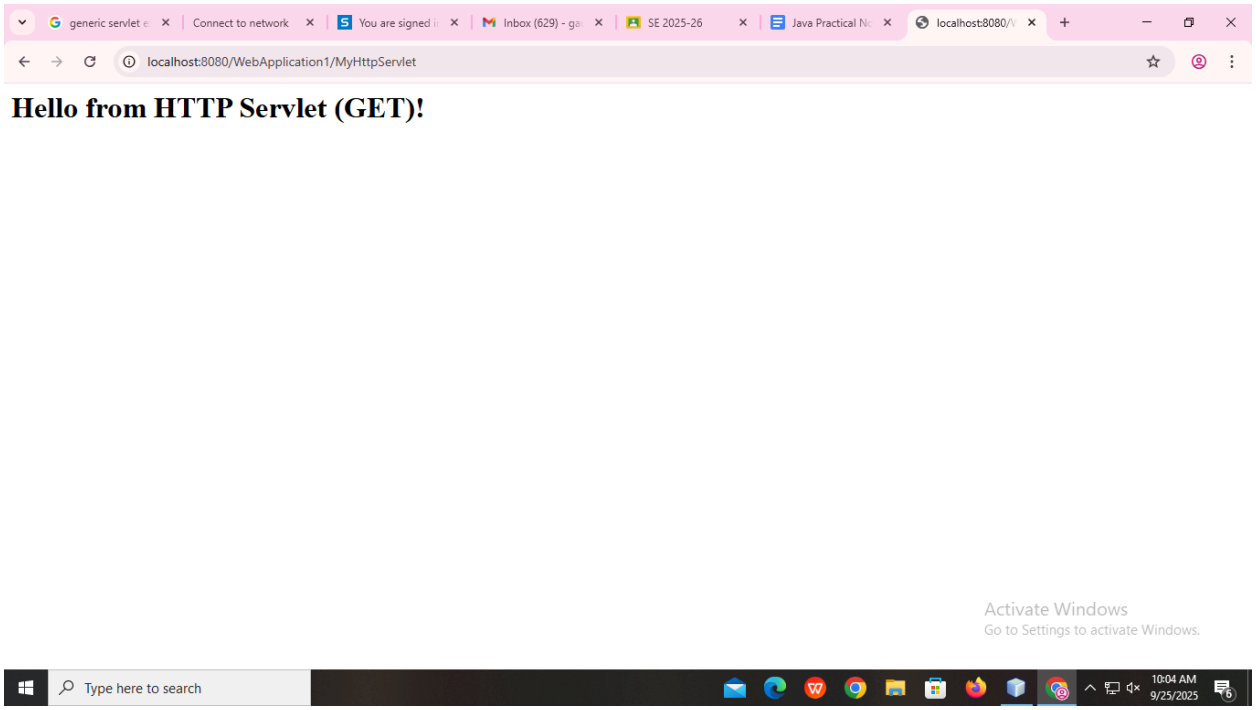
```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        response.getWriter().println("<h1>Hello from HTTP Servlet (GET)!</h1>");
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        response.getWriter().println("<h1>Hello from HTTP Servlet (POST)!</h1>");
        // Process form data, etc.
    }
}
```

Output:

generic Servlet



http Servlet



Conclusion:

This Java program successfully demonstrates servlet implementation.