# Spring

1. Spring的开发步骤：
   1. 导入坐标
   2. 创建Bean
   3. 创建配置文件applicationContext.xml
   4. 在配置文件中配置Bean
   5. 创建applicationContext对象getBean
2. Spring配置文件的配置：
   1. Bean标签的基本配置; 用于配置对象交由**Spring**容器来创建，默认情况下，它调用的是类中的**无参构造**，如果没有无参构造函数，则不能成功。

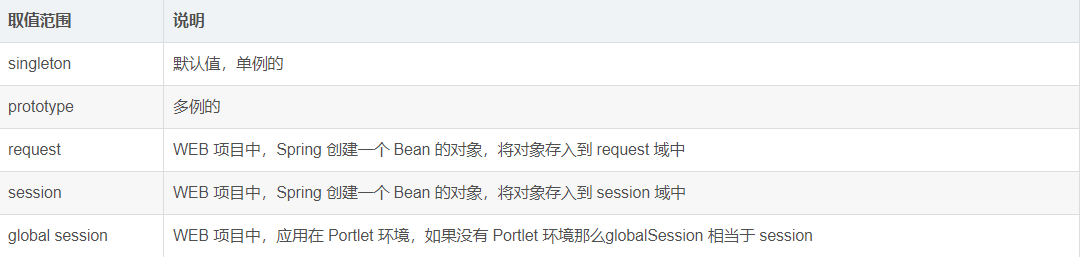


基本属性：

Id: Bean实例在Spring容器中的唯一标识

Class:Bean的全限定名。

* 1. Bean标签的范围配置:



* + 1. 当scope=’singleton’ 时

Bean 的实例化个数：1个

Bean 的实例化时机：当Spring的核心配置文件被加载时，实例化配置的Bean实例

Bean 的生命周期：

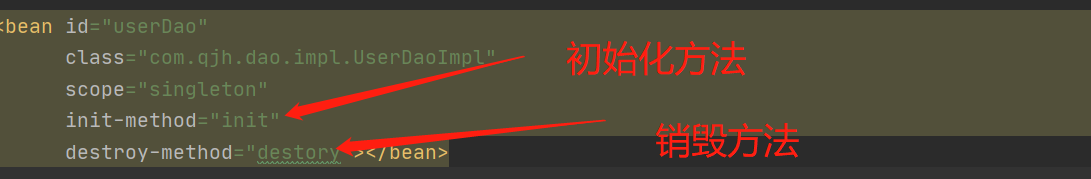
* 对象的创建：当应用加载，创建容器时，对象就被创建了
* 对象的运行：只要容器在，对象就一直活着
* 对象的销毁：当应用卸载，销毁容器时，对象就会被销毁
  + 1. 当scope=’prototype’ 时

Bean 的实例化个数：多个

Bean 的实例化时机：当调用getBean()方法时实例话Bean

Bean 的生命周期

* 对象的创建：当使用对象时，创建新的对象的实例
* 对象的运行：只要对象在使用，对象就一直活着
* 对象的销毁：当对象长时间不用时，被Java的垃圾回收器回收
  1. Bean的生命周期配置



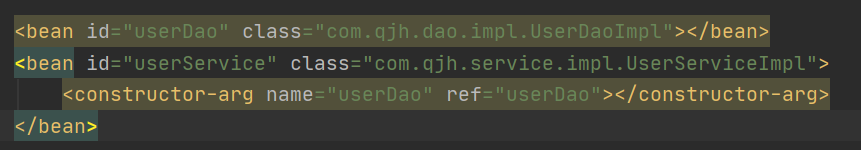
* 1. Bean实例化的三种方式
     1. **无参构造方法实例化**：最常用
     2. 工厂静态方法实例化：
     3. 工厂实例方法实例化：必须要先用工厂对象再去调用方法

