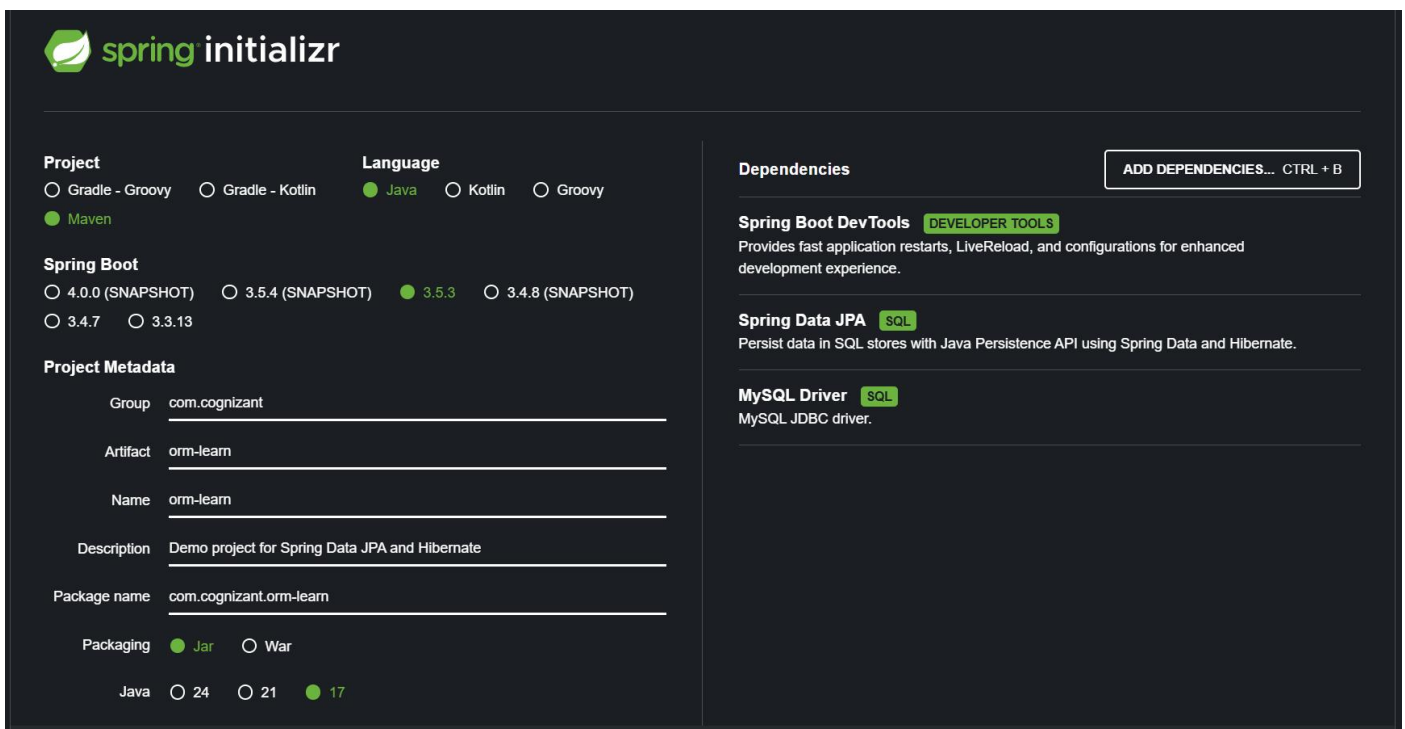


Spring Data JPA with Spring Boot, Hibernate

Hands on 1

Spring Data JPA - Quick Example

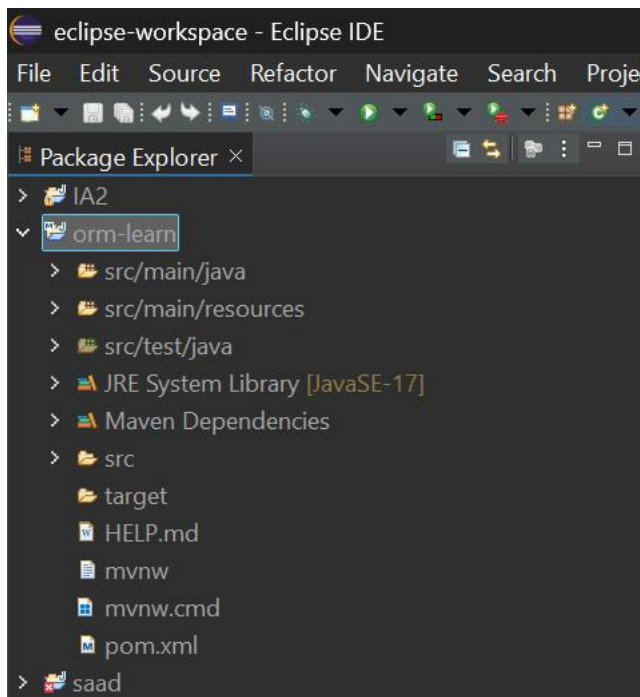
Step 1: Generate Spring Boot Project



