# TaCloud410大文件上传下载

-技术支持：钟志博

## 一.大文件上传

框架提供了uploader标签是基于webuploader 封装的,支持分片,提交的时候分片存储,然后在成功回调里面发起合并请求,将后台分片的文件合并. 但需要修改spring-mvc.xml 文件上传大小限制配置maxUploadSize

### 1.前台标签

<**ta:uploader url="uploaderController!upload.do" id="uploadfile" autoSubmit="true" multiple="false" threads="1" chunkedOptions="{chunked:true,chunkSize:5242880,chunkRetry:2}"**

**custom="true" fileQueued="fnFileQueued" uploadSuccess="fnUploadSuccess" columnWidth="0.3"**/>

属性说明:

url: 上传地址，后台action中需定义File成员变量及getset方法（File数组-多文件）

id: 设置组件id

autoSubmit:自动提交

multiple:是否为多文件

threads: 上传并发数。允许同时最大上传进程数

chunkedOptions: 设置分片上传属性，不设置则默认不使用分片

custom: 自定义模式，自定义各种接口，不按照框架默认

fileQueued: 当有文件被添加进队列的时候触发回调

### 2.前台方法

**function** *fnFileQueued*(file){  
 ***Base***.setValue(**"uploadfileName"**, file.**name**);  
}  
**function** *fnUploadSuccess*(file,response) {  
 **var** opt={};  
 opt.**filename**=file.**name**;  
 opt.**chunks**=***Math***.ceil(file.**size**/(5\*1024\*1024));  
 ***Base***.submit(**""**,**"uploaderController!together.do"**,opt);  
}

### 3.后台方法

@RequestMapping(**"uploaderController!upload.do"**)  
**public** String sync(  
 @RequestPart(**"uploadfile"**) MultipartFile file,  
 HttpServletRequest request, HttpServletResponse response)  
 **throws** Exception {  
  
 String chunk = request.getParameter(**"chunk"**); *// 当前分片* String chunks = request.getParameter(**"chunks"**); *// 分片总计* String realPath = request.getSession().getServletContext()  
 .getRealPath(**"/Upload/"**);  
 String Ogfilename = file.getOriginalFilename();  
 String fileName = Ogfilename.substring(0,Ogfilename.lastIndexOf(**"."**));  
 File tempFile;  
 **if** (chunks != **null**) {  
 tempFile = **new** File(realPath+fileName+File.***separator***+**"temp"**, fileName+chunk);  
 } **else** {  
 tempFile = **new** File(realPath+fileName+File.***separator***+**"temp"**, fileName+**"0"**);  
 }  
 *//copy* FileUtils.*copyInputStreamToFile*(file.getInputStream(), tempFile);  
 **return *JSON***;  
}  
  
@RequestMapping(**"uploaderController!together.do"**)  
**public** String together(@RequestParam(**"filename"**) String filename, @RequestParam(**"chunks"**) **int** chunks, HttpServletRequest request, HttpServletResponse response) **throws** Exception{  
 String realPath = request.getSession().getServletContext()  
 .getRealPath(**"/Upload/"**);  
  
 String fileName = filename.substring(0,filename.lastIndexOf(**"."**));  
  
 System.***out***.println(**"chunks="**+chunks+**";fileName="**+fileName);  
 */\*\*  
 \* 进行文件合并  
 \*/* File file = **new** File(realPath+fileName+File.***separator***+**"temp"**);  
 */\*\*  
 \* 判断分片数量是否正确  
 \*/* **if**(file.list().**length** != chunks ){  
 **throw new** AppException(**"大文件分片提交分片数不正确"**);  
 }  
 */\*\*  
 \* 进行文件合并  
 \*/* File newFile = **new** File(realPath+fileName+File.***separator***+filename);  
 FileOutputStream outputStream = **new** FileOutputStream(newFile, **true**);*//文件追加写入* **byte**[] byt = **new byte**[10\*1024\*1024];  
 **int** len;  
 FileInputStream temp = **null**;*//分片文件* **for**(**int** i = 0 ; i<chunks ; i++){  
 temp = **new** FileInputStream(**new** File(realPath+fileName+File.***separator***+**"temp"**, fileName+i));  
 **while**((len = temp.read(byt))!=-1){  
 System.***out***.println(len);  
 outputStream.write(byt, 0, len);  
 }  
 temp.close();  
 }  
 */\*\*  
 \* 当所有追加写入都写完 才可以关闭流  
 \*/* outputStream.close();  
 **return *JSON***;  
}

### 4.excel文件解析

框架提供了ExcelFileUtilsNew.java类,对上传的excel类型文件进行解析.

