# Hibernate Validator 校验框架

## 为什么使用校验框架

参数验证是一个常见的问题，例如验证用户输入的密码是否为空、邮箱是否合法等。但是无论是前端还是后台，都需对用户输入进行验证，以此来保证系统数据的正确性。对于web来说，有些人可能理所当然的想在前端验证就行了，但这样是非常错误的做法，前台的验证一般是通过JavaScript，js代码是可以被禁用和篡改的，所以相对后台检验而言，安全性会低一些。前端代码对于用户来说是透明的，稍微有点技术的人就可以绕过这个验证，直接提交数据到后台。无论是前端网页提交的接口，还是提供给外部的接口，参数验证随处可见，也是必不可少的。总之，一切用户的输入都是不可信的。

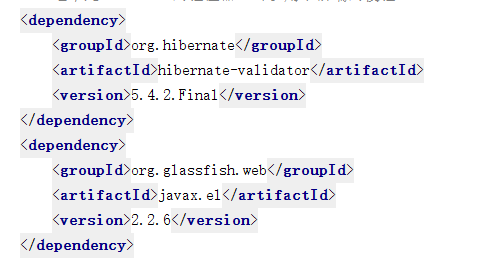
基于这样的常识，在后端的代码开发中，参数校验是一个永远也绕不开的话题。然而编写参数校验代码的过程是一个技术含量不高，及其耗时，繁琐的过程。比如极大多数的参数校验代码就是：判断一下用户名是否已经存在、用户名长度是否满足要求、用户填写的邮箱是否合法。年龄参数是不是一个整数等等。为了减少重复代码的开发，许多有技术积淀的公司，一般都会提供一套或多套特有的参数校验工具类。以此减少代码量。这种做法在项目中非常普遍，有了这些工具类库，程序员在编写业务代码的过程中，工作效率可以大大提升。

## 校验框架Hibernate Validator

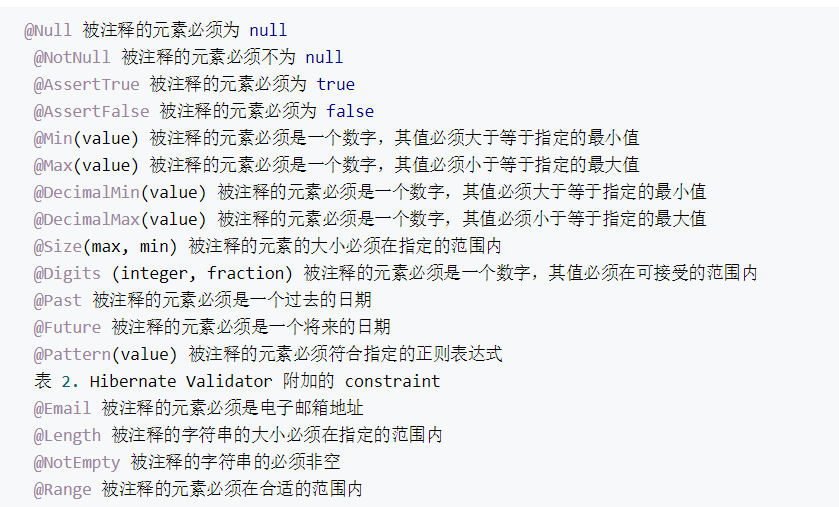
#### 简单介绍

Hibernate Validator，The Bean Validation reference implementation. Express validation rules in a standardized way using annotation-based constraints and benefit from transparent integration with a wide variety of frameworks. 反正就是通过注解进行校验

#### 配置环境



#### 基本注解了解



[校验框架](<https://blog.csdn.net/liuchuanhong1/article/details/52042294>)

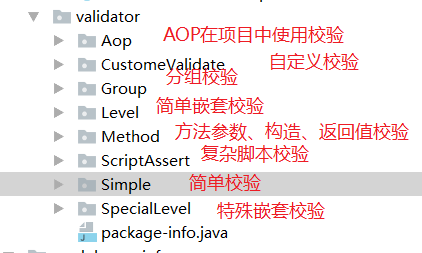
[参考文档](<http://jinnianshilongnian.iteye.com/blog/1990081>)

[参考文档](<http://docs.jboss.org/hibernate/stable/validator/reference/en-US/html_single/#preface>)

