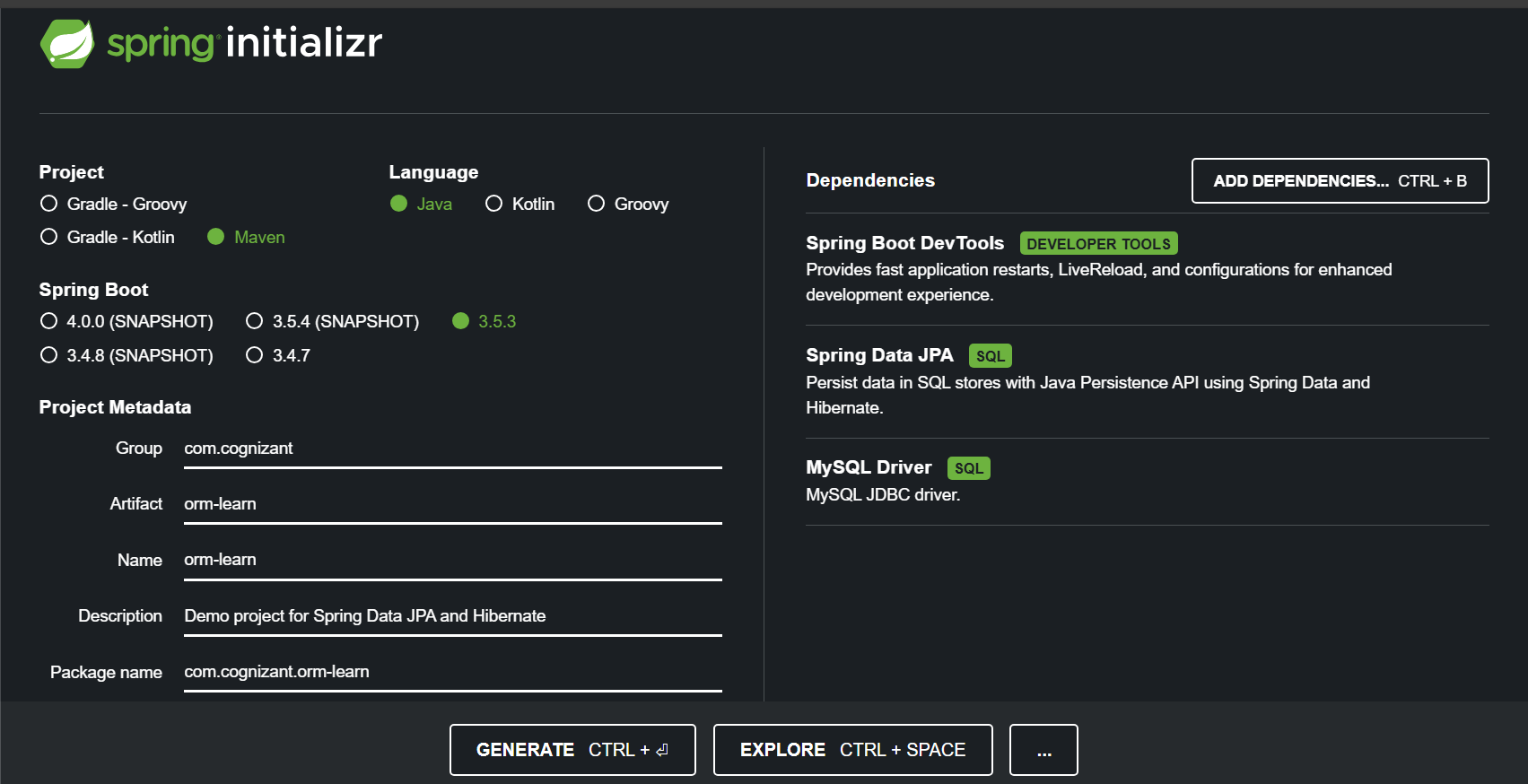
**WEEK 3**

**SPRING DATA JPA WITH HIBERNATE**

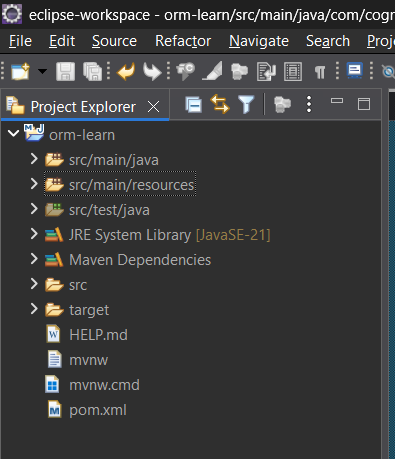
**Hands on 1**

**Spring Data JPA - Quick Example**

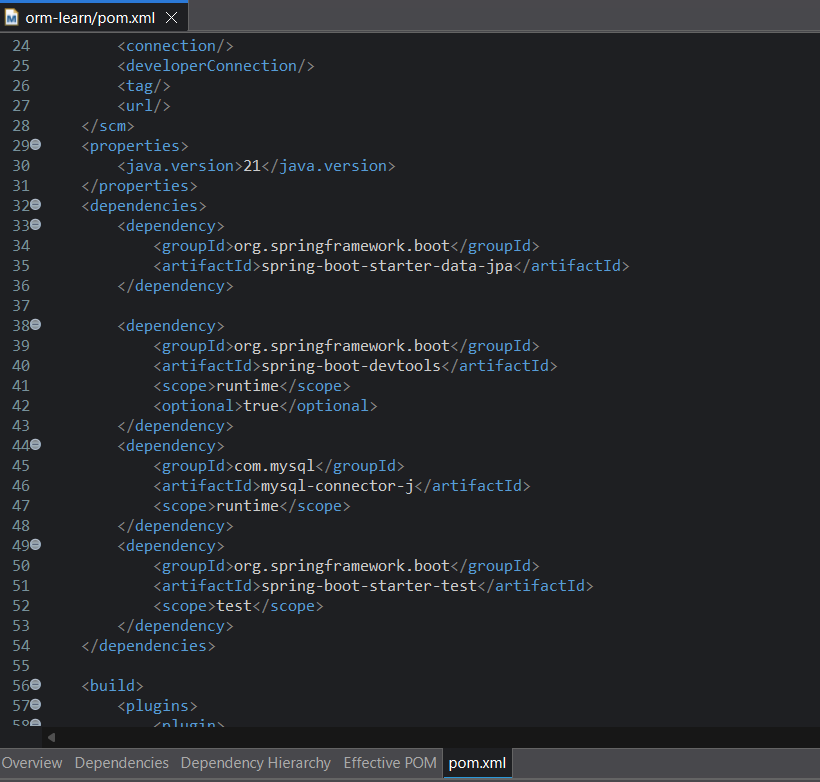
**1. Eclipse Project using Spring Initializr**



