Spring

# Spring介绍

## 1Spring基础概念

### 1.1 什么是Spring

Spring是分层的JavaSE/EE full-stack 轻量级开源框架

JavaEE中程序是被分为三层 WEB层【表现层】 、业务逻辑层、数据访问层【持久层】

\*\*\*\*Struts2:表现层 MVC的框架

\*\*\*\*Hibernate是数据访问层 完全的ORM框架

\*\*\*\*Spring框架 **包含了JavaEE三层的每一层解决方案 (一站式)**

\*\* Web层: SpringMVC

\*\*业务层 Spring对Bean管理、AOP、事务

\*\*持久层 SpringJdbcTempleate /ORM模块

### 1.2 Spring的核心

* 以IoC（Inverse of Control 反转控制）和AOP（Aspect Oriented Programming 面向切面编程为内核）
* IOC就是把创建对象的权限反转到Spring容器 由Spring进行管理对象的创建。

User user =new User();

User user =applicationContext.getBean(“xxx.xml”)； 通过配置文件进行反射的方式进行创对象

* AOP是面向切面的编程 OOP

DI：依赖注入

### 1.3Spring的由来

Rod Johson

* Spring的出现是为了取代EJB的臃肿、低效、脱离现实
  + *Expert One-to-One J2EE Design and Development （分析J2EE开发使用技术 EJB） 2002*
  + *Expert One-to-One J2EE Development without EJB (客观分析J2EE开发需要什么 推出了一个全新的框架 就是现在的Spring框架) 2004*

Spring 的出现就是解决JavaEE开发实际问题，轻量级的 相比EJB比较繁重 复杂的解决方案。

### 1.4Spring的优势

* 方便解耦，简化开发
  + Spring就是一个大工厂，可以将所有对象创建和依赖关系维护，交给Spring管理
* AOP编程的支持
  + Spring提供面向切面编程，可以方便的实现对程序进行权限拦截、运行监控等功能
* 声明式事务的支持
  + 只需要通过配置就可以完成对事务的管理，而无需手动编程
* 方便程序的测试
  + Spring对Junit4支持，可以通过注解方便的测试Spring程序
* 方便集成各种优秀框架
  + Spring不排斥各种优秀的开源框架，其内部提供了对各种优秀框架（如：Struts、Hibernate、MyBatis、Quartz等）的直接支持
* 降低JavaEE API的使用难度
  + Spring 对JavaEE开发中非常难用的一些API（JDBC、JavaMail、远程调用等），都提供了封装，使这些API应用难度大大降低

