链滴

实现 SFTP 连接池进行大数据量文件上传

作者：sigeisment

原文链接：https://ld246.com/article/1617185029813

来源网站：链滴

实现 SFTP 连接池进行大数据量文件上传

SFTP 连接池实现

主要基于 apache 的 commons-pool2 技术对 SFTP 连接对象进行池化。实现类：

- SftpContractUtils (SFTP 连接对象)
- SftpContractProperties (SFTP 连接配置)
- SftpContractFactory (SFTP 连接对象创建工厂)
- SftpContractPool (SFTP 连接池)
- SftpPoolAutoConfiguration （SFTP 连接池自动装配类）

SFTP 连接对象实现

JSch 是 Java Secure Channel 的缩写。 JSch 是一个 SSH2 的纯 Java 实现。它允许你连接到一个 SS 服务器，并且可以使用端口转发， X11 转发，文件传输等。我们使用使用 JSch 实现的 SFTP 功能。

Maven 依赖（早期版本存在与服务端加密方法协商问题）：

```
<dependency>
    <groupId>com.jcraft</groupId>
    <artifactId>jsch</artifactId>
    <version>0.1.55</version>
</dependency>
```

SftpContractUtils 类：

```
package org.test.infra.util;
```

```
import com.alibaba.fastjson.JSON;
import com.jcraft.jsch.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.test.config.SftpContractProperties;
import org.thymeleaf.util.StringUtils;

import java.io.BufferedReader;
import java.io.File;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.nio.charset.Charset;
import java.util.*;

public class SftpContractUtils {

    /** 日志记录器 */
    private static final Logger logger = LoggerFactory.getLogger(SftpContractUtils.class);

    // 设置编码格式
    private String charset = "UTF-8";

    /** Session */
    private Session session = null;
    /** Channel */
    private ChannelSftp channel = null;

    /** 规避多线程并发不断开问题 */
    //private static ThreadLocal<SftpContractUtils> sftpLocal = new ThreadLocal<>();


    private SftpContractUtils(){

    }

    /**
     * SFTP 安全文件传送协议
     */
    private SftpContractUtils(SftpContractProperties properties){
        logger.info("【{}】start reconnect [contract]sftp-server...",System.currentTimeMillis());
        login(properties);
    }

    /**
     * 是否已连接
     *
     * @return
     */
    public boolean isConnected() {
        return null != channel && channel.isConnected();
    }
```

```
    /**
     * 获取sftp客户端
     * utilType == null || utilType == 1 return icon
     * utilType == 2 return contract
     * @return
     */
    public static SftpContractUtils getSftpUtils(SftpContractProperties properties) {
        return new SftpContractUtils(properties);
    }

    /**
     * 获取本地线程存储的sftp客户端
     * utilType == null || utilType == 1 return icon
     * utilType == 2 return contract
     * @return
     */
//    public static SftpContractUtils getLocalSftpUtils(SftpContractProperties properties) throws
Exception {
//        SftpContractUtils sftpUtils = sftpLocal.get();
//        if(sftpUtils == null || !sftpUtils.isConnected()){
//            sftpLocal.set(new SftpContractUtils(properties));
//        }
//        return sftpLocal.get();
//    }


    /**
     * 关闭通道
     *
     * @throws Exception
     */
    public void closeChannel() {
        if (null != channel) {
            try {
                channel.disconnect();
            } catch (Exception e) {
                logger.error("关闭SFTP通道发生异常:", e);
            }
        }
        if (null != session) {
            try {
                session.disconnect();
            } catch (Exception e) {
                logger.error("SFTP关闭 session异常:", e);
            }
        }
    }



    /**
     * 登陆SFTP服务器
     * @return boolean
     */
```

```
    public boolean login(SftpContractProperties properties) {

        try {
            JSch jsch = new JSch();
            // 用户名
            String contractUsername = properties.getUserName();
            // 密码
            String contractPassword = properties.getPassword();
            // SFTP服务器端口
