

2016

# Android Studio 从入门到精通(1-10)



# 目录

第一章 Android Studio 简介及其环境搭建.....	4
1.1 Android Studio 简介 .....	4
1.1.1 Android Studio 能做什么 .....	4
1.1.2 为什么选择 Android Studio .....	4
1.2 Android 系统基本知识.....	5
1.3 搭建环境.....	6
1.3.1 第一步 安装 JDK(Java Development Kit) .....	6
1.3.2 第二步 更新 hosts .....	7
1.3.3 第三步 安装 Android Studio.....	7
1.4 新建项目.....	13
1.5 运行 App.....	17
1.5.1 运行在模拟器上.....	17
1.5.2 运行在手机上.....	21
第二章 Android Studio 基本概念 .....	22
2.1 开发环境.....	22
2.1.1 菜单.....	22
2.1.2 工具栏.....	23
2.1.3 项目文件.....	23
2.1.4 视图 view .....	24
2.1.5 布局.....	25
2.2 建立简单的用户界面.....	25
2.3 字符串资源文件.....	26
2.4 增加一个按钮.....	27
2.5 创建一个新的窗体步骤.....	28
2.5.1 第一步：创建布局文件.....	28
2.5.2 第二步：创建对应的 Activity.....	29
2.5.3 第三步：布局文件与 Activity 关联.....	30
2.5.4 第四步：在 AndroidManifest.xml 里声明.....	31
2.6 调用一个窗体.....	31
2.6.1 调用方.....	31
2.6.2 被调用方.....	33
无法打开另一个窗体时检查： .....	34
缺少 SDK 平台 .....	34
第三章 Android Studio 编程语言基础 .....	36
3.1 Java 代码的基本知识 .....	37
3.2 数据类型.....	38
3.2.1 常用数据类型.....	38
3.2.2 类型转换.....	40
3.2.3 常量和变量.....	40
3.2.3 在 Android 开发中的运用.....	40
3.3 运算符.....	41
3.3.1 算术运算.....	41

3.3.2 比较运算.....	41
3.3.3 逻辑运算.....	42
3.3.4 示例.....	42
3.4 控制语句.....	42
3.4.1 分支控制语句.....	42
3.4.2 示例.....	43
3.4.3 循环控制语句.....	43
3.4.4 示例.....	44
3.5 异常处理.....	45
3.6 Android Studio 面向对象初步.....	47
3.6.1 类与对象.....	47
3.6.2 使用类.....	48
3.6.3 Java 中的继承.....	49
3.6.4 包.....	50
第 4 章 Android Studio 界面布局.....	52
4.1 相对布局 RelativeLayout.....	52
4.2 帧布局 FrameLayout.....	54
4.3 线性布局 LinearLayout.....	58
4.4 表格布局 TableLayout.....	58
4.5 网格布局 GridLayout.....	60
第 5 章 Android Studio 视图工具箱.....	62
5.1 使用 View.....	62
5.2 常用 View 的使用.....	62
5.2.1 文本框 TextView.....	63
5.2.2 编辑框 EditText.....	63
5.2.3 单项选择 RadioGroup 和 RadioButton.....	63
5.2.4 多项选择 CheckBox.....	65
5.2.5 图片视图 ImageView.....	65
5.2.6.日期选择器 (DatePicker).....	67
5.2.7 下拉列表 Spinner.....	68
5.3 小结.....	69
第 6 章 多线程的实现.....	70
6.1 计时器范例.....	70
6.2 布局文件.....	70
6.2 设置按钮状态.....	71
6.3 更新计时器.....	72
6.4 运行计时器.....	73
第 7 章管理 Activity 生命周期.....	75
7.1 Activity 生命周期理论知识.....	75
7.1.1 Activity 生命周期的 5 种状态.....	75
7.1.2 生命周期状态的调用.....	76
7.2 生命周期概念的应用.....	76
7.2.1 LogCat 观察运行.....	77
7.2.2 修复 Activity 生命周期问题.....	80

7.3 其它有关 Activity 的知识.....	82
7.3.1 销毁 activity.....	82
7.3.2 停止和重启一个 activity.....	82
7.3.3 停止 activity.....	82
7.3.4 启动/重启 activity.....	83
7.3.5 重建 activity.....	84
7.3.6 保存 active 状态.....	85
7.3.7 恢复 active 状态.....	85
第八章 简单数据和文件的存取.....	87
8.1 键-值对数据的存取.....	87
8.2 文件的存取.....	88
8.2.1 选择内部或外部存储器.....	88
8.2.2 获得外部存储许可.....	89
8.2.3 保存文件到内部存储器.....	89
8.2.4 保存文件在外部存储器.....	91
8.2.5 查询空闲空间.....	92
8.2.6 删除文件.....	93
第 9 章 数据库操作.....	94
9.1 创建数据模型.....	94
9.2 SQLiteOpenHelper 类详解.....	94
9.2.1 SQLiteOpenHelper 类.....	95
9.2.2 创建数据库.....	96
9.2.3 数据库升级.....	96
9.3 创建数据库和表.....	97
9.4 数据库操作.....	98
9.4.1 创建菜单.....	98
9.4.2 创建新增、修改的 Activity.....	99
9.4.3 新增记录.....	101
9.4.4 显示数据.....	101
9.4.5 修改数据.....	104
9.5 其它.....	106
第 10 章 Intent 详解--与其它 App 交互.....	107
10.1 调用 Activity 并返回结果.....	107
10.1.1 启动 Activity.....	107
10.1.2 工作过程.....	109
10.2 调用其它 APP 服务.....	110
10.2.1 调用动作.....	110
10.2.2 获取结果.....	113
10.3 让其它 app 启动你的 Activity.....	115
10.3.1 增加一个 intent 过滤器.....	115
10.3.2 显示 app 选择器.....	117
10.3.3 在你的 Activity 处理 intent.....	117
10.3.4 返回结果.....	118

# 第一章 Android Studio 简介及其环境搭建

欢迎来到 Android Studio 应用程序开发世界！我将带领大家熟悉使用 android studio 开发平台。听课者需要有一定的编程基础，最好是 Java 编程基础。每次课我会完成一个案例，边干边学，以此让大家越来越熟悉安卓平台。希望大家学习完整个教程后，都能很好地掌握该平台。

本章学习内容包括 5 个方面：

第一，简要介绍 Android Studio，告诉大家为什么学习它；

第二，因为 Android Studio 开发出来的 App 运行在 Android 平台上，所以要介绍 Android 平台；

第三，重点学习 Android Studio 开发环境的搭建过程，搭建好环境，我们就可以顺利地学习后面的章节了。；

第四，快速创建一个 Android 项目；

第五，运行 App 在模拟器和手机上。

## 1.1 Android Studio 简介

### 1.1.1 Android Studio 能做什么

大家都知道谷歌公司的 Android 操作系统，作为智能手机等移动设备的平台。光有平台还不行，平台上还有各种各样称为 APP 的应用软件。Android Studio 是谷歌公司研发的开发基于 Android 系统的软件开发工具，也就是开发运行在 Android 平台上的 APP，这些应用可以设计为工具、管理、互联网、游戏等等软件，取决于程序员的自由发挥和创意。

Android Studio 开发的产品不仅可以运行在智能手机上，还可以开发智能穿戴、电视、车载设备的应用。

### 1.1.2 为什么选择 Android Studio

从 Android 操作系统诞生的时候，开发基于 Android 系统的 APP 的平台是 Eclipse，关于 Android Studio 相比 Eclipse 的优点网络上讨论很多。这里我不打算展开是使用 Android Studio 还是 Eclipse 的辩论，我要提醒读者的是，作为谷歌在 2013 年为开发者提供的 IDE 环境工具 Android Studio，从几次更新之后 Android Studio 已经成为了非常强大的 IDE 开发环境。谷歌也宣布将在年底前中止对其他 IDE 开发环境的支持。也就是说，开发者是时候正式向 Eclipse 说再见了。安卓产品经理 Jamal Eason 在声明中写道“谷歌将会全力专注于 Android Studio 编译工具的开发和技术支持，中止为 Eclipse 提供官方支持。包括中止对 Eclipse ADT 插件以及 Android Ant 编译系统的支持。”

如果你是一名打算从事基于 Android 系统软件开发的初学者，那么一开始学习 Android Studio 是明智的选择，可以少走弯路，抢占战略制高点。毕竟 Android 操作系统是谷歌的，Android Studio 是谷歌的亲儿子。

## 1.2 Android 系统基本知识

用 Android Studio 开发出来的 APP 是运行在 Android 操作系统上的，所以这里概述安卓系统的基本知识是非常必要的。安卓系统是为移动设备准备的操作系统，已被各大移动设备制造商所采用。它含有一个 Linux 操作系统和一些中间件，通常，它还带有一系列关键应用，如联系人管理、地图应用、浏览器等等。安卓由谷歌开发和维护，它是一个开源项目，因此你有兴趣的话可以下载它的源代码。

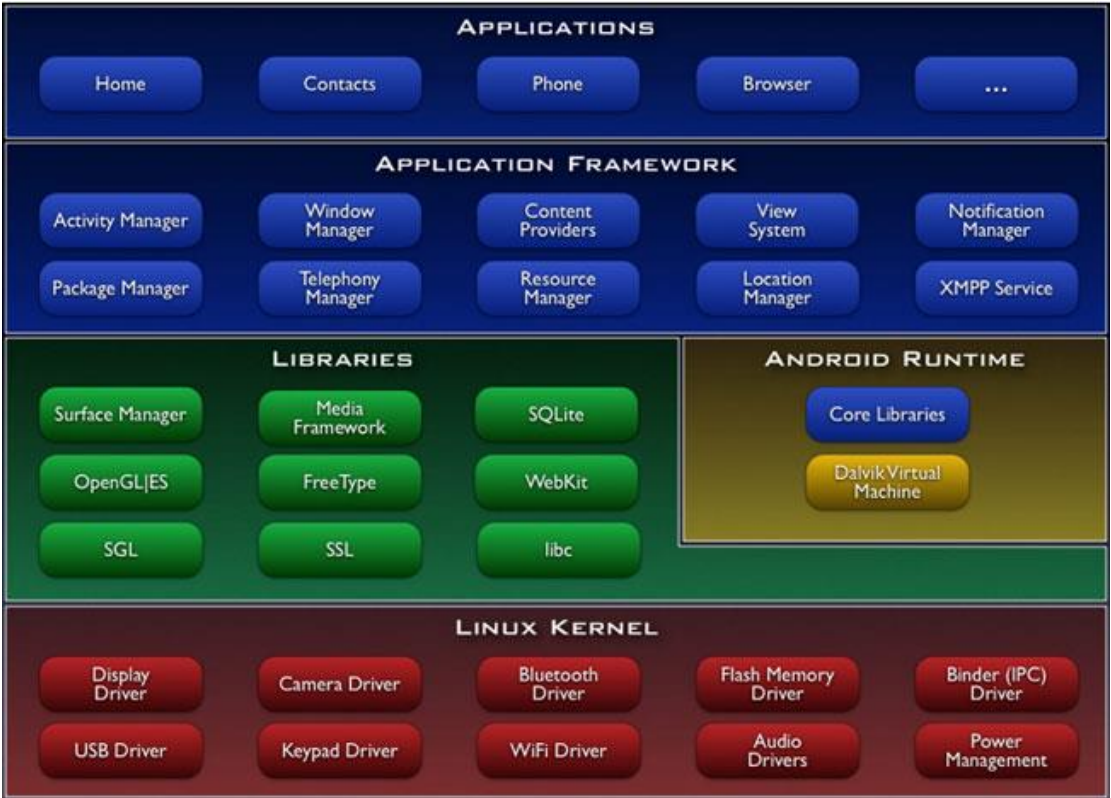


图 1-1 Android 系统

这里把安卓平台按照从下至上顺序分为四层，图中最上面为第四层是应用层，一般的安卓设备都会预装一些应用，所以你有联系人管理、打电话、网络浏览器等应用。安卓的奇妙之处是一个开放式平台，像我们这样的第三方开发者，可以定制自己的应用，并提供给使用安卓设备的用户，这就是在安卓应用框架上建立应用，这就是第三层。注意第三、四两层都是蓝色的，都是用 Java 编写的。因此，移动设备上自带的所有应用，以及第三方或者你自己开发的应用，都是用 Java 语言写的。应用框架是一系列相互紧密联系的应用框架，其中有一系列不同的子框架，有管理活动的框架。活动是什么，后面我会细讲。有窗口管理框架，有管理设备本地内容的框架，有处理电话事件的框架，有位置管理框架等等。我们不能详细解说所有框架，我们只会涉及常用的关键框架和接口。

而框架本身，则是在一系列库之上写出来的，也就是绿色的第二层是安卓运行时。库都是用 C 和 C++写的，这些库是自带的，有一些是由谷歌实现的，其它是集成的开源库。比如 SQLite 是一个轻量级数据库平台，它是安卓设备包含的开源项目，和 iPhone 使用的软件包是一样的都是 SQLite 数据库。还有浏览器的 WebKit 库等等。最后，所有这些都建于位于第一层的 Linux 内核之上，这是硬件之上最直接的软件层，它管理最根本的所有资源，需要强调的是，自带应用程序和你作为开发者所开发的应用，都是基于相同的框架，管理联系人等

等和你的程序所使用的都是完全相同的框架，也就是说，安卓 API 没有特殊和秘密可言。自带程序和你作为开发者开发的程序没有本质差异，这对我们很有好处，是安卓最令人称心的一项特性，它是完全开放的。

## 1.3 搭建环境

安装软件已经为你准备好了，只需按照如下步骤很简单地可以安装上。安装前先检查您电脑的硬件和操作系统是否符合条件：

- Microsoft® Windows® 8/7/Vista/2003 (32 or 64-bit)
- 内存至少 2 GB，推荐内存 4 GB
- 安装完毕后硬盘至少有 400 MB 运行空间
- 至少 1 GB 空间留给 Android SDK，模拟器系统映像，缓存
- 至少 1280 x 800 屏幕分辨率
- 可选加速模拟器: Intel® processor with support for Intel® VT-x, Intel® EM64T (Intel® 64), and Execute Disable (XD) Bit functionality

### 1.3.1 第一步 安装 JDK(Java Development Kit)

打开文件夹step1，

如果你的操作系统是64位的，运行文件jdk-8u77-windows-x64.exe；

如果你的操作系统是32位的，运行文件jdk-8u77-windows-i586.exe。

安装好 JDK 后，设置环境变量：打开控制面板>系统>高级系统设置>高级，单击按钮【环境变量】，在“系统变量”栏，单击【新建】，在弹出窗口设置变量名：JAVA\_HOME，变量值：JDK 安装在你机器上的目录。（例如，我的安装目录是 C:\Program Files\Java\jdk1.8.0\_45\，那么变量值应该是 C:\Program Files\Java\jdk1.8.0\_45\）

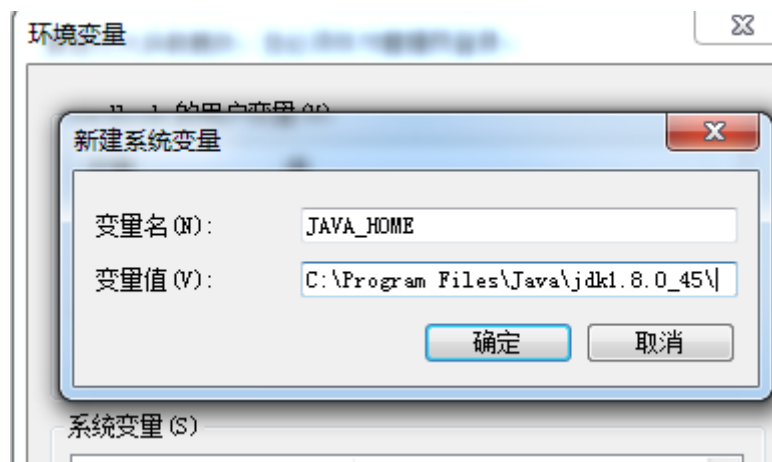


图 1-2



### 1.3.2 第二步 更新 hosts

打开文件夹step2，如现有hosts内容重要的话请备份后再使用（一般不需要）。  
如下图1-3-1所示，win7/win8/win10右键以管理员身份运行“以管理员权限运行.bat”。杀毒软件可能会提醒用户正在修改hosts，如图1-3-2，选择允许修改。完毕后不会有提示，可以进行下一步了。

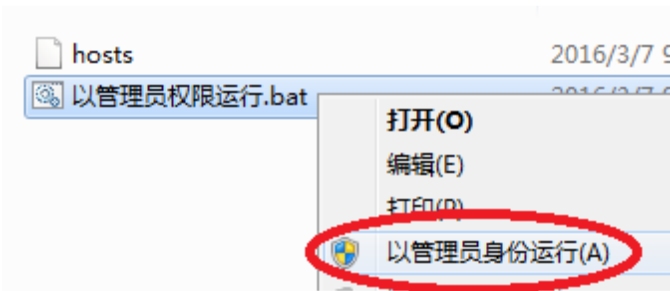


图1-3-1

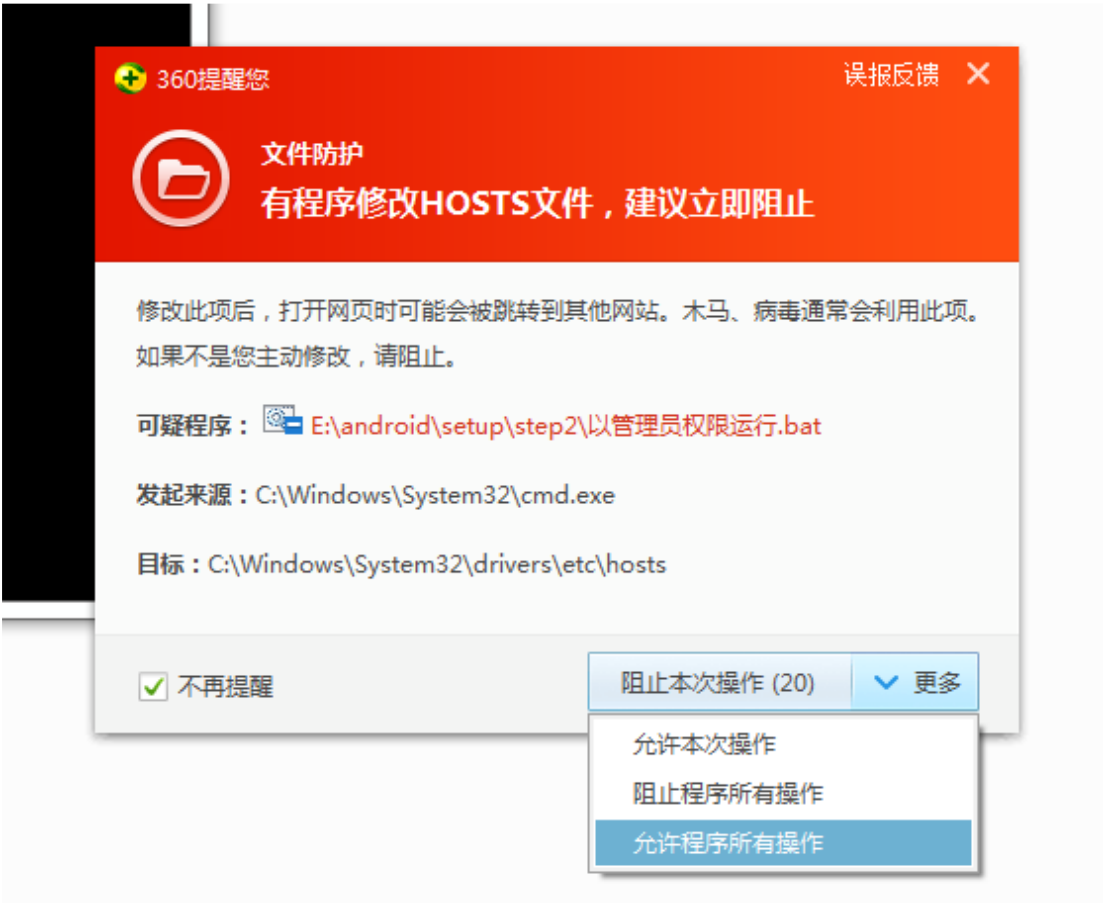


图1-3-2

### 1.3.3 第三步 安装 Android Studio

保持与互联网连接，然后打开文件夹step3，运行 android-studio-bundle-141.2456560-windows.exe 文件，图 1-4 至图 1-16 是安装过程。





