

实验二．HDFS文件的创建和写入

一.实验介绍

1.关于本实验

将本地磁盘的文件内容写入到HDFS上的新文件中

2.实验目的

- 熟悉Linux命令和CentOS操作系统.
- 熟悉JavaSE基础.
- 熟悉HDFS和Hadoop原理,掌握HDFS的API使用.

3.实验环境

- 服务器:华为云 云服务器x3
- 操作系统: CentOS 7.8
- Hadoop版本: Apache Hadoop 2.7.7
- 编译器: IntelliJ IDEA










4.Hadoop部署情况(仅作参考,以个人实际部署为准)

	0	1	2
HDFS	<u>NameNode</u>		<u>SecondaryNameNode</u>
	<u>DataNode</u>	<u>DataNode</u>	<u>DataNode</u>
YARN		<u>ResourceManager</u>	
	<u>NodeManager</u>	<u>NodeManager</u>	<u>NodeManager</u>

二.实验步骤

1.打开服务器虚拟机并启动HDFS

1.1 在华为云控制台启动三台服务器

名称/ID	监控	可用区	状态	规格/镜像
 e-0001		可用区1	 运行中	1vCPUs 2GiB 16.medium.2 CentOS 7.8 64bit
 e-0002		可用区1	 运行中	1vCPUs 2GiB 16.medium.2 CentOS 7.8 64bit
 e-0003		可用区1	 运行中	1vCPUs 2GiB 16.medium.2 CentOS 7.8 64bit

1.2 在NameNode所在节点启动HDFS

例如0号服务器，wyd-0-202214xxxx服务器,执行start-dfs.sh

```
start-dfs.sh
```

```
wyd@wyd-0-202214xxxx ~]$ start-dfs.sh
Starting namenodes on [wyd-0-202214xxxx]
wyd-0-202214xxxx: starting namenode, logging to /opt/module/hadoop-2.7.7/logs/hadoop-wyd-namenode-wyd-0-202214xxxx.out
wyd-2-202214xxxx: starting datanode, logging to /opt/module/hadoop-2.7.7/logs/hadoop-wyd-datanode-wyd-2-202214xxxx.out
wyd-1-202214xxxx: starting datanode, logging to /opt/module/hadoop-2.7.7/logs/hadoop-wyd-datanode-wyd-1-202214xxxx.out
wyd-0-202214xxxx: starting datanode, logging to /opt/module/hadoop-2.7.7/logs/hadoop-wyd-datanode-wyd-0-202214xxxx.out
Starting secondary namenodes [wyd-2-202214xxxx]
wyd-2-202214xxxx: starting secondarynamenode, logging to /opt/module/hadoop-2.7.7/logs/hadoop-wyd-secondarynamenode-wyd-2-202214xxxx.out
```

1.3 在三台服务器分别执行jps命令

例如0号服务器，wyd-0-202214xxxx服务器,执行jps

```
wyd@wyd-0-202214xxxx ~]$ jps
3092 DataNode
2951 NameNode
3325 Jps
```

例如1号服务器，wyd-1-202214xxxx服务器,执行jps

```
wyd@wyd-1-202214xxxx ~]$ jps
2628 Jps
2504 DataNode
```

例如2号服务器，wyd-2-202214xxxx服务器,执行jps

```
wyd@wyd-2-202214xxxx ~]$ jps
2816 Jps
2602 SecondaryNameNode
2490 DataNode
```

1.4 对照jps指令下的进程信息与已部署的HDFS分布情况(不考虑YARN的分布)

