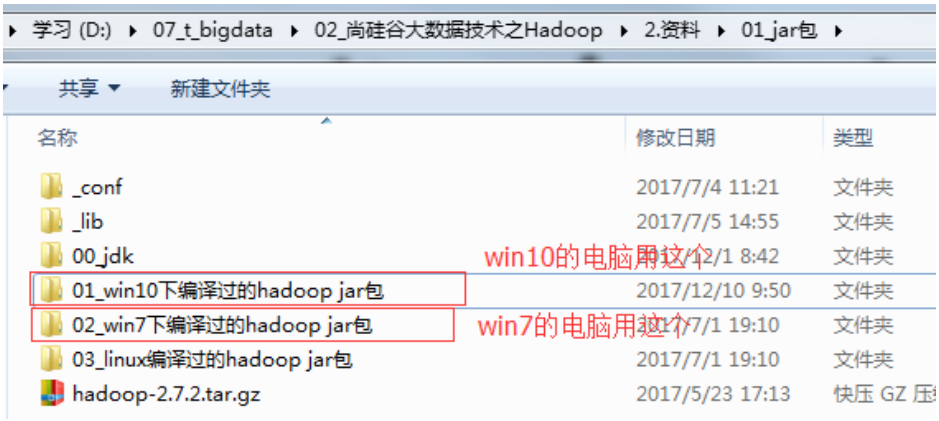
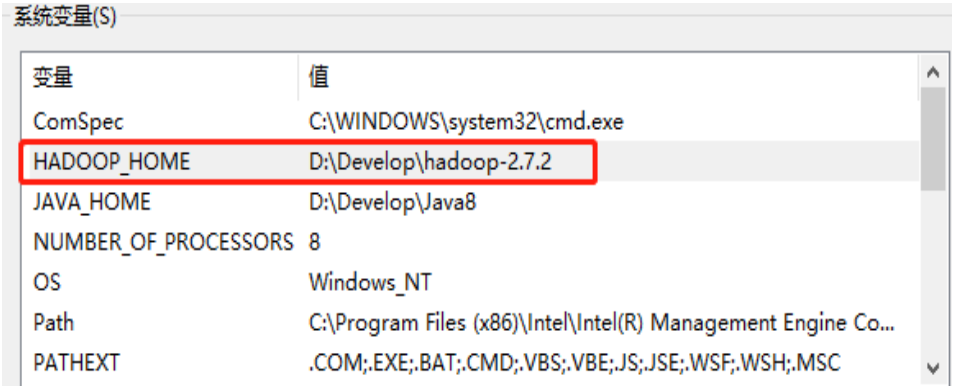
三 HDFS 客户端操作

3.1 HDFS 客户端环境准备

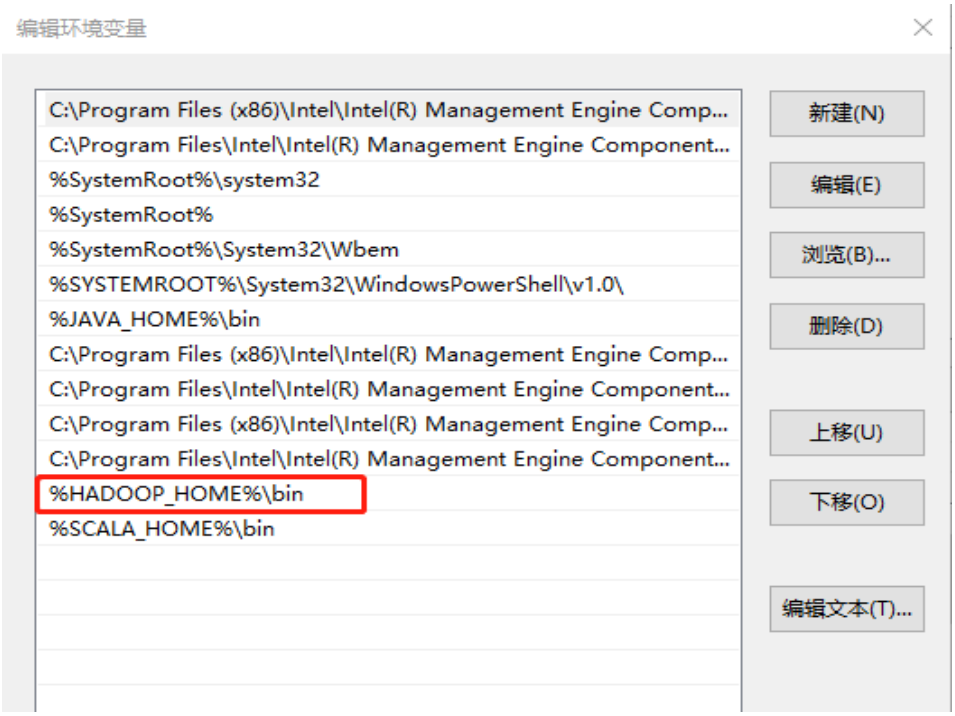
1）根据自己电脑的操作系统拷贝对应的编译后的 hadoop jar 包到非中文路径（例如：D:\Develop\hadoop-2.7.2）。



