SSH-2实现java连接远程服务器并执行脚本命令

原创置顶 ccx\_jy 最后发布于2018-07-04 17:46:50 阅读数 8660 收藏

展开

参考文档：

        maven jar包：https://mvnrepository.com/artifact/ch.ethz.ganymed/ganymed-ssh2

        Ganymed SSH2 API文档 ：http://www.ganymed.ethz.ch/ssh2/javadoc/overview-summary.html

Java的Ganymed SSH2是一个在纯Java中实现SSH-2协议的库(在J2SE 1.4.2和5.0上测试过)。

它允许从Java程序中连接到SSH服务器。

它支持SSH会话(远程命令执行和shell访问)、本地和远程端口转发、本地流转发、X11转发、SCP和SFTP。

不依赖于任何JCE提供程序，因为所有加密功能都包含在内。

API简介

一、API中常用的几个类

Connection：创建服务器连接以及认证（用户名密码认证、秘钥认证等）

Session ：会话

SCPClient：可用于文件在不同服务器之间复制。在服务器端，路径必须存在，如果不存在会报错。

                 线程安全的——您可以同时下载(和上传)不同的文件集，没有任何问题。

                 SCPClient实际上是将每个请求映射到不同的会话。

SFTPv3Client： 执条单个指令（推荐使用） 提供了很多方法

二、使用步骤

        1、creates a Connection object.

        2、calls the connect() method.

        3、calls some of the authentication methods (e.g., authenticateWithPublicKey()).

        4、calls one or several times the openSession() method.

三、需要注意

        1、认证成功后，当前目录就位于/home/username/目录之下，

            你可以指定脚本文件所在的绝对路径，或者通过cd导航到脚本文件所在的目录，

            然后传递执行脚本所需要的参数，完成脚本调用执行。

        2、执行脚本以后，可以获取脚本执行的结果文本，需要对这些文本进行正确编码后返回给客户端，避免乱码产生

        3、如果你需要执行多个linux控制台脚本，比如第一个脚本的返回结果是第二个脚本的入参，你必须打开多个Session,也就是多次调用Session sess = conn.openSession();,使用完毕记得关闭就可以了

四、代码

import java.io.BufferedReader;

import java.io.File;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.UnsupportedEncodingException;

import org.apache.log4j.Logger;

import com.viewhigh.vadp.framework.data.util.StringUtils;

import ch.ethz.ssh2.Connection;

import ch.ethz.ssh2.SCPClient;

import ch.ethz.ssh2.Session;

import ch.ethz.ssh2.StreamGobbler;

/\*\*

\*

\* 描述：连接linux服务器并执行相关的shell命令

\* 作者：cuicx

\* 版本：V1.0

\* 创建日期：2018年7月4日

\* 修改日期: 2018年7月4日

\*/

public class ConnectLinuxCommand {

private static final Logger logger = Logger.getLogger(ConnectLinuxCommand.class);

private static String DEFAULTCHARTSET = "UTF-8";

private static Connection conn;

/\*\*

\*

\* @Title: login

\* @Description: 用户名密码方式 远程登录linux服务器

\* @return: Boolean

\* @throws

\*/

public static Boolean login(RemoteConnect remoteConnect) {

boolean flag = false;

try {

conn = new Connection(remoteConnect.getIp());

conn.connect();// 连接

flag = conn.authenticateWithPassword(remoteConnect.getUserName(), remoteConnect.getPassword());// 认证

if (flag) {

logger.info("认证成功！");

} else {

logger.info("认证失败！");

conn.close();

}

} catch (IOException e) {

e.printStackTrace();

}

return flag;

}

/\*\*

\*

\* @Title: loginByKey

\* @Description: 秘钥方式 远程登录linux服务器

\* @param remoteConnect

\* @param keyFile 一个文件对象指向一个文件，该文件包含OpenSSH密钥格式的用户的DSA或RSA私钥(PEM，不能丢失"-----BEGIN DSA PRIVATE KEY-----" or "-----BEGIN RSA PRIVATE KEY-----"标签

