Digital Nurture 4.0 Deep Skilling - Java FSE
WEEK –3 Additional Hands-on Exercises
Module 5 - Spring Core and Maven

1. Exercise 5: Configuring the Spring IoC Container

Scenario:

The library management application requires a central configuration for beans and dependencies.

Solution:

Code:

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.library</groupId>
  <artifactId>LibraryManagement</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>LibraryManagement</name>
  <url>http://www.example.com</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
```

```
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-engine</artifactId>
      <version>5.7.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-params</artifactId>
      <version>5.7.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.33</version>
</dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
      </plugin>
    </plugins>
  </build>
</project>
```

BookRepository.java

```
package com.library.repository;
public class BookRepository {
    public void saveBook(String bookName) {
        System.out.println("BookRepository: Book saved - " + bookName);
    }
}
```

BookService.java

```
package com.library.service;
import com.library.repository.BookRepository;
public class BookService {
    private BookRepository bookRepository;
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }
    public void addBook(String bookName) {
        System.out.println("BookService: Adding book - " + bookName);
        bookRepository.saveBook(bookName);
    }
}
```

applicationContext.xml

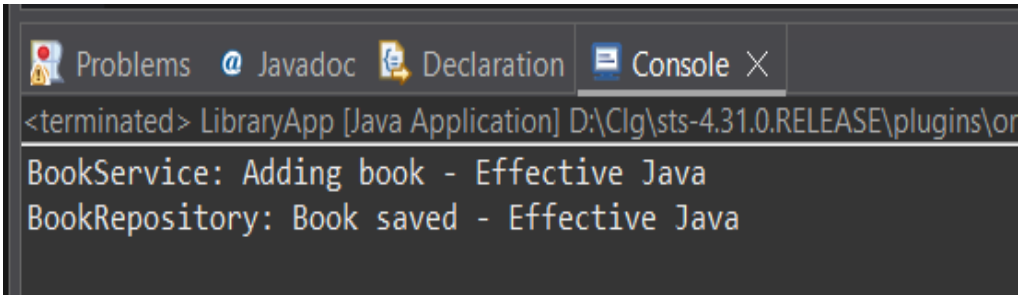
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       https://www.springframework.org/schema/beans/spring-beans.xsd">
     <bean id="bookRepository" class="com.library.repository.BookRepository" />
```

```
    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository" />
    </bean>
</beans>
```

LibraryApp.java

```
package com.library;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.library.service.BookService;
public class LibraryApp {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
        BookService service = (BookService) context.getBean("bookService");
        service.addBook("Effective Java");
    }
}
```

Output:



Explanation:

1. Create applicationContext.xml in src/main/resources.
2. Define beans for BookService and BookRepository in the XML file.
3. Add a setter method in BookService to accept BookRepository.
4. Create a main class to load the Spring context using ClassPathXmlApplicationContext.
5. Retrieve BookService bean from the context and test the configuration.

2. Exercise 7: Implementing Constructor and Setter Injection

Scenario:

The library management application requires both constructor and setter injection for better control over bean initialization.

Solution:

Code:

applicationContext.xml

```
 <?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

       xsi:schemaLocation="http://www.springframework.org/schema/beans

       https://www.springframework.org/schema/beans/spring-beans.xsd">

     <bean id="bookRepository" class="com.library.repository.BookRepository" />

    <bean id="bookService" class="com.library.service.BookService">

        <constructor-arg value="LibraryService v1.0"/>

         <property name="bookRepository" ref="bookRepository"/>

    </bean>

</beans>
```

BookService.java

```
package com.library.service;
```

```
import com.library.repository.BookRepository;

public class BookService {

    private String serviceName; // constructor injection

    private BookRepository bookRepository; // setter injection

    public BookService(String serviceName) {

        this.serviceName = serviceName;

        System.out.println("Constructor injected: " + serviceName);

    }

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook(String bookName) {

        System.out.println(serviceName + " - Adding Book: " + bookName);

        bookRepository.saveBook(bookName);

    }

}
```

BookRepository.java

```
package com.library.repository;

public class BookRepository {

    public void saveBook(String bookName) {
```

```
        System.out.println("BookRepository: Saved book - " + bookName);
    }
}
```

LibraryApp.java

```
package com.library;
import com.library.service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class LibraryApp {
    public static void main(String[] args) {
        ApplicationContext context = new
 ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService = (BookService) context.getBean("bookService");
        bookService.addBook("Clean Code");
    }
}
```

Output:

```
Problems  @ Javadoc  Declaration  Console ×
<terminated> LibraryApp [Java Application] D:\Clg\sts-4.31.0.RELEASE\plugins\org.ecli
Constructor injected: LibraryService v1.0
LibraryService v1.0 - Adding Book: Clean Code
BookRepository: Saved book - Clean Code
```

Explanation:

1. In applicationContext.xml, configure constructor injection for the BookService bean using the <constructor-arg> tag.
2. Ensure the BookService class has a setter method for BookRepository.
3. Configure setter injection in applicationContext.xml using the <property> tag.
4. Run the LibraryManagementApplication main class to verify that both constructor and setter injection work correctly.

3.Exercise 9: Creating a Spring Boot Application

Scenario:

You need to create a Spring Boot application for the library management system to simplify configuration and deployment.

Solution:

Code:

applicationContext.xml

```
spring.application.name=LibraryManagement2

spring.datasource.url=jdbc:h2:mem:librarydb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

spring.jpa.hibernate.ddl-auto=update
```

Book.java

```
package com.example.Library;
import jakarta.persistence.*;
@Entity
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;
    private String title;
    private String author;
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }
    public String getAuthor() { return author; }
    public void setAuthor(String author) { this.author = author; }
}
```

BookRepository.java

```
package com.example.Library.repository;

import com.example.Library.Book;

import org.springframework.data.jpa.repository.JpaRepository;

public interface BookRepository extends JpaRepository<Book, Long> {

}
```

BookController.java

```
package com.example.Library.controller;

import com.example.Library.Book;

import com.example.Library.repository.BookRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.*;
```

```
import java.util.List;

@RestController

@RequestMapping("/books")

public class BookController {

    @Autowired

    private BookRepository repository;

    @GetMapping

    public List<Book> getAllBooks() {

        return repository.findAll();

    }

    @PostMapping

    public Book addBook(@RequestBody Book book) {

        return repository.save(book);

    }

    @GetMapping("/{id}")

    public Book getBook(@PathVariable Long id) {

        return repository.findById(id).orElse(null);

    }

    @PutMapping("/{id}")

    public Book updateBook(@PathVariable Long id, @RequestBody Book updatedBook)
{
```

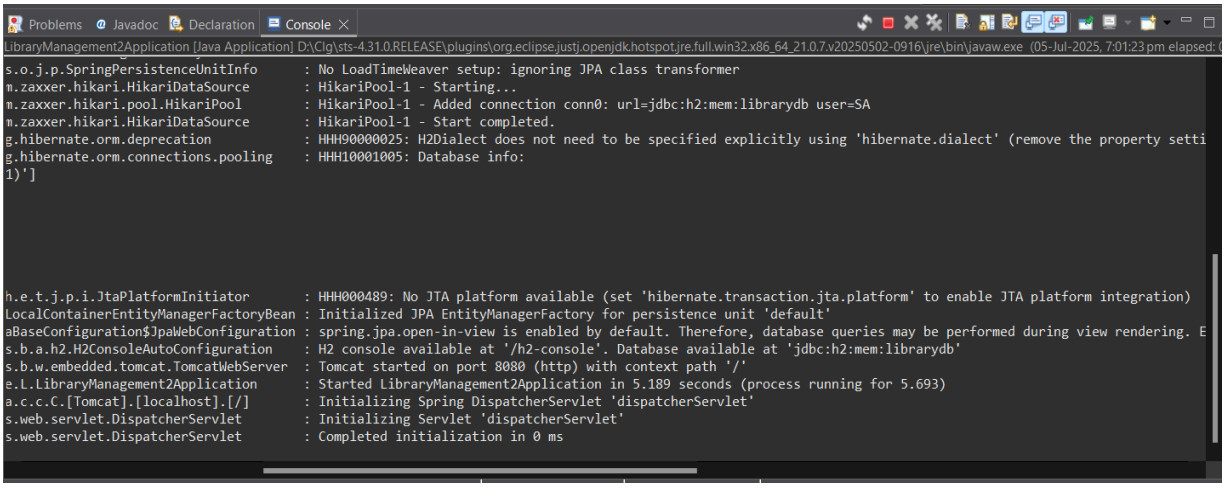
```
        Book book = repository.findById(id).orElse(null);
        if (book != null) {
            book.setTitle(updatedBook.getTitle());
            book.setAuthor(updatedBook.getAuthor());
            return repository.save(book);
        }
        return null;
    }
    @DeleteMapping("/{id}")
    public void deleteBook(@PathVariable Long id) {
        repository.deleteById(id);
    }
}
```

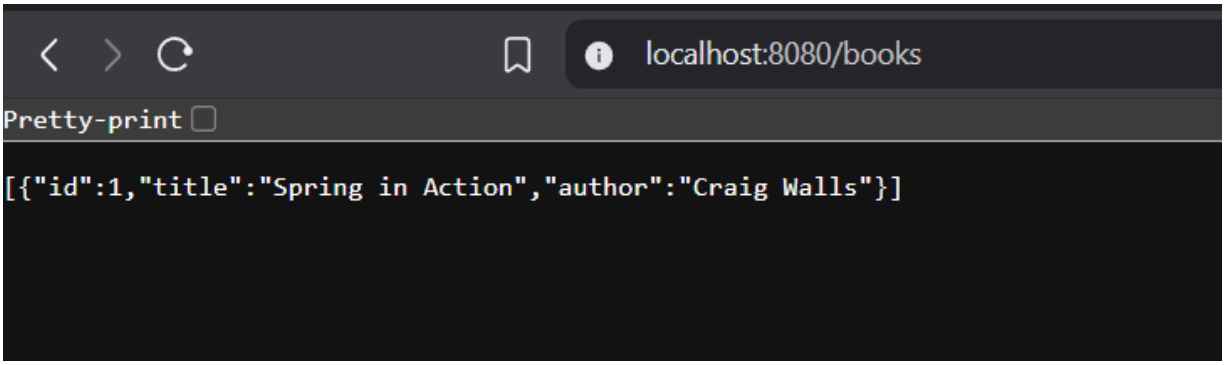
BookController.java

```
package com.example.Library;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class LibraryManagement2Application {
    public static void main(String[] args) {
```

```
        SpringApplication.run(LibraryManagement2Application.class, args);

    }

}
```

Output:





Explanation:

1. Use Spring Initializr to create a new project named LibraryManagement.
2. Add dependencies: Spring Web, Spring Data JPA, H2 Database.
3. Configure database settings in application.properties.
4. Create the Book entity and BookRepository interface.
5. Implement BookController to expose REST endpoints for CRUD operations.
6. Run the Spring Boot application and test the API using Postman or browser.