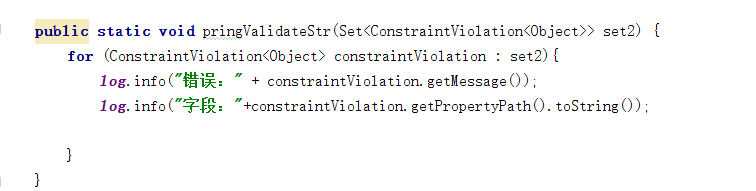
## Hibernate Validator 实践

注意：有些注解必须是当前属性非空的情况下才起作用，一定要定义非空校验注解哦！

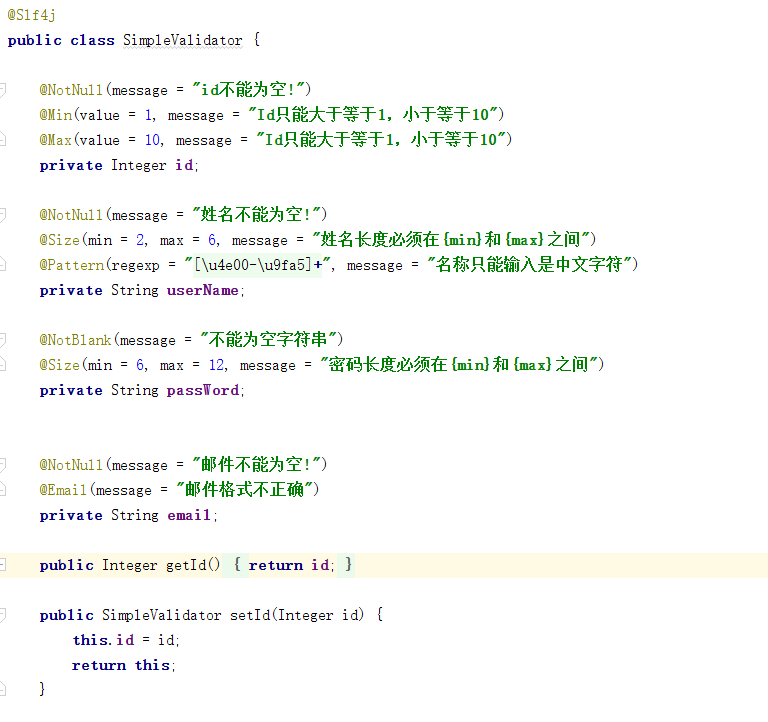
包层次结构



打印错误信息

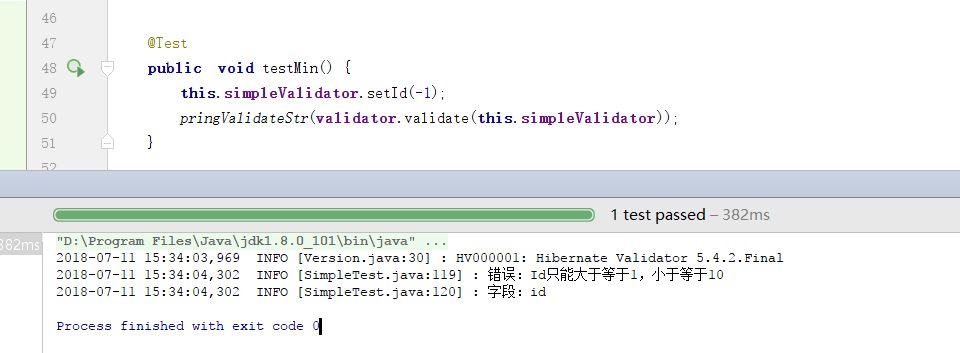


#### SimpleValidator 简单校验一个Java类中的字段





测试错误



