转载请注明链接: https://blog.csdn.net/feather_wch/article/details/88266603

> 快速集成RePlugin以简单实例，讲解内置插件、外置插件、等众多用法。

RePlugin快速集成实战

版本号:2019-03-10(16:30)

宿主App

根目录build.gradle

1. 使用阿里的代理库，不然更新太慢
2. 这里AS3.1配合的gradle-4.5，如果是AS3.3必须要gradle-4.10 会出现不兼容的地方。

```
// Top-level build file where you can add configuration options common to all sub-projects/modu

buildscript {
    repositories {
        maven { url 'http://maven.aliyun.com/nexus/content/groups/public/' }
        maven { url 'http://maven.aliyun.com/nexus/content/repositories/jcenter' }
        maven { url 'http://maven.aliyun.com/nexus/content/repositories/google' }
        maven { url 'http://maven.aliyun.com/nexus/content/repositories/gradle-plugin' }

    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.1.0'
        classpath 'com.qihoo360.replugin:replugin-host-gradle:2.3.1'
    }
}

allprojects {
    repositories {
        maven { url 'http://maven.aliyun.com/nexus/content/groups/public/' }
        maven { url 'http://maven.aliyun.com/nexus/content/repositories/jcenter' }
        maven { url 'http://maven.aliyun.com/nexus/content/repositories/google' }
        maven { url 'http://maven.aliyun.com/nexus/content/repositories/gradle-plugin' }

    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

app的build.gradle

1. apply plugin: 'replugin-host-gradle'要在applicationId后面
2. 根据情况进行配置：repluginHostConfig
3. 不要使用androidx，app运行后会报错
4. 依赖 `implementation 'com.qihoo360.replugin:replugin-host-lib:2.3.1'`

```
apply plugin: 'com.android.application'

android {
    defaultConfig {
        applicationId "com.hao.repluginhost"
        // xxx
    }
}

// ATTENTION!!! Must be PLACED AFTER "android{}" to read the applicationId
apply plugin: 'replugin-host-gradle'

repluginHostConfig {
    /**
     * 是否使用 AppCompat 库
     * 不需要个性化配置时，无需添加
     */
    useAppCompat = true
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    // 不能使用androidx
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'

    // Replugin
    implementation 'com.qihoo360.replugin:replugin-host-lib:2.3.1'
}
```

RePluginApplication

1. 自定义application继承自RePluginApplication
2. AndroidManifest中进行注册

```
    <application
        android:name=".BaseApplication">
    </application>
```

```
public class BaseApplication extends RePluginApplication{
    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);

        // FIXME 允许接收rpRunPlugin等Gradle Task，发布时请务必关掉，以免出现问题
        RePlugin.enableDebugger(base, BuildConfig.DEBUG);
    }

    /**============================================================
     * RePlugin允许提供各种“自定义”的行为，让您“无需修改源代码”，即可实现相应的功能
     *============================================================*/
    @Override
    protected RePluginConfig createConfig() {
        RePluginConfig c = new RePluginConfig();

        // 允许“插件使用宿主类”。默认为“关闭”
        c.setUseHostClassIfNotFound(true);

        // FIXME RePlugin默认会对安装的外置插件进行签名校验，这里先关掉，避免调试时出现签名错误
        c.setVerifySign(!BuildConfig.DEBUG);

        // 针对“安装失败”等情况来做进一步的事件处理
        c.setEventCallbacks(new HostEventCallbacks(this));

        // FIXME 若宿主为Release，则此处应加上您认为"合法"的插件的签名，例如，可以写上"宿主"自己的。
        // RePlugin.addCertSignature("AAAAAAAAA");

        // 在Art上，优化第一次loadDex的速度
        // c.setOptimizeArtLoadDex(true);
        return c;
    }

    @Override
    protected RePluginCallbacks createCallbacks() {
        return new HostCallbacks(this);
    }

    /**
     * 宿主针对RePlugin的自定义行为
     */
    private class HostCallbacks extends RePluginCallbacks {

        private static final String TAG = "HostCallbacks";

        private HostCallbacks(Context context) {
            super(context);
        }

        @Override
        public boolean onPluginNotExistsForActivity(Context context, String plugin, Intent inte
            // FIXME 当插件"没有安装"时触发此逻辑，可打开您的"下载对话框"并开始下载。
            // FIXME 其中"intent"需传递到"对话框"内，这样可在下载完成后，打开这个插件的Activity
            if (BuildConfig.DEBUG) {
```

```
                Log.d(TAG, "onPluginNotExistsForActivity: Start download... p=" + plugin + "; i
            }
            return super.onPluginNotExistsForActivity(context, plugin, intent, process);
        }

    }

    private class HostEventCallbacks extends RePluginEventCallbacks {

        private static final String TAG = "HostEventCallbacks";

        public HostEventCallbacks(Context context) {
            super(context);
        }

