PRACTICAL NO. 05: Servlet Introduction

Java Web Technology | Eclipse IDE | Apache Tomcat 9 | Windows 8.1 (64-bit)

---

a) Write a program for simple servlet, testing servlet and deployment descriptor
b) Implement program using doget and doPost method, life cycle of servlet.
c) Write a program to create servlet file for different database operation.
d) Implement a program for error handling, authentication and context interface

Prerequisites & Setup

Software Required

Software	Details
JDK	Java SE 8 (64-bit) — set JAVA_HOME in Environment Variables
Apache Tomcat	Version 9.x — extract to C:\apache-tomcat-9
Eclipse IDE	Eclipse IDE for Java EE Developers (64-bit) — NOT standard Java edition
MySQL	MySQL Community Server 5.7+ (for Part C only)
MySQL Connector	mysql-connector-java-8.x.x.jar (for Part C only)

Step-by-Step Eclipse & Tomcat Setup

1. Install JDK 8 (64-bit). After installation, open System Properties → Environment Variables. Add JAVA_HOME = C:\Program Files\Java\jdk1.8.0_xxx and append ;%JAVA_HOME%\bin to the PATH variable.
2. Download Apache Tomcat 9 (zip version). Extract to C:\apache-tomcat-9. Do NOT install to Program Files (permission issues on Win 8.1).
3. Open Eclipse. Go to Window → Preferences → Server → Runtime Environments → Click Add.
4. Select Apache Tomcat v9.0 → click Next → Browse to C:\apache-tomcat-9 → Finish.
5. Create a new project: File → New → Dynamic Web Project. Enter name: ServletPractical. Set Target Runtime: Apache Tomcat v9.0. Click Next until the last screen and check Generate web.xml deployment descriptor → Finish.

Project Folder Structure

```
ServletPractical/
├── src/                          ← Java servlet source files (.java)
│   ├── SimpleServlet.java
│   ├── LifecycleServlet.java
│   ├── DBServlet.java
│   └── AuthServlet.java
└── WebContent/                   ← Web resources
    ├── WEB-INF/
    │   ├── web.xml               ← Deployment descriptor
    │   └── lib/                  ← External JARs (e.g. MySQL connector)
    │       └── mysql-connector-java-8.x.x.jar
    ├── index.html                ← Welcome page
    ├── error404.html
    └── error500.html
```

📌 *In Eclipse, src/ maps to Java Resources/src. Right-click src → New → Servlet to create each servlet.*

Part A — Simple Servlet + Deployment Descriptor

A servlet is a Java class that runs on the server and handles HTTP requests. It must extend HttpServlet and override doGet() or doPost(). The web.xml (deployment descriptor) tells Tomcat which URL maps to which servlet class.

Step 1: Create SimpleServlet.java

Right-click src → New → Servlet → Class name: SimpleServlet → Finish. Replace the contents with:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SimpleServlet extends HttpServlet {

    public void init() throws ServletException {
        System.out.println("SimpleServlet initialized!");
    }

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        out.println("<html><body>");
        out.println("<h2 style='color:green;'>Hello from SimpleServlet!</h2>");
        out.println("<p>Server Name: " + request.getServerName() + "</p>");
        out.println("<p>Server Port: " + request.getServerPort() + "</p>");
        out.println("</body></html>");
        out.close();
    }

    public void destroy() {
        System.out.println("SimpleServlet destroyed!");
    }
}
```

Step 2: Configure web.xml (Deployment Descriptor)

Open WebContent/WEB-INF/web.xml and add the servlet registration and URL mapping:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1"
  xmlns="http://xmlns.jcp.org/xml/ns/javaee">

  <servlet>
    <servlet-name>SimpleServlet</servlet-name>
    <servlet-class>SimpleServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>SimpleServlet</servlet-name>
    <url-pattern>/hello</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
```

```
</web-app>
```

Step 3: Run & Test

6. Right-click project → Run As → Run on Server → Select Apache Tomcat v9.0 → Finish.
7. Wait for the Console to print: INFO: Starting Servlet Engine.
8. Open browser and go to: http://localhost:8080/ServletPractical/hello

Expected Output: Hello from SimpleServlet! Server Name: localhost Server Port: 8080

Part B — doGet, doPost & Servlet Lifecycle

The servlet lifecycle has five stages: class loading, instantiation, init(), service() (which calls doGet/doPost), and destroy(). The init() and destroy() methods are called only once; service() is called for every request.

