宝贵建议请发送至：wangzhenyang@itcast.cn

黑马程序员 itheima.com -编程，始于黑马

# Android 课程同步笔记

Beta 0.01 版

By 阳哥

黑马程序员



Copyright©2015 黑马程序员 版权所有

# Android 手机卫士-11

# 简易屏幕适配&代码混淆&异常处理&广告平台

## 1. 简易屏幕适配（★★）

### 1.1 简易屏幕适配简介

由于市场上 Android 手机具备不同的分辨率和尺寸，因此手机适配是所有应用程序必须考虑的重要问题。关于屏幕适配会在智慧北京等以后的项目中做详细的介绍，这里介绍的屏幕适配仅仅是针对手机卫士里面使用的两种改进方法。

### 1.2 使用 ScrollView 解决屏幕适配问题

如果将我们手机卫士部署到一个 240*320 尺寸的手机上 那么我们发现设置中心已经无法将所有的控件显示完全。


因此为了防止控件个数比较多导致的显示不全现象，我们需要将这些控件放在 ScrollView 中，这样就算显示不完也可以进行滚动屏幕来显示。因此我们需要修改设置中心的布局文件。

修改也十分的简单，因为设置中心整体式个 LinearLayout，我们只需在 LinearLayout 外面套上 ScrollView 控件



http://www.itheima.com Copyright©2015 黑马程序员版权所有

即可。

Tips：ScrollView 控件只允许有一个子节点，因此如果我们在使用 ScrollView 的时候需要包含多个子节点，那么我们可以将这些多个子节点包含在一个 LinearLayout 等布局中。

## 1.3 dip 在不同分辨率手机下的适配

将我们的程序分别部署到如下三种（240*320,320*480,480*800）分辨率的模拟器上，出现如下不同效果。


显示弹出层的代码如下片段：

```
//创建一个新的 popupWindow 对象并在指定位置显示
popupWindow = new PopupWindow(layout, -2, -2);
popupWindow.setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
int[] location = new int[2];
view.getLocationInWindow(location);
//相对于一个父元素显示的位置，第三个参数是 pix
popupWindow.showAtLocation(parent, Gravity.TOP + Gravity.LEFT, 60, location[1]);
```

显然是 popupWindow 的 showAtLocation 中的参数 60（距离父控件左侧距离，单位 pix）导致的。

解决上诉问题可以将 pix 单位装换为 dip 单位。在我们的工程中添加 DensityUtil 类，用于 dip 和 pix 之间的转换。

```
public class DensityUtil {
    public static int dip2px(Context context, float dipValue) {
        final float scale = context.getResources().getDisplayMetrics().density;
        return (int) (dipValue * scale + 0.5f);
    }
```


http://www.itheima.com Copyright©2015 黑马程序员版权所有

```
    public static int px2dip(Context context, float pxValue) {
        final float scale = context.getResources().getDisplayMetrics().density;
        return (int) (pxValue / scale + 0.5f);
    }
}
```

对上面代码进行如下修改，定义 dip=60，然后转换成 pix 单位的数值传递给 showAtLocation 方法。

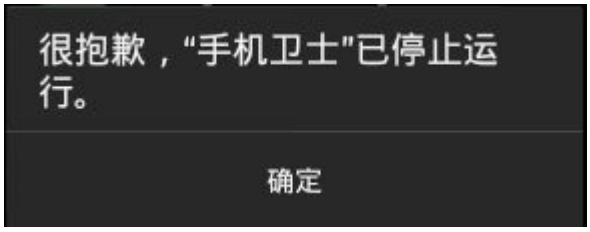
```
//创建一个新的 popupWindow 对象并在指定位置显示
popupWindow = new PopupWindow(layout, -2, -2);
popupWindow.setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
int[] location = new int[2];
view.getLocationInWindow(location);
//相对于一个父元素显示的位置，第三个参数是 pix
int dip = 60;
int pix = DensityUtil.dip2px(AppManagerActivity.this, 60);
popupWindow.showAtLocation(parent, Gravity.TOP + Gravity.LEFT, dip, location[1]);
```

修改完上面代码后重新将新的 apk 部署到三种分辨率的模拟器上，发现适配问题已经解决。

# 2. 异常处理（★★）

## 2.1 Application 异常捕获

如果我们的应用程序中有未捕获的 bug，那么运行时系统会出现如下对话框。




这种错误对于 Android 用户来说是毫无意义的，因此一方面我们尽量避免这类异常的发生，另一方面当此类异常发生后我们也应该有相应的处理手段（比如重启应用程序）。

解决方法其实很简单，我们可以自定义一个 Application 类，继承 Android 的 Application 类，覆写该类的 onCreate 方法，在该方法中给应用线程设置一个未捕获异常处理器。然后将 AndroidManifest.xml 中的 Application


