Eclipse集成

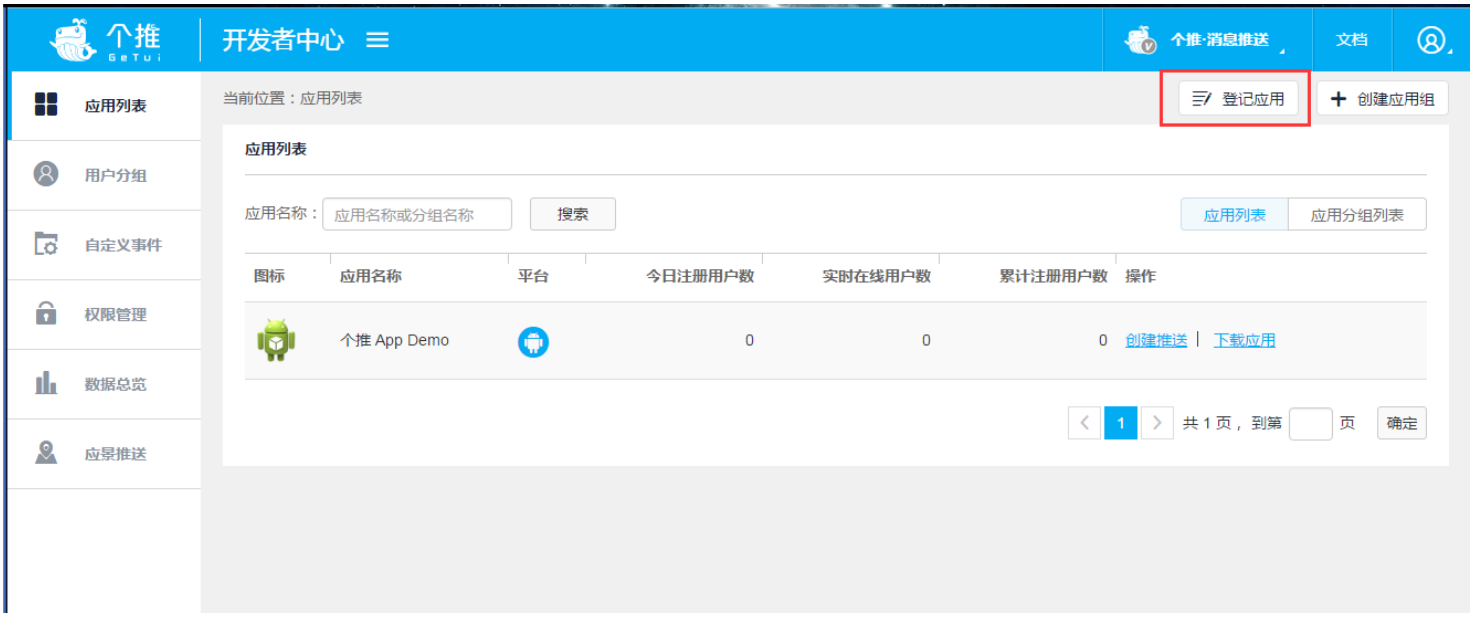
前言

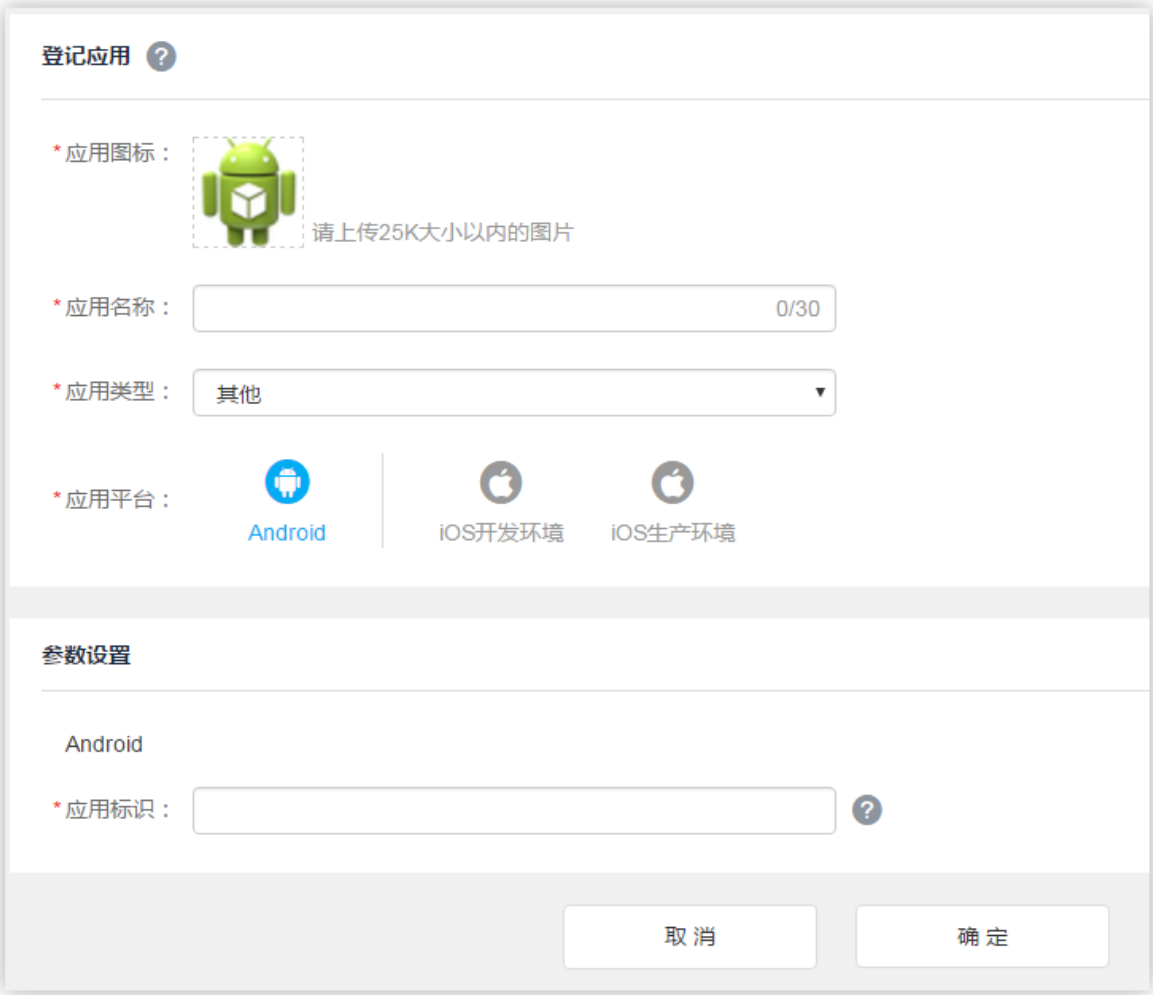
从2016年11月5日起，Google已经正式终止了对 Eclipse Android Developer Tools 的支持，因此我们强烈建议应用开发者尽快迁移到最新的Android Studio开发环境。
可以使用Android Studio提供的 `Import project (Eclipse ADT, Gradle, etc.)` 功能，快速将Eclipse工程转换为Android Studio工程。

