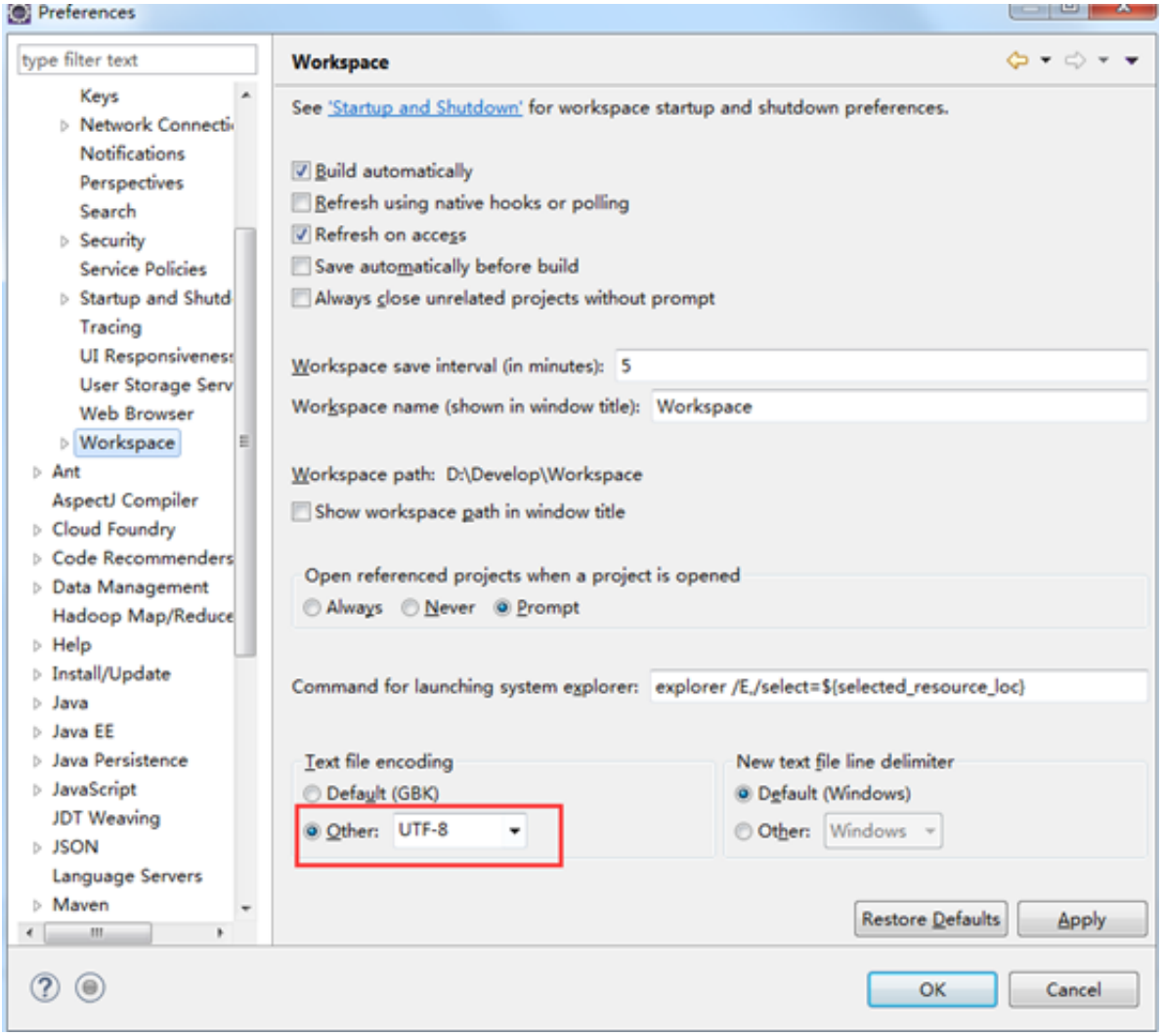
Day03_Eclipse基本设置和hdfs的基本操作

大数据-张军锋 Day03 Eclipse基本设置 hdfs的基本操作 hdfs的基本操作

