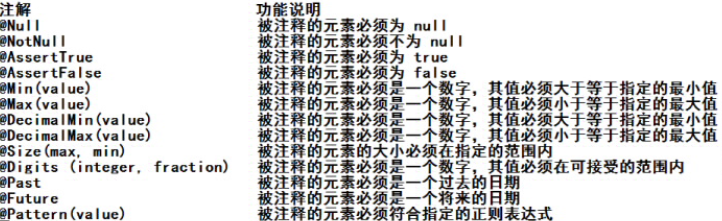
参考文档：https://blog.csdn.net/eson\_15/article/details/51725470

# 数据检验

## JSR 303

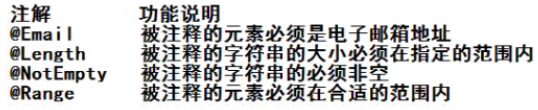
**JSR 303** 是 Java 为 Bean 数据合法性校验提供的标准框架，它已经包含在 JavaEE 6.0 中 .

JSR 303 通过在 **Bean** 属性上标注类似于 @NotNull、@Max等标准的注解指定校验规则，并通过标准的验证接口对 Bean 进行验证



## Hibernate Validator 扩展注解

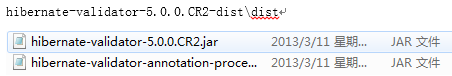
Hibernate Validator 是 JSR 303 的一个参考实现，除支持所有标准的校验注解外，它还支持以下的扩展注解

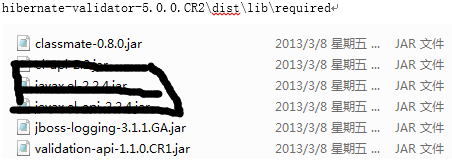


## 如何校验 ? 注解 ?

①. 使用 JSR 303 验证标准

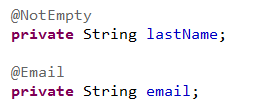
②. 加入 hibernate validator 验证框架的 jar 包





③. 在 SpringMVC 配置文件中添加 <mvc:annotation-driven />

④. 需要在 bean 的属性上添加对应的注解



⑤. 在目标方法 bean 类型的前面添加 @Valid 注解



## 验证出错

注意: 需校验的 Bean 对象和其绑定结果对象或错误对象时成对出现的，它们之间不允许声明其他的入参

Errors 接口提供了获取错误信息的方法，如 getErrorCount() 或getFieldErrors(String field) BindingResult 扩展了 Errors 接口

@RequestMapping(value="emp" , method=RequestMethod.*POST*)

**public** String save(@Valid Employee employee ,BindingResult result

,Map<String, Object> map){



**if**(result.getErrorCount() > 0){

System.*out*.println("出错了!");

**for**(FieldError error:result.getFieldErrors()){

System.*out*.println(error.getField() + ":" + error.getDefaultMessage());

}

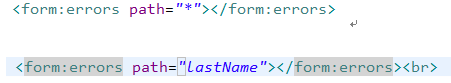
//若验证出错, 则转向定制的页面

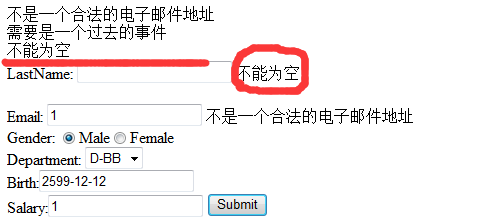
map.put("departments", departmentDao.getDepartments());

**return** "input";

}

## 错误消息 ? 如何显示





## 提示消息的国际化—了解

每个属性在数据绑定和数据校验发生错误时，都会生成一个对应的 FieldError 对象。

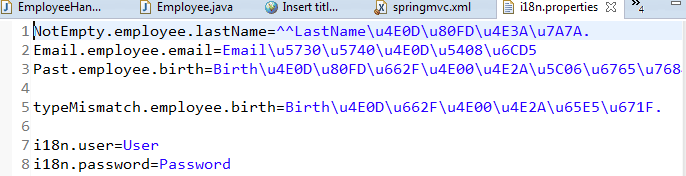
当一个属性校验失败后，校验框架会为该属性生成 4 个消息代码，这些代码以校验注解类名为前缀，结合modleAttribute、属性名及属性类型名生成多个对应的消息代码：例如 User 类中的 password 属性标准了一个 @Pattern 注解，当该属性值不满足 @Pattern 所定义的规则时, 就会产生以下 4个错误代码：