图 1-4

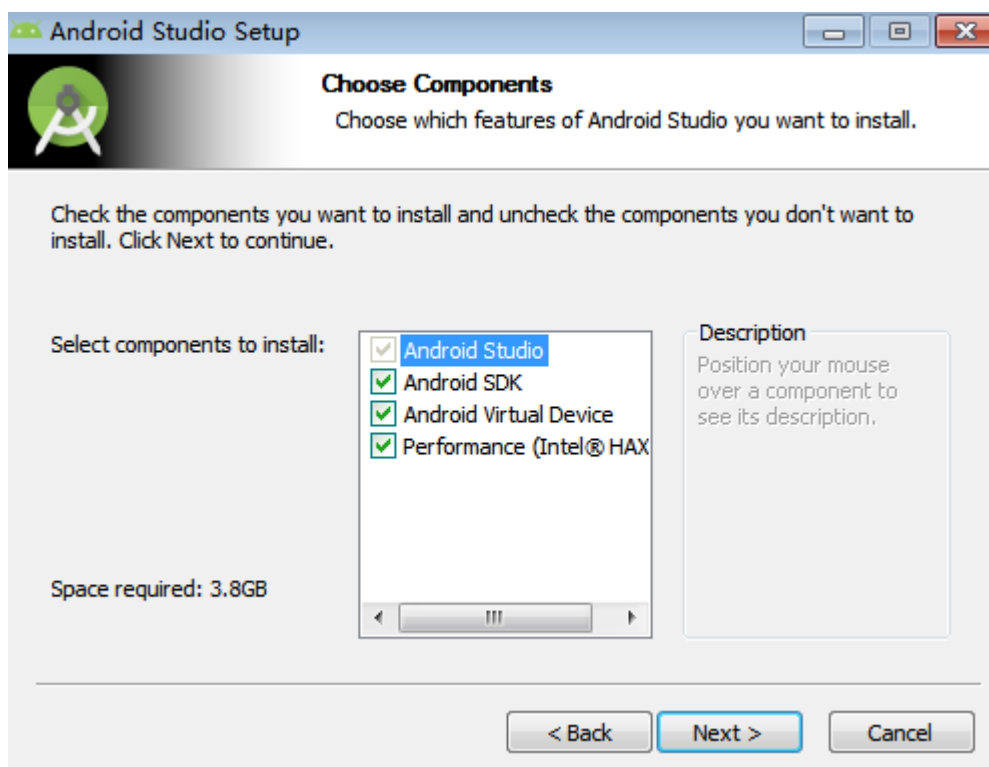


图 1-5

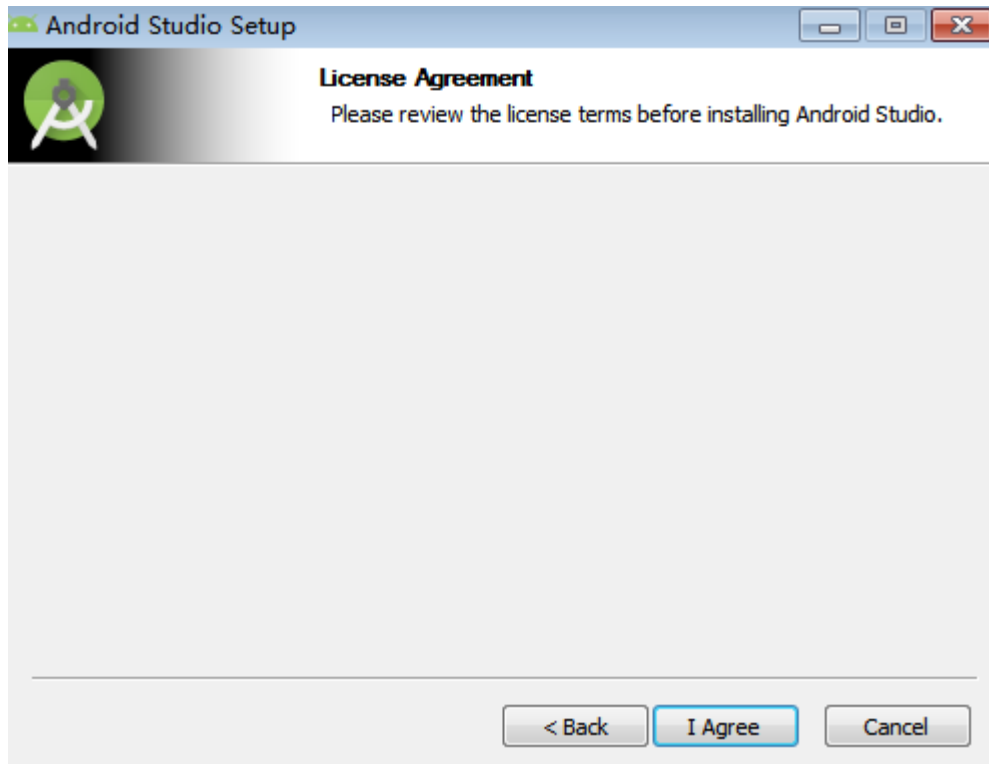


图 1-6

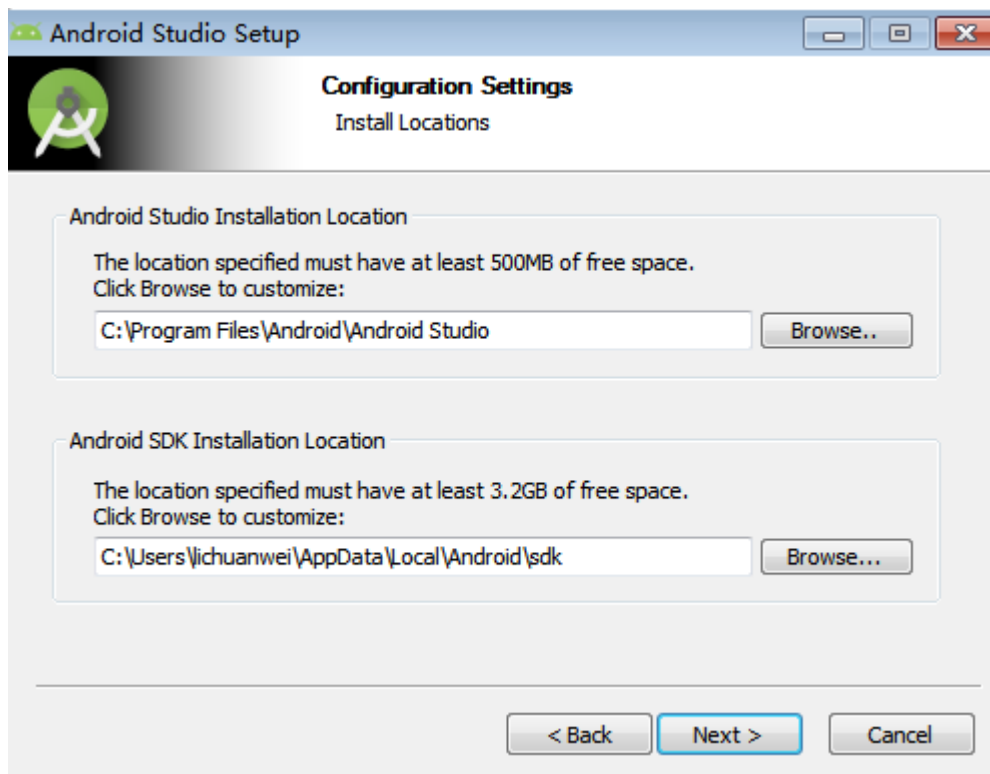


图 1-7

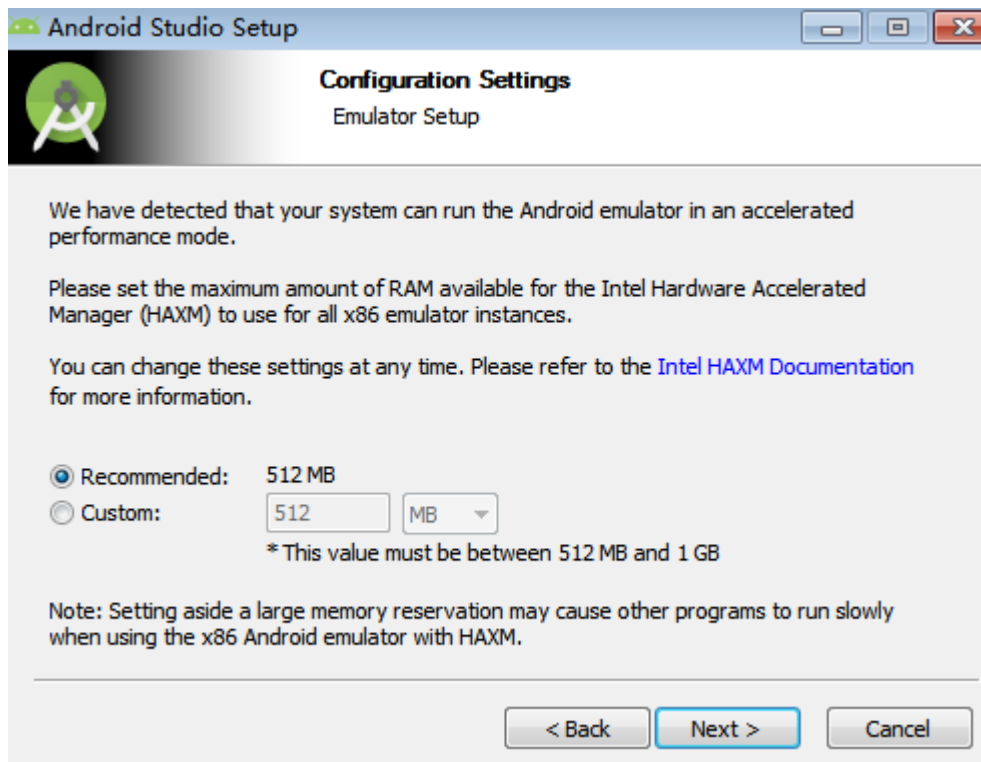


图 1-8

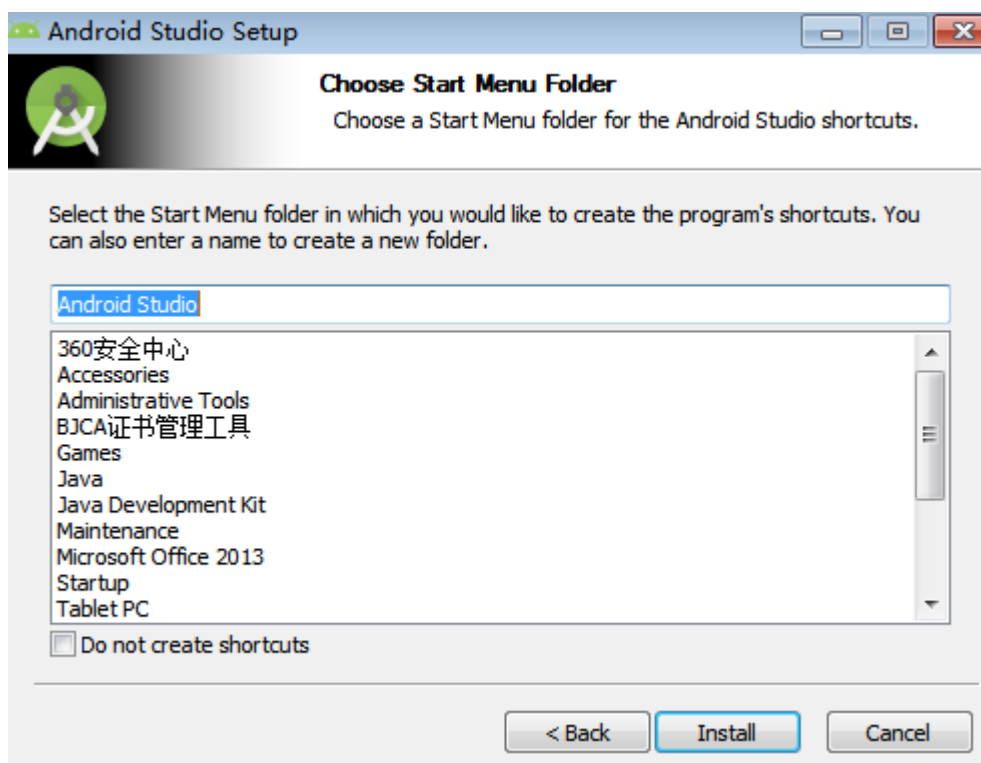


图 1-9

注意：这一步能否成功的关键是否能和谷歌公司网站链接，不要尝试各种跳过这一步的方法了

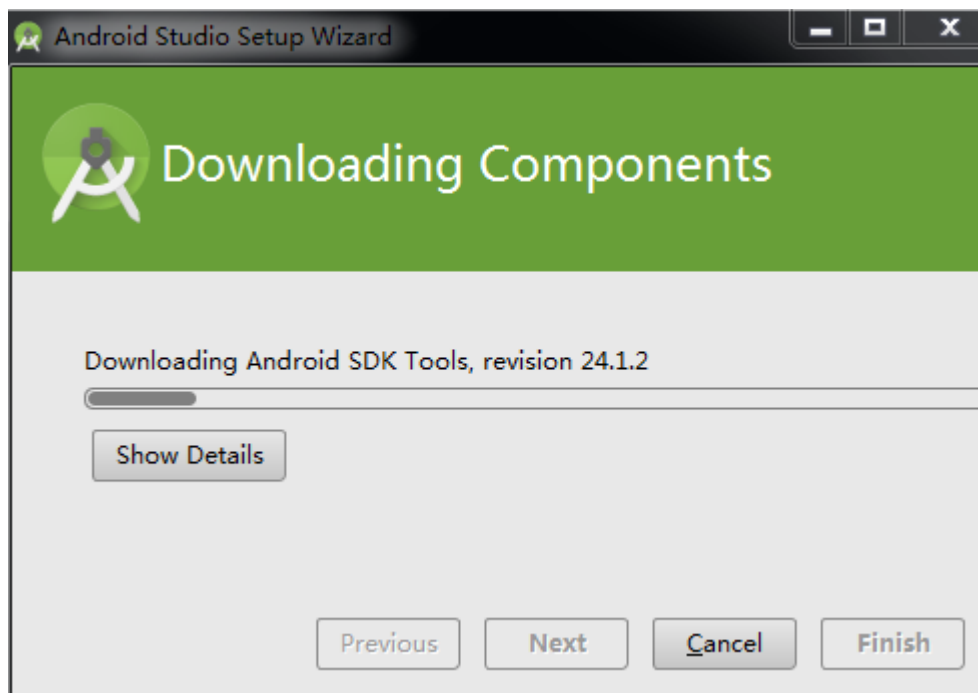


图 1-10

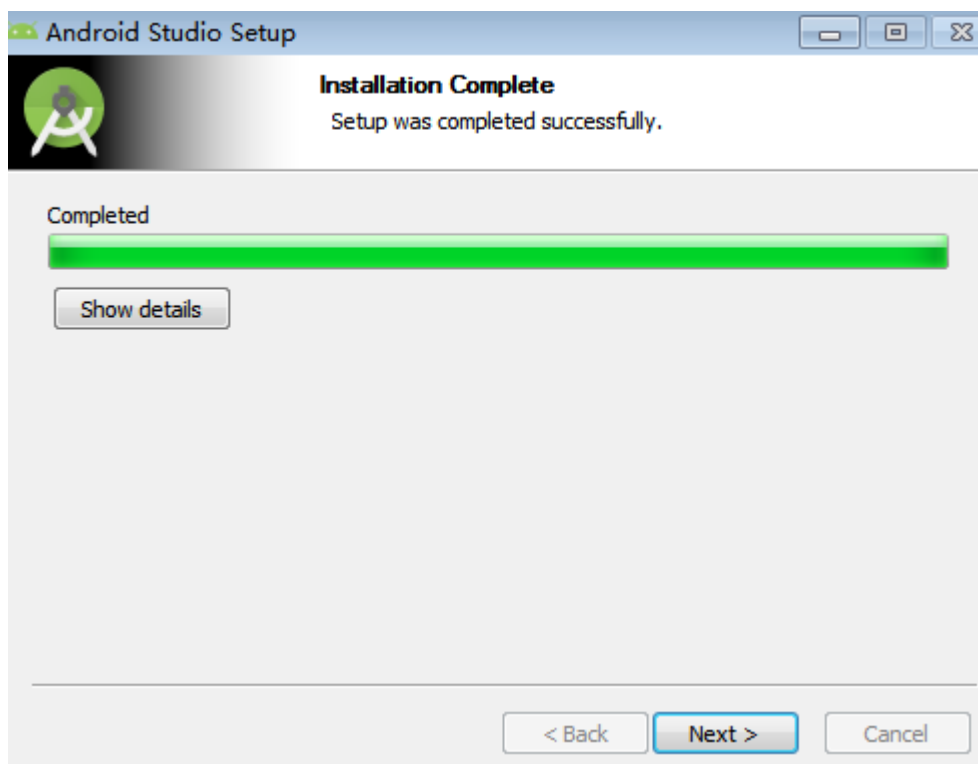


图 1-11

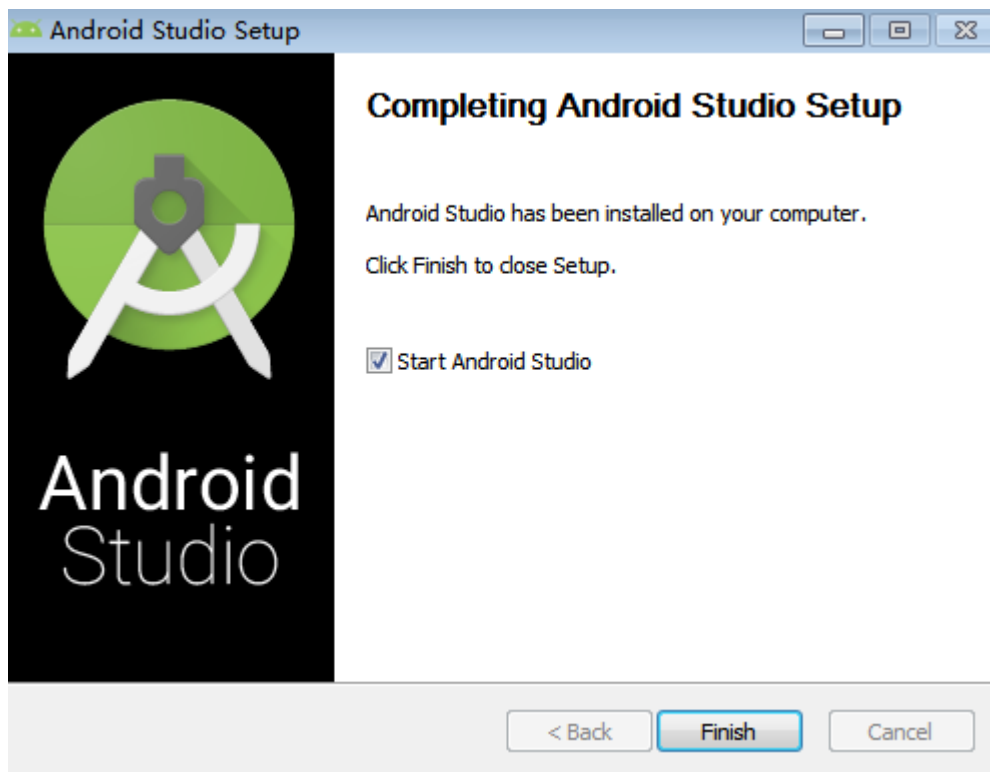


图 1-12

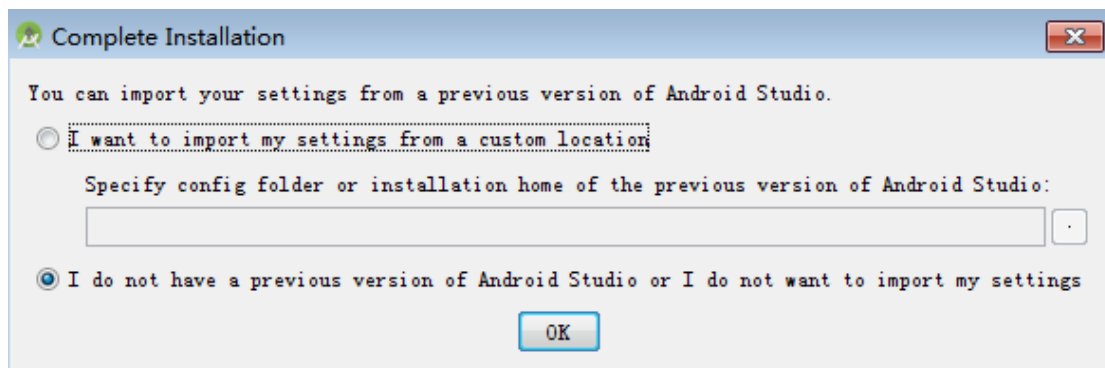


图 1-13

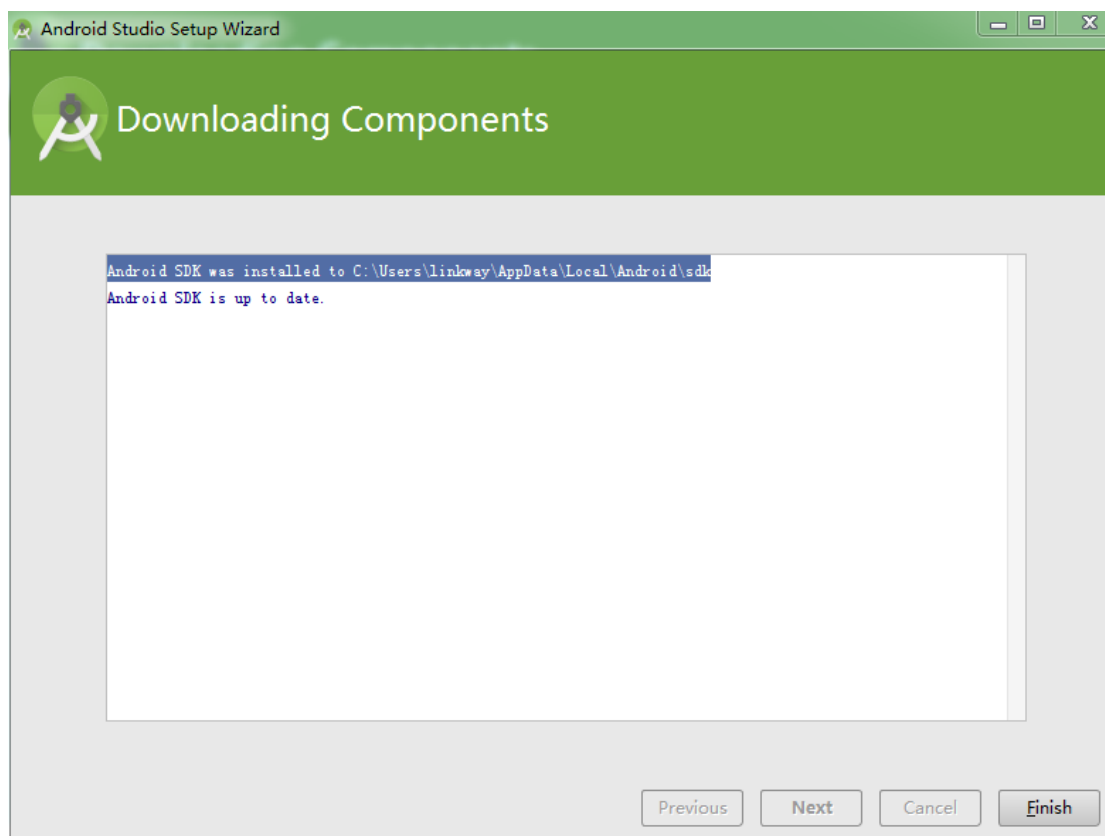


图 1-14

## 1.4 新建项目

安装完成后，我们快速创建一个项目检验一下，后面会详细说明创建项目的细节。打开 Android Studio，如图 1-15 在欢迎界面单击 **Start a new Android Studio project**。（注：如果你已经打开了项目，从 **File** 菜单，选择 **New Project**）

我们目前只需了解第一、二项：

**Start a new Android Studio project:** 开始新建 Android Studio 项目。现在我们要新建一个项目，所以选择这一项。

**Open an existing Android Studio project:** 打开已存在的 Android Studio 项目。

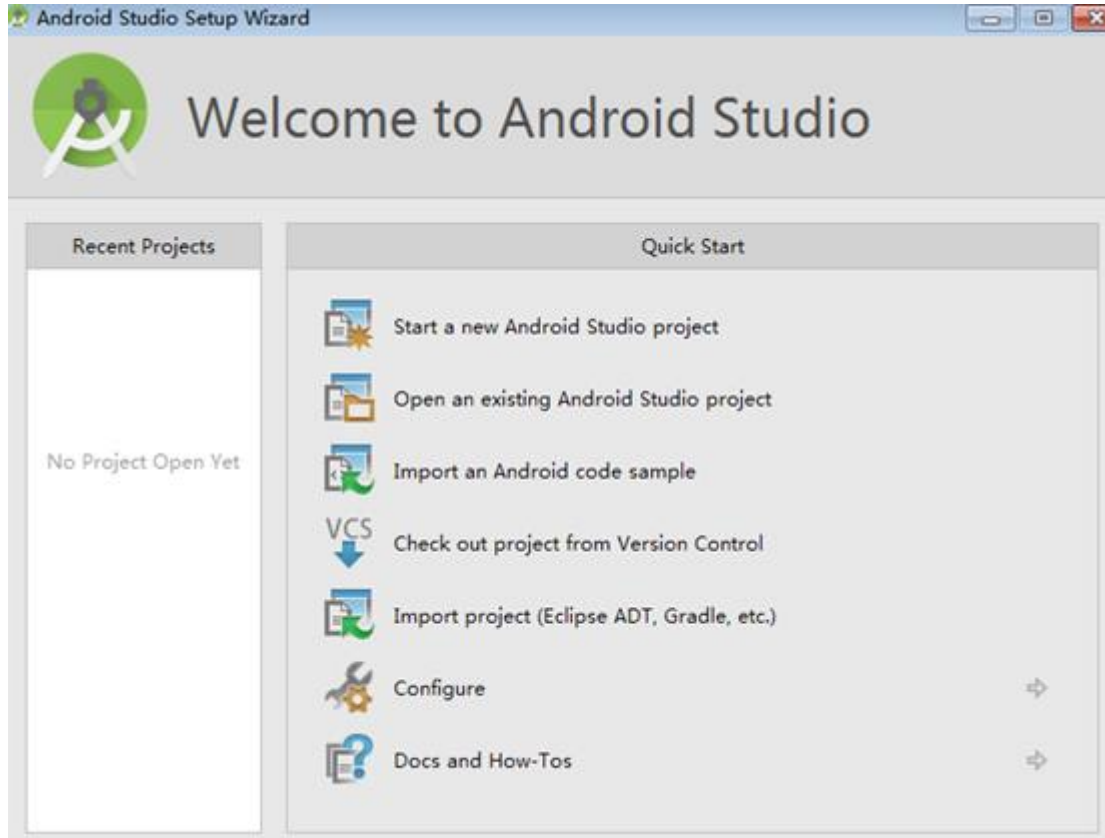


图 1-15 Android Studio 开始界面

单击 Start a new Android Studio project，进入图 1-16 界面，

**Application Name** : HelloWorld;

**Company domain:** mycompany.com。注意这里填写的内容与访问互联网没有任何关系，所以你尽管可以填写不存在的域名；

**Project location:** 保存项目文件的目录。我们学习 Android Studio 时候会创建许多应用程序，为集中管理。我预先新建了文件夹 Androidproject，把本例放在这个文件夹里。

填写 **Application Name** 字段，本例填写 “My First App”，其它可保持默认，单击 Next。

当然可以填写其它值，但这里希望你使用和我的例子一样值，学习后面内容会容易些。



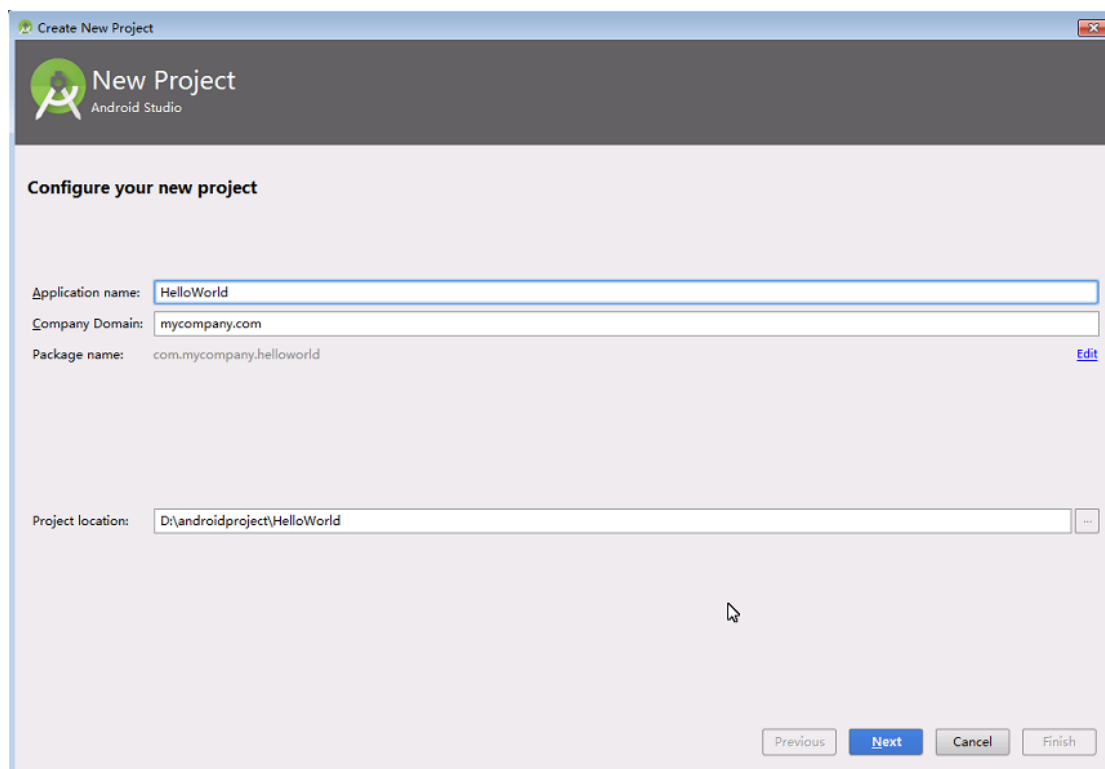


图 1-16

进入图 1-17 界面，勾选 **Phone and Tablet**，Minimum SDK 选择 API 14:Android 4.0 (IceCreamSandwich)，**Minimum SDK** 选择 API 14: Android 4.0 (IceCreamSandwich)。Minimum Required SDK 是 app 所支持的最早 Android 版本。为了尽可能支持更多的设备，你应当把它设置为最低版本，这样 app 可为低版本提供核心可用功能。因为如果你的 app 所有功能仅在最新版本的 Android 上运行的话会导致许多手机的版本不可用。我们目前只学习手机开发，不要勾选其它选项（TV, Wear, Glass），单击 Next.

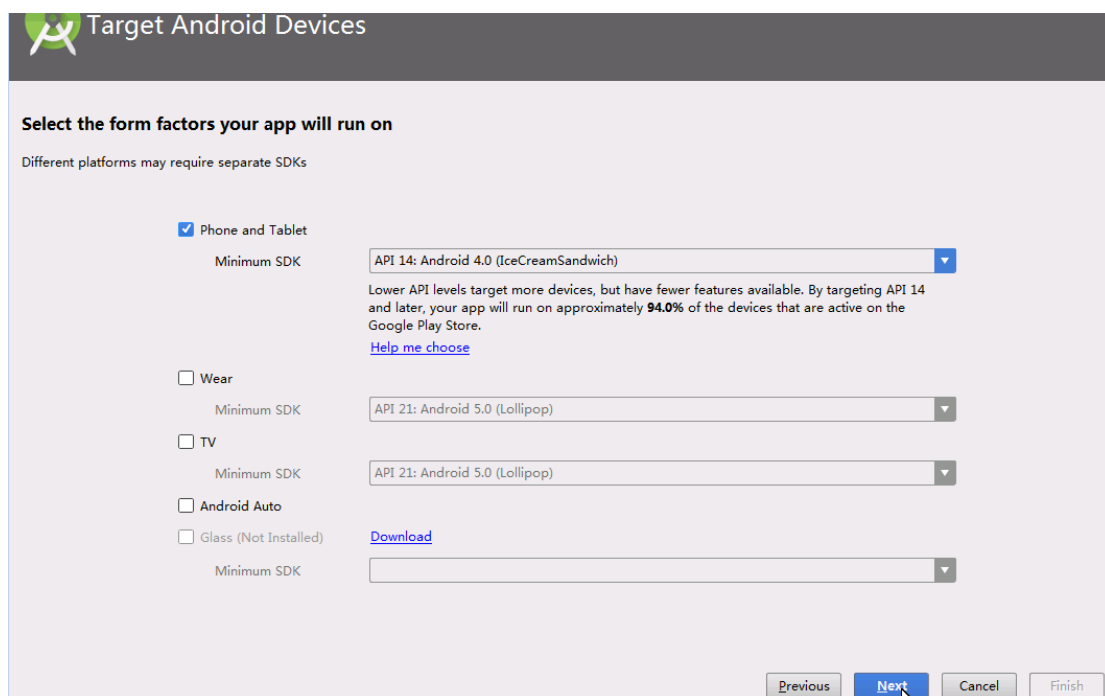


图 1-17

选择界面布局。在图 1-18 选择 **Blank Activity** 单击 **Next**。 选择一个与你开发 APP 相配的界面可以为你节省不少精力，但我们处于学习阶段选择“Blank Activity”，一个空白的布局。

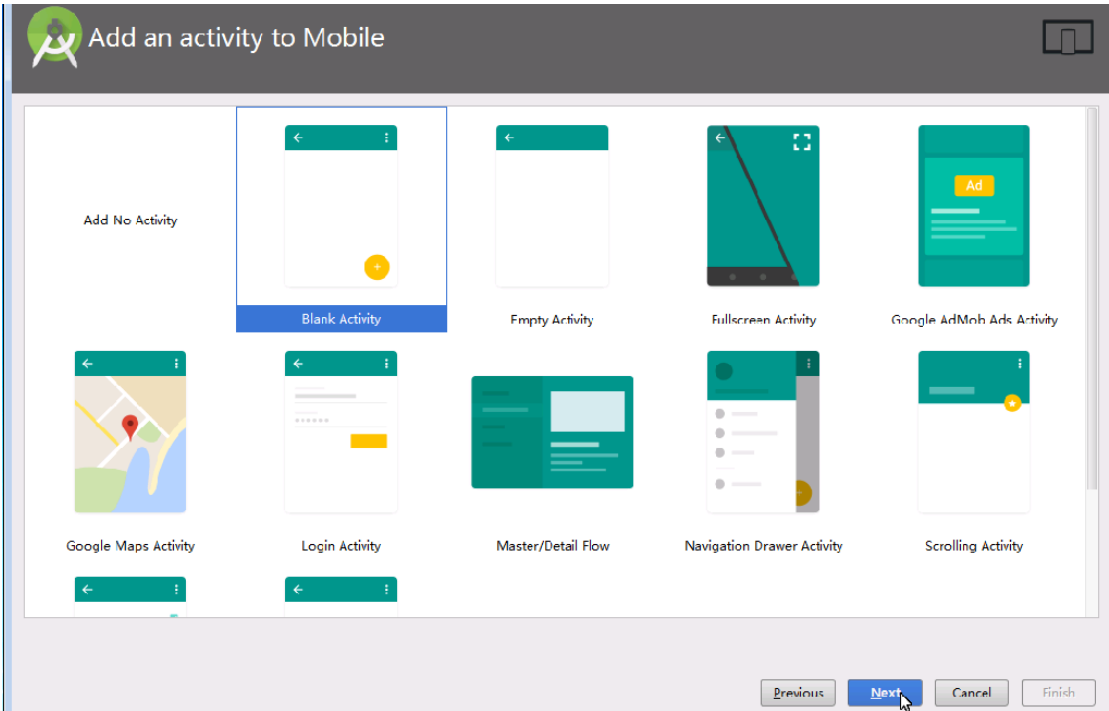


图 1-18

在图 1-19 ,使用默认值，单击 **Finish** 按钮创建了项目。

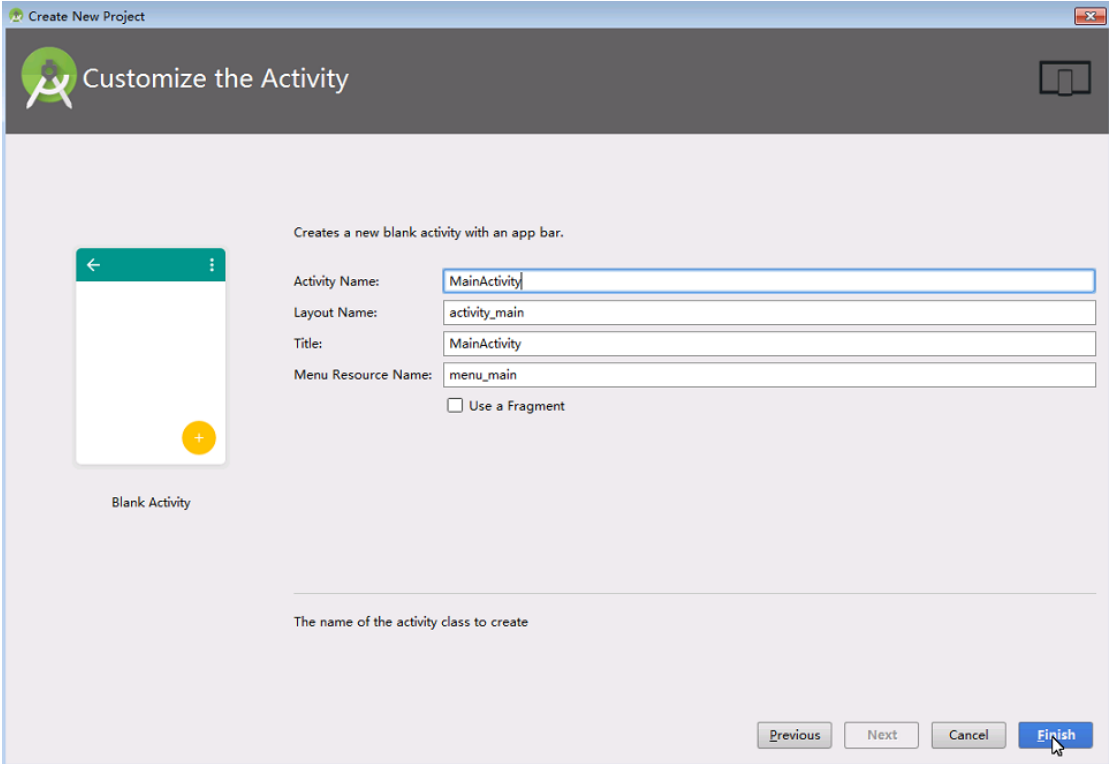


图 1-18

稍等一会儿，Android 项目现在有了基本的“Hello World”app，并且包括一些默认文件，

如图 1-19。

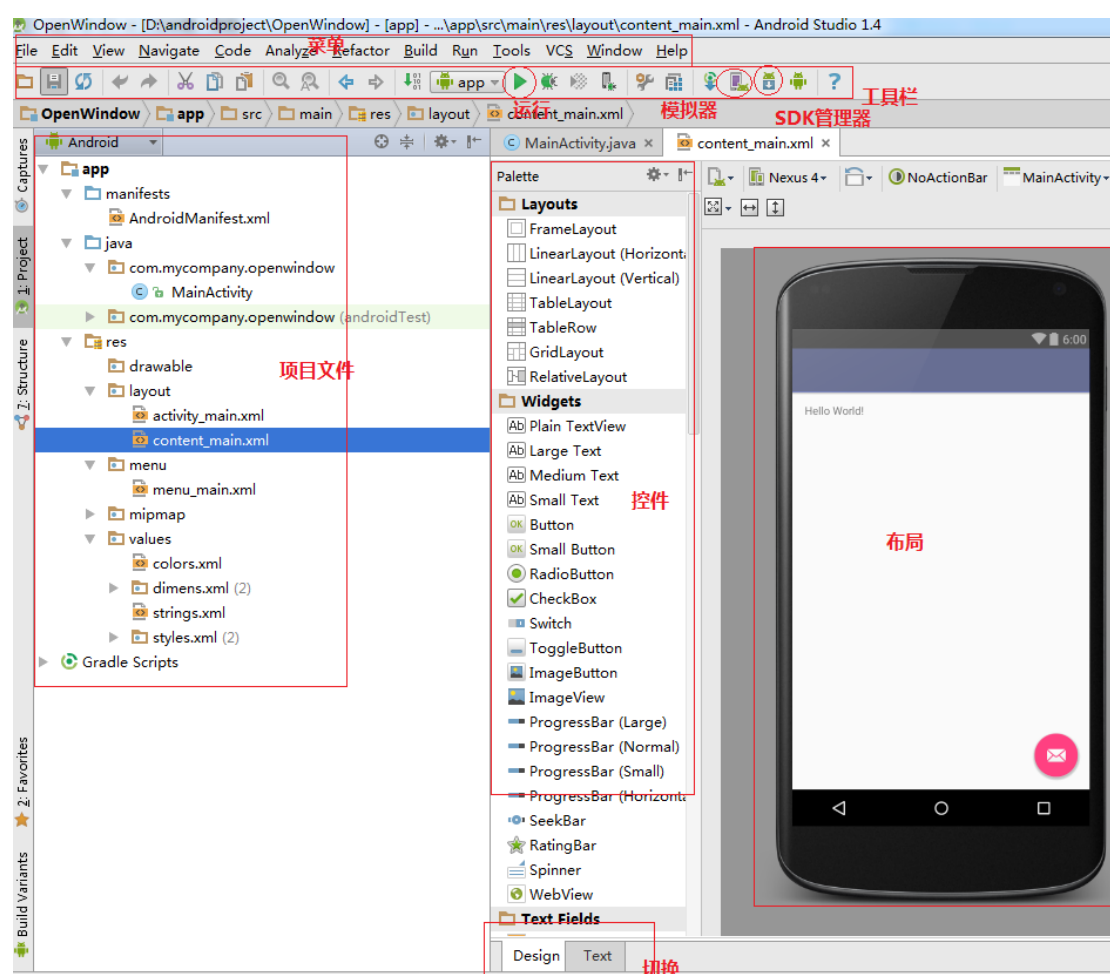


图 1-19

## 1.5 运行 App

学习如何安装、运行 app 在一个实际设备上和一个 Android 模拟器上。实际设备包括智能手机、PDA 等等。下面我们就把实际设备认为一部安装 Android 系统的智能手机，简称手机。

