Spring MVC文件上传

单文件上传

- pom.xml

```
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>1.3.2</version>
</dependency>

<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.2.1</version>
</dependency>
```

- springmvc.xml

```
<bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <!-- 处理中文名乱码 -->
    <property name="defaultEncoding" value="utf-8"></property>
    <!-- 设置多文件上传时，总大小上限，不设置的话没有限制，单位byte -->
    <property name="maxUploadSize" value="1048576"></property>
    <!-- 设置单文件的大小上限 -->
    <property name="maxUploadSizePerFile" value="1048576"></property>
</bean>
```

- JSP
  - input的type设置为file。
  - form表单的method设置为post。
  - form表单的enctype设置为multipart/form-data。

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page isELIgnored="false" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <form action="upload" method="post" enctype="multipart/form-data">
        <input type="file" name="img"/>
```

```
        <input type="submit" value="上传"/>
    </form>
    <c:if test="${requestScope.filePath!=null}">
        <h1>上传的图片</h1>
        <img src="${requestScope.filePath}" width="300px">
    </c:if>
</body>
</html>
```

- FileHandler

```
@PostMapping(value = "upload")
public String upload(@RequestParam("img") MultipartFile multipartFile,
HttpServletRequest request){
    if(multipartFile.getSize()>0){
        String path = request.getServletContext().getRealPath("file");
        String fileName = multipartFile.getOriginalFilename();
        File file = new File(path,fileName);
        try {
            multipartFile.transferTo(file);
            request.setAttribute("filePath","file/"+fileName);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    return "upload";
}
```

多文件上传

- JSP

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page isELIgnored="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <form action="uploads" method="post" enctype="multipart/form-data">
        图片1: <input type="file" name="img"/><br/>
        图片2: <input type="file" name="img"/><br/>
        图片3: <input type="file" name="img"/><br/>
        <input type="submit" value="上传"/>
    </form>
    <c:if test="${requestScope.filePaths!=null}">
        <h1>上传的图片</h1>
        <c:forEach items="${requestScope.filePaths}" var="filePath">
```

```
            <img src="${filePath}" width="300px"/>
        </c:forEach>
    </c:if>
</body>
</html>
```

- FileHandler

```
@PostMapping("uploads")
public String uploads(@RequestParam("img") MultipartFile[]
multipartFiles,HttpServletRequest request){
    List<String> filePaths = new ArrayList<>();
    for (MultipartFile multipartFile:multipartFiles){
        if(multipartFile.getSize()>0){
            String path = request.getServletContext().getRealPath("file");
            String fileName = multipartFile.getOriginalFilename();
            File file = new File(path,fileName);
            try {
                multipartFile.transferTo(file);
                filePaths.add("file/"+fileName);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    request.setAttribute("filePaths",filePaths);
    return "uploads";
}
```

文件下载

- JSP，使用超链接，下载之前上传的文件。

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <a href="download?fileName=1.png">下载图片</a>
</body>
</html>
```

- 业务方法

```
@GetMapping("download")
```

```
public void downloadFile(String fileName, HttpServletRequest request,
HttpServletResponse response){
    if(fileName!=null){
        String path = request.getServletContext().getRealPath("file");
        File file = new File(path,fileName);
        OutputStream outputStream = null;
        if(file.exists()){
            //设置下载文件
            response.setContentType("application/force-download");
            //设置文件名
            response.setHeader("Content-
Disposition","attachment;filename="+fileName);
            try {
                outputStream = response.getOutputStream();
                outputStream.write(FileUtils.readFileToByteArray(file));
                outputStream.flush();
            } catch (IOException e) {
                e.printStackTrace();
            }finally {
                try {
                    outputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

上述是Spring MVC框架对文件上传和下载的支持，文件上传和下载底层是通过IO流完成的，上传就是将客户端的资源通过IO流写入服务端，下载恰好相反，将服务端资源通过IO流写入客户端。Spring MVC提供了一套完善的上传下载机制，可以有效地简化开发步骤。

Spring MVC数据校验

Spring MVC提供了两种数据校验的方式：

- 基于Validator接口
- 使用Annotation JSR-303标准进行校验

基于Validator接口的方式需要自定义Validator验证器，每一条数据的验证规则需要自己手动完成；使用Annotation JSR-303标准则不需要自定义验证器，直接通过注解的方式可以在实体类中添加每个属性的校验规则，这种方式更加方便，实际开发中推荐使用。

基于Validator接口

- 实体类 Student

```
package com.southwind.entity;

public class Student {
    private String name;
    private String password;

    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;
    }
}
```

- 自定义校验器StudentValidator，实现Validator接口，实现其抽象方法。

```
package com.southwind.validator;

import com.southwind.entity.Student;
import org.springframework.validation.Errors;
import org.springframework.validation.ValidationUtils;
import org.springframework.validation.Validator;

public class StudentValidator implements Validator {

    @Override
    public boolean supports(Class<?> clazz) {
        return Student.class.equals(clazz);
    }

    @Override
    public void validate(Object target, Errors errors) {
        ValidationUtils.rejectIfEmpty(errors,"name",null,"姓名不能为空");
        ValidationUtils.rejectIfEmpty(errors,"password",null,"密码不能为空");
    }
}
```

