1、引入jar包

```
<dependency>
    <groupId>com.jcraft</groupId>
    <artifactId>jsch</artifactId>
    <version>0.1.54</version>
</dependency>
```

2、实际例子

```
package com.bluemoon.executor.core.executor;

import com.bluemoon.executor.core.log.XxlJobLogger;
import com.jcraft.jsch.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.*;
import java.nio.charset.Charset;

public class JSchExecutor {
    private static Logger log = LoggerFactory.getLogger(JSchExecutor.class);

    private String charset = "UTF-8"; // 设置编码格式
    private String user; // 用户名
    private String passwd; // 登录密码
    private String host; // 主机IP
    private int port = 22; //默认端口
    private JSch jsch;
    private Session session;

    private ChannelSftp sftp;

    /**
     *
     * @param user 用户名
     * @param passwd 密码
     * @param host 主机IP
     */
    public JSchExecutor(String user, String passwd, String host ) {
        this.user = user;
        this.passwd = passwd;
        this.host = host;
    }

    /**
     *
```

```
     * @param user 用户名
     * @param passwd 密码
     * @param host 主机IP
     */
    public JSchExecutor(String user, String passwd, String host , int port ) {
        this.user = user;
        this.passwd = passwd;
        this.host = host;
        this.port = port;
    }

    /**
     * 连接到指定的IP
     *
     * @throws JSchException
     */
    public void connect() throws JSchException {
        jsch = new JSch();
        session = jsch.getSession(user, host, port);
        session.setPassword(passwd);
        java.util.Properties config = new java.util.Properties();
        config.put("StrictHostKeyChecking", "no");
        session.setConfig(config);
        session.connect();
        Channel channel = session.openChannel("sftp");
        channel.connect();
        sftp = (ChannelSftp) channel;
        log.info("连接到SFTP成功。host: " + host);
    }
    /**
     * 关闭连接
     */
    public void disconnect(){
        if (sftp != null && sftp.isConnected()) {
            sftp.disconnect();
        }
        if(session != null && session.isConnected()){
            session.disconnect();
        }
    }
    /**
     * 执行一条命令
     */
    public int execCmd(String command) throws Exception{
        XxlJobLogger.log( "开始执行命令:" + command);
        int returnCode  = -1;
```

```
        BufferedReader reader = null;
        Channel channel = null;

        channel = session.openChannel("exec");
        ((ChannelExec) channel).setCommand(command);
        channel.setInputStream(null);
        ((ChannelExec) channel).setErrStream(System.err);
        InputStream in = channel.getInputStream();
        reader = new BufferedReader(new InputStreamReader(in));//中文乱码貌似这里不能控制，看连接的服务器的

        channel.connect();
        System.out.println("The remote command is: " + command);
        String buf ;
        while ((buf = reader.readLine()) != null) {
            XxlJobLogger.log(buf);
        }
        reader.close();
        // Get the return code only after the channel is closed.
        if (channel.isClosed()) {
            returnCode = channel.getExitStatus();
        }
        XxlJobLogger.log( "Exit-status:" + returnCode );

       /* StringBuffer buf = new StringBuffer( 1024 );
        byte[] tmp = new byte[ 1024 ];
        while ( true ) {
            while ( in.available() > 0 ) {
                int i = in.read( tmp, 0, 1024 );
                if ( i < 0 ) break;
                buf.append( new String( tmp, 0, i ) );
            }
            if (  channel.isClosed() ) {
                res = channel.getExitStatus();
                XxlJobLogger.log( "Exit-status:" + res );
                System.out.println( "Exit-status:" + res );
                break;
            }
            TimeUnit.MILLISECONDS.sleep(100);
        }
        XxlJobLogger.log( buf.toString() );*/

        channel.disconnect();
        return returnCode;
    }

    /**
```

```
     * 执行相关的命令
     */
    public void execCmd() {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        String command = "";
        BufferedReader reader = null;
        Channel channel = null;

        try {
            while ((command = br.readLine()) != null) {
                channel = session.openChannel("exec");
                ((ChannelExec) channel).setCommand(command);
                channel.setInputStream(null);
                ((ChannelExec) channel).setErrStream(System.err);

                channel.connect();
                InputStream in = channel.getInputStream();
                reader = new BufferedReader(new InputStreamReader(in,
                        Charset.forName(charset)));
