Objectives

- Explain the need and benefit of ORM ○ ORM (Object-Relational Mapping), makes it easier to develop code that interacts with database, abstracts the database system, transactionality
    - ORM Pros and Cons - https://blog.bitsrc.io/what-is-an-orm-and-why-youshould-use-it-b2b6f75f5e2a
    - What is ORM? - https://en.wikipedia.org/wiki/Object-relational_mapping

- Demonstrate the need and benefit of Spring Data JPA ○ Evolution of ORM solutions, Hibernate XML Configuration, Hibernate Annotation Configuration, Spring Data JPA, Hibernate benefits, open source, light weight, database independent query
    - With H2 in memory database - https://www.mkyong.com/spring-boot/springboot-spring-data-jpa/
    - With MySQL - https://www.mkyong.com/spring-boot/spring-boot-spring-datajpa-mysql-example/
    - XML Configuration Example https://www.tutorialspoint.com/hibernate/hibernate_examples.htm
    - Hibernate Configuration Example https://www.tutorialspoint.com/hibernate/hibernate_annotations.htm

- Explain about core objects of hibernate framework ○ Session Factory, Session, Transaction Factory, Transaction, Connection Provider
    - Hibernate Architecture Reference - https://www.tutorialspoint.com/hibernate/hibernate_architecture.htm

- Explain ORM implementation with Hibernate XML Configuration and Annotation Configuration ○ XML Configuration - persistence class, mapping xml, configuration xml, loading hibernate configuration xml file; Annotation Configuration - persistence class, @Entity, @Table, @Id, @Column, hibernate configuration xml file Loading hibernate configuration and interacting with database get the session factory, open session, begin transaction, commit transaction, close session
    - XML Configuration Example - https://www.tutorialspoint.com/hibernate/hibernate_examples.htm
    - Hibernate Configuration Example - https://www.tutorialspoint.com/hibernate/hibernate_annotations.htm

- Explain the difference between Java Persistence API, Hibernate and Spring Data JPA ○ JPA (Java Persistence API), JPA is a specification (JSR 338), JPA does not have implementation, Hibernate is one of the implementation for JPA, Hibernate is a ORM

tool, Spring Data JPA is an abstraction above Hibernate to remove boiler plate code when persisting data using Hibernate.

- Difference between Spring Data JPA and Hibernate - https://dzone.com/articles/what-is-the-difference-between-hibernate-andsprin-1
- Intro to JPA - https://www.javaworld.com/article/3379043/what-is-jpaintroduction-to-the-java-persistence-api.html

- Demonstrate implementation of DML using Spring Data JPA on a single database table ○ Hibernate log configuration and ddl-auto configuration, JpaRepsitory.findById(), defining Query Methods, JpaRespository.save(), JpaRepository.deleteById()
  - Spring Data JPA Ref Repository methods - https://docs.spring.io/springdata/jpa/docs/2.2.0.RELEASE/reference/html/#repositories.core-concepts
  - Query methods - https://docs.spring.io/springdata/jpa/docs/2.2.0.RELEASE/reference/html/#repositories.query-methods

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 Hibernate"
- 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 displaying 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
# 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
```

- Build the project using 'mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.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;
 import
com.cognizant.ormlearn.model.Country;

@Repository public interface CountryRepository extends
JpaRepository<Country, String> {

}
```

Service Class - com.cognizant.orm-learn.service.CountryService

- Create new package com.cognizant.orm-learn.service
- Create new class CountryService
- Include @Service annotation at class level
- Autowire CountryRepository in CountryService
- Include new method getAllCountries() method that returns a list of countries.
- Include @Transactional annotation for this method
- In getAllCountries() method invoke countryRepository.findAll() method and return the result

Testing in OrmLearnApplication.java

- Include a static reference to CountryService in OrmLearnApplication class

```
    private static CountryService countryService;
```

- Define a test method to get all countries from service.

```
    private static void testGetAllCountries() {
        LOGGER.info("Start");
        List<Country> countries = countryService.getAllCountries();
        LOGGER.debug("countries={}", countries);
        LOGGER.info("End");
    }
```

- Modify SpringApplication.run() invocation to set the application context and the CountryService reference from the application context.

```
        ApplicationContext context = SpringApplication.run(OrmLearnApplication.
class, args);         countryService = context.getBean(CountryService.class);
testGetAllCountries();
```

- Execute main method to check if data from ormlearn database is retrieved.

