Spring常用注解总结

Spring的一个核心功能是IOC(具体指什么，现在还没弄得特别清楚)，就是将Bean初始化加载到容器中，Bean是如何加载到容器的，可以使用Spring注解方式或者Spring XML配置方式。

Spring注解方式减少了配置文件内容，更加便于管理，并且使用注解可以大大提高开发效率。下面按照分类讲解Spring中常用的一些注解：

1. 组件类注解

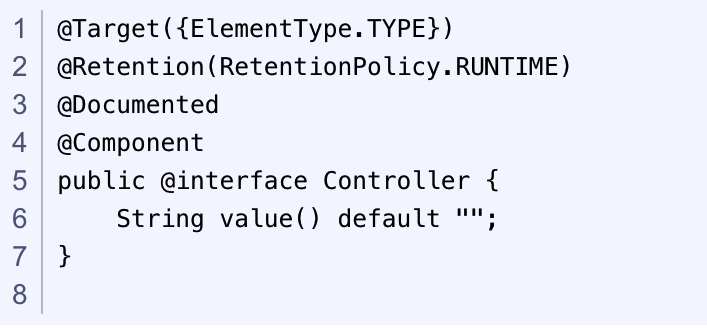
思考：Spring是怎么知道应该把哪些java类当成bean注册到容器中呢？

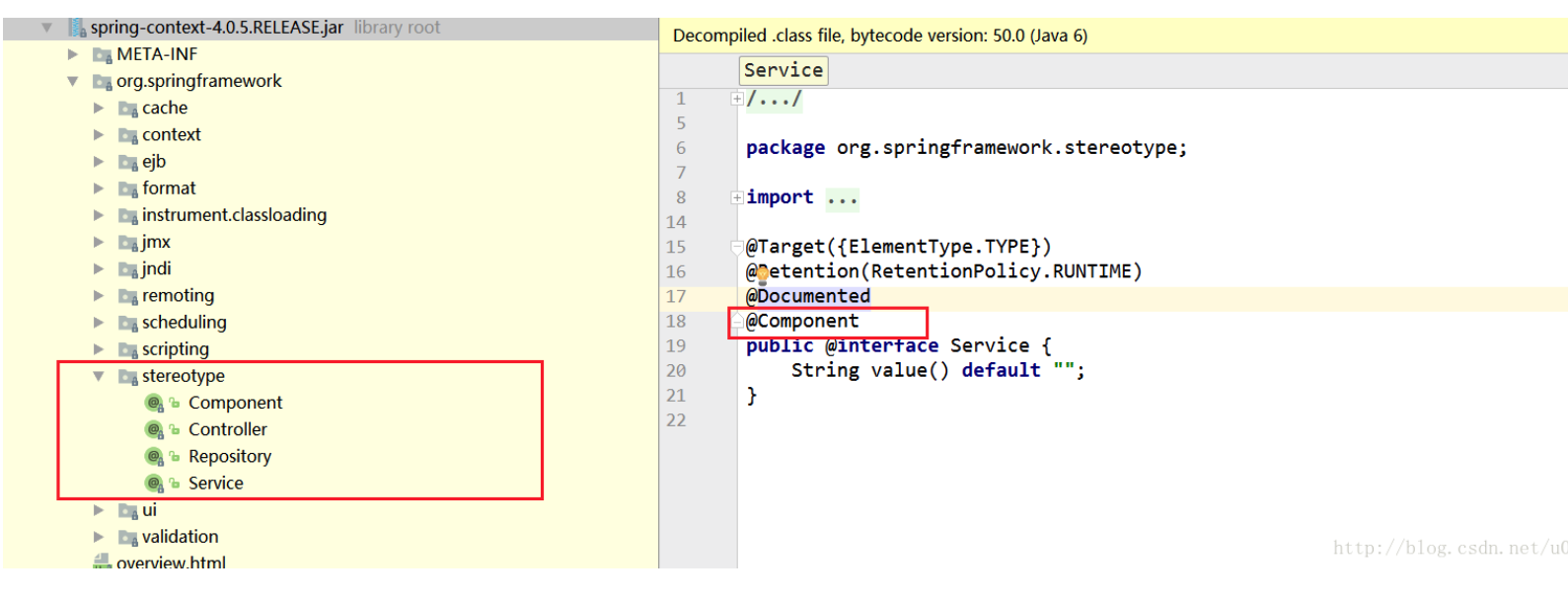
答案：使用配置文件或者注解的方式进行标识需要处理的java类。

1. 注解类介绍

* @Component：标注一个普通的spring Bean类。
* @Repository：标注一个DAO组件类。
* @Service：标注一个业务逻辑组件类。
* @Controller：标注一个控制器组件类。