- **本文档介绍Eclipse开发环境下手动方式导入SDK资源进行集成的步骤，配置相对复杂，需要仔细阅读文档和Demo工程。我们推荐应用开发者采用基于Android Studio Maven的快速集成方案，详见Android Studio快速集成**
- 本文档适用SDK版本：2.9.5.0及以后
- 请参考 `Getui_SDK_Demo_Eclipse_official` Demo工程

1. 创建个推应用

- 请登录 http://dev.getui.com ，选择 登记应用 并填写应用名称和包名信息，完成应用创建：

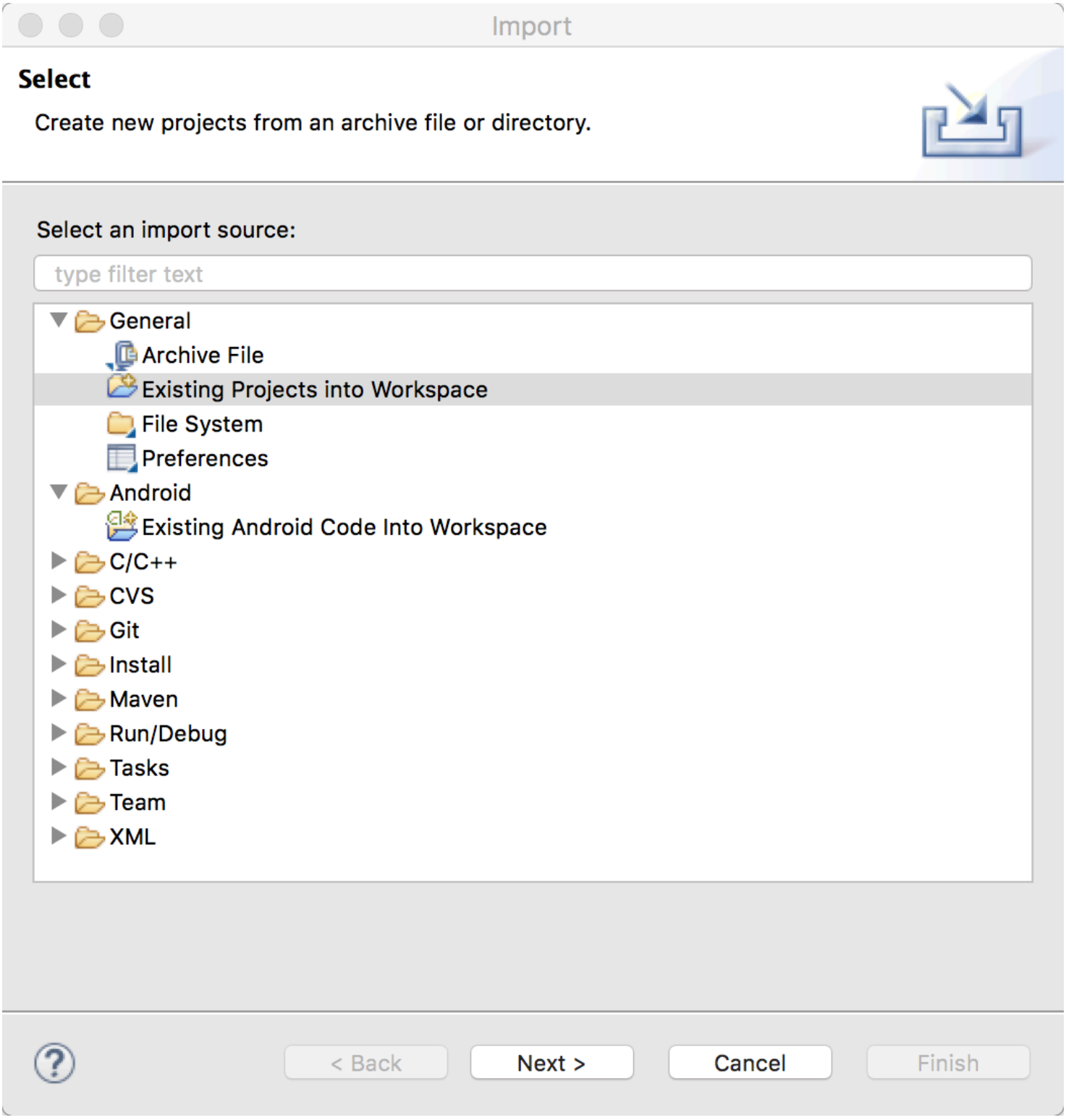




- 点击 `应用配置`，获取到相应的 `AppID` 、 `AppKey` 、 `AppSecret` 信息：

2. 打开项目工程

- 启动Eclipse, 打开或者导入你之前创建的Eclipse项目工程：



3. 添加个推SDK及相关配置

一、老版本（>=2.9.5.0）升级到 2.12.5.0 及以上版本注意事项：

1. 删除原有jni相关目录下的 `libgetuiext2.so` ，并添加 `libgetuiext3.so` 到相同目录
2. 为兼容Android9.0，务必在application节点添加 `android:usesCleartextTraffic="true"`

二、老版本升级到 2.9.5.0 及以上版本注意事项：

1. 替换旧的 `GetuiSDKxxx.jar` ，并删除 `GetuiExt-xxx.jar` 和所有jni相关目录下的 `libgetuiext.so`
2. 删除 `AndroidManifest.xml` 中以下组件相关的配置，最新的SDK已经不再需要这些组件：