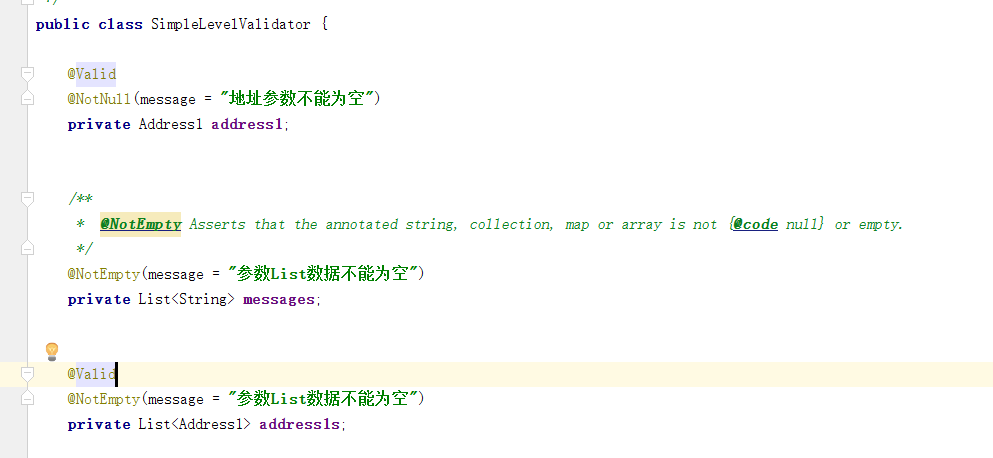
#### 针对某个Class的属性校验

和基本的处理校验一样的，当前类和Simple Validator在一个测试类中

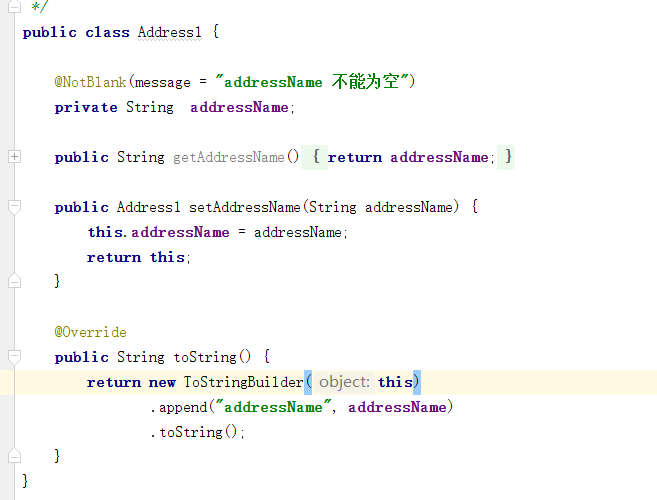


#### 嵌套校验

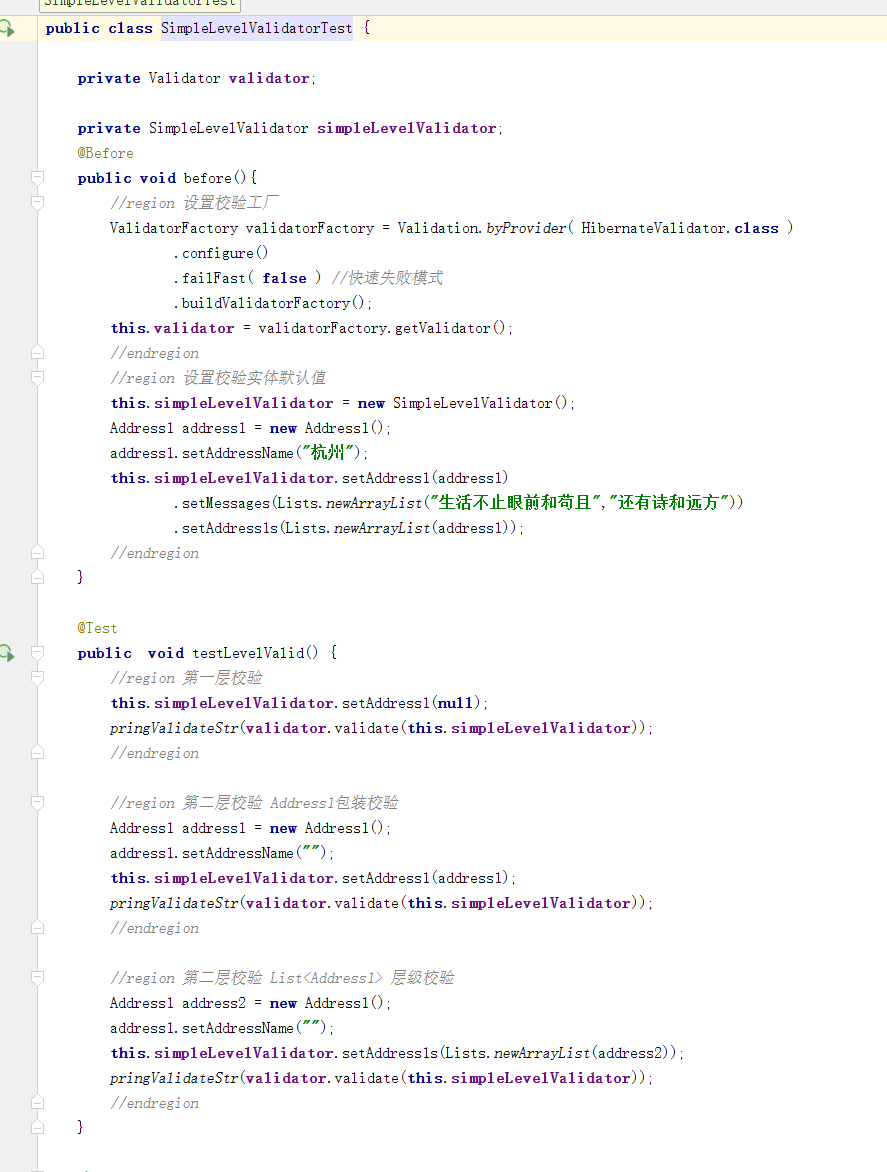
测试类，嵌套校验 List<Address> 、Address、List<String>



