注：该示例用的不是spring自带的MultipartFile上传，而是commons-fileupload 有可能会出错，请注意，若用MultipartFile，请看另一篇笔记，基于这篇修改的。

最近项目上用到文件分片上传，于是找到了百度的一个开源前端控件webuploader。 于是尝试使用。

下载下来后，它提供的服务器端示例代码是php版的，那么Java版的呢？

其实，上传文件都是按照rfc1867标注来的， 只是分段上传需要在前端多做点事情。分段上传原理其实就是在前端使用JavaScript对文件进行分割成不同小块，然后每次ajax请求就post一小块，直到全部收到为止。

但是，为了确保后端能判断文件是否完整的收到，需要得知当前是第几块，一共多少块，每个分段的大小是多少（前后端同学约定好吧），webuploader是有把这些参数传给后端， 但文档里没说明参数名称啊，看了其他后端example，并且自己做了实验才知道。

*参数：*

name 文件名

chunks 一共有多少分片

chunk 当前分片是第几片file 文件对象

**重点来了:后端接收到这些参数应该怎么处理？ 之前看过一个example ，是把每个分片文件都暂时存起来，命名为 文件名.part\_1, 文件名.part\_2, 文件名.part\_3...文件名part\_n, 每次都从1到总分片个数(注:这个值就是chunks的值),**

**遍历这些文件是否存在.如果都存在说明全部上传完成了,则再循环一遍,把所有分段都合并到一个文件里. 这么做虽然是可以,但是如果文件很大, 最后一个分片到达的时候响应可能会很慢,效率低下.**

**那么应该怎么解决呢?考虑了一会, 联想到,以前使用迅雷之类的工具下载,除下载下来的文件以外,还会有一个额外的文件用来存放下载之类的信息.**

**受到这个启发, 我决定这么设计: 每当第n个分片到达时(注:n的值其实就是收到的chunk的值), 使用java的随机文件读写类RandomAccessFile, 定位到 n\*分片大小(注:每个分片大小跟前端约定好的)的位置**

long offset = chunkSize \* param.getChunk();

//定位到该分片的偏移量

accessTmpFile.seek(offset);

**然后写入分片内容.**

//写入该分片数据

accessTmpFile.write(param.getFileItem().get());

**同时,往一个配置文件,*暂且命名为 文件名.conf 设置长度为chunks的值, 也就是分片个数.***

accessConfFile.setLength(param.getChunks());

**然后往第n个位置写入一个Byte.MAX\_VALUE,**

accessConfFile.seek(param.getChunk());

accessConfFile.write(Byte.MAX\_VALUE);

**因为写入的单位就是字节,所以我这么操作就相当于在第n个字节里写入全1的状态. 然后检查,从0到chunks开始每一个字节进行与操作, 一旦到第n个字节发现与运算的结果不是全1(Byte.MAX\_VALUE)那么就说明这个文件的第n个部分没有传输完成.**

**如果conf文件0到chunks的位置全部进行与运算的最后结果还是Byte.MAX\_VALUE,那么就说明这个文件已经传输完成,该干嘛就干嘛..**

//completeList 检查是否全部完成,如果数组里是否全部都是(全部分片都成功上传)

byte[] completeList = FileUtils.readFileToByteArray(confFile);

byte isComplete = Byte.MAX\_VALUE;

for (int i = 0; i < completeList.length && isComplete==Byte.MAX\_VALUE; i++) {

//与运算, 如果有部分没有完成则 isComplete 不是 Byte.MAX\_VALUE

isComplete = (byte)(isComplete & completeList[i]);

System.out.println(prefix + "check part " + i + " complete?:" + completeList[i]);

}

if (isComplete == Byte.MAX\_VALUE) {

System.out.println(prefix + "upload complete !!");

