FPT Software

JAVA SPRING FRAMEWORK

Lab Guides

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

Hanoi, 03/2022

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_Opt1
TYPE:	SHORT
LOC:	200
DURATION:	60 MINUTES

Java Spring Framework Introduction

Objectives:

- Able to create a Maven project using IntelliJ.
- Understand how to configure Spring IoC and Spring Beans.
- Learn how to set up dependency injection and autowiring.
- Write and run a simple Spring application.

Lab Specifications:

Create a basic Spring application to understand the concepts of Spring IoC, Spring Beans, dependency injection, and autowiring.

Problem Description:

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

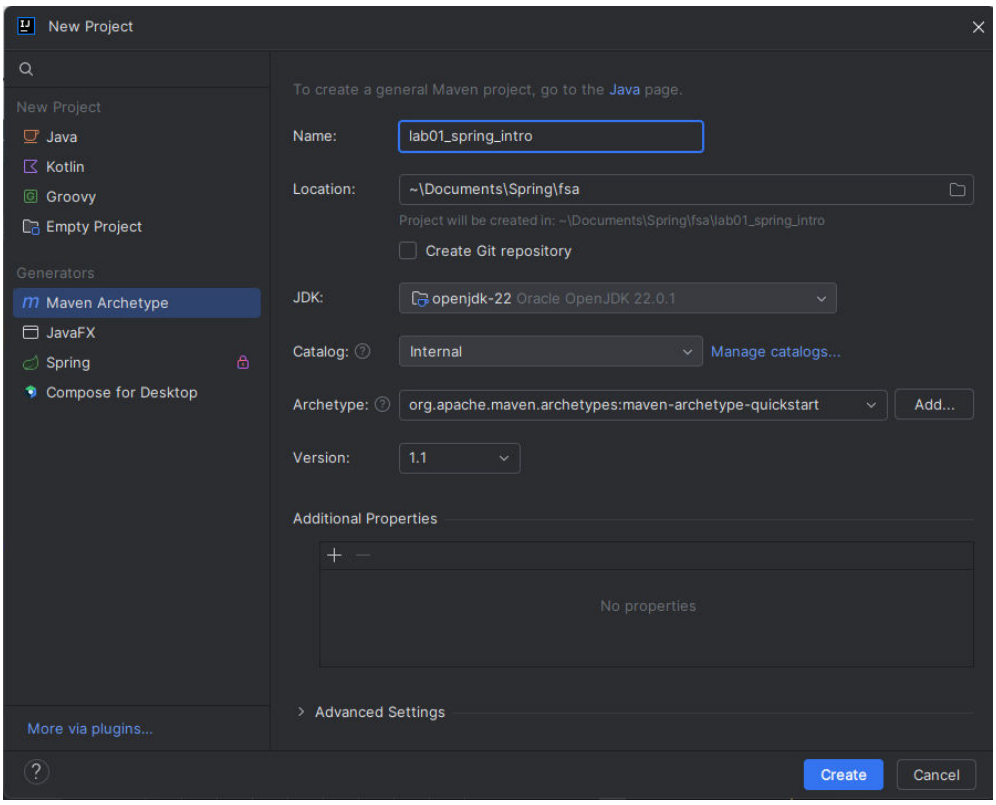
Prerequisites:

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

Guidelines:

Step 1: Create a new Maven project with the name spring_intro in IntelliJ:

1. Open IntelliJ IDEA.
2. Click on File -> New -> Project....
3. Select **Maven** from the project types.
4. Click **Next** and set the project name to lab01_spring_intro.
5. Set the groupId to org.example**.**
6. Click **Create**.



Step 2: Add dependencies and configuration into pom.xml file: Add the Spring Core dependency to your pom.xml file.

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>5.3.20</version>
</dependency>
```

Step 3: Create a Bean: Create a simple Java class named GreetingService.

```
package org.example;

public class GreetingService {
    public void sayHello() {
        System.out.println("Hello, Spring Framework!");
    }
}
```

Step 4: Configure Spring IoC Container: Create an XML configuration file beans.xml in the resources folder.

```
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.xml");
        GreetingService greetingService = (GreetingService)
context.getBean("greetingService");
        greetingService.sayHello();
    }
}
```

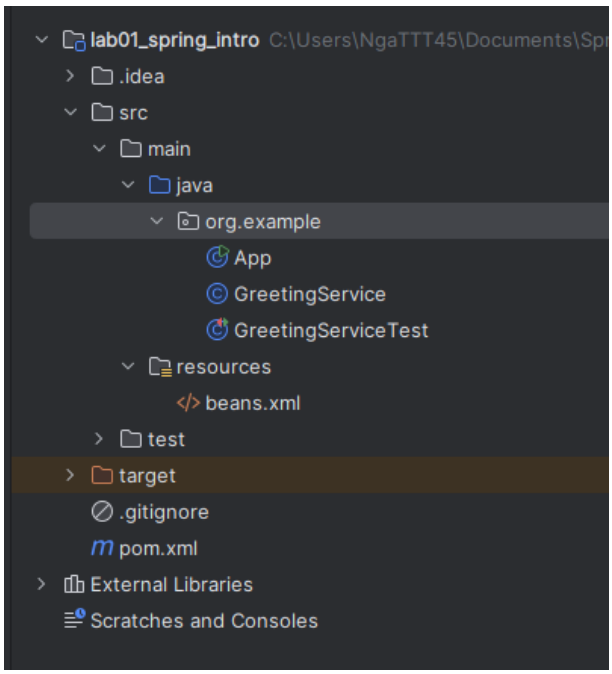
Step 5: Initialize Spring Container and Use the Bean: Create a main class App.java to load the Spring context and retrieve the bean.

```
package org.example;

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

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

Here the structure of the project:



Step 6: Run the Application:

- Run the App.java class.
- Verify that it prints "Hello, Spring Framework!".

Step 7: Write a JUnit Test Case:

Create a JUnit test class to test the GreetingService.

1. Add JUnit dependency to pom.xml.

```
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter</artifactId>
  <version>RELEASE</version>
  <scope>compile</scope>
</dependency>
```

2. Create a test class GreetingServiceTest.java.

```
package org.example;

import org.junit.jupiter.api.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import static org.junit.jupiter.api.Assertions.assertNotNull;

public class GreetingServiceTest {
    @Test
    public void testSayHello() {
        ApplicationContext context = new
ClassPathXmlApplicationContext("beans.xml");
        GreetingService greetingService = (GreetingService)
context.getBean("greetingService");
        assertNotNull(greetingService);
        greetingService.sayHello();
    }
}
```

3. Run the test and verify it passes.

This exercise will help you understand the basics of setting up a Spring application, configuring Spring beans, and performing dependency injection and autowiring.

----oOo-----

THE END