Digital Nurture 4.0 Deep Skilling - Java FSE

WEEK –3 Hands-on Exercises

Module 5 - Spring Core and Maven

1. Exercise 1: Configuring a Basic Spring Application

Scenario:

Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

Solution:

Spring Core:

Spring Core is the foundational module of the Spring Framework, which provides the basic features of dependency injection (DI) and inversion of control (IoC) .

Maven:

Maven is a build automation and dependency management tool used in Java projects. It uses an XML file called pom.xml.

Code:

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.library</groupId>
  <artifactId>LibraryManagement</artifactId>
```

```
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>LibraryManagement</name>
  <url>http://www.example.com</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-engine</artifactId>
      <version>5.7.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-params</artifactId>
      <version>5.7.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.33</version>
</dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
```

```
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
      </plugin>
    </plugins>
  </build>
</project>
```

BookRepository.java

```
package com.library.repository;

public class BookRepository {
    public void saveBook(String bookName) {
        System.out.println("Book saved: " + bookName);
    }
}
```

BookService.java

```
package com.library.service;
import com.library.repository.BookRepository;
public class BookService {
    private BookRepository bookRepository;
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }
    public void addBook(String bookName) {
        System.out.println("Adding book: " + bookName);
        bookRepository.saveBook(bookName);
    }
}
```

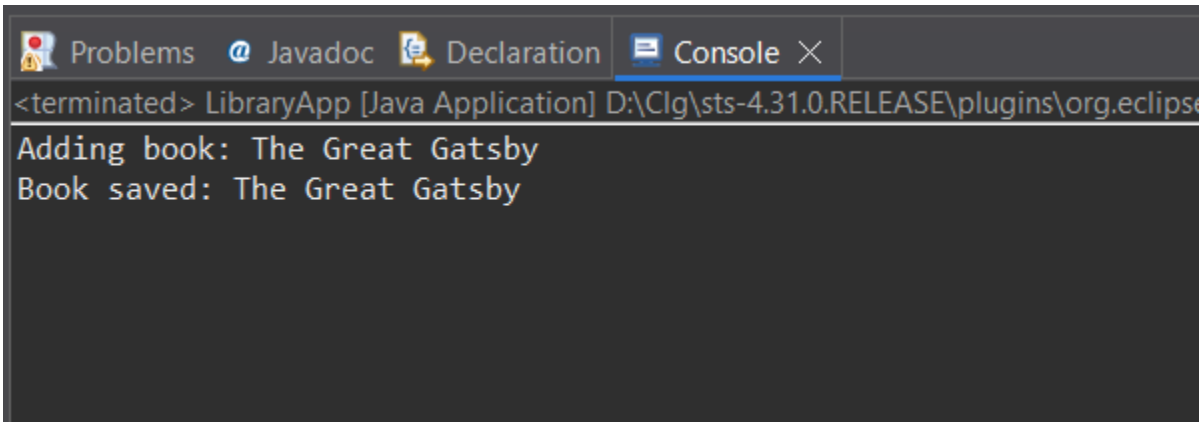
applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="
         http://www.springframework.org/schema/beans
         https://www.springframework.org/schema/beans/spring-beans.xsd">

   <bean id="bookRepository" class="com.library.repository.BookRepository" />

   <bean id="bookService" class="com.library.service.BookService">
      <property name="bookRepository" ref="bookRepository" />
   </bean>

</beans>
```

LibraryApp.java

```
package com.library.LibraryManagement;
import com.library.service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class LibraryApp {
   public static void main(String[] args) {
      ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
      BookService bookService = (BookService) context.getBean("bookService");
      bookService.addBook("The Great Gatsby");
   }
}
```

Output:



Explanation:

1. Create a Maven project named LibraryManagement in your IDE.
2. Add Spring Core dependencies in the pom.xml file.
3. Create an XML file named applicationContext.xml inside src/main/resources.
4. Define beans for BookService and BookRepository in the XML configuration.
5. Create the BookService class inside the com.library.service package.
6. Create the BookRepository class inside the com.library.repository package.
7. Create a main class to load the Spring application context and test the configured beans.

2. Exercise 2: Implementing Dependency Injection

Scenario:

In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

Solution:

Code:

applicationContext.xml

```
 <?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

      xsi:schemaLocation="

         http://www.springframework.org/schema/beans

         https://www.springframework.org/schema/beans/spring-beans.xsd">

   <bean id="bookRepository" class="com.library.repository.BookRepository" />

   <bean id="bookService" class="com.library.service.BookService">

      <property name="bookRepository" ref="bookRepository" />

   </bean>

</beans>
```

BookService.java

```
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

   private BookRepository bookRepository;
```

```
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }
    public void addBook(String bookName) {
        System.out.println("Adding book: " + bookName);
        bookRepository.saveBook(bookName);
    }
}
```

LibraryApp.java

```
package com.library;
import com.library.service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class LibraryApp{
    public static void main(String[] args) {
        ApplicationContext context = new
 ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService = (BookService) context.getBean("bookService");
        bookService.addBook("Spring in Action");
    }
```

```
}
```

Output:

```
Problems  @ Javadoc  Declaration  Console ×
<terminated> LibraryApp [Java Application] D:\Clg\sts-4.31.0.RELEASE\plugins\org.ecli
Adding book: Spring in Action
Book saved: Spring in Action
```

Explanation:

1. Update the applicationContext.xml to define a bean for BookService and inject BookRepository into it using a setter.
2. Add a setter method in the BookService class to receive the BookRepository dependency.
3. Run the LibraryApp main class to load the Spring context and confirm that dependency injection is working correctly.

3. Exercise 4: Creating and Configuring a Maven Project

Scenario:

You need to set up a new Maven project for the library management application and add Spring dependencies.

Solution:

Code:

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                    http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>


    <groupId>com.library</groupId>
    <artifactId>LibraryManagement</artifactId>
    <version>1.0-SNAPSHOT</version>


    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>


    <dependencies>
```

```
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.33</version>
    </dependency>



    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.9.3</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>5.9.3</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

```
<repositories>
    <repository>
        <id>central</id>
        <url>https://repo.maven.apache.org/maven2</url>
    </repository>
</repositories>


<build>
    <plugins>


        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
            </configuration>
        </plugin>
    </plugins>
```

```
    </build>

</project>
```

Output:

```
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  39.576 s
[INFO] Finished at: 2025-07-04T15:43:23+05:30
[INFO] ------------------------------------------------------------------------
```

Explanation:

1. Create a new Maven project named LibraryManagement in your IDE.
2. Add dependencies for Spring Context, Spring AOP, and Spring WebMVC in the pom.xml file.
3. Configure the Maven Compiler Plugin in pom.xml to set the Java version to 1.8.

Digital Nurture 4.0 Deep Skilling - Java FSE

WEEK –3 Hands-on Exercises

Module 6 - Spring Data JPA with Spring Boot, Hibernate

1. Hands on 1: Spring Data JPA - Quick Example

Solution:

Code:

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <parent>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
      <version>3.5.3</version>
      <relativePath/> <!-- lookup parent from repository -->
   </parent>
   <groupId>com.example</groupId>
   <artifactId>ormlearn</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <name>ormlearn</name>
   <description>Demo project for Spring Boot</description>
   <properties>
      <java.version>17</java.version>
   </properties>
```

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <version>8.0.33</version>
        <scope>runtime</scope>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
```

```
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
  </build>
</project>
```

application.properties

```
spring.application.name=ormlearn
# Logging
logging.level.org.springframework=info
logging.level.com.example.ormlearn=debug
logging.level.org.hibernate.SQL=debug
logging.level.org.hibernate.type.descriptor.sql=trace
# Database
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=Shri19
# Hibernate
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
```

Country.java

```
package com.example.ormlearn.model;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import jakarta.persistence.Column;
@Entity
@Table(name = "country")
public class Country {
```

```
    @Id
    @Column(name = "co_code")
    private String code;
    @Column(name = "co_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 + "]";
    }
}
```

CountryRepository.java

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

CountryService.java

```
package com.example.ormlearn.service;
import java.util.List;
import jakarta.transaction.Transactional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.example.ormlearn.model.Country;
import com.example.ormlearn.repository.CountryRepository;
@Service
public class CountryService {
    @Autowired
    private CountryRepository countryRepository;
    @Transactional
    public List<Country> getAllCountries() {
        return countryRepository.findAll();
    }
}
```