            int sftpPort = properties.getPort();
            // SFTP服务器IP地址
            String sftpHost = properties.getHost();
            session = jsch.getSession(contractUsername, sftpHost, sftpPort);
            if(contractPassword != null){
                session.setPassword(contractPassword);
            }
            Properties config = new Properties();
            config.put("StrictHostKeyChecking", "no");
            session.setConfig(config);
            /**
             * 连接超时时间，单位毫秒
             */
            int sftpTimeout = 5000;
            session.setTimeout(sftpTimeout);
            session.connect();
            logger.debug("sftp session connected");

            logger.debug("opening channel");
            channel = (ChannelSftp)session.openChannel("sftp");
            channel.connect();

            logger.debug("connected successfully");
            return true;
        } catch (JSchException e) {
            logger.error("sftp login failed",e);
            return false;
        }
    }

    /**
     * 上传文件
     * <p>
     * 使用示例，SFTP服务器上的目录结构如下：/testA/testA_B/
     * <table border="1">
     * <tr><td>当前目录</td><td>方法</td><td>参数：绝对路径/相对路径</td><td>上传后</
d></tr>
     * <tr><td>/</td><td>uploadFile("testA","upload.txt",new FileInputStream(new File("up.tx
")))</td><td>相对路径</td><td>/testA/upload.txt</td></tr>
     * <tr><td>/</td><td>uploadFile("testA/testA_B","upload.txt",new FileInputStream(new Fi
e("up.txt")))</td><td>相对路径</td><td>/testA/testA_B/upload.txt</td></tr>
     * <tr><td>/</td><td>uploadFile("/testA/testA_B","upload.txt",new FileInputStream(new F
le("up.txt")))</td><td>绝对路径</td><td>/testA/testA_B/upload.txt</td></tr>
     * </table>
     * </p>
```

```
     * @param pathName SFTP服务器目录
     * @param fileName 服务器上保存的文件名
     * @param input 输入文件流
     * @return boolean
     */
    public boolean uploadFile(String pathName,String fileName,InputStream input) throws Sft
Exception {

        String currentDir = currentDir();
        if(!changeDir(pathName)){
            return false;
        }

        try {
            channel.put(input,fileName,ChannelSftp.OVERWRITE);
            if(!existFile(fileName)){
                logger.debug("upload failed");
                return false;
            }
            logger.debug("upload successful");
            return true;
        } catch (SftpException e) {
            logger.error("upload failed",e);
            return false;
        } finally {
            changeDir(currentDir);
        }
    }

    /**
     * 上传文件
     * <p>
     * 使用示例，SFTP服务器上的目录结构如下：/testA/testA_B/
     * <table border="1">
     * <tr><td>当前目录</td><td>方法</td><td>参数：绝对路径/相对路径</td><td>上传后</
d></tr>
     * <tr><td>/</td><td>uploadFile("testA","upload.txt","up.txt")</td><td>相对路径</td><
d>/testA/upload.txt</td></tr>
     * <tr><td>/</td><td>uploadFile("testA/testA_B","upload.txt","up.txt")</td><td>相对路
</td><td>/testA/testA_B/upload.txt</td></tr>
     * <tr><td>/</td><td>uploadFile("/testA/testA_B","upload.txt","up.txt")</td><td>绝对
径</td><td>/testA/testA_B/upload.txt</td></tr>
     * </table>
     * </p>
     * @param pathName SFTP服务器目录
     * @param fileName 服务器上保存的文件名
     * @param localFile 本地文件
     * @return boolean
     */
    public boolean uploadFile(String pathName,String fileName,String localFile) throws SftpExc
ption{

        String currentDir = currentDir();
        if(!changeDir(pathName)){
```

```
            return false;
        }

        try {
            channel.put(localFile,fileName,ChannelSftp.OVERWRITE);
            if(!existFile(fileName)){
