springboot 参数校验

作者：crick77
原文链接：https://ld246.com/article/1512875839848
来源网站：链滴

本文项目已发布到github，后续学习项目也会添加到此工程下，欢迎fork点赞。

https://github.com/wangyuheng/spring-boot-sample

http接口开发过程中的常用场景为，根据提交的表单数据进行格式校验，包括字段长度、数据类型、围等等。。如果每次都写一堆if...else if... 太傻了，所以java提供了一套标准化校验方案**JSR 303**，而准的最佳实践为**Hibernate Validator**。

一句话为，通过注解对bean进行校验，并返回标准文案。

依赖

spring-boot-starter-web已经集成了hibernate-validator, 无需重复引用。

注解

JSR303定义了一套，在javax.validation.constraints包目录下，hibernate-validator扩展了一套，在rg.hibernate.validator.constraints包下，下文会介绍如何自定义注解。

在Bean的字段中增加注解，代码如下：

```
public class User implements Serializable {

    @Min(1)
    private int id;
    @Email
    private String username;
    @NotBlank
    @Length(min = 6, max = 36)
    private String password;
    private Integer age;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
```

```
        this.password = password;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }
}
```

Controller

1. @Validated 声明对参数进行校验
2. MessageSource 用来分离错误提示，也可以实现国际化
3. BindingResult 绑定参数校验结果

```
@RestController
@RequestMapping("/user")
public class UserApi {

    @Autowired
    private MessageSource messageSource;

    @GetMapping("/{id}")
    public Object getInfo(@Validated User user, BindingResult bindingResult) {
        Map<String, Object> result = new HashMap<>();
        if (bindingResult.hasErrors()) {
            List<FieldError> fieldErrors = bindingResult.getFieldErrors();
            Locale locale = LocaleContextHolder.getLocale();
            StringBuilder errorMessage = new StringBuilder();
            fieldErrors.forEach(fieldError -> {
                errorMessage.append(fieldError.getField())
                        .append(":")
                        .append(messageSource.getMessage(fieldError, locale))
                        .append(",");
            });
            result.put("code", 10001);
            result.put("message", errorMessage);
        } else {
            result.put("code", 10000);
            result.put("data", user);
        }
        return result;
    }

}
```

自定义错误提示

框架本身做了错误提示，但是为了友好，通常会自定义提示。

硬编码

可以在 注解中硬编码提示语，如

```
@Email(message = "用户名必须是邮箱")
private String username;
```

ValidationMessages.properties

不过不够灵活。在使用spring-boot的过程中，我们都熟悉了约定大于配置。可以在resources目录下加**ValidationMessages.properties**文件，并在其中复写

```
javax.validation.constraints.Min.message=参数最小为{1}
```

可以实现自定义提示，注意properties中文编码问题。@注1 propertiesEditor

messages.properties

springboot提供的消息文件默认路径为resources下messages.properties，可以把ValidationMessages.properties和messages.properties指定为自定义配置文件

- 在 **application.properties**中配置属性

```
spring.messages.basename=valid
```

- 在resources目录下创建校验提示文件 **valid.properties**

```
org.hibernate.validator.constraints.NotBlank.message={0} can't be blank
user.id.error={0} error
```

- 配置messageSource

```
@Configuration
public class ValidatorConf {

    @Autowired
    private MessageSource messageSource;

    @Bean
    @ConditionalOnClass(MessageSource.class)
    public LocalValidatorFactoryBean localValidatorFactoryBean() {
        LocalValidatorFactoryBean bean = new LocalValidatorFactoryBean();
        bean.setProviderClass(HibernateValidator.class);
        bean.setValidationMessageSource(messageSource);
        return bean;
    }
}
```

简化

需要的校验功能实现了，但是每个restful接口都需要增加这些校验代码？每个参数后面都加一个BindingResult？ 显然不合理。懒惰是美德，试着做一次简化。

通过@RestControllerAdvice和@ExceptionHandler全局捕获restapi的BindException异常，在controller代码中，在需要校验的参数前增加@Validated

```
@RestControllerAdvice
public class GlobalValidator {

    @Autowired
    private MessageSource messageSource;

    @ExceptionHandler(BindException.class)
    public Object bindError(BindException e) {
        Map<String, Object> result = new HashMap<>();
        List<FieldError> fieldErrors = e.getFieldErrors();
        Locale locale = LocaleContextHolder.getLocale();
        StringBuilder errorMessage = new StringBuilder();
        fieldErrors.forEach(fieldError -> {
            errorMessage.append(fieldError.getField())
                    .append(":")
                    .append(messageSource.getMessage(fieldError, locale))
                    .append(",");
        });
        result.put("code", 10001);
        result.put("message", errorMessage);
        return result;
    }
}

@RestController
@RequestMapping("/simpleuser")
public class SimpleUserApi {

    @GetMapping("/{uid}")
    public Object getInfo(@Validated User user) {
        Map<String, Object> result = new HashMap<>();
        result.put("code", 10000);
        result.put("data", user);
        return result;
    }

}
```

测试

mockMvc请求api接口，判断返回值code

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class ValidatorApiTest {

    private MockMvc mockMvc;

    @Autowired
    private WebApplicationContext context;
```

```
    @Before
    public void setup() {
        mockMvc = MockMvcBuilders.webAppContextSetup(context).build();
    }

    private JSONObject requestApi(String path) throws Exception {
        return new JSONObject(mockMvc.perform(MockMvcRequestBuilders.get(path))
                .andExpect(MockMvcResultMatchers.status().isOk())
                .andDo(MockMvcResultHandlers.print())
                .andReturn()
                .getResponse()
                .getContentAsString());
    }

    @Test
    public void test_user_param_valid() throws Exception {
        String rightPath = "/user/" + "12?id=123" + "&password=123456";
        assertTrue(10000 == requestApi(rightPath).getInt("code"));
        String errorPath = "/user/" + "abc";
        assertTrue(10001 == requestApi(errorPath).getInt("code"));
    }


    @Test
    public void test_simpleuser_param_valid() throws Exception {
        String rightPath = "/simpleuser/" + "12?id=123" + "&password=123456";
        assertTrue(10000 == requestApi(rightPath).getInt("code"));
        String errorPath = "/simpleuser/" + "abc";
        assertTrue(10001 == requestApi(errorPath).getInt("code"));
    }

}
```