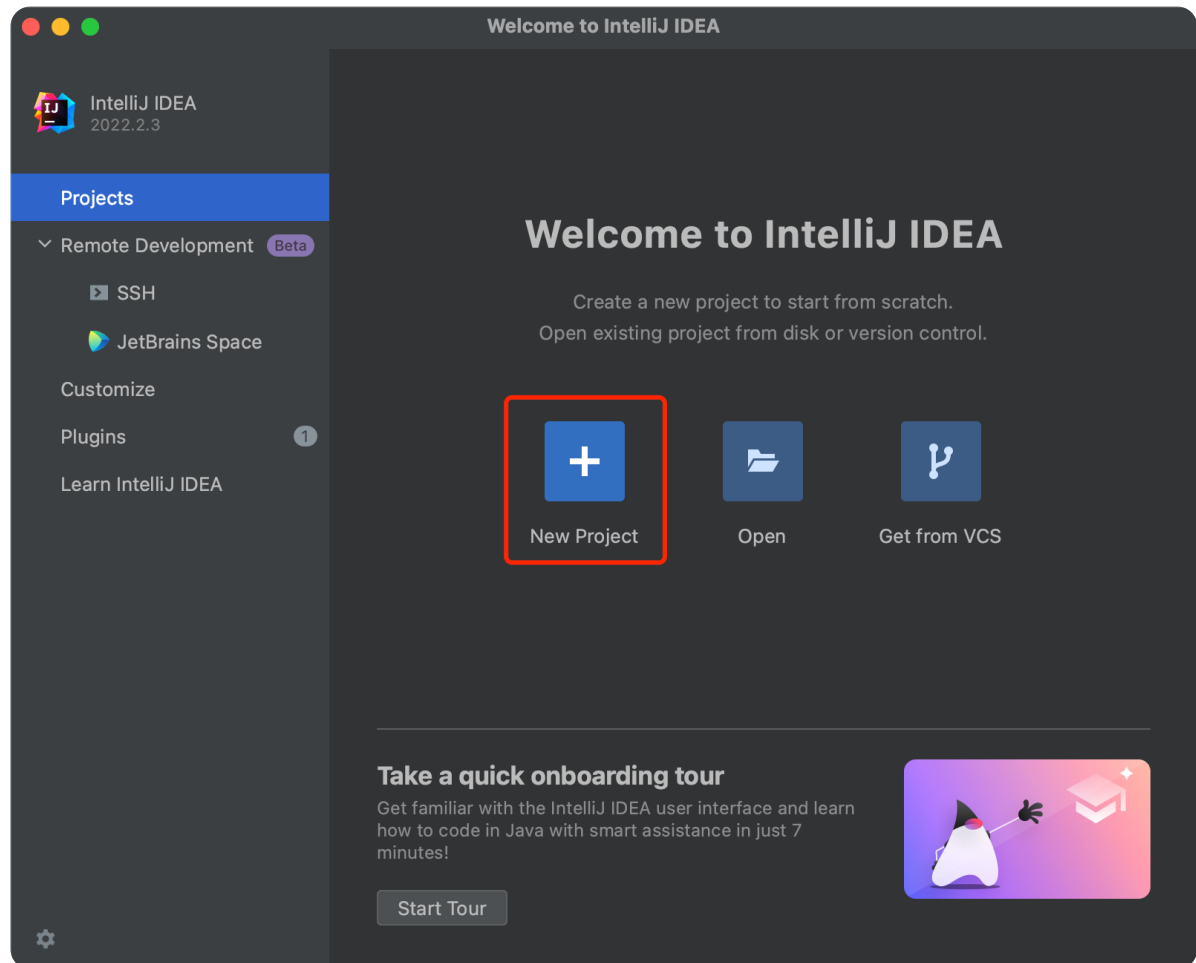
如果未成功启动，请检查理论课实验二的配置，并重新执行以上步骤。

注意：

- Hadoop集群使用结束后,不要直接关机,记得关闭集群!!!
- 如果对配置进行了修改,记得使用rsync命令分发至其他服务器.
- 当对Hadoop配置元数据进行修改时,记得进行format格式化.
- 认真检查各服务器的hosts文件映射是否正确

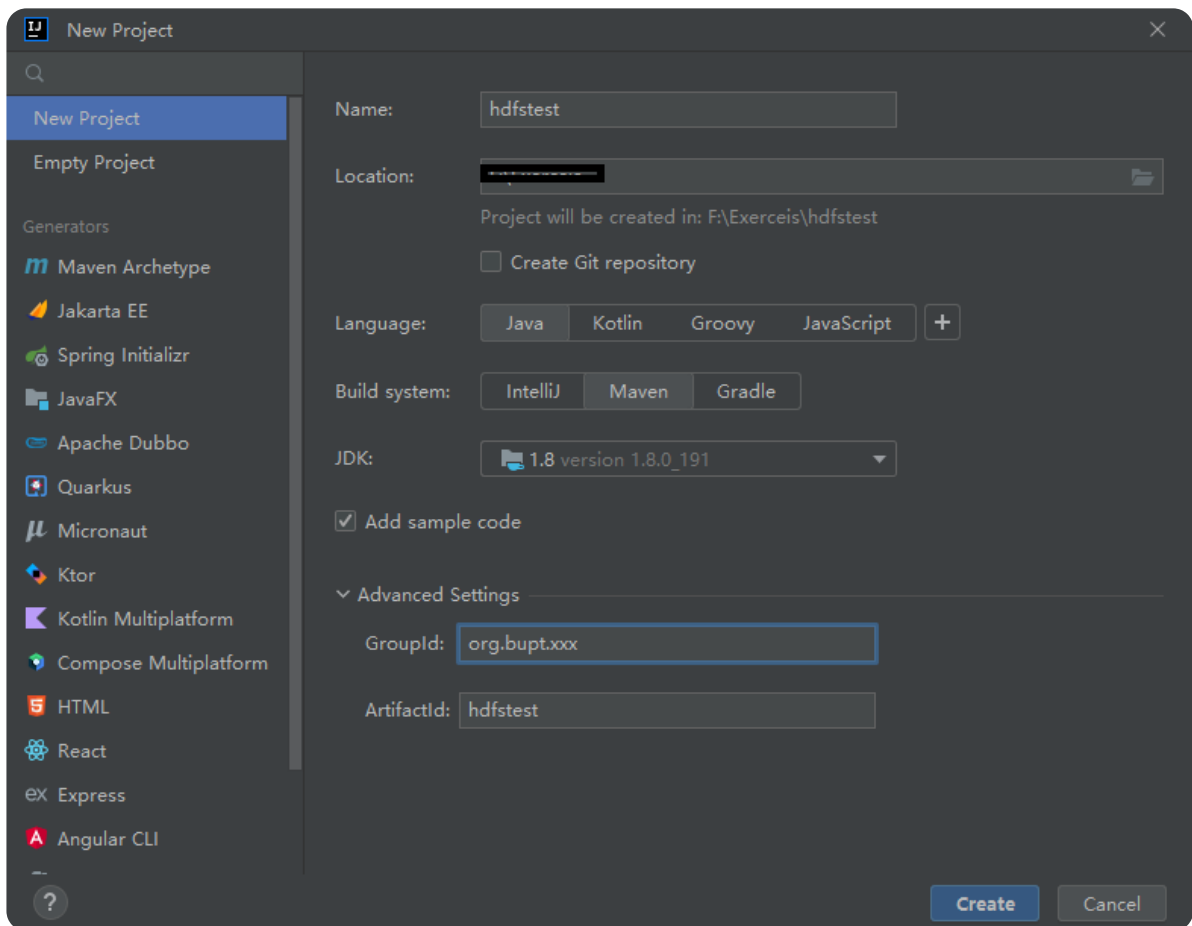
2. 创建项目并测试

2.1 打开编译器，选择Project->New Project



2.2 新建Maven项目

- 点击 New Project
- 命名新项目为 hdfsstest
- 选择语言为 Java
- 项目类型选择 Maven
- 选择JDK版本为 1.8
- 勾选 Add sample code
- 命名 groupId 为 org.bupt.xxx 格式为 名字缩写学号 如 org.bupt.zs2020110000