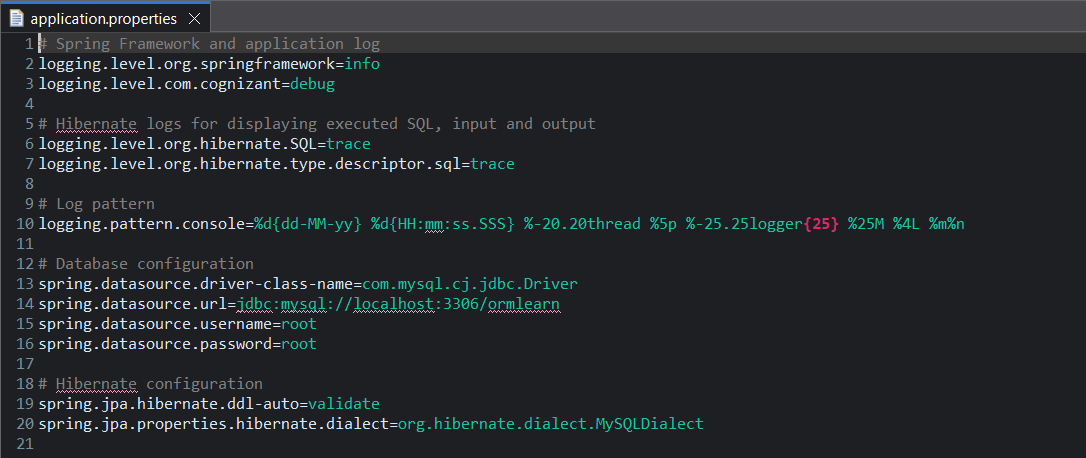
**2. Project Structure**

****

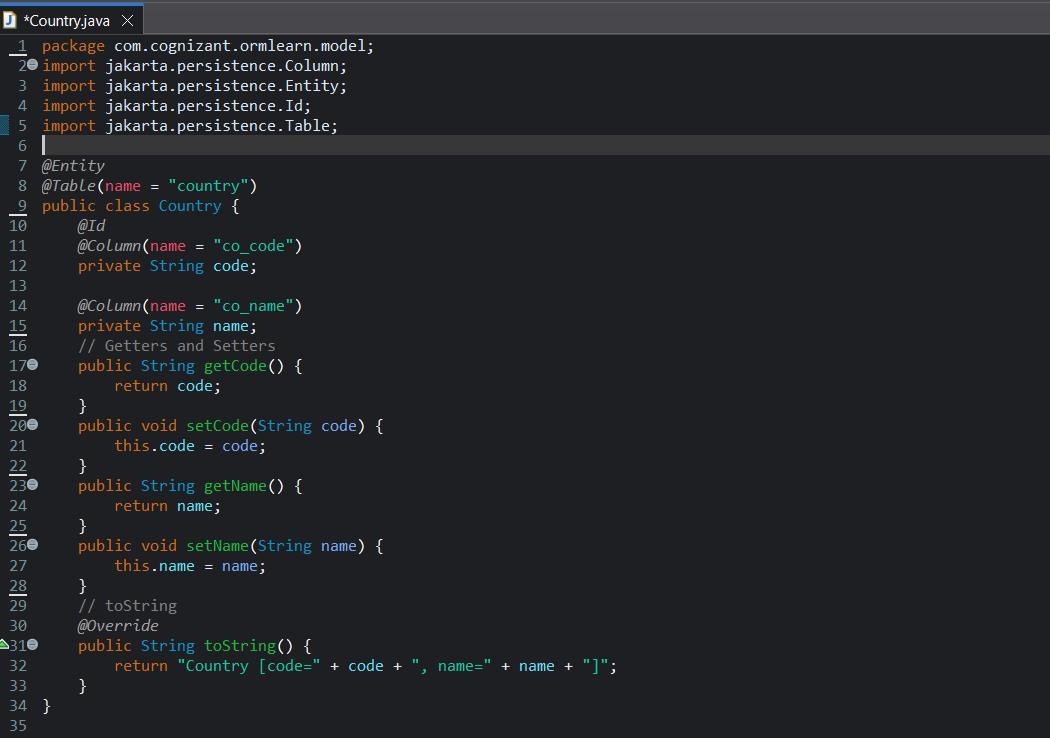
**3. Dependencies in pom.xml**

****

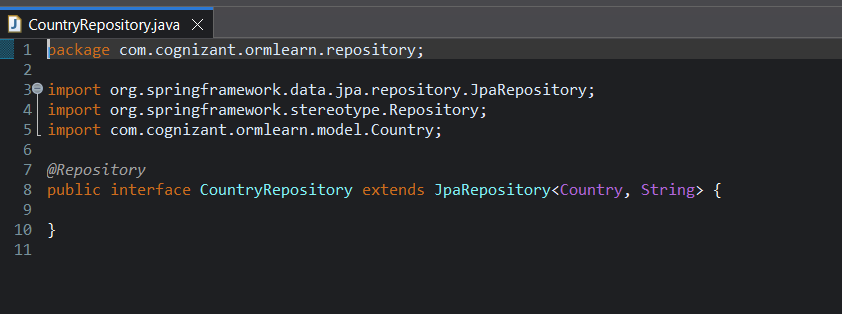
**4. application.properties configuration**

