******HDFS 基本操作篇******

3. HDFS 的 shell(命令行客户端)操作

3.1 HDFS 命令行客户端使用

HDFS 提供 shell 命令行客户端，使用方法如下：

```
[hadoop@hdp-node-01 ~]$ hadoop fs -ls /
15/11/08 17:51:02 WARN util.NativeCodeLoader: Unable to load native-hadoop
Found 4 items
drwxr-xr-x   - hadoop supergroup          0 2015-11-03 03:26 /hbase
drwx-wx-wx   - hadoop supergroup          0 2015-11-04 02:09 /tmp
drwxr-xr-x   - hadoop supergroup          0 2015-11-04 06:46 /user
drwxr-xr-x   - hadoop supergroup          0 2015-11-08 08:40 /weblog
[hadoop@hdp-node-01 ~]$
```

3.2 命令行客户端支持的命令参数

```
[-appendToFile <localsrc> ... <dst>]
[-cat [-ignoreCrc] <src> ...]
[-checksum <src> ...]
[-chgrp [-R] GROUP PATH...]
[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
[-chown [-R] [OWNER][:[GROUP]] PATH...]
[-copyFromLocal [-f] [-p] <localsrc> ... <dst>]
[-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
[-count [-q] <path> ...]
[-cp [-f] [-p] <src> ... <dst>]
[-createSnapshot <snapshotDir> [<snapshotName>]]
[-deleteSnapshot <snapshotDir> <snapshotName>]
[-df [-h] [<path> ...]]
[-du [-s] [-h] <path> ...]
[-expunge]
[-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
[-getfacl [-R] <path>]
[-getmerge [-nl] <src> <localdst>]
```

[-help [cmd ...]]
[-ls [-d] [-h] [-R] [<path> ...]]
[-mkdir [-p] <path> ...]
[-moveFromLocal <localsrc> ... <dst>]
[-moveToLocal <src> <localdst>]
[-mv <src> ... <dst>]
[-put [-f] [-p] <localsrc> ... <dst>]
[-renameSnapshot <snapshotDir> <oldName> <newName>]
[-rm [-f] [-r|-R] [-skipTrash] <src> ...]
[-rmdir [--ignore-fail-on-non-empty] <dir> ...]
[-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>]|[--set <acl_spec> <path>]]
[-setrep [-R] [-w] <rep> <path> ...]
[-stat [format] <path> ...]
[-tail [-f] <file>]
[-test -[defsz] <path>]
[-text [-ignoreCrc] <src> ...]
[-touchz <path> ...]
[-usage [cmd ...]]

3.2 常用命令参数介绍

-help
功能：输出这个命令参数手册

-ls
功能：显示目录信息
示例： hadoop fs -ls hdfs://hadoop-server01:9000/
备注：这些参数中，所有的 hdfs 路径都可以简写
-->hadoop fs -ls / 等同于上一条命令的效果

-mkdir
功能：在 hdfs 上创建目录
示例：hadoop fs -mkdir -p /aaa/bbb/cc/dd

-moveFromLocal
功能：从本地剪切粘贴到 hdfs
示例：hadoop fs - moveFromLocal /home/hadoop/a.txt /aaa/bbb/cc/dd

-moveToLocal
功能：从 hdfs 剪切粘贴到本地
示例：hadoop fs - moveToLocal /aaa/bbb/cc/dd /home/hadoop/a.txt

--appendToFile
功能：追加一个文件到已经存在的文件末尾

示例：hadoop fs -appendToFile ./hello.txt hdfs://hadoop-server01:9000/hello.txt
可以简写为：
Hadoop fs -appendToFile ./hello.txt /hello.txt

-cat
功能：显示文件内容
示例：hadoop fs -cat /hello.txt

-tail
功能：显示一个文件的末尾
示例：hadoop fs -tail /weblog/access_log.1
-text
功能：以字符形式打印一个文件的内容
示例：hadoop fs -text /weblog/access_log.1