                logger.debug("upload failed");
                return false;
            }
            logger.debug("upload successful");
            return true;
        } catch (SftpException e) {
            logger.error("upload failed",e);
            return false;
        } finally {
            changeDir(currentDir);
        }
    }

    /**
     * 下载文件
     * <p>
     * 使用示例，SFTP服务器上的目录结构如下：/testA/testA_B/
     * <table border="1">
     * <tr><td>当前目录</td><td>方法</td><td>参数：绝对路径/相对路径</td><td>下载后</
d></tr>
     * <tr><td>/</td><td>downloadFile("testA","down.txt","D:\\downDir")</td><td>相对路
</td><td>D:\\downDir\\down.txt</td></tr>
     * <tr><td>/</td><td>downloadFile("testA/testA_B","down.txt","D:\\downDir")</td><td
相对路径</td><td>D:\\downDir\\down.txt</td></tr>
     * <tr><td>/</td><td>downloadFile("/testA/testA_B","down.txt","D:\\downDir")</td><td
绝对路径</td><td>D:\\downDir\\down.txt</td></tr>
     * </table>
     * </p>
     * @param remotePath SFTP服务器目录
     * @param fileName 服务器上需要下载的文件名
     * @param localPath 本地保存路径
     * @return boolean
     */
    public boolean downloadFile(String remotePath,String fileName,String localPath) throws Sf
pException{

        String currentDir = currentDir();
        if(!changeDir(remotePath)){
            return false;
        }

        try {
            String localFilePath = localPath + File.separator + fileName;
            channel.get(fileName,localFilePath);

            File localFile = new File(localFilePath);
            if(!localFile.exists()){
                logger.debug("download file failed");
```

```
            return false;
        }
        logger.debug("download successful");
        return true;
    } catch (SftpException e) {
        logger.error("download file failed",e);
        return false;
    } finally {
        changeDir(currentDir);
    }
}

/**
 * 切换工作目录
 * <p>
 * 使用示例，SFTP服务器上的目录结构如下：/testA/testA_B/
 * <table border="1">
 * <tr><td>当前目录</td><td>方法</td><td>参数(绝对路径/相对路径)</td><td>切换后的
录</td></tr>
 * <tr><td>/</td><td>changeDir("testA")</td><td>相对路径</td><td>/testA/</td></t
>
 * <tr><td>/</td><td>changeDir("testA/testA_B")</td><td>相对路径</td><td>/testA/te
tA_B/</td></tr>
 * <tr><td>/</td><td>changeDir("/testA")</td><td>绝对路径</td><td>/testA/</td></t
>
 * <tr><td>/testA/testA_B/</td><td>changeDir("/testA")</td><td>绝对路径</td><td>/t
stA/</td></tr>
 * </table>
 * </p>
 * @param pathName 路径
 * @return boolean
 */
public boolean changeDir(String pathName) throws SftpException{
    if(pathName == null || pathName.trim().equals("")){
        logger.error("invalid pathName");
        return false;
    }
    logger.debug("changeDir【{}】,before:directory successfully changed,current dir=" + ch
nnel.pwd(),pathName);
    try {
        channel.cd(pathName.replaceAll("\\\\", "/"));
    } catch (Exception e1) {
        logger.error("cd dir error", e1);
        pathName = pathName.replaceAll("\\\\", "/");
        String[] folders = pathName.split( "/" );
        for ( String folder : folders ) {
            if ( folder.length() > 0 ) {
                try {
                    channel.cd( folder );
                }
                catch ( SftpException e ) {
                    channel.mkdir( folder );
                    channel.cd( folder );
                }
```

```
            }
        }
    }

    logger.debug("changeDir【{}】,after:directory successfully changed,current dir=" + chan
el.pwd(),pathName);
    return true;

}


/**
 * 切换到上一级目录
 * <p>
 * 使用示例，SFTP服务器上的目录结构如下：/testA/testA_B/
 * <table border="1">
 * <tr><td>当前目录</td><td>方法</td><td>切换后的目录</td></tr>
