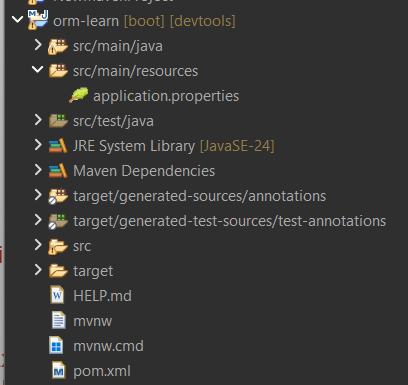
**Hands on 1**

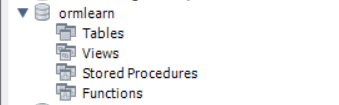
**Spring Data JPA - Quick Example** 

**Create a Eclipse Project using Spring Initializr**

1. Go to <https://start.spring.io/>
2. Change Group as “com.cognizant”
3. Change Artifact Id as “orm-learn”
4. In Options > Description enter "Demo project for Spring Data JPA and Hibernate"
5. Click on menu and select "Spring Boot DevTools", "Spring Data JPA" and "MySQL Driver"
6. Click Generate and download the project as zip
7. Extract the zip in root folder to Eclipse Workspace
8. Import the project in Eclipse "File > Import > Maven > Existing Maven Projects > Click Browse and select extracted folder > Finish"



1. Create a new schema "ormlearn" in MySQL database. Execute the following commands to open MySQL client and create schema.



In orm-learn Eclipse project, open src/main/resources/application.properties and include the below database and log configuration.

spring.application.name=orm-learn

# 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 %-25.25logger{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=Ishita@sql1.

# Hibernate configuration

spring.jpa.hibernate.ddl-auto=update

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

1. 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
2. Include logs for verifying if main() method is called.

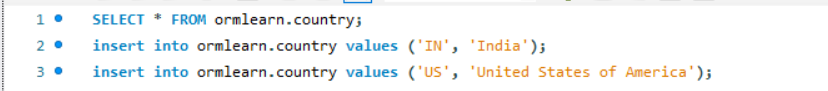


SME to walk through the following aspects related to the project created:

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**

1. 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.



**Persistence Class - com.cognizant.orm-learn.model.Country**

1. Open Eclipse with orm-learn project
2. Create new package com.cognizant.orm-learn.model
3. Create Country.java, then generate getters, setters and toString() methods.
4. Include @Entity and @Table at class level
5. Include @Column annotations in each getter method specifying the column name.
6. package com.cognizant.orm\_learn.model;
7. import jakarta.persistence.Column;
8. import jakarta.persistence.Entity;
9. import jakarta.persistence.Id;
10. import jakarta.persistence.Table;
11. *@Entity*
12. *@Table*(name="country")
13. public class Country {
15. *@Id*
16. *@Column*(name="co\_code")
17. private String code;
18. *@Column*(name="co\_name")
19. private String name;

22. public String getCode() {
23. return code;
24. }
25. public String getName() {
26. return name;
27. }
29. public void setCode(String code) {
30. this.code = code;
31. }
32. public void setName(String name) {
33. this.name = name;
34. }
35. *@Override*
36. public String toString() {
37. return "Country [code=" + code + ", name=" + name + "]";
38. }




44. }

**Repository Class - com.cognizant.orm-learn.CountryRepository**

1. Create new package com.cognizant.orm-learn.repository
2. Create new interface named CountryRepository that extends JpaRepository<Country, String>
3. Define @Repository annotation at class level
4. package com.cognizant.orm\_learn.repository;
5. import org.springframework.data.jpa.repository.JpaRepository;
6. import org.springframework.stereotype.Repository;
7. import com.cognizant.orm\_learn.model.Country;
8. *@Repository*
9. public interface CountryRepository extends JpaRepository<Country, String>{
10. }

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

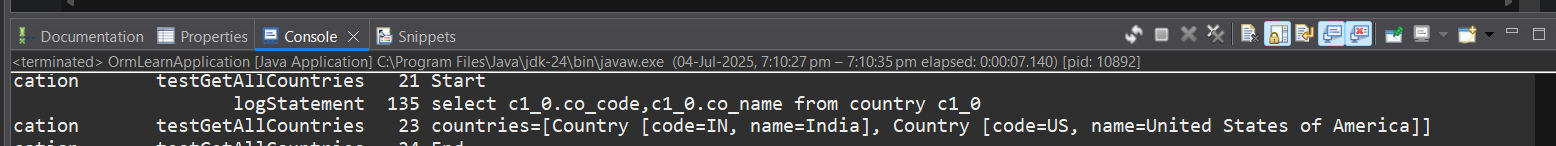
1. Create new package com.cognizant.orm-learn.service
2. Create new class CountryService
3. Include @Service annotation at class level
4. Autowire CountryRepository in CountryService
5. Include new method getAllCountries() method that returns a list of countries.
6. Include @Transactional annotation for this method
7. In getAllCountries() method invoke countryRepository.findAll() method and return the result
8. package com.cognizant.orm\_learn.service;
9. import java.util.List;
10. import org.springframework.beans.factory.annotation.Autowired;
11. import org.springframework.stereotype.Service;
12. import com.cognizant.orm\_learn.model.Country;
13. import com.cognizant.orm\_learn.repository.CountryRepository;
14. import jakarta.transaction.Transactional;
15. *@Service*
16. public class CountryService {
17. *@Autowired*
18. private CountryRepository countryRepository;
20. *@Transactional*
21. public List<Country> getAllCountries() {
23. return countryRepository.findAll();
24. }