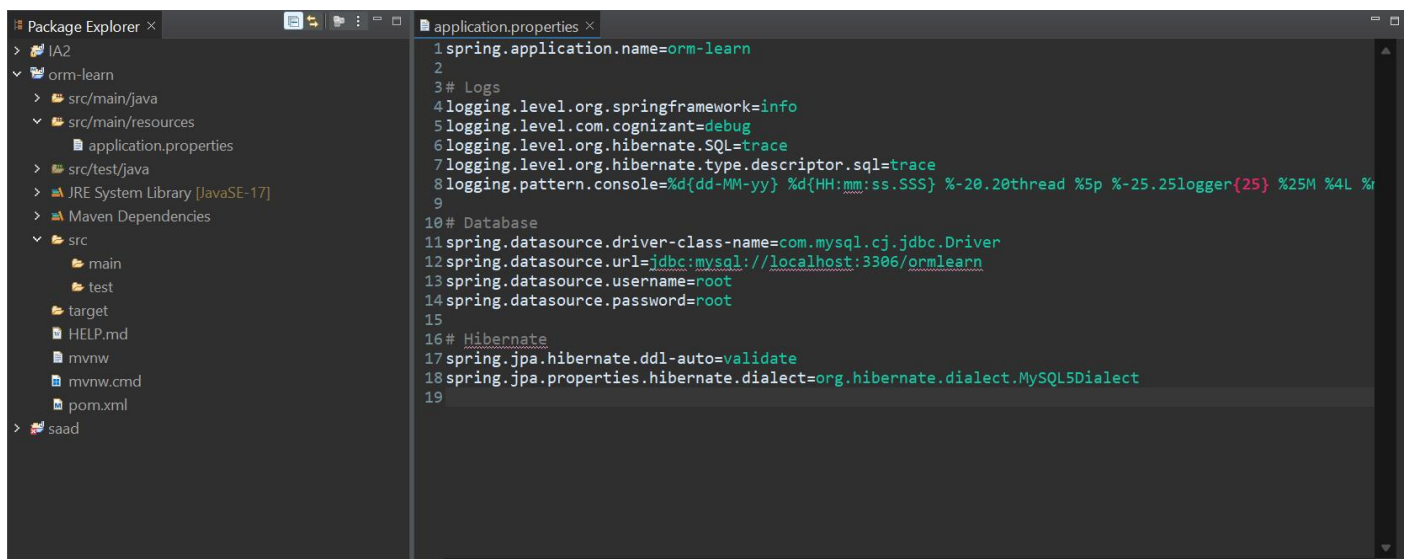
The image shows the Spring Initializr web application interface for configuring a new project. The interface is dark-themed and organized into several sections:

- Project:** Includes radio buttons for **Gradle - Groovy**, **Gradle - Kotlin**, **Maven** (selected), **Language** (with **Java** selected, and **Kotlin**, **Groovy** as options), and **Spring Boot** versions (4.0.0 (SNAPSHOT), 3.5.4 (SNAPSHOT), **3.5.3** (selected), 3.4.8 (SNAPSHOT), 3.4.7, 3.3.13).
- Project Metadata:** A form with fields for **Group** (com.cognizant), **Artifact** (orm-learn), **Name** (orm-learn), **Description** (Demo project for Spring Data JPA and Hibernate), **Package name** (com.cognizant.orm-learn), **Packaging** (**Jar** selected, War as an option), and **Java** version (24, 21, **17** selected).
- Dependencies:** A section on the right with a button **ADD DEPENDENCIES... CTRL + B**. It lists three dependencies:
 - Spring Boot DevTools** (tagged **DEVELOPER TOOLS**): Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
 - Spring Data JPA** (tagged **SQL**): Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
 - MySQL Driver** (tagged **SQL**): MySQL JDBC driver.

Step 2: Import the jar file in Eclipse



Step 3:



Step 4: Open Workbench and create schema, tables and add values

The screenshot shows the MySQL Workbench interface. The left sidebar contains the 'MANAGEMENT' and 'PERFORMANCE' sections. The main window displays a SQL query in the 'Query 1' tab. The query is as follows:

```
4 create table country (  
5     co_code varchar(2) primary key,  
6     co_name varchar(50)  
7 );  
8 insert into country values ('IN', 'India');  
9 insert into country values ('US', 'United States of America');  
10 select*from country;
```

The 'Result Grid' shows the output of the query:

co_code	co_name
IN	India
US	United States of America
NULL	NULL

The 'Output' section at the bottom shows the 'Action Output' table:

#	Time	Action	Message	Duration / Fetch
8	12:25:24	use ormlearn	0 row(s) affected	0.000 sec
9	12:25:24	show tables	0 row(s) returned	0.000 sec / 0.000 sec
10	12:25:24	create table country (co_code varchar(2) primary key, co_name varchar(50))	0 row(s) affected	0.031 sec
11	12:25:24	insert into country values ('IN', 'India')	1 row(s) affected	0.016 sec
12	12:25:24	insert into country values ('US', 'United States of America')	1 row(s) affected	0.000 sec
13	12:25:24	select*from country LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

Step 5: src/main/resources/application.properties

spring.application.name=orm-learn

Logs

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

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

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

Database

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

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

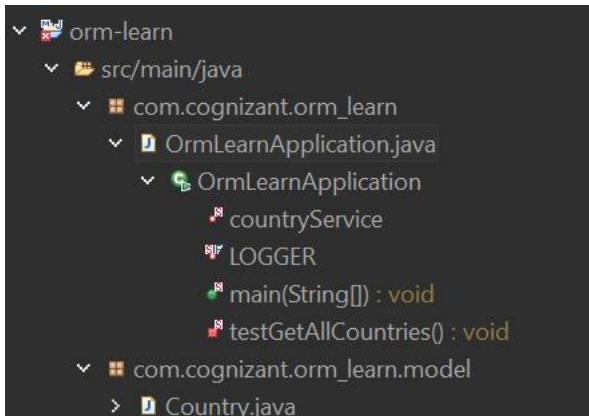
spring.datasource.username=root

```
spring.datasource.password=saadma11
```

Hibernate

```
spring.jpa.hibernate.ddl-auto=validate
```

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



Country.java

```
package com.cognizant.orm_learn.model;
```

```
import jakarta.persistence.*;
```

```
@Entity
```

```
@Table(name = "country")
```

```
public class Country {
```

```
    @Id
```

```
    @Column(name = "co_code")
```

```
    private String code;
```

```
    @Column(name = "co_name")
```

```
    private String 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 + "];"  
}  
}
```

