(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 110519399 A
(43)申请公布日 2019.11.29

(21)申请号 201910938670.0

(22)申请日 2019.09.30

(71)申请人 山东浪潮通软信息科技有限公司
地址 250100 山东省济南市高新区孙村镇科航路2877号

(72)发明人 王宏达 邵辉

(74)专利代理机构 济南信达专利事务所有限公司 37100
代理人 阚恭勇

(51)Int.Cl.
H04L 29/08(2006.01)

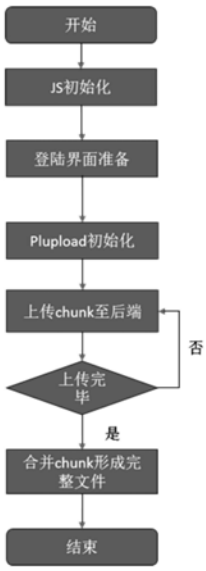
权利要求书1页 说明书6页 附图1页

(54)发明名称

一种基于SpringMVC实现超大文件上传方法

(57)摘要

本发明提供一种基于SpringMVC实现超大文件上传方法，属于文件上传技术领域，本发明基于第三方plupload插件，实现大文件分割成更小的块(Chunk)，基于SpringMVC实现后台接收Chunk流，并根据顺序将上传的Chunk合并，形成原始文件。通过分块上传，可实现：1、分块上传，由大化小，保证浏览器性能；2、可实现断点续传，保证文件上传，保障上传效率。



CN 110519399 A

1.一种基于SpringMVC实现超大文件上传方法，其特征在于，

基于SpringMVC实现后台接收Chunk流，并根据顺序将上传的Chunk合并，形成原始文件。

2.根据权利要求1所述的方法，其特征在于，

基于第三方plupload插件，实现大文件分割成一个以上的的块即Chunk。

3.根据权利要求2所述的方法，其特征在于，

大文件按顺序分块上传，每个块大小一致，且按文件顺序传输。

4.根据权利要求3所述的方法，其特征在于，

首先初始化JS，下载plupload插件，并在项目中引入jQuery.plupload.queue.css，jquery-2.0.0.min.js，plupload.full.min.js，jquery.plupload.queue.js，zh_CN.js文件。

5.根据权利要求4所述的方法，其特征在于，

然后进行前端界面准备，准备上传功能按钮、界面，并实例化一个plupload对象时，即new plupload.Uploader()，需要传入一个对象作为配置参数，并根据需要配置plupload参数。

6.根据权利要求5所述的方法，其特征在于，

所述对象属性包括上传、暂停绑定时间、Chunk大小、支持上传文件的格式。

7.根据权利要求6所述的方法，其特征在于，

最后进行上传功能编写，后端获取plupload对象，通过文件流的方式保存到需要的服务器路径。

8.根据权利要求7所述的方法，其特征在于，

所述获取的plupload对象中包含Chunk、Chunk数量和文件流。

一种基于SpringMVC实现超大文件上传方法

技术领域

[0001] 本发明涉及文件上传技术，尤其涉及一种基于SpringMVC实现超大文件上传方法。

背景技术

[0002] 随着视频网站和大数据应用的普及，特别是高清视频和4K视频应用的到来，超大文件上传已经成为了日常的基础应用需求。但是在很多情况下，无法直接上传超大文件和断点续传，往往在网上找一些简单的程序来实现基本的上传功能，然而在实际使用中会发现，这些基于脚本语言实现的上传功能模块性能很弱，一是不支持2GB以上的内容上传；二是无法支持断点续传；三是效率极低，单台服务器最多支持几十个并发上传连接。

[0003] 超大文件通过浏览器上传时，因为浏览器处理能力及内存不同，会造成浏览器卡死无法响应的现象。

发明内容

[0004] 为了解决以上技术问题，本发明提出了一种基于SpringMVC实现超大文件上传方法，基于SpringMVC+plupload实现超大文件通过浏览器上传、断点续传，plupload可对大文件进行Chunk分割，且支持显示上传进度、图像自动缩略和上传分块，可同时上传多个文件。