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() ○ session.get() ○ session.delete()

Hibernate Annotation Config implementation walk through

SME to provide explanation on the sample Hibernate implementation available in the link below:
https://www.tutorialspoint.com/hibernate/hibernate_annotations.htm

Explanation Topics

- Explain how object to relational database mapping done in persistence class file Employee
- Explain about following aspects of implementing the end to end operations in Hibernate:
    - @Entity ○ @Table ○ @Id ○ @GeneratedValue ○ @Column
    - Hibernate Configuration (hibernate.cfg.xml)
        - Dialect
        - Driver
        - Connection URL
        - Username
        - Password

Hands on 3

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

Hands on 4

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.beginTransaction();          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);
    }
```

Reference Links: https://dzone.com/articles/what-is-the-difference-between-hibernate-and-sprin-1
https://www.javaworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html

Hands on 5

Implement services for managing Country

An application requires for features to be implemented with regards to country. These features needs to be supported by implementing them as service using Spring Data JPA.

- Find a country based on country code
- Add new country
- Update country
- Delete country
- Find list of countries matching a partial country name

Before starting the implementation of the above features, there are few configuration and data population that needs to be incorporated. Please refer each topic below and implement the same.

Explanation for Hibernate table creation configuration

- Moreover the ddl-auto defines how hibernate behaves if a specific table or column is not present in the database.
    - create - drops existing tables data and structure, then creates new tables
    - validate - check if the table and columns exist or not, throws an exception if a matching table or column is not found
    - update - if a table does not exists, it creates a new table; if a column does not exists, it creates a new column
    - create-drop - creates the table, once all operations are completed, the table is dropped

```
# Hibernate ddl auto (create, create-drop, update, validate)

spring.jpa.hibernate.ddl-auto=validate
```

Populate country table

- Delete all the records in Country table and then use the below script to create the actual list of all countries in our world.

```
insert into country (co_code, co_name) values ("AF", "Afghanistan");
insert into country (co_code, co_name) values ("AL", "Albania"); insert
into country (co_code, co_name) values ("DZ", "Algeria"); insert into
country (co_code, co_name) values ("AS", "American Samoa");
```

