Exercise 1: Configuring a Basic Spring Application

App.java

```
package com.qa.nal;
import com.qa.nal.Service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

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

BookService.java

```
package com.qa.nal.Service;
import com.qa.nal.Repository.BookRepository;

public class BookService {

  public void displayService() {
     System.out.println("BookService is working!");
 }
}
```

BookRepository.java

```
package com.qa.nal.Repository;

public class BookRepository {

    public void displayRepository() {
     System.out.println("BookRepository is working!");
 }
}
```

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

    <bean id="bookService" class="com.qa.nal.Service.BookService"/>
    <bean id="bookRepository" class="com.qa.nal.Repository.BookRepository"/>
</beans>
```

Output

```
BookService is working!
```

Exercise 2: Implementing Dependency Injection

App.java

```
package com.qa.nal;
import com.qa.nal.Service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

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

        // Call the service method to test the dependency injection
        bookService.performService();
    }
}
```

BookService.java

```
package com.qa.nal.Service;

import com.qa.nal.Repository.BookRepository;
public class BookService {
private BookRepository bookRepository;
public void setBookRepository(BookRepository bookRepository) {
    this.bookRepository = bookRepository;
}
public void performService() {
    bookRepository.doSomething();
}
}
```

BookRepository.java

```
package com.qa.nal.Repository;
public class BookRepository {
  public void doSomething() {
  System.out.println("BookRepository: Performing database operation...");
}
}
```

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"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd">
<bean id="bookRepository" class="com.qa.nal.Repository.BookRepository"/>
<bean id="bookService" class="com.qa.nal.Service.BookService">
        <property name="bookRepository" ref="bookRepository"/>
    </bean>
</beans>
```

Output

```
BookRepository: Performing database operation…
```

Exercise 3: Implementing Logging with Spring AOP

App.java

```
package com.qa.nal;
import com.qa.nal.Service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

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

        // Invoke methods
        bookService.addBook();
        bookService.removeBook();
    }
}
```

BookService.java

```
package com.qa.nal.Service;
public class BookService {
  public void addBook() {
    // Simulate adding a book
    System.out.println("Adding a book to the library...");
}
public void removeBook() {
    // Simulate removing a book
    System.out.println("Removing a book from the library...");
}
}
```

BookRepository.java

```
package com.qa.nal.Repository;
import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class BookRepository{
@Around("execution(* com.library.service.*.*(..))")
    public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {
        long startTime = System.currentTimeMillis();

        Object proceed = joinPoint.proceed();

        long executionTime = System.currentTimeMillis() - startTime;

        System.out.println(joinPoint.getSignature() + " executed in " + executionTime
+ "ms");

        return proceed;
```

```
    }
}
```

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 support -->
    <aop:aspectj-autoproxy/>

    <!-- Bean definitions -->
    <bean id="bookService" class="com.qa.nal.Service.BookService"/>
    <bean id="loggingAspect" class="com.qa.nal.Repository.BookRepository"/>
</beans>
```

Output

```
Adding a book to the library...
Removing a book from the library...
```

Exercise 4: Creating and Configuring a Maven Project

```
<dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
    <!-- Spring Context -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.22</version>
    </dependency>

    <!-- Spring AOP -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aop</artifactId>
        <version>5.3.22</version>
    </dependency>

    <!-- Spring WebMVC -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.3.22</version>
    </dependency>
  </dependencies>
```

Exercise 5: Configuring the Spring IoC Container

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"
       xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Bean definition for BookRepository -->
    <bean id="bookRepository" class="com.example.repository.BookRepository" />

    <!-- Bean definition for BookService -->
    <bean id="bookService" class="com.example.service.BookService">
        <!-- Injecting BookRepository into BookService -->
        <property name="bookRepository" ref="bookRepository" />
    </bean>

</beans>
```

App.java

```
 package com.example;

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

import com.example.service.BookService;

public class App
{
    public static void main( String[] args )
    {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        // Retrieve the BookService bean
        BookService bookService = (BookService) context.getBean("bookService");

        // Test the configuration by calling a method on the BookService
        bookService.performService();
    }
}
```

BookService.java

```
package com.example.service;

import com.example.repository.BookRepository;

public class BookService {
  private BookRepository bookRepository;

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

```
    }

    // Example method that uses BookRepository
    public void performService() {
        System.out.println("Service started...");
        bookRepository.someRepositoryMethod();
        System.out.println("Service completed.");
    }
}
```

