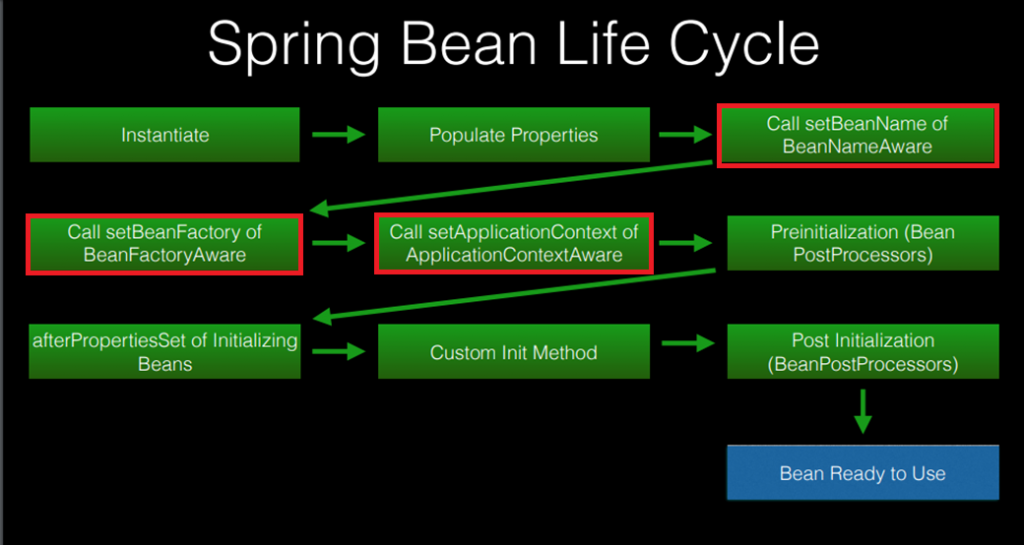
Do you know, how actually a bean object creates ?

To get the answer of above question we have to learn the spring bean life cycle.

Actually what happens is !

Spring takes some steps before serve a bean object to our application. Where spring calls some methods.

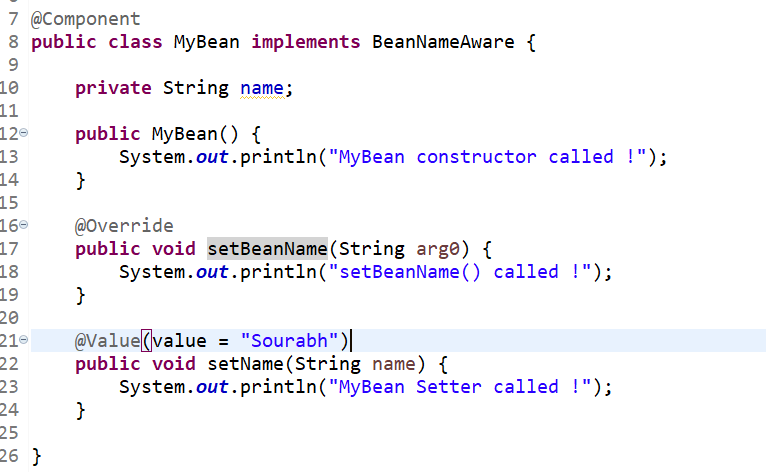
Have a high level overview of bean lifecycle by look up the below diagram :



Step 1 : First spring instantiate the bean object.

Step 2 : After that whatever dependencies required to the bean injects by spring.

Step 3 : As a 3rd step, spring calls the method setBeanName() of BeanNameAware interface.



The above all three step will be taken by spring when spring creates the object of MyBean class shown in the above picture.

As you can see down below, first constructor being called, then setter and then setBeanName(). It is prove that setBeanName() calls after instantiation and setting the property of a bean.

Text

Description automatically generated

**Use of BeanNameAware interface ?**

We can’t access the bean id or name inside the bean class itself.