Copyright©2015 黑马程序员版权所有

指定成我们自定义的 Application 即可。

1）在 activity 包中定义如下类继承 Application，

com.itheima.mobileSafe.activity.MobileGuardApplication，代码清单如下：

```
/**
 * Base class for those who need to maintain global application state. You can
 * provide your own implementation by specifying its name in your
 * AndroidManifest.xml's <application> tag, which will cause that class to be
 * instantiated for you when the process for your application/package is
 * created.
 *
 * There is normally no need to subclass Application. In most situation, static
 * singletons can provide the same functionality in a more modular way. If your
 * singleton needs a global context (for example to register broadcast
 * receivers), the function to retrieve it can be given a
 * android.content.Context which internally uses Context.getApplicationContext()
 * when first constructing the singleton.
 *
 * @author wzy Feb 23, 2015
 *
 */
public class MobileGuardApplication extends Application {

    @Override
    public void onCreate() {
        Thread.currentThread().setUncaughtExceptionHandler(new
MyUncaughtExceptionHandler());
        super.onCreate();
    }

    private class MyUncaughtExceptionHandler implements UncaughtExceptionHandler {

        @Override
        public void uncaughtException(Thread thread, Throwable ex) {
            System.out.println("捕获到未处理异常： " + ex);
            //杀死自己程序 然后程序会自动重启
            android.os.Process.killProcess(android.os.Process.myPid());
        }
    }
}
```



 Copyright©2015 黑马程序员版权所有

2）在 AndroidManifest.xml 中指定自定义的 Application

```
<application
      android:name="com.itheima.mobileSafe.activity.MobileGuardApplication"
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
```

重新部署程序，当系统再次遇到为捕获异常的时候，发现程序自定退出并重新启动。效果类似重启。

## 2.2 应用程序错误日志捕获

在上一节中，我们成功捕获到异常，但是捕获到异常以后我们并没有对异常进行处理。一般情况下，在企业中我们会将捕获到的异常保存到日志文件中，然后在 Splash 界面初始化的时候上传到后台服务器中。

这里我将演示的是如何将日志保存到本地日志文件中。

修改 2.1 节中 MobileGuardApplication 类中的 MyUncaughtExceptionHandler 类的 uncaughtException 方法，在该方法中将用户的操作系统和错误信息保存到日志文件中。

```
public void uncaughtException(Thread thread, Throwable ex) {
          try {
              StringWriter sw = new StringWriter();
              PrintWriter pw = new PrintWriter(sw);
              System.out.println("捕获到未处理异常: " + ex);
              Field[] fields = Build.class.getDeclaredFields();
              for (int i = 0; i < fields.length; i++) {
                  Field field = fields[i];
                  sw.write(field.getName() + "--" + field.get(null) + "\n");
              }
              ex.printStackTrace(pw);
              File file = new
File(Environment.getExternalStorageDirectory(),"log.txt");
              FileOutputStream fos = new FileOutputStream(file);
              fos.write(sw.toString().getBytes());
              fos.close();
              pw.close();
              sw.close();
```



