HDFS客户端操作2 文件操作

连接方式

主要有两种连接HDFS方式：

```
// 如果在resource中设置好了fs.defaultFS，则可以
Configuration conf = new Configuration();
FileSystem fs = FileSystem.get(conf);

//如果没有的话，可以
Configuration conf = new Configuration();
conf.set("fs.defaultFS", "hdfs://master:9000/")
FileSystem fs = FileSystem.get(conf);
```

```
//可以设置登录用户
Configuration configuration = new Configuration();
FileSystem fs = FileSystem.get(new URI("hdfs://master:9000"), configuration,
"hadoop");
```

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();

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

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

    //2. 上传文件
    fs.copyFromLocalFile(new Path("e:/input/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

```
<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下用户自定义配置文件（maven项目的resource文件夹下的.xm配置文件） >（3）服务器默认配置 >（4）默认配置

HDFS文件下载

```
@Test
public void testCopyToLocalFile() throws IOExcpetion, InterruptedException,
URIynaxException{
    //1 获取文件系统
    Configuration configuration = new Configuration();
    FileSystem fs = FileSystem.get(new URI("hdfs://master:9000"), configuration,
"hadoop");

    //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 useRwaLocalFileSystem 是否开启文件校验
    fs.copyToLocalFile(false, new Path("/data.txt"),
newPath("e:input/data.txt"), true);

    //3。关闭资源
    fs.close();
}
```

HDFS文件删除

```
@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://master:9000"),configuration,
"hadoop");

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

    //3. 关闭资源
    fs.colse();
}
```

HDFS文件名更改

```
@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://master:9000"),configuration,
"hadoop");

    //2.执行改名
    fs.rename(new Path("/data.txt"),new Path("/data01.txt"));

    //3. 关闭资源
    fs.colse();
}
```

HDFS文件详情查看

```
@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://master:9000"),configuration,
"hadoop");

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

    while(listFiles.hasNext()){
        LocatedFileStatus status = listFiles.next(); // 文件m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.colse();
}
```

HDFS判断文件和文件夹

```
@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://master:9000"),configuration,
"hadoop");

   // 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.colse();
}
```