玩出花的Bean Validation

技术历史

J2EE6的一项子规范，JSR-303，2009年跟随JAVA6面世，spring系列已整合

技术特性

- 注解编程
- 校验无需侵入业务代码
- 实际使用中可统一处理校验结果

项目应用

整合自定义异常

增加注解校验后，系统框架会转换为自定义异常，最终转为json信息返回至浏览器，校验信息通过 `propertyPath` 绑定字段

```
{
  "rspResult": {
    "bizCode": 420,
    "resultCode": "FAIL_BUS",
    "message": "form validate fail",
    "bvResults": [
      {
        "propertyPath": "name",
        "message": "用户名已存在"
      }
    ],
    "viewName": null
  }
}
```

整合国际化

校验注解message字段可以设置 国际化标签，标签来源 `db_literal` 表，domain类型为 `validator`

如 `validator.constraints.URL.message`

校验注解国际化字段，必须domain为 **validator**，标签前缀为 `validator.constraints`

使用方式

- **简单校验注解**

```
@NotBlank(message = "validator.constraints.NotBlank.message")
private String cnName;              //中文公司名称
```

```
@NotNull(message="******")
@Min(value=1,message="*****")
private Integer currency; //币种 CodeList表中的type为currency
```

注意：可同时增加多个校验注解，**但校验规则间不允许有叠加区域，否则会出现校验结果随机匹配的情况**

如业务有此类需求，另行讨论

- **条件注解**

  增加的校验注解只在特定条件下生效，如针对供应商校验/只在英文环境下有效，诸如此类

  实现方式为校验分组，`Class<?>[] groups() default {};`，例

  ```
  @NotNull(message="*****",groups=CnValidGroup.class)
  private Integer projectType;
  ```

  上面代码校验注解表示只对 中文环境下做校验，`CnValidGroup.class`为海智后台框架内置校验分组。

  已内置校验分组如下

  - `CnValidGroup`(只在中文环境下校验)
  - `NoCnValidGroup`(只在非中文环境下校验)
  - `BuyerValidGroup`(只对采购商用户校验)
  - `SupplierValidGroup`(只对供应商用户校验)
  - `IndValidGroup`（只对工业服务商用户校验）

  以上分组已在后台框架里根据用户local 标记过，直接设置group使用即可

  另有需要在controller方法上标注，需要注解的分组，使用方式如下

  ```
  @ValidGroups(CreateValidGroup.class)
  public GeneralRsp createProject(@RequestBody OssCreateProjectForm form){
          return projectService.createProject(form);
  }

  @Length(min=1,message="****",groups={CreateValidGroup.class})
  private String name;
  ```

  供手动标记的分组已内置如下

  - `CreateValidGroup`
  - `UpdateValidGroup`
  - `SpecValidGroup`

  也可以**自定义分组group**，需要继承海智`ValidGroup`，其实就是一个空的接口作为标记

- **自定义注解**

  - 针对类的注解

    `com.haizol.common.validator.Consistent`一致性校验注解，可用作密码一致性校验，使用方式如下

```
@Consistent(field = "newPwd", verifyField = "confirmPwd", message = "***")
public class UserResetPasswordForm extends BaseForm {

    private static final long serialVersionUID = -7126109220858671518L;
    @Length(min=6,message="***")
    private String newPwd;
    private String confirmPwd;
```

如果确认密码和密码不一致，校验失败，失败信息绑定至 确认密码字段返回给前端

通用的类注解校验字段

`@CustomFieldComplexValid`，可自定义实现校验方法，校验结果绑定字段,例

```
@CustomFieldComplexValid(bindProperty = "age", message = "年龄不符合要求1", vMethod =
"validAge")
@CustomFieldComplexValid(bindProperty = "age1", message = "年龄不符合要求2", vMethod =
"validAge1")
public class DemoForm extends BaseForm {

    private static final long serialVersionUID = -535838510040566693L;

    @CustomFieldValid(springEL = "#f.length()>10", message = "呀呀呀呀")
    private String name;

    private int age;

        /***
     *
     * 校验年龄demo
     * @return
     * @author Vincentzheng
     * @date 2018年9月30日 上午11:36:51
     */
    protected Boolean validAge(){
        if(name != null){
            if(name.length() == age){
                return true;
            }
        }
        return false;
    }

    protected Boolean validAge1(){
        if(name != null){
            if(name.length() == age1+1){
                return true;
            }
        }
        return false;
    }
```

- 针对字段的注解

  `com.haizol.common.web.manager.system.validator.ValidateCode`验证码字段校验，可校验用户输入的验证码及有效性。直接注解验证码字段即可，业务代码已封装

  更通用的字段注解校验

  `@CustomFieldValid`可用spring EL表达式校验自定义校验规则，例

  ```
  @CustomFieldValid(springEL = "#f.length()>10", message = "呀呀呀呀")
  private String name;
  ```

- 更多自定义注解
  - 校验用户已存在，样例代码

    ```
    @UserExist(message="用户名已存在")
    private String name;
    ```

    ```
    @Retention(RetentionPolicy.RUNTIME)
    @Target(ElementType.FIELD)
    @Inherited
    @Documented
    @Constraint(validatedBy = UserExistValidator.class)
    public @interface UserExist {
        String message() default "用户已存在";
        Class<?>[] groups() default {};
        Class<? extends Payload>[] payload() default {};
    }
    class UserExistValidator implements ConstraintValidator<UserExist,String>{

        @Override
        public void initialize(UserExist constraintAnnotation) {

        }

        @Override
        public boolean isValid(String value, ConstraintValidatorContext context) {

            OssProjectService ossProjectService =
    SpringContextUtil.getBean(OssProjectService.class);
            return ossProjectService.checkExsist(value);
        }

    }
    ```

    自定义校验注解目录定义

    `com.haizol.**.**.web.system.validator`

附录：系统注解

JSR303 校验框架注解类：

- @NotNull 注解元素必须是非空
- @Null 注解元素必须是空
- @Digits 验证数字构成是否合法
- @Future 验证是否在当前系统时间之后
- @Past 验证是否在当前系统时间之前
- @Max 验证值是否小于等于最大指定整数值
- @Min 验证值是否大于等于最小指定整数值
- @Pattern 验证字符串是否匹配指定的正则表达式
- @Size 验证元素大小是否在指定范围内
- @DecimalMax 验证值是否小于等于最大指定小数值
- @DecimalMin 验证值是否大于等于最小指定小数值
- @AssertTrue 被注释的元素必须为true
- @AssertFalse 被注释的元素必须为false

HibernateValidator扩展注解类：

- @Email 被注释的元素必须是电子邮箱地址
- @Length 被注释的字符串的大小必须在指定的范围内
- @NotEmpty 被注释的字符串的必须非空
- @Range 被注释的元素必须在合适的范围内