QtsHttp Java Sample Code for Android

Getting Started

Build the develop environment

QtsHttp Java Sample Code is developed using ADT Bundle for Windows. The ADT (Android Developer Tools) Bundle includes:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- The latest Android platform
- The latest Android system image for the emulator

You can download ADT Bundle from Android developer website.
http://developer.android.com/sdk/index.html

And you can read how to setting up the ADT Bundle on website.
http://developer.android.com/sdk/installing/bundle.html

Open project

Unzip sample code zip file, Open Eclipse and select a workspace at unzip folder. Now you can see Qsample in you worksapce.

Basic settings

Open AndroidManifest.xml
“Min SDK version” is equal 9
“Uses Permission” includes android.permission.INTERNET and android.permission.WRITE_EXTERNAL_STORAGE.

User Interface

Login

When you execute this sample code, the first page is login page (Fig. 1). After you type server ip, port, use SSL, user name, and password, click Login button. Wait a short time, then you can see NAS share folder list (Fig. 3).

Display NAS's file list

It will show the file list in the folder that you click it (Fig. 4).

Display device's Download folder file list

Click "Upload to here" button, it will show /mnt/sdcard/Download folder's file list (Fig. 6).

Download file

In Fig.4, if you long click file "1.jpg", the file will be downloaded from NAS to device's Download folder on SD Card. The progress dialog will be showed (Fig. 5). After download file successfully, you can see the file in Download folder (Filg. 6)

Upload file

In Fig. 7, go in to a NAS folder like "/Public/CCC/qqq". If you want to upload file to here, click "Upload to here" button, long click file "1.jpg" (Fig. 6). The progress dialog will be showed (Fig. 8) and the file is uploading. After upload file successfully, you can see the file in NAS folder (Filg. 9).

Logout

If you want to logout NAS, click "Logout" button, and it will go to show login page.

