**WEEK-3**

**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

1. **Create Spring Boot Project using Spring Initializr**

* Go to <https://start.spring.io/>
* Set:
  + Group: com.cognizant
  + Artifact: orm-learn
  + Description: Demo project for Spring Data JPA and Hibernate
* Add Dependencies:
  + Spring Boot DevTools
  + Spring Data JPA
  + MySQL Driver
* Click Generate, download the ZIP.
* File → Import → Maven → Existing Maven Projects
* Browse to extracted folder
* Finish import

1. Create database in MYSQL

Code:

CREATE DATABASE ormlearn;

USE ormlearn;

CREATE TABLE country (

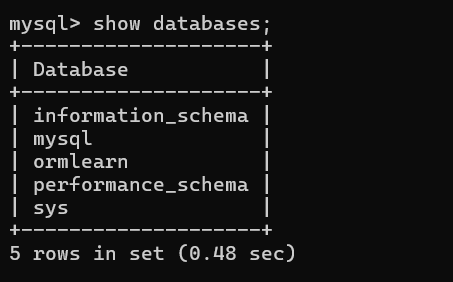
co\_code VARCHAR(2) PRIMARY KEY,

co\_name VARCHAR(50)

);

INSERT INTO country VALUES ('IN', 'India'), ('US', 'United States of America');

**Output:**

:

1. src/main/resources/application.properties:

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=debug

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

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-5level %logger{36} - %msg%n

server.port=8081

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

spring.datasource.username=root

spring.datasource.password=root

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

spring.jpa.hibernate.ddl-auto=validate

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

1. Build Project

* Right click on project -> Run As -> Maven clean
* Right click on project -> Run As -> Maven Install

1. Country.java (Entity Class)

* Src/main/java -> Package (com.cognizant.ormlearn.model) -> Class (Country).

package com.cognizant.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; }

@Override

public String toString() {

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

}

}

1. CountryRepository.java (Interface Repository)

* Src/main/java -> Package (com.cognizant.ormlearn.repository) -> interface (CountryRepository).

package com.cognizant.ormlearn.repository;

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> {

}

1. CountryService.java (Service class)

* Src/main/java -> package (com.cognizant.ormlearn.service) -> class (CountryService).

package com.cognizant.ormlearn.service;

import java.util.List;

import jakarta.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

1. OrmLearnApplication.java

* Src/main/java -> package (com.cognizant.ormlearn)

package com.cognizant.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.cognizant.ormlearn.model.Country;

import com.cognizant.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);

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

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

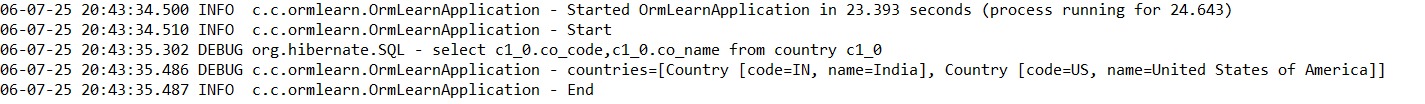
LOGGER.debug("countries={}", countries);

LOGGER.info("End");

}

}

**Output:**



**Explanation:**

* Using Spring Initializr, I made a Spring Boot project called orm-learn, added dependencies for MySQL Driver, DevTools, and Spring Data JPA, and then imported it into my IDE.
* I installed Ormlearn's MySQL database and made a nation table with some basic information, such as "India" and "Russia."
* I made a Country entity class in the code to map to the table, and I used JpaRepository to construct a CountryRepository interface that makes it simple to retrieve data from the database.
* I injected the CountryService class into the main application class and added a function that uses the repository to retrieve all nations.
* The records from the nation table were correctly obtained and logged using SLF4J, according to the Spring Boot console logs when I executed the application.

**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

1. Create Spring Boot Project in Eclipse

* Open Eclipse → Go to File > New > Spring Starter Project.
* Fill in:
* Name: employee-demo
* Group: com.example
* Artifact: employee-demo
* Select dependencies:
* Spring Boot DevTools
* Spring Data JPA
* MySQL Driver
* Click Finish.

1. src/main/resources/application.properties

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

spring.datasource.username=root

spring.datasource.password=root

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

logging.level.org.hibernate.SQL=debug

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

1. Create database in MySQL

Create database employeedb;

1. Employee.java (class)

* Package : com.example.employeedemo.model

package com.example.employeedemo.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.*IDENTITY*)

private Integer id;

@Column(name = "name")

private String name;

public Employee() {}

public Employee(String name) {

this.name = name;

}

public Integer getId() { return id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

@Override

public String toString() {

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

}

}

1. EmployeeRepository.java (interface)

* Package: com.example.employeedemo.repository

package com.example.employeedemo.repository;

import com.example.employeedemo.model.Employee;

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

import org.springframework.stereotype.Repository;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

1. EmployeeService.java (class)

* Package: com.example.employeedemo.service

package com.example.employeedemo.service;

import com.example.employeedemo.model.Employee;

import com.example.employeedemo.repository.EmployeeRepository;

import jakarta.transaction.Transactional;

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

import org.springframework.stereotype.Service;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

1. EmployeeDemoApplication.java (main() class)

* Package: EmployeeDemoApplication.java

package com.example.employeedemo;

import com.example.employeedemo.model.Employee;

import com.example.employeedemo.service.EmployeeService;

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 EmployeeDemoApplication {

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

private static EmployeeService employeeService;

public static void main(String[] args) {

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

employeeService = context.getBean(EmployeeService.class);

LOGGER.info("Inside main");

testAddEmployee();

}

private static void testAddEmployee() {

Employee employee = new Employee("Hamsini");

employeeService.addEmployee(employee);

LOGGER.info("Added Employee: " + employee);

}

}

EmployeeDemoApplication.java -> Run As -> Java Application.

**Output:**

Inside main

Added Employee: Employee [id=27, name=Rakshan]

SELECT \* FROM employee;

**-- Output:**

-- +----+-----------+

-- | id | name |

-- +----+-----------+

-- | 27 | Rakshan |

-- +----+-----------+

**Explanation:**

* I made a new Spring Boot project called employee-demo for this hands-on, adding the required JPA, MySQL, and Spring DevTools requirements.
* I enabled SQL logging, set Hibernate to update the schema automatically, and configured the application.properties file to link to a MySQL database called employeedb.
* I used @GeneratedValue to let the database to automatically produce the main key in an Employee entity class that has fields for id and name.
* I created a service class named EmployeeService with a method to save employee records into the database, as well as an interface called EmployeeRepository that is based on JpaRepository.
* I added a new employee named "Hamsini" to the main application class, and when I ran the program, the data was successfully added to the employee table. I confirmed this using a MySQL query.