

Session 3

Maven and frameworks for J2EE

Maven manages dependencies and the build so that Spring and Hibernate can be added by editing one file. This session sets up the project skeletons every later session builds on, and introduces inversion of control.

All four questions work on one Maven project, `student-admission`, which Sessions 4, 5 and 6 keep extending. Sessions 3 to 6 use classic Spring MVC 6 deployed as a WAR on Tomcat 10.1 (the manual calls this a “J2EE application”); Session 7 switches the same project to Spring Boot.

Objectives

✗ Do not copy. Read for understanding and the viva

- Complete questions 13 to 16 of the manual: maven and frameworks for j2ee
- Prepare the deliverable before the lab and finish it during the session
- Be ready to explain every step in the viva

Questions Covered

✗ Do not copy. Read for understanding and the viva

Question	Requirement	Status
Q13	Create a Maven-based project using Spring Initializr	Complete
Q14	Create a Maven-based J2EE application for Spring MVC	Complete
Q15	Create a Maven-based J2EE application for Hibernate and JPA. Use the same database...	Complete
Q16	Define a new implementation of Teacher Interface for his/her favourite Course in...	Complete

Preparation

✗ Do not copy. Read for understanding and the viva

- Generate a project at start.spring.io with the Spring Web dependency, then import it into the IDE as a Maven project.
- Know what `pom.xml` contains: groupId, artifactId, dependencies, plugins. Be ready to explain what Maven downloads and where (`~/.m2`).
- Inversion of control: the container creates the Teacher implementation and hands it to your code; you never call `new`.

Question 13

Problem Statement

■ Write in lab record

Create a Maven-based project using Spring Initializr.

Solution

■ Write in lab record

Steps

1. Install JDK 17, Maven 3.9 and Eclipse IDE for Enterprise Java (or NetBeans / IntelliJ). Check with `java -version` and `mvn -v`.
2. Open start.spring.io and fill the form: Project **Maven**, Language **Java**, Spring Boot **latest stable 3.x**, Group `com.ignou.lab`, Artifact `student-admission`, Name `student-admission`, Package name `com.ignou.lab.admission`, Packaging **War**, Java **17**.
3. Click **Add dependencies** and pick **Spring Web**. Click **Generate**; `student-admission.zip` downloads.
4. Unzip into your workspace. Look at the tree: `pom.xml`, `mvnw`, `src/main/java/com/ignou/lab/admission/StudentAdmissionApplication.java` and `ServletInitializer.java`, `src/main/resources/application.properties`, `src/test/java`.
5. Eclipse: File → Import → Maven → Existing Maven Projects → Root Directory = the unzipped folder → Finish. Maven downloads every dependency into `~/.m2/repository` (internet needed;

wait for the progress bar in the bottom right).

6. Build once from a terminal in the project folder: `mvn -q package`. `target/student-admission-0.0.1-SNAPSHOT.war` appears.

Program

The generated `pom.xml`, unchanged:

pom.xml (generated by Spring Initializr)

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven
4     <modelVersion>4.0.0</modelVersion>
5     <parent>
6         <groupId>org.springframework.boot</groupId>
7         <artifactId>spring-boot-starter-parent</artifactId>
8         <version>3.3.5</version>
9         <relativePath/> 
10    </parent>
11    <groupId>com.ignou.lab</groupId>
12    <artifactId>student-admission</artifactId>
13    <version>0.0.1-SNAPSHOT</version>
14    <packaging>war</packaging>
15    <name>student-admission</name>
16    <description>Lab project for Student Admission</description>
17    <properties>
18        <java.version>17</java.version>
19    </properties>
20    <dependencies>
21        <dependency>
22            <groupId>org.springframework.boot</groupId>
23            <artifactId>spring-boot-starter-web</artifactId>
24        </dependency>
25        <dependency>
26            <groupId>org.springframework.boot</groupId>
27            <artifactId>spring-boot-starter-tomcat</artifactId>
28            <scope>provided</scope>
29        </dependency>
30        <dependency>
31            <groupId>org.springframework.boot</groupId>
32            <artifactId>spring-boot-starter-test</artifactId>
33            <scope>test</scope>
34        </dependency>
35    </dependencies>
36    <build>
37        <plugins>
38            <plugin>
39                <groupId>org.springframework.boot</groupId>
40                <artifactId>spring-boot-maven-plugin</artifactId>
41            </plugin>
42        </plugins>
```

```
43     </build>
44 </project>
```

Output

Expected result, checked by reading (no JDK or Maven is available where this page was written):

