Spring Web MVC - validate individual request params

I'm running a webapp in Spring Web MVC 3.0 and I have a number of controller methods whose signatures are roughly as follows:

```
@RequestMapping(value = "/{level1}/{level2}/foo", method = RequestMethod.POST)
public ModelAndView createFoo(@PathVariable long level1,
        @PathVariable long level2,
        @RequestParam("foo_name") String fooname,
        @RequestParam(value = "description", required = false) String description);
```

I'd like to add some validation - for example, `description` should be limited to a certain length or `fooname` should only contain certain characters. If this validation fails, I want to return a message to the user rather than just throw some unchecked exception (which would happen anyway if I let the data percolate down to the DAO layer). I'm aware of JSR303 but have not worked with it and don't quite understand how to apply it in a Spring context.

From what I understand, another option would be to bind the `@RequestBody` to an entire domain object and add validation constraints there, but currently my code is set up to accept individual parameters as shown above.

What is the most straightforward way to apply validation to input parameters using this approach?

java spring validation spring-mvc

asked Jun 1 '11 at 15:30

Dan
7,836 4 32 69

I would try using javax.validation annotations there and see if it works. I don't know if it does :) – Bozho Jun 1 '11 at 15:49

Can you elaborate? Would I use the annotation on the parameter declaration itself like I do with `@RequestParam` ? – Dan Jun 1 '11 at 16:30

Also, this is not a Java EE app and not running in a Java EE container, it's just a plain dynamic webapp running in a servlet container. – Dan Jun 1 '11 at 16:46

`@RequestParam @Regex(..) String foo` . It doesn't matter it's not JavaEE as long as there's javax.validation provider, and `mvc:annotation-driven` – Bozho Jun 1 '11 at 16:49

3 Answers

There's nothing built in to do that, not yet anyway. With the current release versions you will still need to use the WebDataBinder to bind your parameters onto an object if you want automagic validation. It's worth learning to do if you're using SpringMVC, even if it's not your first choice for this task.

It looks something like this:

```
public ModelAndView createFoo(@PathVariable long level1,
        @PathVariable long level2,
        @Valid @ModelAttribute() FooWrapper fooWrapper,
        BindingResult errors) {
  if (errors.hasErrors() {
     //handle errors, can just return if using Spring form:error tags.
  }
}

public static class FooWrapper {
  @NotNull
  @Size(max=32)
  private String fooName;
  private String description;
//getset
}
```

If you have Hibernate Validator 4 or later on your classpath and use the default dispatcher setup it should "Just work."

Editing since the comments were getting kind of large:

Any Object that's in your method signature that's not one of the 'expected' ones Spring knows how to inject, such as `HttpRequest` , `ModelMap` , etc, will get data bound. This is accomplished for simple cases just by matching the request param names against bean property names and calling setters. The `@ModelAttribute` there is just a personal style thing, in this case it isn't doing anything. The JSR-303 integration with the @Valid on a method parameter wires in through the `WebDataBinder` . If you use `@RequestBody` , you're using an object marshaller based on the content type spring determines for the request body (usually just from the http header.) The dispatcher servlet (`AnnotationMethodHandlerAdapter` really) doesn't have a way to 'flip the validation switch' for any arbitrary marshaller. It just passes the web request content along to the message converter and gets back a Object. No BindingResult object is generated, so there's nowhere to set the Errors anyway.

You can still just inject your validator into the controller and run it on the object you get, it just doesn't have the magic integration with the `@Valid` on the request parameter populating the `BindingResult` for you.

edited Jan 30 '12 at 16:32

Brabster
28.1k 19 118 168

answered Jun 1 '11 at 17:06

Affe
37.5k 6 64 71

1 What's the difference between `@ModelAttribute` and `@RequestBody` for mapping my input bean? – Dan Jun 1 '11 at 19:27

2 @ModelAttribute specifically uses the web data binder for mapping http parameters onto a POJO. @RequestBody is used for serializing the actual content of the request into a java object using an arbitrary mapping/deserializing technology defined in the dispatcher. The most common example for @RequestBody would be your POST body contains a JSON string and you use Jackson to turn it into a POJO – Affe Jun 1 '11 at 20:20

Ok, so how does a @ModelAttribute work then? Individual named paramters are just mapped into the names of properties in the POJO? Also, why can't I use @RequestBody with validation? – Dan Jun 1 '11 at 20:57

@ModelAttribute isn't even really actually required there. I could have left it out and it would work the same. It's just personal taste to have it explicitly marked where the binder is going to run. By default it's just going to match up HTTP paramter names with bean property names and invoke the setters. – Affe Jun 1 '11 at 21:08

Folded comment into into answer – Affe Jun 1 '11 at 21:35

This seems to be possible now (tried with Spring 4.1.2), see https://raymondhlee.wordpress.com/2015/08/29/validating-spring-mvc-request-mapping-method-parameters/

Extract from above page:

1. Add MethodValidationPostProcessor to Spring @Configuration class:

```
@Bean
public MethodValidationPostProcessor methodValidationPostProcessor() {
    return new MethodValidationPostProcessor();
}
```

2. Add @Validated to controller class
3. Use @Size just before @RequestParam

```
@RequestMapping("/hi")
public String sayHi(@Size(max = 10, message = "name should at most 10 characters long")
@RequestParam("name") String name) {
    return "Hi " + name;

}
```

4. Handle ConstraintViolationException in an @ExceptionHandler method

answered Aug 29 '16 at 18:16

anre
2,366 17 26

I'd like to add that `@Size` annotation can be used to validate Collections, Arrays, and Maps size as well. – naXa May 25 '17 at 11:11

2 I am using Spring Boot and somehow this set up doesn't work for me.Validations are never called. – Sabir Khan Jun 1 '17 at 10:38

In 2017, I am trying with Spring Boot 1.5.3 and it just works if I have a `@Validated` on the controller. I didn't have to add hibernate-validators dependency nor did I configure the `MethodValidationPostProcessor` . – adarshr Jun 19 '17 at 9:22

1 The problem with this method is that we cannot ignore or correct validation errors. We are forced to exit the controller method, by the exception. – Jarekczek Jun 24 '17 at 12:00

If you have multiple request parameters that need to be validated (with Http **GET** or **POST**). You might as well create a **custom model class** and use **@Valid** along with **@ModelAttribute** to validate the parameters. This way you can use Hibernate Validator or javax.validator api to validate the params. It goes something like this:

Request Method:

```
@RequestMapping(value="/doSomething", method=RequestMethod.GET)
public Model dosomething(@Valid @ModelAttribute ModelRequest modelRequest, BindingResult
result, Model model) {

    if (result.hasErrors()) {
        throw new SomeException("invalid request params");
```

```
    }

    //to access the request params
    modelRequest.getFirstParam();
    modelRequest.getSecondParam();

    ...
}
```

ModelRequest class:

```
class ModelRequest {

    @NotNull
    private String firstParam;

    @Size(min = 1, max = 10, message = "You messed up!")
    private String secondParam;

    //Setters and getters

    public void setFirstParam (String firstParam) {
        this.firstParam = firstParam;
    }

    public String getFirstParam() {
        return firstParam;
    }

    ...
}
```

Hope that helps.

edited Mar 4 '15 at 19:11

nerdherd
1,213 1 14 26

answered Nov 15 '12 at 22:33

MasterV
762 10 17

does it mean I have to create many custom model class for different request? – yuxh Apr 19 '17 at 9:26