#3手机应用开发入门

课程简介

课程目标

- Android开发基本概念
- Android 开发环境及相关工具
- 创建HelloWorld
- 命令行运行

Android开发基本概念

- Activities （活动）
- Intents（意图）
- 没有鼠标光标的操作（触摸屏）
- 视图与控件（界面元素）
- 异步调用（多线程支持）
- 后台服务

Activities （活动）

- Android应用由一个或多个Activities组成
- Activities是一个容器，装着你的UI，以及运行UI的代码
- 相当于Windows程序的一个窗口

Intents（意图）

- 用于构成Android的核心消息系统
- 包含要执行的动作（例如：查看、编辑、拨号等）及相关数据（例如：一条联系人信息）
- 用于启动Activity，Android各部件之间通信
- 用户应用可以广播Intents，也可以接收Intents

使用Intents发送消息

- 当广播一个Intent的时候，其实就是告诉Android系统有事情发生了
- 可以告诉当前应用新建一个窗口(Activity)
- 也可以让Android启动另外一个应用

注册Intent接收器

- 必须注册Intent接收器，以监听Intent消息，然后处理之：新建窗口，打开其他应用或者其他动作
- 一个Intent消息可以被多个接收器接收，此时Android系统将弹出选择窗口
- 注意：如果Android没有找到匹配的接收器，发送Intent的应用将会崩溃！

触摸屏操作

- 手指 vs. 鼠标！
- 劣势
  - 鼠标右键怎么实现？长按！
  - 不够精确？应用设计

- 优势
  - 自然
  - 多个手指！
  - 支持手势！

视图与控件（界面元素）

- 视图（Views）
  - 基本的界面元素
  - 屏幕中的一块矩形区域
  - 可响应画图和事件处理
  - 例如：菜单元素
- 控件（Widgets）
  - 高级界面元素
  - 例如：按钮、多选框等等

异步调用

- Android提供了AsyncTask类
- 可很简单的实现多任务
- 不需要自行回收多线程资源
- 而且调用者很容易获得线程运行结果
- 这是一个很简洁、清晰的异步编程模型
- 耗时操作不使用异步调用的后果
  ANR(应用程序无响应)

后台服务

- 类似Windows系统的服务：没有界面的运行方式
- 例如：Android系统里音乐播放器通常提供后台服务的运行方式
  - 可以边收邮件边听歌

Android提供的硬件工具

- Android设备会告诉你：
- Where am I?
- Which way am I walking?
- Is my phone facing up or down?
- Is my phone moving?
- Can I use my Bluetooth headphones?
- How do I record video?

- GPS，罗盘，距离传感器，加速计，蓝牙，相机等

Android提供的软件工具

- Internet： WebOS?
- 音视频支持: 各种格式
- 联系人: 可自由访问，与其他应用组合
- 安全：安装应用时有权限提示
- Google APIs:　支持位置与导航

我们对Android
有了较全面的了解
之后……

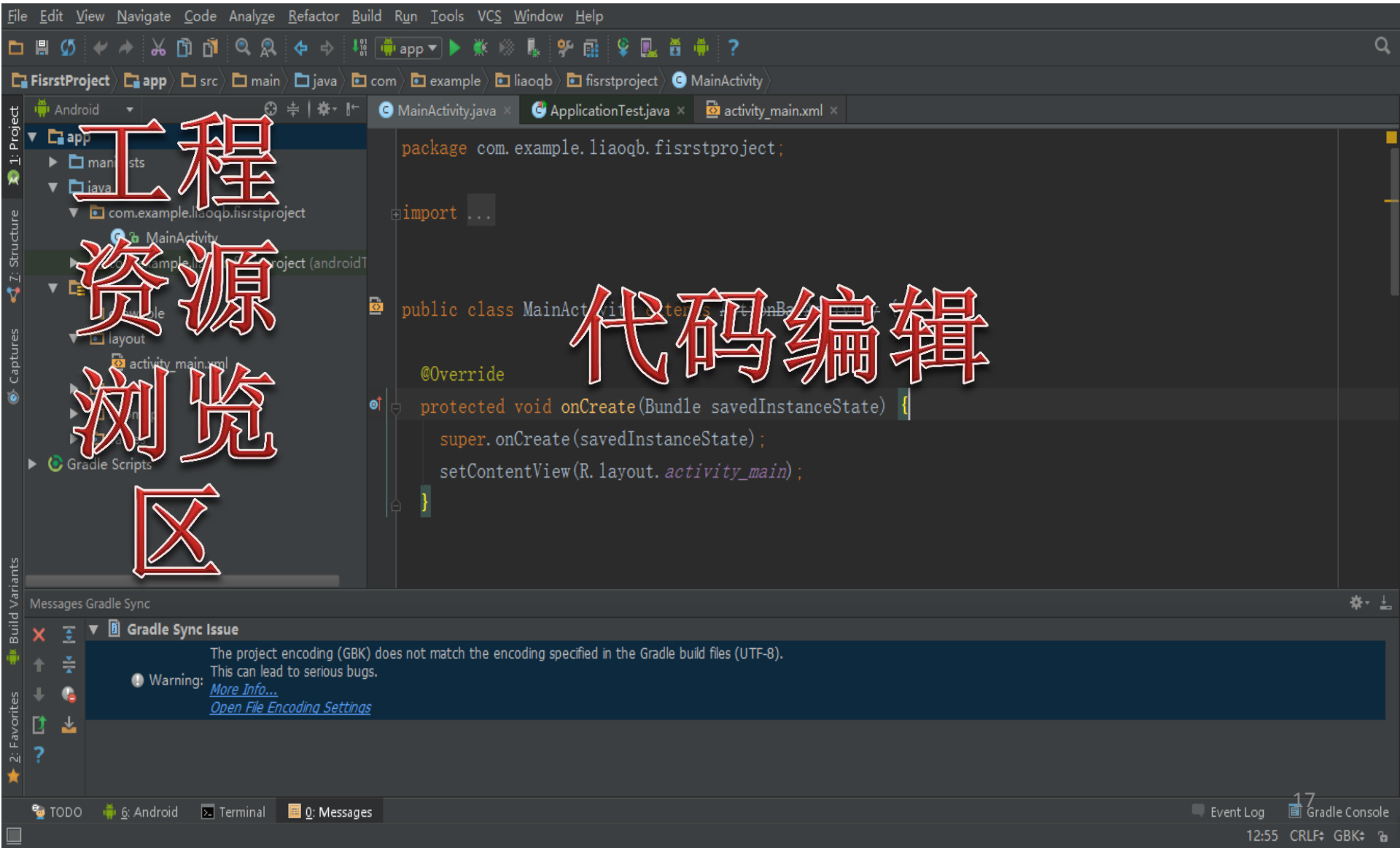
开发前要记住的原则

- KISS： Keep It Simple, Stupid
- 在没有理解内建API之前，不要一头栽进去写代码！
  - 先看文档！不用记住，但一定要看！
  - 写N行代码=一句API调用!
- 不要增加不需要的功能！
  - 使用内建控件就可以完成任何事情

