Legend Option Android SDK 接入文档

集成

1. 将OptionLib-release.aar放在app module下的libs目录下

2. 在根目录地build.gradle中增加

```
allprojects {
    repositories {
        maven { url "https://jitpack.io" }
    }
}
```

3. 在app下的build.gradle中添加依赖

```
// support版本
implementation files('libs/OptionLib-release.aar')
implementation 'com.android.support:appcompat-v7:28.0.0'
implementation 'com.android.support.constraint:constraint-layout:2.0.4'
implementation 'com.android.support:design:28.0.0'
implementation 'android.arch.lifecycle:extensions:1.1.1'
implementation 'com.github.CymChad:BaseRecyclerViewAdapterHelper:2.9.49'
implementation 'com.squareup.okhttp3:okhttp:4.9.1'
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
implementation 'com.github.bumptech.glide:glide:4.8.0'
implementation "org.jetbrains.kotlinx:kotlinx-coroutines-core:1.3.9"
implementation "org.jetbrains.kotlinx:kotlinx-coroutines-android:1.3.9"

// androidx版本
implementation files('libs/OptionLib-release.aar')
implementation 'androidx.appcompat:appcompat:1.3.1'
implementation 'com.google.android.material:material:1.4.0'
implementation 'androidx.constraintlayout:constraintlayout:2.0.2'
implementation 'androidx.swiperefreshlayout:swiperefreshlayout:1.1.0'
implementation 'com.github.CymChad:BaseRecyclerViewAdapterHelper:2.9.49'
implementation 'com.squareup.okhttp3:okhttp:4.9.1'
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
implementation 'com.github.bumptech.glide:glide:4.11.0'
implementation "org.jetbrains.kotlinx:kotlinx-coroutines-core:1.3.9"
implementation "org.jetbrains.kotlinx:kotlinx-coroutines-android:1.3.9"
```

4. 在项目proguard-rules.pro中增加混淆规则

```
-keep class com.legend.option.lib.** {*;}
```

使用

1. **初始化sdk**，在主APP中的自定义**Application**中的**onCreate**方法中调用

```
LegendOptionSdk.init(
    this,
    "http://aa1601f5c51db3149.awsglobalaccelerator.com:38219", /*域名*/
    "hyperw", /*渠道商id*/
    BuildConfig.DEBUG, /*是否debug，为true会输出日志*/
    object : IProvider { /*需要实现的一些方法，供sdk调用*/
        /*当用户token过期时的实现，例如清除登录信息，重新登录*/
        override fun onUserTokenExpired() {
            Toast.makeText(this@BaseApplication, "Token 过期，请重新登录",
Toast.LENGTH_SHORT).show()

            this@BaseApplication.startActivity(
                Intent(this@BaseApplication, LoginActivity::class.java)
                    .addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
            )
        }

        /*为了统一app的toast风格，提供app的toast方法*/
        override fun onToastMessage(context: Context, msg: String) {
            Toast.makeText(context, msg, Toast.LENGTH_SHORT).show()
        }

        /*跳转到去登录的页面*/
        override fun toLogin(context: Context) {
            Toast.makeText(context, "去登录", Toast.LENGTH_SHORT).show()
            context.startActivity(Intent(context, LoginActivity::class.java))
        }

        /*跳转到去授权的页面，如果app的期权不需要授权，则保持这个方法空实现即可*/
        override fun toAuthorize(context: Context) {
            Toast.makeText(context, "去授权", Toast.LENGTH_SHORT).show()
            AlertDialog.Builder(context)
                .setMessage("是否开通牛熊对战")
                .setPositiveButton("开通") {dialog,_ ->
                    LegendOptionSdk.authorize(token)
                    dialog.dismiss()
                }
                .setNegativeButton("取消") {dialog, _ ->
                    dialog.dismiss()
                }
                .show()
        }

        /*跳转到app的划转页面*/
```

```
        override fun toTransfer(context: Context) {
            Toast.makeText(context, "跳转到App的转账页面",
Toast.LENGTH_SHORT).show()
        }
    })
```

2. 当**主APP登录成功**后，调用 **LegendOptionSdk.login()** 方法，通知sdk用户登录。如果期权模块 **不需要用户手动授权** ，继续调用 **LegendOptionSdk.authorize(token)** 方法。如果需要用户手动授权，则需要在授权页面同意按钮点击后调用此方法。

3. 当**主APP退出登录**时，需要调用 **LegendOptionSdk.logout()** 方法来通知SDK用户退出

4. 在需要展示牛熊的页面，引入 **BullBearFragment** 到相应的页面来展示UI

5. 当**主APP切换语言**时，需要在**切换语言处**增加 **LegendOptionSdk.lang = LangType.对应语言**, 目前支持的LangType有---中文 zh_CN, 韩文 ko_KR, 繁体 zh_TW, 英文 en_US, 俄语 ru_RU, vi_VN越南语，th_TH泰语