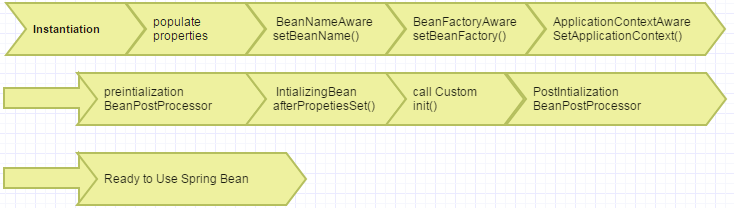
**Spring Bean Life Cycle:**

Spring bean instantiation to destruction there is a series of steps has been performed by Spring container. Each stage has its own predefined task. The really good thing about Spring is we can hook to every stage of Spring through callback methods.

Callback method in this context is developer creates a method and provide some instructions for a stage and telling spring container to execute those instructions when the Spring bean reaches to that particular stage of life cycle.

**Let see the different stages Spring bean**





**Instantiation**: Spring container instantiate Spring Bean.

**Populate Properties**: Spring container set the properties of the bean by injection (from Spring configuration XML).

**SetBeanName** : Set the identifier of the bean.

**SetBeanFactory** : If BeanFactoryAware has been implemented , Container passes the BeanFactory context into the bean.

**SetApplicationContext** : If ApplicationContextAware has been implemented , Container passes the Applicationcontext into the bean.

**PreIntialization** : If BeanPostPeocessor interface has been implemented in a bean container calls bean’s postProcessBeforeIntialization method for doing any preprocess stuff.

**afterPropertySet** : If the old intializingBean interface has been implemented in a bean

it calls the afterPropertySet method of that interface to do any initialization stuff after bean property has been set.

init() : call init method for any initialization stuff.

**PostIntializaton**  : If BeanPostPeocessor interface has been implemented in a bean container calls bean’s postProcessAfterIntialization method for doing any post processing stuff.

**Destroy**: If old disposableBean interface has been implemented by a bean

it calls destroy method of that interface to do any pre destroy stuff before bean deletion from the container.

**Custom Destroy**: call destroy method for any pre-destroy stuff.

While in Later chapter we talking about PostProcessor and Aware interfaces.

Now we will concentrate on the two important callback method in SpringBean Lifecycle

**1. Init**

**2. Destroy**

In ancient days of spring, this can be achieved through implementing InitializingBean and DisposableBean interface

Any pre-initialization instruction was written in afterPropetiesSet() method and any pre destroy stuff was written in destroy() method.

But Now no one uses this as by implementing those interfaces you make your POJO

As Spring dependent POJO. So later If you thought you will not be going to use Spring or want to re-use that POJO in other Application, refactoring has been needed.

**TIP: Please never ever use those InitializingBean, DisposableBean interfaces.**

**init and destroy are mainly use for opening File stream, database connection or closing those costly resources.**

**Problem: Let we try to solve a problem, we want Currency will be showing with its symbol where the input is the amount and the country name.Initdestroy.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-3.0.xsd"*>

<bean id=*"helloWorld"* class=*"com.example.initDestoty.InitDestroyTestBean"* init-method=*"init"*>

<property name=*"price"* value=*"50"*/>

<property name=*"countryName"* value=*"US"*/>

</bean>

</beans>

Java :

package com.example.initDestoty;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class InitDestroyTestBean {

int price;

String countryName;

String msg;

public void init()

{

if("US".equalsIgnoreCase(countryName))

{

this.msg=price + "$";

}

else

{

this.msg=price + "Rupees";

}

}

public void destroy()

{

System.out.println("Call destroy");

}

public String getCountryName() {

return countryName;

}

public void setCountryName(String countryName) {

this.countryName = countryName;

}

public int getPrice() {

return price;

}

public void setPrice(int price) {

this.price = price;

}

public String getMsg() {

return msg;

}

public void setMsg(String msg) {

this.msg = msg;

}

public static void main(String[] args) {

ApplicationContext ctx = new ClassPathXmlApplicationContext("configFiles/initdestroy.xml");

InitDestroyTestBean bean =(InitDestroyTestBean) ctx.getBean("helloWorld");

System.out.println(bean.getMsg());

}

}

Output: 50$

Look at the init method based on the country I populate the msg property as the country is the US we show 50$ as output.