2）配置 HADOOP_HOME 环境变量



2）配置 Path 环境变量



4）创建一个 Maven 工程 HdfsClientDemo

5）导入相应的依赖

```
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
            <version>2.8.2</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
            <version>2.7.2</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-client</artifactId>
            <version>2.7.2</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
```

```
            <artifactId>hadoop-hdfs</artifactId>
            <version>2.7.2</version>
        </dependency>
    </dependencies>
```

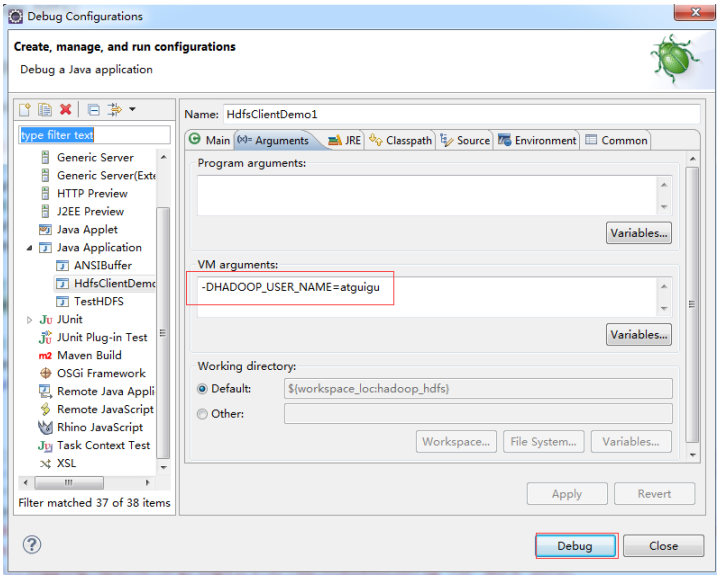
6）创建包名：com.atguigu.hdfs

7）创建 HdfsClient 类

```
public class HdfsClient{
@Test
    public void testMkdirs() throws IOException, InterruptedException, URISyntaxException{

        // 1 获取文件系统
        Configuration configuration = new Configuration();
        // 配置在集群上运行
        // configuration.set("fs.defaultFS", "hdfs://hadoop102:9000");
        // FileSystem fs = FileSystem.get(configuration);

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

        // 2 创建目录
        fs.mkdirs(new Path("/1108/daxian/banzhang"));

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

8）执行程序

运行时需要配置用户名称



客户端去操作 hdfs 时，是有一个用户身份的。默认情况下，hdfs 客户端 api 会从 jvm 中获取一个参数来作为自己的用户身份：-DHADOOP_USER_NAME=atguigu，atguigu 为用户名称。

8）注意：如果 eclipse 打印不出日志，在控制台上只显示

```
1.log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
2.log4j:WARN Please initialize the log4j system properly.
3.log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
```

需要在项目的 src/main/resources 目录下，新建一个文件，命名为“log4j.properties”，在文件中填入

```
log4j.rootLogger=INFO, stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d %p [%c] - %m%n
log4j.appender.logfile=org.apache.log4j.FileAppender
log4j.appender.logfile.File=target/spring.log
log4j.appender.logfile.layout=org.apache.log4j.PatternLayout
log4j.appender.logfile.layout.ConversionPattern=%d %p [%c] - %m%n
```

3.2 HDFS 的 API 操作

3.2.1 HDFS 文件上传（测试参数优先级）

1）编写源代码

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

