# Hadoop 学习理解

----HDFS API 实现

笔者：电杆

2018 年 10 月 10 日

# 目录

# 1 使用环境

## 1.1 参考文档：

https://www.cnblogs.com/tyzmzlf/p/7304954.html

## 1.2 前提环境

编写 Java API 前需要先搭建好 hadoop 集群，并成功启动，搭建见文档：《20864_李杰_Hadoop 集群搭建》

## 1.3 编程环境

Eclipse 或者 IDEA

需要先搭建 maven ，搭建 Kevin 参考博客

https://www.cnblogs.com/lzx2509254166/p/7674455.html

https://www.cnblogs.com/lzx2509254166/p/7674455.html

## 2、API 实现

参考文档：

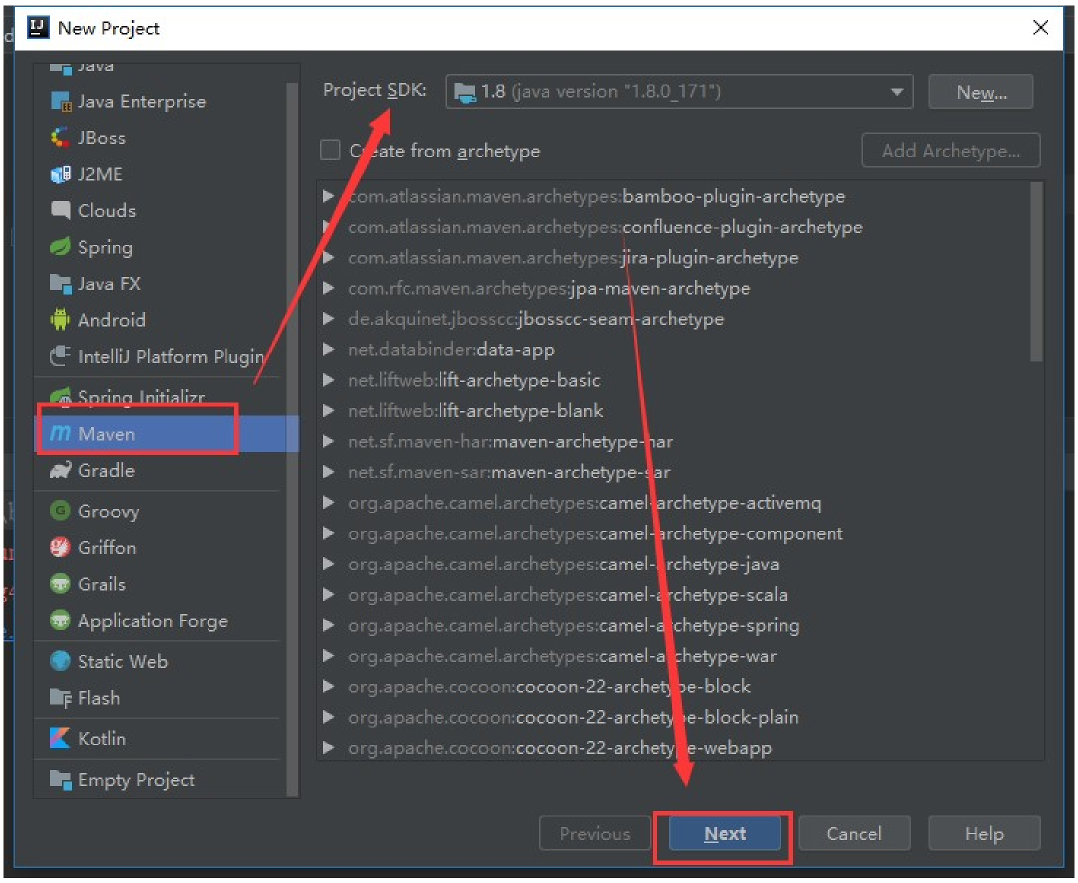
http://blog.51cto.com/jaydenwang/1842908

http://blog.fens.me/hadoop-hdfs-api/

### 2.1 新建 java 项目

使用 IDEA 新建 maven 项目：

选中自己的 jdk 然后下一步


New Project
Java
Java Enterprise
JBoss
J2ME
Clouds
Spring
Java FX
Android
IntelliJ Platform Plugin
Spring Initializr
Maven
Gradle
Groovy
Griffon
Grails
Application Forge
Static Web
Flash
Kotlin
Empty Project
Project SDK: 1.8 (java version "1.8.0_171")
New...
Create from archetype
Add Archetype...
com.atlassian.maven.archetypes:bamboo-plugin-archetype
com.atlassian.maven.archetypes:confluence-plugin-archetype
com.atlassian.maven.archetypes:jira-plugin-archetype
com.rfc.maven.archetypes:jpa-maven-archetype
de.akquinet.jbosscc:jbosscc-seam-archetype
net.databinder:data-app
net.liftweb:lift-archetype-basic
net.liftweb:lift-archetype-blank
net.sf.maven-har:maven-archetype-har
net.sf.maven-sar:maven-archetype-sar
org.apache.camel.archetypes:camel-archetype-activemq
org.apache.camel.archetypes:camel-archetype-component
org.apache.camel.archetypes:camel-archetype-java
org.apache.camel.archetypes:camel-archetype-scala
org.apache.camel.archetypes:camel-archetype-spring
org.apache.camel.archetypes:camel-archetype-war
org.apache.cocoon:cocoon-22-archetype-block
org.apache.cocoon:cocoon-22-archetype-block-plain
org.apache.cocoon:cocoon-22-archetype-webapp
Previous
Next
Cancel
Help