OrmlearnApllication.java

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

Output:

```
Problems  Javadoc  Declaration  Console
<terminated> ormlearn - OrmlearnApplication [Spring Boot App] D:\Clg\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (05-Jul-2025, 11:51:4

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.3)

2025-07-05T11:51:50.276+05:30  INFO 10004 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Starting OrmlearnApplication using Java
2025-07-05T11:51:50.278+05:30 DEBUG 10004 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Running with Spring Boot v3.5.3, Spring
2025-07-05T11:51:50.279+05:30  INFO 10004 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : No active profile set, falling back to
2025-07-05T11:51:50.326+05:30  INFO 10004 --- [ormlearn] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set
2025-07-05T11:51:50.731+05:30  INFO 10004 --- [ormlearn] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositor
2025-07-05T11:51:50.784+05:30  INFO 10004 --- [ormlearn] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scannir
2025-07-05T11:51:51.087+05:30  INFO 10004 --- [ormlearn] [  restartedMain] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitIr
2025-07-05T11:51:51.143+05:30  INFO 10004 --- [ormlearn] [  restartedMain] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 6
2025-07-05T11:51:51.177+05:30  INFO 10004 --- [ormlearn] [  restartedMain] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache disabled
2025-07-05T11:51:51.435+05:30  INFO 10004 --- [ormlearn] [  restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA c
2025-07-05T11:51:51.470+05:30  INFO 10004 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2025-07-05T11:51:51.855+05:30  INFO 10004 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mys
2025-07-05T11:51:51.857+05:30  INFO 10004 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
```

```
Problems  Javadoc  Declaration  Console
<terminated> ormlearn - OrmlearnApplication [Spring Boot App] D:\Clg\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (05-Jul-2025, 11:51

tedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(12, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@983e067) replaced p
tedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(-9, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@1cf58ca4) replaced
tedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(-3, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@58b6a0f2) replaced
tedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4003, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@3432faf0) replaced previous re
tedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4001, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@100be7ae) replaced previous re
tedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4002, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@21fedd8e) replaced previous re
tedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(2004, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@28d824f3) replace
tedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(2005, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@3f7773c4) replace
tedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(2011, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@3de2ac3c) replace
tedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform int
tedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
tedMain] o.s.b.d.a.OptionalLiveReloadServer       : Unable to start LiveReload server
tedMain] c.example.ormlearn.OrmlearnApplication   : Started OrmlearnApplication in 3.213 seconds (process running for 3.934)
tedMain] c.example.ormlearn.OrmlearnApplication   : Inside main
tedMain] c.example.ormlearn.OrmlearnApplication   : Start
tedMain] org.hibernate.SQL                        : select c1_0.co_code,c1_0.co_name from country c1_0
tedMain] c.example.ormlearn.OrmlearnApplication   : countries=[Country [code=IN, name=India], Country [code=US, name=United States of America]]
tedMain] c.example.ormlearn.OrmlearnApplication   : End
ownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
ownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
ownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
```

```
j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
o.s.b.d.a.OptionalLiveReloadServer       : Unable to start LiveReload server
c.example.ormlearn.OrmlearnApplication   : Started OrmlearnApplication in 3.213 seconds (process running for 3.934)
c.example.ormlearn.OrmlearnApplication   : Inside main
c.example.ormlearn.OrmlearnApplication   : Start
org.hibernate.SQL                        : select c1_0.co_code,c1_0.co_name from country c1_0
c.example.ormlearn.OrmlearnApplication   : countries=[Country [code=IN, name=India], Country [code=US, name=United States of America]]
c.example.ormlearn.OrmlearnApplication   : End
j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
```

Explanation:

1. Project Setup

   Use Spring Initializr to create a Maven project with group com.cognizant and artifact orm-learn.
   Select dependencies: Spring Boot DevTools, Spring Data JPA, MySQL Driver.
   Import the generated project into Eclipse.

2. MySQL Configuration
   Create schema ormlearn in MySQL.
   Configure database properties and logging in application.properties.
3. Build the Project
   Use mvn clean package with proxy settings if required.
4. Main Application Setup
   Add SLF4J logger to OrmLearnApplication.
   Run the application and verify logger output from main().
5. Project Structure Overview
   - src/main/java: Java application code.
   - src/main/resources: Config files (application.properties).
   - src/test/java: Unit tests.
   - @SpringBootApplication: Enables component scanning, configuration, and auto-configuration.
   - pom.xml: Contains all project dependencies and build configuration.
6. Database Table Creation
   Create country table with co_code and co_name.
   Insert sample records: IN - India, US - United States of America.
7. Entity Class: Country
   Annotate with @Entity and @Table(name="country").
   Use @Id for primary key and @Column for mapping fields.
8. Repository Interface: CountryRepository
   Extend JpaRepository<Country, String>.
   Annotate with @Repository.
9. Service Class: CountryService
   Annotate with @Service.
   Autowire CountryRepository.
   Add method getAllCountries() with @Transactional.
10. Testing in OrmLearnApplication
    Autowire CountryService using ApplicationContext.
    Add and call testGetAllCountries() to print country list using logger.

2. Hands on 4:Difference between JPA, Hibernate and Spring Data JPA

Solution:

Code:

Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
          http://maven.apache.org/POM/4.0.0
          http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>jpa-hibernate-employee</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>
    <name>jpa-hibernate-employee</name>
    <description>Spring Boot app using Spring Data JPA and MySQL</description>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.2.4</version> <!-- You can update to latest -->
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
```

```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
     <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <scope>runtime</scope>
    </dependency>
     <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
     <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.11.0</version>
```

```
            <configuration>
                <source>17</source>
                <target>17</target>
            </configuration>
        </plugin>
    </plugins>
</build>

</project>
```

application.properties

```
spring.application.name=jpahibernateemployee
# MySQL Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/test_db
spring.datasource.username=root
spring.datasource.password=Shri19
# JPA Properties
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

Employee.java

```
package com.example.employee.model;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
@Entity
public class Employee {
    @Id
    private int id;
    private String name;
    private String department;
```

```
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDepartment() {
        return department;
    }
    public void setDepartment(String department) {
        this.department = department;
    }
}
```

EmployeeRepository.java

```
package com.example.employee.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import com.example.employee.model.Employee;
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {
}
```

EmployeeService.java

```
package com.example.employee.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```
import org.springframework.transaction.annotation.Transactional;
import com.example.employee.model.Employee;
import com.example.employee.repository.EmployeeRepository;
@Service
public class EmployeeService {
   @Autowired
   private EmployeeRepository employeeRepository;
   @Transactional
   public void addEmployee(Employee employee) {
      employeeRepository.save(employee);
   }
}
```

JpahibernateemployeeApllication.java

```
package com.example.employee;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import com.example.employee.model.Employee;
import com.example.employee.service.EmployeeService;
@SpringBootApplication
public class JpahibernateemployeeApplication implements CommandLineRunner {
   @Autowired
   private EmployeeService employeeService;
   public static void main(String[] args) {
      SpringApplication.run(JpahibernateemployeeApplication.class, args);
   }
   @Override
   public void run(String... args) throws Exception {
      Employee emp = new Employee();
```

```
        emp.setId(101);
        emp.setName("John");
        emp.setDepartment("IT");
        employeeService.addEmployee(emp);
    }
}
```

Output:

```
Problems  Javadoc  Declaration  Console
jpahibernateemployee - JpahibernateemployeeApplication [Spring Boot App] D:\Clg\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (05-Ju

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.2.4)

2025-07-05T12:20:01.267+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] c.e.e.JpahibernateemployeeApplication    : Starting Jpahibernateemploy
2025-07-05T12:20:01.270+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] c.e.e.JpahibernateemployeeApplication    : No active profile set, fall
2025-07-05T12:20:01.331+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults
2025-07-05T12:20:01.332+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related
2025-07-05T12:20:01.957+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data J
2025-07-05T12:20:02.014+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data reposi
2025-07-05T12:20:02.482+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with por
2025-07-05T12:20:02.493+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-07-05T12:20:02.493+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [A
2025-07-05T12:20:02.549+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedde
2025-07-05T12:20:02.549+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext:
2025-07-05T12:20:02.694+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing Persi
2025-07-05T12:20:02.751+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] org.hibernate.Version                    : HHH000412: Hibernate ORM cc
2025-07-05T12:20:02.782+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cac
```

```
2025-07-05T12:20:02.549+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedde
2025-07-05T12:20:02.549+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext:
2025-07-05T12:20:02.694+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing Persi
2025-07-05T12:20:02.751+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] org.hibernate.Version                    : HHH000412: Hibernate ORM cc
2025-07-05T12:20:02.782+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cac
2025-07-05T12:20:02.989+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ig
2025-07-05T12:20:03.014+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2025-07-05T12:20:03.364+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connec
2025-07-05T12:20:03.366+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start comple
2025-07-05T12:20:03.415+05:30  WARN 24264 --- [jpahibernateemployee] [  restartedMain] org.hibernate.orm.deprecation            : HHH90000025: MySQL8Dialect
2025-07-05T12:20:03.416+05:30  WARN 24264 --- [jpahibernateemployee] [  restartedMain] org.hibernate.orm.deprecation            : HHH90000026: MySQL8Dialect
2025-07-05T12:20:04.155+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform
2025-07-05T12:20:04.203+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManag
2025-07-05T12:20:04.501+05:30  WARN 24264 --- [jpahibernateemployee] [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is
2025-07-05T12:20:04.765+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer        : LiveReload server is runnir
2025-07-05T12:20:04.807+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080
2025-07-05T12:20:04.816+05:30  INFO 24264 --- [jpahibernateemployee] [  restartedMain] c.e.e.JpahibernateemployeeApplication    : Started Jpahibernateemploye
Hibernate: select e1_0.id,e1_0.department,e1_0.name from employee e1_0 where e1_0.id=?
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	name	department
101	John	IT
NULL	NULL	NULL

Explanation:

1. Understand that **JPA** is a specification (no implementation).
2. Recognize that **Hibernate** is an ORM tool that implements JPA.
3. Know that **Spring Data JPA** is an abstraction layer over JPA providers like Hibernate.
4. Compare coding styles:
   - Hibernate requires manual session and transaction handling.
   - Spring Data JPA uses JpaRepository for CRUD with minimal code.
5. Use annotations like @Autowired and @Transactional in Spring Data JPA for dependency injection and transaction management.
6. Leverage Spring Boot's auto-configuration to manage repositories without XML or manual configuration.

Digital Nurture 4.0 Deep Skilling - Java FSE

WEEK –3 Additional Hands-on Exercises

Module 6 - Spring Data JPA with Spring Boot, Hibernate

I. spring-data-jpa-handson

1. Hands on 5:Implement services for managing Country

Solution:

Code:

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <parent>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
      <version>3.5.3</version>
      <relativePath/> <!-- lookup parent from repository -->
   </parent>
   <groupId>com.example</groupId>
   <artifactId>ormlearn</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <name>ormlearn</name>
   <description>Demo project for Spring Boot</description>
   <properties>
```

```
    <java.version>17</java.version>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <version>8.0.33</version>
        <scope>runtime</scope>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <plugins>
```

```
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
  </build>
</project>
```

application.properties

```
spring.application.name=ormlearn
# Logging
logging.level.org.springframework=info
logging.level.com.example.ormlearn=debug
logging.level.org.hibernate.SQL=debug
logging.level.org.hibernate.type.descriptor.sql=trace
# Database
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=Shri19
# Hibernate
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
```

