Spring Data JPA and Hibernate - Hands-On Walkthrough

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

Project Setup via Spring Initializr:

1. Visit https://start.spring.io/
2. Group: com.cognizant
3. Artifact: orm-learn
4. Description: Demo project for Spring Data JPA and Hibernate
5. Dependencies: Spring Boot DevTools, Spring Data JPA, MySQL Driver
6. Click Generate and import into Eclipse via File > Import > Maven > Existing Maven Projects.
7. Create MySQL schema:

   > mysql -u root -p

   > create schema ormlearn;

application.properties Configuration:

- Logging levels for Spring and Hibernate
- MySQL connection config (URL, username, password)
- Hibernate dialect: MySQL5Dialect
- ddl-auto: validate
- Logging pattern setup

Running Project:

- Add SLF4J Logger to OrmLearnApplication.java
- Add log inside main() to verify application launch

Spring Data JPA and Hibernate - Hands-On Walkthrough

SME Walkthrough Topics:

1. src/main/java: Application code folder

2. src/main/resources: Application config folder

3. src/test/java: Testing code folder

4. OrmLearnApplication.java: Main method

5. @SpringBootApplication: Enables component scanning, config, and auto-configuration

6. pom.xml: Project dependencies and plugin setup

Database Table:

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');

Model Class (Country.java):

@Entity, @Table("country"), @Id for PK, @Column for field mapping

Repository Interface (CountryRepository.java):

Extends JpaRepository<Country, String>, annotated with @Repository

Service Layer (CountryService.java):

Annotated with @Service, uses @Autowired CountryRepository, defines @Transactional method getAllCountries()

Test Service in OrmLearnApplication.java:

- Autowire CountryService

- Define testGetAllCountries()

- Call it from main() using ApplicationContext.getBean()

Hands-on 2: Hibernate XML Configuration Walkthrough

Refer to: https://www.tutorialspoint.com/hibernate/hibernate_examples.htm

Spring Data JPA and Hibernate - Hands-On Walkthrough

Hibernate XML Mapping:

- Uses .hbm.xml files to define mapping between classes and DB tables.

- Example: Employee.hbm.xml includes <class>, <id>, and <property> tags for mapping.

Core Components in Hibernate:

1. SessionFactory: Factory for Session objects, created once during app startup

2. Session: Interface for interacting with database (similar to EntityManager)

3. Transaction: Used for atomic operations

Common Methods:

- beginTransaction(): Begins DB transaction

- commit(): Commits transaction to DB

- rollback(): Reverts changes if error occurs

- session.save(obj): Persists an object

- session.createQuery().list(): Executes HQL and retrieves list

- session.get(Class, id): Fetches object by PK

- session.delete(obj): Deletes object from DB

End-to-end Flow:

- Configure hibernate.cfg.xml

- Load SessionFactory

- Create Session

- Perform CRUD using HQL and transaction APIs