

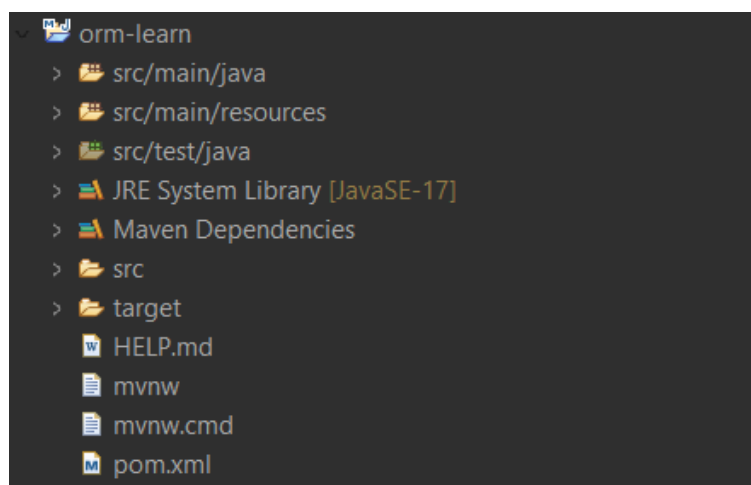
# Spring Data JPA - Quick Example

## Software Pre-requisites

- MySQL Server 8.0
- MySQL Workbench 8
- Eclipse IDE for Enterprise Java Developers 2019-03 R
- Maven 3.6.2

## Create a Eclipse Project using Spring Initializr

- Go to <https://start.spring.io/>
- Change Group as "com.cognizant"
- Change Artifact Id as "orm-learn"
- In Options > Description enter "Demo project for Spring Data JPA and Hibernate"
- Click on menu and select "Spring Boot DevTools", "Spring Data JPA" and "MySQL Driver"
- Click Generate and download the project as zip
- Extract the zip in root folder to Eclipse Workspace
- Import the project in Eclipse "File > Import > Maven > Existing Maven Projects > Click Browse and select extracted folder > Finish"
- 
- Create a new schema "ormlearn" in MySQL database. Execute the following commands to open MySQL client and create schema.



```
> mysql -u root -p
```

```
mysql> create schema ormlearn;
```

- In orm-learn Eclipse project, open `src/main/resources/application.properties` and include the below database and log configuration.

```
# Spring Framework and application log
logging.level.org.springframework=info
logging.level.com.cognizant=debug

# Hibernate logs for displaying executed SQL, input and output
logging.level.org.hibernate.SQL=trace
logging.level.org.hibernate.type.descriptor.sql=trace

# Log pattern
logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25
logger{25} %25M %4L %m%n

# Database configuration
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn
spring.datasource.username=root
spring.datasource.password=root

# Hibernate configuration
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
```

- Build the project using 'mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456' command in command line
- Include logs for verifying if `main()` method is called.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

public static void main(String[] args) {
    SpringApplication.run(OrmLearnApplication.class, args);
    LOGGER.info("Inside main");
}
```

- Execute the OrmLearnApplication and check in log if main method is called.

SME to walk through the following aspects related to the project created:

1. **src/main/java** - Folder with application code
2. **src/main/resources** - Folder for application configuration
3. **src/test/java** - Folder with code for testing the application
4. **OrmLearnApplication.java** - Walkthrough the main() method.
5. Purpose of **@SpringBootApplication** annotation
6. **pom.xml**
  1. Walkthrough all the configuration defined in XML file
  2. Open 'Dependency Hierarchy' and show the dependency tree.

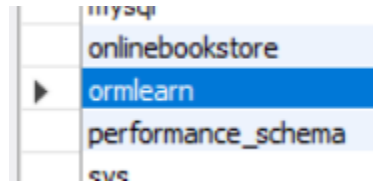
## Country table creation

- Create a new table country with columns for code and name. For sample, let us insert one country with values 'IN' and 'India' in this table.

```
create table country(co_code varchar(2) primary key, co_name varchar(50));
```

- Insert couple of records into the table

```
insert into country values ('IN', 'India');
insert into country values ('US', 'United States of America');
```



## Persistence Class - `com.cognizant.orm-learn.model.Country`

- Open Eclipse with orm-learn project
- Create new package `com.cognizant.orm-learn.model`
- Create `Country.java`, then generate getters, setters and `toString()` methods.
- Include `@Entity` and `@Table` at class level
- Include `@Column` annotations in each getter method specifying the column name.

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="country")
public class Country {

    @Id
    @Column(name="code")
    private String code;

    @Column(name="name")
    private String name;

    // getters and setters

    // toString()

}
```

### Notes:

- @Entity is an indicator to Spring Data JPA that it is an entity class for the application
- @Table helps in defining the mapping database table
- @Id helps in defining the primary key
- @Column helps in defining the mapping table column

```
tityManagerFactory for persistence unit 'default'
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 9.897 s
[INFO] Finished at: 2025-07-01T20:29:43+05:30
[INFO] -----
```

### Repository Class - com.cognizant.orm-learn.CountryRepository

- Create new package com.cognizant.orm-learn.repository
- Create new interface named CountryRepository that extends JpaRepository<Country, String>
- Define @Repository annotation at class level

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.cognizant.ormlearn.model.Country;

@Repository
public interface CountryRepository extends JpaRepository<Country, String> {

}
```

```
... 49 common frames omitted
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 11.474 s
[INFO] Finished at: 2025-07-01T20:47:49+05:30
[INFO] -----
C:\Users\HP\Desktop\orm-learn\orm-learn>
```

## Service Class - com.cognizant.orm-learn.service.CountryService

- Create new package com.cognizant.orm-learn.service
- Create new class CountryService
- Include @Service annotation at class level
- Autowire CountryRepository in CountryService
- Include new method getAllCountries() method that returns a list of countries.
- Include @Transactional annotation for this method
- In getAllCountries() method invoke countryRepository.findAll() method and return the result

## Testing in OrmLearnApplication.java

- Include a static reference to CountryService in OrmLearnApplication class

