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_02_Opt2
TYPE:	SHORT
LOC:	200
DURATION:	60 MINUTES

Java Spring Framework Introduction

Objectives:

- Understand how to use singleton scope with Spring beans.
- Learn to configure and use beans with prototype scope in a real-life scenario.

Lab Specifications:

In a Student Management System, the UniversityService is a singleton-scoped bean since it represents a single instance across the entire application.

Problem Description:

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

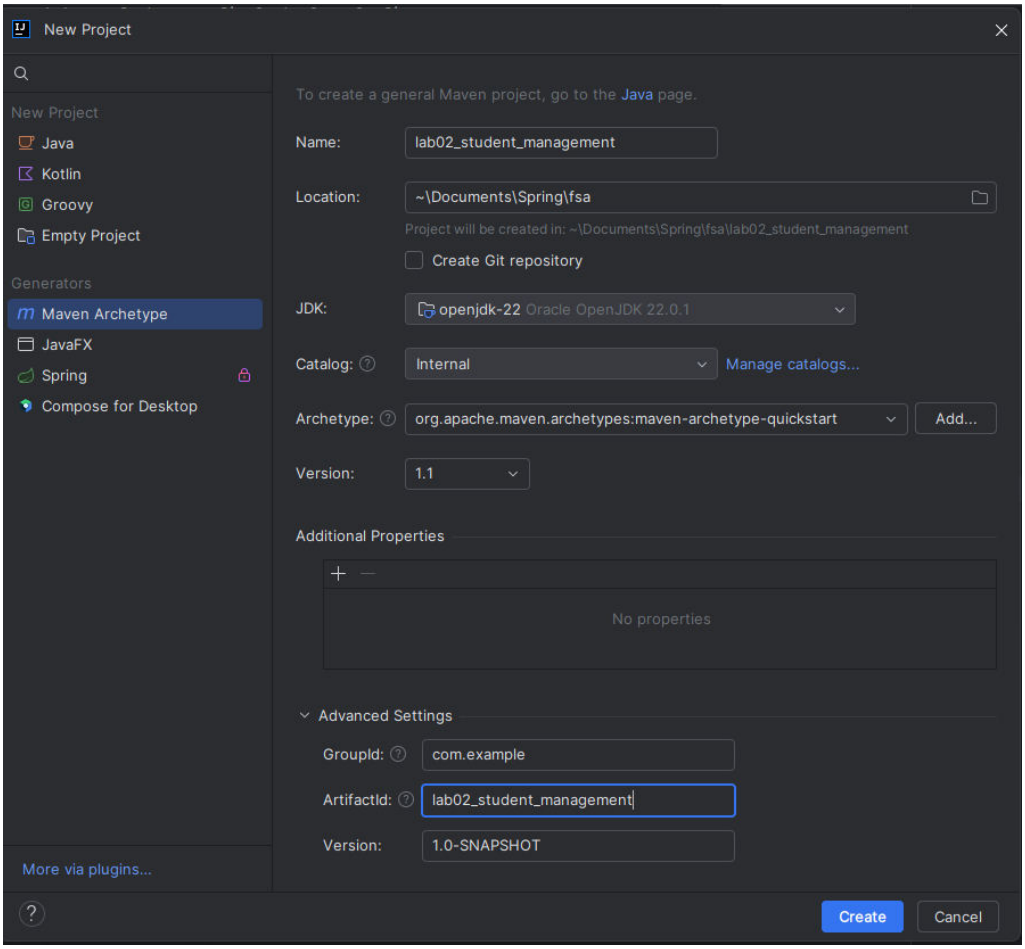
Prerequisites:

- 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 IntelliJ IDEA.
- Click on File -> New -> Project....
- Select Maven from the project types.
- Click Next and set the project name to **lab02_student_management**.
- Set the groupId to com.example and artifactId to **lab02_student_management**.
- Click **Create**.



Step 2: Add dependencies and configuration into pom.xml file:

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.7.5</version>
  <relativePath/>
</parent>
```

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 entity classes:

Create UniversityService class:

```
package com.example;

public class UniversityService {
    private String name;

    public UniversityService() {
        this.name = "Global University";
    }

    public String getName() {
        return name;
```

```
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "UniversityService{name='" + name + "'}";
    }
}
```

Step 4: Configure Beans with Prototype Scope.

Create a configuration class AppConfig

```
package com.example;


import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Scope;

@Configuration
public class AppConfig {

    @Bean
    @Scope("singleton")
    public UniversityService universityService() {
        System.out.println("A new UniversityService instance created");
        return new UniversityService();
    }
}
```

Step 5: Create a Main Class to Test the Singleton Scope:

Create a **MainApp** class:

```
package com.example;

import org.springframework.context.ApplicationContext;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);

        UniversityService service1 = context.getBean(UniversityService.class);
        UniversityService service2 = context.getBean(UniversityService.class);

        service1.setName("Global University");
        service2.setName("International University");

        System.out.println(service1);
        System.out.println(service2);
    }
}
```

Step 6: Run the Application:

- Run the MainApp.java class.
- Verify that it prints:

```
A new UniversityService instance created
UniversityService{name='International University'}
UniversityService{name='International University'}
```

Step 7: Write a JUnit Test Case:

1. Update porm.xml

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

2. Create a test class **UniversityServiceSingletonScopeTest**.java.

```
package com.example;

import org.junit.jupiter.api.Test;
import org.springframework.context.ApplicationContext;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

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

public class UniversityServiceSingletonScopeTest {

    @Test
    public void testSingletonScope() {
        ApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);

        UniversityService service1 = context.getBean(UniversityService.class);
        UniversityService service2 = context.getBean(UniversityService.class);

        // Check that service1 and service2 are the same instance
        assertSame(service1, service2, "The two UniversityService beans should
be the same instance");

        service1.setName("Global University");

        // Check if service2 has the same state as service1
        assertSame("Global University", service2.getName());
    }
}
```

3. Run the test and verify it passes.

This exercise will help you understand how to use prototype scope in Spring with real-life scenarios such as managing employees and customers. The key takeaway is that each time a prototype-scoped bean is requested, a new instance is created, allowing for independent management of each entity.

----oOo-----

THE END