知途APP1.0 核心源码 最后部分

HttpService.kt

```
package com.bird2fish.travelbook.core

import android.app.Service
import android.content.Intent
import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.os.Binder
import android.os.IBinder
import android.util.Log
import com.bird2fish.travelbook.helper.LogHelper
import com.bird2fish.travelbook.helper.PreferencesHelper
import com.bird2fish.travelbook.ui.contact.Friend
import com.bird2fish.travelbook.ui.data.model.CurrentUser
import com.bird2fish.travelbook.ui.data.model.LoggedInUser
import com.google.gson.Gson
import com.google.gson.GsonBuilder
import okhttp3.MediaType.Companion.toMediaType
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.MultipartBody
import okhttp3.Request
import okhttp3.RequestBody
import okhttp3.RequestBody.Companion.asRequestBody
import okhttp3.RequestBody.Companion.toRequestBody
import org.json.JSONArray
import org.json.JSONObject
import java.io.ByteArrayOutputStream
import java.io.File
import java.security.SecureRandom
import java.security.cert.X509Certificate
import java.util.*
import javax.net.ssl.SSLContext
import javax.net.ssl.SSLSocketFactory
import javax.net.ssl.TrustManager
import javax.net.ssl.X509TrustManager


class HttpService : Service() {

    private var host: String = "" //""127.0.0.1:7787"
    private var schema: String = "http"


    fun setSchema(mode: String) {
        this.schema = mode
    }

    fun setServer(url: String) {
```

```
        this.host = url
    }

    fun initServer() {
        host = PreferencesHelper.getHostName()
        schema = PreferencesHelper.getHostSchema()
    }

    // 获取此类的引用
    inner class HttpBinder : Binder() {
        fun getService(): HttpService? {
            return this@HttpService
        }
    }

    //通过binder实现调用者client与Service之间的通信
    private val binder = HttpBinder()

    //private val generator: Random = Random()

    override fun onCreate() {
        Log.i("HttpService", "HttpService -> onCreate, Thread: " +
Thread.currentThread().name)
        super.onCreate()
        host = PreferencesHelper.getHostName()
        schema = PreferencesHelper.getHostSchema()
    }

    override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
        Log.i(
            "HttpService",
            "HttpService -> onStartCommand, startId: " + startId + ", Thread: " +
Thread.currentThread().name
        )
        return START_NOT_STICKY
    }

    override fun onBind(intent: Intent?): IBinder? {
        Log.i("HttpService", "HttpService -> onBind, Thread: " +
Thread.currentThread().name)
        return binder
    }

    override fun onUnbind(intent: Intent): Boolean {
        Log.i("HttpService", "HttpService -> onUnbind, from:" +
intent.getStringExtra("from"))
        return false
    }

    override fun onDestroy() {
        Log.i("DemoLog", "TestService -> onDestroy, Thread: " +
Thread.currentThread().name)
        super.onDestroy()
```

```
    }

    //getRandomNumber是Service暴露出去供client调用的公共方法
//    fun getRandomNumber(): Int {
//        return generator.nextInt()
//    }

    val unsafeSslSocketFactory: SSLSocketFactory
        get() = try {
            val trustAllCerts = arrayOf<TrustManager>(
                object : X509TrustManager {
                    override fun checkClientTrusted(
                        chain: Array<X509Certificate>,
                        authType: String
                    ) {
                    }

                    override fun checkServerTrusted(
                        chain: Array<X509Certificate>,
                        authType: String
                    ) {
                    }

                    override fun getAcceptedIssuers(): Array<X509Certificate> {
                        return arrayOf()
                    }
                }
            )
            val sslContext = SSLContext.getInstance("SSL")
            sslContext.init(null, trustAllCerts, SecureRandom())
            sslContext.socketFactory
        } catch (e: java.lang.Exception) {
            throw RuntimeException(e)
        }


    // https://blog.csdn.net/yzpbright/article/details/115333339
    // 注册
    // http://localhost:7817/v1/user/regist?type=1&pwd=111111&username=robin
    fun register(name: String, pwd: String): LoggedInUser {

        val fakeUser = LoggedInUser("", "", "", "0", "")
        //构建url地址
        var url = "${schema}://${host}/v1/user/regist?type=1&pwd=${pwd}&username=${name}"

        try {
            var client = HttpsUtil.getClient(null)


            val request = Request.Builder()
                .url(url)
                .get() //以post的形式添加requestBody
                .build()
```

```
            var response = client.newCall(request).execute()
            val responseData = response.body?.string()
            if (responseData != null) {
                val jsonObject = JSONObject(responseData)
                val state = jsonObject.getString("state")