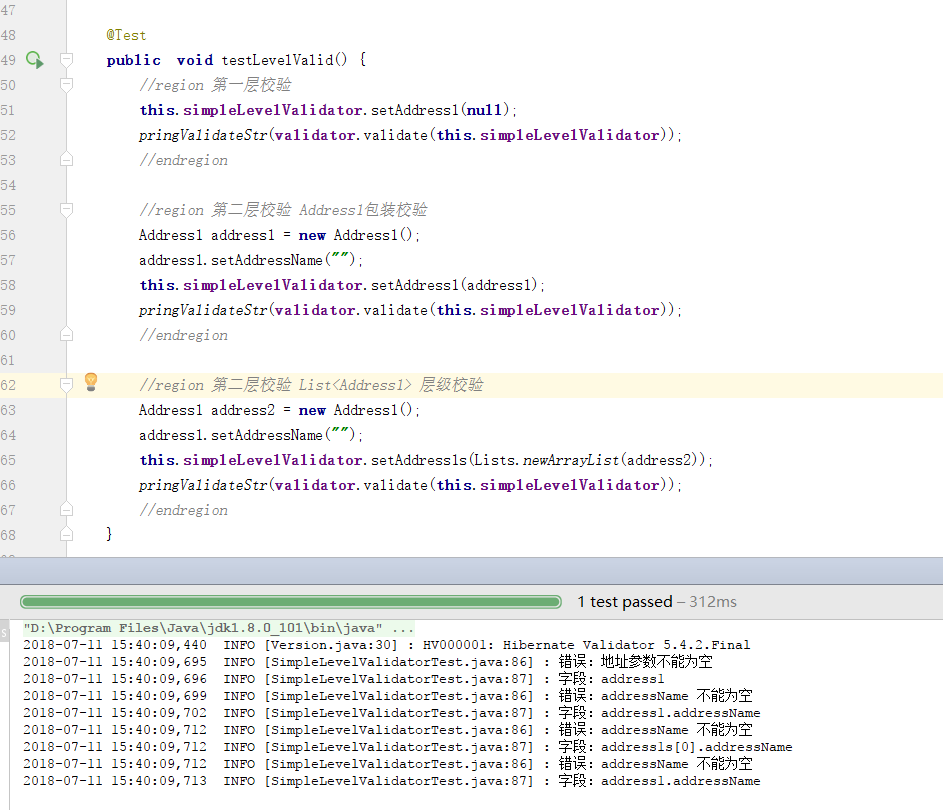
嵌套信息类 Address1



测试校验

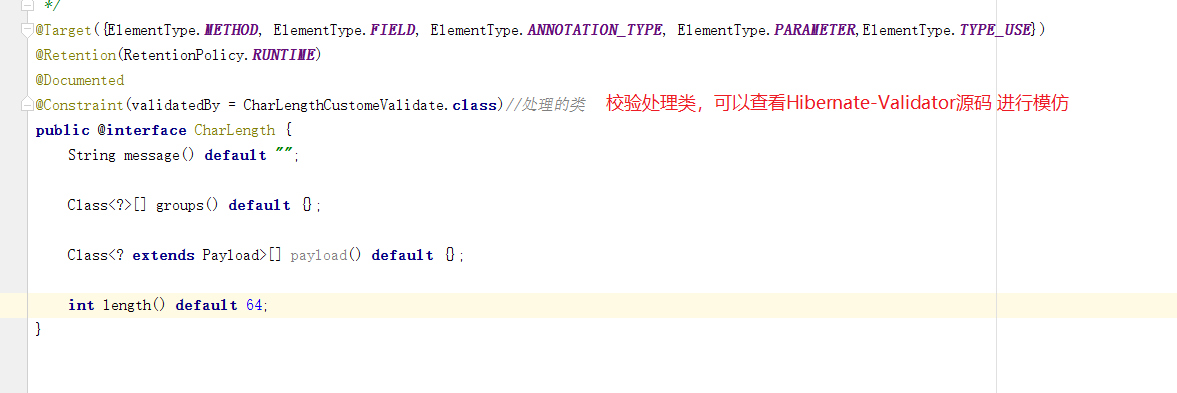


测试结果：



#### 自定义校验

自定义校验处理的注解：字符串长度校验



校验处理实现类：



使用：和基本的校验注解一样的，放在字段上即可；

#### 特殊嵌套校验

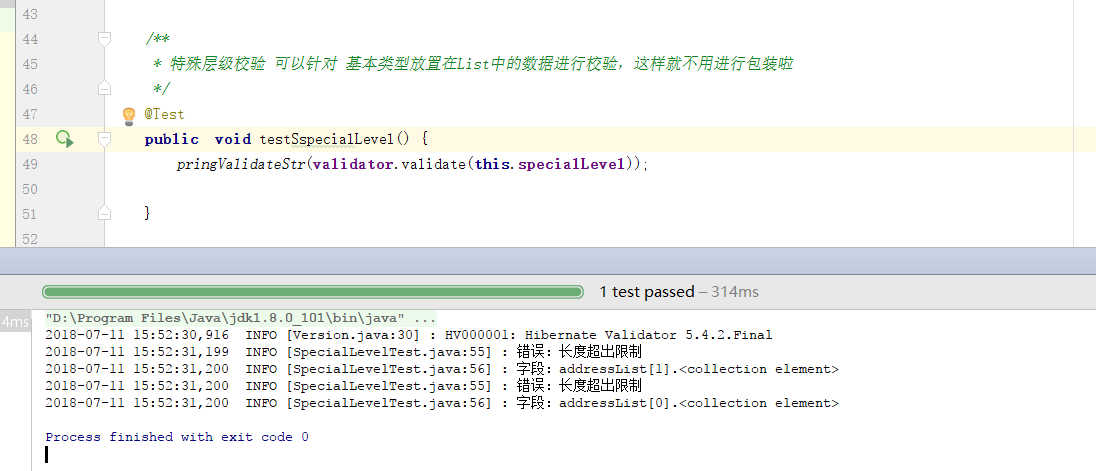
为何特殊啦 List<String> ，如果我想要校验String 泛型参数的数据，如何处理？