****

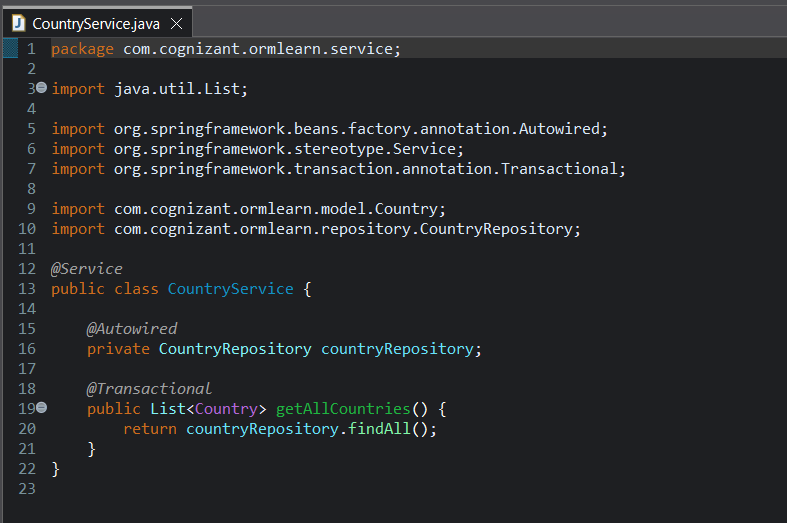
**5. Country Entity Class**

****

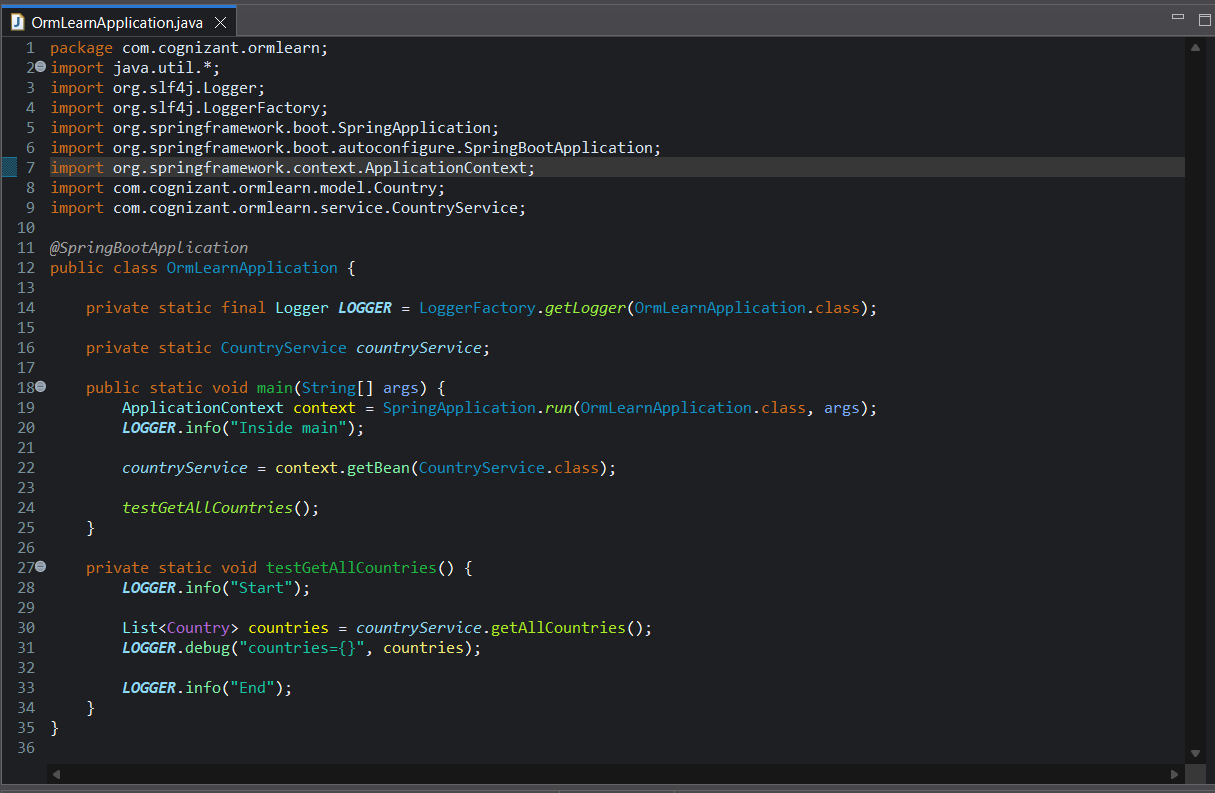
**6. Country Repository Interface**

****

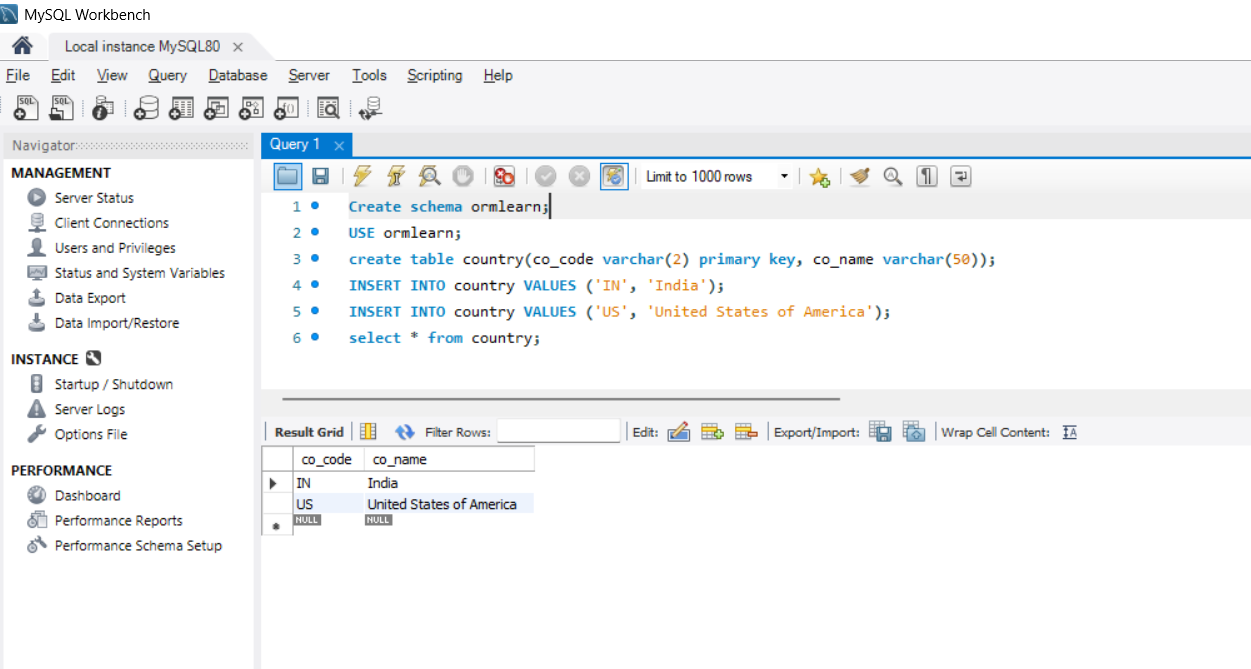
**7. Service Class**

****

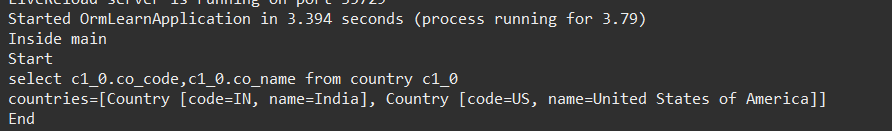
**8. OrmLearnApplication Main Class for testing**

****

**9. country table creation and view in MySQL Database**

****

**10. OUTPUT :**

****

**Hands on 4**

**Difference between JPA, Hibernate and Spring Data JPA**

My Answer:

**JPA (Java Persistence API):**

* It is basically a blueprint containing a set of rules for ORM (object relational mapping), which defines a way for java applications to communicate with relational databases
* It provides annotations for mapping java objects to database tables and also defines how to perform CRUD (create, read, update and delete) operations.
* It is not an implementation, just a specification.

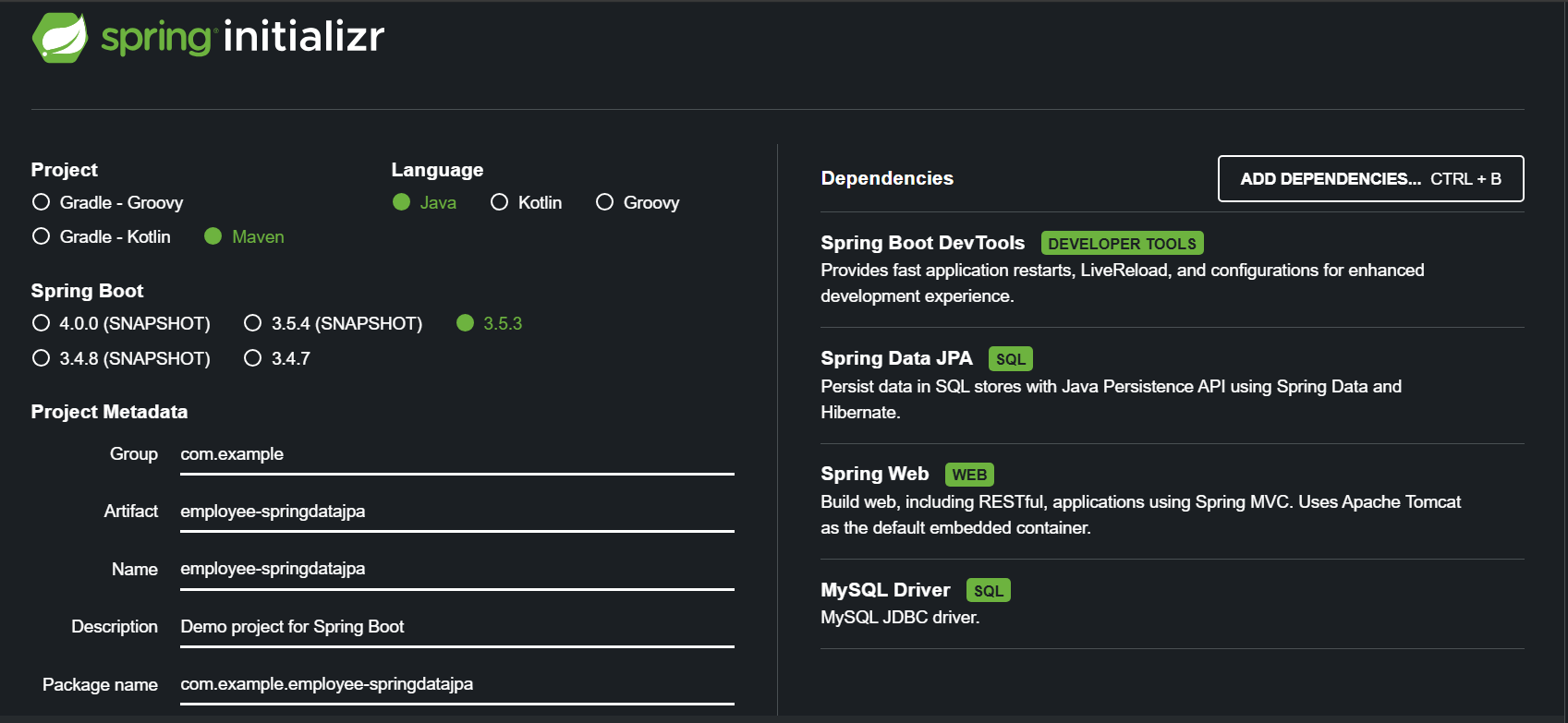
**Hibernate:**

* It is the ORM tool/framework which actually implements the JPA specification.
* It provides the actual code and features to interact with database and handles all operations.

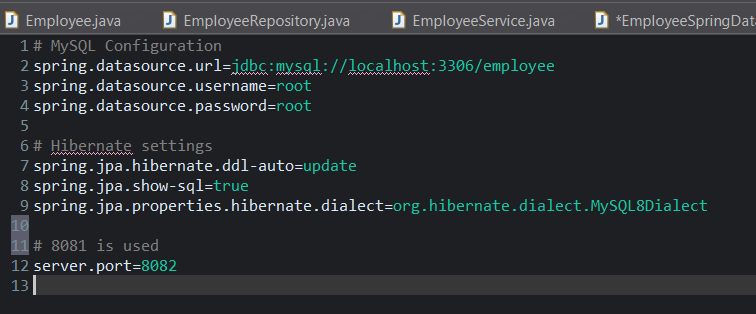
**Spring Data JPA:**

* It can be seen as a abstraction layer which is build on top of JPA and hibernate which simplifies data access.
* It reduces boilerplate code by providing different interfaces like JpaRepository, which automatically generates implementation for common database operations.