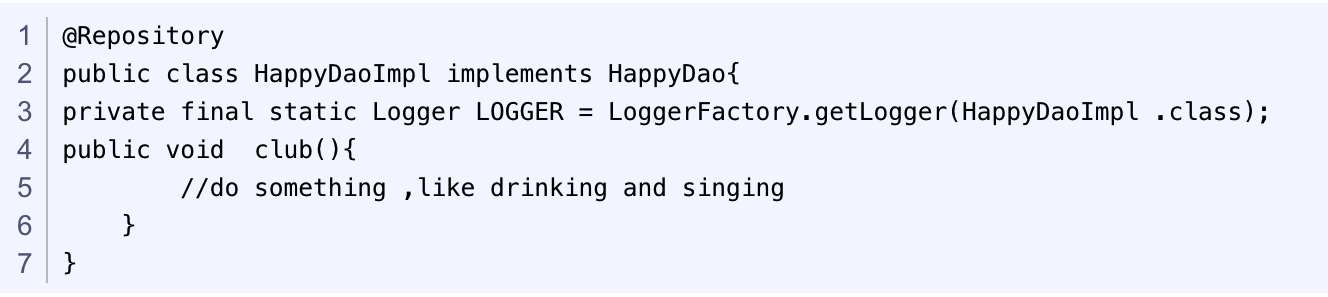
这些都是注解在平时的开发过程中出镜率极高，@Component、@Repository、@Service、@Controller实质上是属于同一类注解，用法相同，功能相同，区别在于标识组件的类型。@Component可以替代@Repository、@Service、@Controller，因为这三个注解是被@Component标注的。如下代码：



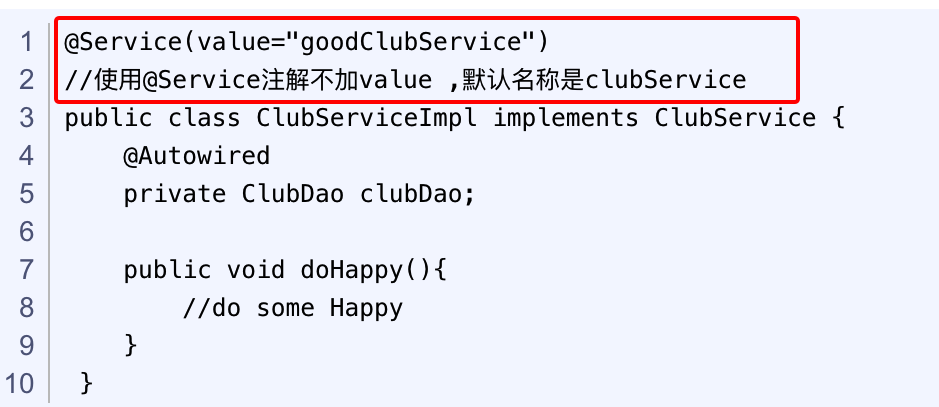


1. 举例详解

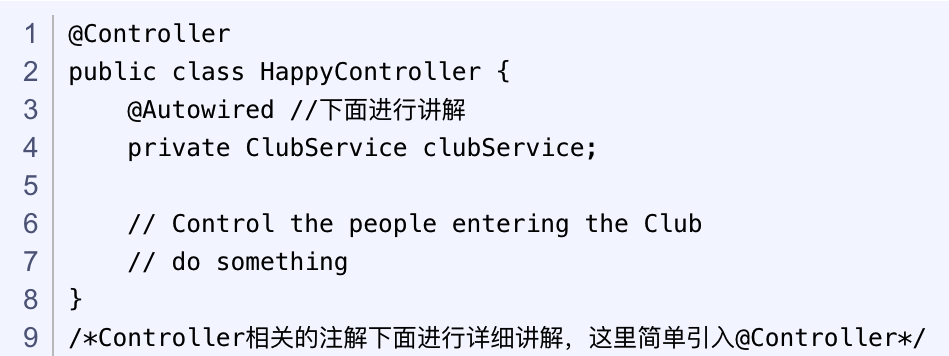
当一个组件代表数据访问层(DAO)的时候，我们使用@Repository进行注解，如下：