Country.java

```
package com.example.ormlearn.model;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import jakarta.persistence.Column;
@Entity
```

```
@Table(name = "country")
public class Country {
    @Id
    @Column(name = "co_code")
    private String code;
    @Column(name = "co_name")
    private String name;
    public Country() {}
    public Country(String code, String name) {
        this.code = code;
        this.name = 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; }
}
```

CountryRepository.java

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

CountryService.java

```
package com.example.ormlearn.service;
import com.example.ormlearn.model.Country;
```

```
import com.example.ormlearn.repository.CountryRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
import java.util.Optional;
@Service
public class CountryService {
    @Autowired
    private CountryRepository countryRepository;
    public Country findCountryByCode(String code) throws Exception {
        Optional<Country> result = countryRepository.findById(code);
        if (result.isEmpty()) throw new Exception("Country Not Found");
        return result.get();
    }
    public void addCountry(Country country) {
        countryRepository.save(country);
    }
    public void updateCountry(Country country) throws Exception {
        if (!countryRepository.existsById(country.getCode())) {
            throw new Exception("Country Not Found");
        }
        countryRepository.save(country);
    }
    public void deleteCountry(String code) throws Exception {
        if (!countryRepository.existsById(code)) {
            throw new Exception("Country Not Found");
        }
        countryRepository.deleteById(code);
    }
    public List<Country> searchCountriesByPartialName(String partialName) {
        return countryRepository.findByNameContaining(partialName);
```

```
    }
}
```

OrmlearnApllication.java

```
package com.example.ormlearn;
import com.example.ormlearn.model.Country;
import com.example.ormlearn.service.CountryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {
    @Autowired
    private CountryService countryService;
    public static void main(String[] args) {
        SpringApplication.run(OrmLearnApplication.class, args);
    @Override
    public void run(String... args) throws Exception {
        System.out.println("Find by code: " +
 countryService.findCountryByCode("IN").getName());
        Country newCountry = new Country("ZZ", "Zootopia");
        countryService.addCountry(newCountry);
        newCountry.setName("Zootopia Updated");
        countryService.updateCountry(newCountry);
        countryService.searchCountriesByPartialName("Uni")
            .forEach(c -> System.out.println(c.getCode() + ": " + c.getName()));
        countryService.deleteCountry("ZZ");
    }
}
```

Output:

```
Problems  Javadoc  Declaration  Console
<terminated> ormlearn - OrmlearnApplication [Spring Boot App] D:\Clg\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe  (05-Jul-2025, 7:54:51

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.3)

2025-07-05T19:54:52.488+05:30  INFO 17544 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Starting OrmlearnApplication using Java
2025-07-05T19:54:52.491+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Running with Spring Boot v3.5.3, Spring
2025-07-05T19:54:52.491+05:30  INFO 17544 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : No active profile set, falling back to
2025-07-05T19:54:52.524+05:30  INFO 17544 --- [ormlearn] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set
2025-07-05T19:54:52.896+05:30  INFO 17544 --- [ormlearn] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositor
2025-07-05T19:54:52.957+05:30  INFO 17544 --- [ormlearn] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scannir
2025-07-05T19:54:53.194+05:30  INFO 17544 --- [ormlearn] [  restartedMain] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitIr
2025-07-05T19:54:53.242+05:30  INFO 17544 --- [ormlearn] [  restartedMain] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 6
2025-07-05T19:54:53.277+05:30  INFO 17544 --- [ormlearn] [  restartedMain] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache disabled
2025-07-05T19:54:53.502+05:30  INFO 17544 --- [ormlearn] [  restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA c
2025-07-05T19:54:53.530+05:30  INFO 17544 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2025-07-05T19:54:53.839+05:30  INFO 17544 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mys
2025-07-05T19:54:53.840+05:30  INFO 17544 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2025-07-05T19:54:53.895+05:30  WARN 17544 --- [ormlearn] [  restartedMain] org.hibernate.orm.deprecation            : HHH90000025: MySQLDialect does not neec
2025-07-05T19:54:53.910+05:30  INFO 17544 --- [ormlearn] [  restartedMain] org.hibernate.orm.connections.pooling    : HHH10001005: Database info:
        Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
        Database driver: undefined/unknown
        Database version: 8.0.42
        Autocommit mode: undefined/unknown
        Isolation level: undefined/unknown
        Minimum pool size: undefined/unknown
        Maximum pool size: undefined/unknown
2025-07-05T19:54:53.918+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(12, org.hibernate.type.de
2025-07-05T19:54:53.918+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(-9, org.hibernate.type.de
2025-07-05T19:54:53.918+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(-3, org.hibernate.type.de
2025-07-05T19:54:53.918+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4003, org.hibernate.type.
2025-07-05T19:54:53.918+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4001, org.hibernate.type.
2025-07-05T19:54:53.918+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4002, org.hibernate.type.
2025-07-05T19:54:53.918+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(2004, org.hibernate.type.
```

```
Problems  Javadoc  Declaration  Console
<terminated> ormlearn - OrmlearnApplication [Spring Boot App] D:\Clg\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe  (05-Jul-2025, 7:54:
2025-07-05T19:54:53.840+05:30  INFO 17544 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2025-07-05T19:54:53.895+05:30  WARN 17544 --- [ormlearn] [  restartedMain] org.hibernate.orm.deprecation            : HHH90000025: MySQLDialect does not neec
2025-07-05T19:54:53.910+05:30  INFO 17544 --- [ormlearn] [  restartedMain] org.hibernate.orm.connections.pooling    : HHH10001005: Database info:
        Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
        Database driver: undefined/unknown
        Database version: 8.0.42
        Autocommit mode: undefined/unknown
        Isolation level: undefined/unknown
        Minimum pool size: undefined/unknown
        Maximum pool size: undefined/unknown
2025-07-05T19:54:53.918+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(12, org.hibernate.type.de
2025-07-05T19:54:53.918+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(-9, org.hibernate.type.de
2025-07-05T19:54:53.918+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(-3, org.hibernate.type.de
2025-07-05T19:54:53.918+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4003, org.hibernate.type.
2025-07-05T19:54:53.918+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4001, org.hibernate.type.
2025-07-05T19:54:53.918+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4002, org.hibernate.type.
2025-07-05T19:54:53.918+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(2004, org.hibernate.type.
2025-07-05T19:54:53.918+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(2005, org.hibernate.type.
2025-07-05T19:54:53.918+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(2011, org.hibernate.type.
2025-07-05T19:54:54.512+05:30  INFO 17544 --- [ormlearn] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (s
2025-07-05T19:54:54.548+05:30  INFO 17544 --- [ormlearn] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory fo
2025-07-05T19:54:54.864+05:30  INFO 17544 --- [ormlearn] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35
2025-07-05T19:54:54.881+05:30  INFO 17544 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Started OrmlearnApplication in 2.705 se
2025-07-05T19:54:54.913+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : select c1_0.co_code,c1_0.co_name from c
Find by code: India
2025-07-05T19:54:54.940+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : select c1_0.co_code,c1_0.co_name from c
2025-07-05T19:54:54.955+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : insert into country (co_name,co_code) v
2025-07-05T19:54:55.253+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : select count(*) from country c1_0 where
2025-07-05T19:54:55.259+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : select c1_0.co_code,c1_0.co_name from c
2025-07-05T19:54:55.264+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : update country set co_name=? where co_c
2025-07-05T19:54:55.327+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : select c1_0.co_code,c1_0.co_name from c
US: United States of America
2025-07-05T19:54:55.332+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : select count(*) from country c1_0 where
2025-07-05T19:54:55.334+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : select c1_0.co_code,c1_0.co_name from c
2025-07-05T19:54:55.337+05:30 DEBUG 17544 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : delete from country where co_code=?
2025-07-05T19:54:55.351+05:30  INFO 17544 --- [ormlearn] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for pe
2025-07-05T19:54:55.353+05:30  INFO 17544 --- [ormlearn] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
2025-07-05T19:54:55.358+05:30  INFO 17544 --- [ormlearn] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
```

Explanation:

1. Set spring.jpa.hibernate.ddl-auto=validate in application.properties.
2. Populate country table using the given INSERT SQL script.
3. Create Country entity with annotations: @Entity, @Table, @Id, @Column.

4. Create CountryRepository interface extending JpaRepository<Country, String>.
5. Define method in CountryRepository:
   List<Country> findByNameContainingIgnoreCase(String name);
6. Create CountryService class with @Service annotation.
7. Autowire CountryRepository in CountryService.
8. Implement methods in CountryService:
   - getCountry(String code)
   - addCountry(Country country)
   - updateCountry(Country country)
   - deleteCountry(String code)
   - findByNameContaining(String partialName)
9. Autowire CountryService in OrmLearnApplication.
10. Create test methods in OrmLearnApplication to call service methods.
11. Run the application and verify logs for output.