 * <tr><td>/testA/</td><td>changeToParentDir()</td><td>/</td></tr>
 * <tr><td>/testA/testA_B/</td><td>changeToParentDir()</td><td>/testA/</td></tr>
 * </table>
 * </p>
 * @return boolean
 */
public boolean changeToParentDir() throws SftpException{
    return changeDir("..");
}

/**
 * 切换到根目录
 * @return boolean
 */
public boolean changeToHomeDir() throws SftpException{
    String homeDir = null;
    homeDir = channel.getHome();
    return changeDir(homeDir);
}

/**
 * 创建目录
 * <p>
 * 使用示例，SFTP服务器上的目录结构如下：/testA/testA_B/
 * <table border="1">
 * <tr><td>当前目录</td><td>方法</td><td>参数(绝对路径/相对路径)</td><td>创建成功
的目录</td></tr>
 * <tr><td>/testA/testA_B/</td><td>makeDir("testA_B_C")</td><td>相对路径</td><td>
testA/testA_B/testA_B_C/</td></tr>
 * <tr><td>/</td><td>makeDir("/testA/testA_B/testA_B_D")</td><td>绝对路径</td><td
/testA/testA_B/testA_B_D/</td></tr>
 * </table>
 * <br/>
 * <b>注意</b>，当<b>中间目录不存在</b>的情况下，不能够使用绝对路径的方式期望创建
间目录及目标目录。
 * 例如makeDir("/testNOEXIST1/testNOEXIST2/testNOEXIST3")，这是错误的。
 * </p>
```

```
     * @param dirName 目录
     * @return boolean
     */
    public boolean makeDir(String dirName){
        try {
            channel.mkdir(dirName);
            logger.debug("directory successfully created,dir=" + dirName);
            return true;
        } catch (SftpException e) {
            logger.error("failed to create directory", e);
            return false;
        }
    }

    /**
     * 删除文件夹
     * @param dirName
     * @return boolean
     */
    @SuppressWarnings("unchecked")
    public boolean delDir(String dirName) throws SftpException{
        if(!changeDir(dirName)){
            return false;
        }

        Vector<ChannelSftp.LsEntry> list = null;
        try {
            list = channel.ls(channel.pwd());
        } catch (SftpException e) {
            logger.error("can not list directory",e);
            return false;
        }

        for(ChannelSftp.LsEntry entry : list){
            String fileName = entry.getFilename();
            if(!fileName.equals(".") && !fileName.equals("..")){
                if(entry.getAttrs().isDir()){
                    delDir(fileName);
                } else {
                    delFile(fileName);
                }
            }
        }

        if(!changeToParentDir()){
            return false;
        }

        try {
            channel.rmdir(dirName);
            logger.debug("directory " + dirName + " successfully deleted");
            return true;
        } catch (SftpException e) {
            logger.error("failed to delete directory " + dirName,e);
```

```
            return false;
        }
    }

    /**
     * 删除文件
     * @param fileName 文件名
     * @return boolean
     */
    public boolean delFile(String fileName){
        if(fileName == null || fileName.trim().equals("")){
            logger.debug("invalid filename");
            return false;
        }

        try {
            channel.rm(fileName);
            logger.debug("file " + fileName + " successfully deleted");
            return true;
        } catch (SftpException e) {
            logger.error("failed to delete file " + fileName,e);
            return false;
        }
    }

    /**
     * 当前目录下文件及文件夹名称列表
     * @return String[]
     */
    public String[] ls(){
        return list(Filter.ALL);
    }

    /**
     * 当前目录下文件及文件夹名称列表
     * @return String[]
     */
    public Optional<ChannelSftp.LsEntry> lsFile(String pathName, String fileName) throws Sft
Exception {
        if(!changeDir(pathName)){
            logger.debug(" changeDir to pathName [{}] 失败",pathName);
            return Optional.empty();