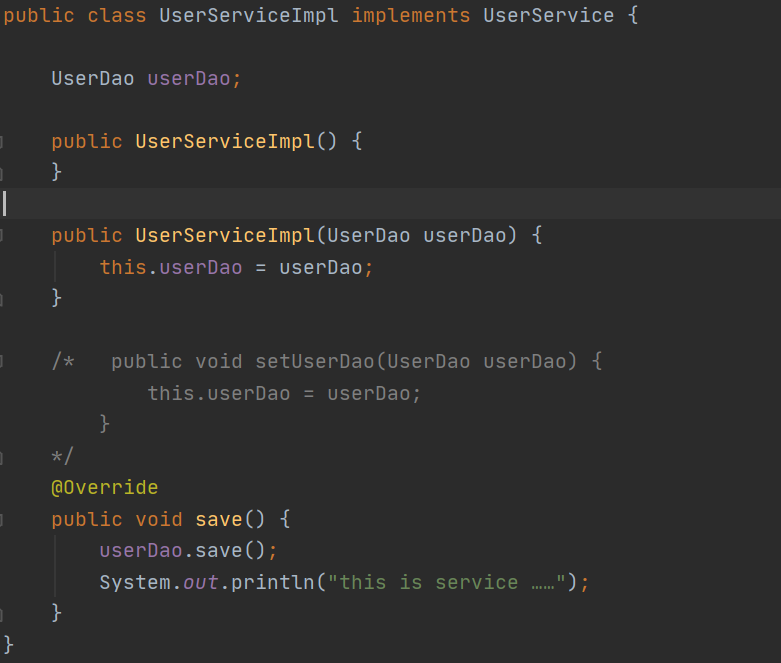


* 1. Bean 的依赖注入

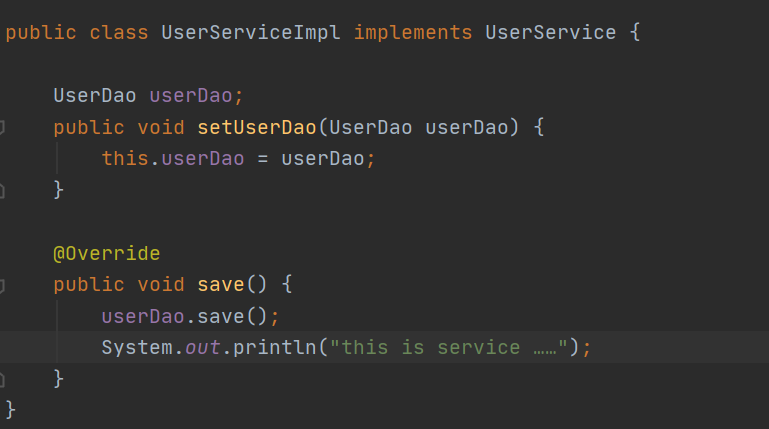
依赖注入（dependency injection）：是spring框架核心IOC的具体体现。在编写程序时，通过控制反转，把对象的创建交给Spring，但是代码中不能出现没有依赖的情况。IOC解耦只是降低了他们的依赖关系，但不会消除

