

快速入门基于valid与validated的参数校验

文章分类: *JavaDemo*; 标签: *JavaCodeSnippet*; 作者: *Hackyle*;
更新时间: *Tue Aug 15 11:12:18 CST 2023*

1. 参数校验

2. 注解类型

3. @Valid

1. 校验表单实体

2. 校验List实体

3. 嵌套校验

4. @Validated

1. 基本用法

2. 校验失败的异常

3. 拦截处理异常

4. 测试

1. JSON实体类

2. 表单实体类

3. 校验单个参数

4. 数组与嵌套

5. 分组校验

背景

- 在平常的项目开发中，我们常需要校验前端传递的参数是否合法
- 对于是新增、修改的数据，校验参数是否合法的意义在于防止异常的数据落到数据库层导致数据库异常
- 对于是查询的数据，校验参数是否合法的意义在于过滤掉一些明显不合法的查询条件，防止恶意查询，减少数据库的查询压力

```
1 @PostMapping("/add")
2 private ApiResponse<String> add(@RequestBody ApiRequest<PersonDto> apiRequest) {
3     PersonDto person = apiRequest.getData();
4     //入参校验的常规做法
5     String name = person.getName();
6     if(null == name || "".equals(name) || name.length() > 20) {
7         return ApiResponse.error(4000, "入参校验不通过: name不合法");
8     }
9     Integer age = person.getAge();
10    if(null == age || age < 0 || age > 100) {
11        return ApiResponse.error(4000, "入参校验不通过: age不合法");
12    }
13    //...
14
15    //personService.add(person);
16
17    return ApiResponse.success(2000, "成功");
18 }
```

本文主要内容

- 介绍两种参数校验方式（valid与validated）的基本用法
- 提供相应的测试案例，供读者深度理解

内容导览

- 参数校验
- 注解类型
- @Valid
 - 校验表单实体
 - 校验List实体
 - 嵌套校验
- @Validated
 - 基本用法
 - 校验失败的异常
 - 拦截处理异常
 - 测试
 - JSON实体类
 - 表单实体类
 - 校验单个参数
 - 数组与嵌套
 - 分组校验

参数校验

hibernate-validator对其进行了实现。

- 可以用在方法、构造函数、方法参数和成员属性（字段）上
- 支持嵌套检测：在一个beanA中，存在另外一个beanB属性。嵌套检测beanA同时也检测beanB
- 不支持分组
- @Valid 进行校验的时候，需要用 BindingResult 来做一个校验结果接收。当校验不通过的时候，如果手动不 return，则并不会阻止程序的执行；

@Validated

- @Validated是只用Spring Validator校验机制使用
- 可以用在类型、方法和方法参数上。但是不能用在成员属性（字段）上
- 不支持嵌套检测
- 支持分组
- @Validated 进行校验的时候，当校验不通过的时候，程序会抛出400异常，阻止方法中的代码执行，这时需要再写一个全局校验异常捕获处理类，然后返回校验提示。

与SpringBoot整合

- 在SpringBootv2.3之前的版本只需要引入 web 依赖就可以了，他包含了validation校验包
- 而在此之后SpringBoot版本就独立出来了需要单独引入依赖

```
1 <dependency>
2   <groupId>org.springframework.boot</groupId>
3   <artifactId>spring-boot-starter-validation</artifactId>
4 </dependency>
5
6 注意：这个依赖的本质还是
7 <dependency>
8   <groupId>org.hibernate</groupId>
9   <artifactId>hibernate-validator</artifactId>
10  <version>5.4.1.Final</version>
11 </dependency>
```

注解类型

字符串空值检查

- @NotBlank(message =) 验证字符串非null，且长度必须大于0
- @NotEmpty 不能为null，集合、数组、map等size()不能为0；字符串trim()后可以等于 ""
- @NotNull 不能为null，可以为空
- @Null 必须为null

字符串格式检查

- @Pattern(regex=,flag=) 被注释的元素必须符合指定的正则表达式
- @Email 被注释的元素必须是电子邮箱地址
- @Length(min=,max=) 被注释的字符串的大小必须在指定的范围内
- @URL(protocol=,host=, port=, regexp=, flags=) 被注释的字符串必须是一个有效的url