2. Hands on 6:Find a country based on country code

Solution:

Code:

CountryService.java

```
package com.example.ormlearn.service;
import com.example.ormlearn.model.Country;
import com.example.ormlearn.repository.CountryRepository;
import com.example.ormlearn.service.exception.CountryNotFoundException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import javax.transaction.Transactional;
import java.util.Optional;
@Service
public class CountryService {
   @Autowired
   private CountryRepository countryRepository;
   @Transactional
   public Country findCountryByCode(String countryCode) throws
CountryNotFoundException {
      Optional<Country> result = countryRepository.findById(countryCode);
      if (!result.isPresent()) {
         throw new CountryNotFoundException("Country with code " + countryCode + "
not found");
      }
      return result.get();
   }
}
```

CountryNotFoundException.java

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

```
}
```

OrmlearnApllication.java

```
package com.example.ormlearn;
import com.example.ormlearn.model.Country;
import com.example.ormlearn.service.CountryService;
import com.example.ormlearn.service.exception.CountryNotFoundException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
@SpringBootApplication
public class OrmLearnApplication {
    private static final Logger LOGGER =
 LoggerFactory.getLogger(OrmLearnApplication.class);
    private static CountryService countryService;
    public static void main(String[] args) throws CountryNotFoundException {
        ApplicationContext context = SpringApplication.run(OrmLearnApplication.class,
args);
        countryService = context.getBean(CountryService.class);
        getCountryTest();
    }
    private static void getCountryTest() throws CountryNotFoundException {
        LOGGER.info("Start");
        Country country = countryService.findCountryByCode("IN");
        LOGGER.debug("Country: {}", country);
        LOGGER.info("End");
    }
}
```

Output:

```
Problems  Javadoc  Declaration  Console
<terminated> ormlearn - OrmlearnApplication [Spring Boot App] D:\Clg\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe  (05-Jul-2025, 8:55

 .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.3)

2025-07-05T20:55:26.833+05:30  INFO 12232 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Starting OrmlearnApplication using Java
2025-07-05T20:55:26.837+05:30 DEBUG 12232 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Running with Spring Boot v3.5.3, Spring
2025-07-05T20:55:26.837+05:30  INFO 12232 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : No active profile set, falling back to
2025-07-05T20:55:26.891+05:30  INFO 12232 --- [ormlearn] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set
2025-07-05T20:55:27.286+05:30  INFO 12232 --- [ormlearn] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositor
2025-07-05T20:55:27.339+05:30  INFO 12232 --- [ormlearn] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scannir
2025-07-05T20:55:27.636+05:30  INFO 12232 --- [ormlearn] [  restartedMain] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitIr
2025-07-05T20:55:27.679+05:30  INFO 12232 --- [ormlearn] [  restartedMain] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 6
2025-07-05T20:55:27.710+05:30  INFO 12232 --- [ormlearn] [  restartedMain] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache disabled
2025-07-05T20:55:27.972+05:30  INFO 12232 --- [ormlearn] [  restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA c
2025-07-05T20:55:28.004+05:30  INFO 12232 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2025-07-05T20:55:28.518+05:30  INFO 12232 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mys
2025-07-05T20:55:28.520+05:30  INFO 12232 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2025-07-05T20:55:28.597+05:30  WARN 12232 --- [ormlearn] [  restartedMain] org.hibernate.orm.deprecation            : HHH90000025: MySQLDialect does not neec
2025-07-05T20:55:28.625+05:30  INFO 12232 --- [ormlearn] [  restartedMain] org.hibernate.orm.connections.pooling    : HHH10001005: Database info:
        Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
        Database driver: undefined/unknown
        Database version: 8.0.42
        Autocommit mode: undefined/unknown
        Isolation level: undefined/unknown
        Minimum pool size: undefined/unknown
        Maximum pool size: undefined/unknown
2025-07-05T20:55:28.635+05:30 DEBUG 12232 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(12, org.hibernate.type.de
2025-07-05T20:55:28.635+05:30 DEBUG 12232 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(-9, org.hibernate.type.de
2025-07-05T20:55:28.635+05:30 DEBUG 12232 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(-3, org.hibernate.type.de
2025-07-05T20:55:28.635+05:30 DEBUG 12232 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4003, org.hibernate.type.
2025-07-05T20:55:28.635+05:30 DEBUG 12232 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4001, org.hibernate.type.
2025-07-05T20:55:28.636+05:30 DEBUG 12232 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4002, org.hibernate.type.
2025-07-05T20:55:28.636+05:30 DEBUG 12232 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(2004, org.hibernate.type.
```

```
12232 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(2011, org.hibernate.type.descriptor.sql.internal.CapacityDepe
12232 --- [ormlearn] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platfo
12232 --- [ormlearn] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
12232 --- [ormlearn] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
12232 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Started OrmlearnApplication in 3.565 seconds (process running for 4.294)
12232 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Start
12232 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : select c1_0.co_code,c1_0.co_name from country c1_0 where c1_0.co_code=?
12232 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Country: com.example.ormlearn.model.Country@d2b3a38
12232 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : End
12232 --- [ormlearn] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
12232 --- [ormlearn] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
12232 --- [ormlearn] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
```

Explanation:

1. Create CountryNotFoundException class in com.cognizant.spring-learn.service.exception.
2. In CountryService, create method findCountryByCode() with @Transactional annotation.
3. Inside findCountryByCode(String countryCode):
   Use countryRepository.findById(countryCode) to get Optional<Country>.
   If not present, throw CountryNotFoundException.
   Else, return result.get().
4. In OrmLearnApplication, define method getAllCountriesTest():
   Call findCountryByCode("IN").

Log the returned Country object.

5. In main() method:
   Call getAllCountriesTest() after setting up application context and service bean.
6. Run and verify output in console logs.

3. Hands on 7: Add a new country

Solution:

Code:

CountryService.java

```
package com.example.ormlearn.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.example.ormlearn.model.Country;
import com.example.ormlearn.repository.CountryRepository;
import com.example.ormlearn.service.exception.CountryNotFoundException;
@Service
public class CountryService {
    @Autowired
    private CountryRepository countryRepository;
    @Transactional
    public Country findCountryByCode(String code) throws CountryNotFoundException {
        return countryRepository.findById(code)
                .orElseThrow(() -> new CountryNotFoundException("Country not found for
code: " + code));
    }
    @Transactional
    public void addCountry(Country country) {
        countryRepository.save(country);
    }
}
```

OrmlearnApllication.java

```
package com.example.ormlearn;
import com.example.ormlearn.model.Country;
import com.example.ormlearn.service.CountryService;
import com.example.ormlearn.service.exception.CountryNotFoundException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
```

```
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
@SpringBootApplication
public class OrmLearnApplication {
    private static final Logger LOGGER =
 LoggerFactory.getLogger(OrmLearnApplication.class);
    private static CountryService countryService;
    public static void main(String[] args) throws CountryNotFoundException {
        ApplicationContext context = SpringApplication.run(OrmLearnApplication.class,
args);
        countryService = context.getBean(CountryService.class);
            testAddCountry();
    }
    private static void getCountryTest() throws CountryNotFoundException {
        LOGGER.info("Start");
        Country country = countryService.findCountryByCode("IN");
        LOGGER.debug("Country: {}", country);
        LOGGER.info("End");
    }
    private static void testAddCountry() throws CountryNotFoundException {
        LOGGER.info("Start");
        Country newCountry = new Country();
        newCountry.setCode("ZZ");
        newCountry.setName("Zootopia");
        countryService.addCountry(newCountry);
        Country addedCountry = countryService.findCountryByCode("ZZ");
        LOGGER.debug("Added Country: {}", addedCountry);
        LOGGER.info("End");
    }
}
```

Output:

```
Problems   Javadoc   Declaration   Console
<terminated> ormlearn - OrmlearnApplication [Spring Boot App] D:\Clg\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe  (05-Jul-2025, 9:27:1

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.3)

2025-07-05T21:27:11.552+05:30  INFO 7888 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Starting OrmlearnApplication using Java
2025-07-05T21:27:11.553+05:30 DEBUG 7888 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Running with Spring Boot v3.5.3, Spring
2025-07-05T21:27:11.554+05:30  INFO 7888 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : No active profile set, falling back to 1
2025-07-05T21:27:11.609+05:30  INFO 7888 --- [ormlearn] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set '
2025-07-05T21:27:12.044+05:30  INFO 7888 --- [ormlearn] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositori
2025-07-05T21:27:12.099+05:30  INFO 7888 --- [ormlearn] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning
2025-07-05T21:27:12.404+05:30  INFO 7888 --- [ormlearn] [  restartedMain] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInf
2025-07-05T21:27:12.447+05:30  INFO 7888 --- [ormlearn] [  restartedMain] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 6.
2025-07-05T21:27:12.482+05:30  INFO 7888 --- [ormlearn] [  restartedMain] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache disabled
2025-07-05T21:27:12.763+05:30  INFO 7888 --- [ormlearn] [  restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA cl
2025-07-05T21:27:12.797+05:30  INFO 7888 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2025-07-05T21:27:13.220+05:30  INFO 7888 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysc
2025-07-05T21:27:13.222+05:30  INFO 7888 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2025-07-05T21:27:13.285+05:30  WARN 7888 --- [ormlearn] [  restartedMain] org.hibernate.orm.deprecation            : HHH90000025: MySQLDialect does not need
2025-07-05T21:27:13.303+05:30  INFO 7888 --- [ormlearn] [  restartedMain] org.hibernate.orm.connections.pooling    : HHH10001005: Database info:
        Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
        Database driver: undefined/unknown
        Database version: 8.0.42
        Autocommit mode: undefined/unknown
        Isolation level: undefined/unknown
        Minimum pool size: undefined/unknown
        Maximum pool size: undefined/unknown
2025-07-05T21:27:13.309+05:30 DEBUG 7888 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(12, org.hibernate.type.des
2025-07-05T21:27:13.310+05:30 DEBUG 7888 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(-9, org.hibernate.type.des
2025-07-05T21:27:13.310+05:30 DEBUG 7888 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(-3, org.hibernate.type.des
2025-07-05T21:27:13.310+05:30 DEBUG 7888 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4003, org.hibernate.type.c
2025-07-05T21:27:13.310+05:30 DEBUG 7888 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4001, org.hibernate.type.c
2025-07-05T21:27:13.311+05:30 DEBUG 7888 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4002, org.hibernate.type.c
2025-07-05T21:27:13.311+05:30 DEBUG 7888 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(2004, org.hibernate.type.c
```

```
rmlearn] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform'
rmlearn] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
rmlearn] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
rmlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Started OrmlearnApplication in 3.315 seconds (process running for 4.021)
rmlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Start
rmlearn] [  restartedMain] org.hibernate.SQL                        : select c1_0.co_code,c1_0.co_name from country c1_0 where c1_0.co_code=?
rmlearn] [  restartedMain] org.hibernate.SQL                        : select c1_0.co_code,c1_0.co_name from country c1_0 where c1_0.co_code=?
rmlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Added Country: Country [code=ZZ, name=Zootopia]
rmlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : End
rmlearn] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
rmlearn] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
rmlearn] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
```

Explanation:

1. In CountryService, create method addCountry() with @Transactional annotation.
2. Inside addCountry(Country country), call countryRepository.save(country).
3. In OrmLearnApplication, define testAddCountry() method:
   Create a new Country object with code and name.
   Call countryService.addCountry(country).
   Call countryService.findCountryByCode(code) to verify.
4. In main() method, call testAddCountry() after setting up the context.
5. Run and check if the new country is present in the database.