[0005] 本发明的技术方案是：

[0006] 一种基于SpringMVC实现超大文件上传方法，基于SpringMVC实现后台接收Chunk流，并根据顺序将上传的Chunk合并，形成原始文件。

[0007] 进一步的，基于第三方plupload插件，实现大文件分割成一个以上的的块即Chunk。超大文件按顺序分块上传，每个块大小一致，且按文件顺序传输。

[0008] 具体如下：

[0009] 1、初始化JS，下载plupload插件，并在项目中需要引入jQuery.plupload.queue.css，jquery-2.0.0.min.js，plupload.full.min.js，jquery.plupload.queue.js，zh_CN.js文件

[0010] 2、前端界面准备，准备上传功能按钮、界面，并实例化一个plupload对象时，也就是new plupload.Uploader()，需要传入一个对象作为配置参数，并根据需要配置plupload参数。对象属性包括上传、暂停绑定时间、Chunk大小、支持上传文件的格式等。

[0011] 3、上传功能编写，后端获取plupload对象，对象中包含Chunk、Chunk数量和文件流，通过文件流的方式保存到需要的服务器路径。

[0012] 本发明的有益效果是

[0013] 可实现超大文件按顺序分块上传，因每个块大小一致，且按文件顺序传输，可实现断点续传，保证文件上传效率和完整。

附图说明

[0014] 图1是本发明的工作流程示意图。

具体实施方式

[0015] 为使本发明实施例的目的、技术方案和优点更加清楚，下面将结合本发明实施例中的附图，对本发明实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例是本发明一部分实施例，而不是全部的实施例，基于本发明中的实施例，本领域普通技术人员在没有做出创造性劳动的前提下所获得的所有其他实施例，都属于本发明保护的范围。

[0016] 本发明的一种基于SpringMVC实现超大文件上传技术，主要包括如下步骤：1、下载Plupload插件并引入相应文件

[0017] 下载地址:http://www.plupload.com/download，里面有要用到的css以及js。在项目中需要引入：jQuery.plupload.queue.css，jquery-2.0.0.min.js，plupload.full.min.js，jquery.plupload.queue.js，zh_CN.js这些文件

[0018] 2、前端准备

[0019] 首先在html中写入如下代码

[0020]

```
<td class="tdTitle"><label><span class="required">*</span>附件：</label></td>
<td id="t0" class="tdRight" colspan="5"> <input id="zxpxKcxxZl" name="zxpxKcxxZl" class="form-control ue-form"
type="hidden" value="" />
<input id="scfj" class="form-control ue-form" type="text" size="70" placeholder="点击右侧浏览，上传附件" value=""
readonly="readonly"></input>
```

[0021] 注意div的id必须是uploader，这在插件源码里是有规定的；设置id为toStop与toStart的按钮，目的是为了实现暂停上传与暂停过后的继续上传。

[0022] 3、页面加载后通过js初始化组件

[0023]

```
            $(function() {
                    // 附件上传目录
                var dirFile = 'XX/XX/XX;
                    //上传
                    var uploader = new plupload.Uploader({
                        runtimes: 'html5,flash,silverlight,html4',
                        browse_button: 'pluploadfiles',
//                      url: contextPath + '/service/plupload/uploads?dirFile='+dirFile,
                        url: 'service/plupload/uploads?dirFile='+dirFile,//后台请求路径
                        max_file_size : '5000mb',
                        chunk_size : '20mb',
                        multi_selection: false,
                        unique_names: false,
                        filters: [
                            {title: "视频格式:mp4", extensions : "mp4"},
                            {title: "文档格式:doc,docx,xls,xlsx,pdf", extensions : "doc ,pdf"},
                        ],
                       flash_swf_url: '../../dist/js/Moxie.swf',
                       silverlight_xap_url: '../../dist/js/Moxie.xap',
                       init: {
                            FilesAdded: function(up, files) {
                                document.getElementById('scfj').value = '';
                            });
                       },
                       UploadProgress: function(up, file) {
                          $('#zxpxKcxxZl').val( file.name);
                            $('#' + file.id).find('.progress-bar')[0].style.width = $('#' +
file.id).find('.progress-bar')[0].innerText = file.percent + '%';
                       },
                       }, });
```

