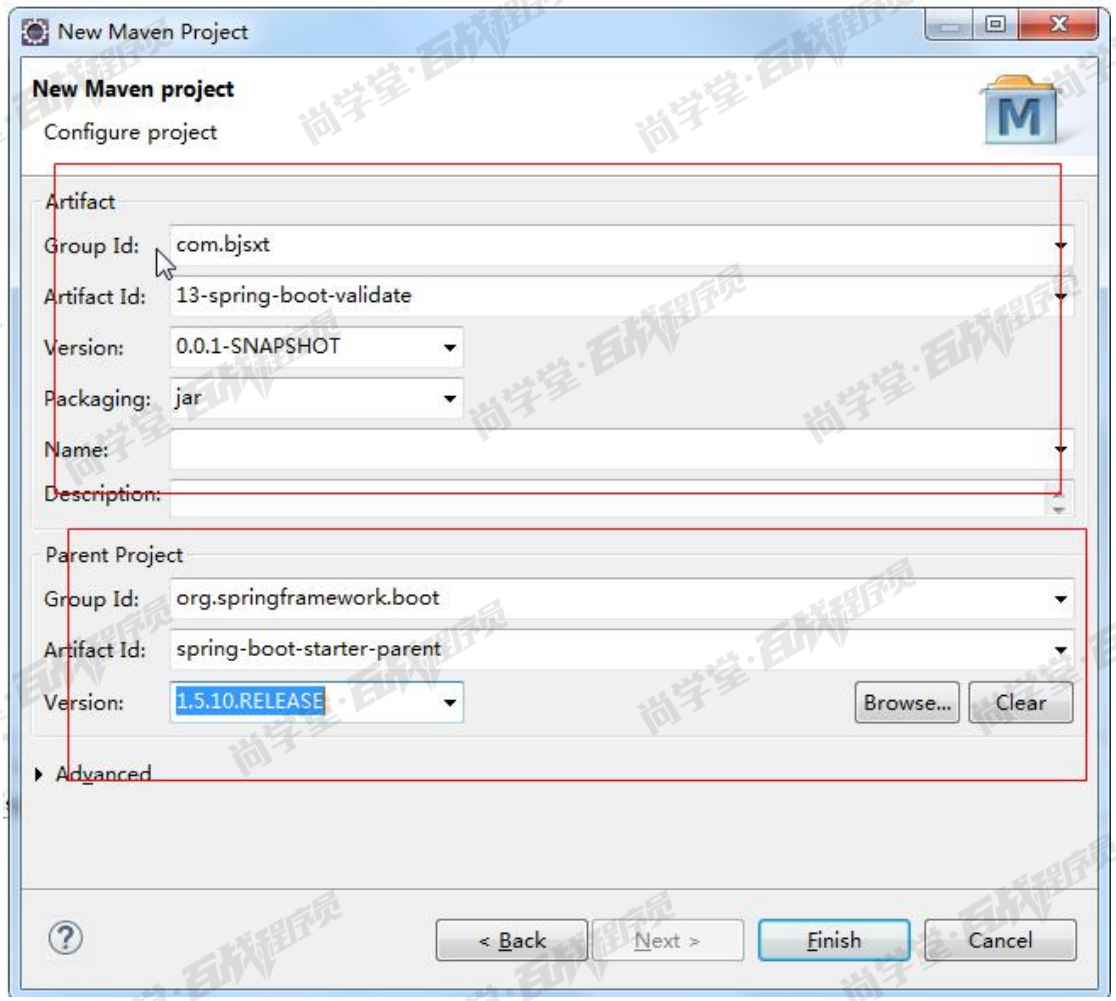
SpringBoot 第一章服务端表单数据校验

（SpringBoot 高级）

一、 实现添加用户功能

1 创建项目



2 修改 POM 文件

```
    <project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
```

```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>1.5.10.RELEASE</version>
        </parent>
        <groupId>com.bjsxt</groupId>
        <artifactId>13-spring-boot-validate</artifactId>
        <version>0.0.1-SNAPSHOT</version>

        <properties>
            <java.version>1.7</java.version>
            <thymeleaf.version>3.0.2.RELEASE</thymeleaf.version>

    <thymeleaf-layout-dialect.version>2.0.4</thymeleaf-layout-dialect.ve
rsion>
        </properties>

        <dependencies>
            <!-- springBoot 的启动器 -->
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-web</artifactId>
            </dependency>
            <!-- thymeleaf 的启动器 -->
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-thymeleaf</artifactId>
            </dependency>
        </dependencies>
    </project>
```

3 编写添加用户功能创建实体类

```
    public class Users {
        private String name;
        private String password;
        private Integer age;
        public String getName() {
            return name;
        }
        public void setName(String name) {
            this.name = name;
        }
        public String getPassword() {
```

```
            return password;
        }
        public void setPassword(String password) {
            this.password = password;
        }
        public Integer getAge() {
            return age;
        }
        public void setAge(Integer age) {
            this.age = age;
        }
        @Override
        public String toString() {
            return "Users [name=" + name + ", password=" + password + ", age="
+ age + "]";
        }

    }
```

3.1编写 Controller

```
    /**
     * SpringBoot 表单数据校验
     *
     *
     */
    @Controller
    public class UsersController {

        @RequestMapping("/addUser")
        public String showPage(){
            return "add";
        }

        /**
         * 完成用户添加
         */
        @RequestMapping("/save")
        public String saveUser(Users users){
            System.out.println(users);
            return "ok";
        }
```

```
}
```