-chgrp
-chmod
-chown
功能：linux 文件系统中的用法一样，对文件所属权限
示例：
hadoop fs -chmod 666 /hello.txt
hadoop fs -chown someuser:somegrp /hello.txt

-copyFromLocal
功能：从本地文件系统中拷贝文件到 hdfs 路径去
示例：hadoop fs -copyFromLocal ./jdk.tar.gz /aaa/
-copyToLocal
功能：从 hdfs 拷贝到本地
示例：hadoop fs -copyToLocal /aaa/jdk.tar.gz

-cp
功能：从 hdfs 的一个路径拷贝 hdfs 的另一个路径
示例： hadoop fs -cp /aaa/jdk.tar.gz /bbb/jdk.tar.gz.2

-mv
功能：在 hdfs 目录中移动文件
示例： hadoop fs -mv /aaa/jdk.tar.gz /

-get
功能：等同于 copyToLocal，就是从 hdfs 下载文件到本地
示例：hadoop fs -get /aaa/jdk.tar.gz
-getmerge
功能：合并下载多个文件
示例：比如 hdfs 的目录 /aaa/ 下有多个文件:log.1, log.2,log.3,...
hadoop fs -getmerge /aaa/log.* ./log.sum

-put
功能：等同于 copyFromLocal
示例：hadoop fs -put /aaa/jdk.tar.gz /bbb/jdk.tar.gz.2

-rm **功能：删除文件或文件夹** *示例：hadoop fs -rm -r /aaa/bbb/* **-rmdir** **功能：删除空目录** *示例：hadoop fs -rmdir /aaa/bbb/ccc*
-df **功能：统计文件系统的可用空间信息** *示例：hadoop fs -df -h /* **-du** **功能：统计文件夹的大小信息** *示例：* *hadoop fs -du -s -h /aaa/**
-count **功能：统计一个指定目录下的文件节点数量** *示例：hadoop fs -count /aaa/*
-setrep **功能：设置 hdfs 中文件的副本数量** *示例：hadoop fs -setrep 3 /aaa/jdk.tar.gz* *<这里设置的副本数只是记录在 namenode 的元数据中，是否真的会有这么多副本，还得看 datanode 的数量>*

******HDFS 应用开发篇******

4. HDFS 的 java 操作

hdfs 在生产应用中主要是客户端的开发，其核心步骤是从 hdfs 提供的 api 中构造一个 HDFS 的访问客户端对象，然后通过该客户端对象操作（增删改查）HDFS 上的文件

4.1 搭建开发环境

1、引入依赖

```
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-client</artifactId>
    <version>2.6.1</version>
</dependency>
```

注：如需手动引入 jar 包，hdfs 的 jar 包----hadoop 的安装目录的 share 下

2、window 下开发的说明

建议在 linux 下进行 hadoop 应用的开发，不会存在兼容性问题。如在 window 上做客户端应用开发，需要设置以下环境：

A、在 windows 的某个目录下解压一个 hadoop 的安装包

B、将安装包下的 lib 和 bin 目录用对应 windows 版本平台编译的本地库替换

C、在 window 系统中配置 HADOOP_HOME 指向你解压的安装包

D、在 windows 系统的 path 变量中加入 hadoop 的 bin 目录

4.2 获取 api 中的客户端对象

在 java 中操作 hdfs，首先要获得一个客户端实例

```
Configuration conf = new Configuration()
FileSystem fs = FileSystem.get(conf)
```