验证数字

- @DecimalMax (value=x) 被注解的元素值小于等于(<=)指定的十进制value 值
- @DecimalMin (value=x) 被验证注解的元素值大于等于指定的十进制value 值
- @Digits(integer=整数位数, fraction=小数位数) 验证注解的元素值的整数位数和小数位数上限
- @Max (value=x) 验证注解的元素值小于等于指定的 value值
- @Min (value=x) 验证注解的元素值大于等于指定的 value值
- @Size(max=, min=) 被注释的元素的大小必须在指定的范围内，可以是 String、Collection、Map、数组
- @Range 值必需在一个范围内
- @Positive: 被注释的元素必须为正数
- @PositiveOrZero: 被注释的元素必须为正数或 0
- @Negative: 被注释的元素必须为负数
- @NegativeOrZero: 被注释的元素必须为负数或 0

验证日期

- @PastOrPresent: 被注释的元素必须是现在或者过去的日期

验证布尔

- @Null 验证元素必须为 null
- @NotNull 验证元素必须不为 null
- @AssertTrue 验证元素必须为 true
- @AssertFalse 验证元素必须为 false

实例

```
1  @Data
2  public class StudentAddDto {
3      @NotBlank(message = "主键不能为空")
4      private String id;
5
6      @NotBlank(message = "名字不能为空")
7      @Size(min=2, max = 4, message = "名字字符长度必须为 2~4个")
8      private String name;
9
10     @Pattern(regexp = "/^1(3\\d|4[5-9]|5[0-35-9]|6[567]|7[0-8]|8\\d|9[0-35-9])\\d{8}$")
11     private String phone;
12
13     @Email(message = "邮箱格式错误")
14     private String email;
15
16     @Past(message = "生日必须早于当前时间")
17     private Date birth;
18
19     @Min(value = 0, message = "年龄必须为 0~100")
20     @Max(value = 100, message = "年龄必须为 0~100")
21     private Integer age;
22
23     @PositiveOrZero
24     private Double score;
25 }
```

@Valid

核心特性

- 在Controller层使用“@Valid”修饰实体类参数
- 在实体类的属性上使用校验注解
- 使用BindingResult接收校验结果，手动控制是否校验通过

```
import lombok.Data;
import org.hibernate.validator.constraints.Range;

import javax.validation.constraints.*;
import java.math.BigDecimal;
import java.util.Date;

@Data
public class PersonAddDto {
    private Long id;

    @NotBlank(message = "请输入名称")
    @Size(message = "名称字符长度在{min}到{max}之间", min = 1, max = 10)
```

```

17     @Range(message = "年龄范围为 {min} 到 {max} 之间", min = 1, max = 100)
18     private Integer age;
19
20     @Max(value = 2, message = "性别限定最大值为2")
21     @Min(value = 1, message = "性别限定最小值为1")
22     @Positive(message = "性别字段只可能为正数")
23     private Integer gender;
24
25     @PastOrPresent(message = "出生日期一定在当前之间之前")
26     private Date birthday;
27
28     private String address;
29
30     @Pattern(regexp = "^1(3[0-9]|4[01456879]|5[0-35-9]|6[2567]|7[0-8]|8[0-9]|9[0-
31     private String tel;
32
33     @Email(message = "邮箱格式错误")
34     private String email;
35
36     @DecimalMax(value = "99999", message = "工资上限为99999")
37     //@DecimalMin(value = "99", message = "工资下限为99")
38     @PositiveOrZero
39     private BigDecimal salary;
40
41 }

