Enterprise Java

Java Server Pages

How to Create :-

- Java with maven -> Web Application -> Finish
- In Source Packages -> new package -> named -> jsp -> Employee.java(Java Class)
- Project Right Click -> new JSP

1. SamplePractise.jsp :-

- For Sample Practise of JSP

2. EmpForm.jsp :-

- Create an Html Design for frontend Logic

3. Emp.jsp :-

- To get value for variables like Empno , Employee name , Salary that put in EmpFom.

4. Employee.java :-

- This **Employee JavaBean** is created so JSP pages can treat an employee as an **object with properties** (empno, ename, salary).
- It provides **getters/setters for automatic data binding** (<jsp:setProperty property="*"/>) and a **validate() method** to check input before displaying orforwarding.
- In short → it separates **business logic (validation, rules)** from JSP view, making the application **clean, reusable, and maintainable**.

What is JSP?

- **JSP = JavaServer Pages**
- It is basically **HTML + Java code embedded inside**.
- Its main **aim/purpose**: **Presentation Layer** (UI).
- It allows you to write Java directly in a web page (<% ... %>), so that the server can generate **dynamic HTML** before sending it to the browser.

So while **Servlets** are Java classes that handle HTTP requests and responses, **JSP** is a page that makes writing HTML easier (with embedded Java).

How does JSP work internally?

When you request SamplePractise.jsp:

1. Server (Payara/Tomcat/GlassFish) **converts JSP into a Servlet class**.
2. That Servlet is compiled to .class and executed.
3. Output (HTML) is sent to browser.

So **JSP = Servlet in disguise**, but with more HTML-friendly syntax.

Where does JSP fit compared to Servlet & EJB?

Think of the architecture like this:

- **Browser (Client)** → sends HTTP request.
- **Servlet (Controller)** → handles request, calls EJB for business logic.
- **EJB (Business Logic)** → does heavy lifting (transactions, DB, rules).
- **JSP (View)** → generates HTML response to show to user.