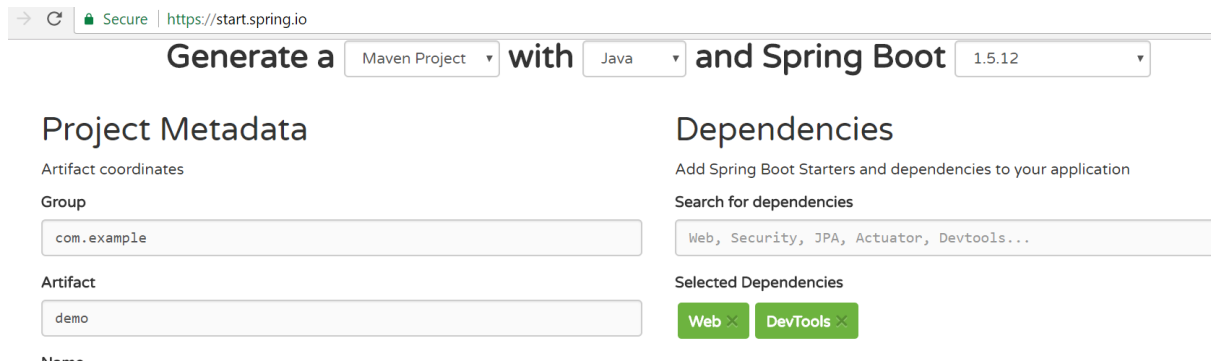


Spring boot with Request Validation

Check this out on github >> <https://github.com/fishyimm/spring-boot-request-validation>

1. Create spring boot project with simple dependencies.

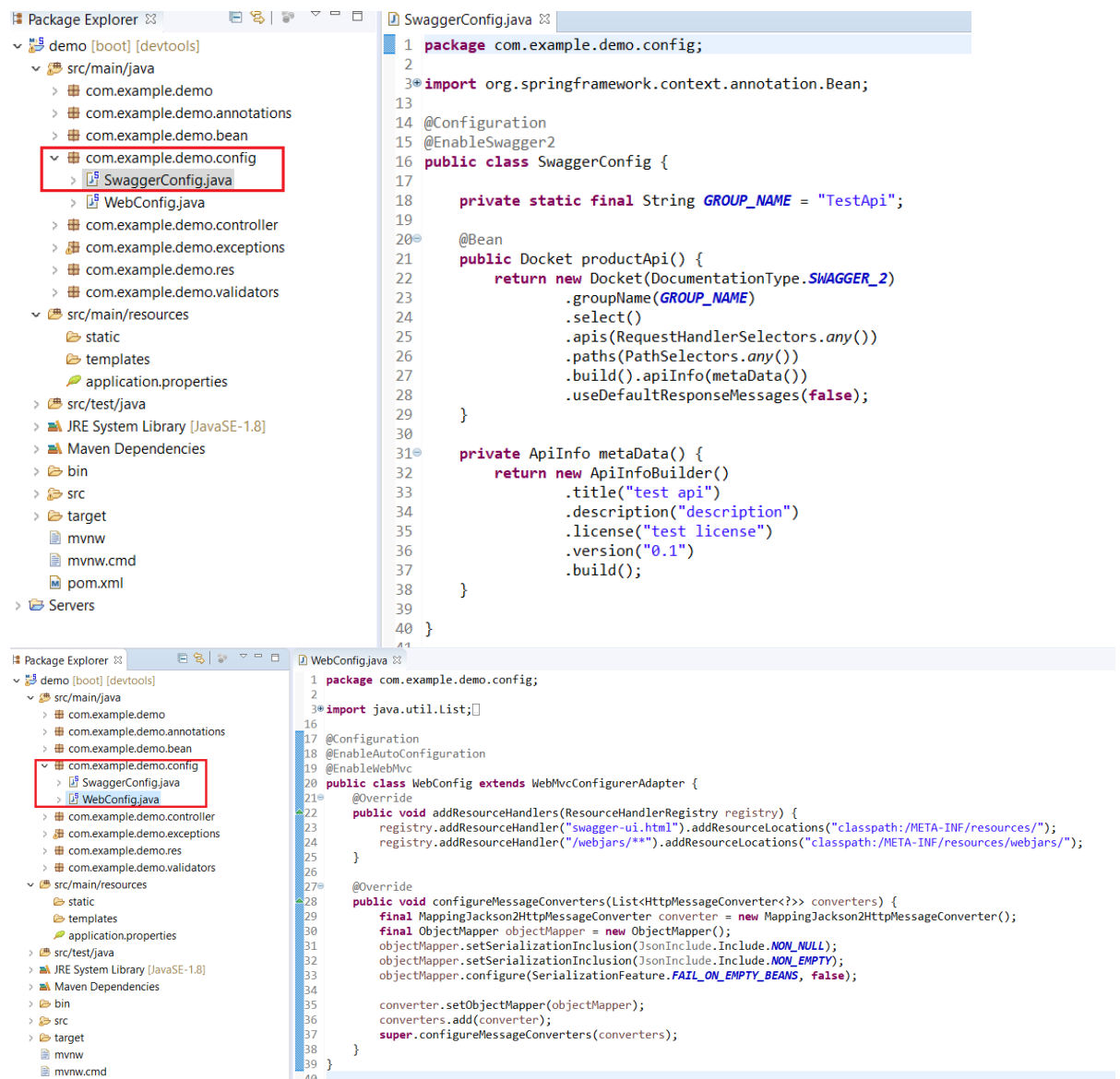
The screenshot shows the Spring Boot Start web application. At the top, there's a navigation bar with a 'Generate a' dropdown set to 'Maven Project', followed by 'with' and a 'Java' dropdown, and 'and Spring Boot' with a version dropdown set to '1.5.12'. Below this, the page is divided into two main sections. The 'Project Metadata' section on the left has a label 'Artifact coordinates' and two input fields: 'Group' with the value 'com.example' and 'Artifact' with the value 'demo'. The 'Dependencies' section on the right has a label 'Add Spring Boot Starters and dependencies to your application' and a search bar with the text 'Search for dependencies'. Below the search bar, it shows 'Selected Dependencies' with two green buttons: 'Web' and 'DevTools', each with a close icon.