当一个组件代表业务层时，我们使用@Service进行注解，如下：

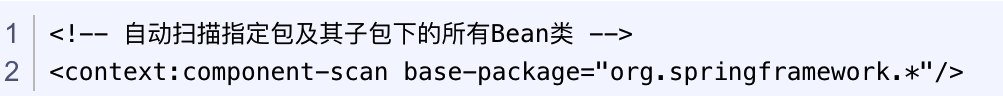


当一个组件作为前端交互的控制层，使用@Controller进行注解，如下：



1. 总结注意点

* 被注解的java类当做Bean实例，Bean实例的名称默认是Bean类的首字母小写，其他部分不变。@Service也可以自定义Bean名称，但是必须是唯一的。
* 尽量使用对应组件注解的类(@Service、@Repository等)来替换@Component注解，在spring未来的版本中，@Controller、@Service、@Repository会携带更多语义。并且便于开发和维护。
* 指定了某些类可作为Spring Bean类使用后，最好还需要让spring搜索指定路径，在Spring配置文件加入如下配置。



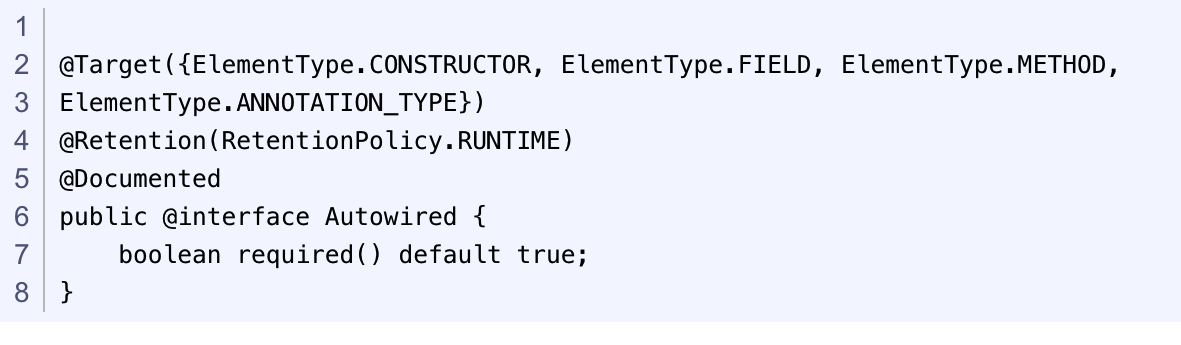
1. 装配bean时常用的注解
2. 注解介绍

* @Autowired：属于Spring的org.springframework.beans.factory.annotation包下，可为类的属性、构造器、方法进行注值。
* @Resource：不属于spring的注解，而是来自于JSR-250位于java.annotation包下，使用该annotation为目标bean指定协作者Bean。
* @PostConstruct和@PreDestroy方法，实现初始化和销毁之前进行的操作。

