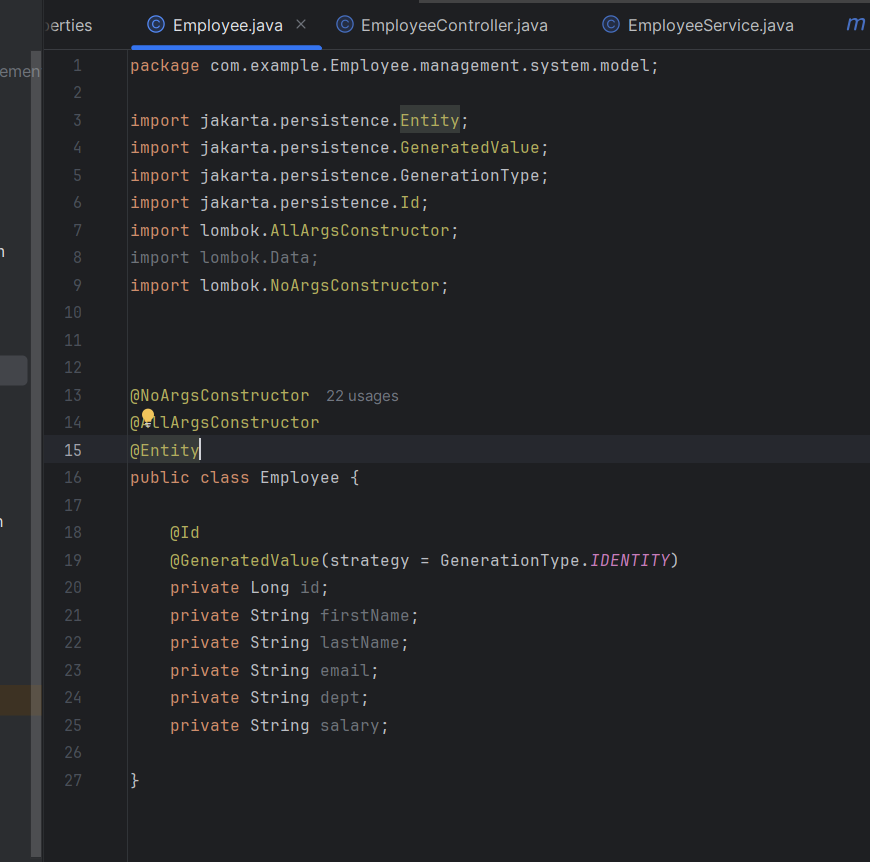
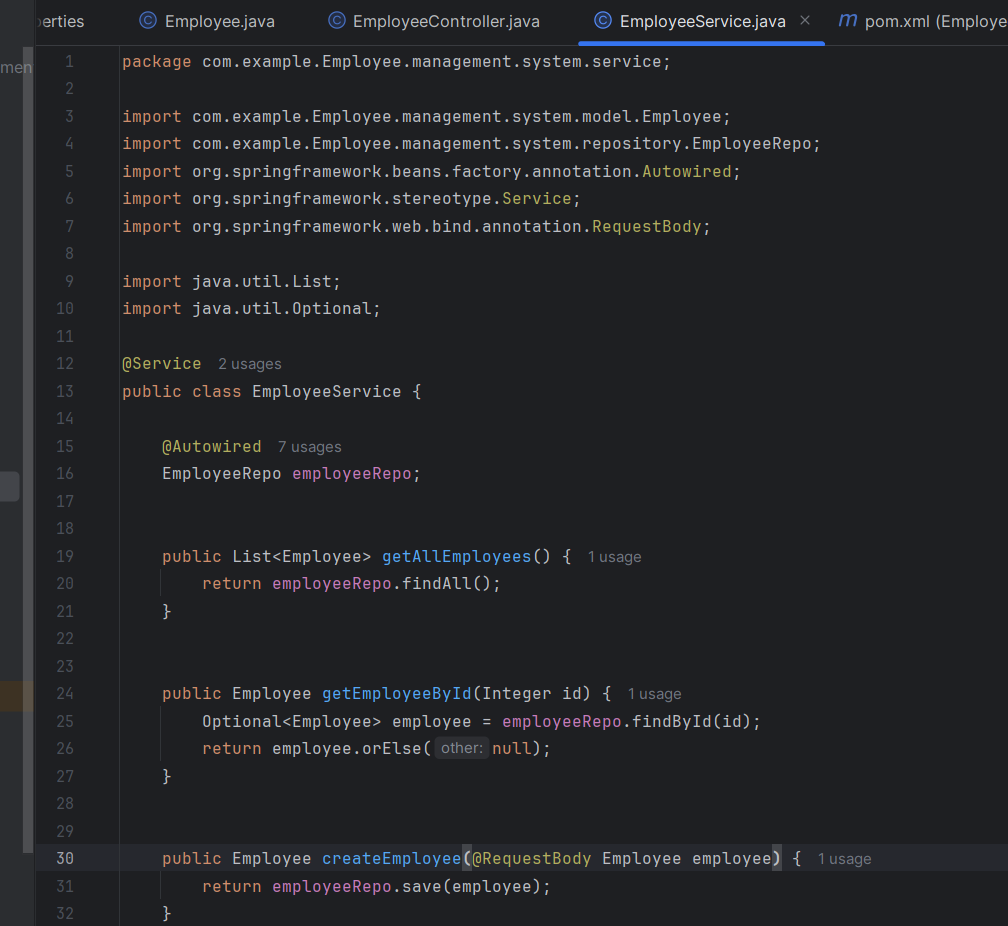