2. You can add swagger dependencies to test your project, add this in your pom.xml

```
<!-- swagger -->
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.7.0</version>
</dependency>
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.7.0</version>
</dependency>
```

3. Create simple configuration for swagger as below:

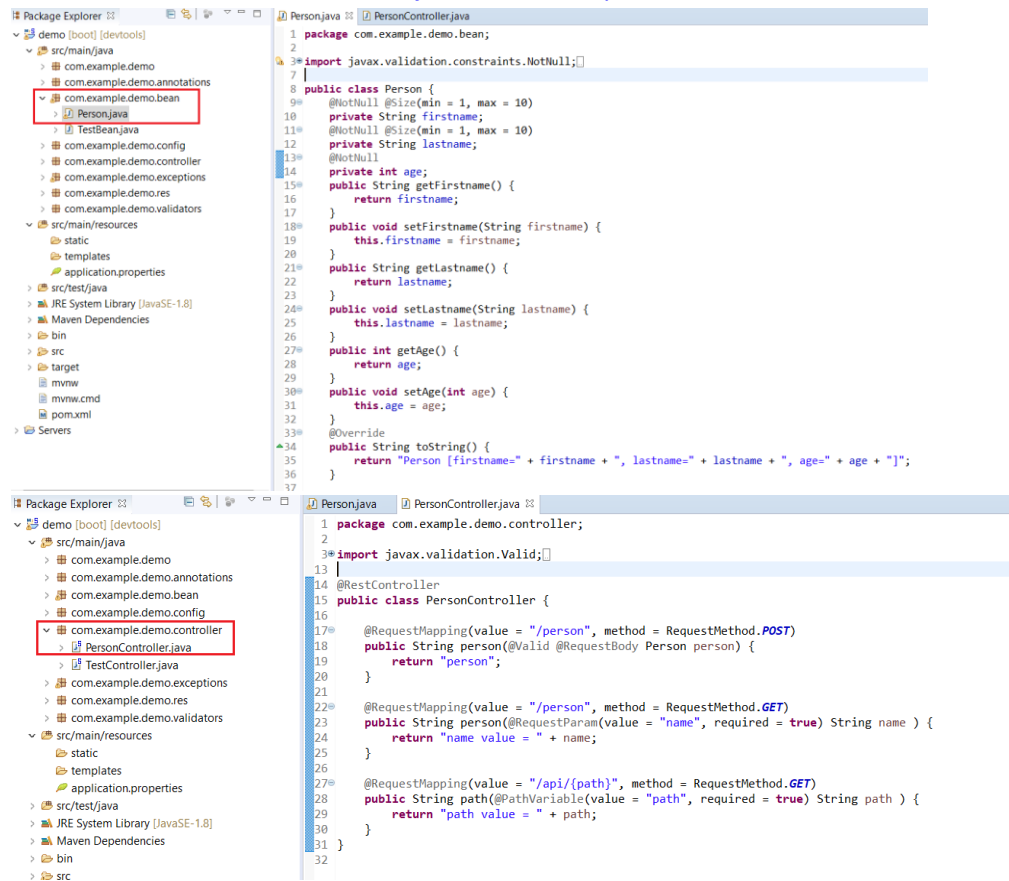
<https://github.com/fishyimm/spring-boot-request-validation/blob/master/src/main/java/com/example/demo/config/>