28. }

**Testing in OrmLearnApplication.java**

* Include a static reference to CountryService in OrmLearnApplication class
* Define a test method to get all countries from service.
* Modify SpringApplication.run() invocation to set the application context and the CountryService reference from the application context.
* Execute main method to check if data from ormlearn database is retrieved.
* package com.cognizant.orm\_learn;
* import com.cognizant.orm\_learn.model.Country;
* import com.cognizant.orm\_learn.service.CountryService;
* import java.util.List;
* import org.slf4j.Logger;
* import org.slf4j.LoggerFactory;
* import org.springframework.boot.SpringApplication;
* import org.springframework.boot.autoconfigure.SpringBootApplication;
* import org.springframework.context.ApplicationContext;
* *@SpringBootApplication*
* public class OrmLearnApplication {
* private static CountryService *countryService*;
* private static void testGetAllCountries()
* {
* ***LOGGER***.info("Start");
* List<Country> countries=*countryService*.getAllCountries();
* ***LOGGER***.debug("countries={}",countries);
* ***LOGGER***.info("End");
* }
* private static final Logger ***LOGGER*** = LoggerFactory.*getLogger*(OrmLearnApplication.class);
* OrmLearnApplication(CountryService countryService) {
* this.*countryService* = countryService;
* }
* public static void main(String[] args) {
* ApplicationContext context=SpringApplication.*run*(OrmLearnApplication.class, args);
* *countryService*=context.getBean(CountryService.class);
* *testGetAllCountries*();
* SpringApplication.*run*(OrmLearnApplication.class, args);
* ***LOGGER***.info("Inside main");
* }
* }

**Output**



**Difference between JPA, Hibernate and Spring Data JPA**   
  
Java Persistence API (JPA)

1. JSR 338 Specification for persisting, reading and managing data from Java objects
2. Does not contain concrete implementation of the specification
3. Hibernate is one of the implementation of JPA

Hibernate

1. ORM Tool that implements JPA

Spring Data JPA

1. Does not have JPA implementation, but reduces boiler plate code
2. This is another level of abstraction over JPA implementation provider like Hibernate
3. Manages transactions

**Refer code snippets below on how the code compares between Hibernate and Spring Data JPA  
Hibernate**

**Employee class**

package com.cognizant.orm\_learn.model;

import java.time.LocalDate;

import java.util.Date;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

import jakarta.persistence.Temporal;

import jakarta.persistence.TemporalType;

*@Entity*

*@Table*(name="employee")

public class Employee {

*@Id*

*@GeneratedValue*(strategy = *GenerationType*.***IDENTITY***)

private int id;

private String name;

private double salary;

private boolean permanent;

*@Column*(name="date\_of\_birth")

private LocalDate dateOfBirth;

public int getId() {

return id;

}

public String getName() {

return name;

}

public double getSalary() {

return salary;

}

public boolean isPermanent() {

return permanent;

}

public LocalDate getDateOfBirth() {

return dateOfBirth;

}

public void setId(int id) {

this.id = id;

}

public void setName(String name) {

this.name = name;

}

public void setSalary(double salary) {

this.salary = salary;

}

public void setPermanent(boolean permanent) {

this.permanent = permanent;

}

public void setDateOfBirth(LocalDate localDate) {

this.dateOfBirth = dateOfBirth;

}

*@Override*

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", salary=" + salary + ", permanent=" + permanent

+ ", dateOfBirth=" + dateOfBirth + "]";

}

}

**EmployeeRepository class**

package com.cognizant.orm\_learn.repository;

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

import org.springframework.stereotype.Repository;

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

*@Repository*

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService class**

package com.cognizant.orm\_learn.service;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

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

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

*@Service*

public class EmployeeService {

*@Autowired*

private EmployeeRepository employeeRepository;

*@Transactional*

public void addEmployee(Employee employee)

{

employeeRepository.save(employee);

}

*@Transactional*(readOnly = true)

public Employee getEmployee(int id)

{

return employeeRepository.findById(id).orElse(null);

}

}

**HibernateEmployeeService Class**

package com.cognizant.orm\_learn.service;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

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

import org.springframework.stereotype.Service;

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

import jakarta.persistence.EntityManagerFactory;

*@Service*

public class HibernateEmployeeService {

private final SessionFactory sessionFactory;

*@Autowired*

public HibernateEmployeeService(EntityManagerFactory emf) {

this.sessionFactory = emf.unwrap(SessionFactory.class);

