

ModelMapper & Exception & Validation

ModelMapper & Exception & Validation

Repo

ModelMapper

Post - Comments fixed

Exception/Error Handling(重点, easy)

Questions

Anotations

Steps

Handle Specific Exception

Handle Global Exception

Validation

Dependency

Steps

Customizing Validation Response

Code

Check diff scenarios using Postman

Validation for Comment feature

Conclusion(非常重要)

Acutator(工具, 知道就行)

Endpoints

Reading

Swagger(工具, 知道就行)

Repo

Repo: <https://github.com/TAIsRich/springboot-redbook.git>

Branch:

- 06_mapper-exception
- 07_01_validation
- 07_02_actuator

ModelMapper

<https://mvnrepository.com/artifact/org.modelmapper/modelmapper/2.4.5>

```
1      <dependency>
2          <groupId>org.modelmapper</groupId>
3          <artifactId>modelmapper</artifactId>
4          <version>2.4.5</version>
5      </dependency>
```

把一个对象（Comment）map成另外一个对象(CommentDto)，反之亦然。

```
1  modelMapper.map(comment, CommentDto.class)
```

本质就是 `comment.setBody(commentDto.getBody())`

@Bean通常用于第三方的包，因为只有自己的source code才能添加@Component, @Service, @Controller, @Repository

```
1  @Bean
2  public ModelMapper modelMapper() {
3      return new ModelMapper();
4  }
```

Post - Comments fixed

获取一个post时候应该获取它的所有comment

- com.chuwa.redbook.payload.PostDto

- `private Set<CommentDto> comments;`
- Related getter setter;

- com.chuwa.redbook.entity

```
1 @OneToMany(mappedBy = "post", cascade =  
    CascadeType.ALL, orphanRemoval = true)  
2     private Set<Comment> comments = new HashSet<>();
```

- Related getter setter;

The screenshot shows a REST client interface with a GET request to `{{host}}/api/v1/posts/11`. The Headers tab is active, showing standard headers like Cache-Control, Postman-Token, Host, User-Agent, and Accept. The Body tab is also active, showing a JSON response:

```
1 {  
2   "id": 11,  
3   "title": "my first",  
4   "description": "my first rest api",  
5   "content": "my first rest api is HTTP POST method, I have learned the requestbody, controller, service, dao, entity, database. and DTO"  
6 }
```

Below the JSON response, the text **no comments** is displayed in red, indicating that the response does not contain any comments.

```
GET {{host}}/api/posts/9

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Body Cookies Headers (5) Test Results Status: 200 OK Time: 269 ms Size: 1.37 KB Save Response

Pretty Raw Preview Visualize JSON

1  {
2    "id": 9,
3    "title": "Curry beat Kobe",
4    "description": "yes or not",
5    "content": "Curry has beaten Kobe and will be the Top 10 Rank in NBA history",
6    "comments": [
7      {
8        "id": 5,
9        "name": "Curry's fan-3",
10       "email": "fans@gmail.com",
11       "body": "Yes, Curry has been definitely beaten Kobe, he should be Rank in top 10, and Lebron should be rank 9"
12     },
13     {
14       "id": 2,
15       "name": "Kobe's fan",
16       "email": "Kobe.fans@gmail.com",
17       "body": "SB, one more champion, more selected to Team 1, more selected to defens team 1. Kobe has Oreal, Curry has Durant"
18     },
19     {
20       "id": 6,
21       "name": "Curry's fan-4",
22       "email": "fans@gmail.com",
23       "body": "Yes, Curry has been definitely beaten Kobe, he should be Rank in top 10, and Lebron should be rank 9"
24     },
25     {
26       "id": 4,
27       "name": "Curry's fan-2",
28       "email": "fans@gmail.com",
29       "body": "Yes, Curry has been definitely beaten Kobe, he should be Rank in top 10, and Lebron should be rank 9"
30     },
31     {
32       "id": 1,
33       "name": "Curry's fan",
34       "email": "fans@gmail.com",
```

Excelption/Error Handling(重点, easy)

