**Spring Bean Scope:**

When Spring beans are creating from Spring containers you can provide instruction to Spring bean by ***scope*** attribute how this bean should be scoped.

Here Scope means How many bean instances will be created for a certain context. In our application we create some classes as Singleton or some classes need to instantiate whenever client invoke it. Furthermore, in web application we want to create instances based on per request, per session, per context etc. Those are called scope of a bean.

Spring has 5 scopes in which first two are used for Spring core and rest of the three are applicable for a Web application.

**Diagram**: **Scopes of Spring bean**

As I said Spring core only uses two scope Singleton and prototype let try to understand them by example.

**Singleton Scope**: Singleton scope says only one instance will be there in Spring container please note that one instance in Spring container not in JVM.

So if I change a property of a bean and again try to load the bean from the container I should able to see that change as only one instance is maintained in the container.

Example:

BeanScope.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=*"bike"* class=*"com.example.scope.Bike"* scope=*"singleton"*>

<property name=*"name"* value=*"HeroHonda"*/>

</bean>

</beans>

**Java:**

**package** com.example.scope;

**public** **class** Bike {

**private** String name;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

@Override

**public** String toString() {

**return** "Bike [name=" + name + "]";

}

}

**package** com.example.scope;

**import** org.springframework.context.ApplicationContext;

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

**public** **class** Application {

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

ApplicationContext ctx = **new** ClassPathXmlApplicationContext("configFiles/BeanScope.xml");

Bike bike =(Bike) ctx.getBean("bike");

System.***out***.println("Bike info ::" + bike);

bike.setName("CBZ");

System.***out***.println("Bike info After ::" + bike);

Bike bike1 =(Bike) ctx.getBean("bike");

System.***out***.println("Requesting container again ::" + bike1);

System.***out***.println("Both instance is same ? " + (bike==bike1));

}

}

Output:

Bike info ::Bike [name=HeroHonda]

Bike info After ::Bike [name=CBZ]

Requesting container again ::Bike [name=CBZ]

Both instance is same ? true

Please note that in SpringBean.xml we set bean scope as the singleton, by doing this we give an instruction to Spring container make this Bike bean as Singleton.

Default scope of Spring bean is Singleton so either we define or omit scope property, spring always creates a singleton bean.

As Spring container manages only one instance of bike bean, after changing the name of the bike that changed name is retain so once again when we request that bike bean it will show the changed name not the name provide by configuration.

Let change the scope to prototype then see what will be the output

**Prototype Scope:** Spring offers a new bean instance per invocation.

<?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=*"bike"* class=*"com.example.scope.Bike"* scope=*"prototype"*>

<property name=*"name"* value=*"HeroHonda"*/>

</bean>

</beans>

**package** com.example.scope;

**import** org.springframework.context.ApplicationContext;

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

**public** **class** Application {

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

ApplicationContext ctx = **new** ClassPathXmlApplicationContext("configFiles/BeanScope.xml");

Bike bike =(Bike) ctx.getBean("bike");

System.***out***.println("Bike info ::" + bike);

bike.setName("CBZ");

System.***out***.println("Bike info After ::" + bike);

Bike bike1 =(Bike) ctx.getBean("bike");

System.***out***.println("Requesting container again ::" + bike1);

System.***out***.println("Both instance is same ? " + (bike==bike1));

}

}

Output:

Bike info ::Bike [name=HeroHonda]

Bike info After ::Bike [name=CBZ]

Requesting container again ::Bike [name=HeroHonda]

Both instance is same ? false

Oops, the output is changed. Because now bean scope is a prototype so every invocation spring bean offers a new bean. So it will offer a new bean when we try to get a bean from container second time.

Pay attention to equality in last line in Application.java that will clear the Idea.