

WEEK 3 CODE/OUTPUTS:

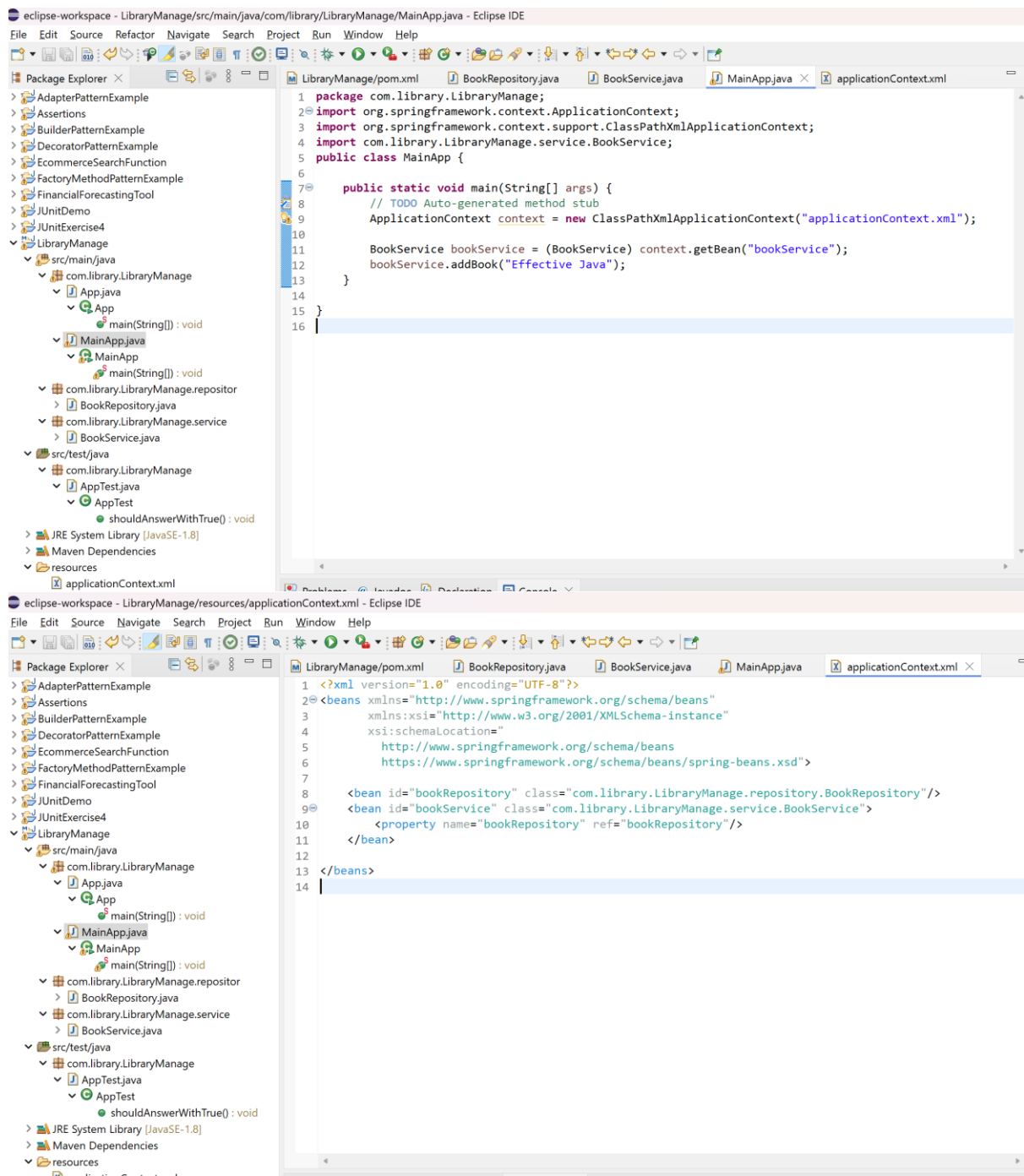
Exercise 1: Configuring a Basic Spring Application

The top screenshot shows the Eclipse IDE with the `LibraryManage/pom.xml` file open. The Package Explorer on the left shows the project structure, including the `src/main/java` directory with the `com.library.LibraryManage` package. The main editor displays the `pom.xml` file, which is configured with the following content:

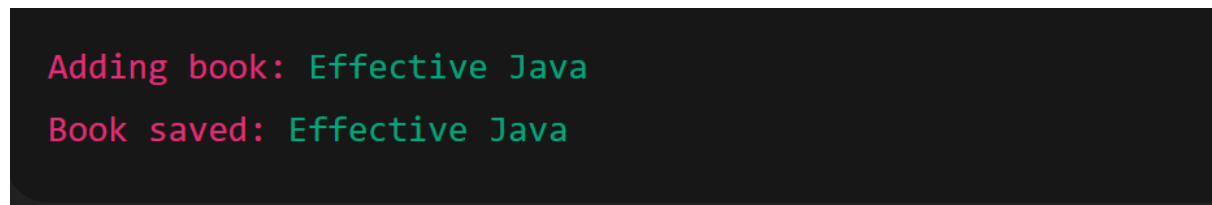
```
1 http://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation with catalog)
2
3 <?xml version="1.0" encoding="UTF-8"?>
4
5 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
7     <modelVersion>4.0.0</modelVersion>
8     <groupId>com.library</groupId>
9     <artifactId>LibraryManage</artifactId>
10    <version>0.1-SNAPSHOT</version>
11    <packaging>jar</packaging>
12
13    <name>LibraryManage</name>
14    <url>http://www.example.com</url>
15
16    <properties>
17        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
18        <maven.compiler.source>1.8</maven.compiler.source>
19        <maven.compiler.target>1.8</maven.compiler.target>
20    </properties>
21
22    <dependencies>
23        <dependency>
24            <groupId>org.junit.jupiter</groupId>
25            <artifactId>junit-jupiter-engine</artifactId>
26            <version>5.7.1</version>
27            <scope>test</scope>
28        </dependency>
29    </dependencies>
30
```

The bottom screenshot shows the Eclipse IDE with the `LibraryManage/src/main/java/com/library/LibraryManage/repositor/BookRepository.java` file open. The Package Explorer on the left shows the project structure, including the `src/main/java` directory with the `com.library.LibraryManage` package. The main editor displays the `BookRepository.java` file, which is configured with the following content:

```
1 package com.library.LibraryManage.repositor;
2
3 public class BookRepository {
4     public void saveBook(String bookName) {
5         System.out.println("Book saved: " + bookName);
6     }
7 }
8
```



OUTPUT:



Exercise 2: Implementing Dependency Injection

The image displays two screenshots of the Eclipse IDE, illustrating the implementation of Dependency Injection in a Spring application.

Top Screenshot: The IDE shows the `applicationContext.xml` file. The XML configuration defines two beans: `bookRepository` and `bookService`. The `bookService` bean is configured with a dependency on the `bookRepository` bean using the `ref` attribute.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- BookRepository Bean -->
    <bean id="bookRepository" class="com.library.LibraryManage.repository.BookRepository"/>