### 1.5.1 运行在模拟器上

开发程序需要无数次的调试，如果每次都安装部署到实际手机上现实的，只有开发的最后阶段才安装到手机上测试。所以需要运行在模拟手机上。

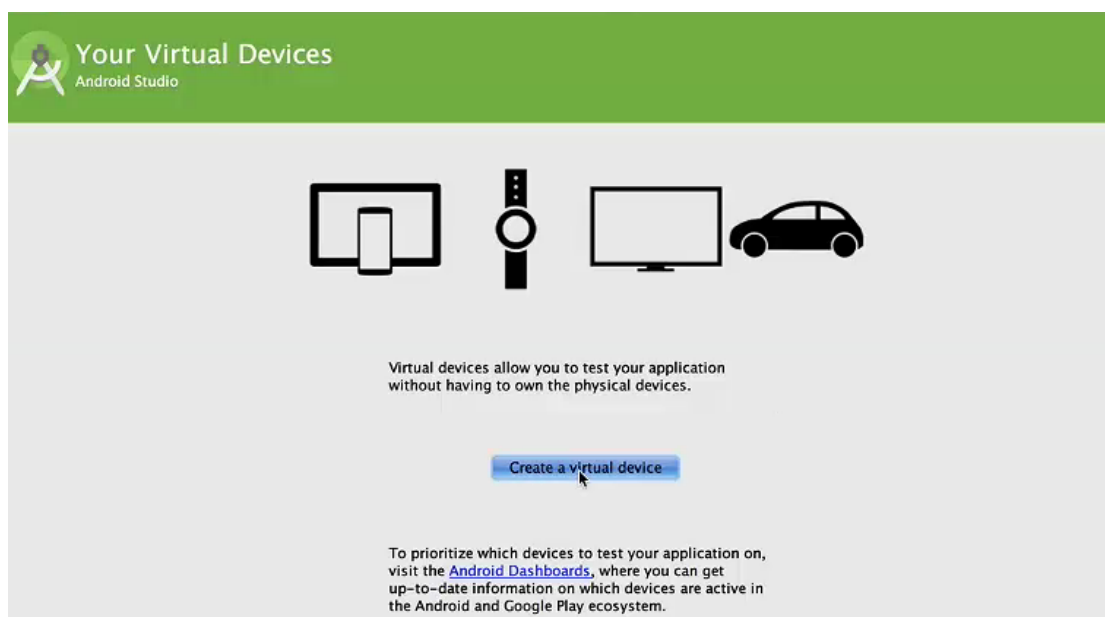
#### 1.创建模拟器

单击图 2-6 的所示模拟器，出现 3-1 界面，单击图中 Action，在弹出菜单中单击“Delete”，删除当前模拟器。



图 3-1

## 2.创建模拟器



3-2

## 3.选择模板

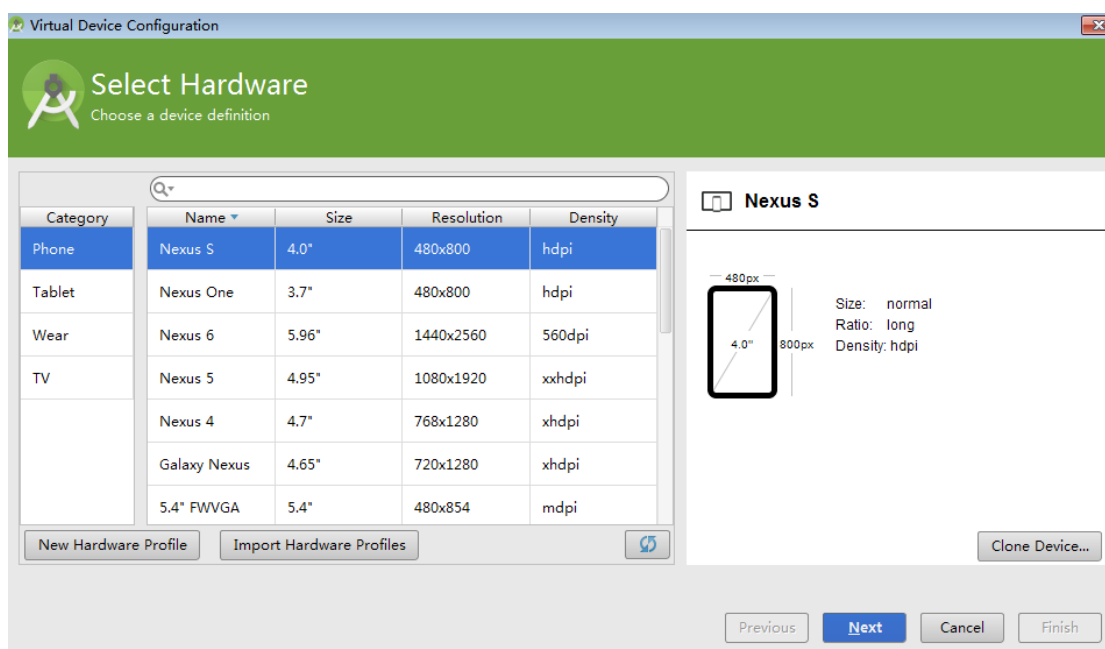


图 3-3

4.选择 Android4.0 的模拟器，如果没安装，单击所在行的 download 下载。需要你的计算机能够和谷歌公司链接。

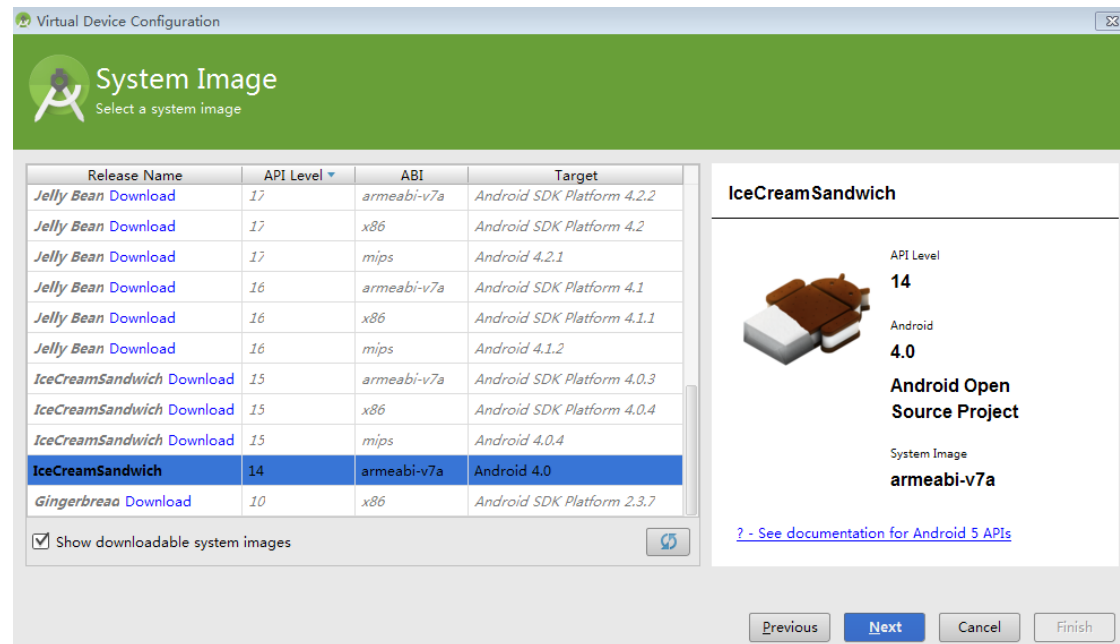


图 3-4

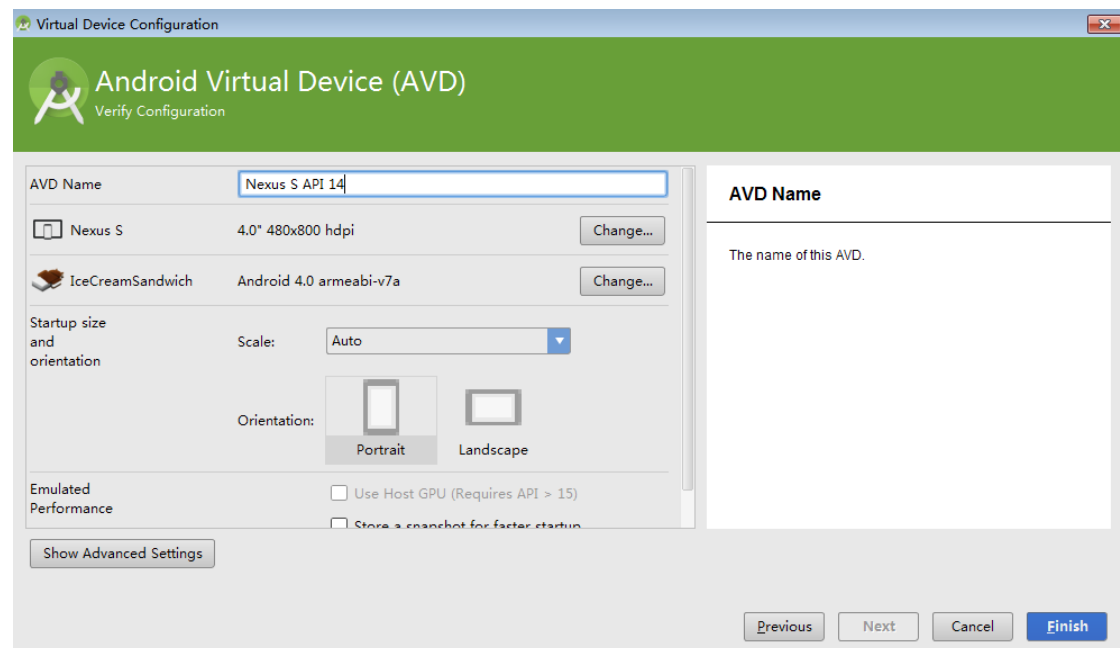


图 3-5

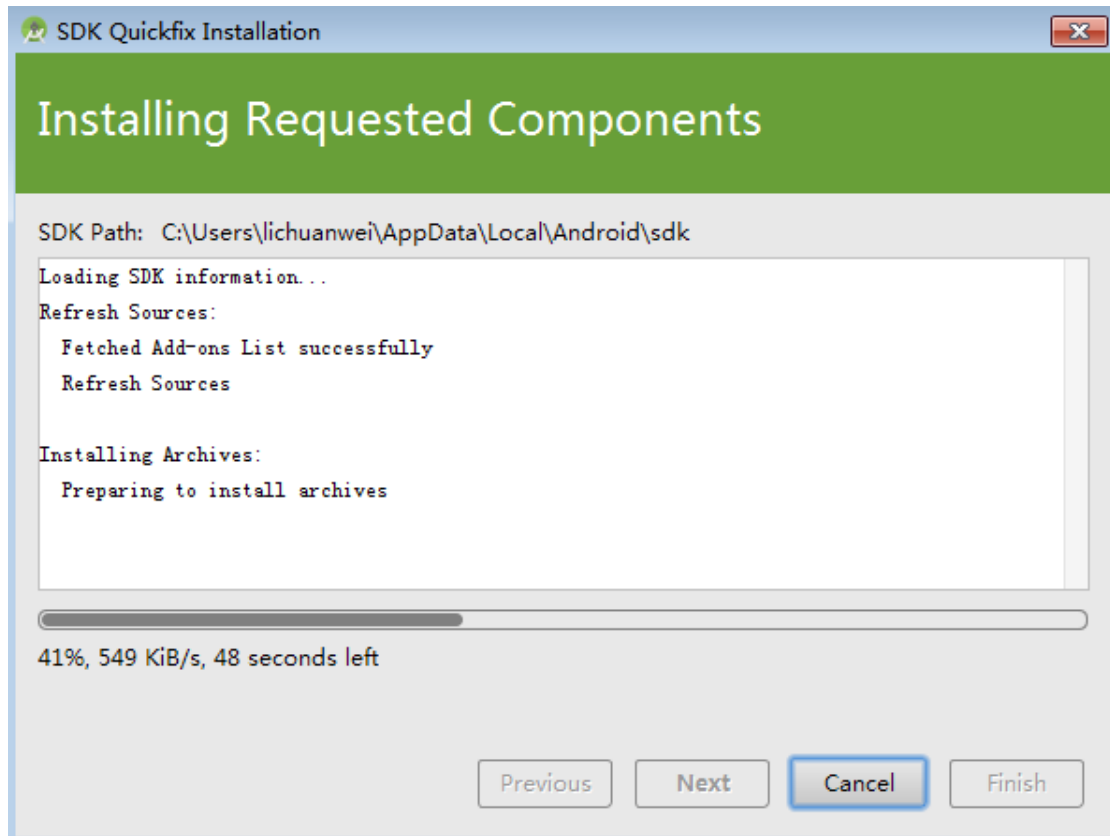


图 3-6

设置好模拟器后，单击图 2-6 运行，便启动模拟器

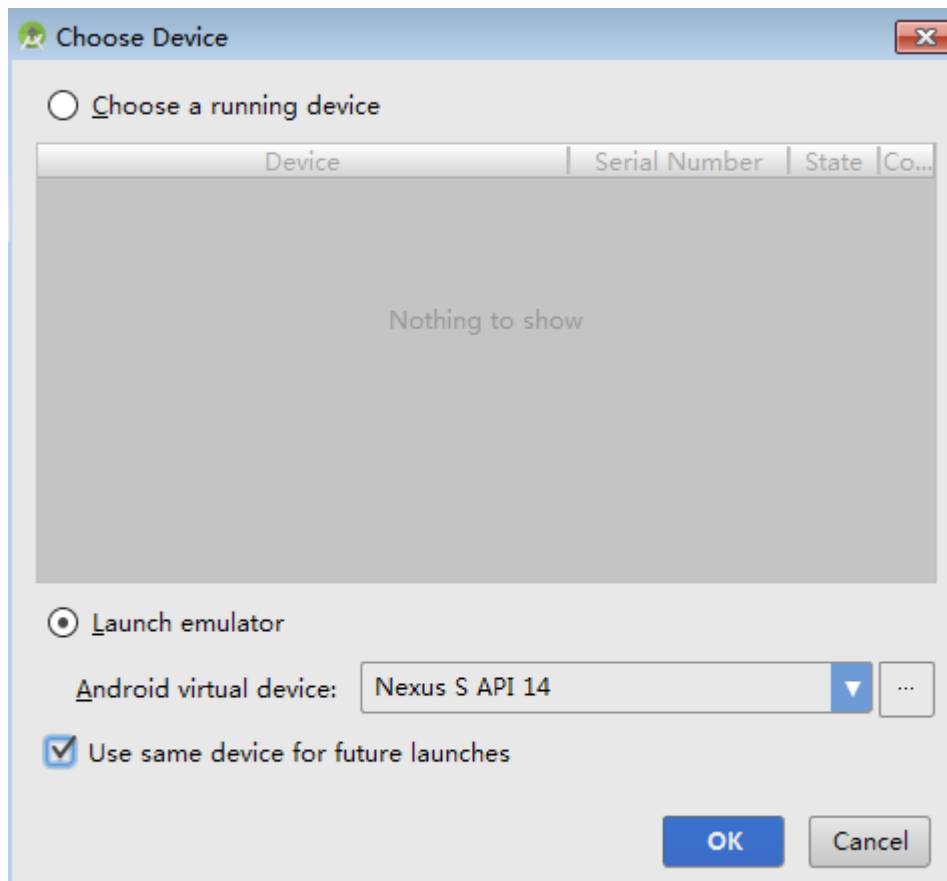


图 3-7

## 1.5.2 运行在手机上

建议初学者仅做了解，跳过这一步，使用后面介绍的模拟器运行。

### 1. 设置手机


1.1 使用 USB 线连接手机和你开发项目的计算机上。

1.2 打开设备的 USB debugging 功能

(1) 在 Android3.2 或更老版本的手机上，设置->应用程序->开发

(2) 在 Android4.0 或更新版本的手机上，开发者选项默认是隐藏的。打开这个功能，使用：设置->关于手机，连续点击**版本号**（英文名 Build number）7 次，返回上一页，会出现**开发人员选项**。

### 2. 从 Android Studio 运行 app

(1) 从工具栏单击 **Run** 

(2) 在弹出 Choose Device 窗口，选择 Choose a running device 单选按钮，选择你的设备，单击 OK

Android Studio 安装 app 到你连接的设备上，然后启动它。



## 第二章 Android Studio 基本概念

第一章快速演示了一个简单项目的建立过程，这一章详细介绍 Android Studio 创建项目的过程,学习 Android Studio 开发环境。本章的项目演示的是一个 APP 窗口打开另一个窗口，并传递参数。

### 2.1 开发环境

Android Studio 功能强大，开发环境要介绍的知识很多，要是全部介绍对于初学者而言太多，不易掌握。这里先学习开发项目中常用到的知识，其它知识需要的时候再逐步掌握。当你学习完第一章后，再次打开 Android Studio 会默认自动打开进入最近的一次项目，如图 2-1 所示。

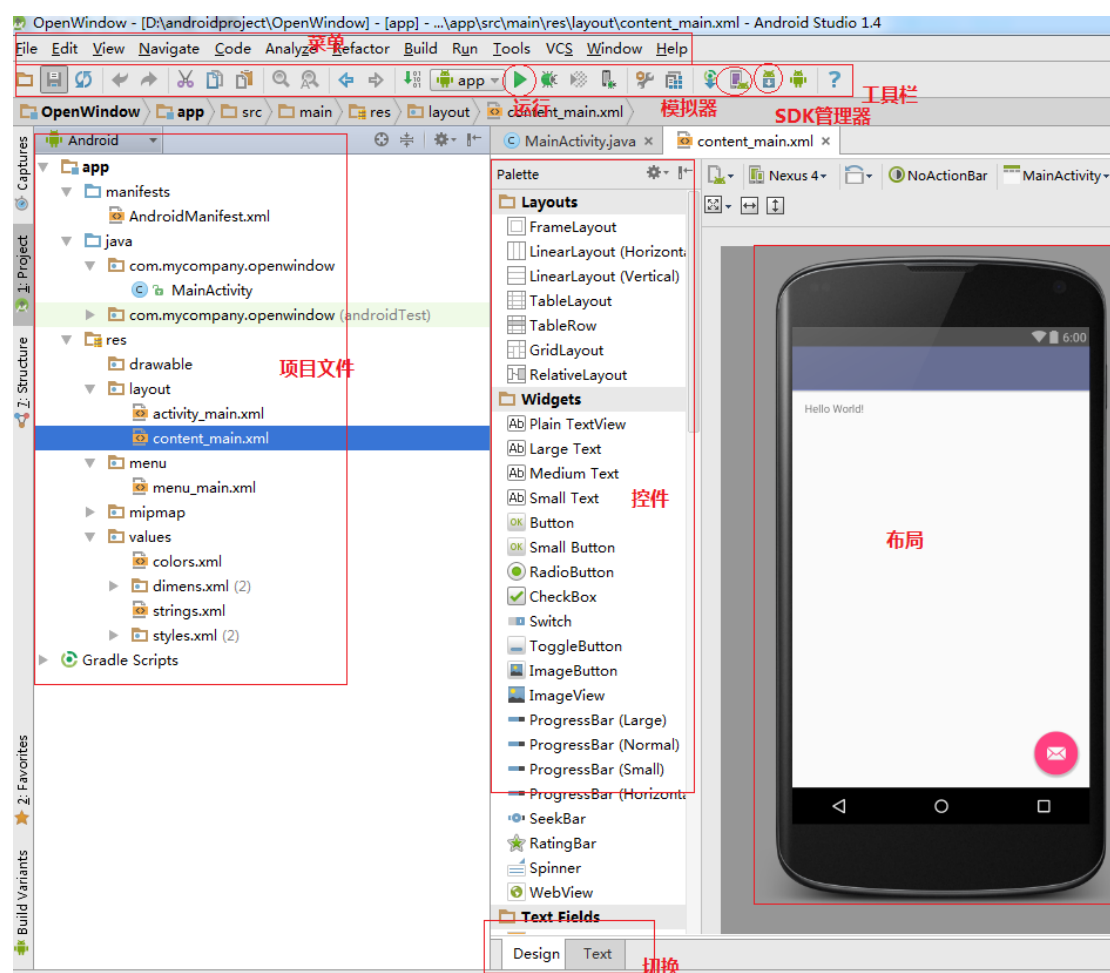


图 2-1 Android Studio 开发环境

#### 2.1.1 菜单

(1)新建项目

File->New->New Project...

(2)打开项目  
File->Open  
(3)关闭项目  
File->Close Project  
(4)退出  
File->Exit

## 2.1.2 工具栏

(1)运行：调试编译并运行项目  
(2)模拟器：模拟器管理器，创建、删除、设置、运行模拟器  
(3)SDK 管理器：下载安装 SDK

## 2.1.3 项目文件

按照功能不同，项目文件分布在不同文件夹里。

(1)Android 清单文件夹 app\manifests\

里面只有一个文件 AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mycompany.helloworld" >
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

这里都是一些关于应用包的元数据。<manifest> 属性有命名空间 `xmlns:android=http://schemas.android.com/apk/res/android` 是自动生成，包名 `package="com.mycompany.helloworld"`，就是我创建项目时填写的。另外还有一些属性没有列出来，如应用程序的版本号，我们可以发布自己的 Android 应用，然后，我们可以修改漏洞、

添加新特性，可以持续更新应用程序，用户就会在手机上收到通知，提示有新版本可用，他们于是可以下载并更新，所以版本号就是这样 1.5、2.0 之类的一段信息。

清单还定义了子元素 `application`(应用)，其属性有 `android:icon`(图标)。我们知道任何手机应用都会有图标，用户通过点击它打开程序，图标还可以自定义，这个元素就是起这个作用。`android:label` (标签)，用来显示应用名。

`Application` 内部有一个或多个 `activity`(活动)元素，所谓 `activity`，就是为用户执行的一项任务，一个 Android 应用程序可以有一个或多个活动。在这个应用程序，只有一个活动名为 `MainActivity`。另外还有 `intent-filter`(意图过滤器)元素，意图是指从活动转到其它活动，这个 `Manifest` 所用的意图过滤器，其名为 `android.intent.action.MAIN`，它的 `category`(类别)是 `launcher`(启动器)，这个特殊的意图就是如果用户点击了菜单中的应用图标，这就是开始要运行的活动，这就像 C/C++ 中的 `main` 程序一样，这个意图过滤器定义了应用程序的进入点。

(2)代码文件夹 `app\java\com.mycompany.helloworld\`

这个文件夹是放代码文件的地方，目前只有文件 `MainActivity`

(3)资源文件夹 `app\res\`

这个文件夹存放项目所需的各种资源，包括子文件夹：

1)图片文件夹 `drawable`

2)布局文件夹 `layout`

布局就是界面设计，Android 使用 XML 文件定义用户界面，用 XML 文件描述屏幕视觉元素的布局，在本应用程序中，当我使用项目创建向导时，它创建了这个 `activity_main.xml` 和作为 `activity_main.xml` 一部分的 `content_main.xml`。

3)菜单文件夹 `menu`

文件夹下有一个文件 `menu_main.xml`

4)图片文件夹 `mipmap`

这个文件夹也是存放图片的，和 `drawable` 作用相同，但是用 `mipmap` 系统会在缩放上提供一定的性能优化。目前这里存放着启动图标，不同的图标适应不同的屏幕。

5)键值

包括颜色、适配不同屏幕、字符串、样式。我们重点学习 `strings.xml`。

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="action_settings">Settings</string>
</resources>
```

用户界面上文本是字符串，这些字符串可以放在 `strings.xml` 里。如 `<string name="app_name">HelloWorld</string>`，它的键是 `app_name`，值是 `HelloWorld`。试着把 `HelloWorld` 改为“你好，世界！”总要指定字符串，这个字符串资源文件可以管理所有的 UI 文本，使得开发人员容易地管理、更新文本，以及 app 多语言国际化。Android 应用或其它移动应用发布到国际上，可能需要应用不同翻译版本，这类数据不应该嵌入源码中，二进制代码不需要改变，中文版和英语版的代码应该一样，所有供用户阅读的文本数据，都应该提出来，放到资源 XML 文件中，需要和代码分开。

## 2.1.4 视图 view

Android Studio 有丰富的控件，如按钮、文本框、进度条等等成为视图 `View`，以后我们会逐步应用到它。

## 2.1.5 布局

布局可以看出界面设计，可以把控件拖拽到布局上，也可以写布局代码设计，单击底部的 Design 和 Text，可视化布局和代码布局切换。

尽管做了介绍，但是只有通过项目才能掌握 Android Studio 开发环境。下面我们通过一个项目边干边学。

## 2.2 建立简单的用户界面

Android Studio 代码和界面（布局）是分开的，如前所述，分别放在不同文件夹下。

一般来说，一个代码文件对应一个界面（布局）文件。如

app\java\com.mycompany.helloworld\MainActivity 是代码文件，它对应的布局文件是 res\layout\activity\_main.xml，Android Studio 用 XML 文件描述界面。可 content\_main.xml 是什么呢？切换到底部 Text 选项卡，出现了 activity\_main.xml 文本，里面有一行：`<include layout="@layout/content_main" />`

content\_main.xml 是 activity\_main.xml 的一部分，activity\_main.xml 定义了顶部和底部的母版，所以我们直接操作 content\_main.xml 就可以了，现在打开它。它是非常简单的 UI 布局，content\_main.xml 的根节点 `RelativeLayout` 是表示相对布局的方式，后面我们会专门学习布局。作为初学者，我们学习最简单的一种布局：LinearLayout（线性布局）。把 `RelativeLayout` 替换成 `LinearLayout`，并删除掉 `<TextView>` 控件，结果是这样：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_main" tools:context=".MainActivity">
</LinearLayout>
```

LinearLayout 能让我们以线性形式布局一系列视图元素，里面的控件可以按从左到右或自上而下自动排列。控制水平布局还是垂直布局的属性 `android:orientation`（方向），设为 `vertical`，这些视图将以垂直方式排列。现在设为 `horizontal`，让它内部控件按水平方向排列。`android:layout_width="match_parent"` 和 `android:layout_height="match_parent"` 分别指示这个边界分别和父容器的宽度和高度同等，它的父容器就是手机屏幕，这样，app 就能填充整个屏幕。

这里要实现的是在 XML 里创建包括文本和按钮的 View，这里 View 可以理解为视图、控件。当按下按钮，会发送文本内容到其它界面处理。现在我们往里面添加控件(view)，用 Design 模式拖拽控件到界面上，但为了省去寻找 view 的麻烦，又训练我们对 view 的理解，在 Text 模式手工填写：

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_main" tools:context=".MainActivity">
    <EditText android:id="@+id/edit_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message"/>
</LinearLayout>

```

添加了编辑框 EditText 和按钮 Button，我们看看 EditText 的写法。

**android:id="@+id/edit\_message"**：识别 view 的唯一的 id，有了这个 id，在程序里可以用 id 操作这个对象。定义 id 的格式为：**@+id/名称**，注意**@+id/**是固定的，反斜杠后的名称才是你自由取名字。引用 id 的格式为：**R.java.资源名称**，我们学习代码时会用到。这种方式资源 ID 只声明一次，引用 ID 不需要加上+号。仅在指定新资源 ID 是才需要+号，诸如 string、布局等具体资源不需要+号。

**android:layout\_width** 和 **android:layout\_height**：可以不用具体的数值，**wrap\_content** 意思是能够包裹住它内容的，仅用于适应视图内容，表示宽度或高度上需要用多少就是多少，而 EditText 使用默认字体，平台知道该字体有多高，由此可知该 EditText 需要的高度。如果是 **match\_parent**，EditText 元素会充满屏幕，因为它将要适应父元素 **LinearLayout** 的大小。

**android:hint="@string/edit\_message"**是编辑框空的时候出现的默认字符串，也就是这个编辑框的提示文本，而文本属性值是**@string/edit\_message**，是因为引用具体资源（而不是标识符），不需要+号。这个字符串的具体值定义在另一个文件的字符串资源上。前面我谈到的软件国际化，应用中的字符串不要硬编码，因为发布的时候，可能需要面向多个国家，这些字符应该独立出来。做法是这样：**@string** 表示该字符串定义在其它 XML 文件中，它在 **values** 目录下，名为 **string.xml**，打开它，其中有一个 **string** 元素，**name** 属性为 **hello**，内容是 **"Hello World,IntroActivity!"**回头看 **main** 文件，该引用是说：要输入到屏幕的这个字段，实际数据在 **string.xml** 文件中，实际显示的是这一段。如果要翻译为英语，只需新建一个文件，一个英语版的 **string.xml** 即可。程序没有变化，只是使用不同资源数据来填充字段。**"@string/edit\_message"**因为尚未定义字符串资源，这里会有个编译错误提示，下面我们定义它。

## 2.3 字符串资源文件

我们把 app 的显示文本定义在字符串资源文件里，可以统一管理。打开 Android 项目字符串资源文件 **app\res\values\strings.xml**，新增两行：

```

<string name="edit_message">输入一条信息</string>
<string name="button_send">发送</string>

```

第一个作为上面编辑框的提示文本，第二个是按钮视图的文本，我们后面会用到。增加

后的文件内容为:

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="action_settings">Settings</string>
    <string name="edit_message">输入一条信息</string>
    <string name="button_send">发送</string>
</resources>
```


## 2.4 增加一个按钮

回到 app\res\layout\activity\_my.xml

1. 在<LinearLayout>元素里, 在<EditText>元素后定义一个<Button>元素
2. 设置按钮的宽和高属性为“wrap\_content”, 这样按钮只适应其文本标签所需的大小
3. 定义按钮文本 android:text 属性值为 button\_send 字符串资源
4. 增加 android:onClick 点击按钮事件.

下面是完成后的内容:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_main" tools:context=".MainActivity">
    <EditText android:id="@+id/edit_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send"/>
</LinearLayout>
```

到此, 可以单击  运行 app 了, 看看效果如何。



## 2.5 创建一个新的窗体步骤

Content\_main.xml 是 Hello World 程序用户界面的定义文件，也许你会问，那如何让程序运行在屏幕上呢？下面我就一步步

### 2.5.1 第一步：创建布局文件

在 app\res\layout\ 上单击右键，如图 5-1，选择 New>Layout resource file，在弹出的窗口填写文件名：activity\_display\_message

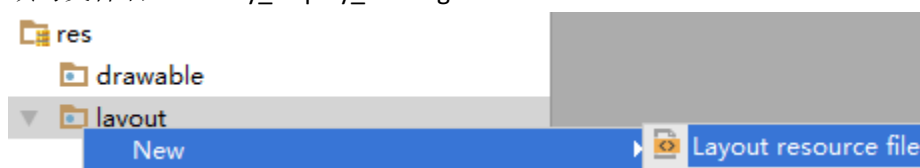
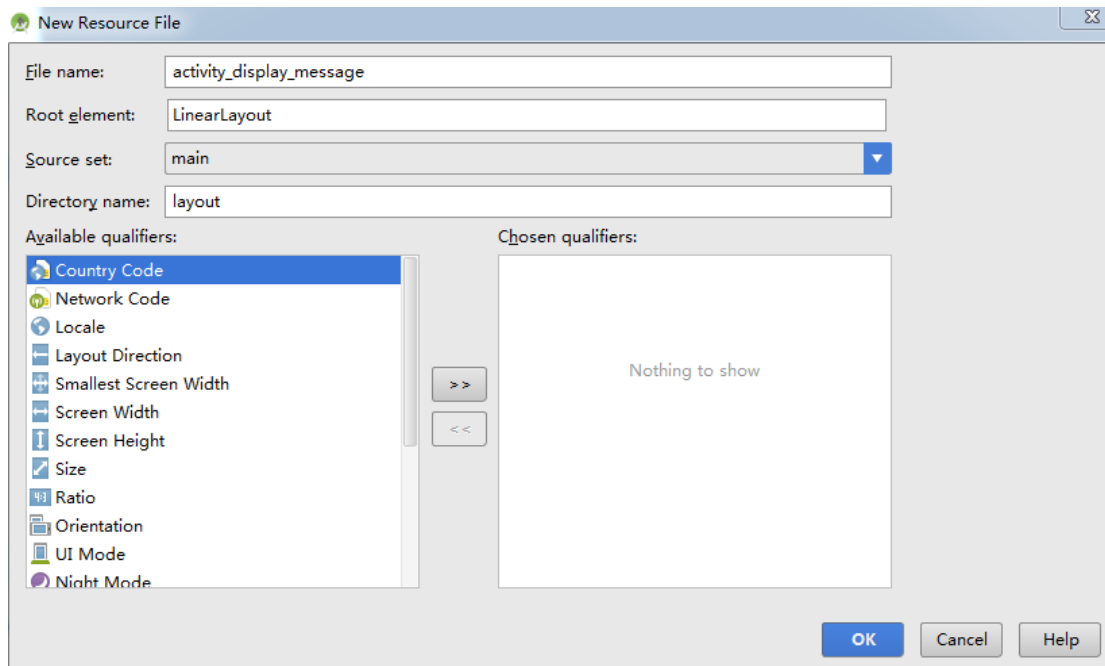


图 5-1



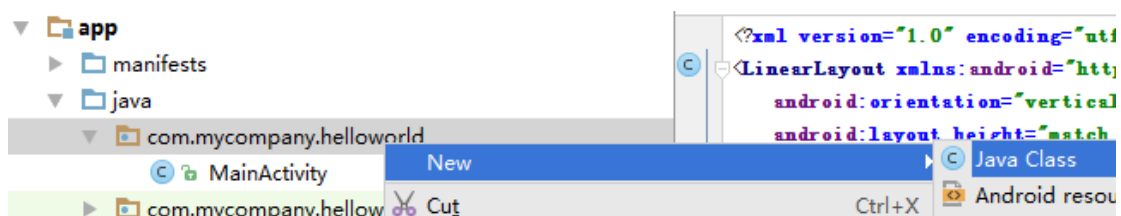


在 activity\_display\_message.xml 增加一个 TextView 视图，TextView 不具有编辑功能，只是显示文本，它将显示传递来的字符串。增加后的内容为：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView android:id="@+id/tvShow"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

## 2.5.2 第二步：创建对应的 Activity

在 app\java\com.mycompany.helloworld 上单击右键，选择 new>Java Class，在弹出窗口中 name 填写：DisplayMessageActivity，然后单击 OK。

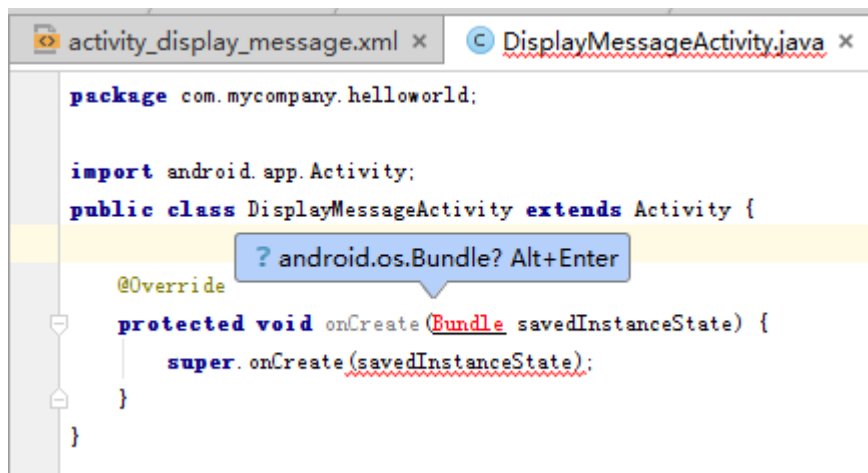


让这个类扩展一个基类 Activity，修改后