## 2.第一个Spring入门程序

### 2.1准备开发环境

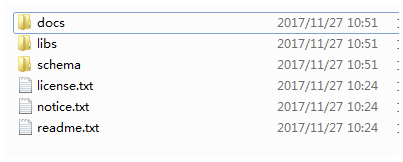
#### 2.1.1添加相关的开发jar包

**1）下载最新的jar包**

http://repo.spring.io/simple/libs-release-local/org/springframework/spring/

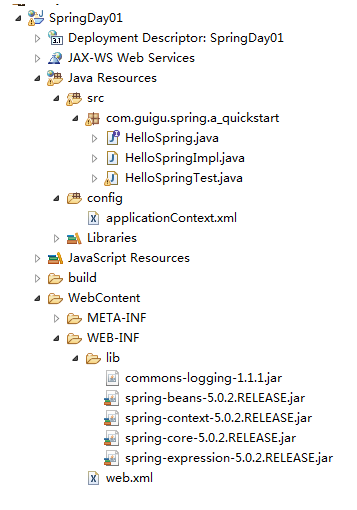
目前最新版本是2017.11.27发布的 5.0.2.RELEASE

2）Spring的目录结构

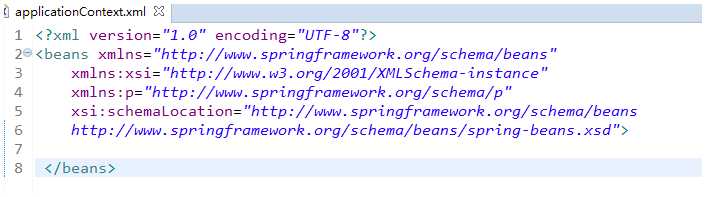


3）将Spring开发的基本jar包导入到工程内

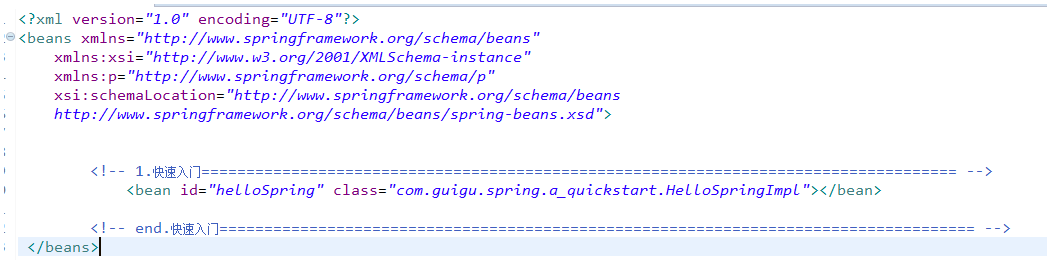
Beans/Core /context/ el /logging



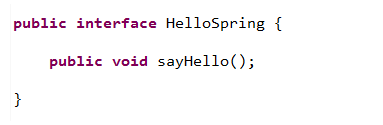
### 2.2准备配置文件

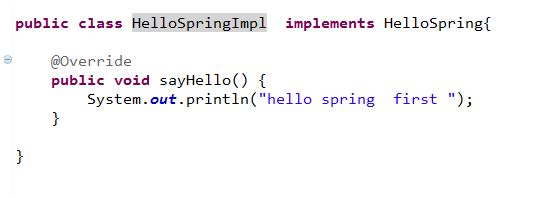


### 2.3编写配置文件信息



### 2.4编写类和接口

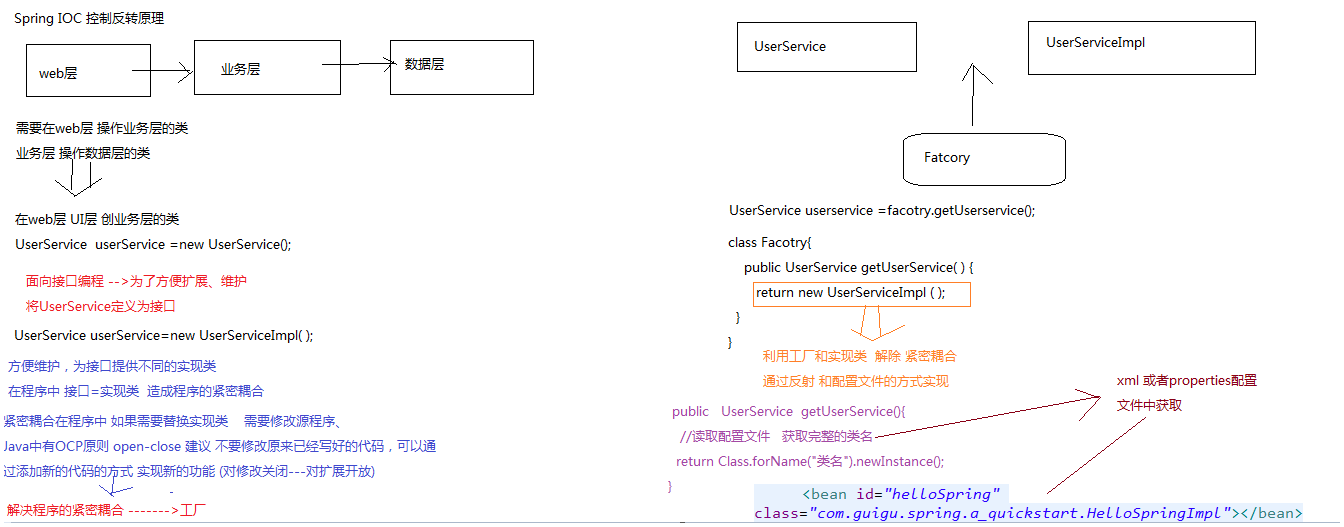




### 2.5编写测试类



### 2.6IOC的原理解释



# 二、IOC容器

IOC容器装配Bean 两种形式 第一种是基于xml配置文件的形式 第二种是基于注解的形式。

## 2.1 IOC装配Bean (XML方式)

Sring装配Bean的三种形式

1. 使用构造函数(有无参的构造函数)