```
insert into country (co_code, co_name) values ("AD", "Andorra"); insert into
country (co_code, co_name) values ("AO", "Angola"); insert into country
(co_code, co_name) values ("AI", "Anguilla"); insert into country (co_code,
co_name) values ("AQ", "Antarctica"); insert into country (co_code, co_name)
values ("AG", "Antigua and Barbuda");
insert into country (co_code, co_name) values ("AR", "Argentina");
insert into country (co_code, co_name) values ("AM", "Armenia");
insert into country (co_code, co_name) values ("AW", "Aruba"); insert
into country (co_code, co_name) values ("AU", "Australia"); insert
into country (co_code, co_name) values ("AT", "Austria"); insert into
country (co_code, co_name) values ("AZ", "Azerbaijan"); insert into
country (co_code, co_name) values ("BS", "Bahamas"); insert into
country (co_code, co_name) values ("BH", "Bahrain"); insert into
country (co_code, co_name) values ("BD", "Bangladesh"); insert into
country (co_code, co_name) values ("BB", "Barbados"); insert into
country (co_code, co_name) values ("BY", "Belarus"); insert into
country (co_code, co_name) values ("BE", "Belgium"); insert into
country (co_code, co_name) values ("BZ", "Belize"); insert into
country (co_code, co_name) values ("BJ", "Benin"); insert into
country (co_code, co_name) values ("BM", "Bermuda"); insert into
country (co_code, co_name) values ("BT", "Bhutan");
insert into country (co_code, co_name) values ("BO", "Bolivia, Plurinational
State of");
insert into country (co_code, co_name) values ("BQ", "Bonaire, Sint Eustatius
and Saba");
insert into country (co_code, co_name) values ("BA", "Bosnia and Herzegovina");
insert into country (co_code, co_name) values ("BW", "Botswana"); insert
into country (co_code, co_name) values ("BV", "Bouvet Island"); insert
into country (co_code, co_name) values ("BR", "Brazil");
insert into country (co_code, co_name) values ("IO", "British Indian Ocean
Territory");
insert into country (co_code, co_name) values ("BN", "Brunei Darussalam");
insert into country (co_code, co_name) values ("BG", "Bulgaria"); insert
into country (co_code, co_name) values ("BF", "Burkina Faso"); insert into
country (co_code, co_name) values ("BI", "Burundi"); insert into country
(co_code, co_name) values ("KH", "Cambodia"); insert into country
(co_code, co_name) values ("CM", "Cameroon"); insert into country
(co_code, co_name) values ("CA", "Canada"); insert into country (co_code,
co_name) values ("CV", "Cape Verde"); insert into country (co_code,
co_name) values ("KY", "Cayman Islands"); insert into country (co_code,
co_name) values ("CF", "Central African Republic");
insert into country (co_code, co_name) values ("TD", "Chad"); insert
into country (co_code, co_name) values ("CL", "Chile"); insert into
country (co_code, co_name) values ("CN", "China");
insert into country (co_code, co_name) values ("CX", "Christmas Island"); insert
into country (co_code, co_name) values ("CC", "Cocos (Keeling) Islands");
insert into country (co_code, co_name) values ("CO", "Colombia");
insert into country (co_code, co_name) values ("KM", "Comoros"); insert
into country (co_code, co_name) values ("CG", "Congo");
insert into country (co_code, co_name) values ("CD", "Congo, the Democratic
Republic of the");
insert into country (co_code, co_name) values ("CK", "Cook Islands"); insert
into country (co_code, co_name) values ("CR", "Costa Rica");
```

```
insert into country (co_code, co_name) values ("HR", "Croatia"); insert
into country (co_code, co_name) values ("CU", "Cuba"); insert into country
(co_code, co_name) values ("CW", "Curaçao"); insert into country (co_code,
co_name) values ("CY", "Cyprus"); insert into country (co_code, co_name)
values ("CZ", "Czech Republic"); insert into country (co_code, co_name)
values ("CI", "Côte d'Ivoire"); insert into country (co_code, co_name)
values ("DK", "Denmark"); insert into country (co_code, co_name) values
("DJ", "Djibouti"); insert into country (co_code, co_name) values ("DM",
"Dominica"); insert into country (co_code, co_name) values ("DO",
"Dominican Republic");
insert into country (co_code, co_name) values ("EC", "Ecuador"); insert
into country (co_code, co_name) values ("EG", "Egypt"); insert into
country (co_code, co_name) values ("SV", "El Salvador"); insert into
country (co_code, co_name) values ("GQ", "Equatorial Guinea");
insert into country (co_code, co_name) values ("ER", "Eritrea"); insert
into country (co_code, co_name) values ("EE", "Estonia"); insert into
country (co_code, co_name) values ("ET", "Ethiopia"); insert into
country (co_code, co_name) values ("FK", "Falkland Islands
(Malvinas)");
insert into country (co_code, co_name) values ("FO", "Faroe Islands");
insert into country (co_code, co_name) values ("FJ", "Fiji"); insert into
country (co_code, co_name) values ("FI", "Finland"); insert into country
(co_code, co_name) values ("FR", "France"); insert into country (co_code,
co_name) values ("GF", "French Guiana"); insert into country (co_code,
co_name) values ("PF", "French Polynesia"); insert into country (co_code,
co_name) values ("TF", "French Southern Territories");
insert into country (co_code, co_name) values ("GA", "Gabon"); insert
into country (co_code, co_name) values ("GM", "Gambia"); insert into
country (co_code, co_name) values ("GE", "Georgia"); insert into
country (co_code, co_name) values ("DE", "Germany"); insert into
country (co_code, co_name) values ("GH", "Ghana"); insert into country
(co_code, co_name) values ("GI", "Gibraltar"); insert into country
(co_code, co_name) values ("GR", "Greece"); insert into country
(co_code, co_name) values ("GL", "Greenland"); insert into country
(co_code, co_name) values ("GD", "Grenada"); insert into country
(co_code, co_name) values ("GP", "Guadeloupe"); insert into country
(co_code, co_name) values ("GU", "Guam"); insert into country
(co_code, co_name) values ("GT", "Guatemala"); insert into country
(co_code, co_name) values ("GG", "Guernsey"); insert into country
(co_code, co_name) values ("GN", "Guinea"); insert into country
(co_code, co_name) values ("GW", "Guinea-Bissau"); insert into country
(co_code, co_name) values ("GY", "Guyana"); insert into country
(co_code, co_name) values ("HT", "Haiti");
insert into country (co_code, co_name) values ("HM", "Heard Island and McDonald
Islands");
insert into country (co_code, co_name) values ("VA", "Holy See (Vatican City
State)");
insert into country (co_code, co_name) values ("HN", "Honduras");
insert into country (co_code, co_name) values ("HK", "Hong Kong");
insert into country (co_code, co_name) values ("HU", "Hungary");
insert into country (co_code, co_name) values ("IS", "Iceland");
insert into country (co_code, co_name) values ("IN", "India"); insert
into country (co_code, co_name) values ("ID", "Indonesia");
```