4. Create simple bean and simple controller for person.

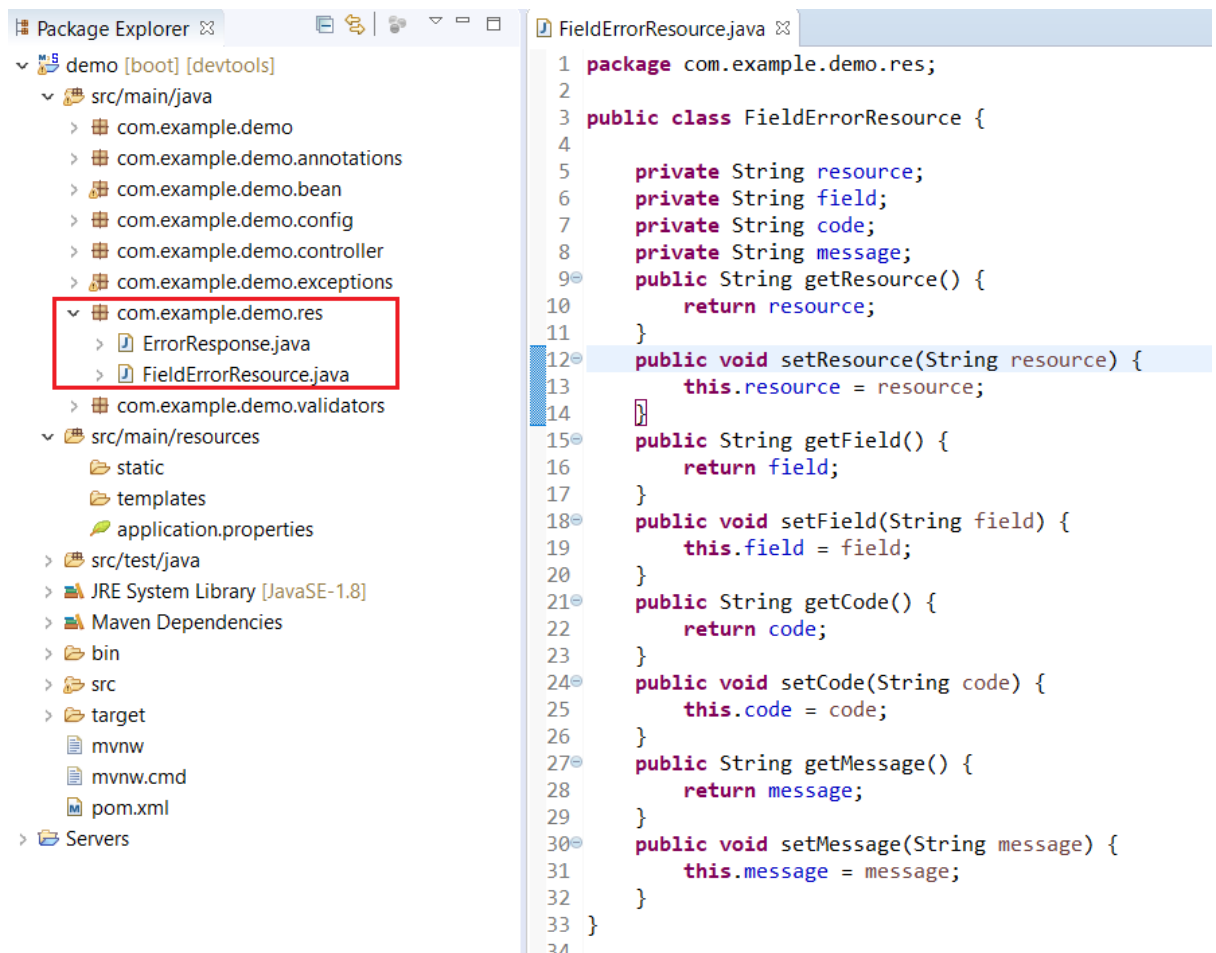
<https://github.com/fishyimm/spring-boot-request-validation/blob/master/src/main/java/com/example/demo/bean/Person.java>