```

校验表单实体

关键语法

- 在Controller层使用“@Valid”修饰实体类参数
- 在实体类中使用校验注解
- 使用BindingResult接收校验结果

```

import com.alibaba.fastjson.JSON;
import com.ks.demo.vv.dto.ApiResponse;
import com.ks.demo.vv.dto.PersonAddDto;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import javax.validation.Valid;

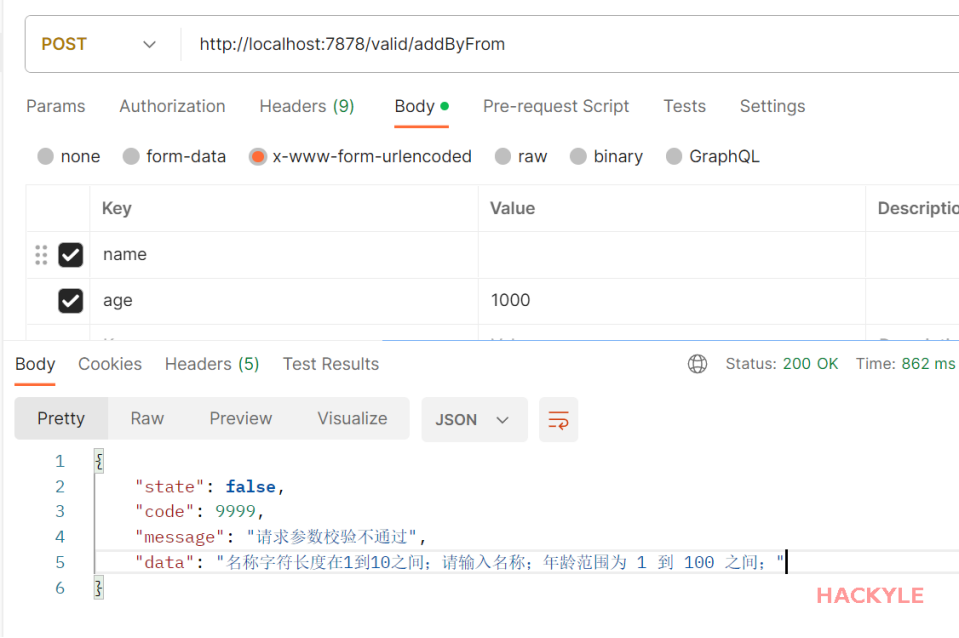
@RequestMapping("/valid")
@RestController
public class ValidController {

    @PostMapping("/addByFrom")
    private ApiResponse<String> addByFrom(@Valid PersonAddDto personAddDto, BindingResult bindingResult) {

        if(bindingResult.hasErrors()) {
            //return ApiResponse.error(9999, "请求参数校验不通过", JSON.toJSONString(
            StringBuilder sb = new StringBuilder();
            bindingResult.getAllErrors().forEach(ele -> sb.append(ele.getDefaultMessage()));
            return ApiResponse.error(9999, "请求参数校验不通过", sb.toString());
        }
    }
}

```

```
26
27
28     return ApiResponse.success(2000, "成功");
29 }
```



校验List实体



- 在Controller层使用“ @Valid List<PersonAddDto> personDtoList” ，是无法检测List中的实体属性的
- 最佳解决方案: @RequestBody @Valid ValidList<PersonAddDto> personDtoList, 也即将List包一层, 并使用@Valid修饰该个List

```
import com.alibaba.fastjson.JSON;
import com.ks.demo.vv.dto.ApiResponse;
import com.ks.demo.vv.dto.PersonAddDto;
import com.ks.demo.vv.dto.ValidList;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import javax.validation.Valid;
import java.util.List;

@RequestMapping("/valid")
@RestController
public class ValidController {

    /**
     * 校验失效, 不支持检验List中的实体
     */
    @PostMapping("/add")
    private ApiResponse<String> add(@RequestBody @Valid List<PersonAddDto> personD
        if(bindingResult.hasErrors()) {
            //return ApiResponse.error(9999, "请求参数校验不通过", JSON.toJSONString
```

```
28     }
29
30     System.out.println(JSON.toJSONString(personDtoList));
31
32     return ApiResponse.success(2000, "成功");
33 }
34
35 /**
36  * 解决校验List中的实体：使用一个对象包装一层List，其本质是"嵌套校验"
37  * 最佳的解决方案是下文将要介绍的将请求体以ApiRequest封装，在"T data"上使用@Valid修
38  */
39 @PostMapping("/addListByNest")
40 private ApiResponse<String> addListByNest(@RequestBody @Valid ValidList<Persor
41     if(bindingResult.hasErrors()) {
42         //return ApiResponse.error(9999, "请求参数校验不通过", JSON.toJSONString
43         StringBuilder sb = new StringBuilder();
44         bindingResult.getAllErrors().forEach(ele -> sb.append(ele.getDefaultMe
45         return ApiResponse.error(9999, "请求参数校验不通过", sb.toString());
46     }
47
48     System.out.println(JSON.toJSONString(personDtoList));
49
50     return ApiResponse.success(2000, "成功");
51 }
52
53 }
54
55 import lombok.Data;
56
57 import javax.validation.Valid;
58 import java.util.List;
59
60 @Data
61 public class ValidList<T> {
62     @Valid
63     private List<T> dataList;
64
65 }
```