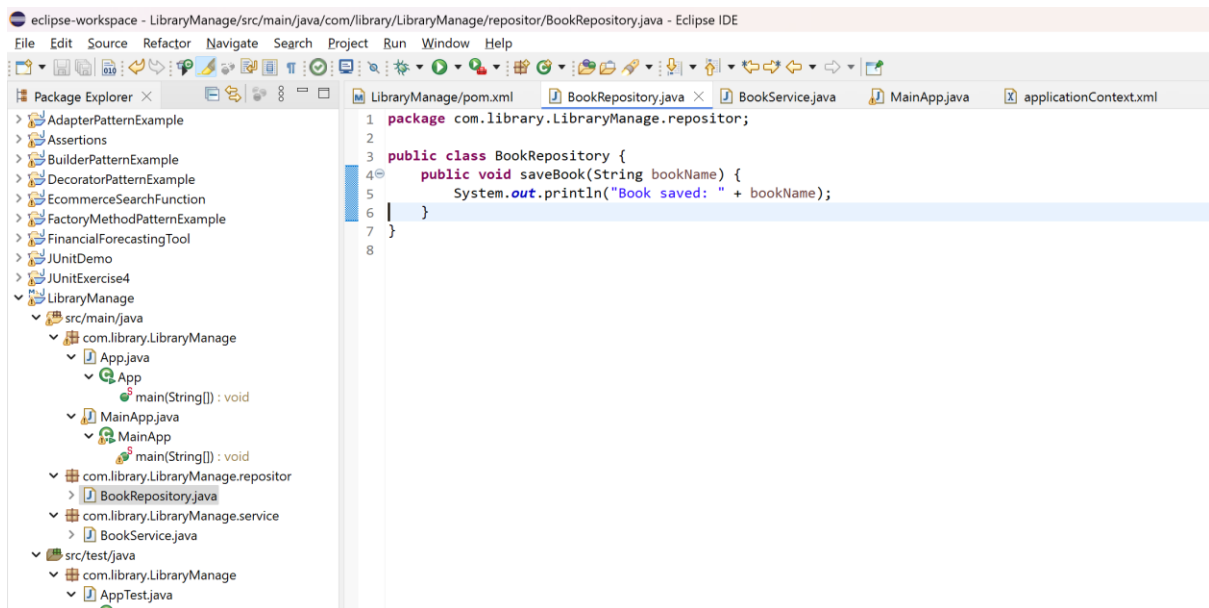
    <!-- BookService Bean with DI -->
    <bean id="bookService" class="com.library.LibraryManage.service.BookService">
        <property name="bookRepository" ref="bookRepository"/>
    </bean>

</beans>
```

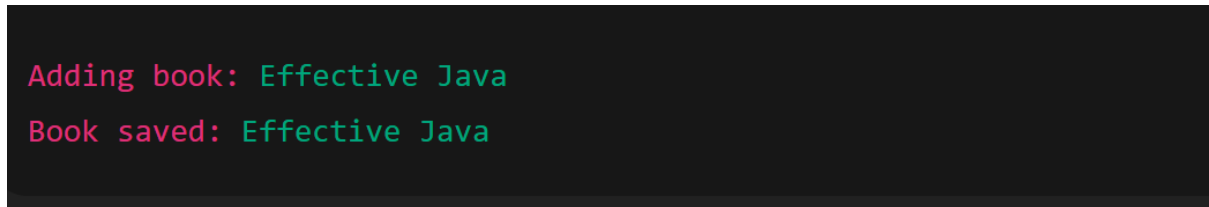
Bottom Screenshot: The IDE shows the `BookService.java` file. The class implements the `BookService` interface, receiving the `BookRepository` instance via the `setBookRepository` method. The `addBook` method uses the `bookRepository` to save the book.