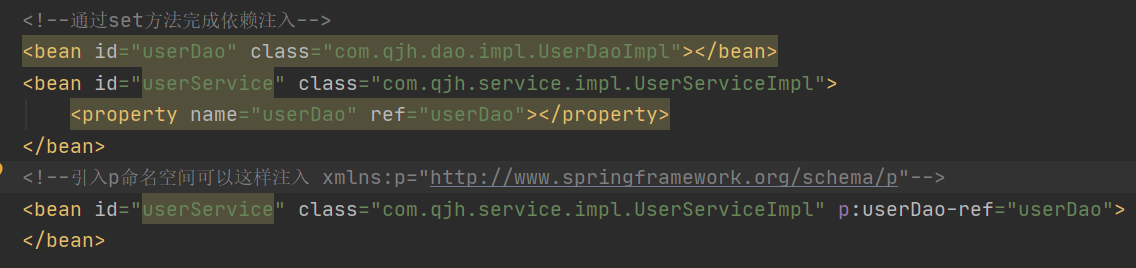
* + 1. 通过构造方法（有参构造）注入



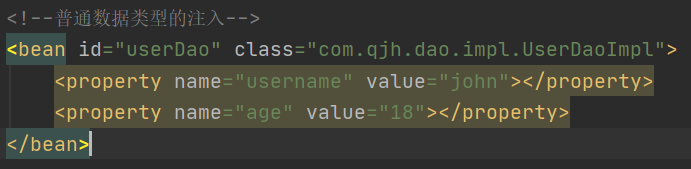


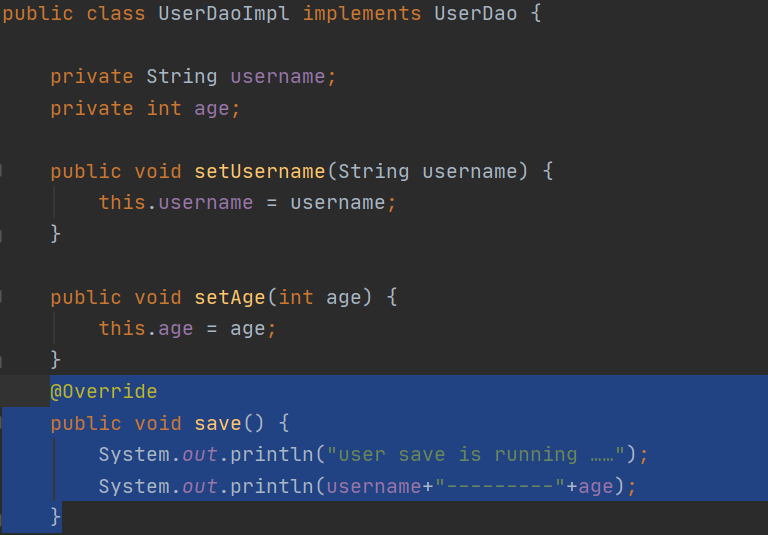
* + 1. 通过set方法注入



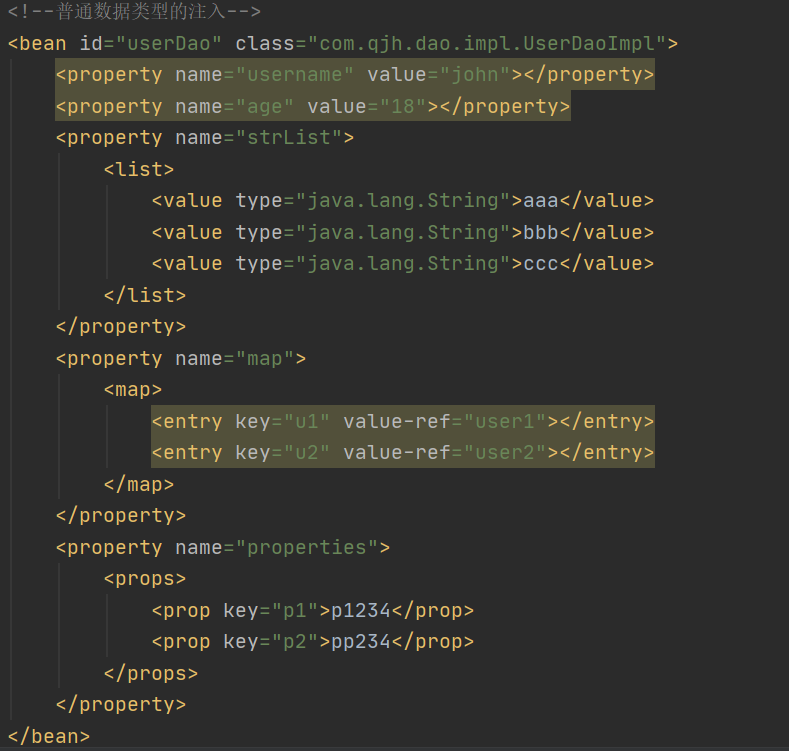


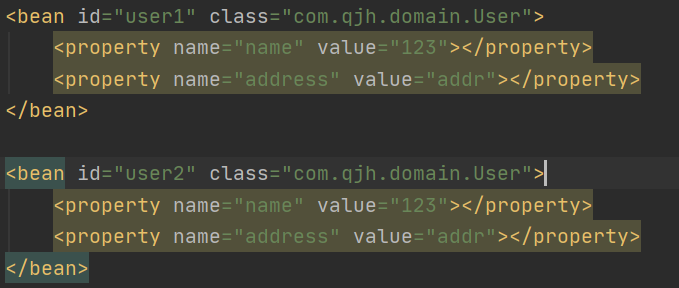
* 1. Bean的依赖注入的数据类型
     1. 普通数据类型

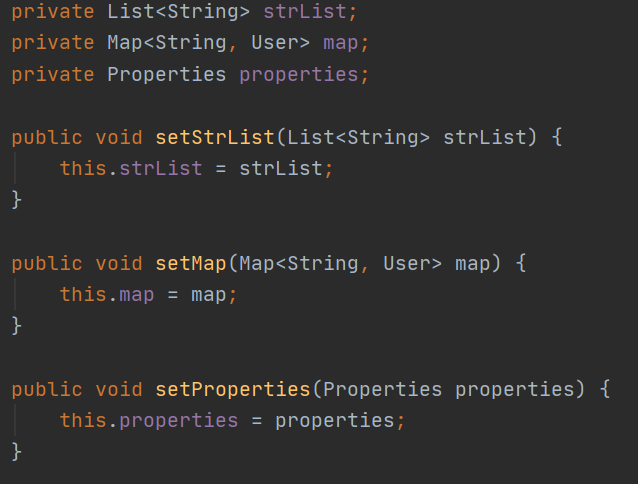




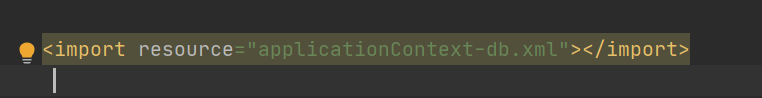