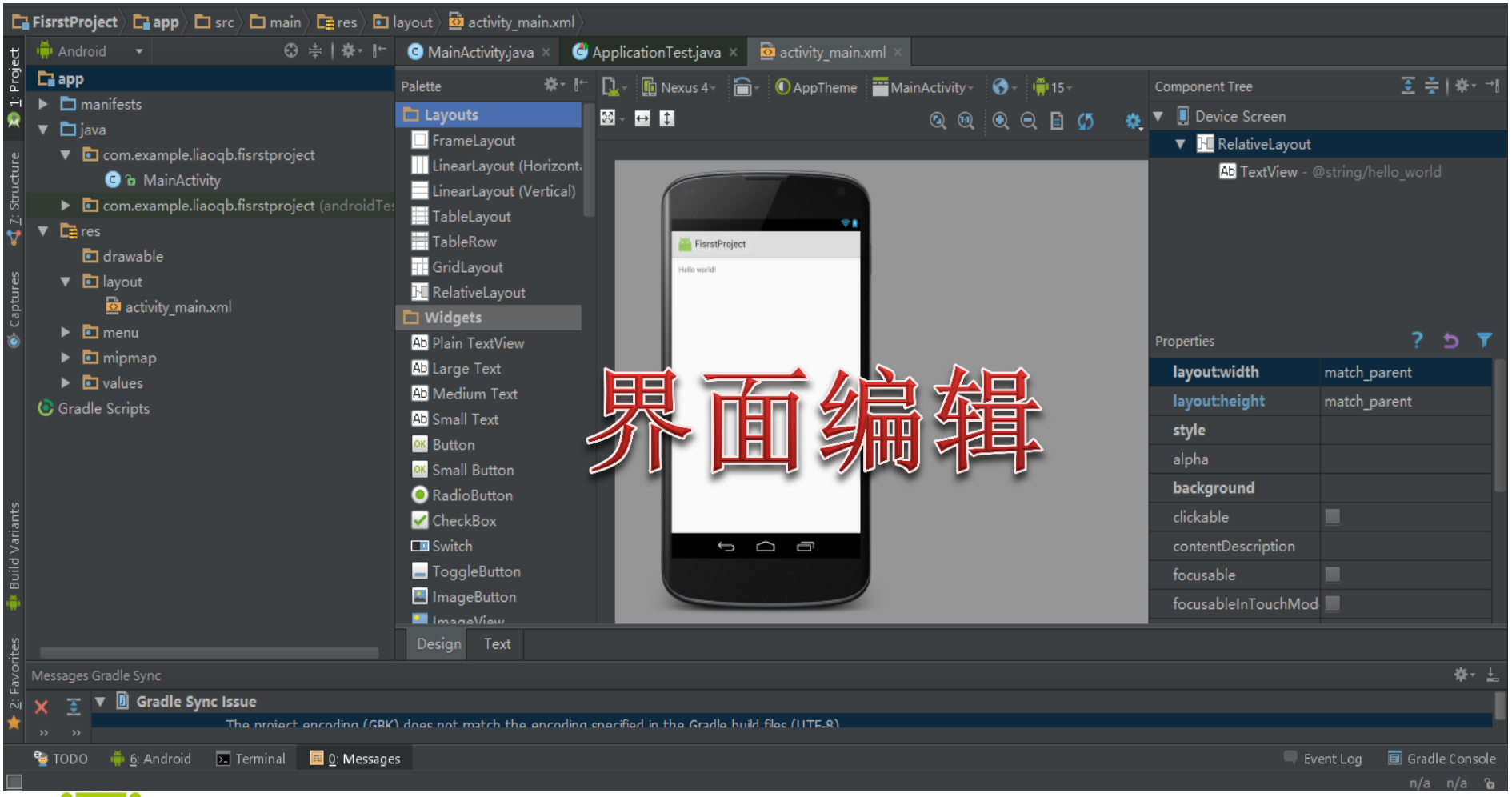
开发环境

- Android Studio
  https://developer.android.google.cn/studio/install.html
  - Android Studio
  - 最新Android SDK
  - 绿色，直接安装即可