http://www.itheima.com Copyright©2015 黑马程序员版权所有

```
// 杀死自己程序 然后程序会自动重启
            android.os.Process.killProcess(android.os.Process.myPid());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

# 3. 广告平台（★）

## 3.1 广告平台简介

广告是目前大多数 app 主要的盈利渠道之一。在我们的的应用中插入一部分广告会给我们带来一定的利润收入。

广告平台是一个平台或者中介，连接着应用开发者和广告主。在平台上，开发者提供应用，广告主提供广告，而移动广告平台会提供相应手机系统的 SDK。因此我们使用哪个广告平台首先需要在其网站进行注册，然后下载其 SDK。然后将其 SDK 内置到我们的代码中。

app 盈利模式主要有以下三种：

- 1）付费下载（在国内基本不流行）
- 2）IAP（internet 接入服务商（Internet Access Provider，IAP ））），比如在玩手机游戏时闯关需要购买一些钻石、道具等
- 3）免费+广告（很流行，也是很多软件采用的盈利模式）

常见广告平台（太多了，只列出其中的几种）：

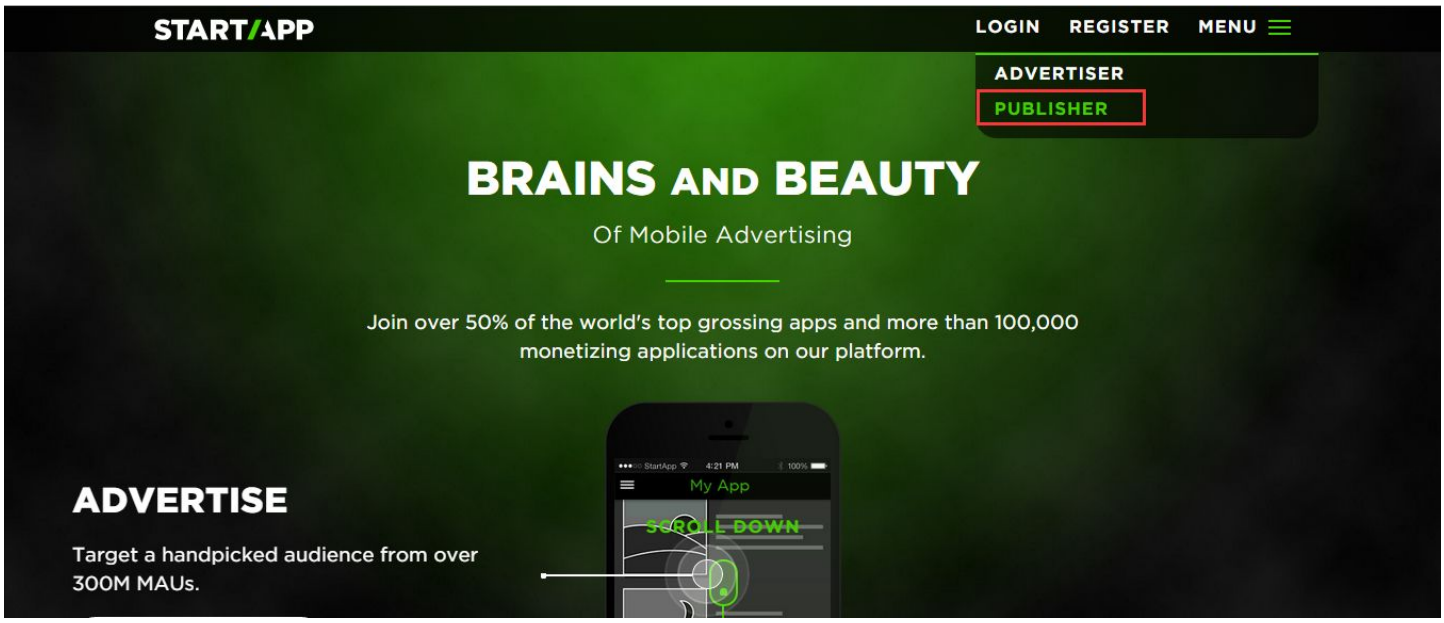
