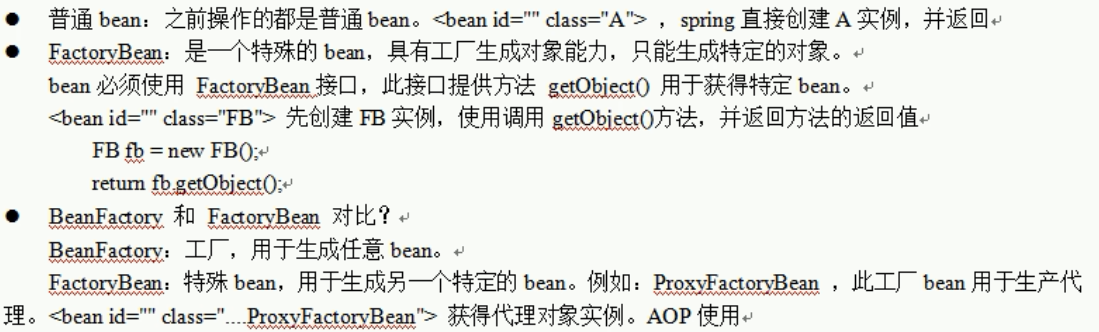
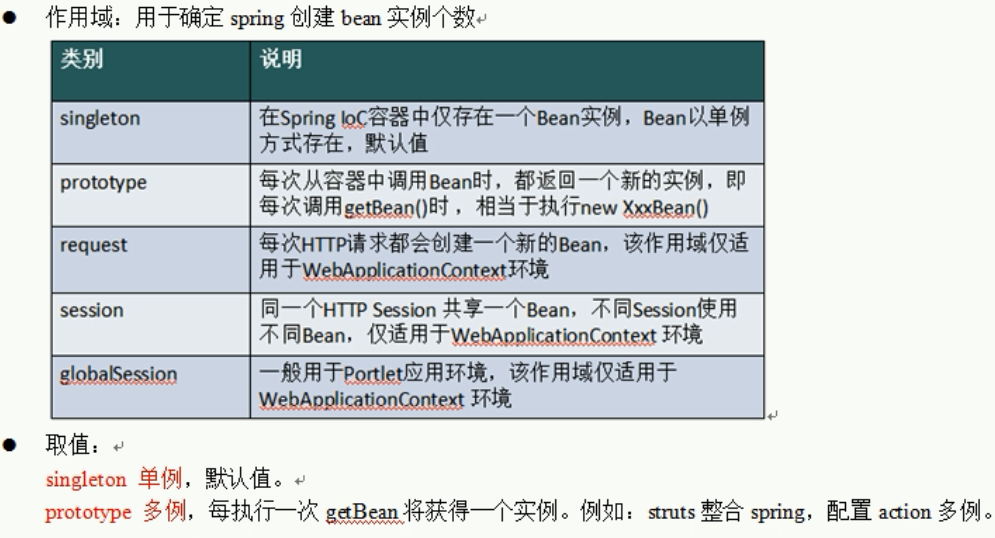
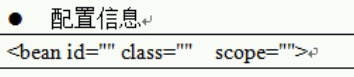
## Spring框架