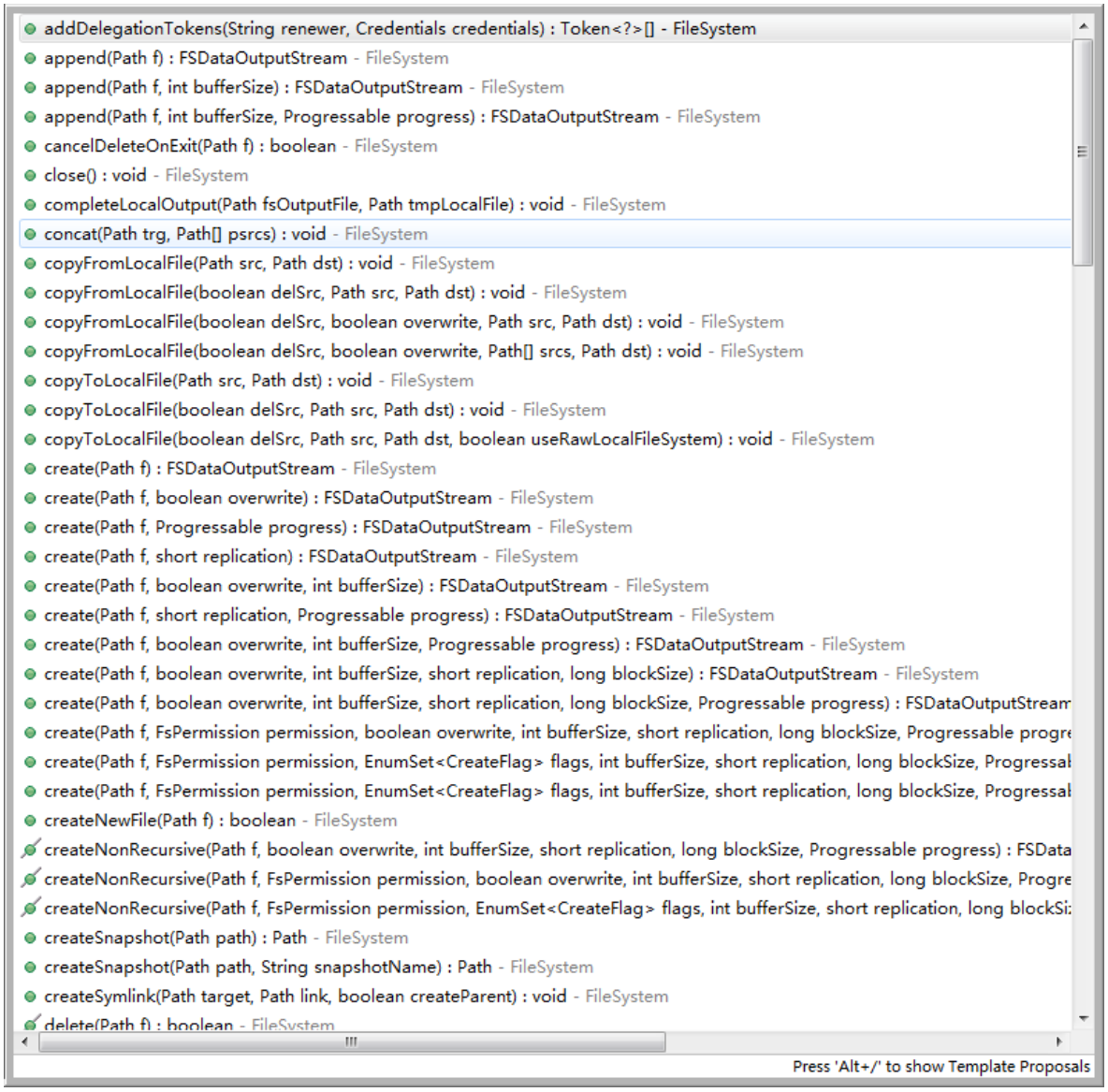
而我们的操作目标是 HDFS，所以获取到的 fs 对象应该是 DistributedFileSystem 的实例；

get 方法是从何处判断具体实例化那种客户端类呢？

——从 conf 中的一个参数 fs.defaultFS 的配置值判断；

如果我们的代码中没有指定 fs.defaultFS，并且工程 classpath 下也没有给定相应的配置，conf 中的默认值就来自于 hadoop 的 jar 包中的 core-default.xml，默认值为： **错误！超链接引用无效。**，则获取的将不是一个 DistributedFileSystem 的实例，而是一个本地文件系统的客户端对象

