Spring Core and Maven

Exercise 1: Configuring a Basic Spring Application

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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>
    <packaging>jar</packaging>

    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <spring.version>5.3.21</spring.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>${spring.version}</version>
        </dependency>
```

```
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
            <version>${spring.version}</version>
        </dependency>
    </dependencies>
</project>
```

src/main/resources/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.library.repository.BookRepository"/>

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

src/main/java/com/library/repository/BookRepository.java

```
package com.library.repository;
import java.util.ArrayList;
import java.util.List;
public class BookRepository {
    private List<String> books;
    public BookRepository() {
        this.books = new ArrayList<>();
        books.add("Spring Framework Guide");
        books.add("Java Programming");
        books.add("Maven Build Tool");
```

```
    }
    public List<String> getAllBooks() {
        return books;
    }
    public void addBook(String book) {
        books.add(book);
    }
    public void displayRepositoryInfo() {
        System.out.println("BookRepository initialized with " + books.size() + " books");
    }
}
```

src/main/java/com/library/LibraryManagementApplication.java

```
package com.library;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.library.service.BookService;
import com.library.repository.BookRepository;
public class LibraryManagementApplication {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService = (BookService) context.getBean("bookService");
        BookRepository bookRepository = (BookRepository)
context.getBean("bookRepository");
        System.out.println("Spring Application Context loaded successfully!");
        bookRepository.displayRepositoryInfo();
        bookService.displayServiceInfo();
        bookService.displayAllBooks();
    }
}
```

src/main/java/com/library/service/BookService.java

```
package com.library.service;
import com.library.repository.BookRepository;
import java.util.List;
public class BookService {
    private BookRepository bookRepository;
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }
    public void displayAllBooks() {
        List<String> books = bookRepository.getAllBooks();
        System.out.println("Available Books:");
        for (String book : books) {
            System.out.println("- " + book);
        }
    }
    public void addNewBook(String bookName) {
        bookRepository.addBook(bookName);
        System.out.println("Book added: " + bookName);
    }
    public void displayServiceInfo() {
        System.out.println("BookService is ready to manage books");
    }
}
```

Output:

```
Spring Application Context loaded successfully!
BookRepository initialized with 3 books
BookService is ready to manage books
Available Books:
- Spring Framework Guide
- Java Programming
- Maven Build Tool
```

Exercise 2: Implementing Dependency Injection

src/main/resources/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.library.repository.BookRepository"/>
   <bean id="bookService" class="com.library.service.BookService">
       <property name="bookRepository" ref="bookRepository"/>
   </bean>
</beans>
```

src/main/java/com/library/service/BookService.java

```
package com.library.service;
import com.library.repository.BookRepository;
import java.util.List;
public class BookService {
   private BookRepository bookRepository;
   public void setBookRepository(BookRepository bookRepository) {
       this.bookRepository = bookRepository;
```

```
    System.out.println("BookRepository dependency injected successfully");
}
public void displayAllBooks() {
    if (bookRepository != null) {
        List<String> books = bookRepository.getAllBooks();
        System.out.println("Available Books from injected repository:");
        for (String book : books) {
            System.out.println("- " + book);
        }
    } else {
        System.out.println("BookRepository is not injected");
    }
}
public void addNewBook(String bookName) {
    if (bookRepository != null) {
        bookRepository.addBook(bookName);
        System.out.println("Book added successfully: " + bookName);
    }
}
public void testDependencyInjection() {
    System.out.println("Testing dependency injection...");
    if (bookRepository != null) {
        System.out.println("Dependency injection working correctly!");
    } else {
        System.out.println("Dependency injection failed!");
    }
```

```
    }
}
```

src/main/java/com/library/LibraryManagementApplication.java

```
package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class LibraryManagementApplication {

    public static void main(String[] args) {

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

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

        System.out.println("Testing Dependency Injection in Spring");

        bookService.testDependencyInjection();

        bookService.displayAllBooks();

        bookService.addNewBook("Design Patterns");

        bookService.displayAllBooks();

    }

}
```

