

android  
developers

# Android编程 andBook

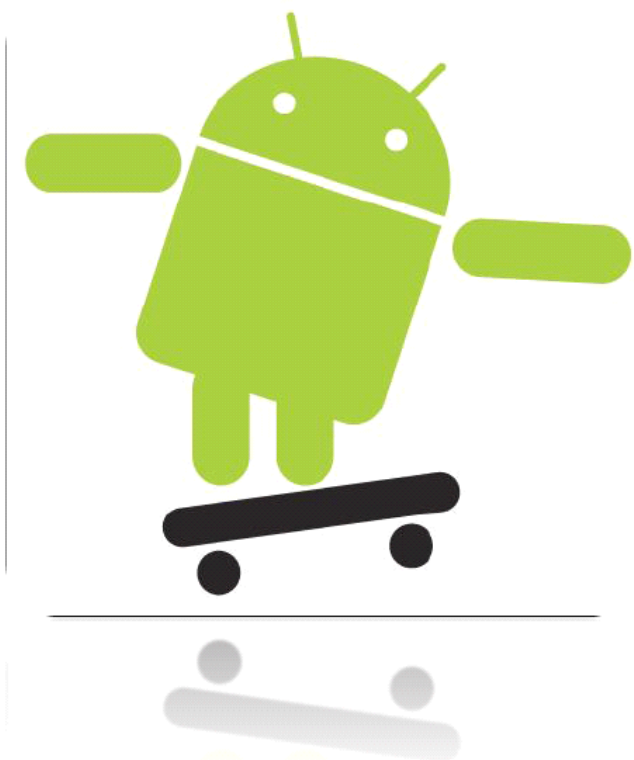


android



# *andbook!*

发布 .002



## Android 编程

附带教程来自 [anddev.org-Community](http://anddev.org-Community)

(本文仅供学习交流)原文版权属于

<http://www.anddev.org/>

Nicolas Gramlich 撰写

张劲锋 译

在 <http://andbook.anddev.org> 检查最新版本

由 [anddev.org](http://anddev.org) 提供

2





# 目录

序言/如何阅读本文 .....	5
介绍 .....	6
什么是 Android --一款 GPhone? .....	7
俯瞰 Android .....	8
开放性 .....	8
所有应用程序都是平等地创建 .....	9
打破应用程序局限 .....	9
快速 & 简单的应用程序开发 .....	9
第一周 .....	12
Dalvik.equals(Java)==false .....	13
与普通 Java 虚拟机的差异 .....	13
Android Code Challenge .....	14
创建 Android 应用程序 .....	15
剖析一个 Android 应用程序 .....	15
Activity .....	15
Intent Receiver .....	17
Service .....	17
Content Provider .....	18
Android 用户界面 .....	19
屏幕元素层次 .....	19
Android UI 元素与 Swing UI 元素比较 .....	22
AndroidManifest.xml 文件 .....	23
一般的 .....	24
<manifest> .....	25
<uses-permission> .....	25
<permission> .....	25
<instrumentation> .....	25
<application> .....	25