4.3 DistributedFileSystem 实例对象所具备的方法



4.4 HDFS 客户端操作数据代码示例：

4.4.1 文件的增删改查

```
public class HdfsClient {

    FileSystem fs = null;

    @Before
    public void init() throws Exception {
```

```
        // 构造一个配置参数对象，设置一个参数：我们要访问的 hdfs 的 URI
        // 从而 FileSystem.get()方法就知道应该是去构造一个访问 hdfs 文件系统的客户端，以及 hdfs 的访问地址
        // new Configuration();的时候，它就会去加载 jar 包中的 hdfs-default.xml
        // 然后再加载 classpath 下的 hdfs-site.xml
        Configuration conf = new Configuration();
        conf.set("fs.defaultFS", "hdfs://hdp-node01:9000");
        /**
         * 参数优先级： 1、客户端代码中设置的值 2、classpath 下的用户自定义配置文件 3、然后是服务器的默认配置
         */
        conf.set("dfs.replication", "3");

        // 获取一个 hdfs 的访问客户端，根据参数，这个实例应该是 DistributedFileSystem 的实例
        // fs = FileSystem.get(conf);

        // 如果这样去获取，那 conf 里面就可以不要配"fs.defaultFS"参数，而且，这个客户端的身份标识已经是 hadoop 用户
        fs = FileSystem.get(new URI("hdfs://hdp-node01:9000"), conf, "hadoop");

    }

    /**
     * 往 hdfs 上传文件
     *
     * @throws Exception
     */
    @Test
    public void testAddFileToHdfs() throws Exception {

        // 要上传的文件所在的本地路径
        Path src = new Path("g:/redis-recommend.zip");
        // 要上传到 hdfs 的目标路径
        Path dst = new Path("/aaa");
        fs.copyFromLocalFile(src, dst);
        fs.close();
    }

    /**
     * 从 hdfs 中复制文件到本地文件系统
     *
     * @throws IOException
     * @throws IllegalArgumentException
     */
```

```
@Test
public void testDownloadFileToLocal() throws IllegalArgumentException, IOException {
        fs.copyToLocalFile(new Path("/jdk-7u65-linux-i586.tar.gz"), new Path("d:/"));
        fs.close();
}

@Test
public void testMkdirAndDeleteAndRename() throws IllegalArgumentException, IOException {

        // 创建目录
        fs.mkdirs(new Path("/a1/b1/c1"));

        // 删除文件夹 ，如果是非空文件夹，参数 2 必须给值 true
        fs.delete(new Path("/aaa"), true);

        // 重命名文件或文件夹
        fs.rename(new Path("/a1"), new Path("/a2"));

}

/**
 * 查看目录信息，只显示文件
 *
 * @throws IOException
 * @throws IllegalArgumentException
 * @throws FileNotFoundException
 */
@Test
public void testListFiles() throws FileNotFoundException, IllegalArgumentException, IOException {

        // 思考：为什么返回迭代器，而不是 List 之类的容器
        RemoteIterator<LocatedFileStatus> listFiles = fs.listFiles(new Path("/"), true);

        while (listFiles.hasNext()) {
                LocatedFileStatus fileStatus = listFiles.next();
                System.out.println(fileStatus.getPath().getName());
                System.out.println(fileStatus.getBlockSize());
                System.out.println(fileStatus.getPermission());
                System.out.println(fileStatus.getLen());
                BlockLocation[] blockLocations = fileStatus.getBlockLocations();
                for (BlockLocation bl : blockLocations) {
                        System.out.println("block-length:" + bl.getLength() + "--" + "block-offset:" + bl.getOffset());
                        String[] hosts = bl.getHosts();
                        for (String host : hosts) {
```

```
                    System.out.println(host);
                }
            }
            System.out.println("----------为 angelababy 打印的分割线----------");
        }
    }

    /**
     * 查看文件及文件夹信息
     *
     * @throws IOException
     * @throws IllegalArgumentException
     * @throws FileNotFoundException
     */
    @Test
    public void testListAll() throws FileNotFoundException, IllegalArgumentException, IOException {

        FileStatus[] listStatus = fs.listStatus(new Path("/"));

        String flag = "d--             ";
        for (FileStatus fstatus : listStatus) {
            if (fstatus.isFile())  flag = "f--         ";
            System.out.println(flag + fstatus.getPath().getName());
        }
    }
}
```