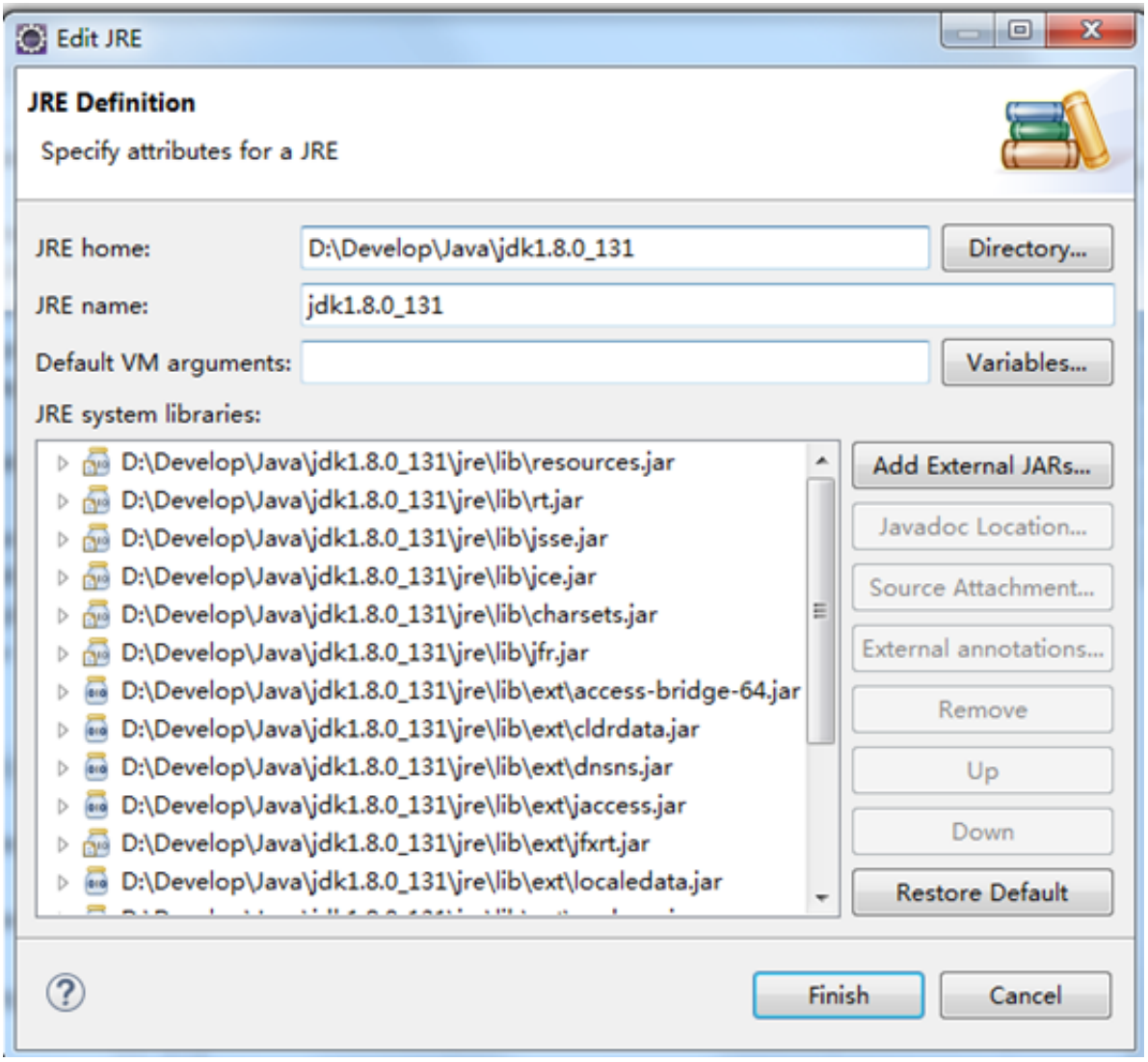
Eclipse基本设置

hadoop开发之前需要对Eclipse进行基础的设置，否则会出现各种各样的麻烦

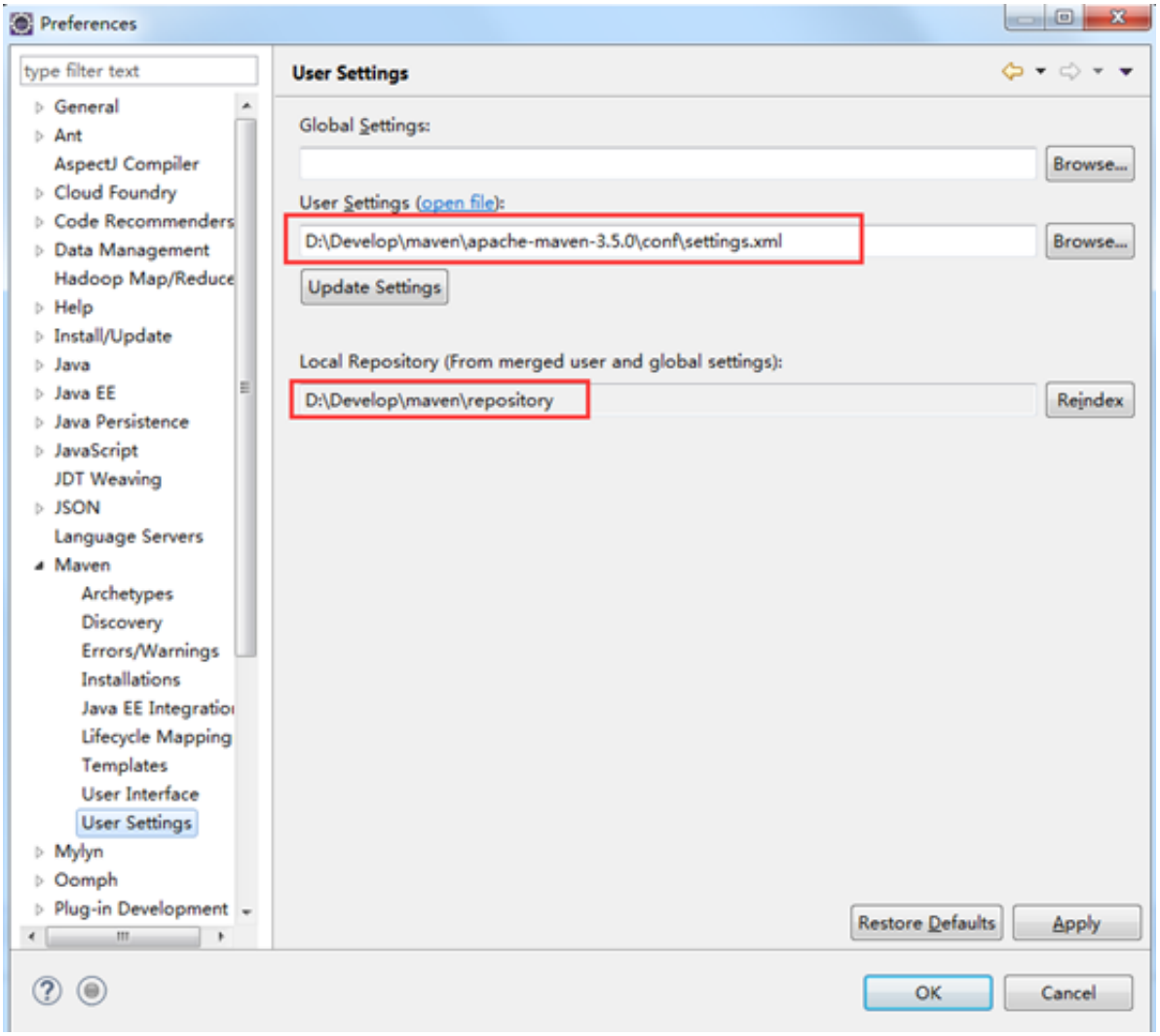
1. 设置编码格式



