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 %-25.25logger{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 is defining the primary key
- @Column helps in defining the mapping table column

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> {

}
```

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.

CODE:

OrmLearnApplication.java

```
package com.cognizant.orm_learn;
import java.util.List;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.domain.EntityScan;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

import com.cognizant.orm_learn.repository.CountryRepository;
import com.cognizant.ormlearn.model.Country;

@SpringBootApplication
@EntityScan("com.cognizant.ormlearn.model")
@EnableJpaRepositories("com.cognizant.orm_learn.repository")
@ComponentScan("com.cognizant.orm_learn")
public class OrmLearnApplication {

    private static CountryRepository countryRepository;

    public static void main(String[] args) {
        ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);
        countryRepository = context.getBean(CountryRepository.class);
        System.out.println("Inside main");
        testGetAllCountries();
    }

    public static void testGetAllCountries() {
```

```
        System.out.println("Start");
        List<Country> countries = countryRepository.findAll();
        System.out.println("countries = " + countries);
        System.out.println("End");
    }
}
```

Country.java

```
package com.cognizant.ormlearn.model;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

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

    @Id
    private String code;
    private String name;

    public String getCode() {
        return code;
    }

    public void setCode(String code) {
        this.code = code;
    }

    public String getName() {
```

```
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "Country [code=" + code + ", name=" + name + "]";
    }
}
```

CountryRepository.java

```
package com.cognizant.orm_learn.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import com.cognizant.ormlearn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {
}
```

application.properties

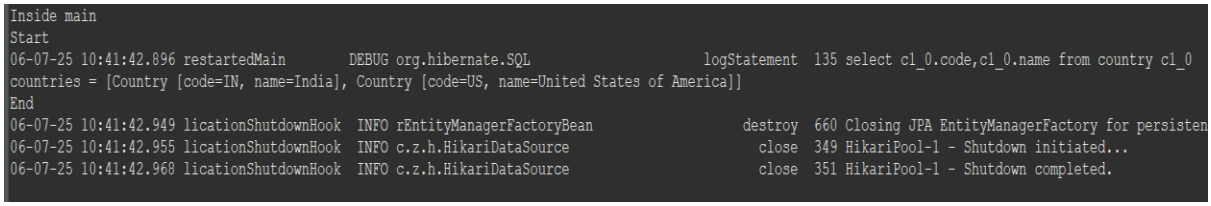
```
# Logging
logging.level.org.springframework=info
logging.level.com.cognizant=debug
logging.level.org.hibernate.SQL=trace
logging.level.org.hibernate.type.descriptor.sql=trace

# Log pattern
logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{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.database-platform=org.hibernate.dialect.MySQLDialect
```

OUTPUT:



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

EmployeeRespository.java

```
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {


}
```

EmployeeService.java

```
  @Autowire
  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-sprin-1

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

SOLUTION :

Difference between JPA, Hibernate, and Spring Data JPA

1. Java Persistence API (JPA)

- JPA is a specification (JSR 338) for a standard for Object Relational Mapping (ORM) in Java.
- It specifies how the Java objects are mapped to database tables and how the persistent data should be managed.
- JPA does not implement anything on its own.
- It utilizes implementations such as Hibernate, EclipseLink, etc., to actually carry out database operations.

2. Hibernate

- Hibernate is an ORM framework that follows the JPA specification.
- Provides mapping of Java objects to relational database tables.
- Provides features like lazy loading, caching, and dirty checking.
- Can be utilized with or without JPA.

3. Spring Data JPA

- Spring Data JPA is an abstraction of JPA and ORM frameworks such as Hibernate.
- Reduces boilerplate code by offering pre-built repository interfaces.
- Automatically generates queries based on method names.
- Plays nicely with Spring Boot and includes capabilities such as pagination, sorting, and custom queries.

Hibernate Code Example

```
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 Example

EmployeeRepository.java

```
public interface EmployeeRepository extends JpaRepository<Employee, Integer> { }
```

EmployeeService.java

```
@Autowired
private EmployeeRepository employeeRepository;

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

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.