<bean name="anotherExample" class="examples.ExampleBeanTwo"/>

2）使用静态工厂方法

<bean id="clientService"

class="examples.ClientService"

factory-method="createInstance"/>

3）使用实例工厂方法

<bean id="serviceLocator" class="examples.DefaultServiceLocator">

<!-- inject any dependencies required by this locator bean -->

</bean>

<!-- the bean to be created via the factory bean -->

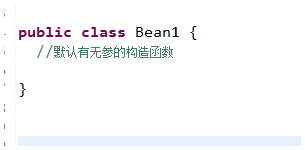
<bean id="clientService"

factory-bean="serviceLocator"

factory-method="createClientServiceInstance"/>

### 2.1.1使用构造函数装配Bean

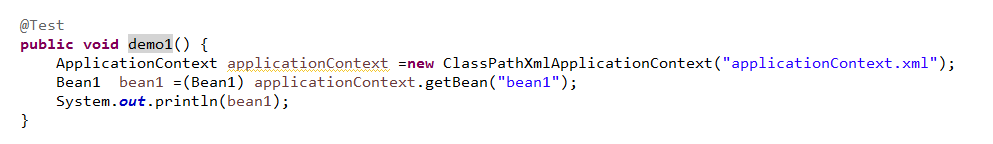
**类**



**配置文件**

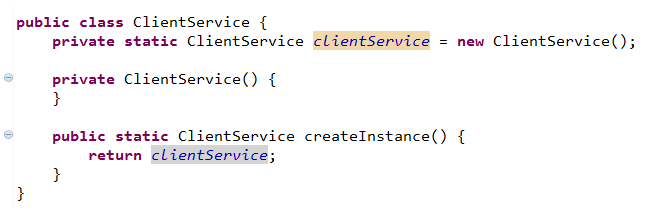


**测试类**



### 2.1.2使用静态工厂方法装配Bean

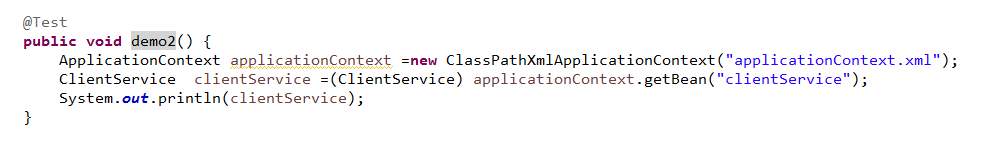
类



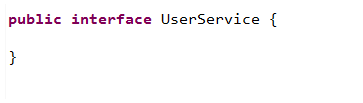
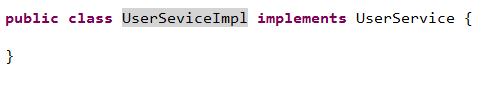
配置文件

