**Bean Scopes in Spring**

In Spring we get the bean from the [**Spring container**](http://www.dineshonjava.com/2012/06/spring-ioc-container.html)with some default configuration. **Default behavior is that every beans in the** [**Spring container**](http://davjavaservices.blogspot.com/2012/06/spring-ioc-container.html) **are initialized when** [**bean configuration file**](http://www.dineshonjava.com/2012/06/what-is-bean-factory-in-spring.html) **loaded to the** [**JVM**](http://www.dineshonjava.com/2012/07/what-is-jvm.html).  Whenever **getBean** is called container recognized the **bean by given bean id and return that bean to the caller**. **One more default behavior is that every beans has only one instance in the** [**spring container**](http://www.dineshonjava.com/2012/06/spring-ioc-container.html)**.**

Bean Scope means which is used to decide which type of bean instance should be return from [**Spring container**](http://www.dineshonjava.com/2012/06/spring-ioc-container.html) **back to the caller.**

We can control not only the various [**dependencies and configuration**](http://www.dineshonjava.com/2012/06/dependency-injection-in-spring.html) values that are to be plugged into an object that is created from a particular bean definition, but also the *scope* of the objects created from a particular bean definition.

1. singleton – Return a single bean instance per Spring IoC container
2. prototype – Return a new bean instance each time when requested
3. request – Return a single bean instance per HTTP request. \*
4. session – Return a single bean instance per HTTP session. \*
5. globalSession – Return a single bean instance per global HTTP session. \*

\* ***means only valid in the context of a web-aware Spring ApplicationContext***

In most cases, you may only deal with the Spring’s core scope – singleton and prototype, and the default scope is singleton.

1. [**Singleton Bean Scope**](http://www.dineshonjava.com/2012/06/singleton-bean-scope-in-spring.html)**:** Scopes a single bean definition to a single object instance per [Spring IoC container](http://www.dineshonjava.com/2012/06/spring-ioc-container.html). This is the default behavior of the [spring container](http://www.dineshonjava.com/2012/06/spring-ioc-container.html).

When a bean is a **singleton**, only one *shared* instance of the bean will be managed, and all requests for beans with an ***id or ids*** matching that bean definition will result in that one specific bean instance being returned by the [Spring container](http://www.dineshonjava.com/2012/06/spring-ioc-container.html).

We can say another way, **when you define a bean definition and it is scoped as a singleton,** then the [Spring IoC container](http://www.dineshonjava.com/2012/06/spring-ioc-container.html) will create ***exactly one* *instance*** of the object defined by that bean definition. This single instance will be stored in a cache of such singleton beans, and ***all subsequent requests and references*** for that named bean will result in the **cached object being returned**.  
To define a singleton scope, you can set the **scope** property to **singleton** in the bean configuration file, as shown below:

<bean class="com.dineshonjava.sdnext.beanscope.Point" id="zeroPoint" scope="singleton"> <property name="x" value="0"></property>

<property name="y" value="0"></property>

</bean>

**Using Annotation for Bean Scope:**

@Service

@Scope("singleton")

public class Point

{

private int x;

private int y;

public void setX(int x){

this.x = x;

}

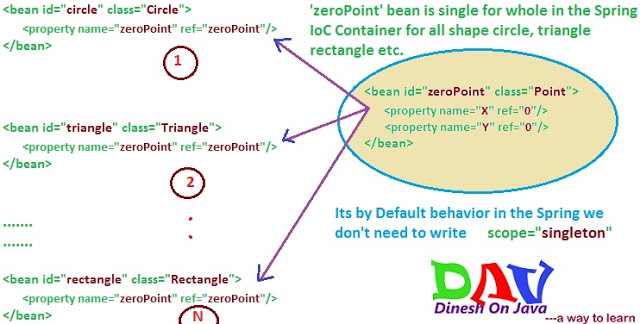
public void setY(int y){

this.y = y;

}

}

**NOTE** : This singleton is differ from the singleton pattern in Java Class. **Single pattern in java mean you can create the only one instance of a that class in** [**JVM**](http://www.dineshonjava.com/2012/07/what-is-jvm.html). **But In spring singleton bean scope means every container can create only single bean in the** [**Spring IoC Container**](http://www.dineshonjava.com/2012/06/spring-ioc-container.html) **but a** [**JVM**](http://www.dineshonjava.com/2012/07/what-is-jvm.html) **can have multiple** [**Spring IoC Container**](http://www.dineshonjava.com/2012/06/spring-ioc-container.html) **so** [**JVM**](http://www.dineshonjava.com/2012/07/what-is-jvm.html) **can multiple beans rather than bean singleton bean scope.**



**2.** [**Prototype Bean Scope**](http://www.dineshonjava.com/2012/06/prototype-bean-scope-with-annotation.html)**:**   
If scope is set to prototype, the [Spring IoC container](http://www.dineshonjava.com/2012/06/spring-ioc-container.html) creates new bean instance of the object every time a request for that specific bean is made. **As a rule, use the prototype scope for all state-full beans and the singleton.**

To define a prototype scope, you can set the **scope** property to **prototype** in the bean configuration file

<bean class="com.dineshonjava.sdnext.beanscope.Point" id="zeroPoint" **scope="prototype"><**property name="x" value="0"></property>

<property name="y" value="0"></property>

</bean>

**Using Annotation for Bean Scope:**

@Service

@Scope("prototype")

public class Point

{

private int x;

private int y;

public void setX(int x){

this.x = x;

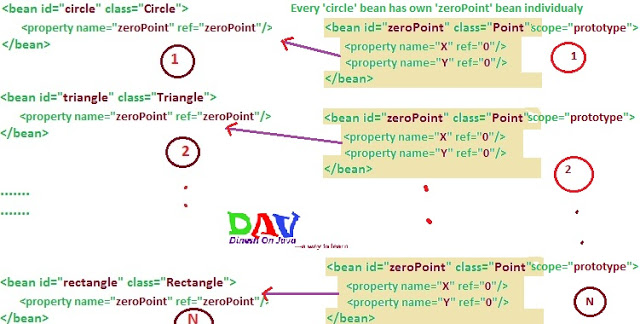
}

public void setY(int y){

this.y = y;

}

}



**There are three web application aware bean scope are given below.**

**1. Request Scope:** This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext.

<bean class="com.dineshonjava.Point" id="point" scope="request"></bean>

**2. Session Scope:** This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

<bean class="com.dineshonjava.Point" id="point" scope="session"></bean>

**3. global-session:** This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

<bean class="com.dineshonjava.Point" id="point" scope="globalSession"></bean>

The global session scope is similar to the standard HTTP Session scope (described immediately above), and really only makes sense in the context of portlet-based web applications. The portlet specification defines the notion of a global Session that is shared amongst all of the various portlets that make up a single portlet web application. Beans defined at the global session scope are scoped (or bound) to the lifetime of the global portlet Session.

Please note that if you are writing a standard Servlet-based web application and you define one or more beans as having global session scope, the standard HTTP Session scope will be used, and no error will be raised.