```
package com.mycompany.helloworld;
import android.app.Activity;
public class DisplayMessageActivity extends Activity {
    }
}
```

## 2.5.3 第三步：布局文件与 Activity 关联

显示到屏幕上的是布局文件，如何让它显示到屏幕上呢？要做的工作是让代码文件 Activity 启动时把布局文件显示出来。在类 DisplayMessageActivity 重写了唯一方法 onCreate，后面我们会讲 Activity 的生命周期，Activity 首次启用时的状态是创建 onCreate。这就要调用 onCreate 钩子方法，基类中的 onCreate 方法：`super.onCreate(savedInstanceState)`；每个 activity 都要进行一些必要的初始化，而这些初始化就是通过调用父类的构造函数来进行的，举个例子，生成一个空白的 activity，起码有一个标题，这个标题是怎样关联和生成的，就是通过构造函数来处理。代码如图所示，这时会出现错误，这是因为相关的类没有导入，按下 `Alt + Enter` 自动导入缺失的类。



接着要调用方法 `setContentView(R.layout.activity_display_message)`；`setContentView` 就是说活动创建以后，这就是我需要渲染到屏幕上的视图——这个视图就是 `R.layout.activity_display_message`，也就是就是在第一步我们编写 XML 布局文件描述 UI 的 `activity_display_message` 文件，放置在 `layout` 目录下。`setContentView(R.layout.activity_display_message)` 是在告诉机器，`activity_display_message` 中定义有布局 `layout`，这就是我要输出到屏幕的。这就是 Java 中用 XML 定义 UI，及程序上引用和操纵它们的方式。完整的文件如下：

```
package com.mycompany.helloworld;

import android.app.Activity;
import android.os.Bundle;

public class DisplayMessageActivity extends Activity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_display_message);
    }
}
```

## 2.5.4 第四步：在 AndroidManifest.xml 里声明

打开 app\manifests\AndroidManifest.xml 声明第三步添加的 activity，在</application>前添加一行：

```
<activity android:name=".DisplayMessageActivity"/>
```

完成后，完整的 AndroidManifest.xml 为：

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportRtl="true"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".DisplayMessageActivity"/>
    </application>

</manifest>
```

## 2.6 调用一个窗体

这里重点理解 Intent。如何从活动转到另一个活动，当应用中包含一个或多个活动时，什么能让我们从一个转到另一个，在 Android Studio 中，我们称之为 Intent(意图)。Intent 是从一个活动传到另一个的异步信息，一般 Intent 包括两段数据，首先是动作，然后是该动作所需数据。

### 2.6.1 调用方

上面我们建立了完整的 Activity，下面要调用它。点击在 ContentMain.xml 定义的按钮，打开上面创建的 Activity。在 ContentMain.xml 按钮定义里增加一个 android:onClick 属性：

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

"sendMessage"是按钮单击事件的函数，我们要在 MainActivity 实现它，在 onCreate 后新增方法 sendMessage。

```
protected void sendMessage(View view) {

}
```

(1) 创建 Intent 用于启动 DisplayMessageActivity 窗体。

打开 app\java\com.mycompany.helloworld\MainActivity.java，在里面添加一个方法 sendMessage()。

```
public void sendMessage(View view) {
    Intent intent = new Intent(this, DisplayMessageActivity.class);
}
```

Intent 有两个参数

第一个是 Context 类型参数，使用 this 因为 Activity 类是 Context 的子类

第二个是系统要传递的目标(这里是要启动的窗体)app 组件的类

这时，Android Studio 会指示你必须导入 Intent 类。在文件顶部，导入 Intent 类

```
import android.content.Intent;
```

注意：按下 Alt + Enter 自动导入缺失的类

(2) 使用 findViewById() 获取当前编辑框 EditText 对象，把编辑框里的值取出赋值到字符串变量 message。

```
EditText editText=(EditText) findViewById(R.id.edit_message);
String message=editText.getText().toString();
```

注：导入 EditText 类 import android.widget.EditText;

(3) 用 putExtra() 方法把 message 作为附加数据加到 intent

```
intent.putExtra("EXTRA_MESSAGE",message);
```

intent.putExtra() 是 Intent 附加参数的方法，可以附加一个键-值对数据类型的值，其第一个参数"EXTRA\_MESSAGE"是键，第二个参数 message 是值，接收方根据"EXTRA\_MESSAGE"取出 message 值。

(4) 调用 startActivity 方法，传递 Intent 对象

```
startActivity(intent);
```

```
package com.mycompany.helloworld;
```

```
import android.content.Intent;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
```

```

import android.view.Menu;
import android.view.MenuItem;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show();
            }
        });
    }

    public void sendMessage(View view) {
        Intent intent = new Intent(this, DisplayMessageActivity.class);
        EditText editText=(EditText) findViewById(R.id.edit_message);
        String message=editText.getText().toString();
        intent.putExtra("EXTRA_MESSAGE",message);
        startActivity(intent);
    }
}

```

## 2.6.2 被调用方

被调用的 activity 启动时要处理的工作在 onCreate() 方法里实现。activity 收到带有 message 的 Intent，然后处理它。onCreate() 方法也必须用 setContentView() 定义 activity 布局。打开 app\java\com.mycompany.helloworld\activity\_display\_message.java， getIntent() 获取传入的 Intent 调用方，getStringExtra("EXTRA\_MESSAGE") 取得"EXTRA\_MESSAGE"代表的值 message。

获取 xml 定义的 TextView，把 message 显示出来。完整代码如下：

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_display_message);

    Intent intent=getIntent();
    String message=intent.getStringExtra("EXTRA_MESSAGE");
}

```

```
TextView textView=(TextView)findViewById(R.id. tvShow);  
textView.setText(message);//设置文本为 message  
}
```

最后运行文件

因为模拟器没装中文输入法，只能输入英文

## 无法打开另一个窗体时检查：

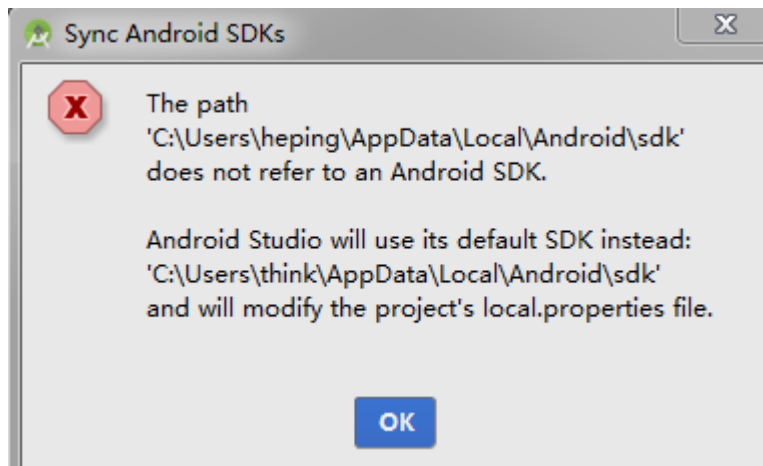
1. onCreate()

setContentView(R.layout.窗体布局文件);

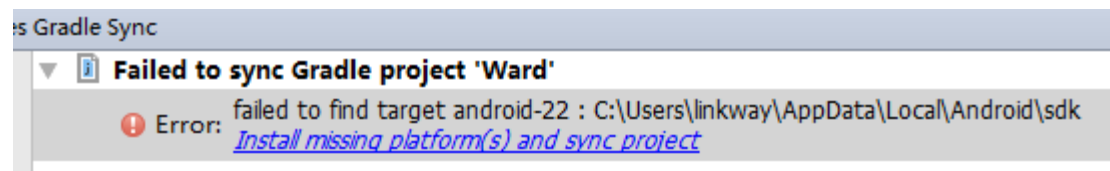
2. Manifest

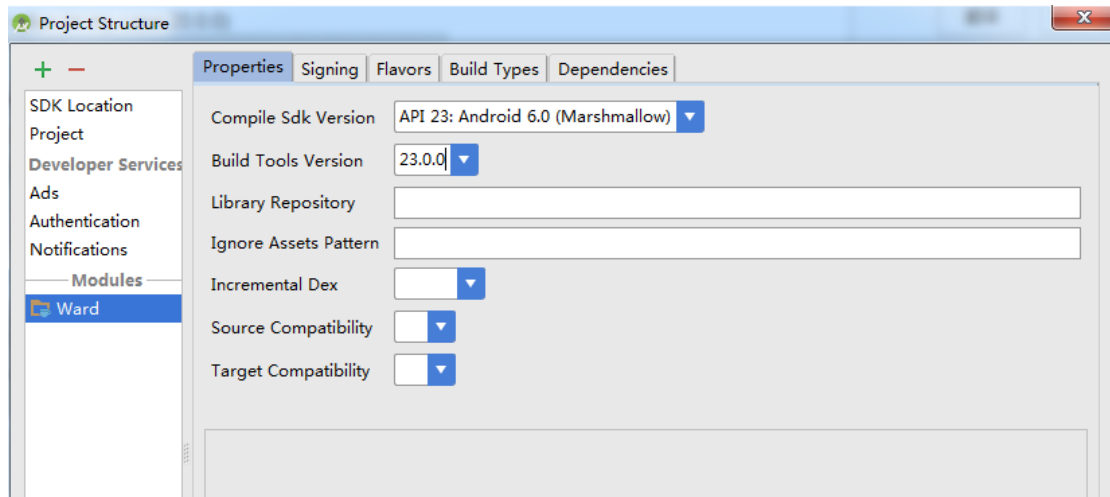
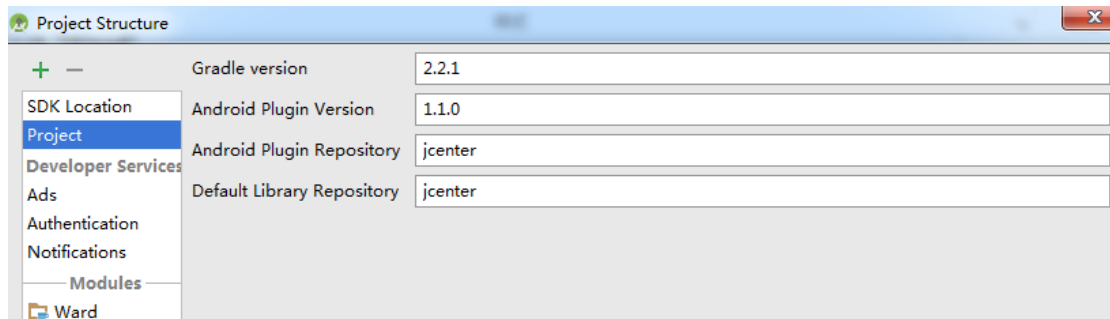
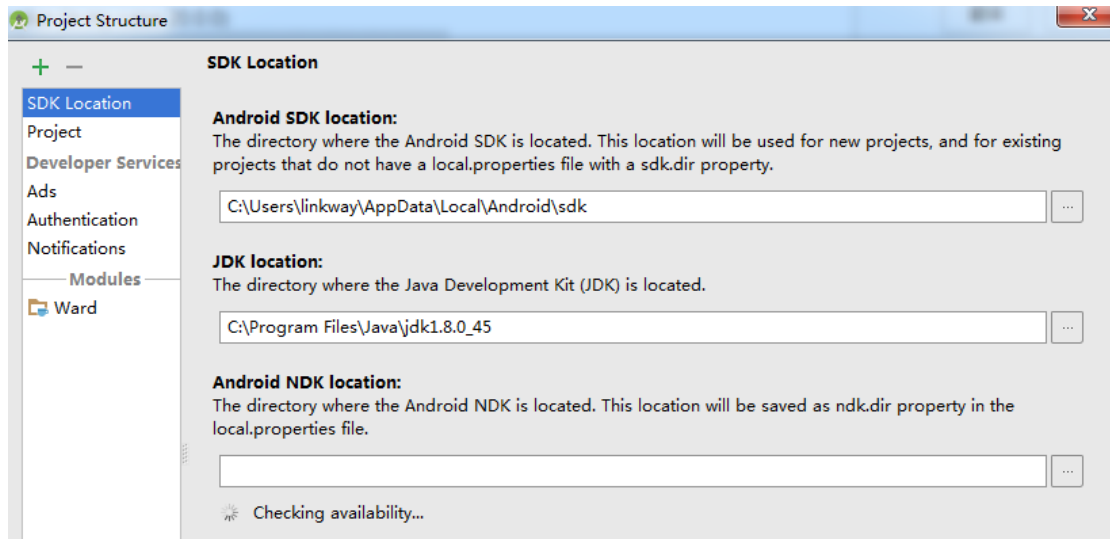
3.所有页面控件名称必须唯一

建议你一步一步创建项目，如果你直接打开使用示例项目，初次打开需要重新设置这个项目路径，单击“OK”即可



## 缺少 SKD 平台







## 第三章 Android Studio 编程语言基础

Android Studio 是 app 开发工具，编写 APP 程序的语言是 Java。Java 语言内容丰富，在大学里至少要学习一个学期，所以我们在这一节课内不可能全面讲解 Java，我们只是学习 Android Studio 开发常用的部分。有过 Java 编程经验的同学也不要跳过，看看 Java 在 Android Studio 的应用；没有 Java 编程经验的同学除了要学习这次课，还要在课外更多地补充 Java 编程技术。

我们一开始学习 Android Studio 就接触到的 JDK (Java Software Develop Kit) 是 Java 软件开发工具包。JDK 是 Java 的核心，包括了 Java 运行环境，一系列 Java 开发工具和 Java 基础的类库。我们还设置了 JDK 环境变量，其中

**PATH:** 提供给操作系统寻找到 Java 命令工具的路径。通常是配置到 JDK 安装路径\bin  
**JAVA\_HOME:** 提供给其它基于 Java 的程序使用，让它们能够找到 JDK 的位置。通常配置到 JDK 安装路径。注意：这个必须书写正确，全部大写，中间用下划线。

**CLASSPATH:** 提供程序在运行期寻找所需资源的路径，比如：类、文件、图片等等。

注意：在 windows 操作系统上，最好在 classpath 的配置里面，始终在前面保持 “.” 的配置，在 windows 里面 “.” 表示当前路径。

首先我们按照前面的方法新建一个项目，应用程序名称为 MyJava，其它设置类似前一个项目。把 app\res\layout\content\_main.xml 按下面修改

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_main" tools:context=".MainActivity">
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <Button
            android:onClick="btnDivideClick"
            android:text="计算"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <EditText
            android:id="@+id/etFirst"
            android:layout_width="50dp"
            android:layout_height="wrap_content" />
```

```

<TextView
    android:text="+"
    android:layout_width="40dp"
    android:layout_height="wrap_content" />

<EditText
    android:id="@+id/etSecond"
    android:layout_width="50dp"
    android:layout_height="wrap_content" />
<TextView
    android:text="="
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<TextView
    android:id="@+id/tvResult"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
</LinearLayout>
</LinearLayout>

```

这个布局的意思是：单击按钮【计算】，把编辑框`etFirst`和编辑框`etSecond` 的两个数相加，结果写入`vResult`。

## 3.1 Java 代码的基本知识

### 1. 语句

用分号；结尾的一行代码就是语句，Java 中语句必须以；结尾。

如：int a = 10;

### 2. 注释

用来对Java 代码进行说明，但不影响程序。Java 中有三种注释方式

//：注释单行语句

int a = 10; //定义一个值为10 的int 变量

/\* \*/：多行注释

/\*

这是一个注释，不会被Java 用来运行

这是第二行注释，可以有任意多行

\*/

/\*\* \*/：文档注释

紧放在变量、方法或类的声明之前的文档注释，表示该注释应该被放在自动生成的文档

### 3. 空格

在一个Java 程序中任何数量的空格都是允许的

### 4. 标识符

标识符是赋予变量、类或方法的名称。首字母只能以字母、下划线、\$开头,其后可以跟字母下划线、\$和数字。如：\$abc、\_ab、ab123 等都是有效的。标识符区分大小写，标识符不能是关键字，标识符长度没有限制。

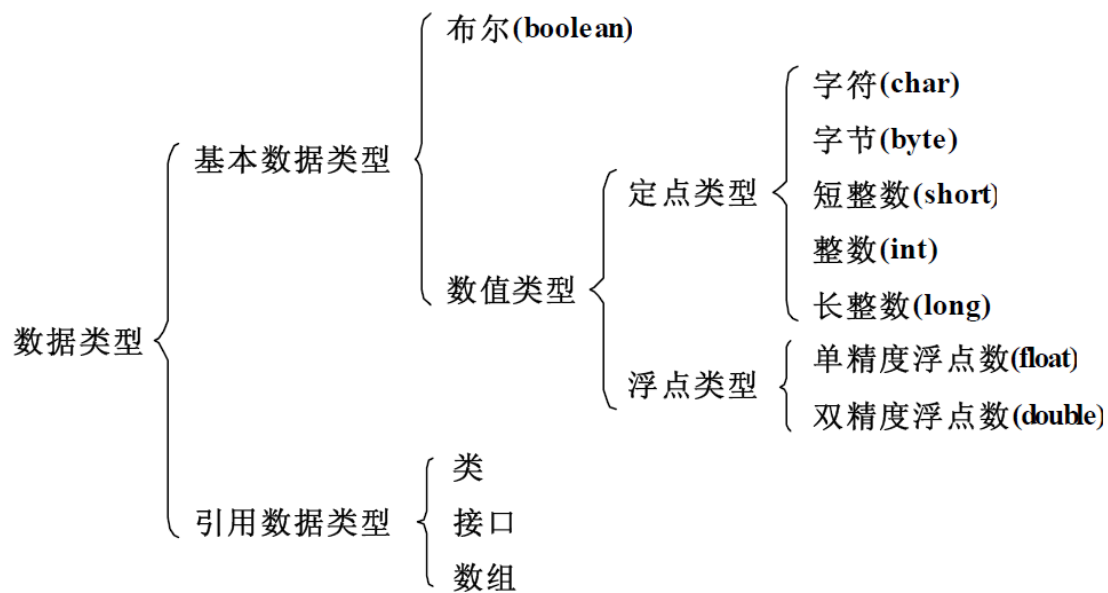
## 5.声明

声明为 **Java** 程序实体引入标识符，能够使用这些名字访问到这些实体，声明实体包括：类名、属性名、方法名、变量名、参数名、接口名等等。其实简单点说就是定义某个东西并对外宣称它。

## 6.赋值

赋值就是为一个声明的变量或者常量赋予具体的值，也就是赋予值的意思。用“=”来表示。`int a = 5;`这句话的意思就是，声明一个类型为 `int` 的变量 `a`，并将它赋值为 `5`。

# 3.2 数据类型



**Java 数据类型层次图。**

数据类型简单的说就是对数据的分类，对数据各自的特点进行类别的划分，划分的每种数据类型都具有区别于其它类型的特征，每一类数据都有相应的特点和操作功能。例如数字类型的就能够进行加减乘除的操作。**Java** 里面的数据类型从大的方面分为两类，一是基本数据类型，一是引用类型。

## 3.2.1 常用数据类型

1. 整数型：`int`、`long`，取值范围不同，整数型的值，如果没有特别指明，默认是 `int` 型
  2. 浮点型：`float`、`double`，浮点型的值，如果没有特别指明，默认是 `double` 型的
- Java** 用浮点型来表示实数，简单地说就是带小数的数据。如果一个数字包括小数点或指数部分，或者在数字后带有字母 `F` 或 `f` (`float`)、`D` 或 `d` (`double`)，则该数字文字为浮点型的。定义 `float` 型的时候，一定要指明是 `float` 型的，可以通过在数字后面添加“`F`”或者“`f`”来表示。定义 `double` 型的时候，可以不用指明，默认就是 `double` 型的，也可以通过在数字后面添加“`D`”或者“`d`”来表示。

`float abc = 5.6F;`

`float abc = 5.6f;`

`double abc = 5.6;`

```
double abc = 5.6D;
```

```
double abc = 5.6d;
```

3. 字符型：char，用来表示单个字符。

```
‘a’ //表示简单的字符
```

```
‘1’ //用数字也可以表示字符
```

转义字符是指，用一些普通字符的组合来代替一些特殊字符，由于其组合改变了原来字符表示的含义，因此称为“转义”。常见的转义字符：

```
\n 回车(\u000a)
```

```
\t 水平制表符(\u0009)
```

```
\b 空格(\u0008)
```

```
\r 换行(\u000d)
```

```
\f 换页(\u000c)
```

```
\' 单引号(\u0027)
```

```
\" 双引号(\u0022)
```

```
\\ 反斜杠(\u005c)
```

4. 逻辑型：boolean，有两个文字值，即true 和false。

```
boolean truth = true; //声明变量truth
```

5. 字符串型 String，Java 中使用String这个类来表示多个字符，表示方式是用双引号把要表示的字符串引起来，字符串里面的字符数量是任意多个。字符本身符合Unicode 标准，且上述char 类型的反斜线符号（转义字符）适用于String。

```
// 声明两个字符串变量并初始化它们
```

```
String greeting = "Good Morning !! \n" ;
```

```
String str1,str2 ; // 声明两个字符串变量
```

```
String s = "" ; //表示空串
```

**注意：**

(1)：String **不是原始的数据类型，而是一个类(class)**

(2)：String包含的字符数量是任意多个，而字符类型只能是一个。

要特别注意：“a” 表示的是字符串，而’a’表示的是字符类型，它们具有不同的功能。

(3)：String的默认值是null

6.布尔型，值为true或false boolean flag=true;

下列程序显示了如何为整数、浮点数、boolean、字符和String类型变量声明和赋值

```
1. public class Assign {
```

```
2. public static void main(String args []) {
```

```
3. int x, y; // 声明 int 变量
```

```
4. float z = 3.414f; // 声明并赋值 float
```

```
5. double w = 3.1415; //声明并赋值 double
```

```
6. boolean truth = true; //声明并赋值boolean
```

```
7. char c; // 声明字符变量
```

```
8. String str; // 声明String 字符串
```

```
9. String str1 = "bye"; //声明并赋值 String 变量
```

```
10. c = 'A'; // 给字符变量赋值
```

```
11. str = "Hi out there!"; // 给String 变量赋值
```

```
12. x = 6;
```

```
13. y = 1000; // 给int 变量赋值
```

14....

15.}

16.}

非法赋值举例

`y = 3.1415926;` // 3.1415926 不是一个 int.

// 需要类型转换并且小数位要截掉

`w = 175,000;` // 逗号(,) 不能够出现

`truth = 1;` // 一个优秀的C/C++程序员常犯的错误, 在Java 语言中boolean 型变量只能为true或false

`z = 3.14156 ;` // double 型的值不能赋给float 变量, 需要类型转换

对于引用数据类型放到后面再学, 先看看常量和变量。

### 3.2.2 类型转换

在开发中常用到类型转换。

1. long与int直接转换

`long bigValue = 99L;`

`int squashed = (int) (bigValue);`

2. 字符串转换其它类型

整数=`Integer.parseInt(字符串)`

小数= `Double.parseDouble(字符串)`

3. 其它类型转字符串

字符串=`String.valueOf(其它类型)`

### 3.2.3 常量和变量

1.什么是常量

常量是值不可以改变的标识符。常量的定义, 建议大家尽量全部大写, 并用下划线将词分隔。如: `JAVASS_CLASS_NUMBER` , `FILE_PATH`

2: 什么是变量

变量是值可以改变, 存储着某一类型的数据。如声明了整数类型的变量 i, 赋值为 9, i 里存放着数据 9。 `int i=9;`

### 3.2.3 在 Android 开发中的运用

我们做一个简单的除法器来学习上面学习到的内容。

声明两个类型是编辑框的变量:

`EditText etFirst=(EditText)findViewById(R.id.etFirst);`

`EditText etSecond=(EditText)findViewById(R.id.etSecond);`

还有一个是文本框的变量:

`TextView tvResult=(TextView)findViewById(R.id.tvResult);`

注意我们在程序中引用 View 时, 使用方法 `findViewById()`, 就是通过 view 的 Id 找到这

个 View，Id 的名称的获得方式为：R.id.你定义的名称。注意 findViewById()前面要加上强类型转换的类型名称。这样可以操作 View 了。

定义两个 double 类型变量，初始值为 0:

```
double numFirst=0;
double numSecond=0;
```

定义两个字符串变量，它们的值分别为用户填写在编辑框里的内容，取编辑框内容的方法为：

编辑框名称.getText()，后面加上.toString()是把内容转换成字符串：

```
String strFirst=etFirst.getText().toString();
String strSecond=etSecond.getText().toString();
```

取到编辑框的内容并且存放在字符串变量里后，还要赋值到 double 型变量里才能进行运算，我们用上面学习的字符串转换小数的方法：

```
numFirst=Double.parseDouble(strFirst);
numSecond=Double.parseDouble(strSecond);
```

上面所有内容如下：

```
public void btnDivideClick(View view)
{
    EditText etFirst=(EditText)findViewById(R.id. etFirst);
    EditText etSecond=(EditText)findViewById(R.id. etSecond);
    TextView tvResult=(TextView)findViewById(R.id. tvResult);
    double numFirst=0;
    double numSecond=0;
    String strFirst=etFirst.getText().toString();
    String strSecond=etSecond.getText().toString();
    numFirst=Double.parseDouble(strFirst);
    numSecond=Double.parseDouble(strSecond);
}
```

## 3.3 运算符

### 3.3.1 算术运算

加：+

减：-

乘：\*

整除：/， 5/2 = 2 // 不是 2.5

mod：%

### 3.3.2 比较运算

比较运算是指：>、<、>=、<=、==、!= 等类似运算

### 3.3.3 逻辑运算

执行一个以上布尔逻辑表达式的组合判断

&&: “与”

||: “或”

### 3.3.4 示例

```
double result=numFirst+numSecond;  
tvResult.setText(String.valueOf(result));
```

## 3.4 控制语句

### 3.4.1 分支控制语句

分支语句又称条件语句，条件语句使部分程序可根据某些表达式的值被有选择地执行。Java 编程语言支持双路if 和多路switch 分支语句。

#### 1. if , else语句

if, else 语句的基本句法是：

```
if (布尔表达式) {  
    语句或块;  
} else {  
    语句或块;  
}
```

#### 2. switch语句

switch 表示选择分支的情况，switch 语句的句法是：

```
switch (expr1) {  
    case expr2:  
        statements;  
        break;  
    case expr3:  
        statements;  
        break;  
    default:  
        statements;  
        break;  
}
```

如果没有break 语句作为某一个case 代码段的结束句，则程序的执行将继续到下一个case，而不检查case表达式的值。

### 3.4.2 示例

我们这里要丰富上面计算器的功能，把只能做相加的功能改为可以让用户输入运算符，我们根据运算符计算。首先把 app\res\layout\content\_main.xml 文件里的

```
<TextView
    android:text="+"
    android:layout_width="40dp"
    android:layout_height="wrap_content" />
```

替换为：

```
<EditText
    android:id="@+id/etOperator"
    android:layout_width="40dp"
    android:layout_height="wrap_content" />
```

这样，我们获取这个运算符的对象，判断用户输入的是什么运算符，来决定做何种运算。注意这里字符串比较不要用==，而是用字符串比较方法 equals。

```
EditText etOperator=(EditText)findViewById(R.id. etOperator);
String operator=etOperator.getText().toString();
double result=0.0;
if(operator.equals("+")) result=numFirst+numSecond;
if(operator.equals("-")) result=numFirst-numSecond;
if(operator.equals("*")) result=numFirst*numSecond;
if(operator.equals("/")) result=numFirst/numSecond;
tvResult.setText(String.valueOf(result));
```

### 3.4.3 循环控制语句

循环语句使语句或块的执行得以重复进行。Java编程语言支持三种循环构造类型：for,while 和do 循环。for和while循环是在执行循环体之前测试循环条件，而do 循环是在执行完循环体之后测试循环条件。这就意味着for和while循环可能连一次循环体都未执行，而 do 循环将至少执行一次循环体。

#### 1. for 循环

for 循环的句法是：

```
for (初值表达式; boolean 测试表达式; 改变量表达式) {
    语句或语句块
}
```

例如：

```
for (int i = 0; i < 10; i++) {

}
```

#### 2. while 循环

while 循环的句法是：

```
while (布尔表达式) {
```



语句或块

}

例如:

```
int i = 0;
```

```
while (i < 10) {
```

```
    i++;
```

```
};
```

请确认循环控制变量在循环体被开始执行之前已被正确初始化, 并确认循环控制变量是真时, 循环体才开始执行。控制变量必须被正确更新以防止死循环。

### 3. do 循环

do 循环的句法是:

```
do {
```

```
    语句或块;
```

```
} while (布尔测试);
```

例如:

```
int i = 0;
```

```
do {
```

```
    i++;
```

```
} while (i < 10);
```

象while 循环一样, 请确认循环控制变量在循环体中被正确初始化和测试并被适时更新。作为一种编程惯例, for循环一般用在那种循环次数事先可确定的情况, 而while和do 用在那种循环次数事先不可确定的情况。

do 循环与while 循环的不同这处在于, 前者至少执行一次, 而后者可能一次都不执行。

## 3.4.4 示例

著名数学家高斯, 最著名的故事莫过于小学时计算  $1+2+3+\dots+100$  的值。我们用循环的方法计算吧。在 app\res\layout\content\_main.xml 里新增 view:

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    >
    <Button
        android:onClick="btnSumClick"
        android:text="计算"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:id="@+id/tvSum"
        android:text="1+2+3+4+5+...+100="
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

当单击【计算】按钮，计算结果附到 tvSum 的文本后面。下面把按钮事件写到 app\java\com.mycompany.myjava\MainActivity 里：

```
public void btnSumClick(View view)
{
    int sum=0;
    for(int i=1;i<=100;i++)
    {
        sum+=i;
    }
    TextView tvSum=(TextView)findViewById(R.id. tvSum);
    String strSum=tvSum.getText().toString()+String.valueOf(sum);
    tvSum.setText(strSum);
}
```

首先声明一个整数变量 sum，赋值为 0，循环从 1 开始，每循环一次自动加 1，直到 100，累加值到 sum。把原先 tvSum 的文本和计算结果连接，最后显示到 tvSum。

## 3.5 异常处理

上面的演示示例是无法交付用户使用的，因为应用程序的健壮性还不够。设想，用户胡乱输入，除数为0等等，程序会异常。在Java 编程语言中，异常类定义程序中可能遇到的轻微的错误条件。可以写代码来处理异常并继续程序执行，而不是让程序中断。例如，发生下列情况时，会出现异常：

- 想打开的文件不存在
- 网络连接中断
- 受控操作数超出预定范围
- 非常感兴趣的正在装载的类文件丢失

Java 提供了一种异常处理模型，它使您能检查异常并进行相应的处理。它实现的是异常处理的抓抛模型。使用此模型，您只需要注意有必要加以处理的异常情况。下面是Java语言中的异常处理块的模型：

```
try{
//放置可能出现异常的代码
}catch(Exception1 el){
//这里进行异常处理
}
finally{
//不管是否有异常发生，始终执行这个代码块
}
```

### 1: try块

try 块由一组可执行语句组成，在执行它们时可能会抛出异常。

也就是说：在try语句块中包含可能会产生异常的语句

会抛出一个异常，原因是试图用0 去除一个数。程序会被成功编译，但当运行该程序时，程序将会发生异常而中断，异常可在catch块中被捕获

### 2: catch块

catch 块，从其名称就可以看出，是用来捕获并处理try中抛出的异常的代码块。没有

try 块，catch 块不能单独存在。下面是catch 块的语法

```
try{
}catch(异常类型 e){
}
```

这里，e 是异常类型类的对象，利用这一对象，我们可以输出这一异常的详细信息

### 3: finally块

finally 块表示：**无论是否出现异常，都会运行的块**

通常在 finally 块中可以编写资源返还给系统的语句，通常，这些语句包括但不限于：

- (1) 释放动态分配的内存块；
- (2) 关闭文件；
- (3) 关闭数据库结果集；
- (4) 关闭与数据库建立的连接；

它紧跟着最后一个块，是可选的，不论是否抛出异常，finally 块总会被执行。

把btnDivideClick方法加上try..catch..finally，注意把View放在try的外面声明，这样catch和finally也能访问到。

```
public void btnDivideClick(View view) {
    TextView tvResult = (TextView) findViewById(R.id. tvResult);
    EditText etFirst = (EditText) findViewById(R.id. etFirst);
    EditText etSecond = (EditText) findViewById(R.id. etSecond);
    try {
        double numFirst = 0;
        double numSecond = 0;
        String strFrist = etFirst.getText().toString();
        String strSecond = etSecond.getText().toString();
        numFirst = Double.parseDouble(strFrist);
        numSecond = Double.parseDouble(strSecond);
        EditText etOperator = (EditText) findViewById(R.id. etOperator);
        String operator = etOperator.getText().toString();
        double result = 0.0;
        if (operator.equals("+")) result = numFirst + numSecond;
        if (operator.equals("-")) result = numFirst - numSecond;
        if (operator.equals("*")) result = numFirst * numSecond;
        if (operator.equals("/")) result = numFirst / numSecond;
        tvResult.setText(String.valueOf(result));
    } catch (Exception exc)
    {
        Toast.makeText(getApplicationContext(), exc.getMessage(),
        Toast.LENGTH_LONG).show();
    }
    finally {
        etFirst.setText("");
        etSecond.setText("");
    }
}
```