* + 1. 引用数据类型
    2. 集合数据类型



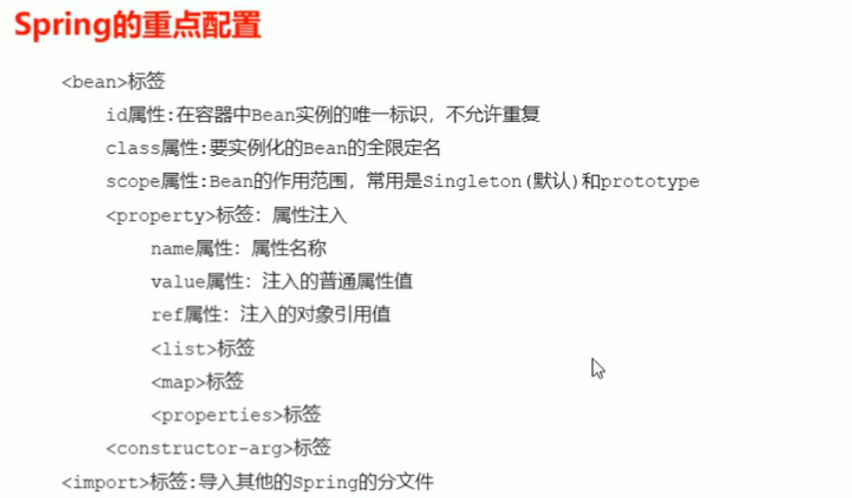




* 1. 引入其他配置文件

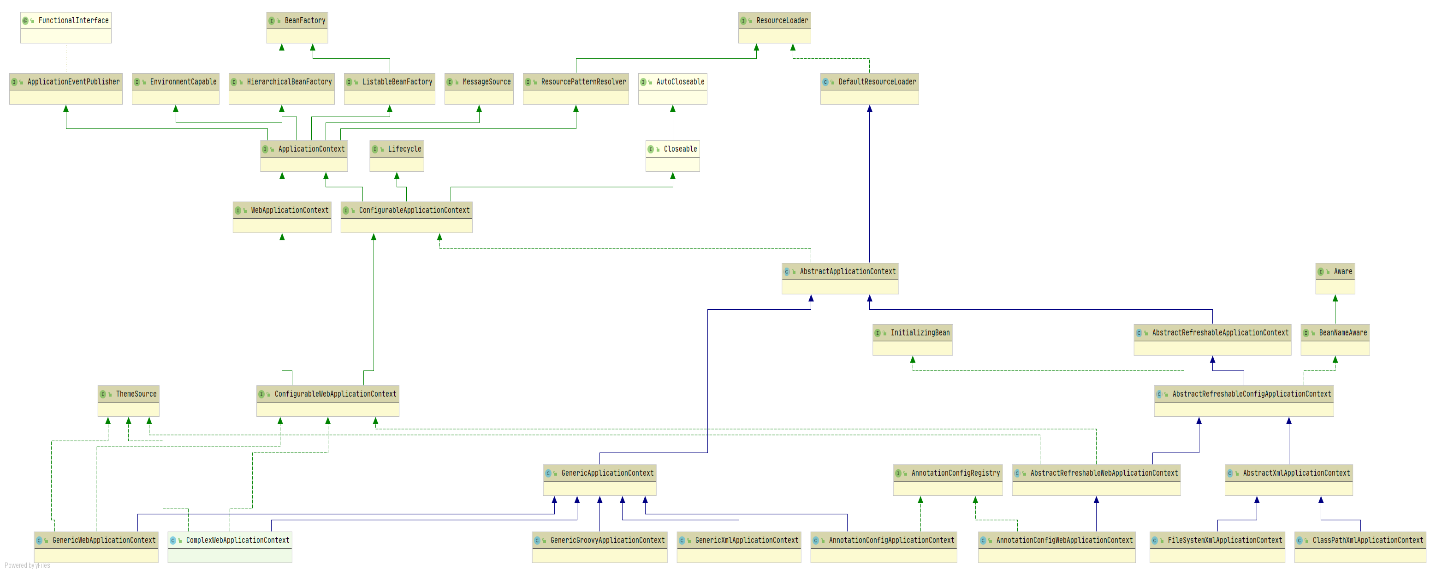


* 1. Spring的重点配置



1. Spring相应的API
   1. ApplicationContext的继承体系

ApplicationContext：接口类型。代表应用上下文可以通过其实例活得Spring容器中的Bean对象

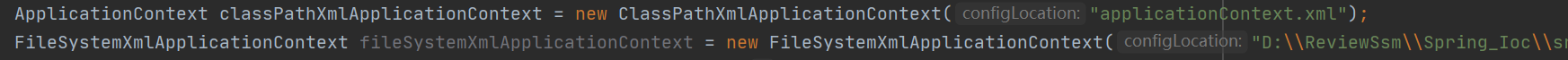


* 1. ApplicationContext的实现类
     1. ClassPathXmlApplicationContext

