Java FTP客户端类库比较

目录

[1. Java实现FTP上传下载 1](#_Toc426282658)

[1.1. Apache FTPClient 1](#_Toc426282659)

[1.2. ftp4j 2](#_Toc426282660)

[1.3. JDk1.7 4](#_Toc426282661)

[2. 效率比较 6](#_Toc426282662)

[2.1. 上传效率比较 6](#_Toc426282663)

[2.2. 下载效率比较 6](#_Toc426282664)

# Java实现FTP上传下载

## Apache FTPClient

FTPClient是Apache commons.net包中的一个常用类，实现了FTP客户端的相关协议，它经常被用来开发Java FTP客户端应用程序，提供了链接FTP服务器、获取服务器目录下的文件一览、上传文件、下载文件、创建目录、删除文件、重命名文件等常用功能。

Apache FTPClient操作文件上传下载伪代码：

* 登录FTP，并返回登录是否成功的Boolean值

|  |
| --- |
| **public** **boolean** login(String host, **int** port, String user, String password) { **boolean** flag = **true**; **try** { client.connect(host, port); client.login(user, password); } **catch** (Exception e) { e.printStackTrace(); flag = **false**; } **return** flag;} |

* 上传文件

|  |
| --- |
| /\*\***@param** path保存FTP位置**@param** file要上传的文件**@param** remoteName在FTP保存时的名字\*/**public** **void** upload(String path, File file, String remoteName) { **try** { **if**(cdAssignPath(path)) {  client.storeFile(remoteName, **new** FileInputStream(file)); } }**catch** (Exception e) { e.printStackTrace(); } } |

* 下载文件

|  |
| --- |
| /\*\***@param** remotePath**@param** remoteName**@param** localPath**@param** localName\*/**public** **void** download(String remotePath, String remoteName, String localPath, String localName) { **if**(cdAssignPath(remotePath)) { **try** { File file = **new** File(localPath); **if**(!file.exists()) { file.mkdirs(); } FileOutputStream write = **new** FileOutputStream(**new** File(localPath + "/" + localName)); client.retrieveFile(remoteName, write); write.close(); } **catch** (Exception e) { e.printStackTrace(); } }} |

## ftp4j

[ftp4j](http://www.sauronsoftware.it/projects/ftp4j/)是一个FTP客户端Java类库，实现了FTP客户端应具有的大部分功能。ftp4j提供多种方式连接到远程FTP服务器包括：通过 TCP/IP直接连接，通过FTP代理、HTTP代理、SOCKS4/4a代理和SOCKS5代理连接，通过SSL安全链接。

[ftp4j](http://www.sauronsoftware.it/projects/ftp4j/)操作文件上传下载伪代码：

* 登录FTP，并返回登录是否成功的Boolean值

|  |
| --- |
| **private** FTPClient client = **new** FTPClient(); /\*\***@param** host**@param** port**@param** user**@param** password**@return**\*/**public** **boolean** login(String host, **int** port, String user, String password) { **boolean** flag = **true**; **try** { client.connect(host, port); client.login(user, password); //数据传输方式 client.setType(FTPClient.TYPE\_BINARY); } **catch** (Exception e) { e.printStackTrace(); flag = **false**; } **return** flag;}  |

* 上传文件

|  |
| --- |
| /\*\***@param** path保存FTP位置**@param** file要上传的文件\*/**public** **void** upload(String path, File file) { **if**(cdAssignPath(path)) { **try** { client.upload(file); } **catch** (Exception e) { e.printStackTrace(); } }} |

* 下载文件

|  |
| --- |
| /\*\***@param** remotePath**@param** remoteName**@param** localPath**@param** localName\*/**public** **void** download(String remotePath, String remoteName, String localPath, String localName) { **if**(cdAssignPath(remotePath)) { **try** { File file = **new** File(localPath); **if**(!file.exists()) { file.mkdirs(); } client.download(remoteName, **new** File(localPath + "/" + localName)); } **catch** (Exception e) { e.printStackTrace(); } }} |

## JDk1.7

sun.net.ftp.FtpClient该类库主要提供了用于建立FTP连接的类。利用这些类的方法可以远程登录到FTP服务器，列举该服务器上的目录，设置传输协议，以及传送文件。

FtpClient操作文件上传下载伪代码：

* 登录FTP，并返回登录是否成功的Boolean值

|  |
| --- |
| **private** FtpClient client = FtpClient.create();/\*\***@param** host**@param** port**@param** user**@param** password**@return**\*/**public** **boolean** login(String host, **int** port, String user, String password) { **try** {SocketAddress addr = **new** InetSocketAddress(ip, port); client.connect(addr); client.login(user, password.toCharArray()); } **catch** (Exception e) { e.printStackTrace(); flag = **false**; } **return** flag;} |

* 上传文件

|  |
| --- |
| /\*\***@param** path保存FTP位置**@param** file要上传的文件**@param** remoteName在FTP保存时的名字\*/**public** **void** upload(String path, File file, String remoteName) { OutputStream os = **null**; FileInputStream is = **null**; **try** { **if**(cdAssignPath(path)) { is = **new** FileInputStream(file);// 创建一个缓冲区 **byte**[] bytes = **new** **byte**[1024]; **int** c; **while** ((c = is.read(bytes)) != -1) { os.write(bytes, 0, c); } }**catch** (Exception e) { e.printStackTrace(); } **finally** { **if** (read != **null**) {**try** { read.close(); } **catch** (IOException e) { e.printStackTrace(); } } **if** (write != **null**) { **try** { write.close(); } **catch** (IOException e) { e.printStackTrace(); } }}} |

* 下载文件

|  |
| --- |
| /\*\***@param** remotePath**@param** remoteName**@param** localPath**@param** localName\*/**public** **void** download(String remotePath, String remoteName, String localPath, String localName) { **if**(cdAssignPath(remotePath)) { **try** { InputStream is = client.getFileStream(remoteName); File file = **new** File(localPath); **if**(!file.exists()) { file.mkdirs(); } FileOutputStream out = **new** FileOutputStream(**new** File(localPath + "/" + localName)); **byte**[] bytes = **new** **byte**[1024]; **int** c; **while** ((c = is.read(bytes)) != -1) { os.write(bytes, 0, c); } is.close(); os.close(); } **catch** (IOException e) { e.printStackTrace(); } }} |

# 效率比较

## 上传效率比较

基于三种FTP类库分别测试256KB、1MB、10MB文件上传到FTP服务器所需要的时间如下表所示。从表中可以看出基于JDK1.7的FtpClient实现的FTP上传效率最高，其次是ftp4j.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 256KB | 1MB | 10MB |
| Apache FTPClient | 301ms | 1169ms | 11623.3ms |
| ftp4j | Min=6ms,Max=117ms | Min=7.3ms,Max=119ms | 149.6ms |
| JDK1.7 | 5.3ms | 10.3ms | 70ms |

## 下载效率比较

测试256KB、1MB、10MB文件从FTP服务器下载到本地所需要的时间如下表所示。ftp4j在文件下载时候效率最高，其次是JDK1.7。而Apache FTPClient在文件上传和下载效率都是比较低的。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 256KB | 1MB | 10MB |
| Apache FTPClient | 426ms | 1650ms | 11428.3ms |
| ftp4j | 5.7ms | 7.7ms | 36.3ms |
| JDK1.7 | 6.6ms | 13.5ms | 90ms |