The IDE（AS） for Android



The IDE（AS） for Android



模拟器

Android SDK的内容

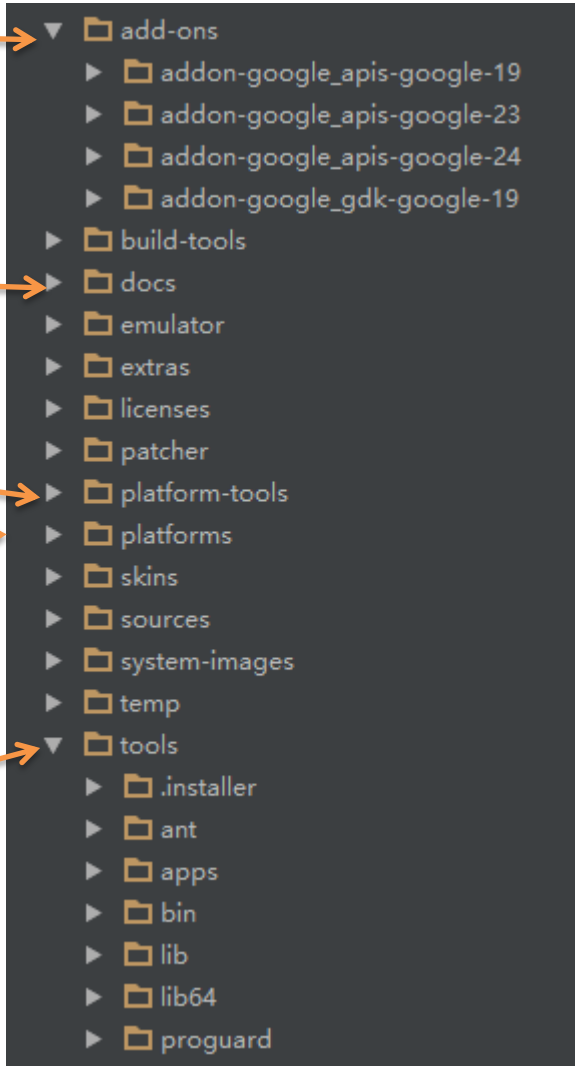
附加库，如：Google Map的API

相关技术文档

Android平台相关通用工具，如：adb

不同版本的Android

SDK提供的工具，如：DDMS

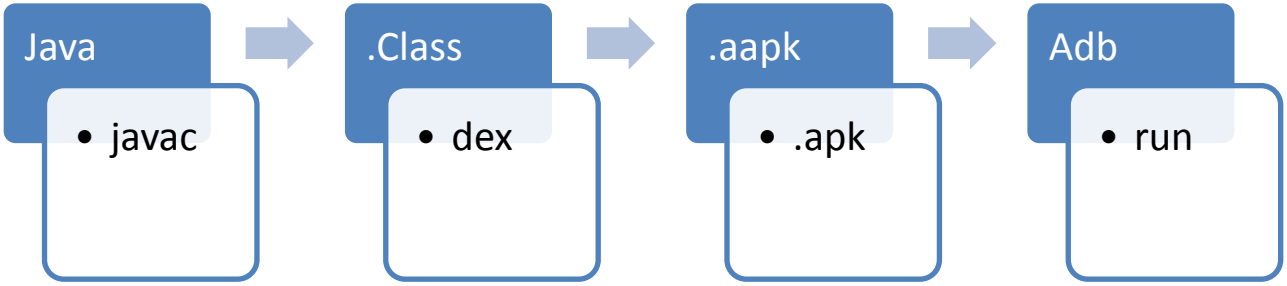


Android SDK提供的工具

- Android模拟器（Android Emulator）
- 集成开发环境插件（ADT）
- 调试监视服务（Dalvik Debug Monitor Service）
- Android 调试桥（Android Debug Bridge）
- Android资源打包工具（aapt.exe）
- Android虚拟设备（Android Virtual Devices）
- ……

Android SDK提供的工具

- ADT: Android Development Tool, an Eclipe plugin
- Two debuggers
  - *adb*: Android Debug Bridge
  - *ddms*: Dalvik Debug Monitor Server
- *aapk*: Android Application package tool
  - All resources are bundled into an archive, called *apk* file.
- *dx*: java byte code to Dalvik executable translator
- Android *emulator*: QEMU



Android的相关文件类型

- Java文件：应用程序源文件
  - android 本身相当一部分都是用java 编写而成
  - android 的应用使用java 来开发。
- Class文件：Java编译后的目标文件
  - Google使用Dalvik 来运行应用程序
  - Android的class 文件是编译过程中的中间目标文件，需要链接成dex 文件才能在Dalvik 上运行。

Dex文件

- Dex文件：Android平台上的可执行文件
  - Android 虚拟机Dalvik 支持的字节码文件格式。
  - 这种虚拟机执行的并非Java 字节码，而是另一种字节码：dex 格式的字节码。
  - 在编译Java 代码之后，通过Android 平台上的工具可以将Java 字节码转换成Dex 字节码。
  - Dalvik VM 针对手机程序与CPU进行过优化，可以同时执行许多VM 而不会占用太多资源。

Apk文件

Apk文件: Android上的安装文件

- Apk 是Android 安装包的扩展名，一个Android 安装包包含了与该Android 应用程序相关的所有文件。
- apk文件将AndroidManifest.xml文件、应用程序代码(.dex 文件)、资源文件和其他文件打成一个压缩包。
- 一个工程打包成一个.apk文件。
- apk 文件的本质是一个zip包。(可以理解为后缀名修改为.apk)

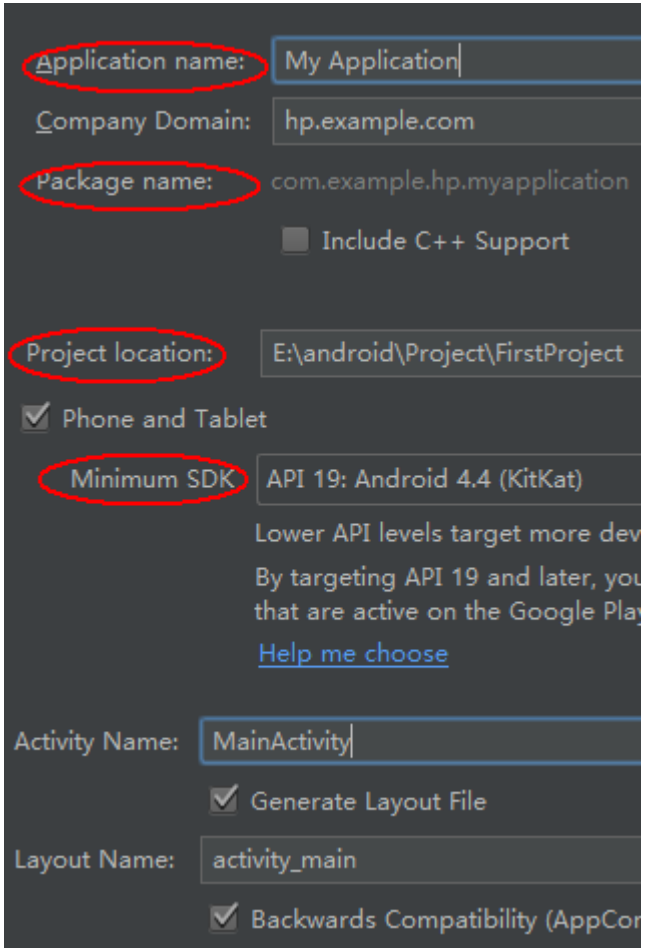
可以先试试SDK的例子程序

http://developer.android.com/resources/samples/

新建Android项目(Project)

Android项目的相关概念

- Application name ： 应用名，应用程序的名称； 最终显示在模拟器上。
- Package name ： 包名， 见Java相关概念
- Project location :在计算机中存储工程的路径;
- Minimum SDK ： 最低SDK版本,如果写2的话,就代表包括1.1 和1.1以上版本的SDK都能运行,写3的话1.1的平台就不能运行了,最终体现在xml申明文件里。
- Activity Name —— UI界面窗口的类名，从Activity继承而来； Activity是一个应用程序的基础，通常是Android Activity的子类。