HACKYLE

首页

文章分类

文章标签

关于

留言

Params

Authorization

Headers (5)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

▼

1

{

2

"dataList": [

3

{

4

"name": "",

5

"age": 999,

6

"email": "dddd"

7

}

8

]

9

}

Body

Cookies

Headers (5)

Test Results

⊕

Status: 200 OK

Time: 315 ms

Size: 351 B

Pretty

Raw

Preview

Visualize

JSON

▼

🔍

1

"state": false,

2

"code": 9999,

3

"message": "请求参数校验不通过",

4

"data": "邮箱格式错误; 请输入名称; 年龄范围为 1 到 100 之间; 名称字符长度在1到10之间;"

5

6

HACKYLE

嵌套校验

嵌套检测：在一个beanA中，存在另外一个beanB属性。嵌套检测beanA同时也检测beanB

```
1  /**
2   * 嵌套检测
3   * 重点关注ApiRequest中"private T data;"上的"@Valid"
4   */
5  @PostMapping("/addByObjNest")
6  private ApiResponse<String> addByObjNest(@RequestBody @Valid ApiRequest<PersonAddE
7      if(br.hasErrors()) {
8          return ApiResponse.error(9999, "请求参数校验不通过", errorInfo(br));
9      }
10
11      System.out.println(JSON.toJSONString(personDto));
12      return ApiResponse.success(2000, "成功");
13  }
```

POST

▼

http://localhost:7878/valid/addByObjNest

S

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

▼

1

{

2

"data": {

3

"name": "",

4

"age": 999,

5

"gender": 3,

6

"emial": "fff",

7

"birthday": "2023-12-12 12:12:12"

8

}

9

}

Body

Cookies

Headers (5)

Test Results

⊕

Status: 200 OK

Time: 64 ms

Size: 400 B

📄 Save as E

Pretty

Raw

Preview

Visualize

JSON

▼

🔍

1

"state": false,

2

"code": 9999,

3

"message": "请求参数校验不通过",

4

"data": "出生日期一定在当前时间之前; 请输入名称; 年龄范围为 1 到 100 之间; 性别限定最大值为2; 名称字符长度在1到10之间;"

5

6

HACKYLE

```
/**
 * 嵌套List
 */
@PostMapping("/addByListNest")
```

HACKYLE

首页

文章分类

文章标签

关于

留言

8

}

9

10

System.out.println(JSON.toJSONString(personDto));

11

12

return ApiResponse.success(2000, "成功");

13

}

POST

http://localhost:7878/valid/addByListNest

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

1

{

2

"data": [

3

{

4

"name": "",

5

"age": 999,

6

"gender": 3,

7

"emial": "fff",

8

"birthday": "2023-12-12 12:12:12"

9

},

10

],

11

"name": "",

12

"age": 999,

13

"gender": 3,

14

"emial": "fff",

15

"birthday": "2023-12-12 12:12:12"

16

}

17

}

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Time: 38 ms

Size: 559 B

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"state": false,

3

"code": 9999,

4

"message": "请求参数校验不通过",

5

"data": "出生日期一定在当前之间之前; 名称字符长度在1到10之间; 请输入名称; 年龄范围为 1 到 100 之间; 性别限定最大值为2; 性别限定最大值为2; 请

6

1 到 100 之间; 名称字符长度在1到10之间; 出生日期一定在当前之间之前; "

7

}

HACKYLE

@Validated

核心特性

- 在Controller层使用” @Validated” 修饰实体类参数
- 在实体类的属性上使用校验注解
- 立即失败：一旦校验失败，自动抛出异常（springframework.validation.BindException），结束正在执行中的流程，本个HTTP请求结束，响应500