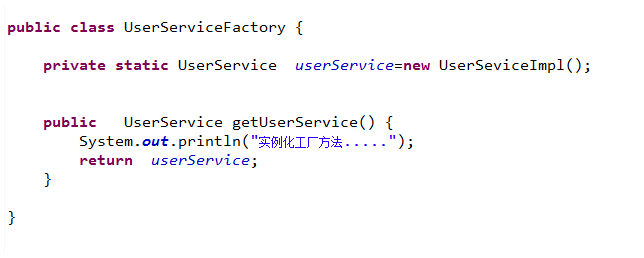


测试类

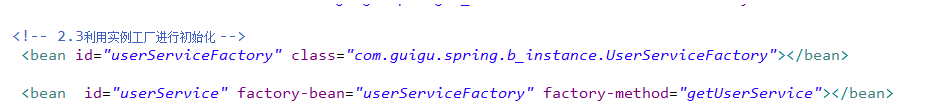


### 2.1.3使用实例化工厂方法

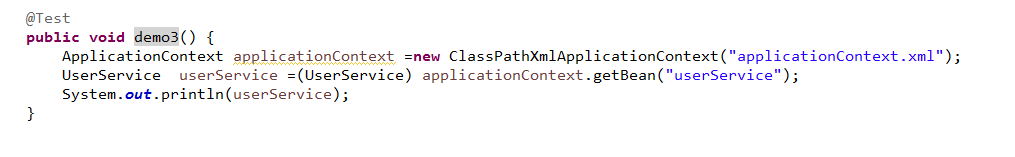
 



配置文件

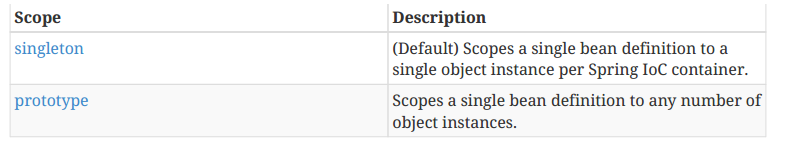


测试代码

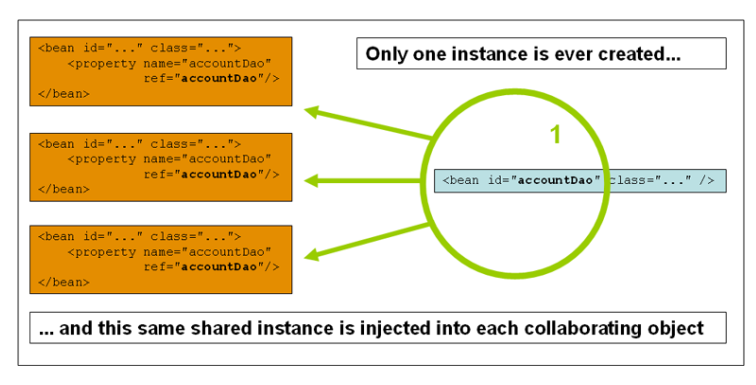


## 2.2 Bean的其他属性的设置

### 2.2.1 Bean的作用域



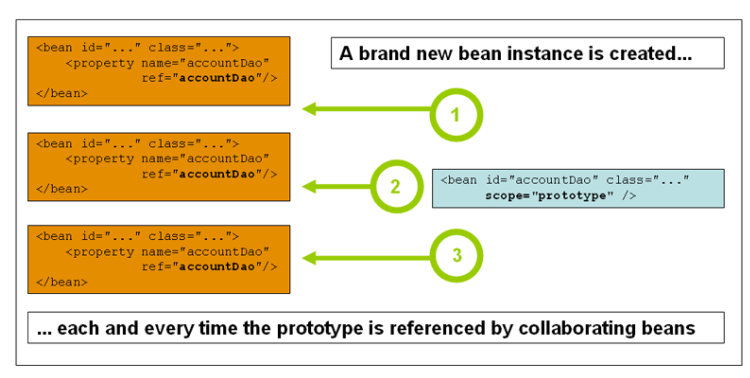
#### 2.2.1.1单例模式



**<bean id="accountService" class="com.foo.DefaultAccountService"/>**

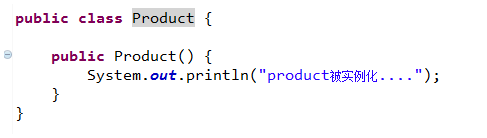
**<bean id="accountService" class="com.foo.DefaultAccountService" scope="singleton"/>**

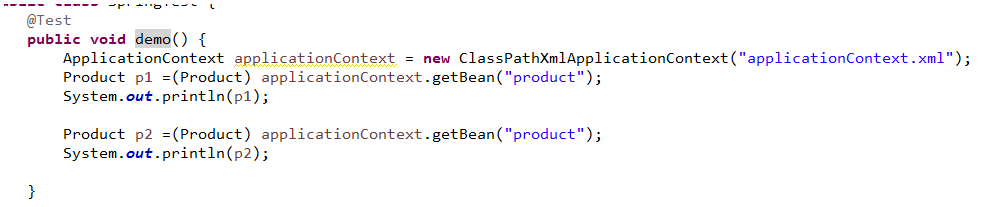
#### 2.2.1.2 多例模式



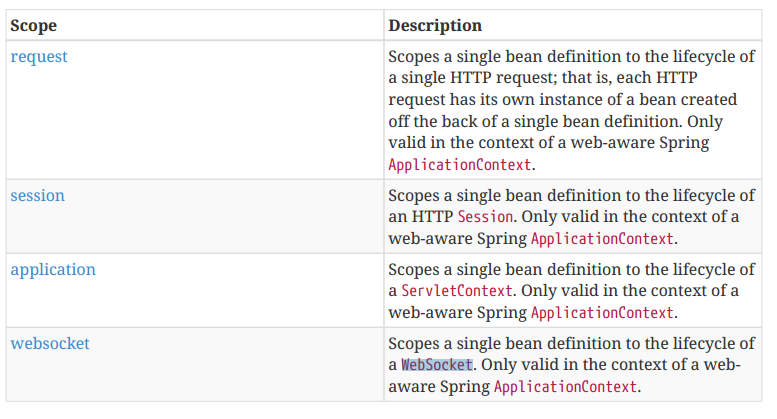
**<bean id="accountService" class="com.foo.DefaultAccountService" scope="prototype"/>**