- 百度移动联盟 http://union.baidu.com/customerLogin.html
- 有米 http://www.youmi.net/
- 万普世纪 http://www.waps.cn/
- 易积分 http://www.yijifen.com/
- StartApp（国外） http://www.startapp.com/


Copyright©2015 黑马程序员版权所有

# 3.2 广告平台使用示例

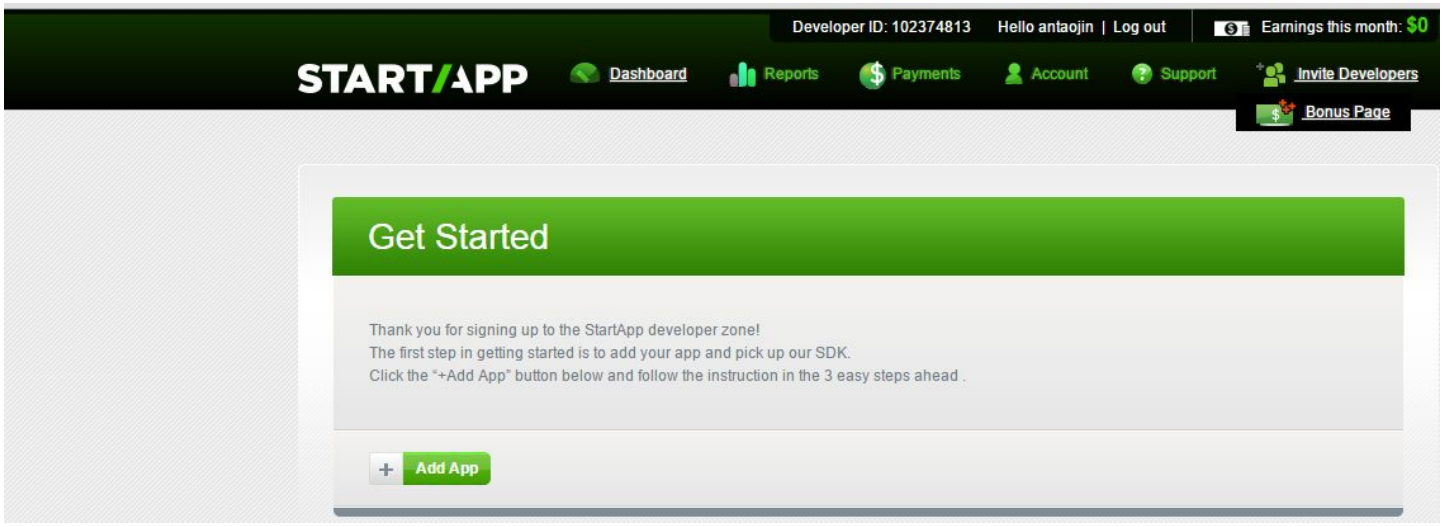
广告平台有很多，但是使用都大同小异，这里以 StartApp 为例进行广告平台的使用示例。

1 登录 StartApp，进行注册 http://www.startapp.com/




如上图所示，我们选择 REGISTER 的 PUBLISHER 进行注册。

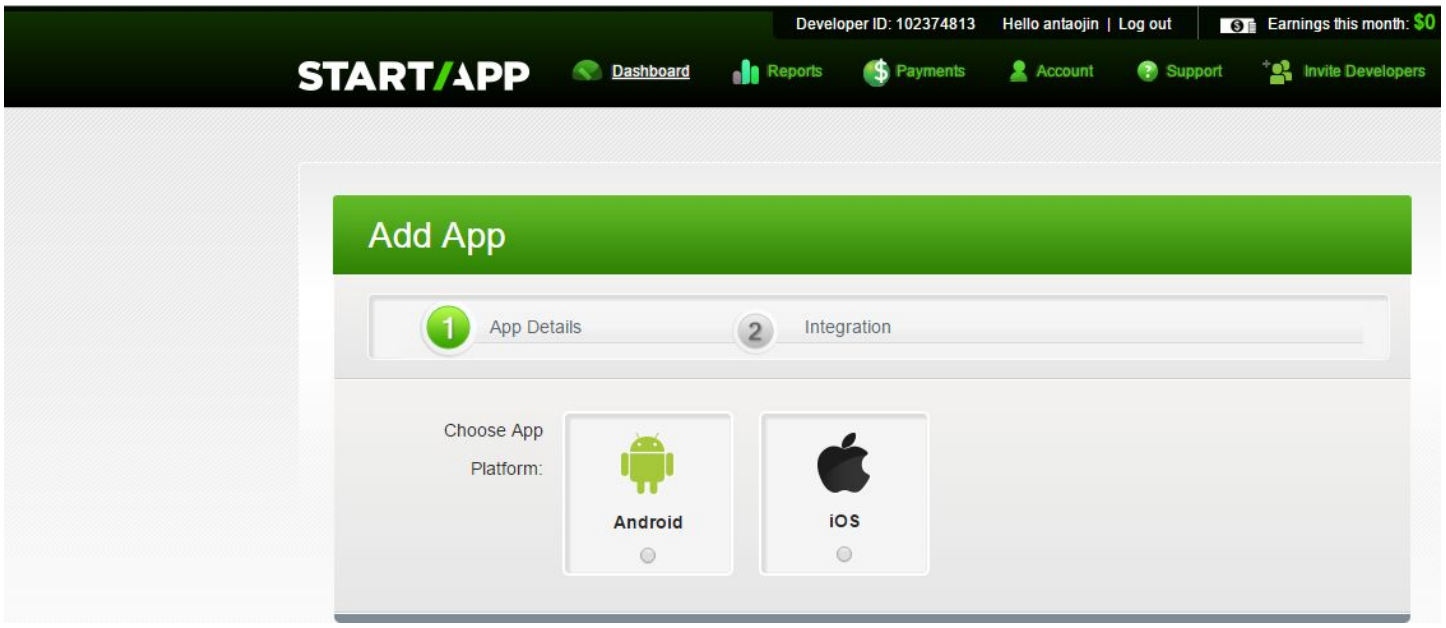
注册好以后如下界面，点击 Add App。




进入如下界面：

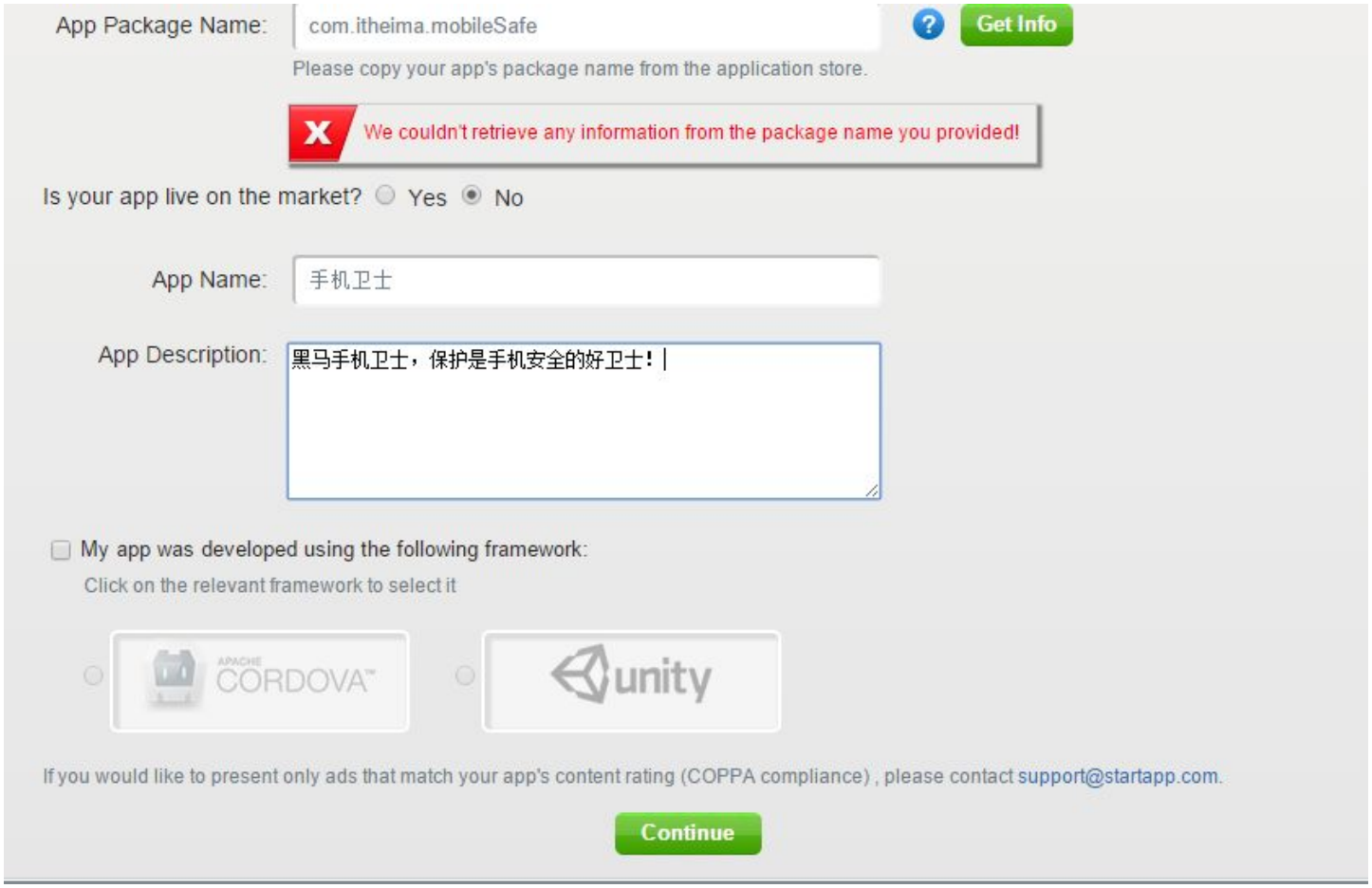


http://www.itheima.com Copyright©2015 黑马程序员版权所有




我们选择 Android 平台。

