Week 3 – SpringDataJPAWithSpringBoot_Hibernate_Mandatory_HandsOn
Skill: Spring Data JPA with Spring Boot, Hibernate
Candidate: Projita Kar
Superset ID: 6407705
Type: Mandatory Hands-On

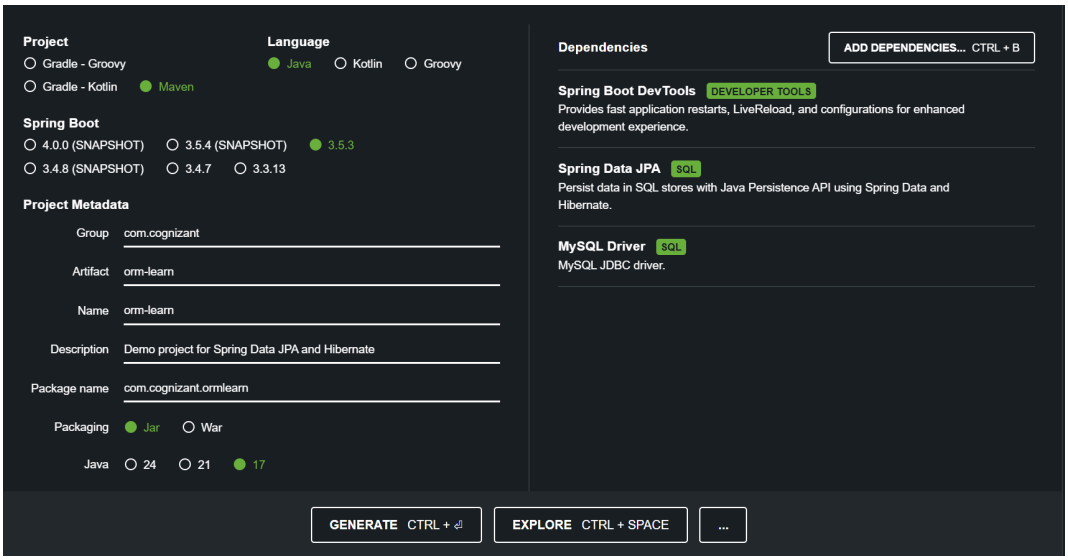
Hands-on 1 : Spring Data JPASpring Data JPA - Quick Example

Step 1: Software Setup

- Mention installed versions:
    - MySQL Server 8.0
    - MySQL Workbench 8
    - Eclipse IDE for Enterprise Java Developers
    - Maven 3.9.10

Step 2: Project Creation via Spring Initializr

- Group: com.cognizant
- Artifact: orm-learn
- Description: "Demo project for Spring Data JPA and Hibernate"
- Dependencies:
    - Spring Boot DevTools
    - Spring Data JPA
    - MySQL Driver
- Screenshot:
    - Spring Initializr settings before generating zip



Step 3: Schema and Properties Configuration

- **SQL command:**
  `MySQL  localhost:33060+ ssl  SQL >` `CREATE SCHEMA ormlearn;`
  `Query OK, 1 row affected (0.0140 sec)`

- application.properties should include:

```
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=********
# Hibernate configuration
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
```

Step 4: Build the Project with Maven

Command:

```
 mvn clean package -Dhttp.proxyHost=proxy.cognizant.com
-Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com
-Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456
```

```
PS C:\Users\proji\eclipse-workspace\orm-learn\orm-learn> mvn clean package -DskipTests
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.cognizant:orm-learn >-----------------------
[INFO] Building orm-learn 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- clean:3.4.1:clean (default-clean) @ orm-learn ---
[INFO] Deleting C:\Users\proji\eclipse-workspace\orm-learn\orm-learn\target
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ orm-learn ---
[INFO] Copying 1 resource from src\main\resources to target\classes
[INFO] Copying 0 resource from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.14.0:compile (default-compile) @ orm-learn ---
[INFO] Recompiling the module because of changed source code.
```

```
Downloaded from central: https://repo.maven.apache.org/maven2/org/jdom/jdom2/2.0.6.
Downloaded from central: https://repo.maven.apache.org/maven2/org/vafer/jdependency
[INFO] Replacing main artifact C:\Users\proji\eclipse-workspace\orm-learn\orm-learn
ed dependencies in BOOT-INF/.
[INFO] The original artifact has been renamed to C:\Users\proji\eclipse-workspace\o
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  6.718 s
[INFO] Finished at: 2025-07-04T04:11:30+05:30
[INFO] ------------------------------------------------------------------------
```

