Spring Data JPA with Spring Boot, Hibernate

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 %

# 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(OrmLearnApplica
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
    a. Walkthrough all the configuration defined in XML file
    b. 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.

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

import jakarta.persistence.*;

@Entity
@Table(name = "country")
public class Country {

    @Id
    @Column(name = "co_code")
    private String code;

    @Column(name = "co_name")
    private String name;

    public String getCode() { return code; }
    public void setCode(String code) { this.code = code; }

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

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

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.

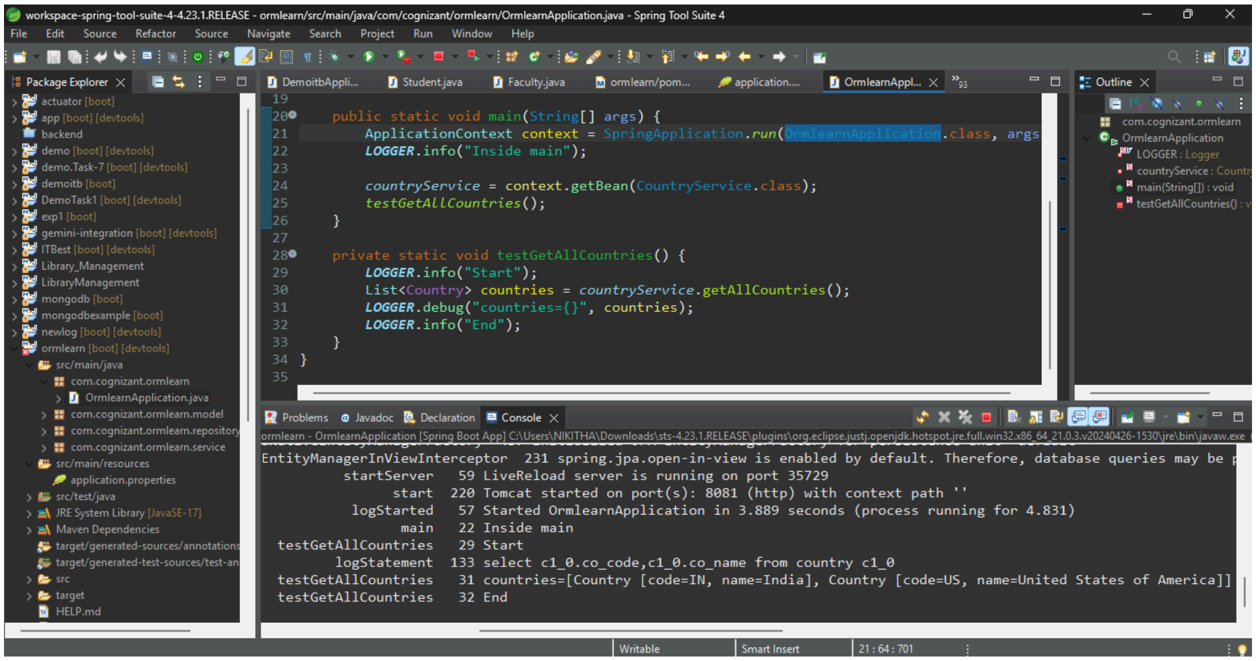
```
private static void testGetAllCountries() {
    LOGGER.info("Start");
    List<Country> countries = countryService.getAllCountries();
    LOGGER.debug("countries={}", countries);
    LOGGER.info("End");
}
```

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

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

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

Output:


workspace-spring-tool-suite-4-4.23.1.RELEASE - ormlearn/src/main/java/com/cognizant/ormlearn/OrmlearnApplication.java - Spring Tool Suite 4

```java
    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");
    }
}
```

ormlearn - OrmlearnApplication [Spring Boot App] C:\Users\NIKITHA\Downloads\sts-4.23.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe

```
EntityManagerInViewInterceptor  231 spring.jpa.open-in-view is enabled by default. Therefore, database queries may be
          startServer   59 LiveReload server is running on port 35729
                start  220 Tomcat started on port(s): 8081 (http) with context path ''
            logStarted   57 Started OrmlearnApplication in 3.889 seconds (process running for 4.831)
                 main   22 Inside main
  testGetAllCountries   29 Start
         logStatement  133 select c1_0.co_code,c1_0.co_name from country c1_0
  testGetAllCountries   31 countries=[Country [code=IN, name=India], Country [code=US, name=United States of America]]
  testGetAllCountries   32 End
```

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

application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/coursedb
spring.datasource.username=root
spring.datasource.password=YourPasswordHere