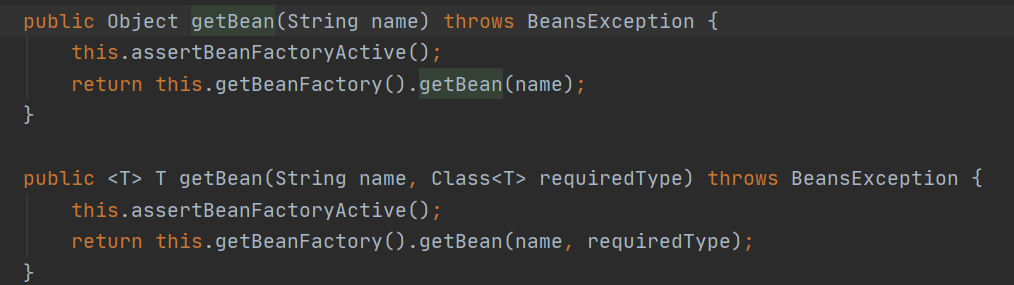
它是从类的根路径下加载配置文件

* + 1. FileSystemXmlApplicationContext

从磁盘路径上加载配置文件



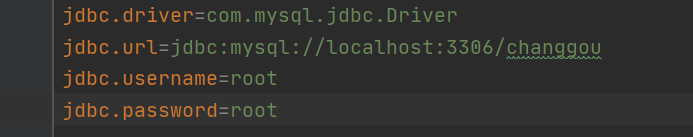
* + 1. AnnotationConfigApplicationContext
  1. getBean()方法的使用（可以看一下源码）



1. Spring配置数据源
   1. 数据源（连接池）的作用
      1. 数据源是为了提高程序性能而出现的
      2. 事先实例化数据源，初始化部分连接资源
      3. 使用连接资源从数据源中获取
      4. 使用完毕后将资源归还给数据源
      5. 常见的数据源（连接池）: c3p0 druid