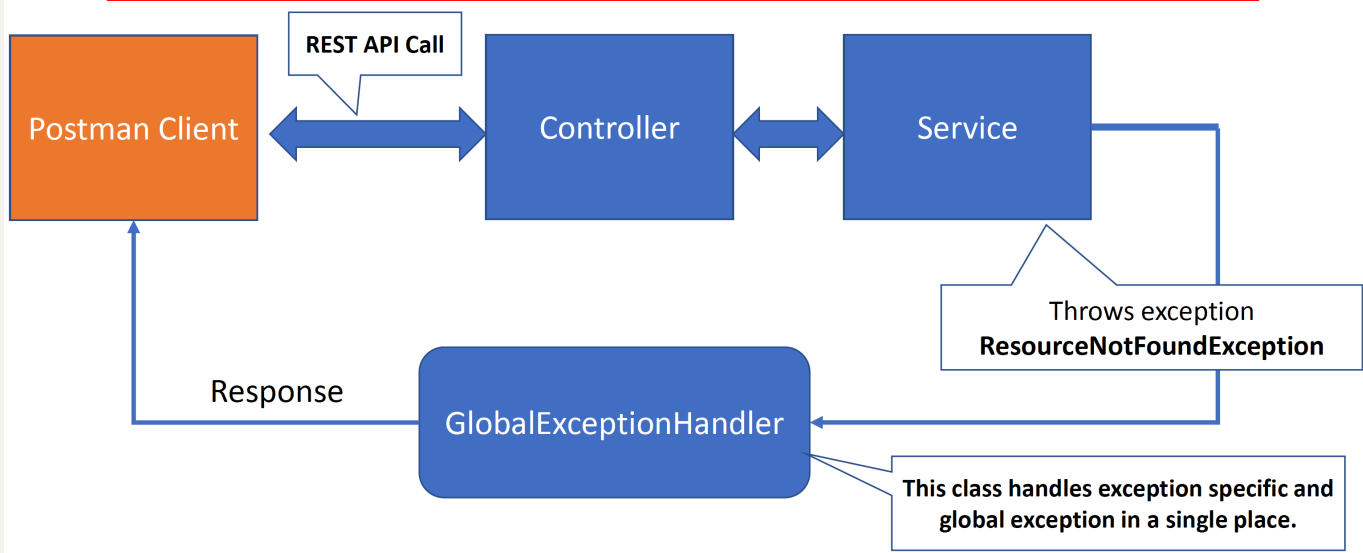
Questions

- How do you do the Error/Exception Handling?

```
1 public void IOABC() throws CustomizedException {
2     String a = "1";
3     throw CustomizedException();
4 }
5
6 public void main (String[] args) throws Exception {
7     // try {
8         IOABC();
9     // } catch (Exception e) {
10        e.printStackTrace();
11    }
12
13 }
```

-

Spring Boot REST API Exception Handling



Anotations

- `@ExceptionHandler`
 - Method Level
 - used to handle the specific exceptions and sending the custom responses to the client
- `@ControllerAdvice`
 - Class Level
 - to handle the exceptions **globally**

Steps

1. create ErrorDetails Class
2. Create GlobalExceptionHandler Class
3. Test using Postman Client

Handle Specific Exception

```
1 @ExceptionHandler(ResourceNotFoundException.class)
2 public ResponseEntity<ErrorDetails>
   handleResourceNotFoundException(ResourceNotFoundException
   exception,
3
4     WebRequest webRequest) {
5     ErrorDetails errorDetails = new ErrorDetails(new Date(),
6     exception.getMessage(),
7     webRequest.getDescription(false));
8
9     return new ResponseEntity<>(errorDetails,
10    HttpStatus.NOT_FOUND);
11 }
12
```

Chuwa / xiaohongshu / postsById

GET `{{host}}/api/posts/9999` Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Body Cookies Headers (5) Test Results Status: 404 Not Found Time: 198 ms Size: 5.44 KB Save Response

Pretty Raw Preview Visualize JSON Copy

```
1 {
2   "timestamp": "2022-06-26T16:54:21.801+00:00",
3   "status": 404,
4   "error": "Not Found",
5   "trace": "com.chuwa.blog.exception.ResourceNotFoundException: Post not found with id : '9999'\n\tat com.chuwa.blog.service.impl.PostServiceImpl.lambda$getPostById$2(PostServiceImpl.java:110)\n\tat java.base/java.util.Optional.orElseThrow(Optional.java:401)\n\tat com.chuwa.blog.service.impl.PostServiceImpl.getPostById(PostServiceImpl.java:118)\n\tat com.chuwa.blog.controller.PostController.getPostById(PostController.java:47)\n\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(Native Method)\n\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:64)\n\tat java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)\n\tat java.base/java.lang.reflect.Method.invoke(Method.java:564)\n\tat org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:205)\n\tat org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:150)\n\tat org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:117)\n\tat org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:895)\n\tat org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:808)\n\tat org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerMethodAdapter.java:87)\n\tat org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1067)\n\tat org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:963)\n\tat org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1006)\n\tat org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:898)\n\tat javax.servlet.http.HttpServlet.service(HttpServlet.java:655)\n\tat org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:883)\n\tat javax.servlet.http.HttpServlet.service(HttpServlet.java:764)\n\tat org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:227)\n\tat org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:162)\n\tat org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)\n\tat org.apache.catalina.core.
```