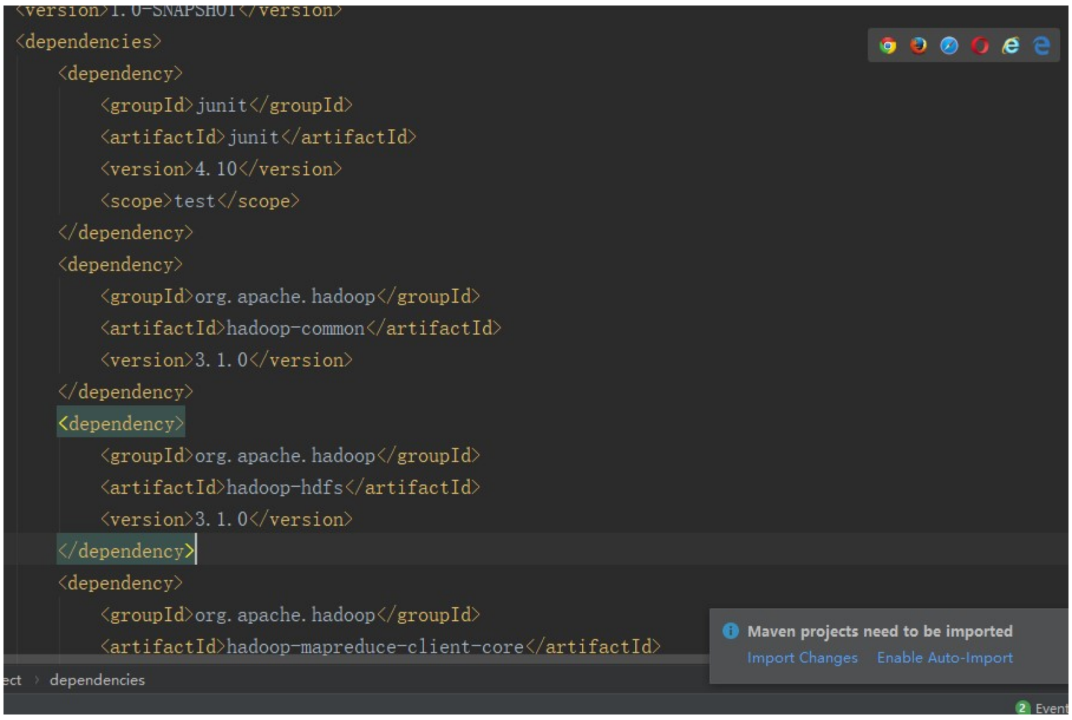
之后填写相应名字，下一步创建完成，

### 2.2 添加相应依赖

在 pom.xml 文件中添加如下配置

<dependencies>

```
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.10</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
            <version>3.1.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-hdfs</artifactId>
            <version>3.1.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-mapreduce-client-core</artifactId>
            <version>3.1.0</version>
        </dependency>
</dependencies>
```

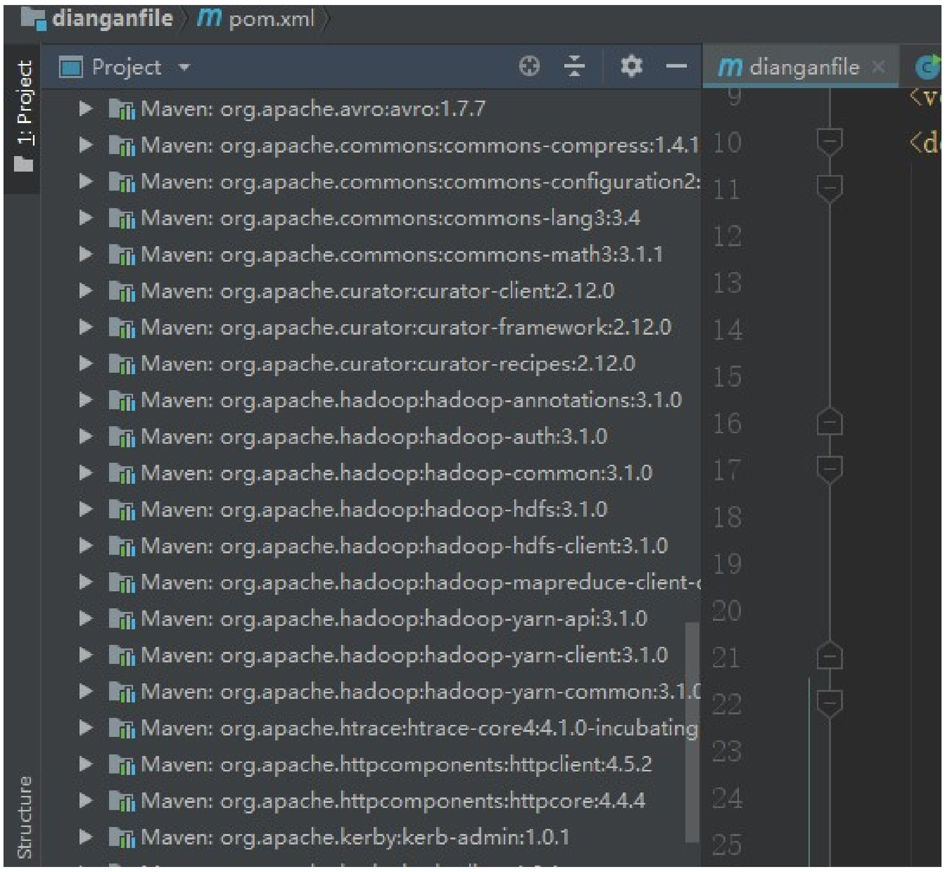


```
<version>1.0-SNAPSHOT</version>
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.10</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>3.1.0</version>
    </dependency>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-hdfs</artifactId>
        <version>3.1.0</version>
    </dependency>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-mapreduce-client-core</artifactId>
```

Maven projects need to be imported
Import Changes   Enable Auto-Import

添加依赖，并选择 import 下载依赖，等待下载完成，

可以看到 maven 自动下载的需要的文件

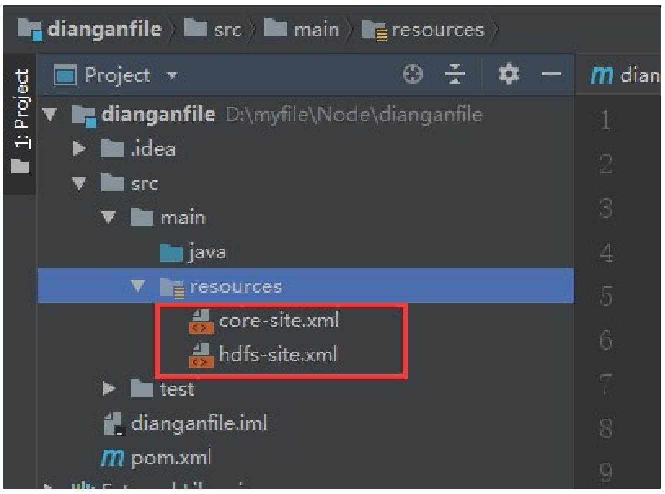


```
Maven: org.apache.avro:avro:1.7.7
Maven: org.apache.commons:commons-compress:1.4.1
Maven: org.apache.commons:commons-configuration2:
Maven: org.apache.commons:commons-lang3:3.4
Maven: org.apache.commons:commons-math3:3.1.1
Maven: org.apache.curator:curator-client:2.12.0
Maven: org.apache.curator:curator-framework:2.12.0
Maven: org.apache.curator:curator-recipes:2.12.0
Maven: org.apache.hadoop:hadoop-annotations:3.1.0
Maven: org.apache.hadoop:hadoop-auth:3.1.0
Maven: org.apache.hadoop:hadoop-common:3.1.0
Maven: org.apache.hadoop:hadoop-hdfs:3.1.0
Maven: org.apache.hadoop:hadoop-hdfs-client:3.1.0
Maven: org.apache.hadoop:hadoop-mapreduce-client-
Maven: org.apache.hadoop:hadoop-yarn-api:3.1.0
Maven: org.apache.hadoop:hadoop-yarn-client:3.1.0
Maven: org.apache.hadoop:hadoop-yarn-common:3.1.0
Maven: org.apache.htrace:htrace-core4:4.1.0-incubating
Maven: org.apache.httpcomponents:httpclient:4.5.2
Maven: org.apache.httpcomponents:httpcore:4.4.4
Maven: org.apache.kerby:kerb-admin:1.0.1
```

