**Hands on 1**

* **Spring Data JPA - Quick Example**

**OrmApplication.java**

package com.cognizant.orm\_learn;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

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

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

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

@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");

}

@Bean

CommandLineRunner run(CountryRepository repo, CountryService countryService) {

return args -> {

// Insert sample data

repo.save(new Country("IN", "India"));

repo.save(new Country("US", "United States of America"));

LOGGER.info("Start fetching countries");

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

countries.forEach(country -> LOGGER.info("{}", country));

LOGGER.info("End fetching countries");

};

}

}

**Country.java**

package com.cognizant.orm\_learn;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

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

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

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

@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");

}

@Bean

CommandLineRunner run(CountryRepository repo, CountryService countryService) {

return args -> {

// Insert sample data

repo.save(new Country("IN", "India"));

repo.save(new Country("US", "United States of America"));

LOGGER.info("Start fetching countries");

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

countries.forEach(country -> LOGGER.info("{}", country));

LOGGER.info("End fetching countries");

};

}

}

**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.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> {

}

**Application.properties**

package com.cognizant.orm\_learn.repository;

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

import org.springframework.stereotype.Repository;

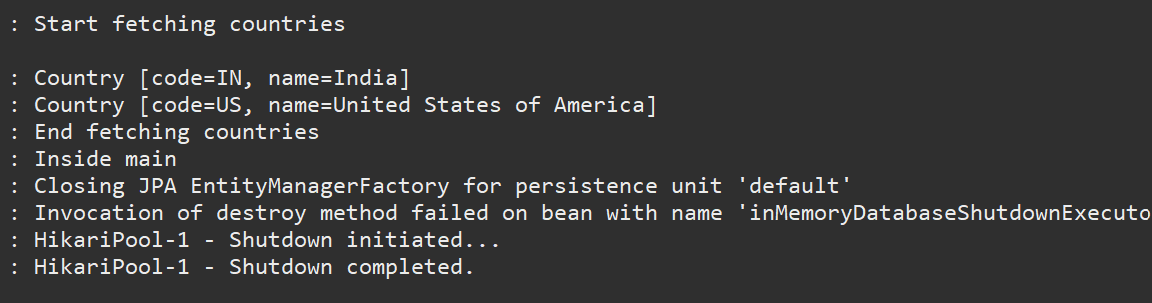
import com.cognizant.orm\_learn.model.Country;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

**Output**



# Hands-on 4

## Difference between JPA, Hibernate, and Spring Data JPA

### 1. Java Persistence API (JPA)

- JPA is a Java specification (JSR 338) for managing relational data in Java applications.  
- It defines standard annotations and interfaces such as @Entity, @Id, and EntityManager, but does not provide an implementation.  
- It is vendor-neutral and requires a concrete implementation like Hibernate to be functional.

### 2. Hibernate

- Hibernate is a popular ORM (Object Relational Mapping) tool and the most widely used implementation of JPA.  
- It provides additional features beyond the JPA specification, including caching and lazy loading.  
- Using Hibernate directly involves manually managing sessions and transactions using classes like SessionFactory, Session, and Transaction.

### 3. Spring Data JPA

- Spring Data JPA is part of the Spring ecosystem and acts as an abstraction layer over JPA.  
- It simplifies data access by providing repository interfaces and using method naming conventions to auto-generate queries.  
- It internally uses a JPA implementation like Hibernate and manages transactions and entity managers automatically.

### 4. Comparison Table

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | JPA | Hibernate | Spring Data JPA |
| Type | Specification | ORM Framework | Abstraction over JPA/Hibernate |
| Implementation Provided | No | Yes | No (uses JPA implementation) |
| Boilerplate Code | Medium | High | Low |
| Requires Configuration | Yes | Yes | Yes (minimal) |
| Repository Abstraction | No | No | Yes (JpaRepository, CrudRepository) |
| Usage in Project | Used annotations like @Entity, @Id | Used internally as JPA provider | Created CountryRepository extending JpaRepository |

### **Code Comparison**

### Using Hibernate:

Session session = factory.openSession();  
Transaction tx = null;  
try {  
 tx = session.beginTransaction();  
 session.save(employee);  
 tx.commit();  
} catch (Exception e) {  
 if (tx != null) tx.rollback();  
} finally {  
 session.close();  
}

**Using Spring Data JPA:**

@Autowired  
private EmployeeRepository employeeRepository;  
  
@Transactional  
public void addEmployee(Employee employee) {  
 employeeRepository.save(employee);  
}

### 6. Summary of Project Implementation

- In the project, we used:  
- JPA annotations in the Country entity class to map it to the database table.  
- Spring Data JPA to define CountryRepository, which extends JpaRepository.  
- Hibernate as the default JPA provider to manage ORM internally.  
- Spring Boot to handle configuration and dependency injection, reducing boilerplate code significantly.