Output:

```
BookRepository dependency injected successfully
Testing Dependency Injection in Spring
Testing dependency injection...
Dependency injection working correctly!
Available Books from injected repository:
- Spring Framework Guide
- Java Programming
- Maven Build Tool
Book added successfully: Design Patterns
Available Books from injected repository:
- Spring Framework Guide
- Java Programming
- Maven Build Tool
- Design Patterns
```

Exercise 4: Creating and Configuring a Maven Project

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<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>

    <packaging>jar</packaging>

    <properties>

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

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

        <spring.version>5.3.21</spring.version>

    </properties>

    <dependencies>
```

```
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aop</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjweaver</artifactId>
        <version>1.9.7</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>
```

src/main/java/com/library/config/MavenConfigTest.java

```
package com.library.config;
public class MavenConfigTest {
    public static void main(String[] args) {
        System.out.println("Maven Project Configuration Test");
        System.out.println("================================");
        String javaVersion = System.getProperty("java.version");
        String javaHome = System.getProperty("java.home");
        System.out.println("Java Version: " + javaVersion);
        System.out.println("Java Home: " + javaHome);
        System.out.println("\nMaven Dependencies Check:");
        try {
            Class.forName("org.springframework.context.ApplicationContext");
            System.out.println("✓ Spring Context dependency loaded");
        } catch (ClassNotFoundException e) {
            System.out.println("✗ Spring Context dependency not found");
        }
        try {
            Class.forName("org.springframework.aop.Advisor");
            System.out.println("✓ Spring AOP dependency loaded");
```

```
        } catch (ClassNotFoundException e) {
            System.out.println("✗ Spring AOP dependency not found");
        }
        try {
            Class.forName("org.springframework.web.servlet.DispatcherServlet");
            System.out.println("✓ Spring WebMVC dependency loaded");
        } catch (ClassNotFoundException e) {
            System.out.println("✗ Spring WebMVC dependency not found");
        }
        System.out.println("\nMaven project configured successfully!");
    }
}
```

Output:

```
Maven Project Configuration Test
================================
Java Version: 1.8.0_291
Java Home: C:\Program Files\Java\jdk1.8.0_291\jre

Maven Dependencies Check:
✓ Spring Context dependency loaded
✓ Spring AOP dependency loaded
✓ Spring WebMVC dependency loaded

Maven project configured successfully!
```

Exercise 5: Configuring the Spring IoC Container

src/main/resources/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.library.repository.BookRepository">
```

```
        <property name="repositoryName" value="Central Library Repository"/>
    </bean>
    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository"/>
        <property name="serviceName" value="Library Management Service"/>
    </bean>
</beans>
```

src/main/java/com/library/LibraryManagementApplication.java

```
package com.library.repository;
import java.util.ArrayList;
import java.util.List;
public class BookRepository {
    private List<String> books;
    private String repositoryName;
    public BookRepository() {
        this.books = new ArrayList<>();
        books.add("Spring Framework Guide");
        books.add("Java Programming");
        books.add("Maven Build Tool");
        books.add("IoC Container Patterns");
    }
    public void setRepositoryName(String repositoryName) {
        this.repositoryName = repositoryName;
    }
    public String getRepositoryName() {
        return repositoryName;
    }
    public List<String> getAllBooks() {
        return books;
    }
```

```
    public void addBook(String book) {
        books.add(book);
    }
    public void displayRepositoryInfo() {
        System.out.println("Repository: " + repositoryName);
        System.out.println("Total books: " + books.size());
    }
}
```

src/main/java/com/library/service/BookService.java

```
package com.library.service;
import com.library.repository.BookRepository;
import java.util.List;
public class BookService {
    private BookRepository bookRepository;
    private String serviceName;
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }
    public void setServiceName(String serviceName) {
        this.serviceName = serviceName;
    }
    public String getServiceName() {
        return serviceName;
    }
    public void displayServiceInfo() {
        System.out.println("Service: " + serviceName);
        if (bookRepository != null) {
            bookRepository.displayRepositoryInfo();
        }
    }
    public void performLibraryOperations() {
        System.out.println("\nPerforming library operations...");
```

```
        displayAllBooks();
        addNewBook("Spring IoC Container");
        System.out.println("Book count after addition: " + bookRepository.getAllBooks().size());
    }
    public void displayAllBooks() {
        List<String> books = bookRepository.getAllBooks();
        System.out.println("\nAvailable Books:");
        for (int i = 0; i < books.size(); i++) {
            System.out.println((i + 1) + ". " + books.get(i));
        }
    }
    public void addNewBook(String bookName) {
        bookRepository.addBook(bookName);
        System.out.println("Added new book: " + bookName);
    }
}
```

