FPT Software

JAVA SPRING FRAMEWORK

Lab Guides

Document Code	25e-BM/HR/HDCV/FSOFT
Version	1.0
Effective Date	01/05/2022

Hanoi, 03/2024

RECORD OF CHANGES

No	Effective Date	Change Description	Reason	Reviewer	Approver
1	06/08/2024	Create a new Lab	Create new		VinhNV

Contents

CODE:	JSFW_Lab_01_Opt2
TYPE:	SHORT
LOC:	200
DURATION:	60 MINUTES

Java Spring Framework Introduction

Objectives:

- Learn how to use dependency injection and autowiring in Spring.
- Understand the difference between XML-based and annotation-based configuration.
- Configure and use beans with dependencies.

Lab Specifications:

Create a Spring application to demonstrate dependency injection and autowiring.

Problem Description:

- Trainees must write scripts to test the methods they have developed.

Prerequisites:

- Using Java SDK version 8.0 at least.
- Using Maven.
- Using Spring Framework 5.0 or higher version.

Guidelines:

Step 1: Extend the previous project to include dependency injection:

Open the existing lab01_spring_intro Maven project in IntelliJ.

Step 2: Create a new MessageService class:

```
package org.example;

public class MessageService {
    public String getMessage() {
        return "Dependency Injection in Spring!";
    }
}
```

Step 3: Modify the GreetingService to use MessageService:

```
package org.example;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class GreetingService {
```

```
    @Autowired
    private MessageService messageService;

    public void setMessageService(MessageService messageService) {
        this.messageService = messageService;
    }

    public void sayHello() {
        System.out.println(messageService.getMessage());
    }
}
```

Step 4: Update the XML Configuration: Modify beans.xml to define the MessageService bean and inject it into the GreetingService bean.

```
<?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="messageService" class="org.example.spring_intro.MessageService"/>

<bean id="greetingService" class="org.example.spring_intro.GreetingService">
    <property name="messageService" ref="messageService"/>
</bean>
</beans>
```

Step 5: Enable Autowiring: Modify the GreetingService to use the @Autowired annotation.

```
package org.example;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class GreetingService {
    @Autowired
    private MessageService messageService;

    public void setMessageService(MessageService messageService) {
        this.messageService = messageService;
    }

    public void sayHello() {
        System.out.println(messageService.getMessage());
    }
}
```

Step 6: Update Configuration for Annotation-Based Wiring: Create a new configuration file beans-annotation.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:annotation-config/>
    <context:component-scan base-package="org.example"/>
```

```
    <bean id="messageService" class="org.example.MessageService"/>
</beans>
```

Step 7: Run the Application with Annotation Configuration: Update App.java to use the new beans-annotation.xml:

```
package org.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("beans-
annotation.xml");
        GreetingService greetingService =
context.getBean(GreetingService.class);
        greetingService.sayHello();
    }
}
```

Step 8: Run the Application:

- Run the App.java class again and verify that it prints "Dependency Injection in Spring!".

Step 9: Write a JUnit Test Case: Create a JUnit test class to test the GreetingService with autowiring.

1. Ensure JUnit dependency is in pom.xml:
2. Create a test class GreetingServiceTest.java:
3. Run the test and verify it passes.

This exercise will help you understand the use of dependency injection and autowiring in Spring, and how to configure beans using both XML and annotations.

----oOo-----

THE END