2. 设置eclipse中jdk



3. eclipse添加额外的maven设置

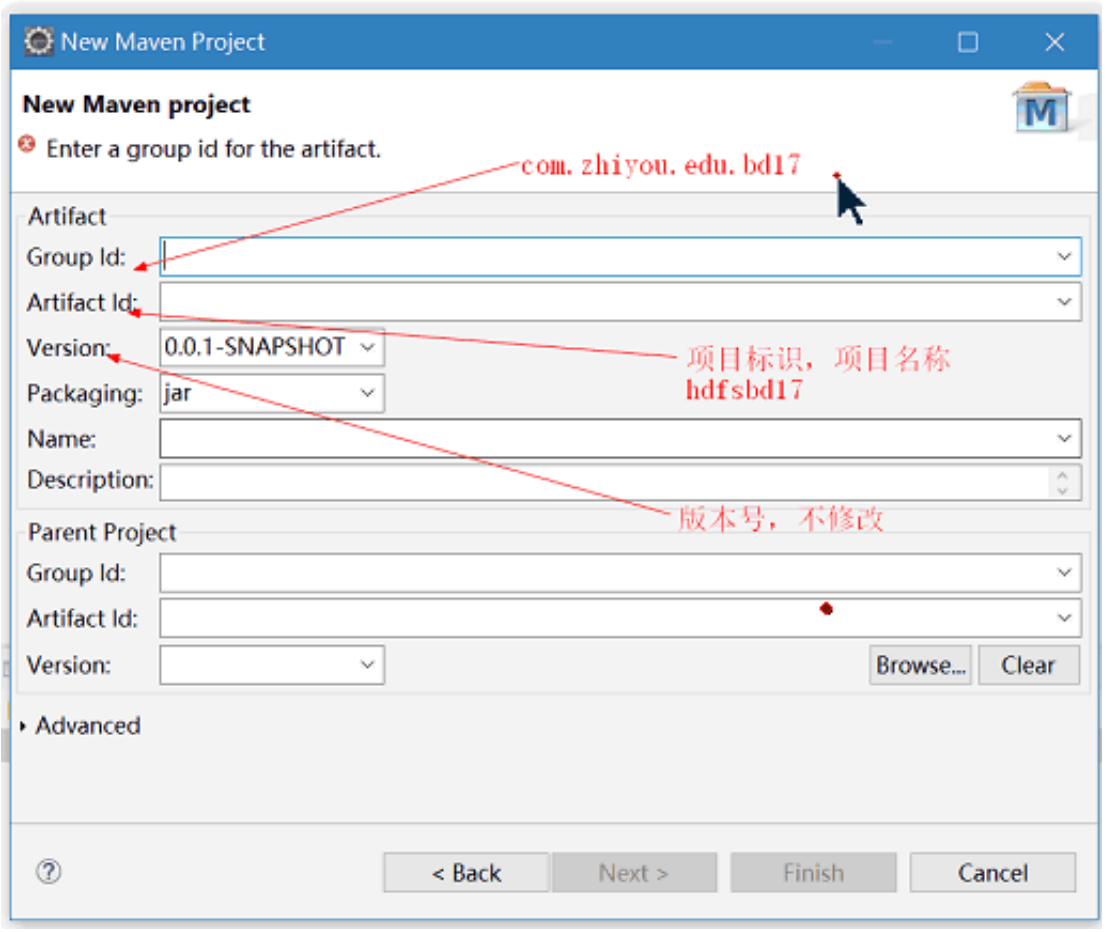


阅读文档的基本步骤

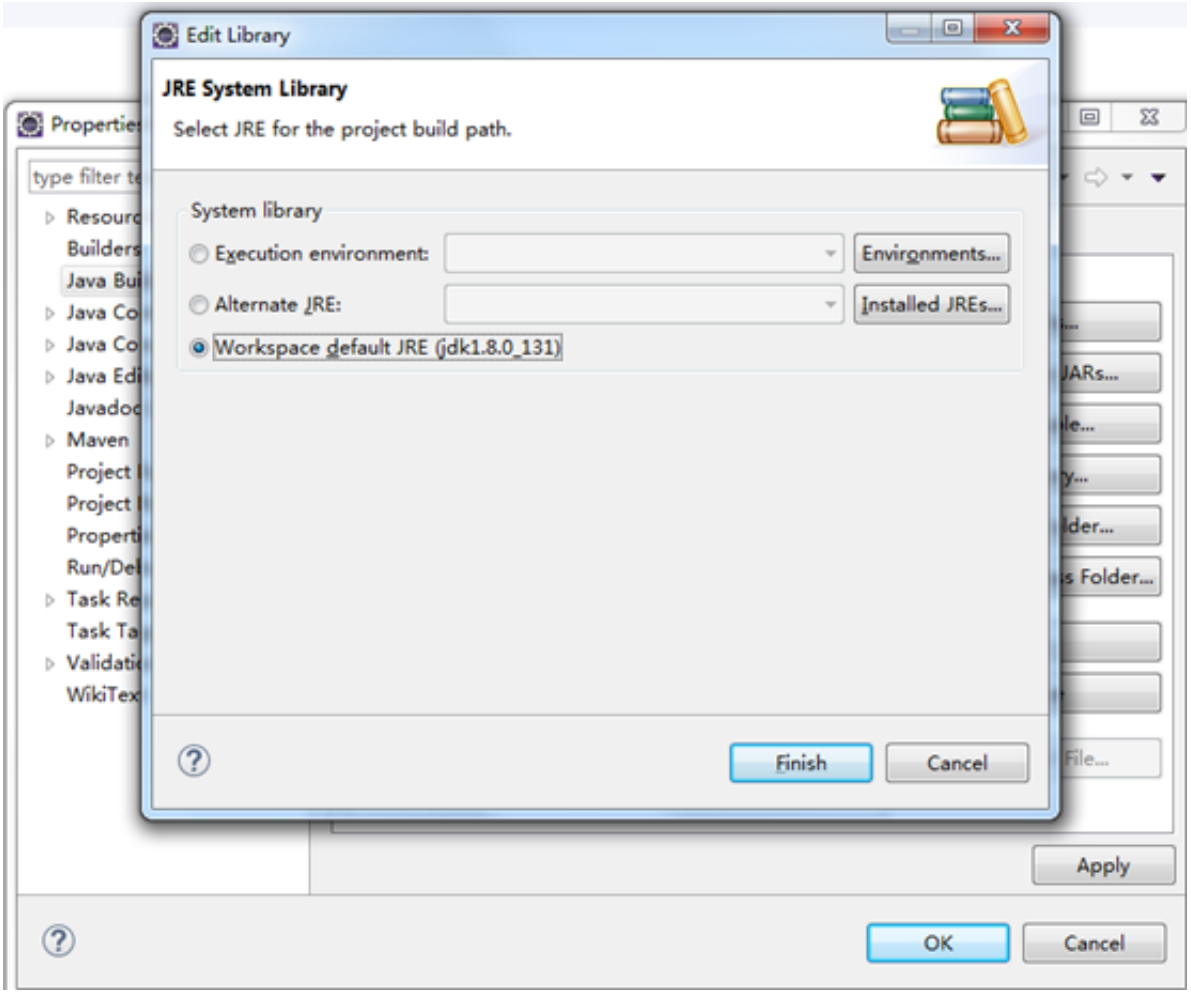
1. read class description
2. read Constructor
3. read static method or Builder and factory
4. read base method，to understand the method of use

