Exercise 1: Configuring a Basic Spring Application

1. Set Up a Spring Project

- **Project Name:** LibraryManagement
- **Group ID:** com.library
- **Artifact ID:** LibraryManagement

Add Spring Core Dependencies in 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.library</groupId>
    <artifactId>LibraryManagement</artifactId>
    <version>1.0-SNAPSHOT</version>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.20</version>
        </dependency>
    </dependencies>
```

```
</project>
```

Configure the Application Context (applicationContext.xml)

```
<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 Definitions -->
    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository"/>
    </bean>


    <bean id="bookRepository"
class="com.library.repository.BookRepository"/>
</beans>
```

BookService.java

```
package com.library.service;


import com.library.repository.BookRepository;


public class BookService {
    private BookRepository bookRepository;
```

```
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void manageBooks() {
        System.out.println("Managing books...");
    }
}
```

BookRepository.java

```
package com.library.repository;

public class BookRepository {
    public void saveBook() {
        System.out.println("Saving book...");
    }
}
```

LibraryManagementApplication.java

```
package com.library;
import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplicationConte
xt;
import com.library.service.BookService;
```

```
public class LibraryManagementApplication {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService =
context.getBean(BookService.class);
        bookService.manageBooks();
    }
}
```

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 (Inversion of Control) and DI (Dependency Injection).

Steps:

Modify the XML Configuration:

```
<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 Definitions -->
  <bean id="bookService" class="com.library.service.BookService">
    <property name="bookRepository" ref="bookRepository"/>
  </bean>

  <bean id="bookRepository" class="com.library.repository.BookRepository"/>
</beans>
```

In this configuration, the bookService bean will have its bookRepository property injected with the bookRepository bean.

Update the BookService Class:

```
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {
    private BookRepository bookRepository;

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

    public void manageBooks() {
        System.out.println("Managing books...");
        bookRepository.saveBook();  // Example usage of
BookRepository
    }
}
```

The setBookRepository method allows Spring to inject the BookRepository instance.

Test the Configuration:

```
package com.library;


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


public class LibraryManagementApplication {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService =
context.getBean(BookService.class);
        bookService.manageBooks();  // Should print "Managing
books..." and "Saving book..."
    }
}
```

Exercise 3: Implementing Logging with Spring AOP

1. Add Spring AOP Dependency

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-aop</artifactId>
    <version>5.3.20</version>
</dependency>
```

Create an Aspect for Logging

LoggingAspect.java

```
package com.library.aspect;

import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Pointcut;

@Aspect
public class LoggingAspect {

    @Pointcut("execution(*
com.library.service.BookService.manageBooks(..))")
    public void manageBooksPointcut() {}
```

```
    @Before("manageBooksPointcut()")
    public void logBefore() {
        System.out.println("Before method execution");
    }

    @After("manageBooksPointcut()")
    public void logAfter() {
        System.out.println("After method execution");
    }
}
```

3. Enable AspectJ Support

Update applicationContext.xml to enable AspectJ support and register the aspect:

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:aop="http://www.springframework.org/schema/aop"

xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop.xsd">

    <bean id="bookService" class="com.library.service.BookService">
```

```
        <property name="bookRepository" ref="bookRepository"/>
    </bean>

    <bean id="bookRepository"
class="com.library.repository.BookRepository"/>

    <!-- Register Logging Aspect -->
    <bean id="loggingAspect"
class="com.library.aspect.LoggingAspect"/>

    <!-- Enable AspectJ Auto Proxying -->
    <aop:aspectj-autoproxy/>
</beans>
```

Exercise 4: Creating and Configuring a Maven Project

- **Create a New Maven Project**
- **Add Spring Dependencies in pom.xml**

```
<dependencies>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-context</artifactId>
<version>5.3.20</version>
</dependency>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-aop</artifactId>
<version>5.3.20</version>
</dependency>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-webmvc</artifactId>
<version>5.3.20</version>
</dependency>
</dependencies>
```

Configure Maven Plugins

Add the Maven Compiler Plugin for Java version 1.8 in pom.xml:

```
<build>
   <plugins>
      <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-compiler-plugin</artifactId>
         <version>3.8.1</version>
         <configuration>
            <source>1.8</source>
            <target>1.8</target>
         </configuration>
      </plugin>
   </plugins>
</build>
```

Exercise 5: Configuring the Spring IoC Container

Create Spring Configuration File:

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

xsi:schemaLocation="http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-
beans.xsd
           http://www.springframework.org/schema/aop
           http://www.springframework.org/schema/aop/spring-aop.xsd">


    <!-- Enable AspectJ auto-proxy support -->
    <aop:aspectj-autoproxy/>


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


    <bean id="bookRepository"
class="com.library.repository.BookRepository"/>
```

```
</beans>
```

BookService Class

```
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {
    private BookRepository bookRepository;

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

    public void manageBooks() {
        System.out.println("Managing books...");
        bookRepository.saveBook();  // Example usage of
BookRepository
    }
}
```

BookRepository Class

```
package com.library.repository;

public class BookRepository {
    public void saveBook() {
        System.out.println("Saving book...");
    }
}
```

LibraryManagementApplication Class

```
package com.library;

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

public class LibraryManagementApplication {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService =
context.getBean(BookService.class);
```

```
    }
}
```

pom.xml (Maven Configuration)

```
<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.library</groupId>
    <artifactId>LibraryManagement</artifactId>
    <version>1.0-SNAPSHOT</version>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.20</version>
        </dependency>
        <dependency>
            <groupId>org.aspectj</groupId>
```

```
            <artifactId>aspectjrt</artifactId>
            <version>1.9.8</version>
        </dependency>
    </dependencies>
</project>
```