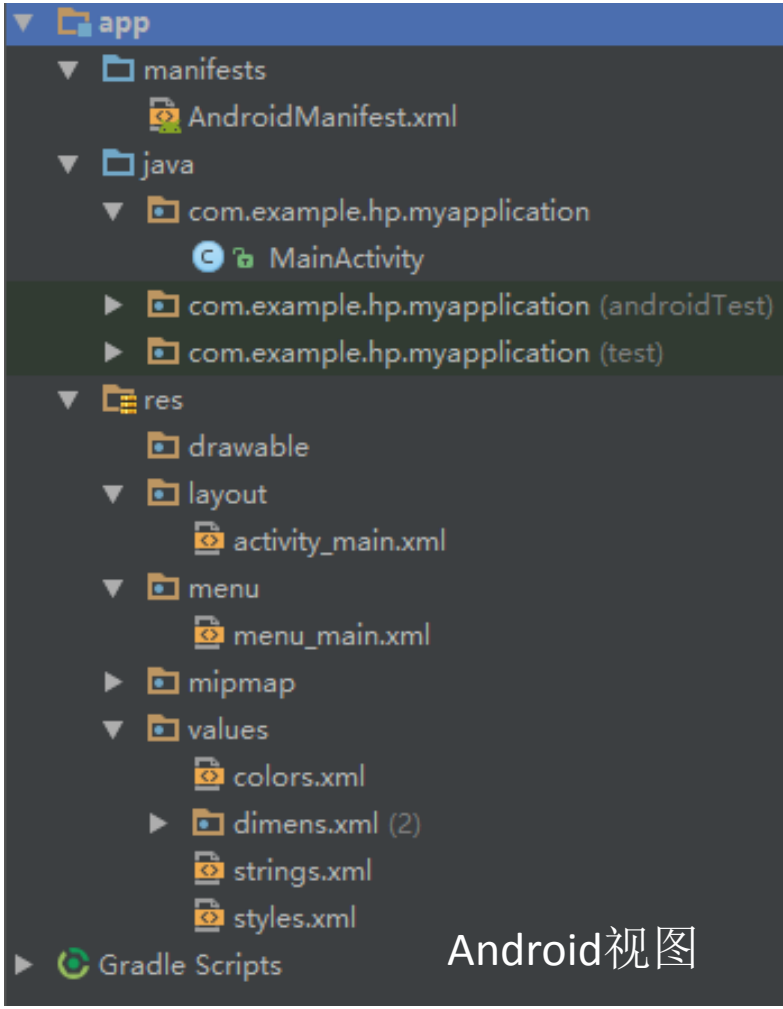


Package是什么？

- package好比java用来组织文件的一种虚拟文件系统。
- package把源代码.java文件，.class文件和其他文件有条理的进行一个组织，以供java来使用。
- package是将文件组织在一颗类似unix，linux文件系统的树结构里面，它有一个根"/"，然后从根开始有目录和文件，目录中也还有文件和目录。

Android应用工程文件组成（1）

- AndroidManifest.xml文件
  - 非常重要，相当于应用的配置文件，声明应用的名称、应用所用到的Activity、Service和receiver等
- Java目录
  - 最重要的，我们的java代码所在的位置
- Res目录
  - 主要放置应用用到的资源文件，分三个目录存放，当目录中的资源文件发生变化时，R文件就会自动发生变化。
  - drawable目录---主要放置图片资源
  - layout目录---主要放置用到的布局文件
  - values目录---主要放置字符串（strings.xml）、颜色(colors.xml)、样式（styles.xml）、尺寸（dimens.xml）
- Asserts目录（默认不创建）
  - 主要放置多媒体等一些文件；

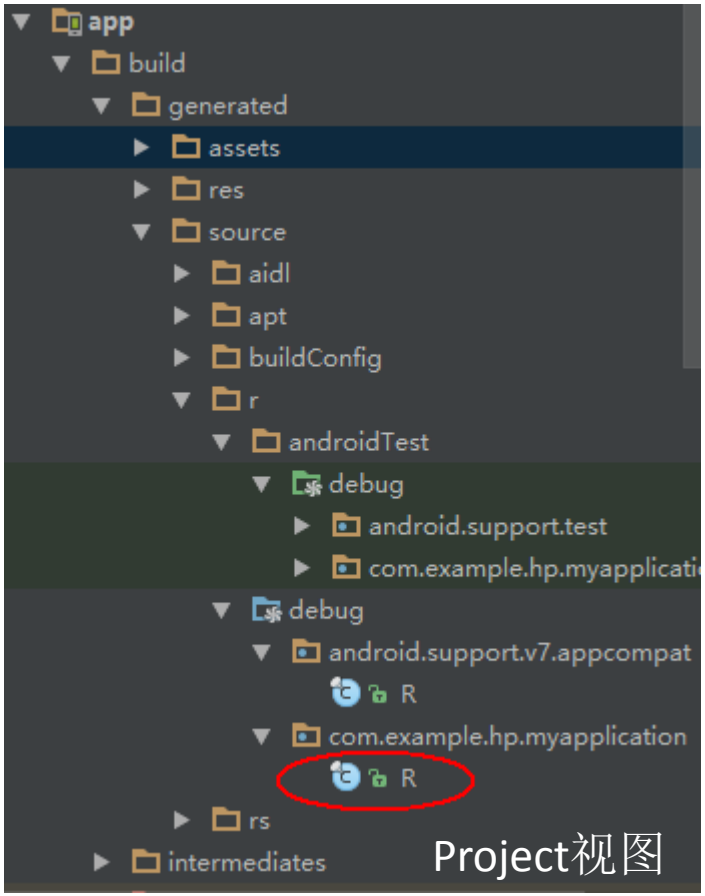


Android视图

Android应用工程文件组成（2）

- build目录下的R.java：对将要用到的资源进行全局索引
  - 自动生成，只读模式，由Android Studio自动来处理
  - Res文件夹中发生任何变化，R.Java都会重新生成

```
public final class R {
    ……
  public static final class string {
     public static final int app_name=0x7f040001;
   }
……
}
```



项目配置文件AndroidManifest.xml

- 声明项目所使用的Activity、Service、Receiver
- 指定程序入口点：类似于Win32程序里的WinMain函数

```
<activity android:name=".Activity01"
          android:label="@string/app_name">
   <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
</activity>
<activity android:name="Activity02"></activity>
```