CODE:

OrmLearnApplication.java

```
package com.cognizant.ormlearn;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;

import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.service.CountryService;
import com.cognizant.ormlearn.service.exception.CountryNotFoundException;

@SpringBootApplication(scanBasePackages = "com.cognizant.ormlearn")
public class OrmLearnApplication {

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

    public static void main(String[] args) {
        ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);
        countryService = context.getBean(CountryService.class);
        testFindCountryByCode();
    }

    private static void testFindCountryByCode() {
        LOGGER.info("Start");

        try {
```

```
            Country country = countryService.findCountryByCode("IN");
            LOGGER.debug("Country: {}", country);
        } catch (CountryNotFoundException e) {
            LOGGER.error("Error: {}", e.getMessage());
        }

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

Country.java

```
package com.cognizant.ormlearn.model;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

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

    @Id
    private String code;
    private String name;

    // Getters and Setters
    public String getCode() {
        return code;
    }
    public void setCode(String code) {
```

```
        this.code = code;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "Country [code=" + code + ", name=" + name + "]";
    }
}
```

CountryRepositry.java

```
package com.cognizant.ormlearn.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.cognizant.ormlearn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {
}
```

CountryService.java

```
package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.repository.CountryRepository;
import com.cognizant.ormlearn.service.exception.CountryNotFoundException;

import java.util.Optional;
```

```
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
public class CountryService {

    @Autowired
    private CountryRepository countryRepository;

    @Transactional
    public Country findCountryByCode(String code) throws CountryNotFoundException {
        Optional<Country> result = countryRepository.findById(code);
        if (!result.isPresent()) {
            throw new CountryNotFoundException("Country with code " + code + " not found.");
        }
        return result.get();
    }

    @Transactional
    public List<Country> getAllCountries() {
        return countryRepository.findAll();
    }
}
```

CountryNotFoundException.java

```
package com.cognizant.ormlearn.service.exception;

public class CountryNotFoundException extends Exception {
```

```
    public CountryNotFoundException(String message) {
        super(message);
    }
}
```

OUTPUT

```
:: Spring Boot :: (v3.5.3)

2025-07-06T18:56:02.909+05:30 INFO 26040 --- [orm-learn] [ main] c.c.ormlearn.OrmLearnApplication : Starting OrmLearnApplication using Java 23.0.2 with PID 26040 (D:\orm-learn\orm-learn\target\classes started by agara in D:\orm-learn\orm-learn)
2025-07-06T18:56:02.910+05:30 INFO 26040 --- [orm-learn] [ main] c.c.ormlearn.OrmLearnApplication : No active profile set, falling back to 1 default profile: "default"
2025-07-06T18:56:03.675+05:30 INFO 26040 --- [orm-learn] [ main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-07-06T18:56:03.732+05:30 INFO 26040 --- [orm-learn] [ main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 42 ms. Found 1 JPA repository interface.
2025-07-06T18:56:04.228+05:30 INFO 26040 --- [orm-learn] [ main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8082 (http)
2025-07-06T18:56:04.245+05:30 INFO 26040 --- [orm-learn] [ main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-07-06T18:56:04.246+05:30 INFO 26040 --- [orm-learn] [ main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.42]
2025-07-06T18:56:04.312+05:30 INFO 26040 --- [orm-learn] [ main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2025-07-06T18:56:04.314+05:30 INFO 26040 --- [orm-learn] [ main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1336 ms
2025-07-06T18:56:04.553+05:30 INFO 26040 --- [orm-learn] [ main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
```

2025-07-06T18:56:04.614+05:30 INFO 26040 --- [orm-learn] [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.6.18.Final

2025-07-06T18:56:04.636+05:30 INFO 26040 --- [orm-learn] [main] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled

2025-07-06T18:56:04.888+05:30 INFO 26040 --- [orm-learn] [main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer

2025-07-06T18:56:04.920+05:30 INFO 26040 --- [orm-learn] [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...

2025-07-06T18:56:05.291+05:30 INFO 26040 --- [orm-learn] [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@7433ca19

2025-07-06T18:56:05.293+05:30 INFO 26040 --- [orm-learn] [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.

2025-07-06T18:56:05.337+05:30 WARN 26040 --- [orm-learn] [main] org.hibernate.orm.deprecation : HHH90000025: MySQLDialect does not need to be specified explicitly using 'hibernate.dialect' (remove the property setting and it will be selected by default)

2025-07-06T18:56:05.356+05:30 INFO 26040 --- [orm-learn] [main] org.hibernate.orm.connections.pooling : HHH10001005: Database info:

Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']

Database driver: undefined/unknown

Database version: 8.0.33

Autocommit mode: undefined/unknown

Isolation level: undefined/unknown

Minimum pool size: undefined/unknown

Maximum pool size: undefined/unknown

2025-07-06T18:56:06.009+05:30 INFO 26040 --- [orm-learn] [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)

2025-07-06T18:56:06.039+05:30 INFO 26040 --- [orm-learn] [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'

2025-07-06T18:56:06.372+05:30 WARN 26040 --- [orm-learn] [main] JpaBaseConfiguration$JpaWebConfiguration : 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

2025-07-06T18:56:06.768+05:30 INFO 26040 --- [orm-learn] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8082 (http) with context path '/'

2025-07-06T18:56:06.778+05:30 INFO 26040 --- [orm-learn] [main] c.c.ormlearn.OrmLearnApplication : Started OrmLearnApplication in 4.307 seconds (process running for 4.63)

2025-07-06T18:56:06.782+05:30 INFO 26040 --- [orm-learn] [main] c.c.ormlearn.OrmLearnApplication : Start

2025-07-06T18:56:06.879+05:30 DEBUG 26040 --- [orm-learn] [main] org.hibernate.SQL : select c1_0.code,c1_0.name from country c1_0 where c1_0.code=?

2025-07-06T18:56:06.930+05:30 INFO 26040 --- [orm-learn] [main] c.c.ormlearn.OrmLearnApplication : End

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

CODE:

OrmLearnApplication.java

