Java Web Application Architecture – Interview Notes

1. Why We Use Maven Instead of Dynamic Web Project

In a Java web application we need many external libraries:

- MySQL JDBC Driver
- Spring
- Hibernate
- Logging libraries, etc.

If we use a **Dynamic Web Project**, we must:

- Manually download `.jar` files
- Add them to the project
- Manage versions ourselves

This is fragile and unscalable.

Maven solves this

Maven is a **build and dependency management tool**.

You simply declare dependencies in:

```
pom.xml
```

Example:

```
<dependency>
   <groupId>mysql</groupId>
   <artifactId>mysql-connector-j</artifactId>
   <version>8.0.33</version>
</dependency>
```

Maven will:

- Download the library
- Resolve versions
- Add it to the project classpath

This is how all enterprise Java projects work.

---

2. Why We Use JSP Instead of HTML

HTML is **static**.
It cannot execute Java code.

JSP (Java Server Pages) is:

HTML + Java

JSP allows:

- Dynamic data rendering
- Displaying database values
- Using Java objects inside pages

So:

- **Servlet** → processing
- **JSP** → display

---

3. Role of Servlet

A **Servlet** is a Java class that:

- Accepts HTTP requests
- Processes business logic
- Interacts with the database
- Sends results to JSP

Servlet is the **Controller**.

---

4. MVC Architecture in Java Web Apps

Enterprise Java web apps follow **MVC**:

Layer	Technology	Responsibility
Model	Java classes + JDBC	Holds data, talks to DB
View	JSP	Displays data
Controller	Servlet	Handles request, coordinates everything

Flow:

```
Browser → Servlet → Model → Servlet → JSP → Browser
```

---

5. User Flow in This Application

User enters:

```
Alien ID → Click Submit
```

Form in `index.jsp`:

```
<form action="getAlien">
   <input type="text" name="aid"/>
   <input type="submit"/>
</form>
```

This sends:

```
getAlien?aid=9
```

---

6. URL Mapping Between JSP and Servlet

Servlet is mapped like this:

```
@WebServlet("/getAlien")
public class GetAlienController extends HttpServlet
```

Or in `web.xml`:

```
<servlet-mapping>
   <url-pattern>/getAlien</url-pattern>
   <servlet-name>GetAlienController</servlet-name>
</servlet-mapping>
```

So:

```
action="getAlien"
→ calls GetAlienController
```

---

7. What Servlet Does

Servlet:

1. Reads request parameter

```
int aid = Integer.parseInt(request.getParameter("aid"));
```

2. Calls model (JDBC)

```
Alien alien = dao.getAlien(aid);
```

3. Sends data to JSP

```
request.setAttribute("alien", alien);
request.getRequestDispatcher("showAlien.jsp").forward(request, response);
```

8. JDBC Role

JDBC connects Java to MySQL.

DAO (Model layer):

```
Connection con = DriverManager.getConnection(...)
PreparedStatement ps = con.prepareStatement("SELECT * FROM alien WHERE
aid=?");
ps.setInt(1, aid);
ResultSet rs = ps.executeQuery();
```

This fetches data and returns a Java object.

---

9. Why This Architecture Is Used in Industry

This design gives:

- **Loose coupling**
- **Separation of concerns**
- **Scalability**
- **Testability**
- **Maintainability**

Servlet does not know SQL
JSP does not know business logic
DAO does not know UI

This is professional enterprise design.

One-Line Interview Answer

In Java web applications we use Maven for dependency management, Servlets as Controllers, JSP for Views, JDBC for database access, and follow the MVC architecture to achieve separation of concerns, scalability, and maintainability.