POM file:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-  
instance"
```

```
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-  
4.0.0.xsd">
```

```
    <modelVersion>4.0.0</modelVersion>
```

```
    <parent>
```

```
        <groupId>org.springframework.boot</groupId>
```

```
        <artifactId>spring-boot-starter-parent</artifactId>
```

```
        <version>3.5.3</version>
```

```
        <relativePath/> <!-- lookup parent from repository -->
```

```
    </parent>
```

```
    <groupId>com.cognizant</groupId>
```

```
    <artifactId>orm-learn</artifactId>
```

```
    <version>0.0.1-SNAPSHOT</version>
```

```
    <name>orm-learn</name>
```

```
    <description>Demo project for Spring Data JPA and Hibernate</description>
```

```
    <url/>
```

```
    <licenses>
```

```
        <license/>
```

```
    </licenses>
```

```
    <developers>
```

```
        <developer/>
```

```
    </developers>
```

```
    <scm>
```

```

        <connection/>
        <developerConnection/>
    </tag/>
    <url/>
</scm>
<properties>
    <java.version>17</java.version>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>

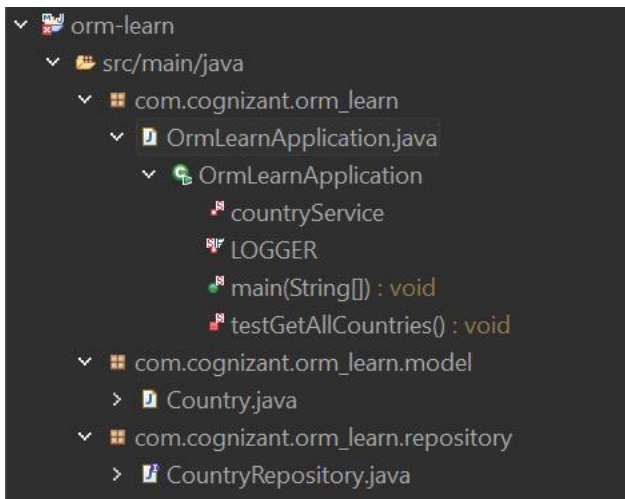
    <dependency>
<groupId>jakarta.persistence</groupId>
<artifactId>jakarta.persistence-api</artifactId>

```

```

<version>3.1.0</version> <!-- or similar -->
</dependency>
</dependencies>
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
</project>

```



CountryRepository.java

```

package com.cognizant.orm_learn.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.cognizant.orm_learn.model.Country;

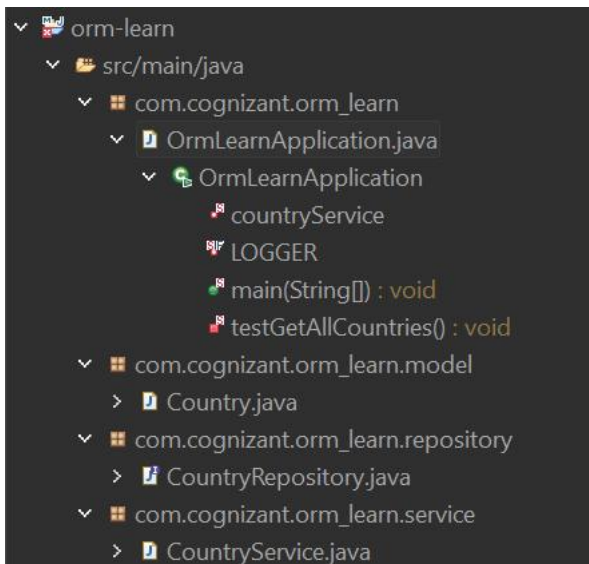
```

@Repository

```

public interface CountryRepository extends JpaRepository<Country, String> {
}

```



CountryService.java

```
package com.cognizant.orm_learn.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.orm_learn.model.Country;
import com.cognizant.orm_learn.repository.CountryRepository;
```

@Service