- Include logs for verifying if main() method is called

```java
package com.cognizant.ormlearn;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class OrmLearnApplication {
  // Logger definition
  private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);
  public static void main(String[] args) {
    SpringApplication.run(OrmLearnApplication.class, args);
    LOGGER.info("Inside main"); // This log will confirm the main() is called
  }
}
```

```
04-07-25 15:59:37.685 restartedMain        DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(12, org.hibernate
04-07-25 15:59:37.686 restartedMain        DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(-9, org.hibernate
04-07-25 15:59:37.686 restartedMain        DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(-3, org.hibernate
04-07-25 15:59:37.686 restartedMain        DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(4003, org.hiberna
04-07-25 15:59:37.686 restartedMain        DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(4001, org.hiberna
04-07-25 15:59:37.686 restartedMain        DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(4002, org.hiberna
04-07-25 15:59:37.687 restartedMain        DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(2004, org.hiberna
04-07-25 15:59:37.687 restartedMain        DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(2005, org.hiberna
04-07-25 15:59:37.687 restartedMain        DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(2011, org.hiberna
04-07-25 15:59:38.422 restartedMain         INFO .p.i.JtaPlatformInitiator          initiateService   59 HHH000489: No JTA platform avai
04-07-25 15:59:38.479 restartedMain         INFO rEntityManagerFactoryBean buildNativeEntityManagerFactory  447 Initialized JPA EntityMan
04-07-25 15:59:38.859 restartedMain         INFO .OptionalLiveReloadServer              startServer   59 LiveReload server is running on
04-07-25 15:59:38.877 restartedMain         INFO c.c.o.OrmLearnApplication              logStarted   59 Started OrmLearnApplication in
04-07-25 15:59:38.886 restartedMain         INFO c.c.o.OrmLearnApplication                    main   16 Inside main
04-07-25 15:59:38.892 licationShutdownHook  INFO rEntityManagerFactoryBean                 destroy  660 Closing JPA EntityManagerFactor
04-07-25 15:59:38.896 licationShutdownHook  INFO c.z.h.HikariDataSource                      close  349 HikariPool-1 - Shutdown initiat
04-07-25 15:59:38.904 licationShutdownHook  INFO c.z.h.HikariDataSource                      close  351 HikariPool-1 - Shutdown complet
```

Step 5: Model Class Creation (Country.java)

- Package: com.cognizant.ormlearn.model
- Class with annotations: @Entity, @Table, @Id, @Column

```java
package com.cognizant.ormlearn.model;
import jakarta.persistence.Entity;
import jakarta.persistence.Table;
```

```java
import jakarta.persistence.Id;
import jakarta.persistence.Column;
@Entity
@Table(name = "country")
public class Country {
  @Id
  @Column(name = "co_code")
  private String code;
  @Column(name = "co_name")
  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;
  }
  // toString() method
  @Override
  public String toString() {
     return "Country [code=" + code + ", name=" + name + "]";
  }
}
```

Step 6: Repository Interface (CountryRepository)

- Package: com.cognizant.ormlearn.repository
- Interface extends JpaRepository<Country, String>
- Annotation: @Repository

```java
package com.cognizant.ormlearn.repository;
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> {
}
```

Step 7: Service Class (CountryService)

- Package: com.cognizant.ormlearn.service
- Annotation: @Service
- Autowire repository and use @Transactional
- Method: getAllCountries()

```java
package com.cognizant.ormlearn.service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.repository.CountryRepository;
@Service
public class CountryService {
  @Autowired
  private CountryRepository countryRepository;
  @Transactional
  public List<Country> getAllCountries() {
    return countryRepository.findAll();
  }
}
```

Step 8: Table and Data in MySQL

- 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.

```sql
CREATE TABLE country (
  co_code VARCHAR(2) PRIMARY KEY,
  co_name VARCHAR(50)
);

INSERT INTO country VALUES ('IN', 'India');
INSERT INTO country VALUES ('US', 'United States of America');
```

Step 9: Modify OrmLearnApplication.java

- Add LOGGER.info("Inside main")
- Setup ApplicationContext
- Get CountryService bean
- Call testGetAllCountries()
- Log output using logger

```java
package com.cognizant.ormlearn;
import java.util.List;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.service.CountryService;
@SpringBootApplication
public class OrmLearnApplication {
  private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);
  private static CountryService countryService;
  public static void main(String[] args) {
    ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);
    LOGGER.info("Inside main");
    countryService = context.getBean(CountryService.class);
    testGetAllCountries();
  }
  private static void testGetAllCountries() {
    LOGGER.info("Start");
    List<Country> countries = countryService.getAllCountries();
    LOGGER.debug("countries={}", countries);
    LOGGER.info("End");
  }
}
```