                String buf = null;
                while ((buf = reader.readLine()) != null) {
                    System.out.println(buf);
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        } catch (JSchException e) {
            e.printStackTrace();
        } finally {
            try {
                reader.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
            channel.disconnect();
        }
    }

    /**
     * 上传文件
     */
    public void uploadFile(String local,String remote) throws Exception {
        File file = new File(local);
        if (file.isDirectory()) {
            throw new RuntimeException(local + "  is not a file");
```

```
        }

        InputStream inputStream = null;
        try {
            String rpath = remote.substring(0,remote.lastIndexOf("/")+1);
            if (!isDirExist(rpath)){
                createDir(rpath);
            }
            inputStream = new FileInputStream(file);
            sftp.setInputStream(inputStream);
            sftp.put(inputStream, remote);
        } catch (Exception e) {
            throw e;
        }finally{
            if(inputStream != null){
                inputStream.close();
            }
        }
    }
    /**
     * 下载文件
     */
    public void downloadFile(String remote,String local) throws Exception{
        OutputStream outputStream = null;
        try {
            sftp.connect(5000);
            outputStream = new FileOutputStream(new File(local));
            sftp.get(remote, outputStream);
            outputStream.flush();
        } catch (Exception e) {
            throw e;
        }finally{
            if(outputStream != null){
                outputStream.close();
            }
        }
    }

    /**
     * 移动到相应的目录下
     * @param pathName 要移动的目录
     * @return
     */
    public boolean changeDir(String pathName){
        if(pathName == null || pathName.trim().equals("")){
            log.debug("invalid pathName");
```

```
            return false;
        }
        try {
            sftp.cd(pathName.replaceAll("\\\\", "/"));
            log.debug("directory successfully changed,current dir=" + sftp.pwd());
            return true;
        } catch (SftpException e) {
            log.error("failed to change directory",e);
            return false;
        }
    }

    /**
     * 创建一个文件目录，mkdir每次只能创建一个文件目录
     * 或者可以使用命令mkdir -p 来创建多个文件目录
     */
    public void createDir(String createpath) {
        try {
            if (isDirExist(createpath)) {
                sftp.cd(createpath);
                return;
            }
            String pathArry[] = createpath.split("/");
            StringBuffer filePath = new StringBuffer("/");
            for (String path : pathArry) {
                if (path.equals("")) {
                    continue;
                }
                filePath.append(path + "/");
                if (isDirExist(filePath.toString())) {
                    sftp.cd(filePath.toString());
                } else {
                    // 建立目录
                    sftp.mkdir(filePath.toString());
                    // 进入并设置为当前目录
                    sftp.cd(filePath.toString());
                }
            }
            sftp.cd(createpath);
        } catch (SftpException e) {
            throw new RuntimeException("创建路径错误：" + createpath);
        }
    }



    /**
```

```
     * 判断目录是否存在
     * @param directory
     * @return
     */
    public boolean isDirExist(String directory)
    {
        boolean isDirExistFlag = false;
        try
        {
            SftpATTRS sftpATTRS = sftp.lstat(directory);
            isDirExistFlag = true;
            return sftpATTRS.isDir();
        }
        catch (Exception e)
        {
            if (e.getMessage().toLowerCase().equals("no such file"))
            {
                isDirExistFlag = false;
            }
        }
        return isDirExistFlag;
    }

}
```

3、用法

```
public static void main(String[] args) {
    JSchExecutor jSchUtil = new JSchExecutor( "domp", "bluemoon2016#","192.168.243.21");
    try {
        jSchUtil.connect();
        jSchUtil.uploadFile("C:\\data\\applogs\\bd-job\\jobhandler\\2020-03-07\\employee.py","/data/applog
        //jSchUtil.execCmd("python /data/bluemoon/kettle/runScript/ods/fact_org_employee.py 'so you is wah
        jSchUtil.execCmd("python /data/applogs/bd-job/jobhandler/gluesource/employee.py 中文名称");
        //jSchUtil.execCmd("cat /data/applogs/bd-job/jobhandler/test");
    } catch (Exception e) {
        e.printStackTrace();
    }finally {
        jSchUtil.disconnect();
    }

}
```