Chuwa / xiaohongshu / postsById

GET `{{host}}/api/posts/9999` Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Body Cookies Headers (5) Test Results Status: 404 Not Found Time: 227 ms Size: 292 B Save Response

Pretty Raw Preview Visualize JSON Copy

```
1 {
2   "timestamp": "2022-06-26T16:52:43.793+00:00",
3   "message": "Post not found with id : '9999'",
4   "details": "uri=/api/posts/9999"
5 }
```

```

1 {
2     "timestamp": "2022-06-26T16:52:43.793+00:00",
3     "message": "Post not found with id : '9999'",
4     "details": "uri=/api/posts/9999"
5 }

```

Handle Global Exception

▼ / xiaohongshu / exceptions / postsByid_StringId

GET (host)/api/posts/9

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

Status: 400 Bad Request Time: 27 ms Size: 6.52 KB

```

1 {
2     "timestamp": "2022-06-26T17:02:35.716+00:00",
3     "status": 400,
4     "error": "Bad Request",
5     "trace": "org.springframework.web.method.annotation.MethodArgumentTypeMismatchException: Failed to convert value of type 'java.lang.String' to required type 'long'; nested exception is java.lang.NumberFormatException: For input string: '\\\"9\\\"'\\n\\tat org.springframework.web.method.annotation.AbstractNamedValueMethodArgumentResolver.resolveArgument(AbstractNamedValueMethodArgumentResolver.java:133)\\n\\tat org.springframework.web.method.support.HandlerMethodArgumentResolverComposite.resolveArgument(HandlerMethodArgumentResolverComposite.java:122)\\n\\tat org.springframework.web.method.support.InvocableHandlerMethod.getMethodArgumentValues(InvocableHandlerMethod.java:179)\\n\\tat org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:146)\\n\\tat org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:895)\\n\\tat org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:808)\\n\\tat org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerMethodAdapter.java:87)\\n\\tat org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1067)\\n\\tat org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:963)\\n\\tat org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1006)\\n\\tat org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:898)\\n\\tat javax.servlet.http.HttpServlet.service(HttpServlet.java:655)\\n\\tat org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:883)\\n\\tat javax.servlet.http

```

▼ / xiaohongshu / exceptions / postsByid_StringId

GET (host)/api/posts/9

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

Status: 500 Internal Server Error Time: 159 ms Size: 403 B

```

1 {
2     "timestamp": "2022-06-26T17:05:38.673+00:00",
3     "message": "Failed to convert value of type 'java.lang.String' to required type 'long'; nested exception is java.lang.NumberFormatException: For input string: '\\\"9\\\"'",
4     "details": "uri=/api/posts/%229%22"
5 }

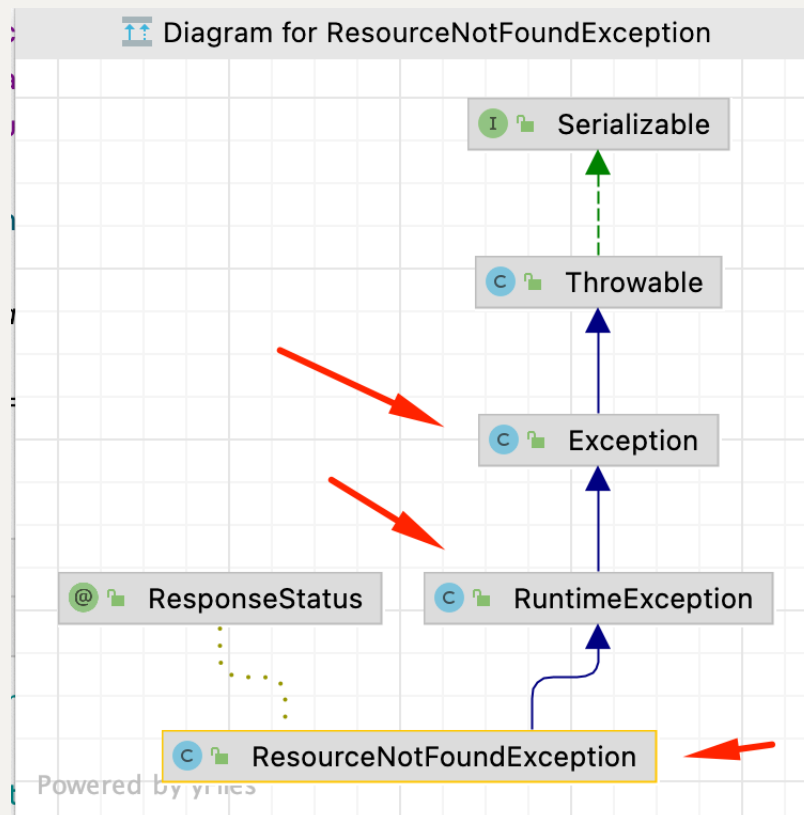
```