```
private static CountryService countryService;
```

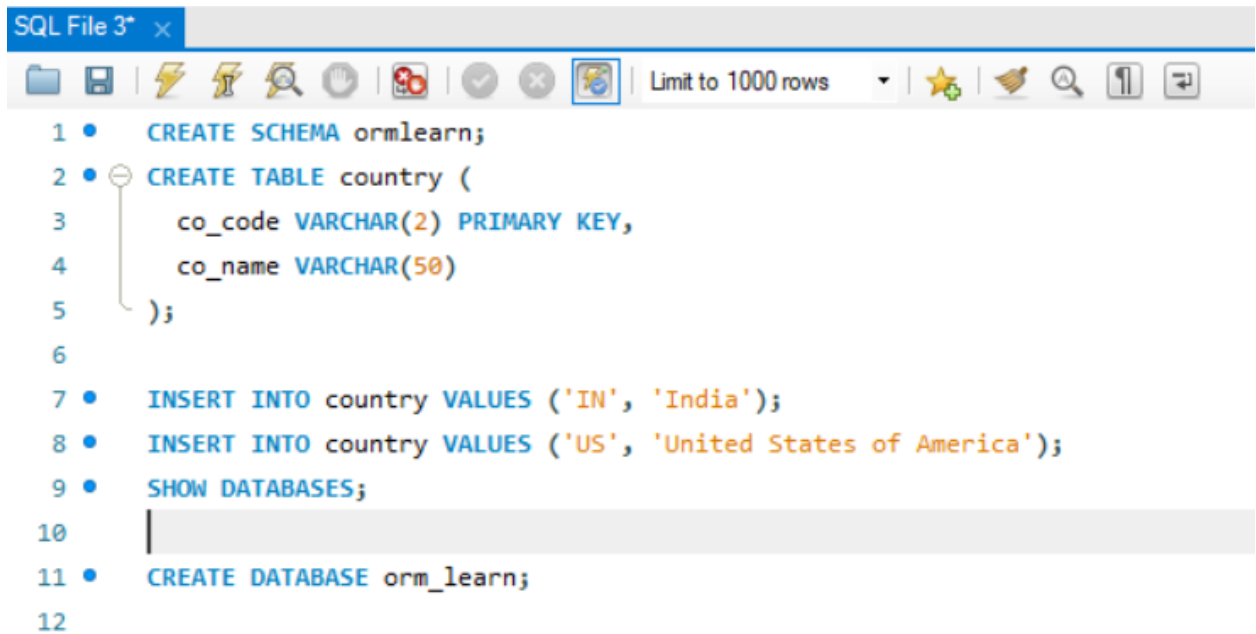
- Define a test method to get all countries from service.

```
private static void testGetAllCountries() {  
    LOGGER.info("Start");  
    List<Country> countries = countryService.getAllCountries();  
    LOGGER.debug("countries={}", countries);  
    LOGGER.info("End");  
}
```

- Modify SpringApplication.run() invocation to set the application context and the CountryService reference from the application context.

```
ApplicationContext context = SpringApplication.run(OrmLearnApplication.  
class, args);  
countryService = context.getBean(CountryService.class);  
  
testGetAllCountries();
```

- Execute main method to check if data from ormlearn database is retrieved.



```
1 • CREATE SCHEMA ormlearn;
2 • CREATE TABLE country (
3     co_code VARCHAR(2) PRIMARY KEY,
4     co_name VARCHAR(50)
5 );
6
7 • INSERT INTO country VALUES ('IN', 'India');
8 • INSERT INTO country VALUES ('US', 'United States of America');
9 • SHOW DATABASES;
10
11 • CREATE DATABASE orm_learn;
12
```

Hands on 2

## Hibernate XML Config implementation walk through

SME to provide explanation on the sample Hibernate implementation available in the link below:

[https://www.tutorialspoint.com/hibernate/hibernate\\_examples.htm](https://www.tutorialspoint.com/hibernate/hibernate_examples.htm)

### Explanation Topics

- Explain how object to relational database mapping done in hibernate xml configuration file
- Explain about following aspects of implementing the end to end operations in Hibernate:
  - SessionFactory
  - Session
  - Transaction
  - beginTransaction()
  - commit()
  - rollback()
  - session.save()
  - session.createQuery().list()
  - session.get()

- session.delete()

```
mysql> USE hibernatedb;
Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_hibernatedb |
+-----+
| employee               |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM employee;
Empty set (0.00 sec)
```

```
Hibernate: create table employee (id integer not null auto_increment, first_name varchar(255), last_name varchar(255),
primary key (id)) engine=InnoDB
Jul 02, 2025 12:22:00 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Hibernate: insert into employee (first_name, last_name) values (?, ?)
Hibernate: select employee0_.id as id1_0_, employee0_.first_name as first_na2_0_, employee0_.last_name as last_nam3_0_ from
employee employee0_
1: Alice Smith
Hibernate: update employee set first_name=?, last_name=? where id=?
Hibernate: delete from employee where id=?
```

Hands on 3

## Hibernate Annotation Config implementation walk through

SME to provide explanation on the sample Hibernate implementation available in the link below:

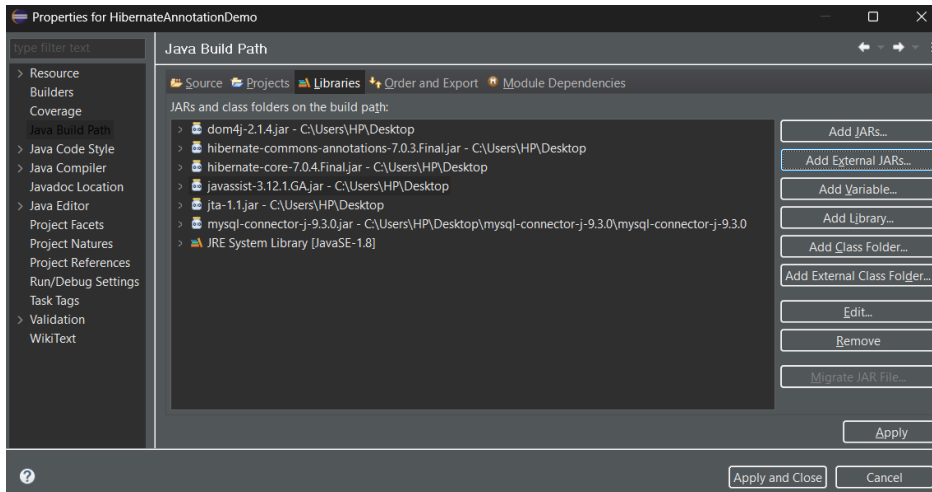
[https://www.tutorialspoint.com/hibernate/hibernate\\_annotations.htm](https://www.tutorialspoint.com/hibernate/hibernate_annotations.htm)

### Explanation Topics

- Explain how object to relational database mapping done in persistence class file Employee
- Explain about following aspects of implementing the end to end operations in Hibernate:
  - @Entity
  - @Table
  - @Id
  - @GeneratedValue
  - @Column



- Hibernate Configuration (hibernate.cfg.xml)
  - Dialect
  - Driver
  - Connection URL
  - Username
  - Password



```

[INFO] Installing C:\hibernate-annotation-demo\pom.xml to C:\Users\HP\.m2\repository\com\example\hibernate-annotation-de
mo\1.0\hibernate-annotation-demo-1.0.pom
[INFO] Installing C:\hibernate-annotation-demo\target\hibernate-annotation-demo-1.0.jar to C:\Users\HP\.m2\repository\co
m\example\hibernate-annotation-demo\1.0\hibernate-annotation-demo-1.0.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 15.028 s
[INFO] Finished at: 2025-07-02T13:20:46+05:30
[INFO] -----

```

```

Hibernate: insert into employee (first_name, last_name, salary) values (?, ?, ?)
Employee saved!
Jul 02, 2025 1:26:12 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnection

```