BookRepository.java

```
  package com.example.repository;

public class BookRepository {
  public void someRepositoryMethod() {
    System.out.println("BookRepository method called.");
}
}
```

Output

```
Service started...
BookRepository method called.
Service completed.
```

Exercise 6: Configuring Beans with Annotations
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: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">

    <!-- Enable component scanning -->
    <context:component-scan base-package="com.example" />

    <!-- Other bean definitions can go here -->

</beans>
```

App.java

```
package com.example;

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

import com.example.service.BookService;

public class App
{
    public static void main( String[] args )
    {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        // Retrieve the BookService bean
        BookService bookService = (BookService) context.getBean("bookService");

        // Test the configuration by calling a method on the BookService
        bookService.performService();
    }
}
```

BookService.java

```
package com.example.service;

import com.example.repository.BookRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class BookService {

    private final BookRepository bookRepository;

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

    // Example method that uses BookRepository
    public void performService() {
        System.out.println("Service started...");
        bookRepository.someRepositoryMethod();
        System.out.println("Service completed.");
    }
}
```

BookRepository.java

```
  package com.example.repository;
@Repository
public class BookRepository {
  public void someRepositoryMethod() {
    System.out.println("BookRepository method called.");
}
}
```

Output

```
Service started...
BookRepository method called.
Service completed.
```

Exercise 7: Implementing Constructor and Setter Injection

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

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

    <!-- Bean for BookService with Constructor and Setter Injection -->
    <bean id="bookService" class="com.example.service.BookService">
        <!-- Constructor Injection -->
        <constructor-arg ref="bookRepository" />

        <!-- Setter Injection (optional, can be removed if not needed) -->
        <property name="bookRepository" ref="bookRepository" />
    </bean>

</beans>
```

App.java

```
 package com.example;

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

import com.example.service.BookService;

public class App
{
    public static void main( String[] args )
    {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        // Retrieve the BookService bean
        BookService bookService = (BookService) context.getBean("bookService");

        // Test the configuration by calling a method on the BookService
        bookService.performService();
```

```
    }
}
```

BookService.java

```
package com.example.service;

import com.example.repository.BookRepository;

    public class BookService {

    private BookRepository bookRepository;
    // Constructor injection
    public BookService(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

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

    public void performService() {
        System.out.println("BookService is performing an operation...");
        bookRepository.saveBook();
    }

}
```

BookRepository.java

```
  package com.example.repository;

public class BookRepository {
    public void saveBook() {
        System.out.println("Book saved to the repository.");
    }
}
```

Output

```
BookService is performing an operation...
Book saved to the repository.
```

Exercise 8: Implementing Basic AOP with Spring

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:context="http://www.springframework.org/schema/context"
        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/context
            http://www.springframework.org/schema/context/spring-context.xsd
            http://www.springframework.org/schema/aop
            http://www.springframework.org/schema/aop/spring-aop.xsd">

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

    <aop:aspectj-autoproxy />

    <!-- Bean for BookRepository (if you have one) -->
    <bean id="bookRepository" class="com.example.repository.BookRepository" />

    <!-- Bean for BookService -->
    <bean id="bookService" class="com.example.service.BookService" />

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

</beans>
```

App.java

```
package com.example;

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

import com.example.service.BookService;

public class App
{
    public static void main( String[] args )
    {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        // Retrieve the BookService bean
        BookService bookService = (BookService) context.getBean("bookService");

       bookService.addBook("Harry Potter");
        bookService.deleteBook("Harry Potter");

    }
}
```

BookService.java

```
package com.example.service;

import com.example.repository.BookRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class BookService {

     public void addBook(String bookName) {
         System.out.println("Book added: " + bookName);
     }

     public void deleteBook(String bookName) {
         System.out.println("Book deleted: " + bookName);
     }

}
```

BookRepository.java

```
  package com.example.repository;
@Repository
public class BookRepository {
  public void saveBook() {
        System.out.println("Book saved to the repository.");
    }
}
```

LoggingAspect.java

```
 package com.example.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;

@Component
@Aspect
public class LoggingAspect {
  @Before("execution(* com.example.service.BookService.*(..))")
  public void logBefore(JoinPoint joinPoint) {
      System.out.println("Before method: " + joinPoint.getSignature().getName());
  }

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

```
}
```

Output

```
Before method: addBook
Book added: Harry Potter
After method: addBook
Before method: deleteBook
Book deleted: Harry Potter
After method: deleteBook
```

Exercise 9: Creating a Spring Boot Application