Spring Framework Exercises Solutions

Exercise 1: Configuring a Basic Spring Application

Set Up a Spring Project:
- Create a Maven project named LibraryManagement.
- Add Spring Core dependencies in the pom.xml file:

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>


  <properties>
     <spring.version>5.3.20</spring.version>
  </properties>


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

Configure the Application Context:
Create 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/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>
```

Define Service and Repository Classes:
BookRepository.java:

```
package com.library.repository;


public class BookRepository {
  // Repository methods
}
```

BookService.java:

```
package com.library.service;


import com.library.repository.BookRepository;


public class BookService {
  private BookRepository bookRepository;


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

Run the Application:
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 = context.getBean("bookService", BookService.class);
     System.out.println("Spring context loaded successfully!");
  }
}
```

Exercise 2: Implementing Dependency Injection

- Modify applicationContext.xml to wire BookRepository into BookService.
- Ensure BookService has a setter for BookRepository.
- Run the main application to test DI.

Exercise 3: Implementing Logging with Spring AOP

Add Spring AOP Dependency:
pom.xml:

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

Create Aspect:
LoggingAspect.java:

```
package com.library.aspect;


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 LoggingAspect {
  @Around("execution(* com.library.service.*.*(..))")
  public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {
      long start = System.currentTimeMillis();
      Object proceed = joinPoint.proceed();
      long executionTime = System.currentTimeMillis() - start;
      System.out.println(joinPoint.getSignature() + " executed in " + executionTime + "ms");
      return proceed;
  }
}
```

Update applicationContext.xml:

```
<beans ... >
```

```
 <context:component-scan base-package="com.library"/>
 <aop:aspectj-autoproxy/>


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


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

Exercise 4: Creating and Configuring a Maven Project

Create a Maven Project and add dependencies:

pom.xml:

```
<dependencies>
 <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>${spring.version}</version>
 </dependency>
 <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>${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>
</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 5: Configuring the Spring IoC Container

Same as Exercise 1: Define beans in applicationContext.xml, inject dependencies, and load context in the main class.

Exercise 6: Configuring Beans with Annotations

Update applicationContext.xml:

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

BookRepository.java:

```
@Repository
public class BookRepository { }
```

BookService.java:

```
@Service
public class BookService {
  @Autowired
  public void setBookRepository(BookRepository bookRepository) {
     this.bookRepository = bookRepository;
  }
}
```

Exercise 7: Implementing Constructor and Setter Injection

BookService.java:

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

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

applicationContext.xml:

```
<bean id="bookService" class="com.library.service.BookService">
  <constructor-arg ref="bookRepository"/>
</bean>
```

Exercise 8: Implementing Basic AOP with Spring

LoggingAspect.java:

```
@Before("execution(* com.library.service.*.*(..))")
public void logBefore(JoinPoint joinPoint) {
  System.out.println("Before executing: " + joinPoint.getSignature().getName());
}

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

applicationContext.xml:

```
<context:component-scan base-package="com.library"/>
<aop:aspectj-autoproxy/>
```

Exercise 9: Creating a Spring Boot Application

Create Spring Boot Project using Spring Initializr

pom.xml:
Add spring-boot-starter-web, spring-boot-starter-data-jpa, h2 dependencies

application.properties:

```
spring.datasource.url=jdbc:h2:mem:librarydb
spring.jpa.hibernate.ddl-auto=update
```

Book.java:

```
@Entity
public class Book {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;
  private String title;
  private String author;
  private String isbn;
}
```

BookRepository.java:

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

BookController.java:

```
@RestController
@RequestMapping("/api/books")
public class BookController {
  @Autowired
  private BookRepository bookRepository;


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

LibraryManagementApplication.java:

```
@SpringBootApplication
public class LibraryManagementApplication {
  public static void main(String[] args) {
     SpringApplication.run(LibraryManagementApplication.class, args);
  }
}
```