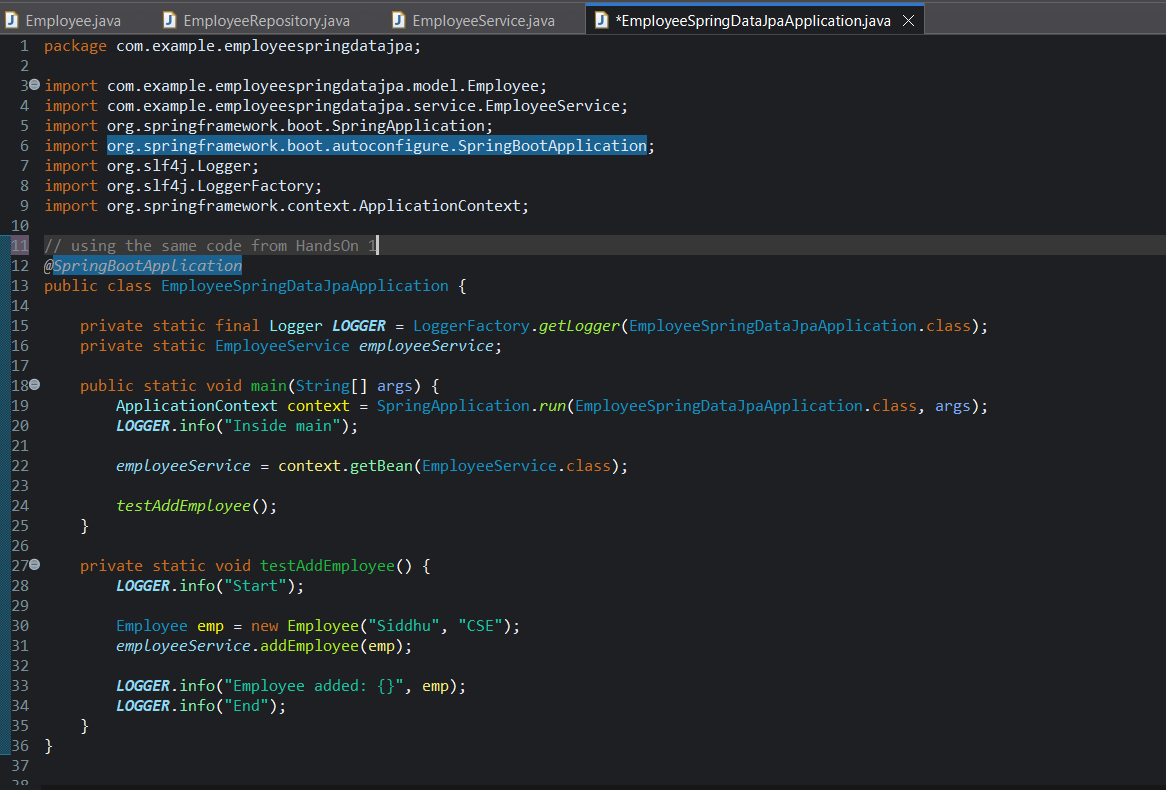
**1. using spring initializr for creating project**

****

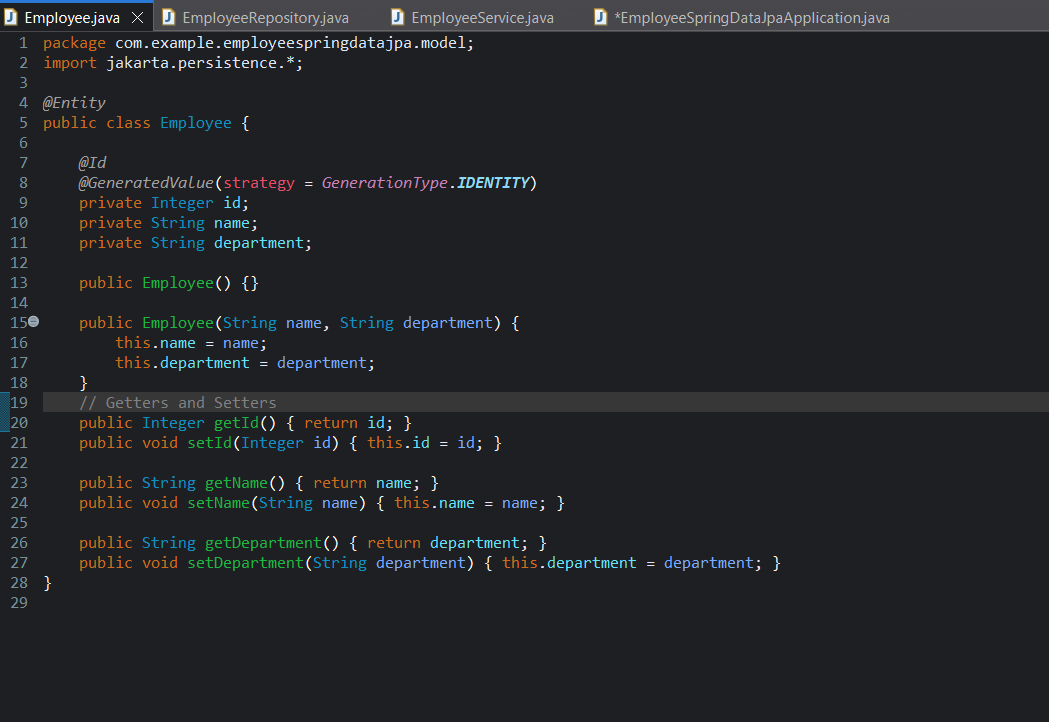
**2. application.properties for configuration:**

****

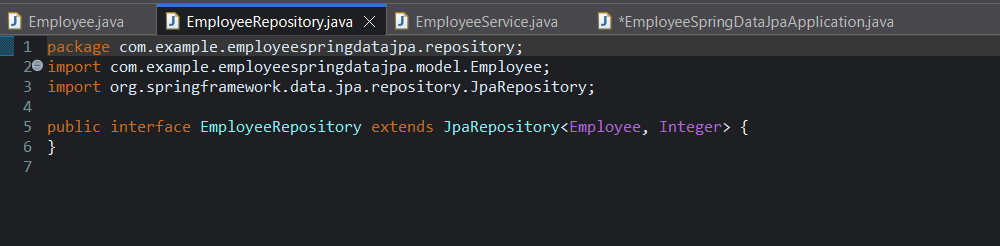
**3. Main Class**

****

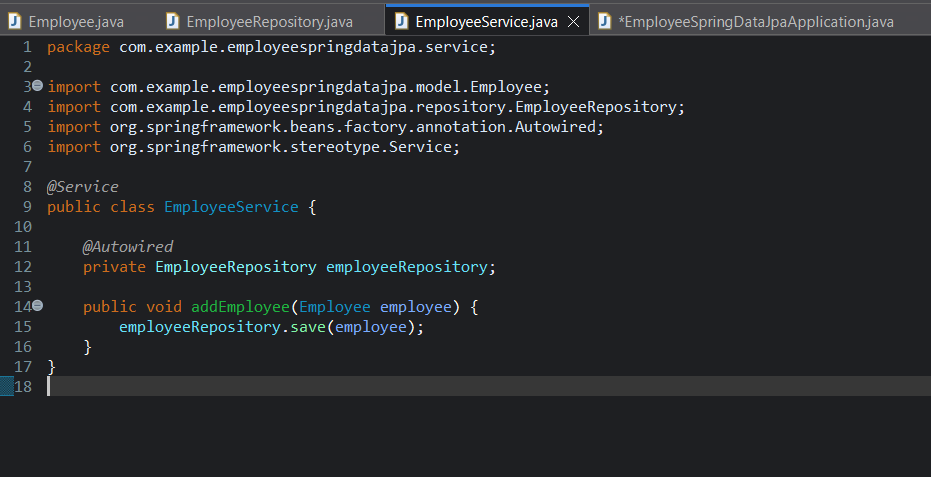
**4. Entity Employee class**

****

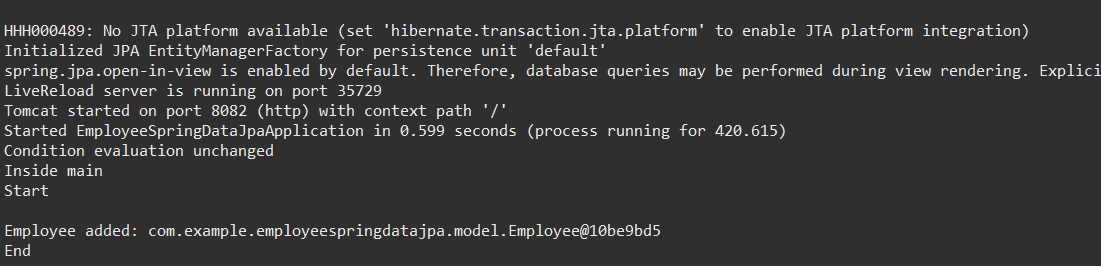
**5. Spring Data JPA EmployeeRespository.java**