```

mysql> SELECT * FROM employee;
+----+-----+-----+-----+
| id | first_name | last_name | salary |
+----+-----+-----+-----+
| 2  | Alice      | Smith     | 50000  |
+----+-----+-----+-----+
1 row in set (0.00 sec)

```

## Difference between JPA, Hibernate and Spring Data JPA

### Java Persistence API (JPA)

- JSR 338 Specification for persisting, reading and managing data from Java objects
- Does not contain concrete implementation of the specification
- Hibernate is one of the implementation of JPA

### Hibernate

- ORM Tool that implements JPA

### Spring Data JPA

- Does not have JPA implementation, but reduces boiler plate code
- This is another level of abstraction over JPA implementation provider like Hibernate
- Manages transactions

### Refer code snippets below on how the code compares between Hibernate and Spring Data JPA Hibernate

```
/* Method to CREATE an employee in the database */
public Integer addEmployee(Employee employee){
    Session session = factory.openSession();
    Transaction tx = null;
    Integer employeeID = null;

    try {
        tx = session.beginTransaction();
        employeeID = (Integer) session.save(employee);
        tx.commit();
    } catch (HibernateException e) {
        if (tx != null) tx.rollback();
        e.printStackTrace();
    }
}
```

```

    } finally {
        session.close();
    }
    return employeeID;
}

```

## Spring Data JPA

### EmployeeRepository.java

```

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

```

### EmployeeService.java

```

@Autowired
private EmployeeRepository employeeRepository;

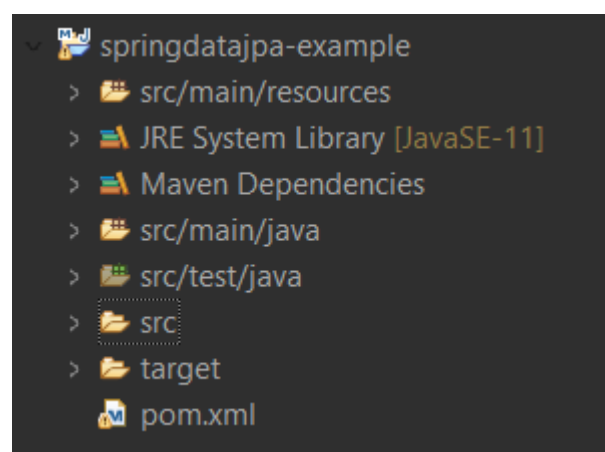
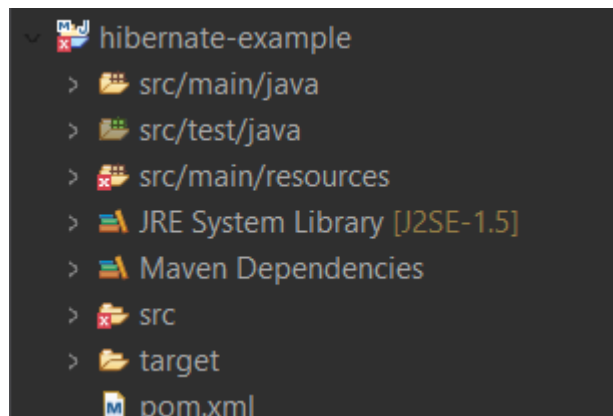
@Transactional
public void addEmployee(Employee employee) {
    employeeRepository.save(employee);
}

```

### Reference Links:

<https://dzone.com/articles/what-is-the-difference-between-hibernate-and-spring>

<https://www.javaworld.com/article/3379043/what-is-jpa-introduction-to-the-jpa-persistence-api.html>



```
Console x Debug x JUnit
terminated> SpringDataJpaApp [Java Application] C:\Program Files\ eclipse - java - 2022-12-R-win32-x86_64\ eclipse\ plugins\ org.eclipse.jdt.launcher\ openjdk.hotspot.jre.full.win32.x86_64.17.0.5.v20221102-0933\ jre\ bin\ java.exe (02-Jul-2025, 2:01:07 pm - 2:01:13 pm) [pid: 16420]

:: Spring Boot ::
(v2.7.18)

2025-07-02 14:01:09.494 INFO 16420 --- [main] com.example.SpringDataJpaApp : Starting SpringDataJpaApp using Java 17.0.5 on LAPTOP-9UI3COB6 with PID 16420 (C:\
\springdatajpa-example\target\classes started by HP in C:\springdatajpa-example)
2025-07-02 14:01:09.498 INFO 16420 --- [main] com.example.SpringDataJpaApp : No active profile set, falling back to 1 default profile: "default"
2025-07-02 14:01:10.201 INFO 16420 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-07-02 14:01:10.225 INFO 16420 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 8 ms. Found 0 JPA repository interfaces.
2025-07-02 14:01:10.979 INFO 16420 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2025-07-02 14:01:11.091 INFO 16420 --- [main] org.hibernate.Version : HHH0000412: Hibernate ORM core version 5.6.15.Final
2025-07-02 14:01:11.389 INFO 16420 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations (5.1.2.Final)
2025-07-02 14:01:11.673 INFO 16420 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2025-07-02 14:01:12.389 INFO 16420 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2025-07-02 14:01:12.430 INFO 16420 --- [main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
2025-07-02 14:01:13.028 INFO 16420 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH0000490: Using JtaPlatform implementation:
[org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2025-07-02 14:01:13.046 INFO 16420 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-07-02 14:01:13.211 INFO 16420 --- [main] com.example.SpringDataJpaApp : Started SpringDataJpaApp in 4.402 seconds (JVM running for 5.12)
2025-07-02 14:01:13.225 INFO 16420 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
2025-07-02 14:01:13.228 INFO 16420 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
2025-07-02 14:01:13.246 INFO 16420 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
```

```
2025-07-02 14:06:05.214 INFO 18120 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataS
down completed.
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 9.221 s
[INFO] Finished at: 2025-07-02T14:06:05+05:30
[INFO] -----
```

```
mysql> USE employee_db;
Database changed
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

Hands on 5

## 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

Before starting the implementation of the above features, there are few configuration and data population that needs to be incorporated. Please refer each topic below and implement the same.

