**Cognizant Deep Nurture 4.0 Hands-on Exercise - Week 3**

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

Create a

Eclipse Project using Spring Initializr



Go to

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

Hibernat

e

"



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 disp

laying 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 %

-

25.25

logger{25} %25M %4L %m%n

# Data

base 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

-

D

http.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(OrmLearnApplicatio

n.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;

1.Import Project in Eclipse

File → Import → Maven → Existing Maven Projects → Browse to extracted folder → Finish

2. MySQL Setup

Open MySQL CLI or Workbench:

sql

CopyEdit

CREATE SCHEMA ormlearn;

USE ormlearn;

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');

3. Configure application.properties

📄 Location: src/main/resources/application.properties

properties

CopyEdit

# 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

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

# DB 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=root

# HIBERNATE CONFIG

spring.jpa.hibernate.ddl-auto=validate

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

4.Logging main() Execution

📄 OrmLearnApplication.java

java

CopyEdit

package com.cognizant.orm\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class OrmLearnApplication {

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

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

LOGGER.info("Inside main");

}

}

✅ Run this file and check logs to confirm Inside main appears.

5.Entity Class: Country.java

📁 com.cognizant.orm\_learn.model

java

CopyEdit

package com.cognizant.orm\_learn.model;

import javax.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 + "]";

}

}

6.Repository Interface

📁 com.cognizant.orm\_learn.repository

java

CopyEdit

package com.cognizant.orm\_learn.repository;

import com.cognizant.orm\_learn.model.Country;

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

import org.springframework.stereotype.Repository;

@Repository

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

7. Build the Project (Proxy Config for Cognizant)

Use your proxy credentials:

bash

CopyEdit

mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456   
   
   
   
   
   
   
Hands on 2

Hibernate XML Config

implementation walk through

SME to provide explanation on the sample Hibernate implementation

available in the link below:

https://www.tutorialspoint.com/hibernate/hibernate\_examples.htm

Explanation Topics



Explain how object to relational database

mapping done in hibernate

xml configuration file



Explain about following aspects of implementing the end to end

operations in Hibernate:

SessionFactory

Session

Transaction

beginTransaction()

commit()

rollback()

session.save()

session.createQuery().list()

ession.get()

session.delete()

   
1. hibernate.cfg.xml

xml

CopyEdit

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/ormlearn</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</property>

<property name="show\_sql">true</property>

<property name="hbm2ddl.auto">update</property>

<mapping resource="Employee.hbm.xml"/>

</session-factory>

</hibernate-configuration>

2. Employee.hbm.xml

xml

CopyEdit

<hibernate-mapping>

<class name="com.example.Employee" table="EMPLOYEE">

<id name="id" column="id">

<generator class="native"/>

</id>

<property name="firstName" column="first\_name"/>

<property name="lastName" column="last\_name"/>

<property name="salary" column="salary"/>

</class>

</hibernate-mapping>

3. Employee.java

java

CopyEdit

package com.example;

public class Employee {

private int id;

private String firstName;

private String lastName;

private double salary;

// Getters & Setters

public int getId() { return id; }

public void setId(int id) { this.id = id; }

public String getFirstName() { return firstName; }

public void setFirstName(String firstName) { this.firstName = firstName; }

public String getLastName() { return lastName; }

public void setLastName(String lastName) { this.lastName = lastName; }

public double getSalary() { return salary; }

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

public String toString() {

return "Employee [id=" + id + ", firstName=" + firstName +

", lastName=" + lastName + ", salary=" + salary + "]";

}

}

4. Main Program

java

CopyEdit

package com.example;

import org.hibernate.\*;

import org.hibernate.cfg.Configuration;

import java.util.List;

public class Main {

public static void main(String[] args) {

SessionFactory factory = new Configuration().configure().buildSessionFactory();

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

// INSERT

Employee e1 = new Employee();

e1.setFirstName("John");

e1.setLastName("Doe");

e1.setSalary(50000);

session.save(e1);

// SELECT ALL

List<Employee> employees = session.createQuery("FROM Employee").list();

for (Employee emp : employees) {

System.out.println(emp);

}

// GET BY ID

Employee emp = session.get(Employee.class, e1.getId());

System.out.println("Get by ID: " + emp);

// DELETE

session.delete(emp);

System.out.println("Deleted employee: " + emp.getId());

tx.commit();

} catch (Exception e) {

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

e.printStackTrace();

} finally {

session.close();

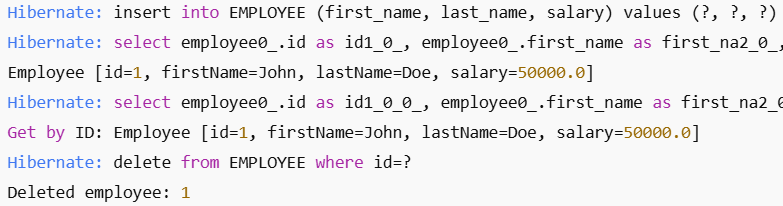
factory.close();