```
1 $ mvn -q package
2 $ ls target
3 classes generated-sources maven-archiver student-admission-0.0.1-SNAPSHOT
```

A full `mvn package` ends with `BUILD SUCCESS` and the time taken. Eclipse shows the project with a small “M” decorator and a `Maven Dependencies` library.

Explanation

- Spring Initializr writes a correct `pom.xml` so nobody types dependency coordinates by hand. The `parent` block imports Spring Boot’s dependency management: child dependencies need no version numbers.
- `groupId` is the organisation, `artifactId` the project, together with `version` they name the WAR in `~/.m2` and in `target/`.
- `packaging` is `war` because the manual deploys on Tomcat. The `spring-boot-starter-tomcat` dependency is `provided`, so the WAR does not carry a second Tomcat.
- Maven phases: `compile` → `test` → `package` → `install`. `mvn package` runs everything up to and including packaging.

Question 14

Problem Statement

■ Write in lab record

Create a Maven-based J2EE application for Spring MVC.

Solution

■ Write in lab record

Steps

1. Keep the folder from Q13. Replace `pom.xml` with the listing below (plain Spring 6, no Boot parent). Delete `StudentAdmissionApplication.java`, `ServletInitializer.java`, `application.properties` and the `src/test` folder; they belong to Spring Boot and return in Session 7.
2. Right-click the project → Maven → Update Project → OK. The `spring-webmvc` jar and its dependencies download.
3. Create `WebAppInitializer.java` and `WebConfig.java` in `com.ignou.lab.admission`, and `HomeController.java` in `com.ignou.lab.admission.web`.
4. Create the folder `src/main/webapp/WEB-INF/views/` and put `home.jsp` in it.
5. Build: `mvn -q package` → `target/student-admission.war` (the `finalName` in the pom drops the version).
6. Deploy: copy the WAR to `$CATALINA_HOME/webapps/` and start Tomcat 10.1 with `bin/startup.sh` (or in Eclipse: Servers view → Tomcat v10.1 → Add and Remove → add the project → Start).
7. Open <http://localhost:8080/student-admission/>.

Program

- Lab record: every tab is one file of the answer. Write all of them.

pom.xml

pom.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven
4     <modelVersion>4.0.0</modelVersion>
5     <groupId>com.ignou.lab</groupId>
6     <artifactId>student-admission</artifactId>
7     <version>0.0.1-SNAPSHOT</version>
8     <packaging>war</packaging>
9     <name>student-admission</name>
10    <description>Lab project for Student Admission using Spring MVC (classic,
11
12    <properties>
13        <maven.compiler.release>17</maven.compiler.release>
14        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15        <spring.version>6.1.14</spring.version>
16    </properties>
17
18    <dependencies>
19        <!-- Spring MVC: DispatcherServlet, controllers, form tags -->
20        <dependency>
21            <groupId>org.springframework</groupId>
22            <artifactId>spring-webmvc</artifactId>
23            <version>${spring.version}</version>
24        </dependency>
25        <!-- Servlet API: Tomcat 10.1 provides it at run time -->
26        <dependency>
27            <groupId>jakarta.servlet</groupId>
28            <artifactId>jakarta.servlet-api</artifactId>
29            <version>6.0.0</version>
30            <scope>provided</scope>
31        </dependency>
32        <!-- JSTL 3.0 (Jakarta namespace) for c:forEach and c:out in JSPs -->
33        <dependency>
34            <groupId>jakarta.servlet.jsp.jstl</groupId>
35            <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
36            <version>3.0.0</version>
37        </dependency>
38        <dependency>
39            <groupId>org.glassfish.web</groupId>
40            <artifactId>jakarta.servlet.jsp.jstl</artifactId>
41            <version>3.0.1</version>
42        </dependency>
43    </dependencies>
```

```
44
45     <build>
46         <finalName>student-admission</finalName>
47         <plugins>
48             <plugin>
49                 <groupId>org.apache.maven.plugins</groupId>
50                 <artifactId>maven-compiler-plugin</artifactId>
51                 <version>3.13.0</version>
52             </plugin>
53             <plugin>
54                 <groupId>org.apache.maven.plugins</groupId>
55                 <artifactId>maven-war-plugin</artifactId>
56                 <version>3.4.0</version>
57                 <configuration>
58                     <!-- no web.xml: WebAppInitializer registers the DispatcherServlet
59                     <failOnMissingWebXml>false</failOnMissingWebXml>
60                 </configuration>
61             </plugin>
62         </plugins>
63     </build>
64 </project>
```

Output

Expected browser result at `/student-admission/` , checked by reading, not executed:

```
1 Welcome to the Lab Project for Student Admission using Spring MVC
2 Spring Framework version: 6.1.14
3 Server time: 2026-09-26T10:42:07.118
4 Students in the IGNOU database   (link; works after Q15)
```

Tomcat's `catalina.out` shows `Initializing Spring DispatcherServlet 'dispatcher'` followed by `Completed initialization` when the WAR starts.