## 2.3 配置 configuration

将配置好的 hadoop 集群中的配置文件拷贝到 java 项目中 resources 处

Core-site.xml、hdfs-site.xml



## 2.4 编写测试 API 类

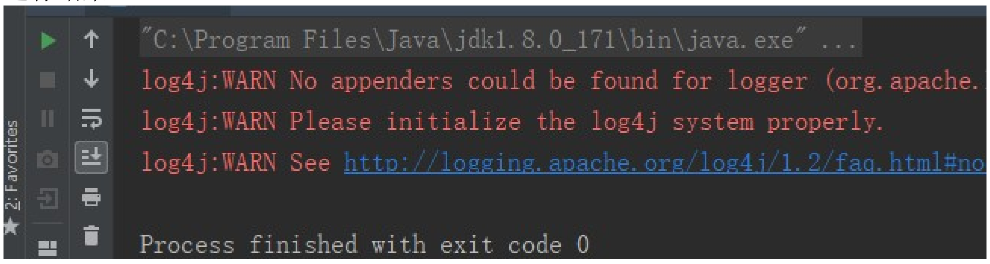
新建 package 并在包中新建 java class，

编写代码如下：

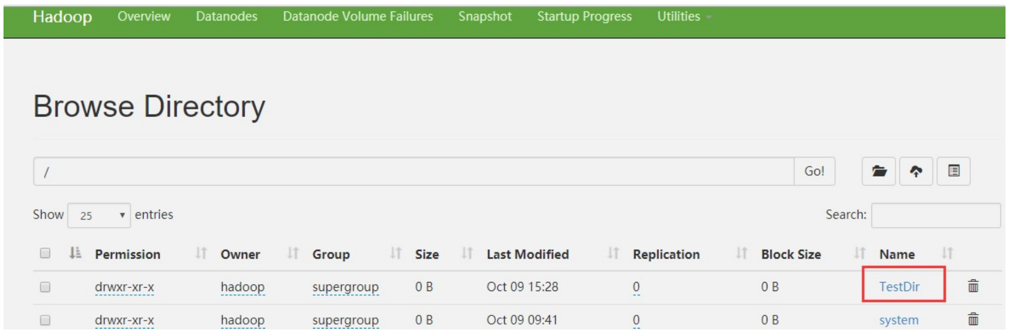
### 2.4.1 新建文件夹：

```
public static void createDir() throws Exception {
        Configuration conf = new Configuration();
        FileSystem hdfs = FileSystem.get(conf);
        Path dfs = new Path("/TestDir");
        hdfs.mkdirs(dfs);
    }
```

运行结果 ：



```
"C:\Program Files\Java\jdk1.8.0_171\bin\java.exe" ...
log4j:WARN No appenders could be found for logger (org.apache.
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#no

Process finished with exit code 0
```

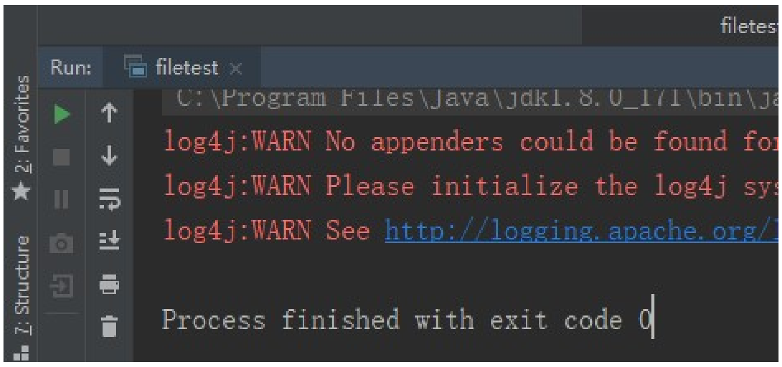


```
Hadoop  Overview  Datanodes  Datanode Volume Failures  Snapshot  Startup Progress  Utilities

Browse Directory

/                                                                 Go!

Show 25 entries                                                   Search:

Permission   Owner    Group        Size  Last Modified  Replication  Block Size  Name
drwxr-xr-x   hadoop   supergroup   0 B   Oct 09 15:28   0            0 B         TestDir
drwxr-xr-x   hadoop   supergroup   0 B   Oct 09 09:41   0            0 B         system
```

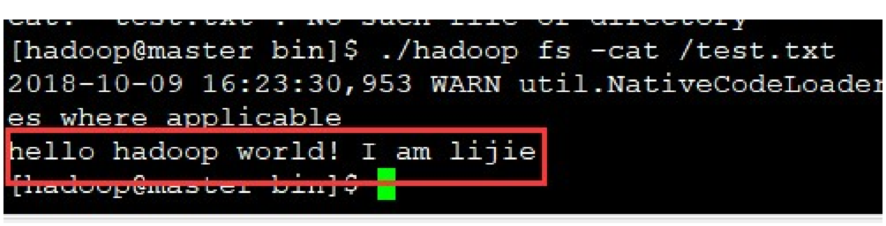
### 2.4.2 新建文件

```
public static void createFile() throws Exception {
        Configuration conf = new Configuration();
        FileSystem hdfs = FileSystem.get(conf);
        byte[] buff = "hello hadoop world! I am lijie\n".getBytes();
        Path dfs = new Path("/test.txt");
        FSDataOutputStream outputStream = hdfs.create(dfs);
        outputStream.write(buff, 0, buff.length);
    }
```

执行结果：



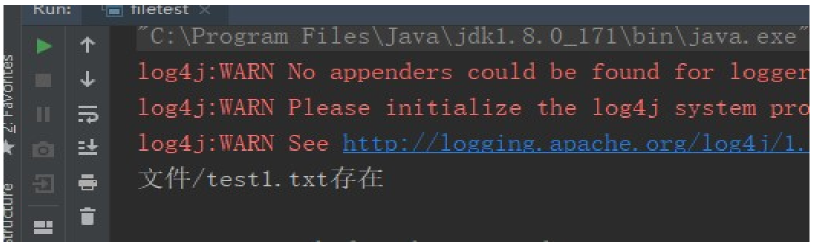
```
Run:  filetest
C:\Program Files\Java\jdk1.8.0_171\bin\ja
log4j:WARN No appenders could be found fo
log4j:WARN Please initialize the log4j sy
log4j:WARN See http://logging.apache.org/

Process finished with exit code 0
```



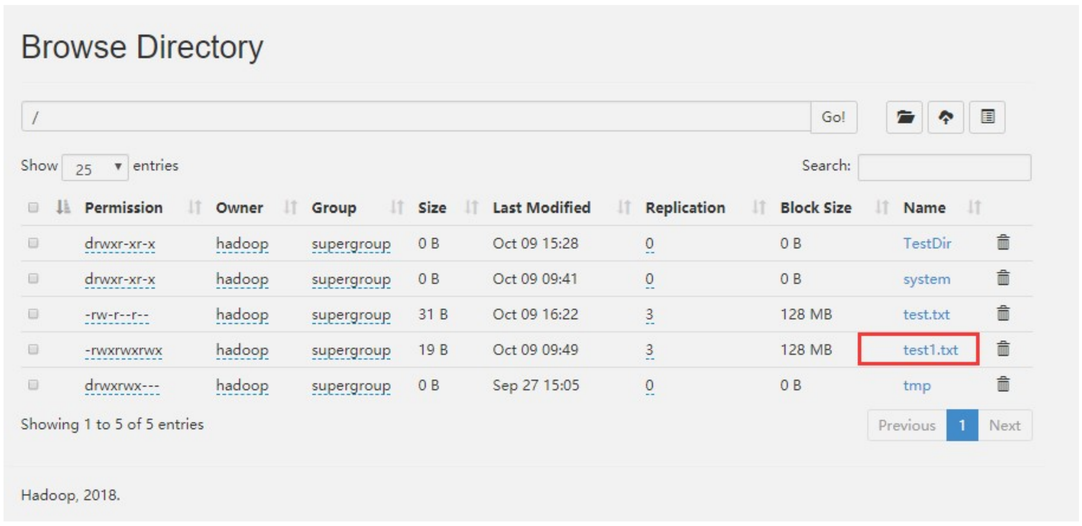
```
cat: test.txt : No such file or directory
[hadoop@master bin]$ ./hadoop fs -cat /test.txt
2018-10-09 16:23:30,953 WARN util.NativeCodeLoade
es where applicable
hello hadoop world! I am lijie
[hadoop@master bin]$
```