`com.igexin.sdk.PushServiceUser`

`com.igexin.sdk.PushManagerReceiver`

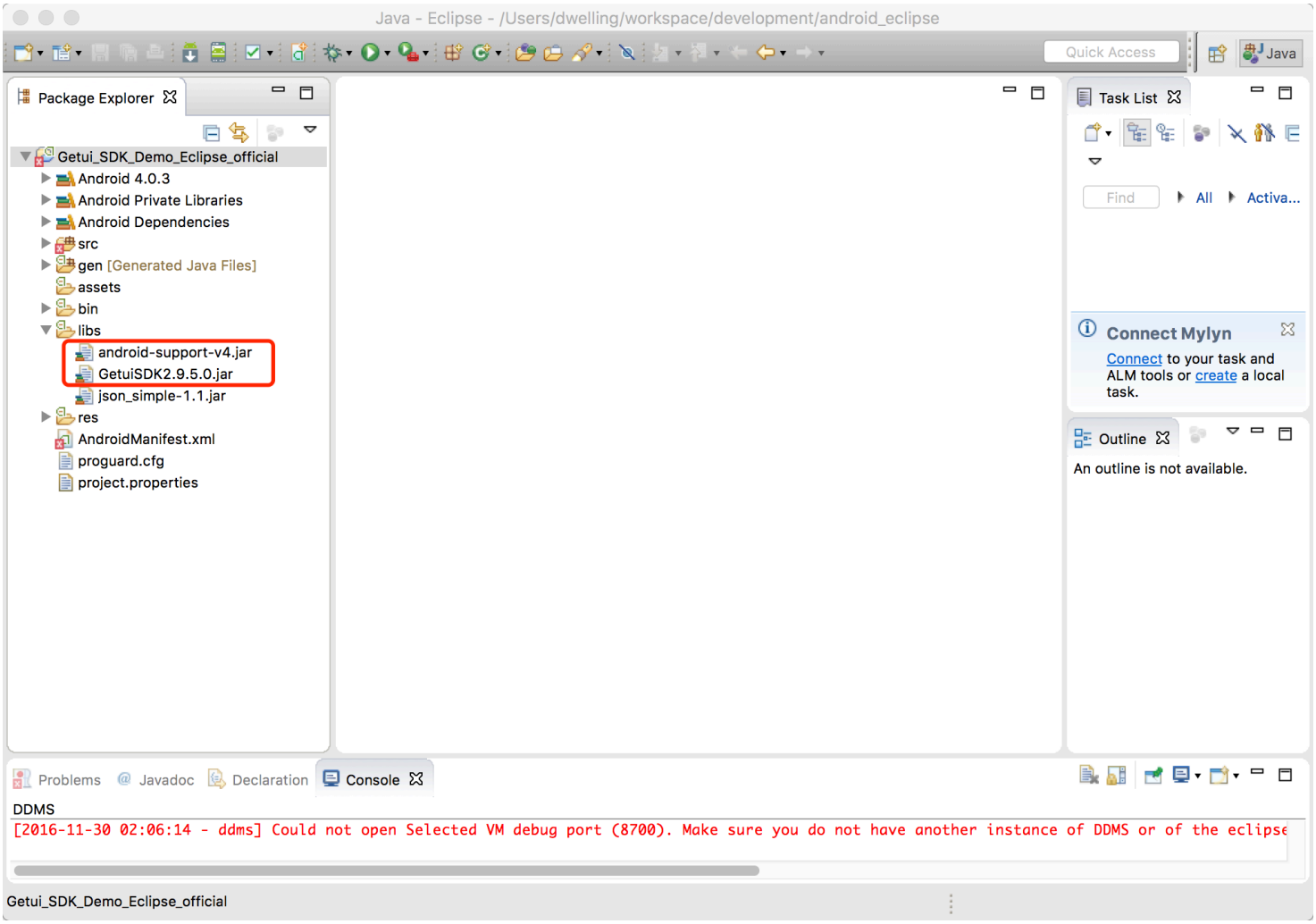
`com.igexin.getuiext.activity.GetuiExtActivity`
`com.igexin.getuiext.service.PayloadReceiver`
`com.igexin.getuiext.service.GetuiExtService`
3. 删除 `res/layout` 目录下原来旧的布局文件，包括 `getui_notification.xml` 、 `notification_inc.xml` 和 `increment_popup_dialog.xml` ，请使用最新SDK所提供的 `getui_notification.xml` 即可
4. 请参考本文档重新进行配置集成

3.1 个推Android SDK资料包结构

```
GETUI_ANDROID_SDK/
  |- readme.txt （SDK资料包说明）
  |- 接入文档/ (Android SDK相关集成文档PDF版本)
  |- 资源文件/
  |    |- res/
  |    |    |- layout/
  |    |    |    |- getui_notification.xml (个推SDK所需的布局文件)
  |    |    |- raw
  |    |    |    |- keep.xml （用于资源保留的描述文件）
  |    |- so/ （各 CPU 架构so库）
  |    |   |- arm64-v8a/
  |    |   |- armeabi/
  |    |   |- armeabi-v7a/
  |    |   |- mips/
  |    |   |- mips64/
  |    |   |- x86/
  |    |   |- x86_64/
  |    |- GetuiSDK2.13.2.0.jar
  | - Demo工程/
  |    |- Getui_SDK_Demo_AS_maven/ （AndroidStudio快速集成Demo工程）
  |    |- Getui_SDK_Demo_AS_official/ （AndroidStudio标准集成Demo工程）
  |    |- Getui_SDK_Demo_Eclipse_official/ （Eclipse集成Demo工程）
```