**Hands On 5: Implement services for managing Country** 

**Country.java**

package com.example.country.entity;  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
  
@Entity  
public class Country {  
 @Id  
 private String code;  
 private String name;  
  
 *// Getters and 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;  
 }  
}

**CountryRepository.java**

package com.example.country.repository;  
  
import com.example.country.entity.Country;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface CountryRepository extends JpaRepository<Country, String> {  
}

**CountryApplication.java**

package com.example.country;  
  
import com.example.country.repository.CountryRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class CountryApplication implements CommandLineRunner {  
  
 @Autowired  
 private CountryRepository countryRepository;  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(CountryApplication.class, args);  
 }  
  
 @Override  
 public void run(String... args) {  
 System.*out*.println("All countries:");  
 countryRepository.findAll().forEach(c -> System.*out*.println(c.getName()));  
  
 }  
}

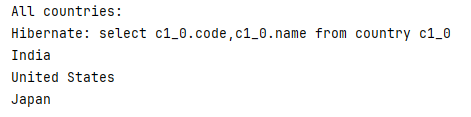
**application.properties**

spring.application.name=country  
spring.datasource.url=jdbc:h2:mem:testdb  
spring.datasource.driver-class-name=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=  
spring.jpa.hibernate.ddl-auto=create  
spring.datasource.initialization-mode=always  
spring.h2.console.enabled=true

**import.sql**

INSERT INTO country (code, name) VALUES ('IN', 'India');  
INSERT INTO country (code, name) VALUES ('US', 'United States');  
INSERT INTO country (code, name) VALUES ('JP', 'Japan');

**Output**

****

**Hands On 6: Find a country based on country code** 

**Country.java**

package com.example.country.entity;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
  
@Entity  
public class Country {  
 @Id  
 private String code;  
 private String name;  
public Country() {}  
 public Country(String code, String name) {  
 this.code = code;  
 this.name = 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 + "]";  
 }  
}

**CountryRepository.java**

package com.example.country.repository;  
  
import com.example.country.entity.Country;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface CountryRepository extends JpaRepository<Country, String> {  
}

**CountryService.java**

package com.example.country.service;  
  
import com.example.country.entity.Country;  
import com.example.country.repository.CountryRepository;  
import com.example.country.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 with code: " + countryCode);  
 }  
 return result.get();  
 }  
}

**CountryNotFoundException.java**

package com.example.country.service.exception;  
  
public class CountryNotFoundException extends Exception {  
 public CountryNotFoundException(String message) {  
 super(message);  
 }  
}

**CountryApplication.java**

package com.example.country;  
  
import com.example.country.entity.Country;  
import com.example.country.repository.CountryRepository;  
import com.example.country.service.exception.CountryNotFoundException;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import com.example.country.service.CountryService;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
@SpringBootApplication  
public class CountryApplication implements CommandLineRunner {  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(CountryApplication.class);  
  
 @Autowired  
 private CountryRepository countryRepository;  
 @Autowired  
 private CountryService countryService;  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(CountryApplication.class, args);  
 }  
  
 @Override  
 public void run(String... args) {  
 getCountryTest();  
 }  
  
 private void getCountryTest() {  
  
 *LOGGER*.info("Start");  
  
 try {  
 Country country = countryService.findCountryByCode("IN");  
 *LOGGER*.debug("Country: {}", country);  
 System.*out*.println("Country: " + country);  
  
 } catch (CountryNotFoundException e) {  
 *LOGGER*.error("Exception: {}", e.getMessage());  
 }  
  
 *LOGGER*.info("End");  
 }  
  
}

**application.properties**

spring.application.name=country  
spring.datasource.url=jdbc:h2:mem:testdb  
spring.datasource.driver-class-name=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=  
spring.jpa.hibernate.ddl-auto=create  
spring.datasource.initialization-mode=always  
spring.h2.console.enabled=true