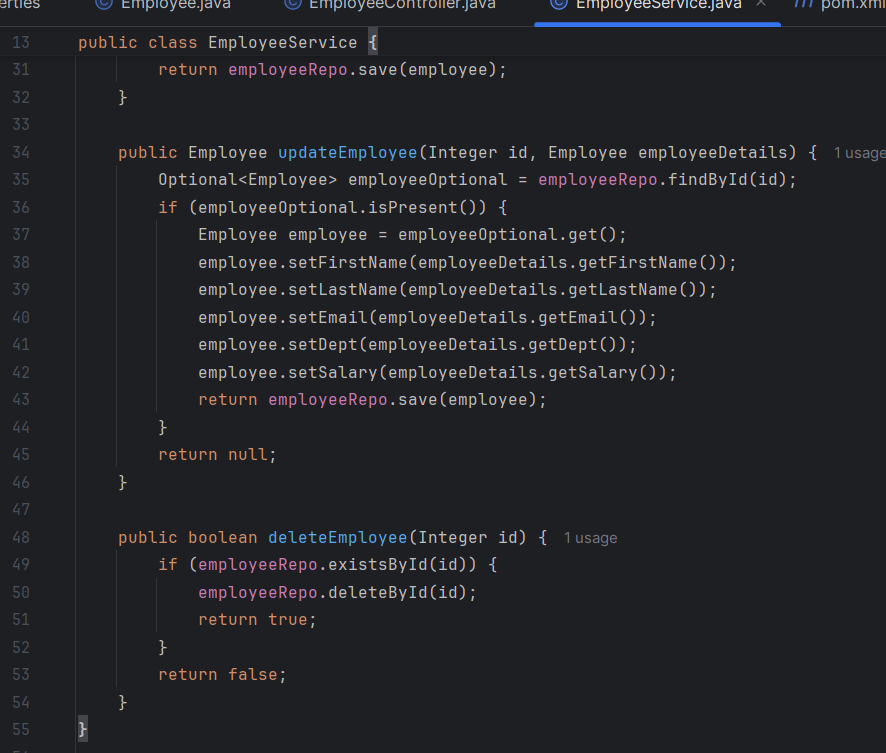
