HDFS中JAVA API的使用

　　HDFS是一个分布式文件系统，既然是文件系统，就可以对其文件进行操作，比如说新建文件、删除文件、读取文件内容等操作。下面记录一下使用JAVA API对HDFS中的文件进行操作的过程。

　　对分HDFS中的文件操作主要涉及一下几个类：

　　Configuration类：该类的对象封转了客户端或者服务器的配置。

　　FileSystem类：该类的对象是一个文件系统对象，可以用该对象的一些方法来对文件进行操作。FileSystem fs = FileSystem.get(conf);通过FileSystem的静态方法get获得该对象。

　　FSDataInputStream和FSDataOutputStream：这两个类是HDFS中的输入输出流。分别通过FileSystem的open方法和create方法获得。

　　具体如何对文件操作清下下面例子：

复制代码

package com.hdfs;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStream;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FSDataOutputStream;

import org.apache.hadoop.fs.FileStatus;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IOUtils;

public class HdfsTest {

//创建新文件

public static void createFile(String dst , byte[] contents) throws IOException{

Configuration conf = new Configuration();

FileSystem fs = FileSystem.get(conf);

Path dstPath = new Path(dst); //目标路径

//打开一个输出流

FSDataOutputStream outputStream = fs.create(dstPath);

outputStream.write(contents);

outputStream.close();

fs.close();

System.out.println("文件创建成功！");

}

//上传本地文件

public static void uploadFile(String src,String dst) throws IOException{

Configuration conf = new Configuration();

FileSystem fs = FileSystem.get(conf);

Path srcPath = new Path(src); //原路径

Path dstPath = new Path(dst); //目标路径

//调用文件系统的文件复制函数,前面参数是指是否删除原文件，true为删除，默认为false

fs.copyFromLocalFile(false,srcPath, dstPath);

//打印文件路径

System.out.println("Upload to "+conf.get("fs.default.name"));

System.out.println("------------list files------------"+"\n");

FileStatus [] fileStatus = fs.listStatus(dstPath);

for (FileStatus file : fileStatus)

{

System.out.println(file.getPath());

}

fs.close();

}

//文件重命名

public static void rename(String oldName,String newName) throws IOException{

Configuration conf = new Configuration();

FileSystem fs = FileSystem.get(conf);

Path oldPath = new Path(oldName);

Path newPath = new Path(newName);

boolean isok = fs.rename(oldPath, newPath);

if(isok){

System.out.println("rename ok!");

}else{

System.out.println("rename failure");

}

fs.close();

}

//删除文件

public static void delete(String filePath) throws IOException{

Configuration conf = new Configuration();

FileSystem fs = FileSystem.get(conf);

Path path = new Path(filePath);

boolean isok = fs.deleteOnExit(path);

if(isok){

System.out.println("delete ok!");

}else{

System.out.println("delete failure");

}

fs.close();

}

//创建目录

public static void mkdir(String path) throws IOException{

Configuration conf = new Configuration();

FileSystem fs = FileSystem.get(conf);

Path srcPath = new Path(path);

boolean isok = fs.mkdirs(srcPath);

if(isok){

System.out.println("create dir ok!");

}else{

System.out.println("create dir failure");

}

fs.close();

}

//读取文件的内容

public static void readFile(String filePath) throws IOException{

Configuration conf = new Configuration();

FileSystem fs = FileSystem.get(conf);

Path srcPath = new Path(filePath);

InputStream in = null;

try {

in = fs.open(srcPath);

IOUtils.copyBytes(in, System.out, 4096, false); //复制到标准输出流

} finally {

IOUtils.closeStream(in);

}

}

public static void main(String[] args) throws IOException {

//测试上传文件

//uploadFile("D:\\c.txt", "/user/hadoop/test/");

//测试创建文件

/\*byte[] contents = "hello world 世界你好\n".getBytes();

createFile("/user/hadoop/test1/d.txt",contents);\*/

//测试重命名

//rename("/user/hadoop/test/d.txt", "/user/hadoop/test/dd.txt");

//测试删除文件

//delete("test/dd.txt"); //使用相对路径

//delete("test1"); //删除目录

//测试新建目录

//mkdir("test1");

//测试读取文件

readFile("test1/d.txt");

}

}