Lifecycle Summary

Method	When Called
init()	Once — when servlet is first loaded into memory
doGet()	Every GET request (browser URL, anchor link, form method=GET)
doPost()	Every POST request (form method=POST)
service()	Called automatically before doGet/doPost — routes the request
destroy()	Once — when server shuts down or servlet is reloaded

`doGet()`

- Handles **HTTP GET** requests
- Data is sent via the **URL query string** → `?name=John&age=25`
- **Visible in browser address bar** — not secure for sensitive data
- Can be **bookmarked and cached**
- Has a **URL length limit** (~2048 characters)
- **Idempotent** — repeated calls produce the same result
- Used for **fetching/retrieving data**

`doPost()`

- Handles **HTTP POST** requests
- Data is sent in the **request body** — not visible in the URL
- **More secure** for sensitive data (passwords, form data)
- **Cannot be bookmarked or cached**
- **No size limit** on data
- **Not idempotent** — repeated calls may have different effects
- Used for **submitting data** (forms, logins, file uploads)

When to Use Which?

Scenario	Use
Search query	`doGet`
Login form	`doPost`
Load a page	`doGet`
File upload	`doPost`
REST API read	`doGet`
REST API create	`doPost`

Create LifecycleServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class LifecycleServlet extends HttpServlet {

    private int requestCount = 0;

    // Called ONCE at startup
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        System.out.println("=== init() called ===");
    }

    // Handles GET requests
    public void doGet(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
        requestCount++;
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<html><body>");
        out.println("<h2>doGet() called</h2>");
        out.println("<p>Requests served: " + requestCount + "</p>");
        out.println("<form method='post' action='lifecycle'>");
        out.println("  Name: <input type='text' name='username'><br><br>");
        out.println("  <input type='submit' value='Submit POST'>");
        out.println("</form></body></html>");
        out.close();
    }

    // Handles POST requests
    public void doPost(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
        requestCount++;
        String name = req.getParameter("username");
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<html><body>");
        out.println("<h2>doPost() called</h2>");
        out.println("<p>Name received: " + name + "</p>");
        out.println("<a href='lifecycle'>Go Back</a>");
        out.println("</body></html>");
        out.close();
    }

    // Called ONCE at shutdown
    public void destroy() {
        System.out.println("=== destroy() called ===");
        System.out.println("Total requests: " + requestCount);
    }
}
```

Add to web.xml

```
<servlet>
  <servlet-name>LifecycleServlet</servlet-name>
  <servlet-class>LifecycleServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>LifecycleServlet</servlet-name>
  <url-pattern>/lifecycle</url-pattern>
</servlet-mapping>
```

📌 *load-on-startup with value 1 means Tomcat loads this servlet immediately at server start, rather than waiting for the first request.*

Test URLs

Action	URL
Test doGet	http://localhost:8080/ServletPractical/lifecycle
Test doPost	Fill the form on the page and click Submit POST

Expected Console Output: === init() called === (on first request) === destroy() called === (on server stop)

Part C — Database Operations Servlet (CRUD)

Step 1: Setup MySQL Database

Open MySQL Command Prompt or MySQL Workbench and run:

```
CREATE DATABASE servletdb;
USE servletdb;
CREATE TABLE students (
    id    INT AUTO_INCREMENT PRIMARY KEY,
    name  VARCHAR(100),
    marks INT
);
```

Step 2: Add MySQL JAR to Project

9. Download mysql-connector-java-8.x.x.jar from MySQL official website.
10. Copy the JAR file to: WebContent/WEB-INF/lib/
11. In Eclipse, right-click project → Build Path → Configure Build Path → Libraries tab → Add JARs → select the JAR from WEB-INF/lib → OK.

Step 3: Create DBServlet.java

```
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class DBServlet extends HttpServlet {

    static final String URL  = "jdbc:mysql://localhost:3306/servletdb";
    static final String USER = "root";
    static final String PASS = "your_password"; // change this

    // doGet: show all records + forms
    public void doGet(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<html><body>");
        out.println("<h2>Student Database</h2>");

        // INSERT form
        out.println("<form method='post' action='dbops'>");
        out.println("Name: <input type='text' name='sname'><br>");
        out.println("Marks: <input type='number' name='marks'><br>");
        out.println("<input type='hidden' name='action' value='insert'>");
        out.println("<input type='submit' value='Insert'></form><hr>");

        // UPDATE form
        out.println("<form method='post' action='dbops'>");
        out.println("ID: <input type='number' name='sid'><br>");
        out.println("New Marks: <input type='number' name='marks'><br>");
        out.println("<input type='hidden' name='action' value='update'>");
        out.println("<input type='submit' value='Update'></form><hr>");