****

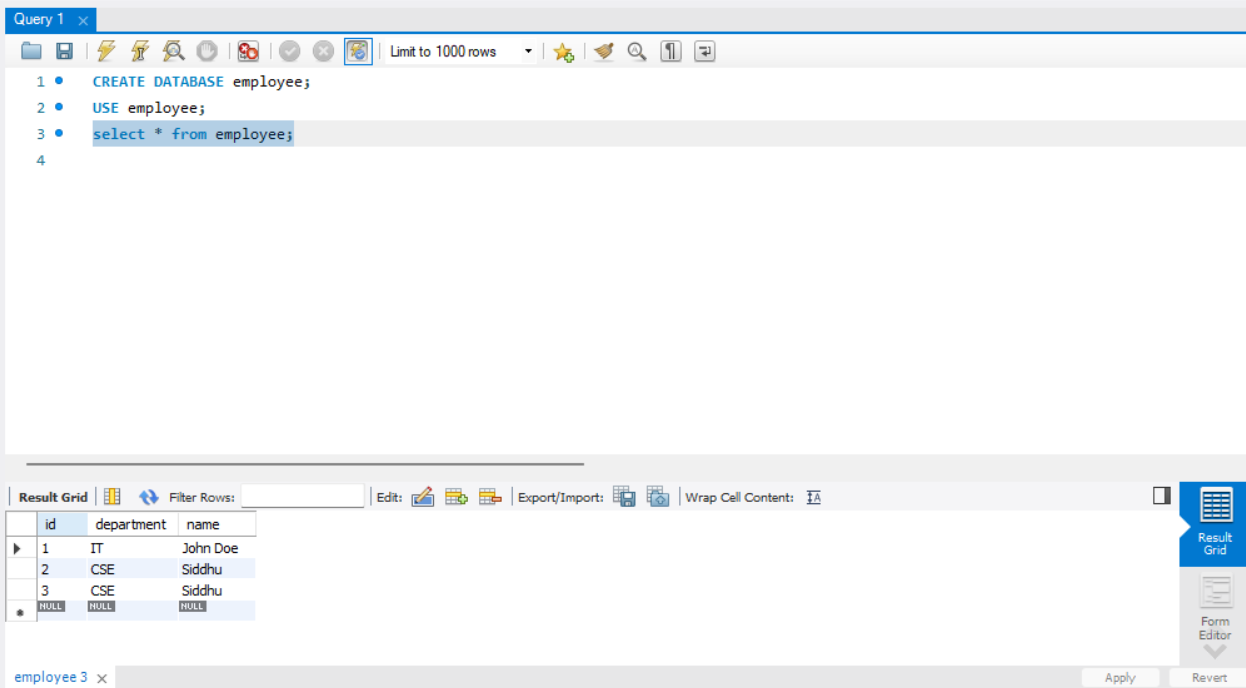
**6. EmployeeService.java**

****

**7. Output:**

****

**8. MySQL TABLE:**

****

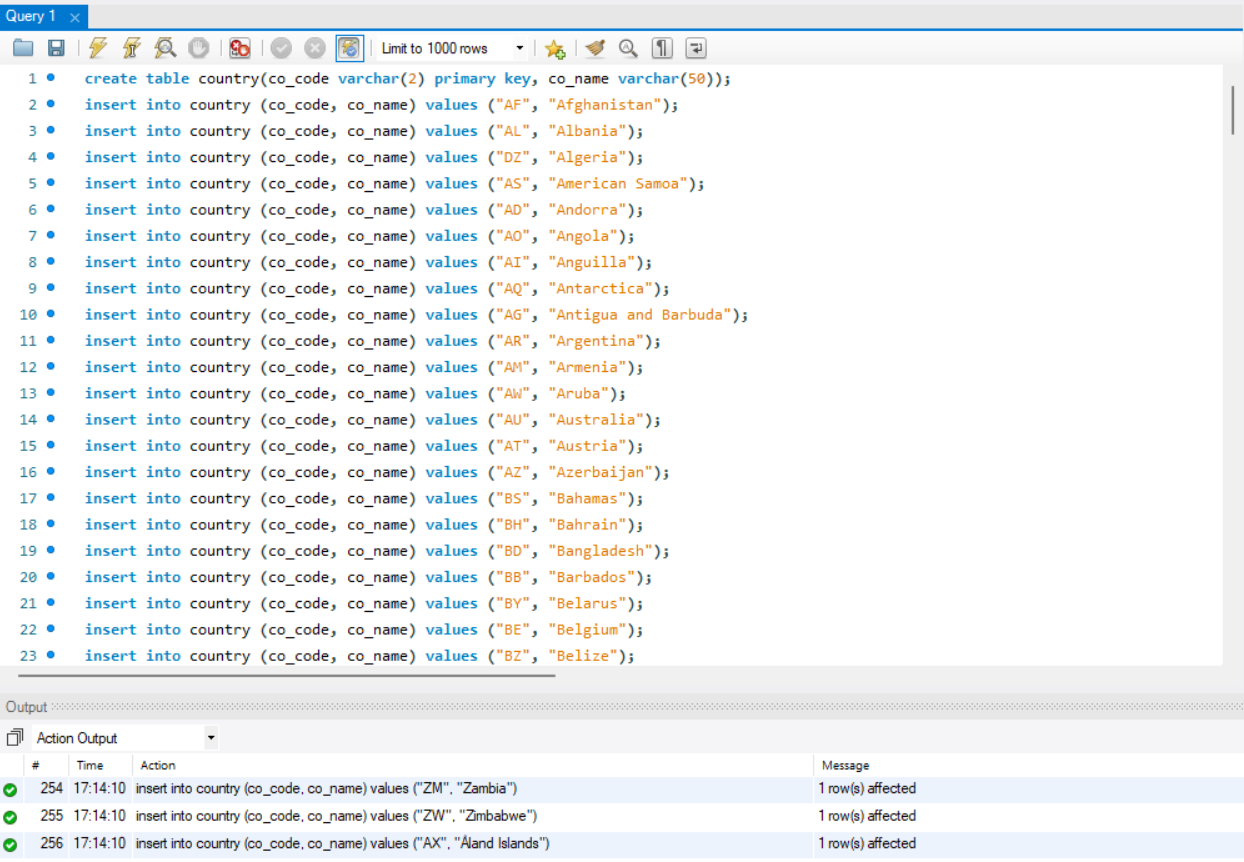
**Hands on 5,6,7,8,9**

**Implement services for managing Country**

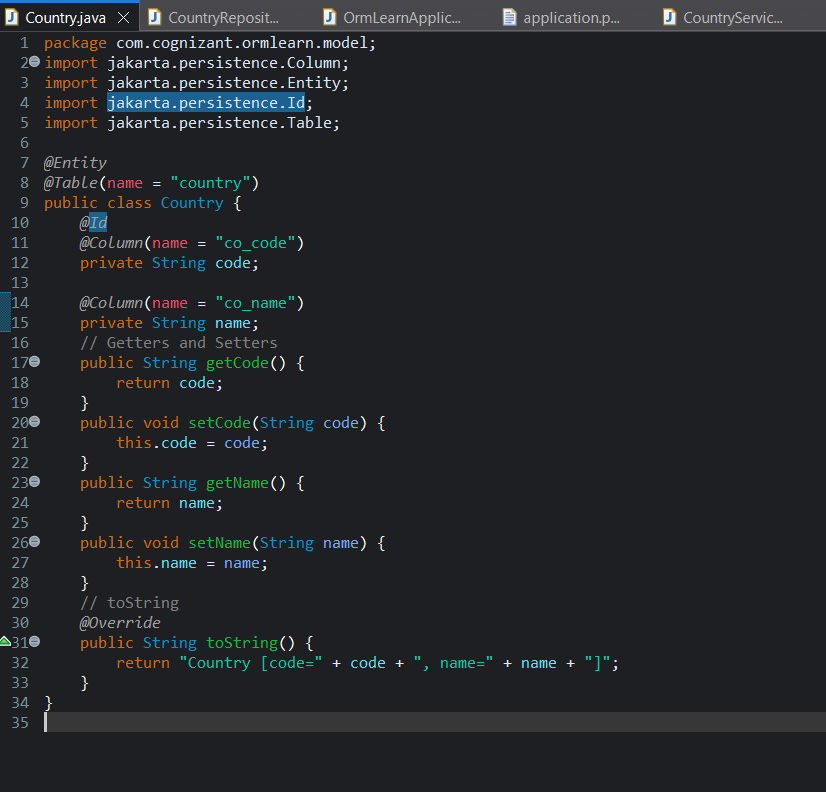
An application requires for features to be implemented with regards to country. These features needs to be supported by implementing them as service using Spring Data JPA.

