## 客户端验证

一般，使用JavaScript进行客户端验证的步骤如下：

编写验证函数

在提交表单的事件中调用验证函数

根据验证函数来判断是否进行表单提交

比如：

**表单提交按钮：**

<button type=*"submit"* class=*"btn"* onclick="isCommit()">提交</button>

<button type=*"reset"* class=*"btn"* onclick="isCancel()">重置</button>

**JavaScript进行验证：**

**function** isCancel() {

document.forms[0].action="";

}

**function** isCommit() {

/\*判断姓名是否为空 \*/

**var** name = document.forms[0].uname.value;

**if**(name == ""){

alert("姓名不能为空！")

document.forms[0].uname.focus();

**return** **false**;

}

/\*判断密码是否为空 \*/

**var** pass = document.forms[0].upass.value;

**if**(pass == ""){

alert("密码不能为空！")

document.forms[0].upass.focus();

**return** **false**;

}

/\*判断确认密码是否为空 \*/

**var** repass = document.forms[0].reupass.value;

**if**(repass == ""){

alert("确认密码不能为空！");

document.forms[0].reupass.focus();

**return** **false**;

}

/\*判断密码和确认密码是否一致 \*/

**if**(pass != repass){

alert("两次密码输入不同！");

document.forms[0].reupass.value="";

document.forms[0].reupass.focus();

**return** **false**;

}

document.forms[0].submit();

**return** **true**;

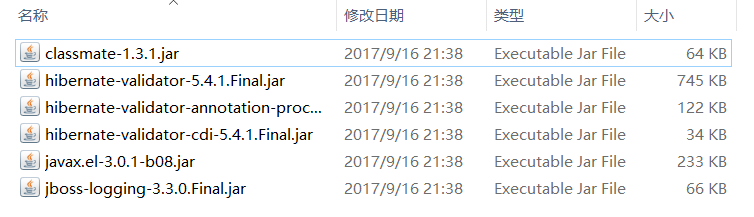
}

## 服务端验证（使用JSR303验证）

使用JSR303进行验证：JSR303验证，目前有两个实现，一个是Hibernate Validator，一个是Apache BVal。本书采用的是Hibernate Validator，注意它和Hibernate无关，只是使用它进行数据验证。

### 2.1 步骤流程

1）导入所需要的jar包



2）配置属性文件与验证器

如果将验证错误消息放在属性文件中，那么需要在配置文件配置属性文件，并将属性文件与Hibernate Validator关联

3）JSR 303不需要编写验证器，但需要利用它的标注类型在领域模型的属性上嵌入约束。

### 3.1 空检查

@Null：验证对象是否为null。

@NotNull：验证对象是否不为null，无法查检长度为0的字符串。

@NotBlank：检查约束字符串是不是null，还有被trim后的长度是否大于0，只对字符

串，且会去掉前后空格。

@NotEmpty：检查约束元素是否为null或者是empty。

示例如下：

@NotBlank(message="{goods.gname.required}")//goods.gname.required为属性文件的错误代码

private String gname;

注意：

//@NotEmpty 用在集合类上面

//@NotBlank 用在String上面

//@NotNull 用在基本类型上

### 3.2 booelan检查

@AssertTrue：验证boolean属性是否为true。

@AssertFalse：验证boolean属性是否为false。

示例如下：

@AssertTrue

private boolean isLogin;

### 3.3 长度检查

@Size(min=, max=)：验证对象（Array，Collection，Map，String）长度是否在给定的范围之内。

@Length(min=, max=)：验证字符串长度是否在给定的范围之内。

示例如下：

@Length(min=1,max=100)

private String gdescription;

### 3.4 日期检查

@Past：验证Date和Calendar对象是否在当前时间之前。

@Future：验证Date和Calendar对象是否在当前时间之后。

@Pattern：验证String对象是否符合正则表达式的规则。

示例如下：

@Past(message="{gdate.invalid}")

private Date gdate;

### 3.5 数值检查

@Min：验证Number和String对象是否大等于指定的值。

@Max：验证Number和String对象是否小等于指定的值。

@DecimalMax：被标注的值必须不大于约束中指定的最大值，这个约束的参数是一个通过BigDecimal定义的最大值的字符串表示，小数存在精度。

@DecimalMin：被标注的值必须不小于约束中指定的最小值，这个约束的参数是一个通过BigDecimal定义的最小值的字符串表示，小数存在精度。

@Digits：验证Number和String的构成是否合法。

@Digits(integer=,fraction=)：验证字符串是否符合指定格式的数字，interger指定整数精度，fraction指定小数精度。

@Range(min=, max=)：检查数字是否介于min和max之间。

@Valid：对关联对象进行校验，如果关联对象是个集合或者数组，那么对其中的元素进行校验，如果是一个map，则对其中的值部分进行校验。

@CreditCardNumber：信用卡验证。

@Email：验证是否是邮件地址，如果为null，不进行验证，通过验证。

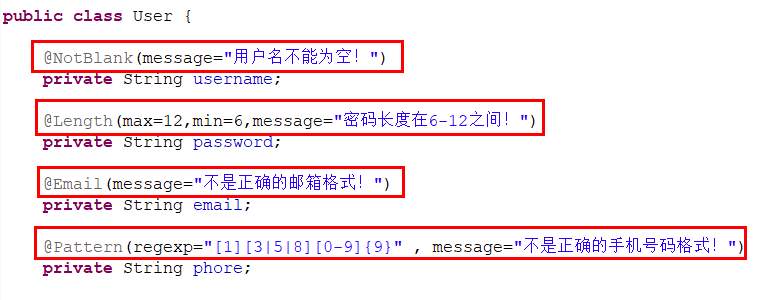
示例如下：

@Range(min=0,max=100,message="{gprice.invalid}")

private double gprice;

## 实验截图（用户注册JSR303验证）

### 3.1 javaBean中“属性验证”注解



### 3.2 在userController类中，使用@Valid验证



### 3.3 register.jsp页面中，展示错误信息

