24 Spring Core vs Spring MVC参数校验

更新时间：2020-08-04 18:23:28

> 先相信你自己，然后别人才会相信你。——屠格涅夫

背景

在任何时候，当你要处理一个应用程序的业务逻辑，数据校验是你必须要考虑和面对的事情。应用程序必须通过某种手段来确保输入进来的数据从语义上来讲是正确的。在通常的情况下，应用程序是分层的，不同的层由不同的开发人员来完成。很多时候同样的数据验证逻辑会出现在不同的层，这样就会导致代码冗余和一些管理的问题，比如说语义的一致性等。为了避免这样的情况发生，最好是将验证逻辑与相应的域模型进行绑定。

Bean Validation 为 JavaBean 验证定义了相应的元数据模型和 API。缺省的元数据是 Java Annotations，通过使用 XML 可以对原有的元数据信息进行覆盖和扩展。Bean Validation 是一个运行时的数据验证框架，在验证之后验证的错误信息会被马上返回。

Hibernate Validator 是 Bean Validation 的参考实现 。Hibernate Validator 提供了 JSR 303、JSR 349 、JSR 380 规范中所有内置 constraint 的实现，除此之外还有一些附加的 constraint。

Spring MVC 使用 Hibernate-Validator 进行参数校验实例

1. 加入 Hibernate 依赖

```
dependencies {
    // This dependency is exported to consumers, that is to say found on their compile classpath.
    api 'org.apache.commons:commons-math3:3.6.1'

    // This dependency is used internally, and not exposed to consumers on their own compile classpath.
    implementation 'com.google.guava:guava:28.0-jre'

    // Use JUnit test framework
    testImplementation 'junit:junit:4.12'

/*  compile(project(":spring-beans"))
    compile(project(":spring-core"))
    compile(project(":spring-aop"))
    compile(project(":spring-context"))
    compile(project(":spring-oxm"))
    compile(project(":spring-jdbc"))*/
    compile(project(":spring-web"))
    compile(project(":spring-webmvc"))
    compile group: 'javax.servlet', name: 'javax.servlet-api', version: '3.0.1'
    runtime 'javax.servlet:jstl:1.1.2'
    compile group: 'com.fasterxml.jackson.core', name: 'jackson-core', version: '2.9.8'
    compile group: 'com.fasterxml.jackson.core', name: 'jackson-databind', version: '2.9.8'
    compile group: 'com.fasterxml.jackson.core', name: 'jackson-annotations', version: '2.9.8'
    compile group: 'org.hibernate.validator', name: 'hibernate-validator', version: '6.1.2.Final'
}
```

2. 定义 Bean

```
package org.framework.davidwang456.controller;

import javax.validation.constraints.Email;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;

import lombok.Data;

@Data
public class User {
    private Long id;

    @Size(min = 5, max = 20)
    private String name;


    @Size(min = 6, max = 15)
    @Pattern(regexp = "\\S+", message = "Spaces are not allowed")
    private String password;

    @NotEmpty
    @Email
    private String email;
}
```

3. 控制器 Controller

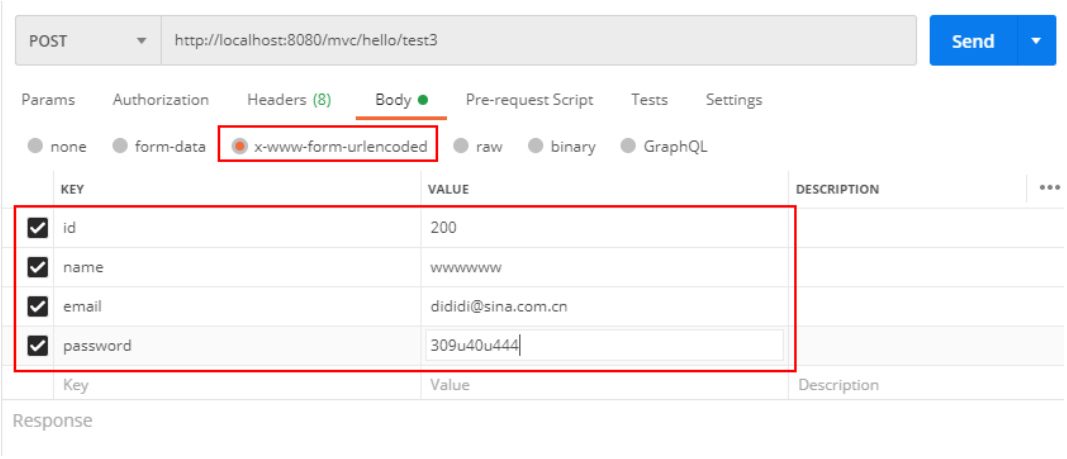
```
@RequestMapping(value="/test3")
@ResponseBody
public String handlePostRequest (@Valid User user, BindingResult bindingResult,
                                 Model model) {

    if (bindingResult.hasErrors()) {
        populateError("name", model, bindingResult);
        populateError("email", model, bindingResult);
        populateError("password", model, bindingResult);
        throw new NullPointerException();
    }
    return "registration-done";
}

private void populateError (String field, Model model, BindingResult bindingResult) {
    if (bindingResult.hasFieldErrors(field)) {
        model.addAttribute(field + "Error", bindingResult.getFieldError(field)
                                                         .getDefaultMessage());
    }
}
```