Exception is the root of all exceptions, so we use it to handle the global exceptions

```

1  @ExceptionHandler(Exception.class)
2  public ResponseEntity<ErrorDetails>
handleGlobalException(Exception exception,
3
4  WebRequest webRequest) {
5      ErrorDetails errorDetails = new ErrorDetails(new Date(),
exception.getMessage(),
6
7      webRequest.getDescription(false));
8
9      return new ResponseEntity<>(errorDetails,
HttpStatus.INTERNAL_SERVER_ERROR);
10 }

```



Validation

验证request Body，并定制化返回一些信息。

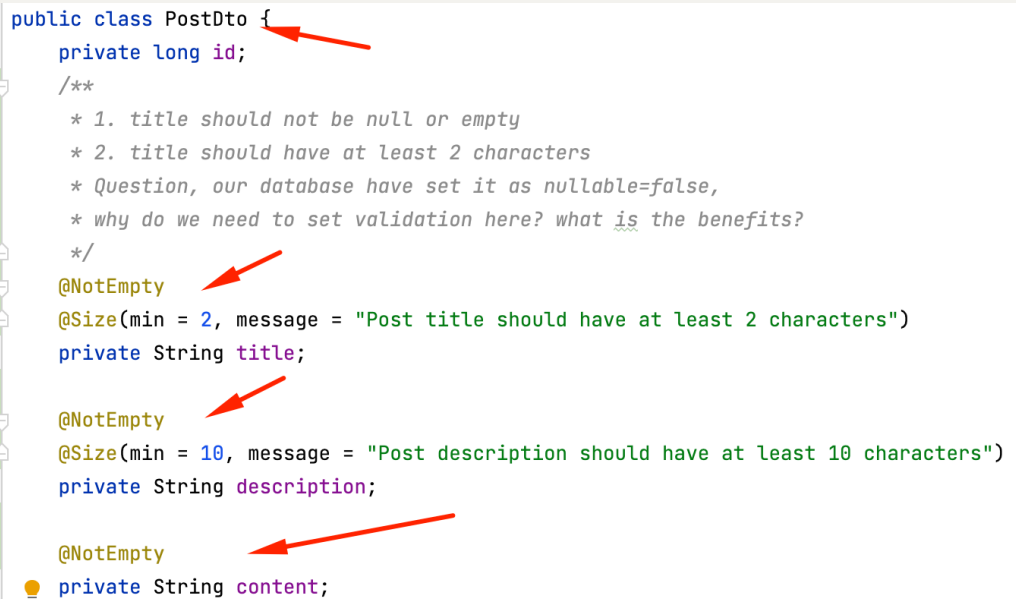
Dependency

```
1 <dependency>
2     <groupId>org.springframework.boot</groupId>
3     <artifactId>spring-boot-starter-validation</artifactId>
4     <version>2.7.0</version>
5 </dependency>
```

Steps

1. import dependency
2. Add validation **Rule** to payload

a.



```
public class PostDto {
    private long id;

    /**
     * 1. title should not be null or empty
     * 2. title should have at least 2 characters
     * Question, our database have set it as nullable=false,
     * why do we need to set validation here? what is the benefits?
     */
    @NotEmpty
    @Size(min = 2, message = "Post title should have at least 2 characters")
    private String title;

    @NotEmpty
    @Size(min = 10, message = "Post description should have at least 10 characters")
    private String description;

    @NotEmpty
    private String content;
```