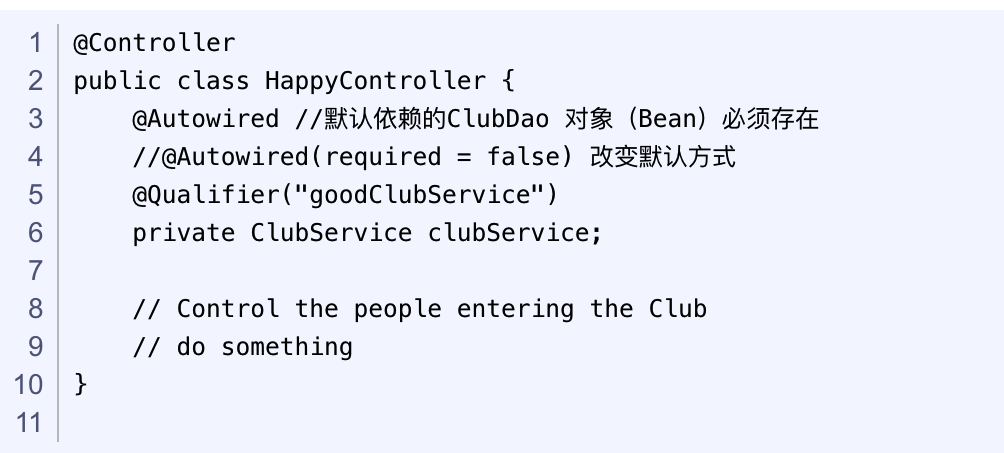
1. 举例说明

1）、@Autowired

@Autowired的定义：

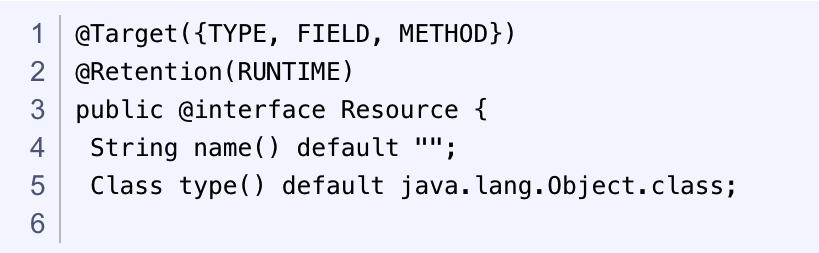


使用@Autowired的实例：

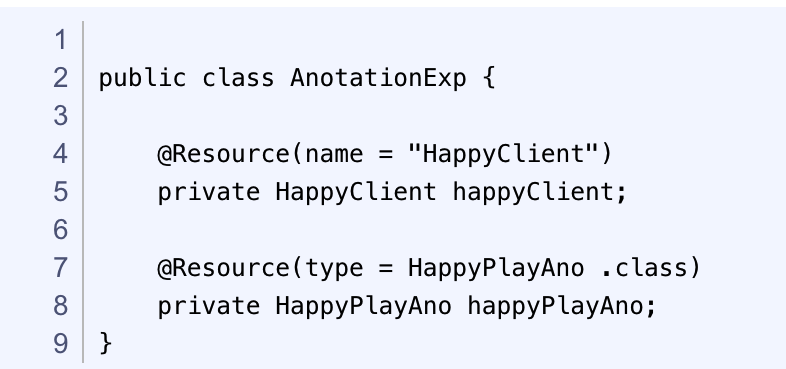


2）@Resource

@Resource的定义：



@Resource的使用实例：



1. 总结

相同点：

@Resource的作用相当于@Autowired，均可标注在字段或者属性的setter方法上。

不同点：

* 提供方：@Autowired是Spring的注解，@Resource是java.annotation的注解
* 注入方式：@Autowired是按照Type注入的; @Resource是默认按照Name自动注入的，也提供按照Type注入。
* 属性：
  + @Autowired
    - @Autowired注解可为类的属性、构造器、方法进行注值。默认情况下，其依赖的对象必须存在（bean可用），如果需要改变这种默认方式，可以设置其required属性为false。
    - @Autowired注解默认按照类型进行装配，如果容器中包含多个同一类型的Bean，那么启动容器时会报找不到指定类型的bean异常，解决方法就是结合@Qualifier注解进行限定，指定注入的bean名称。
  + @Resource中有两个重要的属性：name和type
    - Name属性指定byName，如果没有指定Name属性，当注解标注在字段上，即默认取字段的名称作为bean名称寻找依赖对象，当注解标注在属性的setter方法上，即默认取属性名作为bean名称寻找依赖对象。
* 对比：@Resource注解的使用性更为灵活，可指定名称，也可以指定类型；@Autowired注解进行装配时容易抛出异常，特别是装配的bean类型有多个的时候，而解决的方法是需要在增加@Qualifier进行限定。

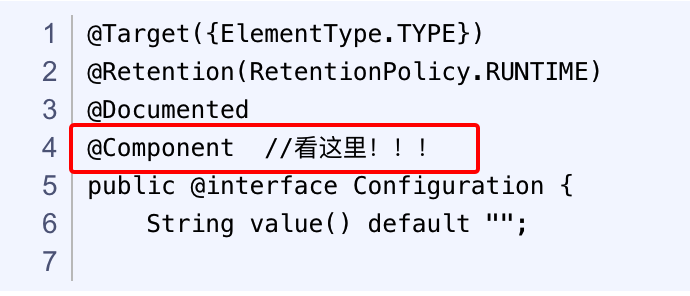
@Resource的装配顺序：

* + - 1. 如果同时指定了name和type，则从spring上下文中寻找唯一匹配的bean进行匹配，找不到则跑出异常
      2. 如果指定了name，则从上下文中查找名称(id)匹配的bean进行装配，找不到则抛出异常。
      3. 如果指定了type，则从上下文中找到类型匹配的唯一bean进行装配，找不到或者找到多个，都会抛出异常。
      4. 如果既没有指定name，也没有指定type，则自动按照byName的方式进行装配；如果没有匹配，则回退为一个原始类型进行匹配，如果没有匹配则自动装配。

