注：前端是使用的百度的webuploader插件实现的分片上传，这个样例最好来处理分为大于1片的文件，在入口处可以检测有没chunk属性来判断是若干片还是只有一片，只有一片的话，直接使用普通方法就好。这里不多赘述。

该方法只是个示例，实际使用应该进行封装。

前端webuploader依赖jquery 版本应高于1.9

前端：

var $list = $("#thelist"),

$list = $('#thelist'),

$btn = $('#ctlBtn'),

state = 'pending',

uploader;

uploader = WebUploader.create({

// swf文件路径

swf: '/webuploader/Uploader.swf',

// 文件接收服务端。

server: 'http://172.16.18.118:8080/upload2.do',

// 选择文件的按钮。可选。

// 内部根据当前运行是创建，可能是input元素，也可能是flash.

pick: {

id: '#picker',

label: '点击选择图片'

},

formData: {

uid: uuid()

},

chunked: true,

chunkSize: 5 \* 1024 \* 1024,

// 不压缩image, 默认如果是jpeg，文件上传前会压缩一把再上传！

resize: false,

// fileNumLimit: 300,

// fileSizeLimit: 200 \* 1024 \* 1024, // 200 M

// fileSingleSizeLimit: 50 \* 1024 \* 1024 // 50 M

});

// 当有文件被添加进队列的时候

uploader.on( 'fileQueued', function( file ) {

$list.append( '<div id="' + file.id + '" class="item">' +

'<h4 class="info">' + file.name + '</h4>' +

'<p class="state">等待上传...</p>' +

'</div>' );

});

// 文件上传过程中创建进度条实时显示。

uploader.on( 'uploadProgress', function( file, percentage ) {

var $li = $( '#'+file.id ),

$percent = $li.find('.progress .progress-bar');

// 避免重复创建

if ( !$percent.length ) {

$percent = $('<div class="progress progress-striped active">' +

'<div class="progress-bar" role="progressbar" style="width: 0%">' +

'</div>' +

'</div>').appendTo( $li ).find('.progress-bar');

}

$li.find('p.state').text('上传中');

$percent.css( 'width', percentage \* 100 + '%' );

});

uploader.on( 'uploadSuccess', function( file ) {

$( '#'+file.id ).find('p.state').text('已上传');

});

uploader.on( 'uploadError', function( file ) {

$( '#'+file.id ).find('p.state').text('上传出错');

});

uploader.on( 'uploadComplete', function( file ) {

$( '#'+file.id ).find('.progress').fadeOut();

});

uploader.on( 'all', function( type ) {

if ( type === 'startUpload' ) {

state = 'uploading';

} else if ( type === 'stopUpload' ) {

state = 'paused';

} else if ( type === 'uploadFinished' ) {

state = 'done';

}

if ( state === 'uploading' ) {

$btn.text('暂停上传');

} else {

$btn.text('开始上传');

}

});

$btn.on( 'click', function() {

if ( state === 'uploading' ) {

uploader.stop();

} else {

uploader.upload();

}

});

function uuid() {

var s = [];

var hexDigits = "0123456789abcdef";

for (var i = 0; i < 36; i++) {

s[i] = hexDigits.substr(Math.floor(Math.random() \* 0x10), 1);

}

s[14] = "4"; // bits 12-15 of the time\_hi\_and\_version field to 0010

s[19] = hexDigits.substr((s[19] & 0x3) | 0x8, 1); // bits 6-7 of the clock\_seq\_hi\_and\_reserved to 01

s[8] = s[13] = s[18] = s[23] = "-";

var uuid = s.join("");

return uuid;

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

<link rel="stylesheet" type="text/css" href="/webuploader/webuploader.css">

</head>

<body>

<div id="uploader" class="wu-example">

<!--用来存放文件信息-->

<div id="thelist" class="uploader-list"></div>

<div class="btns">

<div id="picker">选择文件</div>

<button id="ctlBtn" class="btn btn-default">开始上传</button>

</div>

</div>

<script src="/webuploader/jquery-1.9.1.min.js"></script>

<script type="text/javascript" src="/webuploader/webuploader.js"></script>

<script src="/webuploader/myExtend.js"></script>

</body>

</html>

// TODO 这个可能会有并发问题，比如很大的文件，上传要很久的，没上传完就改了，然后又点，或者没上传完关掉了，（断点续传问题）

@RequestMapping(method = {RequestMethod.POST}, value = {"upload2"})

@ResponseBody

public Map<String, Object> uploadv2(MultipartFileParam param, HttpServletRequest request) throws Exception {

System.out.println(param.getUid());

String prefix = "req\_count:" + counter.incrementAndGet() + ":";

System.out.println(prefix + "start !!!");

//使用 工具类解析相关参数，工具类代码见下面

System.out.println(prefix + "chunks= " + param.getChunks());

System.out.println(prefix + "chunk= " + param.getChunk());

System.out.println(prefix + "chunkSize= " + param.getFile().getSize());

//这个必须与前端设定的值一致

long chunkSize = 5 \* 1024 \* 1024;

WebApplicationContext wac = ContextLoader.getCurrentWebApplicationContext();

String finalDirPath = wac.getServletContext().getRealPath("/").replaceAll("/",File.separator) + "jar" + File.separator;

// String finalDirPath = "/uploads/";

String tempDirPath = finalDirPath;

String tempFileName = param.getName() + "\_tmp";

File tmpDir = new File(tempDirPath);

if (!tmpDir.exists()) {

tmpDir.mkdirs();

}

File confFile = new File(tempDirPath, param.getName() + ".conf");

File tmpFile = new File(tempDirPath, tempFileName);

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.getFile().getBytes());

//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]);

}

accessTmpFile.close();

accessConfFile.close();

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

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

confFile.delete();

// 覆盖，若改名后的文件已存在，则先删掉再改名

File renameFile = new File(finalDirPath, param.getName());

if(renameFile.exists())

renameFile.delete();

tmpFile.renameTo(renameFile);

}

System.out.println(prefix + "end !!!");

Map<String, Object> map = new HashMap<String, Object>();

map.put("status", "ok");

// map.put("msg", jarService.uploadJar(file));

map.put("msg", "ok");

return map;

}

public class MultipartFileParam {

private String uid;

//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;

//文件总大小(单位字节)

private long size = 0L;

//文件名

private String name;

//分片对象

private MultipartFile file;

public String getUid() {

return uid;

}

public void setUid(String uid) {

this.uid = uid;

}

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 getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public MultipartFile getFile() {

return file;

}

public void setFile(MultipartFile file) {

this.file = file;

}

}