        };
        Vector<ChannelSftp.LsEntry> list = null;
        try {
            //ls方法会返回两个特殊的目录，当前目录(.)和父目录(..)
            list = channel.ls(channel.pwd());
        } catch (SftpException e) {
            logger.error("can not list directory",e);
            return Optional.empty();
        }

        for(ChannelSftp.LsEntry entry : list){
            if(StringUtils.equals(entry.getFilename(), fileName)){
```

```
            return Optional.of(entry);
        }
    }
    return Optional.empty();
}

/**
 * 指定目录下文件及文件夹名称列表
 * @return String[]
 */
public String[] ls(String pathName) throws SftpException{
    String currentDir = currentDir();
    logger.debug("currentDir:{} , ls pathName:{} ",currentDir,pathName);
    if(!changeDir(pathName)){
        logger.debug(" changeDir to pathName [{}] 失败",pathName);
        return new String[0];
    };
    String[] result = list(Filter.ALL);
    if(!changeDir(currentDir)){
        logger.debug(" changeDir to currentDir [{}] 失败",currentDir);
        return new String[0];
    }
    logger.debug(" ls pathName result:{} ",JSON.toJSONString(result));
    return result;
}

/**
 * 当前目录下文件名称列表
 * @return String[]
 */
public String[] lsFiles(){
    return list(Filter.FILE);
}

/**
 * 指定目录下文件名称列表
 * @return String[]
 */
public String[] lsFiles(String pathName) throws SftpException{
    String currentDir = currentDir();
    if(!changeDir(pathName)){
        return new String[0];
    };
    String[] result = list(Filter.FILE);
    if(!changeDir(currentDir)){
        return new String[0];
    }
    return result;
}

/**
 * 当前目录下文件夹名称列表
 * @return String[]
 */
```

```
public String[] lsDirs(){
    return list(Filter.DIR);
}

/**
 * 指定目录下文件夹名称列表
 * @return String[]
 */
public String[] lsDirs(String pathName) throws SftpException{
    String currentDir = currentDir();
    if(!changeDir(pathName)){
        return new String[0];
    };
    String[] result = list(Filter.DIR);
    if(!changeDir(currentDir)){
        return new String[0];
    }
    return result;
}

/**
 * 当前目录是否存在文件或文件夹
 * @param name 名称
 * @return boolean
 */
public boolean exist(String name){
    return exist(ls(), name);
}

/**
 * 指定目录下，是否存在文件或文件夹
 * @param path 目录
 * @param name 名称
 * @return boolean
 */
public boolean exist(String path,String name) throws SftpException{
    return exist(ls(path),name);
}

/**
 * 当前目录是否存在文件
 * @param name 文件名
 * @return boolean
 */
public boolean existFile(String name){
    return exist(lsFiles(),name);
}

/**
 * 指定目录下，是否存在文件
 * @param path 目录
 * @param name 文件名
 * @return boolean
 */
```

```
public boolean existFile(String path,String name) throws SftpException{
    return exist(lsFiles(path), name);
}

/**
 * 当前目录是否存在文件夹
 * @param name 文件夹名称
 * @return boolean
 */
public boolean existDir(String name){
    return exist(lsDirs(), name);
}

/**
 * 指定目录下，是否存在文件夹
 * @param path 目录
 * @param name 文家夹名称
 * @return boolean
 */
public boolean existDir(String path,String name) throws SftpException{
    return exist(lsDirs(path), name);
}

/**
 * 当前工作目录
 * @return String
 */
public String currentDir() throws SftpException {
    return channel.pwd();
}

/**
 * 登出
 */
public void logout(){
    if(channel != null){
        channel.quit();
        channel.disconnect();
    }
    if(session != null){
        session.disconnect();
    }
    logger.debug("logout successfully");
}


//------private method ------

/** 枚举，用于过滤文件和文件夹  */
private enum Filter {/** 文件及文件夹 */ ALL ,/** 文件 */ FILE ,/** 文件夹 */ DIR };

/**
 * 列出当前目录下的文件及文件夹
 * @param filter 过滤参数
```

```
     * @return String[]