                //LogHelper.d(" upload date response $state")
                if (state == "ok") {
                    val user = jsonObject.getJSONObject("user")
                    if (user != null) {
                        fakeUser.uid = user.getLong("id").toString()
                        fakeUser.userId = user.getString("name")
                        fakeUser.pwd = pwd
                        fakeUser.sid = ""
                    }
                }
            }
        } catch (e: Exception) {
            //LogHelper.e("Exception")
            //LogHelper.e("$e.message")
        }
        return fakeUser
    }

    // sid登录
    //http://localhost:7817/v1/user/login?uid=1005&sid=6812630841045951575&type=0
    fun loginWithSid(uid: String, sid: String, pwd: String): LoggedInUser {
        val fakeUser = LoggedInUser("", "", "", "0", "")
        //构建url地址
        var url = "${schema}://${host}/v1/user/login?type=0&uid=${uid}&sid=${sid}"
        try {
            var client = HttpsUtil.getClient(null)
            val request = Request.Builder()
                .url(url)
                .get() //以post的形式添加requestBody
                .build()
            var response = client.newCall(request).execute()
            val responseData = response.body?.string()
            if (responseData != null) {
                val jsonObject = JSONObject(responseData)
                val state = jsonObject.getString("state")
                //LogHelper.d(" upload date response $state")
                if (state == "ok") {
                    val user = jsonObject.getJSONObject("session")
                    if (user != null) {
                        fakeUser.uid = user.getLong("id").toString()
                        fakeUser.sid = user.getLong("sid").toString()
                        fakeUser.pwd = pwd
                    }
                } else {
                    PreferencesHelper.delUserSid()
                }
            }
```

```
            // 直接获取信息
            val userInfo = getUserInfo(fakeUser.uid, fakeUser.sid, fakeUser.uid)
            if (userInfo.uid == fakeUser.uid) {
                userInfo.sid = fakeUser.sid
                userInfo.pwd = fakeUser.pwd
                return userInfo
            }
        } catch (e: Exception) {
            PreferencesHelper.delUserSid()
            //LogHelper.e("Exception")
            LogHelper.e("$e.message")
        }

        return fakeUser
    }

    // 用户名密码登录
    // http://localhost:7817/v1/user/login?uid=1005&pwd=111111&type=1
    fun loginWithpwd(uid: String, pwd: String): LoggedInUser {
        val fakeUser = LoggedInUser("", "", "", "0", "")
        //构建url地址
        var url1 = //"http://192.168.1.2:7817/v1/user/login?uid=1005&pwd=123456&type=1"
            "${schema}://${host}/v1/user/login?type=1&pwd=${pwd}&uid=${uid}"

        try {
            var client = HttpsUtil.getClient(null)
            val request = Request.Builder()
                .url(url1)  // "http://www.baidu.com"
                .get() //以post的形式添加requestBody
                .build()
            var response = client.newCall(request).execute()
            val responseData = response.body?.string()
            val jsonObject = JSONObject(responseData)
            val state = jsonObject.getString("state")

            if (state == "ok") {
                val user = jsonObject.getJSONObject("session")
                if (user != null) {
                    fakeUser.uid = user.getLong("id").toString()
                    fakeUser.sid = user.getLong("sid").toString()
                    fakeUser.pwd = pwd
                }
            }

            // 直接获取信息
            val userInfo = getUserInfo(fakeUser.uid, fakeUser.sid, fakeUser.uid)
            if (userInfo.uid == fakeUser.uid) {
                userInfo.sid = fakeUser.sid
                userInfo.pwd = fakeUser.pwd
                return userInfo
            }
        } catch (e: Exception) {
            //LogHelper.e("Exception")
```

```
            e.printStackTrace();
            LogHelper.e("$e.message")
        }


        return fakeUser
    }

    //返回数据
    //    {
    //        "state": "ok",
    //        "user": {
    //        "id": 1005,
    //        "phone": "0",
    //        "icon": "sys:icon/1.jpg",
    //        "name": "robin",
    //        "nick": "robin",
    //        "email": "momo@local.com",
    //        "pwd": "",
    //        "temppwd": "",
    //        "region": "unknow",
    //        "ipv4": "0.0.0.0",
    //        "wxid": "0",
    //        "age": 1,
    //        "gender": 1,
    //        "tm": "2023/09/20 16:04:18",
    //        "sid": 0
    //    }
    //    }
    // 获取某个用户的基本信息
    fun getUserInfo(uid: String, sid: String, fid: String): LoggedInUser {
        val fakeUser = LoggedInUser("", "", "", "0", "")
        //构建url地址
        var url1 = "${schema}://${host}/v1/user/searchfriends?
