**CTS WEEK 3:**

**Module 5 - Spring Core and Maven**

**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.

**Steps:**

1. **Set Up a Spring Project:**
   * Create a Maven project named **LibraryManagement**.
   * Add Spring Core dependencies in the **pom.xml** file.
2. **Configure the Application Context:**
   * Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
   * Define beans for **BookService** and **BookRepository** in the XML file.
3. **Define Service and Repository Classes:**
   * Create a package **com.library.service** and add a class **BookService**.
   * Create a package **com.library.repository** and add a class **BookRepository**.
4. **Run the Application:**
   * Create a main class to load the Spring context and test the configuration.

**SOLUTION:**

**Project Structure**

LibraryManagement1

├── src

│ └── main

│ ├── java

│ │ └── com

│ │ └── library

│ │ ├── MainApp.java

│ │ ├── service

│ │ │ └── BookService.java

│ │ └── repository

│ │ └── BookRepository.java

│ └── resources

│ └── applicationContext.xml

└── pom.xml

**Pom.xml:**

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.32</version>

</dependency>

</dependencies>

Right-click project → Maven → Update Project

**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>

* Package: com.library.repository and Add class BookRepository.java

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

System.out.println("Book \"" + bookName + "\" saved to the repository.");

}

}

* Package: com.library.service and Add class 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 a book...");

bookRepository.saveBook(bookName);

}

}

* Package: com.library and Add class MainApp.java

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context =

new ClassPathXmlApplicationContext("applicationContext.xml");

BookService bookService = context.getBean("bookService", BookService.class);

bookService.addBook("The Great Gatsby");

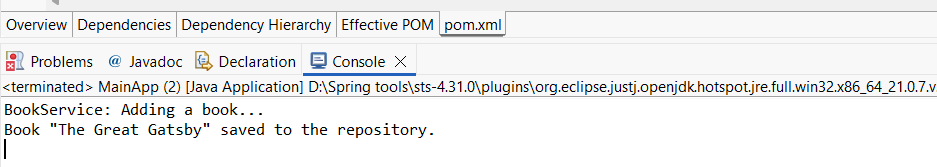
}

}

Right-click MainApp.java

Click Run As → Java Application

**OUTPUT:**

****

**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.

**Steps:**

1. **Modify the XML Configuration:**
   * Update **applicationContext.xml** to wire **BookRepository** into **BookService**.
2. **Update the BookService Class:**
   * Ensure that **BookService** class has a setter method for **BookRepository**.
3. **Test the Configuration:**
   * Run the **LibraryManagementApplication** main class to verify the dependency injection.

**SOLUTION:**

**Modify 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" />

<!-- BookService Bean with Dependency Injection -->

<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("BookService: Adding a book...");

bookRepository.saveBook(bookName);

}

}

**MainApp.java:**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

BookService bookService = (BookService) context.getBean("bookService");

bookService.addBook("The Catcher in the Rye");

}

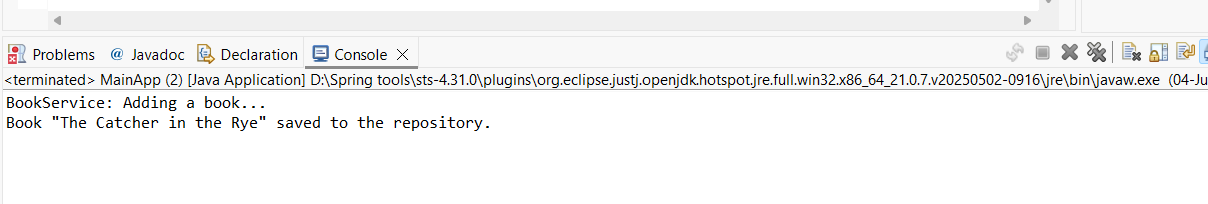
}

**Run MainApp.java:**

Right-click MainApp.java

Click Run As → Java Application

**OUTPUT:**

****

**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.

**Steps:**

1. **Create a New Maven Project:**
   * Create a new Maven project named **LibraryManagement**.
2. **Add Spring Dependencies in pom.xml:**
   * Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.
3. **Configure Maven Plugins:**
   * Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

**SOLUTION:**

New Maven Project - LibraryManagement

Group Id: com.library

Artifact Id: LibraryManagement

Version: 0.0.1-SNAPSHOT (default)

Packaging: jar

Name: LibraryManagement

**Spring Dependencies to 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>

<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.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.32</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.32</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.32</version>

