SpringMVC文件上传

Spring MVC 为文件上传提供了直接的支持，这种支持是通 过即插即用的 MultipartResolver 实现的。

Spring 用 Jakarta Commons FileUpload 技术实现了一个 MultipartResolver 实现类：CommonsMultipartResovler

Spring MVC 上下文中默认没有装配 MultipartResovler，因 此默认情况下不能处理文件的上传工作，如果想使用 Spring 的文件上传功能，需现在上下文中配置 MultipartResolver

配置 MultipartResolver

导入jar

为了让 CommonsMultipartResovler 正确工作，必须先 将

Jakarta Commons FileUpload 及 Jakarta Commons io 的类包添加到类路径下。

```
commons-fileupload-1.3.2
commons-io-2.4.jar
```

配置

```
<!-- SpringMVC文件上传 -->
<bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <!--请求的编码格式必须和用户JSP的编码一致，以便正确读取表单中的内容。
    uploadTempDir:文件上传过程中的临时目录，上传完成后，临时文件会自动删除
    maxUploadSize:设置文件上传大小上限（单位为字节）
      -->
    <property name="defaultEncoding" value="UTF-8" />
    <property name="maxUploadSize" value="1024000"/>
    <property name="uploadTempDir" value="upload/temp"></property>
</bean>
```

defaultEncoding: 必须和用户 JSP 的 pageEncoding 属性 一致，以便正确解析表单的内容

文件上传示例

```
MultipartFile 常用用法
byte[] getBytes()    //获取文件数据
String getContentType()//获取文件MIME类型
InputStream getInputStream()//后去文件流
String getName() //获取表单中文件组件的名字
String getOriginalFilename() //获取上传文件的原名
long getSize()  //获取文件的字节大小，单位byte
boolean isEmpty() //是否为空
void transferTo(File dest) //将该文件保存到一个目标文件中。
```

```
    <h1>文件上传示例</h1>
    <form action="file/upload" enctype="multipart/form-data" method="post">
        上传文件: <input type="file" name="file"><br/>
        <input type="submit" value="上传" />
    </form>
```

```
@RequestMapping("upload")
public String fileUpload(@RequestParam("file")MultipartFile file,HttpServletRequest
request) throws Exception{
    /**
         * MultipartFile 常用用法
            byte[] getBytes()    //获取文件数据
            String getContentType()//获取文件MIME类型
            InputStream getInputStream()//后去文件流
            String getName() //获取表单中文件组件的名字
            String getOriginalFilename() //获取上传文件的原名
            long getSize()  //获取文件的字节大小，单位byte
            boolean isEmpty() //是否为空
            void transferTo(File dest) //将该文件保存到一个目标文件中。
         */
    //上传到指定目录中
    String path=request.getServletContext().getRealPath("/upload/");
    System.out.println(path);
    String path1=request.getSession().getServletContext().getRealPath("/upload/");
    System.out.println(path1);

    if(!file.isEmpty()){
        File f=new File(path+file.getOriginalFilename());
        file.transferTo(f);
    }

    return "/pages/upload.jsp";
}
```

文件下载示例

默认情况下，直接通过a标签 请求即可下载文件。但是如果该文件的文件名为中文，在早些的浏览器上就会导致下载失败；如果使用最新的Firefox、Chrome、Opera、Safari则都可以正常下载文件名为中文的文件.

SpringMVC提供了一个ResponseEntity类型，使用它可以很方便地定义返回的HttpHeaders和HttpStatus

```
//文件下载示例
@RequestMapping("download")
public ResponseEntity<byte[]> download(HttpServletRequest request) throws IOException{
    //指定对应下载文件的路径
    String path=request.getSession().getServletContext().getRealPath("/upload");
    String fileName=path+File.separator+"20180125040256中文文档.txt";
    //请求体
    File file=new File(fileName);
    System.out.println(file);
```

```
    //apache对java io封装提供的工具类
    byte[] body=FileUtils.readFileToByteArray(file);
    //请求头
    HttpHeaders  headers=new HttpHeaders();
    //通知浏览器以attachment（下载方式）打开图片
     //下载显示的文件名，解决中文名称乱码问题
            String downloadFileName = new String("中文文档.txt".getBytes("UTF-8"),"iso-8859-
1");
    headers.add("Content-Disposition","attachment;filename="+downloadFileName);
    //请求码
    HttpStatus status=HttpStatus.OK;
    //请求体、请求头和状态码
    return new ResponseEntity<byte[]>(body, headers, status);
}
```

批量文件上传示例

```
// 批量文件上传
@RequestMapping("demo2")
public String uploadDemo2(@RequestParam("file") MultipartFile files[], HttpServletRequest
        throws IllegalStateException, IOException {
    //根据时间生成不同名称，避免重名
    SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddhhmmss");
    String real = sdf.format(new Date());
    //当前系统中指定上传路径
    String path = request.getSession().getServletContext().getRealPath("/upload/");
    for (MultipartFile file : files) {
        if (file.getSize() > 0) {
            File f = new File(path + real + file.getOriginalFilename());
            file.transferTo(f);
        }
    }
    return "/test.jsp";
}
```

Ajax文件上传 后台/前台

传统的form表单提交会导致页面刷新，但是在有些情况下，我们不希望页面被刷新，这种时候我们需要使用Ajax的方式进行

问题1：传统ajax只能传递一般的参数，上传文件的文件流是无法被序列化并传递 XMLHttpRequest Level 2添加了一个新的接口FormData.利用FormData对象,我们可以通过JavaScript用一些键值对来模拟一系列表单控件 var formData = new FormData(); **Jquery为例：** 需要设置关键是设置：processData 和 contentType

```
function ajaxUploadFile(){
    var formData=new FormData($("#form1")[0]);
        $.ajax({
            type:"post",
            url:"file/uploadFile",
            data:formData,
            cache: false,  //不需要缓存
            contentType: false,   // 不设置请求头
```

```
            processData: false,   //不处理发送数据
            success:function(msg){
              $("#imgId").attr("src",msg);//图片显示
        }
    });
}
```

```
 // ajax文件上传
@ResponseBody
@RequestMapping("uploadFile")
public String uploadFile(@RequestParam("file") MultipartFile file,HttpServletRequest request)
    System.out.println(file.getName());
    System.out.println(file.getOriginalFilename());
    System.out.println(file.getContentType());
     //根据时间来为上传的文件命名
    SimpleDateFormat sdf=new SimpleDateFormat("yyyyMMddhhmmss");
    String timeFileName=sdf.format(new Date());
    String fName=file.getOriginalFilename();
    String realName=fName.substring(fName.lastIndexOf("."),fName.length());
    //得到上传路径位置
    String path=request.getServletContext().getRealPath("/uploadFile/");
    if(!file.isEmpty()){
        file.transferTo(new File(path+timeFileName+realName));
    }
    //返回文件上传的位置和名称，用于前端直接显示
    return "uploadFile/"+timeFileName+realName;
}
```