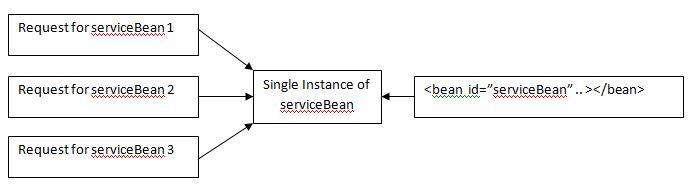
**Spring Bean Scope**

With the help of **Spring Bean Scope** container can decide how particular bean should be created and returned to caller. **Spring Bean Scope** helps the container to decide how particular bean should be treated.   
  
Bean is the actual object of the class that is created by the Spring IoC container. We can create bean definition with the help of XML configuration metadata. When you are defining the metadata configuration for the particular bean, you are actually creating the blueprint or prototype on how IoC container should create the instance and how bean is dependent on other beans.   
  
IoC container reads those information from metadata and create the instance of that bean. There are different types of bean scope; we will discuss them one by one

|  |  |
| --- | --- |
| **Scope Name** | **Description** |
| Singleton | Default Scope, Single object instance per IoC container. |
| Prototype | Scopes a single bean definition to any number of object instances. |
| Request | Each HTTP request will have its own instance. |
| Session | Scopes bean to Life cycle of Session. |
| Global Session | Scopes bean to life cycle of Global Session. Usually in portlet context. |

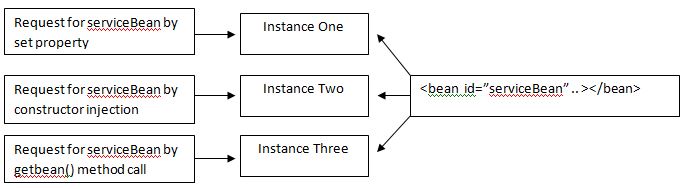
**Singleton Scope**

Singleton scope is the default scope of the bean, only a single shared instance of singleton bean is managed by IoC container. For all requests for that particular singleton bean, there will be one specific bean instance returned by IoC container.   
  
When you define a bean definition as Singleton bean, then IoC container creates only single shared instance of that bean and store it to cache.  
  
  
  [](https://www.ashtpoint.com/wp-content/uploads/2017/05/singleton.jpg) 

**XML Example**

<bean id="serviceBean" class="com.package.ServiceBean" scope="singleton"/>

**Prototype Scope**

Other bean scope is “prototype”, prototype scope of bean will create new bean instance every time a request for that been is raised. That is, whether bean is injected to another bean or request is made through getBean() method. There will be a new instance each time. It is useful for state full beans.   
  
  
  
[](https://www.ashtpoint.com/wp-content/uploads/2017/05/prototype.jpg) 

**XML Example**

<bean id="serviceBean" class="com.package.ServiceBean" scope="prototype"/>

**Note\*:** Spring does not manage complete life cycle of the prototype bean. Although initialization life cycle callback methods are called on all of the prototype objects but destruction life cycle callback methods are not called. The client code must clean prototype objects and release any resource that object is holding up. You can also use bean post processor as well for this particular operation.

**Singleton beans with prototype bean dependency**

One important thing to consider here with prototype and singleton scoped bean is that how they are injected to each other and how they are resolved during instantiation. As we know all Singleton scoped bean are resolved when context is starting up and prototype scoped bean are resolved when request is raised. So when you inject prototype scoped bean into Singleton bean scope then a new prototype bean is instantiated and then injected into singleton bean.

**Request, session, global session, application and Web socket scopes**

All of the scopes can only be used when web-aware spring Application context is implemented such as **XMLWebApplicationContext**. If you try to use them with **ClassPathXMLApplicationContext** then there will be an **IlligalStateException**.   
  
If you access scoped beans within Spring Web MVC, in effect, within a request that is processed by the Spring **DispatcherServlet** or **DispatcherPortlet**, then no special setup is necessary:   
  
**DispatcherServlet** and **DispatcherPortlet** already expose all relevant state.   
  
  
**Scope Example**

<bean id="serviceBean" class="com.package.ServiceBean" scope="request"/>

<bean id="serviceBean" class="com.package.ServiceBean" scope="session"/>

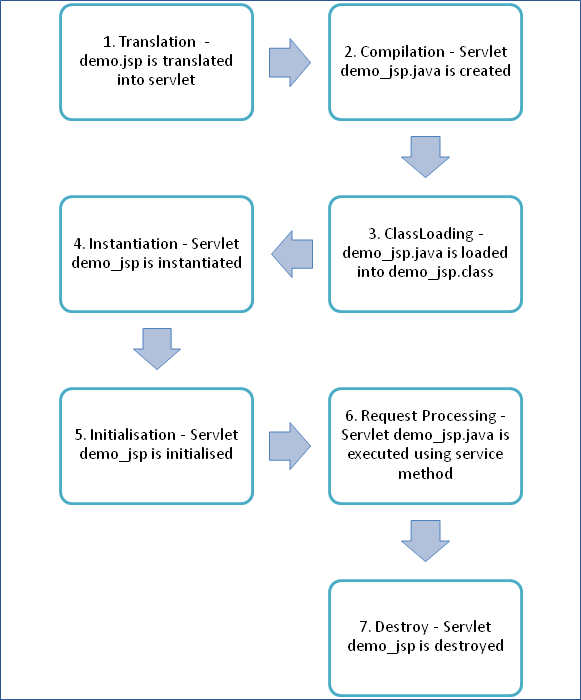
<bean id="serviceBean" class="com.package.ServiceBean" scope="globalsession"/>

<bean id="serviceBean" class="com.package.ServiceBean" scope="prototype"/>

<bean id="serviceBean" class="com.package.ServiceBean" scope="prototype"/>

**JSP Life Cycle:**

JSP Life Cycle is defined as translation of JSP Page into servlet as a JSP Page needs to be converted into servlet first in order to process the service requests.

[](https://cdn.guru99.com/images/jsp/022716_0813_JSPLifeCycl1.png)

Following steps explain the JSP life cycle:

1. Translation of JSP page
2. Compilation of JSP page(Compilation of JSP page into \_jsp.java)
3. Classloading (\_jsp.java is converted to class file \_jsp.class)
4. Instantiation(Object of generated servlet is created)
5. Initialisation(\_jspinit() method is invoked by container)
6. Request Processing(\_jspservice() method is invoked by the container)
7. Destroy (\_jspDestroy() method invoked by the container)