## 3.6 Android Studio 面向对象初步

### 3.6.1 类与对象

现代编程使用面向对象技术，是把现实的世界对象引用到编程技术上。我们通过一个例子来描述，在程序上，把“人”作为一个类(class)，“人”这个类有一些属性，如姓名、年龄等等；还有一些行为，如说话定义为方法。方法就是对象所具有的动态功能。方法可以有参数，参数分实参和形参。形参就是形式参数的意思，是在定义方法名的时候使用的参数，用来标识方法接收的参数类型，在调用该方法时传入。实参就是实际参数的意思，是在调用方法时传递给该方法的实际参数。

一个完整的Java 类通常由下面六个部分组成：

包定义语句

import语句

类定义{

成员变量

构造方法

成员方法

}

在app\java\com.mycompany.myjava上单击右键，选择New->Java Class，填写类名Person，定义的类为：

```
package com.mycompany.myjava;
```

```
public class Person {  
    private String Name;  
    private int Age;  
  
    public String getName() {  
        return Name;  
    }  
  
    public void setName(String name) {  
        Name = name;  
    }  
  
    public int getAge() {  
        return Age;  
    }  
  
    public void setAge(int age) {  
        Age = age;  
    }  
}
```

```

    }
    public String Talk()
    {
        String speak="我是 "+this.getName()+" ,我今年
"+String.valueOf(this.getAge());
        return speak;
    }
}

```

定义了“人”这个类，如何使用它呢？如果有小张、小李、老王等等具体的人，就不必一定定义这些属性和方法了，小张、小李、老王就是人这个类的对象，即使有无数个对象，也可以使用类的属性和方法。

Java 语言允许对类中定义的各种属性和方法进行访问控制，即规定不同的保护等级来限制对它们的使用。Java 语言为对类中的属性和方法进行有效地访问控制，将它们分为四个等级：private，无修饰符，protected，public，具体规则如下：

表 Java 类成员的访问控制

可否直接访问 控制等级	同一个类中	同一个包中	不同包中的 子类的对象	任何场合
private	Yes			
无修饰符	Yes	Yes		
protected	Yes	Yes	Yes	
public	Yes	Yes	Yes	Yes

一个protected 方法或变量可以从同一个包中的类当中的任何方法进行访问，也可以是从任何子类中的任何方法进行访问。当它适合于一个类的子类但不是不相关的类时，就可以使用这种受保护访问来访问成员。

## 3.6.2 使用类

前面学习了如何定义一个类，下面来学习如何使用一个类。首先在app\res\layout\content\_main.xml里新增view:

```

<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <TextView
        android:text="姓名: "
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <EditText
        android:id="@+id/etName"
        android:layout_width="60dp"

```

```

        android:layout_height="wrap_content" />
    <TextView
        android:text="年龄: "
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <EditText
        android:id="@+id/etAge"
        android:layout_width="60dp"
        android:layout_height="wrap_content" />
    <Button
        android:onClick="btnTalkClick"
        android:text="说话"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
<TextView
    android:id="@+id/tvTalk"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

```

### 1. new关键字

那么怎么来使用这个类呢：在你可以使用变量之前，实际内存必须被分配。这是通过使用关键字new 来实现的。

```
Person person=new Person();
```

### 2. 如何使用对象中的属性和方法

要调用对象中的属性和方法，使用“.”操作符。

```
person.setName(etName.getText().toString());
```

### 3. this关键字

关键字this是用来指向当前对象或类实例的。

## 3.6.3 Java 中的继承

### 1.extends关键字

继承关系使用extends 关键字，比如女人类和男人类继承Person类，具有Person 所拥有的所有变量及方法，还可以定义自己的属性和方法。

### 2. 关键字super

关键字super 可被用来引用该类的父类，它被用来引用父类的成员变量或方法。父类行为被调用，就好像该行为是本类的行为一样，而且调用行为不必发生在父类中，它能自动向上层类追溯。

**super关键字的功能：**

- (1) 点取父类中被子类隐藏了的数据成员
- (2) 点取已经覆盖了的方法
- (3) 作为方法名表示父类构造方法

请注意，super.method()格式的调用，如果对象已经具有父类类型，那么它的方法的整

个行为都将被调用，也包括其所有负面效果。该方法不必在父类中定义，它也可以从某些祖先类中继承。也就是说可以从父类的父类去获取，具有追溯性，一直向上去找，直到找到为止，这是一个很重要的特点。

### 3. 调用父类构造方法

在许多情况下，使用默认构造方法来对父类对象进行初始化，当然也可以使用`super`来显示调用父类的构造方法。

**注意：无论是`super`还是`this`，都必须放在构造方法的第一行。**

通常要定义一个带参数的构造方法，并要使用这些参数来控制一个对象的父类部分的构造。可能通过从子类构造方法的第一行调用关键字`super`的手段调用一个特殊的父类构造方法作为子类初始化的一部分。要控制具体的构造方法的调用，必须给`super()`提供合适的参数。当不调用带参数的`super`时，缺省的父类构造方法（即，带0个参数的构造方法）被隐含地调用。在这种情况下，如果没有缺省的父类构造方法，将导致编译错误。当被使用时，`super`或`this`必须被放在构造方法的第一行。显然，两者不能被放在一个单独行中，但这种情况事实上不是一个问题。如果写一个构造方法，它既没有调用`super(...)`也没有调用`this(...)`，编译器自动插入一个调用到父类构造方法中，而不带参数。其它构造方法也能调用`super(...)`或`this(...)`，调用一个`static`方法和构造方法的数据链。最终发生的是父类构造方法（可能几个）将在链中的任何子类构造方法前执行。

## 3.6.4 包

为便于管理数目众多的类，Java 语言中引入了“包”的概念。包是类、接口或其它包的集合，包主要用来将类组织起来成为组，从而对类进行管理。包对于下列工作非常有用，包允许您将包含类代码的文件组织起来，易于查找和使用适当的类；包不止是包含类和接口，还能够包含其它包。形成层次的包空间；它有助于避免命名冲突。当您使用很多类时，确保类和方法名称的唯一性是非常困难的。包能够形成层次命名空间，缩小了名称冲突的范围，易于管理名称。

将多个功能相关的类定义到一个“包”中，以解决命名冲突、引用不方便、安全性等问题。就好似当今的户籍制度，每个公民除有自己的名字“张三”、“李四”外还被规定了他的户籍地。假定有两个人都叫张三，只称呼名字就无法区分他们，但如果事先登记他们的户籍分别在北京和上海，就可以很容易的用“北京的张三”、“上海的张三”将他们区分开来。如果北京市仍有多个张三，还可以细分为“北京市.海淀区的张三”、“北京市.西城区.平安大街的张三”等等，直到能惟一标识每个“张三”为止。

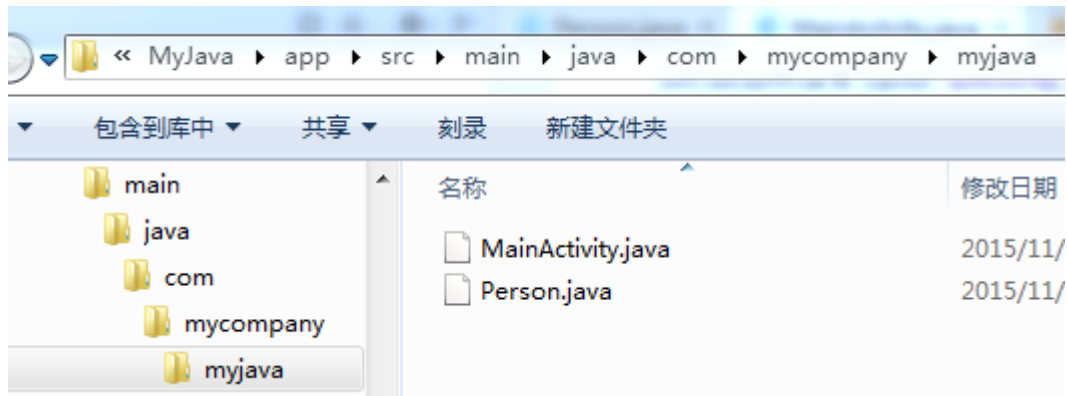
简而言之：**从逻辑上讲，包是一组相关类的集合；从物理上讲，同包即同目录。**

用`package`语句来实现包的定义。`package`语句必须作为Java源文件的第一条语句，指明该文件中定义的类所在的包。若缺省该语句，则指定为无名包，其语法格式为：

```
package pkg1[.pkg2[.pkg3...]]; // “[]”表示可选
```

Java 编译器把包对应于文件系统的目录管理，因此包也可以嵌套使用，即一个包中可以含有类的定义也可以含有子包，其嵌套层数没有限制。`package`语句中，用‘.’来指明包的层次；Java语言要求包声明的层次和实际保存类的字节码文件的目录结构存在对应关系，以便将来使用该类时能通过包名（也就是目录名）查找到所需要的类文件。简单地说就是包的层次结构需要和文件夹的层次对应。

**注意：每个源文件只有一个包的声明，而且包名应该全部小写。**



为了能够使用某一个包的成员，我们需要在Java 程序中明确导入该包。使用“import”语句可完成此功能。在java 源文件中import 语句应位于package 语句之后，所有类的定义之前，可以有0~多条，其语法格式为：  
import package1[.package2...](classname|\*);



## 第 4 章 Android Studio 界面布局

界面布局是研究 app 界面上各种控件的摆放，好的界面布局不仅让用户赏心悦目，而且能提高开发者工作效率，不再把精力纠结在控件与控件的位置关系上。

### 4.1 相对布局 RelativeLayout

相对布局的 XML 元素是<RelativeLayout>，相对布局的特点是可以让控件之间互相确定关系，这样可以保证在屏幕的局部范围内几个控件之间的关系不受外部影响。我们通过具体实例来学习它的用法。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_main" tools:context=".MainActivity">靠左最
顶端是一个<TextView>说明文本
    <TextView android:text="布局种类"
        android:id="@+id/tvButtons"
        android:layout_alignParentTop="true"----对齐到父 UI 的顶端
        android:layout_alignParentLeft="true"---对齐到父 UI 的左端
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="帧"
        android:id="@+id/btnFrame"
        android:onClick="btnFrameClick"
        android:layout_below="@+id/tvButtons"----位于文本框 tvButtons 之下
        android:layout_alignLeft="@+id/tvButtons"/>--与文本框 tvButtons 左边缘对齐
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="线性"
        android:id="@+id/btnLinear"
```

```

        android:onClick="btnLinearClick"
        android:layout_alignTop="@+id/btnFrame"
        android:layout_alignBottom="@+id/btnFrame"
        android:layout_toRightOf="@+id/btnFrame"/>----位于【线性】按钮右侧

```

```

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="表格"
    android:id="@+id/btnTable"
    android:onClick="btnTableClick"
    android:layout_alignTop="@+id/btnFrame"
    android:layout_alignBottom="@+id/btnFrame"
    android:layout_toRightOf="@+id/btnLinear"/>

```

```

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="网格"
    android:id="@+id/btnGrid"
    android:onClick="btnGridClick"
    android:layout_alignTop="@+id/btnFrame"
    android:layout_alignBottom="@+id/btnFrame"
    android:layout_toRightOf="@+id/btnTable"/>

```

</RelativeLayout>

我们把相对位置属性总结一下：

#### (1)属性值为 true 或 false

android:layout\_centerHorizontal 水平居中  
 android:layout\_centerVertical 垂直居中  
 android:layout\_centerInParent 相对于父元素完全居中  
 android:layout\_alignParentBottom 贴紧父元素的下边缘  
 android:layout\_alignParentLeft 贴紧父元素的左边缘  
 android:layout\_alignParentRight 贴紧父元素的右边缘  
 android:layout\_alignParentTop 贴紧父元素的上边缘  
 android:layout\_alignWithParentIfMissing 如果对应的兄弟元素找不到的话就以父元素做参照物

#### (2)属性值必须为 id 的引用名"@id/id-name"

android:layout\_below 在某元素的下方  
 android:layout\_above 在某元素的上方  
 android:layout\_toLeftOf 在某元素的左边  
 android:layout\_toRightOf 在某元素的右边  
 android:layout\_alignTop 本元素的上边缘和某元素的的上边缘对齐  
 android:layout\_alignLeft 本元素的左边缘和某元素的的左边缘对齐  
 android:layout\_alignBottom 本元素的下边缘和某元素的的下边缘对齐  
 android:layout\_alignRight 本元素的右边缘和某元素的的右边缘对齐

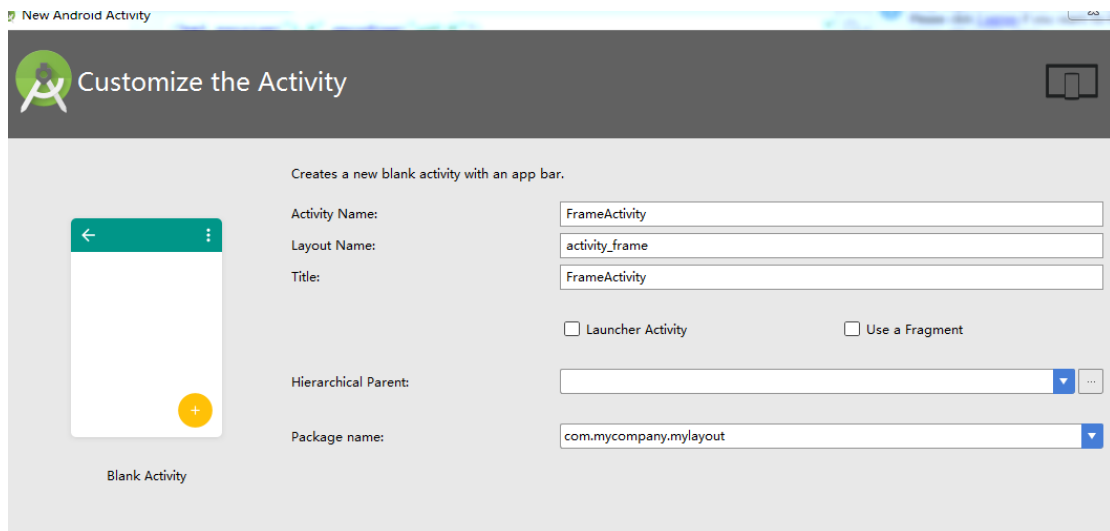
(3)属性值为具体的像素值，如 30dip, 40px

android:layout_marginBottom	离某元素底边缘的距离
android:layout_marginLeft	离某元素左边缘的距离
android:layout_marginRight	离某元素右边缘的距离
android:layout_marginTop	离某元素上边缘的距离

## 4.2 帧布局 FrameLayout

设计 FrameLayout 是为了显示单一项 view。通常，不建议使用 FrameLayout 显示多项内容。因为它们的布局很难调节，不用 layout\_gravity 属性的话，多项内容会重叠；使用 layout\_gravity 的话，能设置不同的位置。

我们用到 FrameLayout 的场合不多，这里就用 FrameLayout 的特点演示一个动画的例子。在 app\java\com.mycompany.mylayout 上单击右键，在弹出菜单上选择 new->Activity->Blank Activity，新建一个名为 FrameActivity。



把布局文件 content\_frame.xml 的布局改为 FrameLayout，然后给它定义名称：`android:id="@+id/myFrame"`

修改后的结果：

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:id="@+id/myFrame"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_frame">
```

```
tools:context="com.mycompany.mylayout.FrameActivity">
</FrameLayout>
```

把示例项目的 8 张图片 m1-m8 拷贝到 app\res\drawable 下，然后在 FrameActivity 编写代码。

1. 首先定义帧布局：FrameLayout **frame** = **null**;

然后在 FrameActivity 启动时候实例化这个 frame:

```
frame = (FrameLayout) findViewById(R.id. myFrame);
```

2. 编写定时器方法 **new** Timer().schedule(参数 1, 参数 2, 参数 3);

第一个参数是要执行的线程任务:

```
new TimerTask() {
    @Override
    public void run() {
        //发送一条信息来通知系统改变前景图片
        handler. sendEmptyMessage(0x123);
    }
}
```

这个线程执行的任务很简单，就是 **handler**. sendEmptyMessage(0x123); 稍后分析它。

第二个参数是这个线程启动后延迟多少微秒后才执行这个任务，为 0 表示立即执行；

第三个参数是间隔多少微秒执行一次，这里是 200 微秒。

完整的定时器方法如下:

```
new Timer().schedule(new TimerTask() {
    @Override
    public void run() {
        //发送一条信息来通知系统改变前景图片
        handler. sendEmptyMessage(0x123);
    }
}, 0, 200);
```

3. 编写一个方法 ChangeImage(int i)，它根据传入的值 i 设置上文定义的 **frame** 的前景图应该是哪个图片。getResources().getDrawable(R.drawable.*m1*) 是取到图片资源文件夹 drawable 下的 m1.png 图片，在程序里不用写图片扩展名，会自动识别图片种类。

```
void ChangeImage(int i)
{
    Drawable a = getResources().getDrawable(R.drawable. m1);
    Drawable b = getResources().getDrawable(R.drawable. m2);
    Drawable c = getResources().getDrawable(R.drawable. m3);
    Drawable d = getResources().getDrawable(R.drawable. m4);
    Drawable e = getResources().getDrawable(R.drawable. m5);
    Drawable f = getResources().getDrawable(R.drawable. m6);
    Drawable g = getResources().getDrawable(R.drawable. m7);
    Drawable h = getResources().getDrawable(R.drawable. m8);
    switch(i)
    {
        case 0:
            frame. setForeground(a);
    }
}
```

```

        break;
    case 1:
        frame.setForeground(b);
        break;
    case 2:
        frame.setForeground(c);
        break;
    case 3:
        frame.setForeground(d);
        break;
    case 4:
        frame.setForeground(e);
        break;
    case 5:
        frame.setForeground(f);
        break;
    case 6:
        frame.setForeground(g);
        break;
    case 7:
        frame.setForeground(h);
        break;
    }
}

```

4. 因为子线程无法直接更新 UI，就用上文提到的 **Handler** 方法可以在子线程中直接用来更新 UI。

所有 **FrameActivity** 代码如下：

```

public class FrameActivity extends AppCompatActivity {
    FrameLayout frame = null;
    //自定义一个用于定时更新 UI 界面的 handler 类对象
    Handler handler = new Handler()
    {
        int i = 0;
        @Override
        public void handleMessage(Message msg) {
            //判断信息是否为本应用发出的
            if(msg.what == 0x123)
            {
                ChangeImage(i);
                i++;
                if(i==8) i=0;
            }
            super.handleMessage(msg);
        }
    }
}

```

```
};
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_frame);  
    frame = (FrameLayout)findViewById(R.id.myFrame);  
    //定义一个定时器对象, 定时发送信息给 handler  
    new Timer().schedule(new TimerTask() {  
        @Override  
        public void run() {  
            //发送一条信息来通知系统改变前景图片  
            handler.sendEmptyMessage(0x123);  
        }  
    }, 0, 200);  
}
```

```
void ChangeImage(int i)
```

```
{  
    Drawable a = getResources().getDrawable(R.drawable.m1);  
    Drawable b = getResources().getDrawable(R.drawable.m2);  
    Drawable c = getResources().getDrawable(R.drawable.m3);  
    Drawable d = getResources().getDrawable(R.drawable.m4);  
    Drawable e = getResources().getDrawable(R.drawable.m5);  
    Drawable f = getResources().getDrawable(R.drawable.m6);  
    Drawable g = getResources().getDrawable(R.drawable.m7);  
    Drawable h = getResources().getDrawable(R.drawable.m8);  
    switch(i)  
    {  
        case 0:  
            frame.setForeground(a);  
            break;  
        case 1:  
            frame.setForeground(b);  
            break;  
        case 2:  
            frame.setForeground(c);  
            break;  
        case 3:  
            frame.setForeground(d);  
            break;  
        case 4:  
            frame.setForeground(e);  
            break;  
        case 5:
```

```

        frame.setForeground(f);
        break;
    case 6:
        frame.setForeground(g);
        break;
    case 7:
        frame.setForeground(h);
        break;
    }
}
}

```

## 4.3 线性布局 LinearLayout

线性布局是一种简单的布局，我们在前面的学习中已经使用过了。

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
</LinearLayout>

```

这是一个线性布局的例子，线性布局里的控件以在 XML 出现的顺序出现在屏幕上。线性布局里的控件是从左到右水平排列，还是从上至下竖直排列是由<LinearLayout>属性 android:orientation 决定的，当它值为"horizontal"时控件是在水平一直排列，不会转到下一行；当它值为"vertical"时控件至上而下排列。

## 4.4 表格布局 TableLayout

这种布局和 Office 里的制作表格类似，把布局看作一个表格，控件放置在每个单元格里。表格行的最大行高与该行最高控件匹配；表格最宽的列与该列最宽控件匹配。如下面例子按钮宽度和高度设为适应文本内容，每个控件在单元格里。

```

<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <TableRow>
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="按钮 1"
            android:id="@+id/btn1" />

        <Button

```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮 2"
        android:id="@+id/btn2" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="按钮 3"
            android:id="@+id/btn3" />
    </TableRow>
    <TableRow>
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="按钮 4"
            android:id="@+id/btn4" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="按钮 5"
            android:id="@+id/btn5" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="按钮 6"
            android:id="@+id/btn6" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="按钮 7"
            android:id="@+id/btn7" />
    </TableRow>
</TableLayout>

```

当把按钮 5 的宽和高修改为下面值，尺寸明显大于其它按钮，则改行的高和该列的高以按钮 5 为准。



```

<Button

android:layout_width="150dp"

android:layout_height="100dp"
    android:text="按钮 5"
    android:id="@+id/btn5" />

```



## 4.5 网格布局 GridLayout

网格布局虽然和表格布局类似，但比表格更灵活。网格布局子控件放置在<GridLayout>里，子控件坐标确定自己在网格位置，即所在的行和列，元素还可以跨行、跨列。如果控件需要 3 行 3 列的布局，控件坐标如下表所示。

0, 0	0, 1	0, 2
1, 0	1, 1	1, 2
2, 0	2, 1	2, 2

子控件的属性 android:layout\_row 和 android:layout\_column 分别表示行和列的坐标。跨行和跨列分别用属性 android:layout\_columnSpan 和 android:layout\_rowSpan 表示。

我们看一个例子，如果按钮按如图所示排列，按钮 4 跨第 1 列的第 2、3 行，按钮 5 跨第 2 行的 2、3 列。



```

<?xml version="1.0" encoding="utf-8"?>
<GridLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">
    <Button
        android:layout_row="0"
        android:layout_column="0"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮 1"
        android:id="@+id/btn1" />

    <Button
        android:layout_row="0"
        android:layout_column="1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮 2"
        android:id="@+id/btn2" />

    <Button

```

```
        android:layout_row="0"
        android:layout_column="2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮 3"
        android:id="@+id/btn3" />
<Button
    android:layout_row="1"
    android:layout_column="0"
    android:layout_rowSpan="2"
    android:layout_width="wrap_content"
    android:layout_height="100dp"
    android:text="按钮 4"
    android:id="@+id/btn4"/>

<Button
    android:layout_row="1"
    android:layout_column="1"
    android:layout_columnSpan="2"
    android:layout_width="175dp"
    android:layout_height="wrap_content"
    android:text="按钮 5"
    android:id="@+id/btn5" />

<Button
    android:layout_row="2"
    android:layout_column="1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="按钮 6"
    android:id="@+id/btn6"/>
<Button
    android:layout_row="2"
    android:layout_column="2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="按钮 7"
    android:id="@+id/btn7" />
</GridLayout>
```

# 第 5 章 Android Studio 视图工具箱

View 视图可以看做是 APP 的零件，包括编辑框、复选框、进度条、按钮等等。如图 5-1 左侧工具箱是 Android Studio 所有的 View，分类摆放。可见 Android Studio 的 View 很多，但我们不必一一学会它们，可以举一反三，学习它们的共同点，掌握它们的使用方法。

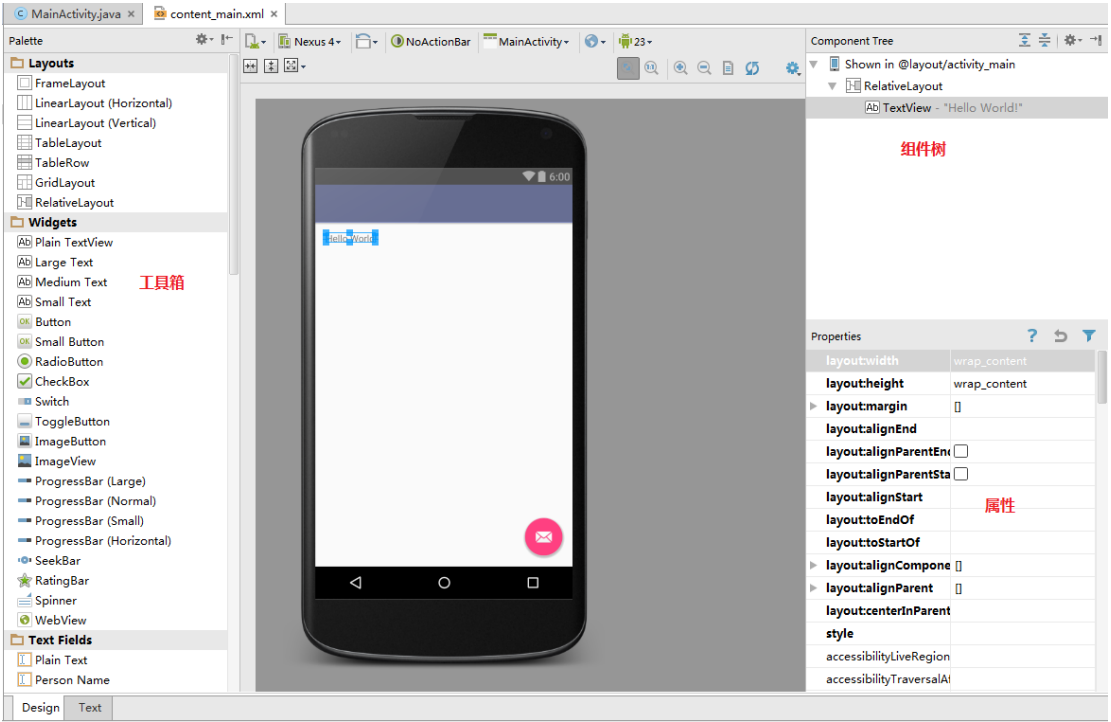


图 5-1

## 5.1 使用 View

### 1.不同的 View 有共同的属性

当选择一个 View，如图 5-1 屏幕上的 TextView，右下角便会出现这个 View 的属性，一个 View 的属性很多，但大部分的属性只需使用它的默认属性即可，无需修改，只是在一些特殊的应用中才会设置；而大部分的 View 只有一小部分属性需要设置，这些共同的属性是：

(1)id。设置 View 的 id，这样程序可以使用 id 操作这个 View，如果无需再程序里操作，可不必设置它。

(2)text。这个 View 显示的文本内容。

(3)layout\_width

(4)layout\_height

### 2.不同的 View 有类似地事件

## 5.2 常用 View 的使用

新建名为 MyView 的项目，删除 app\res\layout\content\_main.xml，直接在 activity\_main.xml 文件里修改布局：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
</LinearLayout>
```

当然你不这样做，在 `content_main.xml` 里修改也行，这里为了让大家更加灵活地学习 Android Studio。下面我们研究常用的 View。

### 5.2.1 文本框 TextView。

用于显示信息，不能输入，但可以在程序中动态修改显示信息。

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="16sp"-----字体大小
    android:textColor="#ffffff"
    android:padding="10dip"-----组件周围空隙大小
    android:background="#cc0000"-----背景颜色
    android:layout_gravity="center"
    android:text="Android Studio 工具箱"/>
```

把它放入上面的 `LinearLayout` 里，看看效果。`TextView` 在前面的章已经应用过多次，大家已经熟悉它了，可以多尝试它的其它属性。

### 5.2.2 编辑框 EditText

让用户输入文字，在程序中用 `getText` 获取用户输入的值。

3.按钮 `Button`。单击按钮会触发 `onClick` 事件，要把代码的方法和这个事件相关联。这里的 `onClick` 事件关联的方法是 `btnActionClick`，`btnActionClick` 需要在代码中实现。

```
<Button
    android:onClick="btnActionClick"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="按钮 1"
    android:id="@+id/btn1" />
```

### 5.2.3 单项选择 RadioGroup 和 RadioButton

多选一的选择模式。因为一组 `RadioButton` 的选择是单选，任何时候只能是一个被选中。只有把这一组 `RadioButton` 放在一个 `RadioGroup` 里，它们之间才是互斥的。`RadioGroup` 是 `RadioButton` 的容器，不仅让它们互斥选择，其 `android:orientation` 属性还能让它们排列是水平还是竖直。

```
<RadioGroup
    android:orientation="horizontal"-----内部的 RadioButton 是水平排列
    android:layout_width="wrap_content"
```

```

        android:layout_height="wrap_content">
        <RadioButton
            android:checked="true"-----表示它是否选中，默认是未选中
            android:onClick="rbClick"-----单击事件
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="看图片"-----显示文本
            android:id="@+id/rb1" />
        <RadioButton
            android:onClick="rbClick"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="时间日期"
            android:id="@+id/rb2" />
        <RadioButton
            android:onClick="rbClick"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="下拉列表"
            android:id="@+id/rb3" />
    </RadioGroup>

```

RadioButton 的 `android:onClick` 事件是同一个 `rbClick` 这个方法的实现：

```

public void rbClick(View view)
{
    int id=view.getId();
    switch (id)
    {
        case R.id. rb1:
            Toast.makeText(getApplicationContext(), "点击了看图片", Toast. LENGTH_SHORT).show();
            break;
        case R.id. rb2:
            Toast.makeText(getApplicationContext(), "点击了日期时间", Toast. LENGTH_SHORT).show();
            break;
        case R.id. rb3:
            Toast.makeText(getApplicationContext(), "点击了下拉列表", Toast. LENGTH_SHORT).show();
            break;
        default:break;
    }
}

```

参数 `view` 代表触发这个事件方法的 `RadioButton`，`int id=view.getId()`；取得这个 `RadioButton` 的 `id` 的整数值，用分支语句 `switch` 判断这个 `id` 与哪个 `RadioButton` 的 `id` 相

符，然后用 Toast 弹出短暂的窗口提示。

#### 5.2.4 多项选择 CheckBox

提供几个选项给用户勾选。因为 CheckBox 不是互斥选择，所以放在 LinearLayout 里排列好布局就可以了。

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <CheckBox
        android:text="语文"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <CheckBox
        android:text="数学"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <CheckBox
        android:text="英语"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <CheckBox
        android:text="体育"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <CheckBox
        android:text="生活"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

#### 5.2.5 图片视图 ImageView

它可以把 drawable 下的图片资源显示出来，首先把示例项目下为您准备好的 9 个图片拷贝到你的 drawable 文件夹下。新建名为活动 ImagesActivity，修改布局文件，这里是表格布局，有 3 行，每行放置 3 个 ImageView

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:gravity="top">

    <TableRow
```

```

        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <ImageView
            android:layout_width="100dp"-----宽度
            android:layout_height="100dp"-----高度
            android:src="@drawable/m1"/>-----指向 m1.png
        <ImageView
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:src="@drawable/m2"/>
        <ImageView
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:src="@drawable/m3"/>
    </TableRow>
    <TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <ImageView
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:src="@drawable/m4"/>
        <ImageView
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:src="@drawable/m5"/>
        <ImageView
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:src="@drawable/m6"/>
    </TableRow>
    <TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <ImageView
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:src="@drawable/m7"/>
        <ImageView
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:src="@drawable/m8"/>
        <ImageView
            android:layout_width="100dp"

```

```

        android:layout_height="100dp"
        android:src="@drawable/m9"/>
    </TableRow>
</TableLayout>

```

回到 MainActivity，把 switch 语句的看图片这个分支的代码由弹出窗口提示改为启动图片 Activity:

```

Intent intentImg=new Intent(this, ImagesActivity.class);
startActivity(intentImg);

```

### 5.2.6.日期选择器 (DatePicker)

新建一个 DateTimeActivity, 会自动生成布局文件 activity\_date\_time 和 content\_date\_time, 在 content\_date\_time 的 Design 模式, 从视图工具箱的 Date&Time 下拖拽 DatePicker 和 TimePicker 到 content\_date\_time:

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_date_time"
    tools:context="com.mycompany.myview.DateTimeActivity">

    <DatePicker
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/datePicker"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <TimePicker
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/timePicker"
        android:layout_below="@+id/datePicker"
        android:layout_centerHorizontal="true" />

</RelativeLayout>

```

打开代码文件 DateTimeActivity, 在 onCreate 里, 找到  
 Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH\_LONG)  
 .setAction("Action", null).show();



替换为:

```
DatePicker datePicker=(DatePicker)findViewById(R.id. datePicker);
int year=datePicker.getYear();
int month= datePicker.getMonth();
int day=datePicker.getDayOfMonth();
String datetime=String. valueOf(year)+"年"+String. valueOf(month)+"月"
+String. valueOf(day)+"日";
Toast. makeText(getApplicationContext(), datetime, Toast. LENGTH_SHORT).show();
```

首先取得日期 **View** 的对象, 分别取得年、月、日, 然后组合在字符串里, 用 **Toast** 弹出当前日期提示。取得时间的方法没有演示, 它需要在 **SDK23** 版本, 有兴趣的同学可以修改 **minSDK**, 创建相应的模拟器。