**import.sql**

insert into country (code, name) values ('AF', 'Afghanistan');  
insert into country (code, name) values ('AL', 'Albania');  
insert into country (code, name) values ('DZ', 'Algeria');  
insert into country (code, name) values ('AS', 'American Samoa');  
insert into country (code, name) values ('AD', 'Andorra');  
insert into country (code, name) values ('AO', 'Angola');  
insert into country (code, name) values ('AI', 'Anguilla');  
insert into country (code, name) values ('AQ', 'Antarctica');  
insert into country (code, name) values ('AG', 'Antigua and Barbuda');  
insert into country (code, name) values ('AR', 'Argentina');  
insert into country (code, name) values ('AM', 'Armenia');  
insert into country (code, name) values ('AW', 'Aruba');  
insert into country (code, name) values ('AU', 'Australia');  
insert into country (code, name) values ('AT', 'Austria');  
insert into country (code, name) values ('AZ', 'Azerbaijan');  
insert into country (code, name) values ('BS', 'Bahamas');  
insert into country (code, name) values ('BH', 'Bahrain');  
insert into country (code, name) values ('BD', 'Bangladesh');  
insert into country (code, name) values ('BB', 'Barbados');  
insert into country (code, name) values ('BY', 'Belarus');  
insert into country (code, name) values ('BE', 'Belgium');  
insert into country (code, name) values ('BZ', 'Belize');  
insert into country (code, name) values ('BJ', 'Benin');  
insert into country (code, name) values ('BM', 'Bermuda');  
insert into country (code, name) values ('BT', 'Bhutan');  
insert into country (code, name) values ('BO', 'Bolivia, Plurinational State of');  
insert into country (code, name) values ('BQ', 'Bonaire, Sint Eustatius and Saba');  
insert into country (code, name) values ('BA', 'Bosnia and Herzegovina');  
insert into country (code, name) values ('BW', 'Botswana');  
insert into country (code, name) values ('BV', 'Bouvet Island');  
insert into country (code, name) values ('BR', 'Brazil');  
insert into country (code, name) values ('IO', 'British Indian Ocean Territory');  
insert into country (code, name) values ('BN', 'Brunei Darussalam');  
insert into country (code, name) values ('BG', 'Bulgaria');  
insert into country (code, name) values ('BF', 'Burkina Faso');  
insert into country (code, name) values ('BI', 'Burundi');  
insert into country (code, name) values ('KH', 'Cambodia');  
insert into country (code, name) values ('CM', 'Cameroon');  
insert into country (code, name) values ('CA', 'Canada');  
insert into country (code, name) values ('CV', 'Cape Verde');  
insert into country (code, name) values ('KY', 'Cayman Islands');  
insert into country (code, name) values ('CF', 'Central African Republic');  
insert into country (code, name) values ('TD', 'Chad');  
insert into country (code, name) values ('CL', 'Chile');  
insert into country (code, name) values ('CN', 'China');  
insert into country (code, name) values ('CX', 'Christmas Island');  
insert into country (code, name) values ('CC', 'Cocos (Keeling) Islands');  
insert into country (code, name) values ('CO', 'Colombia');  
insert into country (code, name) values ('KM', 'Comoros');  
insert into country (code, name) values ('CG', 'Congo');  
insert into country (code, name) values ('CD', 'Congo, the Democratic Republic of the');  
insert into country (code, name) values ('CK', 'Cook Islands');  
insert into country (code, name) values ('CR', 'Costa Rica');  
insert into country (code, name) values ('HR', 'Croatia');  
insert into country (code, name) values ('CU', 'Cuba');  
insert into country (code, name) values ('CW', 'Curaçao');  
insert into country (code, name) values ('CY', 'Cyprus');  
insert into country (code, name) values ('CZ', 'Czech Republic');  
insert into country (code, name) values ('CI', 'Côte d'Ivoire'');  
insert into country (code, name) values ('DK', 'Denmark');  
insert into country (code, name) values ('DJ', 'Djibouti');  
insert into country (code, name) values ('DM', 'Dominica');  
insert into country (code, name) values ('DO', 'Dominican Republic');  