        // DELETE form
```

```
        out.println("<form method='post' action='dbops'>");
        out.println("ID: <input type='number' name='sid'><br>");
        out.println("<input type='hidden' name='action' value='delete'>");
        out.println("<input type='submit' value='Delete'></form><hr>");

        // SELECT ALL
        out.println("<table
border='1'><tr><th>ID</th><th>Name</th><th>Marks</th></tr>");
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(URL, USER, PASS);
            ResultSet rs = con.createStatement().executeQuery(
                           "SELECT * FROM students");
            while (rs.next()) {
                out.println("<tr><td>" + rs.getInt("id") + "</td>");
                out.println("<td>" + rs.getString("name") + "</td>");
                out.println("<td>" + rs.getInt("marks") + "</td></tr>");
            }
            con.close();
        } catch(Exception e) {
            out.println("<tr><td colspan='3'>Error: "+e.getMessage()+"</td></tr>");
        }
        out.println("</table></body></html>");
        out.close();
    }

    // doPost: handle INSERT / UPDATE / DELETE
    public void doPost(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
        String action = req.getParameter("action");
        String msg = "";
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(URL, USER, PASS);
            if ("insert".equals(action)) {
                PreparedStatement ps = con.prepareStatement(
                    "INSERT INTO students(name,marks) VALUES(?,?)");
                ps.setString(1, req.getParameter("sname"));
                ps.setInt(2, Integer.parseInt(req.getParameter("marks")));
                ps.executeUpdate();
                msg = "Record inserted!";
            } else if ("update".equals(action)) {
                PreparedStatement ps = con.prepareStatement(
                    "UPDATE students SET marks=? WHERE id=?");
                ps.setInt(1, Integer.parseInt(req.getParameter("marks")));
                ps.setInt(2, Integer.parseInt(req.getParameter("sid")));
                msg = ps.executeUpdate() > 0 ? "Updated!" : "ID not found.";
            } else if ("delete".equals(action)) {
                PreparedStatement ps = con.prepareStatement(
                    "DELETE FROM students WHERE id=?");
                ps.setInt(1, Integer.parseInt(req.getParameter("sid")));
                msg = ps.executeUpdate() > 0 ? "Deleted!" : "ID not found.";
            }
            con.close();
        } catch(Exception e) { msg = "Error: " + e.getMessage(); }
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<html><body>");
        out.println("<h3 style='color:green;'>" + msg + "</h3>");
        out.println("<a href='dbops'>Go Back</a></body></html>");
        out.close();
    }
}
```

Add to web.xml

```
<servlet>
  <servlet-name>DBServlet</servlet-name>
  <servlet-class>DBServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>DBServlet</servlet-name>
  <url-pattern>/dbops</url-pattern>
</servlet-mapping>
```

Test

12. Visit http://localhost:8080/ServletPractical/dbops
13. Enter a name and marks → click Insert. Page shows: Record inserted!
14. Enter an existing ID and new marks → click Update.
15. Enter an ID → click Delete.
16. The table at the bottom always shows current records from the database.

Part D — Error Handling, Authentication & Context Interface

Key Concepts

Concept	Explanation
ServletContext	Application-wide object shared across all servlets; stores app-level attributes
HttpSession	Per-user session object; stores user-specific data across multiple requests
Error handling	Try-catch inside doPost to catch validation/DB errors; error pages in web.xml for 404/500
Authentication	Checking username and password in doPost; setting session attributes on success

Create AuthServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class AuthServlet extends HttpServlet {

    // init() — set app-wide data in ServletContext
    public void init() throws ServletException {
        ServletContext ctx = getServletContext();
        ctx.setAttribute("appName", "Servlet Practical App");
        ctx.setAttribute("version", "1.0");
    }

    // doGet — show login form
    public void doGet(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
        ServletContext ctx = getServletContext();
        String appName = (String) ctx.getAttribute("appName");
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<html><body>");
        out.println("<h2>" + appName + "</h2>");
        out.println("<form method='post' action='auth'>");
        out.println("Username: <input type='text' name='user'><br><br>");
        out.println("Password: <input type='password' name='pass'><br><br>");
        out.println("<input type='submit' value='Login'>");
        out.println("</form></body></html>");
        out.close();
    }

    // doPost — authenticate with error handling
    public void doPost(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
        String user = req.getParameter("user");
        String pass = req.getParameter("pass");
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        try {
            // Input validation
            if (user == null || user.trim().isEmpty())
```

```
                throw new ServletException("Username cannot be empty!");
            if (pass == null || pass.trim().isEmpty())
                throw new ServletException("Password cannot be empty!");

            // Check credentials
            if ("admin".equals(user) && "1234".equals(pass)) {
                HttpSession session = req.getSession();
                session.setAttribute("loggedUser", user);
                session.setMaxInactiveInterval(300);
                ServletContext ctx = getServletContext();
                out.println("<html><body>");
                out.println("<h2 style='color:green;'>Login Successful!</h2>");
                out.println("<p>Welcome: " + user + "</p>");
                out.println("<p>Session ID: " + session.getId() + "</p>");
                out.println("<p>App: " + ctx.getAttribute("appName") + "</p>");
                out.println("<p>Server: " + ctx.getServerInfo() + "</p>");
                out.println("</body></html>");
            } else {
                res.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
                out.println("<html><body>");
                out.println("<h2 style='color:red;'>Authentication Failed</h2>");
                out.println("<p>Invalid credentials.</p>");
                out.println("<a href='auth'>Try again</a></body></html>");
            }
        } catch (ServletException e) {
            res.setStatus(HttpServletResponse.SC_BAD_REQUEST);
            out.println("<html><body>");
            out.println("<h2 style='color:orange;'>Input Error</h2>");
            out.println("<p>" + e.getMessage() + "</p>");
            out.println("<a href='auth'>Go back</a></body></html>");
        }
        out.close();
    }
}
```