}

**其实还有另一种想法是,前端传来该文件的md5码,然后后端每次接收到都算一次md5码,如果一致则说明上传成功,但是效率应该也不够上面的好,于是没实现. 谁有更好的想法可以留言哈.**

现在可以开始例子：

在前端webuploader source的examples/image-upload/upload.js 中可以看到

复制代码

复制代码

// 实例化 uploader = WebUploader.create({ pick: { id: '#filePicker', label: '点击选择图片' }, formData: { uid: 123 }, dnd: '#dndArea', paste: '#uploader', swf: '../../dist/Uploader.swf', chunked: false, chunkSize: 512 \* 1024, server: '../../server/fileupload.php', // runtimeOrder: 'flash', // accept: { // title: 'Images', // extensions: 'gif,jpg,jpeg,bmp,png', // mimeTypes: 'image/\*' // }, // 禁掉全局的拖拽功能。这样不会出现图片拖进页面的时候，把图片打开。 disableGlobalDnd: true, fileNumLimit: 300, fileSizeLimit: 200 \* 1024 \* 1024, // 200 M fileSingleSizeLimit: 50 \* 1024 \* 1024 // 50 M });

复制代码

复制代码

chunked 被设置为false， 改为true就可以分片上传了。

chunkSize这个后端需要用到，所以前后端需要保持一致。

server改成java后端自己定义的上传文件接口的地址，我这里根据后端例子改成了“http://127.0.0.1:8080/file/test-upload2”

下面是 FileUploadAction.java 示例逻辑都在这里 ,注:有可能因为网络缘故不同分片接收到的时间不一致,并不是顺序的,所以加了个原子计数器方便从标注输出信息看到整个流程.

复制代码

复制代码

package cn.thomastest.controller; import cn.thomastest.controller.request.MultipartFileParam; import cn.thomastest.controller.response.DsResponse; import cn.thomastest.util.MultipartFileUploadUtil; import org.apache.commons.io.FileUtils; import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.ResponseBody; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.io.File; import java.io.RandomAccessFile; import java.util.concurrent.atomic.AtomicLong; /\*\* \* FileUpload \* \* @author ThomasWong \* @date 2016/7/6 15:15 \*/ @Controller @RequestMapping("/file") public class FileUploadAction { private static AtomicLong counter = new AtomicLong(0L); @RequestMapping(method = {RequestMethod.POST}, value = {"test-upload2"}) @ResponseBody public DsResponse uploadv2(HttpServletRequest request, HttpServletResponse resp) throws Exception { String prefix = "req\_count:" + counter.incrementAndGet() + ":"; System.out.println(prefix + "start !!!"); DsResponse response = new DsResponse(); //使用 工具类解析相关参数，工具类代码见下面 MultipartFileParam param = MultipartFileUploadUtil.parse(request); System.out.println(prefix + "chunks= " + param.getChunks()); System.out.println(prefix + "chunk= " + param.getChunk()); System.out.println(prefix + "chunkSize= " + param.getParam().get("chunkSize")); //这个必须与前端设定的值一致 long chunkSize = 512 \* 1024; if (param.isMultipart()) { String finalDirPath = "/data0/uploads/"; String tempDirPath = finalDirPath + param.getId(); String tempFileName = param.getFileName() + "\_tmp"; File confFile = new File(tempDirPath, param.getFileName() + ".conf"); File tmpDir = new File(tempDirPath); File tmpFile = new File(tempDirPath, tempFileName); if (!tmpDir.exists()) { tmpDir.mkdirs(); } RandomAccessFile accessTmpFile = new RandomAccessFile(tmpFile, "rw"); RandomAccessFile accessConfFile = new RandomAccessFile(confFile, "rw"); long offset = chunkSize \* param.getChunk(); //定位到该分片的偏移量 accessTmpFile.seek(offset); //写入该分片数据 accessTmpFile.write(param.getFileItem().get()); //把该分段标记为 true 表示完成 System.out.println(prefix + "set part " + param.getChunk() + " complete"); accessConfFile.setLength(param.getChunks()); accessConfFile.seek(param.getChunk()); accessConfFile.write(Byte.MAX\_VALUE); //completeList 检查是否全部完成,如果数组里是否全部都是(全部分片都成功上传) byte[] completeList = FileUtils.readFileToByteArray(confFile); byte isComplete = Byte.MAX\_VALUE; for (int i = 0; i < completeList.length && isComplete==Byte.MAX\_VALUE; i++) { //与运算, 如果有部分没有完成则 isComplete 不是 Byte.MAX\_VALUE isComplete = (byte)(isComplete & completeList[i]); System.out.println(prefix + "check part " + i + " complete?:" + completeList[i]); } if (isComplete == Byte.MAX\_VALUE) { System.out.println(prefix + "upload complete !!"); } accessTmpFile.close(); accessConfFile.close(); } System.out.println(prefix + "end !!!"); return response; } }