insert into country (code, name) values ('EC', 'Ecuador');  
insert into country (code, name) values ('EG', 'Egypt');  
insert into country (code, name) values ('SV', 'El Salvador');  
insert into country (code, name) values ('GQ', 'Equatorial Guinea');  
insert into country (code, name) values ('ER', 'Eritrea');  
insert into country (code, name) values ('EE', 'Estonia');  
insert into country (code, name) values ('ET', 'Ethiopia');  
insert into country (code, name) values ('FK', 'Falkland Islands (Malvinas)');  
insert into country (code, name) values ('FO', 'Faroe Islands');  
insert into country (code, name) values ('FJ', 'Fiji');  
insert into country (code, name) values ('FI', 'Finland');  
insert into country (code, name) values ('FR', 'France');  
insert into country (code, name) values ('GF', 'French Guiana');  
insert into country (code, name) values ('PF', 'French Polynesia');  
insert into country (code, name) values ('TF', 'French Southern Territories');  
insert into country (code, name) values ('GA', 'Gabon');  
insert into country (code, name) values ('GM', 'Gambia');  
insert into country (code, name) values ('GE', 'Georgia');  
insert into country (code, name) values ('DE', 'Germany');  
insert into country (code, name) values ('GH', 'Ghana');  
insert into country (code, name) values ('GI', 'Gibraltar');  
insert into country (code, name) values ('GR', 'Greece');  
insert into country (code, name) values ('GL', 'Greenland');  
insert into country (code, name) values ('GD', 'Grenada');  
insert into country (code, name) values ('GP', 'Guadeloupe');  
insert into country (code, name) values ('GU', 'Guam');  
insert into country (code, name) values ('GT', 'Guatemala');  
insert into country (code, name) values ('GG', 'Guernsey');  
insert into country (code, name) values ('GN', 'Guinea');  
insert into country (code, name) values ('GW', 'Guinea-Bissau');  
insert into country (code, name) values ('GY', 'Guyana');  
insert into country (code, name) values ('HT', 'Haiti');  
insert into country (code, name) values ('HM', 'Heard Island and McDonald Islands');  
insert into country (code, name) values ('VA', 'Holy See (Vatican City State)');  
insert into country (code, name) values ('HN', 'Honduras');  
insert into country (code, name) values ('HK', 'Hong Kong');  
insert into country (code, name) values ('HU', 'Hungary');  
insert into country (code, name) values ('IS', 'Iceland');  
insert into country (code, name) values ('IN', 'India');  
insert into country (code, name) values ('ID', 'Indonesia');  
insert into country (code, name) values ('IR', 'Iran, Islamic Republic of');  
insert into country (code, name) values ('IQ', 'Iraq');  
insert into country (code, name) values ('IE', 'Ireland');  
insert into country (code, name) values ('IM', 'Isle of Man');  
insert into country (code, name) values ('IL', 'Israel');  
insert into country (code, name) values ('IT', 'Italy');  
insert into country (code, name) values ('JM', 'Jamaica');  
insert into country (code, name) values ('JP', 'Japan');  
insert into country (code, name) values ('JE', 'Jersey');  
insert into country (code, name) values ('JO', 'Jordan');  
insert into country (code, name) values ('KZ', 'Kazakhstan');  
insert into country (code, name) values ('KE', 'Kenya');  
insert into country (code, name) values ('KI', 'Kiribati');  
insert into country (code, name) values ('KP', 'Democratic People's Republic of Korea'');  
insert into country (code, name) values ('KR', 'Republic of Korea');  
insert into country (code, name) values ('KW', 'Kuwait');  
insert into country (code, name) values ('KG', 'Kyrgyzstan');  
insert into country (code, name) values ('LA', 'Lao People's Democratic Republic'');  
insert into country (code, name) values ('LV', 'Latvia');  
insert into country (code, name) values ('LB', 'Lebanon');  
insert into country (code, name) values ('LS', 'Lesotho');  
insert into country (code, name) values ('LR', 'Liberia');  
insert into country (code, name) values ('LY', 'Libya');  
