Exercise 1: Configuring a Basic Spring Application

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>

    <dependencies>
        <!-- Spring Core -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>5.3.8</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.8</version>
        </dependency>
    </dependencies>
</project>
```

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">

    <!-- Define BookService bean -->
    <bean id="bookService" class="com.library.service.BookService"/>

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

BookService.java

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

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

BookRepository.java

```
package com.library.repository;

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

Main.java

```
package com.library;

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

public class Main {
    public static void main(String[] args) {
        // Load the Spring context
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        // Get the BookService bean and test it
        BookService bookService = (BookService) context.getBean("bookService");
        bookService.displayService();

        // Get the BookRepository bean and test it
        BookRepository bookRepository = (BookRepository) context.getBean("bookRepository");
        bookRepository.displayRepository();
    }
}
```

Exercise 2: Implementing Dependency 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">

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

   <!-- Define BookService bean and inject BookRepository -->
   <bean id="bookService" class="com.library.service.BookService">
      <property name="bookRepository" ref="bookRepository"/>
   </bean>
</beans>
```

BookService.java

```
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 displayService() {
      System.out.println("BookService is working...");
      bookRepository.displayRepository();
   }
}
```

BookRepository.java

```
package com.library.repository;

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

```
    }
}
```

Main.java

```
package com.library;

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

public class Main {
    public static void main(String[] args) {
        // Load the Spring context
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        // Get the BookService bean and test it
        BookService bookService = (BookService) context.getBean("bookService");
        bookService.displayService();
    }
}
```

Exercise 3: Implementing Logging with Spring AOP

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>

    <dependencies>
        <!-- Spring Core -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>5.3.8</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
```

```
        <artifactId>spring-context</artifactId>
        <version>5.3.8</version>
    </dependency>
    <!-- Spring AOP -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aop</artifactId>
        <version>5.3.8</version>
    </dependency>
    <!-- AspectJ Weaver -->
    <dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjweaver</artifactId>
        <version>1.9.7</version>
    </dependency>
  </dependencies>
</project>
```

LoggingAspect.java

```
package com.library.aspect;

import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;

@Aspect
public class LoggingAspect {
  @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/>

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

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

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

Main.java

```
package com.library;

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

public class Main {
    public static void main(String[] args) {
        // Load the Spring context
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        // Get the BookService bean and test it
        BookService bookService = (BookService) context.getBean("bookService");
        bookService.displayService();
    }
}
```

Exercise 4: Creating and Configuring a Maven Project

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.example</groupId>
<artifactId>LibraryManagement</artifactId>
<version>1.0-SNAPSHOT</version>

<properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>

<dependencies>
    <!-- Spring Context -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.20</version>
    </dependency>

    <!-- Spring AOP -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aop</artifactId>
        <version>5.3.20</version>
    </dependency>

    <!-- Spring WebMVC -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.3.20</version>
    </dependency>
</dependencies>

<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>
</project>
```

Exercise 5: Configuring the Spring IoC Container Scenario:

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">

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

    <!-- Define BookService bean and inject BookRepository -->
    <bean id="bookService" class="com.example.librarymanagement.service.BookService">
        <property name="bookRepository" ref="bookRepository"/>
    </bean>

</beans>
```

BookService.java

```
package com.example.librarymanagement.service;

import com.example.librarymanagement.repository.BookRepository;

public class BookService {
    private BookRepository bookRepository;

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

    // Business logic methods
    public void performService() {
        System.out.println("BookService is working with BookRepository: " + bookRepository);
    }
}
```

BookRepository.java

```
package com.example.librarymanagement.repository;

public class BookRepository {
    // Repository methods
    public void performRepositoryAction() {
        System.out.println("BookRepository action performed");
    }
}
```

Main.java

```
package com.example.librarymanagement;

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

public class Main {
    public static void main(String[] args) {
        // Load Spring context from XML configuration
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

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

        // Test the configuration
        bookService.performService();
    }
}
```

Exercise 6: Configuring Beans with Annotations Scenario:

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.librarymanagement"/>

</beans>
```

Bookservice.java

```
package com.example.librarymanagement.service;

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

@Service
public class BookService {

    private BookRepository bookRepository;

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

    // Business logic methods
    public void performService() {
        System.out.println("BookService is working with BookRepository: " + bookRepository);
    }
}
```

BookRepository.java

```
package com.example.librarymanagement.repository;

import org.springframework.stereotype.Repository;