@NotEmpty 只能查看是否为空？根据Hibernate 官方文档Java1.8以上版本支持的注解 ElementType.TYPE\_USE 支持放在泛型的内部，这样就可以进行校验啦！之前自定义校验的时候，添加了这个注解在长度校验中，所以可以使用。



测试：



测试结果：

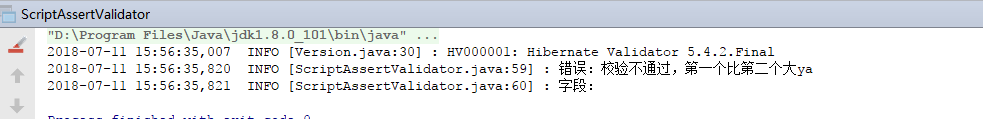


#### 复杂的校验

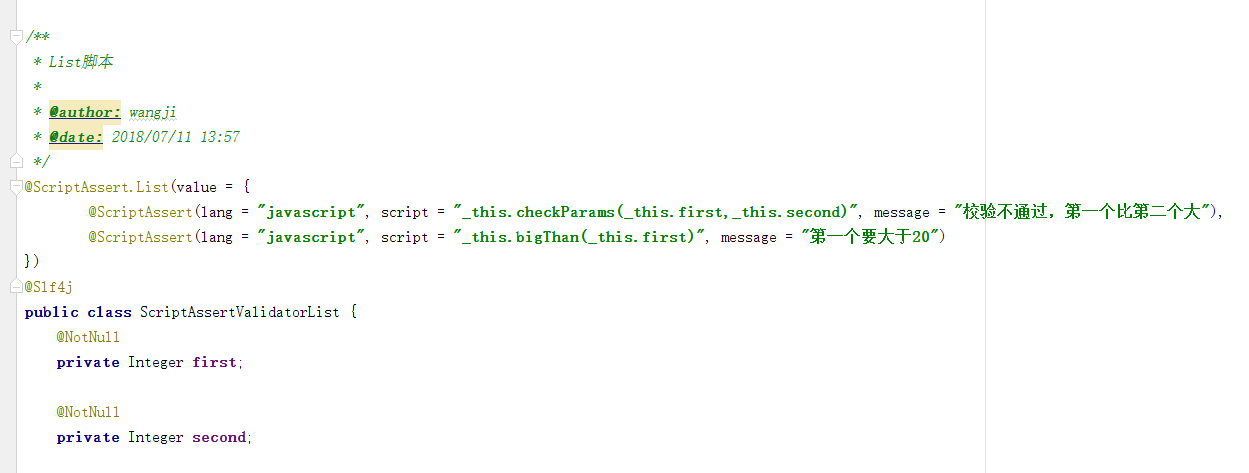
比较复杂的校验需要根据当前字段的值进行处理，这个时候比较复杂，可以通过脚本处理进行简化；