```
public class CountryService {
```

@Autowired

```
private CountryRepository countryRepository;
```

@Transactional

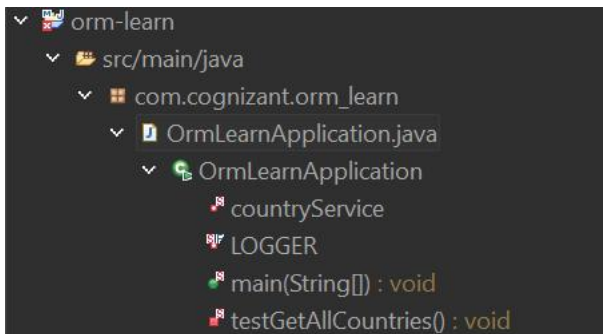
```
public List<Country> getAllCountries() {
```



```

        return countryRepository.findAll();
    }
}

```



OrmLearnApplication.java

```

package com.cognizant.orm_learn;

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.orm_learn.model.Country;

import com.cognizant.orm_learn.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);

```

```

        countryService = context.getBean(CountryService.class);

```

```

        testGetAllCountries();

```

```

    }

```

```

private static void testGetAllCountries() {
    LOGGER.info("Start");

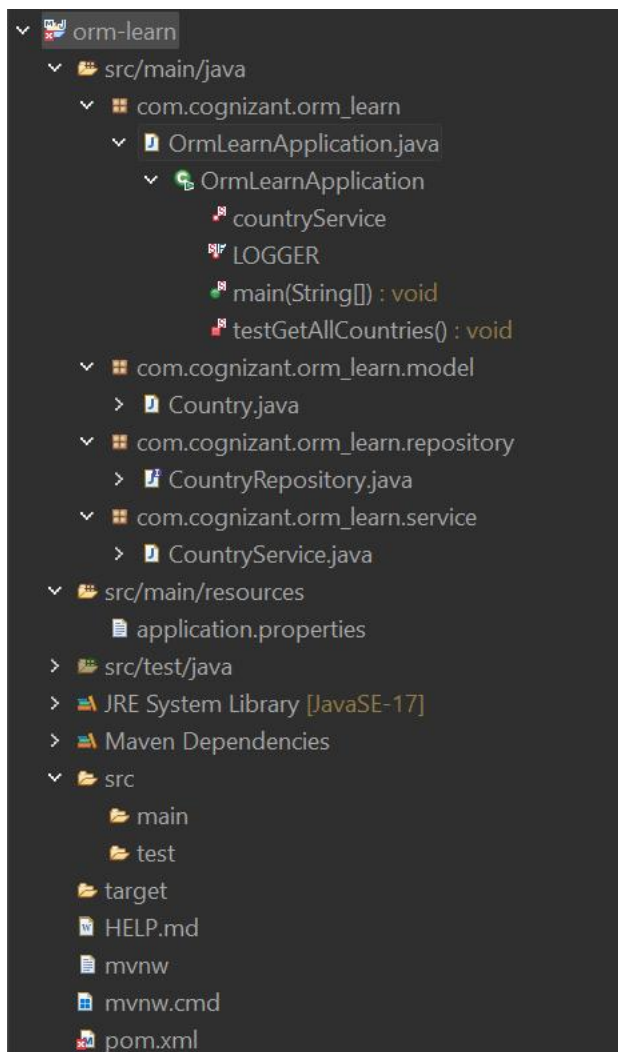
    List<Country> countries = countryService.getAllCountries();

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

    LOGGER.info("End");
}
}

```

Project Workflow/Structure:



```

05-07-25 13:05:26.475 restartedMain INFO c.c.o.OrmLearnApplication logStarting 53 Starting OrmLearnApplication using Java 23.0.1 w
05-07-25 13:05:26.477 restartedMain DEBUG c.c.o.OrmLearnApplication logStarting 54 Running with Spring Boot v3.5.3, Spring v6.2.8
05-07-25 13:05:26.477 restartedMain INFO c.c.o.OrmLearnApplication logStartupProfileInfo 652 No active profile set, falling back to 1 default
05-07-25 13:05:26.528 restartedMain INFO ertyDefaultsPostProcessor logTo 252 Devtools property defaults active! Set 'spring.d
05-07-25 13:05:26.865 restartedMain INFO toryConfigurationDelegate registerRepositoriesIn 145 Bootstrapping Spring Data JPA repositories in DE
05-07-25 13:05:26.909 restartedMain INFO toryConfigurationDelegate registerRepositoriesIn 213 Finished Spring Data repository scanning in 34 m
05-07-25 13:05:27.160 restartedMain INFO o.h.j.i.util.LogHelper logPersistenceUnitInformation 31 HHH000204: Processing PersistenceUnitInfo [n
05-07-25 13:05:27.193 restartedMain INFO org.hibernate.Version logVersion 44 HHH000412: Hibernate ORM core version 6.6.18.Fin
05-07-25 13:05:27.215 restartedMain INFO .i.RegionFactoryInitiator initiateService 50 HHH00026: Second-level cache disabled
05-07-25 13:05:27.474 restartedMain INFO SpringPersistenceUnitInfo addTransformer 87 No LoadTimeWeaver setup: ignoring JPA class tran
05-07-25 13:05:27.495 restartedMain INFO c.z.h.HikariDataSource getConnection 109 HikariPool-1 - Starting...
05-07-25 13:05:27.769 restartedMain INFO c.z.h.pool.HikariPool checkFailFast 575 HikariPool-1 - Added connection com.mysql.cj.jdbc
05-07-25 13:05:27.771 restartedMain INFO c.z.h.HikariDataSource getConnection 122 HikariPool-1 - Start completed.
05-07-25 13:05:27.819 restartedMain WARN o.h.orm.deprecation constructDialect 153 HHH9000025: MySQLDialect does not need to be sp
05-07-25 13:05:27.834 restartedMain INFO o.h.o.connections.pooling logConnectionInfo 163 HHH10001005: Database info:
Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
Database driver: undefined/unknown
Database version: 8.0.42
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown

```

95-07-25	13:05:27.850	restartedMain	DEBUG	h.t.d.s.s.DdlTypeRegistry	addDescriptor	64	addDescriptor(12, org.hibernate.type.descriptor.
95-07-25	13:05:27.851	restartedMain	DEBUG	h.t.d.s.s.DdlTypeRegistry	addDescriptor	64	addDescriptor(-9, org.hibernate.type.descriptor.
95-07-25	13:05:27.851	restartedMain	DEBUG	h.t.d.s.s.DdlTypeRegistry	addDescriptor	64	addDescriptor(-3, org.hibernate.type.descriptor.
95-07-25	13:05:27.851	restartedMain	DEBUG	h.t.d.s.s.DdlTypeRegistry	addDescriptor	64	addDescriptor(4003, org.hibernate.type.descriptor.
95-07-25	13:05:27.852	restartedMain	DEBUG	h.t.d.s.s.DdlTypeRegistry	addDescriptor	64	addDescriptor(4001, org.hibernate.type.descriptor.
95-07-25	13:05:27.852	restartedMain	DEBUG	h.t.d.s.s.DdlTypeRegistry	addDescriptor	64	addDescriptor(4002, org.hibernate.type.descriptor.
95-07-25	13:05:27.852	restartedMain	DEBUG	h.t.d.s.s.DdlTypeRegistry	addDescriptor	64	addDescriptor(2004, org.hibernate.type.descriptor.
95-07-25	13:05:27.852	restartedMain	DEBUG	h.t.d.s.s.DdlTypeRegistry	addDescriptor	64	addDescriptor(2005, org.hibernate.type.descriptor.
95-07-25	13:05:27.852	restartedMain	DEBUG	h.t.d.s.s.DdlTypeRegistry	addDescriptor	64	addDescriptor(2011, org.hibernate.type.descriptor.
95-07-25	13:05:28.421	restartedMain	INFO	p.i.JtaPlatformInitiator	initiateService	59	HHH000489: No JTA platform available (set 'hiber
95-07-25	13:05:28.457	restartedMain	INFO	p.EntityManagerFactoryBean	buildNativeEntityManagerFactory	447	Initialized JPA EntityManagerFactory for p
95-07-25	13:05:28.783	restartedMain	INFO	.OptionalLiveReloadServer	startServer	59	LiveReload server is running on port 35729
95-07-25	13:05:28.794	restartedMain	INFO	c.c.o.OrmLearnApplication	logStarted	59	Started OrmLearnApplication in 2.642 seconds (pr
95-07-25	13:05:28.797	restartedMain	INFO	c.c.o.OrmLearnApplication	testGetAllCountries	29	Start
95-07-25	13:05:28.900	restartedMain	DEBUG	org.hibernate.SQL	logStatement	135	select c1_0.co_code,c1_0.co_name from country c1
95-07-25	13:05:28.925	restartedMain	DEBUG	c.c.o.OrmLearnApplication	testGetAllCountries	31	countries=[Country [code=IN, name=India], Countr
95-07-25	13:05:28.925	restartedMain	INFO	c.c.o.OrmLearnApplication	testGetAllCountries	32	End
95-07-25	13:05:28.930	locationShutdownHook	INFO	p.EntityManagerFactoryBean	destroy	660	Closing JPA EntityManagerFactory for persistence
95-07-25	13:05:28.932	locationShutdownHook	INFO	c.z.h.HikariDataSource	close	349	HikariPool-1 - Shutdown initiated...
95-07-25	13:05:28.937	locationShutdownHook	INFO	c.z.h.HikariDataSource	close	351	HikariPool-1 - Shutdown completed.

