Springmvc 处理文件上传

**预备知识：**

表单提交： Html标签 form

例子： <form action="form\_action.asp" method="get">

<p>First name: <input type="text" name="fname" /></p>

<p>Last name: <input type="text" name="lname" /></p>

<input type="submit" value="Submit" />

</form>

标签属性：

Action：表单提交的服务器地址url

Method：表单提交方式，get、post、delete

Ectype: 表单提交数据前编码方式，三种编码类型（表单提交中ectype只有这三种）

application/x-www-form-urlencoded：默认编码，空格转换为 "+" 加号，特殊符号转换为 ASCII HEX 值。

multipart/form-data：不进行编码，上传文件时使用该方式，并且只能使用post方式，否则无法上传文件。

text/plain 空格转换为+，其他不编码

**补充：**

常见的MIME类型如下：（请求头或响应头中的content-type和accept可能值）

text/html ： HTML格式

text/plain ：纯文本格式

text/xml ： XML格式

image/gif ：gif图片格式

image/jpeg ：jpg图片格式

image/png：png图片格式

以application开头的媒体格式类型：

application/xhtml+xml ：XHTML格式

application/xml ： XML数据格式

application/atom+xml ：Atom XML聚合格式

application/json ： JSON数据格式

application/pdf ：pdf格式

application/msword ： Word文档格式

application/octet-stream ： 二进制流数据（如常见的文件下载）

application/x-www-form-urlencoded ： <form encType=””>中默认的encType，form表单数据被编码为key/value格式发送到服务器（表单默认的提交数据的格式）

另外一种常见的媒体格式是上传文件之时使用的：

multipart/form-data ： 需要在表单中进行文件上传时，就需要使用该格式

**请求过程：（单纯为springmvc控制器映射的处理过程）**

请求根据指定url去匹配处理器，也就是匹配requestmapping,通过验证path（url）、method(请求方式)、consumes（请求的content-type（请求头）是否匹配）、produces（请求的accept（请求头）是否匹配）、params（请求参数）、headers（请求头）等属性，找到对应的映射方法，在这之前的验证如果出错会返回4XX状态码，在进入方法后出错会返回5XX错误。

**springmvc 接受浏览器表单数据上传（文件）**

* **通用情况：**

**配置文件：使用通用文件解析器**

*MultipartResolver 中multipart是指多部分解析，在post提交时请求实体中对表单上传的数据是根据字段分成多个部分在进行传输的，而不是指多个文件。*

|  |
| --- |
| <bean id=*"multipartResolver"该解析器只能指定入口参数为part*  class=*"org.springframework.web.multipart.commons.CommonsMultipartResolver"*>  <property name=*"defaultEncoding"* value=*"utf-8"*></property>  <!-- byte -->  <property name=*"maxUploadSize"* value=*"10485760"*></property>  <!-- byte -->  <property name=*"maxInMemorySize"* value=*"102400"*></property>  </bean> |

1. 页面中表单标签属性 ectype（也就是请求中的content-type）必须指定为multipart/form-data
2. @Requestmapping注解属性中consumers可控制接受指定content-type的请求，默认接受所有类型。
3. 参数使用@requestparam指定multipartFile（spring自定义处理文件上传类）入口参数，在方法中获取文件名、字节码等
4. 使用multipartfile获取字节码，操作文件数据

* **Servlet3.0环境下应用：**

|  |
| --- |
| <bean id=*"multipartResolver" 该解析器只能指定入口参数为part*  class=*"org.springframework.web.multipart.support.StandardServletMultipartResolver"*>  </bean> |

1. 表单中ectype属性指定为multipart/form-data
2. Consumers与请求的content-type一致
3. 在注册前端控制器时指定MultipartConfig（包括限制文件大小等配置）
4. 参数中使用@requestparam指定入口参数类型为 javax.servlet.http.Part

注意：可上传文件数组

**客户端发起的文件上传请求**

在使用restful服务的场景下，非浏览器的客户端也可以直接提交多部分文件请求，和浏览器表单不同的是，数据可能更加复杂了，可能其中包含json、文件、xml等内容类型。

例如：

浏览器提交表单：

|  |
| --- |
| ------WebKitFormBoundaryzuM66baXK1zu7k2x  Content-Disposition: form-data; name="file[0]"; filename="捕获.PNG"  Content-Type: image/png  ------WebKitFormBoundaryzuM66baXK1zu7k2x  Content-Disposition: form-data; name="file[1]"; filename="博客列表.png"  Content-Type: image/png  ------WebKitFormBoundaryzuM66baXK1zu7k2x  Content-Disposition: form-data; name="type"  test （浏览器文本字段是没有类型的）  ------WebKitFormBoundaryzuM66baXK1zu7k2x-- |

Restful服务提交多部分文件请求：

|  |
| --- |
| POST /someUrl  Content-Type: multipart/mixed  --edt7Tfrdusa7r3lNQc79vXuhIIMlatb7PQg7Vp  Content-Disposition: form-data; name="meta-data"  Content-Type: application/json; charset=UTF-8  Content-Transfer-Encoding: 8bit  {  "name": "value"  }  --edt7Tfrdusa7r3lNQc79vXuhIIMlatb7PQg7Vp  Content-Disposition: form-data; name="file-data"; filename="file.properties"  Content-Type: text/xml  Content-Transfer-Encoding: 8bit  ... File Data ... |

这种情况下

使用@requestpart 注解MetaData类来指定复杂类型（json）的对应字段名，文件则可以使用multipartFile接收。

例如：Restful服务提交的文件请求中后台代码

|  |
| --- |
| @RequestMapping(path = "/someUrl", method = RequestMethod.POST)  public String onSubmit(@RequestPart("meta-data") MetaData metadata, @RequestPart("file-data") MultipartFile file) {  // ...  } |

客户端提交文件请求待测试！！！！！！！！！！！！！！！！！！！

传统servlet处理表单上传文件

Servlet3.0 规范后提供了对表单中文件类型的数据支持：

|  |
| --- |
| /\*//请求中content-type的MIME 类型必须是 multipart/form-data才能使用  Part getPart(String name)  Collection<Part> getParts()\*/  @WebServlet(urlPatterns="/fileupload")  @MultipartConfig(maxFileSize=1000000)  **public** **class** FileUploadTestServlet **extends** HttpServlet{  @Override  **protected** **void** doPost(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {  FileOutputStream out = **null**;  Part partImage = req.getPart("image");  String submittedFileName = partImage.getSubmittedFileName();  InputStream inputStream = partImage.getInputStream();    **try** {  **if**(partImage != **null** && partImage.getSubmittedFileName()!= **null**){  File file = **new** File("C:/上传图片——"+partImage.getSubmittedFileName());  **if**(!file.exists()){  file.createNewFile();  }  out = **new** FileOutputStream(file);  **byte**[] b = **new** **byte**[1024];  **int** len = 0;  **while**((len = inputStream.read(b)) != -1){  out.write(b, 0, len);  }  }  } **catch** (Exception e) {  e.printStackTrace();  }**finally**{  **if**(out != **null**){  out.close();  }  **if**(inputStream != **null**){  inputStream.close();  }  }  } |