}

public Integer addEmployee(Employee employee) {

Session session = sessionFactory.openSession(); // Open session from factory

Transaction tx = null;

Integer id = null;

try {

tx = session.beginTransaction();

id = (Integer) session.save(employee);

tx.commit();

} catch (Exception e) {

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

e.printStackTrace();

} finally {

session.close();

}

return id;

}

}

**OrmLearnApplication class**

package com.cognizant.orm\_learn;

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

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

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

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

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

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

import jakarta.annotation.PostConstruct;

import java.time.LocalDate;

import java.util.List;

import java.util.Optional;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class OrmLearnApplication {

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

*@Autowired*

private CountryService countryService;

*@Autowired*

private EmployeeRepository employeeRepository;

*@Autowired*

private EmployeeService employeeService;

*@Autowired*

private HibernateEmployeeService hibernateEmployeeService;

// This method will run automatically after the Spring context is initialized

*@PostConstruct*

public void init() {

testGetAllCountries();

testGetEmployee();

testAddEmployeeJpa();

testAddEmployeeHibernate();

}

private void testGetAllCountries() {

***LOGGER***.info("Start");

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

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

***LOGGER***.info("End");

}

private void testGetEmployee() {

***LOGGER***.info("Start");

Optional<Employee> employee = employeeRepository.findById(1);

if (employee.isPresent()) {

***LOGGER***.debug("Employee: {}", employee.get());

} else {

***LOGGER***.warn("Employee not found");

}

***LOGGER***.info("End");

}

private void testAddEmployeeJpa() {

***LOGGER***.info("Start - Add Employee (JPA)");

Employee emp = new Employee();

emp.setName("Jane Doe");

emp.setPermanent(true);

emp.setSalary(75000.0);

emp.setDateOfBirth(LocalDate.*of*(1992, 8, 20));

employeeService.addEmployee(emp);

***LOGGER***.debug("Saved employee with JPA: {}", emp);

***LOGGER***.info("End - Add Employee (JPA)");

}

private void testAddEmployeeHibernate() {

***LOGGER***.info("Start - Add Employee (Hibernate)");

Employee employee = new Employee();

employee.setName("Hib Emp");

employee.setSalary(50000);

employee.setPermanent(false);

employee.setDateOfBirth(LocalDate.*of*(1992, 2, 20));

int id = hibernateEmployeeService.addEmployee(employee);

***LOGGER***.debug("Saved employee with Hibernate: {}", employee);

***LOGGER***.info("End - Add Employee (Hibernate)");

}

// this method is needed to run the application

public static void main(String[] args) {

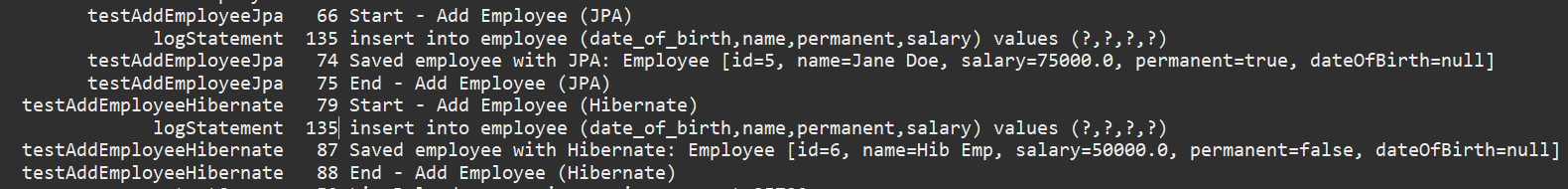
SpringApplication.*run*(OrmLearnApplication.class, args);

***LOGGER***.info("Inside main");

}

}

**Output**



**Explaination**

Comparison: Hibernate vs Spring Data JPA

1. Session Management

Hibernate (Manual):

Session session = sessionFactory.openSession();

Transaction tx = session.beginTransaction();

session.save(employee);

tx.commit();

session.close();

- You manage the session manually.

- You must explicitly begin and commit or roll back transactions.

Spring Data JPA (Automatic):

employeeRepository.save(employee);

- Spring handles session and transaction management internally.

2. Repository / DAO Layer

Hibernate:

- No repository interface.

- Data access logic is written inside a service class.

Spring Data JPA:

- Uses a repository interface:

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

- Spring Boot auto-implements the repository.

3. Configuration and Setup

Hibernate:

sessionFactory = entityManagerFactory.unwrap(SessionFactory.class);

- Manual setup of SessionFactory from EntityManagerFactory.

Spring Data JPA:

@Autowired

private EmployeeRepository employeeRepository;

- Configuration is automatic via Spring Boot.

4. Transaction Handling

Hibernate:

Transaction tx = session.beginTransaction();

try {

session.save(employee);

tx.commit();

} catch (Exception e) {

tx.rollback();

}

- Manual transaction control.

Spring Data JPA:

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

- @Transactional automatically manages transactions.