* Find a country based on country code
* Add new country
* Update country
* Delete country
* Find list of countries matching a partial country name

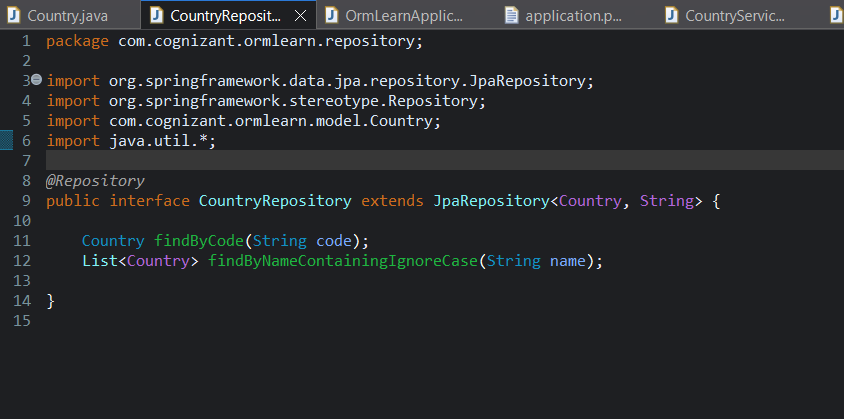
**1. Firstly Populating the table Country**

****

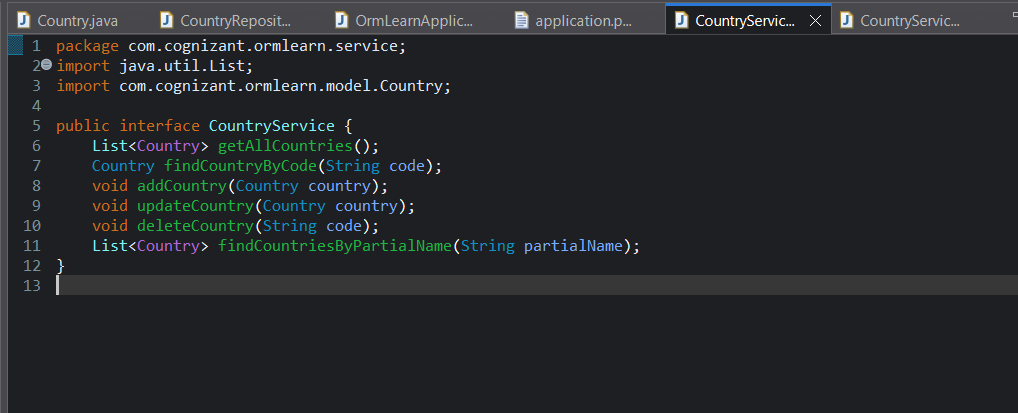
**2. Entity Class Country.java for mapping country table to SQL database:**



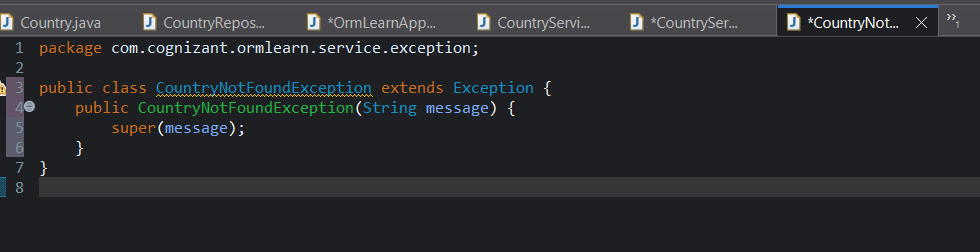
**3. Data Access Layer:**

****

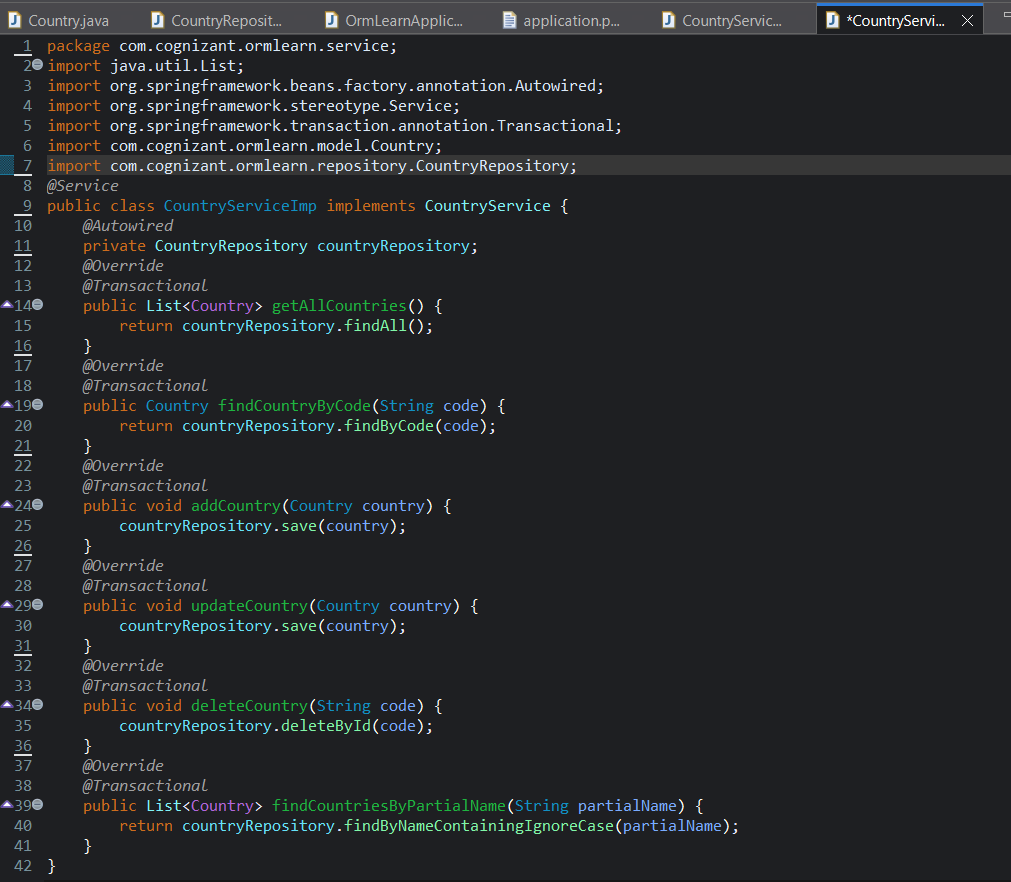
**4. Service.java for business logic methods**