然后在如下界面输入如下信息：

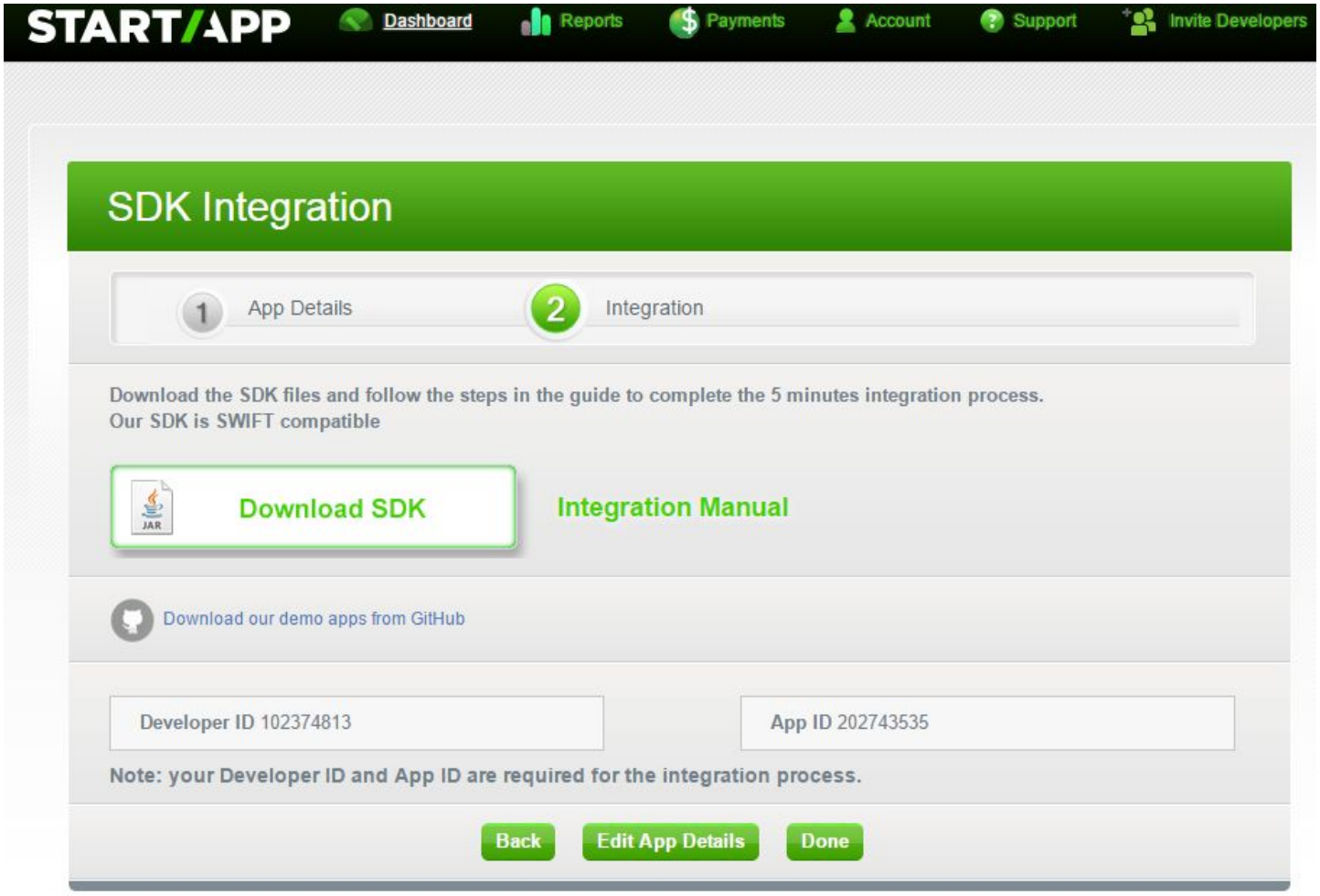



点击 Continue，进入下面界面：



http://www.itheima.com Copyright©2015 黑马程序员版权所有

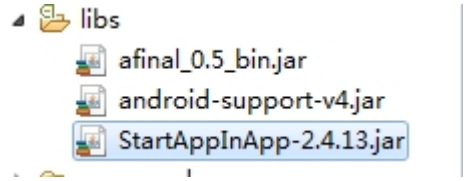
点击 Download SDK，下载 SDK。




下载好 SDK 后我们并不知道如何使用，那么点击上图中的 Integration Manual，进入一个帮助文档界面，为了方便使用，我们可以使用 Ctrl+S 将该网页保存到本地，方便以查看。

2 Adding the SDK JAR to Your Eclipse Project

将我们下载的 SDK 拷贝到 libs 目录中。




3 Updating Your AndroidManifest.xml File

更新我们的 AndroidManifest.xml 文件。

1）在清单文件中，添加如下权限（如果没有）：



http://www.itheima.com Copyright©2015 黑马程序员版权所有

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

2）在清单文件中，添加如下 Activity：

