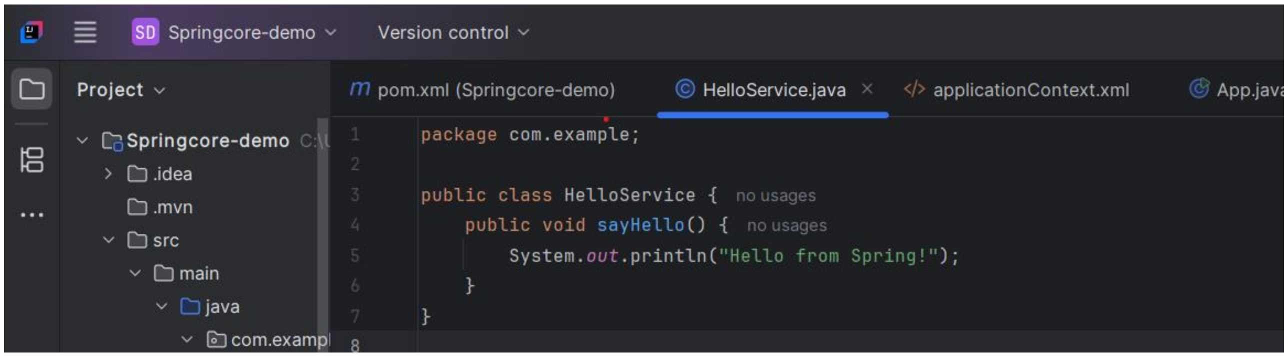
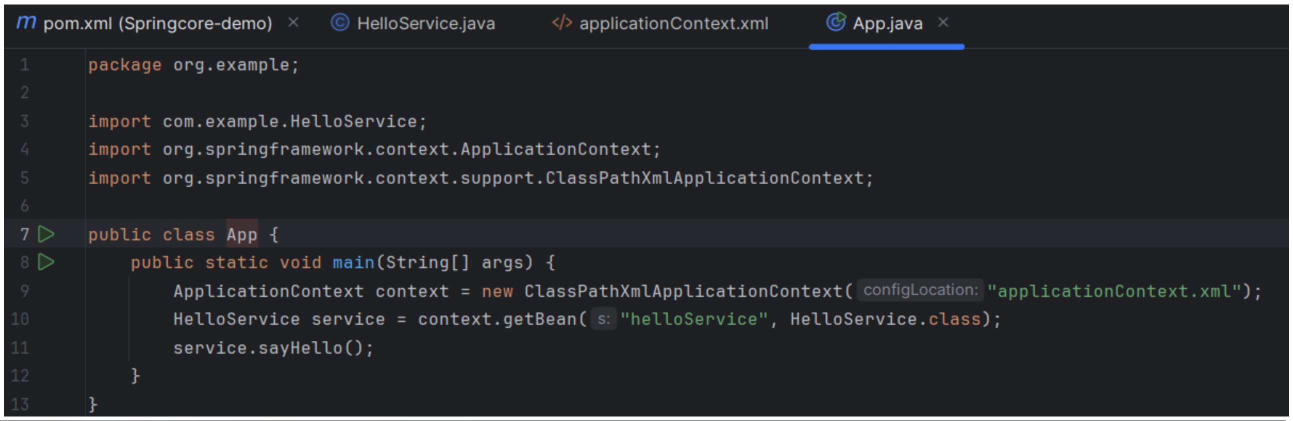
**Week 3**
**Spring Core and Maven**