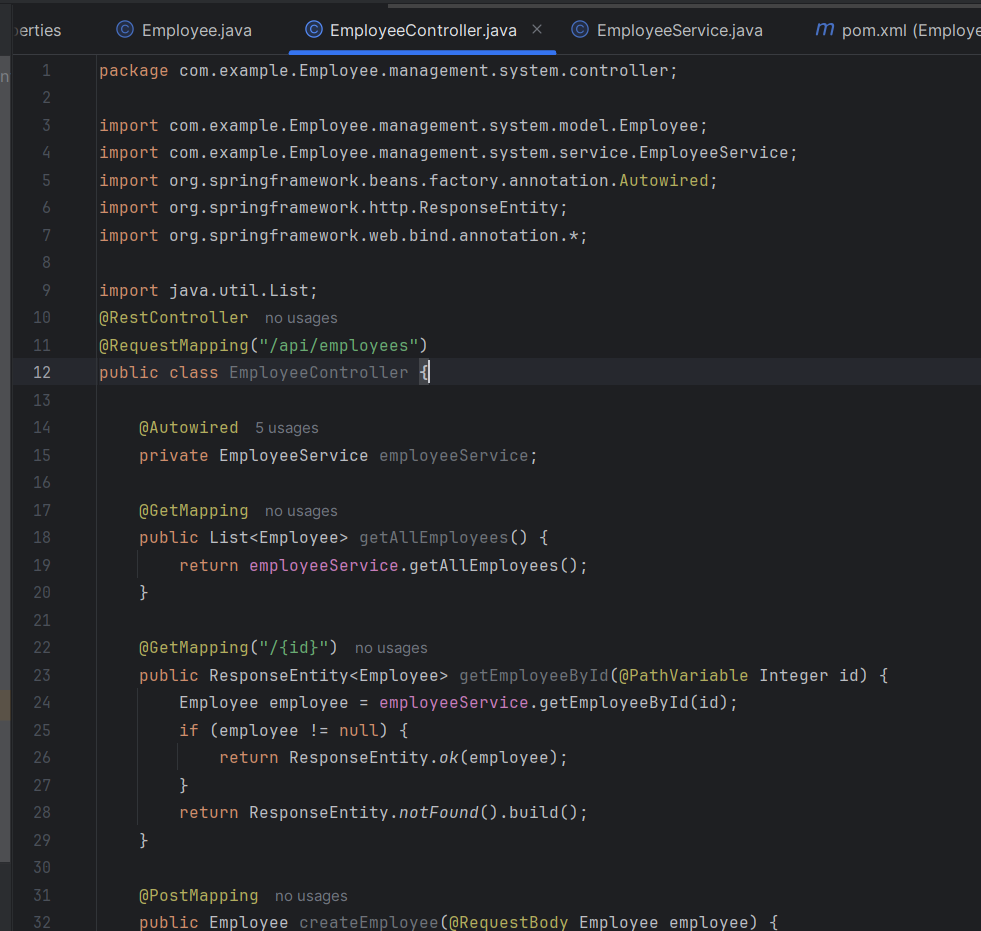
**SPRING DATA JPA WITH SPRING BOOT, HIBERNATE**