insert into country (code, name) values ('LI', 'Liechtenstein');  
insert into country (code, name) values ('LT', 'Lithuania');  
insert into country (code, name) values ('LU', 'Luxembourg');  
insert into country (code, name) values ('MO', 'Macao');  
insert into country (code, name) values ('MK', 'Macedonia, the Former Yugoslav Republic of');  
insert into country (code, name) values ('MG', 'Madagascar');  
insert into country (code, name) values ('MW', 'Malawi');  
insert into country (code, name) values ('MY', 'Malaysia');  
insert into country (code, name) values ('MV', 'Maldives');  
insert into country (code, name) values ('ML', 'Mali');  
insert into country (code, name) values ('MT', 'Malta');  
insert into country (code, name) values ('MH', 'Marshall Islands');  
insert into country (code, name) values ('MQ', 'Martinique');  
insert into country (code, name) values ('MR', 'Mauritania');  
insert into country (code, name) values ('MU', 'Mauritius');  
insert into country (code, name) values ('YT', 'Mayotte');  
insert into country (code, name) values ('MX', 'Mexico');  
insert into country (code, name) values ('FM', 'Micronesia, Federated States of');  
insert into country (code, name) values ('MD', 'Moldova, Republic of');  
insert into country (code, name) values ('MC', 'Monaco');  
insert into country (code, name) values ('MN', 'Mongolia');  
insert into country (code, name) values ('ME', 'Montenegro');  
insert into country (code, name) values ('MS', 'Montserrat');  
insert into country (code, name) values ('MA', 'Morocco');  
insert into country (code, name) values ('MZ', 'Mozambique');  
insert into country (code, name) values ('MM', 'Myanmar');  
insert into country (code, name) values ('NA', 'Namibia');  
insert into country (code, name) values ('NR', 'Nauru');  
insert into country (code, name) values ('NP', 'Nepal');  
insert into country (code, name) values ('NL', 'Netherlands');  
insert into country (code, name) values ('NC', 'New Caledonia');  
insert into country (code, name) values ('NZ', 'New Zealand');  
insert into country (code, name) values ('NI', 'Nicaragua');  
insert into country (code, name) values ('NE', 'Niger');  
insert into country (code, name) values ('NG', 'Nigeria');  
insert into country (code, name) values ('NU', 'Niue');  
insert into country (code, name) values ('NF', 'Norfolk Island');  
insert into country (code, name) values ('MP', 'Northern Mariana Islands');  
insert into country (code, name) values ('NO', 'Norway');  
insert into country (code, name) values ('OM', 'Oman');  
insert into country (code, name) values ('PK', 'Pakistan');  
insert into country (code, name) values ('PW', 'Palau');  
insert into country (code, name) values ('PS', 'Palestine, State of');  
insert into country (code, name) values ('PA', 'Panama');  
insert into country (code, name) values ('PG', 'Papua New Guinea');  
insert into country (code, name) values ('PY', 'Paraguay');  
insert into country (code, name) values ('PE', 'Peru');  
insert into country (code, name) values ('PH', 'Philippines');  
insert into country (code, name) values ('PN', 'Pitcairn');  
insert into country (code, name) values ('PL', 'Poland');  
insert into country (code, name) values ('PT', 'Portugal');  
insert into country (code, name) values ('PR', 'Puerto Rico');  
insert into country (code, name) values ('QA', 'Qatar');  
insert into country (code, name) values ('RO', 'Romania');  
insert into country (code, name) values ('RU', 'Russian Federation');  
insert into country (code, name) values ('RW', 'Rwanda');  
insert into country (code, name) values ('RE', 'Réunion');  
insert into country (code, name) values ('BL', 'Saint Barthélemy');  
insert into country (code, name) values ('SH', 'Saint Helena, Ascension and Tristan da Cunha');  
insert into country (code, name) values ('KN', 'Saint Kitts and Nevis');  
insert into country (code, name) values ('LC', 'Saint Lucia');  
insert into country (code, name) values ('MF', 'Saint Martin (French part)');  