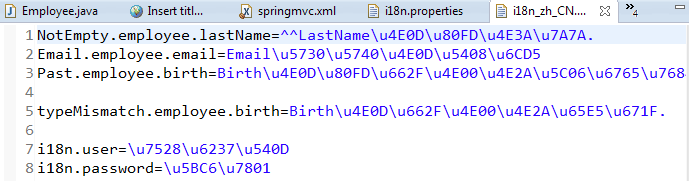
Pattern.user.password ；Pattern.password ；

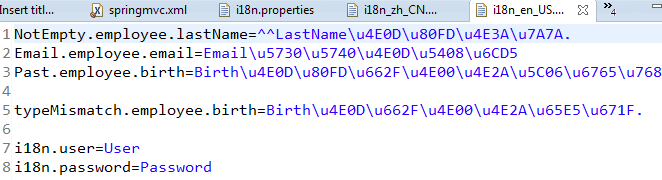
Pattern.java.lang.String ；Pattern



当使用 Spring MVC 标签显示错误• 消息时， Spring MVC 会查看WEB 上下文是否装配了对应的国际化消息，如果没有，则显示默认的错误消息，否则使用国际化消息。







若数据类型转换或数据格式转换时发生错误，或该有的参数不存在，或调用处理方法时发生错误，都会在隐含模型中创建错误消息。其错误代码前缀说明如下：

注册国际化资源文件  
 <!-- 配置国际化资源文件 -->

<bean id=*"messageSource"*

class=*"org.springframework.context.support.ResourceBundleMessageSource"*>

<property name=*"basename"* value=*"i18n"*></property>

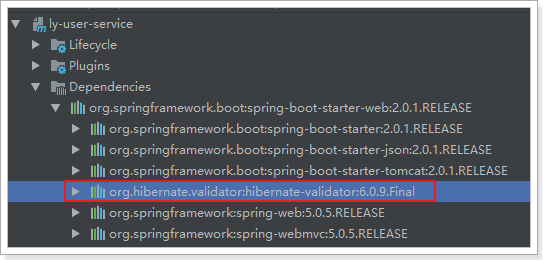
</bean>

# 6.5.hibernate-validate

刚才虽然实现了注册，但是服务端并没有进行数据校验，而前端的校验是很容易被有心人绕过的。所以我们必须在后台添加数据校验功能：

我们这里会使用Hibernate-Validator框架完成数据校验：

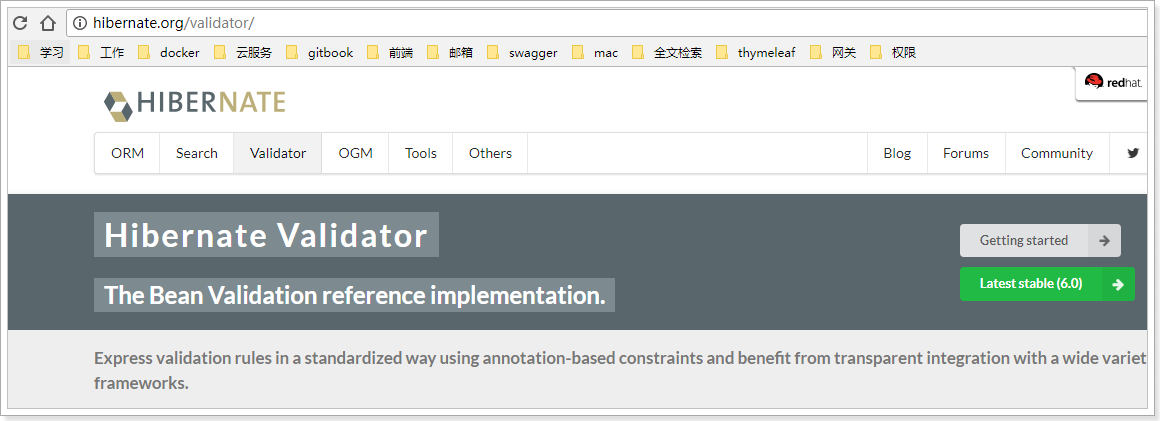
而SpringBoot的web启动器中已经集成了相关依赖：



### 6.5.1.什么是Hibernate Validator

Hibernate Validator是Hibernate提供的一个开源框架，使用注解方式非常方便的实现服务端的数据校验。

官网：<http://hibernate.org/validator/>



**hibernate Validator** 是 Bean Validation 的参考实现 。

Hibernate Validator 提供了 JSR 303 规范中所有内置 constraint（约束） 的实现，除此之外还有一些附加的 constraint。

