# 第五章：springboot数据校验

## 实现添加用户功能

### 修改pom.xml文件

|  |
| --- |
| <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-thymeleaf</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-tomcat</artifactId>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  </dependencies> |

### 编写实体类

|  |
| --- |
| **public** **class** User {  **private** String name;  **private** String password;  **private** Integer age;  **private** String email;    **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** Integer getAge() {  **return** age;  }  **public** **void** setAge(Integer age) {  **this**.age = age;  }  **public** String getPassword() {  **return** password;  }  **public** **void** setPassword(String password) {  **this**.password = password;  }  **public** String getEmail() {  **return** email;  }  **public** **void** setEmail(String email) {  **this**.email = email;  }  @Override  **public** String toString() {  **return** "User [name=" + name + ", password=" + password + ", age=" + age + ", email=" + email + "]";  }  } |

### 编写Controller

|  |
| --- |
| @Controller  **public** **class** UserController {  /\*\*  \* 去到添加用户页面  \* **@return**  \*/  @RequestMapping("/addUser")  **public** String addUser() {  **return** "add";  }  /\*\*  \* 保存用户  \* **@param** user  \* **@return**  \*/  @RequestMapping("/saveUser")  **public** String saveUser(User user) {  System.***out***.println(user);  **return** "ok";  }  } |

### 编写页面

#### 添加页面

|  |
| --- |
| <form th:action=*"@{/saveUser}"* method=*"post"*>  用户姓名：<input type=*"text"* name=*"name"* /><br/>  用户密码：<input type=*"password"* name=*"password"* /><br/>  用户年龄：<input type=*"text"* name=*"age"* /><br/>  <input type=*"submit"* value=*"OK"*/>  </form> |

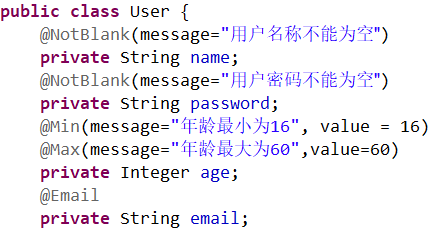
#### 成功页面

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>操作成功</title>  </head>  <body>  OK。。。。  </body>  </html> |

## 数据校验

### 步骤

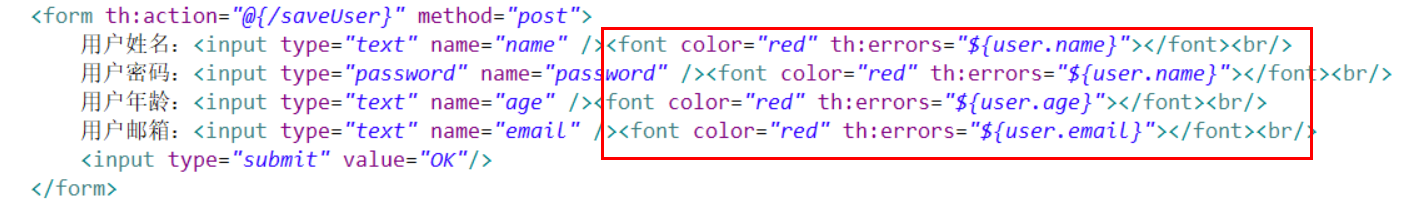
#### 在实体类添加校验注解



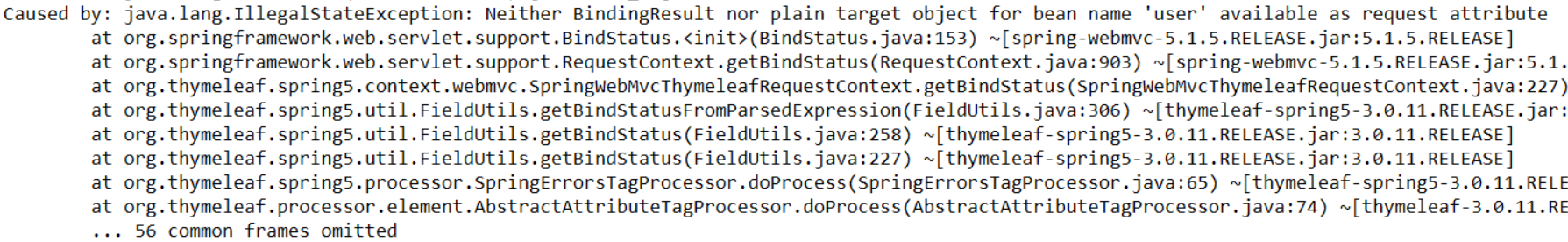
#### 在Controller开启校验



#### 在页面中获取提示信息



### 遇到异常



### 解决异常

#### 固定参数名称

解决异常的方法，在跳转页面的方法中注入一个对象，来解决问题。要求参数对象的

变量名必须是对象的类名的全称首字母小写。





注意：参数名称必须写成类的对象名，首字母小写，遵循驼峰命名法

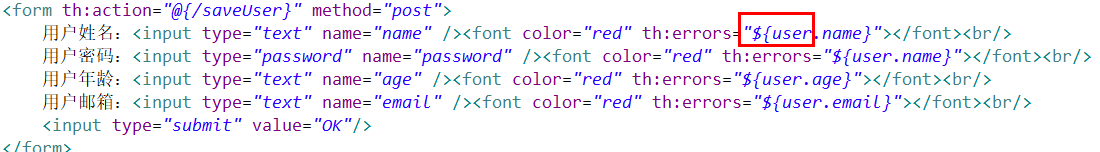
#### 参数名称任意命名

方法：给参数加入@ModelAttribute注解

如果想为传递的对象更改名称，可以使用@ModelAttribute("test")这表示当

前传递的对象的 key 为 test。那么我们在页面中获取该对象的 key 也需要修改为 test





### 其它校验注解

@NotBlank: 判断字符串是否为 null 或者是空串(去掉首尾空格)。

@NotEmpty: 判断字符串是否 null 或者是空串。

@Length: 判断字符的长度(最大或者最小)

@Min: 判断数值最小值

@Max: 判断数值最大值

@Email: 判断邮箱是否合法

## 热部署

### 概念

spring-boot-devtools 是一个为开发者服务的一个模块，其中最重要的功能就是自动应用代码更改到最新的App上面去。原理是在发现代码有更改之后，重新启动应用，但是速度比手动停止后再启动还要更快，更快指的不是节省出来的手工操作的时间

### 用法

在pom.xml文件加入热部署依赖即可，代码配置如下：

|  |
| --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <optional>true</optional>  </dependency> |