Hands-on 1: 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

## 1. Create Spring Boot Project

Go to https://start.spring.io/

Set: Group: com.cognizant, Artifact: orm-learn, Description: Demo project for Spring Data JPA and Hibernate

Select dependencies: Spring Boot DevTools, Spring Data JPA, MySQL Driver

Click Generate, download and extract zip into Eclipse Workspace

## 2. Import into Eclipse

File > Import > Maven > Existing Maven Projects > Browse extracted folder > Finish

## 3. Create Schema in MySQL

mysql -u root -p

create schema ormlearn;

## 4. Configure application.properties

# Logging

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

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

# Log Pattern

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

# Database Configuration

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

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

spring.datasource.username=root

spring.datasource.password=root

# Hibernate Configuration

spring.jpa.hibernate.ddl-auto=validate

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

## 5. Build the Project

mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456

## 6. Log Entry in OrmLearnApplication.java

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

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

LOGGER.info("Inside main");

}

## 7. Database Table Creation

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

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

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

## 8. Create Model Class

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "code")

private String code;

@Column(name = "name")

private String name;

// Getters, setters, toString()

}

## 9. Create Repository Interface

@Repository

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

## 10. Create Service Class

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

## 11. Modify OrmLearnApplication.java to Test

@SpringBootApplication

public class OrmLearnApplication {

private static CountryService countryService;

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

public static void main(String[] args) {

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

countryService = context.getBean(CountryService.class);

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

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

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

LOGGER.info("End");

}

}

# Difference Between JPA, Hibernate, and Spring Data JPA

* Java Persistence API (JPA)

- JPA is a specification (JSR 338) for ORM in Java.

- It provides standard annotations and APIs to map Java objects to database tables.

- It does not include any implementation.

* Hibernate

- Hibernate is a popular ORM tool and one of the implementations of JPA.

- It provides full support for JPA annotations and APIs along with extra features.

- Developers manually manage session, transaction, and entity persistence.

* Spring Data JPA

- It is an abstraction built on top of JPA and often uses Hibernate under the hood.

- It reduces boilerplate code through interfaces like JpaRepository.

- Automatically implements common CRUD and query operations.