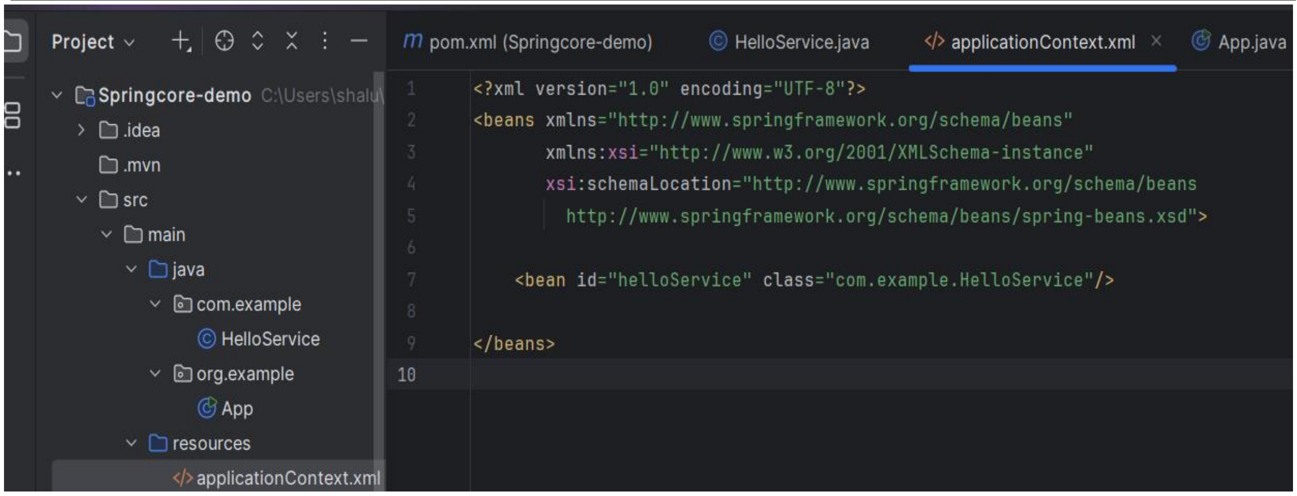
***Excercise 1: Configuring a Basic Spring Application***



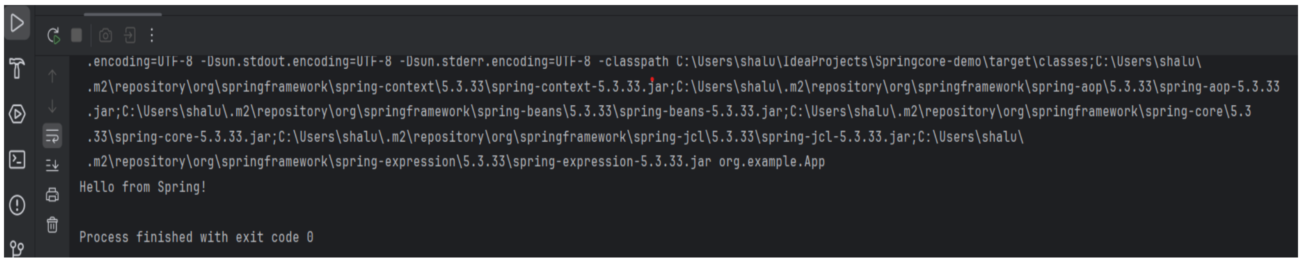
```java
package com.example;

public class HelloService {
    public void sayHello() {
        System.out.println("Hello from Spring!");
    }
}
```



```java
package org.example;

import com.example.HelloService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext( configLocation: "applicationContext.xml");
        HelloService service = context.getBean( s: "helloService", HelloService.class);
        service.sayHello();
    }
}
```

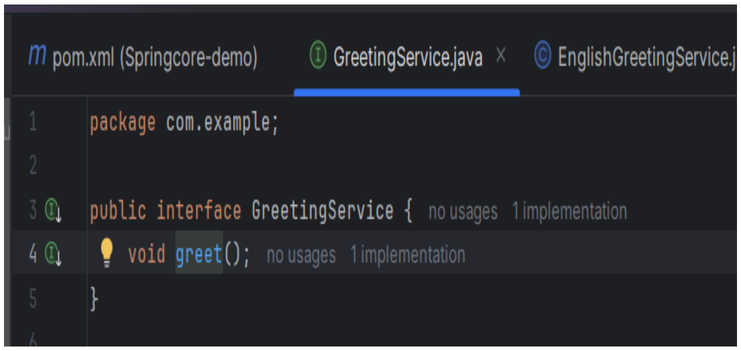


```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
         http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="helloService" class="com.example.HelloService"/>

</beans>
```

