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

FRESHER ACADEMY		
	CODE:	**JSFW_Lab_02_Opt1**
	TYPE:	**SHORT**
	LOC:	**200**
	DURATION:	**60 MINUTES**

Java Spring Framework Introduction

Objectives:

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

Lab Specifications:

Create an Employee Management System where each Employee bean is a prototype-scoped bean, meaning a new instance is created each time the bean is requested.

Problem Description:

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

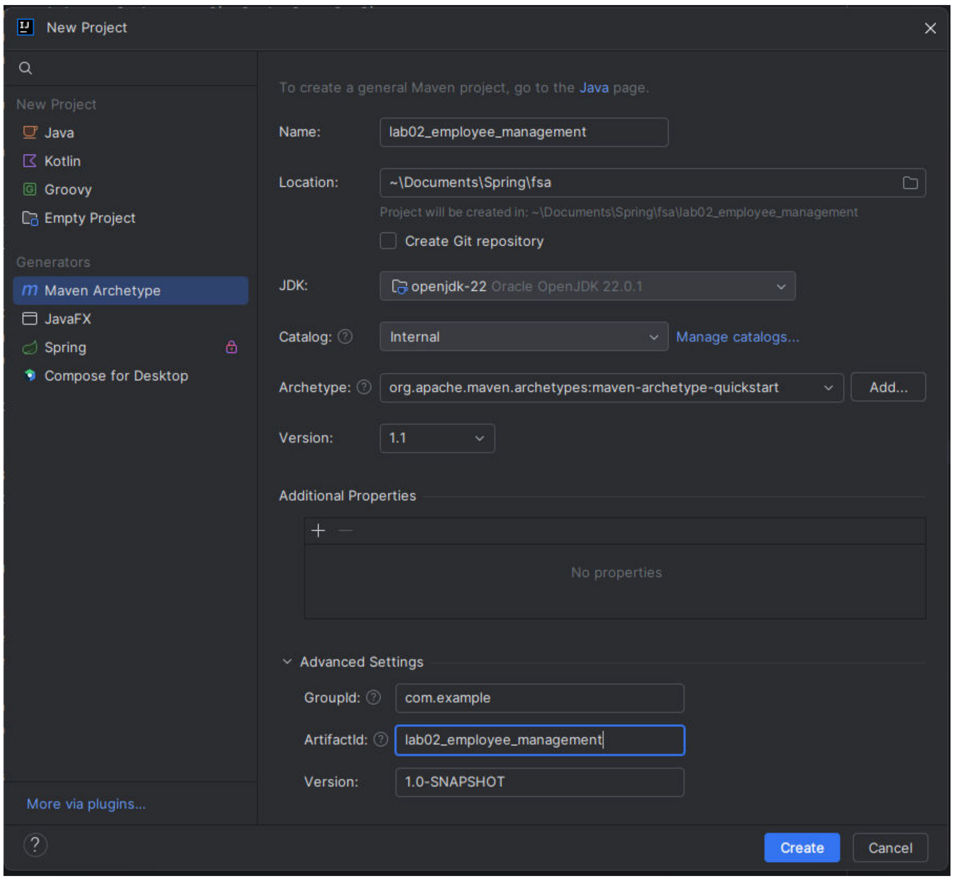
Prerequisites:

- 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_employee_management.
- Set the groupId to com.example and artifactId to lab02_employee_management.
- Click **Create**.


New Project

To create a general Maven project, go to the Java page.

Name: lab02_employee_management

Location: ~\Documents\Spring\fsa

Project will be created in: ~\Documents\Spring\fsa\lab02_employee_management

Create Git repository

JDK: openjdk-22 Oracle OpenJDK 22.0.1

Catalog: Internal

Archetype: org.apache.maven.archetypes:maven-archetype-quickstart

Version: 1.1

Additional Properties: No properties

Advanced Settings

GroupId: com.example

ArtifactId: lab02_employee_management

Version: 1.0-SNAPSHOT

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 entity classes:

Create Customer class:

```
package com.example;

public class Employee {
    private String name;
    private String department;
    private int id;

    public Employee(int id, String name, String department) {
        this.id = id;
        this.name = name;
        this.department = department;
    }

    public String getName() {
        return name;
    }

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

    public void setDepartment(String department) {
        this.department = department;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    @Override
    public String toString() {
        return "Employee{id=" + id + ", name='" + name + "', department='"
+ department + "'}";
    }
}
```

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("prototype")
    public Employee employee() {
        System.out.println("A new Employee instance created");
        return new Employee(1, "John Doe", "IT");
    }
}
```

Step 5: Create a Main Class to Test the Prototype 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);
```

```
        Employee emp1 = context.getBean(Employee.class);
        Employee emp2 = context.getBean(Employee.class);

        emp1.setId(1);
        emp1.setName("John Doe");
        emp1.setDepartment("IT");

        emp2.setId(2);
        emp2.setName("Jane Smith");
        emp2.setDepartment("HR");

        System.out.println(emp1);
        System.out.println(emp2);
    }
}
```

Step 6: Run the Application:

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

```
A new Employee instance created

A new Employee instance created

Employee{id=1, name='John Doe', department='IT'}

Employee{id=2, name='Jane Smith', department='HR'}
```

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 **EmployeePrototypeScopeTest**.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.assertNotSame;

public class EmployeePrototypeScopeTest {

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

        Employee emp1 = context.getBean(Employee.class);
        Employee emp2 = context.getBean(Employee.class);
        assertNotSame(emp1, emp2, "The two Employee beans should be
different instances");

        emp1.setId(1);
        emp1.setName("John Doe");
        emp1.setDepartment("IT");

        emp2.setId(2);
        emp2.setName("Jane Smith");
        emp2.setDepartment("HR");

        System.out.println(emp1);
        System.out.println(emp2);
    }
}
```

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