```
insert into country (co_code, co_name) values ("IR", "Iran, Islamic Republic
of");
insert into country (co_code, co_name) values ("IQ", "Iraq"); insert
into country (co_code, co_name) values ("IE", "Ireland"); insert
into country (co_code, co_name) values ("IM", "Isle of Man"); insert
into country (co_code, co_name) values ("IL", "Israel"); insert into
country (co_code, co_name) values ("IT", "Italy"); insert into
country (co_code, co_name) values ("JM", "Jamaica"); insert into
country (co_code, co_name) values ("JP", "Japan"); insert into
country (co_code, co_name) values ("JE", "Jersey"); insert into
country (co_code, co_name) values ("JO", "Jordan"); insert into
country (co_code, co_name) values ("KZ", "Kazakhstan"); insert into
country (co_code, co_name) values ("KE", "Kenya"); insert into
country (co_code, co_name) values ("KI", "Kiribati");
insert into country (co_code, co_name) values ("KP", "Democratic People's
Republic of Korea");
insert into country (co_code, co_name) values ("KR", "Republic of Korea");
insert into country (co_code, co_name) values ("KW", "Kuwait"); insert
into country (co_code, co_name) values ("KG", "Kyrgyzstan");
insert into country (co_code, co_name) values ("LA", "Lao People's Democratic
Republic");
insert into country (co_code, co_name) values ("LV", "Latvia"); insert
into country (co_code, co_name) values ("LB", "Lebanon"); insert into
country (co_code, co_name) values ("LS", "Lesotho"); insert into
country (co_code, co_name) values ("LR", "Liberia"); insert into
country (co_code, co_name) values ("LY", "Libya"); insert into country
(co_code, co_name) values ("LI", "Liechtenstein"); insert into country
(co_code, co_name) values ("LT", "Lithuania"); insert into country
(co_code, co_name) values ("LU", "Luxembourg"); insert into country
(co_code, co_name) values ("MO", "Macao");
insert into country (co_code, co_name) values ("MK", "Macedonia, the Former
Yugoslav Republic of");
insert into country (co_code, co_name) values ("MG", "Madagascar");
insert into country (co_code, co_name) values ("MW", "Malawi");
insert into country (co_code, co_name) values ("MY", "Malaysia");
insert into country (co_code, co_name) values ("MV", "Maldives");
insert into country (co_code, co_name) values ("ML", "Mali"); insert
into country (co_code, co_name) values ("MT", "Malta");
insert into country (co_code, co_name) values ("MH", "Marshall Islands");
insert into country (co_code, co_name) values ("MQ", "Martinique");
insert into country (co_code, co_name) values ("MR", "Mauritania");
insert into country (co_code, co_name) values ("MU", "Mauritius"); insert
into country (co_code, co_name) values ("YT", "Mayotte"); insert into
country (co_code, co_name) values ("MX", "Mexico");
insert into country (co_code, co_name) values ("FM", "Micronesia, Federated
States of");
insert into country (co_code, co_name) values ("MD", "Moldova, Republic of");
insert into country (co_code, co_name) values ("MC", "Monaco"); insert into
country (co_code, co_name) values ("MN", "Mongolia"); insert into country
(co_code, co_name) values ("ME", "Montenegro"); insert into country (co_code,
co_name) values ("MS", "Montserrat"); insert into country (co_code, co_name)
values ("MA", "Morocco"); insert into country (co_code, co_name) values
("MZ", "Mozambique"); insert into country (co_code, co_name) values ("MM",
"Myanmar"); insert into country (co_code, co_name) values ("NA", "Namibia");
```

