SFTP File Transfer In Java Using Jsch Library

Khairul Islam Azam <Khairulislam.azam@gmail.com>

What is SFTP file transfer?

```
Secure File Transfer Protocol (SFTP) is a secure version of File Transfer Protocol
(FTP) and a part of the SSH Protocol for easy data transfer and data access over a
Secure Shell (SSH) data stream. SFTP is also known as SSH File Transfer Protocol.
```

Jsch library for SFTP

***JSch is a pure Java implementation of SSH2. JSch allows you to connect to** an sshd server and use port forwarding, X11 forwarding, file transfer, etc., and you can integrate its functionality into your own Java programs. JSch is licensed under BSD style license.*

Features

- Connection Method
- Disconnect Method
- File Upload Method
- Single File Download Method
- List Of File Download Method
- File Delete Method

Installation

1. Needs to install Sftp server
2. Needs Dependency of Jsch
    - Jsch library Website

IMPORTANT

Configuration

Before going to use all the method there are some rules , such as:

NOTE	host, port ,user, password, privatekey, passphrase must be configure in application.properties
TIP	User can use password or ssh key to communicate with a server. To first time communicate with server through ssh key one need to install ssh in the server.
WARNING	when one use ssk key. The key (public and private key must be in .ssh directory).Public key paste it into the server authorize_key file in the .ssh directory. And also when client try to first time connect to server with ssh key then he need to set passPharse for private key.

Table 1. All the method and it's demo format of access url in postman api. In postman api first select body and then enter the key and value

Method	Url Description
`public boolean uploadFile(MultipartFile[] file, String remoteFilePath)`	**file**: type is file and then select file, **remotepath**: /home/azam/Desktop/abc
`public boolean downloadFile(String fileName, String localFilePath, String remoteFilePath)`	**fileName**: type is text. Example: abc.jpeg or abc.txt, **remotepath**: /home/azam/Desktop/abc/, **localpath**: C:\Users\\BS585\Desktop\Newfolder\
`public boolean downloadListOfFile(String localFilePath, String remoteFilePath)`	**remotepath**: /home/azam/Desktop/abc/, **localpath**: C:\Users\\BS585\Desktop\Newfolder\
`public boolean deleteFile(String fileName, String remoteFilePath)`	**fileName**: type is text. Example: abc.jpeg or abc.txt, **remotepath**: /home/azam/Desktop/abc/

Connection method

This method is used for creating sftpchannel so that user can upload or download or deleting file. First we need jsch class after that creating session from jsch object giving the username, host, port, after that a Channel connected to sftp server (as a subsystem of the ssh server).

- **@return** : after connecting the channel it should return the connect sftpchannel.

click here to see the code

```
     private ChannelSftp createChannelSftp() {
        try {
            JSch jSch = new JSch();
            jSch.addIdentity(privatekey, passphrase.getBytes());
            Session session = jSch.getSession(username, host, port);
            session.setConfig("StrictHostKeyChecking", "no");
            session.connect(sessionTimeout);
            Channel channel = session.openChannel("sftp");
            channel.connect(channelTimeout);
            return (ChannelSftp) channel;
        } catch (JSchException ex) {
            logger.error("Create ChannelSftp error", ex);
        }

        return null;
    }
```

Disconnect method

This method is used for close all the connection.

- **@param** : channelSftp is the channel which connect successfully in connect() method *.

click here to see the code

```
    private void disconnectChannelSftp(ChannelSftp channelSftp) {
        try {
            if (channelSftp == null)
                return;

            if (channelSftp.isConnected())
                channelSftp.disconnect();

            if (channelSftp.getSession() != null)
                channelSftp.getSession().disconnect();

        } catch (Exception ex) {
            logger.error("SFTP disconnect error", ex.getMessage());
        }
    }
```

File Upload method

This method is used for file upload file from client to server.