<meta-data>.....	27
<receiver>.....	27
<service>.....	27
<provider>.....	28

## 资源和魔幻的 R.java 文件.....29

资源.....	29
---------	----

资源列表.....	29
在代码中使用资源.....	30
参考资源.....	31

可选资源和局部化.....	32
---------------	----

魔幻的 R.java.....	33
-----------------	----

## Hello World --Andoird 方式..... 34

安装 Android SDK .....	35
----------------------	----

Android 开发工具(ADT).....	35
------------------------	----

安装完毕,我们来写点代码 .....	37
--------------------	----

创建一个新 Android 项目.....	37
-----------------------	----

运行你第一个应用程序.....	37
-----------------	----

UI--Java 方式.....	45
------------------	----

System.out.println(.....)?.....	46
---------------------------------	----

LogCat.....	46
-------------	----

## 使用 Intent..... 49

启动(子)Activity.....	49
--------------------	----

在 Java 代码中找到 XML 定义的 View.....	52
--------------------------------	----

为 View 处理点击.....	53
------------------	----

从子 Activity 返回数值.....	56
-----------------------	----

给(子)Activity 传递数据.....	46
------------------------	----

## 重要的 Layout 和 ViewGroup..... 60

RelativeLayout.....	60
---------------------	----





参考书目 .....	61
缺失章节 .....	62
社区 .....	62







## 序言/如何阅读本文

本文为以前已经使用 Java 编程并想开始为 Android 平台开发的开发者编写。我试着尽可能地多的"展开",在合适的每个地方放上示例代码。我也试着尽可能多的插入很多图片,因为它们可以活跃学习过程和吸引读者眼球。



但不庆幸的,不是每样都编上代码;你不得不学习一些 Android 平台基础原理来完全理解它。那就是在第一页中描述的。没有必要通读所有的那些描述页,但那样较好些。你可以把它当成一种参考。当在"展开"-过程期间你应该读也被说明了。因此你可以直接从 Hello World--Android 方式开始。

所有你在本文(所有工作空间)中看到的代码可以在以下站点获得:

<http://andbook.anddev.org/sources>

阅读愉快!!



由 [anddev.org](http://anddev.org) 提供



## 介绍

2007 年 11 月 12 日，开放手机联盟(Open Handset Alliance<sup>1)</sup>) 发布 Google Android SDK,一周前已经宣布。冲击力是难以置信的，几乎每个 IT/编程相关的报纸放出关于 Android SDK 发布的消息--在这一两天内，Google 组织就以超过 2000 消息而镇压群雄。



Android 平台的思想是并且一直是一个震惊和一门每天都吸引越来越多的开发者的课程。尤其是基于 Inten 和甚至可以替换主页面应用程序开源的架构，该架构正真给予整个平台很大数量的灵活性。

"Android ——无限想象"

<sup>1</sup> Nicolas Gramlich --anddev.org 网站--管理员





# 什么是 **Android**——一款 **GPhone**?

在 Google 发布 Android SDK 之前的几周和几月就有关于一个叫 GPhone 的许多传言。它被说成是一种由 Google 生产并通过向该设备用户展示敏感-背景广告方式提供免费通信的移动设备。



图 1 可能的 GPhone 三维图

但是到 2007 年 11 月 5 日 Andy Rubin<sup>2</sup> 宣布:

"Anroid 平台----更有深意和野心，而不是一个单纯的手机."

开放手机联盟成员 Google 为移动设备发布一套完整的软件：一个操作系统、中间件和核心移动应用。一周前发布的不是一个最终产品，而是一个很多地方都没实现的"预览 SDK"。那些主要的新闻站点抓取一些说 Android 平台满是漏洞并严重丢失文档的开发者的不适。但主要是说 Android 在该领域上比任何一个别的软件小。

2 Andy Rubin --Google 移动平台主管





## 俯瞰 Android

让我们开一下 OHA 在他的 Android 平台上强调的：

### 开放性

"Android 从底层被构建,使开发者可以创造引人注意的手机应用程序充分利用所有的付费手机,它被建构成正真的开放。例如，一个应用程序可以调用任何一个手机的核心功能，例如打电话、发送短信或是使用照相机，它允许开发者为用户创建丰富并身临其境的体验。"

这是真的，作为一个开发者你可以做每样事情，从发送短信只需 2 行代码，甚至替换你手机的主屏幕。你可以几周内容易地创建完整定制的操作系统，不再提供预置的 Google 应用程序给用户。

"Android 基于开源的 Linux 内核构建。此外，它利用一个定制的虚拟机，这个虚拟机专门设计来使在手机环境中的内存和硬件资源更完善。Android 会是开源的；它可以不受限制地扩展来融合新的技术（可移植的、可嵌入的，当这些技术出现时）。这个平台将会继续促进开发者社区协同工作来创建创新的手机应用程序。"

