# Java后台校验

**项目位置：<https://github.com/coldbloodanimal/validation>**

**核心依赖**

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-validation</artifactId>  
</dependency>

**项目中应用位置例子：**ice.snowflake.controller.ValidateController，方法参数中有@Valid注解

**概述**：java后台校验，可以有更友好的提示（比起不做校验）等。常用两种方式：java编码方式、注解方式。编码写起来代码较多，写起来麻烦。所以我们就说说注解校验。

**注解简单分类有**：

基础注解：@NotBlank 字符串不为空，@ NotNull 对象不为空 @Max 数字最大值 @Min 数字最小值 @Size 字符串长度等，它们是用在属性上，都拥有groups属性，另外@Valid和@Validated 声明某个对象将会被验证,一般用在方法的参数中

分组注解：如上基础注解都有一个属性 groups，它的主要作用：比如对象的id增加(不需要)和修改(需要)操作时候,我们可以分组来区分，例子在下面

常用基础注解有NotBlank， NotNull

嵌套注解：一个对象中包含的其它对象也需要校验的时候，就需要用到嵌套注解，在包含的对相声使用@Valid注解就可以了  
@Valid和@Validated的区别：前者支持嵌套校验，后者支持分组校验，如果两者都需要的时候，没处理过，大概两个注解都加上就可以了。先放下，以后用到了，我在处理。

其它人写的较好的资料：https://blog.csdn.net/kinginblue/article/details/70186586

测试案例写在：test 源文件下

基础校验和分组校验

@Data

public class Apple {

/\*\*

\* Update和Delete只是两个标记接口，没有任何内容，下面的意思是修改和删除时候都要校验id，但是新增的时候就不需要

\* 具体可看ValidateControllerTest.group()方法

\* notblank字符串不能为空

\* \*/

@NotBlank(groups={Update.class,Delete.class},message="id不能为空")

private String id;

@NotBlank(groups=Add.class,message="名称不能为空")

private String name;

/\*\*

\* 对象不能为空

\* \*/

@NotNull(groups=Add.class,message="电话号码不能为空")

private String phonenumber;

/\*\*

\* 数字最大长值不能超过200，

\* 最小值不能小于0

\* \*/

@Max(value=200,groups=Add.class)

@Min(value=0,groups=Add.class)

private int age;

@Max(3)

@Min(0)

private int sex;

@Size(max=20)

private String description;

}

嵌套注解：

@Data

public class Summoner {

private String username;

private int age;

private String sex;

/\*\*

\* valid支持嵌套校验，就是在传入Summoner对象的时候，也会校验Pet中的合法性

\* 具体可看ValidateControllerTest.addsummoner() .addsummoner1() addsummoner2()方法

\* \*/

@Valid

@NotNull(message = "你需要一个宠物")

private Pet pet;

}

@Data

public class Pet {

@NotBlank(message = "宠物名称不能为空")

private String name;

private String sex;

}

对于错误的全局处理：

/\*\*

\* 从名字看出这玩意儿是个controller层次的通知，应该是aop的一种使用

\* \*/

@ControllerAdvice

public class GlobalExceptionHandler {

/\*\*

\* 对于参数错误统一处理，负责就需要单个处理，如: ValidateController.delete

\* \*/

@ExceptionHandler(MethodArgumentNotValidException.class)

@ResponseBody

String handleMethodArgumentNotValidException(MethodArgumentNotValidException e){

throw new RuntimeException(e.getBindingResult().getFieldError().getDefaultMessage());

}

}

**声明**：绝大部分内容为个人理解，仅供参考