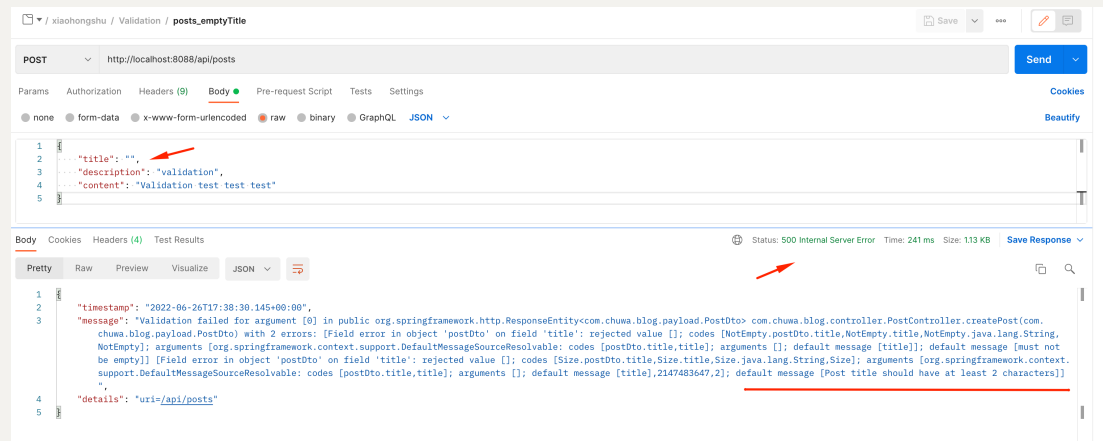
3. Add `@valid` to controller to **apply the Rule** here

a.

```
// create_blog_post
@PostMapping
public ResponseEntity<PostDto> createPost(@Valid @RequestBody PostDto postDto) {
    return new ResponseEntity<>(postService.createPost(postDto), HttpStatus.CREATED);
}
```

4. Check it in Postman

a.



b. Status should not be 500, it should be **Bad_Request**

Customizing Validation Response

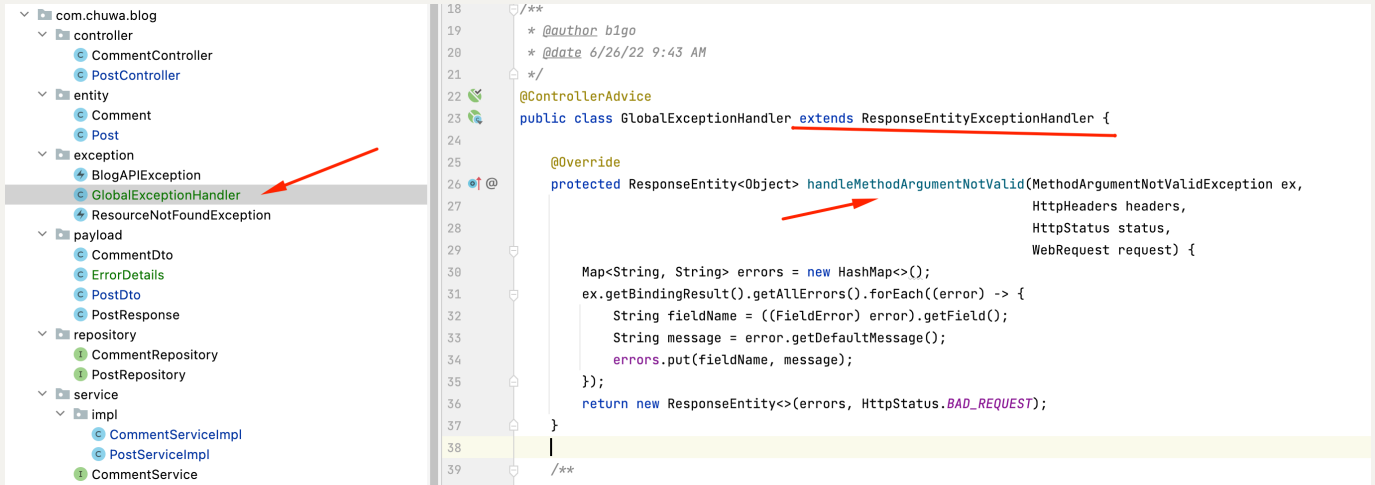
Code

```
1  /**
2   * Approach 1
3   */
4  @ControllerAdvice
5  public class GlobalExceptionHandler extends
6      ResponseEntityExceptionHandler {
7      @Override
8      protected ResponseEntity<Object>
9      handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
10
11      HttpHeaders headers,
12
13      HttpStatus status,
```

```

10     WebRequest request) {
11         Map<String, String> errors = new HashMap<>();
12         ex.getBindingResult().getAllErrors().forEach((error) -> {
13             String fieldName = ((FieldError) error).getField();
14             String message = error.getDefaultMessage();
15             errors.put(fieldName, message);
16         });
17         return new ResponseEntity<>(errors,
18     HttpStatus.BAD_REQUEST);
19     }
20
21     /**
22      * Approach 2
23      */
24     @ControllerAdvice
25     public class GlobalExceptionHandler {
26         @ExceptionHandler(MethodArgumentNotValidException.class)
27         protected ResponseEntity<Object>
28     handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
29
30     HttpHeaders headers,
31
32     HttpStatus status,
33
34     WebRequest request) {
35         Map<String, String> errors = new HashMap<>();
36         ex.getBindingResult().getAllErrors().forEach((error) -> {
37             String fieldName = ((FieldError) error).getField();
38             String message = error.getDefaultMessage();
39             errors.put(fieldName, message);
40         });
41         return new ResponseEntity<>(errors,
42     HttpStatus.BAD_REQUEST);
43     }

