

Spring @PostConstruct and @PreDestroy example

By mkyong (<http://www.mkyong.com/author/mkyong/>) | March 29, 2010 | Updated : June 13, 2011

In Spring, you can either implements InitializingBean and DisposableBean (<http://www.mkyong.com/spring/spring-initializingbean-and-disposablebean-example/>) interface or specify the init-method and destroy-method (<http://www.mkyong.com/spring/spring-init-method-and-destroy-method-example/>) in bean configuration file for the initialization and destruction callback function. In this article, we show you how to use annotation **@PostConstruct** and **@PreDestroy** to do the same thing.

Note

The **@PostConstruct** and **@PreDestroy** annotation are not belong to Spring, it's located in the J2ee library – common-annotations.jar.

@PostConstruct and @PreDestroy

A CustomerService bean with @PostConstruct and @PreDestroy annotation

```
package com.mkyong.customer.services;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;

public class CustomerService
{
    String message;

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    @PostConstruct
    public void initIt() throws Exception {
        System.out.println("Init method after properties are set : " + message);
    }

    @PreDestroy
    public void cleanUp() throws Exception {
        System.out.println("Spring Container is destroy! Customer clean up");
    }
}
```

By default, Spring will not aware of the `@PostConstruct` and `@PreDestroy` annotation. To enable it, you have to either register '**CommonAnnotationBeanPostProcessor**' or specify the '**<context:annotation-config />**' in bean configuration file,

1. CommonAnnotationBeanPostProcessor

```
<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-2.5.xsd">

    <bean class="org.springframework.context.annotation.CommonAnnotationBeanPostProcessor" />

    <bean id="customerService" class="com.mkyong.customer.services.CustomerService">
        <property name="message" value="i'm property message" />
    </bean>

</beans>
```

2. <context:annotation-config />

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-2.5.xsd">

  <context:annotation-config />

  <bean id="customerService" class="com.mkyong.customer.services.CustomerService">
    <property name="message" value="i'm property message" />
  </bean>

</beans>
```

Run it

```
package com.mkyong.common;

import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.mkyong.customer.services.CustomerService;

public class App
{
    public static void main( String[] args )
    {
        ConfigurableApplicationContext context =
            new ClassPathXmlApplicationContext(new String[] {"Spring-Customer.xml"});

        CustomerService cust = (CustomerService)context.getBean("customerService");

        System.out.println(cust);

        context.close();
    }
}
```

Output

```
Init method after properties are set : im property message
com.mkyong.customer.services.CustomerService@47393f
...
INFO: Destroying singletons in org.springframework.beans.factory.
support.DefaultListableBeanFactory@77158a:
defining beans [customerService]; root of factory hierarchy
Spring Container is destroy! Customer clean up
```

The **init()** method (**@PostConstruct**) is called, after the message property is set, and the **cleanup()** method (**@PreDestroy**) is call after the context.close();

Download Source Code

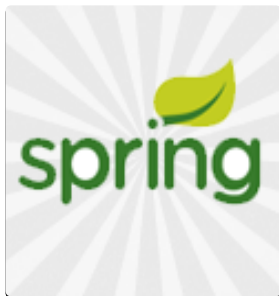
Download It – Spring-PostConstruct-PreDestroy-Example.zip
 (<http://www.mkyong.com/wp-content/uploads/2010/03/Spring-PostConstruct-PreDestroy-Example.zip>)

Tags : [spring \(http://www.mkyong.com/tag/spring/\)](http://www.mkyong.com/tag/spring/)

Share this article on

Twitter ([https://twitter.com/intent/tweet?text=Spring @PostConstruct and @PreDestroy example&url=http://www.mkyong.com/spring/spring-postconstruct-and-predestroy-example/&via=mkyong](https://twitter.com/intent/tweet?text=Spring%20@PostConstruct%20and%20@PreDestroy%20example&url=http://www.mkyong.com/spring/spring-postconstruct-and-predestroy-example/&via=mkyong)) Facebook (<https://www.facebook.com/sharer/sharer.php?u=http://www.mkyong.com/spring/spring-postconstruct-and-predestroy-example/>) Google+ (<https://plus.google.com/share?url=http://www.mkyong.com/spring/spring-postconstruct-and-predestroy-example/>)

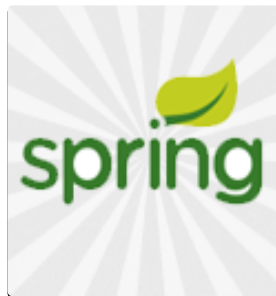
Reader also read :



Spring – Mixing XML and JavaConfig
 (<http://www.mkyong.com/spring/spring-mixing-xml-and-javaconfig/>)



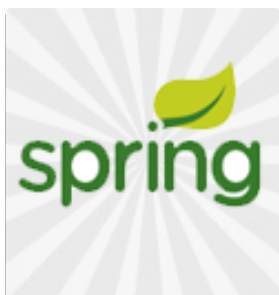
Spring embedded database examples
 (<http://www.mkyong.com/spring/spring-embedded-database-examples/>)



Spring MethodInvokingFactoryBean
 (<http://www.mkyong.com/spring/spring-methodinvokingfactorybean-example/>)



Spring @Value – Import a list from properties file
 (<http://www.mkyong.com/spring/spring-value-import-a-list-from-properties-file/>)



Spring
 @PropertySource