### 2.4.3 检查文件是否存在

```
public static void checkFile() throws Exception {
        Configuration conf = new Configuration();
        FileSystem hdfs = FileSystem.get(conf);
        Path findf = new Path("/test1.txt");
        boolean isExists = hdfs.exists(findf);
        if(isExists){
            System.out.println("文件"+findf+"存在");
        }
        else {
            System.out.println("文件"+findf+"不存在");
        }
    }
```



```
"C:\Program Files\Java\jdk1.8.0_171\bin\java.exe"
log4j:WARN No appenders could be found for logger
log4j:WARN Please initialize the log4j system pro
log4j:WARN See http://logging.apache.org/log4j/1.
文件/test1.txt存在
```



Browse Directory

/ Go!

Show 25 entries Search:

| Permission | Owner | Group | Size | Last Modified | Replication | Block Size | Name |
|---|---|---|---|---|---|---|---|
| drwxr-xr-x | hadoop | supergroup | 0 B | Oct 09 15:28 | 0 | 0 B | TestDir |
| drwxr-xr-x | hadoop | supergroup | 0 B | Oct 09 09:41 | 0 | 0 B | system |
| -rw-r--r-- | hadoop | supergroup | 31 B | Oct 09 16:22 | 3 | 128 MB | test.txt |
| -rwxrwxrwx | hadoop | supergroup | 19 B | Oct 09 09:49 | 3 | 128 MB | test1.txt |
| drwxrwx--- | hadoop | supergroup | 0 B | Sep 27 15:05 | 0 | 0 B | tmp |

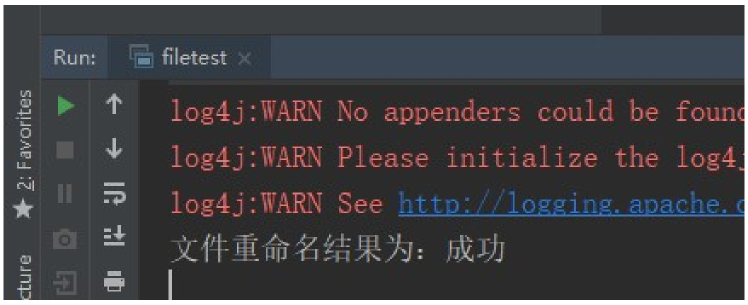
Showing 1 to 5 of 5 entries

Previous 1 Next

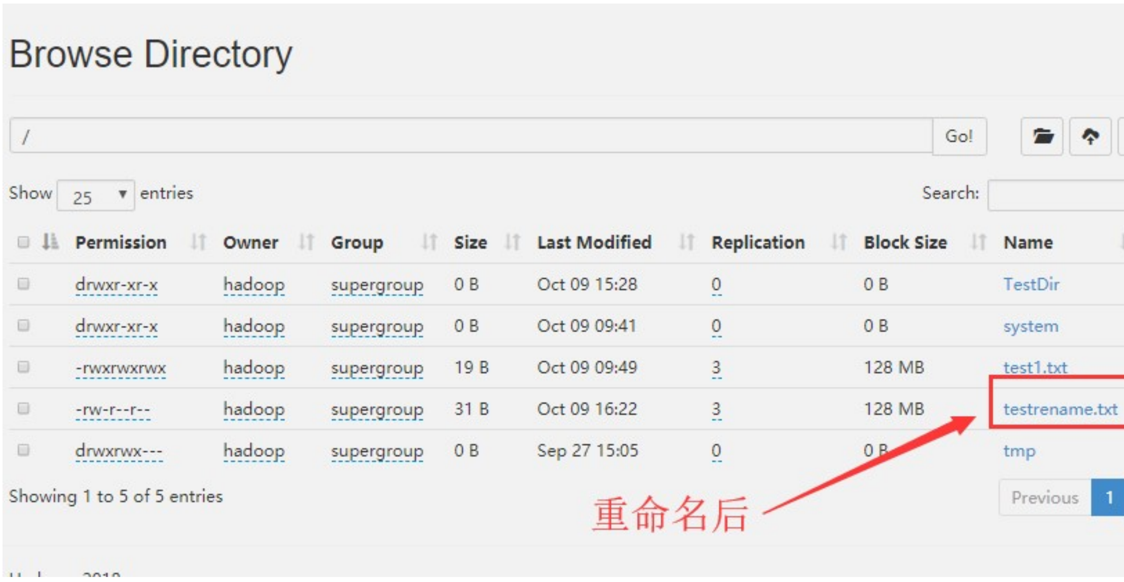
Hadoop, 2018.

### 2.4.4 文件重命名

```
public static void rename() throws Exception {
        Configuration conf = new Configuration();
        FileSystem hdfs = FileSystem.get(conf);
        Path frpaht = new Path("/test.txt");      //旧的文件名
        Path topath = new Path("/testrename.txt");      //新的文件名
        boolean isRename = hdfs.rename(frpaht, topath);
        String result = isRename ? "成功" : "失败";
        System.out.println("文件重命名结果为：" + result);
    }
```



```
log4j:WARN No appenders could be found
log4j:WARN Please initialize the log4j
log4j:WARN See http://logging.apache.c
文件重命名结果为：成功
```



Browse Directory

/ Go!

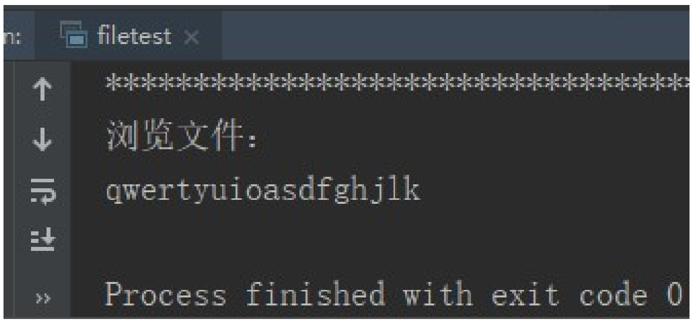
Show 25 entries Search:

| Permission | Owner | Group | Size | Last Modified | Replication | Block Size | Name |
|---|---|---|---|---|---|---|---|
| drwxr-xr-x | hadoop | supergroup | 0 B | Oct 09 15:28 | 0 | 0 B | TestDir |
| drwxr-xr-x | hadoop | supergroup | 0 B | Oct 09 09:41 | 0 | 0 B | system |
| -rwxrwxrwx | hadoop | supergroup | 19 B | Oct 09 09:49 | 3 | 128 MB | test1.txt |
| -rw-r--r-- | hadoop | supergroup | 31 B | Oct 09 16:22 | 3 | 128 MB | testrename.txt |
| drwxrwx--- | hadoop | supergroup | 0 B | Sep 27 15:05 | 0 | 0 B | tmp |