```

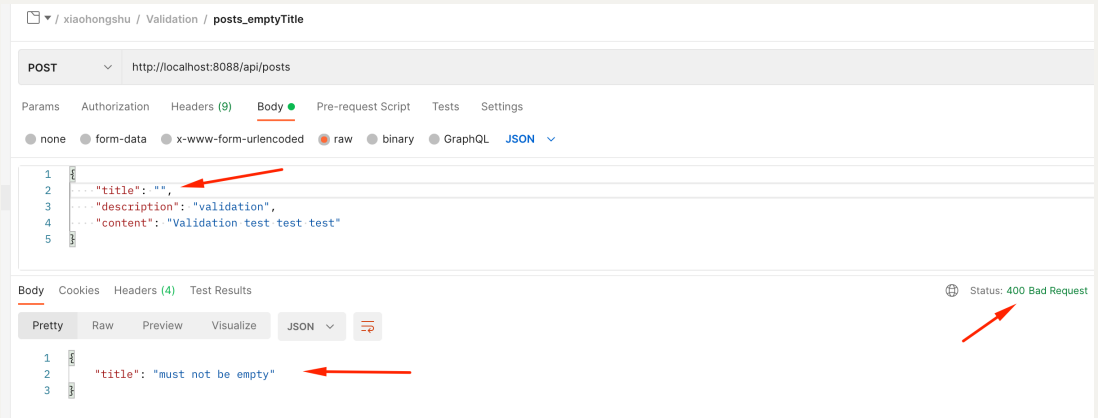


Check diff scenarios using Postman

it will respond with invalid attributes

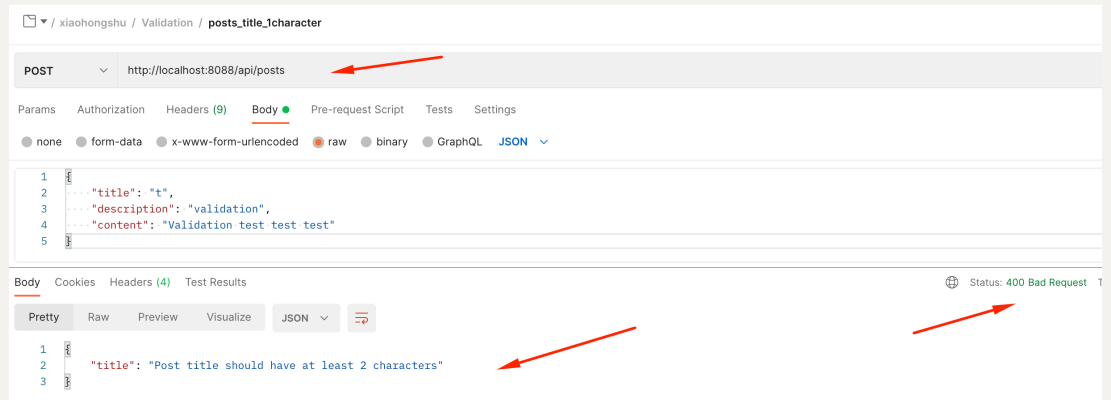
- empty title

•



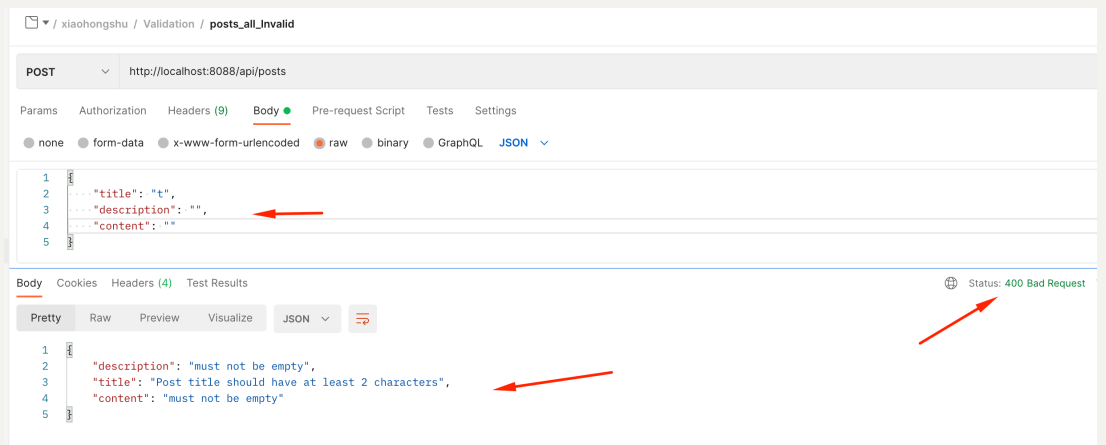
- short title

•



- short for all attributes

•



Validation for Comment feature

```


1     private long id;
2     @NotEmpty(message = "Name should not be null or empty")
3     private String name;
4
5     @NotEmpty(message = "Email should not be null or empty")
6     @Email
7     private String email;
8     @NotEmpty
9     @Size(min = 5, max = 144, message = "Comment body must be
    minimum 5 characters")
10    private String body;

```

```

1      @PostMapping("/posts/{postId}/comments")
2      public ResponseEntity<CommentDto>
3      createComment(@PathVariable(value = "postId") long id,
4                      @Valid
5                      @RequestBody CommentDto commentDto) {
6
7          return new ResponseEntity<>
8              (commentService.createComment(id, commentDto),
9              HttpStatus.CREATED);
10     }
11
12     @PutMapping("/posts/{postId}/comments/{id}")
13     public ResponseEntity<CommentDto>
14     updateComment(@PathVariable(value = "postId") Long postId,
15                  @PathVariable(value = "id") Long commentId,
16                  @Valid
17                  @RequestBody CommentDto commentDto) {
18
19         CommentDto updateComment =
20             commentService.updateComment(postId, commentId, commentDto);