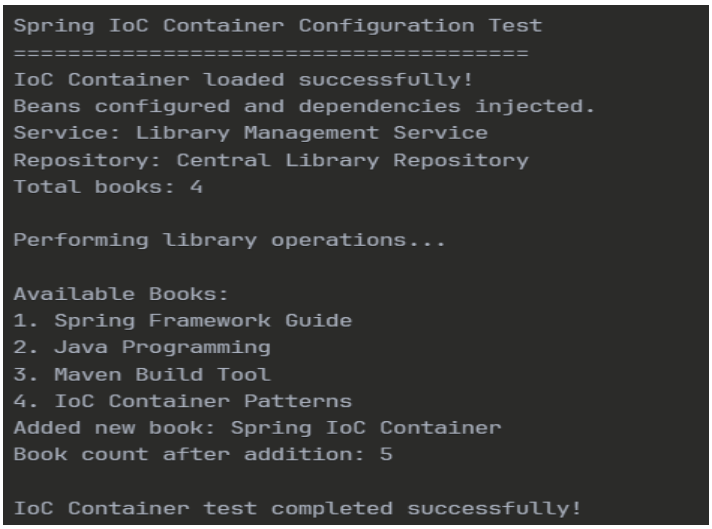
src/main/java/com/library/LibraryManagementApplication.java

```
package com.library;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.library.service.BookService;
public class LibraryManagementApplication {
    public static void main(String[] args) {
        System.out.println("Spring IoC Container Configuration Test");
        System.out.println("======================================");
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService = (BookService) context.getBean("bookService");
        System.out.println("IoC Container loaded successfully!");
        System.out.println("Beans configured and dependencies injected.");

        bookService.displayServiceInfo();
```

```
        bookService.performLibraryOperations();
        System.out.println("\nIoC Container test completed successfully!");
    }
}
```

Output:



Exercise 6: Configuring Beans with Annotations

src/main/resources/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">
    <context:component-scan base-package="com.library"/>
```

```
</beans>
```

src/main/java/com/library/repository/BookRepository.java

```
package com.library.repository;
import org.springframework.stereotype.Repository;
import java.util.ArrayList;
import java.util.List;
@Repository
public class BookRepository {
    private List<String> books;
    public BookRepository() {
        this.books = new ArrayList<>();
        books.add("Spring Framework Guide");
        books.add("Java Programming");
        books.add("Maven Build Tool");
        books.add("Annotation Configuration");
        System.out.println("BookRepository bean created using @Repository annotation");
    }
    public List<String> getAllBooks() {
        return books;
    }
    public void addBook(String book) {
        books.add(book);
    }
    public int getBookCount() {
        return books.size();
    }

    public void displayRepositoryInfo() {
        System.out.println("Repository initialized with " + books.size() + " books");
    }
}
```

src/main/java/com/library/service/BookService.java

```
package com.library.service;
import com.library.repository.BookRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class BookService {
    @Autowired
    private BookRepository bookRepository;
    public BookService() {
        System.out.println("BookService bean created using @Service annotation");
    }
    public void displayAllBooks() {
        List<String> books = bookRepository.getAllBooks();
        System.out.println("\nBooks managed by annotated service:");
        for (int i = 0; i < books.size(); i++) {
            System.out.println((i + 1) + ". " + books.get(i));
        }
    }
    public void addNewBook(String bookName) {
        bookRepository.addBook(bookName);
        System.out.println("Book added through annotated service: " + bookName);
    }
    public void performAnnotationTest() {
        System.out.println("\nTesting annotation-based configuration:");
        displayAllBooks();
        addNewBook("Spring Annotations Guide");
        System.out.println("Total books after addition: " + bookRepository.getBookCount());
    }
    public void displayConfigurationInfo() {
        System.out.println("Service configured using annotations:");
```

