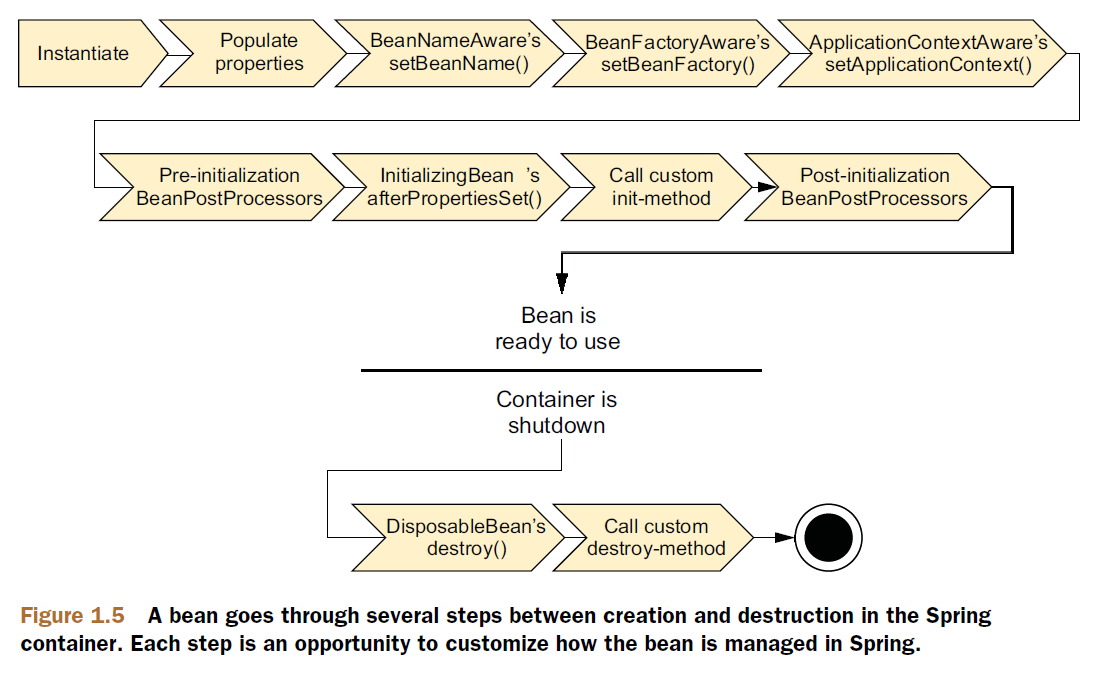
***A bean’s life:***

In a traditional Java application, the lifecycle of a bean is simple. Java’s *new* keyword is used to instantiate the bean, and it’s ready to use. Once the bean is no longer in use, it’s eligible for garbage collection and eventually goes to the big bit bucket in the sky.

* In contrast, the lifecycle of a bean in a Spring bean is more elaborate. It’s advantage of some of the opportunities that Spring bean, because you may want to take advantage of some of the opportunities that spring offers to customize how a bean is created.



* Let’s break it down into more details:
  + Spring instantiates the bean.
  + Spring injects values and bean references into the bean’s properties.
  + If the bean implements *BeanNameAware*, Spring passes the bean’s ID to the *setBeanName*() method.
  + If the bean implements *BeanFactoryAware*, Spring calls the setBeanFactory*()* method, passing in the bean factory itself.
  + If the bean implements *ApplicationContextAware,* Spring calls the *setApplicationContext()* method, passing in a reference to the enclosing application context.
  + If the bean implements *BeanPostProcessor,* Spring calls its *postProcessAfterInitialization()* method.
  + At this point, the bean is ready to be used by the application and remains in the application context until the application context is destroyed.
  + If the bean implements the *DisposableBean* interface, Spring calls tis destroy() method. Likewise, if the bean was declared with a *destroyed-method,*  the specified method is called.