有赞云开发文档

(Android)

目录

有赞云开发文档(Android)

适用有赞云AppSDK版本:6.1.0

简介

本节内容:

有赞云AppSDK是为移动端应用打造的电商交易系统, 将有赞的交易服务在APP内轻松集成.

有赞AppSDK提供两种版本: 基础版和混合版.

两者的区别:

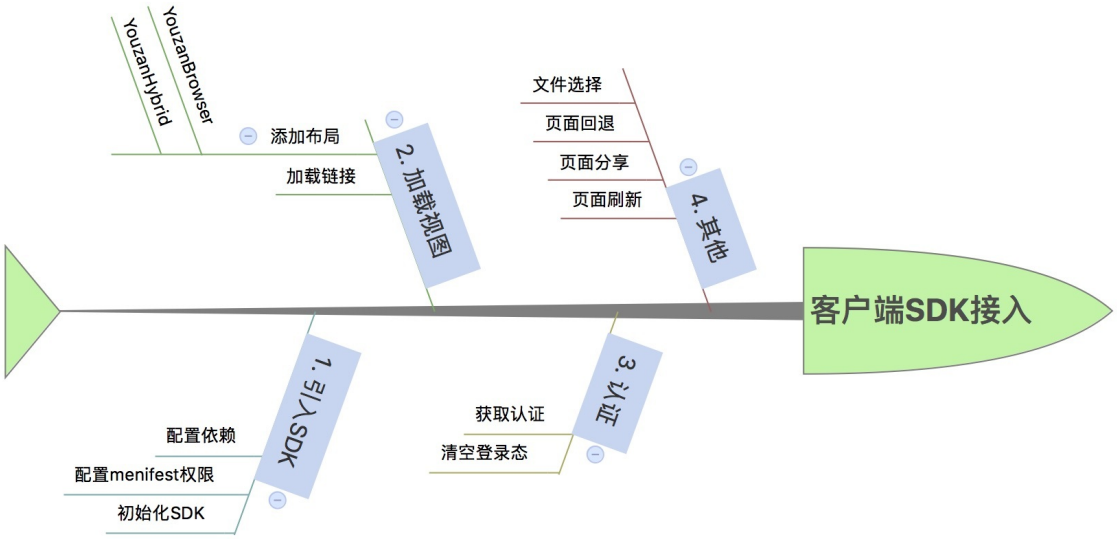
- 基础版基于webview将有赞提供的Html5页面嵌入到App;
- 混合版着力于提供原生化的页面体验, 在后续更新升级中不断优化体验(部分页面还未原生化);
- 基础版中现商品详情页面需登录可见, 混合版中无需登录即可浏览.

两者的相同点:

- 都是基于视图(View)对外提供页面加载服务, 统一的调用方式基本上更换视图类即可完成SDK切换.

可根据您实际业务中对页面的要求, 选择其中一种(也可混合使用)客户端产品接入.

接入流程



引入SDK

在项目根目录的build.gradle中声明maven仓库, 如下所示:

```
allprojects {
    repositories {
        jcenter()
        maven {url 'https://dl.bintray.com/youzanyun/maven/'}
    }
}
```

在子项目build.gradle的dependencies中引入依赖:

基础版SDK

```
compile ('com.youzanyun.open.mobile:basic:6.1.0@aar') {
    transitive = true
}
```

混合版SDK

```
compile ('com.youzanyun.open.mobile:hybrid:6.1.0@aar') {
    transitive = true
}
```

混淆规则

```
# Youzan SDK
-dontwarn com.youzan.androidsdk.***
-keep class com.youzan.androidsdk.**{*;}

# OkHttp
-dontwarn okhttp3.**
-dontwarn okio.**
-dontwarn com.squareup.okhttp.**
-keep class okio.**{*;}
-keep class com.squareup.okhttp.** { *; }
-keep interface com.squareup.okhttp.** { *; }

-dontwarn java.nio.file.*
-dontwarn javax.annotation.**
-dontwarn org.codehaus.mojo.animal_sniffer.IgnoreJRERequirement

# Image Loader
-keep class com.squareup.picasso.Picasso
-keep class com.android.volley.toolbox.Volley
-keep class com.bumptech.glide.Glide
-keep class com.nostra13.universalimageloader.core.ImageLoader
-keep class com.facebook.drawee.backends.pipeline.Fresco
```

初始化SDK

在Application或者加载有赞页面之前对SDK进行初始化, 示例如下:

```
public class MyApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        YouzanSDK.init(this, "client_id", new YouzanBasicSDKAdapter()); // 基础版适配
        // client_id即你在有赞申请的客户端标识（老用户用UA）
        // 若用混合版将第三个参数替换成 new YouzanHybridSDKAdapter();
          YouzanPreloader.preloadHtml(this, "preload_html_url");
          // 若对html预加载时调用， 提升html打开速度，preload_html_url不能是重定向页面
          // 建议对首屏html调用此方法，提升首屏体验
    }
}
```