**具体案例**





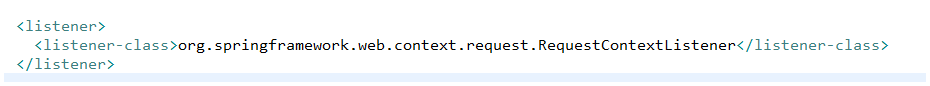
#### 2.2.1.3Web相关的作用域

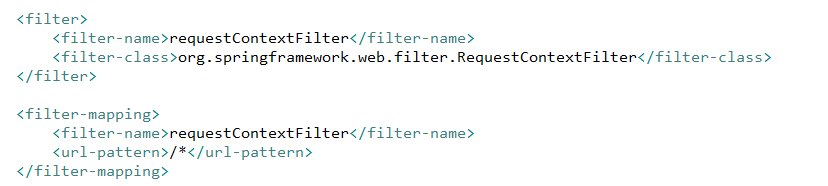


**Request作用域的设置**

首先配置 监听 然后在指定作用域

**在web.xml中配置RequestContextListener**

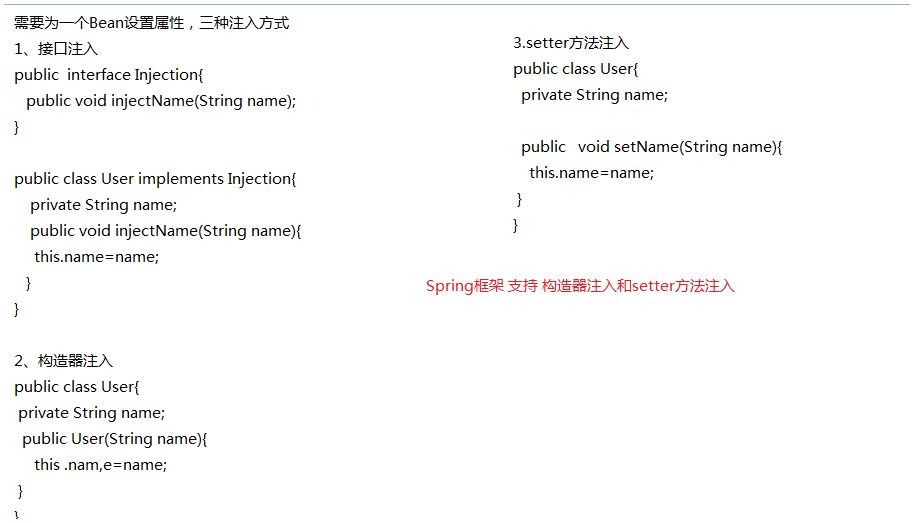




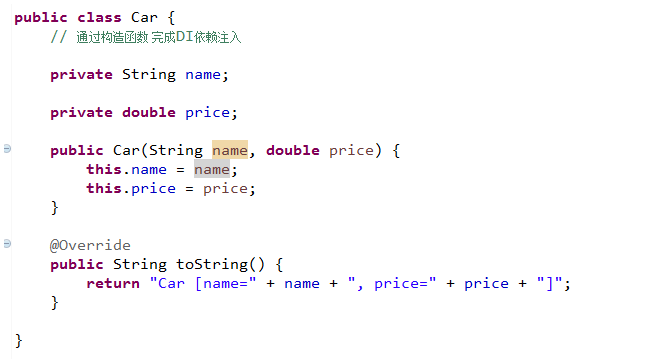
**在代码中指定作用域**



## 2.3 Bean属性的依赖注入



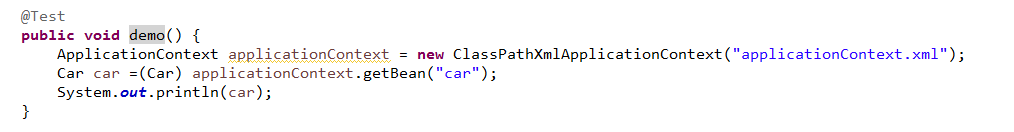
### 2.3.1构造器注入



**配置文件**

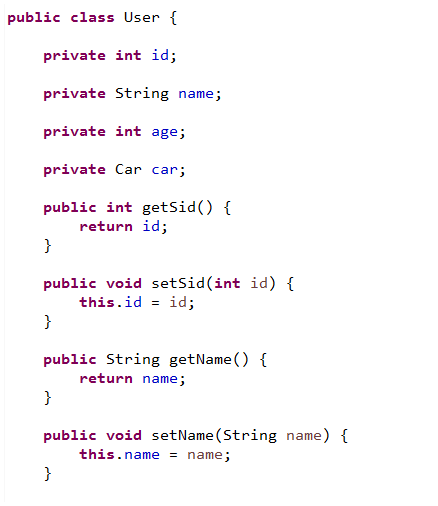