* + 1. Jdbc.properties





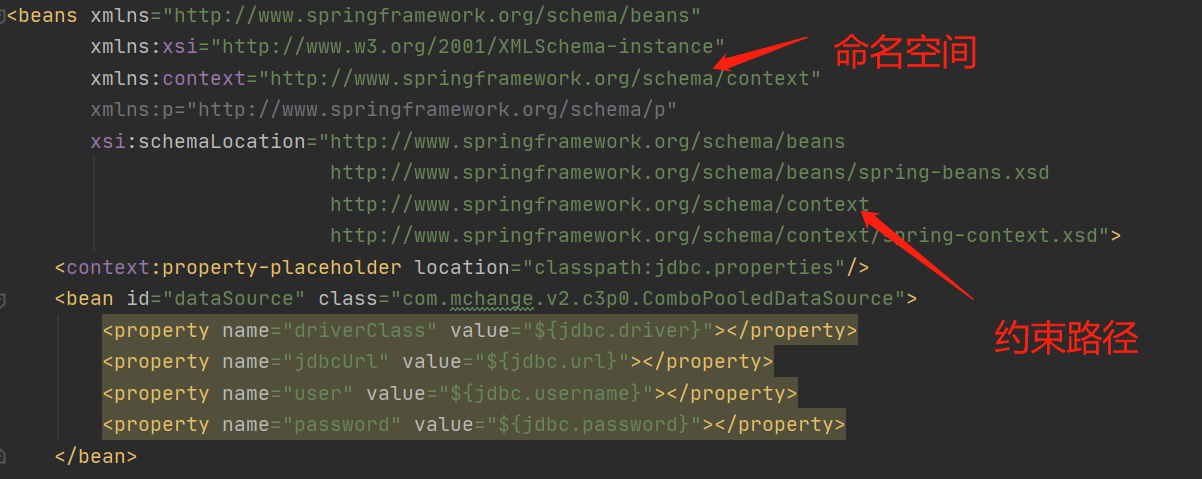
* 1. Spring配置数据源



* 1. 抽取jdbc配置文件

applicationContext.xml 加载jdbc.properties配置文件获取连接信息

首先，需要引入context命名空间和约束路径



1. Spring注解开发
   1. Spring原始注解
      1. Spring的原始注解主要是替代<Bean>的配置



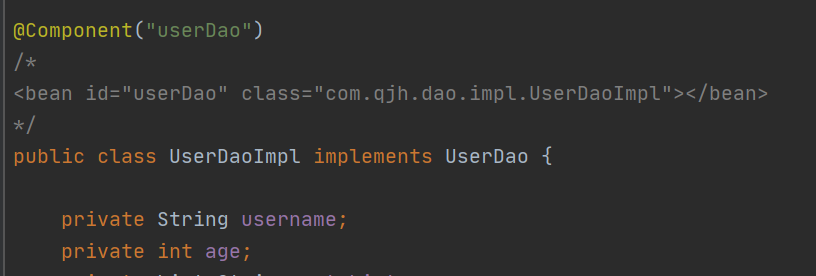
**注：@Autowired 和 @Resource的区别：**

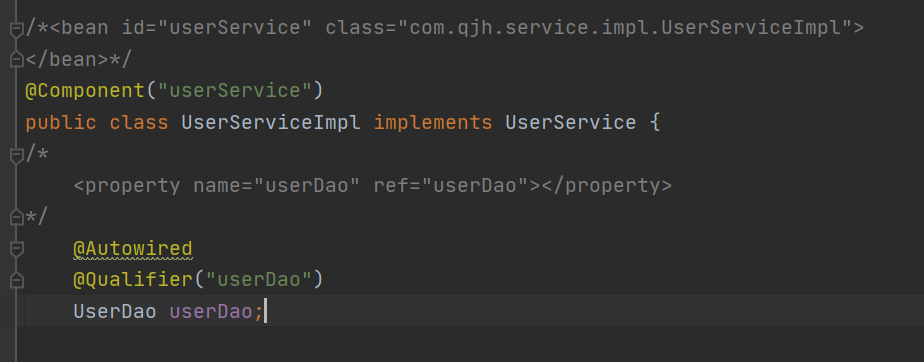
1、@Autowired与@Resource都可以用来装配Bean,都可以写在字段上，或写在setter方法上

