**9. Spring Configuration with Java Annotations - Bean Scopes**

**Bean Scope**:

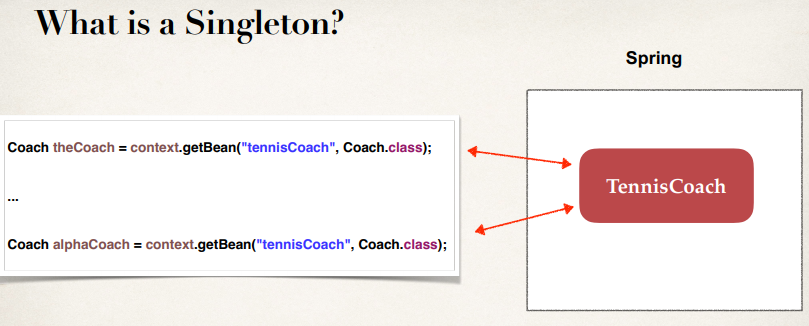
1. Scope refers to the lifecycle of a bean
2. How long does the bean live?
3. How many instances are created?
4. How is the bean shared?

**Default Scope**:

The default scope of a bean is singleton.

**What is singleton**:

1. When the scope is singleton spring creates only one instance of the bean, by default.
2. It cached in memory
3. All requests for the bean will return a SHARED reference to the SAME bean



**Explicitly Specify Bean Scope**:

@Component

@Scope("singleton")

public class TennisCoach implements Coach {

...

}

**Additional Spring Bean Scopes**:

|  |  |
| --- | --- |
| **Scope** | **Description** |
| singleton | Create a single shared instance of the bean. Default scope. |
| prototype | Creates a new bean instance for each container request. |
| request | Scoped to an HTTP web request. Only used for web apps. |
| session | Scoped to an HTTP web session. Only used for web apps. |
| global-session | Scoped to a global HTTP web session. Only used for web |

**Prototype Scope Example**:

When the scope is prototype Spring will create a new object for each bean.

@Component

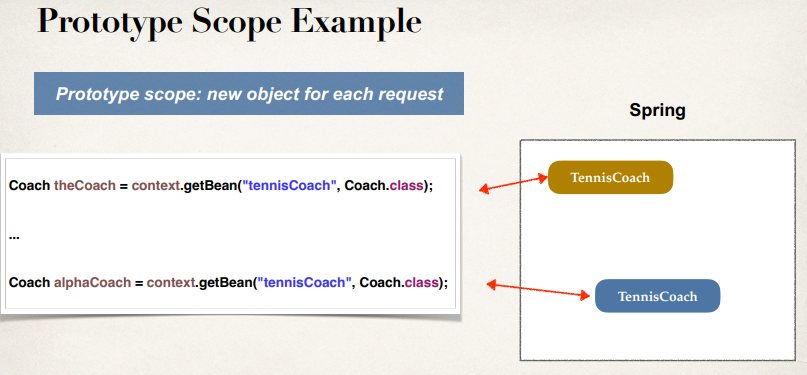
@Scope("prototype")

public class TennisCoach implements Coach {

...

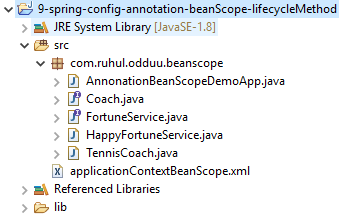
}

**Prototype Scope Example**:



**Example**:

**Project Structure**:



**File: Coach.java**:

**package** com.ruhul.odduu.beanscope;

**public** **interface** Coach {

**public** String getDailyWorout();

**public** String getDailyFortune();

}

**File: FortuneService.java**:

**package** com.ruhul.odduu.beanscope;

**public** **interface** FortuneService {

**public** String getFortune();

}

**File: HappyFortuneService.java:**

**package** com.ruhul.odduu.beanscope;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** HappyFortuneService **implements** FortuneService {

@Override

**public** String getFortune() {

**return** "Today is your lucky day!!!";

}

}

**File: TennisCoach.java**:

**package** com.ruhul.odduu.beanscope;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** TennisCoach **implements** Coach {

// field injection

@Autowired

**private** FortuneService fortuneService;

// default constructor

**public** TennisCoach() {

System.***out***.println(">> TennisCoach: inside default constructor");

}

@Override

**public** String getDailyWorout() {

**return** "Practice your backhand volley";

}

@Override

**public** String getDailyFortune() {

**return** fortuneService.getFortune();

}

}

**File: applicationContextBeanScope.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"*

xmlns:context=*"http://www.springframework.org/schema/context"*

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

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context.xsd"*>

<!-- add entry to enable component scanning -->

<context:component-scan

base-package=*"com.ruhul.odduu.beanscope"* />

</beans>

**File: AnnonationBeanScopeDemoApp.java**:

**package** com.ruhul.odduu.beanscope;

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

**public** **class** AnnonationBeanScopeDemoApp {

**public** **static** **void** main(String[] args) {

// load spring configuration file

ClassPathXmlApplicationContext context = **new** ClassPathXmlApplicationContext("applicationContextBeanScope.xml");

// retrieve bean from spring container

Coach theCoach = context.getBean("tennisCoach", Coach.**class**);

Coach alphaCoach = context.getBean("tennisCoach", Coach.**class**);

// check if they are the same

**boolean** resut = (theCoach == alphaCoach);

System.***out***.println("Point to the same object: " + resut);

// memory address of bean

System.***out***.println("Memory location of theCoach: " + theCoach);

System.***out***.println("Memory location of theCoach: " + alphaCoach);

// close the context

context.close();

}

}

/\*

**Output**:

>> TennisCoach: inside default constructor

Point to the same object: true

Memory location of theCoach: com.ruhul.odduu.beanscope.TennisCoach@61ca2dfa

Memory location of theCoach: com.ruhul.odduu.beanscope.TennisCoach@61ca2dfa

**\*/**

Here we don’t specify any bean-scope explicitly. The default scope of Spring is singleton. Hence the address of two bean is same.

Now we add the scope explicitly. We add scope “prototype”

**File: TennisCoach.java**:

**package** com.ruhul.odduu.beanscope;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

@Component

@Scope("prototype")

**public** **class** TennisCoach **implements** Coach {

// field injection

@Autowired

**private** FortuneService fortuneService;

// default constructor

**public** TennisCoach() {

System.***out***.println(">> TennisCoach: inside default constructor");

}

@Override

**public** String getDailyWorout() {

**return** "Practice your backhand volley";

}

@Override

**public** String getDailyFortune() {

**return** fortuneService.getFortune();

}

}

/\*

**Output**:

>> TennisCoach: inside default constructor

>> TennisCoach: inside default constructor

Point to the same object: false

Memory location of theCoach: com.ruhul.odduu.beanscope.TennisCoach@5fcd892a

Memory location of theCoach: com.ruhul.odduu.beanscope.TennisCoach@8b87145

**\*/**

Since the scope is prototype Spring will create two separate objects. Since two object is created so the constructor invoke two time.

**Life-Cycle Methods**:

9. Spring Configuration with Java Annotations - Bean Scopes and Lifecycle Methods