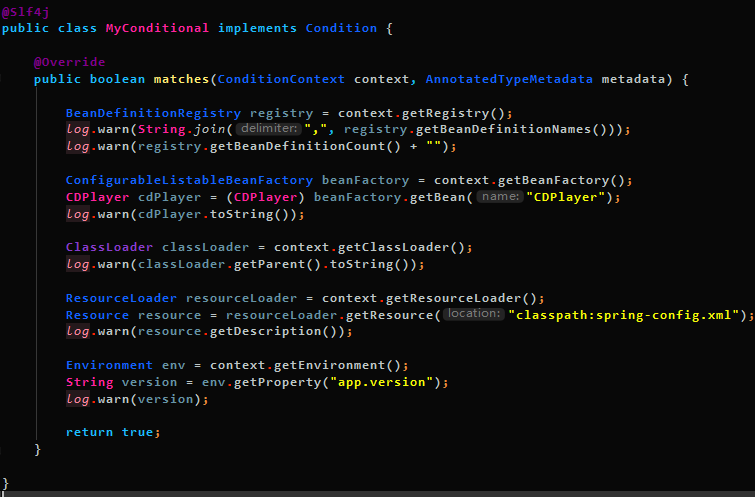
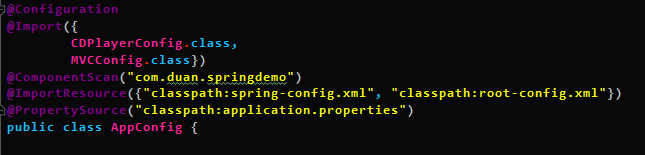
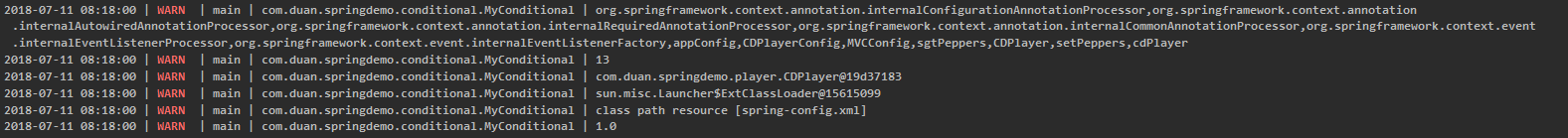
## **条件化 Bean**

@Conditional 注解，可以应用到带有 @Bean 注解的方法上，如果给定的条件为 true，Bean 才会被创建，否则不创建。









## **自动装配的歧义性**

自动装配仅在只有一个 bean 匹配时有效，如果不只有一个 bean 的话，就会阻碍 spring 自 动装配属性、构造器参数或方法参数。

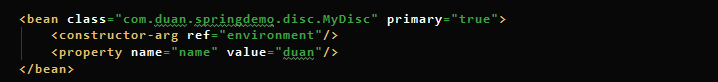
解决：

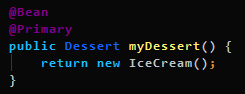
1. 将可选 bean 中的每一个设为首选（ primary ）

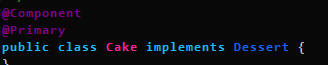
2. 使用限定符来帮助 spring 缩小范围（ qualifier ）

**标识首选的 bean**

@Primary 可与 @Component 和 @Bean 注解配合使用。xml 中对于 bean 标签有对应的 boolean 属性 primary。



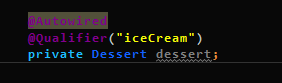




当有两个可选的 Dessert 时，spring 无法从多个首选 bean 中做出选择，因此 @Primary 只在唯一时有效。

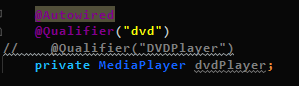
**明确限定 bean**

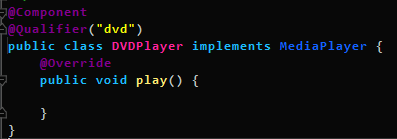
@Qualifier 注解是使用限定符的主要方式，可以与 @Autoware 和 @Inject 注解配合使用。



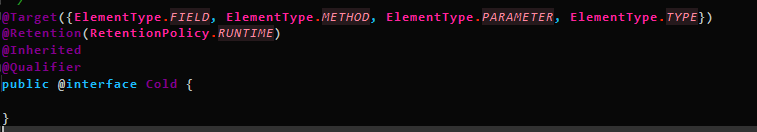
缺点：@Qualifier 指定的 bean id 与类名（方法名）紧耦合，类名修改时 @Qualifer 注解将失效。