```
package com.cognizant.ormlearn;


import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;


import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.service.CountryService;
import com.cognizant.ormlearn.service.exception.CountryNotFoundException;


@SpringBootApplication(scanBasePackages = "com.cognizant.ormlearn")
public class OrmLearnApplication {
```

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

    public static void main(String[] args) {
        ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);
        countryService = context.getBean(CountryService.class);
        testFindCountryByCode();
       testAddCountry();
    }
private static void testAddCountry() {
    LOGGER.info("Start");
    Country newCountry = new Country();
    newCountry.setCode("ZZ");
    newCountry.setName("Zootopia");

    countryService.addCountry(newCountry);

    try {
        Country addedCountry = countryService.findCountryByCode("ZZ");
        LOGGER.debug("Added Country: {}", addedCountry);
    } catch (CountryNotFoundException e) {
        LOGGER.error("Country not found after adding: {}", e.getMessage());
    }

    LOGGER.info("End");
}

    private static void testFindCountryByCode() {
        LOGGER.info("Start");
```

```
        try {
            Country country = countryService.findCountryByCode("IN");
            LOGGER.debug("Country: {}", country);
        } catch (CountryNotFoundException e) {
            LOGGER.error("Error: {}", e.getMessage());
        }

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

Country.java

```
package com.cognizant.ormlearn.model;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

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

    @Id
    private String code;
    private String name;

    // Getters and Setters
    public String getCode() {
        return code;
```

```
    }
    public void setCode(String code) {
        this.code = code;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "Country [code=" + code + ", name=" + name + "]";
    }
}
```

CountryRepositry.java

```
package com.cognizant.ormlearn.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.cognizant.ormlearn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {
}
```

CountryService.java

```
package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.repository.CountryRepository;
import com.cognizant.ormlearn.service.exception.CountryNotFoundException;
```

```
import java.util.Optional;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
public class CountryService {

    @Autowired
    private CountryRepository countryRepository;

    @Transactional
    public Country findCountryByCode(String code) throws CountryNotFoundException {
        Optional<Country> result = countryRepository.findById(code);
        if (!result.isPresent()) {
            throw new CountryNotFoundException("Country with code " + code + " not found.");
        }
        return result.get();
    }
@Transactional
public void addCountry(Country country) {
    countryRepository.save(country);
}
    @Transactional
    public List<Country> getAllCountries() {
        return countryRepository.findAll();
    }
```

}

CountryNotFoundException.java

```
package com.cognizant.ormlearn.service.exception;


public class CountryNotFoundException extends Exception {
   public CountryNotFoundException(String message) {
      super(message);
   }
}
```

OUTPUT:

```
:: Spring Boot :: (v3.5.3)


2025-07-06T19:10:32.501+05:30 INFO 22856 --- [orm-learn] [ main] c.c.ormlearn.OrmLearnApplication : Starting OrmLearnApplication using Java 23.0.2 with PID 22856 (D:\orm-learn\orm-learn\target\classes started by agara in D:\orm-learn\orm-learn)

2025-07-06T19:10:32.505+05:30 INFO 22856 --- [orm-learn] [ main] c.c.ormlearn.OrmLearnApplication : No active profile set, falling back to 1 default profile: "default"

2025-07-06T19:10:33.229+05:30 INFO 22856 --- [orm-learn] [ main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.

