Spring Maven

Exercise 1: Configuring a Basic Spring Application

Scenario:

Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

Steps:

1. **Set Up a Spring Project:**
   - Create a Maven project named **LibraryManagement**.
   - Add Spring Core dependencies in the **pom.xml** file.
2. **Configure the Application Context:**
   - Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
   - Define beans for **BookService** and **BookRepository** in the XML file.
3. **Define Service and Repository Classes:**
   - Create a package **com.library.service** and add a class **BookService**.
   - Create a package **com.library.repository** and add a class **BookRepository**.
4. **Run the Application:**
   - Create a main class to load the Spring context and test the configuration.

Program:

pom.xml

```
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.32</version>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

applicationContext.xml:

```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">

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

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

</beans>
```

BookServece.java:

```java
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {
    private BookRepository bookRepository;

    // Setter for Dependency Injection
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void displayBooks() {
        System.out.println(x:"BookService: Displaying books...");
        bookRepository.fetchBooks();
    }
}
```

BookRepository:

```java
package com.library.repository;

public class BookRepository {
    public void fetchBooks() {
        System.out.println(x:"Fetching list of books from repository...");
    }
}
```

MainApp.java:

```java
package com.library;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.library.service.BookService;

public class MainApp {
    Run | Debug
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext(configLocation:"applicationContext.xml");

        BookService bookService = (BookService) context.getBean("bookService");
        bookService.displayBooks();
    }
}
```

Output:

```
BookService: Displaying books...
Fetching list of books from repository...
```

Exercise 2: Implementing Dependency Injection

Scenario:

In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

Steps:

1. **Modify the XML Configuration:**
    - Update **applicationContext.xml** to wire **BookRepository** into **BookService**.
2. **Update the BookService Class:**
    - Ensure that **BookService** class has a setter method for **BookRepository**.
3. **Test the Configuration:**
    - Run the **LibraryManagementApplication** main class to verify the dependency injection.

Program:

applicationContext.xml:

```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">

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

    <!-- Service Bean with Setter-based Dependency Injection -->
    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository"/>
    </bean>

</beans>
```

BookService.java:

```java
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {
    private BookRepository bookRepository;

    // Setter for Spring DI
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void displayBooks() {
        System.out.println(x:"BookService: Displaying books...");
        bookRepository.fetchBooks();
    }
}
```

Main.java:

```java
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.library.service.BookService;

public class MainApp {
    Run | Debug
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext(configLocation:"applicationContext.xml");

        BookService bookService = (BookService) context.getBean("bookService");
        bookService.displayBooks();
    }
}
```

Output:

```
BookService: Displaying books...
Fetching list of books from repository...
```

Exercise 4: Creating and Configuring a Maven Project

Scenario:

You need to set up a new Maven project for the library management application and add Spring dependencies.

Steps:

1. **Create a New Maven Project:**
    - Create a new Maven project named **LibraryManagement**.
2. **Add Spring Dependencies in pom.xml:**
    - Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.
3. **Configure Maven Plugins:**
    - Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

Program:

LibrarymanagementApplication.java:

```java
package com.library;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class LibrarymanagementApplication {
    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(primarySource:LibrarymanagementApplication.class, args);
    }
}
```

RootController.java:

```java
package com.library.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class RootController {
    @GetMapping("/")
    public String home() {
        return "Welcome to the Library Management System!";
    }
}
```

BookController.java:

```java
package com.library.controller;

import com.library.service.BookService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/books")
public class BookController {

    @Autowired
    private BookService bookService;

    @GetMapping("/{id}")
    public String getBook(@PathVariable int id) {
        String result = bookService.findBookById(id);
        if (result == null || result.isEmpty()) {
            return "Book not found for ID: " + id;
        }
        return result;
    }

    @PostMapping
    public String addBook(@RequestBody String bookName) {
        return bookService.addBook(bookName);
    }

    @GetMapping({ "/", "" })
    public String home() {
        return "Welcome to the Library Management System!";
    }
}
```

LoggingAspect.java:

```java
package com.library.aspect;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.*;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class LoggingAspect {

    @Before("execution(* com.library.service.*.*(..))")
    public void beforeMethod(JoinPoint joinPoint) {
        System.out.println("🔍 Entering method: " + joinPoint.getSignature().getName());
    }

    @AfterReturning(pointcut = "execution(* com.library.service.*.*(..))", returning = "result")
    public void afterReturning(JoinPoint joinPoint, Object result) {
        System.out.println("✅ Exiting method: " + joinPoint.getSignature().getName() + " with result: " + result);
    }
}
```

Output:

Welcome Message:

Welcome to the Library Management System!

Retrieved book using ID:

Book with ID 101

SPRING DATA JPA HANDSON

Spring Data JPA- Quick Example

Program:

application.properties:

```
spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn
spring.datasource.username=rootnet stop mysql
spring.datasource.password=yourpassword
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
```

Department.java:

```
package com.example;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Department {

    @Id
    private int id;
    private String name;

    // Getters & Setters
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    // toString
    @Override
    public String toString() {
        return "Department [id=" + id + ", name=" + name + "]";
    }
}
```

schema.sql:

```sql
CREATE TABLE department (
    id INT PRIMARY KEY,
    name VARCHAR(100)
);
```

DepartmentRepository.java:

```java
package com.example;

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

public interface DepartmentRepository extends JpaRepository<Department, Integer> {
}
```

OrmLearnApplication.java

```java
package com.example;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {

    @Autowired
    private DepartmentRepository departmentRepository;

    public static void main(String[] args) {
        SpringApplication.run(OrmLearnApplication.class, args);
    }

    @Override
    public void run(String... args) {
        Department dept = new Department();
        dept.setId(1);
        dept.setName("HR");

        departmentRepository.save(dept);

        List<Department> departments = departmentRepository.findAll();
        departments.forEach(System.out::println);
    }
}
```

App.java

```java
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.domain.EntityScan;
import org.springframework.boot.CommandLineRunner;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.List;

@SpringBootApplication
@EntityScan(basePackages = ".")
public class App implements CommandLineRunner {

    @Autowired
    private DepartmentRepository departmentRepository;

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(App.class, args);
    }

    @Override
    public void run(String... args) {
        Department dept = new Department();
        dept.setId(1);
        dept.setName("HR");

        departmentRepository.save(dept);

        List<Department> departments = departmentRepository.findAll();
        departments.forEach(System.out::println);
    }
}
```

pom.xml:

```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.example</groupId>
<artifactId>orm-learn</artifactId>
<version>1.0.0</version>

<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.7.18</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>

<properties>
    <java.version>11</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-starter</artifactId>
    </dependency>

    <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
    </dependency>
</dependencies>

<build>
    <plugins>
    <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
    </plugins>
</build>
</project>
```

Output:

Difference between JPA, Hibernate and Spring Data JPA

Program:

Department.java:

```java
package com.example;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Department {

    @Id
    private int id;
    private String name;

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "Department [id=" + id + ", name=" + name + "]";
    }
}
```

DepartmentRepository.java

```java
package com.example;

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

public interface DepartmentRepository extends JpaRepository<Department, Integer> {
}
```

pom.xml:

```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.example</groupId>
<artifactId>orm-learn</artifactId>
<version>1.0.0</version>

<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.7.18</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>

<properties>
    <java.version>11</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-starter</artifactId>
    </dependency>

    <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
    </dependency>
</dependencies>

<build>
    <plugins>
    <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-pl gin</artifactId>
    </plugin>
    </plugins>
</build>
</project>
```

OrmLearningApplication.java

```java
package com.example;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class OrmLearnApplication implements CommandLineRunner {

    @Autowired
    private StudentRepository studentRepository;

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(OrmLearnApplication.class, args);
    }

    @Overrid
    public void run(String... args) throws Exception {
        studentRepository.save(new Student(1, "Alice"));
        studentRepository.save(new Student(2, "Bob"));

        System.out.println(x:"All Students:");
        studentRepository.findAll().forEach(System.out::println);

        System.out.println(x:"Students named Bob:");
        studentRepository.findByName("Bob").forEach(System.out::println);
    }
}
```

Output:


:: Spring Boot ::
(v2.7.18)
2025-07-04 19:16:53.164 INFO 8604 --- [main] com.example.OrmLearnApplication : Starting OrmLearnApplication using Java 21.0.7 on DESKTOP-B9PFVGD w
ith PID 8604 (E:\CTS\orm-learn-flat\target\classes started by Abhinithaa in E:\CTS\orm-learn-flat)
2025-07-04 19:16:53.167 INFO 8604 --- [main] com.example.OrmLearnApplication : No active profile set, falling back to 1 default profile: "default"
2025-07-04 19:16:53.766 INFO 8604 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-07-04 19:16:53.836 INFO 8604 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 57 ms. Found 2 JPA repo
sitory interfaces.
2025-07-04 19:16:54.480 INFO 8604 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2025-07-04 19:16:54.557 INFO 8604 --- [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 5.6.15.Final
2025-07-04 19:16:54.782 INFO 8604 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2025-07-04 19:16:54.987 INFO 8604 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2025-07-04 19:16:55.252 INFO 8604 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2025-07-04 19:16:55.276 INFO 8604 --- [main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.H2Dialect
2025-07-04 19:16:56.026 INFO 8604 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.
transaction.jta.platform.internal.NoJtaPlatform]
2025-07-04 19:16:56.041 INFO 8604 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-07-04 19:16:56.642 INFO 8604 --- [main] com.example.OrmLearnApplication : Started OrmLearnApplication in 4.815 seconds (JVM running for 4.585
)
All Students:
Student [id=1, name=Alice]
Student [id=2, name=Bob]
Students named Bob:
Student [id=2, name=Bob]
2025-07-04 19:16:56.915 INFO 8604 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
2025-07-04 19:16:56.918 INFO 8604 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
2025-07-04 19:16:56.922 INFO 8604 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 8.529 s
[INFO] Finished at: 2025-07-04T19:16:57+05:30
[INFO] --
E:\CTS\orm-learn-flat>