**Spring Bean 作用域**

项目地址：https://github.com/xuegang20/mystudyrecode/tree/main/code/SpringExample

## Bean 的作用域

当在 Spring 中定义一个 bean 时，你必须声明该 bean 的作用域的选项。例如，为了强制 Spring 在每次需要时都产生一个新的 bean 实例，你应该声明 bean 的作用域的属性为 **prototype**。同理，如果你想让 Spring 在每次需要时都返回同一个bean实例，你应该声明 bean 的作用域的属性为 **singleton**。

Spring 框架支持以下五个作用域，分别为 singleton、prototype、request、session 和 global session，5种作用域说明如下所示，

注意，如果你使用 web-aware ApplicationContext 时，其中三个是可用的。

|  |  |
| --- | --- |
| **作用域** | **描述** |
| singleton | 在spring IoC容器仅存在一个Bean实例，Bean以单例方式存在，默认值 |
| prototype | 每次从容器中调用Bean时，都返回一个新的实例，即每次调用getBean()时，相当于执行newXxxBean() |
| request | 每次HTTP请求都会创建一个新的Bean，该作用域仅适用于WebApplicationContext环境 |
| session | 同一个HTTP Session共享一个Bean，不同Session使用不同的Bean，仅适用于WebApplicationContext环境 |
| global-session | 一般用于Portlet应用环境，该作用域仅适用于WebApplicationContext环境 |

本章将讨论前两个范围，当我们将讨论有关 web-aware Spring ApplicationContext 时，其余三个将被讨论。

## singleton 作用域：

singleton 是默认的作用域，也就是说，当定义 Bean 时，如果没有指定作用域配置项，则 Bean 的作用域被默认为 singleton。

当一个bean的作用域为 Singleton，那么 Spring IoC 容器中只会存在一个共享的 bean 实例，并且所有对 bean 的请求，只要 id 与该 bean 定义相匹配，则只会返回 bean 的同一实例。

也就是说，当将一个 bean 定义设置为 singleton 作用域的时候，Spring IoC 容器只会创建该 bean 定义的唯一实例。

Singleton 是单例类型，就是在创建起容器时就同时自动创建了一个 bean 的对象，不管你是否使用，他都存在了，每次获取到的对象都是同一个对象。注意，Singleton 作用域是 Spring 中的缺省作用域。你可以在 bean 的配置文件中设置作用域的属性为 singleton，如下所示：

<!-- A bean definition with singleton scope -->

<bean id="..." class="..." scope="singleton">

<!-- collaborators and configuration for this bean go here -->

</bean>

## 例子

这里是 **HelloWorld.java** 文件的内容：

package com.tutorialspoint;

public class HelloWorld {

private String message;

public void setMessage(String message){

this.message = message;

}

public void getMessage(){

System.out.println("Your Message : " + message);

}

}

下面是 MainApp.java 文件的内容：

package com.tutorialspoint;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("Beans.xml");

HelloWorld objA = (HelloWorld) context.getBean("helloWorld");

objA.setMessage("I'm object A");

objA.getMessage();

HelloWorld objB = (HelloWorld) context.getBean("helloWorld");

objB.getMessage();

}

}

下面是 singleton 作用域必需的配置文件 **Beans.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"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

<bean id="helloWorld" class="com.tutorialspoint.HelloWorld"

scope="singleton">

</bean>

</beans>

一旦你创建源代码和 bean 配置文件完成后，我们就可以运行该应用程序。如果你的应用程序一切都正常，将输出以下信息：

Your Message : I'm object A

Your Message : I'm object A

## prototype 作用域

当一个 bean 的作用域为 Prototype，表示一个 bean 定义对应多个对象实例。Prototype 作用域的 bean 会导致在每次对该 bean 请求（将其注入到另一个 bean 中，或者以程序的方式调用容器的 getBean() 方法）时都会创建一个新的 bean 实例。Prototype 是原型类型，它在我们创建容器的时候并没有实例化，而是当我们获取bean的时候才会去创建一个对象，而且我们每次获取到的对象都不是同一个对象。根据经验，对有状态的 bean 应该使用 prototype 作用域，而对无状态的bean则应该使用 singleton 作用域。

为了定义 prototype 作用域，你可以在 bean 的配置文件中设置作用域的属性为 prototype，如下所示：

<!-- A bean definition with singleton scope -->

<bean id="..." class="..." scope="prototype">

<!-- collaborators and configuration for this bean go here -->

</bean>

## 例子

这里是 **HelloWorld.java** 文件的内容：

package com.tutorialspoint;

public class HelloWorld {

private String message;

public void setMessage(String message){

this.message = message;

}

public void getMessage(){

System.out.println("Your Message : " + message);

}

}

下面是 **MainApp.java** 文件的内容：

package com.tutorialspoint;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("Beans.xml");

HelloWorld objA = (HelloWorld) context.getBean("helloWorld");

objA.setMessage("I'm object A");

objA.getMessage();

HelloWorld objB = (HelloWorld) context.getBean("helloWorld");

objB.getMessage();

}

}

下面是 **prototype** 作用域必需的配置文件 Beans.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"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

<bean id="helloWorld" class="com.tutorialspoint.HelloWorld"

scope="prototype">

</bean>

</beans>

一旦你创建源代码和 Bean 配置文件完成后，我们就可以运行该应用程序。如果你的应用程序一切都正常，将输出以下信息：

Your Message : I'm object A

Your Message : null