\* @param keyfilePass 如果秘钥文件加密 需要用该参数解密，如果没有加密可以为null

\* @return Boolean

\* @throws

\*/

public static Boolean loginByFileKey(RemoteConnect remoteConnect, File keyFile, String keyfilePass) {

boolean flag = false;

// 输入密钥所在路径

// File keyfile = new File("C:\\temp\\private");

try {

conn = new Connection(remoteConnect.getIp());

conn.connect();

// 登录认证

flag = conn.authenticateWithPublicKey(remoteConnect.getUserName(), keyFile, keyfilePass);

if (flag) {

logger.info("认证成功！");

} else {

logger.info("认证失败！");

conn.close();

}

} catch (Exception e) {

e.printStackTrace();

}

return flag;

}

/\*\*

\*

\* @Title: loginByCharsKey

\* @Description: 秘钥方式 远程登录linux服务器

\* @param remoteConnect

\* @param keys 一个字符[]，其中包含用户的DSA或RSA私钥(OpenSSH密匙格式，您不能丢失“----- begin DSA私钥-----”或“-----BEGIN RSA PRIVATE KEY-----“标签。char数组可以包含换行符/换行符。

\* @param keyPass 如果秘钥字符数组加密 需要用该字段解密 否则不需要可以为null

\* @return Boolean

\* @throws

\*/

public static Boolean loginByCharsKey(RemoteConnect remoteConnect, char[] keys, String keyPass) {

boolean flag = false;

// 输入密钥所在路径

// File keyfile = new File("C:\\temp\\private");

try {

conn = new Connection(remoteConnect.getIp());

conn.connect();

// 登录认证

flag = conn.authenticateWithPublicKey(remoteConnect.getUserName(), keys, keyPass);

if (flag) {

logger.info("认证成功！");

} else {

logger.info("认证失败！");

conn.close();

}

} catch (Exception e) {

e.printStackTrace();

}

return flag;

}

/\*\*

\*

\* @Title: execute

\* @Description: 远程执行shll脚本或者命令

\* @param cmd 脚本命令

\* @return: result 命令执行完毕返回结果

\* @throws

\*/

public static String execute(String cmd){

String result = "";

try {

Session session = conn.openSession();// 打开一个会话

session.execCommand(cmd);// 执行命令

result = processStdout(session.getStdout(), DEFAULTCHARTSET);

// 如果为得到标准输出为空，说明脚本执行出错了

if (StringUtils.isBlank(result)) {

result = processStdout(session.getStderr(), DEFAULTCHARTSET);

}

conn.close();

session.close();

} catch (IOException e) {

e.printStackTrace();

}

return result;

}

/\*\*

\* @Title: executeSuccess

\* @Description: 远程执行shell脚本或者命令

\* @param shell脚本或者命令

\* @return String 命令执行成功后返回的结果值，如果命令执行失败，返回空字符串，不是null

\* @throws

\*/

public static String executeSuccess(String cmd){

String result = "";

try {

Session session = conn.openSession();// 打开一个会话

session.execCommand(cmd);// 执行命令

result = processStdout(session.getStdout(), DEFAULTCHARTSET);

conn.close();

session.close();

} catch (IOException e) {

e.printStackTrace();

}

return result;

}

/\*\*

\*

\* @Title: processStdout

\* @Description: 解析脚本执行的返回结果

\* @param in 输入流对象

\* @param charset 编码

\* @return String 以纯文本的格式返回

\* @throws

\*/

public static String processStdout(InputStream in, String charset){

InputStream stdout = new StreamGobbler(in);

StringBuffer buffer = new StringBuffer();

try {

BufferedReader br = new BufferedReader(new InputStreamReader(stdout, charset));

String line = null;

while ((line = br.readLine()) != null) {

buffer.append(line + "\n");

}

} catch (UnsupportedEncodingException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return buffer.toString();