     */
    @SuppressWarnings("unchecked")
    private String[] list(Filter filter){
        Vector<ChannelSftp.LsEntry> list = null;
        try {
            //ls方法会返回两个特殊的目录，当前目录(.)和父目录(..)
            list = channel.ls(channel.pwd());
        } catch (SftpException e) {
            logger.error("can not list directory",e);
            return new String[0];
        }

        List<String> resultList = new ArrayList<String>();
        for(ChannelSftp.LsEntry entry : list){
            if(filter(entry, filter)){
                resultList.add(entry.getFilename());
            }
        }
        return resultList.toArray(new String[0]);
    }

    /**
     * 判断是否是否过滤条件
     * @param entry LsEntry
     * @param f 过滤参数
     * @return boolean
     */
    private boolean filter(ChannelSftp.LsEntry entry, Filter f){
        if(f.equals(Filter.ALL)){
            return !".".equals(entry.getFilename()) && !"..".equals(entry.getFilename());
        } else if(f.equals(Filter.FILE)){
            return !".".equals(entry.getFilename()) && !"..".equals(entry.getFilename()) && !entry.g
tAttrs().isDir();
        } else if(f.equals(Filter.DIR)){
            return !".".equals(entry.getFilename()) && !"..".equals(entry.getFilename()) && entry.ge
Attrs().isDir();
        }
        return false;
    }

    /**
     * 根目录
     * @return String
     */
    private String homeDir() throws SftpException {
        return channel.getHome();
    }

    /**
     * 判断字符串是否存在于数组中
     * @param strArr 字符串数组
     * @param str 字符串
     * @return boolean
```

```
     */
    private boolean exist(String[] strArr,String str){
        if(strArr == null || strArr.length == 0){
            return false;
        }
        if(str == null || "".equals(str.trim())){
            return false;
        }
        for(String s : strArr){
            if(s.equalsIgnoreCase(str)){
                return true;
            }
        }
        return false;
    }

    /**
     * 远程打包zip，目标目录下的所有文件
     * @param zipPathName 要打包的目录
     * @param zipFileName 打包的文件名
     * @return
     * @throws SftpException
     */
    public Boolean zipRemoteFiles(String zipPathName,String zipFileName) throws Exception{

        if(!changeDir(zipPathName)){
            return false;
        }
        String[] files = lsFiles();
        if(files == null || files.length == 0){
            return null;
        }
        String currentDir = currentDir();
        changeDir(currentDir);
        String comd = "zip -r -o -q "+currentDir+"/"+zipFileName+" "+currentDir+"/*";
        String cmdRtn = execCmd(comd);
        logger.info("currentDir:{} execCmd[{}] return:【{}】",currentDir,comd,cmdRtn);
        return true;
    }


    /**
     * 执行一条命令
     */
    public String execCmd(String command) {
        InputStream in = null;
        BufferedReader reader = null;
        Channel channel = null;

        String buf = null;
        try {
            channel = session.openChannel("exec");
            ((ChannelExec) channel).setCommand(command);
            channel.setInputStream(null);
```

```
            ((ChannelExec) channel).setErrStream(System.err);
            channel.connect();
            in = channel.getInputStream();
            reader = new BufferedReader(new InputStreamReader(in, Charset.forName(charset)));

            while ((buf = reader.readLine()) != null) {
                System.out.println(buf);
            }
            channel.disconnect();
        } catch (Exception e) {
            logger.error(" SFTP execCmd ERROR:",e);
        }
        return buf;
    }
}
```