21
22         return new ResponseEntity<>(updateComment, HttpStatus.OK);
23     }

```

POST ⌵ {{host}}/api/posts/1/comments 


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


● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** ⌵


```

1  {
2  ... "name": "",
3  ... "email": "fp.com",
4  ... "body": "你好看"
5  }

```




Body Cookies Headers (4) Test Results ⊕ Status: 400 Bad Request 

Pretty Raw Preview Visualize **JSON** ⌵ 

```

1  {
2  "name": "Name should not be null or empty",
3  "body": "Comment body must be minimum 5 characters",
4  "email": "must be a well-formed email address"
5  }

```



Conclusion(非常重要)

- payload -> Rule
- @Valid -> where apply Rule, if, invalid, throw exception
- global exception -> accept and handle

Acuator(工具，知道就行)

Monitor the applicaiton

- pom.xml

```
1 <dependency>
2     <groupId>org.springframework.boot</groupId>
3     <artifactId>spring-boot-starter-
  actuator</artifactId>
4 </dependency>
```

- applicaiton.properties

```
1 # acuator, JMX
2 management.endpoints.web.exposure.include=*
```

<http://localhost:8080/actuator/health>

GET ▼ `{{host}}/actuator/health`

Params Authorization Headers (7) Body Pre-request Script

Query Params

KEY
Key

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ▼

```
1 {  
2   "status": "UP"  
3 }
```

GET ▼ `{{host}}/actuator/beans`

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Body Cookies Headers (5) Test Results ⊞ Status: 200 OK Time: 535 ms