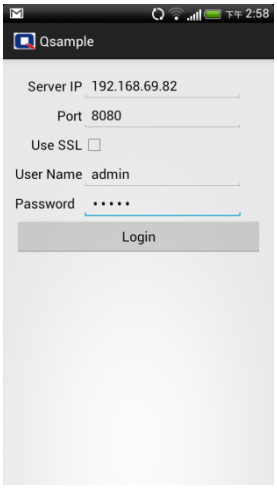

Fig. 1

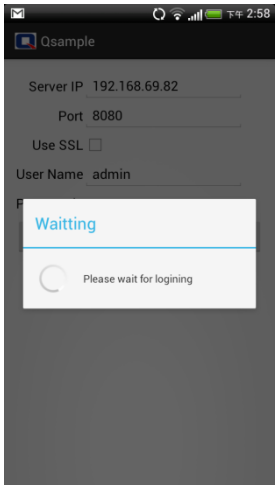

Fig. 2

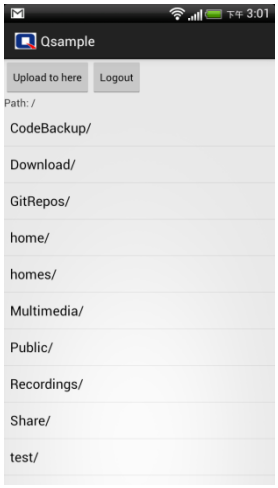

Fig. 3

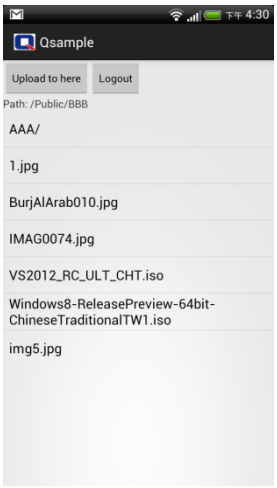

Fig. 4

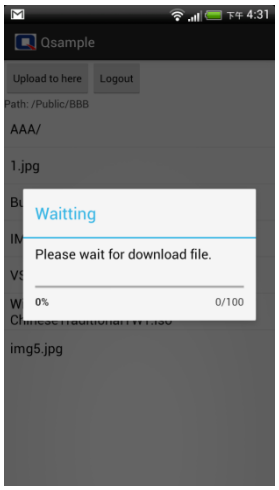

Fig. 5

Fig. 6

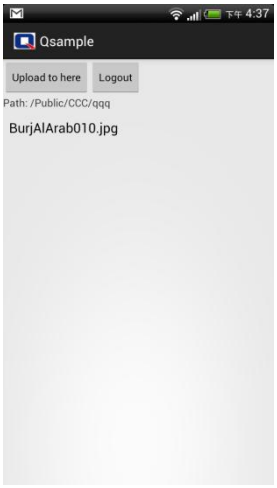

Fig. 7

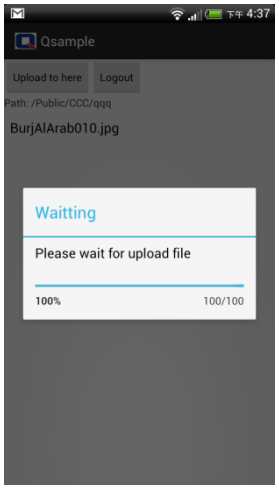

Fig. 8

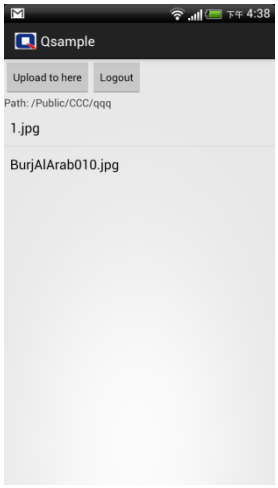

Fig. 9

QDK and App

The sample code includes two parts, QDK and App.

QDK

QDK provides the basic function of QNAP NAS. It is easy to expand function because it is high modular. In this sample code we provide six File Station's functions.

Package: con.qnap.qdk.qtshttp

Class/Interfance/Enum	Description
QtsHttpServer	It represent QNAP NAS and it be used to create File Station
QtsHttpServerInfo	Save server information
QtsHttpSession	Keeps File Station connecting information
QtsHttpStationType	List station type
QtsHttpConnection	Be used to execute URL
QtsHttpResponse	Be used to return URL response content and response code
QtsHttpCancelController	Cancel controller
IQtsHttpTransferedProgressListener	Interface of transferred progress listener

Package: com.qnap.qdk.filestation

Class/Interfance/Enum	Description
IQtsHttpFileStation	Provide several File station functions
QtsHttpFileStation	Implement File Station functions
QtsHttpFileEntry	Be represented a file/folder
QtsHttpFileStationApiVersion	List file sation api version
QtsHttpFileStationStatus	File station status

Package: com.qnap.qdk.exception

Exception
QtsHttpException
QtsHttpAuthorizationFailedException
QtsHttpFileExistException
QtsHttpFileNotExistException
QtsHttpNetworkConnectionFailedException

QtsHttpNetworkNoConnectionExceptioin
QtsHttpNewtorkTimeoutException
QtsHttpNoAuthorizedException
QtsHttpParameterInvalidException
QtsHttpPermissionDeniedException
QtsHttpServerDisconnectException
QtsHttpServerNotExistException
QtsHttpStationNotEnabledException
QtsHttpSystemOutOfMemoryException

Package: com.qnap.qdk.parser

Class/Interfance/Enum	**Description**
JsonParser	Json parser

Package: com.qnap.qdk.util

Class/Interfance/Enum	**Description**
StringTranslator	Support encode password and URL preprocess functions

App

App includes UI (User interface) and behavior control.

Package: com.qnap.qsample

Class/Interfance/Enum	**Description**
MainActivity	Login page
EntryList	Display NAS file list and provide file uploading/downloading
LocalList	Display local file list in SD card's Download folder

Functions

Login to File Station

1. Create **serverInfo** object and set server information like host name, user name, password, using SSL conection, and computer name.

```
String serverIP = "192.168.0.1";
String userName = "admin";
String password = "admin";
String agentName = "QtsHttp";
String computerName = "PC";
boolean isSSL = false;
SSLQtsHttpServerInfo serverInfo = new QtsHttpServerInfo(serverIP,
userName, password, isSSL, computerName);
```

2. Create a **server** object based on ***QtsHttpServer*** using **serverInfo** object.

```
QtsHttpServer server = new QtsHttpServer(serverInfo);
```

3. Setup connection Non-SSL/SSL port.

```
server.setFileStationSSLPortNum(443);
server.setFileStationPortNum(8080);
```

4. Use **server**.**login()** method to login File Station.

```
server.login(QtsHttpStationType.QTS_HTTP_STATION_TYPE_FILE_STATION);
```

5. Use **server**.**openFileStation()** method to get **fileStation** object based on ***IQtsHttpFileStation***.

```
public static IQtsHttpFileStation fileStation;
fileStation = server.openFileStation();
```

6. Now you login into File Station and get **fileStation** object.

Get share folder list

In QNAP Turbo NAS, the root folders are called “Share folder”. If you want to get the share folder list, use **fileStation**.**getShareFolderList()** method to get the share folder list. It will return **ArrayList<QtsHttpFileEntry>**, and you can get type (folder/file), name, path, and file size from each **QtsHttpfileEntry**.