fid=${fid}&uid=${uid}&sid=${sid}"

        try {
            var client = HttpsUtil.getClient(null)
            val request = Request.Builder()
                .url(url1)
                .get()
                .build()
            var response = client.newCall(request).execute()
            val responseData = response.body?.string()
            val jsonObject = JSONObject(responseData)
            val state = jsonObject.getString("state")

            if (state == "ok") {
                val user = jsonObject.getJSONObject("user")
                if (user != null) {
                    fakeUser.uid = user.getLong("id").toString()
                    //fakeUser.sid = user.getLong("sid").toString()
                    fakeUser.nickName = user.getString("nick")
```

```
                fakeUser.userId = user.getString("name")
                fakeUser.icon = user.getString("icon")

                if (user.getInt("age") != null) {
                    fakeUser.age = user.getInt("age").toString()
                } else {
                    fakeUser.age = "0"
                }


                fakeUser.phone = user.getString("phone")
                fakeUser.email = user.getString("email")

                fakeUser.ip = user.getString("ipv4")
                fakeUser.region = user.getString("region")
                if (user.getInt("gender") == 1) {
                    fakeUser.gender = "男"
                } else {
                    fakeUser.gender = "女"
                }
            }
        }
    } catch (e: Exception) {
        //LogHelper.e("Exception")
        e.printStackTrace();
        LogHelper.e("$e.message")
    }

    return fakeUser
}

// 解析返回的粉丝数据
private fun parseFollower(node: JSONArray): LinkedList<Friend> {
    var lst = LinkedList<Friend>()
    for (i in 0 until node.length()) {
        try {
            val user = node.getJSONObject(i);
            var fakeUser = Friend()
            fakeUser.uid = user.getString("id")
            fakeUser.icon = user.getString("icon")
            fakeUser.nick = user.getString("alias")
            val mask = user.getInt("visible")
            fakeUser.show = (mask and 8) != 0
            fakeUser.isFriend = true

            lst.add(fakeUser)
        } catch (e: Exception) {

        }
    }
    return lst
}
```

```
// 查询自己关注列表
/*
    {
    "state": "ok",
    "count": 1,
    "list": [
        {
            "id": 1004,
            "icon": "sys:icon/1.jpg",
            "ctm": 1695200220490,
            "alias": "robin",
            "label": "",
            "visible": 7,
            "block": false
        }
    ]
}
 */
fun getFollowList(uid: String, sid: String): LinkedList<Friend> {
    //构建url地址
    var url1 = "${schema}://${host}/v1/user/listfriends?type=1&&uid=${uid}&sid=${sid}"

    try {
        var client = HttpsUtil.getClient(null)
        val request = Request.Builder()
            .url(url1)
            .get()
            .build()
        var response = client.newCall(request).execute()
        val responseData = response.body?.string()
        val jsonObject = JSONObject(responseData)
        val state = jsonObject.getString("state")

        if (state == "ok") {
            val data = jsonObject.getJSONArray("list")
            var lst = parseFollower(data)
            return lst
        }
    } catch (e: Exception) {