在日常开发中，Hibernate Validator经常用来验证bean的字段，基于注解，方便快捷高效。

### 6.5.2.Bean校验的注解

常用注解如下：

|  |  |
| --- | --- |
| **Constraint** | **详细信息** |
| **@Valid** | 被注释的元素是一个对象，需要检查此对象的所有字段值 |
| **@Null** | 被注释的元素必须为 null |
| **@NotNull** | 被注释的元素必须不为 null |
| **@AssertTrue** | 被注释的元素必须为 true |
| **@AssertFalse** | 被注释的元素必须为 false |
| **@Min(value)** | 被注释的元素必须是一个数字，其值必须大于等于指定的最小值 |
| **@Max(value)** | 被注释的元素必须是一个数字，其值必须小于等于指定的最大值 |
| **@DecimalMin(value)** | 被注释的元素必须是一个数字，其值必须大于等于指定的最小值 |
| **@DecimalMax(value)** | 被注释的元素必须是一个数字，其值必须小于等于指定的最大值 |
| **@Size(max, min)** | 被注释的元素的大小必须在指定的范围内 |
| **@Digits (integer, fraction)** | 被注释的元素必须是一个数字，其值必须在可接受的范围内 |
| **@Past** | 被注释的元素必须是一个过去的日期 |
| **@Future** | 被注释的元素必须是一个将来的日期 |
| **@Pattern(value)** | 被注释的元素必须符合指定的正则表达式 |
| **@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 |
| **@NotBlank** | 被注释的字符串的必须非空 |
| **@URL(protocol=,host=, port=,regexp=, flags=)** | 被注释的字符串必须是一个有效的url |
| **@CreditCardNumber** | 被注释的字符串必须通过Luhn校验算法，银行卡，信用卡等号码一般都用Luhn计算合法性 |

### 6.5.3.给User添加校验

我们在leyou-user-interface中添加Hibernate-Validator依赖：

<dependency>  
 <groupId>org.hibernate.validator</groupId>  
 <artifactId>hibernate-validator</artifactId>  
</dependency>

我们在User对象的部分属性上添加注解：

@Table(name = "tb\_user")  
public class User {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
 @Length(min = 4, max = 30, message = "用户名只能在4~30位之间")  
 private String username;// 用户名  
  
 @JsonIgnore  
 @Length(min = 4, max = 30, message = "密码只能在4~30位之间")  
 private String password;// 密码  
  
 @Pattern(regexp = "^1[35678]\\d{9}$", message = "手机号格式不正确")  
 private String phone;// 电话  
  
 private Date created;// 创建时间  
  
 @JsonIgnore  
 private String salt;// 密码的盐值  
}

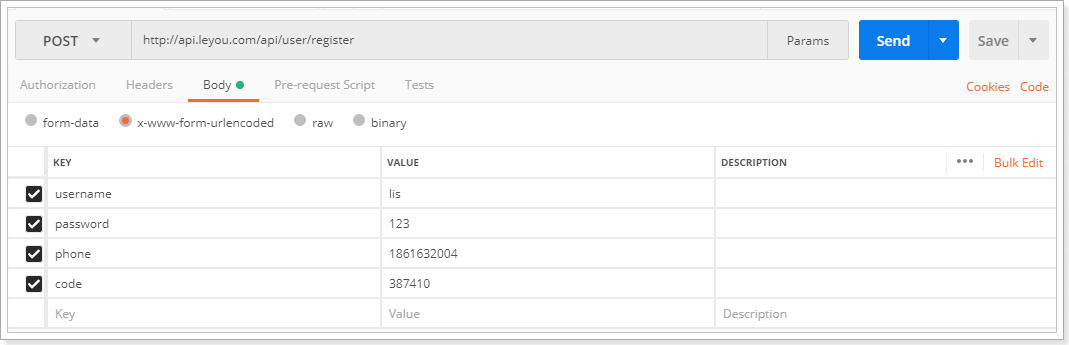
### 6.5.4.在controller上进行控制

在controller中改造register方法，只需要给User添加 @Valid注解即可。



### 6.5.5.测试

我们故意填错：



然后SpringMVC会自动返回错误信息：