Pretty Raw Preview Visualize JSON ▼

```
1 {  
2   "contexts": {  
3     "application": {  
4       "beans": {  
5         "spring.jpa-org.springframework.boot.autoconfigure.orm.jpa.JpaProperties": {  
6           "aliases": [],  
7           "scope": "singleton",  
8           "type": "org.springframework.boot.autoconfigure.orm.jpa.JpaProperties",  
9           "resource": null,  
10          "dependencies": []  
11        },  
12        "endpointCachingOperationInvokerAdvisor": {  
13          "aliases": [],  
14          "scope": "singleton",  
15          "type": "org.springframework.boot.actuate.endpoint.invoker.cache.CachingOperationInvokerAdvisor",  
16          "resource": "class path resource [org.springframework.boot/actuate/autoconfigure/endpoint/EndpointAutoConfiguration.class]",  
17          "dependencies": [  
18            "org.springframework.boot.actuate.autoconfigure.endpoint.EndpointAutoConfiguration",  
19            "environment"  
20          ]  
21        },  
22        "defaultServletHandlerMapping": {  
23          "aliases": [],  
24          "scope": "singleton",  
25          "type": "org.springframework.web.servlet.HandlerMapping",  
26          "resource": "class path resource [org.springframework.boot/actuate/autoconfigure/web/servlet/WebMvcAutoConfiguration$EnableWebMvcConfiguration.class]",  
27          "dependencies": [  
28            "org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoConfiguration$EnableWebMvcConfiguration"  
29          ]  
30        },  
31        "globalExceptionHandler": {  
32          "aliases": [],  
33          "scope": "singleton",  
34          "type": "com.chuwa.redbook.exception.GlobalExceptionHandler",  
35          "resource": "file [/Users/bigo/Documents/githubProjects/chuwa/redbook/target/classes/com/chuwa/redbook/exception/GlobalExceptionHandler.class]",  
36          "dependencies": []  
37        },  
38        "metricsRestTemplateCustomizer": {
```


Endpoints

```
1  {
2    "_links": {
3      "self": {
4        "href": "http://localhost:8080/actuator",
5        "templated": false
6      },
7      "beans": {
8        "href": "http://localhost:8080/actuator/beans",
9        "templated": false
10     },
11     "caches-cache": {
12       "href":
13       "http://localhost:8080/actuator/caches/{cache}",
14       "templated": true
15     },
16     "caches": {
17       "href": "http://localhost:8080/actuator/caches",
18       "templated": false
19     },
20     "health": {
21       "href": "http://localhost:8080/actuator/health",
22       "templated": false
23     },
24     "health-path": {
25       "href":
26       "http://localhost:8080/actuator/health/{*path}",
27       "templated": true
28     },
29     "info": {
30       "href": "http://localhost:8080/actuator/info",
31       "templated": false
32     },
33     "conditions": {
34       "href": "http://localhost:8080/actuator/conditions",
```

```
33         "templated": false
34     },
35     "configprops-prefix": {
36         "href":
37         "http://localhost:8080/actuator/configprops/{prefix}",
38         "templated": true
39     },
40     "configprops": {
41         "href": "http://localhost:8080/actuator/configprops",
42         "templated": false
43     },
44     "env": {
45         "href": "http://localhost:8080/actuator/env",
46         "templated": false
47     },
48     "env-toMatch": {
49         "href":
50         "http://localhost:8080/actuator/env/{toMatch}",
51         "templated": true
52     },
53     "loggers": {
54         "href": "http://localhost:8080/actuator/loggers",
55         "templated": false
56     },
57     "loggers-name": {
58         "href":
59         "http://localhost:8080/actuator/loggers/{name}",
60         "templated": true
61     },
62     "heapdump": {
63         "href": "http://localhost:8080/actuator/heapdump",
64         "templated": false
65     },
66     "threaddump": {
67         "href": "http://localhost:8080/actuator/threaddump",
68         "templated": false
69     }
70 }
```

```

66     },
67     "metrics": {
68         "href": "http://localhost:8080/actuator/metrics",
69         "templated": false
70     },
71     "metrics-requiredMetricName": {
72         "href":
73         "http://localhost:8080/actuator/metrics/{requiredMetricName}",
74         "templated": true
75     },
76     "scheduledtasks": {
77         "href":
78         "http://localhost:8080/actuator/scheduledtasks",
79         "templated": false
80     },
81     "mappings": {
82         "href": "http://localhost:8080/actuator/mappings",
83         "templated": false
84     }
85 }

```

Reading

按照顺序读

- Basic understanding: <https://www.liaoxuefeng.com/wiki/1252599548343744/1282386381766689>
- more undersstanding: <https://springframework.guru/actuator-in-spring-boot/#:~:text=Spring%20Boot%20Actuator%20is%20a,metrics%20from%20production%2Dready%20applications.>
- JMX: <https://docs.spring.io/spring-boot/docs/2.1.11.RELEASE/reference/html/production-ready-endpoints.html>
- <https://www.javatpoint.com/spring-boot-actuator>

Swagger(工具，知道就行)

https://petstore.swagger.io/?_ga=2.7729466.929353116.1662000236-847606602.1662000236#/pet/addPet