**Week 3  
Spring Core and Maven  
  
*Excercise 1: Configuring a Basic Spring Application***

A screen shot of a computer

AI-generated content may be incorrect.



A screen shot of a computer

AI-generated content may be incorrect.

A black screen with many small letters

AI-generated content may be incorrect.

***A screen shot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.Exercise 2: Implementing Dependency Injection***

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.

***A screen shot of a computer

AI-generated content may be incorrect.A screen shot of a computer program

AI-generated content may be incorrect.A screen shot of a computer

AI-generated content may be incorrect.Exercise 3: Creating and Configuring a Maven Project***

***Exercise 4:***

***Spring Data JPA - Quick Example using Spring Boot and Hiberante***

**What is Spring Data JPA?**

Spring Data JPA is a part of the Spring Framework that helps you easily interact with databases using Java objects — without writing complex SQL queries.

It is built on top of JPA (Java Persistence API) and uses Hibernate behind the scenes to manage the actual database communication.

**What is Hibernate?**

Hibernate is a tool (ORM - Object Relational Mapping) that maps Java classes to database tables. It makes it easier to save, read, and update data in the database using simple Java code.

**Instead of writing:**

SELECT \* FROM students WHERE id = 1;

**You just do this in Java:**

Student student = studentRepository.findById(1L).get();

**Tools Used in the eg:**

Spring Boot: Simplifies project setup and auto-configuration.

Spring Data JPA: Handles database CRUD operations easily.

Hibernate: Actual engine doing the work behind JPA.

H2 Database: In-memory database for testing.

Lombok (optional): Reduces boilerplate code.

**What’s in the Project?**

1. **Entity class:**

A Java class mapped to a database table

**Example: @**Entity

public class Student {

@Id

private Long id;

private String name;

}

1. **Repository interface:**

No need to write SQL — just extend JpaRepository

Eg: public interface StudentRepository extends JpaRepository<Student, Long> {}

1. **Controller class:**

Handles HTTP requests like POST, GET, PUT, DELETE

Eg: @RestController

@RequestMapping("/students")

public class StudentController { ... }

1. **application.properties:**

Configues DB and Hiberante settings

Eg: spring.datasource.url=jdbc:h2:mem:testdb

spring.jpa.hibernate.ddl-auto=update

**What You Can Do With It**

Save a new student → POST /students

View all students → GET /students

Get one student by ID → GET /students/{id}

Update student → PUT /students/{id}

Delete student → DELETE /students/{id}

***Exercise 5: Difference between JPA, Hibernate and Spring Data JPA***

| **Feature** | **JPA (Java Persistence API)** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **Type** | Specification / Interface | Implementation of JPA | Abstraction over JPA with Spring support |
| **Provided By** | Java (official, part of Java EE / Jakarta EE) | Third-party library (by Red Hat) | Spring Framework |
| **Purpose** | Defines how Java objects should map to DB tables | Actually, performs the object-to-table mapping | Simplifies JPA by reducing boilerplate code |
| **Requires Implementation?** | Yes (you need Hibernate) | No (it is already an implementation) | Uses Hibernate (or any JPA provider) underneath |
| **Configuration** | Requires manual configuration | Requires configuration | Auto-configured by Spring Boot |
| **Boilerplate Code** | Needs EntityManager, transactions, etc. | Slightly easier than raw JPA | Minimal code, just extends JpaRepository. |
| **Common Use** | For standard JPA-based projects | When using Hibernate-specific features | For Spring Boot apps with database interaction |

**JPA (Java Persistence API)**

It's just a set of rules and interfaces.

It defines how Java classes should interact with a database.

But it doesn't do anything itself — it needs a provider like Hibernate.

Think of JPA as a blueprint.

**Hibernate**

It's a popular implementation of JPA.

It converts Java objects into database tables and vice versa.

Can also work standalone without JPA, with more features.

Think of Hibernate as the builder that follows JPA’s blueprint.

**Spring Data JPA**

It's a Spring module that builds on top of JPA and Hibernate.

It provides ready-made repository interfaces like JpaRepository so you don’t need to write boilerplate code.

It auto-generates SQL queries behind the scenes using method names.

Think of Spring Data JPA as a smart helper that simplifies both JPA and Hibernate.