SFTP 连接配置

配置 IP ,用户名，密码，连接池相关配置

```
package org.test.config;

import org.springframework.boot.context.properties.ConfigurationProperties;

@ConfigurationProperties(prefix = SftpContractProperties.PROJECT_PREFIX)
public class SftpContractProperties {
    public static final String PROJECT_PREFIX = "hzero.sftp";
    /**
     * 用户名
     */
    private String userName;
    /**
     * 密码
     */
    private String password;
    /**
     * SFTP服务器端口
     */
    private int port;
    /**
     * SFTP服务器IP地址
     */
    private String host;

    private SftpContractPoolProperties poolProperties;

    public static class SftpContractPoolProperties {
        private String poolPrefix = "hzero.sftp";
        /**
         * 最大空闲
         */
        private int maxIdle = 5;
        /**
         * 最大总数
```

```
 */
private int maxTotal = 10;
/**
 * 最小空闲
 */
private int minIdle = 0;

/**
 * 初始化连接数
 */
private int initialSize = 3;

public String getPoolPrefix() {
    return poolPrefix;
}

public SftpContractPoolProperties setPoolPrefix(String poolPrefix) {
    this.poolPrefix = poolPrefix;
    return this;
}

public int getMaxIdle() {
    return maxIdle;
}

public SftpContractPoolProperties setMaxIdle(int maxIdle) {
    this.maxIdle = maxIdle;
    return this;
}

public int getMaxTotal() {
    return maxTotal;
}

public SftpContractPoolProperties setMaxTotal(int maxTotal) {
    this.maxTotal = maxTotal;
    return this;
}

public int getMinIdle() {
    return minIdle;
}

public SftpContractPoolProperties setMinIdle(int minIdle) {
    this.minIdle = minIdle;
    return this;
}

public int getInitialSize() {
    return initialSize;
}

public SftpContractPoolProperties setInitialSize(int initialSize) {
    this.initialSize = initialSize;
```

```
            return this;
        }
    }

    public String getUserName() {
        return userName;
    }

    public SftpContractProperties setUserName(String userName) {
        this.userName = userName;
        return this;
    }

    public String getPassword() {
        return password;
    }

    public SftpContractProperties setPassword(String password) {
        this.password = password;
        return this;
    }

    public int getPort() {
        return port;
    }

    public SftpContractProperties setPort(int port) {
        this.port = port;
        return this;
    }

    public String getHost() {
        return host;
    }

    public SftpContractProperties setHost(String host) {
        this.host = host;
        return this;
    }

    public SftpContractPoolProperties getPoolProperties() {
        return poolProperties;
    }

    public SftpContractProperties setPoolProperties(SftpContractPoolProperties poolProperties
{
        this.poolProperties = poolProperties;
        return this;
    }
}
```

SFTP 连接对象创建工厂

继承 commons-pool2 的 BasePooledObjectFactory 类，实现 create，wrap 等方法就可以实现一

对象创建工厂。

```
package org.test.infra.util;

import org.apache.commons.pool2.BasePooledObjectFactory;
import org.apache.commons.pool2.PooledObject;
import org.apache.commons.pool2.impl.DefaultPooledObject;
import org.test.config.SftpContractProperties;


public class SftpContractFactory extends BasePooledObjectFactory<SftpContractUtils> {
    private SftpContractProperties properties;

    public SftpContractFactory(SftpContractProperties properties){
        this.properties = properties;
    }

    @Override
    public SftpContractUtils create() throws Exception {
        // 创建新对象
        return SftpContractUtils.getSftpUtils(properties);
    }

    @Override
    public PooledObject<SftpContractUtils> wrap(SftpContractUtils sftpContractUtils) {
        // 池化对象
        return new DefaultPooledObject<>(sftpContractUtils);
    }

    @Override
    public void passivateObject(PooledObject<SftpContractUtils> p) throws Exception {
        // 将对象返回池时进行的操作，此处将工作目录设置为根目录
        p.getObject().changeToHomeDir();
    }

    @Override
    public void destroyObject(PooledObject<SftpContractUtils> p) throws Exception {
        p.getObject().closeChannel();
    }

    @Override
    public boolean validateObject(PooledObject<SftpContractUtils> p) {
        return p.getObject().isConnected();
    }
}
```

