1. SpringMVC 标签

2. 数据校验

2.1. 数据验证形式

- 数据校验有效防止用户的误输入以及恶意输入,通过数据校验可以防止非法输入进入系统,从而保证系统的健壮性;
- 数据校验分为客户端校验和服务端校验,客户端校验主要是过滤正常用户的误操作,通常使用JavaScript代码完成;
- 服务端校验是整个应用阻止非法数据的最后防线,在后台应用程序中通过编程来实现。

`SprinMVC` 的 `validation` 框架

`JSR303` 实现校验功能

2.2. Hibernate

0）【注意】：整合hibernate-validator的版本问题，否则各种报错
Spring用4.0.3.RELEASE，hibernate-validator用4.3.0.Final，已验证

1）pom.xml中导入spring-webmvc

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>4.0.3.RELEASE</version>
</dependency>
```

2）pom.xml中导入hibernate-validator

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>4.3.0.Final</version>
</dependency>
```

3）修改SpringMVC的配置文件，增加context、aop、mvc

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd
      http://www.springframework.org/schema/context
      http://www.springframework.org/schema/context/spring-context.xsd
      http://www.springframework.org/schema/aop
      http://www.springframework.org/schema/aop/spring-aop.xsd
      http://www.springframework.org/schema/mvc
      http://www.springframework.org/schema/mvc/spring-mvc.xsd">
```

4）修改SpringMVC的配置文件，配置hibernate校验器并注入

```
<bean id="validator"
class="org.springframework.validation.beanvalidation.LocalValidatorFactoryBean">
  <property name="providerClass" value="org.hibernate.validator.HibernateValidator" />
</bean>
<mvc:annotation-driven validator="validator"></mvc:annotation-driven>
```

5）POJO中添加校验规则

```
public class User {
  @NotBlank(message="用户名不能为空！")
  private String name;
  @NotBlank(message="用户名不能为空！")
  private String pwd;
}
```

6）处理器中编写请求映射方法
@Valid注解表示该参数绑定时进行校验，在【要校验的POJO后边添加】BingdingResult，
也可以使用BingdingResult的子类Errors，处理器自动把校验结果绑定到BindingResult中：

```
@RequestMapping(value="/login2")
public String login2(@Valid User u, Errors errors)  {
  System.out.println("login2222222222222");
  if(errors.hasErrors()) {
    for(ObjectError e : errors.getAllErrors())
      System.out.println(e.getDefaultMessage());
    return "relogin.jsp";
  }
  else
    return "success.jsp";
}
```

如果方法的参数列表中，BingdingResult没有紧跟着写在POJO后面，则报错404

```
@RequestMapping(value="/login3")
public String login3(@Valid User u, String str, Errors errors)  {
  System.out.println("login333333333333");
  if(errors.hasErrors()) {
    for(ObjectError e : errors.getAllErrors())
      System.out.println(e.getDefaultMessage());
    return "relogin.jsp";
  }
  else
    return "success.jsp";
}
```

7）编写relogin.jsp显示错误信息

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<html>
<head></head>
<body>
  <form:form modelAttribute="user" action="login" method="post">
            登录名: <form:input path="name"/>
            <form:errors path="name" cssStyle="color:red"/>
            密码: <form:input path="pwd"/>
            <form:errors path="pwd" cssStyle="color:red"/>
            <input type="submit" value="登录" />
  </form:form>
</body>
```

2.3. JSR303

注解	功能
@Null	验证对象是否为空
@NotNull	验证对象是否不为空
@AssertTrue	验证Boolean对象是否为true
@AssertFalse	验证Boolean对象是否为false
@Max(value)	验证Number和String 是否小于指定值
@Min(value)	验证Number和String 是否大于指定值
@DecimalMax(value)	不能大于约束中指定的值