1. Bean的种类

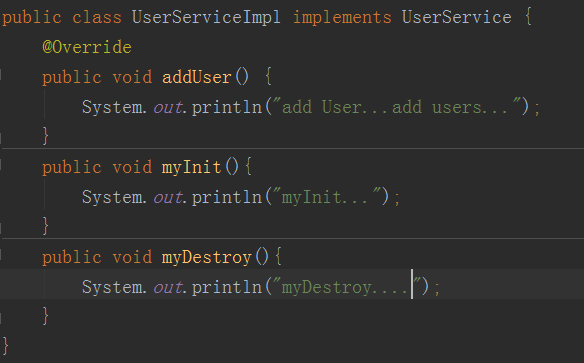


1. Spring容器中Bean的作用域（了解）

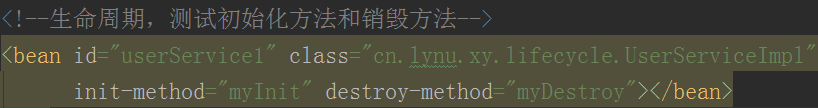




1. Spring容器中Bean的生命周期（11个步骤，了解）
2. 初始化方法和销毁方法
3. 定义接口和实现类



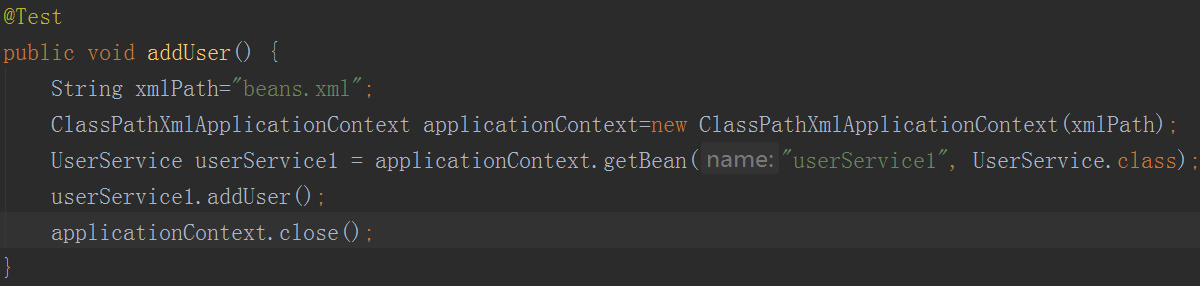
1. 在xml中配置



说明：class中的值是一个类的全限定类名，这个类是哪个，就表示创建哪个类的实例对象

比如：<bean id="a" class="A"></bean> ，就相当于A a = new A（）；

1. 测试



**说明：**

\* beans.xml配置文件默认从src下找（和mybatis一样）

\* ApplicationContext是一个接口，ClassPathXmlApplicationContext是该接口的一个实现类

\* 容器（applicationContext）必须close，必须是单例的，销毁方法才能这执行

\* 此方法ApplicationContext接口中没有定义，实现类中有，所以上面的代码中， applicationContext返回的类型改为实现类的类型，即ClassPathXmlApplicationContext

1. BeanProstProcessor后处理（没看懂？）
2. 使用setter方法注入属性值
3. 注入普通类型的值（两种方式）

\* 第一种： <bean id="person" class="cn.lynu.xy.setter.Person">

<property name="name" value="阳志"></property>

<property name="age" value="18"></property>

<property name="address" ref="homeAddress"></property>

</bean>

\* 第二种：<bean id="person" class="cn.lynu.xy.setter.Person">

<property name="name">

<value>阳志</value>

</property>

<property name="age">

<value>18</value>

</property>

</bean>

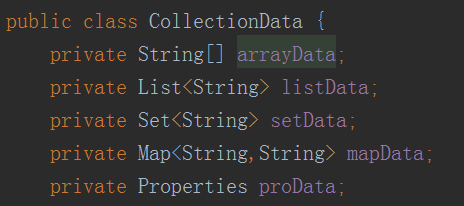
1. 注入引用类型的值（两种方式）

\* 第一种： <property name="address" ref="address"></property>

ref：表示引用的对象的id

\* 第二种：同上面第二种

1. P命名空间（了解，只是setter方法的简化，一般不用）
2. SpEL（了解）
3. 集合注入
4. 定义集合类



1. Xml中配置集合

集合的注入都是在 <property>中添加子标签

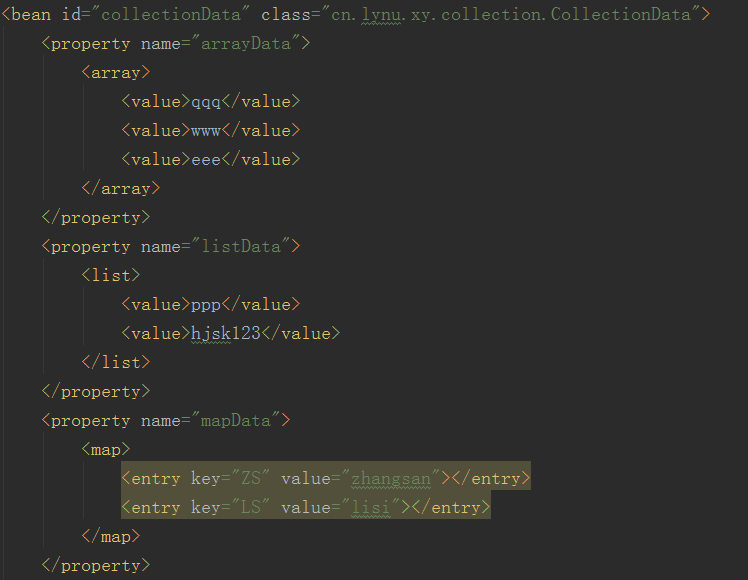
数组标签：<array>

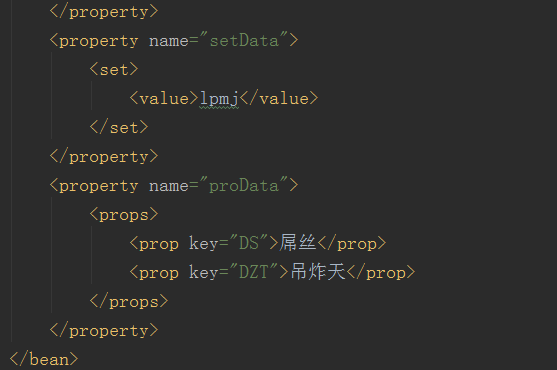
List标签：<list>

Set标签：<set>

Map标签：<map> 使用<entry>描述

Properties标签：<props> <prop key="">值</prop>：值写在标签体内





说明：集合的注入都是在 <property>中添加子标签

数组标签：<array>

List标签：<list>

Set标签：<set>

Map标签：<map> 使用<entry>描述

Properties标签：<props> <prop key="">值</prop>：值写 在标签体内

1. 测试