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
server.port=8081
```

Course Entity

```
package com.example.course.entity;

import jakarta.persistence.*;

@Entity
public class Course {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;
    private String author;
    private int price;

    // Constructors
    public Course() {}

    public Course(String name, String author, int price) {
        this.name = name;
        this.author = author;
        this.price = price;
    }
    // Getters & Setters
}
```

Course Repository

```
package com.example.course.repository;

import com.example.course.entity.Course;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CourseRepository extends JpaRepository<Course, Integer> {
}
```

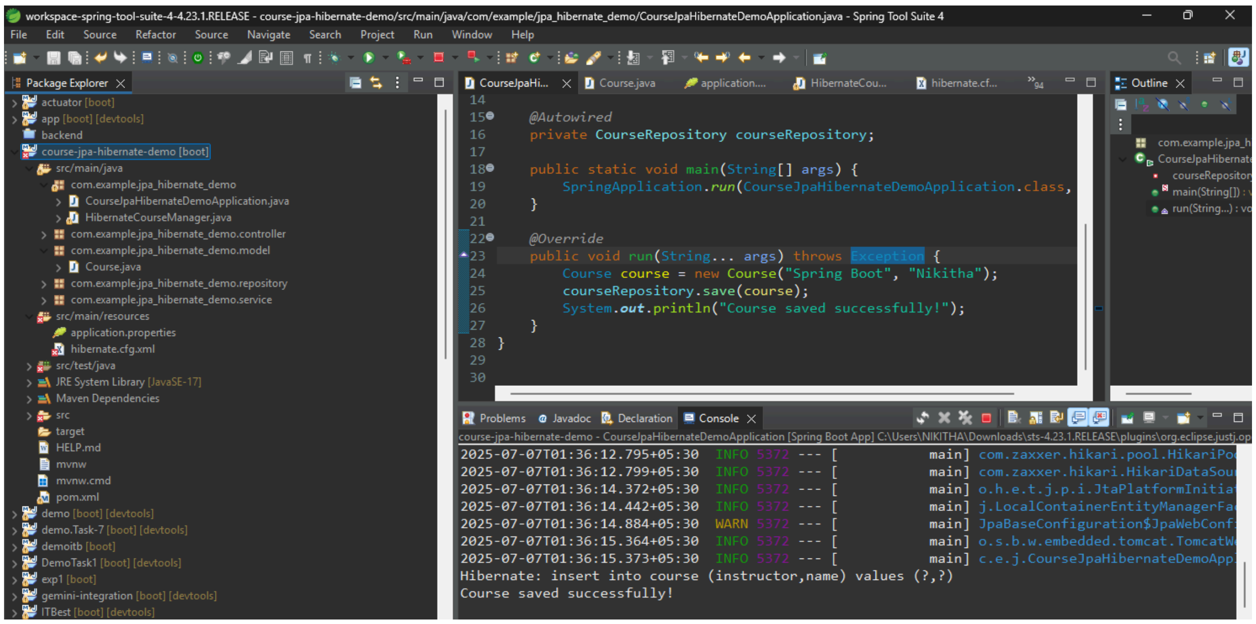
Main Class with Runner

```
package com.example;

import com.example.course.entity.Course;
import com.example.course.repository.CourseRepository;
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 CourseJpaHibernateDemoApplication implements CommandLine
Runner {

    @Autowired
    private CourseRepository courseRepository;