**测试代码**

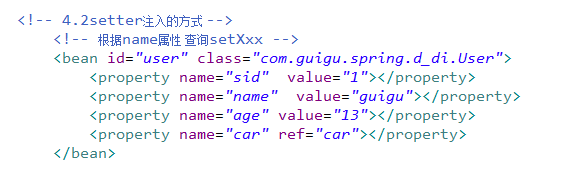


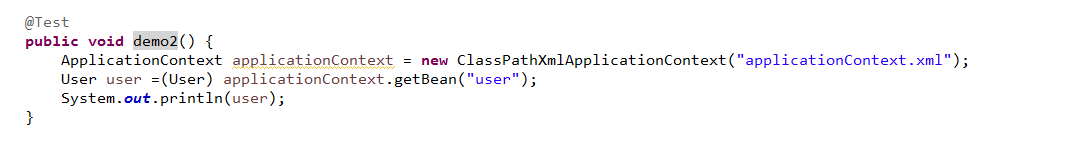
### 2.3.2Setter方法注入

类



配置文件

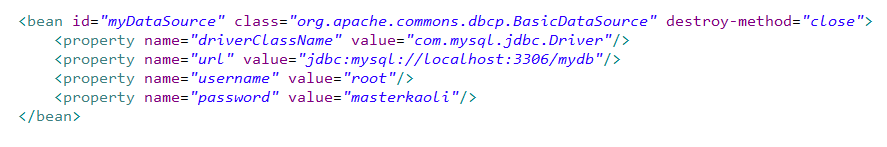




### 2.3.3依赖注入的细节

#### 2.3.3.1配置DBCP数据库连接池

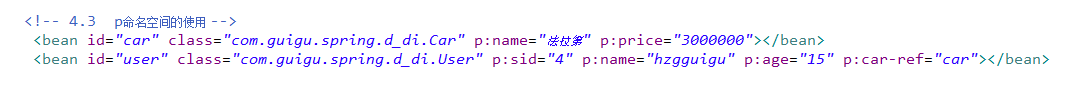
案例一 ：利用Spring配置DBCP数据库连接池



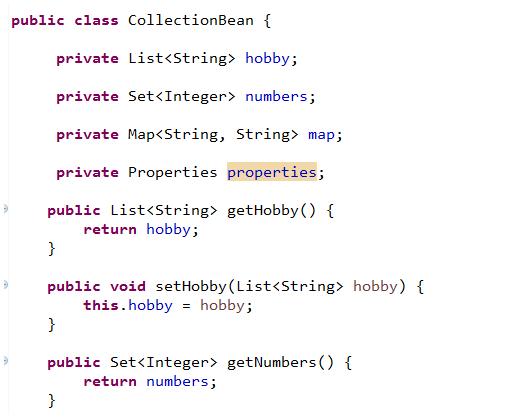
#### 2.3.3.2 p命名空间

把上方的配置DBCP数据库连接池 改为p命名空间的使用。





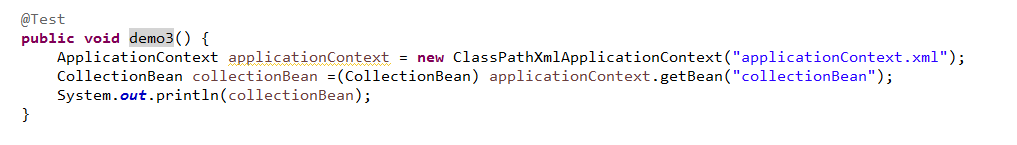
#### 2.3.3.3 集合属性的注入



配置文件



测试代码



#### 2.3.3.4 属性的合并