Exercise 6: Configuring Beans with Annotations

1. **Enable Component Scanning**

Update applicationContext.xml to include component scanning:

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-
context.xsd">

    <context:component-scan base-package="com.library"/>

    <aop:aspectj-autoproxy/>
</beans>
```

Annotate Classes

BookService.java

```
package com.library.service;
```

```
import org.springframework.stereotype.Service;

@Service
public class BookService {
    private final BookRepository bookRepository;

    public BookService(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void manageBooks() {
        System.out.println("Managing books...");
    }
}
```

BookRepository.java

```
package com.library.repository;
import org.springframework.stereotype.Repository;
@Repository
public class BookRepository {
    public void saveBook() {
        System.out.println("Saving book...");
    }
}
```

Exercise 7: Implementing Constructor and Setter Injection

1. Configure Constructor Injection

Update the applicationContext.xml to configure constructor injection for BookService:

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

xsi:schemaLocation="http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/aop
           http://www.springframework.org/schema/aop/spring-aop.xsd">

    <!-- Enable AspectJ auto-proxy support -->
    <aop:aspectj-autoproxy/>

    <!-- Bean Definitions -->
    <bean id="bookService" class="com.library.service.BookService">
        <constructor-arg ref="bookRepository"/>
    </bean>

    <bean id="bookRepository" class="com.library.repository.BookRepository"/>
```

```
</beans>
```

2. BookService Class with Constructor Injection

Update the BookService class to include a constructor for dependency injection:

```
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {
    private final BookRepository bookRepository;

    // Constructor for Constructor Injection
    public BookService(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void manageBooks() {
        System.out.println("Managing books...");
        bookRepository.saveBook();  // Example usage of BookRepository
    }
}
```

3. Test the Injection

You can test this setup by running the LibraryManagementApplication class from the previous exercises:

```
package com.library;


import org.springframework.context.ApplicationContext;

import
org.springframework.context.support.ClassPathXmlApplicationConte
xt;

import com.library.service.BookService;


public class LibraryManagementApplication {

   public static void main(String[] args) {

       ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

       BookService bookService =
context.getBean(BookService.class);

       bookService.manageBooks();

   }

}
```

Exercise 8: Implementing Basic AOP with Spring

1. Define an Aspect

LoggingAspect.java

```
package com.library.aspect;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class LoggingAspect {

    @Before("execution(* com.library.service.BookService.manageBooks(..))")
    public void logBefore(JoinPoint joinPoint) {
        System.out.println("Before method: " + joinPoint.getSignature().getName());
    }
```

```
    @After("execution(* com.library.service.BookService.manageBooks(..))")
    public void logAfter(JoinPoint joinPoint) {
        System.out.println("After method: " + joinPoint.getSignature().getName());
    }
}
```

Configure the Aspect:

applicationContext.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"
       xmlns:aop="http://www.springframework.org/schema/aop"

xsi:schemaLocation="http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/aop
           http://www.springframework.org/schema/aop/spring-aop.xsd">

    <!-- Enable AspectJ auto-proxy support -->
    <aop:aspectj-autoproxy/>
```

```
    <!-- Bean Definitions -->
    <bean id="bookService" class="com.library.service.BookService">
        <constructor-arg ref="bookRepository"/>
    </bean>

    <bean id="bookRepository"
class="com.library.repository.BookRepository"/>

    <!-- Define Aspect -->
    <bean id="loggingAspect"
class="com.library.aspect.LoggingAspect"/>

</beans>
```

Exercise 9: Creating a Spring Boot Application

1. Create a Spring Boot Project

- Group ID: com.library
- Artifact ID: LibraryManagement
- Dependencies: Spring Web, Spring Data JPA, H2 Database

2. Add Dependencies

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

3. Create application.properties

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=password
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.h2.console.enabled=true
```

4. Define Entities and Repositories

Book.java

```
package com.library.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String title;
    private String author;
```

```
    // Getters and Setters
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }
}
```

BookRepository.java

```
package com.library.repository;

import com.library.entity.Book;
import org.springframework.data.jpa.repository.JpaRepository;

public interface BookRepository extends JpaRepository<Book,
Long> {
}
```

5. Create a REST Controller

BookController.java

```
package com.library.controller;

import com.library.entity.Book;
import com.library.repository.BookRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
```

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

    @Autowired
    private BookRepository bookRepository;

    @GetMapping
    public List<Book> getAllBooks() {
        return bookRepository.findAll();
    }

    @PostMapping
    public Book createBook(@RequestBody Book book) {
        return bookRepository.save(book);
    }

    @GetMapping("/{id}")
    public Book getBookById(@PathVariable Long id) {
        return bookRepository.findById(id).orElse(null);
    }

    @PutMapping("/{id}")
    public Book updateBook(@PathVariable Long id, @RequestBody
Book bookDetails) {
        Book book = bookRepository.findById(id).orElse(null);
```

```
        if (book != null) {
            book.setTitle(bookDetails.getTitle());
            book.setAuthor(bookDetails.getAuthor());
            return bookRepository.save(book);
        }
        return null;
    }

    @DeleteMapping("/{id}")
    public void deleteBook(@PathVariable Long id) {
        bookRepository.deleteById(id);
    }
}
```

6. Run the Application

LibraryManagementApplication.java

```
package com.library;

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

```
@SpringBootApplication
public class LibraryManagementApplication {
    public static void main(String[] args) {
        SpringApplication.run(LibraryManagementApplication.class, args);
    }
}
```