Step 10: Verify Output in Console

- Verify logs:
  - Main method entered
  - Start message
  - SQL select log
  - Countries printed
  - End message
- Screenshot:

```
DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(12, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@2d266557) rep
DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(-9, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@419abda7) rep
DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(-3, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@3ee160d2) rep
DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(4003, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@64281614) replaced previ
DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(4001, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@6c1eca49) replaced previ
DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(4002, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@6797e920) replaced previ
DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(2004, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@42809807) r
DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(2005, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@74f8f67) re
DEBUG h.t.d.s.s.DdlTypeRegistry            addDescriptor   64 addDescriptor(2011, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@2b9ba7a1) r
 INFO .p.i.JtaPlatformInitiator          initiateService   59 HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platfo
 INFO rEntityManagerFactoryBean buildNativeEntityManagerFactory  447 Initialized JPA EntityManagerFactory for persistence unit 'default'
 INFO .OptionalLiveReloadServer              startServer   59 LiveReload server is running on port 35729
 INFO c.c.o.OrmLearnApplication               logStarted   59 Started OrmLearnApplication in 3.654 seconds (process running for 4.03)
 INFO c.c.o.OrmLearnApplication                    main   24 Inside main
 INFO c.c.o.OrmLearnApplication      testGetAllCountries   31 Start
DEBUG org.hibernate.SQL                     logStatement  135 select c1_0.co_code,c1_0.co_name from country c1_0
DEBUG c.c.o.OrmLearnApplication      testGetAllCountries   33 countries=[Country [code=IN, name=India], Country [code=US, name=United States of America]]
 INFO c.c.o.OrmLearnApplication      testGetAllCountries   34 End
 INFO rEntityManagerFactoryBean                 destroy  660 Closing JPA EntityManagerFactory for persistence unit 'default'
 INFO c.z.h.HikariDataSource                      close  349 HikariPool-1 - Shutdown initiated...
 INFO c.z.h.HikariDataSource                      close  351 HikariPool-1 - Shutdown completed.
```

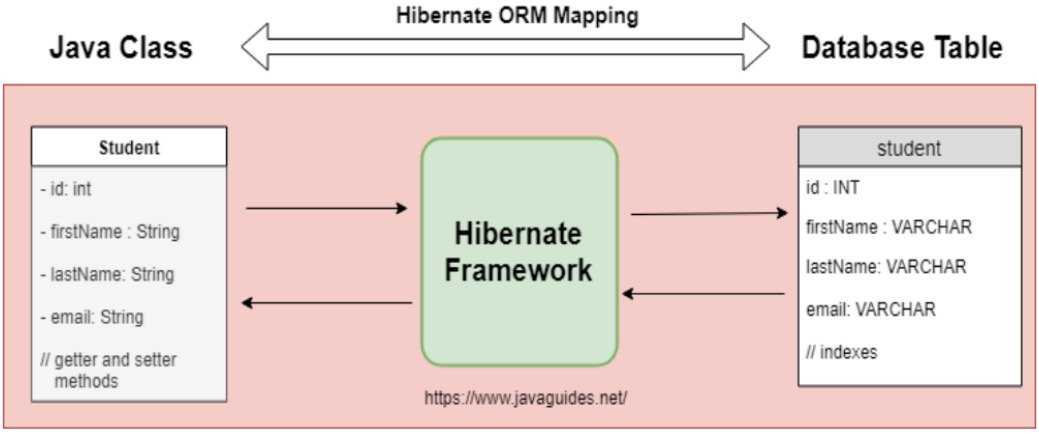
Hands-on 4: Difference between JPA, Hibernate and Spring Data JPA

Java Persistence API (JPA)

- JPA is a **Java specification** (JSR 338) for managing relational data in Java applications.
- It defines a set of interfaces and rules for persisting, reading, and managing data using Java objects.
- **JPA is not an implementation**; it only provides the blueprint.
- Example implementations of JPA: **Hibernate, EclipseLink, OpenJPA**.

Hibernate

- **Hibernate is an ORM (Object Relational Mapping) tool** and the most popular implementation of the JPA specification.
- It provides a concrete implementation of the interfaces defined by JPA.
- Hibernate manages database interactions via session objects.



Sample Hibernate Code:

```
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

- Spring Data JPA is a **Spring project** that adds an abstraction layer on top of JPA providers like Hibernate.
- It **simplifies the data access layer** by reducing boilerplate code and managing transactions automatically.
- It does **not implement JPA itself**, but **relies on a JPA provider** (e.g., Hibernate) under the hood.

Sample Spring Data JPA Code:

```
// EmployeeRepository.java
public interface EmployeeRepository extends JpaRepository<Employee,
Integer> {
}

// EmployeeService.java
@Autowired
private EmployeeRepository employeeRepository;

@Transactional
public void addEmployee(Employee employee) {
    employeeRepository.save(employee);
}
```

Summary Table

Aspect	JPA	Hibernate	Spring Data JPA
Type	Specification	Implementation	Abstraction Layer
Purpose	Define API for persistence	Provide ORM mapping	Simplify JPA usage

Boilerplate	Medium	High	Low
Transaction Management	Programmer-defined	Programmer-defined	Handled by Spring
Requires configuration	Yes	Yes	Minimal in Spring Boot

References:

- What is the difference between Hibernate and Spring Data JPA?