[0024] 4、后端上传逻辑编写

[0025]

Controller层

```
@Autowired
private PluploadService pluploadService;
 /**Plupload 文件上传处理方法*/
@RequestMapping(value="/pluploadUpload")
public void upload(Plupload plupload,HttpServletRequest request,HttpServletResponse response) {
     String FileDir = "pluploadDir";//文件保存的文件夹
    plupload.setRequest(request);//手动传入 Plupload 对象 HttpServletRequest 属性
     int userId = ((User)request.getSession().getAttribute("user")).getUserId();
     //文件存储绝对路径,会是一个文件夹，项目相应 Servlet 容器下的"pluploadDir"文件夹，还会以用户唯一 id 作划分
    File dir = new File(request.getSession().getServletContext().getRealPath("/") + FileDir+"/"+userId);
    if(!dir.exists()){
        dir.mkdirs();//可创建多级目录，而 mkdir()只能创建一级目录
    }
    //开始上传文件
    pluploadService.upload(plupload, dir);
}
```

[0026]

PluploadService层

```
public class PluploadService {
     @Autowired
    private youandmeService youandmeService;
     public void upload(Plupload plupload,File pluploadDir){
        String fileName = ""+System.currentTimeMillis()+plupload.getName();//在服务器内生成唯一文件名
        upload(plupload, pluploadDir, fileName);
    }
     private void upload(Plupload plupload,File pluploadDir,String fileName){
         int chunks = plupload.getChunks();//用户上传文件被分隔的总块数
        int nowChunk = plupload.getChunk();//当前块，从 0 开始
        //这里 Request 请求类型的强制转换可能出错，配置文件中向 SpringIOC 容器引入 multipartResolver 对象即可。
  MultipartHttpServletRequest multipartHttpServletRequest
=(MultipartHttpServletRequest)plupload.getRequest();
        //调试发现 map 中只有一个键值对
        MultiValueMap<String,MultipartFile> map = multipartHttpServletRequest.getMultiFileMap();
         if(map!=null){
            try{
                Iterator<String> iterator = map.keySet().iterator();
                while(iterator.hasNext()){
                     String key = iterator.next();
                    List<MultipartFile> multipartFileList = map.get(key);
```

[0027]

```
                for(MultipartFile multipartFile:multipartFileList){//循环只进行一次
                    plupload.setMultipartFile(multipartFile);//手动向Plupload对象传入MultipartFile属性值
                File targetFile = new File(pluploadDir+"/"+fileName);//新建目标文件
                    if(chunks>1){//用户上传资料总块数大于1，要进行合并
                        File tempFile = new File(pluploadDir.getPath()+"/"+multipartFile.getName());
                        //第一块直接从头写入，不用从末端写入
                    savePluploadFile(multipartFile.getInputStream(),tempFile,nowChunk==0?false:true);
                        if(chunks-nowChunk==1){//全部块已经上传完毕，此时targetFile要改文件名字
                            tempFile.renameTo(targetFile);
                            //每当文件上传完毕，将上传信息插入数据库
                            Timestamp now = new Timestamp(System.currentTimeMillis());
                            youandmeService.uploadInfo(fileName,((User)(plupload.getRequest().getSession().getA
ttribute("user"))).getUsername(),now);
                        }  }
                        else{
                        //只有一块，就直接拷贝文件内容
                        multipartFile.transferTo(targetFile);
                       //每当文件上传完毕，将上传信息插入数据库
                        Timestamp now = new Timestamp(System.currentTimeMillis());
                        youandmeService.uploadInfo(fileName, ((User) (plupload.getRequest().getSession().getAtt
ribute("user"))).getUsername(), now);
                    }
                }
            }
            catch (IOException e){
                e.printStackTrace();
            }
        }
    }
    private void savePluploadFile(InputStream inputStream,File tempFile,boolean flag){
        //文件流保存代码，此处不体现
        }
    }
}
```

