验证（Validation）

javax validation

自定义验证annotation

1.使用javax.validation自带验证

Step1(声明字段约束):

为字段添加注解:

```
        //
        @NotBlank(message = "{name.empty.error}")//messagei18n
        private String name;

        //
        @Size(min = 6)
        private String pass;

        //
        @Past
        private Date pastDate;

        //
        @Future
        private Date futureDate;

        //
        @Pattern(regexp = "^[0-9]+")
        private String patterStr;

        //
        @Range(min = 0, max = 100)
        private Integer num;

        //
        @Email
        private String email;

        //
        @Valid
        private SubEntity subEntity;
```

在i18n文件中添加:

```
name.empty.error=name\u4E0D\u80FD\u4E3A\u7A7A
```

Step2(开启参数验证)：

在接口传入参数前添加@Valid注解：

```
        @PostMapping("test")
    public ValidTestEntity test(@RequestBody @Valid ValidTestEntity
entity) {
        return entity;
    }
```

2. 自定义验证逻辑

Step1(注解类)：

```
@Target({ FIELD })
@Retention(RUNTIME)
//
@Constraint(validatedBy = MyValidator.class)
public @interface MyChecker {
    String message() default "{myChecker.error}";

    //
    Class<?>[] groups() default {};

    //
    Class<? extends Payload>[] payload() default {};
}
```

Step2(验证逻辑)：

```
public class MyValidator implements ConstraintValidator<MyChecker, String>
{//
    @Override
    public void initialize(MyChecker constraintAnnotation) {
        System.out.println(constraintAnnotation.message());//
    }

    @Override
    public boolean isValid(String s, ConstraintValidatorContext
constraintValidatorContext) {
                //
        if ("myValidator".equals(s))
            return true;
        return false;
    }
}
```

Step3(自定义注解的使用)：

与javax.validation提供的注解无异

样例:http://git.acca.com.cn:7990/projects/OPRA-GIT/repos/opra-mas/browse/opra-mas-server

~~是否需要架构提供一些预制的验证，主文件相关？~~