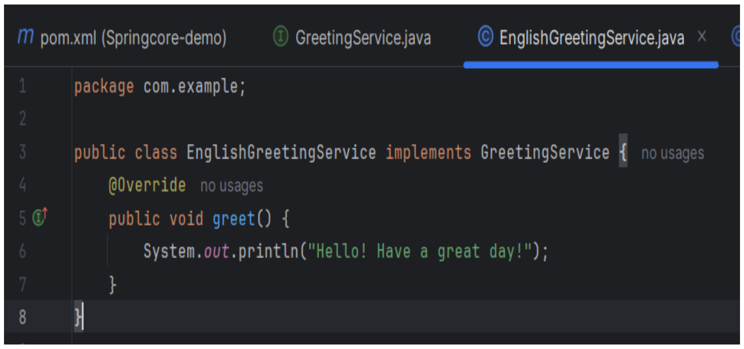


```
 .encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath C:\Users\shalu\IdeaProjects\Springcore-demo\target\classes;C:\Users\shalu\
 .m2\repository\org\springframework\spring-context\5.3.33\spring-context-5.3.33.jar;C:\Users\shalu\.m2\repository\org\springframework\spring-aop\5.3.33\spring-aop-5.3.33
 .jar;C:\Users\shalu\.m2\repository\org\springframework\spring-beans\5.3.33\spring-beans-5.3.33.jar;C:\Users\shalu\.m2\repository\org\springframework\spring-core\5.3
 .33\spring-core-5.3.33.jar;C:\Users\shalu\.m2\repository\org\springframework\spring-jcl\5.3.33\spring-jcl-5.3.33.jar;C:\Users\shalu\
 .m2\repository\org\springframework\spring-expression\5.3.33\spring-expression-5.3.33.jar org.example.App
Hello from Spring!

Process finished with exit code 0
```

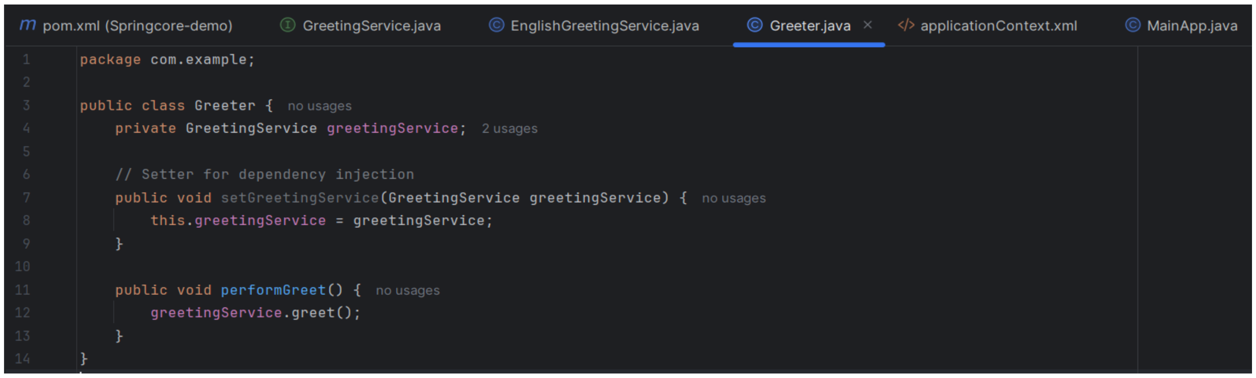
## *Exercise 2: Implementing Dependency Injection*



