# Spring Core and Maven Exercises - Solutions

## Exercise 1: Configuring a Basic Spring Application

1. pom.xml (Spring Core dependency):  
<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.27</version>  
</dependency>  
  
2. applicationContext.xml (src/main/resources):  
<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>  
  
3. Java Classes:  
package com.library.repository;  
public class BookRepository {  
 public void saveBook(String title) {  
 System.out.println("Saving book: " + title);  
 }  
}  
  
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 addBook(String title) {  
 bookRepository.saveBook(title);  
 }  
}  
  
Main Class:  
ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
BookService service = context.getBean(BookService.class);  
service.addBook("Spring Framework Guide");

## Exercise 2: Implementing Dependency Injection

Update applicationContext.xml to wire BookRepository into BookService using setter injection (as above).  
  
Ensure BookService has:  
public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
}

## Exercise 3: Implementing Logging with Spring AOP

1. pom.xml Dependency:  
<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.27</version>  
</dependency>  
  
2. LoggingAspect.java:  
@Aspect  
public class LoggingAspect {  
 @Around("execution(\* com.library.service.\*.\*(..))")  
 public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {  
 long start = System.currentTimeMillis();  
 Object proceed = joinPoint.proceed();  
 long duration = System.currentTimeMillis() - start;  
 System.out.println(joinPoint.getSignature() + " executed in " + duration + "ms");  
 return proceed;  
 }  
}  
  
3. applicationContext.xml updates:  
<aop:aspectj-autoproxy/>  
<bean id="loggingAspect" class="com.library.aspect.LoggingAspect"/>

## Exercise 4: Creating and Configuring a Maven Project

Create new Maven project: LibraryManagement  
  
pom.xml:  
<dependencies>  
 <dependency> <!-- Spring Context --> </dependency>  
 <dependency> <!-- Spring AOP --> </dependency>  
 <dependency> <!-- Spring WebMVC --> </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>

## Exercise 6: Configuring Beans with Annotations

1. Annotations:  
@Service  
public class BookService {}  
  
@Repository  
public class BookRepository {}  
  
2. applicationContext.xml:  
<context:component-scan base-package="com.library"/>  
<context:annotation-config/>

## Exercise 7: Implementing Constructor and Setter Injection

BookService.java:  
public class BookService {  
 private BookRepository bookRepository;  
 public BookService(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
}  
  
applicationContext.xml:  
<bean id="bookRepository" class="com.library.repository.BookRepository"/>  
<bean id="bookService" class="com.library.service.BookService">  
 <constructor-arg ref="bookRepository"/>  
 <property name="bookRepository" ref="bookRepository"/>  
</bean>

## Exercise 9: Creating a Spring Boot Application

1. Use Spring Initializr to create LibraryManagement.  
  
2. application.properties:  
spring.datasource.url=jdbc:h2:mem:librarydb  
spring.datasource.driverClassName=org.h2.Driver  
  
3. Book.java (entity), BookRepository.java (interface), BookController.java (@RestController with CRUD mappings)  
  
Run: LibraryManagementApplication.java (with @SpringBootApplication)