```
insert into country (co_code, co_name) values ("NR", "Nauru"); insert
into country (co_code, co_name) values ("NP", "Nepal"); insert into
country (co_code, co_name) values ("NL", "Netherlands"); insert into
country (co_code, co_name) values ("NC", "New Caledonia"); insert into
country (co_code, co_name) values ("NZ", "New Zealand"); insert into
country (co_code, co_name) values ("NI", "Nicaragua"); insert into
country (co_code, co_name) values ("NE", "Niger"); insert into country
(co_code, co_name) values ("NG", "Nigeria"); insert into country
(co_code, co_name) values ("NU", "Niue"); insert into country (co_code,
co_name) values ("NF", "Norfolk Island"); insert into country (co_code,
co_name) values ("MP", "Northern Mariana Islands");
insert into country (co_code, co_name) values ("NO", "Norway");
insert into country (co_code, co_name) values ("OM", "Oman"); insert
into country (co_code, co_name) values ("PK", "Pakistan"); insert
into country (co_code, co_name) values ("PW", "Palau");
insert into country (co_code, co_name) values ("PS", "Palestine, State of");
insert into country (co_code, co_name) values ("PA", "Panama"); insert into
country (co_code, co_name) values ("PG", "Papua New Guinea"); insert into
country (co_code, co_name) values ("PY", "Paraguay"); insert into country
(co_code, co_name) values ("PE", "Peru"); insert into country (co_code,
co_name) values ("PH", "Philippines"); insert into country (co_code,
co_name) values ("PN", "Pitcairn"); insert into country (co_code, co_name)
values ("PL", "Poland"); insert into country (co_code, co_name) values
("PT", "Portugal"); insert into country (co_code, co_name) values ("PR",
"Puerto Rico"); insert into country (co_code, co_name) values ("QA",
"Qatar"); insert into country (co_code, co_name) values ("RO", "Romania");
insert into country (co_code, co_name) values ("RU", "Russian Federation");
insert into country (co_code, co_name) values ("RW", "Rwanda"); insert into
country (co_code, co_name) values ("RE", "Réunion"); insert into country
(co_code, co_name) values ("BL", "Saint Barthélemy"); insert into country
(co_code, co_name) values ("SH", "Saint Helena, Ascension and Tristan da
Cunha");
insert into country (co_code, co_name) values ("KN", "Saint Kitts and Nevis");
insert into country (co_code, co_name) values ("LC", "Saint Lucia"); insert
into country (co_code, co_name) values ("MF", "Saint Martin (French
part)");
insert into country (co_code, co_name) values ("PM", "Saint Pierre and
Miquelon");
insert into country (co_code, co_name) values ("VC", "Saint Vincent and the
Grenadines");
insert into country (co_code, co_name) values ("WS", "Samoa"); insert
into country (co_code, co_name) values ("SM", "San Marino");
insert into country (co_code, co_name) values ("ST", "Sao Tome and Principe");
insert into country (co_code, co_name) values ("SA", "Saudi Arabia"); insert
into country (co_code, co_name) values ("SN", "Senegal"); insert into country
(co_code, co_name) values ("RS", "Serbia"); insert into country (co_code,
co_name) values ("SC", "Seychelles"); insert into country (co_code, co_name)
values ("SL", "Sierra Leone"); insert into country (co_code, co_name) values
("SG", "Singapore"); insert into country (co_code, co_name) values ("SX",
"Sint Maarten (Dutch
part)");
insert into country (co_code, co_name) values ("SK", "Slovakia"); insert
into country (co_code, co_name) values ("SI", "Slovenia");
```