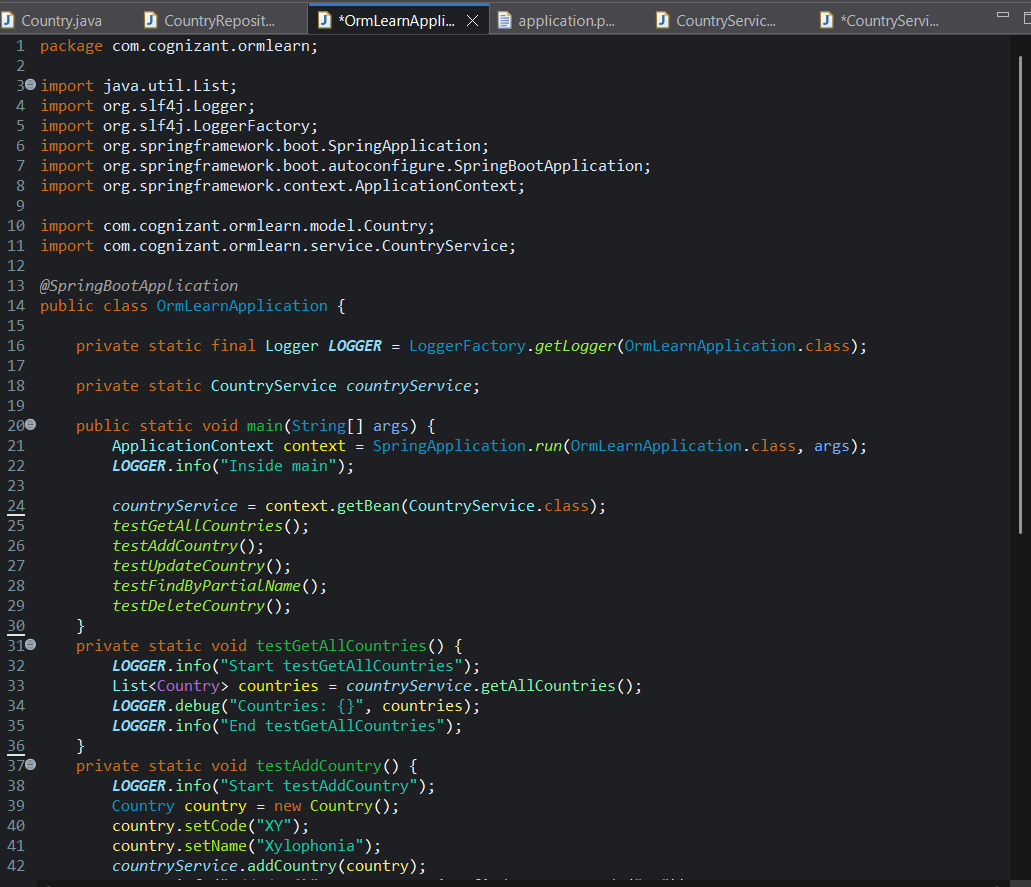
**Exception Class for HandsOn 7**

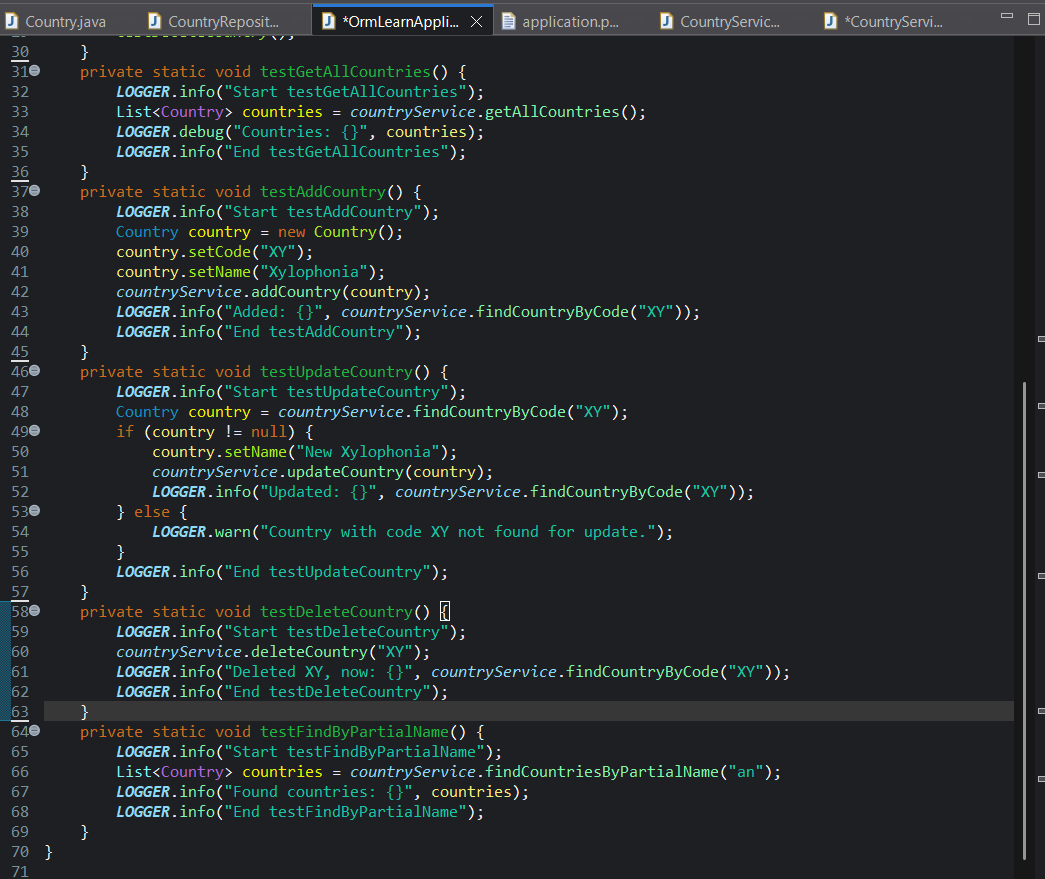
****

**5. CountryServiceImp implements CountryService and uses CountryRepository to interact with database**

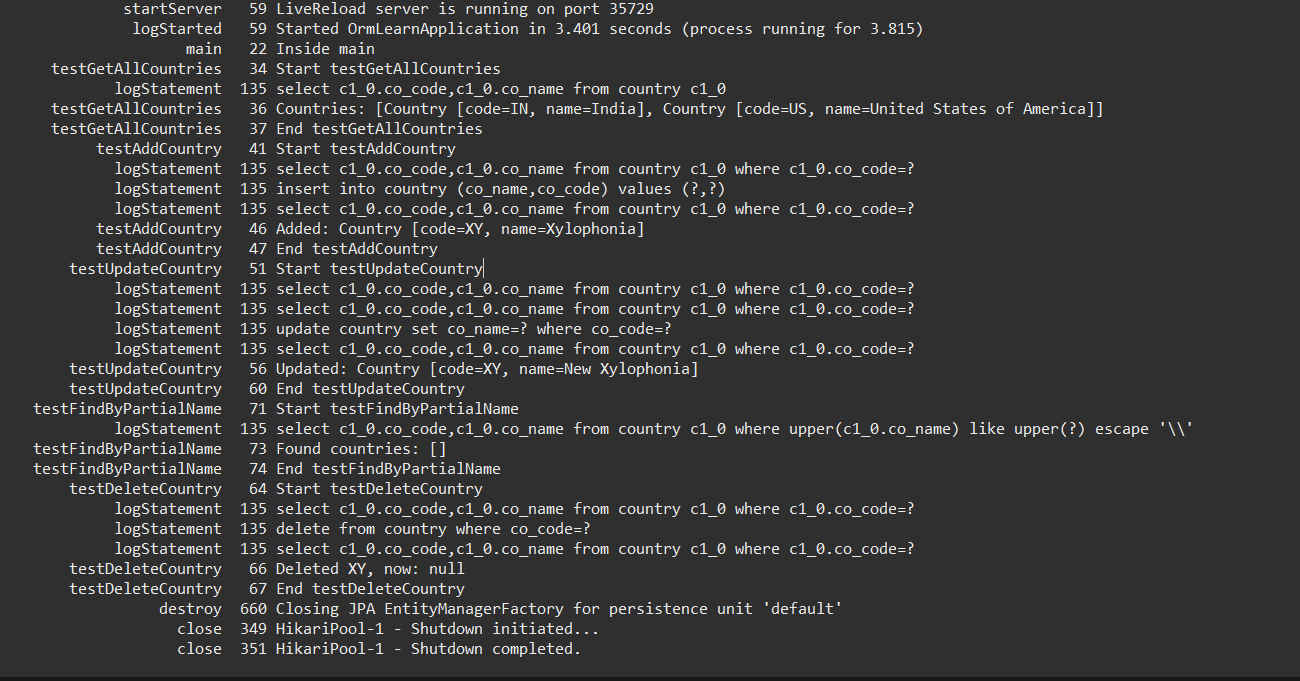
****

**6. Main Class to run the application**

****

****

**7. Output/Results:**

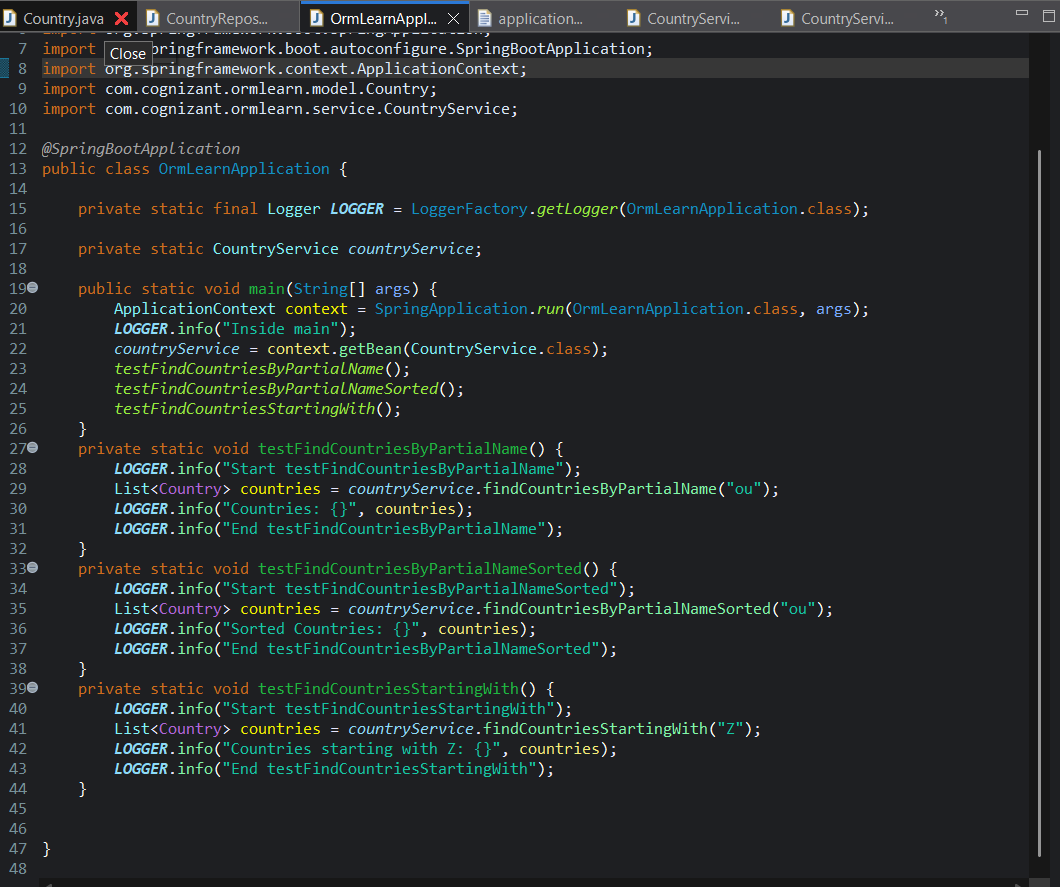
