# 尚马教育 JAVA 高级课程

# fastDFS

文档编号：C14

创建日期： 2017-07-07

最后修改日期：2019-09-09

版 本 号：V3.5

电子版文件名：尚马教育-第三阶段-14.FastDFS分布式文件存储.docx

**文档修改记录：**

|  |  |  |  |
| --- | --- | --- | --- |
| 更新日期 | 更新作者 | 更新说明 | 版本号 |
| 2017-08-25 | 张元林 | 初始版本 | V1.0 |
| 2018-08-15 | 王绍成 | Mybatis版本更新 | V2.0 |
| 2019-09-20 | 冯勇涛 | 课件格式以及课程深度加深 | V3.0 |

目录

[尚马教育 JAVA 高级课程 1](#_Toc15173)

[fastDFS 1](#_Toc18524)

[1. Fastdfs介绍 3](#_Toc27215)

[1.1. 认识fastdfs 3](#_Toc18304)

[1.2. FastDFS组成部分 3](#_Toc19139)

[1.3. FastDFS服务器架构 5](#_Toc1188)

[2. FastDFS客户端开发 5](#_Toc3957)

[2.1. jar包准备 5](#_Toc31580)

[2.2. 配置文件 6](#_Toc12676)

[2.3. 使用客户端api 6](#_Toc1185)

[2.3.1. 初始化fastdfs服务 6](#_Toc16640)

[2.3.2. 上传文件 6](#_Toc26476)

[2.3.3. 下载文件 8](#_Toc11133)

[2.3.4. 删除文件 8](#_Toc7041)

[2.3.5. 生成HttpUrl 9](#_Toc12349)

## Fastdfs介绍

### 认识fastdfs

#### 原作者版本

FastDFS是由国人余庆所开发，其项目地址https://github.com/happyfish100/fastdfs

FastDFS是一个轻量级的开源分布式文件系统，主要解决了大容量的文件存储和高并发访问的问题，文件存取时实现了负载均衡。

支持存储服务器在线扩容,支持相同的文件只保存一份,节约磁盘。

FastDFS只能通过Client API访问，不支持POSIX访问方式。

FastDFS适合中大型网站使用，用来存储资源文件(如：图片、文档、视频等)

#### fastdfs\_client版本

[tobato](https://github.com/tobato)项目团队在原作者YuQing与yuqih发布的java客户端基础上进行了大量重构工作，项目地址：https://github.com/tobato/fastdfs\_client

|  |
| --- |
| 支持对服务端的连接池管理；  支持上传图片时候检查图片格式，并且自动生成缩略图；  并且在SpringBoot当中自动导入依赖。 |

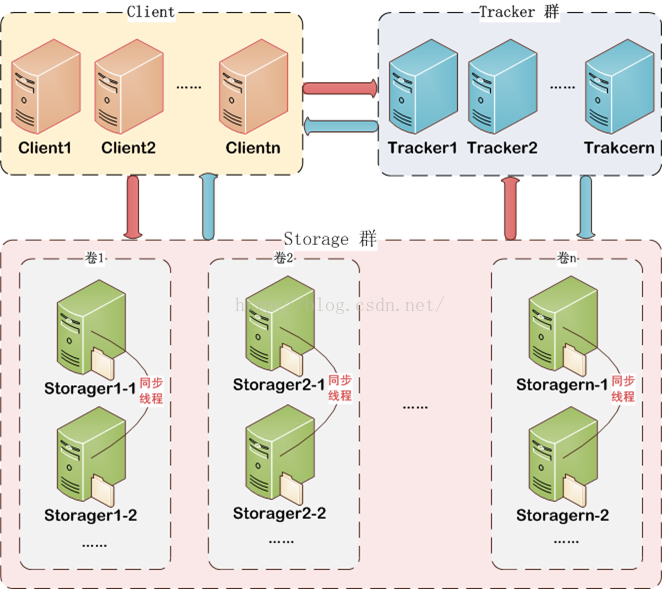
### FastDFS组成部分

FastDFS由跟踪服务器（tracker server）、存储服务器（storage server）和客户端（client）三个部分组成，主要解决了海量数据存储问题，特别适合以中小文件（建议范围：4KB < file\_size <500MB）为载体的在线服务。

|  |
| --- |
| Tracker server(跟踪服务器)  跟踪服务器：用来调度来自客户端的请求。且在内存中记录所有存储组和存储服务器的信息状态。 |
| Storage server(存储服务器)  用来存储文件(data)和文件属性(metadata) |
| Client客户端  提供基本文件访问接口，比如upload、download、append、delete等，以客户端库的方式提供给用户使用。 |