<https://github.com/fishyimm/spring-boot-request-validation/blob/master/src/main/java/com/example/demo/controller/PersonController.java>



5. Create simple response error bean as below:

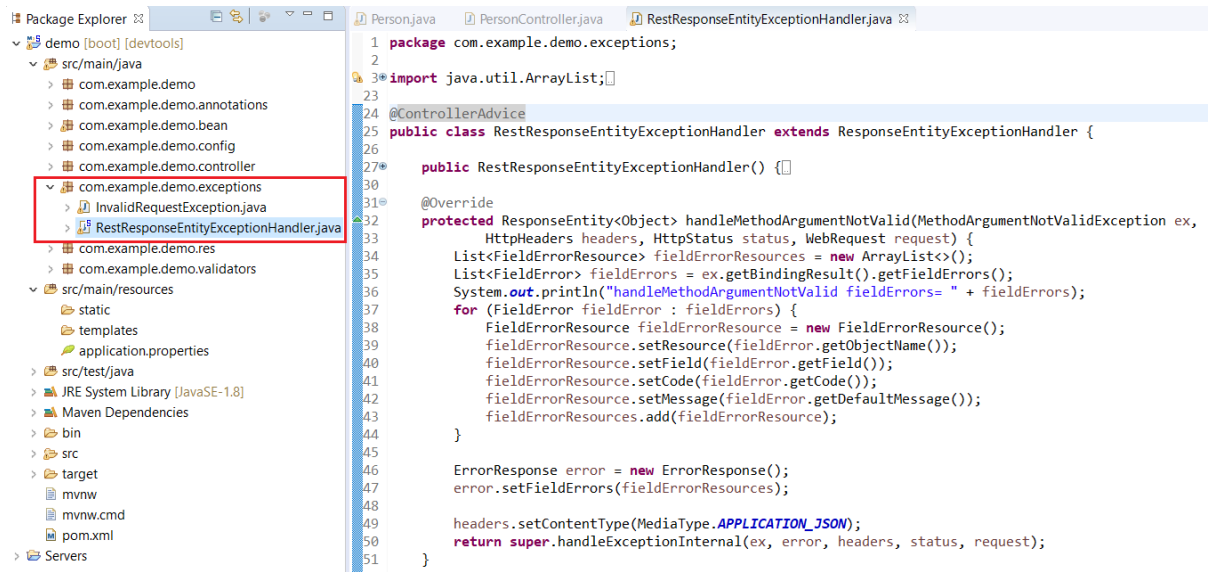
<https://github.com/fishyimm/spring-boot-request-validation/tree/master/src/main/java/com/example/demo/res>