        @Override
        public void onInstallPluginFailed(String path, InstallResult code) {
            // FIXME 当插件安装失败时触发此逻辑。您可以在此处做“打点统计”，也可以针对安装失败情况做‘
            // 大部分可以通过RePlugin.install的返回值来判断是否成功
            if (BuildConfig.DEBUG) {
                Log.d(TAG, "onInstallPluginFailed: Failed! path=" + path + "; r=" + code);
            }
            super.onInstallPluginFailed(path, code);
        }

        @Override
        public void onStartActivityCompleted(String plugin, String activity, boolean result) {
            // FIXME 当打开Activity成功时触发此逻辑，可在这里做一些APM、打点统计等相关工作
            super.onStartActivityCompleted(plugin, activity, result);
        }
    }
}
```

内置插件

build.gradle配置

1、根目录的build.gradle

```
buildscript {

    repositories {
        repositories {
            maven { url 'http://maven.aliyun.com/nexus/content/groups/public/' }
            maven { url 'http://maven.aliyun.com/nexus/content/repositories/jcenter' }
            maven { url 'http://maven.aliyun.com/nexus/content/repositories/google' }
            maven { url 'http://maven.aliyun.com/nexus/content/repositories/gradle-plugin' }
        }
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.1.0'
        classpath 'com.qihoo360.replugin:replugin-plugin-gradle:2.3.1'
    }
}

allprojects {
    repositories {
        repositories {
            maven { url 'http://maven.aliyun.com/nexus/content/groups/public/' }
            maven { url 'http://maven.aliyun.com/nexus/content/repositories/jcenter' }
            maven { url 'http://maven.aliyun.com/nexus/content/repositories/google' }
            maven { url 'http://maven.aliyun.com/nexus/content/repositories/gradle-plugin' }
        }
        google()
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

2、app的build.gradle

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "com.hao.repluginlogin"
        minSdkVersion 23
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'

    // Replugin
    implementation 'com.qihoo360.replugin:replugin-plugin-lib:2.3.1'
}
apply plugin: 'replugin-plugin-gradle'
```

宿主调用内置插件

外置插件

外置插件的项目配置

1、创建一个新功能 `RepluginWeb` ， 根据插件apk的方式进行配置。

> 根目录 build.gradle

```
buildscript {

    repositories {
        maven { url 'http://maven.aliyun.com/nexus/content/groups/public/' }
        maven { url 'http://maven.aliyun.com/nexus/content/repositories/jcenter' }
        maven { url 'http://maven.aliyun.com/nexus/content/repositories/google' }
        maven { url 'http://maven.aliyun.com/nexus/content/repositories/gradle-plugin' }
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.1.0'
        classpath 'com.qihoo360.replugin:replugin-plugin-gradle:2.3.1'
    }
}

allprojects {
    repositories {
        maven { url 'http://maven.aliyun.com/nexus/content/groups/public/' }
        maven { url 'http://maven.aliyun.com/nexus/content/repositories/jcenter' }
        maven { url 'http://maven.aliyun.com/nexus/content/repositories/google' }
        maven { url 'http://maven.aliyun.com/nexus/content/repositories/gradle-plugin' }
    }
}
```

> app的build.gradle

```
apply plugin: 'com.android.application'

android {
    defaultConfig {
        applicationId "com.hao.repluginweb"
        xxx
    }
}

dependencies {
    xxx
    // Replugin
    implementation 'com.qihoo360.replugin:replugin-plugin-lib:2.3.1'
}
apply plugin: 'replugin-plugin-gradle'
```

> AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.hao.repluginweb">

    <application
        xxx>

        <!--插件别名-->
        <meta-data
            android:name="com.qihoo360.plugin.name"
            android:value="webview" />

        <!--插件版本号-->
        <meta-data
            android:name="com.qihoo360.plugin.version.ver"
            android:value="100" />
    </application>

</manifest>
```

2、生成插件apk将其放置到宿主apk的asset/external目录下，用于模拟Apk的下载。

宿主调用外置插件

3、宿主调用外置插件

```
        /**==================================
         * 1、使用WebView的插件中的Activity
         *==================================*/
        findViewById(R.id.webview_btn).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(RePlugin.isPluginInstalled("webview")){
                    RePlugin.startActivity(v.getContext(), new Intent(), "webview", "com.hao.re
                }else{
                    Toast.makeText(MainActivity.this, "You must install webview first", Toast.l
                }
            }
        });
```

宿主下载并加载外置插件

4、宿主下载并且加载外置插件(这里通过文件复制模拟下载流程)

> 1-点击“安装外置插件”Button进行安装

```
/**==================================
 * 2、加载插件apk(webview)-加载外置插件，模拟下载的流程。
 *==================================*/
findViewById(R.id.load_webview_apk).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // 安装插件
        installExternalPlugin("webview.apk");
    }
});
```