</dependency>

</dependencies>

<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>

**Compile with Java 1.8:**

* Package: com.library and Add class MainApp.java

**MainApp.java**

package com.library;

import java.util.Arrays;

import java.util.List;

public class MainApp {

public static void main(String[] args) {

List<String> books = Arrays.asList("A", "B", "C");

books.forEach(System.out::println); // Java 8 lambda expression

}

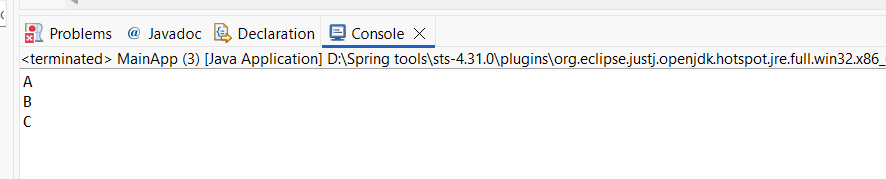
}

**Run MainApp.java**

Right-click MainApp.java

Click Run As → Java Application

**OUTPUT:**

****

**Module 6 - Spring Data JPA with Spring Boot, Hibernate**

**Spring Data JPA - Quick Example   
  
 Software Pre-requisites**

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

**Create a Eclipse Project using Spring Initializr**

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

**> mysql -u root -p**

**mysql> create schema ormlearn;**

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

**# Spring Framework and application log**

**logging.level.org.springframework=info**

**logging.level.com.cognizant=debug**

**# Hibernate logs for displaying executed SQL, input and output**

**logging.level.org.hibernate.SQL=trace**

**logging.level.org.hibernate.type.descriptor.sql=trace**

**# Log pattern**

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

**# Database configuration**

**spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver**

**spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn**

**spring.datasource.username=root**

**spring.datasource.password=root**

**# Hibernate configuration**

**spring.jpa.hibernate.ddl-auto=validate**

**spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect**

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

**import org.slf4j.Logger;**

**import org.slf4j.LoggerFactory;**

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

**public static void main(String[] args) {**

**SpringApplication.run(OrmLearnApplication.class, args);**

**LOGGER.info("Inside main");**

**}**

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

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

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

**Country table creation**

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

**create table country(co\_code varchar(2) primary key, co\_name varchar(50));**

* **Insert couple of records into the table**

**insert into country values ('IN', 'India');**

**insert into country values ('US', 'United States of America');**

**Persistence Class - com.cognizant.orm-learn.model.Country**

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

**import javax.persistence.Column;**

**import javax.persistence.Entity;**

**import javax.persistence.Id;**

**import javax.persistence.Table;**

**@Entity**

**@Table(name="country")**

**public class Country {**

**@Id**

**@Column(name="code")**

**private String code;**

**@Column(name="name")**

**private String name;**

**// getters and setters**

**// toString()**

**}**

***Notes:***

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

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

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

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

**import org.springframework.stereotype.Repository;**

**import com.cognizant.ormlearn.model.Country;**

**@Repository**

**public interface CountryRepository extends JpaRepository<Country, String> {**

**}**

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

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

**Testing in OrmLearnApplication.java**

* **Include a static reference to CountryService in OrmLearnApplication class**

**private static CountryService countryService;**

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

**private static void testGetAllCountries() {**

**LOGGER.info("Start");**

**List<Country> countries = countryService.getAllCountries();**

**LOGGER.debug("countries={}", countries);**

**LOGGER.info("End");**

**}**

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

**ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);**

**countryService = context.getBean(CountryService.class);**

**testGetAllCountries();**

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

**SOLUTION:**

**MY SQL SETUP:**

Create Schema:

CREATE DATABASE ormlearn;

Create Table:

USE ormlearn;

CREATE TABLE country (

code VARCHAR(2) PRIMARY KEY,

name VARCHAR(100)

);

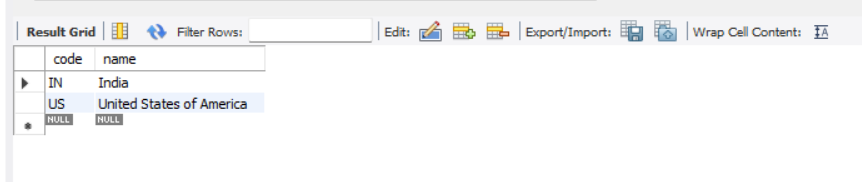
INSERT INTO country (code, name) VALUES

('IN', 'India'),

('US', 'United States of America');

SELECT \* FROM country;

**TABLE OUTPUT:**

**File Structure:**

orm-learn

│

├── src

│ ├── main

│ │ ├── java

│ │ │ └── com

│ │ │ └── cognizant

│ │ │ └── ormlearn

│ │ │ ├── OrmLearnApplication.java

│ │ │ ├── model

│ │ │ │ └── Country.java

│ │ │ ├── repository

│ │ │ │ └── CountryRepository.java

│ │ │ └── service

│ │ │ └── CountryService.java

│ │ └── resources

│ │ └── application.properties

└── pom.xml

**Country.java**

package com.cognizant.ormlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

private String code;

private String name;

public 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;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