Showing 1 to 5 of 5 entries

### 2.4.5 查看文件

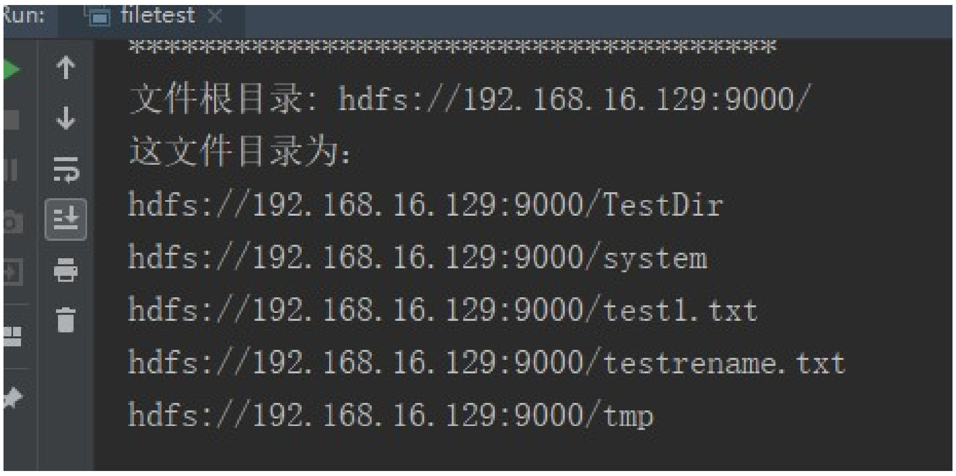
```
public static   void look() throws Exception{
        Configuration conf = new Configuration();
        FileSystem hdfs = FileSystem.get(conf);
        Path findf = new Path("/test1.txt");
        FSDataInputStream fsDataInputStream = hdfs.open(findf);
        System.out.println("************************************");
        System.out.println("浏览文件：");
        int c;
        while((c = fsDataInputStream.read()) != -1){
             System.out.print((char)c);
        }
        fsDataInputStream.close();
   }
```



```
***********************************
浏览文件:
qwertyuioasdfghjlk

Process finished with exit code 0
```

### 2.4.6 列出指定目录的文件列表

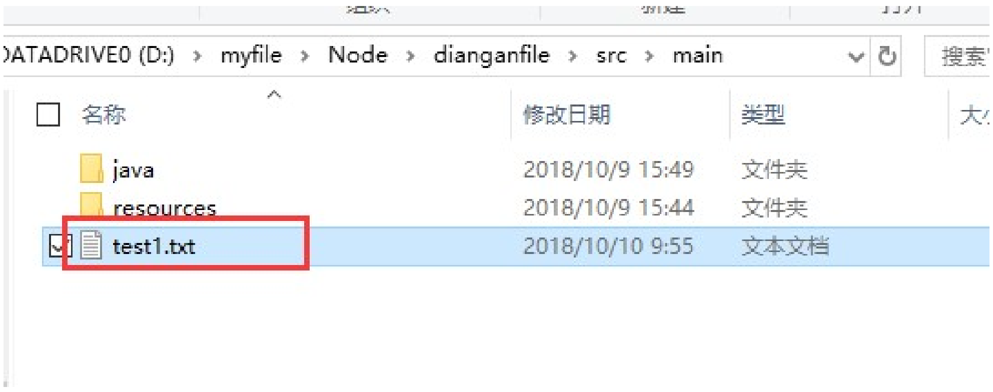
```
   public static void Catalog() throws Exception {
           Configuration conf = new Configuration();
           FileSystem hdfs = FileSystem.get(conf);
           Path findf = new Path("/");
           FileStatus fileStatus = hdfs.getFileStatus(findf);
           System.out.println("***********************************");
           System.out.println("文件根目录: " + fileStatus.getPath());
           System.out.println("这文件目录为：");
           for (FileStatus fs : hdfs.listStatus(findf)) {
                System.out.println(fs.getPath());
           }
      }
```



```
**************************************
文件根目录: hdfs://192.168.16.129:9000/
这文件目录为:
hdfs://192.168.16.129:9000/TestDir
hdfs://192.168.16.129:9000/system
hdfs://192.168.16.129:9000/test1.txt
hdfs://192.168.16.129:9000/testrename.txt
hdfs://192.168.16.129:9000/tmp
```

### 2.4.7 下载文件到本地目录

```
public static void download() throws Exception{
        Configuration conf = new Configuration();
        FileSystem hdfs = FileSystem.get(conf);
        Path findf = new Path("/test1.txt");
        InputStream in = hdfs.open(findf);
        OutputStream out = new
FileOutputStream("D://myfile/Node/dianganfile/src/main/test1.txt");
        IOUtils.copyBytes(in, out, 4096, true);
    }
```

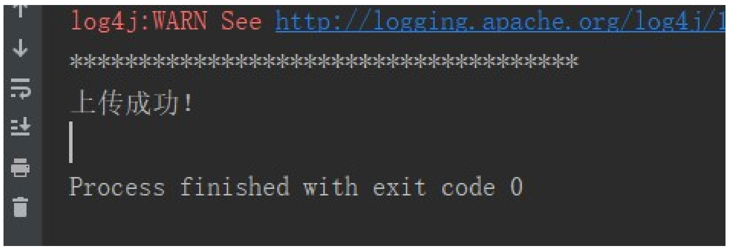


DATADRIVE0 (D:) › myfile › Node › dianganfile › src › main

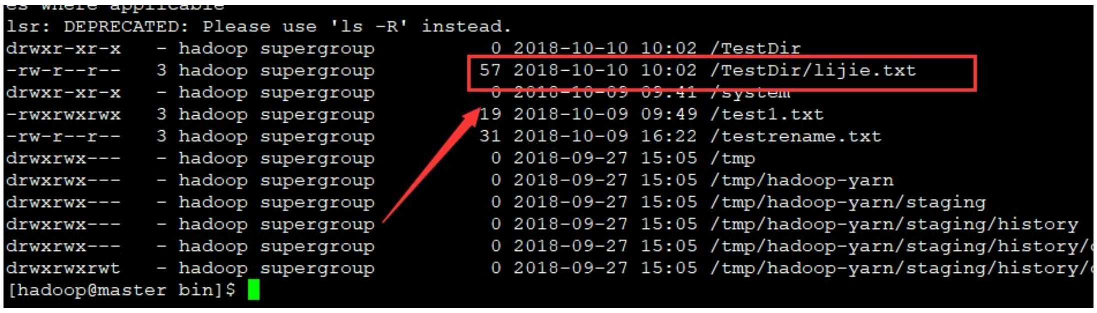
| 名称 | 修改日期 | 类型 |
|---|---|---|
| java | 2018/10/9 15:49 | 文件夹 |
| resources | 2018/10/9 15:44 | 文件夹 |
| test1.txt | 2018/10/10 9:55 | 文本文档 |