例子MainActivity.java

```
package com.example.hp.myapplication;
import android.app.Activity;
import android.os.Bundle;
public class MainActivity extends Activity
{
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
  }
}
```

Activity是一个应用实体，用于处理action。
当 Activity 开始的时候，Android System 将会调用其中的OnCreate()方法，并调用 setContentView()按定义好的布局显示界面

主要类与方法

- android.app.Activity类：
  - 因为几乎所有的活动（activities）都是与用户交互的，所以Activity类关注创建窗口，可以用方法setContentView(View)将自己的UI放到里面。然而活动通常以全屏的方式展示给用户，也可以以浮动窗口或嵌入在另外一个活动中。有两个方法大部分Activity子类都实现：
- onCreate(Bundle)：
  - 初始化活动（Activity），例如完成一些图形的绘制。最重要的是，在这个方法里通常将布局资源（layoutresource）调用setContentView(int)方法定义你的UI，和用findViewById(int)在你的UI中检索你需要编程的交互的小部件（widgets）。setContentView指定由哪个文件指定布局（main.xml），可以将这个界面显示出来，然后进行相关操作，操作会被包装成为一个意图，然后这个意图对应有相关的activity进行处理。
- onPause()：
  - 处理当离开活动时要做的事情。用户做的所有改变应该在这里提交（通常ContentProvider保存数据）。

程序说明

- 导入类android.app.Activity和android.os.Bundle，HelloWorld类继承自Activity，且重写了onCreate方法。
- **@Override**
  - 在重写父类的onCreate时，在方法前面加上@Override系统可以帮你检查方法的正确性。例如，public void onCreate(Bundle savedInstanceState){.....}这种写法是正确的，若写成public void oncreate(Bundle savedInstanceState){.....}这样编译器会报错，以确保正确重写onCreate方法。（oncreate应该为onCreate）
  - 若不加@Override，则编译器将不会检测出错误，会认为你新定义了一个方法oncreate。

布局文件main.xml

- 严格遵循XML定义规则的
- 指定encoding=”utf-8”
- 例如：

```
<LinearLayout xmlns:android=“http://schemas.android.com/apk/res/android”
android:orientation=“vertical”
android:layout_width=“fill_parent”
android:layout_height=“fill_parent”
>
</LinearLayout>
```

- “LinearLayout”是一种常用的样式配置方案，以上下或左右线性布局安排元素
- “xmlns:android=”http://schemas.android.com/apk/res/android””，这是XML命名空间的声明,它是告诉Android的工具, 你将要涉及到公共的属性已被定义在XML命名空间。在每一个Android的布局文件的最外边的标签必须有这个属性。
- 再往下就是具体的该元素的属性配置

UI布局定义main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    android:textSize="@dimen/text_size"
/>
</LinearLayout>
```

“@string/hello”，“@dimen/text_size”是android的变量引用语法。分别在/res/strings.xml和/res/dimens.xml里定义。

字符串定义strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">FirstProject</string>
    <string name="hello">Hello World!</string>
</resources>
```

该行定义了hello变量的值为“Hello World!”

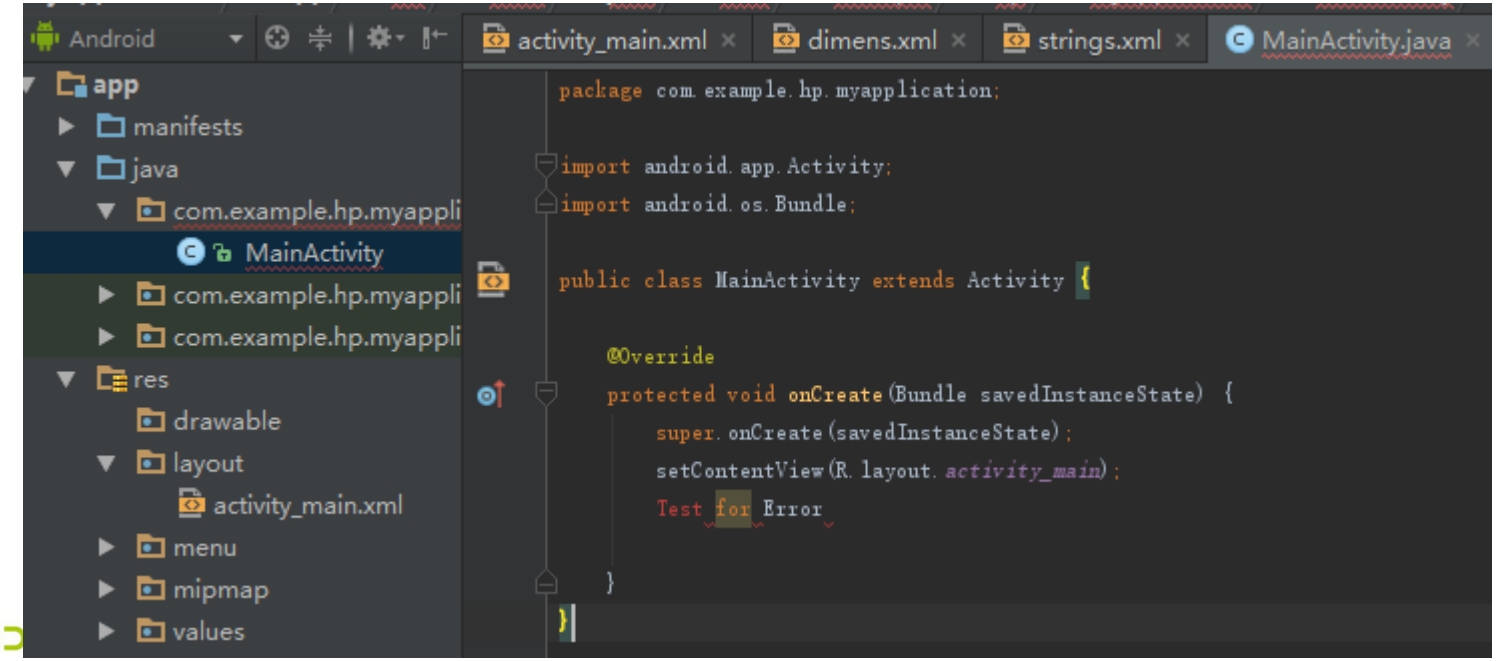
尺寸定义dimens.xml

```
<resources>
    <dimen name="text_size">30sp</dimen>
</resources>
```

