> 只有当知识写进你的长时记忆区，才是真正的学习

HDFS客户端操作 --- 文件操作

参数优先级测试

1.编写测试方法，设置文件副本数量

```
@Test
public void testCopyFromLocalFile() throws IOException, InterruptedException,
URISyntaxException {
        // 1 获取文件系统
        Configuration configuration = new Configuration();

        // 配置文件副本数为2
        configuration.set("dfs.replication", "2");

        FileSystem fs = FileSystem.get(new URI("hdfs://hadoop102:9000"),
configuration, "zhutiansama");

        // 2 上传文件
        fs.copyFromLocalFile(new Path("e:/data.txt"), new Path("/data.txt"));

        // 3 关闭资源
        fs.close();

        System.out.println("over");
}
```

2.将hdfs-site.xml拷贝到resources下，设置副本数为1

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
</configuration>
```

3.参数的优先级

> 参数优先级排序：（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

HDFS文件下载

```
@Test
public void testCopyToLocalFile() throws IOException, InterruptedException,
URISyntaxException{
```

```
        // 1 获取文件系统
        Configuration configuration = new Configuration();
        FileSystem fs = FileSystem.get(new URI("hdfs://hadoop102:9000"),
configuration, "zhutiansama");

        // 2 执行下载操作
        // boolean delSrc 指是否将原文件删除
        // Path src 指要下载的文件路径
        // Path dst 指将文件下载到的路径
        // boolean useRawLocalFileSystem 是否开启文件校验
        fs.copyToLocalFile(false, new Path("/data.txt"), new
Path("e:/data.txt"), true);

        // 3 关闭资源
        fs.close();
}
```

HDFS文件夹删除

```
@Test
public void testDelete() throws IOException, InterruptedException,
URISyntaxException{
    // 1 获取文件系统
    Configuration configuration = new Configuration();
    FileSystem fs = FileSystem.get(new URI("hdfs://hadoop102:9000"),
configuration, "zhutiansama");

    // 2 执行删除
    fs.delete(new Path("/input01/"), true);

    // 3 关闭资源
    fs.close();
}
```

HDFS文件名更名

```
@Test
public void testRename() throws IOException, InterruptedException,
URISyntaxException{
    // 1 获取文件系统
    Configuration configuration = new Configuration();
    FileSystem fs = FileSystem.get(new URI("hdfs://hadoop102:9000"),
configuration, "zhutiansama");

    // 2 修改文件名称
    fs.rename(new Path("/data.txt"), new Path("/datarename.txt"));

    // 3 关闭资源
    fs.close();
}
```

HDFS文件详情查看

```
@Test
public void testListFiles() throws IOException, InterruptedException,
URISyntaxException{

    // 1获取文件系统
    Configuration configuration = new Configuration();

    FileSystem fs = FileSystem.get(new URI("hdfs://hadoop102:9000"),
configuration, "zhutiansama");

    // 2 获取文件详情
    RemoteIterator<LocatedFileStatus> listFiles = fs.listFiles(new Path("/"),
true);

    while(listFiles.hasNext()){
        LocatedFileStatus status = listFiles.next();

        // 文件名称
        System.out.println(status.getPath().getName());
        // 长度
        System.out.println(status.getLen());
        // 权限
        System.out.println(status.getPermission());
        // 分组
        System.out.println(status.getGroup());

        // 获取存储的块信息
        BlockLocation[] blockLocations = status.getBlockLocations();

        for (BlockLocation blockLocation : blockLocations) {

            // 获取块存储的主机节点
            String[] hosts = blockLocation.getHosts();

            for (String host : hosts) {
                System.out.println(host);
            }
        }

    }
// 3 关闭资源
fs.close();
}
```

HDFS判断文件和文件夹

```
@Test
public void testListStatus() throws IOException, InterruptedException,
URISyntaxException{

    // 1 获取文件配置信息
    Configuration configuration = new Configuration();
    FileSystem fs = FileSystem.get(new URI("hdfs://hadoop102:9000"),
configuration, "zhutiansama");

    // 2 判断是文件还是文件夹
```

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

    for (FileStatus fileStatus : listStatus) {

        // 如果是文件
        if (fileStatus.isFile()) {
                System.out.println("f:"+fileStatus.getPath().getName());
            }else {
                System.out.println("d:"+fileStatus.getPath().getName());
            }
        }

    // 3 关闭资源
    fs.close();
}
```

> 上面学的API操都是框架封装好的。如果我们想自己实现上述API的操作应该用IO流方式