### 2.4.8 上传文件到 HDFS

```
public static void upload() throws Exception{
        Configuration conf = new Configuration();
        FileSystem hdfs = FileSystem.get(conf);
        Path srcPath = new Path("D://myfile/Node/dianganfile/src/main/lijie.txt");
        Path dstPath = new Path("/TestDir/lijie.txt");
        hdfs.copyFromLocalFile(false, srcPath, dstPath);
        hdfs.close();
        System.out.println("************************************");
        System.out.println("上传成功！");
    }
```



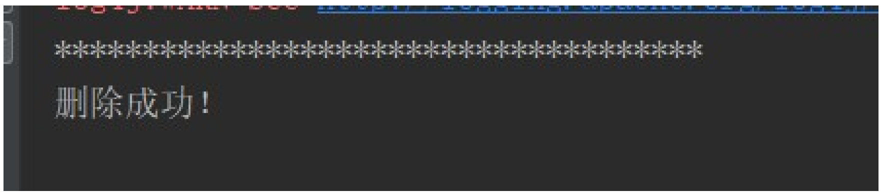
```
log4j:WARN See http://logging.apache.org/log4j/1
****************************************
上传成功！

Process finished with exit code 0
```



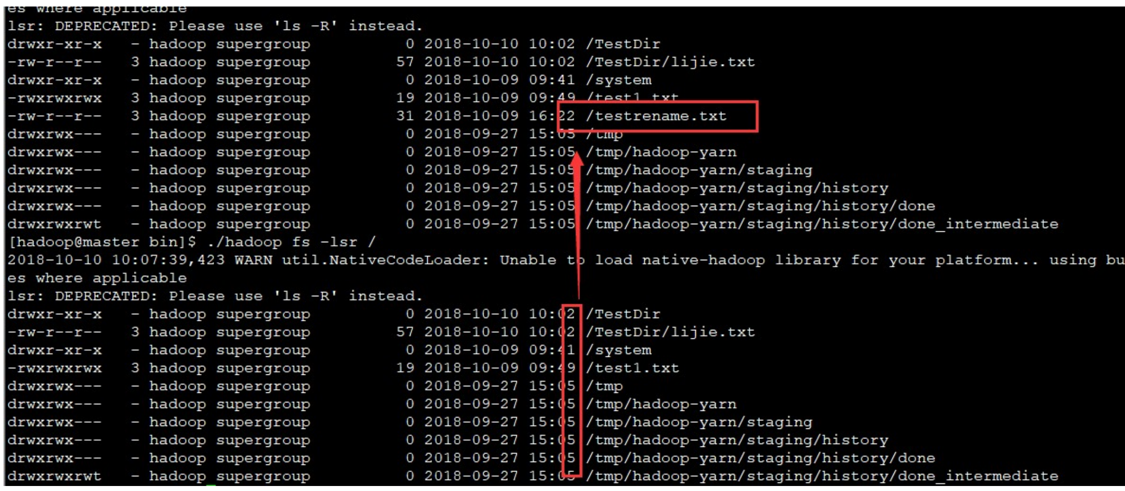
```
es where applicable
lsr: DEPRECATED: Please use 'ls -R' instead.
drwxr-xr-x   - hadoop supergroup          0 2018-10-10 10:02 /TestDir
-rw-r--r--   3 hadoop supergroup         57 2018-10-10 10:02 /TestDir/lijie.txt
drwxr-xr-x   - hadoop supergroup          0 2018-10-09 09:41 /system
-rwxrwxrwx   3 hadoop supergroup         19 2018-10-09 09:49 /test1.txt
-rw-r--r--   3 hadoop supergroup         31 2018-10-09 16:22 /testrename.txt
drwxrwx---   - hadoop supergroup          0 2018-09-27 15:05 /tmp
drwxrwx---   - hadoop supergroup          0 2018-09-27 15:05 /tmp/hadoop-yarn
drwxrwx---   - hadoop supergroup          0 2018-09-27 15:05 /tmp/hadoop-yarn/staging
drwxrwx---   - hadoop supergroup          0 2018-09-27 15:05 /tmp/hadoop-yarn/staging/history
drwxrwx---   - hadoop supergroup          0 2018-09-27 15:05 /tmp/hadoop-yarn/staging/history/
drwxrwxrwt   - hadoop supergroup          0 2018-09-27 15:05 /tmp/hadoop-yarn/staging/history/
[hadoop@master bin]$
```

### 2.4.9 删除 HDFS 文件

```
    public static void delete() throws Exception{
        Configuration conf = new Configuration();
        FileSystem hdfs = FileSystem.get(conf);
        Path path = new Path("hdfs://192.168.16.129:9000/testrename.txt");
        hdfs.delete(path,true);
        System.out.println("************************************");
        System.out.println("删除成功！");
    }
```



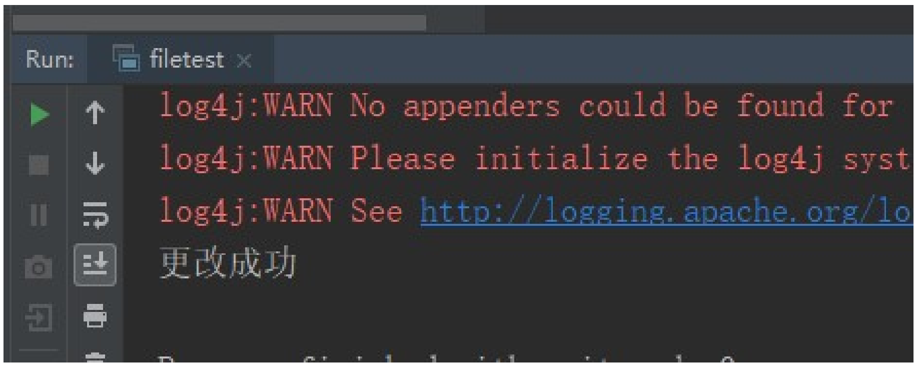
```
****************************************
删除成功！
```