YouzanBrowser和YouzanHybrid

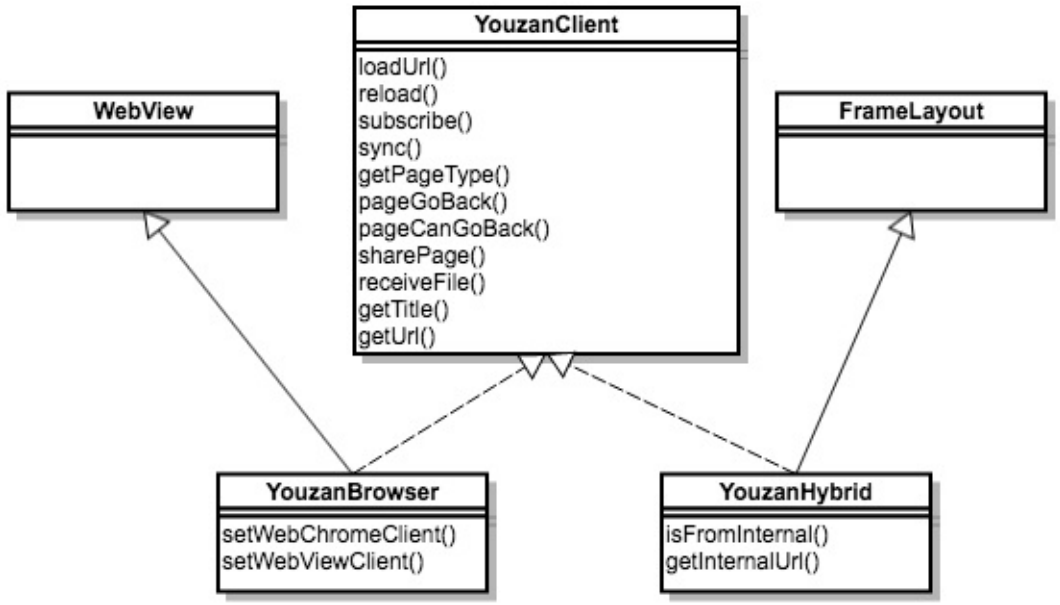
本节内容:

综述

基础版SDK提供了 `YouzanBrowser` 进行页面加载, 混合版SDK则提供了 `YouzanHybrid` 进行页面加载.

> 在调用 `YouzanBrowser` 或者 `YouzanHybrid` 进行页面加载之前都需要先初始化SDK(参考上节初始化SDK).

这两个类的结构图, 如下所示:



`YouzanBrowser` 和 `YouzanHybrid` 是自定义视图类, 开发者也可以在布局文件中使用:

- YouzanBrowser

```
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.youzan.androidsdk.basic.YouzanBrowser
        android:id="@+id/view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</FrameLayout>
```

- YouzanHybrid

```
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.youzan.androidsdk.hybrid.YouzanHybrid
        android:id="@+id/view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</FrameLayout>
```

除了上文指出的业务上的区别, 两者开发上也存在使用上的不同。

> 下文使用 `mView` 表示 `YouzanBrowser` 或者 `YouzanHybrid` 的实例
>
> 开始使用 `YouzanBrowser` 及 `YouzanHybrid` 之前一般都需要注册事件，详情看 事件 一件

自定义跳转

`YouzanBrowser` 和 `YouzanHybrid` 均可以当作一个普通的视图在布局当中使用，不管是屏幕中的一块区域还是整个界面。 当发生点击跳转时，默认将在视图所在区域内切换。但 `YouzanBrowser` 和 `YouzanHybrid` 均支持回调重写url跳转逻辑，完全可开发者自定义。两者的实现方式分别如下:

- 在 `YouzanBrowser` 自定义跳转

  `YouzanBrowser` 实质上是一个WebView，支持 `WebView` 的所有操作，重写 `shouldOverrideUrlLoading` 方法:

```
    YouzanBrowser youzanBrowser = new YouzanBrowser(this);
    youzanBrowser.setWebViewClient(new WebViewClient() {
            @Override
            public boolean shouldOverrideUrlLoading(WebView view, WebResourceRequest request) {

                // TODO: override url, return true or false

                return super.shouldOverrideUrlLoading(view, request);
```

```
            }
        });
```

- 在 `YouzanHybrid` 自定义跳转

  `YouzanHybrid` 提供 `YouzanRouterClient` ，重写 `shouldOverridePageLoading` 方法：

```
    YouzanHybrid youzanHybrid = new YouzanHybrid(this);
    youzanHybrid.setPageRouterClient(new YouzanRouterClient() {
            @Override
            public boolean shouldOverridePageLoading(Context context, PageRequest pageRequest) {
                YouzanLog.d("PageRouterClient override, url:" + pageRequest.getUrl());

                // TODO: override url
                // 返回值： true -- 代表你已处理过，YouzaHybrid不再处理；false -- YouzanHybrid处理，默认在
当前区域跳转
                return false;
            }
        });
```