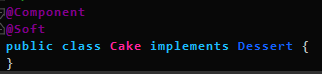
通过在 Bean 定义处和使用处同时使用 @Qualifier 注解，可以实现基于 String 描述的对多个限定特性进行限定（java 8 才支持重复注解，但 @Qualifier 不支持重复注解），因此只能明确限定一个特性。

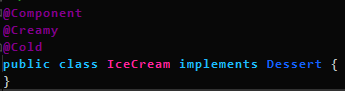
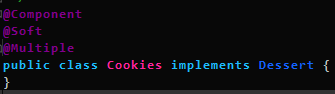




可以通过创建自定义的注解来实现限定符的功能，这些注解需要被 @Qualifier 注解注解，那么这个注解本身可以作为一个限定特性。









原始的 @Qualifier 注解使用 String 进行限定，而基于 @Qualifier 注解的注解使用注解进行限定，类型更为安全，同时能够解耦。

## **bean 的作用域**

在默认的情况下，Spring 上下文的所有 bean 都是作为单例存在的。

单例会保持一些状态，对象会被污染，有时重用时不安全的。

作用域：

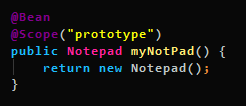
1. 单例（Singleton）：整个应用中，只创建 bean 的一个实例

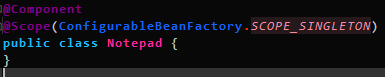
2. 原型（Prototype）：每次注入或通过 Spring 上下文获取时都创建新的实例