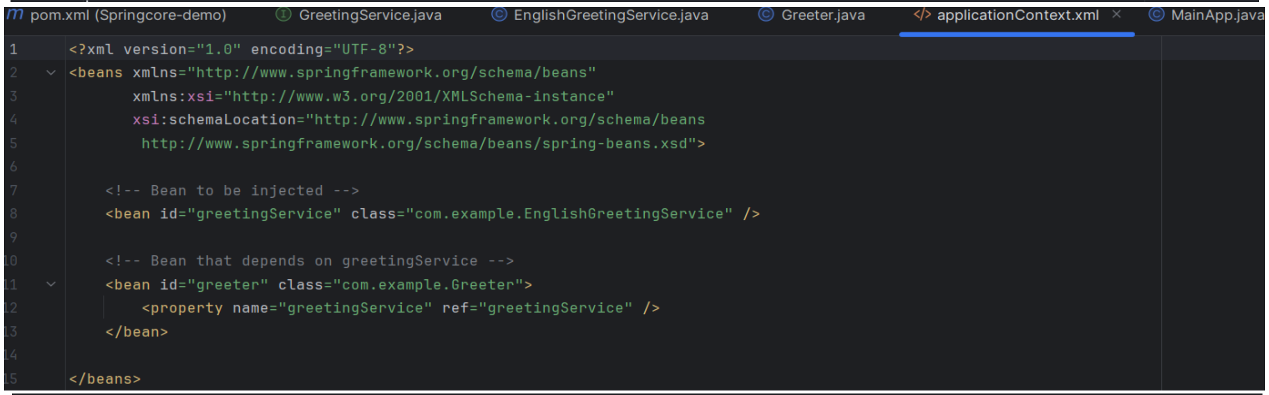
```java
package com.example;

public interface GreetingService {
    void greet();
}
```



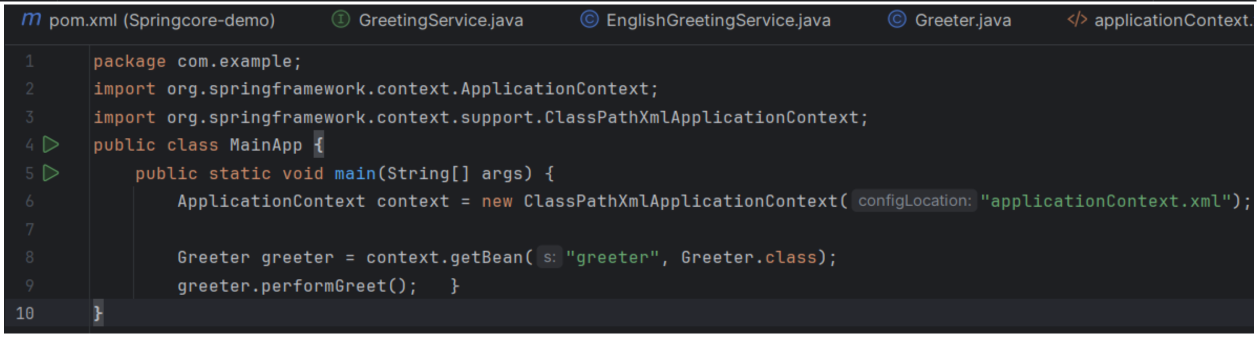
```java
package com.example;

public class EnglishGreetingService implements GreetingService {
    @Override
    public void greet() {
        System.out.println("Hello! Have a great day!");
    }
}
```



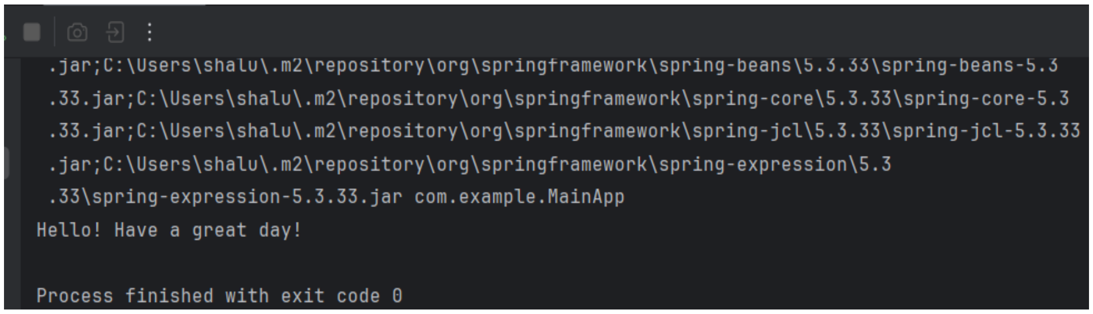
```java
package com.example;

public class Greeter {
    private GreetingService greetingService;

    // Setter for dependency injection
    public void setGreetingService(GreetingService greetingService) {
        this.greetingService = greetingService;
    }

    public void performGreet() {
        greetingService.greet();
    }
}
```



```xml
<?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">

    <!-- Bean to be injected -->
    <bean id="greetingService" class="com.example.EnglishGreetingService" />

    <!-- Bean that depends on greetingService -->
    <bean id="greeter" class="com.example.Greeter">
        <property name="greetingService" ref="greetingService" />
    </bean>

</beans>
```