测试结果：



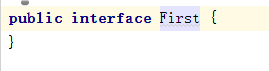
多个校验脚本处理效果一样@ScriptAssert.List



#### 分组校验

何为分组校验，分组校验的意思就是，比如我第一次插入数据不需要校验主键，更新的时候需要校验主键，这种不同情况下校验的选择不同，所以需要分组校验；默认情况下分组为Default.class;

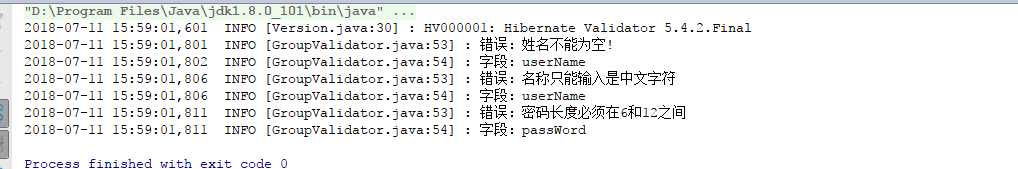
分组标识为自己随便定义一个接口就好啦，如下：



分组校验：由于默认情况下就使用Default.class,所以如果不需要分组，分组信息可以不填的！这里只有NotNull注解未分组，所以就被默认分组包含；



测试结果：

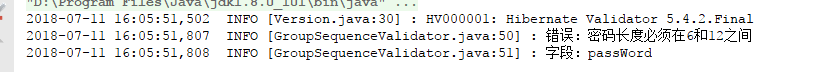


#### 默认分组顺序校验

默认的分组支持顺序校验，当第一个不成功，就不继续校验其它的啦！项目中使用较少；



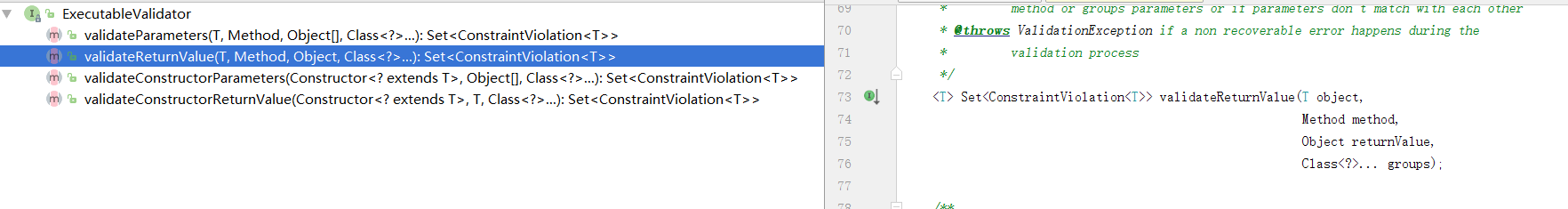
测试结果：



#### 校验方法参数、构造函数、方法返回值

校验这些数据是通过AOP处理Hibernate Validator的基础，通过这些方法能够更好的理解通过AOP处理实现的细节；上面的部分基本上处理的数据都是在基本的Object数据实例的校验；

ExecutableValidator：Validates parameters and return values of methods and constructors.



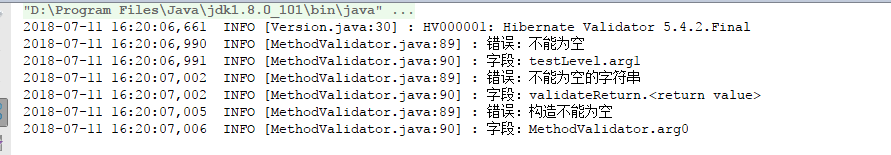
需要校验的方法：



从上到下的处理方法参数、返回值、构造函数



测试结果：



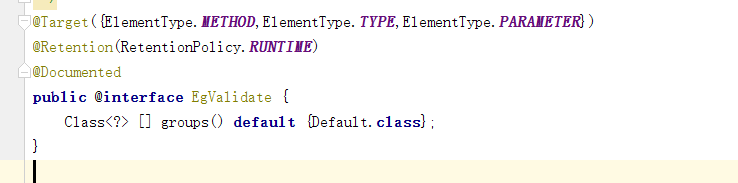
## AOP实现思路

#### 基本思路梳理

我们通过注解能够获取到当前调用方法的参数值，在执行方法调用之前进行参数校验，然后通过Hibernate Validator 对于方法参数校验进行处理，这里处理应该保护当前的校验的分组信息获取（通过注解获取到哦）、对于方法校验还需变量所以的方法参数进行Class中注解信息校验，能够处理好这些数据信息，然后通过抛出错误的异常信息，全局异常进行捕获处理即可完成；