1. Create cancel object based on QtsHttpCancelController.

```
QtsHttpCancelController cancel = new QtsHttpCancelController();
```

2. Get **ArrayList<QtsHttpFileEntry>** and use for loop to get each **QtsHttpFileEntry**.

```
ArrayList<QtsHttpFileEntry> fileEntryList =
fileStation.getShareFolderList(cancel);
for(QtsHttpFileEntry fileEntry : fileEntryList) {
    String fileName = fileEntry.getFileName();
    String filePath = fileEntry.getFilePath();
    boolean isDir = fileEntry.isDir();
}
```

3. If you want to cancel getting share folder list, you can set cancel.setCancel().

```
cancel.setCancel();
```

Get forder and file list

If you want to get the file list in folder, use **fileStation**.**getFileList()** method to get the file list. It will return **ArrayList<QtsHttpFileEntry>**, and you can get type (folder/file), name, path, and file size from each **QtsHttpfileEntry**.

1. Create cancel object based on QtsHttpCancelController.

```
QtsHttpCancelController cancel = new QtsHttpCancelController();
```

2. Set parent path, index of ordering file list, number of getting file entry

```
String parentPath = "/Public/";
int index = 0;
int limit = 50;
```

3. Get **ArrayList<QtsHttpFileEntry>** and use for loop to get each **QtsHttpFileEntry**.

```
ArrayList<QtsHttpFileEntry> fileEntryList =
fileStation.getFileList(parentPath, index, limit, cancel);
for(QtsHttpFileEntry fileEntry : fileEntryList) {
    String fileName = fileEntry.getFileName();
    String filePath = fileEntry.getFilePath();
    boolean isDir = fileEntry.isDir();
}
```

Download file

If you want to download file from NAS, you need to get source file path (from NAS) and destination file paths (to your device). You also need to implement your **TransferedProgressListener** class implementing from **IQtsHttpTransferedProgressListener**. So you can show the progress on you app.

1. Implement your **TransferedProgressListener** class implementing from **IQtsHttpTransFeredProgressListener**. You need to override onProgress() method.

```
public class TransferedProgressListener implements
IQtsHttpTransferedProgressListener {
    private long mTotalFileLengthInBytes;
    public TransferedProgressListener() {
        mTotalFileLengthInBytes = 0;
    }

    @Override
    public void onProgress(long transferedFileLengthInBytes) {
        if (mTotalFileLengthInBytes > 0) {
            long percentage =
Math.round(((double)(transferedFileLengthInBytes * 100) /
mTotalFileLengthInBytes));
            Message msg = new Message();
            Bundle data = msg.getData();
            data.putLong("percent",  percentage);
            msg.what = sMsgUpdateProgress;
            msg.setData(data);
            uiMessageHandler.sendMessage(msg);
        }
    }

    public void setTotalFileLengthInBytes(long
totalFileLengthInBytes) {
        this.mTotalFileLengthInBytes = totalFileLengthInBytes;
    }
}
```

2. Setup source and destination file path. Create cancel controller and transferred progress listener.

```
String downloadSrcFilePath = “/Public/test.txt”;
String downloadDestPath = “/mnt/sdcard/Download/”;
```

```
QtsHttpCancelController cancel = new QtsHttpCancelController();
TransferedProgressListener progListener = new TransferedProgressListener();
```

3. Download file.

```
fileStation.downloadFileByPath(downloadSrcFilePath,
downloadDestPath, 0, cancel, progListener);
```

Upload file

If you want to upload file to NAS, it is similar download file. You need to provide source file path (from your device) and destination file path (to NAS). And you also to implement your **TransferedProgressListener** class like download file or you can use same **TransferedProgressListener** from download file.

1. Implement your **TransferedProgressListener** class implementing from **IQtsHttpTransFeredProgressListener**. You need to override onProgress() method.
2. Setup source and destination file path. Create cancel controller and transferred progress listener.

```
String uploadSrcFilePath = "/mnt/sdcard/Download/photo.jpg";
String uploadDestPath = "/mnt/sdcard/Download/";
QtsHttpCancelController cancel = new QtsHttpCancelController();
TransferedProgressListener progListener = new TransferedProgressListener();
```

3. Upload file.

```
fileStation.uploadFileByPath(uploadSrcFilePath, uploadDestPath,
cancel, progListener);
```

Close file station

1. Using **server** object to close file station first.

```
server.closeFileStation(fileStation);
fileStation = null;
server = null;
```

Expand functions

This sample code just implement six function includes login(), logout(), getShareFolderList(), getFileList(), downloadFile(), and uploadFile(). You can get File management API document (QNAP File Station Web API document) for QNAP Developer Center website: http://www.qnap.com/dev/en/
Go to Resource, and download **Development tooklkit (API & SDK) – File management API document**.
You can read this document and implement other functions.