> 2-外置插件的下载和安装流程

```
/**==============================
 *  加载外置插件，内部模拟下载过程。
 *==========================*/
private void installExternalPlugin(String pluginApkName){
    // 1. apk名
    String apkName = pluginApkName;
    // 2. apk目前位于assets/external/路径中
    String apkAssetsPath = "external" + File.separator + apkName;
    // 3、如果/data/data/webview.apk文件已经存在，需要删除后重新复制(模拟下载流程)
    String pluginFilePath = getFilesDir().getAbsolutePath() + File.separator + apkName;
    File pluginFile = new File(pluginFilePath);
    if(pluginFile.exists()){
        FileUtils.deleteQuietly(pluginFile); // 删除已经存在的apk
    }
    // 4、将assets/external中的文件复制到app的文件下
    copyAssetsFileToAppFiles(apkAssetsPath, apkName);
    // 5、安装。如果数据目录下存在该插件apk文件，就进行安装。
    if(pluginFile.exists()){
        // 外置插件必须使用详细完整的路径名，如“/sdcard/webview.apk”
        RePlugin.install(pluginFilePath);
    }else{
        Toast.makeText(MainActivity.this, "Download webview failed!", Toast.LENGTH_SHORT).s
    }
}


/**==========================================================================
 * @function 将“assets/external”下的外置插件复制到宿主app的数据目录中，模拟外置插件的下载
 * @param assetFileName assets目录中的文件路径名，例如"assets/external/webview.apk"中的"exter
 * @param newFileName 插件名如"webview.apk"
 *==========================================================================*/
private void copyAssetsFileToAppFiles(String assetFileName, String newFileName){

    // 1、assets/external中的外置插件apk作为输入
    try(InputStream inputStream = this.getAssets().open(assetFileName);
        // 2、宿主appp数据目录作为输出
        FileOutputStream fileOutputStream = this.openFileOutput(newFileName, MODE_PRIVATE))

        // 3、IO进行文件的复制
        int byteCount = 0;
        byte[] buffer = new byte[1024];
        while((byteCount = inputStream.read(buffer)) != -1){
            fileOutputStream.write(buffer, 0, byteCount);
        }
        // 4、刷新输出流
        fileOutputStream.flush();

    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

插件的进阶用法

插件的升级

获取插件的服务

获取插件的Fragment

1、xml直接引用插件的Fragment

1、利用Hook支持xml中直接引用插件Fragment

> 1-XML

```
<fragment
    android:name="com.hao.repluginweb.ItemFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</fragment>
```

> 2-Hook替换Fragment

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    /**=======================================
     * 1、XML中使用插件中的Fragment，注册一个全局的Hook，将宿主中的Fragment定向到插件中的Fragm
     *=======================================*/
    String pluginName = "com.hao.repluginweb";
    RePlugin.registerHookingClass("com.hao.repluginweb.ItemFragment",
            RePlugin.createComponentName(pluginName, "com.hao.repluginweb.ItemFragment"), r
    setContentView(R.layout.activity_main_fragment);
}
```

2、代码中加载Fragment

2、代码中可以动态加载插件中的Fragment

```
        /**=======================================
         * 代码使用插件Fragment
         *=========================================*/

        String pluginName = "com.hao.repluginweb";

        // 1、获取插件的ClassLoader
        ClassLoader d1ClassLoader = RePlugin.fetchClassLoader(pluginName);
        try {
            // 2、利用插件的ClassLoader获取到目标Fragment
            Fragment fragment = d1ClassLoader.loadClass("com.hao.repluginweb.ItemFragment").as
            // 3、动态加载Fragment
            getFragmentManager().beginTransaction().add(R.id.fragment_container_cl, fragment).
        } catch (Exception e) {
            e.printStackTrace();
        }
```

cannot be cast to android.support.v4.app.Fragment

1、解决办法1：Fragment继承自android.app.Fragment不会有该类问题

2、解决办法2:

> 插件Apk的build.gradle

```
dependencies {
    implementation fileTree(include: ['*.jar'], dir: 'libs')
    implementation 'com.android.support:appcompat-v7:28.0.0'
    // Replugin
    implementation 'com.qihoo360.replugin:replugin-plugin-lib:2.3.1'
    // 1、骗过编译期
//    provided files('libs/fragment.jar')
    compileOnly files('libs/fragment.jar')
}
// 2、借助Gradle 的exclude语法来解决版本冲突
configurations { all*.exclude group: 'com.android.support', module: 'support-v4' }
```

3、解决方案官方链接：FAQ

插件调用宿主

宿主和插件的通信

插件多进程

插件

问题汇总

参考资料

1. 插件和宿主间的通信方式
2. android.support.v4.app.Fragment和android.app.Fragment区别