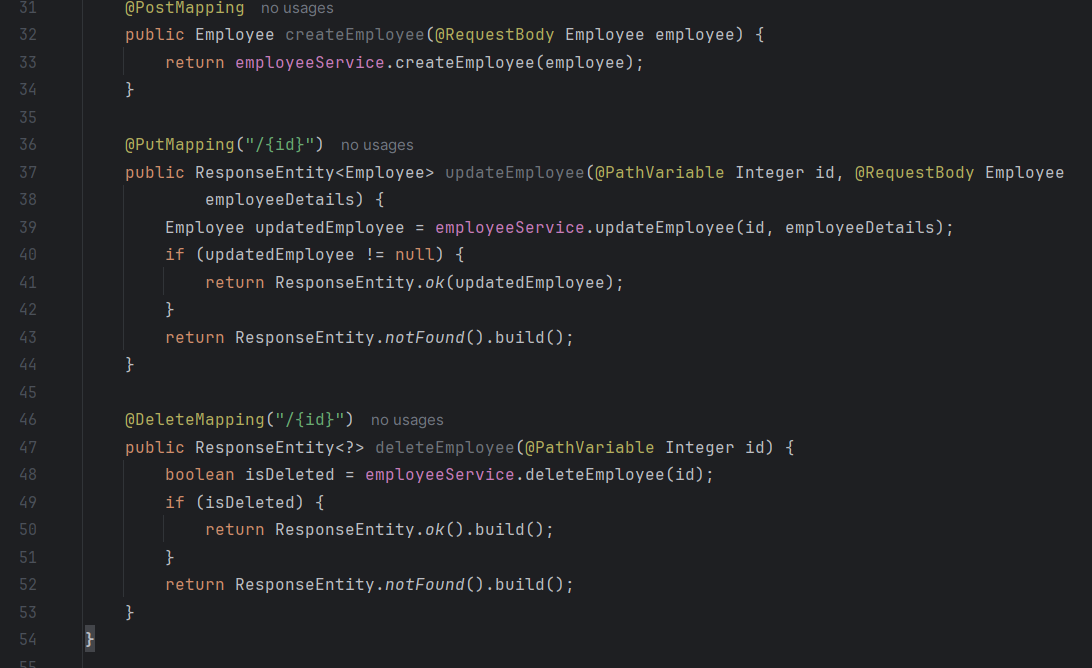
1. **Spring Data JPA - Quick Example:  
     