- 业务方法login参数列表中的@Validated表示参数student是需要校验的对象，BindingResult用来存储错误信息，两者缺一不可，并且必须挨着写，中间不能有其他参数。

```
package com.southwind.controller;

import com.southwind.entity.Student;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class ValidateHandler {

    @GetMapping("login")
    public String login(Model model){
        model.addAttribute(new Student());
        return "login";
    }

    @PostMapping("login")
    public String login(@Validated Student student, BindingResult
bindingResult){
        if(bindingResult.hasErrors()){
            return "login";
        }
        return "index";
    }
}
```

- springmvc.xml中配置validator。

```
<!-- 基于Validator的验证器 -->
<mvc:annotation-driven validator="studentValidator"></mvc:annotation-driven>
<bean id="studentValidator"
class="com.southwind.validator.StudentValidator"></bean>
```

- login.jsp

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
```

```
    <h1>学生登陆</h1>
    <form:form modelAttribute="student" action="login" method="post">
        学生姓名:<form:input path="name"></form:input><form:errors
path="name"></form:errors><br/>
        学生密码:<form:input path="password"></form:input><form:errors
path="password"></form:errors><br/>
        <input type="submit" value="提交"/>
    </form:form>
</body>
</html>
```

Annotation JSR-303标准

使用Annotation JSR-303标准进行验证，需要导入支持这种标准的jar包，HibernateValidator。

标准详解：

- @Null 限制只能为null
- @NotNull 限制必须不为null
- @AssertFalse 必须为false
- @AssertTrue 必须为true
- @DecimalMax(value) 必须小于等于指定值
- @DecimalMin(value) 必须大于等于指定值
- @Digits(integer,fraction) 必须是小数，且整数部分的位数不能超过integer，小数部分的位数不能超过fraction
- @Future 必须是一个未来的日期
- @Max(value) 限制必须为一个不大于指定值的数字
- @Min(value) 限制必须为一个不小于指定值的数字
- @Past 必须是一个过去的日期
- @Pattern(value) 必须满足指定的正则表达式
- @Size(max,min) 字符长度必须在min到max之间
- @NotEmpty 验证不能为空，注意与@NotNull
- @NotBlank 验证不能为空，不同于@NotNull和@NotEmpty，@NotBlank只应用在字符串的验证
- @Email 验证是否符合邮箱格式

具体使用

- pom.xml中添加Hibernate Validator依赖。

```
<!-- JSR-303 -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>5.1.3.Final</version>
</dependency>
<dependency>
    <groupId>javax.validation</groupId>
    <artifactId>validation-api</artifactId>
```

```
    <version>1.1.0.Final</version>
</dependency>
<dependency>
    <groupId>org.jboss.logging</groupId>
    <artifactId>jboss-logging</artifactId>
    <version>3.1.1.GA</version>
</dependency>

<!-- JDK9以上的需要添加如下依赖 -->
<dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.3.0</version>
</dependency>
<dependency>
    <groupId>com.sun.xml.bind</groupId>
    <artifactId>jaxb-impl</artifactId>
    <version>2.3.0</version>
</dependency>
<dependency>
    <groupId>com.sun.xml.bind</groupId>
    <artifactId>jaxb-core</artifactId>
    <version>2.3.0</version>
</dependency>
<dependency>
    <groupId>javax.activation</groupId>
    <artifactId>activation</artifactId>
    <version>1.1.1</version>
</dependency>
```

- 创建实体类User，通过注解的方式给属性指定校验规则。

```
package com.southwind.entity;

import org.hibernate.validator.constraints.Email;
import org.hibernate.validator.constraints.NotEmpty;

import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;

public class User {
    @NotEmpty(message = "用户名不能为空")
    private String username;

    @Size(min = 6,max = 20,message = "密码长度为6-20")
    private String password;

    @Email(regexp = "^[a-zA-Z0-9_.-]+@[a-zA-Z0-9-]+(\\.[a-zA-Z0-9-]+)*\\.[a-
zA-Z0-9]{2,6}$",message = "请输入正确的邮箱格式")
```

```
    private String email;

    @Pattern(regexp = "^((13[0-9])|(14[5|7])|(15(0-3)|[5-9]))|(18[0,5-
9]))\\\\d{8}$",message = "请输入正确的电话号码")
    private String phone;
}
```

- 业务方法中使用@Valid来绑定校验对象。

```
@GetMapping("register")
public String register(Model model){
    model.addAttribute(new User());
    return "register";
}

@PostMapping("register")
public String register(@Valid User user,BindingResult bindingResult){
    if(bindingResult.hasErrors()){
        return "register";
    }
    return "index";
}
```

- springmvc.xml

```
<mvc:annotation-driven></mvc:annotation-driven>
```

- register.jsp

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%--
  Created by IntelliJ IDEA.
  User: southwind
  Date: 2019-01-24
  Time: 21:59
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <form:form modelAttribute="user" action="register" method="post">
        用户名: <form:input path="username"></form:input><form:errors
path="username"></form:errors><br/>
        密码: <form:input path="password"></form:input><form:errors
path="password"></form:errors><br/>
```

```
        邮箱: <form:input path="email"></form:input><form:errors
path="email"></form:errors><br/>
        电话: <form:input path="phone"></form:input><form:errors
path="phone"></form:errors><br/>
        <input type="submit" value="注册"/>
    </form:form>
</body>
</html>
```