Mandatory Hands-on

Hands on 1

Spring Data JPA - Quick Example

Install the software's for the necessity of project, like mysql workbench and mysql server, eclipse.

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.

Creating a eclipse project using spring initializr, following the above guidelines.

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

Application.properties:

```
spring.application.name=orm-learn

# 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/orm_learn
spring.datasource.username=root
spring.datasource.password=Sree@1405
#FIXED: Correct dialect class name for Hibernate 6+ / Spring Boot 3.x
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
# Optional - if you want Hibernate to create tables automatically (during dev only)
# spring.jpa.hibernate.ddl-auto=update
```

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 Entity with JPA in Spring Boot

Requirements:

1. Create a country table in MySQL.
2. Create a JPA Entity class Country.
3. Create a Repository interface to fetch all countries.
4. Write a test method in the main() class to fetch and print the countries.

Database setup (mysql):

```
CREATE DATABASE ormlearn:
USE ormlearn;
CREATE TABLE country (
  code VARCHAR(2) PRIMARY KEY,
  name VARCHAR(50)
);

INSERT INTO country (code, name) VALUES
('IN', 'India'),
('US', 'United States of America'),
('UK', 'United Kingdom');
```

Create a entity class named country, src/main/java create a package com.cognizant.orm_learn and create a class country.

Country.java:

```
package com.cognizant.orm_learn.model;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
@Entity
@Table(name = "country")
public class Country {
    @Id
    @Column(name = "code")
    private String code;
    @Column(name = "name")
    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 + "]";
    }
}
```

Create interface named as CountryRepository that extends JpaRepository, src/main/java create a package com.cognizant.orm_learn.repository.

CountryRepository.java:

```
package com.cognizant.orm_learn.repository;
import com.cognizant.orm_learn.model.Country;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
@Repository
public interface CountryRepository extends JpaRepository<Country, String> {
}
```

Create a class named as CountryService, src/main/java create package com.cognizant.orm_learn.service.

CountryService,java:

```
package com.cognizant.orm_learn.service;
import com.cognizant.orm_learn.model.Country;
import com.cognizant.orm_learn.repository.CountryRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import java.util.List;
```

```
@Service
public class CountryService {
    @Autowired
    private CountryRepository countryRepository;
    @Transactional
    public List<Country> getAllCountries() {
        return countryRepository.findAll();
    }
}
```

Create main class named OrmLearnApplication, src/main/java already package and class will be there direct changing the code.

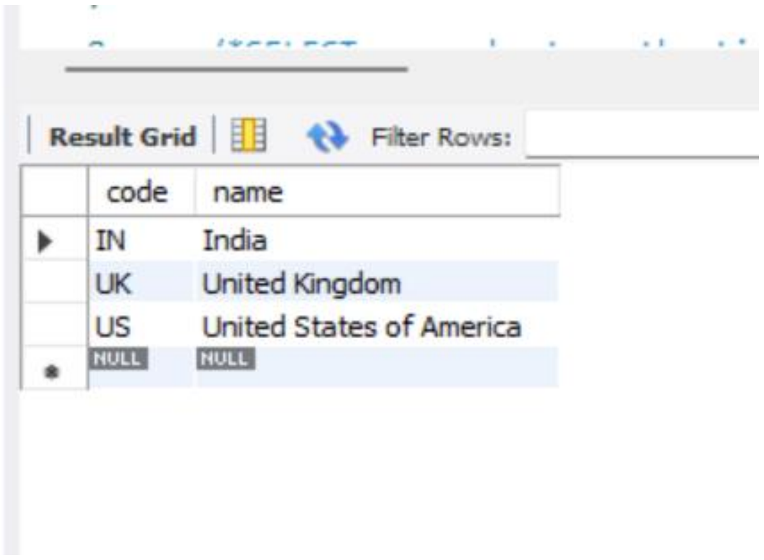
OrmLearnApplication.java:

```
package com.cognizant.orm_learn;
import com.cognizant.orm_learn.model.Country;
import com.cognizant.orm_learn.service.CountryService;
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 java.util.List;
@SpringBootApplication
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);
        LOGGER.info("Inside main");
        countryService = context.getBean(CountryService.class);
        testGetAllCountries();
    }
    private static void testGetAllCountries() {
        LOGGER.info("Start");
        List<Country> countries = countryService.getAllCountries();
        LOGGER.debug("countries={}", countries);
        LOGGER.info("End");
    }
}
```

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

It has successfully exwcuted and connected through database, and created a table for the country.