```
<activity android:name="com.startapp.android.publish.list3d.List3DActivity"
          android:theme="@android:style/Theme" />
<activity android:name="com.startapp.android.publish.AppWallActivity"
          android:theme="@android:style/Theme.Translucent"
          android:configChanges="orientation|keyboardHidden|screenSize" />
```

4 Initialization，This is a mandatory step

进行初始化操作，这一步是必须的，不然是无法使用。

在我们要内置广告的 Activity 的 onCreate 方法中进行广告平台的初始化。

```
StartAppSDK.init(this, "Your Developer Id", "Your App ID", true);
```

将上面的 Developer Id 和 App ID 替换成我们自己的，然后我们把这段代码放到 HomeActivity 中。

```
StartAppSDK.init(this, "102374813", "202743535", true);
```

5 在布局文件中添加控件

因为我们的广告显示在控件中，因此需要在要显示广告的地方添加如下控件。

```
 <com.startapp.android.publish.banner.Banner
          android:id="@+id/startAppBanner"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_centerHorizontal="true"/>
```

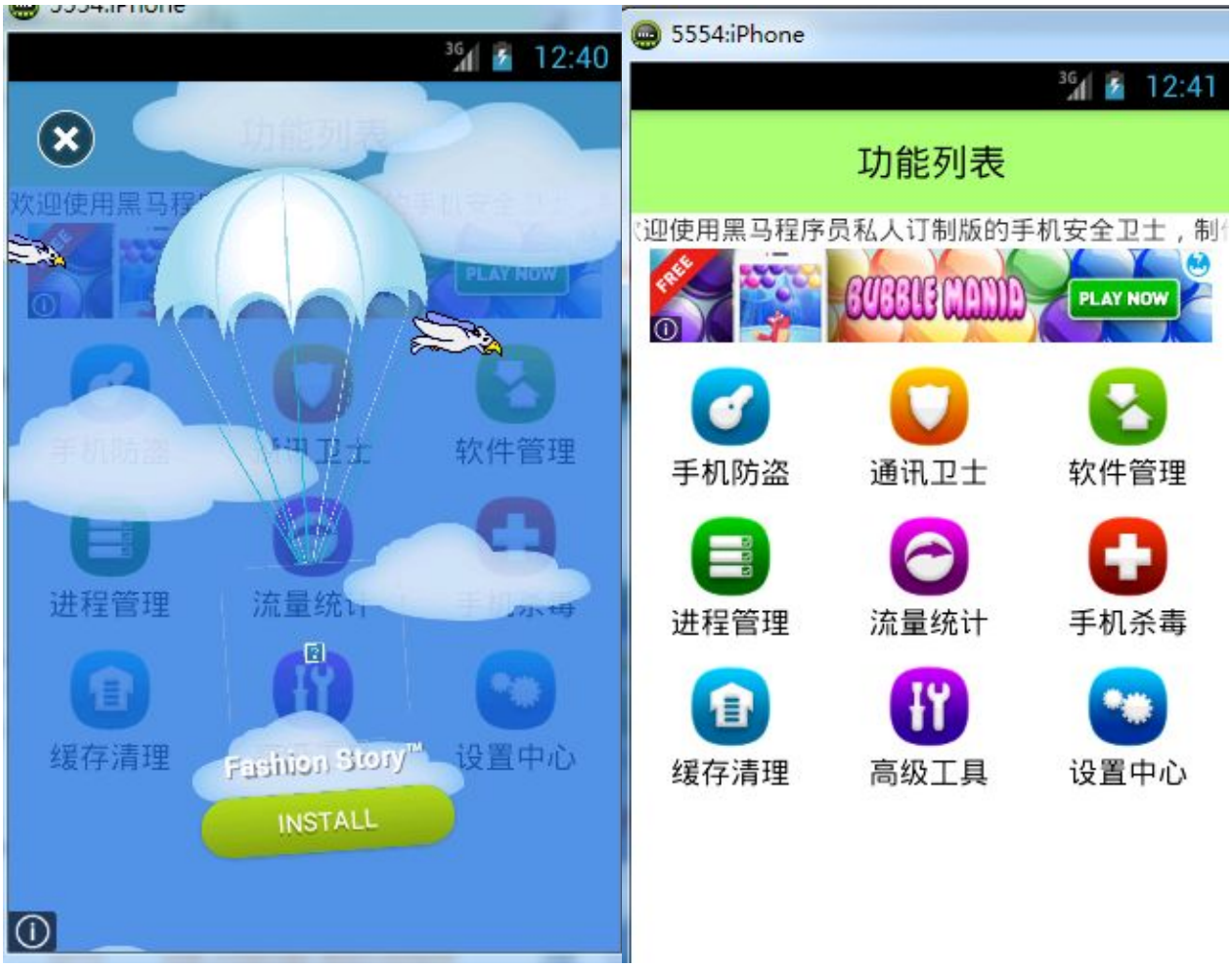
6 广告形式有多种，我们这里使用其推荐的形式，Splash 模式，在 onCreate 方法中添加如下代码即可。

