Enterprice Java

➡ First Download Payara Server Community Edition 6.2025.7(Full)

- https://www.payara.fish/downloads/payara-platform-community-edition/

➡ How to Create project When using Servlet :

- *Apache Netbeans 25 -> New Project -> Java with Maven -> Web Application -> Payara Server -> Jakarata EE 10 Web ->Source Packages ->Servlet Package -> Servlet.java(If Network not Available then Java with Ant -> Java Web -> Web Application)*

1.BookQueryStringServlet.java

1. Purpose of BookQueryStringServlet :

- **Learning HTTP GET vs POST :**

=> You've overridden both doGet() and doPost() and directed them to processRequest().

=> If you call servlet via a URL → GET request.

=> If you submit a form with method="POST" → POST request.

=> This way you can handle both GET and POST with same logic.

- **Dynamic Response Generation :-**

=> The servlet dynamically prints an HTML page showing the book details entered in query string/form.

2. URL-Mapping :

- *http://localhost:8080/Web1/BookQueryStringServlet?bname=Java+Programming\&authname=James+Gosling\&pname=Sun+Microsystems\&synopsis=Core+concepts+of+Java*

1.Employee.java & 2.OptimizedLogic.java & 3. DataLogic.java & 4.EmployeeServlet.java

1.How to add dependencies while perform CRUD Using Servlet :

- *in mavenproject -> Project Files ->*

-> **pom.xml :-- Add below Dependencies**

```
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <version>8.0.33</version>
   <scope>provided</scope>
</dependency>
```

=> mysql-connector-java is the official MySQL JDBC driver.It allows your Servlet (Java code) to connect to a MySQL database.

=> Adding this dependency makes Maven download the MySQL driver jar and put it in project's classpath.

=> add this dependency so Servlet can connect and interact with a MySQL database.

2.How to perform CRUD using Servlet :

- *in mavenproject -> Source Package -> new package create -> model -> OptimizedLogic.java :-*

-> Model ::

[1] **OptimizedLogic.java :-**

=> Above class is a Data Access Layer (DAL) or DAO (Data Access Object).It acts as a bridge between your Servlet (Java code) and the MySQL database.

=> Instead of writing SQL directly inside your servlet, you put all database logic here → making code cleaner, reusable, and easier to maintain.

- *in mavenproject -> Source Package -> new package create -> model ->Employee.java :-*

[2] Employee.java :-

=> In Java, we need an object to store that row's data.

=> That's what Employee is → it's a blueprint (class) to hold empno, ename, and salary.

-> Servlet ::

- *in mavenproject -> Source Package -> new package create -> Servlet ->EmployeeServlet.java :-*

[3] EmployeeServlet.java :-

=> The EmployeeServlet acts as the controller in the MVC architecture of this project.

-> Purpose ::

=> Handles HTTP requests (GET/POST) and interacts with the model classes (Employee, OptimizedLogic) to fetch employee data from the database.

-> Flow ::

=> Browser sends a request to /EmployeeServlet

=> Servlet calls OptimizedLogic to retrieve all employees

=> Generates an HTML response containing:

=> Employee List (Emp No, Name, Salary) in a table

=> Gross Salary & Max Salary summary

=> Sends the response back to the browser

3.URL-Mapping :

- *http://localhost:8080//EmployeeServlet*

1.CookieServlet.java

1. Purpose of CookieServlet :

- **Track user visits using cookies :**

=> The servlet remembers if a user has visited before by storing a visit cookie in their browser.

=> Each time the user comes back, the cookie value is updated and shown.

- **Demonstrate Cookie Handling in Servlets :**
- **How to create a cookie:**

new Cookie("visit", "1");

- **How to read cookies from request:**

request.getCookies();

- **How to update and send cookies back to the client:**

response.addCookie(visitor);

- **State Management in Web Applications :**

=> HTTP is stateless (server forgets user after each request).

=> Cookies help maintain state across requests (like remembering visits, login sessions, preferences, cart items, etc.).

---

1. SessionServlet1.java & 2.SessionServlet2.java

1. Purpose of SessionServlet1 and SessionServlet2 :

- **SessionServlet1 :**

=> Creates an HTTP session (if it doesn't exist).

- **Stores data in the session using:**

session.setAttribute("user", "admin");

session.setAttribute("rno", String.valueOf(Math.random()));

- **SessionServlet2 :**

=> Retrieves the same session.

- **Reads the data stored by SessionServlet1:**

session.getAttribute("user");

session.getAttribute("rno");

2. When to Use HttpSession :

- **Login / Authentication**

=> Store user details (username, roles) after login → available across all pages.

- **Shopping Cart**

=> Store cart items in session while the user browses.

- **Wizard / Multi-step Forms**

=> Save form data temporarily across multiple steps.

- **Tracking User-Specific Data**

=> Store temporary user preferences (language, theme, etc.).

2. Difference Between Cookie & Session (in your examples) :

- **CookieServlet** → Stores info on the ***client side*** (browser).

- **SessionServlet1 & 2** → Stores info on the ***server side***, identified by JSESSIONID cookie.

1.SumInitServlet.java

1. How to create :

- *Project -> source package -> new package -> servlet ->Configure Servlet Deployement ->new add Name and Value -> Finish*

2. Purpose of SumInit Servlet :

=> To demonstrate how to use servlet initialization parameters (initParams) with @WebServlet.

=> These parameters act like configuration values (instead of hardcoding values inside the servlet).

=> Example here → two numbers x=40 and y=34 are defined as init parameters, and the servlet calculates their sum.

---

1. MyFilter.java & 2.AnotherFilter.java

1. How to create Filter in java Apache netbeans 25 :

- *First create filter package -> new -> other -> Web -> Filter -> Filter name(MyFilter) -> (if Filter name and Applies to add) else -> Finish*

2. What is a Filter in Servlets?

- A Filter is a component in Java EE (Jakarta EE) that can intercept requests and responses before they reach a servlet or JSP.
- Filters are not servlets themselves, but they work in the middleware layer between the client (browser) and the servlet.

=> They are mostly used for:

- **Pre-processing (before request goes to servlet)**
- **Post-processing (after servlet generates response)**

◆ **Common Purposes of Filters :**

1. Logging & Debugging :-

Track requests and responses (headers, parameters, execution time).

2. *Authentication & Authorization :-*

Check if a user is logged in before allowing access to certain servlets/pages.

3. *Input Validation & Sanitization :-*

Prevent SQL Injection or XSS attacks by filtering inputs.

4. *Compression :-*

GZIP compress the response before sending to client.

5. *Character Encoding :-*

Set UTF-8 encoding for requests/responses globally.

6. *Centralized Code Execution :-*

Instead of repeating code in every servlet, keep it once in a filter.

3. AnotherFilter.java works :

-> You annotated it with:

@WebFilter(filterName = "AnotherFilter", urlPatterns = {"/*"})

This means the filter will run for all URLs (/*).

Your filter methods:

doBeforeProcessing() → runs before servlet executes.

System.out.println("I am Another Filter as Request");

doAfterProcessing() → runs after servlet response is generated.

System.out.println("I am AnotherFilter as Response");

chain.doFilter(request, response) → hands over control to the next filter or servlet.