Create Error Pages

Create error404.html in WebContent/:

```
<!DOCTYPE html>
<html>
<head><title>404 Not Found</title></head>
<body style='text-align:center; margin-top:60px;'>
  <h1 style='color:red;'>404 - Page Not Found</h1>
  <p>The page you requested does not exist.</p>
  <a href='index.html'>Go to Home</a>
</body>
</html>
```

Create error500.html in WebContent/:

```
<!DOCTYPE html>
<html>
<head><title>500 Server Error</title></head>
<body style='text-align:center; margin-top:60px;'>
  <h1 style='color:red;'>500 - Internal Server Error</h1>
  <p>Something went wrong on the server.</p>
  <a href='index.html'>Go to Home</a>
</body>
</html>
```

Add to web.xml

```
<servlet>
```

```
  <servlet-name>AuthServlet</servlet-name>
  <servlet-class>AuthServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>AuthServlet</servlet-name>
  <url-pattern>/auth</url-pattern>
</servlet-mapping>

<!-- Error pages -->
<error-page>
  <error-code>404</error-code>
  <location>/error404.html</location>
</error-page>
<error-page>
  <error-code>500</error-code>
  <location>/error500.html</location>
</error-page>
```

Test Scenarios

Test	Expected Output
Visit /auth (GET)	Shows login form with app name from ServletContext
Submit admin / 1234	Login Successful! Shows session ID, server info
Submit wrong credentials	Authentication Failed — HTTP 401
Submit empty username	Input Error — Username cannot be empty! — HTTP 400
Visit a non-existent URL	Custom error404.html page is shown

Common Errors & Solutions

Error	Solution
HTTP 404 on servlet URL	Check <url-pattern> in web.xml matches the browser URL exactly. Ensure servlet class name in web.xml matches the .java filename.
ClassNotFoundException: com.mysql.cj.jdbc.Driver	MySQL JAR is missing from WEB-INF/lib. Right-click project → Build Path → Add External JARs.
Access denied for user root	Wrong password in DBServlet.java. Check MySQL service is running via services.msc (Win+R → services.msc).
Blank response in browser	Call response.setContentType("text/html") BEFORE getWriter(). Check for missing out.println() statements.
Port 8080 already in use	Another process is using port 8080. Stop it or change Tomcat port in conf/server.xml to 8081.
ClassCastException in web.xml	Servlet class does not extend HttpServlet. Check import and extends declaration.
Changes not reflected after edit	Eclipse may cache the old class. Right-click project → Refresh, then restart Tomcat.

Quick Reference — All Test URLs

Part	URL
A — Simple Servlet	http://localhost:8080/ServletPractical/hello
B — Lifecycle (GET)	http://localhost:8080/ServletPractical/lifecycle
C — DB Operations	http://localhost:8080/ServletPractical/dbops
D — Authentication	http://localhost:8080/ServletPractical/auth

📌 *Default Tomcat port is 8080. If you changed it in server.xml, replace 8080 in the URLs above.*