创建maven项目注意事项

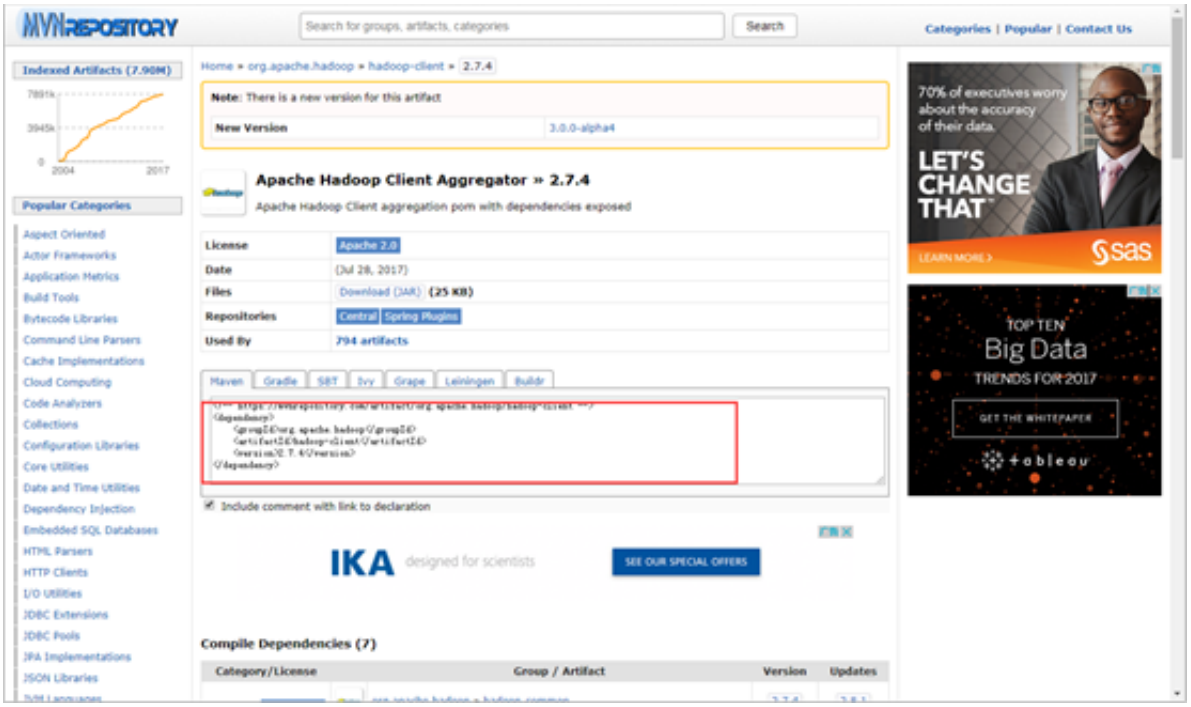
1. 创建项目


New Maven Project
New Maven project
Enter a group id for the artifact.
com.zhiyou.edu.bd17
Artifact
Group Id:
Artifact Id:
Version:
0.0.1-SNAPSHOT
Packaging:
jar
Name:
Description:
项目标识，项目名称
hdfsbd17
版本号，不修改
Parent Project
Group Id:
Artifact Id:
Version:
Browse...
Clear
Advanced
< Back
Next >
Finish
Cancel

2. 修改jdk的版本


Edit Library
JRE System Library
Select JRE for the project build path.
System library
Execution environment:
Environments...
Alternate JRE:
Installed JREs...
Workspace default JRE (jdk1.8.0_131)
Finish
Cancel
Apply
OK
Cancel

3. 添加依赖，添加依赖的网址 https://mvnrepository.com



Java代码操作hdfs

- 创建文件系统的配置文件 `CONF = new Configuration();`
  - 默认加载 `classpath` 下 `core-site.xml` 文件,所以需要将hadoop的配置文件拷贝下来
  - 也可以通过 `CONF` 的 `set` 方法进行设置如: `conf.set("fs.defaultFS", "hdfs://hadoop:9000");`

- 创建文件系统 `hdfs = FileSystem.get(CONF);`
  - `FileSystem` 是抽象类,返回值为具体的子类,如果是 `HDFS` 返回值的类型是 `DistributedFileSystem`

- 创建路径 `Path path = new Path(fileName);`
- 判断路径是否存在 `boolean orExist = hdfs.exists(path);`
- 判断是否是文件夹 `hdfs.isDirectory(path)`
- 判断是否是文件 `hdfs.isFile(path)`
- 创建输入流 `FSDataInputStream input = hdfs.open(path);`
- 创建输出流 `FSDataOutputStream output = hdfs.create(path);`
- 上传文件 `hdfs.copyFromLocalFile(srcPath, desPath);`
- 下载文件 `hdfs.copyToLocalFile(srcPath, desPath);`
- 递归删除文件 `hdfs.delete(path, true);`
- 创建文件夹 `boolean orSuccess = hdfs.mkdirs(path)`
- 获取目录下所有FileSatus `RemoteIterator<LocatedFileStatus> listFiles = hdfs.listFiles(path, true);` 或者 `FileStatus[] status = hdfs.listStatus(path);`
- Path相关

    - 通过Path获取路径 `path.toString()`
    - 通过Path获取文件名 `path.getName()`

- FileStatus相关
    - 通过Path生成FileStatus `hdfs.getFileStatus(path)`
    - 通过FileSatus获取Path `fileStatus.getPath()`

编写hdfsUtils

```
public class HdfsUtils {
    public static final Configuration CONF = new Configuration();
    public static FileSystem hdfs;
    static {
        try {
            hdfs = FileSystem.get(CONF);
        } catch (IOException e) {
            System.out.println("无法连接hdfs,请检查配置...");
            e.printStackTrace();
        }
    }
}
```

在hdfs上创建文件，并写入数据

```
// 创建一个新的文件，将数据写入到hdfs文件中
    public static void createFile(String fileName, String content)
throws Exception {
        Path path = new Path(fileName);
        if (hdfs.exists(path)) {
            System.out.println("文件 " + fileName + " 已存在");
        } else {
            FSDataOutputStream outputStream = hdfs.create(path);
            outputStream.writeUTF(content); // 会添加一些特殊的字符
            // outputStream.write(content.getBytes()); //可以解决特殊
的字符
            outputStream.close();
            outputStream.flush();
        }
    }
```

读取hdfs上已有的文件

```
public static void readFile(String fileName) throws Exception {
        Path path = new Path(fileName);
        if (!hdfs.exists(path) || hdfs.isDirectory(path)) {
            System.out.println("给定路径 " + fileName + "不存在,或者不
是一个");
        } else {
            FSDataInputStream inputStream = hdfs.open(path);
            String content = inputStream.readUTF();
            System.out.println(content);
        }
    }
```

删除hdfs上已经有的文件或文件夹

```
public static void deleteFile(String fileName) throws Exception {
        Path path = new Path(fileName);
        if (!hdfs.exists(path)) {
            System.out.println("文件不存在");
        } else {
            hdfs.delete(path, true);
            System.out.println("删除成功");
        }
    }
```

上传文件

```
public static void upload(Path src,Path dst) throws Exception{
        if(hdfs.exists(dst)){
            System.out.println("文件已经存在");
        }else {
            hdfs.copyFromLocalFile(false,src, dst);
        }
    }
```