3. 会话（Session）：在 Web 应用中，为每个会话创建一个 bean 实例

4. 请求（Request）：在 Web 应用中，为每个请求创建一个 bean 实例

使用 @Scop 组合 @Bean 和 @Component（包括自注解，被 @Component 注解的注解）。

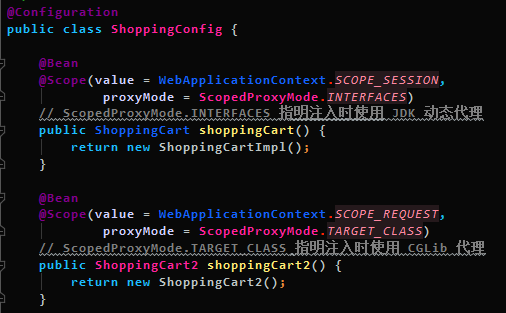




在 xml 中，bean 标签有对应的 socp 属性。

## **会话和请求作用域**

就购物车而言，会话作用域最为合适。



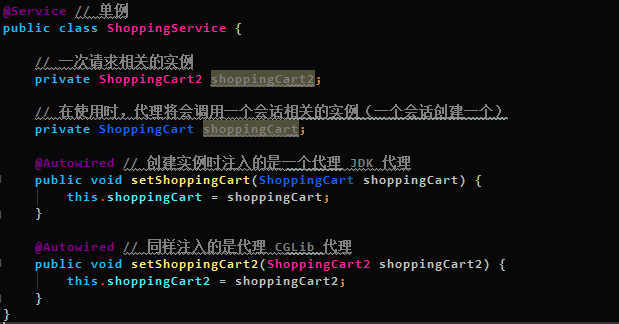
SCOPE\_SESSION 对于一次会话会创建新的实例。

proxyMode 解决了将会话或请求作用域 bean 注入到单例 bean 中的问题。

实际注入时会注入目标 bean 的一个代理，使用时代理会对其进行懒解析并将调用委托给会话作用域内正真的 bean。

ScopedProxyMode.INTERFACE 指明目标类是一个接口，注入时需使用 JDK 动态代理

ScopedProxyMode.TARGET\_CLASS 指明目标类是类，注入时使用 CGLib 代理



xml 中也提供了相应属性：bean 元素的 scop 属性，以及 bean 元素下的子元素 aop:scoped-proxy 。

<aop:scoped-proxy proxy-target-class="false"/>

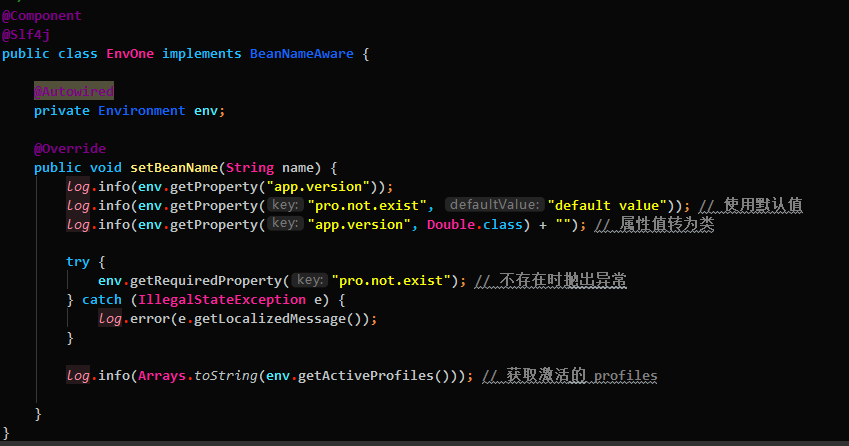
通过 proxy-target-class 属性指明目标 bean 是接口还是类。

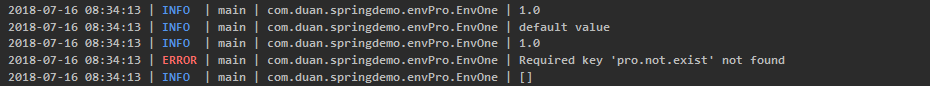
## **运行时注入**

spring 提供了两种在运行时求值的方式：

1. 属性占位符（Property placeholder）

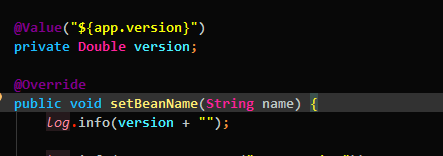
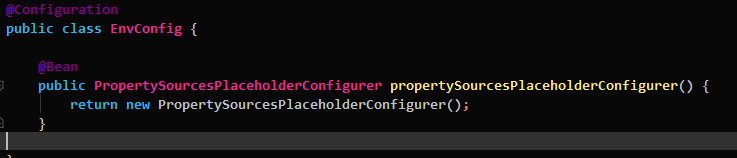
2. spring 表达式语言（SpEL）





**属性占位符**

形式为 "${...}" ，使用 @Value 注解，java 配置需要配置 PropertySourcesPlaceholderConfigurer bean，xml 可通过 <context:property-placeholder /> 开启。