YouzanClient

两者都实现了 `YouzanClient` 接口，这里对这个通用对外调用接口进行说明.

loadUrl(String url)

> 加载给定的链接

对于*YouzanBrowser*

```
final String url = "https://h5.youzan.com/xxxx";
//店铺链接，可以从有赞后台`店铺=>店铺概况=>访问店铺`复制到相应的链接

mView.loadUrl(url);
```

对于*YouzanHybrid*

```
final String url = "https://h5.youzan.com/xxxx";
//店铺链接，可以从有赞后台`店铺=>店铺概况=>访问店铺`复制到相应的链接

mView.loadUrl(url);
```

reload()

> 重新加载当前页面

说明:

可将 `YouzanBrowser` 或 `YouzanHybrid` 嵌套在SwipeRefreshLayout内实现下拉刷新.

subscribe(Event event)

> 订阅事件

说明:

基于发布订阅模式设计的事件回调, 当用户进行操作触发相应事件开发者需要实现.

现SDK内可供订阅的事件包括:

- 认证事件(必须实现)
- 分享事件
- 页面状态事件
- 文件选择事件

sync(YouzanToken token)

> 同步用户态

说明:

当认证事件被触发时, 开发者需要将认证信息拼装成 `YouzanToken` 对像通过 `sync()` 方法回传, 并在相应的页面进行刷新. SDK在获取 `YouzanToken` 后会进行本地缓存, 直到相关的认证信息失效后认证事件再次被触发再次自行同样的逻辑.

syncNot()

> 同步用户未登录状态

说明:

当认证事件被触发时, 如果用户选择不登录，则需调用 `syncNot()` 方法告知SDK。SDK内部会将当前页面重置或者回退。当 `syncNot()` 返回 `false` 时，需要外部处理。这意味着当前加载页面一定需要登录，如果不登录就不能进入，此时需要将当面界面 `finish`.

getPageType()

> 获取当前页面类型

返回值:

```
/**
 * 未知类型, 当页面处于加载中的状态时可能为该类型
 */
int PAGE_TYPE_UNKNOWN = 0x0;
/**
 * 原生:商品详情.
 */
int PAGE_TYPE_NATIVE_GOODS = 0x11;
/**
 * 原生:购物车列表.
 */
int PAGE_TYPE_NATIVE_CART = 0x12;
/**
 * 原生:支付结果.
 */
int PAGE_TYPE_NATIVE_PAY_RESULT = 0x13;
/**
```

```
     * 原生:订单列表
     */
    int PAGE_TYPE_NATIVE_TRADE_LIST = 0x14;
    /**
     * 页面主要是由{@link android.webkit.WebView}加载的html5页面构成.
     */
    int PAGE_TYPE_HTML5 = 0x01;
```

说明:

使用 `YouzanBrowser` 返回值固定为 `YouzanClient.PAGE_TYPE_HTML5` .

pageGoBack()

> 返回上一页

说明:

区别于 `WebView.goBack()` , 建议使用该方法回访上一页. 代码实践如下:

```
@Override
public void onBackPressed() {
    if (!mView.pageGoBack()) {
        super.onBackPressed();
    }
}
```

pageCanGoBack()

> 判断是否可以返回上一页

说明:

区别于 `WebView.canGoBack()` , 配合pageGoBack()方法使用.

sharePage()

> 分享页面, 触发分享事件

说明:

在页面加载完成后, 调用本方法会触发分享事件以获取当前页面的分享信息.

receiveFile(int requestCode, Intent data)

> 传入文件选择数据

说明:

当前文件选择事件被触发时, 开发者去实现具体的打开文件选择器. 如果是在新的页面打开文件选择器, 则需要在 `onActivityResult` 中回传数据. 代码实践如下:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
```