   employee.java:  
   **

**EmployeeService.java:  
  
**

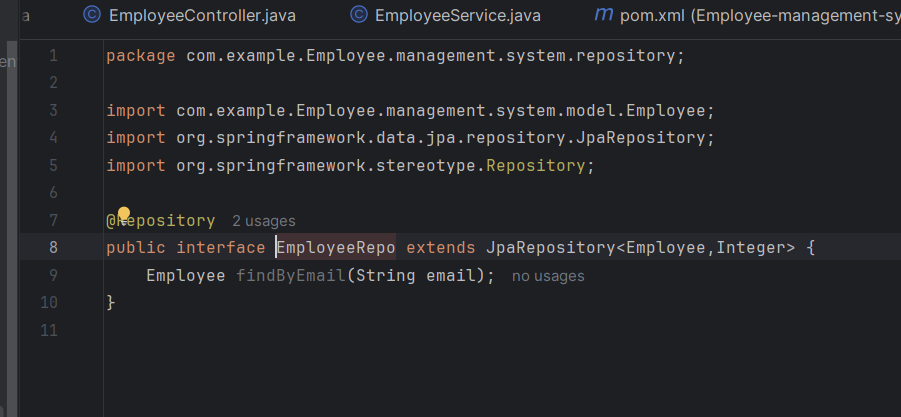
****

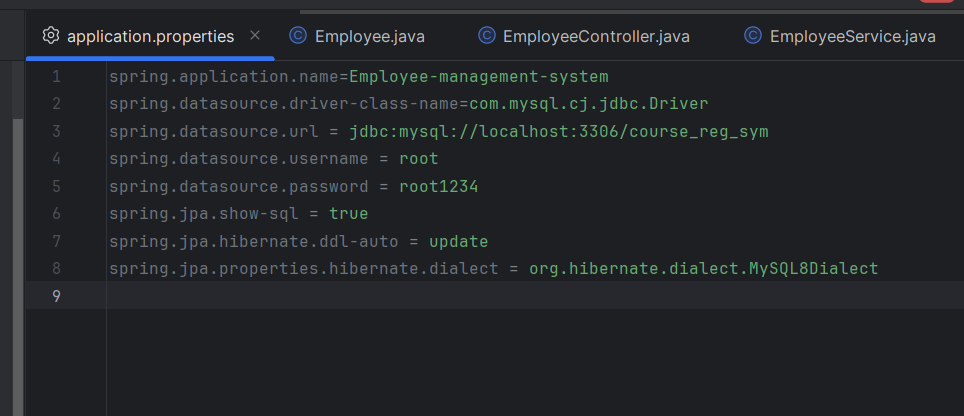
**EmployeeController.java:**

****

****

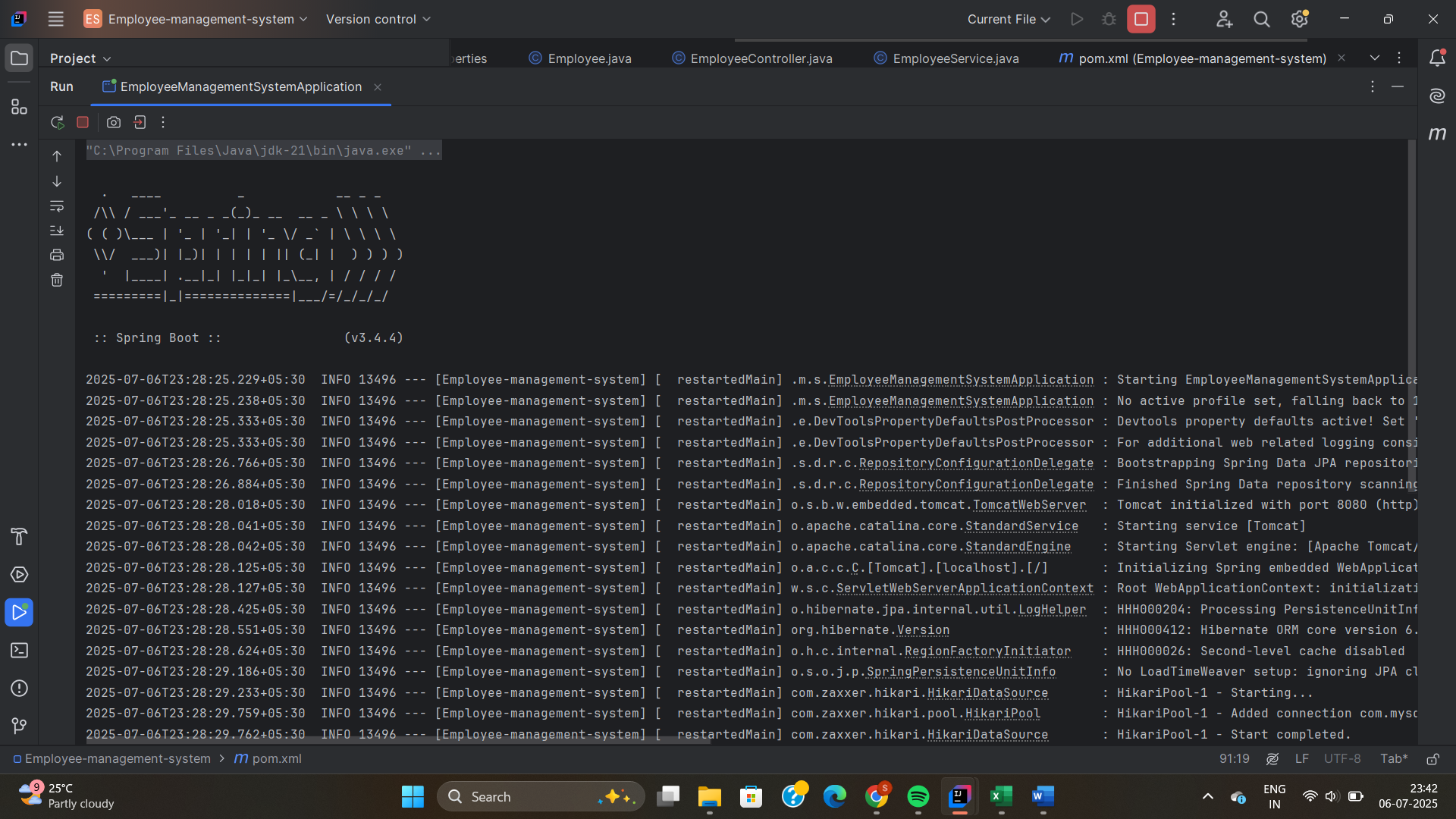
**EmployeeRepository.java:**

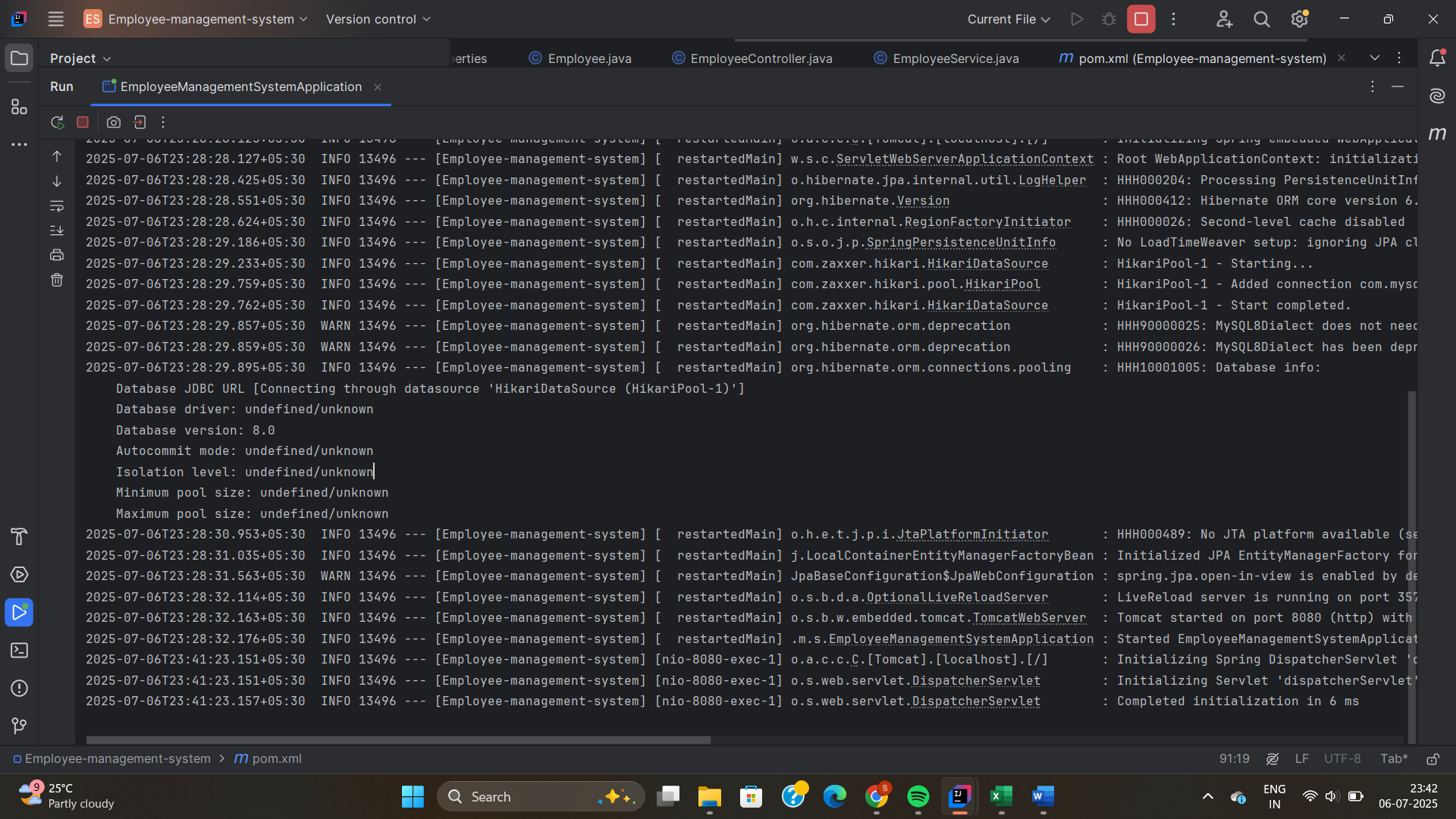
****

**Application.properties:**  


**Pom.xml :**  
<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.4.4</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.example</groupId>  
 <artifactId>Employee-management-system</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>Employee-management-system</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>21</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <configuration>  
 <annotationProcessorPaths>  
 <path>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 </path>  
 </annotationProcessorPaths>  
 </configuration>  
 </plugin>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 <configuration>  
 <excludes>  
 <exclude>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 </exclude>  
 </excludes>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

**Output :**





**2.** **Difference between Java Persistence API, Hibernate and Spring Data JPA:**

While developing backend applications using Java, especially in enterprise or full-stack development, we often need to interact with a database. Instead of writing complex SQL queries manually, we can use Object Relational Mapping (ORM) tools that allow us to interact with the database using Java objects.