```
insert into country (co_code, co_name) values ("SB", "Solomon Islands");
insert into country (co_code, co_name) values ("SO", "Somalia"); insert
into country (co_code, co_name) values ("ZA", "South Africa"); insert into
country (co_code, co_name) values ("GS", "South Georgia and the South
Sandwich Islands");
insert into country (co_code, co_name) values ("SS", "South Sudan");
insert into country (co_code, co_name) values ("ES", "Spain"); insert
into country (co_code, co_name) values ("LK", "Sri Lanka"); insert
into country (co_code, co_name) values ("SD", "Sudan"); insert into
country (co_code, co_name) values ("SR", "Suriname");
insert into country (co_code, co_name) values ("SJ", "Svalbard and Jan Mayen");
insert into country (co_code, co_name) values ("SZ", "Swaziland"); insert
into country (co_code, co_name) values ("SE", "Sweden"); insert into country
(co_code, co_name) values ("CH", "Switzerland"); insert into country
(co_code, co_name) values ("SY", "Syrian Arab Republic"); insert into country
(co_code, co_name) values ("TW", "Taiwan, Province of
China");
insert into country (co_code, co_name) values ("TJ", "Tajikistan");
insert into country (co_code, co_name) values ("TZ", "Tanzania, United Republic
of");
insert into country (co_code, co_name) values ("TH", "Thailand");
insert into country (co_code, co_name) values ("TL", "Timor-Leste");
insert into country (co_code, co_name) values ("TG", "Togo"); insert
into country (co_code, co_name) values ("TK", "Tokelau"); insert
into country (co_code, co_name) values ("TO", "Tonga");
insert into country (co_code, co_name) values ("TT", "Trinidad and Tobago");
insert into country (co_code, co_name) values ("TN", "Tunisia");
insert into country (co_code, co_name) values ("TR", "Turkey"); insert
into country (co_code, co_name) values ("TM", "Turkmenistan"); insert
into country (co_code, co_name) values ("TC", "Turks and Caicos
Islands");
insert into country (co_code, co_name) values ("TV", "Tuvalu"); insert
into country (co_code, co_name) values ("UG", "Uganda"); insert into
country (co_code, co_name) values ("UA", "Ukraine");
insert into country (co_code, co_name) values ("AE", "United Arab Emirates");
insert into country (co_code, co_name) values ("GB", "United Kingdom");
insert into country (co_code, co_name) values ("US", "United States"); insert
into country (co_code, co_name) values ("UM", "United States Minor Outlying
Islands");
insert into country (co_code, co_name) values ("UY", "Uruguay"); insert
into country (co_code, co_name) values ("UZ", "Uzbekistan"); insert
into country (co_code, co_name) values ("VU", "Vanuatu");
insert into country (co_code, co_name) values ("VE", "Venezuela, Bolivarian
Republic of");
insert into country (co_code, co_name) values ("VN", "Viet Nam"); insert
into country (co_code, co_name) values ("VG", "Virgin Islands,
British");
insert into country (co_code, co_name) values ("VI", "Virgin Islands, U.S.");
insert into country (co_code, co_name) values ("WF", "Wallis and Futuna");
insert into country (co_code, co_name) values ("EH", "Western Sahara");
insert into country (co_code, co_name) values ("YE", "Yemen"); insert into
country (co_code, co_name) values ("ZM", "Zambia"); insert into country
(co_code, co_name) values ("ZW", "Zimbabwe"); insert into country (co_code,
co_name) values ("AX", "Åland Islands");
```

Refer subsequent hands on exercises to implement the features related to country.

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:
    - Method signature