6. Create `RestExceptionHandler` with `@ControllerAdvice` to handle exception that occur in controller as below:

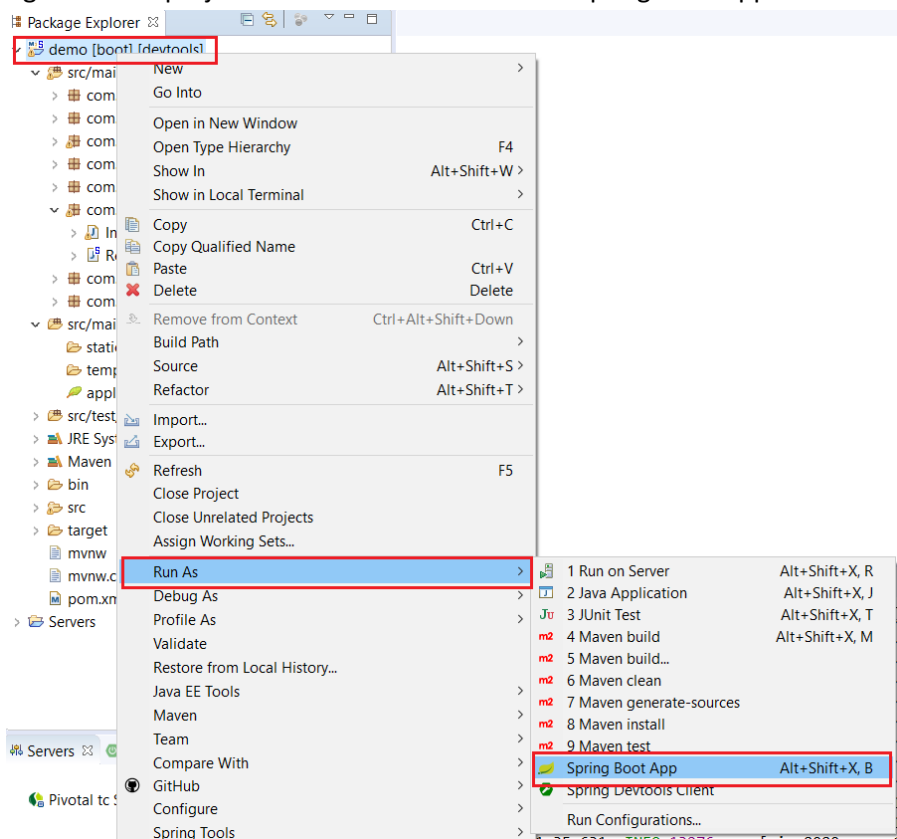
<https://github.com/fishyimm/spring-boot-request-validation/tree/master/src/main/java/com/example/demo/res>

[validation/blob/master/src/main/java/com/example/demo/exceptions/RestResponseEntityExceptionHandler.java](#)



7. Now test it with swagger-ui >> start your project and go to <http://localhost:8080/swagger-ui.html>

right click on project > choose run as >> choose Spring Boot App in STS IDE



localhost:8080/swagger-ui.html#/

POST

/person

Response Class (Status 200)

string

Response Content Type

Parameters

Parameter	Value
person	<pre>{ "age": 0, "firstname": "string", "lastname": "string" }</pre>

Parameter content type:

Try it out!

Hide Response

Request URL

http://localhost:8080/person

Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body

```
"person"
```

Response Code

200

with normal request it won't occur any error but try as below and see the result

POST

/person

Response Class (Status 200)

string

Response Content Type */* ▾

Parameters

Parameter	Value	Description
person	<pre>{ "age": 0, "firstname": "stringstringstring", "lastname": "stringstringstringstring" }</pre>	person

Parameter content type: application/json ▾

Try it out!

Hide Response

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: applica
  "age": 0, \
  "firstname": "stringstringstring", \
  "lastname": "stringstringstringstring" \
}' 'http://localhost:8080/person'
```

Request URL