在 **MainActivity**, 修改 **switch**, 以便显示出 **DateTimeActivity**:

```
public void rbClick(View view)
{
    int id=view.getId();
    switch (id)
    {
        case R.id. rb1:
            Intent intentImg=new Intent(this, ImagesActivity. class);
            startActivity(intentImg);
            break;
        case R.id. rb2:
            Intent intentDate=new Intent(this, DateTimeActivity. class);
            startActivity(intentDate);
            break;
        case R.id. rb3:
            Toast. makeText(getApplicationContext(), "点击了下拉列表"
, Toast. LENGTH_SHORT).show();
            break;
        default:break;
    }
}
```

运行后, 单击“时间日期”单选按钮, 弹出的窗口, 点击右下角的邮件图标, 会显示日期。

### 5.2.7 下拉列表 Spinner

**Spinner** 提供一种下拉列表选择的输入方式, 这种方式在手机应用上很常见, 软件开发的一个“潜规则”是能让用户选择的时候不让用户填写, 因为下拉列表选择的数据规范, 而用户填写的内容可能千奇百怪, 需要在程序中处理, 否则会导致程序崩溃。

新建一个 **SpinnerActivity**, 切换布局文件 **content\_spinner.xml** 至 **Design** 模式, 从工具箱的 **Widgets** 栏拖入 **Spinner** 到屏幕上。打开 **SpinnerActivity**, 编写一个方法:

```
private void find_and_modify_view() {
String[] mProvinces={"北京市", "上海市", "天津市", "重庆市", "安徽省";
```

```
Spinner spinner=(Spinner)findViewById(R.id. spinner);
ArrayList arrayList=new ArrayList<String>();
for(int i=0;i<mProvinces.length;i++)
{
    arrayList.add(mProvinces[i]);
}
ArrayAdapter arrayAdapter=new
ArrayAdapter<String>(this, android.R.layout. simple_spinner_dropdown_item, arrayL
ist);
arrayAdapter.setDropDownViewResource(android.R.layout. simple_spinner_dropdown_
item);
spinner.setAdapter(arrayAdapter);
}
```

这个方法首先定义数组 `mProvinces`，定义了一些直辖市和省的名称，然后使用循环把这些数据赋值给 `ArrayList` 类型的 `arrayList` 变量。在实际项目中，这些数据是从数据库或网络取出的，这里的 `mProvinces` 充当数据源。然后把 `arrayList` 绑定到适配器 `arrayAdapter`，最后把 `spinner` 与适配器关联便可以显示数据。

## 5.3 小结

大家有兴趣可以继续拖动 `View` 到 `activity_main` 上看看效果，比如进度条(`ProgressBar`)、评分组件(`Rating`)等等。这里不再一一讲述所有的 `View` 了，在实际需要中学习、应用 `View` 更有意义。

## 第 6 章 多线程的实现

Activity 只是 Android 应用程序的一个屏幕或者用户界面，它是与用户交互的一个全屏应用或者浮动窗口。Android 应用程序由不同的 Activity 组成，这些 Activity 既与用户进行交互，也与其它 Activity 进行交互。例如，一个简单的接收器仅使用单个 Activity。如果完善计算器应用程序，在简单版本和科学版本之间切换，你将使用两个 Activity。

在计算机编程中，一个基本的概念就是同时对多个任务加以控制。许多程序设计问题都要求程序能够停下手头的工作，改为处理其他一些问题，再返回主进程。每个 Android 应用程序都运行在单独的进程中。为了运行应用程序，进程不断地被启动和终止。为了节省内存和资源，进程可能会被杀掉。而 Activity 则轮流地运行在应用程序进程的 UI 主线程中。

下面我们通过一个例子来学习 Android Studio 里多线程的应用和实现。

### 6.1 计时器范例

我们要做的示例是一个计时器，在界面上记录逝去的时间。时间一分一秒地过去，也就是要在界面上显示分分秒秒的时间。主线程就是正在运行的、显示在手机屏幕上的线程，另一个线程就是计时并显示。按以前学的知识，新建一个应用程序名称为 Lifecycle 的项目。

### 6.2 布局文件

为简化起见，删除布局文件 content\_main.xml，把 activity\_main.xml 改为：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_main" tools:context=".MainActivity">
    <TextView android:id="@+id/timer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
```

```

        android:orientation="horizontal" >
        <Button
            android:id="@+id/start_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/start_button"
            android:onClick="clickedStart" />

        <Button
            android:id="@+id/stop_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/stop_button"
            android:onClick="clickedStop" />
    </LinearLayout>
</RelativeLayout>

```

另外，res/values/strings.xml 需要加上：

```

<string name="start_button">开始</string>
<string name="stop_button">停止</string>

```

把代码文件 MainActivity 的 onCreate 方法只保留第 1、2 行。



图 6-1 布局效果

## 6.2 设置按钮状态

声明一个显示时间的文本和两个按钮：

```

protected TextView counter;
protected Button start;
protected Button stop;

```

在 onCreate 里把它们实例化：

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    counter = (TextView) findViewById(R.id.timer);
    start = (Button) findViewById(R.id.start_button);
    stop = (Button) findViewById(R.id.stop_button);
}

```

这里要让两个按钮互斥，也就是 **start\_button** 和 **stop\_button** 不能同时有效，所以建一个互斥变量：**protected boolean timerRunning**;  
定义两个按钮事件：

```

public void clickedStart(View view) {
    timerRunning = true;
    enableButtons();
}

public void clickedStop(View view) {
    timerRunning = false;
    enableButtons();
}

```

这两个按钮设置了 **timerRunning** 的值后，都调用 **enableButtons()** 决定禁用或启用按钮。

```

protected void enableButtons() {
    start.setEnabled(!timerRunning);
    stop.setEnabled(timerRunning);
}

```

## 6.3 更新计时器

(1) 新建两个 **long** 类型变量：

```

protected long startedAt;
protected long lastStopped;

```

(2) 在单击【开始】和【结束】按钮方法中，分别设置 **startedAt**、**lastStop** 的值为当前时间，单位是毫秒：

```

startedAt = System.currentTimeMillis();
lastStopped = System.currentTimeMillis();

```

这样，先单击【开始】按钮，后单击【结束】按钮，便可计算出两次点击的时间差。

(3) 创建方法 **setTimeDisplay()**，显示正在经历的时间，并把它设置到 **counter** 的文本中。

```

protected void setTimeDisplay() {
    String display;
    long timeNow;
    long diff;
    long seconds;
    long minutes;
    long hours;
}

```

```

    if (timerRunning) {
        timeNow = System.currentTimeMillis();
    } else {
        timeNow = lastStopped;
    }
    diff = timeNow - startedAt;
    //没有负值的时间
    if (diff < 0) {
        diff = 0;
    }
    seconds = diff / 1000;
    minutes = seconds / 60;
    hours = minutes / 60;
    seconds = seconds % 60;
    minutes = minutes % 60;

    display = String.format("%d", hours) + ":"
        + String.format("%02d", minutes) + ":"
        + String.format("%02d", seconds);
    counter.setText(display);
}

```

## 6.4 运行计时器

似乎使用 `setTimeDisplay()` 方法可以定期更新 UI 显示以便能够显示当前时间，但在 Android 系统中，这并不那么简单。Activity 的用户界面运行在单独的线程中，如果阻塞该线程太长时间，Android 操作系统认为应用程序以及冻结，会弹出一个应用程序没有反应 (Application Not Responding) 的对话框，下文我们简称 ANR。解决方案之一是创建另外一个线程，在该线程中完成所有的工作，这样就不会阻塞主 UI 线程，可以避免任何 ANR。遗憾的是，简单的使用标准的 Java 计时器或线程并不能解决问题。这是因为 Android SDK 不是线程安全的，任何用这种方式创建的线程都不能更新 UI 显示，只有 UI 线程可以更新 UI 显示。

那么解决方案是什么呢？答案是可以通过使用 `Runnable` 接口和 `Handler` 类来创建一个定时器。

`Runnable` 接口定义了一个名为 `run` 的方法，你需要实现该方法（它不接受参数并返回 `void`）。当新的线程启动时这个 `run` 方法就会被调用一次。

`Handler` 类可以对 `Runnable` 类中的 `run` 方法（和一些其他方法）的调用进行排队。可以使用这个类来实现每隔一段时间执行的定时器。

还有一些其他方法可以实现该功能。例如，使用 `AsyncTask` 或 `Services`。但是使用 `Runnable` 和 `Handler` 是最简单的。下面使用 `Runnable` 和 `Handler` 来创建定时器。

(1) 在 `MainActivity.java` 类的顶部，创建一个 `long` 类型的静态变量 `UPDATE_ERVERY`。为该变量赋值 200，这是更新屏幕 `counter` 的频率。如果设置为 1000，那么有可能不能准确地显示每一秒，定时器的显示可能会错过几秒。你可以多尝试几次，找到效果最好的值。

```
private static long UPDATE_ERVERY = 200;
```

(2) 创建一个实现接口 Runnable 的新类 UpdateTimer，定义一个 run 方法。

```
class UpdateTimer implements Runnable {  
    public void run() {  
  
        }  
    }  
}
```

(3) 为该类创建一个 handler 属性和 updateTimer 属性：

```
protected Handler handler;  
protected UpdateTimer updateTimer;
```

(4) 在【开始】按钮方法里的底部，创建这两个属性的实例，并调用 handler 的 postDelayed 方法。这将导致 UpdateTimer 的 run 方法在 200 毫秒后被调用。

```
handler = new Handler();  
updateTimer = new UpdateTimer();  
handler.postDelayed(updateTimer, UPDATE_EVERY);
```

(5) 在【停止】按钮方法的底部，通过调用 removeCallbacks 方法来停止所有等待调用 run 方法的事件，并设置 handler 为 null

```
handler.removeCallbacks(updateTimer);  
handler = null;
```

(6) 在 run 方法中，添加对定时器显示的调用，并于 200 毫秒后再次调用 run 方法（通过调用 postDelayed）。

```
if(handler!=null) {  
    handler.postDelayed(this, UPDATE_EVERY);  
}
```

(7) 运行该应用程序。按下【开始】按钮，定时器开始计数，当【停止】按钮按下后，定时器停止计数。

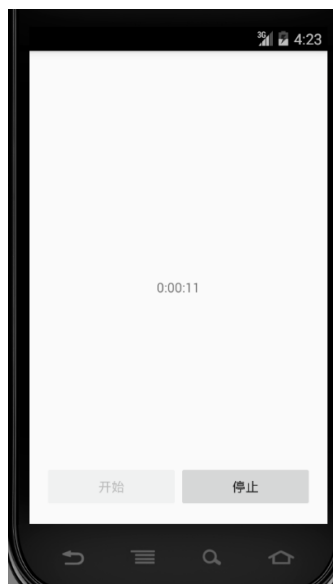


图 6-2 运行效果

## 第 7 章管理 Activity 生命周期

Activity 启动后就会进入一个生命周期 (lifecycle)。生命周期是一个术语，它指的是当用户（或者操作系统）与 Activity 交互时，Activity 经历的状态。Android 提供了特别的回调方法来让你针对 Activity 生命周期的变化做出反应。那我们为什么要研究生命周期呢？回答这个问题前，我们先把第六章做的范例拿出来再次运行。如果你把计时器通过 USB 接口安装在手机上运行该应用程序，启动计时器，等待一会儿，然后旋转屏幕，看看发生了什么？如果你不愿意麻烦，让它在一个模拟器上运行，然后同时按下 Ctrl+F12，可以旋转模拟器屏幕，你会发现计时器没有正常工作。要搞清楚这个原因、修改更加健壮的应用程序，就需要学习 Activity 的生命周期，并彻底搞清楚其原理。

### 7.1 Activity 生命周期理论知识

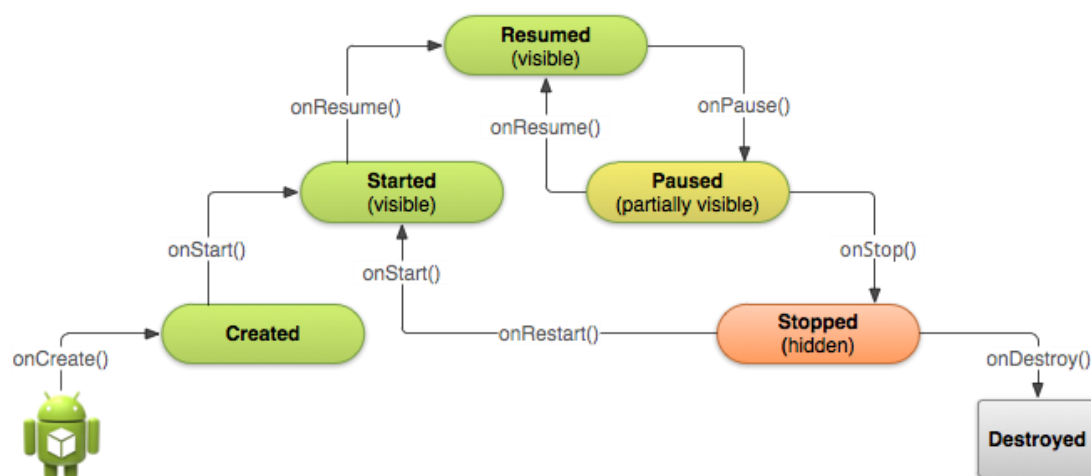


图 7-1 生命周期方法的调用

#### 7.1.1 Activity 生命周期的 5 种状态

Activity 生命周期有 4 中状态。

1. **Resumed**: 当 Activity 位于应用程序的前台时，它是运行中的 Activity，用户可见、可交互。在一段时间内只有一个 Activity 处于运行状态。

2. **Paused**: 如果 Activity 失去焦点，被另一个前台 activity 部分或全部覆盖，但仍然可见（比如一个更小的 Activity 出现在栈顶），那么 Activity 处于暂停状态，此时不接收用户输入，不执行任何代码。

3. **Stopped**: 如果 Activity 被另一个运行中的 Activity 彻底覆盖，完全隐藏在后台，用户不可见，处于停止状态。当 Activity 停止时，将失去所有状态，这样就需要在重启 Activity 时，重新创建用户界面的当前状态。

4. **Destroyed**: 当 Activity 被暂停或者停止后，系统为了回收内存可以结束它。在这之后用户可以重新启动 Activity。

5. 其它（Created 和 Started）: 是短暂的，系统快速从它们一个状态切换到下一个状态。



也就是说，系统调用 `onCreate()` 后，迅速调用 `onStart()`，接着 `onResume()`。下面我们看看系统对生命周期状态的调用。

## 7.1.2 生命周期状态的调用

app 不像其它传统程序（如 C 语言）那样启动 `main()` 方法，Android 系统激活指定的、与它生命周期对应阶段的方法。它在一个 `Activity` 实例里初始化代码，从启动到销毁 `Activity` 都有一系列的回调方法，从而完成整个应用程序的执行。

当应用程序在不同状态间迁移时，`android.app.Activity` 的生命周期方法（或者回调）会被系统调用。这些回调方法如下：

`onCreate(Bundle savedInstanceState)`：在 `Activity` 第一次被创建时调用。此时应初始化数据、创建初始视图或者回复 `Activity` 之前保存的冻结状态。`onCreate` 之后通常是调用 `onStart`。

`onStart()`：在 `Activity` 变为可见时被调用。这里最适合编写应用程序用户界面的相关代码，如处理与用户交互的事件。`onStart` 之后通常是调用 `onResume`。但如果 `Activity` 变为隐藏的，那么将调用 `onStop`

`onResume()`：在 `Activity` 变为前台运行并且可以与用户交互时被调用。它之后通常是调用 `onPause`。

`onPause()`：在其他 `Activity` 切换到前台时被调用。`onPause` 的实现必须很快，因为在该函数返回前，其他 `Activity` 不能运行。如果 `Activity` 重新回到前台，那么 `onPause` 之后调用的是 `onResume`。如果 `Activity` 变为不可见，那么 `onPause` 之后调用的是 `onStop`。

`onStop()`：在 `Activity` 变为不可见时被调用。无论是新 `Activity` 被启动（或者已经存在的 `Activity` 被恢复）还是当前 `Activity` 被销毁，当前 `Activity` 的 `onStop` 回调都会被调用。如果 `Activity` 重新回到前台，那么 `onStop` 之后将调用 `onRestart`

`onRestart()`：在 `Activity` 被重新启动并回到前台时被调用。`onRestart` 之后通常是调用 `onStart`

`onDestroy()`：在 `Activity` 被销毁前被调用，无论是因为 `Activity` 运行结束还是因为系统要回收当前 `Activity` 使用的内存，系统均会调用它。

图 7-1 中阐释了 `Activity` 经历的各种状态，以及回调方法的调用顺序：①用户离开，系统调用 `onStop()` 停止 `activity`；②用户返回停止的 `activity`，系统调用 `onRestart()`；③紧接着 `onStart()` 和④ `onResume()`。注意无论什么情况导致 `activity` 停止，系统永远在 `onStop()` 之前调用 `onPause()`

## 7.2 生命周期概念的应用

根据 `activity` 的具体情况，实际开发时候可能不需要实现所有的生命周期方法。然而，理解每个方法的实现，保证用户各种动作的需求是很重要的。熟悉 `activity` 生命周期所有方法可能保证你开发的 app 可以应对各种情况，包括：

- （1）正在使用 app 时，如果用户接听电话或切换其它 app 时不能崩溃；
- （2）当用户没有使用 app，就不要消耗手机宝贵系统资源；
- （3）用户暂时离开 app，返回时不要丢掉用户进程；
- （4）横竖切换屏幕时不能崩溃或丢失用户进程。

一个 `activity` 在图 1-1 所述的不同状态转换，其中三种状态是静态的，也就是能在三个状态的其中一个长时间存在。

## 7.2.1 LogCat 观察运行

我们打开第六章的示例项目，打开 MainActivity.java 代码文件，发现上面所述的生命周期方法，这里只有 onCreate()，其它的没有。其实，并不是所有的 Activity 都要写出所有的生命周期调用方法，要根据实际的需要写出。在 MainActivity 里任意位置单击右键，选择“Generate...”>“Override Methods...”>“Override Methods...”，如图 7-2、3、4 所示。

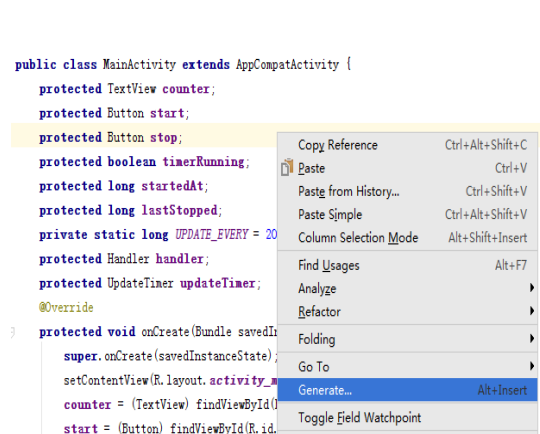


图 7-2

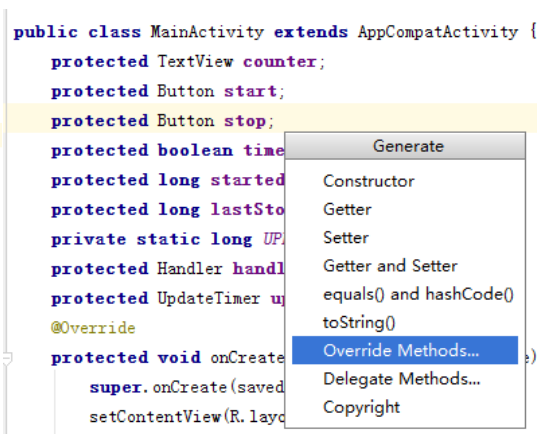


图 7-3

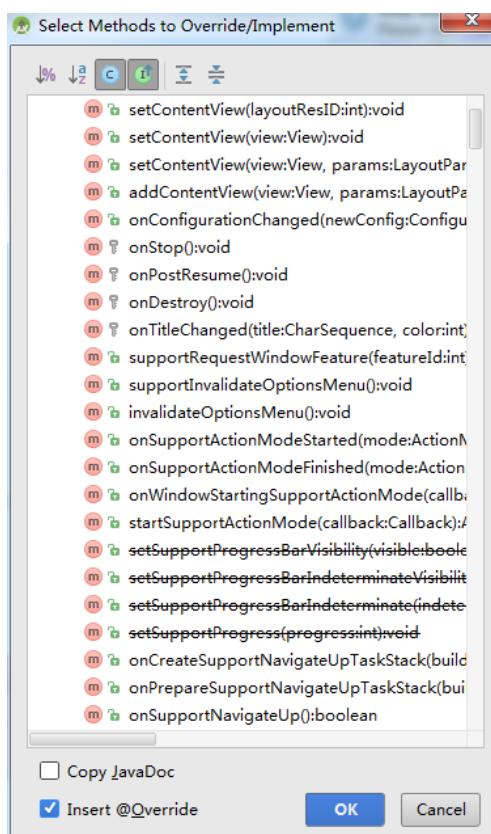


图 7-4

你会发现有许多方法，其中就有前面所提到的生命周期所有的方法。你可以从图 7-4 选择写入，也可以手动写入。最后 MainActivity 多出了如下生命周期方法：

```
@Override
protected void onStart() {
    super.onStart();
}
```

```
@Override
protected void onPause() {
    super.onPause();
}
```

```
@Override
protected void onResume() {
    super.onResume();
}
```

```
@Override
protected void onStop() {
    super.onStop();
}
```

```
@Override
protected void onDestroy() {
    super.onDestroy();
}
```

```
@Override
protected void onRestart() {
    super.onRestart();
}
```

由于每个方法里没有执行程序，而我们需要观察这些方法的运行，通过在 LogCat 视图中查看输出的 debug 级别的日志信息，这些信息显示了应用程序的状态变化。先加上变量：

```
private static String CLASS_NAME;
```

然后这些生命周期方法里加上日志信息，方便我们观察它们的执行情况：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d(CLASS_NAME, "onCreate");
    setContentView(R.layout.activity_main);
    counter = (TextView) findViewById(R.id.timer);
    start = (Button) findViewById(R.id.start_button);
    stop = (Button) findViewById(R.id.stop_button);
}

@Override
protected void onStart() {
```

```
        super.onStart();
        Log.d(CLASS_NAME, "onStart");
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.d(CLASS_NAME, "onPause");
    }

    @Override
    protected void onResume() {
        super.onResume();
        Log.d(CLASS_NAME, "onResume");
    }

    @Override
    protected void onStop() {
        super.onStop();
        Log.d(CLASS_NAME, "onStop");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d(CLASS_NAME, "onDestroy");
    }

    @Override
    protected void onRestart() {
        super.onRestart();
        Log.d(CLASS_NAME, "onRestart");
    }
}
```

运行 APP 后，观察 Android Studio 底部 LogCat，会出现我们加入的日志信息：

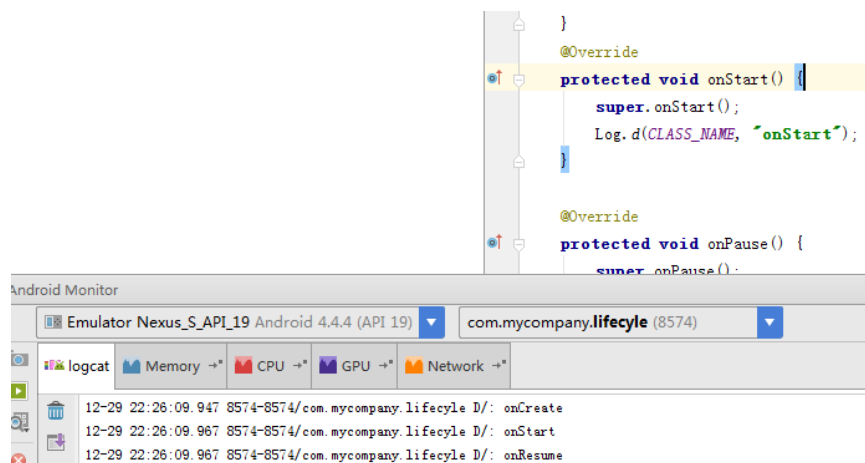


图 7-5 日志

我们看到执行顺序为 onCreate>onStart>onResume。然后同时按下 ctrl+Enter，导致了计时器失效，然后再次观察：

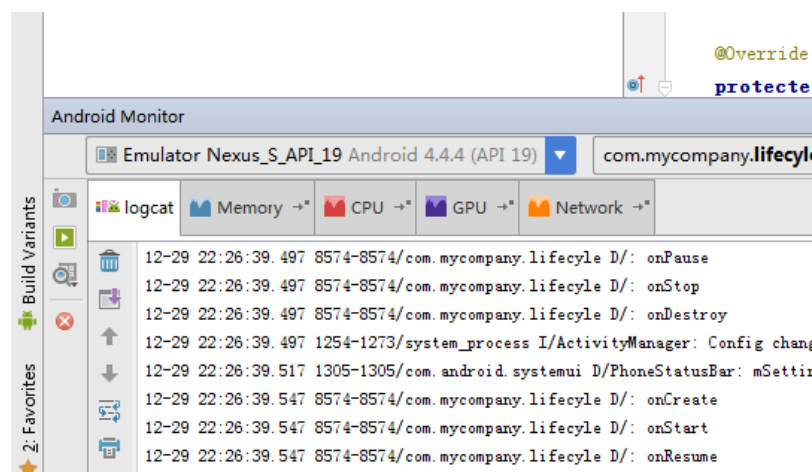


图 7-6

这种行为导致了 onPause>onStop>onDestroy>onCreate>onStart>onResume，Activity 会被重新创建，并丢失了所有状态，所以计时器失效。

## 7.2.2 修复 Activity 生命周期问题

如你所见，当应用程序不再允许，没有必要再更新计时器显示。当计时器 Activity 被重新创建时，需要刷新 UI 显示使其进入正确的状态。为解决这些问题，需要在正确的时间更新屏幕，还要把是否正在运行的标志 timerRunning 保存在手机里。

(1) 修改单击【开始】按钮事件

```
public void clickedStart(View view) {
    timerRunning = true;
    enableButtons();
    handler = new Handler();
    updateTimer = new UpdateTimer();
    handler.postDelayed(updateTimer, UPDATE_EVERY);
    saveState();
}
```

下面的方法是把 timerRunning 的值保存在手机里，其工作原理我们后面章节有讲解

```
public void saveState() {  
    SharedPreferences sharedPref = this.getSharedPreferences("SAVEDSTART",  
Context.MODE_PRIVATE);  
    SharedPreferences.Editor editor = sharedPref.edit();  
    editor.putBoolean("timerRunning", timerRunning);  
    editor.commit();  
}
```

(2) 修改 onStart

当 onStart 被调用且计时器仍在运行时，读取保存在手机上的 timerRunning 的值，再次调用 UpdateTimer 的 run 方法。

```
protected void onStart() {  
    super.onStart();  
    Log.d(CLASS_NAME, "onStart");  
    SharedPreferences sp = getSharedPreferences("SAVEDSTART",  
Context.MODE_PRIVATE);  
    timerRunning=sp.getBoolean("timerRunning", false);  
    if (timerRunning) {  
        handler = new Handler();  
        updateTimer = new UpdateTimer();  
        handler.postDelayed(updateTimer, UPDATE_EVERY);  
    }  
}
```

(3) 当 onStop 被调用时，不需要再更新时间显示，将如下代码添加到 onStop 方法中：

```
protected void onStop() {  
    super.onStop();  
    Log.d(CLASS_NAME, "onStop");  
    if (timerRunning) {  
        handler.removeCallbacks(updateTimer);  
        updateTimer = null;  
        handler = null;  
    }  
}
```

(4) 当 onResume 被调用时，需要刷新时间显示，添加如下代码：

```
protected void onResume() {  
    super.onResume();  
    Log.d(CLASS_NAME, "onResume");  
    enableButtons();  
    setTimeDisplay();  
}
```

最后，调试该程序，当计时器运行时，选择屏幕，应用程序按期待的方式运行了。

## 7.3 其它有关 Activity 的知识

### 7.3.1 销毁 activity

activity 生命周期最后一个调用方法是 `onDestroy()`，这个方法完成后 activity 实例被彻底从系统内存移除。大多 app 不需要你去实现这个方法，然而，如果 activity 包括后台线程或长时间运转的资源，不正确关闭可能会泄露内存，那需要在 `onDestroy()` 里杀死这些线程

```
@Override
public void onDestroy()
{
    super.onDestroy();//永远调用这个超类
    android.os.Debug.stopMethodTracing();//停止追踪 onCreate()期间启动的 activity
}
```

### 7.3.2 停止和重启一个 activity

恰当停止和重启 activity 是生命周期重要过程，它确保用户数据的保存，让你的 app 可靠健壮，不会丢失进程。停止和重启用于下面关键场景：

- (1) 用户从当前运行的 app 切换到另一个 app，那么当前运行在前台的 activity 被停止。如果用户从主屏幕重新打开或返回第一个 app 时，activity 需要重启；
- (2) 用户在 app 里从一个 activity 启动一个新的 activity，当第二个 activity 创建时第一个 activity 要被停止。接着用户按下返回按钮，第一个 activity 要重启；
- (3) 用户正在使用 app 时接听电话。

Activity 类提供两个生命周期方法：`onStop()` 和 `onRestart()`，允许你处理 activity 停止和重启时编程写入一些特别的操作。这不象暂停状态那样部分 UI 被阻碍，停止状态的 UI 不再可见，用户焦点只能在一个的 activity 上。注：因为系统保存停止的 activity 实例在系统内存里，你不必实现 `onStop()` 和 `onRestart()`。大多 activity 相对简单，可以用 `onPause()` 停止和重启。

### 7.3.3 停止 activity

当 activity 收到 `onStop()` 调用，就不再可见，释放所有不再需要的资源。一旦 activity 停止了，系统由于需要回收系统内存可能要销毁实例。极端情况下，系统只是简单地杀掉 app 进程，而不调用 activity 最终的 `onDestroy()`，因此用 `onStop()` 写处理程序释放可能内存泄露的资源很重要。

尽管 `onPause()` 方法在 `onStop()` 之前调用，应该让 `onStop()` 承担更多任务，类似 CPU 关闭前会密集地操作，如写信息到数据库等。如：这里的 `onStop()` 方法的实现保存（下面代码不是示例项目）

```
protected void onStop() {
    super.onStop(); //永远是第一个调用超类
    // 保存
```

```

 ContentValues values = new ContentValues();
 values.put(NotePad.Notes.COLUMN_NAME_NOTE, getCurrentNoteText());
 values.put(NotePad.Notes.COLUMN_NAME_TITLE, getCurrentNoteTitle())
 getContentResolver().update(
     mUri,      // 更新 note 的 URL
     values,    // 列名地图，应用的新值。
     null,      //
     null       //
 );
 }

```

当 **activity** 停止了，**activity** 对象驻留在内存，当 **activity** 继续时候调用。你不必重新初始化已创建的组件。系统也会保持跟踪布局 **view** 的当前状态，因此，如果用户输入文本到 **EditText**，内容会保留，不必保存恢复它。

### 7.3.4 启动/重启 activity

当 **activity** 收到 **onRestart()**调用，从停止状态返回前台。**onRestart()**方法，仅当 **activity** 从停止状态恢复时调用，因此你可以用它执行特别的恢复工作。一般不用 **onRestart** 恢复 **activity** 状态，因为 **onStop()**方法应该根本清除所有 **activity** 资源，重启时候你要重新初始化它们。当你的 **activity** 第一次创建（不存在 **activity** 实例）也需要初始化它们。所以，你应该经常使用 **onStart()**方法作为对应的 **onStop()**，因为系统在创建 **activity**、从停止窗体重启都要调用 **onStart()**。

例如，因为用户返回前可能要长时间离开你的 **app**，这时候一些条件如 **GPS** 是否正常等等需要在 **onStart()**里重新检查必需的系统功能是否可用。（下面代码不是示例项目）

```

@Override
protected void onStart() {
    super.onStart(); // 先调用超类
    // activity 重启或者第一次启动，检验 GPS 是否可用
    LocationManager locationManager =
        (LocationManager) getSystemService(Context.LOCATION_SERVICE);
    boolean gpsEnabled =
        locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
    if (!gpsEnabled) {
        // 这里创建一个对话框要求用户打开 GPS, 用一个 intent
    }
}