注：JDK版本直接选择version:1.8 Vendor: Amazon Corretto

2.3 导入所需依赖

- 修改项目根目录下的pom.xml文件，导入hadoop-client依赖。
- 注意：将下面代码直接覆盖粘贴到pom.xml文件，并修改groupId。

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.bupt.xxx</groupId>
    <artifactId>hdfstest</artifactId>
    <version>1.0-SNAPSHOT</version>
    <!-- 配置阿里云仓库 -->
    <repositories>
```

```
<repository>

  <id>aliyun-repos</id>

  <url>https://maven.aliyun.com/repository/public</url>

  <releases>

    <enabled>true</enabled>

  </releases>

  <snapshots>

    <enabled>false</enabled>

  </snapshots>

</repository>

</repositories>

<pluginRepositories>

  <pluginRepository>

    <id>aliyun-repos</id>

    <url>https://maven.aliyun.com/repository/public</url>

    <releases>

      <enabled>true</enabled>

    </releases>

    <snapshots>

      <enabled>false</enabled>

    </snapshots>

  </pluginRepository>

</pluginRepositories>

<properties>

  <maven.compiler.source>8</maven.compiler.source>

  <maven.compiler.target>8</maven.compiler.target>

  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

  <dependency>

    <groupId>junit</groupId>

    <artifactId>junit</artifactId>

    <version>4.13.2</version>

  </dependency>

</dependencies>
```

```

    <dependency>

        <groupId>org.apache.logging.log4j</groupId>

        <artifactId>log4j-core</artifactId>

        <version>2.19.0</version>

    </dependency>

    <!-- hdfs 程序需要的依赖 -->

    <dependency>

        <groupId>org.apache.hadoop</groupId>

        <artifactId>hadoop-client</artifactId>

        <version>2.7.7</version>

    </dependency>

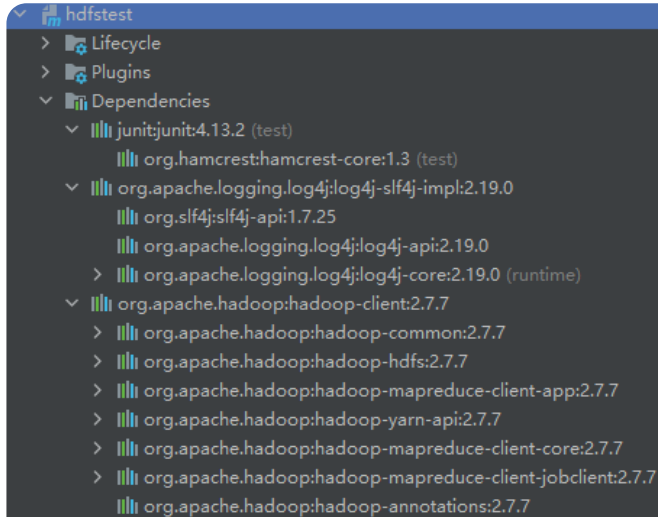
</dependencies>

</project>

```

- 编译器根据配置导入依赖包,等待下载完成pom.xml文件,当该文件没有错误提示时即可。

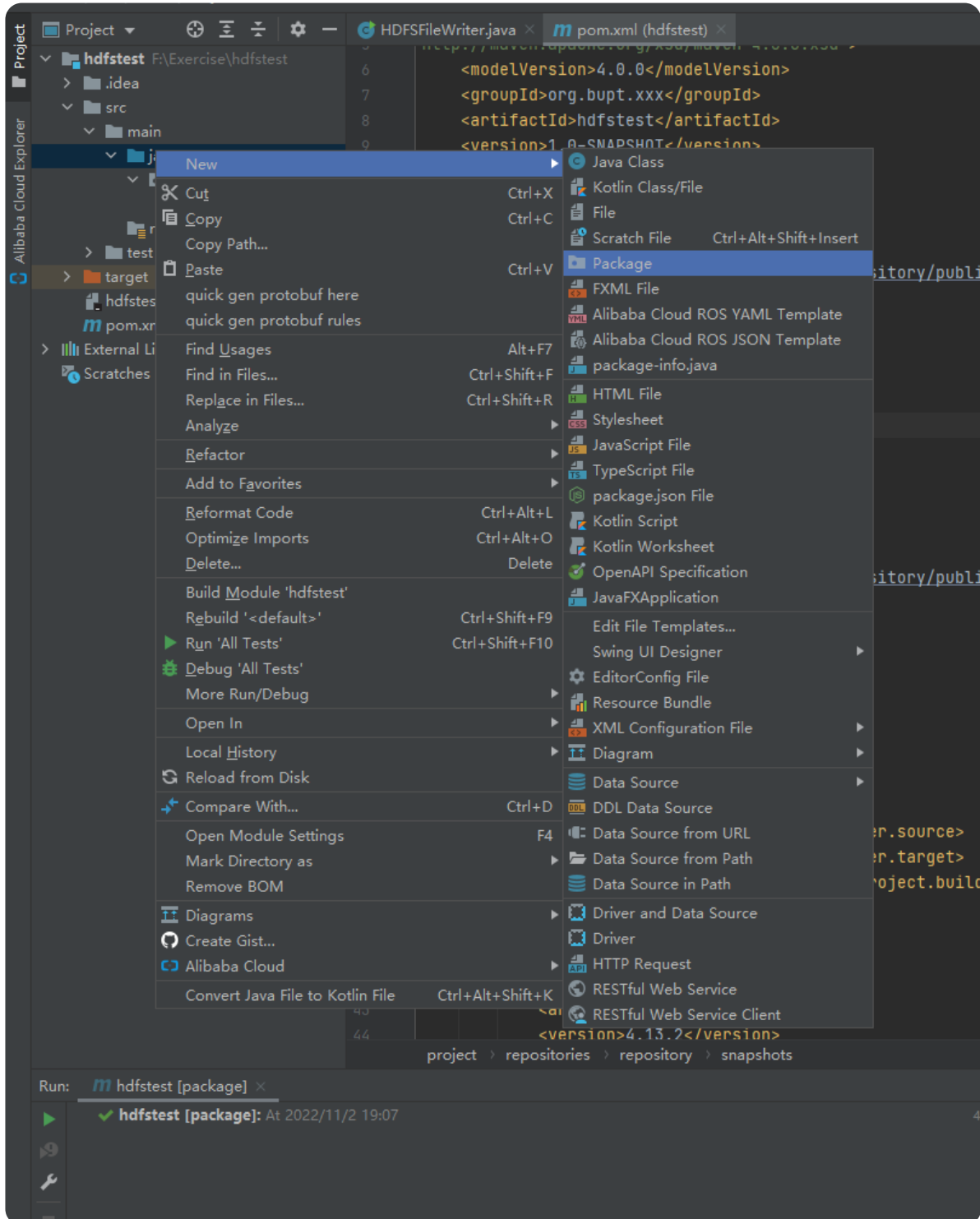
2.4 检查依赖项是否已经导入



3. 基于HDFS的API进行编码开发

3.1 新建org.bupt.xxx.hdfstest包

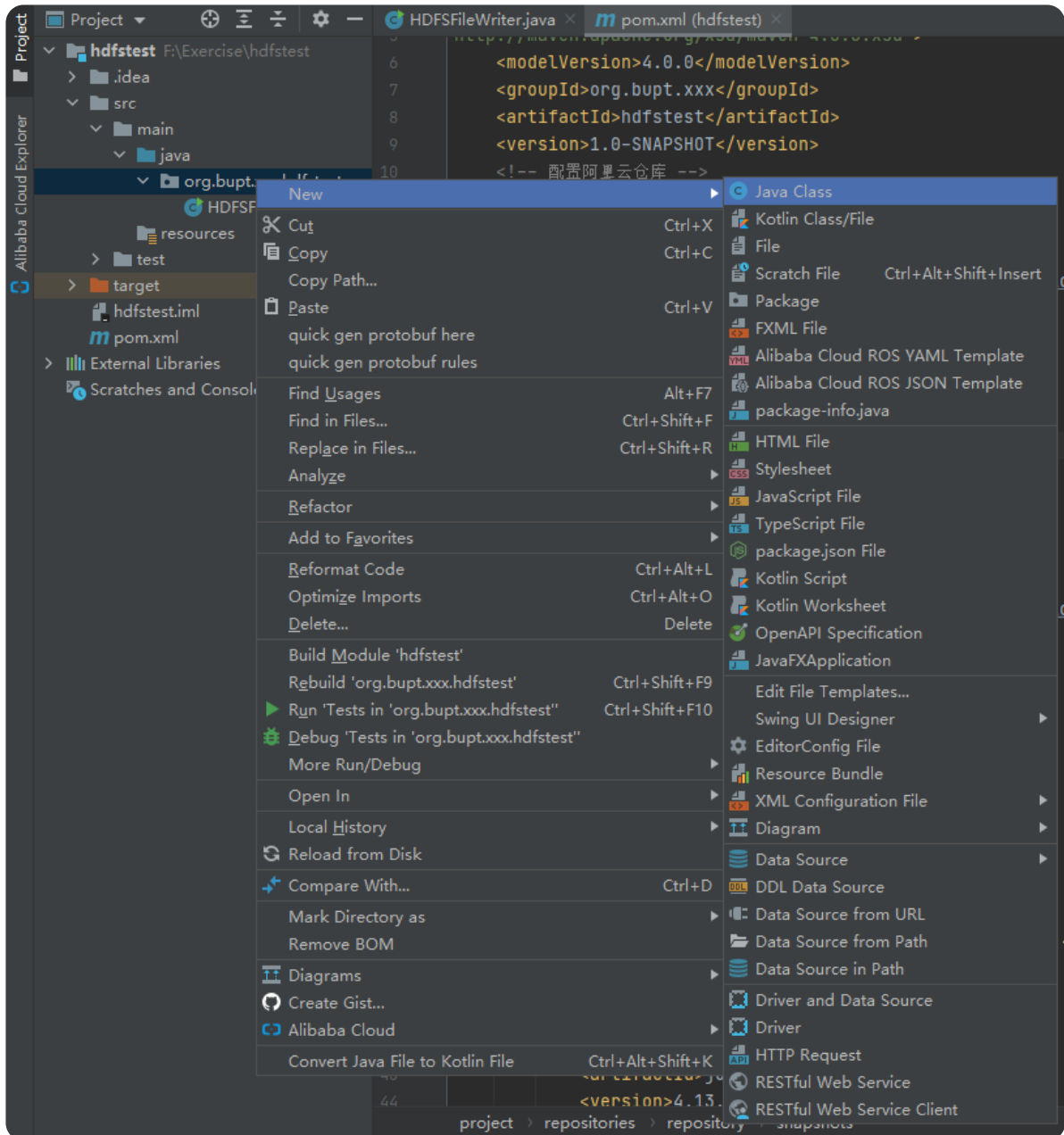
- xxx格式为 名字缩写学号 如 org.bupt.zs2020110000.hdfstest
- 右键单击 java 选择 New -> Package 填入包名



3.2 编写HDFSFileWriter类

- 右键单击新建的包，选择 New -> Java Class 在弹出的窗口中选择Class，对象名称为 HDFSFileWriter

- 在创建的代码文件中编写HDFSFileWriter代码



- 在文件中写入如下代码,注意修改包名(建议复制阅读修改)