基本用法

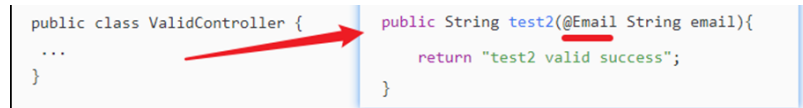
- 校验的参数是**实体**，@Validated注解直接放在该模型参数前即可，属性校验放在实体类的属性上。

```
@PostMapping(value = "test1")
public String test1(@Validated @RequestBody ValidEntity validEntity){
    return "test1 valid success";
}
```

```
@Max(value = 50, message = "年龄超出，出现异常")
private Integer age;
```

在实体类的属性上加验证注解

- 校验的参数是**普通参数**，@Validated要直接放在类上，在具体的参数前加上校验注解。



```
public class ValidController {
    ...
    public String test2(@Email String email){
        return "test2 valid success";
    }
}
```

校验失败的异常

背景：由于校验不通过则立即失败，抛出异常，本次请求结束，响应500。

异常抛出：

- 校验从@RequestBody来的实体，失败抛出：
org.springframework.web.bind.MethodArgumentNotValidException
- 校验普通实体失败抛出：springframework.validation.BindException
- 校验普通参数失败抛出：validation.ConstraintViolationException

```
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import net.hackyle.boot.entity.Person;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import javax.validation.constraints.Email;

@Api("ViolationController")
@Validated
@RestController
public class ViolationController {
    @ApiOperation(value = "test01", notes = "校验从@RequestBody来的实体")
    //校验从@RequestBody来的实体，失败抛出:springframework.MethodArgumentNotValidExc
    @PostMapping("/test01")
    public String test01(@Validated @RequestBody Person person) {
        return "通过校验" + " " + person.toString();
    }

    @ApiOperation(value = "test02", notes = "校验普通实体")
    //校验普通实体失败抛出: org.springframework.validation.BindException
    @GetMapping("/test02")
    public String test02(@Validated Person person) {
        return "通过校验" + " " + person.toString();
    }

    @ApiOperation(value = "test03", notes = "校验普通参数")
    //校验普通参数失败抛出: javax.validation.ConstraintViolationException
    @PostMapping("/test03")
    public String test03(@RequestBody @Email String email) {
        return "通过校验" + " " + email;
    }
}

public class Person {
    @Max(value = 50)
    private Integer age;
```

拦截处理异常

背景:

- 由于校验不通过则立即失败，抛出异常，本次请求结束，响应500。
- 为了不让用户对响应的500而产生疑惑，所以我们需要在全局异常捕获器中捕获Validator的异常，并响应给客户看得懂的信息。