insert into country (code, name) values ('PM', 'Saint Pierre and Miquelon');  
insert into country (code, name) values ('VC', 'Saint Vincent and the Grenadines');  
insert into country (code, name) values ('WS', 'Samoa');  
insert into country (code, name) values ('SM', 'San Marino');  
insert into country (code, name) values ('ST', 'Sao Tome and Principe');  
insert into country (code, name) values ('SA', 'Saudi Arabia');  
insert into country (code, name) values ('SN', 'Senegal');  
insert into country (code, name) values ('RS', 'Serbia');  
insert into country (code, name) values ('SC', 'Seychelles');  
insert into country (code, name) values ('SL', 'Sierra Leone');  
insert into country (code, name) values ('SG', 'Singapore');  
insert into country (code, name) values ('SX', 'Sint Maarten (Dutch part)');  
insert into country (code, name) values ('SK', 'Slovakia');  
insert into country (code, name) values ('SI', 'Slovenia');  
insert into country (code, name) values ('SB', 'Solomon Islands');  
insert into country (code, name) values ('SO', 'Somalia');  
insert into country (code, name) values ('ZA', 'South Africa');  
insert into country (code, name) values ('GS', 'South Georgia and the South Sandwich Islands');  
insert into country (code, name) values ('SS', 'South Sudan');  
insert into country (code, name) values ('ES', 'Spain');  
insert into country (code, name) values ('LK', 'Sri Lanka');  
insert into country (code, name) values ('SD', 'Sudan');  
insert into country (code, name) values ('SR', 'Suriname');  
insert into country (code, name) values ('SJ', 'Svalbard and Jan Mayen');  
insert into country (code, name) values ('SZ', 'Swaziland');  
insert into country (code, name) values ('SE', 'Sweden');  
insert into country (code, name) values ('CH', 'Switzerland');  
insert into country (code, name) values ('SY', 'Syrian Arab Republic');  
insert into country (code, name) values ('TW', 'Taiwan, Province of China');  
insert into country (code, name) values ('TJ', 'Tajikistan');  
insert into country (code, name) values ('TZ', 'Tanzania, United Republic of');  
insert into country (code, name) values ('TH', 'Thailand');  
insert into country (code, name) values ('TL', 'Timor-Leste');  
insert into country (code, name) values ('TG', 'Togo');  
insert into country (code, name) values ('TK', 'Tokelau');  
insert into country (code, name) values ('TO', 'Tonga');  
insert into country (code, name) values ('TT', 'Trinidad and Tobago');  
insert into country (code, name) values ('TN', 'Tunisia');  
insert into country (code, name) values ('TR', 'Turkey');  
insert into country (code, name) values ('TM', 'Turkmenistan');  
insert into country (code, name) values ('TC', 'Turks and Caicos Islands');  
insert into country (code, name) values ('TV', 'Tuvalu');  
insert into country (code, name) values ('UG', 'Uganda');  
insert into country (code, name) values ('UA', 'Ukraine');  
insert into country (code, name) values ('AE', 'United Arab Emirates');  
insert into country (code, name) values ('GB', 'United Kingdom');  
insert into country (code, name) values ('US', 'United States');  
insert into country (code, name) values ('UM', 'United States Minor Outlying Islands');  
insert into country (code, name) values ('UY', 'Uruguay');  
insert into country (code, name) values ('UZ', 'Uzbekistan');  
insert into country (code, name) values ('VU', 'Vanuatu');  
insert into country (code, name) values ('VE', 'Venezuela, Bolivarian Republic of');  
insert into country (code, name) values ('VN', 'Viet Nam');  
insert into country (code, name) values ('VG', 'Virgin Islands, British');  
insert into country (code, name) values ('VI', 'Virgin Islands, U.S.');  
insert into country (code, name) values ('WF', 'Wallis and Futuna');  
insert into country (code, name) values ('EH', 'Western Sahara');  
insert into country (code, name) values ('YE', 'Yemen');  
insert into country (code, name) values ('ZM', 'Zambia');  
insert into country (code, name) values ('ZW', 'Zimbabwe');  
insert into country (code, name) values ('AX', 'Åland Islands');