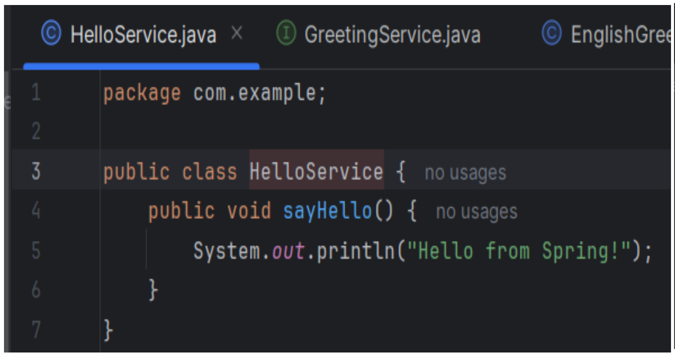


```java
package com.example;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext( configLocation: "applicationContext.xml");

        Greeter greeter = context.getBean( s: "greeter", Greeter.class);
        greeter.performGreet();   }
}
```

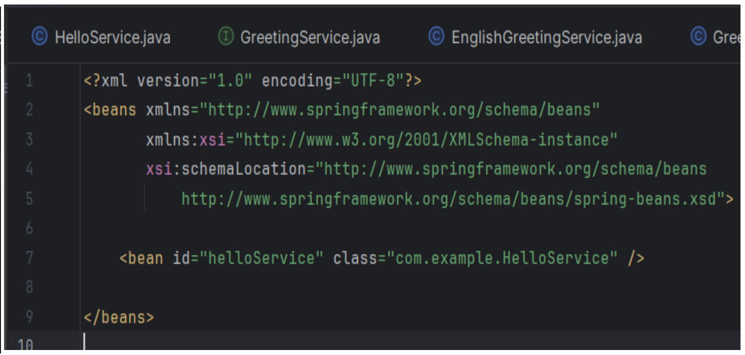


```
 .jar;C:\Users\shalu\.m2\repository\org\springframework\spring-beans\5.3.33\spring-beans-5.3
 .33.jar;C:\Users\shalu\.m2\repository\org\springframework\spring-core\5.3.33\spring-core-5.3
 .33.jar;C:\Users\shalu\.m2\repository\org\springframework\spring-jcl\5.3.33\spring-jcl-5.3.33
 .jar;C:\Users\shalu\.m2\repository\org\springframework\spring-expression\5.3
 .33\spring-expression-5.3.33.jar com.example.MainApp
Hello! Have a great day!

Process finished with exit code 0
```