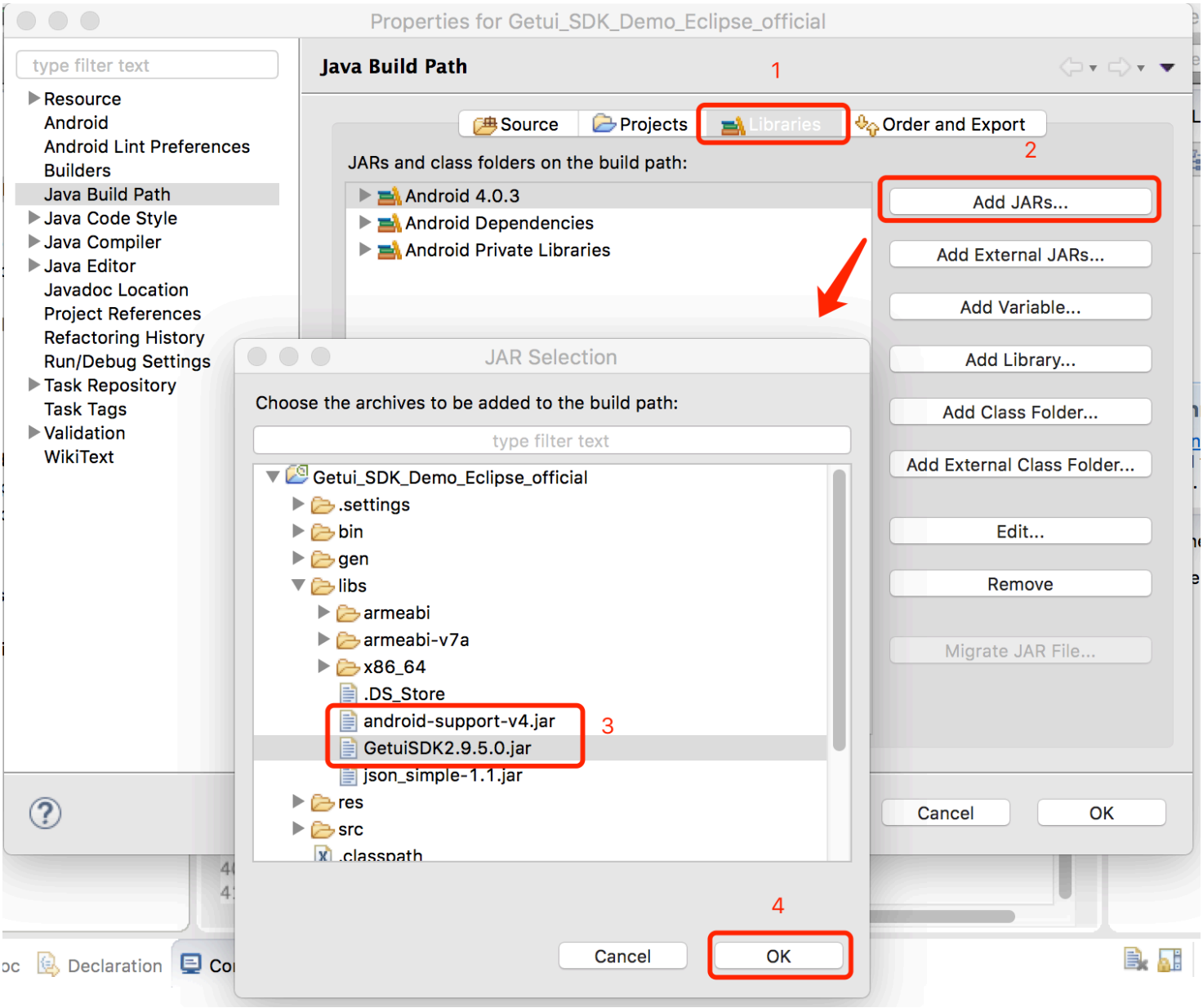
3.2 导入个推SDK

- 将SDK资料包 `GETUI_ANDROID_SDK/资源文件` 目录下的 `GetuiSDKxxx.jar` 复制到Eclipse工程根目录下的 `libs` 目录下，如图所示：



若没有 `libs` 目录，则选中项目工程后点右键，在弹出菜单中选择 `New` -> `Folder` ，将新目录命名为 `libs` 。

- 添加SDK库引用： **（最新的ADT工具会自动导入 `libs` 目录下的jar包，该步骤可以省略）**右键单击工程，选择 `Build Path` 中的 `Configure Build Path...` ，选中 `Libraries` 标签，点击 `Add Jars...` 按钮导入工程 `libs` 目录下的 `GetuiSdk-xxx.jar` 文件：

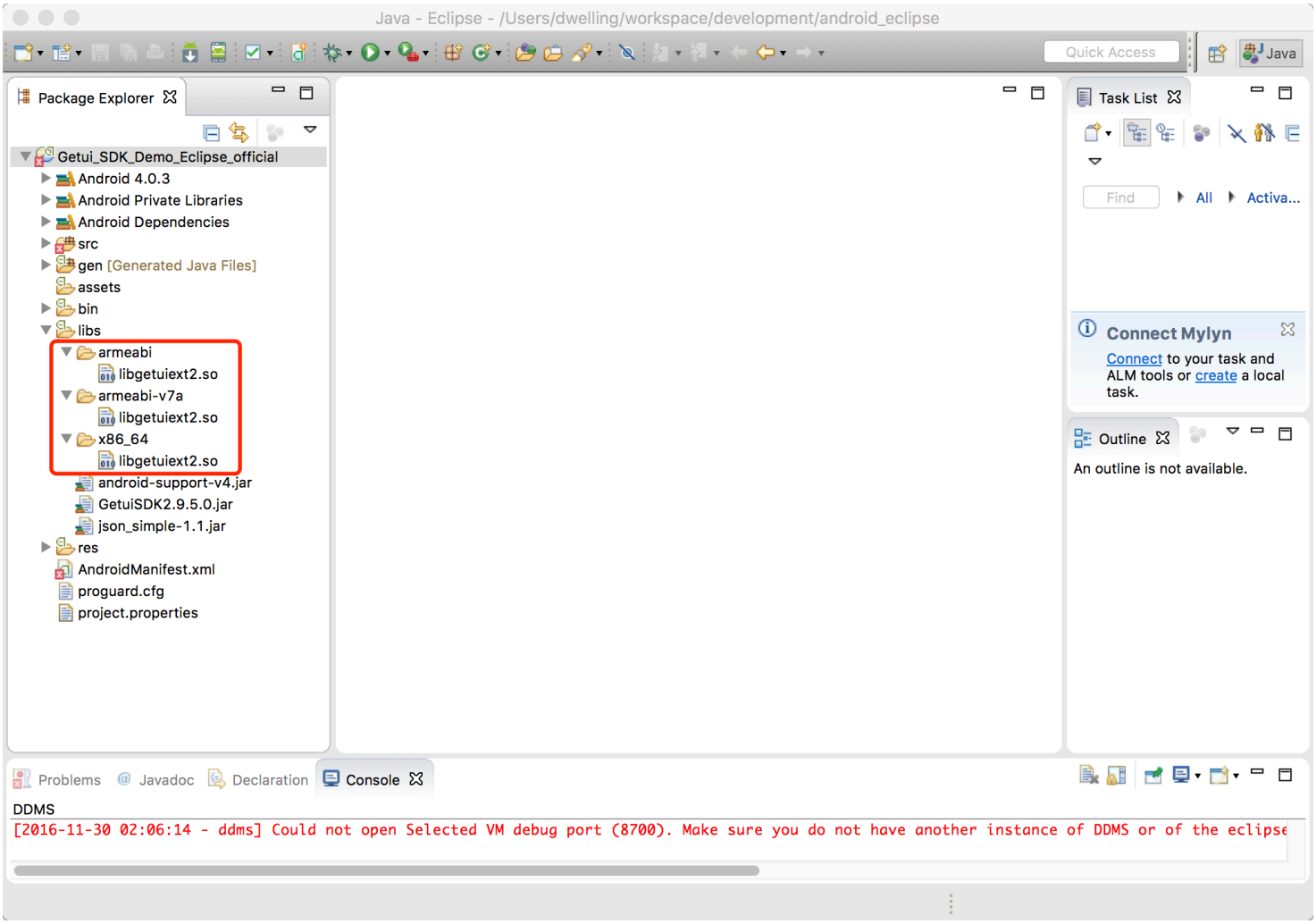


3.3 导入个推so库

目前个推SDK支持 `armeabi` 、 `armeabi-v7a` 、 `arm64-v8a` 、 `mips` 、 `mips64` 、 `x86` 、 `x86_64` 这几种 CPU 架构，请根据项目情况指定所需的架构。

如果项目中包含的其他 so 库只支持其中某几种 cpu 架构，那么应该根据其他 so 库所支持的 CPU 架构的最小集来配置。否则如果在特定架构上未能支持所有 so 库，则很可能导致程序运行异常。切记！

