Spring Data JPA with Spring Boot, Hibernate

Spring Data JPA - Quick Example

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" ...>
	<modelVersion>4.0.0</modelVersion>
	<groupId>com.cognizant</groupId>
	<artifactId>orm-learn</artifactId>
	<version>1.0.0</version>
	<description>Demo project for Spring Data JPA and Hibernate</description>
	<properties>
		<java.version>1.8</java.version>
		<spring-boot.version>2.5.6</spring-boot.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>mysql</groupId>
			<artifactId>mysql-connector-java</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-devtools</artifactId>
			<scope>runtime</scope>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
```

```
            <artifactId>spring-boot-starter-logging</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

application.properties

```
# 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

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

# MySQL Database Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn
spring.datasource.username=root
```

```
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# Hibernate
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
```

sql

```
CREATE DATABASE IF NOT EXISTS ormlearn;
USE ormlearn;
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');
```

Country.java

```
package com.cognizant.ormlearn.model;
import javax.persistence.*;
@Entity
@Table(name = "country")
public class Country {
	@Id
	@Column(name = "co_code")
	private String code;
	@Column(name = "co_name")
	private String name;
	public Country() {}
	public Country(String code, String name) {
		this.code = code;
		this.name = 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 + "'}";
    }
}
```

CountryRepository.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> {
}
```

CountryService.java

```
package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.repository.CountryRepository;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import javax.transaction.Transactional;
import java.util.List;

@Service
public class CountryService {
	@Autowired
	private CountryRepository countryRepository;
	@Transactional
	public List<Country> getAllCountries() {
		return countryRepository.findAll();
	}
}
```

OrmLearnApplication.java (Main Class)

```
package com.cognizant.ormlearn;
import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.service.CountryService;
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 java.util.List;

@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();
        countries.forEach(country -> LOGGER.debug("Country: {}", country));
        LOGGER.info("End");
    }
    System.out.println("Starting OrmLearn Application...\n");
LOGGER.info("Inside main method.");
LOGGER.info("Getting list of countries from service...\n");
System.out.println("Constructor Injection: CountryService initialized.");
System.out.println("Repository Injection: CountryRepository injected.\n");
System.out.println("📘 Listing Countries:");
List<Country> countries = countryService.getAllCountries();
for (Country country : countries) {
    System.out.println(country);
}
System.out.println("\nDone.");
}
```

Output :

```
Starting OrmLearn Application...

INFO  : Inside main method.
INFO  : Getting list of countries from service...

Constructor Injection: CountryService initialized.
Repository Injection: CountryRepository injected.

📘 Listing Countries:
Country{code='IN', name='India'}
Country{code='US', name='United States of America'}

Done.
```

Difference between JPA, Hibernate and Spring Data JPA

Feature	JPA	Hibernate	Spring Data JPA
Type	Specification (API)	JPA Implementation and Standalone ORM	Framework/Abstraction over JPA
Provided by	Oracle / Java Community	Red Hat	Spring Framework
Implementation	No	Yes	Yes (wraps JPA provider like Hibernate)
Boilerplate Code	Required	Required	Minimal (Auto-generated methods)
Ease of Use	Medium	Medium	High
ORM Capability	Defined via annotations	Full ORM support	Simplified ORM via repositories
Query Support	JPQL	JPQL, HQL, Native SQL	Derived queries, JPQL, Native SQL
Caching Support	Not defined	Yes (First-level, Second-level)	Yes (uses Hibernate or other providers)
Entity Mapping	Yes	Yes	Yes (uses JPA annotations)
Spring Integration	No direct integration	Works with Spring	Full Spring ecosystem integration
Common Interfaces	EntityManager, @Entity, etc.	Session, Criteria, HQL, etc.	CrudRepository, JpaRepository, PagingAndSortingRepository
Use Case	Defines ORM standards	Implements JPA and adds extra ORM features	Simplifies database operations in Spring applications