To do so, we have to implements this interface and spring will call its method setBeanName() by passing the bean name as argument.

Further, we can set this argument in our data member.

Graphical user interface, text, application

Description automatically generated

Step 4 : As a 4th step, spring calls the setBeanFactory() method of BeanFactoryAware interface.

In the same way, spring will call the setBeanFactory() method by passing the BeanFactory object as a argument and similarly we can store this object into a data member.

We don’t need to cover the use of BeanFactory because it is legacy of ApplicationContext. But it is the part of bean lifecycle we are just going through it.

Graphical user interface, text, application

Description automatically generated

Step 5 : As a 5th step, spring will call the setApplicationContext() method which is coming from ApplicationContextAware interface.

Now assume, you may need another bean class object in your bean class.

To do so, there is one way that we can Autowired the bean object.

But what if we need to provide a method in our bean class which returns the number of bean present in IOC container also.

So the best way is we can do implement ApplicationContextAware and have ApplicationContext object in our bean class. Now using this object we can get the bean object without using autowired and also there is a method in ApplicationContext i.e. getBeanDefinitionCount() which returns the number of bean present in the IOC container.

Both task can done using this way !

Overall conclusion is, by implementing ApplicationContextAware interface in our bean class we’ll have application context object and we can call any method of it whichever we want.

Graphical user interface, text, application, email

Description automatically generated

**Remaining Steps**

The remaining steps will be covered here in one go.

To understand **BeanPostProcessor,** look at the below class :

**Graphical user interface, text, application, email

Description automatically generated**

Don’t be scared by looking at the above class, simply what we have done here is, just created a separate class with name **MyBeanPostProcessor** and it is implementing the **BeanPostProcessor** interface.

**BeanPostProcessor** interface has two default methods that have been overridden in the above class.

Now just annotate this class from @Component.

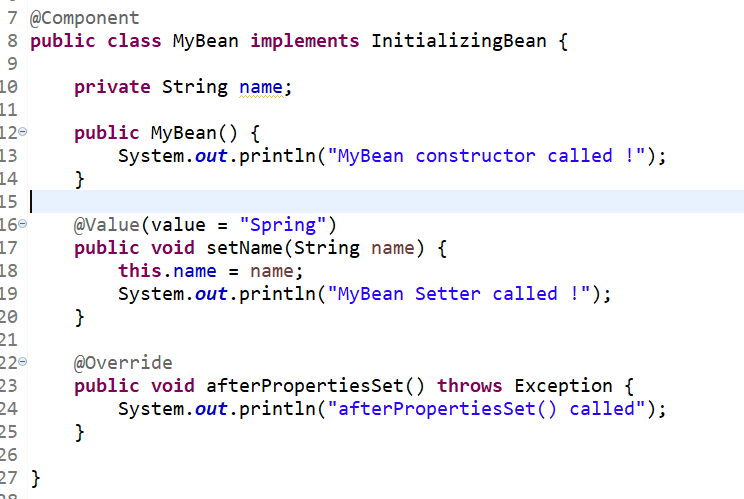
Step 6 : As a step six, spring will call the method postProcessBeforeInitialization() as a six step.

Note : After completion of 5th step, spring looks up for a class which is implementing BeanPostProcessor and then calls its postProcessBeforeInitialization() method.

Step 7 : As a next step, spring will call **afterPropertiesSet()** belong to IntializingBean interface.

We have already discussed about this interface when we were discussing the lifecycle method using interface.

So just after postProcessBeforeInitialization() method the **afterPropertiesSet()** method calls.



Step 8 : And just after **afterPropertiesSet()** methods call, spring will call the BeanPostProcessor second method i.e. postProcessAfterInitialization()

And this is the last step of spring lifecycle. After this, your bean is ready to use.

A picture containing graphical user interface

Description automatically generated

Text

Description automatically generated with medium confidence