```
        if(resultCode == RESULT_OK){
            //...
            mView.receiveFile(requestCode, data);
        }
    }
```

getTitle()

> 获取当前页面的标题

说明:

在页面加载完成后(页面状态事件), 调用本方法才会返回正确数据.

getUrl()

> 获取当前页面的链接

说明:

在页面加载完成后(页面状态事件), 调用本方法才会返回正确数据.

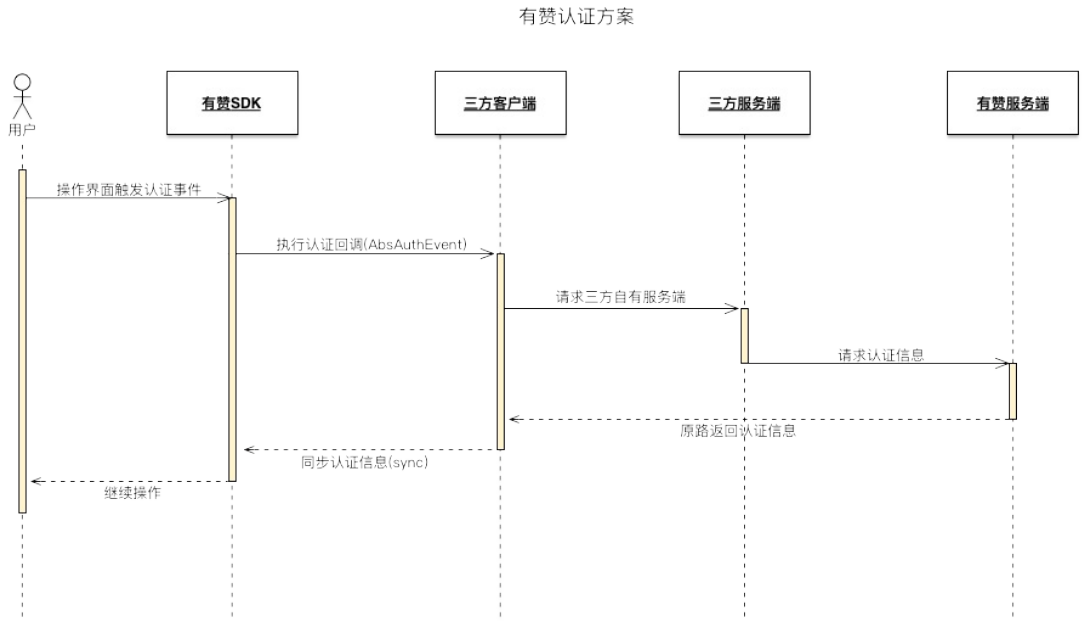
认证授权

本节内容:

综述

出于更好的安全性考虑, SDK从5.0版本开始调整了用户认证方式, 并且5.0以下版本SDK使用的用户认证方式将不再做支持.

新的认证方案如下图所示:



其中从有赞服务端返回认证信息是一个包含用户登录态信息的JSON, 是完成APP内登录态认证的必要条件.

三方服务端获取认证信息可以参考服务端接入简介

获取认证信息

需订阅认证事件(AbsAuthEvent)来监听来自SDK内的认证回调, 并编码实现请求三方自有服务端来获取认证数据.

在获取到用户信息后, 需要将数据按要求填充到 `YouzanToken` , 然后调用sync()方法回传给SDK来完成认证.

示例代码:

```
mView.sync(token);
```

清空登录态

在三方App内发生**用户切换**或**用户退出**的时, 为了避免出现客户信息混乱, 请务必执行如下代码登出有赞用户角色.

代码示例:

```
YouzanSDK.userLogout(context);
```

事件

本节内容:

- 认证事件 (必须实现)
- 分享事件
- 页面状态事件
- 文件选择事件

有赞SDK基于发布/订阅模式设计了事件回调接口, 开发者需要给视图(`YouzanBrowser` 或 `YouzanHybrid`)订阅需要的事件完成具体的业务实现.

认证事件(AbsAuthEvent)

是否必须实现: 是

代码实践(伪代码):

```
final static int REQUEST_CODE_LOGIN= 0x11;

mView.subscribe(new AbsAuthEvent() {
    /**
    * @param needLogin 标识当需要的认证信息类型.
    * needLogin为true: 需要带有用户角色的认证信息, 可以通过请求有赞服务端的登录(login)接口获取;
    * needLogin为false: 无需用户角色的认证信息, 既可通过初始化(initToken)接口获取也可通过登录(login)接口获取认证信息.
     */
    @Override
    public void call(View view, boolean needLogin) {
        //伪代码
        if(AppUserManager.isLogin()){ //判断App内的用户是否登录
            //调用login接口, 获取数据, 组装成YouzanToken, 回传给 mView.sync()
        } else if (needLogin){
            Intent intent = new Intent(YouzanActivity.this, LoginActivity.class);
            startActivityForResult(intent, REQUEST_CODE_LOGIN);
        } else {
            //调用initToken接口, 获取数据, 组装成YouzanToken, 回传给 mView.sync()
        }
    }
});

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(resultCode == RESULT_OK){
        if(requestCode == REQUEST_CODE_LOGIN){
            //调用login接口, 获取数据, 组装成YouzanToken, 回传给 mView.sync()
              //若未登录, 则回调:
            //if (!mView.syncNot()) {
            //     onBackpressed();
            //}
        }
        }
        //处理文件选择
        mView.receiveFile(requestCode, data);
    }
}
```

说明:

认证是SDK接入流程中最为重要的部分, 也是最复杂部分, 希望开发者能仔细浏览文档.

当访问的页面需要新的认证信息时(之前未认证过或者认证信息过期), 该事件会被回调.

开发者需按照代码实践中的建议实现编码(制作参考根据实际情况调整), 从服务端获取到认证信息后拼装成 `YouzanToken` 对象后调用sync()方法回传给SDK来完成整个操作.

其中, `YouzanToken` 的数据模型:

字段	说明
accessToken	有赞服务端返回的 `access_token` 字段, 用于网络接口认证
cookieKey	有赞服务端返回的 `cookie_key` 字段, 用于网页内认证
cookieValue	有赞服务端返回的 `cookie_value` 字段, 用于网页内认证

> 初始化token接口
>
> 登录接口

触发方式:

SDK内部触发.

分享事件(AbsShareEvent)

是否必须实现: 否

代码实践:

```
mView.subscribe(new AbsShareEvent() {
    @Override
    public void call(View view, GoodsShareModel data) {
       //调用系统默认的分享
        String content = data.getDesc() + " " + data.getLink();
        Intent sendIntent = new Intent();
        sendIntent.setAction(Intent.ACTION_SEND);
        sendIntent.putExtra(Intent.EXTRA_TEXT, content);
        sendIntent.putExtra(Intent.EXTRA_SUBJECT, data.getTitle());
        sendIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        sendIntent.setType("text/plain");
        startActivity(sendIntent);
    }
});
```

说明:

事件被回调后, 可获取到 `GoodsShareModel` 页面分享信息实例.

开发者可以调用Android系统默认的分享进行分享, 或者使用其他分享SDK来提高分享体验.

`GoodsShareModel` 包含的字段有:

字段	说明
title	页面标题
link	当前页面链接
desc	商品详细的描述
imgUrl	商品图片链接

触发方式:

开发者可主动调用sharePage()方法触发.

页面状态事件(AbsStateEvent)

是否必须实现: 否

代码实践:

```
mView.subscribe(new AbsStateEvent() {
    @Override
    public void call(View view) {
        //mActionBar.setTitle(mView.getTitle());
    }
});
```

说明:

事件被回调表示页面加载完成. 开发者可以基于这个时机获取页面标题等.

触发方式:

SDK内部触发.

文件选择事件(AbsChooserEvent)

是否必须实现: 否

代码实践:

```
mView.subscribe(new AbsChooserEvent() {
    @Override
    public void call(View view, Intent intent, int requestCode) throws ActivityNotFoundException {
        startActivityForResult(intent, requestCode);
    }
});

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(resultCode == RESULT_OK){
        //...
        mView.receiveFile(requestCode, data);
    }
}
```

说明:

事件被回调表示SDK内发起了一个文件选择的请求, 之后开发者需要调用receiveFile()方法回传数据.

触发方式:

如果页面内有文件的选框, 点击选框触发.


店铺歇业中，无法下单
随便买的东西
¥0.01 - 1000.00
新版
旧版
古早味
潮人
购买数量:
每人限购3件
1
图片1
仅限一张
拍照 或 选择照片
文本
输入文本
日期
时间
ID
输入18位身份证号码
数字
输入数字
邮箱
输入邮箱
凑够五个字
输入文本
多行文本
点击填写段落文本
图片2
仅限一张
拍照 或 选择照片
卖家云小风扇
¥0.01
×1
说点什么吧，你的感受对其他朋友很重要哦
总体感受
好评
中评
差评
服务打分
商品描述
卖家服务
发货速度
提交
有赞

Html页面打开加载

===

从AppSDK6.1.0版本开始，提供了 `YouzanPreloader.preloadHtml(context, url)` 方法，做H5页面预取。当在 `YouzanBrowser.loadUrl(url)` 时，会优先读取已预取的内容，这样打开H5页面的速度会更快，体验更好。

`YouzanPreloader.preloadHtml` 方法目前只针对 `h5.youzan.com` 这个域名生效，并且无法处理包含重写向的页面。所以在给出 `url` 时一定特别注意。不要对发起支付、提交订单等包含重定向页面调用 `YouzanPreloader.preloadHtml(context, url)` 。

并不建议对app中所有页面 `YouzanBrowser.loadUrl` 方法，因为毕竟这是一个流量和时间的交换，可能已经预取了这个页面，但用户实际并未打开。建议只预取H5页面首屏的 `url` 。

`YouzanPreloader.preloadHtml(context, url)` 的调用时机，建议在 `Application.onCreate()` 方法中做，也可以选择app中其他可以预测的时机，但只有确保预加载已经完成才能有效果。

另外，使用了 `YouzanPreloader.preloadHtml(context, url)` ,在页面回退栈中也会直接使用预取内容。

工具

本节内容:

WebUtil.isYouzanPage(String url)

> 判断给定的链接是否为有赞页面
>
> 返回: true表示属于有赞页面

WebUtil.isYouzanHost(String host)

> 判断给定的域名是否属于有赞
>
> 返回: true表示域名属于有赞

WebUtil.sync(Context context, YouzanToken token)

> 回传认证信息
>
> 全局调用, 可用于提前完成认证信息同步优化体验.

WebUtil.isTokenInactive(String code)

> 当访问有赞服务端API返回错误码(code), 判断该错误码是否标识当前错误是由于认证信息(token)无效造成的.

YouzanSDK.isDebug(boolean debug)

> 开启调试模式
>
> 入参debug:true标识开启

YouzanSDK.userLogout(Context context)

> 本地操作清空当前用户的信息
>
> 在三方App内发生用户切换或用户退出的时, 为了避免出现客户信息混乱, 请务必执行如下代码登出有赞用户角色.

其他

本节内容:

图片加载框架

说明:

如果使用原生SDK, 则需要引入一个图片加载框架.

原生SDK内已自动适配的图片加载框架如下:

图片加载框架	兼容版本	适配工厂类
Picasso	2.5.2	PicassoFactory
UniversalImageLoader	1.9.5	UILFactory
Glide	3.3.0+	GlideFactory
Fresco	0.8.0+	FrescoFactory
Volley	1.0.0+	VolleyFactory

> 如果项目之前没有接入过图片加载框架的话, 推荐使用Picasso

开发者只需要选择上面的一种图片加载框架在项目中导入即可, 如果在工程中添加了这份混淆规则SDK内部会自动去匹配项目中的图片加载框架, 无需开发者另做配置.

当然, 也可以手工指定有赞混合版SDK使用的图片加载框架, 示例代码如下:

```
//在展示界面前调用
YouzanHybrid.setImageLoaderFactory(new PicassoFactory());
```

独立下单弹框

是否必须实现: 否

使用场景:

开发者根据有赞云开放API自行实现了商品列表页, 需要对单个商品进行快速操作(加入购物车，立即购买), 可以使用SDK内提供的 `GoodsCardDialog` 类

代码实践:

```
new GoodsCardDialog
    .Builder(this, YouzanActivity.class)
    .setAlias(alias)
    .build(new DialogCallback() {
        @Override
        public void onCreated(Dialog dialog) {
            //获取商品购买弹框依附的dialog, 开发者可以基于此进行操作.
            dialog.show();
        }
        @Override
        public void onBuy(String url) {
            // 用mView去加载此url发起购买流程
        }

        @Override
        public void onAddCart() {

        }
    });
```

说明:

运行 `GoodsCardDialog` 之前需要调用WebUtil.sync()方法完成认证信息同步.

其中 `YouzanActivity.class` 为 `YouzanHybrid` 所依附的Activity的类对象, 并在回调的 `onCreated` 方法中获取到Dialog实例.

方法	说明	是否必需
setAlias(String)	商品别名	
setNumId(long)	商品数字编号	调用时,参数 numId 和 alias 必须选其一

效果如下:


AFU阿芙玫瑰绿白乳液100ml 增白 提亮肤色 补水保湿乳液
¥120.00
尺寸
大
中
小
购买数量:
1
加入购物车
立即购买

混淆规则

为保证SDK正常调用, 请务必在项目中混淆规则中添加如下规则:

```
# Youzan SDK
-dontwarn com.youzan.androidsdk.***
-keep class com.youzan.androidsdk.**{*;}

# OkHttp
-dontwarn okhttp3.**
-dontwarn okio.**
-dontwarn com.squareup.okhttp.**
-keep class okio.**{*;}
-keep class com.squareup.okhttp.** { *; }
-keep interface com.squareup.okhttp.** { *; }

-dontwarn java.nio.file.*
-dontwarn javax.annotation.**
-dontwarn org.codehaus.mojo.animal_sniffer.IgnoreJRERequirement

# Image Loader
-keep class com.squareup.picasso.Picasso
-keep class com.android.volley.toolbox.Volley
-keep class com.bumptech.glide.Glide
-keep class com.nostra13.universalimageloader.core.ImageLoader
-keep class com.facebook.drawee.backends.pipeline.Fresco
```

有赞云数据API

本节内容:

如果开发者想要基于有赞开放接口做自定义开发, 可以使用基础版SDK提供的API来完成.

网络调用框架

SDK内网络调用框架是基于OkHttp封装.

代码示例:

```
//请求获取alias为3nvfrf9auwizk的商品详情数据
Http.attach(this)
    .put("alias", "3nvfrf9auwizk")
    .with(new GoodsItemQuery() {
        @Override
        protected void onSuccess(@NonNull GoodsDetailModel data) {
            //请求成功
        }