**Output**



**Hands On 7: Add a new country**

**Country.java**

package com.example.country.entity;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
  
@Entity  
public class Country {  
 @Id  
 private String code;  
 private String name;  
public Country() {}  
 public Country(String code, String name) {  
 this.code = code;  
 this.name = 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 + "]";  
 }  
}

**CountryRepository.java**

package com.example.country.repository;  
  
import com.example.country.entity.Country;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface CountryRepository extends JpaRepository<Country, String> {  
}

**CountryService.java**

package com.example.country.service;  
  
import com.example.country.entity.Country;  
import com.example.country.repository.CountryRepository;  
import com.example.country.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 with code: " + countryCode);  
 }  
 return result.get();  
 }  
 @Transactional  
 public void addCountry(Country country) {  
 countryRepository.save(country);  
 }  
  
}

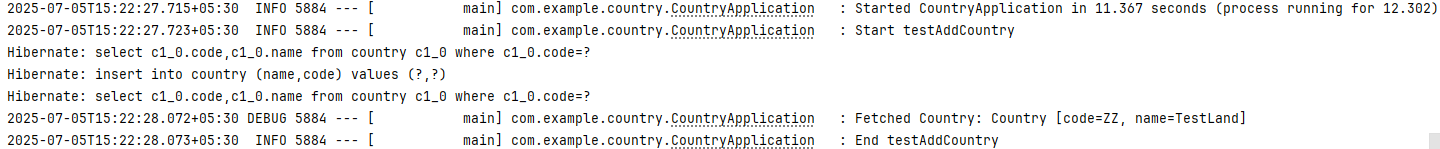
**CountryApplication.java**

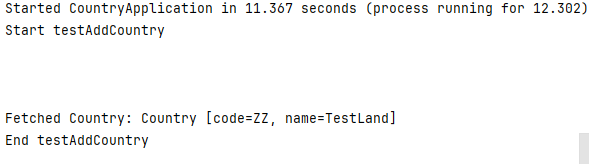
package com.example.country;  
  
import com.example.country.entity.Country;  
import com.example.country.repository.CountryRepository;  
import com.example.country.service.exception.CountryNotFoundException;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import com.example.country.service.CountryService;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
@SpringBootApplication  
public class CountryApplication implements CommandLineRunner {  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(CountryApplication.class);  
  
 @Autowired  
 private CountryRepository countryRepository;  
 @Autowired  
 private CountryService countryService;  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(CountryApplication.class, args);  
 }  
  
 @Override  
 public void run(String... args) {  
 testAddCountry();  
 }  
  
 private void testAddCountry() {  
 *LOGGER*.info("Start testAddCountry");  
  
 Country newCountry = new Country();  
 newCountry.setCode("ZZ");  
 newCountry.setName("TestLand");  
  
 countryService.addCountry(newCountry);  
  
 try {  
 Country fetched = countryService.findCountryByCode("ZZ");  
 *LOGGER*.debug("Fetched Country: {}", fetched);  
 } catch (CountryNotFoundException e) {  
 *LOGGER*.error("Country not found: {}", e.getMessage());  
 }  
  
 *LOGGER*.info("End testAddCountry");  
 }  
}

**CountryApplication.java**

spring.h2.console.enabled=true  
spring.h2.console.path=/h2-console  
  
spring.datasource.url=jdbc:h2:mem:testdb  
spring.datasource.driverClassName=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=  
  
spring.jpa.show-sql=true  
spring.jpa.hibernate.ddl-auto=create  
spring.datasource.initialization-mode=always  
  
logging.level.com.example.country=DEBUG

**Output**





**Hands On 8: Update a country based on code**   
**Country.java**

package com.example.country.entity;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
  
@Entity  
public class Country {  
 @Id  
 private String code;  
 private String name;  
  
 *// Constructors* public Country() {}  
 public Country(String code, String name) {  
 this.code = code;  
 this.name = 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 + "]";  
 }  
}

**CountryRepository.java**

package com.example.country.repository;  
  
import com.example.country.entity.Country;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface CountryRepository extends JpaRepository<Country, String> {  
}

**CountryService.java**

package com.example.country.service;  
  
import com.example.country.entity.Country;  
import com.example.country.repository.CountryRepository;  
import com.example.country.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 with code: " + countryCode);  
 }  
 return result.get();  
 }  
 @Transactional  
 public void addCountry(Country country) {  
 countryRepository.save(country);  
 }

@Transactional  
public void updateCountry(String code, String newName) throws CountryNotFoundException {  
 Country country = countryRepository.findById(code)  
 .orElseThrow(() -> new CountryNotFoundException("Country not found for code " + code));  
 country.setName(newName);  
 countryRepository.save(country);  
}

}

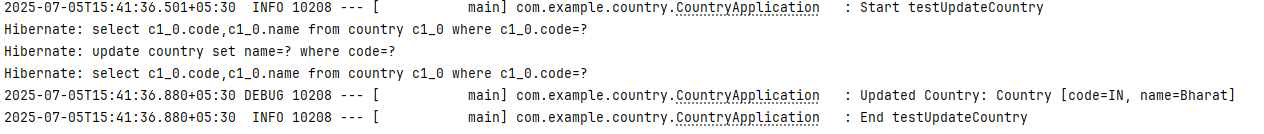
**CountryApplication.java**

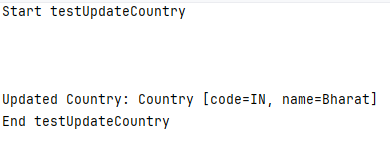
package com.example.country;  
  
import com.example.country.entity.Country;  
import com.example.country.repository.CountryRepository;  
import com.example.country.service.exception.CountryNotFoundException;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import com.example.country.service.CountryService;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
@SpringBootApplication  
public class CountryApplication implements CommandLineRunner {  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(CountryApplication.class);  
  