        //LogHelper.e("Exception")
        e.printStackTrace();
        LogHelper.e("$e.message")
    }

    return LinkedList<Friend>()
}

fun getFollowList(): LinkedList<Friend> {
    val user = CurrentUser.getUser()
    if (user != null) {
        return getFollowList(user.uid, user.sid)
    } else {
        return LinkedList<Friend>()
```

```
        }

    }

    // 搜索人员添加好友
    fun searchForFriend(fid: String): LinkedList<Friend> {
        val user = CurrentUser.getUser()
        if (user != null) {

            val tempUser = getUserInfo(user.uid, user.sid, fid)
            if (tempUser.uid == fid) {
                var friend = Friend()
                friend.fromUser(tempUser)
                var lst = LinkedList<Friend>()
                lst.add(friend)
                return lst
            }
        }
        return LinkedList<Friend>()

    }

    // 设置关注的好友，是否显示等信息，备注等；取消关注好友等
    fun setFollowUserInfo(friend: Friend): Boolean {

        val user = CurrentUser.getUser()
        if (user == null)
            return false
        var str = "show"
        if (friend.show == false) {
            str = "ignore"
        }

        var url1 =
            "${schema}://${host}/v1/user/setfriendinfo?
uid=${user.uid}&sid=${user.sid}&fid=${friend.uid}&param=${str}"

        try {
            var client = HttpsUtil.getClient(null)
            val request = Request.Builder()
                .url(url1)
                .get()
                .build()
            var response = client.newCall(request).execute()
            val responseData = response.body?.string()
            val jsonObject = JSONObject(responseData)
            val state = jsonObject.getString("state")

            if (state == "ok") {

                return true
            }
        } catch (e: Exception) {
```

```
            //LogHelper.e("Exception")
            e.printStackTrace();
            LogHelper.e("$e.message")
        }
        return false
    }

    fun removeFriend(friend: Friend): Boolean {
        val user = CurrentUser.getUser()
        if (user == null)
            return false
        var url1 =
            "${schema}://${host}/v1/user/setfriendinfo?
uid=${user.uid}&sid=${user.sid}&fid=${friend.uid}&param=remove"

        try {
            var client = HttpsUtil.getClient(null)
            val request = Request.Builder()
                .url(url1)
                .get()
                .build()
            var response = client.newCall(request).execute()
            val responseData = response.body?.string()
            val jsonObject = JSONObject(responseData)
            val state = jsonObject.getString("state")

            if (state == "ok") {

                return true
            }
        } catch (e: Exception) {
            //LogHelper.e("Exception")
            e.printStackTrace();
            LogHelper.e("$e.message")
        }
        return false

    }

    // 设置关注某人
    // http://localhost:7817/v1/user/addfriendreq?uid=1005&sid=6812630841045951575&fid=1004
    /*
    {
        "state": "ok",
        "detail": "add friend ok"
    }
     */
    fun setFollowHim(friend: Friend): Boolean {
        //构建url地址
        val user = CurrentUser.getUser()
        if (user == null)
            return false
        var url1 =
```

```
            "${schema}://${host}/v1/user/addfriendreq?