II. spring-data-jpa-handson

1. Demonstrate implementation of Query Methods feature of Spring Data JPA

- **Query Methods - Search by containing text, sorting, filter with starting text, fetch between dates, greater than or lesser than, top**

Solution:

Code:

Country.java

```
package com.example.ormlearn.model;
import jakarta.persistence.*;
import java.time.LocalDate;
@Entity
@Table(name = "country")
public class Country {
    @Id
    @Column(name = "co_code")
    private String code;
    @Column(name = "co_name")
    private String name;
    @Column(name = "created_date")
    private LocalDate createdDate;
    public Country() {}
    public Country(String code, String name, LocalDate createdDate) {
        this.code = code;
        this.name = name;
        this.createdDate = createdDate;
    }
    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; }
    public LocalDate getCreatedDate() { return createdDate; }
    public void setCreatedDate(LocalDate createdDate) { this.createdDate = createdDate; }

    @Override
    public String toString() {
        return "Country{" + "code='" + code + '\'' + ", name='" + name + '\'' + ", createdDate="
+ createdDate + '}';
```

```
    }
}
```

CountryRepository.java

```
package com.example.ormlearn.repository;
import com.example.ormlearn.model.Country;
import org.springframework.data.jpa.repository.JpaRepository;
import java.time.LocalDate;
import java.util.List;
public interface CountryRepository extends JpaRepository<Country, String> {
    // Find countries by name containing text
    List<Country> findByNameContaining(String keyword);

    // Find countries by name starting with
    List<Country> findByNameStartingWith(String prefix);

    // Find countries created after a date
    List<Country> findByCreatedDateAfter(LocalDate date);

    // Find countries created between two dates
    List<Country> findByCreatedDateBetween(LocalDate start, LocalDate end);

    // Find top 3 countries by name ascending
    List<Country> findTop3ByOrderByNameAsc();

    // Find all countries sorted by name descending
    List<Country> findAllByOrderByNameDesc();
}
```

CountryService.java

```
package com.example.ormlearn.service;
```

```
import com.example.ormlearn.model.Country;
import com.example.ormlearn.repository.CountryRepository;
import com.example.ormlearn.service.exception.CountryNotFoundException;
import jakarta.transaction.Transactional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.time.LocalDate;
import java.util.List;
import java.util.Optional;
@Service
public class CountryService {
    @Autowired
    private CountryRepository countryRepository;
    @Transactional
    public Country findCountryByCode(String countryCode) throws
CountryNotFoundException {
        Optional<Country> result = countryRepository.findById(countryCode);
        if (!result.isPresent()) {
            throw new CountryNotFoundException("Country Not Found");
        }
        return result.get();
    }
    @Transactional
    public void addCountry(Country country) {
        countryRepository.save(country);
    }
    public List<Country> findByNameContaining(String keyword) {
        return countryRepository.findByNameContaining(keyword);
    }
    public List<Country> findByNameStartingWith(String prefix) {
        return countryRepository.findByNameStartingWith(prefix);
```

```
    }
    public List<Country> findByCreatedDateAfter(LocalDate date) {
        return countryRepository.findByCreatedDateAfter(date);
    }
    public List<Country> findByCreatedDateBetween(LocalDate start, LocalDate end) {
        return countryRepository.findByCreatedDateBetween(start, end);
    }
    public List<Country> findTop3ByOrderByNameAsc() {
        return countryRepository.findTop3ByOrderByNameAsc();
    }
    public List<Country> findAllByOrderByNameDesc() {
        return countryRepository.findAllByOrderByNameDesc();
    }
}
```

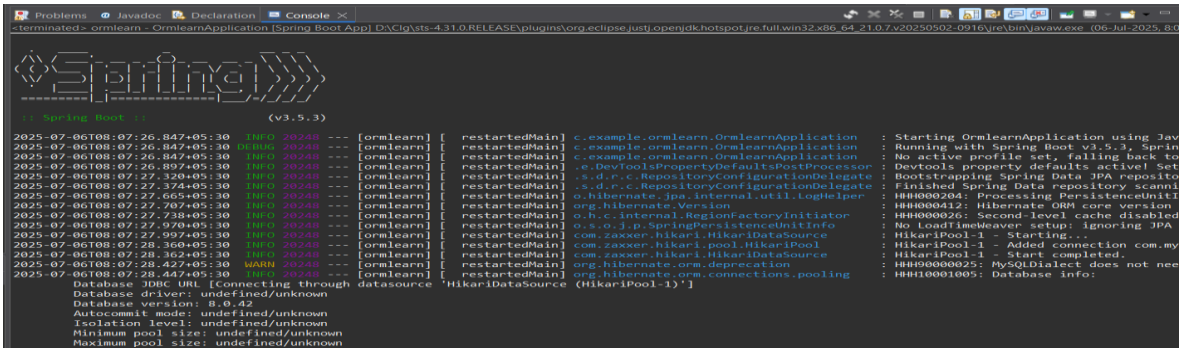
OrmlearnApllication.java

```
package com.example.ormlearn;
import com.example.ormlearn.model.Country;
import com.example.ormlearn.service.CountryService;
import com.example.ormlearn.service.exception.CountryNotFoundException;
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.time.LocalDate;
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) throws CountryNotFoundException {
        ApplicationContext context = SpringApplication.run(OrmlearnApplication.class,
args);
        countryService = context.getBean(CountryService.class);
        // getCountryTest();
        // testAddCountry();
        testQueryMethods();
    }
    private static void getCountryTest() throws CountryNotFoundException {
        LOGGER.info("Start getCountryTest");
        Country country = countryService.findCountryByCode("IN");
        LOGGER.debug("Country: {}", country);
        LOGGER.info("End getCountryTest");
    }
    private static void testAddCountry() throws CountryNotFoundException {
        LOGGER.info("Start testAddCountry");
        Country newCountry = new Country();
        newCountry.setCode("ZZ");
        newCountry.setName("Zootopia");
        newCountry.setCreatedDate(LocalDate.now());
        countryService.addCountry(newCountry);
        Country addedCountry = countryService.findCountryByCode("ZZ");
        LOGGER.debug("Added Country: {}", addedCountry);
        LOGGER.info("End testAddCountry");
    }
    private static void testQueryMethods() {
        LOGGER.info("Start Query Methods");
        List<Country> countriesWithAn = countryService.findByNameContaining("an");
        LOGGER.debug("Countries containing 'an': {}", countriesWithAn);
```

```
        List<Country> countriesStartingWithU = countryService.findByNameStartingWith("U");
        LOGGER.debug("Countries starting with 'U': {}", countriesStartingWithU);
        List<Country> createdAfter = countryService.findByCreatedDateAfter(LocalDate.of(2024, 1, 1));
        LOGGER.debug("Countries created after 2024-01-01: {}", createdAfter);
        List<Country> createdBetween = countryService.findByCreatedDateBetween(
            LocalDate.of(2023, 1, 1), LocalDate.of(2024, 12, 31));
        LOGGER.debug("Countries created between 2023 and 2024: {}", createdBetween);
        List<Country> top3Countries = countryService.findTop3ByOrderByNameAsc();
        LOGGER.debug("Top 3 countries by name ASC: {}", top3Countries);
        List<Country> countriesDesc = countryService.findAllByOrderByNameDesc();
        LOGGER.debug("All countries by name DESC: {}", countriesDesc);
        LOGGER.info("End Query Methods");
    }
}
```

Output:



```
JtaPlatformInitiator        : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)
erEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
onalLiveReloadServer        : LiveReload server is running on port 35729
earn.OrmlearnApplication    : Started OrmlearnApplication in 3.148 seconds (process running for 3.931)
earn.OrmlearnApplication    : Start Query Methods
SQL                         : select c1_0.co_code,c1_0.created_date,c1_0.co_name from country c1_0 where c1_0.co_name like ? escape '\\'
earn.OrmlearnApplication    : Countries containing 'an': [Country{code='CA', name='Canada', createdDate=1867-07-01}, Country{code='FR', name='France', create
SQL                         : select c1_0.co_code,c1_0.created_date,c1_0.co_name from country c1_0 where c1_0.co_name like ? escape '\\'
earn.OrmlearnApplication    : Countries starting with 'U': [Country{code='US', name='United States of America', createdDate=1776-07-04}]
SQL                         : select c1_0.co_code,c1_0.created_date,c1_0.co_name from country c1_0 where c1_0.created_date>?
earn.OrmlearnApplication    : Countries created after 2024-01-01: [Country{code='ZZ', name='Zootopia', createdDate=2025-01-01}]
SQL                         : select c1_0.co_code,c1_0.created_date,c1_0.co_name from country c1_0 where c1_0.created_date between ? and ?
earn.OrmlearnApplication    : Countries created between 2023 and 2024: [Country{code='ZA', name='South Africa', createdDate=2023-05-31}]
SQL                         : select c1_0.co_code,c1_0.created_date,c1_0.co_name from country c1_0 order by c1_0.co_name limit ?
earn.OrmlearnApplication    : Top 3 countries by name ASC: [Country{code='AU', name='Australia', createdDate=1901-01-01}, Country{code='BR', name='Brazil', cr
SQL                         : select c1_0.co_code,c1_0.created_date,c1_0.co_name from country c1_0 order by c1_0.co_name desc
earn.OrmlearnApplication    : All countries by name DESC: [Country{code='ZZ', name='Zootopia', createdDate=2025-01-01}, Country{code='US', name='United States
earn.OrmlearnApplication    : End Query Methods
erEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
ari.HikariDataSource        : HikariPool-1 - Shutdown initiated...
ari.HikariDataSource        : HikariPool-1 - Shutdown completed.
```

Explanation:

1. Define Query Methods in Repository Interface:
   List<Country> findByNameContaining(String keyword);
   List<Country> findByNameStartingWith(String prefix);
   List<Country> findTop3ByOrderByNameAsc();
   (For date/number fields, if present:)
   List<Entity> findByDateBetween(Date start, Date end);
   List<Entity> findByValueGreaterThan(int value);
2. Create corresponding test methods in OrmLearnApplication:
   Call the query methods from CountryRepository.
   Log and verify the output.
3. Run the main application and observe results in the log.