```
package org.bupt.xxx.hdfstest;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IOUtils;
```



```
import java.io.*;

import java.net.URI;

public class HDFSFileWriter {

    public static void main(String[] args) throws IOException, InterruptedException {

        String file1 = args[0]; // /root/file1.txt

        String file2 = args[1]; // hdfs://wyd_0_202214xxxx:9000//file2.txt

        // 创建带有缓冲的文件输入流BufferedInputStream(读取大文件时很有优势)
        InputStream in = new BufferedInputStream(new FileInputStream(file1));

        // 创建Configuration对象conf,会加载hadoop的配置文件,从而访问hdfs
        Configuration conf = new Configuration();

        // 根据file2的生成URI对象,再利用此对象和conf对象生成FileSystem对象fs,最后一个参数
        // 为user, root用户填"root"
        FileSystem fs = FileSystem.get(URI.create(file2),conf,"root");

        // 根据要创建的文件路径生成一个Path对象,调用fs的create方法,将此对象作为create方法的
        // 参数,返回一个输出流
        FSDataOutputStream out = fs.create(new Path(file2));

        // 使用流拷贝工具类IOUtils的copyBytes方法,将输入流in拷贝到标准输出流out中,第1,2个参
        // 数为输入流,输出流,
        // 第三个参数为缓冲区大小. 第四个参数为是否关闭数据流
        IOUtils.copyBytes(in,out,4096,true);

        // 手动关闭输入流
        IOUtils.closeStream(in);

        // 手动关闭输出流
        IOUtils.closeStream(out);

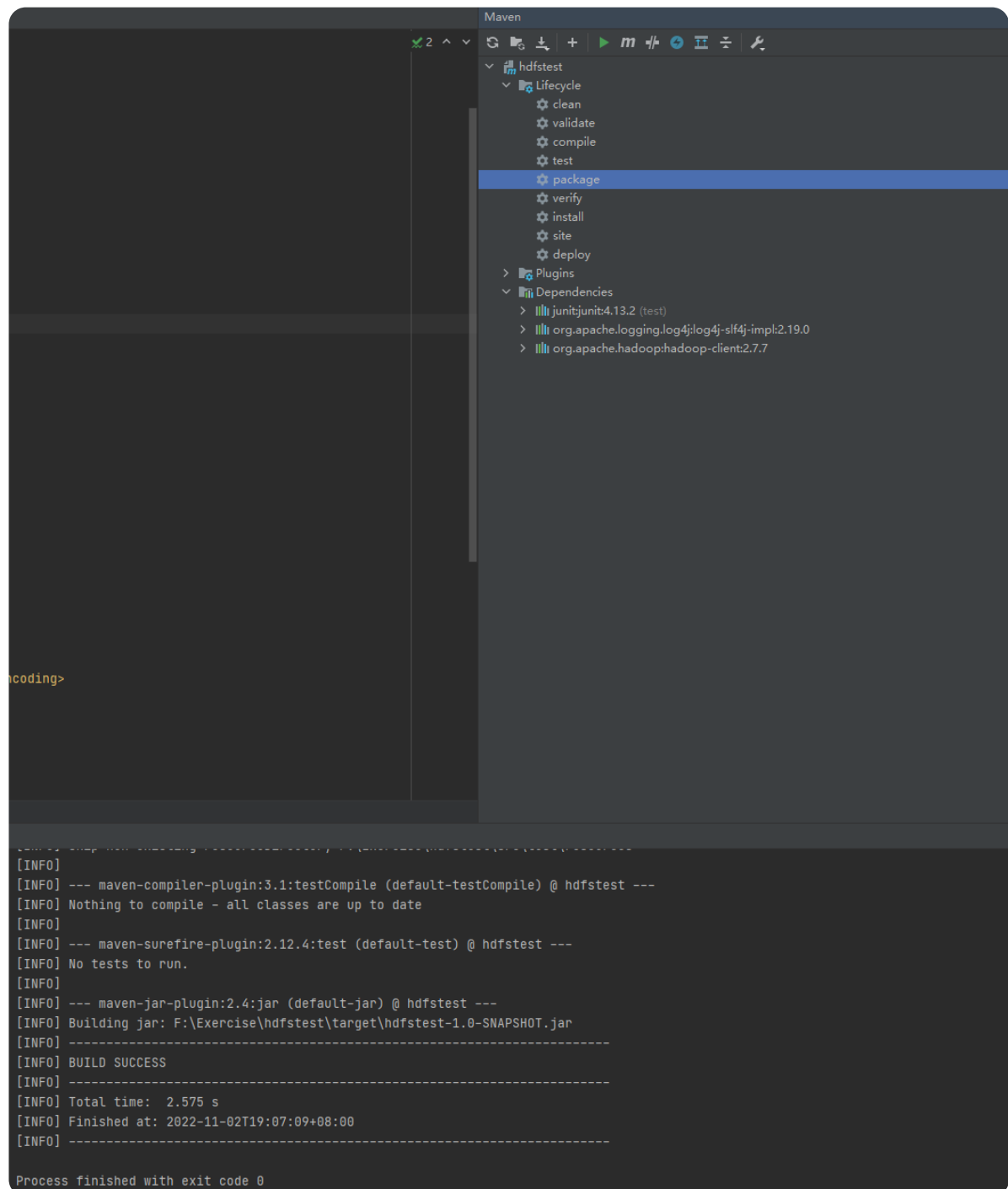
        // 手动关闭FileSystem连接
        fs.close();