#### 实践实现AOP校验

##### 定义校验的注解（包含一个数据分组信息）



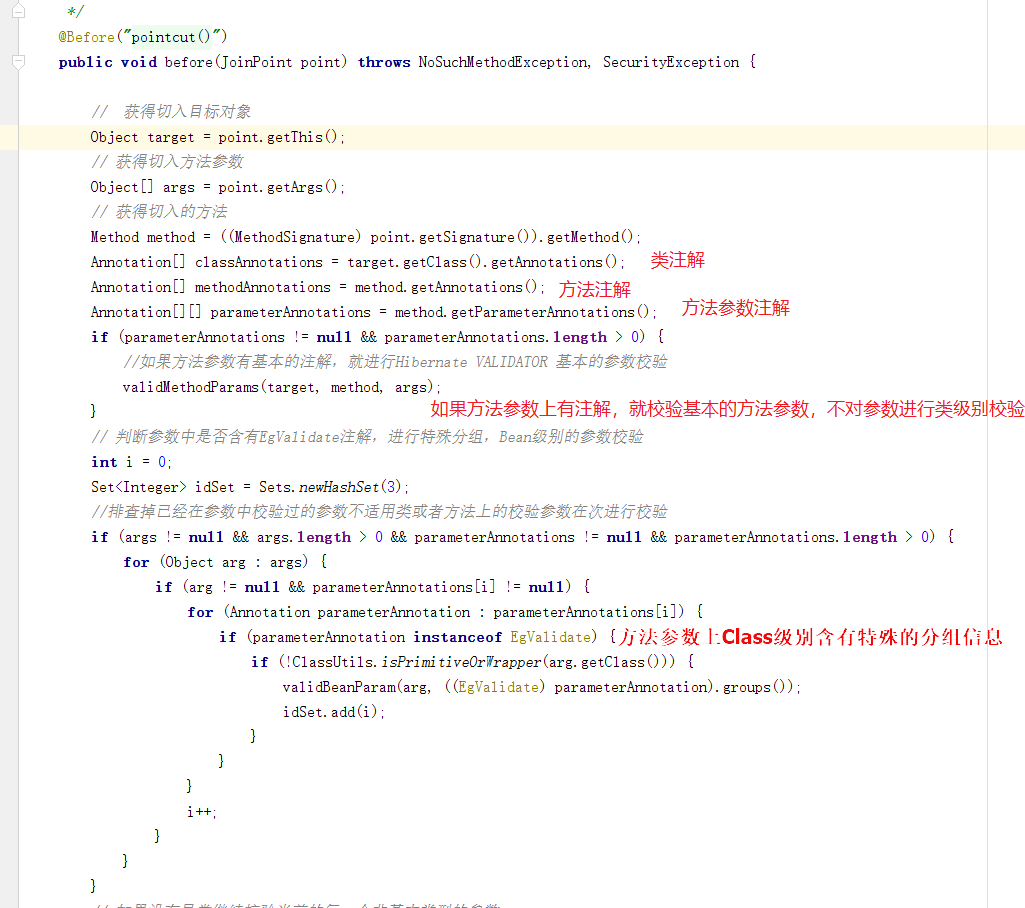
##### 定义AOP拦截特定的注解

放在类上面拦截所有的方法、放在方法上只拦截当前方法



##### 在方法执行之前进行拦截处理

这个阶段是重点处理阶段，需要获取方法参数、当前执行方法、执行方法注解、方法参数注解、方法参数等等；还需要获取当前类中是否定义类级别的分组信息、方法上分组信息，如果方法上有分组信息就使用方法上的，按照优先级使用分组信息；



基本的方法参数校验处理完成啦，就开始处理Class级别的参数校验（非基本数据类型）,如果方法参数上有@EgValidator 可以获取到分组信息就立即进行Class级别数据校验，否则下面进行通过获取方法上注解、类上注解，以优先级为原则，选取默认的分组信息处理，最后进行类级别数据校验；