```
        System.out.println("- @Service annotation on BookService");
        System.out.println("- @Repository annotation on BookRepository");
        System.out.println("- @Autowired annotation for dependency injection");
        bookRepository.displayRepositoryInfo();
    }
}
```

src/main/java/com/library/LibraryManagementApplication.java

```
package com.library;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.library.service.BookService;
public class LibraryManagementApplication {
    public static void main(String[] args) {
        System.out.println("Spring Annotation-Based Configuration Test");
        System.out.println("==========================================");
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService = context.getBean(BookService.class);
        System.out.println("\nComponent scanning completed!");
        System.out.println("Beans created and autowired successfully.\n");

        bookService.displayConfigurationInfo();
        bookService.performAnnotationTest();
        System.out.println("\nAnnotation-based configuration test completed!");
    }
}
```

Output:

```
Spring Annotation-Based Configuration Test
========================================
BookRepository bean created using @Repository annotation
BookService bean created using @Service annotation

Component scanning completed!
Beans created and autowired successfully.

Service configured using annotations:
- @Service annotation on BookService
- @Repository annotation on BookRepository
- @Autowired annotation for dependency injection
Repository initialized with 4 books

Testing annotation-based configuration:

Books managed by annotated service:
1. Spring Framework Guide
2. Java Programming
3. Maven Build Tool
4. Annotation Configuration
Book added through annotated service: Spring Annotations Guide
Total books after addition: 5

Annotation-based configuration test completed!
```

Exercise 7: Implementing Constructor and Setter Injection

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/beanshttp://www.springframework.org/schema/beans/spring-beans.xsd">
   <bean id="bookRepository" class="com.library.repository.BookRepository"/>
   <bean id="bookService" class="com.library.service.BookService">
      <constructor-arg ref="bookRepository"/>
   </bean>
</beans>
```

com/library/service/BookService.java

```
package com.library.service;
import com.library.repository.BookRepository;
public class BookService {
   private BookRepository bookRepository;
   public BookService(BookRepository bookRepository) {
      this.bookRepository = bookRepository;
      System.out.println("BookService created with constructor injection.");
   }
   public void setBookRepository(BookRepository bookRepository) {
```

```
        this.bookRepository = bookRepository;
        System.out.println("BookRepository set using setter injection.");
    }
    public void listBooks() {
        bookRepository.getAllBooks();
    }
}
```

com/library/repository/BookRepository.java

```
package com.library.repository;
public class BookRepository {
    public void getAllBooks() {
        System.out.println("Retrieving all books from the database.");
    }
}
```

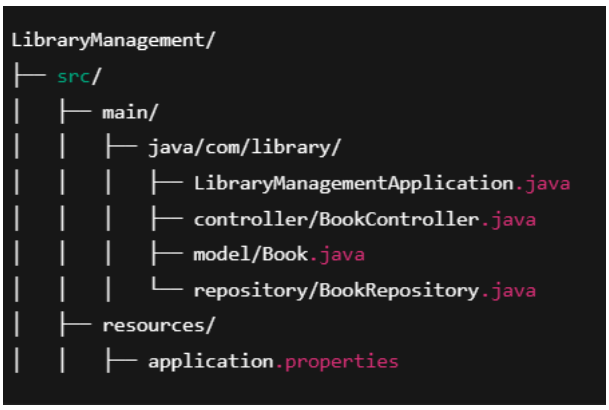
LibraryManagementApplication.java

```
package com.library;
import com.library.service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class LibraryManagementApplication {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
        BookService bookService = (BookService) context.getBean("bookService");
        System.out.println("BookService instance: " + bookService);
        bookService.listBooks();
    }
}
```

Output:

```
BookService created with constructor injection.
BookService instance: com.library.service.BookService@<some_hash_code>
Retrieving all books from the database.
```

Exercise 9: Creating a Spring Boot Application



pom.xml

```
<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>
    </dependency>
</dependencies>
```

Book.java (Entity)

```
package com.library.model;
import jakarta.persistence.*;
@Entity
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String title;
```

```
    private String author;
    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.model.Book;
import org.springframework.data.jpa.repository.JpaRepository;
public interface BookRepository extends JpaRepository<Book, Long> {
}
```

BookController.java

```
package com.library.controller;
import com.library.model.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);
    }
}
```

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);
    }
}
```