tinker的使用

1、注册

在Tinker官网注册，新建APP，获取appKey,官网地址： http://www.tinkerpatch.com

2、集成

项目的build.gradle

```
dependencies {
    classpath 'com.android.tools.build:gradle:3.4.1'
    // TinkerPatch 插件
    classpath "com.tinkerpatch.sdk:tinkerpatch-gradle-plugin:1.2.6"

}
```

module的build.gradle

引入tinker的gradle

```
apply plugin: 'com.android.application'
apply from: 'tinker-support.gradle'
```

引入jar包

```
    compile 'com.android.support:multidex:1.0.1'
    // 多dex配置
    implementation 'com.tinkerpatch.sdk:tinkerpatch-android-sdk:1.2.6'
```

配置签名，正式环境要用

```
signingConfigs {
    config {
        keyAlias 'rcss'
        keyPassword 'rcss2018'
        storeFile file('C:/Users/18030213.SHDM/Desktop/richangshishi.jks')
        storePassword 'richangshishi'
    }
}
```

Application里面配置轮询请求Tinker后台和MultiDex分包

```
private void initTinkerPatch() {
    Log.e(TAG, msg: "lpf--initTinkerPatch");
    tinkerApplicationLike = TinkerPatchApplicationLike.getTinkerPatchApplicationLike();
    // 初始化TinkerPatch SDK, 更多配置可参照API章节中的,初始化SDK
    TinkerPatch.init(tinkerApplicationLike)
            .reflectPatchLibrary()
            .setPatchRollbackOnScreenOff(true)
            .setPatchRestartOnSrceenOff(true)
            .setFetchPatchIntervalByHours(1);

    // 每隔1个小时(通过setFetchPatchIntervalByHours设置)去访问后台时候有更新,通过handler实现轮训的
    TinkerPatch.with().fetchPatchUpdateAndPollWithInterval();
    //启动时立即去后台检查
    TinkerPatch.with().fetchPatchUpdate(true);
}
```

```
@Override
public void attachBaseContext(Context base) {
    super.attachBaseContext(base);
    //you must install multiDex whatever tinke
    MultiDex.install(base);
}
```

引入tinkerpatch-support，根据自身需求修改

文件如下：

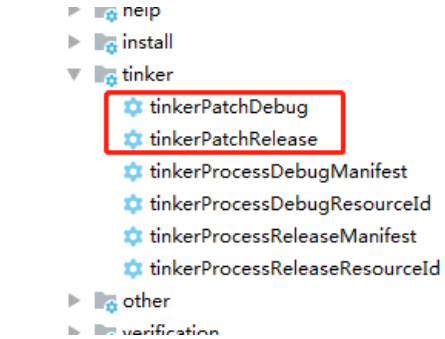
```
apply plugin: 'tinkerpatch-support'

/**
 * TODO: 请按自己的需求修改为适应自己工程的参数
 */
def bakPath = file("${buildDir}/bakApk/")
def baseInfo = "tinker-1.0.0-0712-17-00-48"
def variantName = "debug"

/**
 * 对于插件各参数的详细解析请参考
 * http://tinkerpatch.com/Docs/SDK
 */
tinkerpatchSupport {
    /** 可以在debug的时候关闭 tinkerPatch **/
    /** 当disable tinker的时候需要添加multiDexKeepProguard和proguardFiles,
     这些配置文件本身由tinkerPatch的插件自动添加，当你disable后需要手动添加
     你可以copy本示例中的proguardRules.pro和tinkerMultidexKeep.pro,
     需要你手动修改'tinker.sample.android.app'本示例的包名为你自己的包名, com.xxx前缀的包名不用
修改
     **/
    tinkerEnable = true
    reflectApplication = true

    /**
     * 是否开启加固模式，只能在APK将要进行加固时使用，否则会patch失败。
     * 如果只在某个渠道使用了加固，可使用多flavors配置
     **/
    protectedApp = true
    /**
     * 实验功能
     * 补丁是否支持新增 Activity (新增Activity的exported属性必须为false)
     **/
    supportComponent = true

    autoBackupApkPath = "${bakPath}"

    appKey = "9f084231115d28d2"

    /** 注意: 若发布新的全量包, appVersion一定要更新 **/
```

```
    appVersion = "1.0.0"

    def pathPrefix = "${bakPath}/${baseInfo}/${variantName}"
    def name = "${project.name}-${variantName}"

    baseApkFile = "${pathPrefix}/${name}.apk"
    baseProguardMappingFile = "${pathPrefix}/${name}-mapping.txt"
    baseResourceRFile = "${pathPrefix}/${name}-R.txt"

    /**
     *  若有编译多flavors需求, 可以参照:  https://github.com/TinkerPatch/tinkerpatch-flavors-sample
     *  注意: 除非你不同的flavor代码是不一样的,不然建议采用zip comment或者文件方式生成渠道信息（相关工具: walle 或者 packer-ng）
     **/
}

/**
 * 用于用户在代码中判断tinkerPatch是否被使能
 */
android {
    defaultConfig {
        buildConfigField "boolean", "TINKER_ENABLE",
"${tinkerpatchSupport.tinkerEnable}"
    }
}

/**
 * 一般来说,我们无需对下面的参数做任何的修改
 * 对于各参数的详细介绍请参考:
 * https://github.com/Tencent/tinker/wiki/Tinker-%E6%8E%A5%E5%85%A5%E6%8C%87%E5%8D%97
 */
tinkerPatch {
    ignoreWarning = false
    useSign = true
    dex {
        dexMode = "jar"
        pattern = ["classes*.dex"]
        loader = []
    }
    lib {
        pattern = ["lib/*/*.so"]
    }

    res {
        pattern = ["res/*", "r/*", "assets/*", "resources.arsc",
"AndroidManifest.xml"]
```

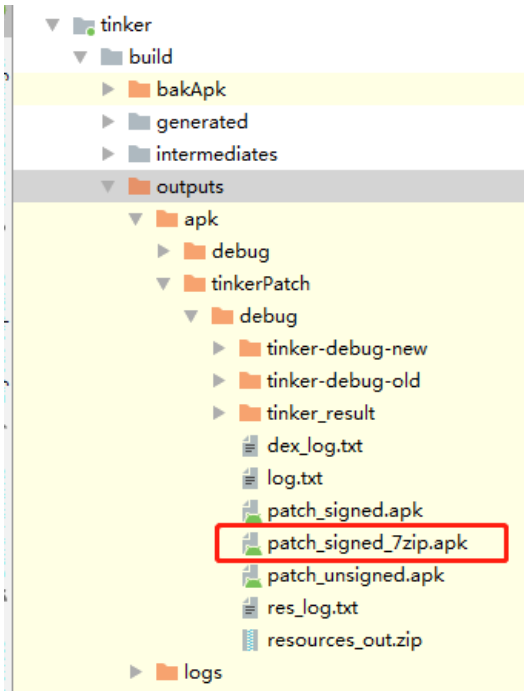
```
        ignoreChange = []
        largeModSize = 100
    }

    packageConfig {
    }
    sevenZip {
        zipArtifact = "com.tencent.mm:SevenZip:1.1.10"
    }
    buildConfig {
        keepDexApply = false
    }
}
```

3、插件生成

1、执行task任务



2、插件包的位置



3、将插件包传给Tinker后台