FastDFS

(感觉可以做一个云图库)

# Maven依赖

<**dependency**>  
 <**groupId**>org.csource</**groupId**>  
 <**artifactId**>fastdfs-client-java</**artifactId**>  
 <**version**>1.27-SNAPSHOT</**version**>  
</**dependency**>

# 配置文件

新建fdfs\_client.conf文件

**connect\_timeout** = 60  
**network\_timeout** = 60  
**charset** = UTF-8  
**tracker\_server** = 47.100.138.136:22122

application.properties

*#文件上传***spring.servlet.multipart.enabled**=**true  
spring.servlet.multipart.max-file-size**=**10MB  
spring.servlet.multipart.max-request-size**=**10MB**

# 工具类

*/\*\*  
 \* 将MultipartFile转成当前类  
 \*/***public class** FastDFSFile {  
 **private** String **name**; *//文件名* **private byte**[] **content**; *//文件二级制数据* **private** String **ext**; *//后缀* **private** String **md5**; *//加密信息* **private** String **author**; *//作者* **public** FastDFSFile(String name, **byte**[] content, String ext, String author) {  
 **this**.**name** = name;  
 **this**.**content** = content;  
 **this**.**ext** = ext;  
 **this**.**author** = author;  
 }  
 **public** FastDFSFile(String name, **byte**[] content, String ext) {  
 **this**.**name** = name;  
 **this**.**content** = content;  
 **this**.**ext** = ext;  
 }  
 **public** String getName() {  
 **return name**;  
 }  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
 **public byte**[] getContent() {  
 **return content**;  
 }  
 **public void** setContent(**byte**[] content) {  
 **this**.**content** = content;  
 }  
 **public** String getExt() {  
 **return ext**;  
 }  
 **public void** setExt(String ext) {  
 **this**.**ext** = ext;  
 }  
 **public** String getMd5() {  
 **return md5**;  
 }  
 **public void** setMd5(String md5) {  
 **this**.**md5** = md5;  
 }  
 **public** String getAuthor() {  
 **return author**;  
 }  
 **public void** setAuthor(String author) {  
 **this**.**author** = author;  
 }  
}

*/\*\*  
 \* FastDFS连接工具类  
 \*/***public class** FastDFSClient {  
 **private static** org.slf4j.Logger *logger* = LoggerFactory.*getLogger*(FastDFSClient.**class**);  
 *//静态块,连接初始化* **static** {  
 **try** {  
 String filePath = **new** ClassPathResource(**"fdfs\_client.conf"**).getFile().getAbsolutePath();;  
 ClientGlobal.*init*(filePath);  
 } **catch** (Exception e) {  
 *logger*.error(**"FastDFS Client Init Fail!"**,e);  
 }  
 }  
 *//上传操作* **public static** String[] upload(FastDFSFile file) {  
 *logger*.info(**"File Name: "** + file.getName() + **"File Length:"** + file.getContent().**length**);  
 NameValuePair[] meta\_list = **new** NameValuePair[1];  
 meta\_list[0] = **new** NameValuePair(**"author"**, file.getAuthor());  
 **long** startTime = System.*currentTimeMillis*();  
 String[] uploadResults = **null**;  
 StorageClient storageClient=**null**;  
 **try** {  
 storageClient = *getTrackerClient*();  
 uploadResults = storageClient.upload\_file(file.getContent(), file.getExt(), meta\_list);  
 } **catch** (IOException e) {  
 *logger*.error(**"IO Exception when uploadind the file:"** + file.getName(), e);  
 } **catch** (Exception e) {  
 *logger*.error(**"Non IO Exception when uploadind the file:"** + file.getName(), e);  
 }  
 *logger*.info(**"upload\_file time used:"** + (System.*currentTimeMillis*() - startTime) + **" ms"**);  
 **if** (uploadResults == **null** && storageClient!=**null**) {  
 *logger*.error(**"upload file fail, error code:"** + storageClient.getErrorCode());  
 }  
 String groupName = uploadResults[0];  
 String remoteFileName = uploadResults[1];  
 *logger*.info(**"upload file successfully!!!"** + **"group\_name:"** + groupName + **", remoteFileName:"** + **" "** + remoteFileName);  
 **return** uploadResults;  
 }  
 */\*\*  
 \* 获得文件信息  
 \** ***@param groupName*** *group1  
 \** ***@param remoteFileName*** *M00/00/00/rBEJ4V2zvlOAUKxjAAAqEW9s6-A777.jpg  
 \** ***@return*** *文件信息  
 \*/* **public static** FileInfo getFile(String groupName, String remoteFileName) {  
 **try** {  
 StorageClient storageClient = *getTrackerClient*();  
 **return** storageClient.get\_file\_info(groupName, remoteFileName);  
 } **catch** (IOException e) {  
 *logger*.error(**"IO Exception: Get File from Fast DFS failed"**, e);  
 } **catch** (Exception e) {  
 *logger*.error(**"Non IO Exception: Get File from Fast DFS failed"**, e);  
 }  
 **return null**;  
 }  
 */\*\*  
 \* 获得文件的输入流  
 \** ***@param groupName*** *group1  
 \** ***@param remoteFileName*** *M00/00/00/rBEJ4V2zvlOAUKxjAAAqEW9s6-A777.jpg  
 \** ***@return*** *InputStream  
 \*/* **public static** InputStream downFile(String groupName, String remoteFileName) {  
 **try** {  
 StorageClient storageClient = *getTrackerClient*();  
 **byte**[] fileByte = storageClient.download\_file(groupName, remoteFileName);  
 InputStream ins = **new** ByteArrayInputStream(fileByte);  
 **return** ins;  
 } **catch** (IOException e) {  
 *logger*.error(**"IO Exception: Get File from Fast DFS failed"**, e);  
 } **catch** (Exception e) {  
 *logger*.error(**"Non IO Exception: Get File from Fast DFS failed"**, e);  
 }  
 **return null**;  
 }  
 */\*\*  
 \* 删除文件  
 \** ***@param groupName*** *group1  
 \** ***@param remoteFileName*** *M00/00/00/rBEJ4V2zvlOAUKxjAAAqEW9s6-A777.jpg  
 \** ***@return*** *有值说明删除成功  
 \** ***@throws*** *Exception  
 \*/* **public static int** deleteFile(String groupName, String remoteFileName)  
 **throws** Exception {  
 StorageClient storageClient = *getTrackerClient*();  
 **int** i = storageClient.delete\_file(groupName, remoteFileName);  
 *logger*.info(**"delete file successfully!!!"** + i);  
 **return** i;  
 }  
 **public static** StorageServer[] getStoreStorages(String groupName)  
 **throws** IOException {  
 TrackerClient trackerClient = **new** TrackerClient();  
 TrackerServer trackerServer = trackerClient.getConnection();  
 **return** trackerClient.getStoreStorages(trackerServer, groupName);  
 }  
 **public static** ServerInfo[] getFetchStorages(String groupName,  
 String remoteFileName) **throws** IOException {  
 TrackerClient trackerClient = **new** TrackerClient();  
 TrackerServer trackerServer = trackerClient.getConnection();  
 **return** trackerClient.getFetchStorages(trackerServer, groupName, remoteFileName);  
 }  
 **public static** String getTrackerUrl() **throws** IOException {  
 **return "http://"**+*getTrackerServer*().getInetSocketAddress().getHostString()+**"/"**;  
 }  
 **private static** StorageClient getTrackerClient() **throws** IOException {  
 TrackerServer trackerServer = *getTrackerServer*();  
 StorageClient storageClient = **new** StorageClient(trackerServer, **null**);  
 **return** storageClient;  
 }  
 **private static** TrackerServer getTrackerServer() **throws** IOException {  
 TrackerClient trackerClient = **new** TrackerClient();  
 TrackerServer trackerServer = trackerClient.getConnection();  
 **return** trackerServer;  
 }  
}