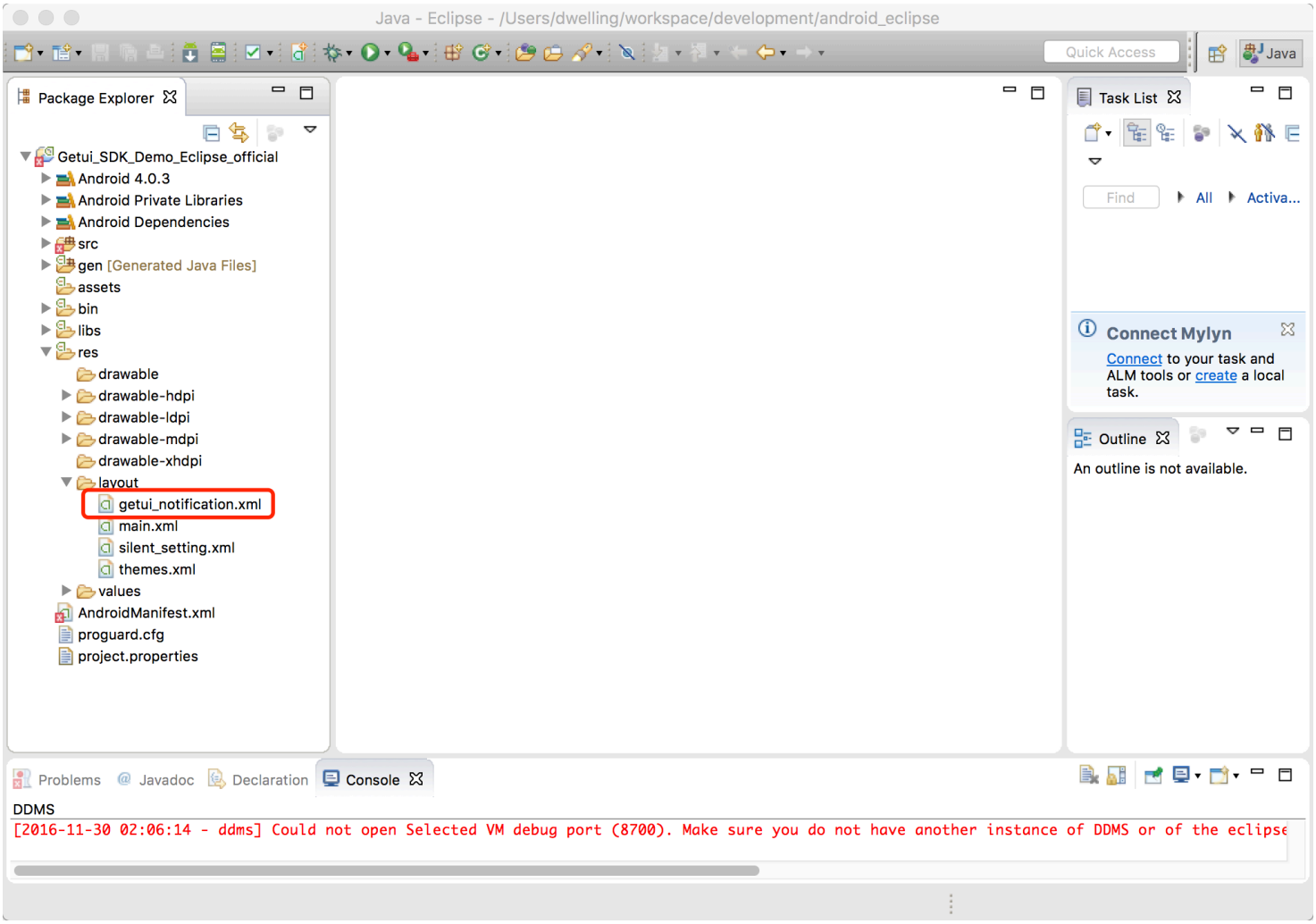
- 将SDK资料包中 `GETUI_ANDROID_SDK/资源文件/so` 目录下所需CPU架构的目录拷贝到Eclipse工程根目录下的 `libs` 目录下，如图所示：



3.4 导入布局文件

为了支持最新的展开式样式和浮动通知，必须使用最新的 `getui_notification.xml` 布局文件

- 将SDK资料包中 `GETUI_ANDROID_SDK/资源文件/layout` 目录下的 `getui_notification.xml` 布局文件复制到项目工程的 `res/layout/` 目录下，如图所示：



3.5 Manifest配置

- 在 `AndroidManifest.xml` 中需要正确配置个推SDK所需的Service、Activity、以及BroadcastReceiver等组件。请在 `<application>` 标签内增加以下组件配置：

```
<!-- 个推SDK配置开始 -->
<!-- Android9.0以上默认不支持http通信，为保证SDK正常使用，请在application节点下新增该属性 -->
<application android:usesCleartextTraffic="true">

<!-- 配置的第三方参数属性 -->
<meta-data
      android:name="PUSH_APPID"
      android:value="你的APPID" /> <!-- 替换为第三方应用的APPID -->
<meta-data
      android:name="PUSH_APPKEY"
      android:value="你的APPKEY" /> <!-- 替换为第三方应用的APPKEY -->
<meta-data
      android:name="PUSH_APPSECRET"
      android:value="你的APPSECRET" /> <!-- 替换为第三方应用的APPSECRET -->

<!-- 配置SDK核心服务 -->
<!-- android:permission在2.13.1.0版本后必须配置 -->
<service
      android:name="com.igexin.sdk.PushService"
      android:permission="android.permission.BIND_JOB_SERVICE"
      android:exported="true"
      android:label="NotificationCenter"
      android:process=":pushservice" >
        <intent-filter>
         <action android:name="com.igexin.sdk.action.service.message"/>
        </intent-filter>
</service>

<receiver android:name="com.igexin.sdk.PushReceiver" >
      <intent-filter>
          <action android:name="android.intent.action.BOOT_COMPLETED" />
          <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
          <action android:name="android.intent.action.USER_PRESENT" />
          <!-- 以下三项为可选的action声明，可大大提高service存活率和消息到达速度 -->
          <action android:name="android.intent.action.MEDIA_MOUNTED" />
          <action android:name="android.intent.action.ACTION_POWER_CONNECTED" />
          <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED" />
      </intent-filter>
</receiver>
<activity
       android:name="com.igexin.sdk.PushActivity"
       android:excludeFromRecents="true"
       android:exported="false"
       android:process=":pushservice"
       android:taskAffinity="com.igexin.sdk.PushActivityTask"
       android:theme="@android:style/Theme.Translucent.NoTitleBar" />
<activity
       android:name="com.igexin.sdk.GActivity"
       android:excludeFromRecents="true"
       android:exported="true"
       android:process=":pushservice"
       android:taskAffinity="com.igexin.sdk.PushActivityTask"
       android:theme="@android:style/Theme.Translucent.NoTitleBar" />

<!-- 个推SDK配置结束 -->
```