三、@Component vs @Configuration and @Bean

1、简单介绍

Spring的官方团队说@Component可以替代@Configuration注解，事实上我们看源码也可以发现看到，如下：

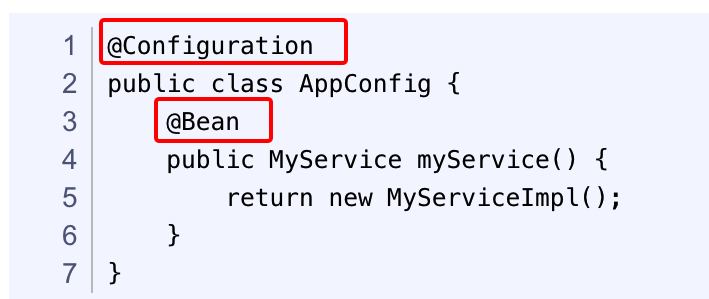


虽然说可以替代，但是两个注解之间还是有区别的！

Bean注解主要用于方法上，有点类似于工厂方法，当然使用了@Bean注解，我们可以连续使用多种定义bean时用到的注解，譬如用@Qualifier注解定义工厂方法的名称，用@Scope注解定义该bean的作用域范围，譬如singleton还是prototype。

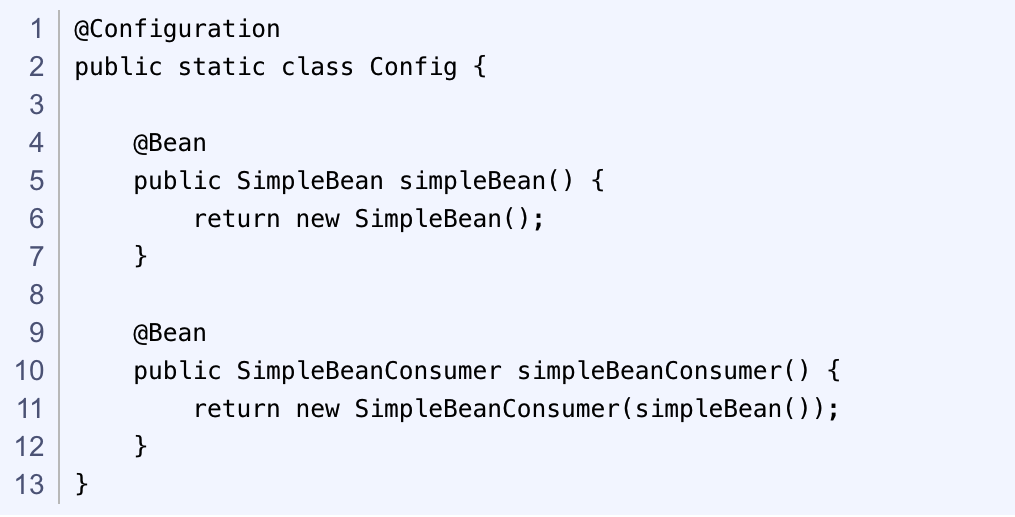
Spring中新的java配置支持的核心就是@Configuration注解的类。这些类主要包括@Bean注解的方法来为Spring的ioc容器管理的对象定义实例，配置和初始化逻辑。