Explanation

- No `web.xml` : Tomcat calls `WebAppInitializer` because it extends `AbstractAnnotationConfigDispatcherServletInitializer` . It registers a `DispatcherServlet` on `/` backed by `WebConfig` , plus a UTF-8 `CharacterEncodingFilter` so Indian-language names survive form posts.

- `@EnableWebMvc` switches on annotation-driven controllers, `@ComponentScan` finds every `@Controller`, `@Service`, `@Repository` and `@Configuration` under `com.ignou.lab.admission`.
- `InternalResourceViewResolver` turns the string `"home"` into `/WEB-INF/views/home.jsp`. JSPs under `WEB-INF` cannot be opened directly; only a controller can forward to them.
- `jakarta.servlet-api` is `provided`: Tomcat has it. JSTL 3.0 is bundled because Tomcat does not ship it; its tag URI is `jakarta.tags.core`, not the old `java.sun.com` one.

Question 15

Problem Statement

■ Write in lab record

Create a Maven-based J2EE application for Hibernate and JPA. Use the same database created in Exercise no. 5 of Session 1.

Solution

■ Write in lab record

Steps

1. Make sure the `ignou` database and its `student` table from Session 1 exist. If you are starting fresh, run `ignou-student.sql` in MySQL Workbench (Query tab → paste → Execute) or `mysql -u root -p < ignou-student.sql`.
2. Add the three dependencies from `pom-hibernate-fragment.xml` inside the `dependencies` element of `pom.xml`, then Maven → Update Project.
3. Add `src/main/resources/db.properties` with your MySQL user and password.
4. Create `JpaConfig.java` next to `WebConfig.java`, `Student.java` in `com.ignou.lab.admission.entity`, `StudentRepository.java` in `com.ignou.lab.admission.repo`, `StudentController.java` in `com.ignou.lab.admission.web` and `students.jsp` in `WEB-INF/views`.
5. Rebuild and redeploy. Open <http://localhost:8080/student-admission/students>.

Program

- Lab record: every tab is one file of the answer. Write all of them.

pom.xml

pom.xml (Q15 additions)

```
1  <!-- Q15: add inside <dependencies> of pom.xml, then Maven > Update Project
2  <!-- Spring's JPA glue: LocalContainerEntityManagerFactoryBean, JpaTransacti
3  <dependency>
4      <groupId>org.springframework</groupId>
5      <artifactId>spring-orm</artifactId>
6      <version>${spring.version}</version>
7  </dependency>
8  <!-- Hibernate 6 = JPA 3.1 provider; pulls in jakarta.persistence-api →
9  <dependency>
10     <groupId>org.hibernate.orm</groupId>
11     <artifactId>hibernate-core</artifactId>
12     <version>6.6.3.Final</version>
13 </dependency>
14 <!-- MySQL Connector/J 8 (driver class com.mysql.cj.jdbc.Driver) →
15 <dependency>
16     <groupId>com.mysql</groupId>
17     <artifactId>mysql-connector-j</artifactId>
18     <version>8.4.0</version>
19 </dependency>
```

Output

Expected, checked by reading, not executed. Tomcat's console shows the SQL Hibernate ran because `show_sql` is on:

```
1  Hibernate:
2      select
3          s1_0.id,
4          s1_0.address,
5          s1_0.courses,
6          ...
7  from
8      student s1_0
9  order by
10     s1_0.id
```

The browser table at `/students` lists the two seeded rows: `2201234567 Asha Verma ... MCA MCS-218,MCS-220` and `2201234568 RahuL Singh ... MCA MCS-219`. If `db.properties` has a wrong password, deployment fails with `Access denied for user 'root'@'localhost'` in the log.

Explanation

- JPA is the standard API (`jakarta.persistence`); Hibernate is the provider that implements it. Code talks to `EntityManager`, Hibernate translates to MySQL SQL.
- `LocalContainerEntityManagerFactoryBean` scans the `entity` package for `@Entity` classes and builds one `EntityManagerFactory` when Tomcat starts. `JpaTransactionManager` plus `@EnableTransactionManagement` make `@Transactional` methods open and commit a transaction.
- `@PersistenceContext` injects a thread-safe `EntityManager` proxy; each transaction gets its own real one.
- `hibernate.hbm2ddl.auto=update` compares the entity with the table and adds missing columns (here `hostel_required`), so an older Session 1 table keeps working. It never drops or alters existing columns.
- `@Transactional(readOnly = true)` on `findAll` lets Hibernate skip dirty checking; the JSP receives plain entity objects.