解决方案: 新建一个配置类，并添加@RestControllerAdvice注解，然后在具体方法中通过ExceptionHandler指定需要处理的异常

```
import com.hackyle.boot.common.pojo.ApiResponse;
import org.springframework.validation.BindException;
import org.springframework.web.HttpMediaTypeNotSupportedException;
import org.springframework.web.HttpRequestMethodNotSupportedException;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;
import org.springframework.web.servlet.NoHandlerFoundException;

import javax.naming.AuthenticationException;
import javax.validation.ConstraintViolationException;

@RestControllerAdvice
public class GlobalExceptionHandler {
    /**
     * Validator校验RequestBody实体不通过抛出的异常
     */
    @ExceptionHandler(MethodArgumentNotValidException.class)
    public ApiResponse<String> methodArgumentNotValidException(MethodArgumentNotValidException e) {
        //LOGGER.info("全局异常捕获器-捕获到MethodArgumentNotValidException: ", e);
        return ApiResponse.error(9999, "校验RequestBody的实体不通过"); //这里为了代码
    }
    /**
     * Validator校验单个参数校验失败抛出的异常
     */
    @ExceptionHandler(ConstraintViolationException.class)
    public ApiResponse<String> constraintViolationException(ConstraintViolationException e) {
        //LOGGER.info("全局异常捕获器-捕获到ConstraintViolationException: ", e);
        return ApiResponse.error(9999, "处理单个参数校验失败抛出的异常");
    }
    /**
     * Validator校验普通实体失败抛出的异常
     */
    @ExceptionHandler(BindException.class)
    public ApiResponse<String> bindException(BindException e) {
        //LOGGER.info("全局异常捕获器-捕获到BindException: ", e);
        return ApiResponse.error(9999, "校验普通实体失败抛出的异常");
    }
    /**
     * 请求方法不被允许异常
     */
    @ExceptionHandler(HttpRequestMethodNotSupportedException.class)
    public ApiResponse<String> httpRequestMethodNotSupportedException(HttpRequestMethodNotSupportedException e) {
        //LOGGER.info("全局异常捕获器-捕获到HttpRequestMethodNotSupportedException: ", e);
        return ApiResponse.error(9999, "请求方法不被允许异常");
    }
}
```

```
50 public ApiResponse<String> httpMediaTypeNotSupportedException(HttpMediaTypeNot
51 //LOGGER.info("全局异常捕获器-捕获到HttpRequestMethodNotSupportedException:
52 return ApiResponse.error(9999, "HTTP请求不支持");
53 }
54
55 @ExceptionHandler(NoHandlerFoundException.class)
56 public ApiResponse<String> noHandlerFoundException(NoHandlerFoundException e)
57 //LOGGER.info("全局异常捕获器-捕获到NoHandlerFoundException: ", e);
58 return ApiResponse.error(9999, "接口不存在");
59 }
60
61 @ExceptionHandler(AuthenticationException.class)
62 public ApiResponse<String> authenticationException(AuthenticationException e)
63 //LOGGER.info("全局异常捕获器-捕获到AuthenticationException: ", e);
64 return ApiResponse.error(9999, "认证异常");
65 }
66
67 /**
68 * 总异常：只要出现异常，总会被这个拦截，因为所有的异常父类为：Exception
69 */
70 @ExceptionHandler(Exception.class)
71 public ApiResponse<String> exception(Exception e) {
72 //LOGGER.info("全局异常捕获器-捕获到Exception: ", e);
73 return ApiResponse.error(9999, "总异常");
74 }
75 }
```

测试

主要测试点

- 校验JSON传来的实体类
- 校验表单传来的实体类
- 校验单个参数
- 校验嵌套传来的JSON实体类，注意在嵌套检测处要使用 “@Valid”

JSON实体类

```
1 //校验从@RequestBody来的实体，失败抛出:springframework.MethodArgumentNotValidExcepti
2 @PostMapping("/requestBody")
3 public String requestBody(@Validated @RequestBody PersonAddDto person) {
4     return "通过校验" + " " + JSON.toJSONString(person);
5 }
```

HACKYLE

首页

文章分类

文章标签

关于

留言

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

4

5

6

7

{

... "name": "",

... "age": 666,

... "gender": 3,

... "emial": "fff",

... "birthday": "2023-12-12 12:12:12"

}

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

{

... "state": false,

... "code": 9999,

... "message": "校验RequestBody的实体不通过",

... "data": null

}

POST

http://localhost:7878/validated/requestBody

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

4

5

6

7

{

... "name": "adfad",

... "age": 66,

... "gender": 2,

... "emial": "fff@aa.com",

... "birthday": "2022-12-12 12:12:12"

}

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Time:

Pretty

Raw

Preview

Visualize

Text

1

通过校验 {"age":66,"birthday":"2022-12-12 20:12:12","gender":2,"name":"adfad"}

表单实体类

```
1 //校验普通实体失败抛出: org.springframework.validation.BindException
2 @PostMapping("/entity")
3 public String entity(@Validated PersonAddDto person) {
4     return "通过校验" + " " + JSON.toJSONString(person);
5 }
```

HACKYLE

首页 文章分类 文章标签 关于 留言

Params Authorization Headers (5) **Body** Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL

	Key	Value	Descriptio
<input checked="" type="checkbox"/>	name		
<input checked="" type="checkbox"/>	age	1000	
	Key	Value	Descriptio

Body Cookies Headers (5) Test Results

🌐 Status: 200 OK Time: 902 ms

Pretty Raw Preview Visualize

JSON

↻

1 {

2 "state": false,

3 "code": 9999,

4 "message": "校验普通实体失败抛出的异常",

5 "data": null

6 }

POST

⌵

http://localhost:7878/validated/entity

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL

	Key	Value
<input checked="" type="checkbox"/>	name	啊啊啊啊
<input checked="" type="checkbox"/>	age	10
	Key	Value