### FastDFS服务器架构

* 存储服务器会定期向跟踪服务器发送状态信息（心跳包）。
* 客户端发起上传请求时,向跟踪服务器查询可用的存储服务器。
* 客户端向存储服务器上传文件内容,存储服务器保存文件并生成文件路径以及文件名
* 存储服务器将路径及文件名返回给客户端



## FastDFS客户端开发

### jar包准备

Maven:

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

### 配置文件

fdfs\_client.conf

|  |
| --- |
| connect\_timeout = 2 network\_timeout = 30 charset = UTF-8 http.tracker\_http\_port = 8111 tracker\_server = 192.168.20.252:22122 |

### 使用客户端api

#### 初始化fastdfs服务

|  |
| --- |
| **private static** TrackerClient *tc* = **null**; **private static** String *httpPort* = **null**;  **static** {  URL resource = FastdfsUtil.**class**.getClassLoader().getResource(**"fdfs\_client.conf"**);  **try** {  ClientGlobal.*init*(resource.getPath());*//加载conf配置文件  httpPort* = ClientGlobal.*getG\_tracker\_http\_port*() + **""**;// 得到http服务端口  *tc* = **new** TrackerClient();*//所有的跟踪服务器信息在TrackerClient对象中。*  } **catch** (Exception e) {  e.printStackTrace();  }} |

#### 上传文件

|  |
| --- |
| **public static** String upload(**byte**[] bytes, String fileName, **int** size) {  String savePath = **null**;  TrackerServer ts = **null**;*//跟踪服务对象* StorageServer ss = **null**;*//存储服务对象* **try** {  ts = *tc*.getConnection();*//得到一台具体的跟踪服务器* StorageClient1 sc = **new** StorageClient1(ts, ss);*//TrackerServer分配StorageServer  //文件后缀* String extName = *getExtName*(fileName);  *//元数据列表,(类似写的注释)上传文件习惯把上传日期，文件大小，文件真实名* NameValuePair[] meta\_list = {  **new** NameValuePair(***UPLOAD\_TIME***, System.*currentTimeMillis*() + **""**),  **new** NameValuePair(***FILE\_SIZE***, size + **""**),  **new** NameValuePair(***FILE\_REALNAME***, fileName)  };   savePath = sc.upload\_file1(bytes, extName, meta\_list);  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  *//释放连接* **try** {  **if** (ts != **null**) ts.close();  **if** (ss != **null**) ss.close();  } **catch** (IOException e) {  e.printStackTrace();  }  }  **return** savePath; } |

#### 下载文件

|  |
| --- |
| **public static byte**[] downloadFile(String savePath) {  **byte**[] bs = **null**;  TrackerServer ts = **null**;  StorageServer ss = **null**;  **try** {  ts = *tc*.getConnection();  StorageClient1 sc = **new** StorageClient1(ts, ss);  bs = sc.download\_file1(savePath);  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {  **if** (ts != **null**) ts.close();  **if** (ss != **null**) ss.close();  } **catch** (IOException e) {  e.printStackTrace();  }  }  **return** bs; } |

#### 删除文件

|  |
| --- |
| **public static boolean** deleteFile(String savePath) {  **boolean** i=**false**;  TrackerServer ts = **null**;  StorageServer ss = **null**;  **try** {  ts = *tc*.getConnection();  StorageClient1 sc = **new** StorageClient1(ts, ss);  **int** isDel = sc.delete\_file1(savePath);*//表示删除文件，0成功，2失败* i = isDel==0?**true**:**false**;  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {  **if** (ts != **null**) ts.close();  **if** (ss != **null**) ss.close();  } **catch** (IOException e) {  e.printStackTrace();  }  }  **return** i; } |

#### 生成HttpUrl

|  |
| --- |
| **public static** String getHttpUrl(String savePath){  String url =**"http://"**;  TrackerServer ts = **null**;  **try** {  ts = *tc*.getConnection();*//得到一台具体的跟踪服务器* InetSocketAddress address = ts.getInetSocketAddress();  String hostName = address.getHostName();*//得到跟踪服务器的主机名* url = url+hostName+**":"**+*httpPort*+**"/"**+savePath;  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {  **if** (ts != **null**) ts.close();  } **catch** (IOException e) {  e.printStackTrace();  }  }  **return** url; } |