```
@Transactional
public Country findCountryByCode(String countryCode) throws CountryNotFoundException
```

- 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.

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:
  - Create new instance of country with a new code and name ○ Call countryService.addCountry() passing the country created in the previous step. ○ Invoke countryService.findCountryByCode() passing the same code used when adding a new country
  - Check in the database if the country is added

Hands on 8

Update a country based on code

- Create a new method updateCountry() in CountryService with parameters code and name. Annotate this method with @Transactional. Implement following steps in this method.
  - Get the reference of the country using findById() method in repository
  - In the country reference obtained, update the name of country using setter method
  - Call countryRepository.save() method to update the name
- Include new test method in OrmLearnApplication, which invokes updateCountry() method in CountryService passing a country's code and different name for the country.
- Check in database table if name is modified.

Hands on 9

Delete a country based on code

- Create new method deleteCountry() in CountryService. Annotate this method with @Transactional.
- In deleteCountry() method call deleteById() method of repository.
- Include new test method in OrmLearnApplication with following steps
  - Call the delete method based on the country code during the add country hands on
- Check in database if the country is deleted

OrmLearnApplication.java

```
package com.cognizant.ormlearn;


import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;
```

Hands on 10

```
import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import java.util.List;


@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();

    }
```

Hands on 11

```
    private static void testGetAllCountries() {

        LOGGER.info("Start");

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

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

        LOGGER.info("End");

    }

}
```

Country.java

```
package com.cognizant.ormlearn.model;



import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;
```

Hands on 12

```
@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; }
```

Hands on 13

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


    @Override
    public String toString() {
        return "Country [code=" + code + ", name=" + name + "]";
    }
}
```

Application.propoerties

```
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
logging.level.org.hibernate.SQL=debug
```

Hands on 14

spring.datasource.url=jdbc:mysql://localhost:3308/orm_learn

spring.datasource.username=root

spring.datasource.password=12345

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

spring.jpa.hibernate.ddl-auto=validate

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

spring.jpa.hibernate.ddl-auto=validate

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

Hands on 15

OUTPUT:

Hands on 16

```
<terminated> OrmLearnApplication (1) [Java Application] C:\Users\ADMIN\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe  (04-Jul-2025, 6:13:07 pm – 6:13:29 pm elapsed: 0:00:21.465) [pid: 9236]

 :: Spring Boot ::                (v3.3.13)

04-07-25 18:13:19.499 restartedMain        INFO c.c.o.OrmLearnApplication             logStarting   50 Starting OrmLearnApplication using Java 21.0.7 with PID 9236 (C:\Users\ADMIN\Downlc
04-07-25 18:13:19.506 restartedMain       DEBUG c.c.o.OrmLearnApplication             logStarting   51 Running with Spring Boot v3.3.13, Spring v6.1.21
04-07-25 18:13:19.507 restartedMain        INFO c.c.o.OrmLearnApplication    logStartupProfileInfo  654 No active profile set, falling back to 1 default profile: "default"
04-07-25 18:13:19.638 restartedMain        INFO ertyDefaultsPostProcessor                 logTo  252 Devtools property defaults active! Set 'spring.devtools.add-properties' to 'false'
04-07-25 18:13:21.349 restartedMain        INFO toryConfigurationDelegate   registerRepositoriesIn  147 Bootstrapping Spring Data JPA repositories in DEFAULT mode.
04-07-25 18:13:21.485 restartedMain        INFO toryConfigurationDelegate   registerRepositoriesIn  215 Finished Spring Data repository scanning in 108 ms. Found 1 JPA repository interfac
04-07-25 18:13:22.872 restartedMain        INFO o.h.j.i.util.LogHelper   logPersistenceUnitInformation   31 HHH000204: Processing PersistenceUnitInfo [name: default]
04-07-25 18:13:23.133 restartedMain        INFO org.hibernate.Version                  logVersion   44 HHH000412: Hibernate ORM core version 6.5.3.Final
04-07-25 18:13:23.270 restartedMain        INFO .i.RegionFactoryInitiator         initiateService   50 HHH000026: Second-level cache disabled
04-07-25 18:13:24.038 restartedMain        INFO SpringPersistenceUnitInfo          addTransformer   87 No LoadTimeWeaver setup: ignoring JPA class transformer
04-07-25 18:13:24.081 restartedMain        INFO c.z.h.HikariDataSource             getConnection  109 HikariPool-1 - Starting...
04-07-25 18:13:24.851 restartedMain        INFO c.z.h.pool.HikariPool              checkFailFast  554 HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@414312d6
04-07-25 18:13:24.858 restartedMain        INFO c.z.h.HikariDataSource             getConnection  122 HikariPool-1 - Start completed.
04-07-25 18:13:25.051 restartedMain        WARN o.h.orm.deprecation              constructDialect  153 HHH90000025: MySQLDialect does not need to be specified explicitly using 'hibernate
04-07-25 18:13:25.139 restartedMain       DEBUG h.t.d.s.s.DdlTypeRegistry           addDescriptor   64 addDescriptor(12, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlTy
04-07-25 18:13:25.141 restartedMain       DEBUG h.t.d.s.s.DdlTypeRegistry           addDescriptor   64 addDescriptor(-9, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlTy
04-07-25 18:13:25.142 restartedMain       DEBUG h.t.d.s.s.DdlTypeRegistry           addDescriptor   64 addDescriptor(-3, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlTy
04-07-25 18:13:25.143 restartedMain       DEBUG h.t.d.s.s.DdlTypeRegistry           addDescriptor   64 addDescriptor(4003, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@1604a1b9
04-07-25 18:13:25.148 restartedMain       DEBUG h.t.d.s.s.DdlTypeRegistry           addDescriptor   64 addDescriptor(4001, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@1f36d355
04-07-25 18:13:25.150 restartedMain       DEBUG h.t.d.s.s.DdlTypeRegistry           addDescriptor   64 addDescriptor(4002, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@45c2effb
04-07-25 18:13:25.152 restartedMain       DEBUG h.t.d.s.s.DdlTypeRegistry           addDescriptor   64 addDescriptor(2004, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdl
04-07-25 18:13:25.154 restartedMain       DEBUG h.t.d.s.s.DdlTypeRegistry           addDescriptor   64 addDescriptor(2005, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdl
04-07-25 18:13:25.158 restartedMain       DEBUG h.t.d.s.s.DdlTypeRegistry           addDescriptor   64 addDescriptor(2011, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdl
04-07-25 18:13:26.708 restartedMain        INFO .p.i.JtaPlatformInitiator         initiateService   59 HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to e
04-07-25 18:13:26.791 restartedMain        INFO rEntityManagerFactoryBean buildNativeEntityManagerFactory  437 Initialized JPA EntityManagerFactory for persistence unit 'default'
04-07-25 18:13:28.238 restartedMain        INFO .OptionalLiveReloadServer            startServer   59 LiveReload server is running on port 35729
04-07-25 18:13:28.379 restartedMain        INFO c.c.o.OrmLearnApplication             logStarted   56 Started OrmLearnApplication in 10.305 seconds (process running for 11.612)
04-07-25 18:13:28.404 restartedMain        INFO c.c.o.OrmLearnApplication      testGetAllCountries   24 Start
04-07-25 18:13:28.829 restartedMain       DEBUG org.hibernate.SQL                    logStatement  135 select c1_0.co_code,c1_0.co_name from country c1_0
04-07-25 18:13:28.889 restartedMain       DEBUG c.c.o.OrmLearnApplication      testGetAllCountries   26 countries=[Country [code=IN, name=India], Country [code=US, name=United States]]
04-07-25 18:13:28.890 restartedMain        INFO c.c.o.OrmLearnApplication      testGetAllCountries   27 End
04-07-25 18:13:28.906 licationShutdownHook  INFO rEntityManagerFactoryBean                destroy  650 Closing JPA EntityManagerFactory for persistence unit 'default'
04-07-25 18:13:28.917 licationShutdownHook  INFO c.z.h.HikariDataSource                    close  349 HikariPool-1 - Shutdown initiated...
04-07-25 18:13:28.942 licationShutdownHook  INFO c.z.h.HikariDataSource                    close  351 HikariPool-1 - Shutdown completed.
```

Hands on 17