    }

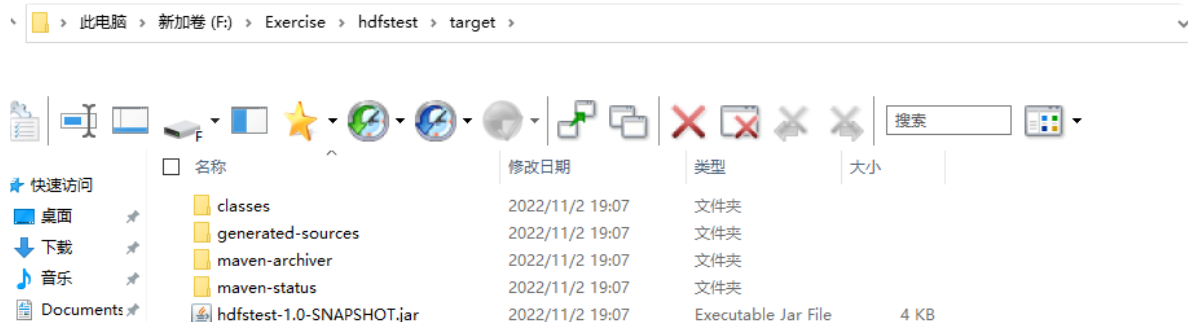
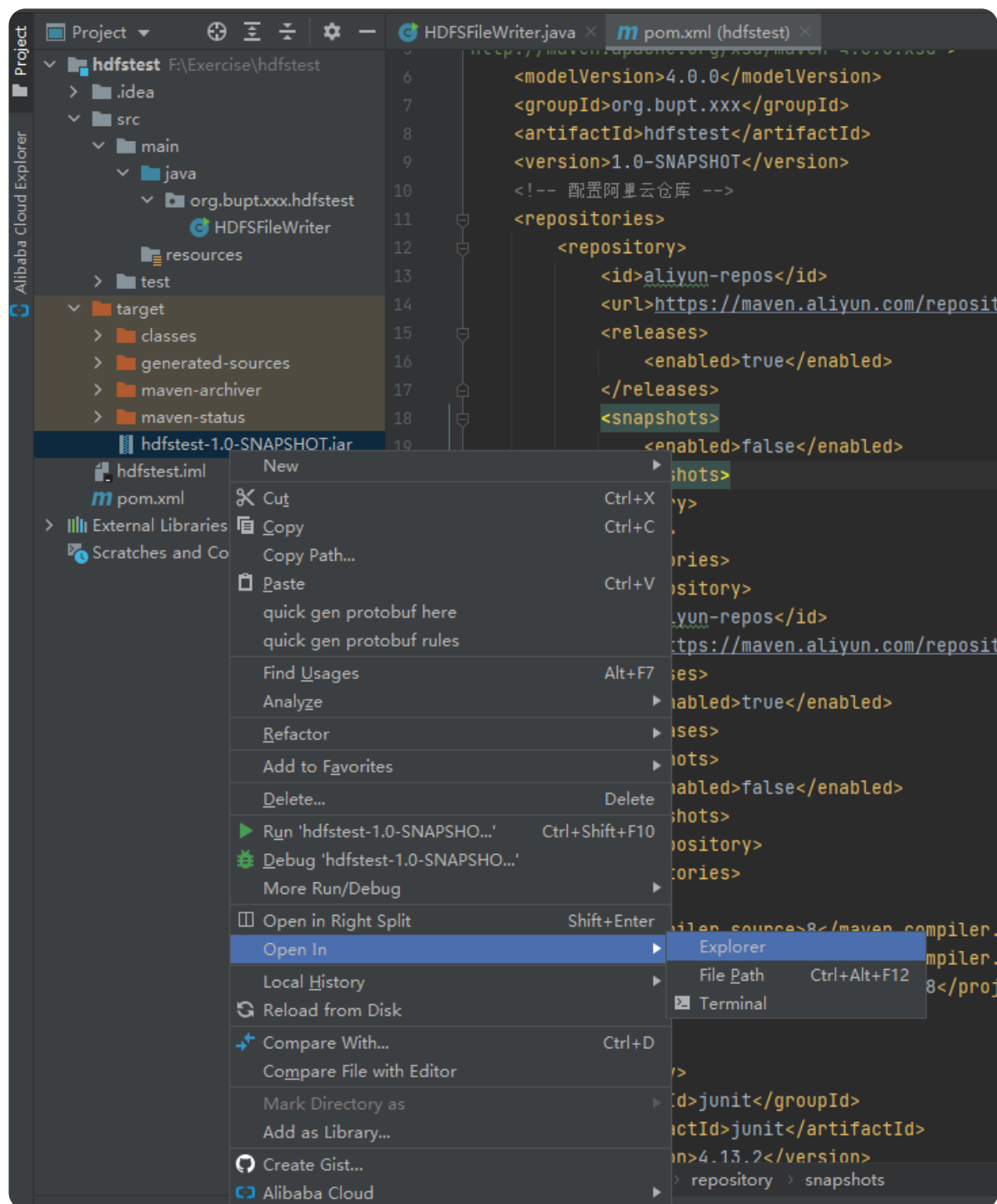
}
```

3.3 打包Maven项目

- 检查代码文件是否齐全(HDFSFileWriter)
- 使用IDEA右侧的Maven窗口提供的工具进行打包
 - 点击Lifecycle -> Package
 - 出现build success后表示打包成功



- 可以看到左侧的项目架构里，项目根目录出现了**target**目录(没有的话请尝试刷新目录)，在**target**目录下出现后缀为**.jar**的可执行文件，可以在文件系统浏览器中打开



3.4 上传Jar包和数据包

利用Xshell ftp文件工具,或者scp/rsync命令, 将所需的文件上传到0号服务器目录下
需要上传的文件为hdfstest.jar(在本地重命名)

4.创建file1.txt文件

0号服务器

4.1 新建file1.txt