下载文件

```
public static void download() throws Exception{
        // 方式一
        /*InputStream in = hdfs.open(new Path("/aa/a.txt"));
        FileOutputStream out = new FileOutputStream("d:/a.txt");
        IOUtils.copyBytes(in, out, 4096,true);*/
        // 方式二
        Path src = new Path("/aa/a.txt");
        Path dst = new Path("d:/aa.txt");
        hdfs.copyToLocalFile(src, dst);
}
```

迭代文件

```
public static void getAllFileStatus(Path path) throws Exception {
        FileStatus[] fileStatus = hdfs.listStatus(path);
        if(hdfs.isDirectory(path)){
            for (FileStatus fs : fileStatus) {
                 path = fs.getPath();
                 getAllFileStatus(path);
                 System.out.println(fs);
            }
        }
    }
```

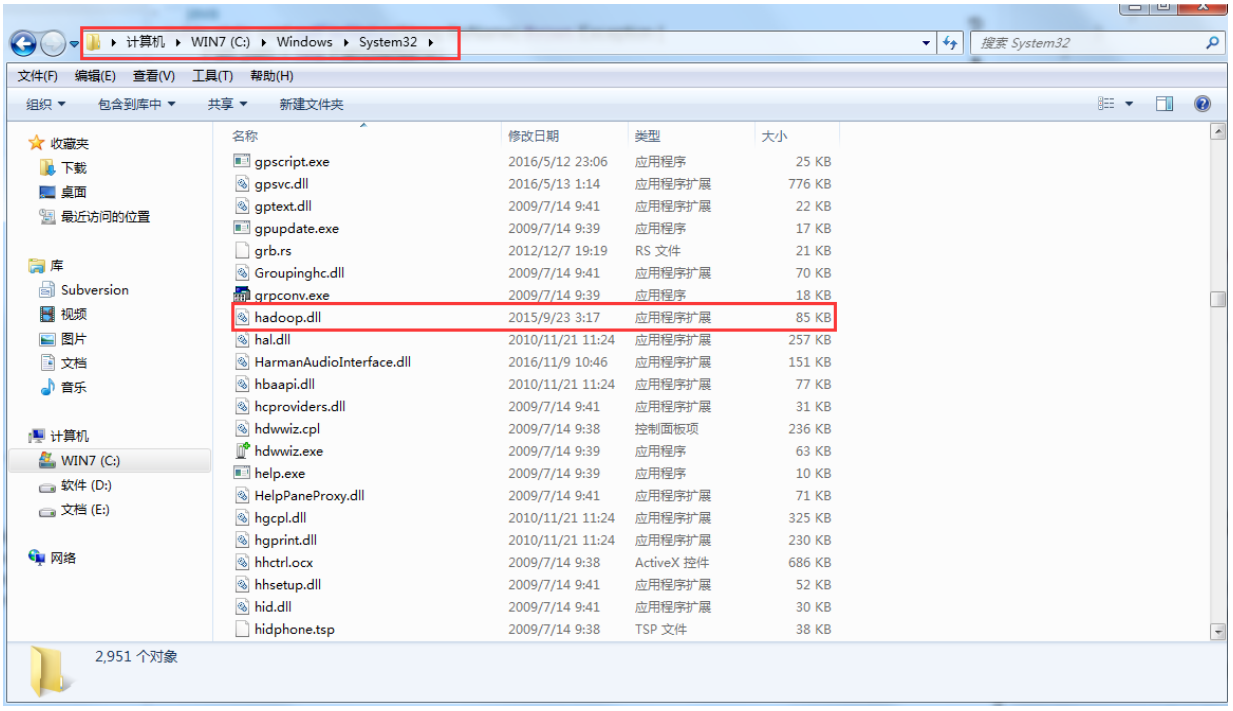
查看文件状态

```
public static void getFileStatus(String fileName) throws Exception
{
        Path path = new Path(fileName);
        FileStatus[] fileStatus = hdfs.listStatus(path);
        for (FileStatus fs : fileStatus) {
            System.out.println(fs);
        }
    }
```