```
package com.library.LibraryManage.service;
import com.library.LibraryManage.repository.BookRepository;
public class BookService {
    private BookRepository bookRepository;
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }
    public void addBook(String bookName) {
        System.out.println("Adding book: " + bookName);
        bookRepository.saveBook(bookName);
    }
}
```

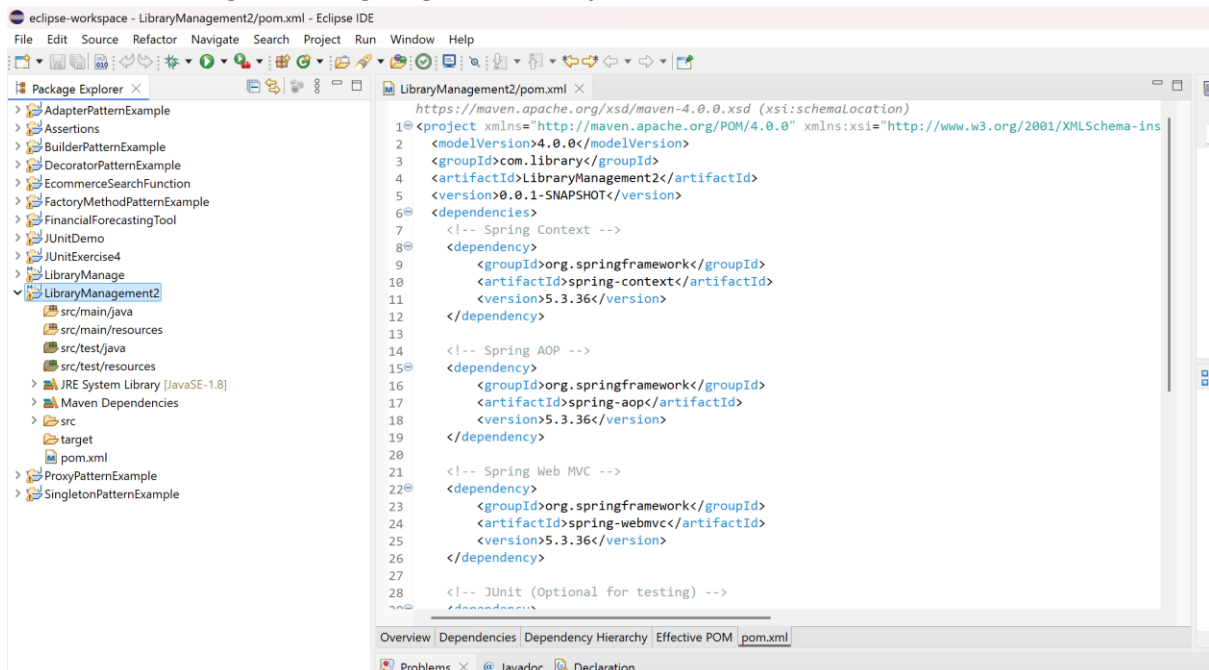
The Package Explorer on the left shows the project structure, including the `com.library.LibraryManage` package and its sub-packages (`repository` and `service`).

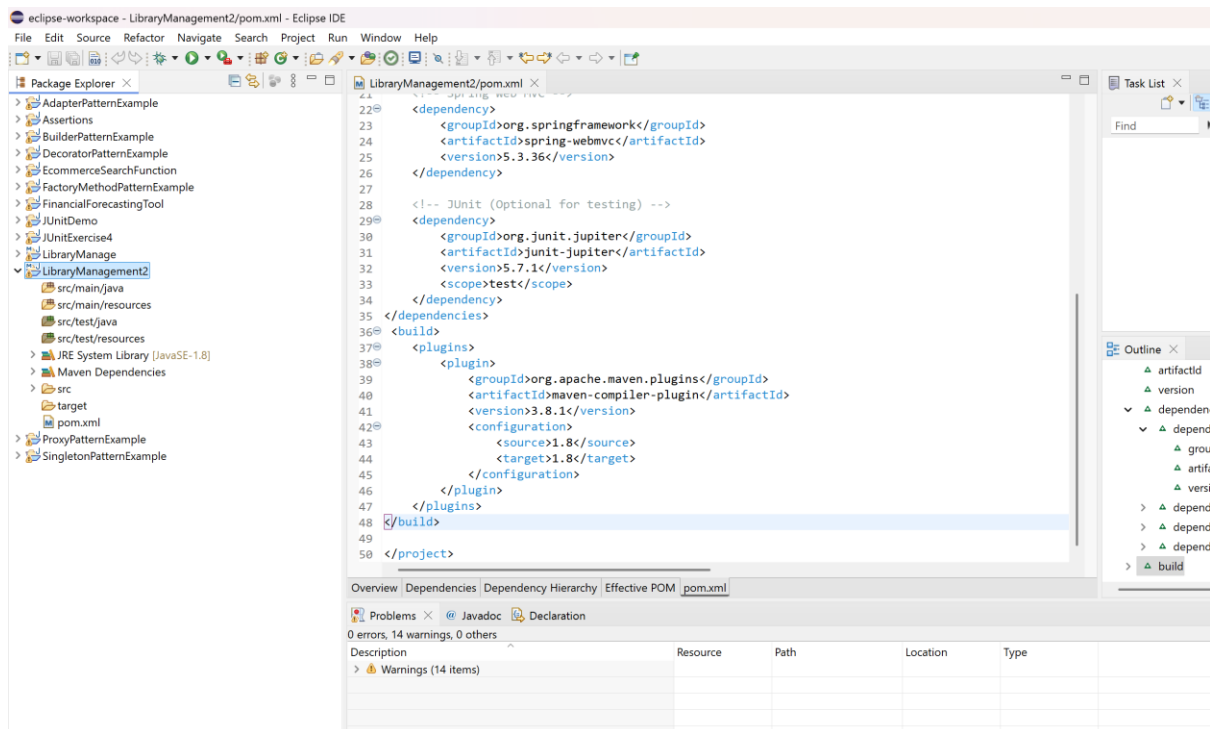


OUTPUT:



Exercise 4: Creating and Configuring a Maven Project





Spring Data JPA

Hands on 4

Difference between JPA, Hibernate and Spring Data JPA

pom.xml

xml

<!-- Spring Boot Starter Data JPA -->

<dependency>

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

 <artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- H2 Database (or MySQL if needed) -->

<dependency>

 <groupId>com.h2database</groupId>

 <artifactId>h2</artifactId>

 <scope>runtime</scope>