        @Override
        protected void onFailure(@NonNull YouzanException error) {
            //请求失败, error.getMessage()可获取失败信息
        }
    });
```

说明:

调用有赞云开发*API*都是需要完成接口认证, 即在调用前按照认证生成 `YouzanToken` 传给*Utils.sync()*方法完成认证.

调用方式:

- `put(String key, Object value)` 方法: 可以传入请求的API接口所对应的键值对参数, 具体可以参考API文档的入参说明.
- `puts(Map<String, String> parameters)` 方法: 可以将参数以map的形式一次性传入.
- `with(Query<MODEL> query)` 方法: 传入具体API请求类, 并开始网络请求.
- `intercept(HttpInterceptor interceptor)` 方法: 对HTTP response进行拦截, 可实现 `HttpInterceptor` 接口根据返回HTTP BODY以控制是否拦截回调.

> 获取更多开放*API*

开放API

本节内容:

了解开放API调用方式请参考上一节网络调用框架, 后续将会根据需求集成更多API.

商品

获取单个商品信

> 调用类: GoodsItemQuery
>
> 有赞云开放API: youzan.item.get

获取商品优惠详情

> 调用类: GoodsPromotionQuery
>
> 有赞云开放API: youzan.ump.promotion.get

商品评价和评论的统计数

> 调用类: GoodsRateCountQuery
>
> 有赞云开放API: youzan.item.reviews.count

商品评价和评论，复合条件查询

> 调用类: GoodsReviewsQuery
>
> 有赞云开放API: youzan.item.reviews.query

店铺

店铺基本信息

> 调用类: ShopBasicQuery
>
> 有赞云开放API: youzan.shop.get

交易

下单入口，进入结算页

> 调用类: TradeBillUrlQuery

> 有赞云开放API: youzan.trade.bill.goods.url.get

多商品下单入口，进入结算页

> 调用类: TradeBillGoodsQuery
>
> 有赞云开放API: youzan.trade.bill.goods.url.get

添加商品至购物车

> 调用类: TradeCartAddQuery
>
> 有赞云开放API: youzan.trade.cart.add

查询用户在购物车中的商品数量

> 调用类: TradeCartCountQuery
>
> 有赞云开放API: youzan.trade.cart.count

删除购物车中的商品

> 调用类: TradeCartDeleteQuery
>
> 有赞云开放API: youzan.trade.cart.delete

显示购物车中的商品列表

> 调用类: TradeCartListQuery
>
> 有赞云开放API: youzan.trade.cart.list

工具

获取七牛文件上传Token

> 调用类: UploadTokenQuery
>
> 说明: 返回值即 `token` 即为UploadQiniuQuery调用类的入参.

七牛文件上传

> 调用类: UploadQiniuQuery
>
> 七牛API文档: up.qbox.me

更新说明

===

目录:

版本5.2.0

更新内容:

- 混合版视图支持页面回退，提供通过 `YouzanRouterClient` 重写url跳转的方式，并去除之前版本url跳转时重新start当前Activity的限制;
- 订单列表页支持提示用户绑定有赞账号显示买家在其他渠道的订单信息，商家后台可设置是否显示;
- 客服页账号打通，免登录咨询;
- 修复若干bug;

接入修改注意:

- 包名由 `com.youzan.sdk` 规范为 `com.youzan.androidsdk`，相应基础版视图类为 `com.youzan.androidsdk.basic.YouzanBrowser`，混合版视图类为 `com.youzan.androidsdk.hybrid.YouzanHybrid`；
- `YouzanSDK.init` 方法需增加 `adapter` 参数，控制初始化基础版 or 混合版;
- 混合版页面跳转需要注意，不再基于当前Activity传入url重新start，需要通过 `YouzanHybrid.setPageRouterClient(YouzanRouterClient client)` 自定义实现;

版本5.2.2

更新内容:

- 修复了混合版本的若干bug;

版本5.3.3

更新内容:

- 增加h5页面静态资源缓存，提升h5加载速度;
- 增加版本号统计;
- 修复若干bug;

版本6.0.0

- 动态获取H5页面静态资源更新，优化预加载策略，提升H5页面初次及二次资源加载的缓存命中率及页面加载速度（30%以上的提升）;

- 修复一些已知bug;

版本6.1.0

- html页面加速，增加初始化并行、缓存及预取接口，优化首屏html加载体验;
- 优化静态资源下载，复用webview资源流，节省用户流量;
- 修复接入端使用GrowingIO、听云oneapm等插件改写字节码可能造成的问题;
- 修复大用户量下发现的一些偶现异常;

常见问题

===

目录:

关于Token与Cookie有效期

access_token有效期7天， 而cookie_key, cookie_value有效期只有半小时（当买家在页面有操作时，cookie_key,cookie_value的有效期会自动延长。

但当前的接口中Token和Cookie是在同一个接口返回的，所以当调用登录接口时，其实三者是同步更新了。

可以考虑在应用启动时调用登录接口获取Token及Cookie, 并调用 `YouzanSDK.sync(token)`，这样，当真的需要token时，就可以直接加载，不需要再从远端取。可以提高加载速度。

软键盘覆盖问题

由于有赞云SDK对外输出的只是一个View，无法控制软键盘模式。请在Activity的声明中将windowSoftMode设置为 **android:windowSoftInputMode="stateHidden|adjustResize"**

初次进入登录态页面耗时较长

这个问题在一些网络不佳的情况容易出现，原因是多方面的。我们针对h5资源做缓存的加速方案正在开发中，预计可以取得一些效果。

另者：1）请检查页面是否图片资源过多、过大; 2）query token的流程可以在后台异步先做掉（参考上述Token），节省这部分时间;

混合版无法展示图片

在确定无网络问题且确定项目中已经集成相应的图片加载框架的情况下, 无法显示页面内的图片, 效果如图:

解决方法:

先确定项目中存在以下图片加载框架中的一种:

图片加载框架	兼容版本	适配器
Picasso	2.5.2	PicassoFactory
UniversalImageLoader	1.9.5	UILFactory
Glide	3.3.0+	GlideFactory
Fresco	0.8.0+	FrescoFactory
Volley	1.0.0+	VolleyFactory

一般是由于代码混淆造成的SDK无法找到图片加载框架, 解决方案有两种.

一种是修改项目的代码混淆策略， 参考混淆规则;

另一种是主动为SDK设定加载框架的适配器, 需要在展示 `YouzanHybrid` 之前调用:

```
YouzanHybrid.setImageLoaderFactory(new PicassoFactory());
```

> 其中 `PicassoFactory` 为Picasso的适配器, 其他适配器还有: `FrescoFactory` , `GlideFactory` , `UILFactory` , `VolleyFactory` 等.

找不到constraint layout相关资源

错误类似:

```
AAPT: No resource identifier found for attribute 'layout_constraintLeft_toLeftOf'
```

请在项目中build.gradle中,引入:

```
repositories {
     maven {
        url 'https://maven.google.com'
    }
}
```

需要登录时，用户返回未登录状态处理

当收到AbsAuthEvent要求登录获取token时，如果用户选择取消登录，则mView容器需要重置自身状态，需要回调mView.syncNot()方法处理。syncNot方法会根据当前页面栈状态判断是重置组件状态还是回退。

调用方法:

```
if(!mView.syncNot()) { // 如果返回false表示mView内部无法处理(重置或回退，一般此时页面view栈只有一层，需要外层视图容器处理)
  onBackPressed();
}
```

关于使用GrowingIO、听云oneapm等插件的兼容及可能造成的问题

在YouzanBrowser中，为了处理状态同步、资源加速等问题，重写了WebView、WebViewClient、WebChromeClient的一些方法。

而GrowingIO、听云oneapm等插件为了能够自动收集页面数据也可能改写相关方法，由此可能造成不兼容的问题。

我们尽量分析了GrowingIO、听云oneapm等库，包括其不同版本的改写方式，做了兼容实现。所以一般情况下，用户不会遇到此类问题，但不排除有未尽情况，或者此类三方库后续版本变化又未实时发现的情况。

因此，以下做一些说明:

1、GrowingIO会自己生成一个WebChromeClient将用户自己设置的WebChromeClient包装起来，具体调用方式:

```
VdsAgent.setWebChromeClient((WebView)this, paramContext); // paramContext 是用户定义的WebChromeClient
```

目前未发现GrowingIO改写出现问题的情况，可以正常使用。

2、听云oneapm的改写要粗糙得多，如果重写了setWebViewClient方式，它会在最后一句加上它自己的调用:

```
Callback.setWebViewClient(this, paramWebViewClient);
```

这里就有可能造成死循环了， AppSDK中已经处理这种情况，可能正常使用。

但是在听云5.1.0.0版本中发现了以下改写:

```
public WebResourceResponse shouldInterceptRequest(WebView paramWebView, String paramString)
{
  if (Build.VERSION.SDK_INT > 10) {
    if (Callback.access$100(paramString)) {
      return this.a.shouldInterceptRequest(paramWebView, paramString);
    }
  }
  try
  {
    localObject1 = new URL(paramString).openConnection();
    localInputStream = ((URLConnection)localObject1).getInputStream();
    localObject1 = ((URLConnection)localObject1).getContentType();
    str2 = "";
    if (TextUtils.isEmpty((CharSequence)localObject1)) {
    if (((String)localObject1).indexOf(";") == -1) {
    localObject1 = ((String)localObject1).split(";");
    str2 = localObject1[0];
    String[] arrayOfString = localObject1[1].trim().split("=");
    if ((localObject1.length != 2) || (!arrayOfString[0].trim().toLowerCase().equals("charset"))) {
      break label185;
    }
    localObject1 = arrayOfString[1].trim();
  }
  catch (MalformedURLException localMalformedURLException)
  {
  catch (IOException localIOException)
  {
  if (str2.contains("text/html"))
  {
    localInputStream = Callback.addBiAgent(localInputStream);
    if ((localInputStream != null) && (Build.VERSION.SDK_INT > 11))
    {
      localObject1 = new WebResourceResponse(str2, (String)localObject1, localInputStream);
      return (WebResourceResponse)localObject1;
    }
  }
}
```

也就是说听云拦截了html的请求，采用URLConnection自己去请求后端，返回内容。在有赞的页面中，需要根据UA处理相应逻辑、返回正确页面。而听云的改写是完全没有考虑设置UA、Cookie等情况，故不能使用。