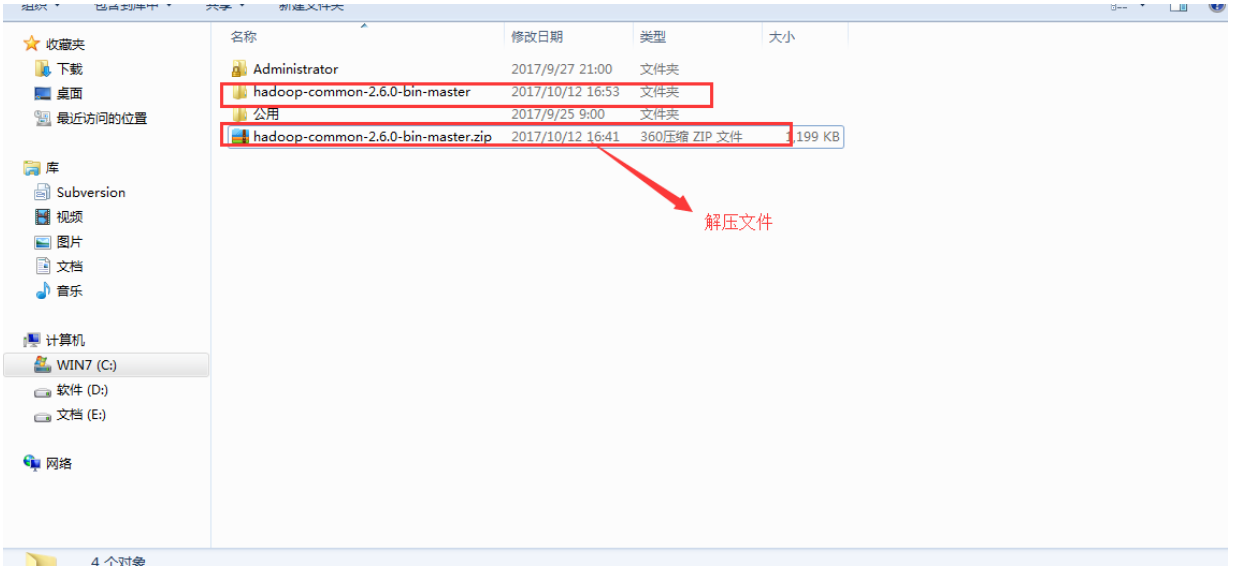
安装hadoop环境

> 对于hadoop的开发，最好是直接在原生的linux上进行开发，但是考虑到linux操作不方便，在windows上进行开发，但是存在一些问题，必须营造一个hadoop的开发环境，对于linux开发有一定经验的大牛们，编译了linux的环境放到网络上了，我们只需要站在巨人的肩膀上就可以了，在百度上输入 hadoop common bin就能找到对应的压缩包，下面是安装的过程

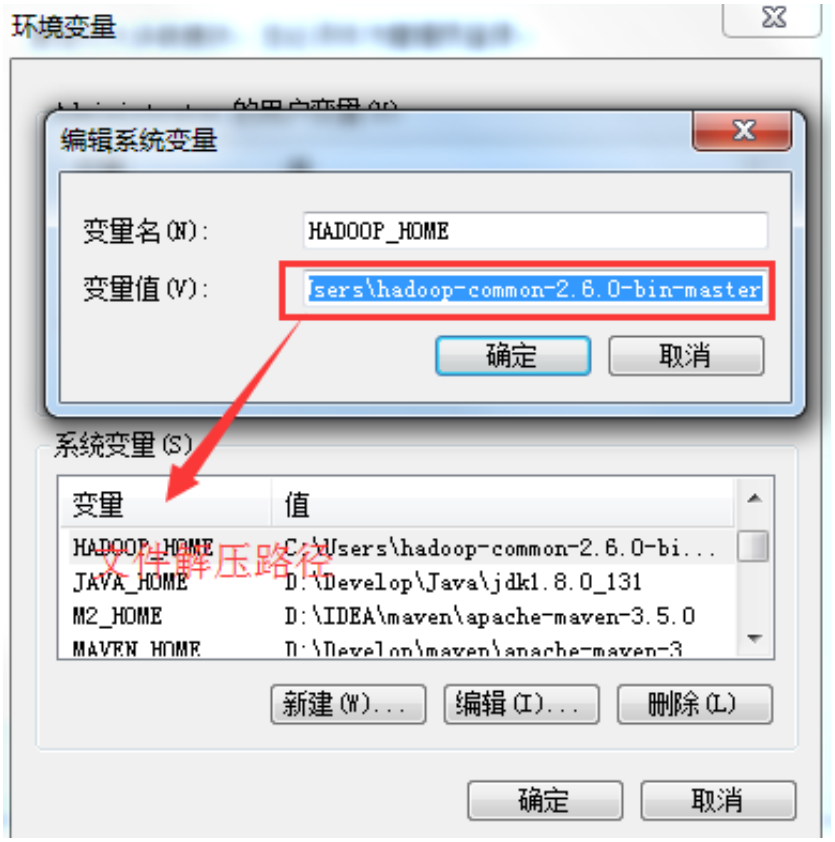
1. 将hadoop.dll文件复制到 `C:\Windows\System32` 文件夹下面



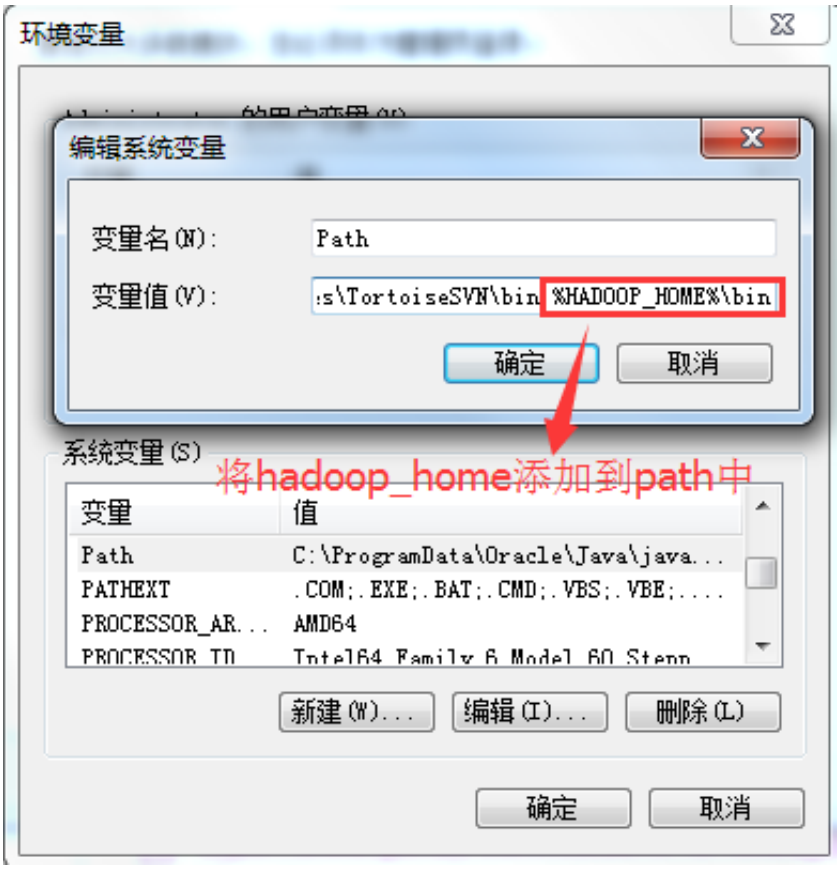
2. 在任意目录下解压下载的 `hadoop-common-2.6.0-bin-master` 文件



