关于在spring  容器初始化 bean 和销毁前所做的操作定义方式有三种：

## [第一种：通过@PostConstruct 和 @PreDestroy 方法 实现初始化和销毁bean之前进行的操作](http://write.blog.csdn.net/postedit/8681497)（看我的test类，里面有用到service类）

## 第二种是：[通过 在xml中定义init-method 和  destory-method方法](http://blog.csdn.net/topwqp/article/details/8681467)

## 第三种是：[通过bean实现InitializingBean和 DisposableBean接口](http://blog.csdn.net/topwqp/article/details/8681573)

|  |
| --- |
| @Service 相当于注册javabean  public class Test {  @PostConstruct  public void test1(){  System.out.println("12345123456789023456789234567891234567aaaaaaaaaaaaaaaaaaaaaaaaaa");  }    @PreDestroy  public void test2(){  System.out.println("12345123456789023456789234567891234567bbbbbbbbbbbbbbbbbbbbbbbbbb");  }  } |

下面演示通过  @PostConstruct 和 @PreDestory

1:定义相关的实现类：

**[java]** [view plain](http://blog.csdn.net/topwqp/article/details/8681497) [copy](http://blog.csdn.net/topwqp/article/details/8681497)

1. **package** com.myapp.core.annotation.init;
3. **import** javax.annotation.PostConstruct;
4. **import** javax.annotation.PreDestroy;
6. **public** **class** PersonService {
8. **private** String  message;
10. **public** String getMessage() {
11. **return** message;
12. }
14. **public** **void** setMessage(String message) {
15. **this**.message = message;
16. }
18. @PostConstruct
19. **public** **void**  init(){
20. System.out.println("I'm  init  method  using  @PostConstrut...."+message);
21. }
23. @PreDestroy
24. **public** **void**  dostory(){
25. System.out.println("I'm  destory method  using  @PreDestroy....."+message);
26. }
28. }

2：定义相关的配置文件：

**[html]** [view plain](http://blog.csdn.net/topwqp/article/details/8681497) [copy](http://blog.csdn.net/topwqp/article/details/8681497)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xsi:schemaLocation="http://www.springframework.org/schema/beans
6. http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
7. http://www.springframework.org/schema/context
8. http://www.springframework.org/schema/context/spring-context-3.1.xsd"**>**
10. <!-- <context:component-scan  base-package="com.myapp.core.jsr330"/> -->
12. **<context:annotation-config** **/>**
14. **<bean** id="personService" class="com.myapp.core.annotation.init.PersonService"**>**
15. **<property** name="message" value="123"**></property>**
16. **</bean>**
18. **</beans>**

其中<context:annotation-config />告诉spring 容器采用注解配置：扫描注解配置；

测试类：

**[java]** [view plain](http://blog.csdn.net/topwqp/article/details/8681497) [copy](http://blog.csdn.net/topwqp/article/details/8681497)

1. **package** com.myapp.core.annotation.init;
3. **import** org.springframework.context.ApplicationContext;
4. **import** org.springframework.context.support.ClassPathXmlApplicationContext;
6. **public** **class** MainTest {
8. **public** **static** **void** main(String[] args) {
10. ApplicationContext  context = **new** ClassPathXmlApplicationContext("resource/annotation.xml");
12. PersonService   personService  =  (PersonService)context.getBean("personService");
14. personService.dostory();
15. }
17. }

测试结果：

I'm  init  method  using  @PostConstrut....123  
I'm  destory method  using  @PreDestroy.....123

其中也可以通过申明加载org.springframework.context.annotation.CommonAnnotationBeanPostProcessor

类来告诉Spring容器采用的 常用 注解配置的方式：

只需要修改配置文件为：

**[html]** [view plain](http://blog.csdn.net/topwqp/article/details/8681497) [copy](http://blog.csdn.net/topwqp/article/details/8681497)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xsi:schemaLocation="http://www.springframework.org/schema/beans
6. http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
7. http://www.springframework.org/schema/context
8. http://www.springframework.org/schema/context/spring-context-3.1.xsd"**>**
10. <!-- <context:component-scan  base-package="com.myapp.core.jsr330"/> -->
12. <!-- <context:annotation-config /> -->

15. **<bean** class="org.springframework.context.annotation.CommonAnnotationBeanPostProcessor" **/>**
16. **<bean** id="personService" class="com.myapp.core.annotation.init.PersonService"**>**
17. **<property** name="message" value="123"**></property>**
18. **</bean>**


22. **</beans>**

同样可以得到以上测试的输出结果。