Spring Bean Scope

Bean Scope refers to the lifecycle of a bean, visibility of a bean, how long does the bean live, how many instances are created, how is the bean shared?

Bean's default scope is a singleton. The spring container will create a single instance of the bean. It is cached in memory. All requests for the bean will return a shared reference of the same bean.

Spring provides @Scope annotation to mark a bean scope.

Spring Bean Scope

The following are the types of bean scope used in the Spring application.

Scope	Description
singleton	It is the default scope of a bean. It represents a single bean scope for each Spring IoC container.
prototype	It is used to set the scope of a single bean definition to any number of object instances.

Spring Bean Scope

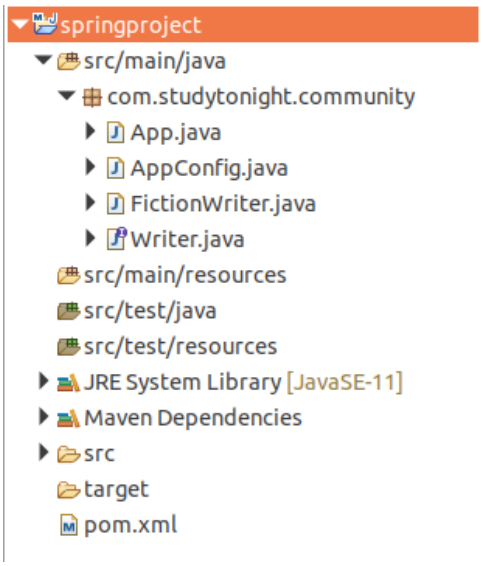
request	This bean scope is set for a single HTTP request.
session	This scope is set for a single bean definition to an HTTP session.
application	It sets bean scope to a ServletContext.
WebSocket	It sets bean scope to a WebSocket.

Example: Bean Singleton Scope (Default Scope)

Let's create an example to mark a bean scope as default and check whether it has a default scope or not.

Spring Bean Scope

Project Structure



Spring Bean Scope

Project Files Source Code:

//App.java

This file contains the code to create an IOC container for our application. The AnnotationConfigApplicationContext class is used to create an object for application context. Here two bean objects are created and checked whether both are equal or not.

```
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
public class App
{

	public static void main(String[] args)
	 {

		AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);
		Writer writer1 = context.getBean("fictionWriter", Writer.class);
		Writer writer2 = context.getBean("fictionWriter", Writer.class);

		boolean isSame = writer1 == writer2;
		System.out.println("Instance One :"+writer1);
		System.out.println("Instance One :"+writer2);
		System.out.println("Both bean instances are same: "+isSame);

		//writer.write();
		// Close the context
		context.close();
	}
}
```

Spring Bean Scope

// AppConfig.java

This is a configuration file in Java which is an alternate of the applicationContext.xml file that we created for the XML-based configuration example. The @Configuration annotation indicates that this is not a simple class but a configuration class and the @ComponentScan annotation is used to indicate the component location in our spring project.

```
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
@Configuration
@ComponentScan("com.studytonight.community")
public class AppConfig
{

}
```

// FictionWriter.java

```
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

@Component
@Scope
public class FictionWriter implements Writer {

	@Override
	public void write() {

		System.out.println("Write Fiction Novels...");
	}
}
```

// Writer.java

```
package com.studytonight.community;
```

Spring Bean Scope

```
public interface Writer
{

	void write();
	void getAward();
}
```

// pom.xml

Spring Bean Scope

This file contains all the dependencies of this project such as spring jars, servlet jars, etc. Put these dependencies into your project to run the application.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.studytonight</groupId>
  <artifactId>springproject</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
  <!-- https://mvnrepository.com/artifact/org.springframework/spring-web -->
  <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-core</artifactId>
                        <version>${spring.version}</version>
                </dependency>
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-context</artifactId>
                        <version>${spring.version}</version>
                </dependency>
        </dependencies>
        <properties>
                <spring.version>5.2.8.RELEASE</spring.version>
        </properties>
  <build>
    <sourceDirectory>src</sourceDirectory>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>

  </build>
</project>
```

Spring Bean Scope

```
Instance One :com.studytonight.community.FictionWriter@6a28ffa4

Instance One :com.studytonight.community.FictionWriter@6a28ffa4

Both bean instances are same: true
```

Example: Bean Scope Prototype

This is another type of bean scope and in this case, two bean objects of the same class are not equal.

// FictionWriter.java

```
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

@Component
@Scope("prototype")
public class FictionWriter implements Writer
{

	@Override
	public void write()
 {

		System.out.println("Write Fiction Novels...");
	}
}
```

```
Instance One :com.studytonight.community.FictionWriter@6a28ffa4
```

Spring Bean Scope

```
Instance One :com.studytonight.community.FictionWriter@48ae9b55

Both bean instances are same: false
```