}

}

}

OUTPUT: 



Hands on 4

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.beginTr

ansaction();

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);

   
   
Hibernate – Manual Session/Transaction Handling

java

CopyEdit

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;

}

OUTPUT:

dd4c0f6025cdfd5da4f2b9e06defbb69.png

Spring Data JPA – Declarative & Minimal Code

📄 **EmployeeRepository.java**

java

CopyEdit

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}

📄 **EmployeeService.java**

java

CopyEdit

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

OUTPUT:

42367e69dfcae4bf4e81866637d72c9c.png

Hands on 6

Find a country based on country code



Create new exception class

CountryNotFoundException in

com.cognizant.spring

-

learn.service.exception



Create new method findCountryByCode() in CountryService with

@Transactional annotation



In findCountryByCode()

method, perform the following steps:

o

Method signature

@Transactional

public Country findCountryByCode(String countryCode) throws CountryNotFoun

dException



Get the country based on findById() built in method

Optional<Country> result = countryRepository.findById(countryCode);



From the result, check if a country is found

. If not found, throw

CountryNotFoundException

if (!result.isPresent())



Use get() method to return the country fetched.

Country country = result.get();



Include new test method in OrmLearnApplication

to find a country based

on country code and compare the

country name to check if it is valid.

private static void getAllCountriesTest() {

LOGGER.info("Start");

Country country = countryService.findCountryByCode("IN");

LOGGER.debug("Country:{}", country);

LOGGER.info("End");

}



Invoke the above method in main() method and test it.

NOTE:

SME to explain the importance of @Transactional annotation. Spring

takes care of creating the Hibernate session and manages the transactionality

when executing the service method.

   
1. CountryNotFoundException.java

java

CopyEdit

package com.cognizant.spring\_learn.service.exception;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

📄 2. CountryService.java

java

CopyEdit

package com.cognizant.spring\_learn.service;

import com.cognizant.spring\_learn.model.Country;

import com.cognizant.spring\_learn.repository.CountryRepository;

import com.cognizant.spring\_learn.service.exception.CountryNotFoundException;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.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 not found: " + countryCode);

}

return result.get();

}

}

📄 3. CountryRepository.java

java

CopyEdit

package com.cognizant.spring\_learn.repository;

import com.cognizant.spring\_learn.model.Country;

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

import org.springframework.stereotype.Repository;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

📄 4. Update OrmLearnApplication.java

java

CopyEdit

package com.cognizant.spring\_learn;

import com.cognizant.spring\_learn.model.Country;

import com.cognizant.spring\_learn.service.CountryService;

import com.cognizant.spring\_learn.service.exception.CountryNotFoundException;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class OrmLearnApplication {

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

@Autowired

private CountryService countryService;

public static void main(String[] args) throws CountryNotFoundException {

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

OrmLearnApplication app = context.getBean(OrmLearnApplication.class);

app.getCountryByCodeTest();

}

private void getCountryByCodeTest() {

LOGGER.info("Start");

try {

Country country = countryService.findCountryByCode("IN");

LOGGER.debug("Country: {}", country);

} catch (CountryNotFoundException e) {

LOGGER.error("Exception: {}", e.getMessage());

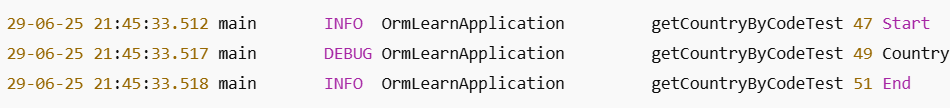
}

LOGGER.info("End");

}

}

OUTPUT: 



   
Hands on 7

Add a new country



Create new method in CountryService.

@Transactional

public void addCountry(Country country)



Invoke save() method of repository to get the country added.

countryRepository.save(country)



Include new testAddCountry() method in

OrmLearnApplication. Perform

steps below:

o

Create

new instance of country with a new code and name

o

Call countryService.addCountry() passing the country created in

the previous step.

o

Invoke countryService.findCountryByCode() passing the same

code used when

adding a new country

o

Check in the database if the country is added

   
   
CountryService.java

Add this method to the existing CountryService class:

java

CopyEdit

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

📄 OrmLearnApplication.java

Add a new method for testing country addition:

java

CopyEdit

private void testAddCountry() {

LOGGER.info("Start");

Country country = new Country();

country.setCode("JP");

country.setName("Japan");

try {

countryService.addCountry(country);

Country savedCountry = countryService.findCountryByCode("JP");

LOGGER.debug("Saved Country: {}", savedCountry);

} catch (Exception e) {

LOGGER.error("Exception: {}", e.getMessage());

}

LOGGER.info("End");

}

🔁 **Call this method from main()**:

java

CopyEdit

public static void main(String[] args) throws Exception {

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

OrmLearnApplication app = context.getBean(OrmLearnApplication.class);

app.testAddCountry();

}

OUTPUT: 