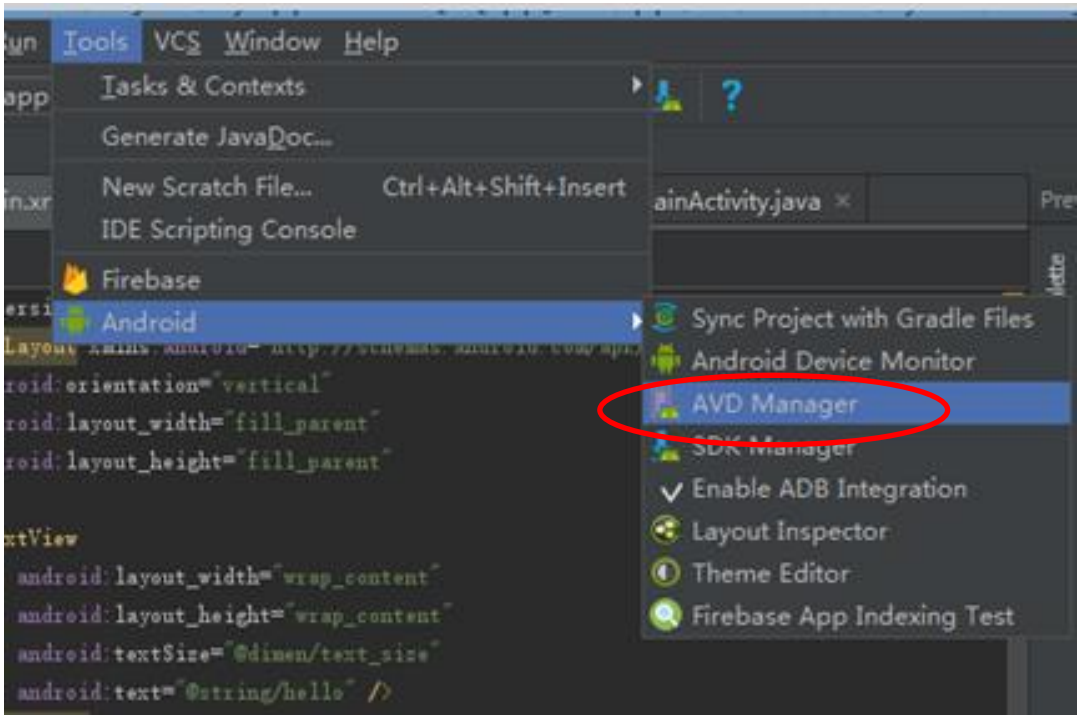
该行定义了text_size变量的值为30sp

处理出错信息

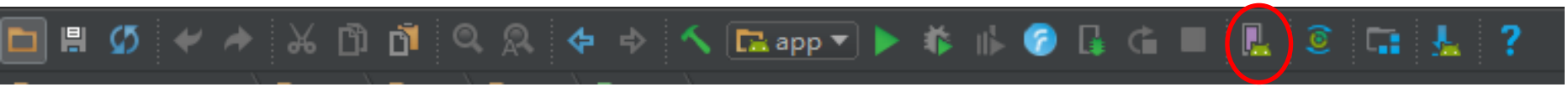
- Android Studio会帮你自动编译，自动查错
- 错误信息“红色波浪线”
- 编译出现错误的文件



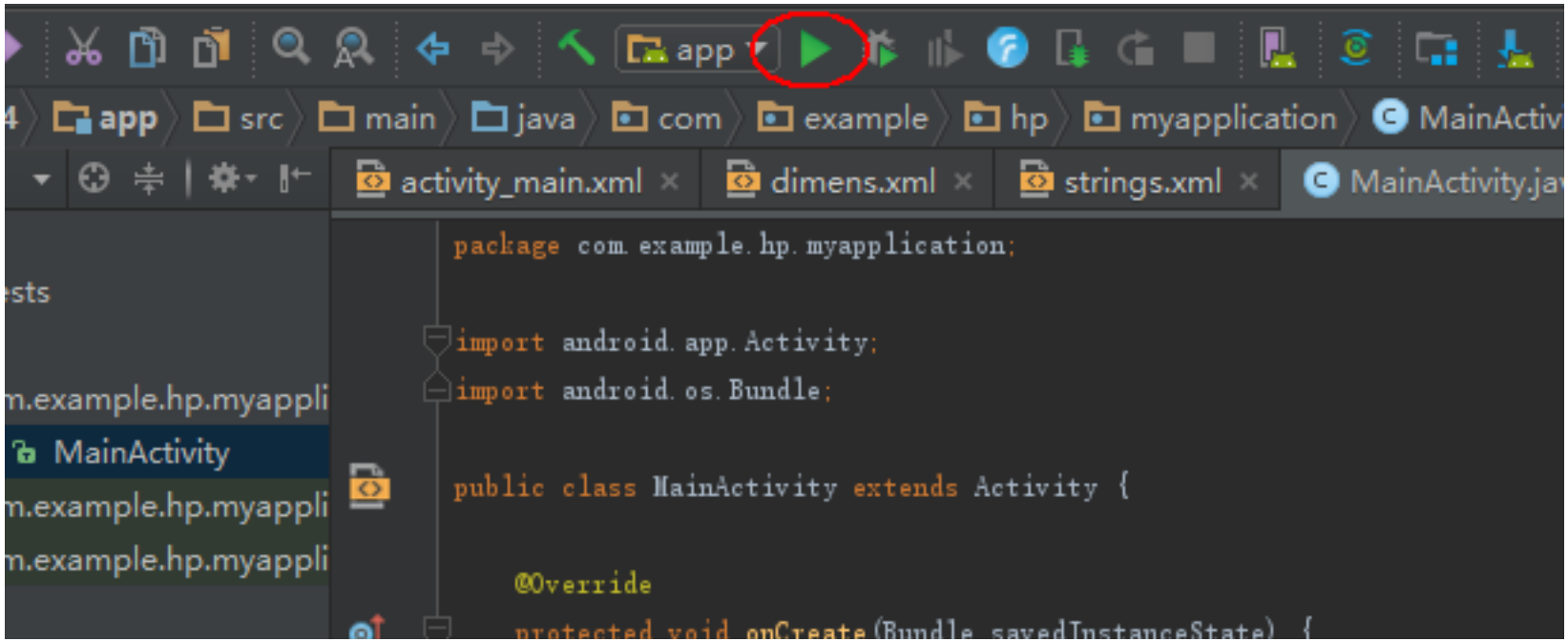
模拟器Emulator设置(AVD Manager)



- 在Android Studio“工具-Android”下的“AVD Manager”
- 可以增删改模拟器



直接点击Run App运行Android程序



模拟运行效果

显示内容：
Hello world!