@Override
protected void onRestart() {
    super.onRestart();
}

```



```
// Activity 从停止状态重启
}
```

系统调用 `onDestroy()` 销毁 `activity`，因为你通常用 `onStop()` 释放大多资源，到收到 `onDestroy()` 调用时候，大多 `app` 不需要做很多了。这个方法是你最后清除资源的、避免内存泄露机会，因此你要确定销毁多余的线程和长时间运行的动作。

### 7.3.5 重建 activity

有一些场景是正常 `app` 操作导致 `activity` 销毁，如用户按下返回按钮，`activity` 调用 `finish()` 指示销毁自己。系统也可以销毁 `activity`，如果它当前停止、长时间不用、前台 `activity` 需要更多资源，系统必须关闭后台进程恢复内存。

由于用户按下返回、`activity` 自我结束而销毁 `activity`，系统认为 `activity` 实例已消失，因为这个行为指示 `activity` 不再需要。然而，如果由于系统的约束条件（不是正常 `app` 行为）销毁了 `activity`，尽管实际 `activity` 实例消失，系统记住它的存在，如果用户导航回来，系统创建一个 `activity` 新实例，使用一套保存描述 `activity` 状态的数据恢复先前状态的、已保存的数据被称为“实例状态，是存在 `Bundle` 对象的键-值对。

注意：用户每次旋转屏幕，`activity` 将销毁和重建。当屏幕改变了方向，系统销毁、重建前台 `activity`，因为屏幕规划改变了，`activity` 需要加载替换资源（如 `layout`）

默认情况系统用 `Bundle` 实例状态保存关于每个 `View` 对象的信息（如输入到 `EditText` 对象的值）因此，如果 `activity` 实例被销毁和重建，你无需写代码恢复先前状态的布局。然而，你的 `activity` 可能有更多状态信息需要保存，如最终用户进程的成员变量。为了系统恢复 `activity` 的 `view` 状态，每个 `view` 必需有 `android:id` 属性提供的唯一 ID。保存 `activity` 状态的附加数据，你必需覆写 `onSaveInstanceState()` 回调方法。系统调用这个方法，当用户正在离开 `activity`、传递它到 `Bundle` 对象，保存在 `activity` 意外销毁事件。如果系统必需以后重建 `activity`，它传递同样的 `Bundle` 对象到 `onRestoreInstanceState()` 和 `onCreate()` 方法

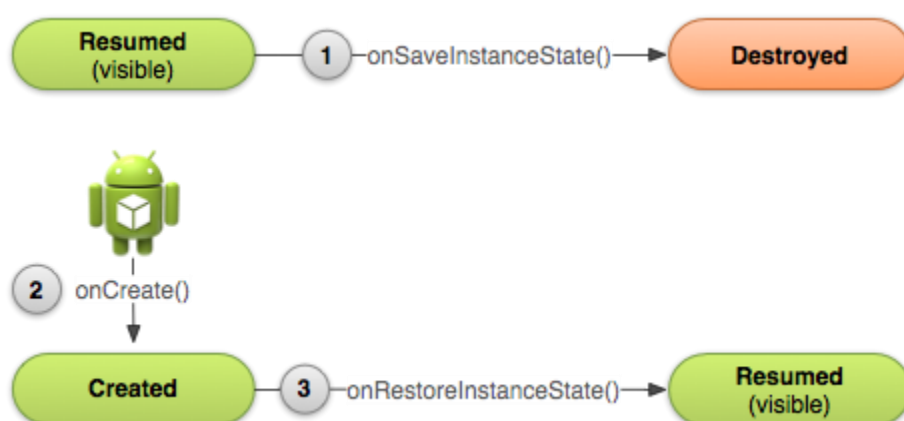


图 3-1

如图 3-1，①系统开始停止 `activity`，调用 `onSaveInstanceState()`，此时你编写程序保存数据，以免重建新的 `activity` 实例。如果 `activity` 销毁，同样的实例必需重建，系统把定义在①的数据传递到②`onCreate()`方法和③`onRestoreInstanceState()`方法

### 7.3.6 保存 active 状态

Active 正在停止时系统调用 `onSaveInstanceState()`，因此 active 能保存状态带有键-值对集合的状态信息。这个方法默认保存 active view 状态信息，如 `EditText` 里的文本或 `ListView` 的滚动条位置。为保存 active 附加状态信息，必须实现 `onSaveInstanceState()`、增加键-值对到 `Bundle` 对象。

```
static final String STATE_SCORE = "playerScore";
static final String STATE_LEVEL = "playerLevel";
...
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    // Save the user's current game state
    savedInstanceState.putInt(STATE_SCORE, mCurrentScore);
    savedInstanceState.putInt(STATE_LEVEL, mCurrentLevel);
    // 调用超类可以保存 view 状态
    super.onSaveInstanceState(savedInstanceState);
}
```

### 7.3.7 恢复 active 状态

当重建先前销毁的 active，你能从 `Bundle` 恢复保存的、系统传递的状态。`onCreate()`和 `onRestoreInstanceState()`回调方法都收到同样的、包含状态信息实例的 `Bundle`。因为系统正在创建为你 activity 一个新实例还是重建，`onCreate()`方法都会调用，你必须检查 `Bundle` 状态是否为空，读取它之前。如果空，系统正在创建 activity 新实例，而不是恢复。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // 检查是否正在重新创建先前销毁的实例
    if (savedInstanceState != null) {
        //从保存的状态恢复变量的值
        mCurrentScore = savedInstanceState.getInt(STATE_SCORE);
        mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);
    } else {
        //用默认值为新实例正确地初始化变量
    }
    ...
}
```

`onStart()`后，你可能选择实现 `onRestoreInstanceState()`而不是在 `onCreate()`期间恢复状态。系统调用 `onRestoreInstanceState()`仅在保存的状态去恢复，因此你不必检查 `Bundle` 为空

```
public void onRestoreInstanceState(Bundle savedInstanceState) {  
    //调用这个超类才能恢复保存的 view 的值  
    super.onRestoreInstanceState(savedInstanceState);  
    // 从保存的实例恢复状态  
    mCurrentScore = savedInstanceState.getInt (STATE_SCORE);  
    mCurrentLevel = savedInstanceState.getInt (STATE_LEVEL);  
}
```

## 第八章 简单数据和文件的存取

绝大多数 Android app 需要保存数据，上一章里涉及到了是保存 app 在 `onPause()` 期间的状态信息，以免用户的进程信息不会丢失。其实多数 app 需要保存用户配置，还有一些 app 必须管理大量文件。这里我们详细学习 Android App 键-值对和文件的存取，首先创建项目 DataSave。

### 8.1 键-值对数据的存取

用过微信的同学知道，第一次登录微信时需要验证登录信息，以后再打开微信时可以直接进入，那是因为把登录信息存储在手机上了。如果你是用 QQ 号码登录就是，你的 QQ 号为 123456，那么设置 QQ 为键，123456 为值，QQ-123456 就是键值对。像这种少量的键-值的集合需要存储，需要使用 `SharedPreferences` API。一个 `SharedPreferences` 对象指向一个包含键-值对的文件，并提供一个简单的读写方法。每个 `SharedPreferences` 由 framework 管理，既能私有又能共享。下面学习如何使用 `SharedPreferences` API 保存和恢复简单的值。

新建一个名为 `KeySaveActivity` 的 Activity 用于演示键值对数据的存取，如图 8-1 所示。



图 8-1

在“写入数据”栏随意写入数据，然后单击按钮【写入】，会把数据保存在手机里；当单击按钮【读取】，会把保存的数据显示在“读取数据”栏。下面我们看看代码的实现。

(1) `private final String KEY_NAME="MYKEY";`

这是声明常量 `KEY_NAME` 为键的名称，用来识别键，如果需要操作多个键值对，就要声明多个。

(2) 下面是【写入】按钮的方法，首先得到“写入数据”编辑框 `etWrite`，以便获取其数据：

获取 `SharedPreferences` 句柄：

```
SharedPreferences sharedPref=this.getSharedPreferences(KEY_NAME,Context.MODE_PRIVATE);  
getSharedPreferences()：用于多方共享文件，第一个参数用来标识名称，第二个参数定义为私有；
```

写入键值对数据，在 `SharedPreferences` 上调用 `edit()` 创建一个 `SharedPreferences.Editor`。用诸如 `putInt()`-取整数和 `putString()`-取字符串方法传递你想要写入的键和值，然后调用 `commit()` 保存。

警告：如果你创建一个带有 `MODE_WORLD_READABLE` 和 `MODE_WORLD_WRITEABLE` 参数的对象，那么任何其它识别出标识符的 app 可以访问数据。

```

public void btnWriteClick(View view)
{
    EditText etWrite=(EditText)findViewById(R.id. etWrite);
    SharedPreferences sharedPref=this.getSharedPreferences(KEY_NAME,
Context. MODE_PRIVATE);
    SharedPreferences.Editor editor=sharedPref.edit();
    editor.putString(KEY_NAME,etWrite.getText().toString());
    editor.commit();
}

```

(3) 【读取】按钮的方法：得到当前键名称为 **KEY\_NAME** 的 **SharedPreferences**，调用诸如 **getInt()**和 **getString()**的方法，取出你想要的键值，如果键不存在会有一个默认值本例是“none”。最后把它的值取出放置在“读取”编辑框里。

```

public void btnReadClick(View view)
{
    SharedPreferences preferences=getSharedPreferences(KEY_NAME, Context. MODE_PRIVATE);
    EditText etRead=(EditText)findViewById(R.id. etRead);
    etRead.setText(preferences.getString(KEY_NAME, "none")); //若没键值，就显示 none
}

```

## 8.2 文件的存取

Android 使用文件系统，类似其它基于磁盘文件系统的平台。我们学习 Android 文件系统如何应用 **File API** 读取、写入文件。一个文件对象适用于大量数据、从头至尾按顺序不可跳跃地读取，例如图片文件或其它网络传输数据。这节课学习在 **app** 执行一个基本的文件相关任务。这里假定你熟悉基本的 **Linux** 文件系统和 **java.io** 标准文件输入输出 **api**。

### 8.2.1 选择内部或外部存储器

所有 Android 设备有两个文件存储区域：“内部”和“外部”存储器。这些名字来源于早期的 Android，那时大多数设备提供了稳定的内置内存，还可以插入可移动的存储媒体，如 **SD 卡**（外部存储器）。一些设备把永久存储空间划分为“内部”和“外部”分区，因此即使没有可移动存储介质，还是有两个存储区域，外部存储器是否可移动的对于 **API** 而言是同样的操作。下面列出了每种存储器的特点

内部存储器	外部存储器
永远可用	不是永远可用，因为用户可以加载或卸载它
默认仅能由 <b>app</b> 访问	可以由外部可以读取文件
当用户卸载 <b>app</b> ，系统删除所有 <b>app</b> 文件	当用户卸载 <b>app</b> ，系统只能删除你使用 <b>getExternalFilesDir()</b> 保存的目录
内部存储器最好用户或 <b>app</b> 访问文件的安全	外部存储器最好用于 <b>app</b> 或用户不受限制地共享文件

## 8.2.2 获得外部存储许可

为了写数据到外部存储器, 必须在 manifest 文件要求 WRITE\_EXTERNAL\_STORAGE 允许, 在 manifests/AndroidManifest.xml 文件里加入:

```
<uses-permission  
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

如果 app 使用 WRITE\_EXTERNAL\_STORAGE 许可, 也意味着有读取外部存储器的权限。无需任何许可保存文件到内部存储器, 应用程序永远有权限读写文件到内部存储器的目录。

## 8.2.3 保存文件到内部存储器

当保存文件到内部存储器, 调用下列两个方法可以得到正确的文件目录:

**getFilesDir():** 返回一个文件代表一个内部存储器的目录

**getCacheDir():** 返回一个文件代表一个内部存储器目录作为 app 的临时缓存文件。

注意删除不再需要的文件, 在给定时间里根据内存分配合理大小, 如果系统开始运转下降, 会不予警告地删除缓存文件。下面我们看看项目示例文件, 保存文件的界面和 8-1 一样, 但代码是不同的。

### (1) 把文本内容写入文件

为了在这些目录里创建一个新文件, 可以使用 **File()**构造器指定内部存储路径的方法传递文件:

```
File file = new File(context.getFilesDir(), filename);
```

另一种办法是你还可以调用 **openFileOutput()**得到 **FileOutputStream** 写文件到你的内部目录。例如, 这里是写文本到文件的办法。

```
public void btnWriteClick(View view)  
{  
    EditText etWrite=(EditText)findViewById(R.id. etWrite); //获取写入文本控件实例  
    String string =etWrite.getText().toString(); //取得写入文本控件的内容  
    FileOutputStream outputStream;  
    try {  
        outputStream = openFileOutput(filename, Context. MODE_PRIVATE);  
        outputStream.write(string.getBytes());  
        outputStream.close();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

或者需要缓存一些文件应该用 **createTempFile()**代替。例如, 下面方法从 URL 提取文件, 创建 app 内部存储缓存目录的名字创建文件

```
public File getTempFile(Context context, String url) {  
    File file;  
    try {  
        String fileName = Uri.parse(url).getLastPathSegment();  
        file = File.createTempFile(fileName, null, context.getCacheDir());  
    }  
}
```

```

        catch (IOException e) {
            //创建文件错误的异常处理
        }
        return file;
    }
}

```

app 内部存储器目录是由 app 包名指定，位于一个特殊的 Android 文件系统的位置。技术上，另一个 app 能在你设置文件模式为可读的情况下读取你的内部文件。但是这个其它的 app 需要知道 app 包名和文件名才能做到。其它 app 不能浏览你的内部目录，除非文件设置为可读或写，否则不能读或写文件。因此只要你在内部存储器上文件用 `MODE_PRIVATE`，它们就不能为其它 app 访问。

## (2) 读取文件内容

首先要获得文件所在路径：`String path=this.getFilesDir()+"/"+filename;`  
`getFilesDir()`得到保存的路径，然后与写入文件所用的文件名组合即可得到完整路径。

```

public void btnReadClick(View view)
{
    String path=this.getFilesDir()+"/"+filename;//获取完整路径
    String content = ""; //文件内容字符串
    //打开文件
    File file = new File(path);
    try {
        InputStream instream = new FileInputStream(file);
        if (instream != null)
        {
            InputStreamReader inputreader = new InputStreamReader(instream);
            BufferedReader buffreader = new BufferedReader(inputreader);
            String line;
            //分行读取
            while (( line = buffreader.readLine()) != null) {
                content += line + "\n";
            }
            instream.close();
        }
    }
    catch (java.io.FileNotFoundException e)
    {
        Log.d("TestFile", "文件不存在");
    }
    catch (IOException e)
    {
        Log.d("TestFile", e.getMessage());
    }
    EditText etRead=(EditText)findViewById(R.id. etRead);
    etRead.setText(content);
}

```

## 8.2.4 保存文件在外部存储器

当用户把外部存储器挂载到计算机上或拔下了 SD 卡，外部存储器可能不可用，所以访问外存之前你应该永远检查是否有效。调用 `getExternalStorageState()` 查询外部存储器状态。如果返回的状态是 `MEDIA_MOUNTED`，那么就可以读写文件。下面例子检查存储器的可用性：

```
/* 检查外部存储器的读写是否有效 */
public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}

/* 检查外部存储器是否可读 */
public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

尽管外部存储器被用户和其它 app 修改，你可以存储两类文件：

- (1)公共文件：可以不受限制地与其它 app 和用户有效。当用户卸载了你的 app，这些文件还对用户有效。例如，你的 app 拍的照片、下载的其它文件。
- (2)私有文件：app 的运转所需文件等，当用户卸载 app 时候应该被删除。因为它们在外部分存储器上，尽管这些文件技术上被用户和其它 app 访问，它们实际上不为 app 以外的用户提供访问。当用户卸载 app，系统会删除所有在外部存储器私有目录下的文件，如 app 下载的额外资源、临时媒体文件。

如果你要保存文件到外部存储器，使用 `getExternalStoragePublicDirectory()` 方法得到一个代表外部存储器正确目录的文件。这个方法有个参数指定你要保存的文件类型，这样能和其它公共文件逻辑上组织好，如 `DIRECTORY_MUSIC` 或 `DIRECTORY_PICTURES`。

```
public File getAlbumStorageDir(String albumName) {
    //获取目录用于用户公共图片目录
    File file = new
    File(Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_PICTURES), albumName);
    if (!file.mkdirs()) {
        Log.e(LOG_TAG, "Directory not created");
    }
}
```



```
    }  
    return file;  
}
```

如果你要保存对你 **app** 而言是私有文件，你可以调用 `getExternalFilesDir()` 获得正确的目录，传递一个自定义的目录类型名称。每个用这种方式创建的目录被加入到封装所有你的 **app** 外部存储文件的父目录，在你卸载 **app** 时系统会删除掉它。  
例如，这个方法你可以用来创建一个目录为一个个人相册。

```
public File getAlbumStorageDir(Context context, String albumName) {  
    //获取目录用于 app 私有图片目录  
    File file = new File(context.getExternalFilesDir(  
        Environment.DIRECTORY_PICTURES), albumName);  
    if (!file.mkdirs()) {  
        Log.e(LOG_TAG, "Directory not created");  
    }  
    return file;  
}
```

如果没有预定义子目录的名字适合你的文件，代替方法是 `getExternalFilesDir()`，传递 `null`。这会为你的外部存储器上 **app** 私有目录返回其根目录。

记住 `getExternalFilesDir()` 是在一个当卸载 **app** 就被删除的目录里创建目录。如果你保存的文件需要在用户卸载你 **app** 后仍然有效，如 **app** 是一个照相机，用户需要保留照片。你应该取代用 `getExternalStoragePublicDirectory()`

不管用 `getExternalPublicDirectory()` 于共享，还是 `getExternalFilesDir()` 用于 **app** 的私有，重要的是你要用诸如 `DIRECTORY_PICTURE` 这样的 API 常量命名目录。这些目录名确保了系统正确地管理文件。例如，保存在 `DIRECTORY_RINGTONES` 的文件被分类后，系统媒体扫描器认为是彩铃而不是音乐。

## 8.2.5 查询空闲空间

如果你在保存前知道数据有多大，你就能查出是否有效空间是可用的，不会调用 `getFreeSpace()` 或 `getTotalSpace()` 引起 `IOException`。这些方法分别提供当前有效空间和所有存储空间。这些信息也是有用的，避免特定线程填满了存储卷。

然而，系统不能保证你能写入和 `getFreeSpace()` 指示的一样多的字节。如果返回数目比你保存的数据多出一些 MB，或者系统文件低于满格的 90%，那就可能安全地进行下去。否则你可能无法写入存储器。

注：保存文件前你如果不能检查可用空间数量。你可以用 `try` 写入文件，一旦出错可以捕获一个 `IOException`。你可能需要这样做，如果你不知道你到底需要多少空间。例如，你在保存文件前改变了文件编码，转 PNG 图片为 JPEG，你事先无法知道文件大小。

## 8.2.6 删除文件

你一定需要删除不再使用的文件。最直接的方法就是让打开的文件自己调用 `delete()`

```
myFile.delete();
```

如果文件保存在内部存储器，你要调用 `deleteFile()`使用 `Context` 定位、删除文件

```
myContext.deleteFile(filename);
```

注：当用户卸载 app，Android 系统删除下列文件：

所有保存在内部存储器上的文件；

所有保存在外部存储器上、用 `getExternalFilesDir()`保存的文件；

然而，你应该手动删除所有 `getCacheDir()`创建的缓存文件。

## 第9章 数据库操作

凡是做过信息系统开发的同学们都知道软件开发绕不过数据库，大家可能有使用过 SQL Server 或 Oracle 的经验。而 Android 操作 SQLite 数据库，SQLite 是一种流行的关系数据库管理系统，它具有以下特征：开源、符合标准、轻量级、单一层。SQLite 非常可靠，是许多消费类电子产品（如 MP3 播放器和智能手机）首先的数据库系统。使用 SQLite 可以为应用程序创建完全封装的关系数据库，创建的数据库在 Android 手机上，使用这些数据库可以存储和管理复杂的、结构化的应用程序数据，并持久化保存。

本章通过一个简单的人员信息管理的案例学习 Android 数据库操作，功能很简单：管理人员姓名和简介，在列表显示，可以增删改。学习本章内容假定你学习过数据库，因为在计算机课程里数据库本身就是一门课，这里不会详细介绍数据库理论，只学习 AndroidStudio 针对 SQLite 数据库的操作。为了将来方便做实际项目，我们用面向对象的方法讲解内容。首先按照前面的方法创建项目 PersonInfo

### 9.1 创建数据模型

应用程序需要保存并显示人员信息，所以讲数据建模为一个实体（类），这个类包括：人员姓名 name 和简介 info。因为我们重点学习数据库操作，所以这个类很简单，在实际项目中大家可以做更复杂的逻辑。这个类还有一个私有属性\_id。Android Studio 经常使用\_id 字段作为数据库表的主键，所以最好遵守这个约定。现在在 java/com.mycompany.personinfo 下定义一个类 Person：

```
public class Person {  
    private int _id;  
    public String name;  
    public String info;  
}
```

### 9.2 SQLiteOpenHelper 类详解

就像保存在设备内部存储器里的文件，Android 保存数据库于关联应用程序的私有磁盘空间。这个数据库是安全的，因为默认情况下这些区域不与其它应用程序有关联。Android 数据库操作有一个很有用的类—SQLiteOpenHelper，可用这个类获取数据库的引用进行存取操作，当需要创建或者更新数据库时，Android 会自动调用该类的方法，这个方法我们要事先写好。

要创建数据库，需要使用 SQL 语言，创建表的 SQL 语句形式：

```
create table <表名> (<字段名><字段类型>,<字段名><字段类型>,...);
```

经常可看到 SQL 用全大写字母书写，但是对 SQLite 来说不需要这样。实际上用小写字母的话还能够使 SQL 语句可读性强一点。与你以前学习的数据库字段类型不同，SQLite 仅支持如下数据类型：

int: 有符号数

real: 浮点数

text: 字符串

blob: 数据块

还有一些可以应用在数据库字段上的重要修饰符，第一个是 `not null`，它表明该字段必须包含一个值。第二个是 `auto-increment`，它可以用在一个主键字段上，当往表中添加新行后，相应字段的值会自动增加。

SQLite 没有 `boolean` 和 `date` 数据类型，但可用 `int` 类型的值 0 和 1 来表示 `boolean`，可以将数据保存为字符串或 `int` 来表示日期。

## 9.2.1 SQLiteOpenHelper 类

有一个名为 `SQLiteOpenHelper` 的类是用来创建、管理数据库的，通常我们不直接在 `SQLiteOpenHelper` 类里写逻辑，是创建一个类继承它，然后把操作数据库的逻辑写在这个子类里。下面，我们在包 `java/com.pisonsoft.ontheway.helpers` 下新建类 `SQLiteHelper`，继承 `SQLiteOpenHelper`，作为维护和管理数据库的基类，如下所示。

```
public class SQLiteHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "persondata.db";
    private static final int DATABASE_VERSION = 1;
    public SQLiteHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase database) {
        Person.create(database);
    }
    public void onConfigure(SQLiteDatabase database) {
        database.execSQL("pragma foreign_keys=ON;");
    }
    @Override
    public void onUpgrade(SQLiteDatabase database, int oldVersion, int newVersion)
    {
        Person.drop(database);
        onCreate(database);
    }
    public SQLiteDatabase open() { return getWritableDatabase(); }
    public void create() { open(); }
}
```

代码解释如下：

- (1) `private static final String DATABASE_NAME = "persondata.db";`  
`private static final int DATABASE_VERSION = 1;`

定义数据库文件名称和数据库版本的变量，系统发现当数据库版本 `DATABASE_VERSION` 的值变化了就会自动更新数据库，更新的方法我们后面给出；

- (2) `public SQLiteHelper(Context context) {`  
`super(context, DATABASE_NAME, null, DATABASE_VERSION);`  
`}`

构造函数，在构造函数内部调用父类的构造函数，并传递数据库名和版本号。

```
(3) public void onCreate(SQLiteDatabase database) {
    Person.create(database);
}
```

`onCreate` 方法接受 `SQLiteDatabase` 类型的单个参数，是重写父类的对应方法。它在类创建时会自动执行，如果数据库不存在的话创建数据库。

## 9.2.2 创建数据库

上面 (3) 的 `Person.create(database)`; 实际是执行 SQL 语句，它的代码写在 `Person` 类里：

```
public class Person {
    public int _id;
    public String name;
    public String info;
    public static void create(SQLiteDatabase database) {
        String createTable = "create table if not exists persons ("
            + "_id integer primary key autoincrement, "
            + "name text not null, "
            + "info text not null );";
        database.execSQL(createTable);
    }
}
```

## 9.2.3 数据库升级

要从数据库中移除表，可以执行 SQL 的 `drop` 命令

`drop table if exists <tablename>`

`if exists` 部分是可选的，但因为表可能不存在，所以包含该部分会更安全。当删除一个表，必须首先删除通过外键关联到它的表。

数据库中表的结构改变是软件维护中经常遇到的事情。当我们数据库结构更新的时候，版本号变量 `DATABASE_VERSION` 增加时，`Android` 就自动执行 `onUpgrade` 方法。从该方法我们看出，所谓的升级就是删除数据库的所有表及其数据，然后重写创建。

```
public void onUpgrade(SQLiteDatabase database, int oldVersion, int newVersion)
{
    Person.drop(database);
    onCreate(database);
}
```

我们完成 `drop` 方法，在 `Person` 类添加删除表的方法：

```
public static void drop(SQLiteDatabase database)
{
    String dropTable="drop table if exists persons";
    database.execSQL(dropTable);
}
```

## 9.3 创建数据库和表

Application 是 Android 框架的一个系统组件，当 Android 程序启动时系统会创建一个 Application 对象，用来存储系统的一些信息。

Android 系统自动会为每个程序运行时创建一个 Application 类的对象且只创建一个，所以 Application 可以说是单例（singleton）模式的一个类。

通常我们是不需要指定一个 Application 的，系统会自动帮我们创建，如果需要创建自己的 Application，那也很简单！创建一个类继承 Application 并在 AndroidManifest.xml 文件中的 application 标签中进行注册（只需要给 application 标签增加 name 属性，并添加自己的 Application 的名字即可）。

启动 Application 时，系统会创建一个 PID，即进程 ID，所有的 Activity 都会在此进程上运行。那么我们在 Application 创建的时候初始化全局变量，同一个应用的所有 Activity 都可以取到这些全局变量的值，换句话说，我们在某一个 Activity 中改变了这些全局变量的值，那么在同一个应用的其他 Activity 中值就会改变。

Application 对象的生命周期是整个程序中最长的，它的使用寿命就等于这个程序的生命周期。因为它是全局的单例的，所以在不同的 Activity、Service 中获得的对象都是同一个对象。所以可以通过 Application 来进行一些，如：数据传递、数据共享和数据缓存等操作。

(1) 创建名为 PersonApplication 的类，继承 Application，添加 SQLiteHelper 属性及获取方法：

```
public class PersonApplication extends Application {  
    private SQLiteHelper helper;  
    public SQLiteHelper getSQLiteHelper() {  
        if (helper == null) {  
            helper = new SQLiteHelper(this);  
        }  
        return helper;  
    }  
  
    @Override  
    public void onCreate() {  
        super.onCreate();  
        getSQLiteHelper().create();  
    }  
}
```

(2) 修改 AndroidManifest.xml 文件，将 android:name 特性设置为 .PersonApplication，这样应用程序将使用类 PersonApplication 作为其 application

```
<application  
    android:name=".PersonApplication"  
    android:allowBackup="true"  
    android:icon="@mipmap/ic_launcher"  
    android:label="@string/app_name"  
    android:supportRtl="true"  
    android:theme="@style/AppTheme">
```

当 APP 启动时系统创建 PersonApplication 对象，用来存储系统的全局信息。PersonApplication 生成时执行 onCreate，这生成 SQLiteHelper 对象，然后 getSQLiteHelper().create() 执行了 open() 方法，open() 方法执行了 getWritableDatabase()，因为 getWritableDatabase 内部自动调用了 mContext.openOrCreateDatabase，即打开或创建数据库方法，数据库不存在时会执行创建方法。

现在，当我们启动 APP 时，就会创建好了数据库和表。后面我们研究如何操作数据库。

## 9.4 数据库操作

### 9.4.1 创建菜单

为了操作方便，我们制作增加、删除、修改的菜单。

- (1) 打开 menu\menu\_main.xml 文件，添加

```
<item
    android:id="@+id/menu_add"
    android:orderInCategory="200"
    android:showAsAction="always|withText"
    android:title="@string/menu_add"/>
<item
    android:id="@+id/menu_del"
    android:orderInCategory="200"
    android:showAsAction="always|withText"
    android:title="@string/menu_del"/>
<item
    android:id="@+id/menu_edit"
    android:orderInCategory="200"
    android:showAsAction="always|withText"
    android:title="@string/menu_edit"/>
```

- (2) 打开 values\strings.xml，添加：

```
<string name="menu_add">新增</string>
<string name="menu_del">删除</string>
<string name="menu_edit">修改</string>
```

- (3) 打开 MainActivity，修改已有方法 onOptionsItemSelected，用户点击不同菜单执行相应的方法：

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    switch (id)
    {
        case R.id.menu_add:
            clickAdd();
            return true;
    }
}
```

```

        case R.id.menu_del:
            clickDel();
            return true;
        case R.id.menu_edit:
            clickEdit();
            return true;
    }
    return super.onOptionsItemSelected(item);
}

```

(4) 把上面所述的方法写下，以备后面使用

```

private void clickAdd()
{
}

private void clickDel()
{
}

private void clickEdit()
{
}

```

## 9.4.2 创建新增、修改的 Activity

创建一个名为 EditActivity 的空 Activity，用于提供给用户新增、修改数据的界面。建好后打开 app\res\layout\content\_edit.xml，修改为：

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.mycompany.personinfo.EditActivity"
    tools:showIn="@layout/activity_edit">
    <LinearLayout
        android:id="@+id/layoutEditor"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <TextView

```



```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="姓名: "
        android:id="@+id/textView2"/>
<EditText
    android:layout_width="100dp"
    android:layout_height="50dp"
    android:id="@+id/etName"/>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="介绍: " />
<EditText
    android:layout_width="200dp"
    android:layout_height="50dp"
    android:id="@+id/etInfo" />
</LinearLayout>
<LinearLayout
    android:layout_below="@+id/layoutEditor"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <Button
        android:layout_width="60dp"
        android:layout_height="wrap_content"
        android:text="增加"
        android:onClick="add"/>
    <Button
        android:layout_width="60dp"
        android:layout_height="wrap_content"
        android:text="修改"
        android:onClick="update"/>
</LinearLayout>
</RelativeLayout>

```



### 9.4.3 新增记录

在 `java\com.mycompany.personinfo\EditActivity` 里添加【增加】按钮的方法，该方法首先取得编辑框 `etName` 和 `etInfo` 用户输入的值，然后取得 `PersonApplication` 类创建的 `SQLiteHelper` 类的对象 `helper`，通过 `helper` 得到 `SQLiteDatabase` 类的对象 `database`，它执行插入数据的 SQL 语句。最后给出提示。

```
private EditText etName;
private EditText etInfo;
etName=(EditText)findViewById(R.id.etName);
etInfo=(EditText)findViewById(R.id.etInfo);

public void add(View view) {
    String name=etName.getText().toString();
    String info=etInfo.getText().toString();
    try {
        SQLiteHelper helper = ((PersonApplication)
getApplication()).getSQLiteHelper();
        SQLiteDatabase database = helper.getWritableDatabase();
        String sql = "insert into persons (name,info) values ('" + name +
",'" + info + "')";
        database.execSQL(sql);
        Toast.makeText(getApplicationContext(), "新增成功",
Toast.LENGTH_LONG).show();
    } catch (Exception exc)
    {
        Toast.makeText(getApplicationContext(), exc.getMessage(),
Toast.LENGTH_LONG).show();
    }
}
```

为了能够调用这个 Activity，返回 MainActivity，把 `clickAdd()` 补充完整。

```
private void clickAdd()
{
    Intent intent=new Intent(this,EditActivity.class);
    startActivity(intent);
}
```

### 9.4.4 显示数据

打开 `res\layout\content_main.xml`，添加一个 `ListView`：

```
<ListView
    android:id="@+id/listperson"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

打开 MainActivity, 声明一个 ArrayList 变量:

```
ArrayList<Map<String, String>> list = new ArrayList<Map<String, String>>();
```

添加方法 listPerson, 然后在 onCreate() 里调用它:

```
private void listPerson()
{
    SQLiteHelper helper = ((PersonApplication)
getApplication()).getSQLiteHelper();
    SQLiteDatabase database = helper.open();
    List<Person> persons= Person.getAll(database); // note not async
    helper.close();
    for (Person person : persons) {
        HashMap<String, String> map = new HashMap<String, String>();
        map.put("name", person.name);
        map.put("info", person.info);
        list.add(map);
    }
    SimpleAdapter adapter = new SimpleAdapter(this, list,
android.R.layout.simple_list_item_2,
        new String[]{"name", "info"}, new int[]{android.R.id.text1,
android.R.id.text2});
    ListView listView = (ListView) findViewById(R.id.listperson);
    listView.setAdapter(adapter);
}
```

该方法首先取得 SQLiteDatabase 对象, 传入 Person.getAll 方法, 取得 Person 对象集 persons, 把 persons 的值复制到 ArrayList 类型的 list, 把 list 与适配器 SimpleAdapter 关联, 最后设置 listView 的适配器为 SimpleAdapter。