使用@Configuration来注解类表示类可以被Spring的ioc容器所使用，作为bean定义的资源。

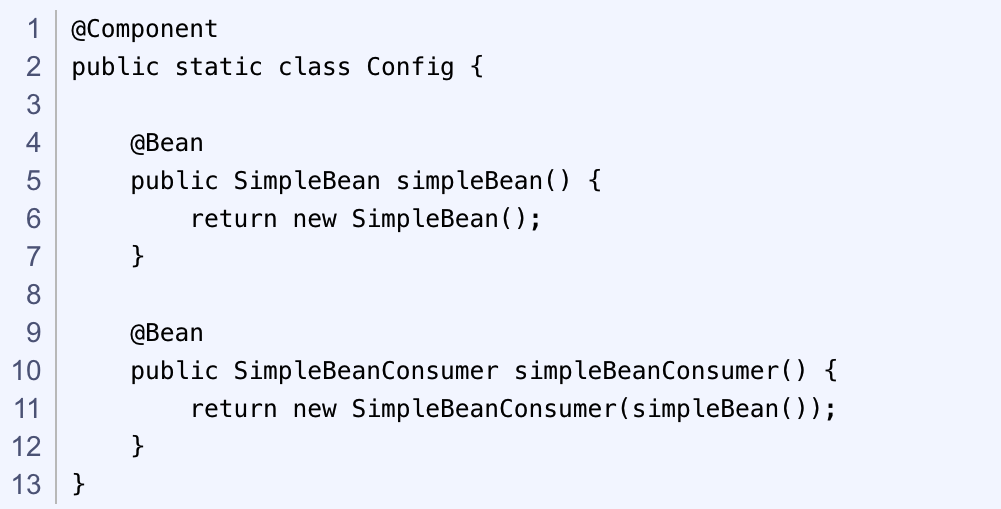


2、举例说明@Component和@Configuration

@Configuration的例子：



@Component的例子：

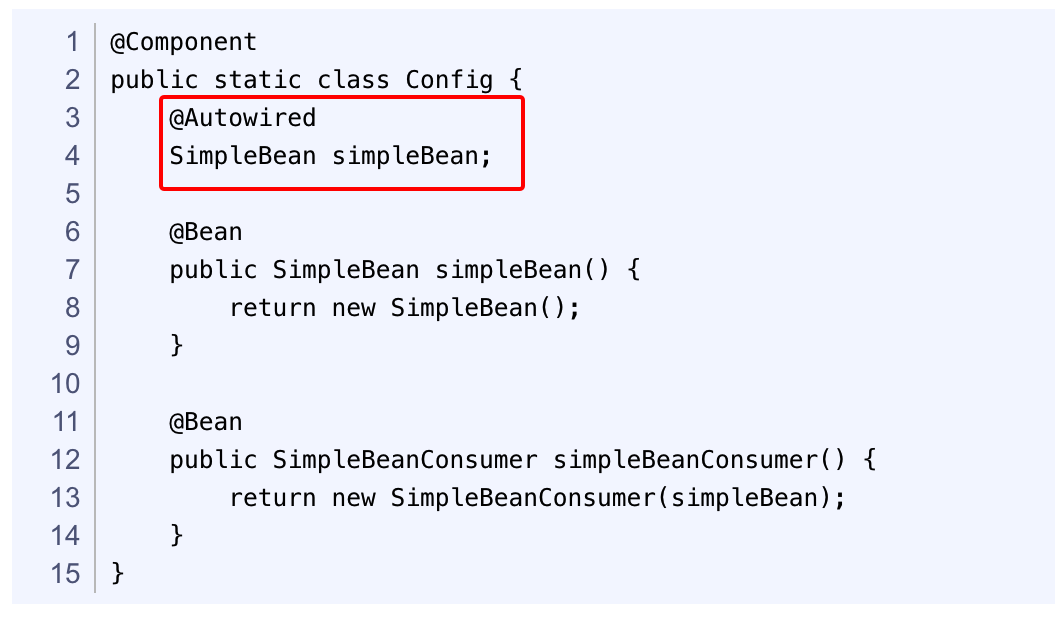


第一个代码@Configuration的例子正常工作，正如预期的那样，SimpleBeanConsumer将会得到一个单例SimpleBean的链接。第二个的配置是完全错误的，因为Spring会创建一个SimpleBean的单例bean，但是SimpleBeanConsumer将获得另一个SimpleBean实例（也就是相当于直接调用new SimpleBean(), 这个bean不归Spring管理的），即new SimpleBean()实例是Spring上下文控件之外的。

1. 原因总结

使用@Configuration，所有标记为@bean的方法将被包装成一个CGLIB包装器，它的工作方式就是这个方法的第一次调用时，那么原始的主体将被执行，生成的对象将在spring的上下文中注册，所有下一次的调用都只返回从上下文检索得到的bean。