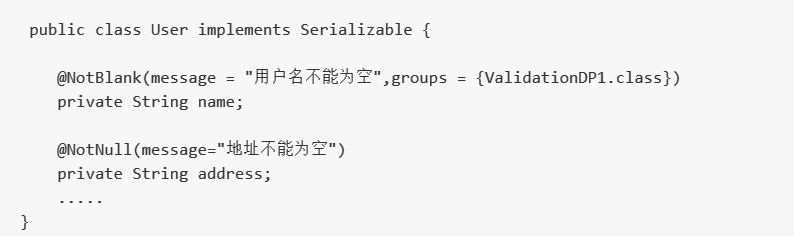


然后检测到有异常就抛出去就行啦！



#### 实践

##### 定义一个Class



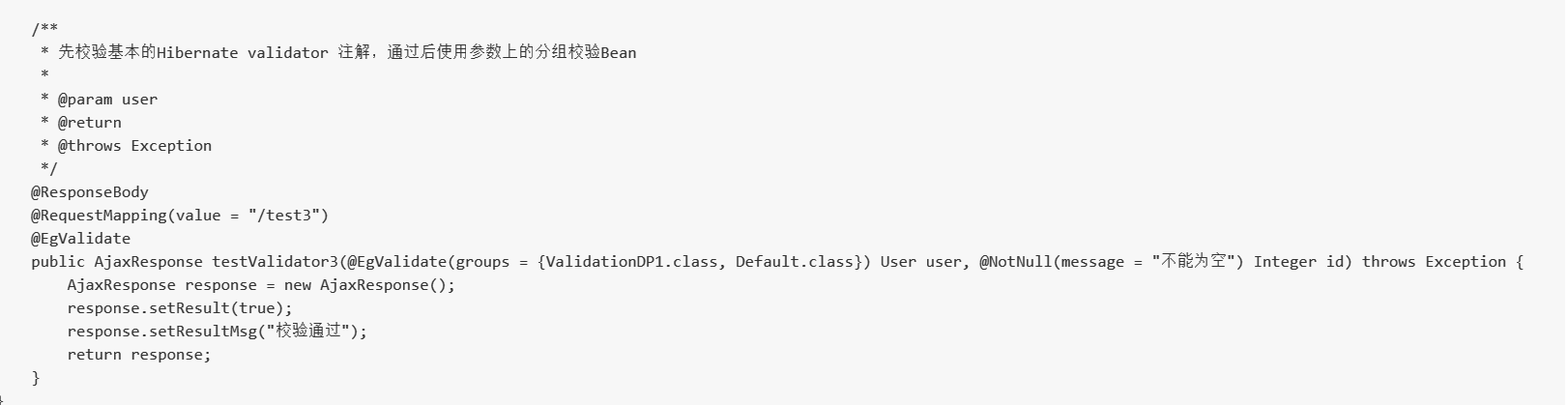
##### 不使用校验分组



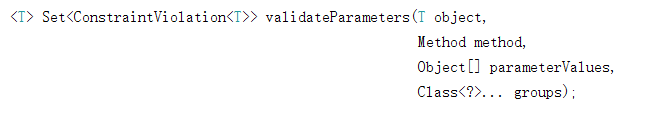
##### 使用参数上校验分组



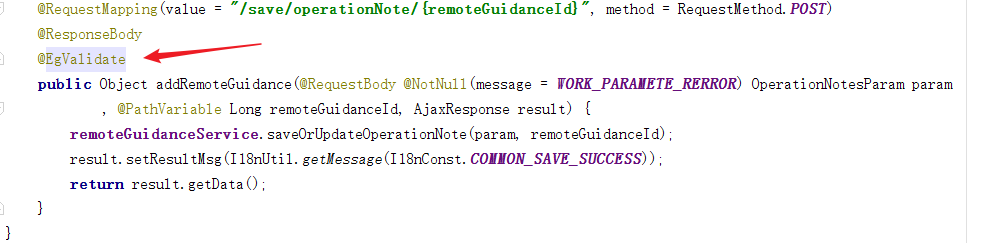
##### 使用参数上分组，还有基本的注解（这种情况基本不常见，Service层校验可能会有这种奇葩）



##### 正常情况下不会出现这么复杂，所以这里实现的时候没有把方法上基本类型的校验分组进行处理；一般情况下分组都是在一个类中使用，所以这里没有进行处理；处理的情况比较复杂实用性不是很强！



基本上正常情况下处理都是这样的，分组都没有定义！直接处理ok！



## 参考文档

[Hibernate-validator校验架](<https://blog.csdn.net/liuchuanhong1/article/details/52042294>)

[记录Hibernate Validator实践[](https://blog.csdn.net/u012881904/article/details/79538895)](%5d(https://blog.csdn.net/u012881904/article/details/79538895))

[官方文档](<http://docs.jboss.org/hibernate/stable/validator/reference/en-US/html_single/#preface>)

[学习记录文档](<http://jinnianshilongnian.iteye.com/blog/1990081>)