This is where JPA, Hibernate, and Spring Data JPA come in. Although they are related to each other, they serve different roles in the Java persistence architecture.

1. Java Persistence API (JPA):

JPA stands for Java Persistence API. It is a specification (just a set of rules or interfaces) that defines how Java objects should be stored and retrieved from relational databases.

JPA was introduced as part of the Java EE (Enterprise Edition) standards to make Java persistence more standardized across different tools.

🔸 Key Characteristics:

* Part of Java EE, not specific to any vendor.
* Defines how to map Java classes to database tables.
* Does not provide any implementation.
* To use JPA, you need a JPA provider (like Hibernate, EclipseLink, etc.).

🔸 Common Annotations Provided by JPA:

| Annotation | Purpose |
| --- | --- |
| @Entity | Specifies that the class is an entity (mapped to a table) |
| @Id | Marks the primary key of the entity |
| @GeneratedValue | Specifies how the primary key is generated |
| @Table, @Column | Customize the table or column mapping |
| @OneToMany, @ManyToOne | Define relationships between entities |

Example:

import jakarta.persistence.\*;

@Entity

@Table(name = "employees")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String department;

}

2. Hibernate — *Implementation of JPA*

Hibernate is an ORM (Object Relational Mapping) tool that provides a working implementation of the JPA specification. It’s one of the most widely used tools for handling database operations in Java applications.

If JPA defines what should happen, Hibernate actually does it.