需要将上述注释部分的参数替换为【步骤1】获取的个推应用参数

- 请在 `<manifest>` 根标签下加入个推SDK所必需的权限，配置如下：

```
<!-- 个推SDK权限配置开始 -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.GET_TASKS" />
<!-- 浮动通知权限 -->
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>

<!-- 自定义权限 -->
<uses-permission android:name="getui.permission.GetuiService.${applicationId}" />

<permission
    android:name="getui.permission.GetuiService.${applicationId}"
    android:protectionLevel="normal"></permission>
<!-- 个推SDK权限配置结束 -->
```

需要将上述注释部分的包名替换为应用实际的包名

- 一个配置后完整的 `AndroidManifest.xml` 示例如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.getui.demo"
          android:versionCode="2"
          android:versionName="2.0.0">

    <!-- 个推SDK权限配置开始 -->
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="android.permission.GET_TASKS"/>
    <!-- 浮动通知权限 -->
    <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>

    <!-- 自定义权限 -->
    <uses-permission android:name="getui.permission.GetuiService.${applicationId}" />

    <permission
        android:name="getui.permission.GetuiService.${applicationId}"
        android:protectionLevel="normal"></permission>
    <!-- 个推SDK权限配置结束 -->

    <application
        android:usesCleartextTraffic="true"
        android:icon="@drawable/demo"
        android:label="@string/app_name"
        android:persistent="true">
        <!-- 第三方应用配置 -->
        <activity
            android:name=".GetuiSdkDemoActivity"
            android:label="@string/app_name"
            android:launchMode="singleTop">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
```

```
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <!-- 在上面加入你的你的activity配置 -->
        <!-- 个推SDK配置开始 -->
        <!-- 配置的第三方参数属性 -->
        <meta-data
            android:name="PUSH_APPID"
            android:value="wT1tGnaFC98Kgpfoi2u7g6"/>
        <meta-data
            android:name="PUSH_APPKEY"
            android:value="wT1tGnaFC98Kgpfoi2u7g6"/>
        <meta-data
            android:name="PUSH_APPSECRET"
            android:value="my9R0U9s2Y8vLSfeToj6N5"/>

        <!-- 配置SDK核心服务 -->
        <service
            android:name="com.igexin.sdk.PushService"
            android:permission="android.permission.BIND_JOB_SERVICE"
            android:exported="true"
            android:label="NotificationCenter"
            android:process=":pushservice">
            <intent-filter>
                <action android:name="com.igexin.sdk.action.service.message"/>
            </intent-filter>
        </service>

        <receiver android:name="com.igexin.sdk.PushReceiver">
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED"/>
                <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
                <action android:name="android.intent.action.USER_PRESENT"/>
                <!-- 以下三项为可选的action声明，可大大提高service存活率和消息到达速度 -->
                <action android:name="android.intent.action.MEDIA_MOUNTED"/>
                <action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
                <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
            </intent-filter>
        </receiver>
        <activity
            android:name="com.igexin.sdk.PushActivity"
            android:excludeFromRecents="true"
            android:exported="false"
            android:process=":pushservice"
            android:taskAffinity="com.igexin.sdk.PushActivityTask"
            android:theme="@android:style/Theme.Translucent.NoTitleBar"/>
        <activity
            android:name="com.igexin.sdk.GActivity"
            android:excludeFromRecents="true"
            android:exported="true"
            android:process=":pushservice"
            android:taskAffinity="com.igexin.sdk.PushActivityTask"
            android:theme="@android:style/Theme.Translucent.NoTitleBar"/>

        <!-- 个推SDK配置结束 -->
    </application>
</manifest>
```

3.6 配置自定义推送服务

为了让推送服务在部分主流机型上更稳定运行，从2.9.5.0版本开始，个推支持第三方应用配置使用自定义Service来作为推送服务运行的载体。

- 在项目源码中添加一个继承自com.igexin.sdk.PushService的自定义Service：

```
package com.getui.demo;

import com.igexin.sdk.PushService;

public class DemoPushService extends PushService {

}
```

- 在 `AndroidManifest.xml` 中添加上述自定义Service：

```
<service
  android:name="com.getui.demo.DemoPushService"
  android:exported="true"
  android:label="PushService"
  android:process=":pushservice">
</service>
```

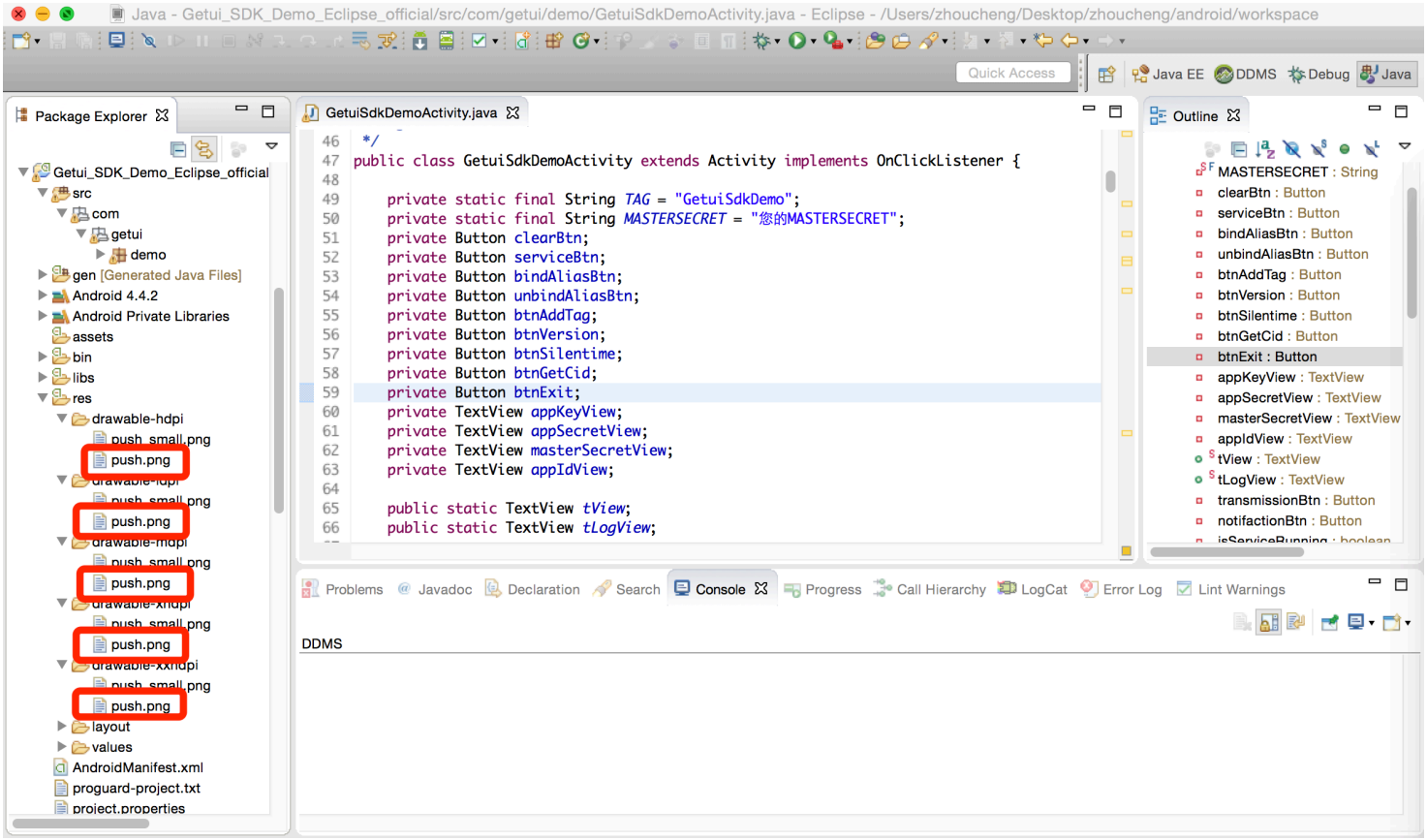
3.7 配置可选权限

- 上述接入方式已包含个推服务所需的所有必备权限。在此之外，您还可以配置以下可选权限，以便使用个推3.0提供的电子围栏功能。请在 `AndroidManifest.xml` 的 `<manifest>` 根标签下添加如下配置：