命令行操作*

- 启动 Android Studio，然后再启动模拟器，这样不但浪费时间，而且Android Studio又占用内存，如何不用启动Android Studio就可以使用模拟器？
- 若想让系统理解你输入的命令(如:输入android命令)有两种方法：

  1.设置环境变量.(和设置java路径一样，具体网上查);

  2.直接进入你SDK里tools目录(cd:sdkpath/tools)

*：高级操作技巧

命令行操作*

- 查看可用的TARGET列表: android list targets
  - 例如输出：

Available Android targets:

---

id: 1 or "android-10"

Name: Android 2.3.3

Type: Platform

API level: 10

Revision: 2

Skins: HVGA, QVGA, WQVGA400, WQVGA432, WVGA800 (default), WVGA854

ABIs : armeabi

命令行运行*

- android create avd -n MyAVD233 -t 1
  - 创建虚拟机，名字MyAVD233，目标SDK：1
- emulator -avd MyAVD233
  - 启动虚拟机MyAVD233
- adb install XXX.apk.
  - 这条命令是安装apk文件，如果有多个设备(而你想把apk安装到emulator-5554这个模拟器上)则要输入:adb install -s emulator-5554 D:/XXX.apk.

命令行运行*

- android create avd -n MyAVD233 -t 1
  - 创建虚拟机，名字MyAVD233，目标SDK：1
- emulator -avd MyAVD233
  - 启动虚拟机MyAVD233
- adb install XXX.apk.
  - 这条命令是安装apk文件，如果有多个设备(而你想把apk安装到emulator-5554这个模拟器上)则要输入:adb install -s emulator-5554 D:/XXX.apk.

命令行运行*

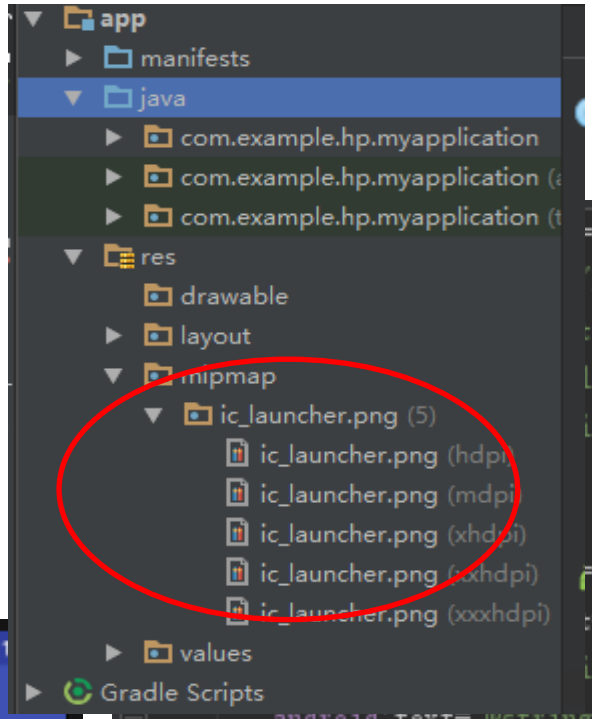
- android create project --target 1 --name MyFirstApp --path <保存路径>\MyFirstApp --activity MainActivity --package com.example.myfirstapp
  - 该命令创建工程
  - 注意：不要创建到AS的workspace下，否则Import时会报错。

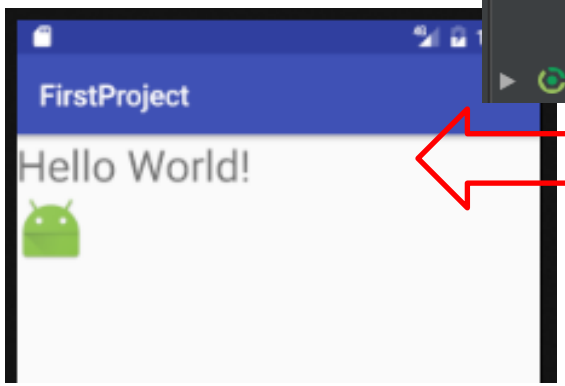
模拟器屏幕大小

- HVGA-L：480x320横屏
- HVGA-P：320x480竖屏（默认）
- QVGA-L：320x240横屏
- QVGA-P：240x320竖屏
- WVGA-L：800x480横屏

显示图像

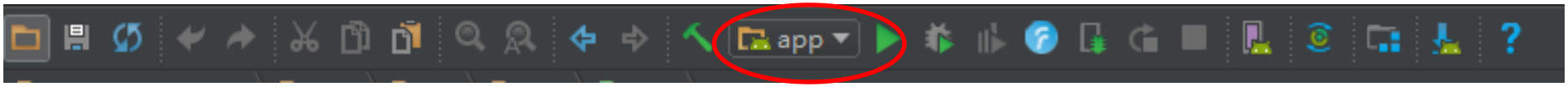
- 添加图片到/res相应目录
- 布局增加ImageView
- 设置图片引用