uid=${user.uid}&sid=${user.sid}&fid=${friend.uid}"

        try {
            var client = HttpsUtil.getClient(null)
            val request = Request.Builder()
                .url(url1)
                .get()
                .build()
            var response = client.newCall(request).execute()
            val responseData = response.body?.string()
            val jsonObject = JSONObject(responseData)
            val state = jsonObject.getString("state")

            if (state == "ok") {

                return true
            }
        } catch (e: Exception) {
            //LogHelper.e("Exception")
            e.printStackTrace();
            LogHelper.e("$e.message")
        }
        return false
    }

    // 查询好友信息
    /*
    {
        "id": 1005,
        "phone": "",
        "icon": "sys:icon/2.jpg",
        "name": "robin",
        "nick": "飞鸟",
        "email": "",
        "pwd": "123456",
        "temppwd": "",
        "region": "unknow",
        "ipv4": "0.0.0.0",
        "wxid": "0",
        "age": 42,
        "gender": 1,
        "tm": "2023/09/20 16:04:18",
        "sid": 6812630841045951575
    }
     */

    fun updateInfo(user: LoggedInUser): Boolean {

        var jsonObject = JSONObject()

        try {
            val intNumber = user.uid.toInt()
```

```
            jsonObject.put("id", intNumber)

        } catch (e: NumberFormatException) {
            return false
        }

        try {
            val intNumber = user.sid.toLong()
            jsonObject.put("session", intNumber)

        } catch (e: NumberFormatException) {
            return false
        }



        jsonObject.put("pwd", user.pwd)
        jsonObject.put("icon", user.icon)
        jsonObject.put("nick", user.nickName)
        jsonObject.put("name", user.userId)


        try {
            val intNumber = user.age.toInt()
            println("转换后的整数: $intNumber")
            jsonObject.put("age", intNumber)
        } catch (e: NumberFormatException) {
            println("无法将字符串转换为整数: $e")
            jsonObject.put("age", 0)
        }

        if (user.gender == "男") {
            jsonObject.put("gender", 1)
        } else {
            jsonObject.put("gender", 0)
        }

        jsonObject.put("phone", user.phone)
        jsonObject.put("email", user.email)

        //System.out.println(tmStr)
        //jsonObject.put("tmStr", tmStr)
        var jsonStr = jsonObject.toString()
        //System.out.println(jsonStr)

        //调用请求
        val contentType = "application/json".toMediaType()
        var requestBody = jsonStr.toRequestBody(contentType)

        val uid = user.uid
        val sid = user.sid

        //构建url地址
```

```
    var url1 = "${schema}://${host}/v1/user/setbaseinfo?&uid=${uid}&sid=${sid}"

    try {
        var client = HttpsUtil.getClient(null)
        val request = Request.Builder()
            .url(url1)
            .post(requestBody)
            .build()
        var response = client.newCall(request).execute()
        val responseData = response.body?.string()
        val jsonObject = JSONObject(responseData)
        val state = jsonObject.getString("state")

        if (state == "ok") {

            return true
        }
    } catch (e: Exception) {
        //LogHelper.e("Exception")
        e.printStackTrace();
        LogHelper.e("$e.message")
    }
    return false
}

////////////////////////////////////////////////////////////////////////

fun parseNews(data: JSONArray): LinkedList<News> {
    val newsList = LinkedList<News>()

    for (i in 0 until data.length()) {
        val newsObject: JSONObject = data.getJSONObject(i)

        val nid = newsObject.getString("nid")
        val uid = newsObject.getString("uid")
        val nick = newsObject.getString("nick")
        val icon = newsObject.getString("icon")
        val lat = newsObject.getDouble("lat")
        val log = newsObject.getDouble("log")
        val alt = newsObject.getDouble("alt")
        val tm = newsObject.getLong("tm")
        val title = newsObject.getString("title")
        val content = newsObject.getString("content")

        val imagesArray = newsObject.getJSONArray("images")
        val imagesList = mutableListOf<String>()
        for (j in 0 until imagesArray.length()) {
            val imageUrl = imagesArray.getString(j)
            imagesList.add(imageUrl)
        }

        val tagsArray = newsObject.getJSONArray("tags")
        val tagsList = mutableListOf<String>()
```

```
            for (j in 0 until tagsArray.length()) {
                val tag = tagsArray.getString(j)
                tagsList.add(tag)
            }

            val type = newsObject.getString("type")
            val trackFile = newsObject.getString("trackfile")
            val likes = newsObject.getInt("likes")
            val favs = newsObject.getInt("favs")
            val deleted = newsObject.getBoolean("deleted")
            val delTm = newsObject.getLong("deltm")

            val news = News(
                nid, uid, nick, icon, lat, log, alt, tm, title, content,
                imagesList, tagsList, type, trackFile, likes, favs, deleted, delTm
            )

            newsList.add(news)
        }

        return newsList
    }

    fun removeNews(uid: String, sid: String, nid:String):Boolean{
        //构建url地址
        var url1 =  "${schema}://${host}/v1/news/delete?&uid=${uid}&sid=${sid}&nid=${nid}"
        try {
            var client = HttpsUtil.getClient(null)
            val request = Request.Builder()
                .url(url1)
                .get()
                .build()
            var response = client.newCall(request).execute()
            val responseData = response.body?.string()

            System.out.println(responseData)
            val jsonObject = JSONObject(responseData)
            val state = jsonObject.getString("state")

            if (state.equals("ok", true)) {
                return true
            }
        } catch (e: Exception) {
            //LogHelper.e("Exception")
            e.printStackTrace();
            LogHelper.e("$e.message")
        }

        return false
    }

    fun getNewsRecent(uid: String, sid: String, page: Int, pagesize: Int): LinkedList<News>
{
```

```
        //构建url地址
        var url1 =
            "${schema}://${host}/v1/news/recent?