Question 16

Problem Statement

■ Write in lab record

Define a new implementation of Teacher Interface for his/her favourite Course in Spring using Inversion of Control and retrieving the information from the new teacher implementation.

Solution

■ Write in lab record

Steps

1. Create package `com.ignou.lab.admission.ioc` with `Teacher.java`, `JavaTeacher.java` and `TeacherApp.java`.

2. Run `TeacherApp` as a plain Java program: right-click → Run As → Java Application, or from the terminal `mvn -q compile exec:java -Dexec.mainClass=com.ignou.lab.admission.ioc.TeacherApp`.
3. Add a second implementation (say `NetworksTeacher` returning MCS-218) and run again to see Spring refuse with `NoUniqueBeanDefinitionException`; remove it or add `@Primary` to one. This is the viva question.

Program

- Lab record: every tab is one file of the answer. Write all of them.

Teacher.java

ioc/Teacher.java

```
1 package com.ignou.lab.admission.ioc;
2
3 public interface Teacher {
4     String getName();
5     String getFavouriteCourse();
6 }
```

Output

Expected console output, checked by reading, not executed:

```
1 Bean class : JavaTeacher
2 Dr. Rao prefers MCS-220 Web Technologies
```

Explanation

- Inversion of control: `TeacherApp` never writes `new JavaTeacher()`. It hands the class to `AnnotationConfigApplicationContext`, the container instantiates it, and the program asks for it by the interface `Teacher`.
- `@Component` marks the class as a bean. Because the request is by interface, swapping in another implementation touches one file only.
- The context is `AutoCloseable`; the try-with-resources closes it and destroys the beans.
- Session 4 injects a `CourseService` into `JavaTeacher` so that the favourite course changes on every call.

Viva Questions

✗ Do not copy. Read for understanding and the viva

- **Q:** What does Maven download and where does it keep it? **A:** Every dependency jar and its transitive dependencies, in the local repository `~/.m2/repository`, keyed by groupId, artifactId and version.
- **Q:** Why is the servlet API marked provided? **A:** Tomcat already has it; shipping a second copy inside the WAR causes class-loading clashes.
- **Q:** Where is web.xml? **A:** Not needed. `WebAppInitializer` uses the Servlet 3+ initializer API to register the `DispatcherServlet` in code.
- **Q:** What is the difference between JPA and Hibernate? **A:** JPA is the specification (annotations and `EntityManager`); Hibernate is one implementation of it and also has its own `Session` API.
- **Q:** What does `hbm2ddl.auto=update` do, and would you use it in production? **A:** Adds missing tables and columns at start-up. No; production schemas are migrated with scripts.
- **Q:** What is inversion of control? **A:** The container creates objects and wires them; application code asks for them instead of constructing them.
- **Q:** Why ask for the bean by interface rather than by `JavaTeacher`? **A:** So the caller does not depend on one implementation; a different teacher can be plugged in without changing `TeacherApp`.

Common Mistakes

✗ Do not copy. Read for understanding and the viva

- Using `javax.servlet` or `javax.persistence` imports. Tomcat 10 and Spring 6 use the `jakarta` packages; the old ones compile and then fail at run time.
- Forgetting Maven → Update Project after editing `pom.xml`, then reporting “cannot resolve import”.
- Writing the JSTL taglib URI as `http://java.sun.com/jsp/jstl/core`; with JSTL 3 on Tomcat 10 it is `jakarta.tags.core`.
- Committing `db.properties` with the real root password. Use a lab user with rights on one database.
- Putting JSPs outside `WEB-INF/views`; they are then reachable without a controller and the view resolver prefix no longer matches.

Session Summary

■ Write in lab record

- Question 13: Spring Initializr project `student-admission` (Maven, WAR, Java 17), imported and built with `mvn package`
- Question 14: classic Spring MVC 6 on Tomcat 10.1: `pom.xml`, `WebAppInitializer`, `WebConfig`, `HomeController`, `home.jsp`
- Question 15: Hibernate 6 / JPA on the Session 1 `ignou` database: `JpaConfig`, `Student` entity, `StudentRepository`, `/students` page
- Question 16: `Teacher` interface with `JavaTeacher` created by the Spring container and read back through the interface

[✎ Edit this page](#)

[< Previous](#)
Session 2

Session 4 [Next >](#)