```
cd ~  
vim file1.txt
```

4.2 在file1.txt文件中输入自己的IP,姓名以及学号

4.3 验证创建成功

```
[wyd@wyd-0-202214xxxx ~]$ ls  
experiment file1.txt hdfstest.jar
```

5.提交hdfstest任务

5.1 利用Hadoop jar命令运行jar包

在0号服务器 执行如下命令

```
hadoop jar hdfstest.jar org.bupt.xxx.hdfstest.HDFSFileWriter /root/file1.txt  
hdfs://wyd-0-202214xxxx:9000//file2.txt
```

5.2 在HDFS上查看是否创建成功

在0号服务器 执行如下命令

```
hadoop fs -ls /  
hadoop fs -cat /file2.txt
```

```
[wyd@wyd-0-202214xxxx ~]$ hadoop fs -cat /file2.txt  
202214xxxx  
wyd
```

5.3 在WEB端查看HDFS文件系统

114.116.203.150:50070/explorer.html#/

Hadoop Overview Datanodes Snapshot Startup Progress Utilities							
Browse Directory							
/							Go!
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	wyd	supergroup	15 B	2022/11/5 17:22:52	2	128 MB	file2.txt
drwxr-xr-x	wyd	supergroup	0 B	2022/11/2 21:08:54	0	0 B	sanguo

Hadoop, 2018.

5.4 验收前删除HDFS中的file2.txt文件,验收时重新执行步骤5.1,5.2,5.3

```
hadoop fs -rm -f /file2.txt
```

6. 关闭HDFS,关机

6.1 关闭HDFS

在0号服务器

```
stop-dfs.sh
```

6.2 执行jps确认所有的hdfs组件均已关闭

三台服务器

6.3 关机

三台服务器

```
shutdown -h now
```

6.4 确认三台华为云服务器均已关闭

课堂实验结束!

三.扩展实验(选做)

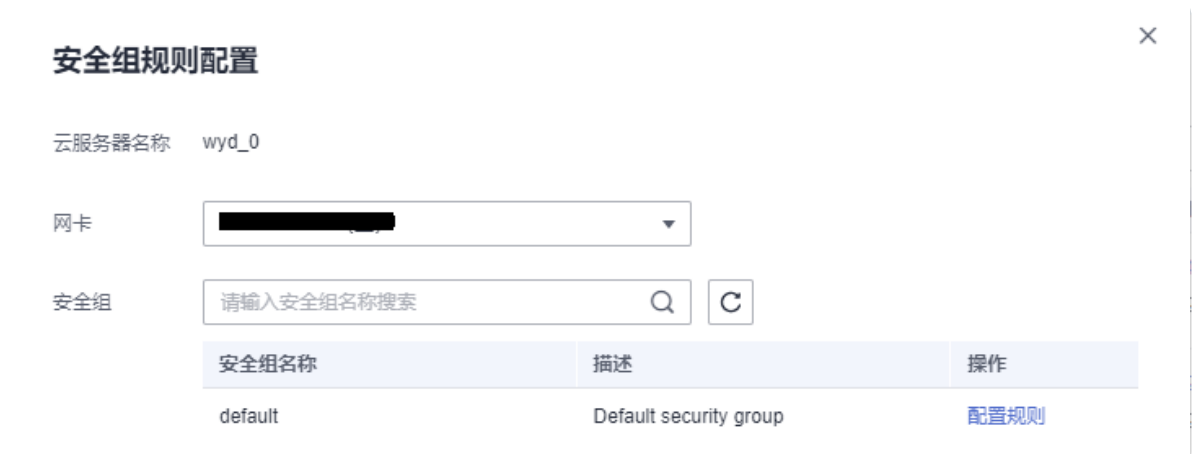
选做实验二选一即可,扩展实验可打包Maven,也可只在本地运行

0. 开放HDFS的华为云安全组端口(便于在本地调试)

0.1 登陆华为云,在自己的服务器界面选择 更多->网络设置->安全组规则配置



0.2 在安全组规则配置 选择配置规则



0.3 选择入方向规则, 添加9000端口,具体配置如图,更多华为云安全组配置策略参见 规则指导



1. 上传选做实验数据

- 通过Xshell ftp 或者 scp/rsync上传 sanguo.tar.gz
- 在服务器解压sanguo.tar.gz

- `tar zxvf sanguo.tar.gz`

- ```
wyd@wyd-0-202214xxxx ~]$ ls
experiment file1.txt hdfstest.jar sanguo sanguo.tar.gz
[wyd@wyd-0-202214xxxx ~]$ cd sanguo
[wyd@wyd-0-202214xxxx sanguo]$ ls
diaochan.txt dongzhuo.txt huatuo.txt lvbu.txt shuguo weiguo wuguo
```

- 将解压后的sanguo文件夹上传至hdfs文件系统

```
wyd@wyd-0-202214xxxx ~]$ hadoop fs -put ~/sanguo /
[wyd@wyd-0-202214xxxx ~]$ hadoop fs -ls /
Found 1 items
drwxr-xr-x - wyd supergroup 0 2022-11-02 21:08 /sanguo
```

## 2. 进行选做实验

若在本地进行调试测试,可参考该本地远程连接NameNode的配置代码.

若直接打包Maven项目,可忽略此部分.