在上面的@Component的例子的代码中，为了纠正第二个代码块，使前后得两个simpleBean相同，我们可以这样做：



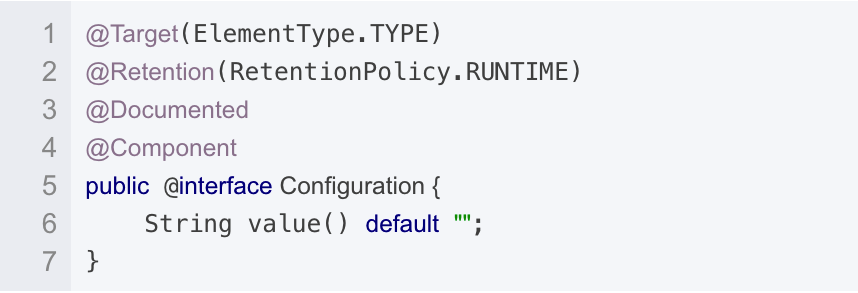
@Autowired可以为类的属性进行注值。

附几个知识点：

**3.1 Spring中的@Configuration和@Component的区别**

一句话概括就是@Configuration中所有带@Bean注解的方法都会被动态处理，因此调用该方法返回的都是同一个实例。

下面看看实现的细节，@Configuration注解：



从定义上看，@Configuration注解本质上还是@Component。

@Configuration标记的类必须符合下面的要求：

* 配置类必须以类的形式提供（不能是工厂方法返回的实例），允许通过生成子类在运行时增强（cglib动态代理）。
* 配置类不能是final类（没法动态代理）
* 配置注解通常为了通过@Bean注解生成Spring容器管理的类。
* 配置类必须是非本地的（即不能再方法中声明，不能是private）。
* 任何嵌套配置类都必须声明为static。
* @Bean方法可能不会反过来创建进一步的配置类（也就是返回的bean如果带有@Configuration，也不会被特殊处理，只会作为普通的bean）。

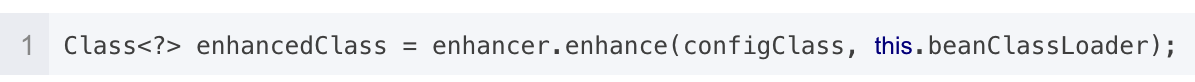
**加载过程**

Spring容器在启动时，会加载默认的一些PostRocessor，其中就有ConfigurationClassPostProcessor，这个后置处理程序专门处理带有@Configuration注解的类，这个程序会在bean定义加载完成后，在bean初始化前进行处理。主要处理的过程就是使用cglib动态代理增强类，而且是对其中带有@Bean注解的方法进行处理。

在ConfigurationClassPostProcessor中的postProcessBeanFactory方法中调用了下面的方法：



在上图中的第一个for循环，查找到所有带有@Configuration注解的定义，然后再第二个for循环中，通过下面的方法对类进行增强：



然后再使用增强后的类替换掉原有的beanClass：



所以到此时，所有带有@Configuration注解的bean都已经变成了增强的类。

对增强的类进行一个概述：

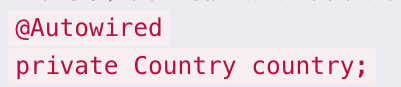
* Spring中存在这样的一个工具类ConfigurationClassEnhancer，它会对应用中每个配置类，也就是一般通过@Configuration注解定义的类进行一个增强。通过增强以后，配置类中使用@Bean注解的bean定义方法就不再是普通的方法了，他们具有了如下跟bean作用域有关的能力，以单例bean为例：
  + 1、它们首次被调用时，相应方法逻辑会被执行用于创建bean实例。
  + 2、再次被调用时，不会在执行创建bean实例，而是根据bean名称返回首次该方法被执行时创建的bean实例。
  + 该增强是通过为相应配置类创建一个CGLIB子类来完成的，该工具类会被ConfigurationClassPostProcessor在应用启动阶段发现和注册配置类中的所有bean定义之后，应用于所有的配置类对他们进行增强。

**@Bean注解方法执行策略**

先给一个简单的示例代码：



相信大多数人第一次看到上面userInfo()中调用country()时，会认为这里的Country()和上面的@Bean方法返回的Country可能不是同一个对象，因此可能会通过下面的方式来代替这种方式：



实际上不需要这么做（后面会给出需要这么做的场景），直接调用country()方法返回的是同一个实例。

**@Component注解**

@Component注解并没有通过cglib来代理@Bean方法的调用，因此像下面这样配置时，就是两个不同的conutry。



有些特殊情况下，我们不希望MyBeanConfig被代理，就得用@Component，这种情况下，上面的写法就需要改写成下面这样：

 这种方式可以保证使用的是同一个Country实例。