3.2编写页面 add.html ok.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>添加用户</title>
</head>
<body>
    <form th:action="@{/save}" 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>
</body>
</html>
```

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

二、 SpringBoot 对表单做数据校验

1 SpringBoot 对表单数据校验的技术特点

1.1 SpringBoot 中使用了 Hibernate-validate 校验框架

2 SpringBoot 表单数据校验步骤

2.1 在实体类中添加校验规则

```
    public class Users {
        @NotBlank //非空校验
        private String name;
        @NotBlank //密码非空校验
        private String password;
        private Integer age;
        public String getName() {
            return name;
        }
        public void setName(String name) {
            this.name = name;
        }
        public String getPassword() {
            return password;
        }
        public void setPassword(String password) {
            this.password = password;
        }
        public Integer getAge() {
            return age;
        }
        public void setAge(Integer age) {
            this.age = age;
        }
        @Override
        public String toString() {
            return "Users [name=" + name + ", password=" + password + ", age="
+ age + "]";
        }

    }
```

2.2在 Controller 中开启校验

```
    /**
         * 完成用户添加
         *@Valid 开启对 Users 对象的数据校验
         *BindingResult:封装了校验的结果
         */
        @RequestMapping("/save")
        public String saveUser(@Valid Users users,BindingResult result){
            if(result.hasErrors()){
                return "add";
            }
            System.out.println(users);
            return "ok";
        }
```

2.3在页面中获取提示信息

```
    <!DOCTYPE html>
    <html>
    <head>
    <meta charset="UTF-8">
    <title>添加用户</title>
    </head>
    <body>
        <form th:action="@{/save}" method="post">
            用户姓名：<input type="text" name="name"/><font color="red"
th:errors="${users.name}"></font><br/>
            用户密码：<input type="password" name="password" /><font
color="red" th:errors="${users.password}"></font><br/>
            用户年龄：<input type="text" name="age" /><font color="red"
th:errors="${users.age}"></font><br/>
            <input type="submit" value="OK"/>
        </form>
    </body>
    </html>
```

2.4遇到异常

```
java.lang.IllegalStateException: Neither BindingResult nor plain target object for bean name 'users' available as req
        at org.springframework.web.servlet.support.BindStatus.<init>(BindStatus.java:144) ~[spring-webmvc-4.3.14.RELE
        at org.thymeleaf.spring4.util.FieldUtils.getBindStatusFromParsedExpression(FieldUtils.java:307) ~[thymeleaf-s
        at org.thymeleaf.spring4.util.FieldUtils.getBindStatus(FieldUtils.java:258) ~[thymeleaf-spring4-3.0.2.RELEASE
```

三、 解决数据校验时的异常问题

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。

代码

```
    /**
         * 解决异常的方式。可以在跳转页面的方法中注入一个 Uesrs 对象。
         * 注意：由于 springmvc 会将该对象放入到 Model 中传递。key 的名称会使用该对象的驼峰式的命名规则来作为 key。
         * 参数的变量名需要与对象的名称相同。将首字母小写。
         *
         * @param users
         * @return
         */
        @RequestMapping("/addUser")
        public String showPage( Users users){
            return "add";
        }
```

```
    /**
         * 完成用户添加
         *@Valid 开启对 Users 对象的数据校验
         *BindingResult:封装了校验的结果
         */
        @RequestMapping("/save")
        public String saveUser( @Valid Users users,BindingResult result){
            if(result.hasErrors()){
                return "add";
            }
            System.out.println(users);
            return "ok";
        }
```

```
    <!DOCTYPE html>
    <html>
    <head>
    <meta charset="UTF-8">
    <title>添加用户</title>
    </head>
    <body>
        <form th:action="@{/save}" method="post">
            用户姓名：<input type="text" name="name"/><font color="red" th:errors="${users.name}"></font><br/>
            用户密码：<input type="password" name="password" /><font color="red" th:errors="${users.password}"></font><br/>
```

```
            用户年龄：<input type="text" name="age" /><font color="red"
th:errors="${users.age}"></font><br/>
            <input type="submit" value="OK"/>
        </form>
    </body>
    </html>
```

如果参数的名称需要做改变

```
        /**
         *
         * 如果想为传递的对象更改名称，可以使用@ModelAttribute("aa")这表示当
前传递的对象的 key 为 aa。
         * 那么我们在页面中获取该对象的 key 也需要修改为 aa
         * @param users
         * @return
         */
        @RequestMapping("/addUser")
        public String showPage(@ModelAttribute("aa") Users users){
            return "add";
        }
```

```
    /**
         * 完成用户添加
         *@Valid 开启对 Users 对象的数据校验
         *BindingResult:封装了校验的结果
         */
        @RequestMapping("/save")
        public String saveUser(@ModelAttribute("aa") @Valid Users
users,BindingResult result){
            if(result.hasErrors()){
                return "add";
            }
            System.out.println(users);
            return "ok";
        }
```

```
    <!DOCTYPE html>
    <html>
    <head>
    <meta charset="UTF-8">
    <title>添加用户</title>
    </head>
    <body>
        <form th:action="@{/save}" method="post">
            用户姓名：<input type="text" name="name"/><font color="red"
```

```
th:errors="${aa.name}"></font><br/>
            用户密码：<input type="password" name="password" /><font
color="red" th:errors="${aa.password}"></font><br/>
            用户年龄：<input type="text" name="age" /><font color="red"
th:errors="${aa.age}"></font><br/>
            <input type="submit" value="OK"/>
        </form>
    </body>
    </html>
```

四、 其他校验规则

@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