问题：本人在做大学服务外包竞赛的项目，遇到“通过java调用python代码”的问题。通过资料查找，发现网上多是推荐用Process pr=Runtime.getRuntime().exec(args1);这一方法（具体内容请自行搜索）。<https://www.cnblogs.com/nuccch/p/8435693.html>（还是帮你们搜吧）

一开始，本人也选择了该方法，因为确实简单，还能使用第三方库。但是本人调用中遇到了问题。java上找到了指定文件，也顺利传入参数，但由于python和java不在同一工程中，python内的一些文件指定不是用的绝对路径，导致python运行失败。而这问题在服务器内用cmd运行python时不会出现，即cmd能成功执行python。

重点来了，这一FileNotfound的错误在用Runtime方法直接执行python文件时不会出现错误提示，且在python终断后，继续执行Java代码。这给我一种Runtime执行有时限的错觉（就让我一段时间内不知道是python里指定文件没用绝对路径的问题），网上也有网友这样质疑：“java Runtime调用python无法执行完？”

<https://zhidao.baidu.com/question/1993832437599756187.html?hideOtherAnswer=true&newAnswer=1>

后来，本人才用于cmd结合的方法和错误处理，如下：（这一方法可以在Tomcat，即服务器中看到python文件执行的错误提示。个人强推！）就java先自己打开cmd，再执行pathon，遇python错误，报错。

import java.io.InputStream;

import java.io.OutputStream;

import java.io.PrintWriter;

public class CMD命令 {

public static void main(String[] args) {

docmd();

}

static void docmd() {

String[] command = { "cmd", };

Process p = null;

try {

p = Runtime.getRuntime().exec(command);

new Thread(new SyncPipe(p.getErrorStream(), System.err)).start();

new Thread(new SyncPipe(p.getInputStream(), System.out)).start();

PrintWriter stdin = new PrintWriter(p.getOutputStream());

/\*\* 以下可以输入自己想输入的cmd命令 \*/

stdin.println("python "+"要执行的python文件的绝对路径 "+传入python的参数);//此处自行填写，切记有空格，跟cmd的执行语句一致。

stdin.close();

} catch (Exception e) {

throw new RuntimeException("编译出现错误：" + e.getMessage());

}

}

}

class SyncPipe implements Runnable {

private final OutputStream ostrm\_;

private final InputStream istrm\_;

public SyncPipe(InputStream istrm, OutputStream ostrm) {

istrm\_ = istrm;

ostrm\_ = ostrm;

}

public void run() {

try {

final byte[] buffer = new byte[1024];

for (int length = 0; (length = istrm\_.read(buffer)) != -1;) {

ostrm\_.write(buffer, 0, length);

}

} catch (Exception e) {

throw new RuntimeException("处理命令出现错误：" + e.getMessage());

}

}

}

这个代码看起来比较长，但是绝对比单纯Runtime有效，有错误处理，代码拷到自己的工具类里，加入路径和参数就能直接用，也没比单用Runtime麻烦多少，强推！

————————————————

版权声明：本文为CSDN博主「MrBang\_」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/MrBang\_/article/details/88054391