4. postman 测试

header：Content-Type application/x-www-form-urlencoded



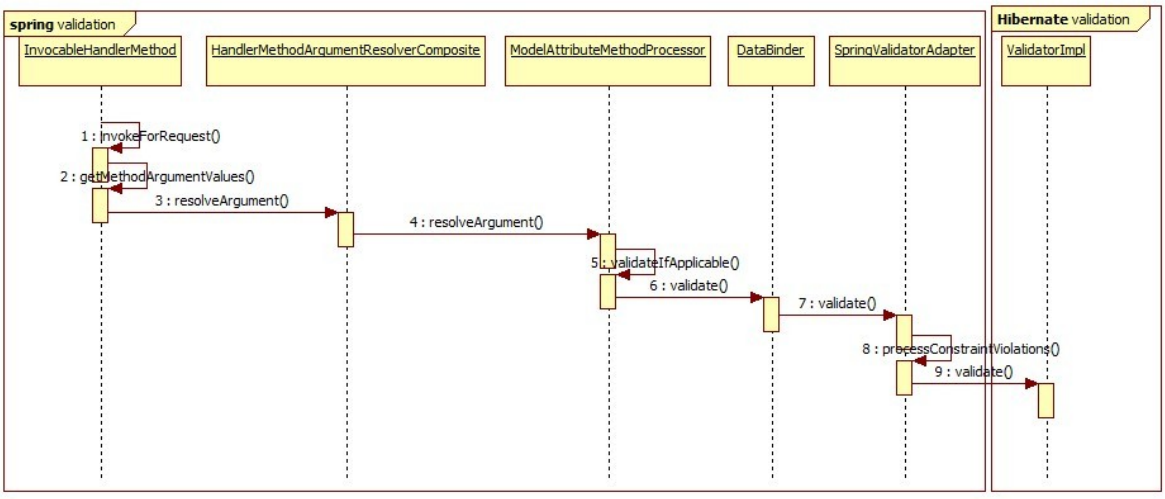
测试结果：

```
registration-done
```

原理探秘

将参数 email 置空，此时抛出异常：

可以看出异常发生在获取参数，验证参数时发生异常。在 InvocableHandlerMethod.java#invokeForRequest 方法打断点，一步步追踪。



其中，重点是 Spring 通过适配器 SpringValidatorAdapter 来调用 Hibernate 的 ValidatorImpl.java#validate 实现：

```
@SuppressWarnings({ "unchecked", "rawtypes" })
@Override
public void validateValue(
        Class<?> targetType, String fieldName, @Nullable Object value, Errors errors, Object... validationHints) {

    if (this.targetValidator != null) {
        processConstraintViolations(this.targetValidator.validateValue(
                (Class) targetType, fieldName, value, asValidationGroups(validationHints)), errors);
    }
}
```

@Valid vs @Validated

Spring Validation 验证框架对参数的验证机制提供了@Validated（Spring's JSR-303 规范，是标准 JSR-303 的一个变种），javax 提供了 @Valid（标准JSR-303规范），配合 BindingResult 可以直接提供参数验证结果。

在检验 Controller 的入参是否符合规范时，使用 @Validated 或者 @Valid 在基本验证功能上没有太多区别。但是在分组、注解地方、嵌套验证等功能上两个有所不同。

1. 分组

@Validated：提供了一个分组功能，可以在入参验证时，根据不同的分组采用不同的验证机制，可参考高效使用hibernate-validator校验框架；

@Valid：作为标准JSR-303规范，不支持分组的功能。

2. 注解作用的位置

@Validated：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；

@Valid：可以用在方法、构造函数、方法参数和成员属性（字段）上。

总结

对于任何一个应用而言在客户端做的数据有效性验证都不是安全有效的，这时候就要求我们在开发的时候在服务端也对数据的有效性进行验证。 SpringMVC 自身对数据在服务端的校验（Hibernate Validator）有一个比较好的支持，它能将我们提交到服务端的数据按照我们事先的约定进行数据有效性验证，对于不合格的数据信息 SpringMVC 会把它保存在错误对象中（Errors接口的子类），这些错误信息我们也可以通过 SpringMVC 提供的标签（form:errors）在前端 JSP 页面上进行展示。或者使用拦截器 after 方法对处理错误信息进行处理后传递给页面（我们使用JSON请求的时候就需要这样做）。

}

23 Spring Core参数校验之JSR-303_JSR-349注解

25 Spring Core 数据绑定之BeanWrapper实现示例及背后原理探究