**Spring Core Maven Exercises - Solution**

**Exercise 1: Configuring a Basic Spring Application**

Solution:

1. \*\*Project Setup:\*\*

```xml

<!-- pom.xml -->

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.10</version>

</dependency>

</dependencies>

2. XML Configuration:

```xml

<!-- applicationContext.xml -->

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

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

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

</bean>

**Exercise 2: Implementing Dependency Injection**

Solution:

1. Setter Injection:

```java

// BookService.java

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

}

```

**Exercise 3: Implementing Logging with Spring AOP**

Solution:

1. Add AOP Dependency:

```xml

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.10</version>

</dependency>

```

2. Logging Aspect:

```java

@Aspect

public class LoggingAspect {

@Before("execution(\* com.library.service.\*.\*(..))")

public void logBefore(JoinPoint joinPoint) {

System.out.println("Method executing: " + joinPoint.getSignature());

}

}

**Exercise 4: Maven Project Setup**

Solution:

```xml

<!-- pom.xml -->

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.10</version>

</dependency>

</dependencies>

```

**Exercise 5: Spring IoC Container**

Solution:

```xml

<!-- applicationContext.xml -->

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

<constructor-arg ref="bookRepository"/>

</bean>

```

**Exercise 6: Annotation-Based Configuration**

Solution:

1. Enable Scanning:

```xml

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

```

2. Annotate Classes:

```java

@Service

public class BookService { ... }

@Repository

public class BookRepository { ... }

```

**Exercise 7: Constructor and Setter Injection**

Solution:

```java

// Constructor Injection

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

// Setter Injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

```

**Exercise 8: Basic AOP Implementation**

Solution:

```java

@Aspect

public class TransactionAspect {

@Around("execution(\* com.library.service.\*.\*(..))")

public Object manageTransaction(ProceedingJoinPoint pjp) throws Throwable {

System.out.println("Starting transaction");

Object result = pjp.proceed();

System.out.println("Committing transaction");

return result;

}

}

```

**Exercise 9: Spring Boot Application**

Solution:

1. Main Class:

```java

@SpringBootApplication

public class LibraryApplication {

public static void main(String[] args) {

SpringApplication.run(LibraryApplication.class, args);

}

}

```

2.Controller:

```java

@RestController

@RequestMapping("/books")

public class BookController {

@GetMapping

public List<Book> getAllBooks() { ... }

}

```