[0028] 1)在Controller的最后一行PluploadService.upload(plupload,dir);中将客户端提交至服务器生成的plupload对象与规定保存的文件夹目录,以参数的形式传入PluploadService的upload方法中。

[0029] 2)在upload(Plupload plupload,File pluploadDir)方法中,为文件生成一个唯一的文件名以便存储在服务器中。

[0030] 3)chunks是用户一次性选中要上传的文件中当前文件被分隔后的总块数;nowChunk是这次上传中块的编号,从0开始。Plupload依次地将块从客户端提交至服务器,因此在文件上传中,会有很多次Http请求,而同一个文件的chunks是不变的,nowChunk会一次次增加。

[0031] 4)将HttpServletRequest强制转换为MultipartHttpServletRequest时可能会出错,但这个错误可以避免,只需在SpringIOC容器中注入一个名为multipartResolver的对象

[0032] 5)通过MultipartHttpServletRequest拿到MultiValueMap(经过调试发现这个map只有一对键值对),其Value类型为MultipartFile,这个MultipartFile其实就是当前的块。

[0033] 6)plupload.setMultipartFile(multipartFile);手动为plupload对象传入MultipartFile属性值。

[0034] 7)如果总块数chunks大于1,那就将上传过来的一个个小块合成一个文件,否则那就直接拷贝块文件到目标文件multipartFile.transferTo(targetFile);

[0035] 8)在chunks大于1时,首先要新建一个临时文件tempFile,用于不断不断将一个个小块写入这个tempFile,等写完后(chunks-nowChunk==1),就将其重命名(tempFile.renameTo(targetFile);)。

[0036] 9)savePluploadFile(multipartFile.getInputStream(),tempFile,nowChunk==0?false:true);方法用于合并一个个小块文件,如果是第一块的话,就从头开始写入(newFileOutputStream(tempFile)),否则全部从末端写入(newFileOutputStream(tempFile,true))。

[0037] 以上所述仅为本发明的较佳实施例,仅用于说明本发明的技术方案,并非用于限定本发明的保护范围。凡在本发明的精神和原则之内所做的任何修改、等同替换、改进等,均包含在本发明的保护范围内。

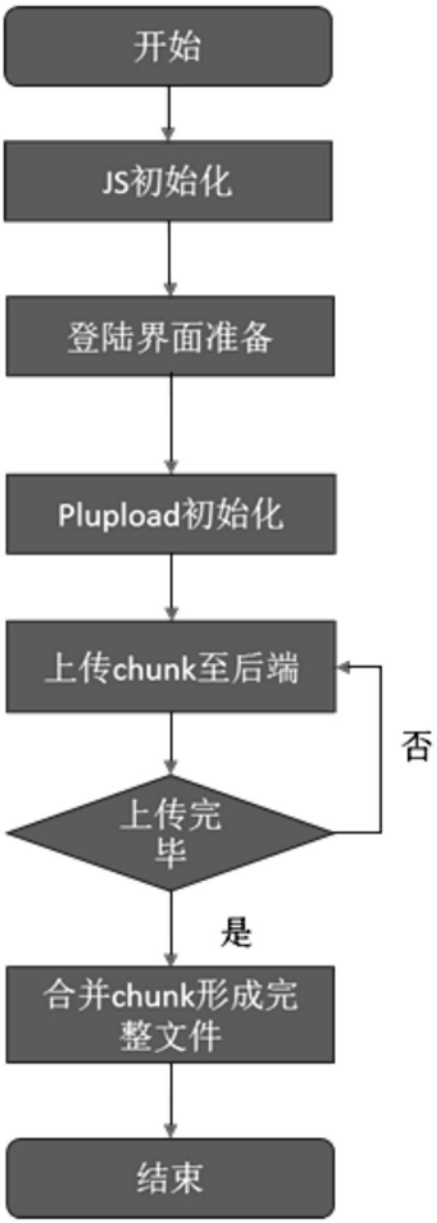


图1