```
LiveReload server is running on port 35729
Started OrmLearnApplication in 2.642 seconds (process running for 2.959)
Start
select c1_0.co_code,c1_0.co_name from country c1_0
countries=[Country [code=IN, name=India], Country [code=US, name=United States of America]]
End
Closing JPA EntityManagerFactory for persistence unit 'default'
```

Hands on 4

Difference between JPA, Hibernate and Spring Data JPA

Feature	JPA	Hibernate	Spring Data JPA
Type	Specification	Implementation of JPA	Abstraction over JPA and Hibernate
Provided by	javax.persistence or jakarta.persistence	org.hibernate	org.springframework.data.jpa.repository
Config Required	Yes (persistence.xml)	Yes (hibernate.cfg.xml)	No (Spring Boot auto-config)
Uses	Interfaces & annotations	JPA + native Hilbernate APIs	JPA + Spring Repositories.
Boilerplate Code	Moderate	High	Minimal
Transaction Mgmt	Manual	Manual	Auto-managed
Ease of Use	Medium	Harder	Very easy
Purpose	Defines ORM behavior	Implements ORM	Simplifies JPA usage in Spring apps
Default Query Language	JPQL	JPQL + HQL +native SQL	JPQL + derived queries + @Query support.

JPA Example (Using EntityManager):

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("persistenceUnit");
```

```
EntityManager em = emf.createEntityManager();
```

```
em.getTransaction().begin();
```

```
Country country = new Country();
```

```
country.setCode("FR");
```

```
country.setName("France");
```

```
em.persist(country);
```

```
em.getTransaction().commit();  
em.close();
```

Hibernate Example (Using Session and Transaction):

```
SessionFactory factory = new Configuration().configure().buildSessionFactory();  
Session session = factory.openSession();  
Transaction tx = session.beginTransaction();
```

```
Country country = new Country();  
country.setCode("FR");  
country.setName("France");  
session.save(country);
```

```
tx.commit();  
session.close();
```

Spring Data JPA Example (what we used in Hands-on 1)

@Autowired

```
private CountryRepository countryRepository;
```

```
public void saveCountry() {  
    Country country = new Country();  
    country.setCode("FR");  
    country.setName("France");  
    countryRepository.save(country);  
}
```