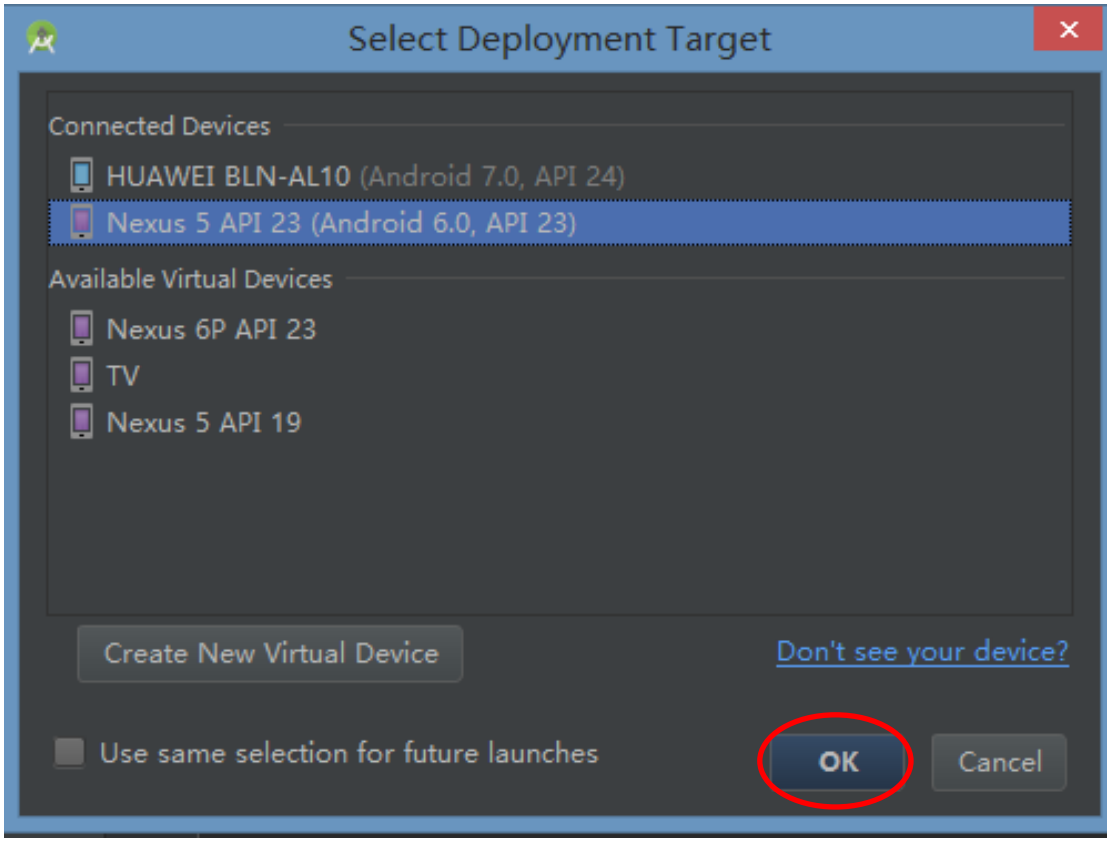




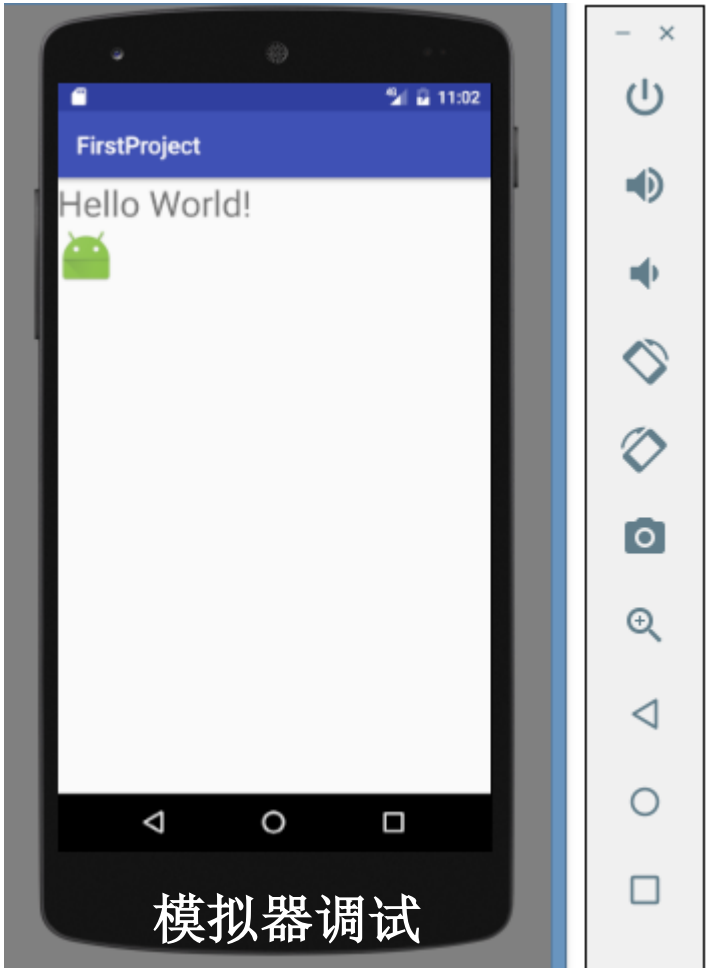

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@mipmap/ic_launcher"/>
</LinearLayout>
```

系统运行

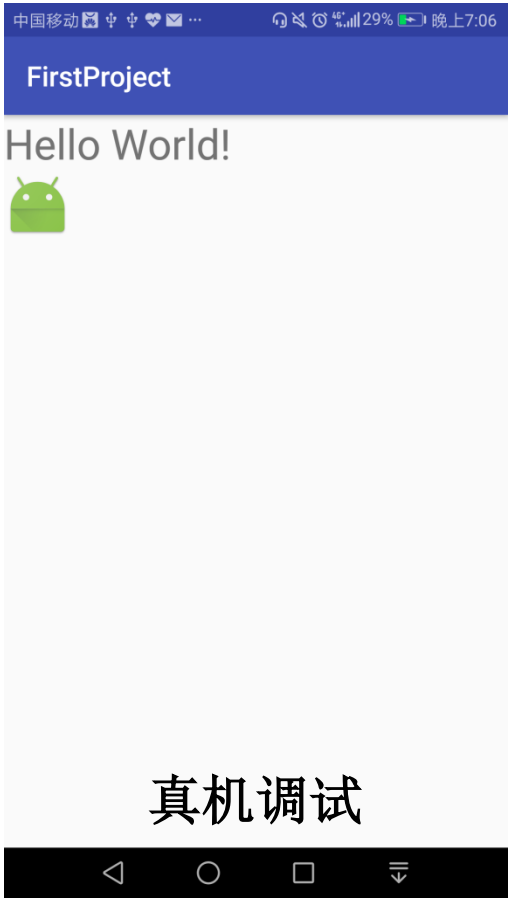

app


Select Deployment Target
Connected Devices
HUAWEI BLN-AL10 (Android 7.0, API 24)
Nexus 5 API 23 (Android 6.0, API 23)
Available Virtual Devices
Nexus 6P API 23
TV
Nexus 5 API 19
Create New Virtual Device
Don't see your device?
Use same selection for future launches
OK
Cancel

运行效果



模拟器调试



真机调试


Questions?