```
StartAppAd.showSplash(this, savedInstanceState);
```

7 重新将我们的应用部署到模拟器上，同时保证我们的模拟器能链接到网络，启动以后，发现广告来了，效果


http://www.itheima.com Copyright©2015 黑马程序员版权所有

图如下：



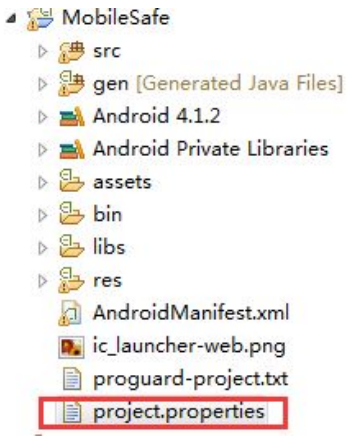

# 4. 代码混淆（★★）

代码混淆可以在一定程度上加强应用程序的反编译难度，就算被反编译也不容易被看懂，因此我们的劳动成果得以保护。

代码混淆是通过 ProGuard 来实现，其原理就是动态的替换包名、类名、方法名。代码混淆功能是 SDK 内置实现的。

- 1）在工程中修改文件（project.properties）参数：


http://www.itheima.com Copyright©2015 黑马程序员版权所有




将

```
#proguard.config=${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt
```

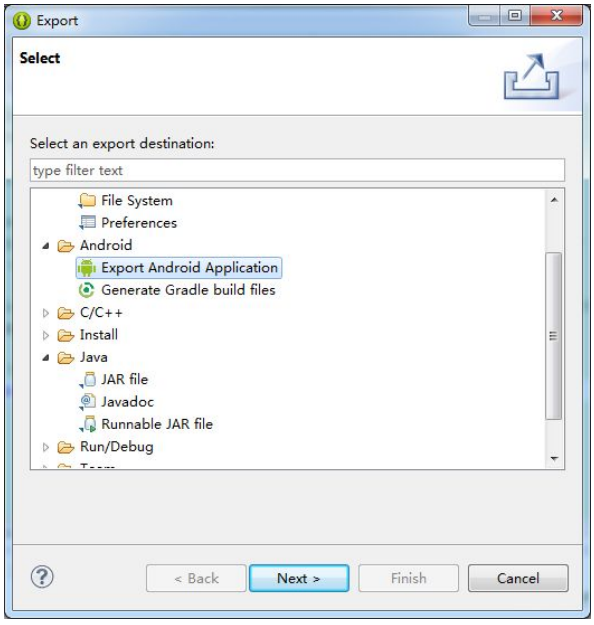
改为