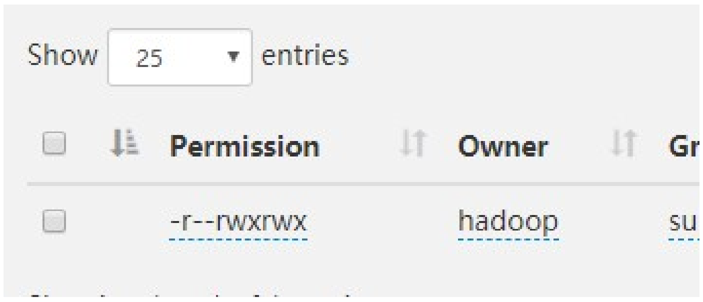
```
es where applicable
lsr: DEPRECATED: Please use 'ls -R' instead.
drwxr-xr-x   - hadoop supergroup          0 2018-10-10 10:02 /TestDir
-rw-r--r--   3 hadoop supergroup         57 2018-10-10 10:02 /TestDir/lijie.txt
drwxr-xr-x   - hadoop supergroup          0 2018-10-09 09:41 /system
-rwxrwxrwx   3 hadoop supergroup         19 2018-10-09 09:49 /test1.txt
-rw-r--r--   3 hadoop supergroup         31 2018-10-09 16:22 /testrename.txt
drwxrwx---   - hadoop supergroup          0 2018-09-27 15:05 /tmp
drwxrwx---   - hadoop supergroup          0 2018-09-27 15:05 /tmp/hadoop-yarn
drwxrwx---   - hadoop supergroup          0 2018-09-27 15:05 /tmp/hadoop-yarn/staging
drwxrwx---   - hadoop supergroup          0 2018-09-27 15:05 /tmp/hadoop-yarn/staging/history
drwxrwx---   - hadoop supergroup          0 2018-09-27 15:05 /tmp/hadoop-yarn/staging/history/done
drwxrwxrwt   - hadoop supergroup          0 2018-09-27 15:05 /tmp/hadoop-yarn/staging/history/done_intermediate
[hadoop@master bin]$ ./hadoop fs -lsr /
2018-10-10 10:07:39,423 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using bu
es where applicable
lsr: DEPRECATED: Please use 'ls -R' instead.
drwxr-xr-x   - hadoop supergroup          0 2018-10-10 10:02 /TestDir
-rw-r--r--   3 hadoop supergroup         57 2018-10-10 10:02 /TestDir/lijie.txt
drwxr-xr-x   - hadoop supergroup          0 2018-10-09 09:41 /system
-rwxrwxrwx   3 hadoop supergroup         19 2018-10-09 09:49 /test1.txt
drwxrwx---   - hadoop supergroup          0 2018-09-27 15:05 /tmp
drwxrwx---   - hadoop supergroup          0 2018-09-27 15:05 /tmp/hadoop-yarn
drwxrwx---   - hadoop supergroup          0 2018-09-27 15:05 /tmp/hadoop-yarn/staging
drwxrwx---   - hadoop supergroup          0 2018-09-27 15:05 /tmp/hadoop-yarn/staging/history
drwxrwx---   - hadoop supergroup          0 2018-09-27 15:05 /tmp/hadoop-yarn/staging/history/done
drwxrwxrwt   - hadoop supergroup          0 2018-09-27 15:05 /tmp/hadoop-yarn/staging/history/done_intermediate
```

### 2.4.10 更改文件权限

```
public   static void UpdatePermission ()throws IOException{
        Configuration conf = new Configuration();
        FileSystem hdfs = FileSystem.get(conf);
        Path findf = new Path("/TestDir/lijie.txt");
      //   FsPermission permission = new FsPermission(FsAction.ALL,FsAction.ALL,FsAction.ALL)
        if(hdfs.exists(findf)){
            hdfs.setPermission(findf,new FsPermission(FsAction.ALL,FsAction.ALL,FsAction.ALL));
        }
    }
```



```
log4j:WARN No appenders could be found for
log4j:WARN Please initialize the log4j syst
log4j:WARN See http://logging.apache.org/lo
更改成功
```