## Explanation for Hibernate table creation configuration

- Moreover the ddl-auto defines how hibernate behaves if a specific table or column is not present in the database.
  - create - drops existing tables data and structure, then creates new tables
  - validate - check if the table and columns exist or not, throws an exception if a matching table or column is not found
  - update - if a table does not exists, it creates a new table; if a column does not exists, it creates a new column
  - create-drop - creates the table, once all operations are completed, the table is dropped

```
# Hibernate ddl auto (create, create-drop, update, validate)
spring.jpa.hibernate.ddl-auto=validate
```

## Populate country table

- Delete all the records in Country table and then use the below script to create the actual list of all countries in our world.

```
insert into country (co_code, co_name) values ("AF", "Afghanistan");
insert into country (co_code, co_name) values ("AL", "Albania");
insert into country (co_code, co_name) values ("DZ", "Algeria");
insert into country (co_code, co_name) values ("AS", "American Samoa");
insert into country (co_code, co_name) values ("AD", "Andorra");
insert into country (co_code, co_name) values ("AO", "Angola");
insert into country (co_code, co_name) values ("AI", "Anguilla");
insert into country (co_code, co_name) values ("AQ", "Antarctica");
insert into country (co_code, co_name) values ("AG", "Antigua and Barbuda");
insert into country (co_code, co_name) values ("AR", "Argentina");
insert into country (co_code, co_name) values ("AM", "Armenia");
insert into country (co_code, co_name) values ("AW", "Aruba");
insert into country (co_code, co_name) values ("AU", "Australia");
insert into country (co_code, co_name) values ("AT", "Austria");
insert into country (co_code, co_name) values ("AZ", "Azerbaijan");
insert into country (co_code, co_name) values ("BS", "Bahamas");
insert into country (co_code, co_name) values ("BH", "Bahrain");
insert into country (co_code, co_name) values ("BD", "Bangladesh");
insert into country (co_code, co_name) values ("BB", "Barbados");
insert into country (co_code, co_name) values ("BY", "Belarus");
insert into country (co_code, co_name) values ("BE", "Belgium");
insert into country (co_code, co_name) values ("BZ", "Belize");
insert into country (co_code, co_name) values ("BJ", "Benin");
insert into country (co_code, co_name) values ("BM", "Bermuda");
insert into country (co_code, co_name) values ("BT", "Bhutan");
```

```

insert into country (co_code, co_name) values ("BO", "Bolivia, Plurinational
State of");
insert into country (co_code, co_name) values ("BQ", "Bonaire, Sint Eustatius
and Saba");
insert into country (co_code, co_name) values ("BA", "Bosnia and Herzegovina");
insert into country (co_code, co_name) values ("BW", "Botswana");
insert into country (co_code, co_name) values ("BV", "Bouvet Island");
insert into country (co_code, co_name) values ("BR", "Brazil");
insert into country (co_code, co_name) values ("IO", "British Indian Ocean
Territory");
insert into country (co_code, co_name) values ("BN", "Brunei Darussalam");
insert into country (co_code, co_name) values ("BG", "Bulgaria");
insert into country (co_code, co_name) values ("BF", "Burkina Faso");
insert into country (co_code, co_name) values ("BI", "Burundi");
insert into country (co_code, co_name) values ("KH", "Cambodia");
insert into country (co_code, co_name) values ("CM", "Cameroon");
insert into country (co_code, co_name) values ("CA", "Canada");
insert into country (co_code, co_name) values ("CV", "Cape Verde");
insert into country (co_code, co_name) values ("KY", "Cayman Islands");
insert into country (co_code, co_name) values ("CF", "Central African
Republic");
insert into country (co_code, co_name) values ("TD", "Chad");
insert into country (co_code, co_name) values ("CL", "Chile");
insert into country (co_code, co_name) values ("CN", "China");
insert into country (co_code, co_name) values ("CX", "Christmas Island");
insert into country (co_code, co_name) values ("CC", "Cocos (Keeling)
Islands");
insert into country (co_code, co_name) values ("CO", "Colombia");
insert into country (co_code, co_name) values ("KM", "Comoros");
insert into country (co_code, co_name) values ("CG", "Congo");
insert into country (co_code, co_name) values ("CD", "Congo, the Democratic
Republic of the");
insert into country (co_code, co_name) values ("CK", "Cook Islands");
insert into country (co_code, co_name) values ("CR", "Costa Rica");
insert into country (co_code, co_name) values ("HR", "Croatia");
insert into country (co_code, co_name) values ("CU", "Cuba");
insert into country (co_code, co_name) values ("CW", "Curaçao");
insert into country (co_code, co_name) values ("CY", "Cyprus");
insert into country (co_code, co_name) values ("CZ", "Czech Republic");
insert into country (co_code, co_name) values ("CI", "Côte d'Ivoire");
insert into country (co_code, co_name) values ("DK", "Denmark");
insert into country (co_code, co_name) values ("DJ", "Djibouti");
insert into country (co_code, co_name) values ("DM", "Dominica");
insert into country (co_code, co_name) values ("DO", "Dominican Republic");
insert into country (co_code, co_name) values ("EC", "Ecuador");
insert into country (co_code, co_name) values ("EG", "Egypt");
insert into country (co_code, co_name) values ("SV", "El Salvador");
insert into country (co_code, co_name) values ("GQ", "Equatorial Guinea");
insert into country (co_code, co_name) values ("ER", "Eritrea");
insert into country (co_code, co_name) values ("EE", "Estonia");
insert into country (co_code, co_name) values ("ET", "Ethiopia");
insert into country (co_code, co_name) values ("FK", "Falkland Islands
(Malvinas)");
insert into country (co_code, co_name) values ("FO", "Faroe Islands");
insert into country (co_code, co_name) values ("FJ", "Fiji");

```

```

insert into country (co_code, co_name) values ("FI", "Finland");
insert into country (co_code, co_name) values ("FR", "France");
insert into country (co_code, co_name) values ("GF", "French Guiana");
insert into country (co_code, co_name) values ("PF", "French Polynesia");
insert into country (co_code, co_name) values ("TF", "French Southern
Territories");
insert into country (co_code, co_name) values ("GA", "Gabon");
insert into country (co_code, co_name) values ("GM", "Gambia");
insert into country (co_code, co_name) values ("GE", "Georgia");
insert into country (co_code, co_name) values ("DE", "Germany");
insert into country (co_code, co_name) values ("GH", "Ghana");
insert into country (co_code, co_name) values ("GI", "Gibraltar");
insert into country (co_code, co_name) values ("GR", "Greece");
insert into country (co_code, co_name) values ("GL", "Greenland");
insert into country (co_code, co_name) values ("GD", "Grenada");
insert into country (co_code, co_name) values ("GP", "Guadeloupe");
insert into country (co_code, co_name) values ("GU", "Guam");
insert into country (co_code, co_name) values ("GT", "Guatemala");
insert into country (co_code, co_name) values ("GG", "Guernsey");
insert into country (co_code, co_name) values ("GN", "Guinea");
insert into country (co_code, co_name) values ("GW", "Guinea-Bissau");
insert into country (co_code, co_name) values ("GY", "Guyana");
insert into country (co_code, co_name) values ("HT", "Haiti");
insert into country (co_code, co_name) values ("HM", "Heard Island and McDonald
Islands");
insert into country (co_code, co_name) values ("VA", "Holy See (Vatican City
State)");
insert into country (co_code, co_name) values ("HN", "Honduras");
insert into country (co_code, co_name) values ("HK", "Hong Kong");
insert into country (co_code, co_name) values ("HU", "Hungary");
insert into country (co_code, co_name) values ("IS", "Iceland");
insert into country (co_code, co_name) values ("IN", "India");
insert into country (co_code, co_name) values ("ID", "Indonesia");
insert into country (co_code, co_name) values ("IR", "Iran, Islamic Republic
of");
insert into country (co_code, co_name) values ("IQ", "Iraq");
insert into country (co_code, co_name) values ("IE", "Ireland");
insert into country (co_code, co_name) values ("IM", "Isle of Man");
insert into country (co_code, co_name) values ("IL", "Israel");
insert into country (co_code, co_name) values ("IT", "Italy");
insert into country (co_code, co_name) values ("JM", "Jamaica");
insert into country (co_code, co_name) values ("JP", "Japan");
insert into country (co_code, co_name) values ("JE", "Jersey");
insert into country (co_code, co_name) values ("JO", "Jordan");
insert into country (co_code, co_name) values ("KZ", "Kazakhstan");
insert into country (co_code, co_name) values ("KE", "Kenya");
insert into country (co_code, co_name) values ("KI", "Kiribati");
insert into country (co_code, co_name) values ("KP", "Democratic People's
Republic of Korea");
insert into country (co_code, co_name) values ("KR", "Republic of Korea");
insert into country (co_code, co_name) values ("KW", "Kuwait");
insert into country (co_code, co_name) values ("KG", "Kyrgyzstan");
insert into country (co_code, co_name) values ("LA", "Lao People's Democratic
Republic");
insert into country (co_code, co_name) values ("LV", "Latvia");