This is the table I have got after creating a table, and inserting values to it:



code	name
IN	India
UK	United Kingdom
US	United States of America
NULL	NULL

The connection that established from eclipse and through mysql workbench has successfully executed and connected.

Hands-On 4: Difference between JPA, Hibernate, and Spring Data JPA

1. Java Persistence API (JPA)

- JPA is a Java specification (JSR 338) for accessing, persisting, and managing data between Java objects and relational databases.
- It defines a set of interfaces and annotations but does not provide an actual implementation.
- JPA requires an implementation to work — Hibernate is the most popular one.

2. Hibernate

- Hibernate is a popular ORM (Object-Relational Mapping) tool that provides a concrete implementation of the JPA specification.
- It also comes with features beyond JPA like caching, lazy/eager loading, and native query support.

3. Spring Data JPA

- Spring Data JPA is not an implementation of JPA.
- Instead, it provides a higher-level abstraction over JPA implementations (like Hibernate).
- Its main goal is to reduce boilerplate code and ease repository creation.
- It integrates with Spring Framework's DI and transaction management.

4. Code Comparison

Hibernate Code (Manual Session Management)

```
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 Code (Declarative, Cleaner)

EmployeeRepository.java

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

EmployeeService.java

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

Advantages of Spring Data JPA:

- Less boilerplate
- Built-in CRUD methods
- Declarative transactions (@Transactional)
- Cleaner and more maintainable

Comparison Table:

Feature	**JPA**	**Hibernate**	**Spring Data JPA**
Type	Specification	Implementation	Abstraction over JPA
Provides API?	Yes (Interfaces)	Yes (APIs + Impl)	Yes (Auto-generated Repos)
Implementation	No	Yes	No (Delegates to JPA provider)
Boilerplate Code	High	Medium	Low
Transaction Handling	Needs manual setup	Manual/Declarative	Declarative using Spring

Additional Hands-on

Hands on 5

Implement services for managing Country

Implement the following **CRUD + search** operations for the Country entity:

1. Find a country by code
2. Add a new country
3. Update an existing country
4. Delete a country
5. Find countries matching a partial name

Entity: Country.java

```
package com.cognizant.orm_learn.repository;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import com.cognizant.orm_learn.model.Country;
public interface CountryRepository extends JpaRepository<Country, String> {
   List<Country> findByNameContainingIgnoreCase(String namePart);
}
```

Service: CountryService,java

```
package com.cognizant.orm_learn.service;
import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.cognizant.orm_learn.model.Country;
import com.cognizant.orm_learn.repository.CountryRepository;
@Service
public class CountryService {
   @Autowired
   private CountryRepository countryRepository;
```

```
    @Transactional(readOnly = true)
    public Country findCountryByCode(String code) {
        Optional<Country> optional = countryRepository.findById(code);
        return optional.orElseThrow(() -> new RuntimeException("Country not found with code: " +
code));
    }
    @Transactional
    public void addCountry(Country country) {
        countryRepository.save(country);
    }
    @Transactional
    public void updateCountry(Country country) {
        if (!countryRepository.existsById(country.getCode())) {
            throw new RuntimeException("Cannot update. Country with code " + country.getCode() + "
not found.");
        }
        countryRepository.save(country);
    }
    @Transactional
    public void deleteCountry(String code) {
        if (!countryRepository.existsById(code)) {
            throw new RuntimeException("Cannot delete. Country with code " + code + " not found.");
        }
        countryRepository.deleteById(code);
    }
    @Transactional(readOnly = true)
    public List<Country> findCountriesByName(String namePart) {
        return countryRepository.findByNameContainingIgnoreCase(namePart);
    }
}
```

Test in OrmLearnApplication.java:

```
package com.cognizant.orm_learn;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import com.cognizant.orm_learn.model.Country;
import com.cognizant.orm_learn.service.CountryService;
import jakarta.annotation.PostConstruct;
@SpringBootApplication
public class OrmLearnApplication {
  @Autowired
  private CountryService countryService;
  public static void main(String[] args) {
    SpringApplication.run(OrmLearnApplication.class, args);
  }
  @PostConstruct
  public void testCountryOperations() {
    System.out.println("----- Find IN -----");
    System.out.println(countryService.findCountryByCode("IN"));
    System.out.println("----- Add ZZ -----");
    countryService.addCountry(new Country("ZZ", "Zootopia"));
    System.out.println(countryService.findCountryByCode("ZZ"));
    System.out.println("----- Update ZZ -----");
    countryService.updateCountry(new Country("ZZ", "Zanzibar"));
    System.out.println(countryService.findCountryByCode("ZZ"));
    System.out.println("----- Search 'land' -----");
    List<Country> matches = countryService.findCountriesByName("land");
    matches.forEach(System.out::println);
    System.out.println("----- Delete ZZ -----");
    countryService.deleteCountry("ZZ");
    }
}
```

application.properties:

```
spring.datasource.url=jdbc:mysql://localhost:3306/your_database_name
spring.datasource.username=root
spring.datasource.password=your_password
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.show-sql=true
```

Hands on 6

Find a country based on country code

1. **Create custom exception:**

src/main/java/com/cognizant/orm_learn/service/exception/CountryNotFoundException.java

```
package com.cognizant.orm_learn.service.exception;
public class CountryNotFoundException extends Exception {
    public CountryNotFoundException(String message) {
        super(message);
    }
}
```

2. **CountryService.java:**
   Updated CountryService.java class

```
package com.cognizant.orm_learn.service;
import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.cognizant.orm_learn.model.Country;
import com.cognizant.orm_learn.repository.CountryRepository;
import com.cognizant.orm_learn.service.exception.CountryNotFoundException;
@Service
public class CountryService {
    @Autowired
    private CountryRepository countryRepository;
    @Transactional(readOnly = true)
    public Country findCountryByCode(String code) throws CountryNotFoundException {
        Optional<Country> optional = countryRepository.findById(code);
        if (!optional.isPresent()) {
            throw new CountryNotFoundException("Country not found with code: " + code);
        }
        return optional.get();
    }
    @Transactional
    public void addCountry(Country country) {
        countryRepository.save(country);
    }
    @Transactional
    public void updateCountry(Country country) throws CountryNotFoundException {
        if (!countryRepository.existsById(country.getCode())) {
            throw new CountryNotFoundException("Cannot update. Country with code " +
country.getCode() + " not found.");
        }
        countryRepository.save(country);
    }
    @Transactional
    public void deleteCountry(String code) throws CountryNotFoundException {
        if (!countryRepository.existsById(code)) {
            throw new CountryNotFoundException("Cannot delete. Country with code " + code + "
not found.");
        }
        countryRepository.deleteById(code);
    }
    @Transactional(readOnly = true)
    public List<Country> findCountriesByName(String namePart) {
        return countryRepository.findByNameContainingIgnoreCase(namePart);
    }
}
```

3. **OrmLearnApplication.java:**
   Updated main class

```
package com.cognizant.orm_learn;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import com.cognizant.orm_learn.model.Country;
import com.cognizant.orm_learn.service.CountryService;
import com.cognizant.orm_learn.service.exception.CountryNotFoundException;
import org.springframework.context.ApplicationContext;
```

```
@SpringBootApplication
public class OrmLearnApplication {
   private static CountryService countryService;
   private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);
   public static void main(String[] args) {
      ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);
      countryService = context.getBean(CountryService.class);