```
http://localhost:8080/person
```

Response Body

```
{
  "fieldErrors": [
    {
      "resource": "person",
      "field": "firstname",
      "code": "Size",
      "message": "size must be between 1 and 10"
    },
    {
      "resource": "person",
      "field": "lastname",
      "code": "Size",
      "message": "size must be between 1 and 10"
    }
  ]
}
```

Response Code

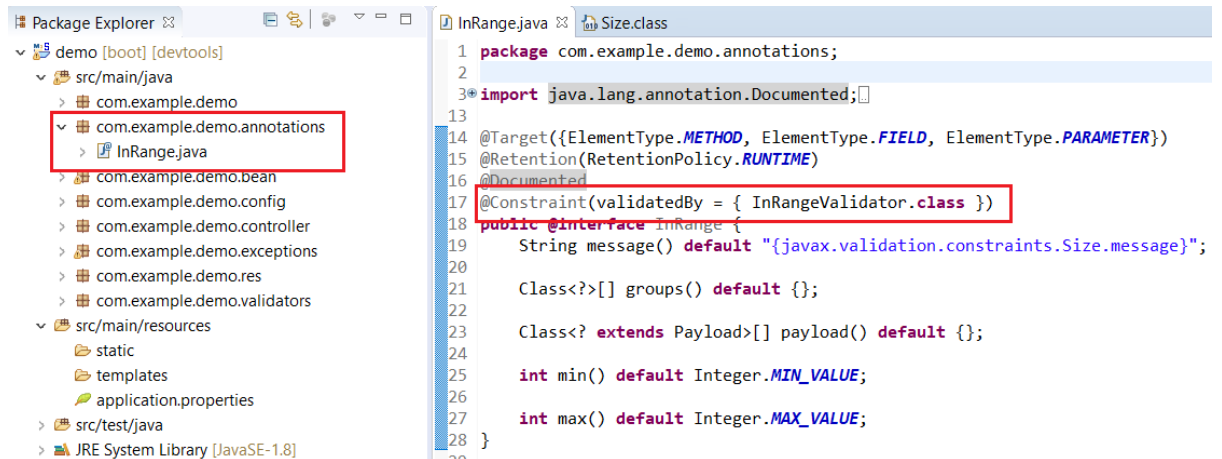
```
400
```

as we set **@Size** and **@NotNull** if your person null or parameter exceed the length error will be occurred but how about **parameter age(int)** how to validate the size of it, since **@Size** cannot validate it so we need to create new validation for **int value**

8. Create new annotation for validation you can check **Size** for example

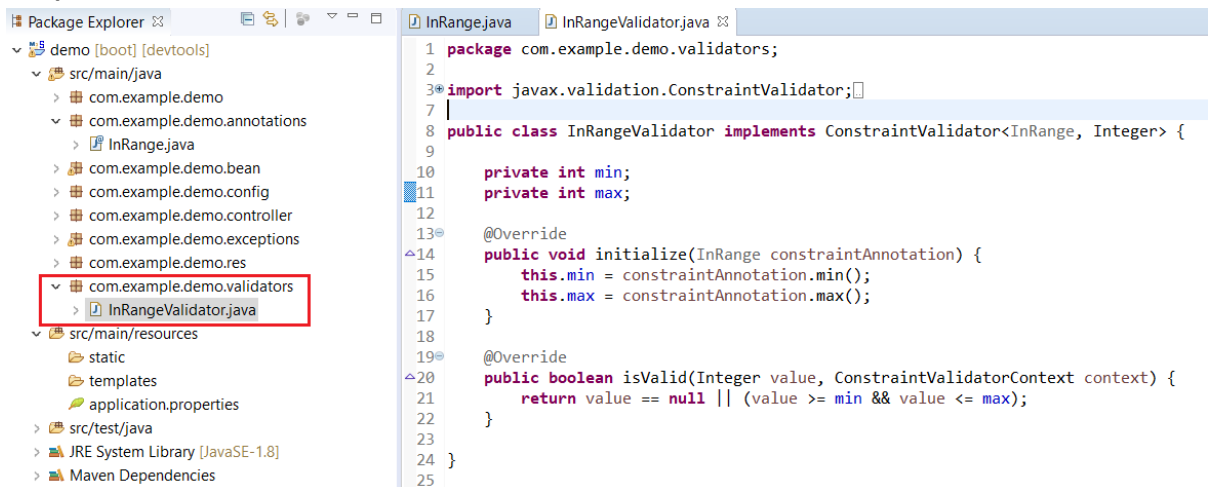
```
InRange.java Size.class
20 * JBoss, Home of Professional Open Source
17 package javax.validation.constraints;
18
19 import java.lang.annotation.Documented;
20 import java.lang.annotation.Retention;
21 import java.lang.annotation.Target;
22 import javax.validation.Constraint;
23 import javax.validation.Payload;
24
25 import static java.lang.annotation.ElementType.ANNOTATION_TYPE;
26 import static java.lang.annotation.ElementType.CONSTRUCTOR;
27 import static java.lang.annotation.ElementType.FIELD;
28 import static java.lang.annotation.ElementType.METHOD;
29 import static java.lang.annotation.ElementType.PARAMETER;
30 import static java.lang.annotation.RetentionPolicy.RUNTIME;
31
32 * The annotated element size must be between the specified boundaries (included).
47 @Target({ METHOD, FIELD, ANNOTATION_TYPE, CONSTRUCTOR, PARAMETER })
48 @Retention(RUNTIME)
49 @Documented
50 @Constraint(validatedBy = { })
51 public @interface Size {
52
53     String message() default "{javax.validation.constraints.Size.message}";
54
55     Class<?>[] groups() default { };
56
57     Class<? extends Payload>[] payload() default { };
58
59     * @return size the element must be higher or equal to
62 int min() default 0;
63
64     * @return size the element must be lower or equal to
67 int max() default Integer.MAX_VALUE;
68
69     * Defines several {@link Size} annotations on the same element.
77 @interface List {
81 }
82
```

and create **inrange** annotation for validate int value what u need to create more is **validatedBy** as picture below:



```
InRange.java Size.class
1 package com.example.demo.annotations;
2
3 import java.lang.annotation.Documented;
13
14 @Target({ElementType.METHOD, ElementType.FIELD, ElementType.PARAMETER})
15 @Retention(RetentionPolicy.RUNTIME)
16 @Documented
17 @Constraint(validatedBy = { InRangeValidator.class })
18 public @interface InRange {
19     String message() default "{javax.validation.constraints.Size.message}";
20
21     Class<?>[] groups() default { };
22
23     Class<? extends Payload>[] payload() default { };
24
25     int min() default Integer.MIN_VALUE;
26
27     int max() default Integer.MAX_VALUE;
28 }
29
```


this validator will validate the value of the parameter size now try to add this annotation in our **person bean**



POST /person

Response Class (Status 200)
string

Response Content Type

Parameters

Parameter	Value	Description
person	<pre>{ "age": 0, "firstname": "string", "lastname": "string" }</pre>	person

Parameter content type:

[Try it out!](#) [Hide Response](#)

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: */*' \
  -d '{
  "age": 0, \
  "firstname": "string", \
  "lastname": "string" \
}' 'http://localhost:8080/person'
```

Request ID:

here we got error already cause of we sending age = 0 and validator give us an error as expected

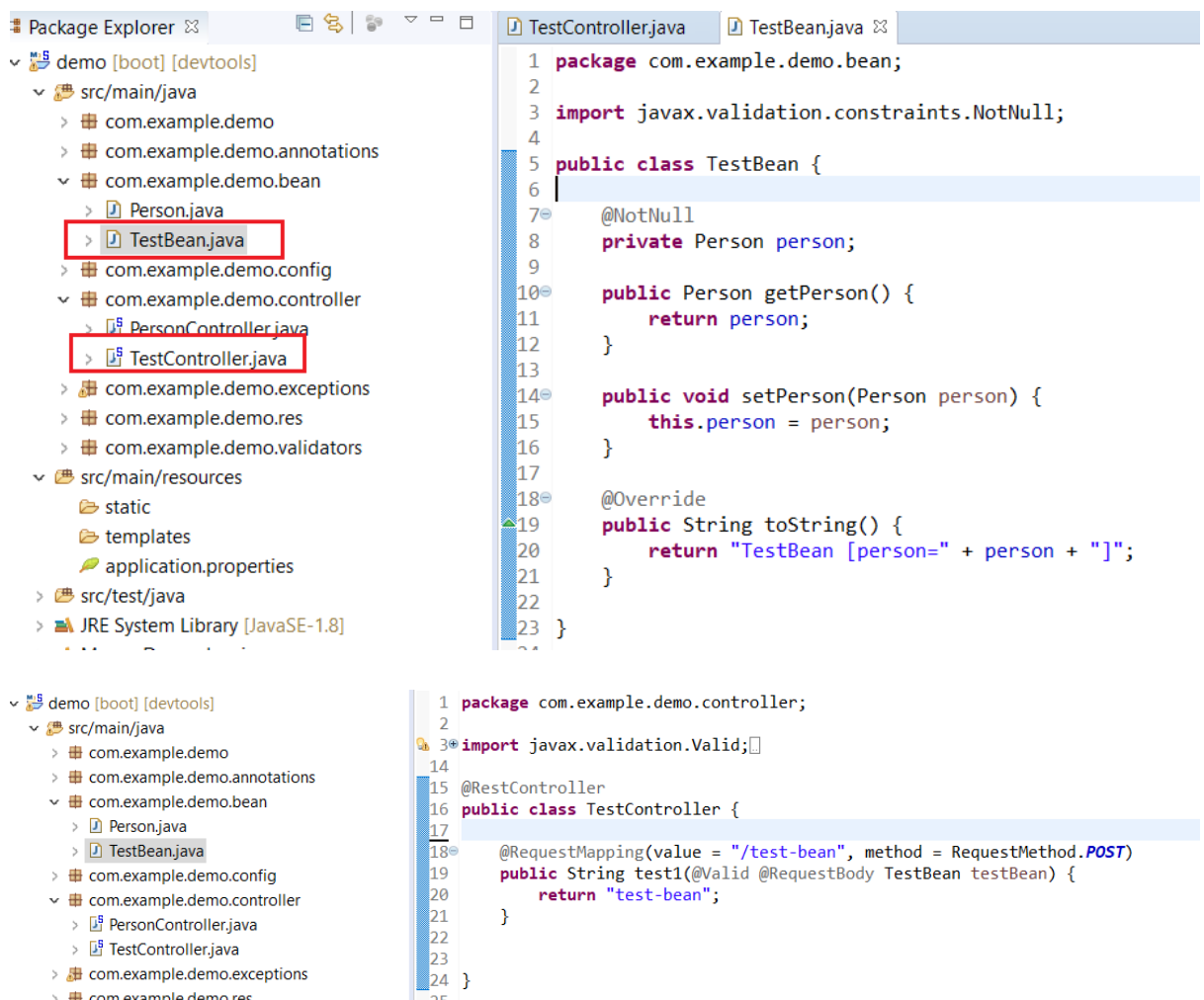
```
Response Body

{
  "fieldErrors": [
    {
      "resource": "person",
      "field": "age",
      "code": "InRange",
      "message": "size must be between 1 and 10"
    }
  ]
}

Response Code

400
```