复制代码

复制代码

MultipartFileUploadUtil.java 负责从request对象中取出文件以及相应的参数

复制代码

复制代码

package cn.thomastest.util; import cn.thomastest.controller.request.MultipartFileParam; import org.apache.commons.fileupload.FileItem; import org.apache.commons.fileupload.FileItemFactory; import org.apache.commons.fileupload.disk.DiskFileItemFactory; import org.apache.commons.fileupload.servlet.ServletFileUpload; import org.apache.commons.lang.math.NumberUtils; import javax.servlet.http.HttpServletRequest; import java.util.List; /\*\* \* Created by ThomasWong on 2016/9/8. \*/ public class MultipartFileUploadUtil { /\*\* \* 在HttpServletRequest中获取分段上传文件请求的信息 \* @param request \* @return \*/ public static MultipartFileParam parse(HttpServletRequest request) throws Exception { MultipartFileParam param = new MultipartFileParam(); boolean isMultipart = ServletFileUpload.isMultipartContent(request); param.setMultipart(isMultipart); if(isMultipart){ FileItemFactory factory = new DiskFileItemFactory(); ServletFileUpload upload = new ServletFileUpload(factory); // 得到所有的表单域，它们目前都被当作FileItem List<FileItem> fileItems = upload.parseRequest(request); for (FileItem fileItem : fileItems) { System.out.println("field name has:"+fileItem.getFieldName()); if (!"file".equals(fileItem.getFieldName())){ System.out.println("field val has:"+fileItem.getString()); } if (fileItem.getFieldName().equals("id")) { param.setId(fileItem.getString()); } else if (fileItem.getFieldName().equals("name")) { param.setFileName(new String(fileItem.getString().getBytes( "ISO-8859-1"), "UTF-8")); } else if (fileItem.getFieldName().equals("chunks")) { param.setChunks(NumberUtils.toInt(fileItem.getString())); } else if (fileItem.getFieldName().equals("chunk")) { param.setChunk(NumberUtils.toInt(fileItem.getString())); } else if (fileItem.getFieldName().equals("file")) { param.setFileItem(fileItem); param.setSize(fileItem.getSize()); } else{ param.getParam().put(fileItem.getFieldName(), fileItem.getString()); } } } return param; } }

复制代码

复制代码

MultipartFileParam.java 上面取出的参数都暂时存在这个类的实例里

复制代码

复制代码

package cn.thomastest.controller.request; import org.apache.commons.fileupload.FileItem; import java.util.HashMap; /\*\* \* 特殊参数. 分段文件上传 \* Created by ThomasWong on 2016/9/8. \*/ public class MultipartFileParam { //该请求是否是multipart private boolean isMultipart; //任务ID private String id; //总分片数量 private int chunks; //当前为第几块分片 private int chunk; //当前分片大小 private long size = 0L; //文件名 private String fileName; //分片对象 private FileItem fileItem; //请求中附带的自定义参数 private HashMap<String, String> param = new HashMap<>(); public boolean isMultipart() { return isMultipart; } public void setMultipart(boolean multipart) { isMultipart = multipart; } public String getId() { return id; } public void setId(String id) { this.id = id; } public int getChunks() { return chunks; } public void setChunks(int chunks) { this.chunks = chunks; } public int getChunk() { return chunk; } public void setChunk(int chunk) { this.chunk = chunk; } public long getSize() { return size; } public void setSize(long size) { this.size = size; } public String getFileName() { return fileName; } public void setFileName(String fileName) { this.fileName = fileName; } public FileItem getFileItem() { return fileItem; } public void setFileItem(FileItem fileItem) { this.fileItem = fileItem; } public HashMap<String, String> getParam() { return param; } public void setParam(HashMap<String, String> param) { this.param = param; } }