```

```

insert into country (co_code, co_name) values ("LB", "Lebanon");
insert into country (co_code, co_name) values ("LS", "Lesotho");
insert into country (co_code, co_name) values ("LR", "Liberia");
insert into country (co_code, co_name) values ("LY", "Libya");
insert into country (co_code, co_name) values ("LI", "Liechtenstein");
insert into country (co_code, co_name) values ("LT", "Lithuania");
insert into country (co_code, co_name) values ("LU", "Luxembourg");
insert into country (co_code, co_name) values ("MO", "Macao");
insert into country (co_code, co_name) values ("MK", "Macedonia, the Former
Yugoslav Republic of");
insert into country (co_code, co_name) values ("MG", "Madagascar");
insert into country (co_code, co_name) values ("MW", "Malawi");
insert into country (co_code, co_name) values ("MY", "Malaysia");
insert into country (co_code, co_name) values ("MV", "Maldives");
insert into country (co_code, co_name) values ("ML", "Mali");
insert into country (co_code, co_name) values ("MT", "Malta");
insert into country (co_code, co_name) values ("MH", "Marshall Islands");
insert into country (co_code, co_name) values ("MQ", "Martinique");
insert into country (co_code, co_name) values ("MR", "Mauritania");
insert into country (co_code, co_name) values ("MU", "Mauritius");
insert into country (co_code, co_name) values ("YT", "Mayotte");
insert into country (co_code, co_name) values ("MX", "Mexico");
insert into country (co_code, co_name) values ("FM", "Micronesia, Federated
States of");
insert into country (co_code, co_name) values ("MD", "Moldova, Republic of");
insert into country (co_code, co_name) values ("MC", "Monaco");
insert into country (co_code, co_name) values ("MN", "Mongolia");
insert into country (co_code, co_name) values ("ME", "Montenegro");
insert into country (co_code, co_name) values ("MS", "Montserrat");
insert into country (co_code, co_name) values ("MA", "Morocco");
insert into country (co_code, co_name) values ("MZ", "Mozambique");
insert into country (co_code, co_name) values ("MM", "Myanmar");
insert into country (co_code, co_name) values ("NA", "Namibia");
insert into country (co_code, co_name) values ("NR", "Nauru");
insert into country (co_code, co_name) values ("NP", "Nepal");
insert into country (co_code, co_name) values ("NL", "Netherlands");
insert into country (co_code, co_name) values ("NC", "New Caledonia");
insert into country (co_code, co_name) values ("NZ", "New Zealand");
insert into country (co_code, co_name) values ("NI", "Nicaragua");
insert into country (co_code, co_name) values ("NE", "Niger");
insert into country (co_code, co_name) values ("NG", "Nigeria");
insert into country (co_code, co_name) values ("NU", "Niue");
insert into country (co_code, co_name) values ("NF", "Norfolk Island");
insert into country (co_code, co_name) values ("MP", "Northern Mariana
Islands");
insert into country (co_code, co_name) values ("NO", "Norway");
insert into country (co_code, co_name) values ("OM", "Oman");
insert into country (co_code, co_name) values ("PK", "Pakistan");
insert into country (co_code, co_name) values ("PW", "Palau");
insert into country (co_code, co_name) values ("PS", "Palestine, State of");
insert into country (co_code, co_name) values ("PA", "Panama");
insert into country (co_code, co_name) values ("PG", "Papua New Guinea");
insert into country (co_code, co_name) values ("PY", "Paraguay");
insert into country (co_code, co_name) values ("PE", "Peru");
insert into country (co_code, co_name) values ("PH", "Philippines");

```



```

insert into country (co_code, co_name) values ("PN", "Pitcairn");
insert into country (co_code, co_name) values ("PL", "Poland");
insert into country (co_code, co_name) values ("PT", "Portugal");
insert into country (co_code, co_name) values ("PR", "Puerto Rico");
insert into country (co_code, co_name) values ("QA", "Qatar");
insert into country (co_code, co_name) values ("RO", "Romania");
insert into country (co_code, co_name) values ("RU", "Russian Federation");
insert into country (co_code, co_name) values ("RW", "Rwanda");
insert into country (co_code, co_name) values ("RE", "Réunion");
insert into country (co_code, co_name) values ("BL", "Saint Barthélemy");
insert into country (co_code, co_name) values ("SH", "Saint Helena, Ascension
and Tristan da Cunha");
insert into country (co_code, co_name) values ("KN", "Saint Kitts and Nevis");
insert into country (co_code, co_name) values ("LC", "Saint Lucia");
insert into country (co_code, co_name) values ("MF", "Saint Martin (French
part)");
insert into country (co_code, co_name) values ("PM", "Saint Pierre and
Miquelon");
insert into country (co_code, co_name) values ("VC", "Saint Vincent and the
Grenadines");
insert into country (co_code, co_name) values ("WS", "Samoa");
insert into country (co_code, co_name) values ("SM", "San Marino");
insert into country (co_code, co_name) values ("ST", "Sao Tome and Principe");
insert into country (co_code, co_name) values ("SA", "Saudi Arabia");
insert into country (co_code, co_name) values ("SN", "Senegal");
insert into country (co_code, co_name) values ("RS", "Serbia");
insert into country (co_code, co_name) values ("SC", "Seychelles");
insert into country (co_code, co_name) values ("SL", "Sierra Leone");
insert into country (co_code, co_name) values ("SG", "Singapore");
insert into country (co_code, co_name) values ("SX", "Sint Maarten (Dutch
part)");
insert into country (co_code, co_name) values ("SK", "Slovakia");
insert into country (co_code, co_name) values ("SI", "Slovenia");
insert into country (co_code, co_name) values ("SB", "Solomon Islands");
insert into country (co_code, co_name) values ("SO", "Somalia");
insert into country (co_code, co_name) values ("ZA", "South Africa");
insert into country (co_code, co_name) values ("GS", "South Georgia and the
South Sandwich Islands");
insert into country (co_code, co_name) values ("SS", "South Sudan");
insert into country (co_code, co_name) values ("ES", "Spain");
insert into country (co_code, co_name) values ("LK", "Sri Lanka");
insert into country (co_code, co_name) values ("SD", "Sudan");
insert into country (co_code, co_name) values ("SR", "Suriname");
insert into country (co_code, co_name) values ("SJ", "Svalbard and Jan Mayen");
insert into country (co_code, co_name) values ("SZ", "Swaziland");
insert into country (co_code, co_name) values ("SE", "Sweden");
insert into country (co_code, co_name) values ("CH", "Switzerland");
insert into country (co_code, co_name) values ("SY", "Syrian Arab Republic");
insert into country (co_code, co_name) values ("TW", "Taiwan, Province of
China");
insert into country (co_code, co_name) values ("TJ", "Tajikistan");
insert into country (co_code, co_name) values ("TZ", "Tanzania, United Republic
of");
insert into country (co_code, co_name) values ("TH", "Thailand");
insert into country (co_code, co_name) values ("TL", "Timor-Leste");