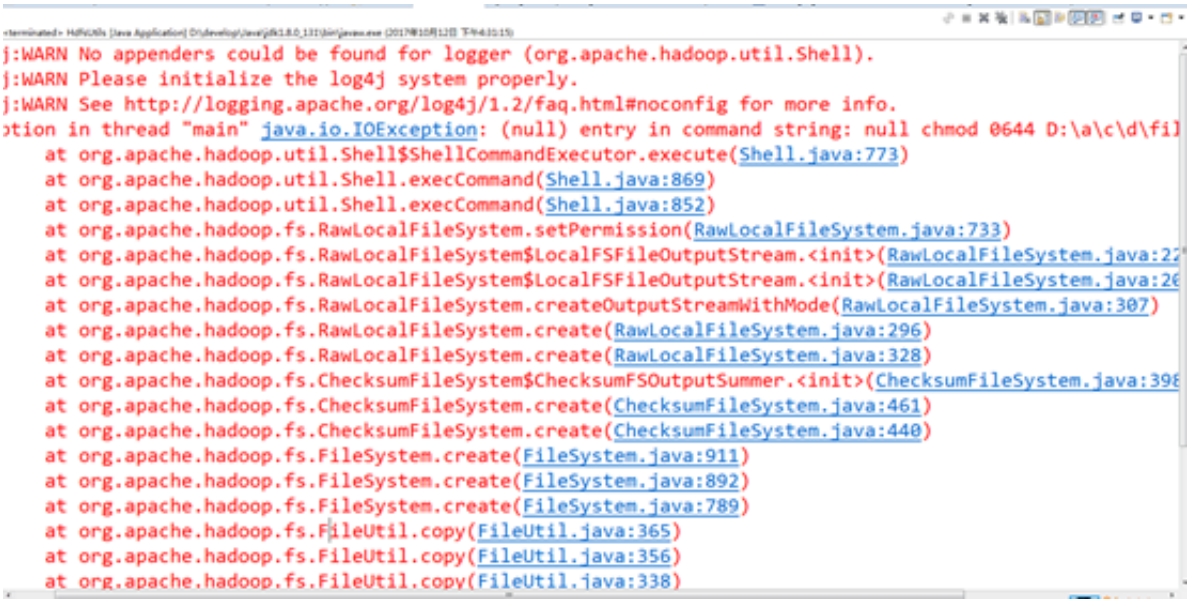
3. 新建系统变量



5. 添加到path



注意如果不安装hadoop环境执行hadoop程序的时候可能会出现如下图所示的错误信息。


j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
j:WARN Please initialize the log4j system properly.
j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
ɔtion in thread "main" java.io.IOException: (null) entry in command string: null chmod 0644 D:\a\c\d\fil
at org.apache.hadoop.util.Shell$ShellCommandExecutor.execute(Shell.java:773)
at org.apache.hadoop.util.Shell.execCommand(Shell.java:869)
at org.apache.hadoop.util.Shell.execCommand(Shell.java:852)
at org.apache.hadoop.fs.RawLocalFileSystem.setPermission(RawLocalFileSystem.java:733)
at org.apache.hadoop.fs.RawLocalFileSystem$LocalFSFileOutputStream.<init>(RawLocalFileSystem.java:22
at org.apache.hadoop.fs.RawLocalFileSystem$LocalFSFileOutputStream.<init>(RawLocalFileSystem.java:26
at org.apache.hadoop.fs.RawLocalFileSystem.createOutputStreamWithMode(RawLocalFileSystem.java:307)
at org.apache.hadoop.fs.RawLocalFileSystem.create(RawLocalFileSystem.java:296)
at org.apache.hadoop.fs.RawLocalFileSystem.create(RawLocalFileSystem.java:328)
at org.apache.hadoop.fs.ChecksumFileSystem$ChecksumFSOutputSummer.<init>(ChecksumFileSystem.java:398
at org.apache.hadoop.fs.ChecksumFileSystem.create(ChecksumFileSystem.java:461)
at org.apache.hadoop.fs.ChecksumFileSystem.create(ChecksumFileSystem.java:440)
at org.apache.hadoop.fs.FileSystem.create(FileSystem.java:911)
at org.apache.hadoop.fs.FileSystem.create(FileSystem.java:892)
at org.apache.hadoop.fs.FileSystem.create(FileSystem.java:789)
at org.apache.hadoop.fs.FileUtil.copy(FileUtil.java:365)
at org.apache.hadoop.fs.FileUtil.copy(FileUtil.java:356)
at org.apache.hadoop.fs.FileUtil.copy(FileUtil.java:338)