&uid=${uid}&sid=${sid}&page=${page}&size=${pagesize}"

        try {
            var client = HttpsUtil.getClient(null)
            val request = Request.Builder()
                .url(url1)
                .get()
                .build()
            var response = client.newCall(request).execute()
            val responseData = response.body?.string()

            System.out.println(responseData)
            val jsonObject = JSONObject(responseData)
            val state = jsonObject.getString("state")

            if (state == "ok") {
                val data = jsonObject.getJSONArray("data")
                var lst = parseNews(data)
                return lst
            }
        } catch (e: Exception) {
            //LogHelper.e("Exception")
            e.printStackTrace();
            LogHelper.e("$e.message")
        }

        return LinkedList<News>()
    }


    fun convertNewsToJson(news: News): String {
        val gson = Gson()
        return gson.toJson(news)
    }

    fun postNews(news: News, uid: String, sid: String): Boolean {

        val jsonStr = convertNewsToJson(news)
        //调用请求
        val contentType = "application/json".toMediaType()
        var requestBody = jsonStr.toRequestBody(contentType)

        //构建url地址
        var url1 = "${schema}://${host}/v1/news/publish?&uid=${uid}&sid=${sid}"

        try {
            var client = HttpsUtil.getClient(null)
            val request = Request.Builder()
                .url(url1)
                .post(requestBody)
```

```
                .build()
            var response = client.newCall(request).execute()
            val responseData = response.body?.string()
            val jsonObject = JSONObject(responseData)
            val state = jsonObject.getString("state")

            if (state == "ok") {

                return true
            }
        } catch (e: Exception) {
            //LogHelper.e("Exception")
            e.printStackTrace();
            LogHelper.e("$e.message")
        }
        return false
    }

    fun getImageUrl(fileName: String): String {
        val url = "${schema}://${host}/file/${fileName}"
        return url
    }

    fun uploadAndScaleImageFile(imageFilePath: String) :String {

        val imageFile = File(imageFilePath)
        // 获取原始图片的尺寸信息
        val options = BitmapFactory.Options()
        options.inJustDecodeBounds = true
        System.out.println(imageFile.path)
        BitmapFactory.decodeFile(imageFile.path, options)
        if (options.outHeight == 0){
            return ""
        }


        // 计算缩放比例
        options.inSampleSize = calculateInSampleSizeByWidth(options, 800)

        // 加载并返回缩放后的图片
        options.inJustDecodeBounds = false
        val scaledBitmap = BitmapFactory.decodeFile(imageFile.path, options)
        if (scaledBitmap == null){
            return ""
        }

        // 将 Bitmap 转换为字节数组
        val byteArrayOutputStream = ByteArrayOutputStream()
        scaledBitmap.compress(Bitmap.CompressFormat.JPEG, 100, byteArrayOutputStream)
        val byteArray = byteArrayOutputStream.toByteArray()

        // 构建 Multipart 请求体
        val mediaType = "image/*".toMediaTypeOrNull()
```

```
        val requestBody: RequestBody = MultipartBody.Builder()
            .setType(MultipartBody.FORM)
            .addFormDataPart("file", "image.jpg",
                byteArray.toRequestBody(mediaType, 0,byteArray.size)
            )
            .build()

        // 构建请求
        val url = "${schema}://${host}/uploadfile"
        val request: Request = Request.Builder()
            .url(url)
            .post(requestBody)
            .build()

        try {
            var client = HttpsUtil.getClient(null)
            var response = client.newCall(request).execute()
            val responseData = response.body?.string()
            val jsonObject = JSONObject(responseData)
            val state = jsonObject.getString("state")

            if (state.equals("ok", ignoreCase = true)) {

                val filename = jsonObject.getString("newname")

                return filename
            }
        } catch (e: Exception) {
            //LogHelper.e("Exception")
            e.printStackTrace();