```

```

insert into country (co_code, co_name) values ("TG", "Togo");
insert into country (co_code, co_name) values ("TK", "Tokelau");
insert into country (co_code, co_name) values ("TO", "Tonga");
insert into country (co_code, co_name) values ("TT", "Trinidad and Tobago");
insert into country (co_code, co_name) values ("TN", "Tunisia");
insert into country (co_code, co_name) values ("TR", "Turkey");
insert into country (co_code, co_name) values ("TM", "Turkmenistan");
insert into country (co_code, co_name) values ("TC", "Turks and Caicos
Islands");
insert into country (co_code, co_name) values ("TV", "Tuvalu");
insert into country (co_code, co_name) values ("UG", "Uganda");
insert into country (co_code, co_name) values ("UA", "Ukraine");
insert into country (co_code, co_name) values ("AE", "United Arab Emirates");
insert into country (co_code, co_name) values ("GB", "United Kingdom");
insert into country (co_code, co_name) values ("US", "United States");
insert into country (co_code, co_name) values ("UM", "United States Minor
Outlying Islands");
insert into country (co_code, co_name) values ("UY", "Uruguay");
insert into country (co_code, co_name) values ("UZ", "Uzbekistan");
insert into country (co_code, co_name) values ("VU", "Vanuatu");
insert into country (co_code, co_name) values ("VE", "Venezuela, Bolivarian
Republic of");
insert into country (co_code, co_name) values ("VN", "Viet Nam");
insert into country (co_code, co_name) values ("VG", "Virgin Islands,
British");
insert into country (co_code, co_name) values ("VI", "Virgin Islands, U.S.");
insert into country (co_code, co_name) values ("WF", "Wallis and Futuna");
insert into country (co_code, co_name) values ("EH", "Western Sahara");
insert into country (co_code, co_name) values ("YE", "Yemen");
insert into country (co_code, co_name) values ("ZM", "Zambia");
insert into country (co_code, co_name) values ("ZW", "Zimbabwe");
insert into country (co_code, co_name) values ("AX", "Åland Islands");

```

Refer subsequent hands on exercises to implement the features related to country.

```

port(s): 8080 (http) with context path ''
2025-07-02T15:57:48.754+05:30 INFO 10152 --- [          main] c.example.CountryManagementApplication : Started
CountryManagementApplication in 7.412 seconds (process running for 8.124)

```

```

[INFO]
[INFO] --- install:3.1.1:install (default-install) @ country-management ---
[INFO] Installing C:\country-management\pom.xml to C:\Users\HP\.m2\repository\com\example\country-management\0.0.1-SNAPS
HOT\country-management-0.0.1-SNAPSHOT.pom
[INFO] Installing C:\country-management\target\country-management-0.0.1-SNAPSHOT.jar to C:\Users\HP\.m2\repository\com\ex
ample\country-management\0.0.1-SNAPSHOT\country-management-0.0.1-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 14.215 s
[INFO] Finished at: 2025-07-02T16:02:45+05:30
[INFO] -----

```

## Hands on 6

# Find a country based on country code

- Create new exception class `CountryNotFoundException` in `com.cognizant.spring-learn.service.exception`
- Create new method `findCountryByCode()` in `CountryService` with `@Transactional` annotation
- In `findCountryByCode()` method, perform the following steps:
  - Method signature

```
@Transactional
public Country findCountryByCode(String countryCode) throws CountryNotFoundException
```

- Get the country based on `findById()` built in method

```
Optional<Country> result = countryRepository.findById(countryCode);
```

- From the result, check if a country is found. If not found, throw `CountryNotFoundException`

```
if (!result.isPresent())
```

- Use `get()` method to return the country fetched.

```
Country country = result.get();
```

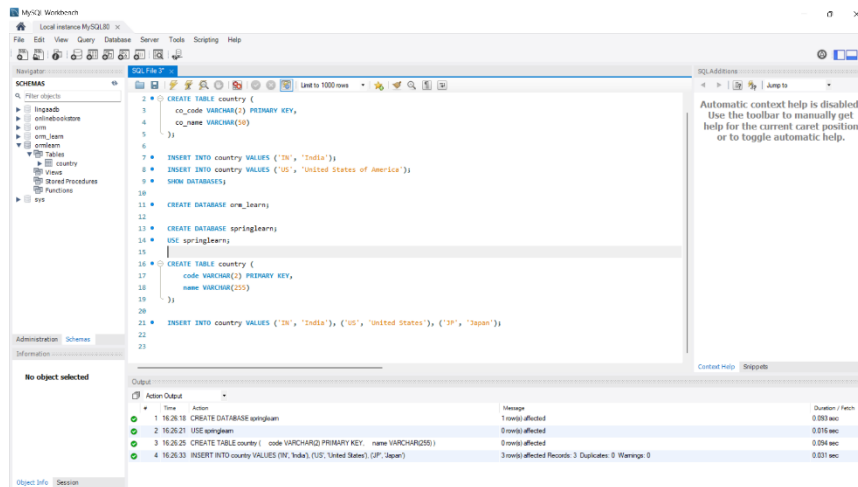
- Include new test method in `OrmLearnApplication` to find a country based on country code and compare the country name to check if it is valid.

```
private static void getAllCountriesTest() {
    LOGGER.info("Start");
    Country country = countryService.findCountryByCode("IN");
    LOGGER.debug("Country:{", country);
}
```

```
LOGGER.info("End");
}
```

- Invoke the above method in main() method and test it.