```
<!-- iBeancon功能所需权限 -->;
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<!-- 个推3.0电子围栏功能所需权限 -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

3.8 导入通知栏图标

- 为了修改默认的通知图标以及通知栏顶部提示小图标，请在资源目录的 `res/drawable-ldpi/` 、 `res/drawable-mdpi/` 、 `res/drawable-hdpi/` 、 `res/drawable-xhdpi/` 、 `res/drawable-xxhdpi/` 等各分辨率目录下，放置相应尺寸的文件名为 `push.png` 和 `push_small.png` 图片，如图所示：



- 建议的 `push.png` 图片尺寸如下：

```
ldpi:    48*48
mdpi:    64*64
hdpi:    96*96
xhdpi:   128*128
xxhdpi:  192*192
```

- 该图标 `push.png` 将会作为通知图标，如下所示：

- 建议的 `push_small.png` 图片尺寸如下：

```
ldpi:    18*18
mdpi:    24*24
hdpi:    36*36
xhdpi:   48*48
xxhdpi:  72*72
xxxhdp:  96*96
```

- 该图标 `push_small.png` 将会作为通知图标展示在通知栏顶部，如下所示：

- `push_small.png` 设计规范请参考状态栏图标设计规范

3.9 Proguard混淆配置

- 如果你的项目需要使用Proguard进行混淆打包，为了避免个推SDK被错误混淆导致功能异常，请在项目的 `proguard.cfg` 文件中添加如下代码：

```
-dontwarn com.igexin.**
-keep class com.igexin.**{*;}
-keep class org.json.** { *; }

-keep class android.support.v4.app.NotificationCompat { *; }
-keep class android.support.v4.app.NotificationCompat$Builder { *; }
```

如果在proguard.cfg中已经添加 `-dontwarn` ，则无需再添加 `-dontwarn com.igexin.**`

3.10 AndResGuard混淆配置

- 如果您的工程使用了 `AndResGuard` 进行资源精简，为了避免个推SDK所需资源被错误精简导致功能异常，需要您在 `config.xml` 文件中的 `<issue id=whitelist>` 节点下添加如下代码

```
<issue id="whitelist" isactive="true">
    <path value="<your_package_name>.R.drawable.push"/>
    <path value="<your_package_name>.R.drawable.push_small"/>
    <path value="<your_package_name>.R.layout.getui_notification"/>
    <path value="<your_package_name>.R.id.getui_*"/>
    <!-- 若您需要使用其他自定义推送图标，也需要在此处添加 -->
</issue>
```

4. 编写集成代码

4.1 初始化SDK

我们建议应用开发者在Activity或Service类中调用个推SDK的初始化方法，确保SDK在各种情况下都能正常运行。一般情况下可以在主Activity的onCreate()或者onResume()方法中调用，也可以在多个主要界面Activity的onCreate()或onResume()方法中调用。反复调用SDK初始化并不会有什么副作用。

- 在应用的 Activity 里导入 `PushManager` 类，如下所示：

```
import com.igexin.sdk.PushManager;
```

- 然后在 Activity 的onCreate()或者onResume()方法中调用个推SDK初始化方法。如果使用了自定义推送服务，初始化方法还需要传入新的自定义推送服务名：

```
// com.getui.demo.DemoPushService 为第三方自定义推送服务
PushManager.getInstance().initialize(getApplicationContext(), com.getui.demo.DemoPushService.class);
```

4.2 接收推送服务事件

从2.9.5.0版本开始，为了解决小概率发生的Android广播丢失问题，我们推荐应用开发者使用新的IntentService方式来接收推送服务事件（包括CID获取通知、透传消息通知等）

- 在项目源码中添加一个继承自com.igexin.sdk.GTIntentService的类，用于接收CID、透传消息以及其他推送服务事件。请参考下列代码实现各个事件回调方法：

```
package com.getui.demo;

import android.content.Context;
import android.os.Message;
import android.util.Log;

import com.igexin.sdk.GTIntentService;
import com.igexin.sdk.PushConsts;
import com.igexin.sdk.PushManager;
import com.igexin.sdk.message.FeedbackCmdMessage;
import com.igexin.sdk.message.GTCmdMessage;
import com.igexin.sdk.message.GTTransmitMessage;
import com.igexin.sdk.message.SetTagCmdMessage;

/**
 * 继承 GTIntentService 接收来自个推的消息，所有消息在线程中回调，如果注册了该服务，则务必要在 AndroidManifest中声明，否则无法接受消息<br>
 * onReceiveMessageData 处理透传消息<br>
 * onReceiveClientId 接收 cid <br>
 * onReceiveOnlineState cid 离线上线通知 <br>
 * onReceiveCommandResult 各种事件处理回执 <br>
 */
public class DemoIntentService extends GTIntentService {

    public DemoIntentService() {

    }

    @Override
    public void onReceiveServicePid(Context context, int pid) {
    }

    @Override
    public void onReceiveMessageData(Context context, GTTransmitMessage msg) {
        // 透传消息的处理，详看SDK demo
    }

    @Override
    public void onReceiveClientId(Context context, String clientid) {
        Log.e(TAG, "onReceiveClientId -> " + "clientid = " + clientid);
    }

    @Override
    public void onReceiveOnlineState(Context context, boolean online) {
    }

    @Override
    public void onReceiveCommandResult(Context context, GTCmdMessage cmdMessage) {
    }

    @Override
    public void onNotificationMessageArrived(Context context, GTNotificationMessage msg) {
    }

    @Override
    public void onNotificationMessageClicked(Context context, GTNotificationMessage msg) {
    }
}
```

- 在 `AndroidManifest.xml` 中配置上述 IntentService 类：

```
<!-- android:permission在2.13.1.0版本后必须配置 -->
<service
    android:name="com.getui.demo.DemoIntentService"
    android:permission="android.permission.BIND_JOB_SERVICE"/>
```

