Week 3 – SpringCoreAndMaven_Mandatory_HandsOn
Skill: Spring Core and Maven
Candidate: Projita Kar
Superset ID: 6407705
Type: Mandatory Hands-On

Exercise 1: Configuring a Basic Spring Application

Scenario:
Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

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

```
<?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>

   <properties>
       <maven.compiler.source>21</maven.compiler.source>
       <maven.compiler.target>21</maven.compiler.target>
       <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
   </properties>
   <dependencies>
       <dependency>
           <groupId>org.springframework</groupId>
           <artifactId>spring-context</artifactId>
           <version>5.3.34</version>
       </dependency>
   </dependencies>

</project>
```

2. Configure the Application Context:
- Create an XML configuration file named applicationContext.xml in the src/main/resources directory.
- Define beans for BookService and BookRepository in the XML file.

```
<?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="bookService" class="com.library.service.BookService" />
   <bean id="bookRepository"
class="com.library.repository.BookRepository" />

</beans>
```

3. Define Service and Repository Classes:
   - Create a package com.library.service and add a class BookService.

```
package com.library.service;

public class BookService {
   public void greet() {
       System.out.println("Hello from BookService!");
   }
}
```

   - Create a package com.library.repository and add a class BookRepository.

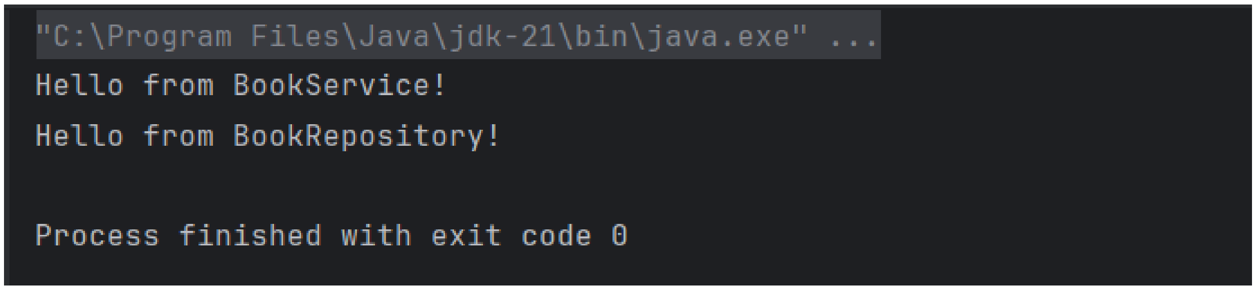
```
package com.library.repository;

public class BookRepository {
   public void showMessage() {
       System.out.println("Hello from BookRepository!");
   }
}
```

4. Run the Application:
   - Create a main class to load the Spring context and test the configuration.

```

package com.library.main;

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

```

```
        BookService bookService = context.getBean("bookService",
BookService.class);
        bookService.greet();

        BookRepository bookRepository = context.getBean("bookRepository",
BookRepository.class);
        bookRepository.showMessage();
    }
}
```

Output:-



```
"C:\Program Files\Java\jdk-21\bin\java.exe" ...
Hello from BookService!
Hello from BookRepository!

Process finished with exit code 0
```

Exercise 2: Implementing Dependency Injection

Scenario:
In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

Steps:
1. Modify the XML Configuration:

- Update applicationContext.xml to wire BookRepository into BookService.

```
<?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>
```

2. Update the BookService Class:

- Ensure that BookService class has a setter method for BookRepository.

  BookService class:-

```java
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {
   private BookRepository bookRepository;

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

   public void listBooks() {
       System.out.println("BookService: Listing books");
       bookRepository.fetchBooks();
   }
}
```

  BookRepository class:-

```java
package com.library.repository;

public class BookRepository {
   public void fetchBooks() {
       System.out.println("Fetching books from the repository...");
   }
}
```

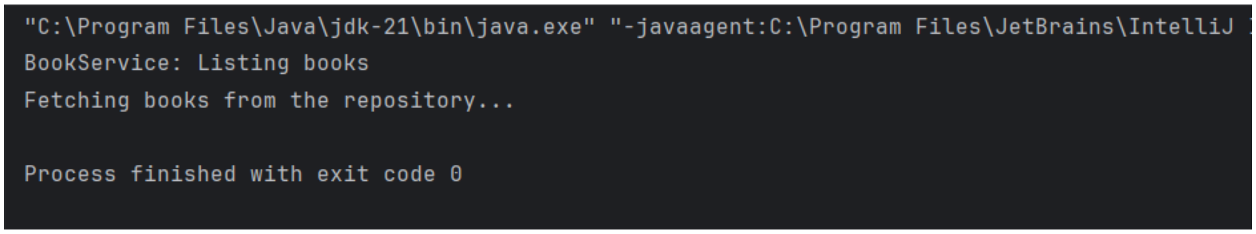
3. Test the Configuration:

- Run the LibraryManagementApplication main class to verify the dependency injection.

```java
package com.library.main;

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

```

```
        BookService bookService = context.getBean("bookService",
BookService.class);
        bookService.listBooks();
    }
}
```

Output:-



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ 
BookService: Listing books
Fetching books from the repository...

Process finished with exit code 0
```

Exercise 4: Creating and Configuring a Maven Project

Scenario:
You need to set up a new Maven project for the library management application and add Spring Dependencies.

Steps:
1. Create a New Maven Project:
    - Create a new Maven project named LibraryManagement.
2. Add Spring Dependencies in pom.xml:
    - Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.
3. Configure Maven Plugins:
    - Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

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_Exercise1</artifactId>
   <version>1.0-SNAPSHOT</version>

   <properties>
       <maven.compiler.source>21</maven.compiler.source>
       <maven.compiler.target>21</maven.compiler.target>
       <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
```

```xml
   </properties>

   <dependencies>
       <!-- Spring Core -->
       <dependency>
           <groupId>org.springframework</groupId>
           <artifactId>spring-context</artifactId>
           <version>5.3.32</version>
       </dependency>

       <!-- Spring AOP -->
       <dependency>
           <groupId>org.springframework</groupId>
           <artifactId>spring-aop</artifactId>
           <version>5.3.32</version>
       </dependency>

       <!-- Spring WebMVC -->
       <dependency>
           <groupId>org.springframework</groupId>
           <artifactId>spring-webmvc</artifactId>
           <version>5.3.32</version>
       </dependency>
   </dependencies>

   <!-- Step 3: Maven Compiler Plugin -->
   <build>
       <plugins>
           <plugin>
               <groupId>org.apache.maven.plugins</groupId>
               <artifactId>maven-compiler-plugin</artifactId>
               <version>3.10.1</version>
               <configuration>
                   <source>1.8</source>
                   <target>1.8</target>
               </configuration>
           </plugin>
       </plugins>
   </build>

</project>
```