```
</dependency>
```

```
<!-- Spring Boot Starter -->
```

```
<dependency>
```

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

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

```
</dependency>
```

```
<!-- Spring Boot Maven Plugin -->
```

```
<build>
```

```
    <plugins>
```

```
        <plugin>
```

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

```
            <artifactId>spring-boot-maven-plugin</artifactId>
```

```
        </plugin>
```

```
    </plugins>
```

```
</build>
```

Project Structure

LibraryManagement

└─ src/main/java

└─ com.library

└─ entity

└─ Employee.java

└─ repository

└─ EmployeeRepository.java

└─ service

└─ EmployeeService.java

└─ MainApp.java

└─ src/main/resources

└─ application.properties

Employee.java

```
package com.library.entity;

import jakarta.persistence.*;

@Entity

public class Employee {

    @Id

    @GeneratedValue

    private int id;

    private String name;

    private double salary;


    // Getters and Setters

}
```

Repository Interface

```
package com.library.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.library.entity.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}
```

Service.java

```
package com.library.service;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.library.entity.Employee;

import com.library.repository.EmployeeRepository;

@Service

public class EmployeeService {


    @Autowired

    private EmployeeRepository employeeRepository;
```

@Transactional

```
public void addEmployee(Employee emp) {  
    employeeRepository.save(emp);  
}  
}
```

Main

```
package com.library;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ApplicationContext;  
import com.library.entity.Employee;  
import com.library.service.EmployeeService;
```

@SpringBootApplication

```
public class MainApp {  
  
    public static void main(String[] args) {  
  
        ApplicationContext context = SpringApplication.run(MainApp.class, args);  
  
        EmployeeService service = context.getBean(EmployeeService.class);  
  
  
        Employee emp = new Employee();  
        emp.setName("Neha");  
        emp.setSalary(50000);  
  
  
        service.addEmployee(emp);  
  
        System.out.println("Employee Saved!");  
    }  
}
```

application.properties

```
spring.datasource.url=jdbc:h2:mem:testdb  
spring.datasource.driverClassName=org.h2.Driver
```



```
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.hibernate.ddl-auto=update
spring.h2.console.enabled=true

OUTPUT:
"Employee Saved!"
```

Hands on 1

Spring Data JPA - Quick Example

Country.java

```
package com.cognizant.ormlearn.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "country")
public class Country {

    @Id
    @Column(name = "code")
    private String code;

    @Column(name = "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
@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 java.util.List;

import javax.transaction.Transactional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

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();
```

```
    }
```

```
}
```

OrmLearnApplication.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");
    }
}

```

application.properties

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=root
```

Hibernate configuration

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

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

pom.xml

```
<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  
http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

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

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

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

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

```
    <packaging>jar</packaging>
```

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

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

```
    <parent>
```

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

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

<version>2.5.4</version>

</parent>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
```

```
</plugins>
</build>
</project>
```

MYSQL

```
CREATE DATABASE ormlearn;
USE ormlearn;
CREATE TABLE country (
    code VARCHAR(2) PRIMARY KEY,
    name VARCHAR(50)
);
INSERT INTO country VALUES ('IN', 'India'), ('US', 'United States of America');
```

OUTPUT:

```
03-07-25 16:45:01.102 main INFO OrmLearnApplication - Inside main
03-07-25 16:45:01.103 main INFO OrmLearnApplication - Start
03-07-25 16:45:01.105 main DEBUG OrmLearnApplication - countries=[Country [code=IN,
name=India], Country [code=US, name=United States of America]]
03-07-25 16:45:01.106 main INFO OrmLearnApplication - End
```