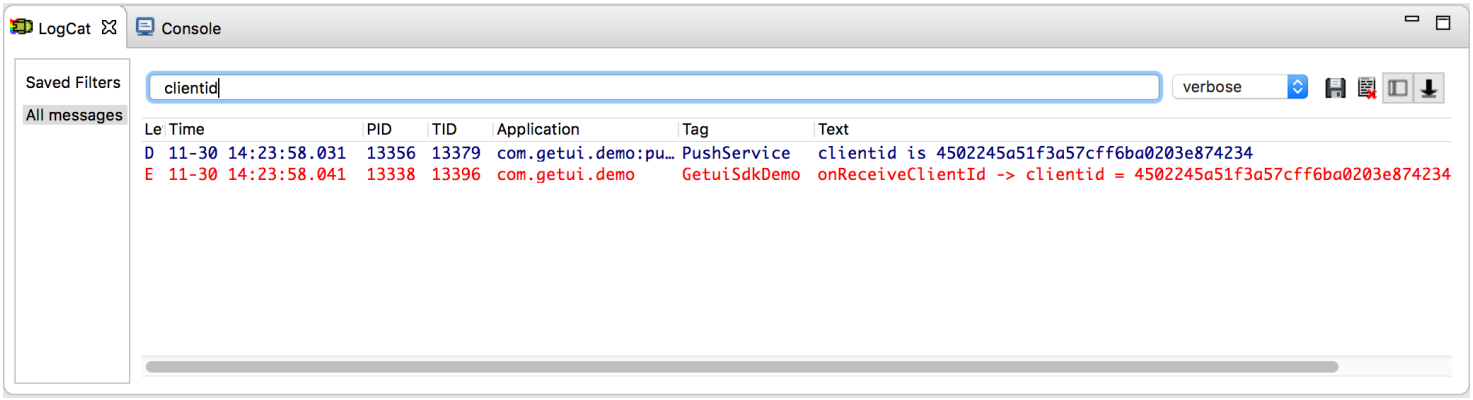
- 在个推SDK初始化后，注册上述 IntentService 类：

```
// com.getui.demo.DemoIntentService 为第三方自定义的推送服务事件接收类
PushManager.getInstance().registerPushIntentService(getApplicationContext(), com.getui.demo.DemoIntentService.class
);
```

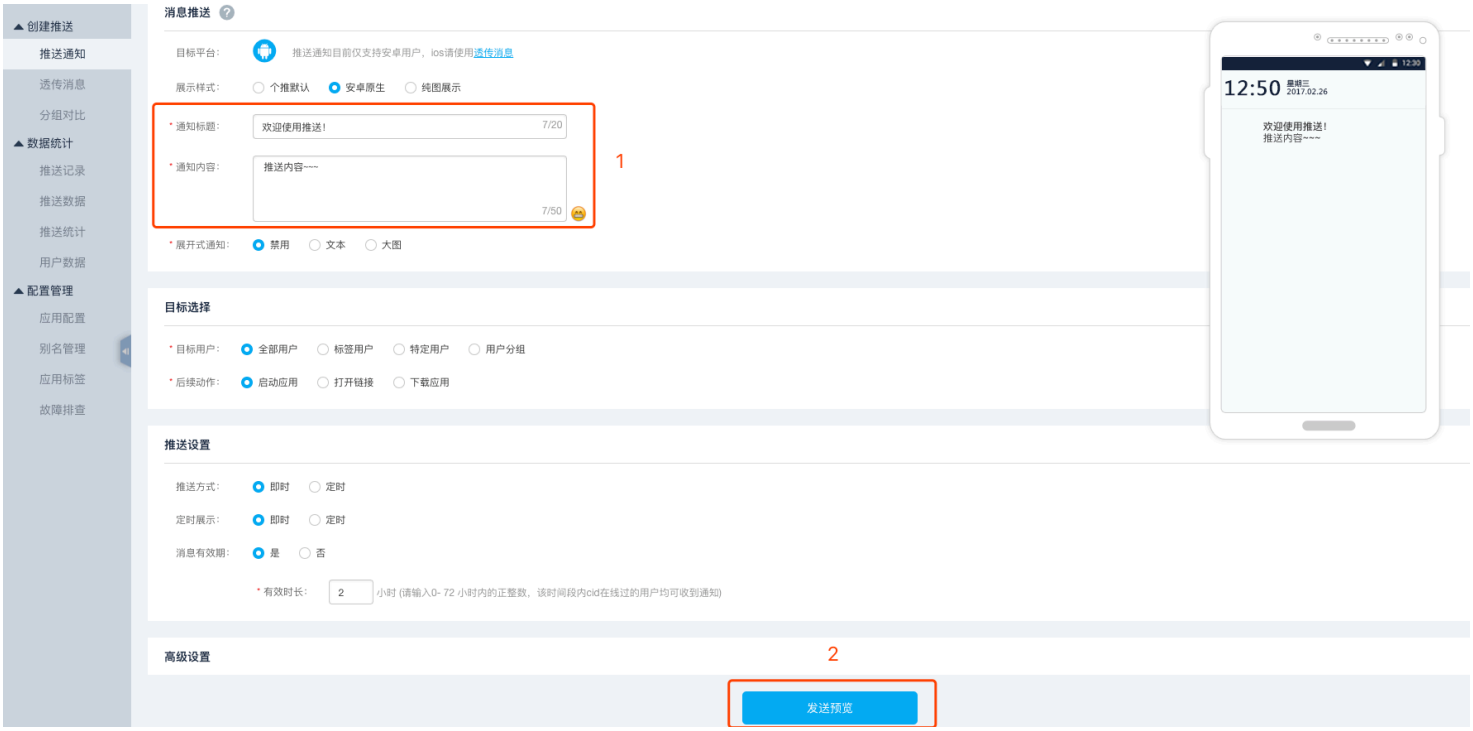
关于原有广播方式和新的IntentService方式兼容性说明：
1. 如果调用了registerPushIntentService方法注册自定义IntentService，则SDK仅通过IntentService回调推送服务事件；
2. 如果未调用registerPushIntentService方法进行注册，则原有的广播接收器仍然可以继续使用。

5. 测试

- 连接手机或启动Android模拟器，编译运行你的工程，查看logcat信息。在搜索框中输入 `clientid` ，如果能显示 `clientid is xxx` 日志，则说明个推SDK已经成功运行起来了：



- 登录 个推开发者中心，点击【创建推送】，进入待测试应用的推送通知界面：



- 依次填写 `通知标题` 和 `通知内容` ，点击 `发送` 按钮即可向该推送应用名下所有CID推送通知消息。具体推送操作方法详见：创建推送通知

- 如果手机或模拟器收到消息，显示下图所示通知，那么恭喜您，个推SDK接入测试已经成功完成！