2025-07-06T19:10:33.278+05:30 INFO 22856 --- [orm-learn] [ main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 40 ms. Found 1 JPA repository interface.

2025-07-06T19:10:33.842+05:30 INFO 22856 --- [orm-learn] [ main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8083 (http)

2025-07-06T19:10:33.861+05:30 INFO 22856 --- [orm-learn] [ main] o.apache.catalina.core.StandardService : Starting service [Tomcat]

2025-07-06T19:10:33.861+05:30 INFO 22856 --- [orm-learn] [ main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.42]

2025-07-06T19:10:33.935+05:30 INFO 22856 --- [orm-learn] [ main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
```

2025-07-06T19:10:33.936+05:30 INFO 22856 --- [orm-learn] [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1360 ms

2025-07-06T19:10:34.169+05:30 INFO 22856 --- [orm-learn] [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]

2025-07-06T19:10:34.230+05:30 INFO 22856 --- [orm-learn] [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.6.18.Final

2025-07-06T19:10:34.265+05:30 INFO 22856 --- [orm-learn] [main] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled

2025-07-06T19:10:34.519+05:30 INFO 22856 --- [orm-learn] [main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer

2025-07-06T19:10:34.559+05:30 INFO 22856 --- [orm-learn] [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...

2025-07-06T19:10:34.902+05:30 INFO 22856 --- [orm-learn] [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@5d221b20

2025-07-06T19:10:34.904+05:30 INFO 22856 --- [orm-learn] [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.

2025-07-06T19:10:34.961+05:30 WARN 22856 --- [orm-learn] [main] org.hibernate.orm.deprecation : HHH90000025: MySQLDialect does not need to be specified explicitly using 'hibernate.dialect' (remove the property setting and it will be selected by default)

2025-07-06T19:10:34.980+05:30 INFO 22856 --- [orm-learn] [main] org.hibernate.orm.connections.pooling : HHH10001005: Database info:

Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']

Database driver: undefined/unknown

Database version: 8.0.33

Autocommit mode: undefined/unknown

Isolation level: undefined/unknown

Minimum pool size: undefined/unknown

Maximum pool size: undefined/unknown

2025-07-06T19:10:35.712+05:30 INFO 22856 --- [orm-learn] [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)

2025-07-06T19:10:35.741+05:30 INFO 22856 --- [orm-learn] [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'

2025-07-06T19:10:36.127+05:30 WARN 22856 --- [orm-learn] [main] JpaBaseConfiguration$JpaWebConfiguration : 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

2025-07-06T19:10:36.547+05:30 INFO 22856 --- [orm-learn] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8083 (http) with context path '/'

2025-07-06T19:10:36.558+05:30 INFO 22856 --- [orm-learn] [main] c.c.ormlearn.OrmLearnApplication : Started OrmLearnApplication in 4.481 seconds (process running for 4.812)

2025-07-06T19:10:36.562+05:30 INFO 22856 --- [orm-learn] [main] c.c.ormlearn.OrmLearnApplication : Start

2025-07-06T19:10:36.645+05:30 DEBUG 22856 --- [orm-learn] [main] org.hibernate.SQL : select c1_0.code,c1_0.name from country c1_0 where c1_0.code=?

2025-07-06T19:10:36.694+05:30 INFO 22856 --- [orm-learn] [main] c.c.ormlearn.OrmLearnApplication : End

2025-07-06T19:10:36.695+05:30 INFO 22856 --- [orm-learn] [main] c.c.ormlearn.OrmLearnApplication : Start

2025-07-06T19:10:36.702+05:30 DEBUG 22856 --- [orm-learn] [main] org.hibernate.SQL : select c1_0.code,c1_0.name from country c1_0 where c1_0.code=?

2025-07-06T19:10:36.716+05:30 DEBUG 22856 --- [orm-learn] [main] org.hibernate.SQL : insert into country (name,code) values (?,?)

2025-07-06T19:10:36.728+05:30 DEBUG 22856 --- [orm-learn] [main] org.hibernate.SQL : select c1_0.code,c1_0.name from country c1_0 where c1_0.code=?

2025-07-06T19:10:36.730+05:30 INFO 22856 --- [orm-learn] [main] c.c.ormlearn.OrmLearnApplication : End