2. Demonstrate implementation of O/R Mapping

- **@ManyToOne, @JoinColumn, @OneToMany, FetchType.EAGER, FetchType.LAZY, @ManyToMany, @JoinTable, mappedBy**

Solution:

Code:

Country.java

```
package com.example.ormlearn.model;
import jakarta.persistence.*;
@Entity
```

```
@Table(name = "country")
public class Country {
    @Id
    @Column(name = "co_code")
    private String code;
    @Column(name = "co_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;
    }
}
```

Employee.java

```
package com.example.ormlearn.model;
import jakarta.persistence.*;
import java.util.List;
@Entity
@Table(name = "employee")
public class Employee {
    @Id
    private int id;
    @Column(name = "emp_name")
```

```
private String name;
@ManyToOne
@JoinColumn(name = "co_code")
private Country country;
@ManyToMany(fetch = FetchType.LAZY)
@JoinTable(
    name = "employee_skill",
    joinColumns = @JoinColumn(name = "employee_id"),
    inverseJoinColumns = @JoinColumn(name = "skill_id")
)
private List<Skill> skills;
public int getId() {
    return id;
}
public void setId(int id) {
    this.id = id;
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public Country getCountry() {
    return country;
}
public void setCountry(Country country) {
    this.country = country;
}
public List<Skill> getSkills() {
    return skills;
```

```
    }
    public void setSkills(List<Skill> skills) {
        this.skills = skills;
    }
}
```

CountryRepository.java

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

EmployeeRepository.java

```
package com.example.ormlearn.repository;
import com.example.ormlearn.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {
}
```

EmployeeService.java

```
package com.example.ormlearn.service;
import com.example.ormlearn.model.Employee;
import com.example.ormlearn.model.Skill;
import com.example.ormlearn.repository.EmployeeRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
@Service
public class EmployeeService {
```

```
    @Autowired
    private EmployeeRepository employeeRepository;
    @Transactional
    public void fetchEmployeeWithSkills(int id) {
        Employee employee = employeeRepository.findById(id).orElse(null);
        if (employee != null) {
            System.out.println("Employee: " + employee.getName());
            System.out.println("Country: " + employee.getCountry().getName());
             System.out.println("Skills:");
            for (Skill skill : employee.getSkills()) {
                System.out.println(" - " + skill.getName());
            }
        } else {
            System.out.println("Employee not found.");
        }
    }
}
```

OrmlearnApllication.java

```
package com.example.ormlearn;
import com.example.ormlearn.service.EmployeeService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class OrmlearnApplication implements CommandLineRunner {
    @Autowired
    private EmployeeService employeeService;
    public static void main(String[] args) {
```

```
        SpringApplication.run(OrmlearnApplication.class, args);
    }
    @Override
    public void run(String... args) throws Exception {
        employeeService.fetchEmployeeWithSkills(1); // Replace 1 with a valid employee ID
in your DB
    }
}
```

Output:

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.3)

2025-07-06T16:44:41.887+05:30  INFO 12896 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Starting OrmlearnApplication using Java
2025-07-06T16:44:41.888+05:30 DEBUG 12896 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Running with Spring Boot v3.5.3, Spring
2025-07-06T16:44:41.890+05:30  INFO 12896 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : No active profile set, falling back to
2025-07-06T16:44:41.934+05:30  INFO 12896 --- [ormlearn] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set
2025-07-06T16:44:42.338+05:30  INFO 12896 --- [ormlearn] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositor
2025-07-06T16:44:42.413+05:30  INFO 12896 --- [ormlearn] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scannir
2025-07-06T16:44:42.708+05:30  INFO 12896 --- [ormlearn] [  restartedMain] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitIr
2025-07-06T16:44:42.769+05:30  INFO 12896 --- [ormlearn] [  restartedMain] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 6
2025-07-06T16:44:42.804+05:30  INFO 12896 --- [ormlearn] [  restartedMain] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache disabled
2025-07-06T16:44:43.071+05:30  INFO 12896 --- [ormlearn] [  restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA c
2025-07-06T16:44:43.104+05:30  INFO 12896 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2025-07-06T16:44:43.424+05:30  INFO 12896 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mys
2025-07-06T16:44:43.425+05:30  INFO 12896 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2025-07-06T16:44:43.486+05:30  WARN 12896 --- [ormlearn] [  restartedMain] org.hibernate.orm.deprecation            : HHH90000025: MySQLDialect does not need
2025-07-06T16:44:43.505+05:30  INFO 12896 --- [ormlearn] [  restartedMain] org.hibernate.orm.connections.pooling    : HHH10001005: Database info:
        Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
        Database driver: undefined/unknown
        Database version: 8.0.42
        Autocommit mode: undefined/unknown
        Isolation level: undefined/unknown
        Minimum pool size: undefined/unknown
        Maximum pool size: undefined/unknown
2025-07-06T16:44:43.512+05:30 DEBUG 12896 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(12, org.hibernate.type.de
```

```
  : addDescriptor(12, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@75458280) replaced previous registration(org.hibernate.type.desc
  : addDescriptor(-9, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@59ab601d) replaced previous registration(org.hibernate.type.desc
  : addDescriptor(-3, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@4e29b5ce) replaced previous registration(org.hibernate.type.desc
  : addDescriptor(4003, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@1ab1c619) replaced previous registration(org.hibernate.type.descriptor.sql
  : addDescriptor(4001, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@529a39d) replaced previous registration(org.hibernate.type.descriptor.sql.i
  : addDescriptor(4002, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@4e4f0575) replaced previous registration(org.hibernate.type.descriptor.sql
  : addDescriptor(2004, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@6f7ca830) replaced previous registration(org.hibernate.type.de
  : addDescriptor(2005, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@57f36a43) replaced previous registration(org.hibernate.type.de
  : addDescriptor(2011, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@74fc0908) replaced previous registration(org.hibernate.type.de
  : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)
  : Initialized JPA EntityManagerFactory for persistence unit 'default'
  : LiveReload server is running on port 35729
  : Started OrmlearnApplication in 3.272 seconds (process running for 4.196)
  : select e1_0.id,c1_0.co_code,c1_0.co_name,e1_0.emp_name from employee e1_0 left join country c1_0 on c1_0.co_code=e1_0.co_code where e1_0.id=?



  : select s1_0.employee_id,s1_1.id,s1_1.name from employee_skill s1_0 join skill s1_1 on s1_1.id=s1_0.skill_id where s1_0.employee_id=?


  : Closing JPA EntityManagerFactory for persistence unit 'default'
  : HikariPool-1 - Shutdown initiated...
  : HikariPool-1 - Shutdown completed.
```

```
2025-07-06T16:44:43.512+05:30 DEBUG 12896 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(12, org.hibernate.type.de
2025-07-06T16:44:43.513+05:30 DEBUG 12896 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(-9, org.hibernate.type.de
2025-07-06T16:44:43.513+05:30 DEBUG 12896 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(-3, org.hibernate.type.de
2025-07-06T16:44:43.513+05:30 DEBUG 12896 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4003, org.hibernate.type.
2025-07-06T16:44:43.513+05:30 DEBUG 12896 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4001, org.hibernate.type.
2025-07-06T16:44:43.513+05:30 DEBUG 12896 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(4002, org.hibernate.type.
2025-07-06T16:44:43.513+05:30 DEBUG 12896 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(2004, org.hibernate.type.
2025-07-06T16:44:43.514+05:30 DEBUG 12896 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(2005, org.hibernate.type.
2025-07-06T16:44:43.514+05:30 DEBUG 12896 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(2011, org.hibernate.type.
2025-07-06T16:44:44.386+05:30  INFO 12896 --- [ormlearn] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (s
2025-07-06T16:44:44.435+05:30  INFO 12896 --- [ormlearn] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory fo
2025-07-06T16:44:44.770+05:30  INFO 12896 --- [ormlearn] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35
2025-07-06T16:44:44.791+05:30  INFO 12896 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Started OrmlearnApplication in 3.272 se
2025-07-06T16:44:44.852+05:30 DEBUG 12896 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : select e1_0.id,c1_0.co_code,c1_0.co_nam
Employee: John Doe
Country: India
Skills:
2025-07-06T16:44:44.922+05:30 DEBUG 12896 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : select s1_0.employee_id,s1_1.id,s1_1.na
 - Java
 - Python
2025-07-06T16:44:44.967+05:30  INFO 12896 --- [ormlearn] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for pe
2025-07-06T16:44:44.969+05:30  INFO 12896 --- [ormlearn] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
2025-07-06T16:44:44.980+05:30  INFO 12896 --- [ormlearn] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
```

Explanation:

1. Create Entity Classes
   Annotate classes with @Entity and @Table.
2. Implement Relationships
   Use @ManyToOne and @JoinColumn for many-to-one mapping.
   Use @OneToMany(mappedBy = "...", fetch = FetchType.LAZY/EAGER) for one-to-many.
   Use @ManyToMany and @JoinTable for many-to-many.
3. Create Repositories
   Extend JpaRepository for each entity.
4. Autowire in Service Layer
   Autowire repositories into a service class.
5. Create Test Methods
   Fetch entity with related data and log the output.
6. Run Application

III. <u>spring-data-jpa-handson</u>

3. Demonstrate writing Hibernate Query Language and Native Query

- **HQL stands for Hibernate Query Language, JPQL stands for Java Persistence Query Language, Compare HQL and JPQL, @Query annotation, HQL fetch keyword, aggregate functions in HQL, Native Query, nativeQuery attribute**

Solution:

HQL (Hibernate Query Language) is an object-oriented query language similar to SQL, but it operates on Java objects instead of database tables.

JPQL (Java Persistence Query Language) is part of JPA and is also object-oriented. It works similarly to HQL and is considered vendor-agnostic (works across different JPA providers).

Code:

Employee.java