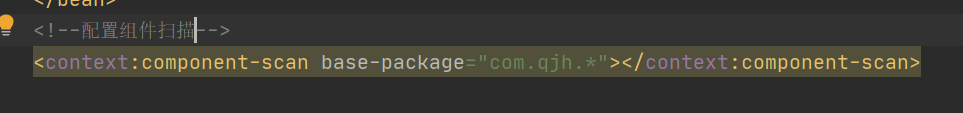
2、@Autowired(属于Spring)默认按类型装配，默认情况下必须要求依赖对象必须存在，如果要允许null值，可以设置它的required属性为false

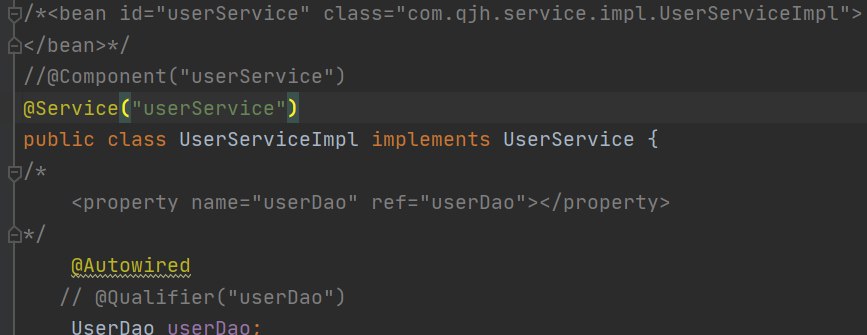
3、@Resource（属于JavaEE），默认按照名称进行装配，名称可以通过name属性来指定，如果没有指定name属性，当注解写在字段上时，默认取字段名进行安装名称查找，如果注解写在setter 方法上默认取属性名进行装配，当找不到名称匹配的bean时才按照类型进行装配，需注意，name属性一旦指定，就只会按照名称进行装配。

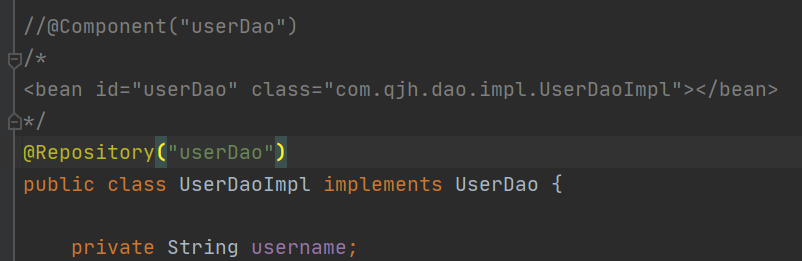
* + 1. 注解的使用



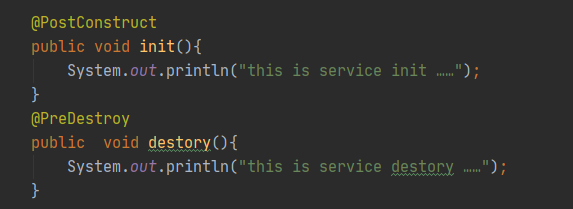












* 1. Spring 的新注解
     1. 使用上面的注解，还不能全部替代xml配置文件，还需要使用注解替代的配置如下：
        1. 非定义的Bean 的配置<bean>
        2. 加载properties文件的配置<context:property-placeholder/>
        3. 组件扫描的配置：<context:component-scan>
        4. 引入其他文件：<import>
     2. Spring的新注解

|  |  |
| --- | --- |
| 注解 | 说明 |
| @Configuration | 用于指定当前类是一个Spring配置类，当创建容器时会从该类上加载注解 |
| @ComponentScan | 用于指定Spring的初始化容器时要扫描的包‘  作用和spring的xml配置文件中的  <context:component-scan>的作用一样 |
| @Bean | 使用在方法上，标注该方法的返回值存储到Spring容器中 |
| @PropertySource | 用于加载Propertis文件中的配置 |
| @Import | 用于导入配置类 |

1. Spring整合Junit