}

/\*\*

\*

\* @Title: ConnectLinux

\* @Description: 通过用户名和密码关联linux服务器

\* @return

\* @return String

\* @throws

\*/

public static boolean connectLinux(String ip,String userName,String password,String commandStr) {

logger.info("ConnectLinuxCommand scpGet===" + "ip:" + ip + " userName:" + userName + " commandStr:"

+ commandStr);

String returnStr = "";

boolean result = true;

RemoteConnect remoteConnect = new RemoteConnect();

remoteConnect.setIp(ip);

remoteConnect.setUserName(userName);

remoteConnect.setPassword(password);

try {

if (login(remoteConnect)) {

returnStr = execute(commandStr);

System.out.println(result);

}

} catch (Exception e) {

e.printStackTrace();

}

if (StringUtils.isBlank(returnStr)) {

result = false;

}

return result;

}

/\*\*

\*

\* @Title: scpGet

\* @Description: 从其他服务器获取文件到本服务器指定目录

\* @param host ip(其他服务器)

\* @param username 用户名(其他服务器)

\* @param password 密码(其他服务器)

\* @param remoteFile 文件位置(其他服务器)

\* @param localDir 本服务器目录

\* @throws IOException

\* @return void

\* @throws

\*/

public static void scpGet(String ip, String userName, String password, String remoteFile, String localDir)

throws IOException {

logger.info("ConnectLinuxCommand scpGet===" + "ip:" + ip + " userName:" + userName + " remoteFile:"

+ remoteFile + " localDir:" + localDir);

RemoteConnect remoteConnect = new RemoteConnect();

remoteConnect.setIp(ip);

remoteConnect.setUserName(userName);

remoteConnect.setPassword(password);

if (login(remoteConnect)) {

SCPClient client = new SCPClient(conn);

client.get(remoteFile, localDir);

conn.close();

}

}

/\*\*

\*

\* @Title: scpPut

\* @Description: 将文件复制到其他计算机中

\* @param host

\* @param username

\* @param password

\* @param localFile

\* @param remoteDir

\* @throws IOException

\* @return void

\* @throws

\*/

public static void scpPut(String ip, String userName, String password, String localFile, String remoteDir)

throws IOException {

logger.info("ConnectLinuxCommand scpPut===" + "ip:" + ip + " userName:" + userName + " localFile:"

+ localFile + " remoteDir:" + remoteDir);

RemoteConnect remoteConnect = new RemoteConnect();

remoteConnect.setIp(ip);

remoteConnect.setUserName(userName);

remoteConnect.setPassword(password);

if (login(remoteConnect)) {

SCPClient client = new SCPClient(conn);

client.put(localFile, remoteDir);

conn.close();

}

}

}

实体类：

public class RemoteConnect {

private String ip;

private String userName;

private String password;

public String getIp() {

return ip;

}

public void setIp(String ip) {

this.ip = ip;

}

public String getUserName() {

return userName;

}

public void setUserName(String userName) {

this.userName = userName;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

测试用例

public class CcxTest {

@Test

public void testDemo() {

//cd /home/ubuntu/Desktop/music\_rec/user\_sim/result;cat xyy\_result\_m10d.json

//String commandStr="cd /home;rm -rf aa.txt";

String commandStr="cd /home;cat aa.txt;rm -rf aa.txt";

Boolean result=ConnectLinuxCommand.connectLinux("192.168.42.201","root","123456",commandStr);

System.out.println(result);

}

@Test

public void testDemo2() throws IOException {

try {

ConnectLinuxCommand.scpGet("192.168.42.201","root","123456", "/home/aa.txt", "d:/aa");

} catch (Exception e) {

e.printStackTrace();

}

}

@Test

public void testDemo3() {

try {

ConnectLinuxCommand.scpPut("192.168.42.201","root","123456", "d:/aa/aa.txt", "/home/bb");

} catch (Exception e) {

e.printStackTrace();

}

}

}

————————————————

版权声明：本文为CSDN博主「ccx\_jy」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/chuanxincui/java/article/details/80916205