Body Cookies Headers (5) Test Results

🌐 Status: 200 OK

Pretty Raw Preview Visualize

Text

↻

1 通过校验 {"age":10,"name":"啊啊啊啊"}

HACKYLE

校验单个参数

```
1 //校验普通参数失败抛出: javax.validation.ConstraintViolationException
2 //注意: 在使用属性校验参数前一定要额外加"@Validated", 也可以加在类上
3 @PostMapping("/param")
4 public String param(@Validated @Email @RequestParam("email") String email) {
5     return "通过校验" + " " + email;
6 }
```

https://blog.hackyle.com/article/java-demo/valid-validated-demo

13/18

HACKYLE

首页

文章分类

文章标签

关于

留言

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	Key	Value
<input type="checkbox"/>	name	啊啊啊啊
<input type="checkbox"/>	age	10
<input checked="" type="checkbox"/>	email	aaa
	Key	Value

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

"state": false,

"code": 9999,

"message": "处理单个参数校验失败抛出的异常",

"data": null

HACKYLE

POST

http://localhost:7878/validated/param

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	Key	Value
<input type="checkbox"/>	name	啊啊啊啊
<input type="checkbox"/>	age	10
<input checked="" type="checkbox"/>	email	aaa@aa.com
	Key	Value

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Pretty

Raw

Preview

Visualize

Text

1

通过校验

aaa@aa.com

HACKYLE

数组与嵌套

```
1 //数组与嵌套
2 @PostMapping("/listNest")
3 public String listNest(@Validated @RequestBody ApiRequest<List<PersonAddDto>> apiRequest) {
4     return "通过校验" + " " + JSON.toJSONString(apiRequest);
5 }
```

HACKYLE

首页

文章分类

文章标签

关于

留言

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

● JSON

1

{

2

"data": [

3

{

4

"name": "",

5

"age": 999,

6

"gender": 3,

7

"emial": "fff",

8

"birthday": "2023-12-12 12:12:12"

9

},

10

{

11

"name": "",

12

"age": 999,

13

"gender": 3,

14

"emial": "fff",

15

"birthday": "2023-12-12 12:12:12"

16

}

17

]

18

}

Body

Cookies

Headers (5)

Test Results

⌕

Status: 200 OK

Time: 16 ms

Size: 251 B

Pretty

Raw

Preview

Visualize

JSON

↕

1

{

2

"state": false,

3

"code": 9999,

4

"message": "校验RequestBody的实体不通过",

5

"data": null

6

}

HACKYLE

POST

⌵

http://localhost:7878/validated/listNest

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

● JSON

1

{

2

"data": [

3

{

4

"name": "adfad",

5

"age": 66,

6

"gender": 2,

7

"emial": "fff@aa.com",

8

"birthday": "2022-12-12 12:12:12"

9

},

10

{

11

"name": "adfad",

12

"age": 66,

13

"gender": 2,

14

"emial": "fff@aa.com",

15

"birthday": "2022-12-12 12:12:12"

16

}

17

]

18

}

Body

Cookies

Headers (5)

Test Results

⌕

Status: 200 OK

Time: 289 ms

Size: 328 B

📄

Pretty

Raw

Preview

Visualize

Text

↕

1

通过校验 { "data": [{ "age": 66, "birthday": "2022-12-12 20:12:12", "gender": 2, "name": "adfad" }, { "age": 66,

2

"birthday": "2022-12-12 20:12:12", "gender": 2, "name": "adfad" }] }

HACKYLE

分组校验

背景:

- 对于同一实体，在不同场景下需要校验的参数也是不同的。
- 例如，在增加操作的时候，需要校验username、password；在删除操作的时候，需要校验ID；在修改操作的时候，也需要校验ID和某些字段；在查询操作的时候需要校验ID。

语法:

- 创建分组校验
- 在实体类上加各个分组标识
- 在Controller层的Validated注解中也加入分组标识

定义分组标识

```
1 | import javax.validation.groups.Default;
2 |
3 | public interface ValidatedGroup extends Default {
4 |
5 | }
```

```
7      interface Update extends ValidatedGroup {
8      }
9
10     interface Query extends ValidatedGroup {
11     }
12
13     interface Delete extends ValidatedGroup {
14     }
15 }
```