```
        configuration.set("dfs.replication", "2");
        FileSystem fs = FileSystem.get(new URI("hdfs://hadoop102:9000"), configuration, "atguigu");

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

        // 3 关闭资源
        fs.close();

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

2）将 hdfs-site.xml 拷贝到项目的根目录下

```
<?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

3.2.2 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, "atguigu");

        // 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("/hello1.txt"), new Path("e:/hello1.txt"), true);
```

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

3.2.3 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, "atguigu");

        // 2 执行删除
        fs.delete(new Path("/1108/"), true);

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

3.2.4 HDFS 文件名更改

```
    @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, "atguigu");

        // 2 修改文件名称
        fs.rename(new Path("/hello.txt"), new Path("/hello6.txt"));

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

3.2.5 HDFS 文件详情查看

查看文件名称、权限、长度、块信息

```
    @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, "atguigu");

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

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

            // 输出详情
            // 文件名称
            System.out.println(status.getPath().getName());
            // 长度
            System.out.println(status.getLen());
            // 权限
            System.out.println(status.getPermission());
            // z 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);
                }
            }

            System.out.println("----------------班长的分割线----------");
        }
    }
```

3.2.6 HDFS 文件和文件夹判断

```
    @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,
```

```
"atguigu");

        // 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();
    }
```

3.3 HDFS 的 I/O 流操作

3.3.1 HDFS 文件上传

```
    @Test
    public void putFileToHDFS() 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,
"atguigu");

        // 2 创建输入流
        FileInputStream fis = new FileInputStream(new File("e:/hello.txt"));

        // 3 获取输出流
        FSDataOutputStream fos = fs.create(new Path("/hello4.txt"));

        // 4 流对拷
        IOUtils.copyBytes(fis, fos, configuration);

        // 5 关闭资源
        IOUtils.closeStream(fis);
        IOUtils.closeStream(fos);
    }
```

3.3.2 HDFS 文件下载

1）需求：从 HDFS 上下载文件到本地 e 盘上。

2）编写代码：

```
    // 文件下载
    @Test
    public void getFileFromHDFS() 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, "atguigu");

        // 2 获取输人流
        FSDataInputStream fis = fs.open(new Path("/hello1.txt"));

        // 3 获取输出流
        FileOutputStream fos = new FileOutputStream(new File("e:/hello1.txt"));

        // 4 流的对拷
        IOUtils.copyBytes(fis, fos, configuration);

        // 5 关闭资源
        IOUtils.closeStream(fis);
        IOUtils.closeStream(fos);
        fs.close();
    }
```

3.3.3 定位文件读取

1）下载第一块

```
    @Test
    public void readFileSeek1() 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, "atguigu");

        // 2 获取输入流
        FSDataInputStream fis = fs.open(new Path("/hadoop-2.7.2.tar.gz"));

        // 3 创建输出流
        FileOutputStream fos = new FileOutputStream(new
```

```
File("e:/hadoop-2.7.2.tar.gz.part1"));

        // 4 流的拷贝
        byte[] buf = new byte[1024];

        for(int i =0 ; i < 1024 * 128; i++){
            fis.read(buf);
            fos.write(buf);
        }

        // 5 关闭资源
        IOUtils.closeStream(fis);
        IOUtils.closeStream(fos);
    }
```

2）下载第二块

```
    @Test
    public void readFileSeek2() 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, "atguigu");

        // 2 打开输入流
        FSDataInputStream fis = fs.open(new Path("/hadoop-2.7.2.tar.gz"));

        // 3 定位输入数据位置
        fis.seek(1024*1024*128);

        // 4 创建输出流
        FileOutputStream fos = new FileOutputStream(new File("e:/hadoop-2.7.2.tar.gz.part2"));

        // 5 流的对拷
        IOUtils.copyBytes(fis, fos, configuration);

        // 6 关闭资源
        IOUtils.closeStream(fis);
        IOUtils.closeStream(fos);
    }
```

3）合并文件

在 window 命令窗口中执行

type hadoop-2.7.2.tar.gz.part2 >> hadoop-2.7.2.tar.gz.part1