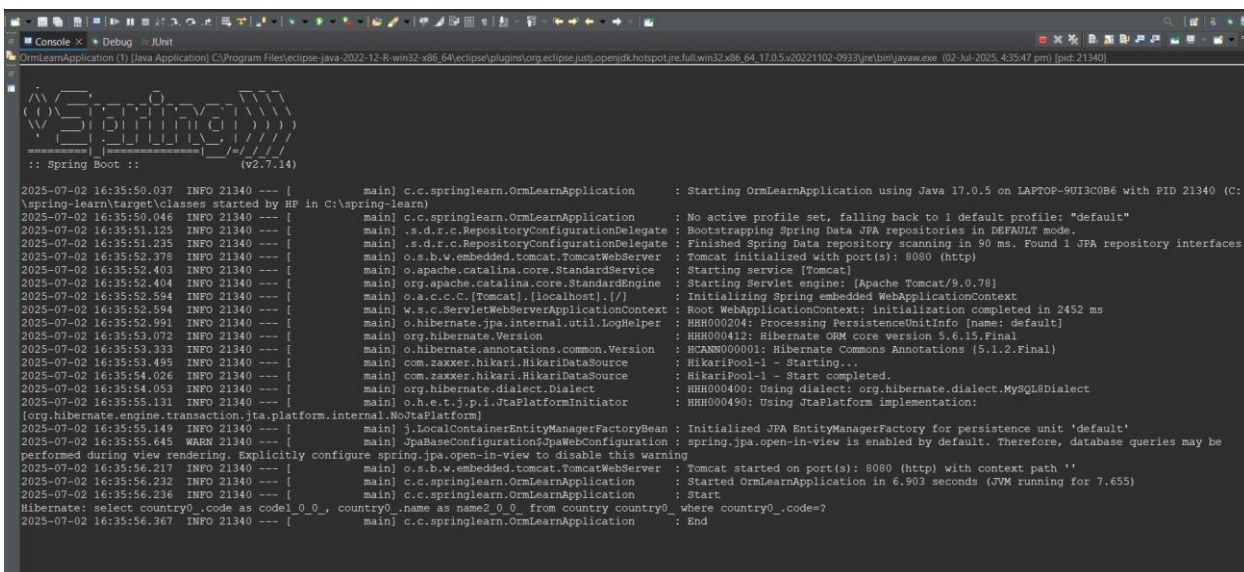
**NOTE:** SME to explain the importance of @Transactional annotation. Spring takes care of creating the Hibernate session and manages the transactionality when executing the service method.



```

[INFO] Installing C:\spring-learn\target\spring-learn-0.0.1-SNAPSHOT.jar to C:\Users\HP\.m2\repository\com\cognizant\spr
ing-learn\0.0.1-SNAPSHOT\spring-learn-0.0.1-SNAPSHOT.jar
[INFO] Installing C:\spring-learn\pom.xml to C:\Users\HP\.m2\repository\com\cognizant\spring-learn\0.0.1-SNAPSHOT\spring
-learn-0.0.1-SNAPSHOT.pom
[INFO]
-----
[INFO] BUILD SUCCESS
[INFO]
-----
[INFO] Total time: 10.425 s
[INFO] Finished at: 2025-07-02T16:32:29+05:30
[INFO]
-----

```



## Add a new country

- Create new method in CountryService.

```
@Transactional
public void addCountry(Country country)
```

- Invoke save() method of repository to get the country added.

```
countryRepository.save(country)
```

- Include new testAddCountry() method in OrmLearnApplication. Perform steps below:
  - Create new instance of country with a new code and name
  - Call countryService.addCountry() passing the country created in the previous step.
  - Invoke countryService.findCountryByCode() passing the same code used when adding a new country
  - Check in the database if the country is added

```
Hibernate: select country0_.code as code1_0_0_, country0_.name as name2_0_0_ from country c
=?
Hibernate: insert into country (name, code) values (?, ?)
Country added.
```

```
Hibernate: select country0_.code as code1_0_0_, country0_.name as name2_0_0_ from country country0_ where country0_.code
=?
Retrieved Country: Xyland
```

```
[INFO] --- resources:3.2.0:testResources (default-testResources) @ spring-learn ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Using 'UTF-8' encoding to copy filtered properties files.
[INFO] Copying 0 resource
[INFO] --- compiler:3.10.1:testCompile (default-testCompile) @ spring-learn ---
[INFO] Changes detected - recompiling the module!
[INFO] --- surefire:2.22.2:test (default-test) @ spring-learn ---
[INFO] --- jar:3.2.2:jar (default-jar) @ spring-learn ---
[INFO] Building jar: C:\spring-learn\target\spring-learn-0.0.1-SNAPSHOT.jar
[INFO] --- spring-boot:2.7.14:repackage (repackage) @ spring-learn ---
[INFO] Replacing main artifact with repackaged archive
[INFO] --- install:2.5.2:install (default-install) @ spring-learn ---
[INFO] Installing C:\spring-learn\target\spring-learn-0.0.1-SNAPSHOT.jar to C:\Users\HP\.m2\repository\com\cognizant\spring-learn\0.0.1-SNAPSHOT\spring-learn-0.0.1-SNAPSHOT.jar
[INFO] Installing C:\spring-learn\pom.xml to C:\Users\HP\.m2\repository\com\cognizant\spring-learn\0.0.1-SNAPSHOT\spring-learn-0.0.1-SNAPSHOT.pom
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 6.893 s
[INFO] Finished at: 2025-07-02T17:03:32+05:30
[INFO] -----
C:\spring-learn>
```