```
//在本地配置hdfs namenode的远程客户端连接

String hdfs_address = "hdfs://你的公网ip:9000"; //hdfs namenode 公网地址

Configuration conf = new Configuration(); //创建Configuration

FileSystem fs= FileSystem.get(new URI(hdfs_address),conf,"root"); //创建hdfs虚拟文件系统连接

System.out.println("Connected successful!");
```

### 参考API

```
org.apache.hadoop.conf.Configuration;

org.apache.hadoop.fs.*

org.apache.hadoop.fs.FileSystem

org.apache.hadoop.fs.listStatus
```

参考文档 [文档](#)

建议参考网络资源

## 选做1.通过Java API实现类似于Linux的文件系统ls命令。

要求： 输入:hdfs的文件路径 例如 /sanguo , /sanguo/shuguo

输出结果为该目录路径下的所有目录和文件的信息 以及 目录和文件分别的数目。

实验结果例图(仅供参考)

```
[org.apache.hadoop.ipc.Client]-IPC Client (530653666) connection to /114.116.203.150:9000 from wyd got value #0
[org.apache.hadoop.ipc.ProtobufRpcEngine]-Call: getListing took 112ms
Type Permission Owner Group Size Last Modified Replication Block Size Name
file rw-r--r-- wyd supergroup 8B 2022-11-02 21:08:53 2 128MB diaochan.txt
file rw-r--r-- wyd supergroup 8B 2022-11-02 21:08:53 2 128MB dongzhuo.txt
file rw-r--r-- wyd supergroup 6B 2022-11-02 21:08:54 2 128MB huatuo.txt
file rw-r--r-- wyd supergroup 6B 2022-11-02 21:08:54 2 128MB lvbu.txt
directory rwxr-xr-x wyd supergroup 0B 2022-11-02 21:08:54 0 0MB shuguo
directory rwxr-xr-x wyd supergroup 0B 2022-11-02 21:08:54 0 0MB weiguo
directory rwxr-xr-x wyd supergroup 0B 2022-11-02 21:08:54 0 0MB wuguo
directory: 3 files:4
```

```
[org.apache.hadoop.ipc.ProtobufRpcEngine]-Call: getListing took 98ms
Type Permission Owner Group Size Last Modified Replication Block Size Name
file rw-r--r-- wyd supergroup 6B 2022-11-02 21:08:54 2 128MB guanyu.txt
file rw-r--r-- wyd supergroup 10B 2022-11-02 21:08:54 2 128MB huangzhong.txt
file rw-r--r-- wyd supergroup 6B 2022-11-02 21:08:54 2 128MB liubei.txt
file rw-r--r-- wyd supergroup 6B 2022-11-02 21:08:54 2 128MB machao.txt
file rw-r--r-- wyd supergroup 7B 2022-11-02 21:08:54 2 128MB mifuren.txt
file rw-r--r-- wyd supergroup 8B 2022-11-02 21:08:54 2 128MB zhangfei.txt
file rw-r--r-- wyd supergroup 7B 2022-11-02 21:08:54 2 128MB zhaoyun.txt
file rw-r--r-- wyd supergroup 10B 2022-11-02 21:08:54 2 128MB zhugeliang.txt
directory: 0 files:8
```

测试时，需至少测试两个目录。

---

## 选做2.通过Java API实现类似于Linux的文件系统的tree命令



```
[wyd@wyd-0-202214xxxx ~]$ cd sanguo
[wyd@wyd-0-202214xxxx sanguo]$ tree
.
├── diaochan.txt
├── dongzhuo.txt
├── huatuo.txt
├── lvbu.txt
├── shuguo
│ ├── guanyu.txt
│ ├── huangzhong.txt
│ ├── liubei.txt
│ ├── machao.txt
│ ├── mifuren.txt
│ ├── zhangfei.txt
│ ├── zhaoyun.txt
│ └── zhugeliang.txt
├── weiguo
│ ├── caocao.txt
│ ├── caopi.txt
│ ├── caozhi.txt
│ ├── simayi.txt
│ └── zhenji.txt
└── wuguo
 ├── daqiao.txt
 ├── sunjian.txt
 ├── sunquan.txt
 ├── xiaoqiao.txt
 └── zhouyu.txt

3 directories, 22 files
```

要求：不需要输出像Linux tree命令结果的样式，能区分出文件的层级结构即可，可不统计目录个数以及文件个数。

实验结果例图(仅供参考)

Connected successful!

```
.
├── diaochan.txt
├── dongzhuo.txt
├── huatuo.txt
├── lvbu.txt
├── shuguo
│ ├── guanyu.txt
│ ├── huangzhong.txt
│ ├── liubei.txt
│ ├── machao.txt
│ ├── mifuren.txt
│ ├── zhangfei.txt
│ ├── zhaoyun.txt
│ └── zhugeliang.txt
├── weiguo
│ ├── caocao.txt
│ ├── caopi.txt
│ ├── caozhi.txt
│ ├── simayi.txt
│ └── zhenji.txt
└── wuguo
 ├── daqiao.txt
 ├── sunjian.txt
 ├── sunquan.txt
 ├── xiaoqiao.txt
 └── zhouyu.txt
```

Closed successful!