Bean lifecycle means, there are some methods is going to be call when the bean object will create and destroy.

There are 3 ways to manage bean lifecycle methods :

1. **Using XML**
2. **Using Annotation**
3. **By Implementing Interfaces**

The 3rd approach is least recommended i.e. **Implementing interface.** But we will still learn it for interview point of view.

Using XML

**Example 1**

**Person.java :**

package com.beanlifecycle.xml;

public class Person {

public void init() {

System.out.println("Entry - Take entry coin");

}

public void chilling() {

System.out.println("Consuming alcohol, dancing and watching beauties");

}

public void destroy() {

System.out.println("Exit - Give back entry coin");

}

}

Here we are introducing two special methods i.e. **init()** and **destroy()**

Although we can give any name to them but init() and destroy() is recommended one.

The **init()** method will always run after successfully creation of object and all the dependencies has been injected.

The **destroy()** method will always run before when the bean object is going to be dead.

**How will the bean object become dead ?**

When we call the **close()** method of **ClassPathXmlApplicationContext** class.

**Config.xml :**

<?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.xsd">

<bean id="person" class="com.beanlifecycle.xml.Person" init-method="init" destroy-method="destroy" />

</beans>

Here we are introducing two new <bean> tag attributes,

init-method="init"

This attribute will take the name of your init method. As we have discussed we can give any name to our inti method but it is recommended to our init method should always be named as **init().**

destroy-method="destroy"

This attribute will take the name of your destroy method.

**Bar.java :**

public class Bar {

public static void main(String[] args) {

ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("com/beanlifecycle/xml/config.xml");

Person p = context.getBean("person", Person.class);

p.chilling();

context.close();

}

}

**Example 2**

**Girl.java :**

public class Girl {

public void init() {

System.out.println("Girl Entry - Take entry coin");

}

public void chilling() {

System.out.println("Girl - Consuming alcohol, dancing and watching handsomes");

}

public void destroy() {

System.out.println("Girl Exit - Give back entry coin");

}

}

**Boy.java :**

public class Boy {

public void init() {

System.out.println("Boy Entry - Take entry coin");

}

public void chilling() {

System.out.println("Boy - Consuming alcohol, dancing and watching beauties");

}

public void destroy() {

System.out.println("Boy Exit - Give back entry coin");

}

}

**Config.xml :**

<?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.xsd">

<bean id="girl" class="com.beanlifecycle.xml2.Girl" init-method="init" destroy-method="destroy" />

<bean id="boy" class="com.beanlifecycle.xml2.Boy" init-method="init" destroy-method="destroy" />

</beans>

**Bar.java :**

public class Bar {

public static void main(String[] args) {

ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("com/beanlifecycle/xml2/config.xml");

Girl girl = context.getBean("girl", Girl.class);

girl.chilling();

Boy boy = context.getBean("boy", Boy.class);

boy.chilling();

context.close();

}

}

In this particular example, the noticeable thing is that we are mentioning the **init-method** and **destroy-method** attribute for both the beans whereas the name of init and destroy method in both the classes is similar.

It is okay if we have less number bean classes present but it will be hectic and not recommended for large number of classes.

**So what is best solution ?**

First of all, spring recommend that always create init and destroy method in your every bean class with the name **init()** and **destroy()**. By which every bean class will have init and destroy method with same name.

Now if we follow this spring recommendation we can use a **<beans>** tag attributes called **default-init-method** and **default-destroy-method.** And now spring can easily detect that every bean containing init and destroy method with same name as you have provided inside these attributes.

<?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.x sd"

default-destroy-method="destroy" default-init-method="init">

<bean id="girl" class="com.beanlifecycle.xml3.Girl" />

<bean id="boy" class="com.beanlifecycle.xml3.Boy" />

</beans>