    public static void main(String[] args) {
        SpringApplication.run(CourseJpaHibernateDemoApplication.class, args);
    }
    @Override
    public void run(String... args) throws Exception {
        Course course = new Course("Spring Boot", "Nikitha");
        courseRepository.save(course);
        System.out.println("Course saved.");
    }
}
```

pom.xml Snippet

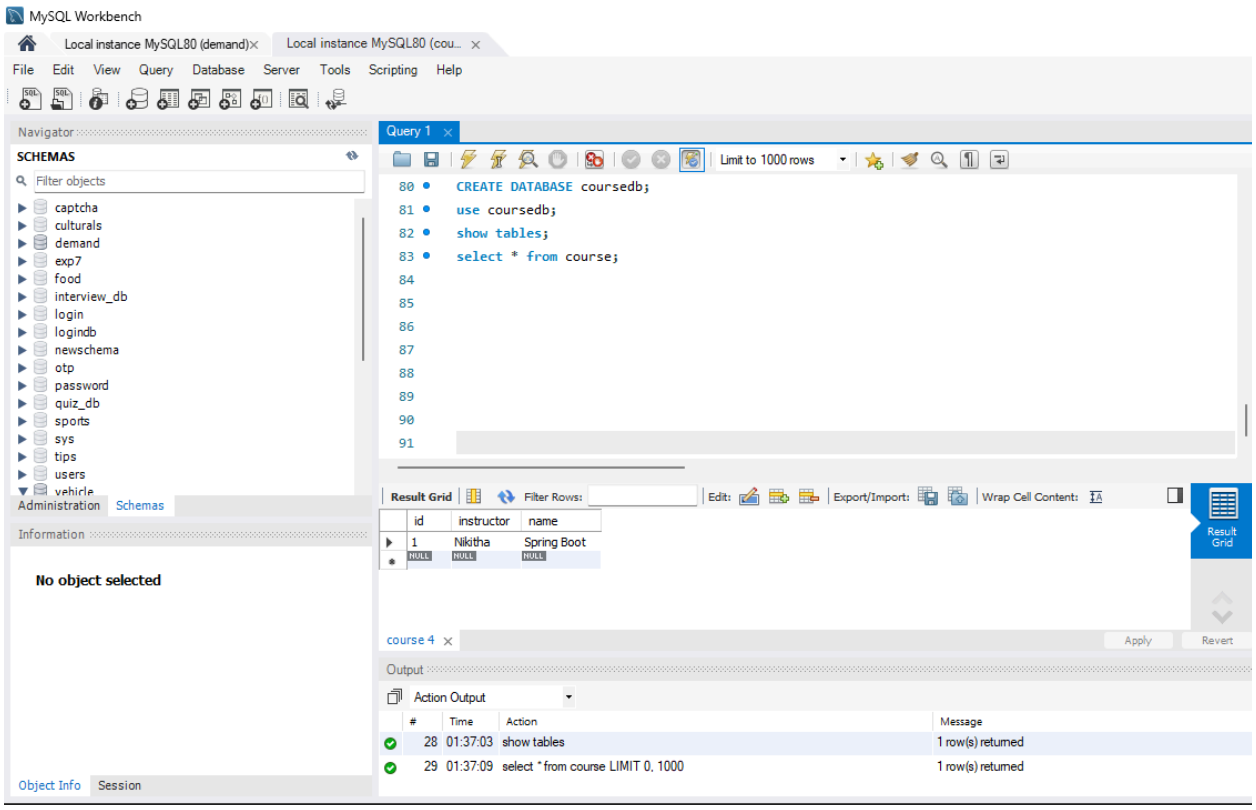
```xml
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

Output:



```java
    @Autowired
    private CourseRepository courseRepository;

    public static void main(String[] args) {
        SpringApplication.run(CourseJpaHibernateDemoApplication.class,
    }

    @Override
    public void run(String... args) throws Exception {
        Course course = new Course("Spring Boot", "Nikitha");
        courseRepository.save(course);
        System.out.println("Course saved successfully!");
    }
}
```

```
course-jpa-hibernate-demo - CourseJpaHibernateDemoApplication [Spring Boot App] C:\Users\NIKITHA\Downloads\sts-4.23.1.RELEASE\plugins\org.eclipse.justj.op
2025-07-07T01:36:12.795+05:30  INFO 5372 --- [           main] com.zaxxer.hikari.pool.HikariPo
2025-07-07T01:36:12.799+05:30  INFO 5372 --- [           main] com.zaxxer.hikari.HikariDataSou
2025-07-07T01:36:14.372+05:30  INFO 5372 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitia
2025-07-07T01:36:14.442+05:30  INFO 5372 --- [           main] j.LocalContainerEntityManagerFa
2025-07-07T01:36:14.884+05:30  WARN 5372 --- [           main] JpaBaseConfiguration$JpaWebConf
2025-07-07T01:36:15.364+05:30  INFO 5372 --- [           main] o.s.b.w.embedded.tomcat.TomcatWe
2025-07-07T01:36:15.373+05:30  INFO 5372 --- [           main] c.e.j.CourseJpaHibernateDemoApp
Hibernate: insert into course (instructor,name) values (?,?)
Course saved successfully!
```



```sql
CREATE DATABASE coursedb;
use coursedb;
show tables;
select * from course;
```

id	instructor	name
1	Nikitha	Spring Boot
NULL	NULL	NULL

#	Time	Action	Message
28	01:37:03	show tables	1 row(s) returned
29	01:37:09	select * from course LIMIT 0, 1000	1 row(s) returned