# Controller参考

@Controller  
**public class** UploadController {  
 **private static** Logger *logger* = LoggerFactory.*getLogger*(UploadController.**class**);  
 @GetMapping(**"/"**)  
 **public** String index() {  
 **return "upload"**;  
 }  
 @PostMapping(**"/upload"**) *//new annotation since 4.3* **public** String singleFileUpload(@RequestParam(**"file"**) MultipartFile file,  
 RedirectAttributes redirectAttributes) {  
 **if** (file.isEmpty()) {  
 redirectAttributes.addFlashAttribute(**"message"**, **"Please select a file to upload"**);  
 **return "redirect:uploadStatus"**;  
 }  
 **try** {  
 *// Get the file and save it somewhere* String path=saveFile(file);  
 redirectAttributes.addFlashAttribute(**"message"**,  
 **"You successfully uploaded '"** + file.getOriginalFilename() + **"'"**);  
 redirectAttributes.addFlashAttribute(**"path"**,  
 **"file path url '"** + path + **"'"**);  
 } **catch** (Exception e) {  
 *logger*.error(**"upload file failed"**,e);  
 }  
 **return "redirect:/uploadStatus"**;  
 }  
 @GetMapping(**"/uploadStatus"**)  
 **public** String uploadStatus() {  
 **return "uploadStatus"**;  
 }  
 */\*\*  
 \** ***@param multipartFile*** *\** ***@return*** *\** ***@throws*** *IOException  
 \*/* **public** String saveFile(MultipartFile multipartFile) **throws** IOException {  
 String[] fileAbsolutePath={};  
 String fileName=multipartFile.getOriginalFilename();  
 String ext = fileName.substring(fileName.lastIndexOf(**"."**) + 1);  
 **byte**[] file\_buff = **null**;  
 InputStream inputStream=multipartFile.getInputStream();  
 **if**(inputStream!=**null**){  
 **int** len1 = inputStream.available();  
 file\_buff = **new byte**[len1];  
 inputStream.read(file\_buff);  
 }  
 inputStream.close();  
 FastDFSFile file = **new** FastDFSFile(fileName, file\_buff, ext);  
 **try** {  
 fileAbsolutePath = FastDFSClient.*upload*(file); *//upload to fastdfs* } **catch** (Exception e) {  
 *logger*.error(**"upload file Exception!"**,e);  
 }  
 **if** (fileAbsolutePath==**null**) {  
 *logger*.error(**"upload file failed,please upload again!"**);  
 }  
 String path=FastDFSClient.*getTrackerUrl*()+fileAbsolutePath[0]+ **"/"**+fileAbsolutePath[1];  
 **return** path;  
 }  
 @RequestMapping(**"/getFileInfo"**)  
 @ResponseBody  
 **public** String getFileInfo(String groupName, String remoteFileName){  
 FileInfo fileInfo = FastDFSClient.*getFile*(groupName, remoteFileName);  
 **if** (fileInfo != **null**){  
 **return** fileInfo.toString();  
 }  
 **return "参数有误!"**;  
 }  
 @RequestMapping(**"/deleteFile"**)  
 @ResponseBody  
 **public** Integer deleteFile(String groupName, String remoteFileName){  
 **int** i = 0;  
 **try** {  
 i = FastDFSClient.*deleteFile*(groupName, remoteFileName);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **return** i;  
 }  
}