****

**2. spring-data-jpa-handson**

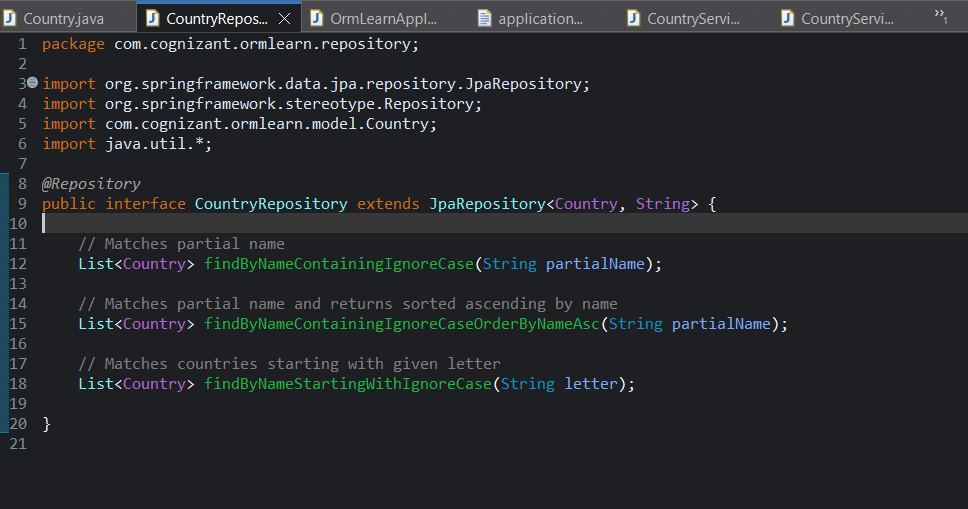
**Hands on 1**

**Write queries on country table using Query Methods**

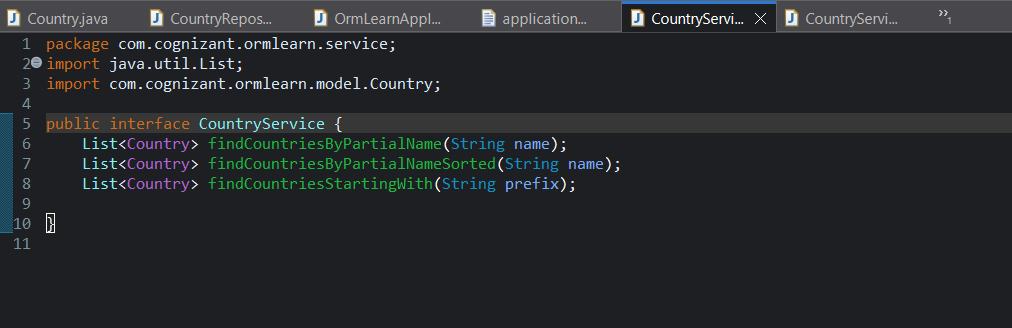
**1. Main Class for calling the query methods:**

****

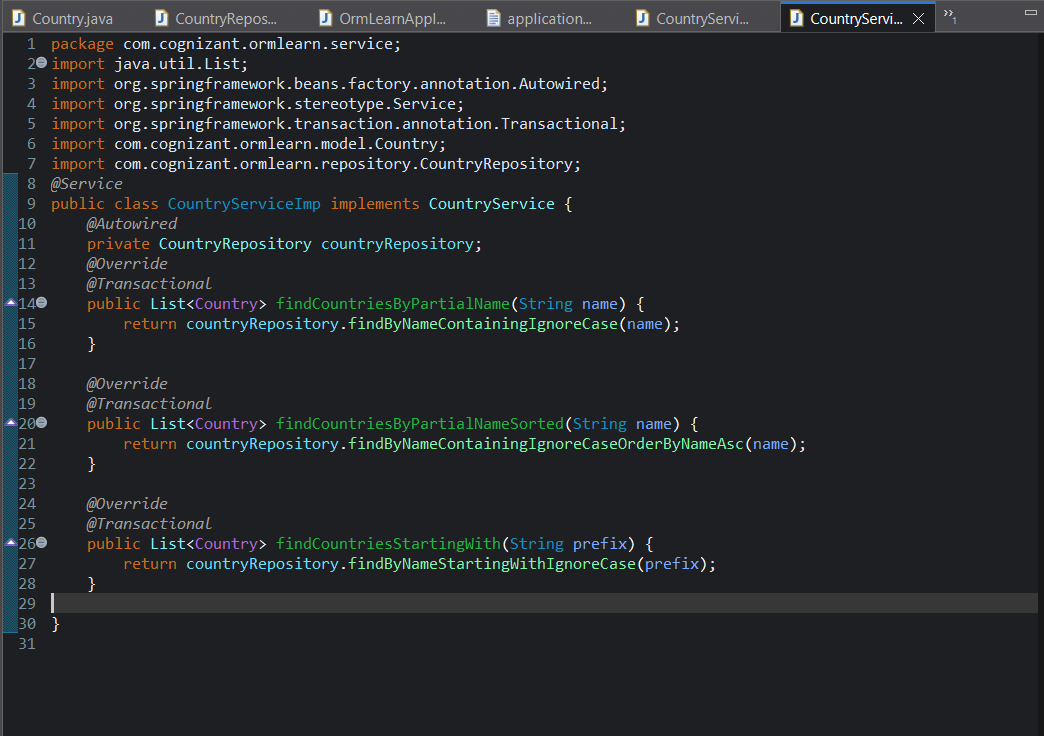
**2. Data Access Layer**

****

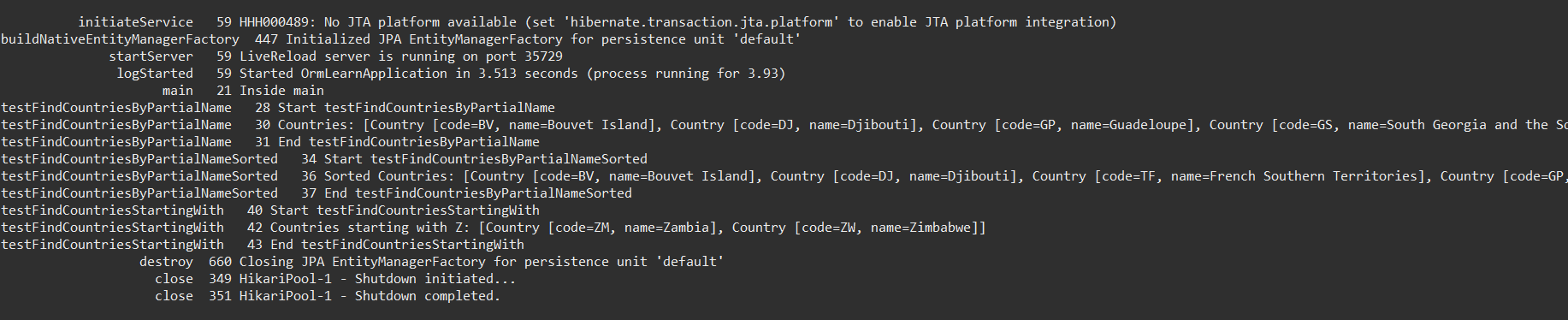
**3. Service class for business logic**

****

**4. Service implementation class**

****

**5. Output:**

****