- **@param localFilePath** : Here localFilePath is the local directory used for a client who upload file from his pc or any other device
- **@param remoteFilePath** : Here remoteFilepath is the directory of your server where you can store your file
- **@return** : after storing your file this method, return boolean value to ensure that your file is saved or not.

click here to see the code

```
    public boolean uploadFile(MultipartFile[] file, String remoteFilePath) {
        ChannelSftp channelSftp = createChannelSftp();

        if (remoteDirectoryCk(remoteFilePath)) {
            try {
                File convFile = convertMultiPartToFile(file);
                if (convFile.canRead()) {
                    channelSftp.put(new FileInputStream(convFile),
                            remoteFilePath + "/" + convFile.getName());
                    return true;
                }

            } catch (SftpException | FileNotFoundException ex) {
                logger.error("Error upload file", ex);
            } catch (IOException e) {
                logger.error("Error upload file", e);
            } finally {
                disconnectChannelSftp(channelSftp);
            }
        }
        return false;
    }
```

File Download method

This method is used for download file from server to client.

- **@param localFilePath** : localFilepath is the directory of the client path where download file should be store.
- **@param remoteFilePath** : Here remoteFilepath is the directory of your server can access your file
- **@return** : if it fetches data successfully then it sends true otherwise send false .

click here to see the code

```
   public boolean downloadFile(String fileName, String localFilePath, String
remoteFilePath) {
        ChannelSftp channelSftp = createChannelSftp();
        OutputStream outputStream;

        try {
            File file = new File(localFilePath + fileName);
            outputStream = new FileOutputStream(file);
            channelSftp.get(remoteFilePath + fileName, outputStream);
            file.createNewFile();
            return true;
        } catch (SftpException | IOException ex) {
            logger.error("Error download file", ex.getMessage());
        } finally {
            disconnectChannelSftp(channelSftp);
        }

        return false;
    }
```

List Of File Download method

This method is used downloading the list of file from server directory to client directory.

- **@param localFilePath** : url of client directory where file should be stored
- **@param remoteFilePath** : url of server directory from where list of files should fetch
- **@return** : if it fetches all data successfully then it sends true otherwise send false.

click here to see the code

```
   public boolean downloadListOfFile(String localFilePath, String remoteFilePath)
{
        ChannelSftp channelSftp = createChannelSftp();
        try {

            Vector<ChannelSftp.LsEntry> fileList = channelSftp.ls(remoteFilePath);

            for (ChannelSftp.LsEntry file : fileList) {
                if (!file.getFilename().startsWith(".")) {
                    channelSftp.get(remoteFilePath + file.getFilename(),
                            localFilePath + file.getFilename());
                }
            }

            return true;
        } catch (SftpException ex) {
            logger.error("Error downloading file list", ex.getMessage());
        } finally {
            disconnectChannelSftp(channelSftp);
        }

        return false;
    }
```

Delete method

This method is used for deleting file in server directory.

- **@param fileName** : fileName for delete
- **@param remoteFilePath** : server directory where the file stored
- **@return** : if it deleted successfully then it sends true otherwise send false.

click here to see the code

```
public boolean deleteFile(String fileName, String remoteFilePath) {
        ChannelSftp channelSftp = createChannelSftp();
        try {

            Vector fileList = channelSftp.ls(remoteFilePath);
            for (int i = 0; i < fileList.size(); i++) {
                ChannelSftp.LsEntry lsEntry = (ChannelSftp.LsEntry) fileList.get(
i);
                String file = lsEntry.getFilename();
                logger.info("access file name: " + file);
                if (file.equalsIgnoreCase(fileName)) {
                    channelSftp.rm(remoteFilePath + fileName);
                    return true;
                } else {
                    logger.info(fileName + " " + "not found");
                }
            }
        } catch (SftpException ex) {
            logger.error("Error deleting file file", ex.getMessage());
        } finally {
            disconnectChannelSftp(channelSftp);
        }
        return false;
    }
```

Additional Resources

Read

- Jsch. Official Website

Watch

- How to perform SFTP Operation Using Java

References

- [1] Jsch. Official Website
- [2] Sftp operation docs. github jsch