提供了如下方法解析csv格式文件

**public static** List getCSVInputStream2ObjectList(InputStream inputStream,  
 String fieldNames, String domainClassName, **int** titleRow) {

提供了如下方法对xls/xlsx文件进行解析

**public static** List getExcelInputStream2ObjectListWithParam(InputStream is, String fieldNames, String domainClassName, **int** rowStart, **int** rowEnd, **int** cellStart, **int** cellEnd)  
 **throws** IllegalInputAppException {

## 二.大文件导出

框架自身的全部导出支持csv格式,且提供了导出csv格式的工具类

### A.框架自身全部导出

#### 前台tag标签

<**ta:dataGridToolPaging url="dataGridController!queryInfo.do" parameter="{'grid6':'grid6'}" selectExpButtons="1,2,3" showExcel="true" pageSize="10" submitIds="text1,grid6" successCallBack="grid6successCallBack" validateForm="grid6validateForm" sqlStatementName="demo.queryAa10" resultType="java.util.HashMap" exportFileName="grid6\_exl" expKeyOrName="false"  
 suffix="csv" exportWithHead="true" exportBean="dao"**/>

属性说明:

url: 数据获取地址,如果showPagingBar显示分页则该项必须

parameter:分页请求参数

selectExpButtons: 导出按钮控制,1导出当前页,2导出选择行,3导出全部.例如selectExpButtons="1,2"表示只显示当前和选择按钮

showExcel: 是否显示导出excel按钮

pageSize: 设置默认分页大小

submitIds: 提交数据ids，包括查询条件的所有id

successCallBack: 分页每次请求获取数据成功时的回调函数，默认传入参数（data）为当前请求返还的表格数据

validateForm: 分页查询中的校验函数，如果校验不通过，不查询返回数据，该属性为一个function。返回true为通过，false为不通过'

sqlStatementName: 导出全部数据时的必填项，sql查询，例如:sqlStatementName="ac01.getList"

resultType: 导出全部数据时的必填项，sql查询时的返回类型

exportFileName: excel导出的文件名，默认 export

expKeyOrName: true/false,导出时是导出码值还是描述值,默认true,表示导出码值,false表示导出描述值

suffix: 文件后缀名xlsx/xls/csv

exportWithHead: 导出是否有表头,true/false,默认true

exportBean: 全部导出使用的dao的beanid,默认为dao

#### 注意事项:

1.全部导出需要showExcel为true,且selectExpButtons中有3

2.全部导出需要设置sqlStatementName和resultType

3.需要大数据导出的时候,需要设置suffix为csv

4.需要导出其他数据库数据时,需要将exportBean设置为对应数据源的dao的beanid

### B.工具类说明

ExcelFileUtilsNew.java工具类提供了如下两个方法用于导出csv格式文件

**public static void** exportDaoHandlerDataToHttpDownLoadCsv(String[] columnsName, **final** String[] columns, String csvName, IDao dao, String sqlId, Object object, **final** HttpServletResponse response, **final** Map<String, Map<String, String>> mapCollection, Boolean withHead) **throws** Exception {

**public static void** exportDaoHandlerDataToHttpDownLoadCsv(String[] columnsName, **final** String[] columns, String csvName, IDao dao, String sqlId, Object object, **final** HttpServletResponse response) **throws** Exception {

参数说明:

columnsName:为列的名称数组

columns:为列的字段名称数组

csvName:为导出文件名

dao:为导出数据所对应的数据源的dao

sqlId:为前台的sqlStatementName

object:为map形式的参数

mapCollection:为所需转码的列的码值

withHead:在导出文件中是否带第一行列的名称