```
proguard.config=proguard-android.txt
```

Tips：这一步其实并不是必须的，但是由于 eclipse 可能存在未知的 bug 导致找不到`${sdk.dir}`（sdk 根目录）目录，因此我们一般会将 sdk 中的 `proguard-android.txt` 文件直接拷贝到工程目录中。

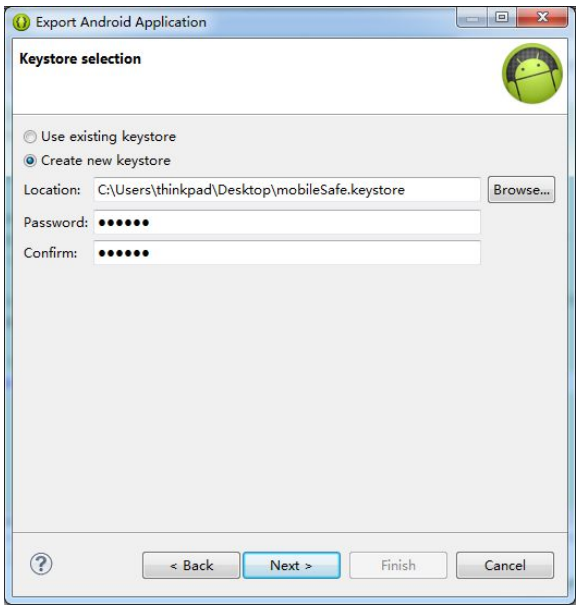
2）将`${sdk.dir}/tools/proguard` 目录下的 `proguard-android.txt` 文件拷贝到工程根目录中。

3）导出 APK，eclipse->Export，弹出如下界面，选择 Export Android Application

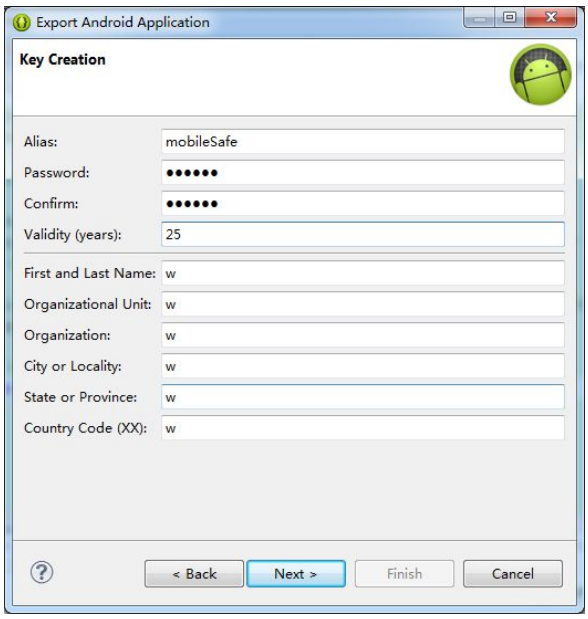



点击 Next 进入下一步，如下图所示，选择 Create new keystore


Copyright©2015 黑马程序员版权所有



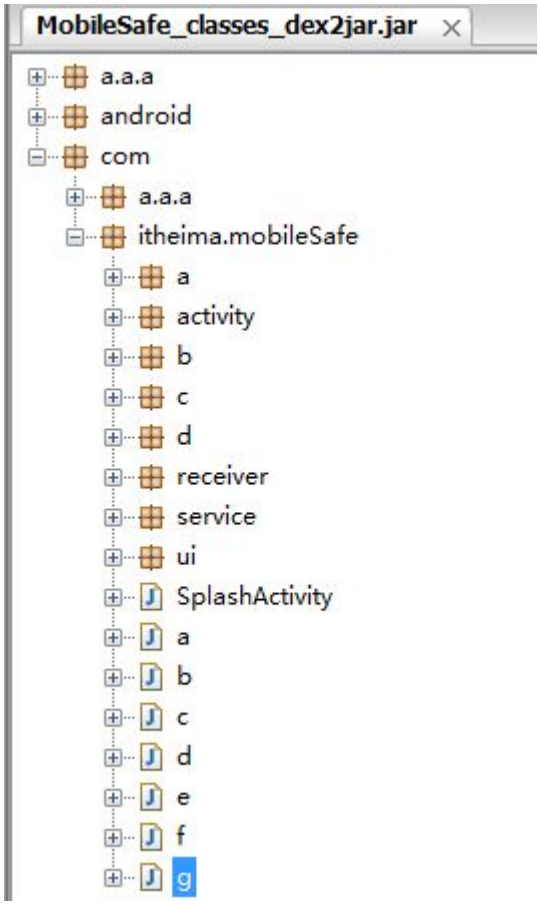

输入密码并确认，然后 Next，进入如下界面。




填入相应的信息，然后 Next，进入最后一个界面，该界面比较简单，不再给出截图。在该界面中选择将生成的 apk 保存路径，然后点击 finish 即可。

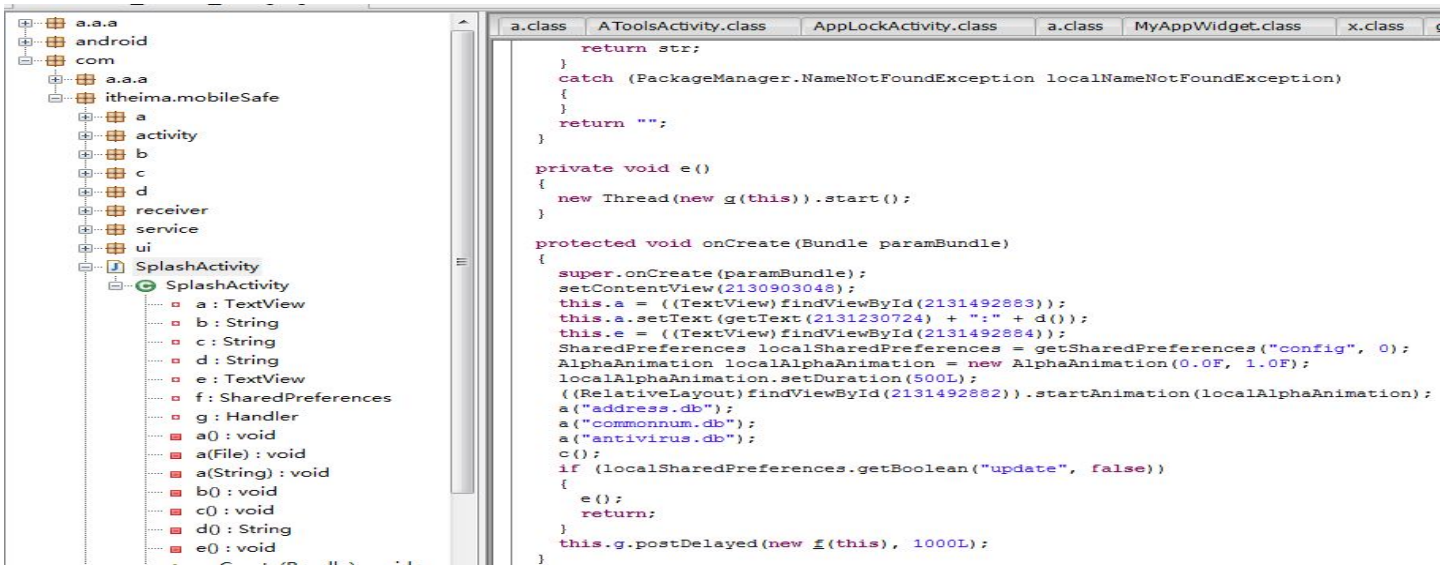
这时我们的 apk 已经实现了代码混淆，为了进行验证，我们使用逆向助手进行反编译，查看反编译以后的代码状况。用逆向助手提取 dex 文件，然后将 dex 装换成 jar 文件，用 jd 工具打开 jar 如下图所示：


http://www.itheima.com Copyright©2015 黑马程序员版权所有




发现除了四大组件（因为需要在 AndroidManifest.xml 中注册）名字没有被替换外，其他类名以及包名已经被替换。

为了进一步观察结果，我们打开 SplashActivity，如下图所示：




发现其中的方法名和变量名也已经被修改。




Copyright©2015 黑马程序员版权所有

***至此，本文档完！***

2015 年 2 月 23 日 星期一 20:45:10

河南省济源市梨林镇



http://www.itheima.com Copyright©2015 黑马程序员版权所有