SFTP 连接池

继承 GenericObjectPool ,连接池将工厂对象与配置对象结合。 GenericObjectPool 中有 borrowObject , returnObject 等方法供我们获取，回归对象。

```
package org.test.infra.util;

import org.apache.commons.pool2.PooledObjectFactory;
```

```
import org.apache.commons.pool2.impl.AbandonedConfig;
import org.apache.commons.pool2.impl.GenericObjectPool;
import org.apache.commons.pool2.impl.GenericObjectPoolConfig;

public class SftpContractPool extends GenericObjectPool<SftpContractUtils> {
    public SftpContractPool(PooledObjectFactory<SftpContractUtils> factory) {
        super(factory);
    }

    public SftpContractPool(PooledObjectFactory<SftpContractUtils> factory, GenericObjectP
olConfig config) {
        super(factory, config);
    }

    public SftpContractPool(PooledObjectFactory<SftpContractUtils> factory, GenericObjectP
olConfig config, AbandonedConfig abandonedConfig) {
        super(factory, config, abandonedConfig);
    }
}
```

SFTP 连接池自动装配类

```
package org.test.autoconfigure;

import org.apache.commons.pool2.impl.GenericObjectPoolConfig;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.autoconfigure.condition.ConditionalOnClass;
import org.springframework.boot.context.properties.EnableConfigurationProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.test.config.SftpContractProperties;
import org.stest.infra.util.SftpContractFactory;
import org.test.infra.util.SftpContractPool;

@EnableConfigurationProperties(SftpContractProperties.class)
@Configuration
public class SftpPoolAutoConfiguration {
    private SftpContractPool pool;
    private SftpContractProperties properties;

    @Autowired
    public SftpPoolAutoConfiguration(SftpContractProperties properties) {
        this.properties = properties;
    }

    @ConditionalOnClass({SftpContractFactory.class})
    @Bean
    protected SftpContractPool faceSDKPool() {
        SftpContractFactory faceSDKFactory = new SftpContractFactory(properties);
        //设置对象池的相关参数
        SftpContractProperties.SftpContractPoolProperties poolProperties = properties.getPoolP
operties();
        if (poolProperties == null) {
            poolProperties = new SftpContractProperties.SftpContractPoolProperties();
```

```
        }
        GenericObjectPoolConfig poolConfig = new GenericObjectPoolConfig();
        poolConfig.setMaxIdle(poolProperties.getMaxIdle());
        poolConfig.setMaxTotal(poolProperties.getMaxTotal());
        poolConfig.setMinIdle(poolProperties.getMinIdle());
        poolConfig.setBlockWhenExhausted(true);
        poolConfig.setTestOnBorrow(true);
        poolConfig.setTestOnReturn(true);
        poolConfig.setTestWhileIdle(true);
        poolConfig.setTimeBetweenEvictionRunsMillis(1000 * 60 * 30);
        //一定要关闭jmx，不然springboot启动会报已经注册了某个jmx的错误
        poolConfig.setJmxEnabled(false);

        //新建一个对象池,传入对象工厂和配置
        pool = new SftpContractPool(faceSDKFactory, poolConfig);

        return pool;
    }
}
```

通过管道流进行文件上传

实例：

```
PipedInputStream in = new PipedInputStream();
PipedOutputStream out = new PipedOutputStream(in);
new Thread(
        () -> {
            generate(out);
        }
).start();
SftpContractUtils sftpContractUtils = sftpContractPool.borrowObject();
String fileName = "test1.txt";
String path = "/test"
boolean uploadFileSuccess = sftpContractUtils.uploadFile(path, fileName, in);
sftpContractPool.returnObject(sftpContractUtils);
```

注意：

管道输入流应该连接到管道输出流；管道输入流提供要写入管道输出流的所有数据字节。通常，数据某个线程从 PipedInputStream 对象读取， 并由其他线程将其写入到相应的 PipedOutputStream 不建议对这两个对象尝试使用单个线程，因为这样可能死锁线程。管道输入流包含一个缓冲区，可在冲区限定的 范 围内将读操作和写操作分离开。如果向连接管道输出流提供数据字节的线程不再存在则认为该管道已损坏。

PipedInputStream 的缓冲区大小默认是 1024 字节，管道的读写操作是互相阻塞的，当缓冲区为空，读操作阻塞；当缓冲区满时，写操作阻塞。