@Repository
public class BookRepository {
    // Repository methods
    public void performRepositoryAction() {
        System.out.println("BookRepository action performed");
    }
}
```

LibraryManagementApplication.java

```
package com.example.librarymanagement;

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

public class LibraryManagementApplication {
    public static void main(String[] args) {
        // Load Spring context from XML configuration
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

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

        // Test the configuration
        bookService.performService();
    }
}
```

Exercise 7: Implementing Constructor and Setter Injection Scenario:

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.librarymanagement"/>

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

    <!-- Define BookService bean with constructor injection -->
    <bean id="bookService" class="com.example.librarymanagement.service.BookService">
        <constructor-arg ref="bookRepository"/>
        <!-- Configure setter injection -->
```

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

</beans>
```

Bookserice.java

```
package com.example.librarymanagement.service;

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

@Service
public class BookService {
    private BookRepository bookRepository;

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

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

    // Business logic methods
    public void performService() {
        System.out.println("BookService is working with BookRepository: " + bookRepository);
    }
}
```

BookRepository.java

```
package com.example.librarymanagement.repository;

import org.springframework.stereotype.Repository;

@Repository
public class BookRepository {
    // Repository methods
    public void performRepositoryAction() {
        System.out.println("BookRepository action performed");
```

```
    }
}
```

LibraryManagementApplication.java

```
package com.example.librarymanagement;

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

public class LibraryManagementApplication {
    public static void main(String[] args) {
        // Load Spring context from XML configuration
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

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

        // Test the configuration
        bookService.performService();
    }
}
```

Exercise 8: Implementing Basic AOP with Spring Scenario:

LogginAspect.java

```
 package com.example.librarymanagement.aspect;

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 {

    // Advice method for logging before method execution
    @Before("execution(* com.example.librarymanagement.service.BookService.*(..))")
    public void logBefore() {
        System.out.println("LoggingAspect: Before method execution");
    }
```

```
    // Advice method for logging after method execution
    @After("execution(* com.example.librarymanagement.service.BookService.*(..))")
    public void logAfter() {
        System.out.println("LoggingAspect: After method execution");
    }
}
```

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">

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

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

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

    <!-- Define BookService bean with constructor injection -->
    <bean id="bookService" class="com.example.librarymanagement.service.BookService">
        <constructor-arg ref="bookRepository"/>
        <!-- Configure setter injection -->
        <property name="bookRepository" ref="bookRepository"/>
    </bean>

</beans>
```

LibraryMAnagementApplication.java

```
package com.example.librarymanagement;

import com.example.librarymanagement.service.BookService;
import org.springframework.context.ApplicationContext;
```

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

public class LibraryManagementApplication {
    public static void main(String[] args) {
        // Load Spring context from XML configuration
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

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

        // Test the configuration
        bookService.performService();
    }
}
```

Exercise 9: Creating a Spring Boot Application Scenario:

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.example</groupId>
    <artifactId>librarymanagement</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>LibraryManagement</name>
    <description>Library Management System</description>
    <packaging>jar</packaging>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.5</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-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>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

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

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
```

Book.java

```
package com.example.librarymanagement.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.example.librarymanagement.repository;

import com.example.librarymanagement.entity.Book;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
```

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

BookController.java

```
package com.example.librarymanagement.controller;

import com.example.librarymanagement.entity.Book;
import com.example.librarymanagement.repository.BookRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

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

    @Autowired
    private BookRepository bookRepository;

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

    @GetMapping("/{id}")
    public ResponseEntity<Book> getBookById(@PathVariable Long id) {
        Optional<Book> book = bookRepository.findById(id);
        if (book.isPresent()) {
            return ResponseEntity.ok(book.get());
        } else {
            return ResponseEntity.notFound().build();
        }
    }

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

```
    @PutMapping("/{id}")
    public ResponseEntity<Book> updateBook(@PathVariable Long id, @RequestBody Book
bookDetails) {
        Optional<Book> book = bookRepository.findById(id);
        if (book.isPresent()) {
            Book existingBook = book.get();
            existingBook.setTitle(bookDetails.getTitle());
            existingBook.setAuthor(bookDetails.getAuthor());
            return ResponseEntity.ok(bookRepository.save(existingBook));
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteBook(@PathVariable Long id) {
        Optional<Book> book = bookRepository.findById(id);
        if (book.isPresent()) {
            bookRepository.delete(book.get());
            return ResponseEntity.noContent().build();
        } else {
            return ResponseEntity.notFound().build();
        }
    }
}
```

LibraryManagementApplication.java

```
package com.example.librarymanagement;

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);
    }
}
```