SPRING - BEAN SCOPES

https://www.tutorialspoint.com/spring/spring_bean_scopes.htm

When defining a <bean> you have the option of declaring a scope for that bean. For example, to force Spring to produce a new bean instance each time one is needed, you should declare the bean's scope attribute to be **prototype**. Similarly, if you want Spring to return the same bean instance each time one is needed, you should declare the bean's scope attribute to be **singleton**.

The Spring Framework supports the following five scopes, three of which are available only if you use a web-aware ApplicationContext.

Sr.No.	Scope & Description
1	**singleton** This scopes the bean definition to a single instance per Spring IoC container $default$.
2	**prototype** This scopes a single bean definition to have any number of object instances.
3	**request** This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext.
4	**session** This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.
5	**global-session** This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

In this chapter, we will discuss about the first two scopes and the remaining three will be discussed when we discuss about web-aware Spring ApplicationContext.

The singleton scope

If a scope is set to singleton, the Spring IoC container creates exactly one instance of the object defined by that bean definition. This single instance is stored in a cache of such singleton beans, and all subsequent requests and references for that named bean return the cached object.

The default scope is always singleton. However, when you need one and only one instance of a bean, you can set the **scope** property to **singleton** in the bean configuration file, as shown in the following code snippet –

```
<!-- A bean definition with singleton scope -->
<bean id = "..." class = "..." scope = "singleton">
   <!-- collaborators and configuration for this bean go here -->
</bean>
```

Example

Let us have a working Eclipse IDE in place and take the following steps to create a Spring application –

Steps	Description
1	Create a project with a name *SpringExample* and create a package *com.tutorialspoint* under the **src** folder in the created project.
2	Add required Spring libraries using *Add External JARs* option as explained in the *Spring Hello World Example* chapter.
3	Create Java classes *HelloWorld* and *MainApp* under the *com.tutorialspoint* package.
4	Create Beans configuration file *Beans.xml* under the **src** folder.
5	The final step is to create the content of all the Java files and Bean Configuration file and run the application as explained below.

Here is the content of **HelloWorld.java** file –

```
package com.tutorialspoint;

public class HelloWorld {
   private String message;

   public void setMessage(String message){
      this.message  = message;
   }
   public void getMessage(){
      System.out.println("Your Message : " + message);
   }
}
```

Following is the content of the **MainApp.java** file –

```
package com.tutorialspoint;

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

public class MainApp {
   public static void main(String[] args) {
      ApplicationContext context = new ClassPathXmlApplicationContext("Beans.xml");
      HelloWorld objA = (HelloWorld) context.getBean("helloWorld");

      objA.setMessage("I'm object A");
      objA.getMessage();

      HelloWorld objB = (HelloWorld) context.getBean("helloWorld");
      objB.getMessage();
   }
}
```

Following is the configuration file **Beans.xml** required for singleton scope –

```
<?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 = "helloWorld" class = "com.tutorialspoint.HelloWorld" scope = "singleton">
   </bean>

</beans>
```

Once you are done creating the source and bean configuration files, let us run the application. If everything is fine with your application, it will print the following message –

```
Your Message : I'm object A
Your Message : I'm object A
```

The prototype scope

If the scope is set to prototype, the Spring IoC container creates a 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 scope for stateless beans.

To define a prototype scope, you can set the **scope** property to **prototype** in the bean configuration file, as shown in the following code snippet –

```
<!-- A bean definition with prototype scope -->
<bean id = "..." class = "..." scope = "prototype">
   <!-- collaborators and configuration for this bean go here -->
</bean>
```

Example

Let us have working Eclipse IDE in place and follow the following steps to create a Spring application –

Steps	Description
1	Create a project with a name *SpringExample* and create a package *com.tutorialspoint* under the **src** folder in the created project.
2	Add required Spring libraries using *Add External JARs* option as explained in the *Spring Hello World Example* chapter.
3	Create Java classes *HelloWorld* and *MainApp* under the *com.tutorialspoint* package.
4	Create Beans configuration file *Beans.xml* under the **src** folder.
5	The final step is to create the content of all the Java files and Bean Configuration file and run the application as explained below.

Here is the content of **HelloWorld.java** file

```
package com.tutorialspoint;

public class HelloWorld {
   private String message;

   public void setMessage(String message){
      this.message  = message;
   }
   public void getMessage(){
      System.out.println("Your Message : " + message);
   }
}
```

Following is the content of the **MainApp.java** file –

```
package com.tutorialspoint;

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

public class MainApp {
   public static void main(String[] args) {
      ApplicationContext context = new ClassPathXmlApplicationContext("Beans.xml");
      HelloWorld objA = (HelloWorld) context.getBean("helloWorld");

      objA.setMessage("I'm object A");
      objA.getMessage();

      HelloWorld objB = (HelloWorld) context.getBean("helloWorld");
      objB.getMessage();
   }
}
```

Following is the configuration file **Beans.xml** required for prototype scope –

```
<?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 = "helloWorld" class = "com.tutorialspoint.HelloWorld" scope = "prototype">
   </bean>

</beans>
```

Once you are done creating the source and bean configuration files, let us run the application. If everything is fine with your application, it will print the following message –

```
Your Message : I'm object A
Your Message : null
```