            LogHelper.e("$e.message")
        }

        return ""
    }

//    private fun asytest(){
//        var client = HttpsUtil.getClient()
//        client.newCall(request).enqueue(object : Callback {
//            override fun onFailure(call: Call, e: IOException) {
//                // 上传失败的处理
//                e.printStackTrace()
//            }
//
//            override fun onResponse(call: Call, response: Response) {
//                // 上传成功的处理
//                if (response.isSuccessful) {
//                    val responseBody = response.body()?.string()
//                    // 处理服务器响应数据
//                } else {
//                    // 上传失败的处理
//                }
//            }
```

```
//        })
//    }


    private fun calculateInSampleSize(options: BitmapFactory.Options, maxSize: Long): Int {
        val height = options.outHeight
        val width = options.outWidth
        var inSampleSize = 1

        val byteCount = height * width * 4 // 每个像素占4字节

        while (byteCount / inSampleSize > maxSize) {
            inSampleSize *= 2
        }

        return inSampleSize
    }

    private fun calculateInSampleSizeByWidth(options: BitmapFactory.Options, maxWidth:
Long): Int {
        val height = options.outHeight
        val width = options.outWidth
        var inSampleSize = 1
        if (options.outHeight == 0){
            return inSampleSize
        }

        val tmp  = options.outWidth.toDouble() / maxWidth.toDouble()
        inSampleSize = tmp.toInt()

        return inSampleSize
    }

    fun compressBitmap(bitmap: Bitmap, quality: Int): ByteArray {
        val byteArrayOutputStream = ByteArrayOutputStream()
        bitmap.compress(Bitmap.CompressFormat.JPEG, quality, byteArrayOutputStream)
        return byteArrayOutputStream.toByteArray()
    }

    fun uploadTextFile(filePath: String): String {
        val textFile = File(filePath)

        // 构建请求体，指定为纯文本类型
        //val requestBody: RequestBody = RequestBody.create("text/plain".toMediaType(),
textFile)

        val mediaType = "text/plain".toMediaTypeOrNull()
        val requestBody: RequestBody = MultipartBody.Builder()
            .setType(MultipartBody.FORM)
            .addFormDataPart("file", "track.gpx", textFile.asRequestBody(mediaType))
            .build()

        val url = "${schema}://${host}/uploadfile"
```

```
        val request = Request.Builder()
            .url(url)
            .post(requestBody)
            .build()

        val client = HttpsUtil.getClient(null)

        try {
            val response = client.newCall(request).execute()
            val responseData = response.body?.string()
            val jsonObject = JSONObject(responseData)
            val state = jsonObject.getString("state")

            if (state.equals("ok", true)) {
                val filename = jsonObject.getString("newname")
                return filename
            }
        } catch (e: Exception) {
            e.printStackTrace()
            LogHelper.e("$e.message")
        }

        return ""
    }

    fun uploadImage(imageFilePath: String):String {

        val imageFile = File(imageFilePath)
        val mediaType = "image/*".toMediaTypeOrNull()
        val requestBody: RequestBody = MultipartBody.Builder()
            .setType(MultipartBody.FORM)
            .addFormDataPart("file", "image.jpg", imageFile.asRequestBody(mediaType))
            .build()

        val url = "${schema}://${host}/uploadfile"
        val request: Request = Request.Builder()
            .url(url)
            .post(requestBody)
            .build()

        var client = HttpsUtil.getClient(null)
        try {

            var response = client.newCall(request).execute()
            val responseData = response.body?.string()
            val jsonObject = JSONObject(responseData)
            val state = jsonObject.getString("state")

            if (state.equals("ok", true)) {
                val filename = jsonObject.getString("newname")
                return filename
            }
        } catch (e: Exception) {
```

```
            //LogHelper.e("Exception")
            e.printStackTrace();
            LogHelper.e("$e.message")
        }

        return ""
    }