      testFindCountryByCode(); // Call the test method
      testFindInvalidCountryCode();
   }
   private static void testFindCountryByCode() {
      LOGGER.info("Start");
      try {
         Country country = countryService.findCountryByCode("IN");
         LOGGER.debug("Country: {}", country);
      } catch (CountryNotFoundException e) {
         LOGGER.error("Exception: {}", e.getMessage());
      }
      LOGGER.info("End");
   }
   private static void testFindInvalidCountryCode() {
      LOGGER.info("Start");
      try {
         Country country = countryService.findCountryByCode("ZZ");
         LOGGER.debug("Country: {}", country);
      } catch (CountryNotFoundException e) {
         LOGGER.error("Exception: {}", e.getMessage());
      }
      LOGGER.info("End");
   }
}
```

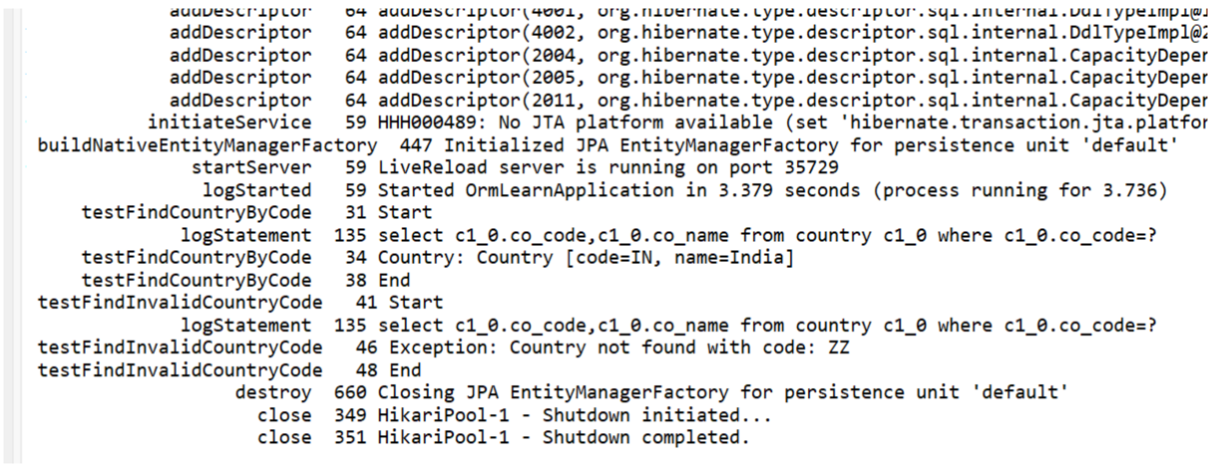
4. **Test the application:**
   Testing the main class named OrmLearnApplication.java, run as>>java application.

   This is the output from console:

```
logStatement  135 select c1_0.co_code,c1_0.co_name from country c1_0 where c1_0.co_code=?
04-07-25 22:01:59.880 restartedMain        DEBUG c.c.o.OrmLearnApplication
testFindCountryByCode   34 Country: Country [code=IN, name=India]
04-07-25 22:01:59.880 restartedMain         INFO c.c.o.OrmLearnApplication
testFindCountryByCode   38 End

logStatement  135 select c1_0.co_code,c1_0.co_name from country c1_0 where c1_0.co_code=?
04-07-25 22:01:59.885 restartedMain        ERROR c.c.o.OrmLearnApplication
testFindInvalidCountryCode   46 Exception: Country not found with code: ZZ  04-07-25
```

22:01:59.885 restartedMain INFO c.c.o.OrmLearnApplication testFindInvalidCountryCode 48 End



```
            addDescriptor   64 addDescriptor(4001, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@
            addDescriptor   64 addDescriptor(4002, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@
            addDescriptor   64 addDescriptor(2004, org.hibernate.type.descriptor.sql.internal.CapacityDeper
            addDescriptor   64 addDescriptor(2005, org.hibernate.type.descriptor.sql.internal.CapacityDeper
            addDescriptor   64 addDescriptor(2011, org.hibernate.type.descriptor.sql.internal.CapacityDeper
          initiateService   59 HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platfor
buildNativeEntityManagerFactory  447 Initialized JPA EntityManagerFactory for persistence unit 'default'
              startServer   59 LiveReload server is running on port 35729
               logStarted   59 Started OrmLearnApplication in 3.379 seconds (process running for 3.736)
    testFindCountryByCode   31 Start
             logStatement  135 select c1_0.co_code,c1_0.co_name from country c1_0 where c1_0.co_code=?
    testFindCountryByCode   34 Country: Country [code=IN, name=India]
    testFindCountryByCode   38 End
testFindInvalidCountryCode   41 Start
             logStatement  135 select c1_0.co_code,c1_0.co_name from country c1_0 where c1_0.co_code=?
testFindInvalidCountryCode   46 Exception: Country not found with code: ZZ
testFindInvalidCountryCode   48 End
                  destroy  660 Closing JPA EntityManagerFactory for persistence unit 'default'
                    close  349 HikariPool-1 - Shutdown initiated...
                    close  351 HikariPool-1 - Shutdown completed.
```