**Week-3: Spring Data JPA with Spring Boot, Hibernate**

| **Hands-on-2: Spring Data JPA - Quick Example**  **Project Structure:**  **Code:**  application.properties  //application.properties  #spring.application.name=orm-learn  # Logging configuration  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  # Database config  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=Anasuya\_15  # Hibernate config  spring.jpa.hibernate.ddl-auto=update  #spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect  spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect  server.port=8081  Country.java: package com.cognizant.orm\_learn.model;  import jakarta.persistence.Column;  import jakarta.persistence.Entity;  import jakarta.persistence.Id;  import jakarta.persistence.Table;  @Entity  @Table(name = "country")  public class Country {  @Id  @Column(name = "code")  private String code;  @Column(name = "name")  private String name;  // Getters & Setters  public String getCode() {  return code;  }  public void setCode(String code) {  this.code = code;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  @Override  public String toString() {  return "Country [code=" + code + ", name=" + name + "]";  }  }  CountryRepository.java:  package com.cognizant.orm\_learn.repository;  import org.springframework.data.jpa.repository.JpaRepository;  import org.springframework.stereotype.Repository;  import com.cognizant.orm\_learn.model.Country;  @Repository  public interface CountryRepository extends JpaRepository<Country, String> {  }  CountryService.java  package com.cognizant.orm\_learn.service;  import com.cognizant.orm\_learn.model.Country;  import com.cognizant.orm\_learn.repository.CountryRepository;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import org.springframework.transaction.annotation.Transactional;  import java.util.List;  @Service  public class CountryService {  @Autowired  private CountryRepository countryRepository;  @Transactional  public List<Country> getAllCountries() {  return countryRepository.findAll();  }  }  OrmLearnApplication.java  package com.cognizant.orm\_learn;  import com.cognizant.orm\_learn.model.Country;  import com.cognizant.orm\_learn.service.CountryService;  import org.slf4j.Logger;  import org.slf4j.LoggerFactory;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  import org.springframework.context.ApplicationContext;  import java.util.List;  @SpringBootApplication  public class OrmLearnApplication {  private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);  private static CountryService countryService;  public static void main(String[] args) {  ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);  LOGGER.info("Inside main");  countryService = context.getBean(CountryService.class);  testGetAllCountries();  }  private static void testGetAllCountries() {  LOGGER.info("Start");  List<Country> countries = countryService.getAllCountries();  LOGGER.debug("countries={}", countries);  LOGGER.info("End");  }  }    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/> </parent>  <groupId>com.cognizant</groupId>  <artifactId>orm-learn</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>orm-learn</name>  <description>Demo project for Spring Data JPA and Hibernate</description>  <properties>  <java.version>21</java.version> </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-jpa</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <scope>runtime</scope>  <optional>true</optional>  </dependency>  <dependency>  <groupId>com.mysql</groupId>  <artifactId>mysql-connector-j</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.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project>  **Output**      **Hands-on 4: Difference Between JPA, Hibernate, and Spring Data JPA**  **Java Persistence API (JPA)**   * JPA is a **Java specification** (JSR 338) for Object-Relational Mapping (ORM). * It provides a set of **annotations and interfaces** to map Java objects to database tables. * **It is not an implementation**; it requires a provider like Hibernate.   Common Annotations:  @Entity  @Table(name = "employee")  public class Employee {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int id;  private String name;  private double salary;  // getters and setters  } **Hibernate**  * Hibernate is an **open-source ORM framework** and one of the most widely used implementations of JPA. * It provides additional features such as **lazy loading**, **caching**, and **custom dialects**. * When used directly (without Spring), you must manage the **session factory, sessions, and transactions manually**.   Code Example (Manual Hibernate - add employee):  public Integer addEmployee(Employee employee){  Session session = HibernateUtil.getSessionFactory().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**  * Spring Data JPA is a **Spring framework module** built on top of JPA and Hibernate. * It **abstracts away the boilerplate code** and allows you to define **interfaces instead of full DAO implementations**. * Spring handles the session and transaction management automatically.   EmployeeService.java  @Service  public class EmployeeService {  @Autowired  private EmployeeRepository employeeRepository;  @Transactional  public void addEmployee(Employee employee) {  employeeRepository.save(employee);  }  } |
| --- |