Key Characteristics:

* Implements JPA specification.
* Can be used with or without JPA.
* Offers additional features not covered in JPA.
* Translates Java code into SQL queries automatically.
* Manages database sessions, connections, transactions, and caching.

Extra Features Provided by Hibernate:

Hibernate Example (without Spring):

Session session = sessionFactory.openSession();

Transaction tx = session.beginTransaction();

Employee emp = new Employee("John", "IT");

session.save(emp);

tx.commit();

session.close();

3. Spring Data JPA — *High-Level Abstraction on Top of JPA and Hibernate*

Spring Data JPA is a Spring Framework module that builds on top of JPA and Hibernate and makes working with databases much easier by removing the need to write most of the boilerplate code.

* We don’t have to write SQL/HQL.
* We don’t even have to write implementations of your repository interfaces.
* It handles most CRUD operations for you automatically.

Key Characteristics:

* Built on top of JPA and Hibernate.
* Part of the larger Spring Data project.
* Focuses on simplifying data access layers.
* Supports automatic query generation by method naming conventions.
* Integrates smoothly with Spring Boot.

Common Interfaces:

| Interface | Description |
| --- | --- |
| CrudRepository | Provides basic CRUD operations |
| JpaRepository | Adds paging, sorting, and more features |
| PagingAndSortingRepository | Adds paging and sorting functionality |

Example:

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

List<Employee> findByDepartment(String department);

}