**Bean Life Cycle**

**Life cycle**: life cycle means it has life start to end which have to live (execute in sequence).

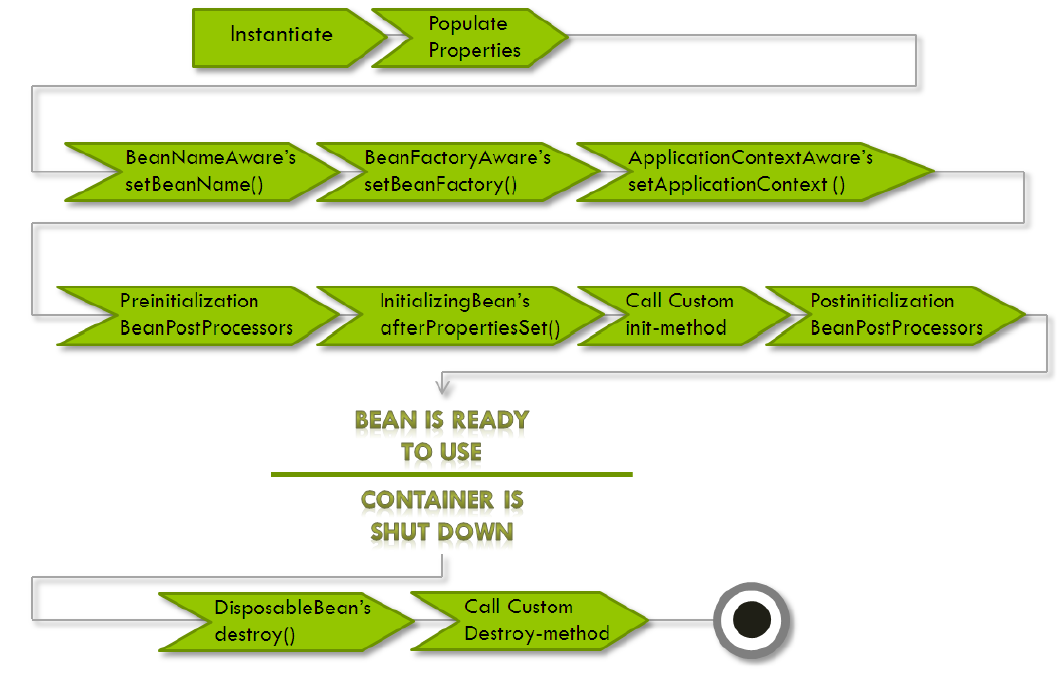
Same Bean also has life cycle which are going to discuss below:

**Bean**: Spring beans are just object instances that are managed by the spring container, namely, they are created and wired by the framework and put into a "bag of objects" (the container) from where you can get them later.

The "wiring" part there is what dependency injection is all about, what it means is that you can just say "I will need this thing" and the framework will follow some rules to get you the proper instance.

In spring, those objects that form the backbone of your application and that are managed by the Spring IoC container are referred to as beans. A bean is simply an object that is instantiated, assembled and otherwise managed by a Spring IoC container.

This is a diagram of bean life cycle how bean works in spring: and discuss one by one all below



The life cycle of a spring bean is easy to understand. When a bean is instantiated, it may be required to perform some initialization to get it into a usable state. Similarly, when the bean is no longer required and is removed from the [container](http://www.dineshonjava.com/2012/06/spring-ioc-container.html), some cleanup may be required.

**Step 1:** load bean and instantiated the bean

When run client App ( i.e. SpringDOJMainClass.java) and create the object of ApplicationContext then load the configuration file (i.e. bean.xml) and instantiate all the beans which is defined in this configuration file.

But if we are using BeanFactory then configuration file will not be loaded automatically it will be loaded when you will call getBean(…..) method.

**Step2**: Populate properties- Spring IoC container injects the bean’s properties.

If any listeners is using then inject those.

**Step3**: Set Bean Name- Spring container sets the bean name. If the bean implements BeanNameAware, spring container passes the bean’s id to setBeanName() method.

**Step4**: Set Bean Factory-If the bean implements BeanFactoryAware, Spring container passes theBeanFactory to setBeanFactory().

In case of BeanFactoryAware implementation otherwise

**OR**

**Step4**: Set ApplicationContext- If the bean implements ApplicationContextAware

setApplicationContext method of ApplicationContextAware class. Set the ApplicationContext that this object runs in.

If you want to use configuration file data without loading again this file then you should implement this interface to get ApplicationContext object.

In your project any class want to access ApplicationContext object without load configuration file then you have to implement this interface.

**Step5**: Pre Initialization-This stage is also called the bean postprocess . If there are any BeanPostProcessors, theSpring container calls the postProcesserBeforeInitialization () method.

Means If there are any BeanPostProcessors associated with the BeanFactory that loads the Bean, then the spring will call postProcessBeforeInitialization() method before the properties for the Bean are injected.

This is BeanFactory Specific you should implement in separate class.

**Step6**: Initialize beans- If the bean implements IntializingBean,its afterPropertySet()method is called. If the bean has init method declaration, the specified initialization method is called.

Means If the Bean class implements the InitializingBean interface, then the spring will callafterPropertiesSet() method once all the Bean properties defined in the Configuration file are injected.

**Step7**: If there is any custom init-method declared in the configuration file, that method will be called.

This is custom method which is called after afterPropertiesSet method this is alternate of afterpropertiesSet method. The name of the custom initialization method to invoke after setting an properties. The method must have no arguments, but may throw any exception. This is an alternative to implementing Spring's InitializingBean interface or marking a method with the PostConstruct annotation.

**Step8:** Post Initialization- IfBeanPostProcessors is implemented by the bean, the Spring container calls their postProcessAfterinitalization() method.

Means If there are any BeanPostProcessors associated with the BeanFactory that loads the Bean, then the spring will call postProcessAfterInitialization() method.

Same like pre Initialization.This is BeanFactory Specific you should implement in separate class.

**Step9**: Ready to Use- Now the bean is ready to be used by the application.

**Step10**: Destroy- The bean is destroyed during this stage. If the bean implements DisposableBean, the Spring IoC container will call the destroy() method . If a custom destroy () method is defined, the container calls the specified method.

This method is called if you close the session of applicationcontext( by calling registerShutdownHook(); of AbstractApplicationContext or by calling the close() method of ClassPathXmlApplicationContext) then destroy method will be called.