SPRING DATA JPA HANDS ON 1

Superset ID : 6384831

Name : Mohana Priya N

E-mail : mohanapriya.2205056@srec.ac.in

Mandatory Questions

1) Hands on 1: Spring Data JPA - Quick Example

Solution:

//LearnApplication.java

```
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);

        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");
    }
}
```

2) Hands on 4: Difference between JPA, Hibernate and Spring Data JPA

Solution:

Hibernate CRUD example:

```
Session session = factory.openSession();
Transaction tx = session.beginTransaction();
Employee employee = new Employee();
employee.setName("John Doe");
session.save(employee);
tx.commit();
session.close();
```

Spring Data JPA CRUD example:

```
@Repository
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}
@Service
public class EmployeeService {
    @Autowired
    private EmployeeRepository employeeRepository;
    @Transactional
    public void addEmployee(Employee employee) {
        employeeRepository.save(employee);
    }
}
```

Other hands on:

3) Hands on 2 : Hibernate XML Config implementation walk through

Solution:

//Hibernate xml

```
<hibernate-configuration>
   <session-factory>
      <property name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</property>
      <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.hbm2ddl.auto">update</property>
   </session-factory>
</hibernate-configuration>
```

4) Hands on 3 : Hibernate Annotation Config implementation walk through

Solution:

//java

```
package com.cognizant.ormlearn.model;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Column;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
@Entity
@Table(name = "employee")
public class Employee {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    @Column(name = "id")
    private int id;
    @Column(name = "name")
    private String name;
    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; }
}
```

5) Hands on 5: Implement services for managing Country

Solution:

//**country.java**

```
package com.cognizant.ormlearn.service;
import java.util.List;
import javax.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();
    }
    @Transactional
    public Country findCountryByCode(String code) {
        return countryRepository.findById(code).orElse(null);
    }
    @Transactional
    public void addCountry(Country country) {
        countryRepository.save(country);
    }
    @Transactional
    public void updateCountry(String code, String name) {
        Country country = countryRepository.findById(code).orElse(null);
        if (country != null) {
            country.setName(name);
            countryRepository.save(country);
        }
    }
    @Transactional
    public void deleteCountry(String code) {
        countryRepository.deleteById(code);
    }
}
```

6) Hands on 6: Find a country based on country code

Solution:

//country.java

```
package com.cognizant.ormlearn.service.exception;
public class CountryNotFoundException extends Exception {
    public CountryNotFoundException(String message) {
        super(message);
    }
}
```

//code.java

```
@Transactional
public Country findCountryByCode(String code) throws CountryNotFoundException {
   return countryRepository.findById(code)
           .orElseThrow(() -> new CountryNotFoundException("Country not found for code: " + code));
}
```

7) Hands on 7: Add a new country

Solution:

```
@Transactional
public void addCountry(Country country) {
   countryRepository.save(country);
}
```

8) Hands on 8 : Update a country based on code

Solution:

```
@Transactional
public void updateCountry(String code, String name) {
   Country country = countryRepository.findById(code).orElse(null);
   if (country != null) {
       country.setName(name);
       countryRepository.save(country);
   }
}
```

9) Hands on 9 : Delete a country based on code

Solution:

```
@Transactional
public void deleteCountry(String code) {
   countryRepository.deleteById(code);}
```