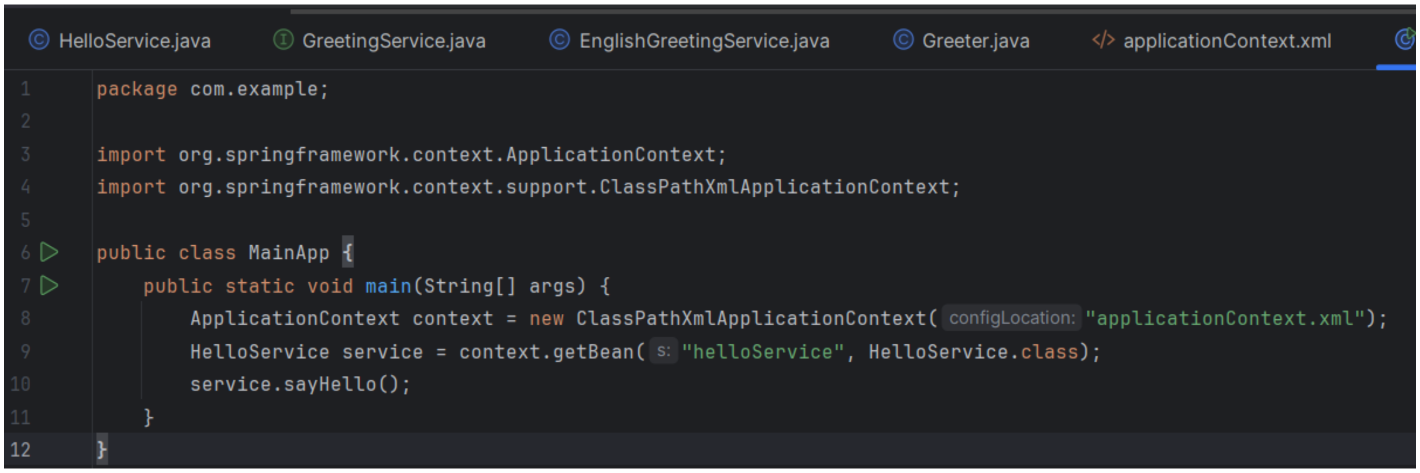
*Exercise 3: Creating and Configuring a Maven Project*



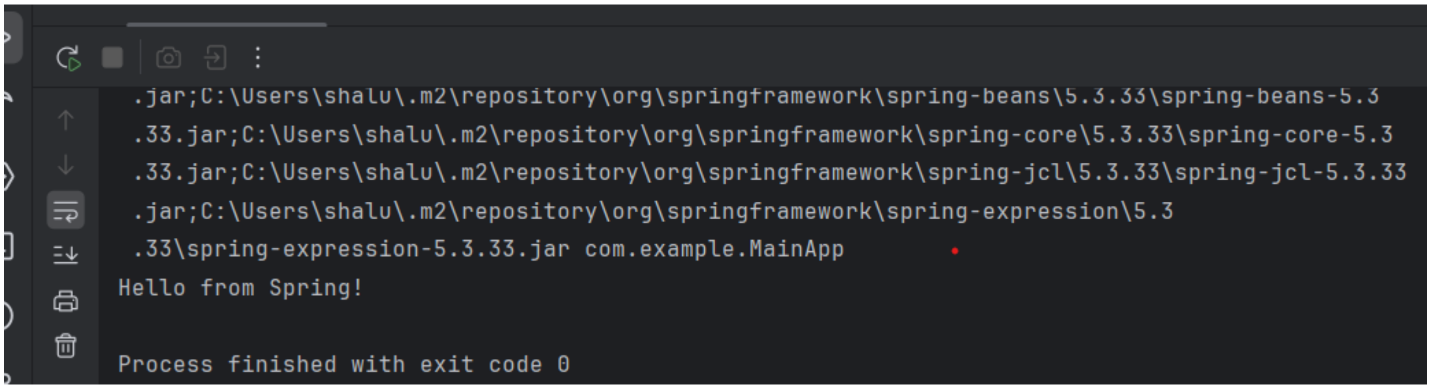
```java
package com.example;

public class HelloService {  no usages
    public void sayHello() {  no usages
        System.out.println("Hello from Spring!");
    }
}
```



```xml
<?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">

    <bean id="helloService" class="com.example.HelloService" />

</beans>
```



```java
package com.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext( configLocation: "applicationContext.xml");
        HelloService service = context.getBean( s: "helloService", HelloService.class);
        service.sayHello();
    }
}
```



```
.jar;C:\Users\shalu\.m2\repository\org\springframework\spring-beans\5.3.33\spring-beans-5.3
.33.jar;C:\Users\shalu\.m2\repository\org\springframework\spring-core\5.3.33\spring-core-5.3
.33.jar;C:\Users\shalu\.m2\repository\org\springframework\spring-jcl\5.3.33\spring-jcl-5.3.33
.jar;C:\Users\shalu\.m2\repository\org\springframework\spring-expression\5.3
.33\spring-expression-5.3.33.jar com.example.MainApp
Hello from Spring!

Process finished with exit code 0
```

***Exercise 4:***

***Spring Data JPA - Quick Example using Spring Boot and Hiberante***

**What is Spring Data JPA?**

Spring Data JPA is a part of the Spring Framework that helps you easily interact with databases using Java objects — without writing complex SQL queries.