**CountryRepository.java**

package com.cognizant.ormlearn.repository;

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

import com.cognizant.ormlearn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

}

**CountryService.java**

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**OrmLearnApplication.java**

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.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;

import java.util.List;

@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) {

System.out.println("Start");

List<Country> countries = countryService.getAllCountries();

System.out.println("Countries: " + countries);

System.out.println("End");

}

}

**Application.properties**

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=root123

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

**Pom.xml**

<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>

</dependency>

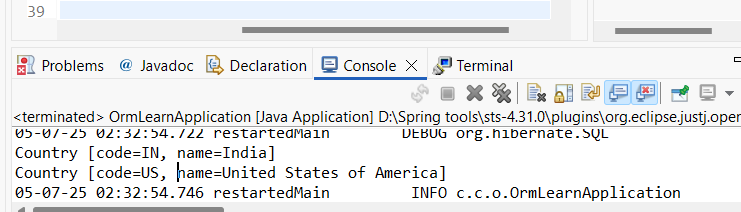
</dependencies>

**Run the project:**

cd /d "D:\Spring tools\orm-learn\orm-learn"

mvn spring-boot:run

**OUTPUT:**

****

**Difference between JPA, Hibernate and Spring Data JPA**   
  
 Java Persistence API (JPA)

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

Hibernate

* ORM Tool that implements JPA

Spring Data JPA

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

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

/\* Method to CREATE an employee in the database \*/

public Integer addEmployee(Employee employee){

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try {

tx = session.beginTransaction();

employeeID = (Integer) session.save(employee);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

}

return employeeID;

}

**Spring Data JPA** EmployeeRespository.java

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

EmployeeService.java

@Autowire

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

​​​​​​​   
**SOLUION:**

**MY SQL SETUP:**

Create Schema:

CREATE DATABASE ormlearn;

Create Table:

USE ormlearn;

CREATE TABLE employee (

id INT AUTO\_INCREMENT PRIMARY KEY,

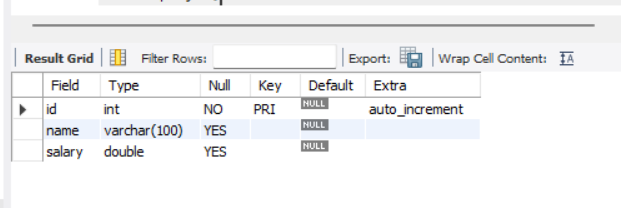
name VARCHAR(100),

salary DOUBLE

);

Desc employee;

**TABLE OUTPUT:**



STRUCTURE:  
orm-learn/

└── src/

└── main/

├── java/

│ └── com/cognizant/ormlearn/

│ ├── OrmLearnApplication.java

│ ├── model/Employee.java

│ ├── repository/EmployeeRepository.java

│ └── service/EmployeeService.java

└── resources/

└── application.properties

**Application.properties**

spring.application.name=orm-learn

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace

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

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=root123

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

**Employee.java**

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

private double salary;

// Getters and Setters

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 double getSalary() { return salary; }

public void setSalary(double salary) { this.salary = salary; }

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", salary=" + salary + "]";

}

}

**EmployeeRepository.java**

package com.cognizant.ormlearn.repository;

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

import com.cognizant.ormlearn.model.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

public void printAllEmployees() {

employeeRepository.findAll().forEach(System.out::println);

}

}

**OrmLearnApplication.java**

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.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 {

Employee emp1 = new Employee();

emp1.setName("John Doe");

emp1.setSalary(60000.0);

employeeService.addEmployee(emp1);

Employee emp2 = new Employee();

emp2.setName("Jane Smith");

emp2.setSalary(75000.0);

employeeService.addEmployee(emp2);

System.out.println("All Employees:");

employeeService.printAllEmployees();

}

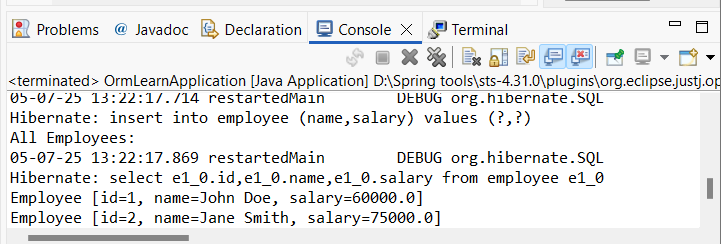
}

**Run the Project:**

cd /d "D:\Spring tools\orm-learn\orm-learn"

mvn spring-boot:run

**OUTPUT:**

****