 @Autowired  
 private CountryRepository countryRepository;  
 @Autowired  
 private CountryService countryService;  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(CountryApplication.class, args);  
 }  
  
 @Override  
 public void run(String... args) {  
 testUpdateCountry();  
 }  
  
 private void testUpdateCountry() {  
 *LOGGER*.info("Start testUpdateCountry");  
  
 try {  
 countryService.updateCountry("IN", "Bharat");  
 Country updated = countryService.findCountryByCode("IN");  
 *LOGGER*.debug("Updated Country: {}", updated);  
 } catch (CountryNotFoundException e) {  
 *LOGGER*.error("Country not found: {}", e.getMessage());  
 }  
  
 *LOGGER*.info("End testUpdateCountry");  
 }  
  
}

**application.properties**

spring.h2.console.enabled=true  
spring.h2.console.path=/h2-console  
  
spring.datasource.url=jdbc:h2:mem:testdb  
spring.datasource.driverClassName=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=  
  
spring.jpa.show-sql=true  
spring.jpa.hibernate.ddl-auto=create  
spring.datasource.initialization-mode=always  
  
logging.level.com.example.country=DEBUG

**Output**





**Hands On 9: Delete a country based on code**   
**Country.java**

package com.example.country.entity;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
  
@Entity  
public class Country {  
 @Id  
 private String code;  
 private String name;  
  
 *// Constructors* public Country() {}  
 public Country(String code, String name) {  
 this.code = code;  
 this.name = 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 + "]";  
 }  
}

**CountryRepository.java**

package com.example.country.repository;  
  
import com.example.country.entity.Country;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface CountryRepository extends JpaRepository<Country, String> {  
}

**CountryService.java**

package com.example.country.service;  
  
import com.example.country.entity.Country;  
import com.example.country.repository.CountryRepository;  
import com.example.country.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 with code: " + countryCode);  
 }  
 return result.get();  
 }  
 @Transactional  
 public void addCountry(Country country) {  
 countryRepository.save(country);  
 }

@Transactional  
public void updateCountry(String code, String newName) throws CountryNotFoundException {  
 Country country = countryRepository.findById(code)  
 .orElseThrow(() -> new CountryNotFoundException("Country not found for code " + code));  
 country.setName(newName);  
 countryRepository.save(country);  
}

@Transactional  
public void deleteCountry(String code) throws CountryNotFoundException {  
 if (!countryRepository.existsById(code)) {  
 throw new CountryNotFoundException("Country with code " + code + " does not exist");  
 }  
 countryRepository.deleteById(code);  
}

}

**CountryApplication.java**

package com.example.country;  
  
import com.example.country.entity.Country;  
import com.example.country.repository.CountryRepository;  
import com.example.country.service.exception.CountryNotFoundException;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import com.example.country.service.CountryService;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
@SpringBootApplication  
public class CountryApplication implements CommandLineRunner {  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(CountryApplication.class);  
  
 @Autowired  
 private CountryRepository countryRepository;  
 @Autowired  
 private CountryService countryService;  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(CountryApplication.class, args);  
 }  
  
 @Override  
 public void run(String... args) {  
 testDeleteCountry();  
 }  
  
 private void testDeleteCountry() {  
 *LOGGER*.info("Start testDeleteCountry");  
  
 try {  
 countryService.deleteCountry("ZZ");  
 *LOGGER*.debug("Country with code ZZ deleted.");  
 } catch (CountryNotFoundException e) {  
 *LOGGER*.error("Delete failed: {}", e.getMessage());  
 }  
  
 *LOGGER*.info("End testDeleteCountry");  
 }  
  
}

**application.properties**

spring.h2.console.enabled=true  
spring.h2.console.path=/h2-console  
  
spring.datasource.url=jdbc:h2:mem:testdb  
spring.datasource.driverClassName=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=  
  
spring.jpa.show-sql=true  
spring.jpa.hibernate.ddl-auto=create  
spring.datasource.initialization-mode=always  
  
logging.level.com.example.country=DEBUG

**Output**

