Spring Data JPA - Quick Example

1. Project Generation

• Used Spring Initializr with:

o Group: com.cognizant

o Artifact: orm-learn

o Dependencies:

 Spring Boot DevTools

 Spring Data JPA

 MySQL Driver

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. MySQL Setup

• MySQL version: 8.0

• Created schema:

sql

CopyEdit

CREATE SCHEMA ormlearn;

• Created table and inserted records:

sql

CopyEdit

CREATE TABLE country (

code VARCHAR(2) PRIMARY KEY,

name VARCHAR(50)

);

INSERT INTO country VALUES ('IN', 'India');

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Configuration in application.properties

properties

CopyEdit

# Logging

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace

# Log format

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n

# DB Config

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

spring.jpa.hibernate.ddl-auto=validate

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

Difference between JPA, Hibernate and Spring Data JPA

1. JPA (Java Persistence API)

• What it is: A specification (set of interfaces and annotations).

• Goal: Standardize ORM in Java across vendors.

• Does not do any real work — it needs a provider like Hibernate.

• Examples:

java

CopyEdit

@Entity

public class User {

@Id

private Long id;

private String name;

}

• Common interfaces: EntityManager, Query, EntityTransaction

• Provider examples: Hibernate, EclipseLink, OpenJPA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

📌 2. Hibernate

• What it is: A JPA implementation + more features.

• Extends JPA: Offers more flexibility than JPA (caching, filters, criteria API, etc.)

• Can be used with or without JPA.

• Vendor-specific features:

o @CreationTimestamp, @UpdateTimestamp

o Native queries

o Hibernate Session API

• When you use Hibernate directly, you might write:

java

CopyEdit

Session session = sessionFactory.openSession();

Transaction tx = session.beginTransaction();

session.save(user);

tx.commit();

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

📌 3. Spring Data JPA

• What it is: A Spring module that builds on JPA and Hibernate to:

o Reduce boilerplate

o Simplify database access

o Use repository interfaces instead of writing queries

• Key features:

o Interface-based repositories (JpaRepository)

o Auto-generated queries from method names

o Pagination, sorting

o Custom queries with @Query

• Example:

java

CopyEdit

public interface UserRepository extends JpaRepository<User, Long> {

List<User> findByName(String name);

}