4.4.2 通过流的方式访问 hdfs

```
/**
 * 相对那些封装好的方法而言的更底层一些的操作方式
 * 上层那些 mapreduce   spark 等运算框架，去 hdfs 中获取数据的时候，就是调的这种底层的 api
 * @author
 *
 */
public class StreamAccess {

    FileSystem fs = null;

    @Before
```

```
public void init() throws Exception {

        Configuration conf = new Configuration();
        fs = FileSystem.get(new URI("hdfs://hdp-node01:9000"), conf, "hadoop");

}



@Test
public void testDownLoadFileToLocal() throws IllegalArgumentException, IOException{

        //先获取一个文件的输入流----针对 hdfs 上的
        FSDataInputStream in = fs.open(new Path("/jdk-7u65-linux-i586.tar.gz"));

        //再构造一个文件的输出流----针对本地的
        FileOutputStream out = new FileOutputStream(new File("c:/jdk.tar.gz"));

        //再将输入流中数据传输到输出流
        IOUtils.copyBytes(in, out, 4096);



}



/**
  * hdfs 支持随机定位进行文件读取，而且可以方便地读取指定长度
  *  用于上层分布式运算框架并发处理数据
  * @throws IllegalArgumentException
  * @throws IOException
  */
@Test
public void testRandomAccess() throws IllegalArgumentException, IOException{
        //先获取一个文件的输入流----针对 hdfs 上的
        FSDataInputStream in = fs.open(new Path("/iloveyou.txt"));



        //可以将流的起始偏移量进行自定义
        in.seek(22);

        //再构造一个文件的输出流----针对本地的
        FileOutputStream out = new FileOutputStream(new File("c:/iloveyou.line.2.txt"));

        IOUtils.copyBytes(in,out,19L,true);
```

```
    }



    /**
     * 显示 hdfs 上文件的内容
     * @throws IOException
     * @throws IllegalArgumentException
     */
    @Test
    public void testCat() throws IllegalArgumentException, IOException{

        FSDataInputStream in = fs.open(new Path("/iloveyou.txt"));

        IOUtils.copyBytes(in, System.out, 1024);
    }
}
```

4.4.3 场景编程

在 mapreduce 、spark 等运算框架中，有一个核心思想就是将运算移往数据，或者说，就是要在并发计算中尽可能让运算本地化，这就需要获取数据所在位置的信息并进行相应范围读取

以下模拟实现：获取一个文件的所有 block 位置信息，然后读取指定 block 中的内容

```
    @Test
    public void testCat() throws IllegalArgumentException, IOException{

        FSDataInputStream in = fs.open(new Path("/weblog/input/access.log.10"));
        //拿到文件信息
        FileStatus[] listStatus = fs.listStatus(new Path("/weblog/input/access.log.10"));
        //获取这个文件的所有 block 的信息
        BlockLocation[] fileBlockLocations = fs.getFileBlockLocations(listStatus[0], 0L, listStatus[0].getLen());
        //第一个 block 的长度
        long length = fileBlockLocations[0].getLength();
        //第一个 block 的起始偏移量
        long offset = fileBlockLocations[0].getOffset();

        System.out.println(length);
        System.out.println(offset);
```

```
            //获取第一个 block 写入输出流
//          IOUtils.copyBytes(in, System.out, (int)length);
            byte[] b = new byte[4096];

            FileOutputStream os = new FileOutputStream(new File("d:/block0"));
            while(in.read(offset, b, 0, 4096)!=-1){
                os.write(b);
                offset += 4096;
                if(offset>=length) return;
            };
            os.flush();
            os.close();
            in.close();
        }
```