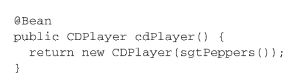
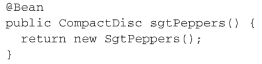
Spring 学习笔记

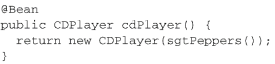
## 1.@Bean注解：

使用了@Bean注解， 这表明这个方法会创建一个bean实例并将其注册到Spring应用上下文中。与方法的名字相同。

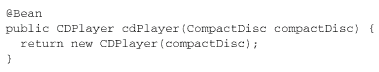


添加了@Bean注解的方法， Spring将会拦截所有对它的调用， 并确保直接返回该方法所创建的bean， 而不是每次都对其进行实际的调用。





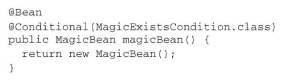
通过调用方法来引用bean的方式有点令人困惑。 其实还有一种理解起来更为简单的方式：



在这里， cdPlayer()方法请求一个CompactDisc作为参数。 当Spring调用cdPlayer()创建CDPlayerbean的时候， 它会自动装配一个CompactDisc到配置方法之中。 然后， 方法体就可以按照合适的方式来使用它。 借助这种技术， cdPlayer()方法也能够将CompactDisc注入到CDPlayer的构造器中， 而且不用明确引用CompactDisc的@Bean方法。通过这种方式引用其他的bean通常是最佳的选择， 因为它不会要求将CompactDisc声明到同一个配置类之中。 在这里甚至没有要求CompactDisc必须要在JavaConfig中声明， 实际上它可以通过组件扫描功能自动发现或者通过XML来进行配置。 你可以将配置分散到多个配置类、 XML文件以及自动扫描和装配bean之中， 只要功能完整健全即可。 不管CompactDisc是采用什么方式创建出来的， Spring都会将其传入到配置方法中， 并用来创建CDPlayer bean。

## 2. @Conditional注解

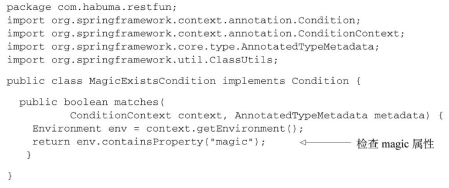
假设你希望一个或多个bean只有在应用的类路径下包含特定的库时才创建。 或者我们希望某个bean只有当另外某个特定的bean也声明了之后才会创建。 我们还可能要求只有某个特定的环境变量设置之后， 才会创建某个bean。使用@Conditional注解条件化地配置了MagicBean。



可以看到， @Conditional中给定了一个Class， 它指明了条件——在本例中， 也就是MagicExistsCondition。 @Conditional将会通过Condition接口进行条件对比：



设置给@Conditional的类可以是任意实现了Condition接口的类型。 可以看出来， 这个接口实现起来很简单直接， 只需提供matches()方法的实现即可。 如果matches()方法返回true， 那么就会创建带有@Conditional注解的bean。 如果matches()方法返回false， 将不会创建这些bean。在本例中， 我们需要创建Condition的实现并根据环境中是否存在magic属性来做出决策。 程序清单展现了MagicExistsCondition， 这是完成该功能的Condition实现类：



在上面的程序清单中， matches()方法很简单但功能强大。 它通过给定的ConditionContext对象进而得到Environment对象， 并使用这个对象检查环境中是否存在名为magic的环境属性。 在本例中， 属性的值是什么无所谓， 只要属性存在即可满足要求。 如果满足这个条件的话， matches()方法就会返回true。 所带来的结果就是条件能够得到满足， 所有@Conditional注解上引用MagicExistsCondition的bean都会被创建。

话说回来， 如果这个属性不存在的话， 就无法满足条件， matches()方法会返回false， 这些bean都不会被创建。MagicExistsCondition中只是使用了ConditionContext得到的Environment， 但Condition实现的考量因素可能会比这更多。 matches()方法会得到ConditionContext和AnnotatedTypeMetadata对象用来做出决策。

通过ConditionContext， 我们可以做到如下几点：

借助getRegistry()返回的BeanDefinitionRegistry检查bean定义；

借助getBeanFactory()返回的ConfigurableListableBeanFactory检查bean是否存在， 甚至探查bean的属性；

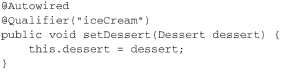
借助getEnvironment()返回的Environment检查环境变量是否存在以及它的值是什么；

读取并探查getResourceLoader()返回的ResourceLoader所加载的资源；

借助getClassLoader()返回的ClassLoader加载并检查类是否存在。

## 3. @Qualifier注解

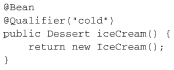
@Qualifier注解是使用限定符的主要方式。 它可以与@Autowired和@Inject协同使用， 在注入的时候指定想要注入进去的是哪个bean。



我们可以为bean设置自己的限定符， 而不是依赖于将bean ID作为限定符。 在这里所需要做的就是在bean声明上添加@Qualifier注解。 例如， 它可以与@Component组合使用， 如下所示：



值得一提的是， 当通过Java配置显式定义bean的时候， @Qualifier也可以与@Bean注解一起使用

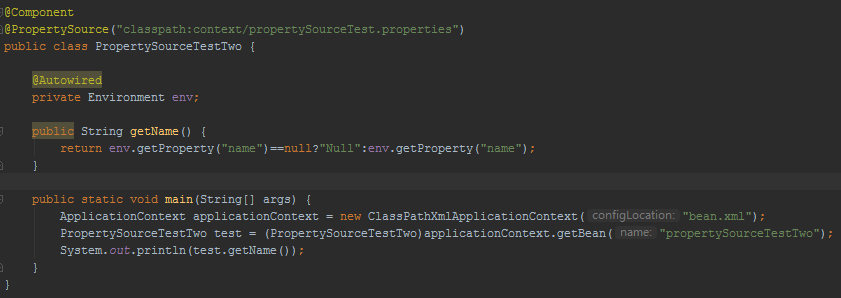


## 4. @PropertySource注解

@PropertySource注解用于将项目中的Property文件引入到Spring上下文当中



上面的方式要想使用@PropertySource读取Property文件，用@Value注解 （@Value既可以用在类属性上，也可以用在方法参数上）用${}占位符注入属性，PropertySourcesPlaceholderConfigurer 这个bean是必须的（可以通过XML配置替换此处的注册bean的方法，如果你想使用XML配置的话，Spring context命名空间中的<context:propertyplaceholder>元素将会为你生成PropertySourcesPlaceholderConfigurer bean），这个就是占位bean,另一种方式是不用value直接用Envirment变量直接getProperty('key')



Envirment中有很多方法帮助处理property文件属性，具体不在这里记录了

## 5. @DeclareParents注解

spring-AOP通过注解@DeclareParents为指定类引入其他类的方法

https://blog.csdn.net/u010502101/article/details/76944753