    fun getGpxFile(filename: String):Track{
            //构建url地址
            var url1 = this.getImageUrl(filename)

            try {
                var client = HttpsUtil.getClient(null)
                val request = Request.Builder()
                    .url(url1)
                    .get()
                    .build()
                var response = client.newCall(request).execute()
                val responseData = response.body?.string()

                System.out.println(responseData)
                val gson = GsonBuilder()
                    .setDateFormat("MM/dd/yy hh:mm a")
                    .create()
                return gson.fromJson(responseData, Track::class.java)

            } catch (e: Exception) {
                //LogHelper.e("Exception")
                e.printStackTrace();
                LogHelper.e("$e.message")
            }

            return Track()
        }

    fun getUserFav(uid:String, nid:String, sid:String): NewsFav{
        var url1 = "${schema}://${host}/v1/news/getfav?&uid=${uid}&sid=${sid}&nid=${nid}"
        val newsFav = NewsFav(nid, uid, false, false, false, 0)
        try {
            var client = HttpsUtil.getClient(null)
            val request = Request.Builder()
                .url(url1)
                .get()
                .build()
            var response = client.newCall(request).execute()
            val responseData = response.body?.string()

            System.out.println(responseData)
            val jsonObject = JSONObject(responseData)
            val state = jsonObject.getString("state")

            if (state.equals("ok", true)) {
```

```
                val data = jsonObject.getJSONObject("data")
                if (data != null){
                    newsFav.fav = data.getBoolean("fav")
                    newsFav.like = data.getBoolean("like")
                    newsFav.hate = data.getBoolean("hate")
                    newsFav.reason = data.getInt("reason")
                }
            }
        } catch (e: Exception) {
            //LogHelper.e("Exception")
            e.printStackTrace();
            LogHelper.e("$e.message")
        }
        return newsFav
    }

    fun updateUserFav(uid:String, nid:String, sid:String, opt :String):Boolean{
        var url1 = "${schema}://${host}/v1/news/setfav?
&uid=${uid}&sid=${sid}&nid=${nid}&opt=${opt}"
        val newsFav = NewsFav(nid, uid, false, false, false, 0)
        try {
            var client = HttpsUtil.getClient(null)
            val request = Request.Builder()
                .url(url1)
                .get()
                .build()
            var response = client.newCall(request).execute()
            val responseData = response.body?.string()

            System.out.println(responseData)
            val jsonObject = JSONObject(responseData)
            val state = jsonObject.getString("state")

            if (state == "ok") {
                val data = jsonObject.getJSONObject("data")
                return true
            }
        } catch (e: Exception) {
            //LogHelper.e("Exception")
            e.printStackTrace();
            LogHelper.e("$e.message")
        }
        return false
    }

    fun updateUserLike(uid:String, nid:String, sid:String, opt :String):Boolean{
        var url1 = "${schema}://${host}/v1/news/setlike?
&uid=${uid}&sid=${sid}&nid=${nid}&opt=${opt}"
        val newsFav = NewsFav(nid, uid, false, false, false, 0)
        try {
            var client = HttpsUtil.getClient(null)
            val request = Request.Builder()
                .url(url1)
```

```
                .get()
                .build()
            var response = client.newCall(request).execute()
            val responseData = response.body?.string()

            System.out.println(responseData)
            val jsonObject = JSONObject(responseData)
            val state = jsonObject.getString("state")

            if (state == "ok") {
                val data = jsonObject.getJSONObject("data")
                return true
            }
        } catch (e: Exception) {
            //LogHelper.e("Exception")
            e.printStackTrace();
            LogHelper.e("$e.message")
        }
        return false
    }


}
```