属性使用校验注解

```
1  import com.ks.demo.vv.config.ValidatedGroup;
2  import lombok.Data;
3  import org.hibernate.validator.constraints.Range;
4
5  import javax.validation.constraints.*;
6  import java.math.BigDecimal;
7  import java.util.Date;
8
9  @Data
10 public class PersonAddDto {
11     //使用了group的, 限定只有在该个标记的情况下才会使得当前校验生效
12     //在Controller层使用: @Validated(ValidatedGroup.Update.class) @RequestBody Pers
13     //含义: 限定在更新和删除时, 必须携带ID
14     @NotNull(groups = {ValidatedGroup.Update.class, ValidatedGroup.Delete.class},
15     private Long id;
16
17     @NotBlank(message = "请输入名称")
18     @Size(message = "名称字符长度在{min}到{max}之间", min = 1, max = 10)
19     private String name;
20
21     @NotNull(message = "请输入年龄")
22     @Range(message = "年龄范围为 {min} 到 {max} 之间", min = 1, max = 100)
23     private Integer age;
24 }
```

在Controller层指定开启那些校验

```
import com.alibaba.fastjson2.JSON;
import com.ks.demo.vv.config.ValidatedGroup;
import com.ks.demo.vv.dto.ApiRequest;
import com.ks.demo.vv.dto.PersonAddDto;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

import javax.validation.constraints.Email;
import java.util.List;

@RequestMapping("/validated")
@RestController
public class ValidatedGroupController {
    /**
     * `@Validated` 中不指定任何分组
     */
}
```



```
20     }
21
22     /**
23      * `@Validated`中限定使用'ValidatedGroup.Update.class'分组
24      * PersonAddDto实体类中的所有加了该个分组标识的校验都生效
25      */
26     @PostMapping("/groupUpdate")
27     public String groupUpdate(@Validated(ValidatedGroup.Update.class) @RequestBody
28         return "通过校验" + " " + JSON.toJSONString(apiRequest);
29     }
30
31     /**
32      * `@Validated`中限定'ValidatedGroup.Delete.class'分组
33      * PersonAddDto实体类中的所有加了该个分组标识的校验都生效
34      */
35     @PostMapping("/groupDel")
36     public String groupDel(@Validated(ValidatedGroup.Delete.class) @RequestBody Ap
37         return "通过校验" + " " + JSON.toJSONString(apiRequest);
38     }
39 }
```

POST

http://localhost:7878/validated/groupUpdate

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```
1 {
2   ... "data": {}
3   ... "id": null,
4   ... "name": "aa",
5   ... "age": 6,
6   ... "gender": 1
7 }
```

Body Cookies Headers (5) Test Results

Pretty

Raw

Preview

Visualize

JSON

```
2   "state": false,
3   "code": 9999,
4   "message": "校验RequestBody的实体不通过",
5   "data": null
6 }
```

2023-08-15 09:57:42.755 WARN 12040 --- [nio-7878-exec-7] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [org.springframework.web.bind

.MethodArgumentNotValidException: Validation failed for argument [0] in public java.lang.String com.ks.demo.vv.controller.ValidatedGroupContr

.ks.demo.vv.dto.ApiRequest<com.ks.demo.vv.dto.PersonAddDto>: [Field error in object 'apiRequest' on field 'data.id': rejected value [null];

.apiRequest.data.id,NotNull.data.id,NotNull.id,NotNull.java.lang.Long,NotNull]; arguments [org.springframework.context.support.DefaultMessage

codes [apiRequest.data.id,data.id]; arguments []; default message [data.id]]; default message [更新操作时不允许id为空]]]

版权声明：非明确标注皆为原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上本文链接及此声明。
原文链接：<https://blog.hackyle.com/article/java-demo/valid-validated-demo>

留下你的评论

Link: Input your phone or website, please

谢谢，解决了我的问题

在SpringBootv2.3之前的版本只需要引入 web 依赖就可以了，他包含了validation校验包而在此之后SpringBoot版本就独立出来了需要单独引入依赖

回复

Designed and Created by HACKYLE SHAWE

备案号：浙ICP备20001706号-2