xml 中也可通过 “${...}” 的形式使用。

**使用 spring 表达式语言装配**

1. 使用 bean id 引用其他 bean

2. 调用方法和访问对象属性

3. 对值进行算术、关系和逻辑运算

4. 正则表达式匹配

5. 集合操作

SpEL 要放到 “#{...}” 表达式中，在 xml 中也可以使用。

**#{T(System).currentTimeMillis()}**

T()表达式用于将括号内表达式视为 java 类，能够访问目标类型的静态方法和常量。也直接装配目标类型。

**#{envOne.version}**

envOne 为 bean 的 id，可获得其可访问成员的值

**#{systemProperties['jnidispatch.path']}**

systemProperties获取系统属性

**#{envOne.getClass().toString().toUpperCase()}**

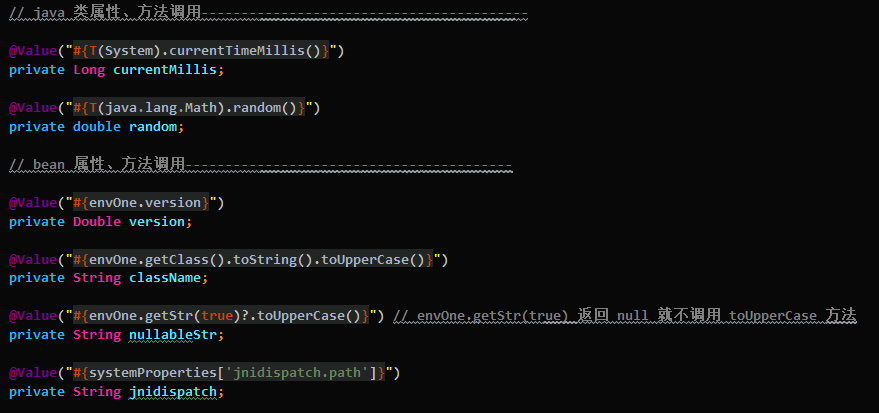
调用 bean 的方法

**#{envOne.getStr(true)?.toUpperCase()}**

envOne.getStr(true) 返回 null 就不调用 toUpperCase 方法

**#{T(java.lang.Math).random()}**

调用 java 类的静态方法



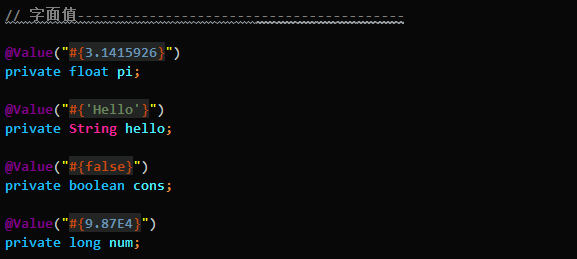
**字面值**

#{3.1415926}

#{'Hello'}

#{false}

#{9.87E4}



**运算符**

算术：+ - \* / % ^

比较：< > == <= >= lt gt eq le ge

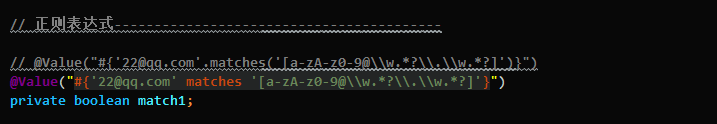
逻辑：and or not |

条件：?: (ternary) ?: (Elvis) 三元运算符，Elvis 表达式

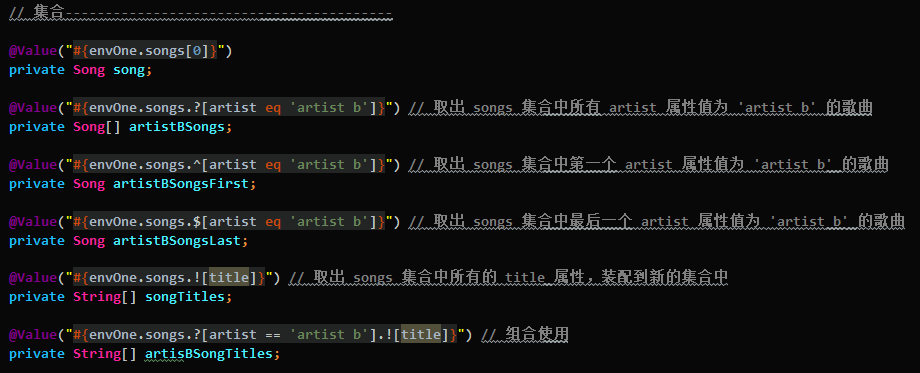
正则：matches

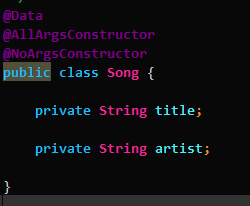


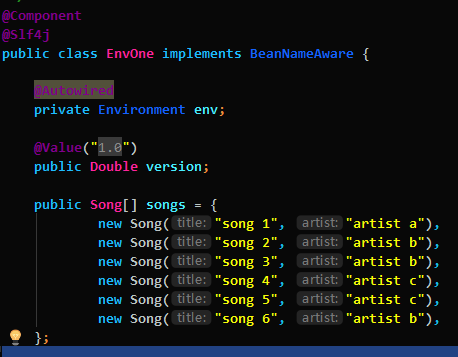
**正则表达式**



**集合**

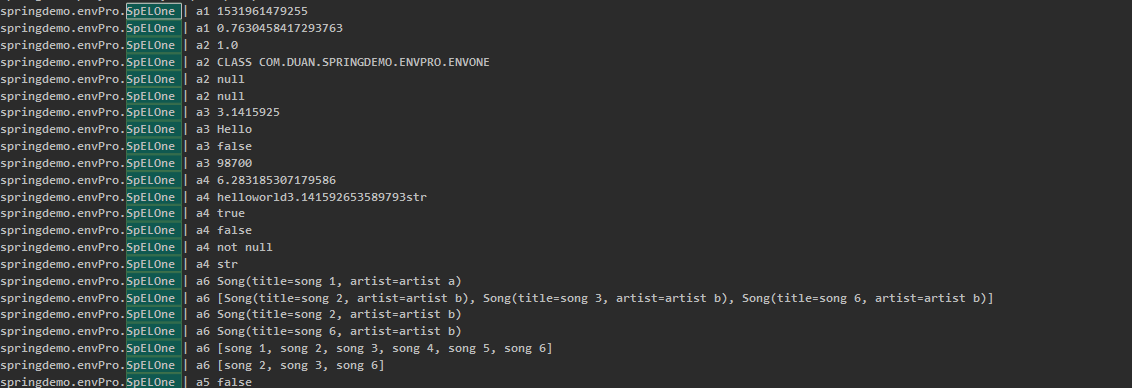






**log输出**





- java 8 允许在一个条目上重复使用一个注解，只要这个注解本身有 @Repeatable 注解。