It is built on top of JPA (Java Persistence API) and uses Hibernate behind the scenes to manage the actual database communication.

**What is Hibernate?**

Hibernate is a tool (ORM - Object Relational Mapping) that maps Java classes to database tables. It makes it easier to save, read, and update data in the database using simple Java code.

**Instead of writing:**

```
SELECT * FROM students WHERE id = 1;
```

**You just do this in Java:**

```
Student student = studentRepository.findById(1L).get();
```

**Tools Used in the eg:**

Spring Boot: Simplifies project setup and auto-configuration.

Spring Data JPA: Handles database CRUD operations easily.

Hibernate: Actual engine doing the work behind JPA.

H2 Database: In-memory database for testing.

Lombok (optional): Reduces boilerplate code.

**What's in the Project?**

1. **Entity class:**

   A Java class mapped to a database table

   **Example: @**Entity

```
public class Student {
@Id
private Long id;
 private String name;
}
```

2. **Repository interface:**

   No need to write SQL — just extend JpaRepository

Eg: public interface StudentRepository extends JpaRepository<Student, Long> {}

3. **Controller class:**

   Handles HTTP requests like POST, GET, PUT, DELETE

   Eg: @RestController

   @RequestMapping("/students")

   public class StudentController { ... }

4. **application.properties:**
   Configues DB and Hiberante settings
   Eg: spring.datasource.url=jdbc:h2:mem:testdb
   spring.jpa.hibernate.ddl-auto=update

**What You Can Do With It**

Save a new student → POST /students

View all students → GET /students

Get one student by ID → GET /students/{id}

Update student → PUT /students/{id}

Delete student → DELETE /students/{id}

***Exercise 5: Difference between JPA, Hibernate and Spring Data JPA***

| **Feature** | **JPA (Java Persistence API)** | **Hibernate** | **Spring Data JPA** |
|---|---|---|---|
| **Type** | Specification / Interface | Implementation of JPA | Abstraction over JPA with Spring support |
| **Provided By** | Java (official, part of Java EE / Jakarta EE) | Third-party library (by Red Hat) | Spring Framework |
| **Purpose** | Defines how Java objects should map to DB tables | Actually, performs the object-to-table mapping | Simplifies JPA by reducing boilerplate code |
| **Requires Implementation?** | Yes (you need Hibernate) | No (it is already an implementation) | Uses Hibernate (or any JPA provider) underneath |
| **Configuration** | Requires manual configuration | Requires configuration | Auto-configured by Spring Boot |
| **Boilerplate Code** | Needs EntityManager, transactions, etc. | Slightly easier than raw JPA | Minimal code, just extends JpaRepository. |
| **Common Use** | For standard JPA-based projects | When using Hibernate-specific features | For Spring Boot apps with database interaction |

**JPA (Java Persistence API)**

It's just a set of rules and interfaces.

It defines how Java classes should interact with a database.

But it doesn't do anything itself — it needs a provider like Hibernate.

Think of JPA as a blueprint.

**Hibernate**

It's a popular implementation of JPA.

It converts Java objects into database tables and vice versa.

Can also work standalone without JPA, with more features.

Think of Hibernate as the builder that follows JPA’s blueprint.

**Spring Data JPA**

It's a Spring module that builds on top of JPA and Hibernate.

It provides ready-made repository interfaces like JpaRepository so you don’t need to write boilerplate code.

It auto-generates SQL queries behind the scenes using method names.

Think of Spring Data JPA as a smart helper that simplifies both JPA and Hibernate.