把调用 Person 的 getAll 方法补充完整, 打开 Person, 添加:

```
static public List<Person> getAll(SQLiteDatabase database) {
    List<Person> list = new ArrayList<Person>();
    Cursor cursor = database.rawQuery("select * from persons", null);
    cursor.moveToFirst();
    while (!cursor.isAfterLast()) {
        Person person = cursorToPerson(cursor);
        list.add(person);
        cursor.moveToNext();
    }
    cursor.close();
    return list;
}

static private Person cursorToPerson(Cursor cursor) {
    Person person = new Person();
    person._id=cursor.getInt(cursor.getColumnIndex("_id"));
    person.name = cursor.getString(cursor.getColumnIndex("name"));
    person.info = cursor.getString(cursor.getColumnIndex("info"));
}
```

```
        return person;
    }
}
```

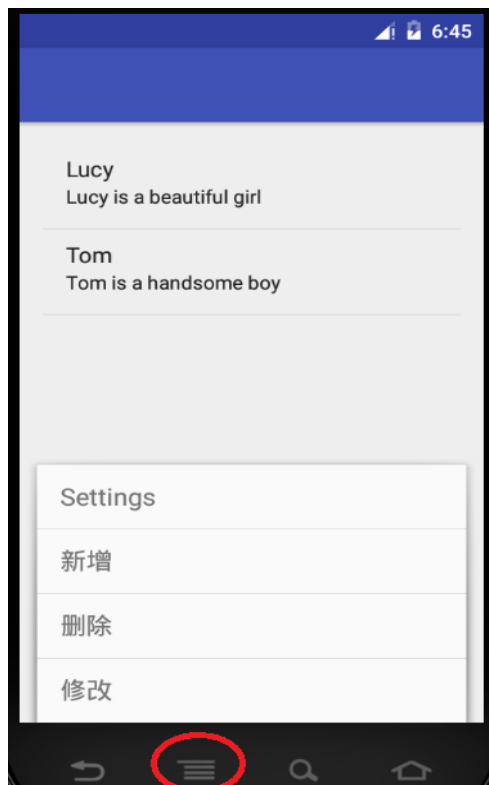
getAll 方法内创建一个 Person 类型的 List 和一个数据库游标，通过调用 rawQuery 来创建该游标的实例，并传递 SQL 语句从而在 persons 表中选择所有行。

移动到游标的第一个元素：cursor.moveToFirst();

继续移动游标，遍历所有元素，调用 Person person = cursorToPerson(cursor) 方法将它们一一转化为 Person 实例，添加到 list 中。最后关闭游标并返回已经填充数据的 list。

还有其它防止 SQL 注入的取数据方法：

```
ArrayList<Person> persons = new ArrayList<Person>();
String[] projection = {"_id","name","info"};
String selection=null;
String[] selectionArgs=null;
String sortOrder = "_id DESC";
Cursor c = db.query(
    "person",    //要查询的表名
    projection,  //返回的列
    selection,   //WHERE 子句里的列名
    selectionArgs,//WHERE 子句里的值
    null,        //行不分组
    null,        //没有行分组过滤
    sortOrder    //排序
);
```



## 9.4.5 修改数据

要修改数据，首先要知道用户选择了哪一条数据，所以要在 `listView` 加上用户点击的监听。还要把选择的数据暂时保存，当用户选择了【修改】菜单后，就显示这些数据提供给用户修改。

声明一个 `Person` 类，用于暂存用户选择的数据：

```
Person selectedPerson;
selectedPerson=new Person();

listPerson 方法添加 listView 的单击项的监听事件，并把所选项的值保存在 seletedPerson
private void listPerson()
{
    SQLiteHelper helper = ((PersonApplication)
getApplication()).getSQLiteHelper();
    SQLiteDatabase database = helper.open();
    List<Person> persons= Person.getAll(database); // note not async
    helper.close();
    for (Person person : persons) {
        HashMap<String, String> map = new HashMap<String, String>();
        map.put("name", person.name);
        map.put("info", person.info);
        list.add(map);
    }
    SimpleAdapter adapter = new SimpleAdapter(this, list,
android.R.layout.simple_list_item_2,
        new String[]{"name", "info"}, new int[]{android.R.id.text1,
android.R.id.text2});
    ListView listView = (ListView) findViewById(R.id.listperson);
    listView.setAdapter(adapter);
    listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long
arg3) {
            selectedPerson._id = arg2;
            selectedPerson.name = list.get(arg2).get("name");
            selectedPerson.info = list.get(arg2).get("info");
        }
    });
}
```

把单击按钮【修改】的方法补充完整，该方法把 `selectedPerson` 的信息作为参数附加到 `intent`，传递给 `EditActivity`：

```
private void clickEdit()
{
    Intent intent=new Intent(this,EditActivity.class);
```

```

Bundle bundle=new Bundle();
bundle.putInt("ID",selectedPerson._id);
bundle.putString("NAME",selectedPerson.name);
bundle.putString("INFO",selectedPerson.info);
intent.putExtras(bundle);
startActivity(intent);
}

```

打开 EditActivity, 在 onCreate 检查是否有 Intent 传入, 有的话取出带入的数据并保存在变量\_id, name 和 info 的值写入编辑框, 以使用户修改。

```

private int _id;
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_edit);
    etName=(EditText)findViewById(R.id.etName);
    etInfo=(EditText)findViewById(R.id.etInfo);
    Intent intent=getIntent();
    if(intent!=null)
    {
        Bundle bundle=intent.getExtras();
        _id=bundle.getInt("ID");
        etName.setText(bundle.getString("NAME")+String.valueOf(_id));
        etInfo.setText(bundle.getString("INFO"));
    }
}

```

【修改】按钮事件方法里, 我们使用了另一种操作数据的方法, 使用 ContentValues 保存要修改的数据。database.update 方法的参数分别为:

if(database.update("表名", 要修改的值, where 条件, where 的值)

```

public void update(View view)
{
    String name=etName.getText().toString();
    String info=etInfo.getText().toString();
    try {
        SQLiteHelper helper = ((PersonApplication)
getApplication()).getSQLiteHelper();
        SQLiteDatabase database = helper.getWritableDatabase();
        ContentValues values=new ContentValues();
        values.put("_id", _id);
        values.put("name", name);
        values.put("info", info);
        if(database.update("persons", values, "_id=?", new
String[] {String.valueOf(_id)}))>0)
            Toast.makeText(getApplicationContext(), "成功",
Toast.LENGTH_LONG).show();
    } catch (Exception exc)

```

```

    {
        Toast.makeText(getApplicationContext(), exc.getMessage(),
Toast.LENGTH_LONG).show();
    }
}

```



## 9.5 其它

如果有若干个 SQL 语句需要执行，可以使用事务，如下所示 `persons` 是若干个 `Person` 对象的列表，如果把它们都插入到数据库，把插入操作放在事务里，如果有一个插入操作失败，则事务回滚为出入前状态，防止有的插入成功有的没有。

```

public void add(List<Person> persons) {
    db.beginTransaction(); //开始事务
    try {
        for (Person person : persons) {
            db.execSQL("INSERT INTO person VALUES(null, ?, ?, ?)", new Object[]{person.name,
person.info});
        }
        db.setTransactionSuccessful(); //设置事务成功完成
    } finally {
        db.endTransaction(); //结束事务
    }
}

```

本章的删除方法没有给出，作为大家练习。另外，插入、修改完数据返回后应该更新显示列表，大家学习完 `Intent` 详解的知识后可以完善。

## 第 10 章 Intent 详解--与其它 App 交互

一个 Android 的 app 通常有几个 activity(活动)，每个 activity 显示一个用户交互界面，以允许用户执行特定任务（如浏览地图或照相）。为了让用户从一个 activity 导航到另一个 Activity，app 必须使用一个称为 Intent(意图)来实现，它定义了你的 app 意图做什么。当你用 startActivity()方法传递一个 Intent 到系统，系统使用 Intent 识别并启动合适的 app 组件，Intent 甚至允许一个 app 启动另一个独立的 app。一个 Intent 可以显示启动一个特定 Activity 实例，或隐式启动任何可调用的动作（如捕获照片）。

Intent 的用处很广，前面我们已经涉及到 Intent，现在我们系统学习它的用法。

### 10.1 调用 Activity 并返回结果

上一章我们学习的数据库操作就用到了调用另一个 Activity，当我们需要修改记录时，打开一个修改记录的 Activity；修改完成返回调用者 Activity 后应该通知调用者修改结果，以便更新数据列表显示。

这里我们不用数据库作为示例，实现的功能是用户在主页面单击【登录】菜单后，打开登录界面，用户输入用户名后返回，然后主页面显示用户名。在实际项目中，可以改为用户验证用户名和密码成功后，返回成功信息给主页面，主页面在加载用户信息。

#### 10.1.1 启动 Activity

(1)在 app\java\com.pisonsoft.myintent 新建一个 blank activity，名为 LoginActivity。创建后修改它的布局文件，打开 app\res\layout\content\_login.xml，修改为

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:paddingTop="60dp"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView android:id="@+id/tvLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:gravity="center_horizontal"
        android:text="请输入用户名和密码："
        android:textSize="25dp"
        android:textColor="#8C6931"/>
    <EditText android:id="@+id/txtName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/tvLogin"
        android:inputType="text"
        android:hint="请输入用户名"
```



```

        />
        <EditText android:id="@+id/txtPassword"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@+id/txtName"
            android:inputType="textPassword"
            android:hint="请输入密码" />

        <Button android:id="@+id/btnLogin"
            android:layout_width="90dp"
            android:layout_height="wrap_content"
            android:onClick="btnLoginClick"
            android:text="登录" />
    </LinearLayout>

```

(2) 打开 res\menu\menu\_main.xml，添加一条菜单：

```

<item
    android:id="@+id/action_login"
    android:orderInCategory="100"
    android:title="@string/action_login"
    app:showAsAction="never" />

```

(3) 打开 res/values/strings.xml，添加一条字符串资源

```

<string name="action_login">登录</string>

```

(4) 把 onOptionsItemSelected 里的

```

if (id == R.id.action_settings) {
    return true;
}

```

改为 switch 语句，并把调用登录的分支预留出来：

```

public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    switch (id)
    {
        case R.id.action_settings: return true;
        case R.id.action_login:

            break;
    }
    return super.onOptionsItemSelected(item);
}

```

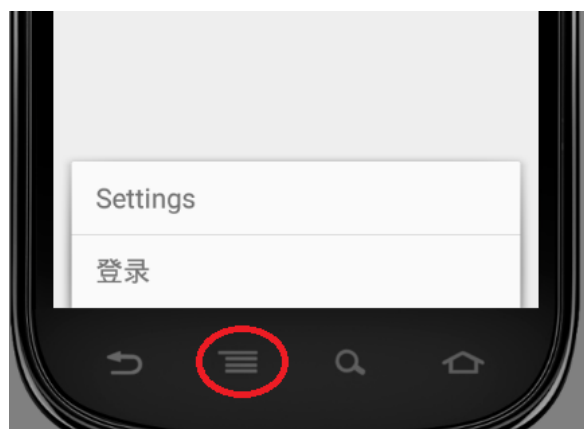


图 10-1

### 10.1.2 工作过程

为获得返回结果而启动 activity 的 Intent 对象没有什么特别之处，但是必须要传递一个附加整型参数到 `startActivityForResult()` 方法。这个整数参数是识别你请求的“请求码”。当你收到结果 Intent，回调也提供了同样的请求码，这样通过比较发送和返回的“请求码”是否一致而正确地识别出是哪个 Activity 调用后返回了，确定如何处理它。

(1)调用方在 `onOptionsItemSelected` 里发起调用，添加调用 `LoginActivity` 的代码，这里 `startActivityForResult` 的第二个参数就是请求码：

```
Intent intent=new Intent(this,LoginActivity.class);
startActivityForResult(intent, CC.LoginRequestCode);
```

这个请求码是一个常量，我们统一放在一个类里。在 `app\java\com.pisonsoft.myintent` 定义一个类 `CC.java`，用于存放常量，在 `CC.java` 定义一个请求码：

```
public final class CC {
    public static final int LoginRequestCode=1;
}
```

(2)在被调用者 `LoginActivity` 编写【登录】按钮代码。`setResult` 设置了返回的结果码，结果码是用户设定的整数，反映执行结果的情况，一般是 `Activity.RESULT_OK`、`Activity.RESULT_CANCELED` 就够了。

```
public void btnLoginClick(View view) {
    //取得用户输入的用户名和密码
    EditText etName = (EditText) findViewById(R.id.txtName);
    EditText etPwd = (EditText) findViewById(R.id.txtPassword);
    String name = etName.getText().toString().trim();
    String pwd = etPwd.getText().toString().trim();
    //返回调用者
    Intent intent = new Intent();
    intent.putExtra(CC.NAME, name);
    intent.putExtra(CC.PASSWORD, pwd);
    setResult(Activity.RESULT_OK, intent);//结果码 Activity.RESULT_OK
    //关闭自己
```

```

        finish();
    }

```

(3)在调用者 MainActivity 添加一个接收返回的方法，该方法首先判断请求码是什么，根据请求码识别出是哪个 Activity 返回的；然后根据返回码做进一步处理。

@Override

```

protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    if(requestCode==CC.LoginRequestCode) //登录返回
    {
        if(resultCode== Activity.RESULT_OK) {
            Bundle bundle = data.getExtras();
            String name = bundle.getString(CC.NAME, "");
            String pwd = bundle.getString(CC.PASSWORD, "");
            String str="用户名: "+name+"; 密码: "+pwd;
            TextView tv = (TextView) findViewById(R.id.tvLogin);
            tv.setText(str);
        }
    }
}

```

别忘了 MainActivity 的 TextView 的 id 为 tvLogin

## 10.2 调用其它 APP 服务

Intent 不仅能调用自己项目的 Activity，还能够调用其它的 app。例如，如果你的 app 里有一个公司地址，要把这个地址它在地图上显示出来，可以不必在自己的 app 建立一个显示地图的 activity 也能实现这个功能。实际上，你可以使用 Intent 创建一个查看地址请求，然后 Android 系统启动一个 app，显示地址在地图上。

### 10.2.1 调用动作

前面使用 intent 在 app 的 activity 之间导航通常使用显式的 Intent，定义需要启动的明确的类名。然而，调用别人的一个独立的 app 执行任务，例如“查看地图”，你必须使用一个隐式的 intent。我们看如何为特定动作创建一个隐式的 intent，如何用它启动一个 activity 执行另一个 app 的动作。这个特定的动作可称为 ActivityIntent。

**ACTION\_ALL\_APPS** 打开一个列出所有已安装应用程序的 Activity，通常此操作有启动器处理；

**ACTION\_ANSWER** 打开一个处理来电的 Activity，通常这个动作有本地电话拨号程序处理；

**ACTION\_BUG\_REPORT** 显示一个可以报告 bug 的 Activity，通常由本地 bug 报告机制处理；

**ACTION\_CALL** 打开一个电话拨号程序，并立即使用 Intent 的数据 URL 所提供的号码拨打一个电话。此动作值应用于代替本地拨号程序的 Activity。大多情况下使用 ACTION\_DIAL；

**ACTION\_CALL\_BUTTON** 当用户按下硬件的“拨打程序”时触发，通常会调用拨号 Activity；

**ACTION\_DELETE** 启动一个 Activity，允许删除 Intent 的数据 URL 中指定的数据；

**ACTION\_DIAL** 打开一个拨号程序，要拨打的号码有 Intent 的数据 URL 预先提供。默认情况下，这是有本地 Android 电话拨号程序进行处理。拨号程序可以规范化大部分号码样式，例

如 <tel:555-1234> 和 [tel:\(0551\)5551212](tel:(0551)5551212) 都是有效的号码;

**ACTION\_EDIT** 请求一个 Activity, 要求改 Activity 可以编辑 Intent 的数据 URL 中的数据;

**ACTION\_INSERT** 打开一个能够在 Intent 的数据 URL 指定的游标出插入新项的 Activity。当作为子 Activity 调用的时候, 它应该返回一个指向新插入项的 URL。

**ACTION\_PICK** 启动一个子 Activity, 它可以让你从 Intent 的数据 URL 指定的 Content Provider 中选择一个项。当关闭的时候, 它应该返回所选择的项的 URL。启动的 Activity 与选择的数据有关。例如, 传递 `content://contacts/people` 将会调用本地联系人列表;

**ACTION\_SEARCH** 用于启动特定的搜索 Activity, 如果没有在特定的 Activity 上触发它, 就会提示从所有支持搜索的应用程序中做出选择。可以使用 `SearchManager.QUERY` 键把搜索词作为一个 Intent 的 extra 中的字符串来提供;

**ACTION\_SEARCH\_LONG\_PRESS** 允许截获对硬件搜索键的长按操作, 通常由系统处理, 提供语言搜索的快捷方法;

**ACTION\_SENDDTO** 启动一个 Activity 来向 Intent 的数据 URL 所指定的联系人发送一条消息;

**ACTION\_SEND** 启动一个 Activity, 该 Activity 会发送 Intent 中指定的数据。接收人需要由解析的 Activity 来选择。使用 `setType` 可以设置传输数据的 MINE 类型。数据本身应该根据他的类型, 使用 `EXTRA_TEXT` 或者 `EXTRA_STREAM` 存储为 extra。对于 E-mail, 本地 Android 应用程序也可以使用 `EXTRA_EMAIL`、`EXTRA_CC`、`EXTRA_BCC` 和 `EXTRA_SUBJECT` 键来接收 extra。应该使用 `ACTION_SEND` 动作像远程接收人 (而不是设备上的另外一个应用程序) 发送数据。

**ACTION\_VIEW** 以最合理的方式查看 Intent 数据 URL 中提供的数据。不同的应用程序将会根据所提供的数据 URL 模式来处理视图请求。一般情况下, `http:地址` 将会打开浏览器, `tel:地址` 将会打开拨号程序以拨打该号码, `geo:地址` 会在 Google 地图应用程序中显示出来, 而联系人信息将会在联系人管理器中显示出来;

**ACTION\_WEB\_SEARCH** 打开一个浏览器, 根据 `SearchManager.QUERY` 键提供的查询执行 Web 搜索。

下面我们选几个动作练习。

(1) 在 `res\menu\menu_main.xml` 添加菜单项:

```
<item
    android:id="@+id/action_call"
    android:orderInCategory="11"
    android:title="@string/action_call"
    app:showAsAction="never" />
<item
    android:id="@+id/action_view"
    android:orderInCategory="12"
    android:title="@string/action_view"
    app:showAsAction="never" />
<item
    android:id="@+id/action_calendar"
    android:orderInCategory="13"
    android:title="@string/action_calendar"
    app:showAsAction="never" />
<item
    android:id="@+id/action_send"
    android:orderInCategory="14"
```

```
android:title="@string/action_send"
app:showAsAction="never" />
```

(2) 在 res\values\strings.xml 添加对应的字符串资源:

```
<string name="action_login">登录</string>
<string name="action_call">打电话</string>
<string name="action_view">浏览网页</string>
<string name="action_calendar">设置日历</string>
<string name="action_send">发送图片</string>
```

(3) 在 MainActivity 增加菜单处理方法

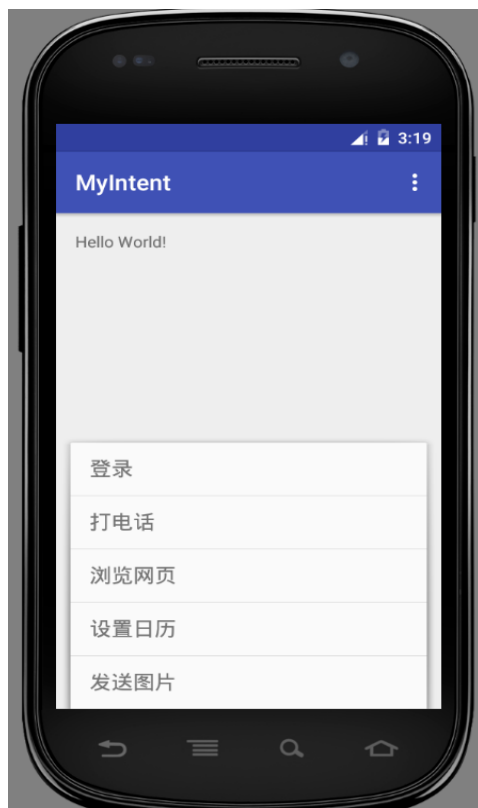
```
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    switch (id)
    {
        case R.id.action_login:
            Intent intent=new Intent(this, LoginActivity.class);
            startActivityForResult(intent, CC.LoginRequestCode);
            break;
        case R.id.action_call:
            Uri number = Uri.parse("tel:5551234");
            Intent callIntent = new Intent(Intent.ACTION_DIAL, number);
            startActivity(callIntent);
            break;
        case R.id.action_view:
            Uri webpage = Uri.parse("http://www.ifeng.com");
            Intent webIntent = new Intent(Intent.ACTION_VIEW, webpage);
            startActivity(webIntent);
            break;
        case R.id.action_calendar:
            Intent calendarIntent = new Intent(Intent.ACTION_INSERT,
CalendarContract.Events.CONTENT_URI);
            Calendar beginTime = Calendar.getInstance();
            beginTime.set(2015, 4, 28, 10, 0);
            Calendar endTime = Calendar.getInstance();
            endTime.set(2015, 4, 28, 12, 0);
            calendarIntent.putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME,
beginTime.getTimeInMillis());
            calendarIntent.putExtra(CalendarContract.EXTRA_EVENT_END_TIME,
endTime.getTimeInMillis());
            calendarIntent.putExtra(CalendarContract.Events.TITLE, "Ninja
class");
            calendarIntent.putExtra(CalendarContract.Events.EVENT_LOCATION,
"Secret dojo");
            startActivity(calendarIntent);
            return true;
    }
}
```

```

        case R.id. action_send:
            Intent picMessageIntent = new
Intent(android.content.Intent. ACTION_SEND);
            picMessageIntent.setType("image/jpeg");
            File downloadedPic = new File(
                Environment.getExternalStoragePublicDirectory(
                    Environment. DIRECTORY_DOWNLOADS), "q.jpeg");
            picMessageIntent.putExtra(Intent. EXTRA_STREAM,
Uri.fromFile(downloadedPic));
            startActivity(Intent.createChooser(picMessageIntent, "使用发送图
片:"));

    }
    return super.onOptionsItemSelected(item);
}

```



## 10.2.2 获取结果

有的 Activity 启动另一个 APP 得 Activity 不是单向的，还要获取所启动 Activity 的返回结果。要收到结果，需要调用 `startActivityForResult()`，而不是 `startActivity()`。例如，你的 app 可以启动一个照相 app 并收到捕获的照片作为结果。还可能启动了联系人 app 目的是为了给用户选择一个联系人，这样你会收到联系人信息作为返回的结果。

当然，能够有响应结果的 activity 必须事先设计为有返回值。当它返回时，把发送结果作为另一个 Intent 对象，这时你的 activity 在 `ActivityResult()` 调用时回收它。调用

`startActivityForResult()`，可以使用隐式或显示的 `intent`。当启动一个你自己的 `activity` 去回收结果，应该使用显示的 `intent` 确保你收到期望的结果

前面我们已经学习了调用返回结果的过程，现在小结一下。当用户完成调用的 `activity` 并返回时，系统自动调用你的 `activity` 的 `onActivityResult()` 方法，这个方法包括 3 个参数：

- (1) 你传递给 `startActivityForResult()` 的请求码
- (2) 所调用的 `activity` 指定的结果代码，`RESULT_OK` 表示操作成功，`RESULT_CANCELED` 表示用户按下后退按钮或其它什么原因导致失败
- (3) 带有返回数据的 `Intent`

这是如何处理“选择内容”`intent` 的处理的模板

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
    //检查 requestCode 是否与发送的请求码匹配
    if (requestCode == PICK_CONTACT_REQUEST) {
        // 检查结果代码是否等于 RESULT_OK，相等表示成功返回
        if (resultCode == RESULT_OK) {
            // 用户选择一个联系人
            // Intent 的数据 Uri 识别所选的联系人
            // 处理操作
        }
    }
}
```

这个例子介绍了 `Android` 联系人 `app` 返回的 `Intent` 结果提供一个 `Uri` 内容，确认用户选择的联系人。

为了成功地处理结果，必须理解 `Intent` 结构的格式是什么，当 `activity` 返回的结果是你自己的 `activity` 时是很容易的。`Android` 平台包括的 `app` 提供他们自己的 `API`，你能得到指定的数据结果。例如，联系人 `app` 永远返回确认所选联系人的 `URI` 内容为结果，照相机 `app` 返回的 `Bitmap` 放在附加数据里。

读取联系数据，上面代码展示了如何从联系人 `app` 得到结果，不必知道如何从结果中实际读取数据的细节，因为它需要更高级的管理内容的讨论。如果你对此好奇，这里有一些更多代码展示如何从所选联系人里读取电话号码

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
    //检查是不是要响应的请求
    if (requestCode == PICK_CONTACT_REQUEST) {
        // 确保成功的请求 Make sure the request was successful
        if (resultCode == RESULT_OK) {
            //获得指向被选联系人的 URI
            Uri contactUri = data.getData();
            String[] projection = {Phone.NUMBER};
```

```

        //在联系人上执行查询获得号码列
        //警告：query() 方法应该从独立线程上调用一方阻塞
        Cursor cursor = getContentResolver()
            .query(contactUri, projection, null, null, null);
        cursor.moveToFirst();

        //从号码列接受电话号码
        int column = cursor.getColumnIndex(Phone.NUMBER);
        String number = cursor.getString(column);
        //处理电话号码...
    }
}
}

```

## 10.3 让其它 app 启动你的 Activity

前面介绍了从你的 App 启动其它 Activity，但是如果你的 App 提供有为其它 App 有用的功能，那么你的 App 应该准备响应来自其它 App 的调用。例如，如果你构建了一个社交 App 共享信息或照片给用户的朋友们，那应该支持 ACTION\_SEND intent，这样用户能从其它 App 初始化一个“共享”动作，启动你的 App 执行动作。

为了允许其它 App 启动你的 Activity，需要在 manifest 文件被调用的<activity>里增加<intent-filter> 元素，称为 intent 过滤器。当你的 app 安装在设备上，系统识别你的 intent 过滤器，并把此信息增加到所有已安装的 app 支持 intent 的内部目录。当一个 app 调用 startActivity()或 startActivityForResult()时，系统就能找到哪个 activity 能响应 intent。

现在我们新建一个项目：SharedApp，用于目标 app 被启动。

### 10.3.1 增加一个 intent 过滤器

Intent 过滤器我们并不陌生，每次新建一个项目，主 Activity 会自动有一个 intent 过滤器，只是我们前面没有专门研究它。我们新建了 Sharedapp 后，打开 manifests\AndroidManifest.xml 文件：

```

<activity
    android:name=".MainActivity"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

```

为了恰当定义你的 Activity 能够处理的 intents，每个新增的 intent 过滤器应该尽可能地在动作类型和接受 Activity 数据特别



系统可以发送一个给定的 Intent 到 Activity，如果这个 Activity 有一个过滤器，能满足下列 Intent 对象标准：

**action:** 执行动作的字符串名。通常是 ACTION\_SEND 或 ACTION\_VIEW。在<action>元素指定 intent 过滤器，这个元素里指定的值必须是 Action 的字符串全名，而不是 API 常量。

**data:** 与 intent 关联的数据描述。它是在<data>元素里指定这个 Intent 过滤器。在这个元素里使用一个或多个属性，你可以指定 MIME 类型，URI 前缀，URI 方案，或这些的组合，其它可接受的数据类型

注意：如果你不需要声明特定的 Uri 数据（例如当你的 Activity 处理其它种类“附加”数据，而不是一个 URL），你应该仅指定 android:mimeType 属性，声明你的 Activity 能处理的数据类型，如 text/plain 或 image/jpeg

**category:** 提供一个额外方法特征化 Activity 处理的 intent，通常是与用户手势、启动位置关联。还有有一些系统支持很少使用的不同种类，默认所有隐式 intent 用 CATEGORY\_DEFAULT 定义。用<category>元素指定这个 intent 过滤器

在 intent 过滤器里，声明你的 Activity 接受的标准，嵌套在相应的<intent-filter>。如下有一个 Activity，带有处理 ACTION\_SEND 数据类型是文本或图片的 intent 的一个 intent 过滤器

```
<activity android:name="ShareActivity">
  <intent-filter>
    <action android:name="android.intent.action.SEND"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/plain"/>
    <data android:mimeType="image/*"/>
  </intent-filter>
</activity>
```

每个进入的 intent 指定一个 action 和一个数据类型，但每个<intent-filter>声明<action>，<category>，<data>的多个实例是没问题的。如果任何两个 Action 和数据对在它们行为中相互排斥，应该创建独立的 intent 过滤器指定哪个 Action 是可接受的匹配数据类型。

例如，假定的 Activity 在 ACTION\_SEND 和 ACTION\_SENDTO 的 intent 上都可处理文本和图片。这种情况下，你必须为两个 Action 定义两个独立的 intent 过滤器，因为一个 ACTION\_SENDTO intent 必须使用数据 Uri 指定接受者的地址，使用 send 或 sendtoURI 方案。

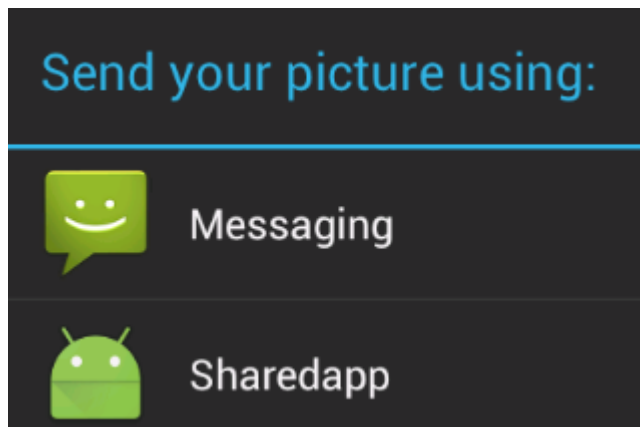
```
<activity android:name="ShareActivity">
  <!--发送文本过滤器；接受带有 sms URI schemes 的 SENDTO action -->
  <intent-filter>
    <action android:name="android.intent.action.SENDTO"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:scheme="sms" />
    <data android:scheme="smsto" />
  </intent-filter>
  <!--发送文本或图片过滤器；接受 SEND action 和文本或图片数据-->
  <intent-filter>
    <action android:name="android.intent.action.SEND"/>
    <category android:name="android.intent.category.DEFAULT"/>
```

```
<data android:mimeType="image/*"/>
<data android:mimeType="text/plain"/>
</intent-filter>
</activity>
```

注意：为了收到隐式 intent，你必须包括 CATEGORY\_DEFAULT 分类在 intent 过滤器。方法 startActivity()和 startActivityForResult()对待所有 intent 和它们声明的 CATEGORY\_DEFAULT 分类一样。如果你不把它声明在你的 intent 过滤器，没有隐式 intent 解决你的 Activity。

### 10.3.2 显示 app 选择器

当你启动一个 activity，传递 intent 到 startActivity()，有多个 app 响应这个 intent，用户能够选择默认打开的 app。当你运行一次 Sharedapp 后，关闭，再次运行前一个程序，单击[多选]按钮会出现选择器，这里因为 Sharedapp 里有一个 Send 动作。



### 10.3.3 在你的 Activity 处理 intent

为了决定什么样的 Action 进入 Activity，可以读取使用的 Intent 启动它。Activity 启动，调用 getIntent()接收启动 Activity 的 Intent。可以在 Activity 生命周期的任何时候做，但应该通常在回调的早期如 onCreate() 或 onStart()做。如：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    //获取启动这个 activity 的 intent
    Intent intent = getIntent();
    Uri data = intent.getData();

    // 根据 intent 类型确定该做什么
```

```
if (intent.getType().indexOf("image/") != -1) {  
    // 处理带图片数据的 intent ...  
} else if (intent.getType().equals("text/plain")) {  
    // 处理带文本的 intent ...  
}  
}
```

### 10.3.4 返回结果

如果要返回一个激活的你的 **Activity** 结果，简单地调用 `setResult()`指定结果代码和结果 **Intent**。当你的操作结束了，用户应该返回到原来的 **Activity**，调用 `finish()`关闭（或销毁）你的 **Activity**。如

```
// 创建 intent 递交一些种类的返回数据  
Intent result = new Intent("com.example.RESULT_ACTION",  
Uri.parse("content://result_uri");  
setResult(Activity.RESULT_OK, result);  
finish();
```

你必须永远在结果里指定结果代码。通常是 [RESULT\\_OK](#) 或 [RESULT\\_CANCELED](#)。需要的话你还可以提供附加数据到 **Intent**。结果默认设置为 [RESULT\\_CANCELED](#) 因此，如果用户在完成 **Action** 前、你设置结果前按下后退按钮，原始 **Activity** 收到“canceled”结果

如果你简单地需要返回一个整数，指示几种结果之一的选项，你可以设置结果代码为任何高于 0 的值。如果你使用返回代码递交一个整数，你不必需要包括 **Intent**，你可以调用 `setResult()`仅传递一个返回码。如：

```
setResult(RESULT_COLOR_RED);  
finish();
```

这个例子返回码是本地定义的整数（大于 0），这个好处是收到返回一个结果到你自己的 app 的 **Activity**，接收结果的 **Activity** 可以引用公共常量以决定返回码的值。