```
package com.example.ormlearn.model;
import jakarta.persistence.*;
import java.util.HashSet;
import java.util.Set;
@Entity
@Table(name = "employee")
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name = "name", nullable = false)
    private String name;
    @Column(name = "department")
    private String department;
    @Column(name = "salary")
    private Double salary;
    @ManyToMany(fetch = FetchType.LAZY)
```

```
@JoinTable(
    name = "employee_skill",
    joinColumns = @JoinColumn(name = "employee_id"),
    inverseJoinColumns = @JoinColumn(name = "skill_id")
)
private Set<Skill> skills = new HashSet<>();
public Long getId() {
    return id;
}
public void setId(Long id) {
    this.id = id;
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public String getDepartment() {
    return department;
}
public void setDepartment(String department) {
    this.department = department;
}
public Double getSalary() {
    return salary;
}
public void setSalary(Double salary) {
    this.salary = salary;
}
public Set<Skill> getSkills() {
```

```
        return skills;
    }
    public void setSkills(Set<Skill> skills) {
        this.skills = skills;
    }
}
```

Skill.java

```
package com.example.ormlearn.model;
import jakarta.persistence.*;
import java.util.Set;
@Entity
public class Skill {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    @ManyToMany(mappedBy = "skills")
    private Set<Employee> employees;
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public Set<Employee> getEmployees() { return employees; }
    public void setEmployees(Set<Employee> employees) { this.employees = employees; }
}
```

EmployeeRepository.java

```
package com.example.ormlearn.repository;
import com.example.ormlearn.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
```

```
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import java.util.List;
public interface EmployeeRepository extends JpaRepository<Employee, Long> {
    // JPQL Query
    @Query("SELECT e FROM Employee e WHERE e.department = :dept")
    List<Employee> findByDepartment(@Param("dept") String department);
    // Fetch Join Query
    @Query("SELECT e FROM Employee e JOIN FETCH e.skills WHERE e.id = :id")
    Employee findEmployeeWithSkills(@Param("id") Long id);
    // Aggregate function
    @Query("SELECT AVG(e.salary) FROM Employee e")
    Double findAverageSalary();
    // Native SQL Query
    @Query(value = "SELECT * FROM employee WHERE salary > :minSalary",
nativeQuery = true)
    List<Employee> findEmployeesWithHighSalary(@Param("minSalary") double
salary);
}
```

TestEmployeeService.java

```
package com.example.ormlearn.service;
import com.example.ormlearn.model.Employee;
import com.example.ormlearn.repository.EmployeeRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import jakarta.annotation.PostConstruct;
import java.util.List;
@Service
public class TestEmployeeService {
    @Autowired
```

```
    private EmployeeRepository employeeRepository;
    @PostConstruct
    public void testQueries() {
        System.out.println("Testing JPQL Query by Department:");
        List<Employee> itEmployees = employeeRepository.findByDepartment("IT");
        itEmployees.forEach(e -> System.out.println(e.getName()));
        System.out.println("\nTesting Fetch Join (Employee with Skills):");
        Employee empWithSkills = employeeRepository.findEmployeeWithSkills(1L);
        System.out.println("Employee: " + empWithSkills.getName());
        empWithSkills.getSkills().forEach(skill -> System.out.println(skill.getName()));
        System.out.println("\nTesting Aggregate (Average Salary):");
        Double avgSalary = employeeRepository.findAverageSalary();
        System.out.println("Average Salary: " + avgSalary);
        System.out.println("\nTesting Native SQL Query:");
        List<Employee> highEarners =
employeeRepository.findEmployeesWithHighSalary(50000.0);
        highEarners.forEach(e -> System.out.println(e.getName() + " - " + e.getSalary()));
    }
}
```

OrmlearnApllication.java

```
package com.example.ormlearn;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class OrmlearnApplication {
    public static void main(String[] args) {
        SpringApplication.run(OrmlearnApplication.class, args);
    }
}
```

Output:

```
Problems  Javadoc  Declaration  Console
<terminated> ormlearn - OrmlearnApplication [Spring Boot App] D:\Clg\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.3)

2025-07-06T17:30:38.800+05:30  INFO 24740 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Starting OrmlearnApplic
2025-07-06T17:30:38.802+05:30 DEBUG 24740 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Running with Spring Boo
2025-07-06T17:30:38.803+05:30  INFO 24740 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : No active profile set,
2025-07-06T17:30:38.854+05:30  INFO 24740 --- [ormlearn] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defau
2025-07-06T17:30:39.309+05:30  INFO 24740 --- [ormlearn] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Da
2025-07-06T17:30:39.376+05:30  INFO 24740 --- [ormlearn] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data re
2025-07-06T17:30:39.710+05:30  INFO 24740 --- [ormlearn] [  restartedMain] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing P
2025-07-06T17:30:39.770+05:30  INFO 24740 --- [ormlearn] [  restartedMain] org.hibernate.Version                    : HHH000412: Hibernate Or
2025-07-06T17:30:39.815+05:30  INFO 24740 --- [ormlearn] [  restartedMain] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level
2025-07-06T17:30:40.061+05:30  INFO 24740 --- [ormlearn] [  restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup
2025-07-06T17:30:40.111+05:30  INFO 24740 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting
2025-07-06T17:30:40.507+05:30  INFO 24740 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added co
2025-07-06T17:30:40.510+05:30  INFO 24740 --- [ormlearn] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start co
2025-07-06T17:30:40.592+05:30  WARN 24740 --- [ormlearn] [  restartedMain] org.hibernate.orm.deprecation            : HHH90000025: MySQLDialec
2025-07-06T17:30:40.613+05:30  INFO 24740 --- [ormlearn] [  restartedMain] org.hibernate.orm.connections.pooling    : HHH10001005: Database i
        Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
        Database driver: undefined/unknown
        Database version: 8.0.42
        Autocommit mode: undefined/unknown
        Isolation level: undefined/unknown
        Minimum pool size: undefined/unknown
        Maximum pool size: undefined/unknown
2025-07-06T17:30:40.621+05:30 DEBUG 24740 --- [ormlearn] [  restartedMain] o.h.t.d.sql.spi.DdlTypeRegistry          : addDescriptor(12, org.h
```

```
2025-07-06T17:30:41.686+05:30  INFO 24740 --- [ormlearn] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityM
2025-07-06T17:30:42.011+05:30  INFO 24740 --- [ormlearn] [  restartedMain] o.s.d.j.r.query.QueryEnhancerFactory     : Hibernate is in classpa
Testing JPQL Query by Department:
2025-07-06T17:30:42.773+05:30 DEBUG 24740 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : select e1_0.id,e1_0.dep

Testing Fetch Join (Employee with Skills):
2025-07-06T17:30:42.810+05:30 DEBUG 24740 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : select e1_0.id,e1_0.dep
Employee: John Doe
Java
Python

Testing Aggregate (Average Salary):
2025-07-06T17:30:42.837+05:30 DEBUG 24740 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : select avg(e1_0.salary)
Average Salary: 63333.333333333336

Testing Native SQL Query:
2025-07-06T17:30:42.928+05:30 DEBUG 24740 --- [ormlearn] [  restartedMain] org.hibernate.SQL                        : SELECT * FROM employee
John Doe - 60000.0
Jane Smith - 75000.0
Pierre Dupont - 55000.0
John Doe - 60000.0
Jane Smith - 75000.0
Pierre Dupont - 55000.0
2025-07-06T17:30:43.087+05:30  INFO 24740 --- [ormlearn] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is ru
2025-07-06T17:30:43.115+05:30  INFO 24740 --- [ormlearn] [  restartedMain] c.example.ormlearn.OrmlearnApplication   : Started OrmlearnApplica
2025-07-06T17:30:43.126+05:30  INFO 24740 --- [ormlearn] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManag
2025-07-06T17:30:43.128+05:30  INFO 24740 --- [ormlearn] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown
2025-07-06T17:30:43.136+05:30  INFO 24740 --- [ormlearn] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown
```

Explanation:

Steps to Demonstrate HQL & JPQL

1. Create HQL/JPQL method in Repository:
   Use @Query annotation for custom queries.
2. JPQL - Find by Department:
   Define: @Query("SELECT e FROM Employee e WHERE e.department = :dept")
   Use parameter binding with @Param.
3. HQL Fetch Join (Many-to-Many Skills):
   Define: @Query("SELECT e FROM Employee e JOIN FETCH e.skills WHERE e.id = :id")
4. HQL Aggregate Function (Average Salary):
   Define: @Query("SELECT AVG(e.salary) FROM Employee e")

Steps to Demonstrate Native SQL Query

5. Create Native Query Method:
   Define: @Query(value = "SELECT * FROM employee WHERE salary > ?1", nativeQuery = true)
6. Run and Test All Methods in Main Class:
   Call each repository method.
   Print/log the results to verify output.