9. Now we gonna try to validate nested bean, create another simple bean and add person in it as below



```
Package Explorer
demo [boot] [devtools]
├── src/main/java
│   ├── com.example.demo
│   ├── com.example.demo.annotations
│   ├── com.example.demo.bean
│   │   ├── Person.java
│   │   └── TestBean.java
│   ├── com.example.demo.config
│   ├── com.example.demo.controller
│   │   ├── PersonController.java
│   │   └── TestController.java
│   ├── com.example.demo.exceptions
│   ├── com.example.demo.res
│   ├── com.example.demo.validators
│   └── src/main/resources
│       ├── static
│       ├── templates
│       └── application.properties
├── src/test/java
└── JRE System Library [JavaSE-1.8]

TestBean.java
1 package com.example.demo.bean;
2
3 import javax.validation.constraints.NotNull;
4
5 public class TestBean {
6
7     @NotNull
8     private Person person;
9
10    public Person getPerson() {
11        return person;
12    }
13
14    public void setPerson(Person person) {
15        this.person = person;
16    }
17
18    @Override
19    public String toString() {
20        return "TestBean [person=" + person + "]";
21    }
22
23 }
```

```
TestController.java
1 package com.example.demo.controller;
2
3 import javax.validation.Valid;
4
5 @RestController
6 public class TestController {
7
8     @RequestMapping(value = "/test-bean", method = RequestMethod.POST)
9     public String test1(@Valid @RequestBody TestBean testBean) {
10        return "test-bean";
11    }
12
13 }
```

10. Now try it on swagger and get success result but why? Validation does not work?

POST

/test-bean

Response Class (Status 200)

string

Response Content Type

/

Parameters

Parameter	Value	Description
testBean	<div><div>{ "person": { "age": 0, "firstname": "string", "lastname": "string" } }</div><div>Parameter content type: <div>application/json</div></div></div> <div>testBean</div>	

Try it out!

Hide Response

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: ap  
  "person": {  
    "age": 0, \  
    "firstname": "string", \  
    "lastname": "string" \  
  } \  
}, 'http://localhost:8080/test-bean'
```

Request Headers

{
 "Accept": "*/*"
}

Response Body

"test-bean"

Response Code

200

11. You need a little trick here just add **@Valid** on person bean inside **TestBean** as picture below:

```
TestBean.java
1 package com.example.demo.bean;
2
3 import javax.validation.Valid;
4
5
6 public class TestBean {
7
8     @NotNull
9     @Valid
10    private Person person;
11
12    public Person getPerson() {
13        return person;
14    }
15
16    public void setPerson(Person person) {
17        this.person = person;
18    }
19
20    @Override
21    public String toString() {
22        return "TestBean [person=" + person + "]";
23    }
24
25 }
```

now try again, you will see that got some error as expected.

POST /test-bean

Response Class (Status 200)

string

Response Content Type */* v

Parameters

Parameter	Value	Description
testBean	<pre>{ "person": { "age": 0, "firstname": "string", "lastname": "string" } }</pre>	testBean

Parameter content type: application/json v

Try it out! Hide Response

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: a
{"person": {
  "age": 0,
  "firstname": "string",
  "lastname": "string"
}}, "http://localhost:8080/test-bean"
```

Response Body

```
{
  "fieldErrors": [
    {
      "resource": "testBean",
      "field": "person.age",
      "code": "InRange",
      "message": "size must be between 1 and 10"
    }
  ]
}
```

Response Code

400