Show 25 entries

| | Permission | Owner | Gr |
|---|---|---|---|
| | -r--rwxrwx | hadoop | su |

### 2.4.11 所有代码：

```
package Filetest;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.hdfs.DistributedFileSystem;
import org.apache.hadoop.hdfs.protocol.DatanodeInfo;
import org.apache.hadoop.io.IOUtils;
import java.io.*;
public class filetest {
    public static void main(String[] args) throws Exception {

        //checkFile();
        //getFileLocation();
        //Catalog();
        look();
        //download();
        //upload();
        //delete();
        //suploadWithStream();
        //listFilestest();
    }
    public static void createFile() throws Exception {
        Configuration conf = new Configuration();
        FileSystem hdfs = FileSystem.get(conf);
        byte[] buff = "hello hadoop world! I am lijie\n".getBytes();
        Path dfs = new Path("/test.txt");
        FSDataOutputStream outputStream = hdfs.create(dfs);
        outputStream.write(buff, 0, buff.length);
        hdfs.close();
    }
    public static void createDir() throws Exception {
        Configuration conf = new Configuration();
        FileSystem hdfs = FileSystem.get(conf);
        Path dfs = new Path("/TestDir");
        hdfs.mkdirs(dfs);
        hdfs.close();
    }
    public static void rename() throws Exception {
        Configuration conf = new Configuration();
        FileSystem hdfs = FileSystem.get(conf);
        Path frpaht = new Path("/test.txt");      //旧的文件名
        Path topath = new Path("/testrename.txt");      //新的文件名
        boolean isRename = hdfs.rename(frpaht, topath);
        String result = isRename ? "成功" : "失败";
        System.out.println("文件重命名结果为： " + result);
        hdfs.close();
    }
    /**
    * 检索文件是否存在
    */
    public static void checkFile() throws Exception {
        Configuration conf = new Configuration();
        FileSystem hdfs = FileSystem.get(conf);
```

```
		Path findf = new Path("/test1.txt");

		boolean isExists = hdfs.exists(findf);

		if (isExists) {

			System.out.println("文件" + findf + "存在");

		} else {

			System.out.println("文件" + findf + "不存在");

		}

		hdfs.close();

	}

	public static void getModifyTime() throws Exception {

		Configuration conf = new Configuration();

		FileSystem hdfs = FileSystem.get(conf);

		Path fpath = new Path("/test1.txt");

		FileStatus fileStatus = hdfs.getFileStatus(fpath);

		long modiTime = fileStatus.getModificationTime();

		System.out.println("test.txt 的修改时间是" + modiTime);

		hdfs.close();

	}

	/**

	 * 通过"FileSystem.getFileBlockLocation（FileStatus file，long start，long len）

	 * 可查找指定文件在 HDFS 集群上的位置，其中 file 为文件的完整路径，start 和 len
来标识查找文件的路径。具体实现如下

	 */

	public static void getFileLocation() throws Exception {

		Configuration conf = new Configuration();

		FileSystem hdfs = FileSystem.get(conf);

		Path fpath = new Path("/test1.txt");

		FileStatus filestatus = hdfs.getFileStatus(fpath);

		BlockLocation[] blkLocations = hdfs.getFileBlockLocations(filestatus, 0,
filestatus.getLen());
```

```
        int blockLen = blkLocations.length;

        for (int i = 0; i < blockLen; i++) {

            String[] hosts = blkLocations[i].getHosts();

            System.out.println("block_" + i + "_location:" + hosts[0]);

        }

        hdfs.close();

    }

    /**

     * 通过"DatanodeInfo.getHostName（）"可获取 HDFS 集群上的所有节点名称

     */

    public static void getList() throws Exception {

        Configuration conf = new Configuration();

        FileSystem fs = FileSystem.get(conf);

        DistributedFileSystem hdfs = (DistributedFileSystem) fs;

        DatanodeInfo[] dataNodeStats = hdfs.getDataNodeStats();

        for (int i = 0; i < dataNodeStats.length; i++) {

            System.out.println("DataNode_" + i + "_Name:" +
dataNodeStats[i].getHostName());

        }

        hdfs.close();

    }

    public static void listFilestest() throws Exception {

        Configuration conf = new Configuration();

        FileSystem hdfs = FileSystem.get(conf);

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

        for (FileStatus fileStatus : fileStatuses) {

            System.out.println("这是一个：" + (fileStatus.isDirectory() ? "文件夹" : "文件
"));

            System.out.println("副本系数：" + fileStatus.getReplication());

            System.out.println("大小：" + fileStatus.getLen());
```

```
			System.out.println("路径：" + fileStatus.getPath() + "\n");
		}
		hdfs.close();
	}
	/**
	 * 列出指定目录的文件列表
	 */
	public static void Catalog() throws Exception {
		Configuration conf = new Configuration();
		FileSystem hdfs = FileSystem.get(conf);
		Path findf = new Path("/");
		FileStatus fileStatus = hdfs.getFileStatus(findf);
		System.out.println("*************************************");
		System.out.println("文件根目录: " + fileStatus.getPath());
		System.out.println("这文件目录为：");
		for (FileStatus fs : hdfs.listStatus(findf)) {
			System.out.println(fs.getPath());
		}
		hdfs.close();
	}
	/**
	 * 查看文件内容
	 */
	public static  void look() throws Exception{
		Configuration conf = new Configuration();
		FileSystem hdfs = FileSystem.get(conf);
		Path findf = new Path("/test1.txt");
		FSDataInputStream fsDataInputStream = hdfs.open(findf);
		System.out.println("*************************************");
```

```
            System.out.println("浏览文件：");
            int c;
            while((c = fsDataInputStream.read()) != -1){
                System.out.print((char)c);
            }
            fsDataInputStream.close();
            hdfs.close();
    }
    /**
     * 下载 HDFS 文件至本地指定目录
     */
    public static void download() throws Exception{
            Configuration conf = new Configuration();
            FileSystem hdfs = FileSystem.get(conf);
            Path findf = new Path("/test1.txt");
            InputStream in = hdfs.open(findf);
            OutputStream out = new
FileOutputStream("D://myfile/Node/dianganfile/src/main/test1.txt");
            IOUtils.copyBytes(in, out, 4096, true);
            hdfs.close();
    }
    /**
     * 上传文件至 HDFS 指定目录
     */
    public static void upload() throws Exception{
            Configuration conf = new Configuration();
            FileSystem hdfs = FileSystem.get(conf);
            Path srcPath = new Path("D://myfile/Node/dianganfile/src/main/lijie.txt");
            Path dstPath = new Path("/TestDir/lijie.txt");
```

```
		hdfs.copyFromLocalFile(false, srcPath, dstPath);
		hdfs.close();
		System.out.println("************************************");
		System.out.println("上传成功！ ");
		hdfs.close();
	}
	/**
	 * 删除指定 HDFS 文件
	 */
	public static void delete() throws Exception{
		Configuration conf = new Configuration();
		FileSystem hdfs = FileSystem.get(conf);
		Path path = new Path("hdfs://192.168.16.129:9000/testrename.txt");
		boolean isExists = hdfs.exists(path);
		hdfs.delete(path,true);
		System.out.println("************************************");
		System.out.println("删除成功！ ");
		hdfs.close();
	}
	public void listFiles() throws IOException {
		Configuration conf = new Configuration();
		FileSystem hdfs = FileSystem.get(conf);
		String dirName = "/test1";
		Path f = new Path(dirName);
		FileStatus[] status = hdfs.listStatus(f);
		System.out.println(dirName + " has all files:");
		if (status.length == 0) {
			System.out.println("nothing !");
		} else {
```

```
            for (int i = 0; i < status.length; i++) {
                System.out.println(status[i].getPath().toString());
            }
        }
        hdfs.close();
    }
    /**
     *  重写文件
     * @throws IOException
     */
    public static void uploadWithStream() throws IOException {
        Configuration conf = new Configuration();
        FileSystem hdfs = FileSystem.get(conf);
        Path topath = new Path("/test1.txt");
        String frompath = "D://myfile/Node/dianganfile/src/main/lijie.txt";
        //HDFS 上的文件流
        FSDataOutputStream outputStream = hdfs.create(topath, true);
        //本地读取的文件流
        FileInputStream inputStream = new FileInputStream(frompath);
        //将输入文件流写到输出文件流
        IOUtils.copyBytes(inputStream,outputStream,4096,false);
        hdfs.close();
    }
    public   static void UpdatePermission ()throws IOException{
            Configuration conf = new Configuration();
            FileSystem hdfs = FileSystem.get(conf);
            Path findf = new Path("/TestDir/lijie.txt");
        //    FsPermission permission = new
FsPermission(FsAction.ALL,FsAction.ALL,FsAction.ALL)
```

```
		if(hdfs.exists(findf)){
			hdfs.setPermission(findf,new
FsPermission(FsAction.ALL,FsAction.ALL,FsAction.ALL));
			System.out.println("更改成功");
		}
		else{
			System.out.println("文件不存在");
		}
	}
}
```

## 3、常见错误：

### 3.1 运行时发生错误：: Server IPC version 9 cannot communicate with client version 4

题的根源在于，工程当中 maven dependencies 里面的包，有个 hadoop-core 的包，版本太低，这样，程序里面所有引用到 org.apache.hadoop 的地方，都是低版本的，你用的是 maven3 的话，默认是 hadoop-core-1.2.1.jar,这个就是那个“ipc client version4”，而一般情况下你的电脑里运行的 hadoop 都是 2.x,显然版本不对

可以在配置文件中配置：

hadoop-common、hadoop-hdfs、hadoop-mapreduce-client-core 然后删除 hadoop-core

配置如下：

```
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.10</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
            <version>3.1.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-hdfs</artifactId>
            <version>3.1.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-mapreduce-client-core</artifactId>
            <version>3.1.0</version>
        </dependency>
    </dependencies>
```

## 3.1 在执行写入时出现权限问题：AccessControlException

```
org.apache.hadoop.security.AccessControlException: Permission denied: user=diangan, access=WRITE, inode="/":hadoop:supergroup:drwxr-xr-x
```

是因为编写代码的主机用户，不具备 hadoop 所在主机用户的权限，因此发生错误，

解决办法大概有三种：

1、在系统的环境变量或 java JVM 变量里面添加 HADOOP_USER_NAME，这个值具体等于多少看自己的情况，以后会运行 HADOOP 上的 Linux 的用户名。（修改完重启 eclipse，不然可能不生效）

2、将当前系统的帐号修改为 hadoop

3、使用 HDFS 的命令行接口修改相应目录的权限，hadoop fs -chmod 777 /user,

后面的/user 是要上传文件的路径，不同的情况可能不一样，比如要上传的文件路径为 hdfs://namenode/user/xxx.doc，则这样的修改可以，如果要上传的文件路径为 hdfs://namenode/java/xxx.doc，则要修改的为 hadoop fs -chmod 777 /java 或者 hadoop fs -chmod 777 /，java 的那个需要先在 HDFS 里面建立 Java 目录，后面的这个是为根目录调整权限。

问题详解可参考博客：

https://blog.csdn.net/xiaoshunzi111/article/details/52062640

## 参考

本文档参考许多网上博客，致谢！！先后不分等级，帮助同等珍贵！

https://www.cnblogs.com/tyzmzlf/p/7304954.html

https://www.cnblogs.com/lzx2509254166/p/7674455.html

https://www.cnblogs.com/lzx2509254166/p/7674455.html

http://blog.51cto.com/jaydenwang/1842908

http://blog.fens.me/hadoop-hdfs-api/

https://blog.csdn.net/xiaoshunzi111/article/details/52062640

电杆