## Hands on 8

# Update a country based on code

- Create a new method `updateCountry()` in `CountryService` with parameters code and name. Annotate this method with `@Transactional`. Implement following steps in this method.
  - Get the reference of the country using `findById()` method in repository
  - In the country reference obtained, update the name of country using setter method
  - Call `countryRepository.save()` method to update the name
- Include new test method in `OrmLearnApplication`, which invokes `updateCountry()` method in `CountryService` passing a country's code and different name for the country.
- Check in database table if name is modified.

```
Console x Debug Run
OrmLearnApplication (1) Java Application C:\Program Files\jetbrains\java-2022-12-R-win32-x86_64\jetbrains\plugins\org.eclipse.justi.openjdk.hotspot.re.full.win32-x86_64-17.0.5.v20221102-0933\jre\bin\java.exe (02-Jul-2025 5:09:41 pm) [pid: 13768]

:: Spring Boot ::
(V2.7.14)

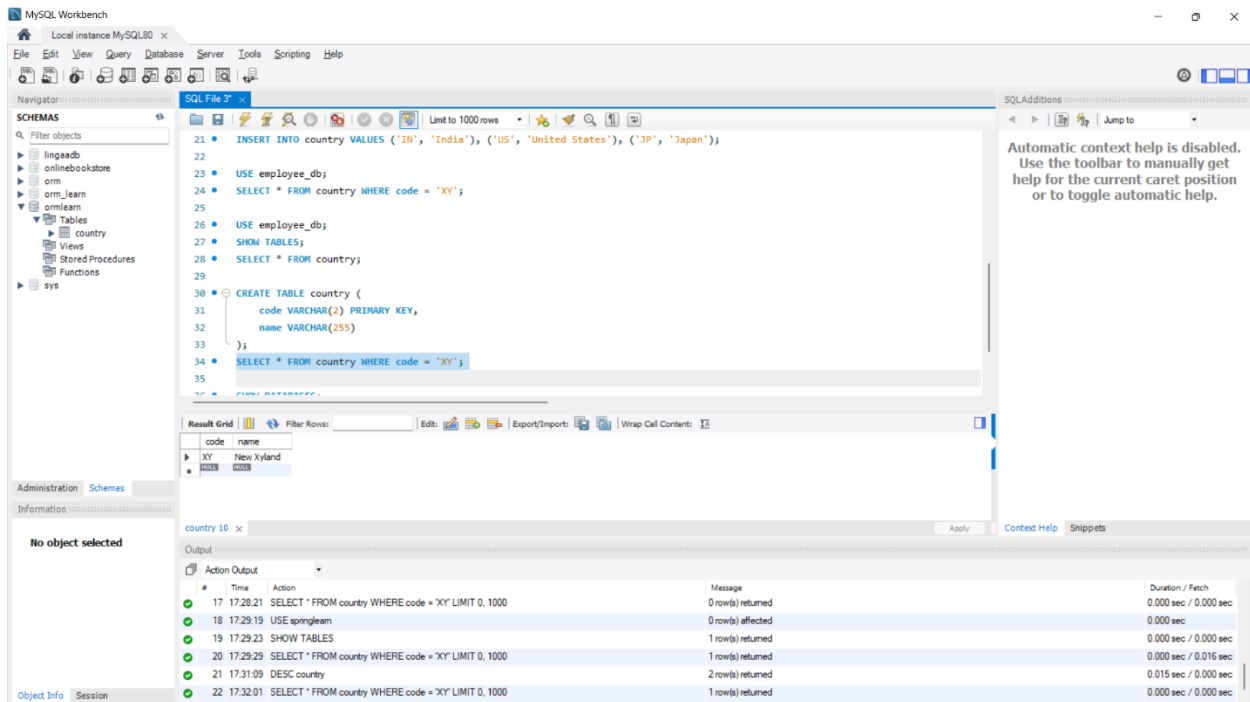
2025-07-02 17:09:43.916 INFO 13768 --- [main] c.c.springlearn.OrmLearnApplication : Starting OrmLearnApplication using Java 17.0.5 on LAPTOP-9UI3C0B6 with PID 13768 (C:\spring-learn\target\classes started by HP in C:\spring-learn)
2025-07-02 17:09:43.924 INFO 13768 --- [main] c.c.springlearn.OrmLearnApplication : No active profile set, falling back to 1 default profile: "default"
2025-07-02 17:09:44.553 INFO 13768 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-07-02 17:09:45.054 INFO 13768 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 79 ms. Found 1 JPA repository interfaces.
2025-07-02 17:09:46.164 INFO 13768 --- [main] o.s.b.w.e.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2025-07-02 17:09:46.190 INFO 13768 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-07-02 17:09:46.190 INFO 13768 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.78]
2025-07-02 17:09:46.390 INFO 13768 --- [main] o.a.c.c.c.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2025-07-02 17:09:46.391 INFO 13768 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2370 ms
2025-07-02 17:09:46.806 INFO 13768 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH0000204: Processing PersistenceUnitInfo [name: default]
2025-07-02 17:09:46.919 INFO 13768 --- [main] org.hibernate.Version : HHH0000412: Hibernate ORM core version 5.6.15.Final
2025-07-02 17:09:47.206 INFO 13768 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations [5.1.2.Final]
2025-07-02 17:09:47.381 INFO 13768 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2025-07-02 17:09:47.922 INFO 13768 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2025-07-02 17:09:47.952 INFO 13768 --- [main] org.hibernate.dialect.Dialect : HHH0000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
2025-07-02 17:09:49.026 INFO 13768 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH0000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2025-07-02 17:09:49.040 INFO 13768 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-07-02 17:09:49.520 WARN 13768 --- [main] UpabaseConfigurationJpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2025-07-02 17:09:50.061 INFO 13768 --- [main] o.s.b.w.e.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2025-07-02 17:09:50.082 INFO 13768 --- [main] c.c.springlearn.OrmLearnApplication : Started OrmLearnApplication in 6.932 seconds (JVM running for 7.694)

==> Adding new country...
Hibernate: select country0_.code as code1_0_0_, country0_.name as name2_0_0_ from country country0_ where country0_.code=?
Country added.
Hibernate: select country0_.code as code1_0_0_, country0_.name as name2_0_0_ from country country0_ where country0_.code=?
Retrieved Country: Country [code=XY, name=Xyland]
==> Updating country...
Hibernate: select country0_.code as code1_0_0_, country0_.name as name2_0_0_ from country country0_ where country0_.code=?
Hibernate: update country set name=? where code=?
Hibernate: select country0_.code as code1_0_0_, country0_.name as name2_0_0_ from country country0_ where country0_.code=?
Updated Country: Country [code=XY, name=New Xyland]
```

```
--- spring-boot:2.7.14:repackage (repackage) @ spring-learn ---
Replacing main artifact with repackaged archive
```

```
BUILD SUCCESS
```

```
Total time: 6.473 s
Finished at: 2025-07-02T17:12:42+05:30
```



```

mysql> SELECT * FROM country WHERE code = 'XY';
+-----+-----+
| code | name      |
+-----+-----+
| XY   | New Xyland |
+-----+-----+
1 row in set (0.00 sec)

```

Hands on 9

## Delete a country based on code

- Create new method deleteCountry() in CountryService. Annotate this method with @Transactional.
- In deleteCountry() method call deleteById() method of repository.
- Include new test method in OrmLearnApplication with following steps
  - Call the delete method based on the country code during the add country hands on
- Check in database if the country is deleted

```
Retrieved Country: Country [code=XY, name=Xyland]
==> Updating country...
Hibernate: select country0_.code as code1_0_0_, country0_.name as name2_0_0_ from country country0_ where country0_.code=?
Hibernate: update country set name=? where code=?
Hibernate: select country0_.code as code1_0_0_, country0_.name as name2_0_0_ from country country0_ where country0_.code=?
Updated Country: Country [code=XY, name=New Xyland]
==> Deleting country with code 'XY'...
Hibernate: select country0_.code as code1_0_0_, country0_.name as name2_0_0_ from country country0_ where country0_.code=?
Country with code XY deleted.
Hibernate: delete from country where code=?
Hibernate: select country0_.code as code1_0_0_, country0_.name as name2_0_0_ from country country0_ where country0_.code=?
Country with code 'XY' was successfully deleted.
```

```
mysql> USE springlearn;
Database changed
mysql> SELECT * FROM country WHERE code = 'XY';
Empty set (0.00 sec)

mysql>
```

**Submitted By:**

Name : Lingaraj Nayak

Superset ID : 6387607