这里 Google 谈到了一个名叫 Dalvik 的虚拟机(Dalvik VM)，它是一个基于寄存器的虚拟机，由 Dan Bornstein 和其他 Google 工程师设计编写，是 Android 平台的一个重要部分。在"基于寄存器"处，我们发现了第一个和普通 Java 虚拟机(基于堆栈)的不同点。参看"`Dalvik.equals(Java)==false`"章节来获取更多关于这个讨论的细节。



## 所有应用程序都是平等地创建



## 打破应用程序局限

"Android 打破创建新的、创新的应用程序的局限。例如：开发者可以让个人手机上的数据和网站上的数据连接--如用户的联系人、日历、地理位置--提供了一个更有意义的用户体验。开发者可以创建一个应用程序，这个程序可以使用户看到他们朋友位置和改变后位置，当他们在附近时可以有和他们联系。"

## 快速&简单的应用程序开发

"Android 提供了宽范围的有用类库和工具，用这些可以创建丰富的应用程序，例如 Android 使开发者可以获得设备的地理位置，并允许设备通过同类对同类的方式的社交程序和其他设备通信。另外，Android 包括一套完整的工具，这些工具提供开发者高度的生产力和深度探究他们的应用程序。"





自从 Web2.0 改革，这让内容丰富的应用程序顷刻间不再是幻想。Android 带来了未知的开发速度。让我举个例子：

一天，我在 Android 文档中的"DrivingDirections"这个时髦词汇处困惑。



构思完成



图 2 Google 驾驶导航仪在 Android 上完成

上图中这个程序的开发过程花了一个半小时!! (包括简单的用户界面和所有你看到的图像)。你能否在其他手机平台上创建一个如此家喻户晓的经典程序？--不。



额外，用几行代码这个程序就能够使用当前流行的 GPS-定位来丰富。



Google 强调 Android 支持基础定位服务的能力。Android 中的 Google Map 如此整洁，就好像是专为 Android 开发的一样。通过添加 3 行 Android 默认应用的 Java 代码和 3 行 XML 代码，你就可以综合一个完整的放大和拖拽地图的能力。

在 Android 中别的易用的好的特性是动画和媒体回放。从 m5 版本，Android SDK 就包含了连续和背面的 GeoCoding 和 mp3 添入，支持：ogg-Vorbis，MIDI 和一些其他格式的回放。





## 第一周



不走运，开发者不得不处理一个不完整实现的预览 SDK(初始发布: "m3-rc20"), 里面甚至一些 SDK 关键部分被证实是不完整的。文档缺失导致 Google 开发组中出现一个 Android 开发者组织。迟早，你不得不重视来自 Google 的声明:

"如果它没有被证实，就不意味着他可用"

许多开发者不了解第一个发布的 SDK 是预览或是开发者预览这个事实，改变了的 API 要得等到。

其他让人厌的漏洞是模拟器声音突变，这个漏洞说是在四周后"m3-r37a"发布中解决,但还是在一些 m5 安装中出现了。



## Dalvik.equals(Java)==false

为什么叫"Dalvik"?--Dalvik 虚拟机由 Bornstein 命名,在 Eyjafjörður (冰岛)

渔村后面的 Dalvík, 他祖先曾住的地方。

正如你可能听说的一样, Dalvik 是 Android 虚拟机的名称。它是一个执行

Dalvik 可执行格式(\*.dex)文件的只直译程序虚拟机,这种 .dex 格式使高效存储和内存映射执行更完善。Dalvik 虚拟机基于寄存器,并能运行被 java 编译器编译过的 class, 这个 class 已经被包含的"dx"工具转换成 Dalvik 自己的格式了。这个虚拟机运行在 Linux2.6 内核上,虚拟机依靠它来获得底层服务(例如线程管理和低级内存管理)。Dalvik 虚拟机也被优化在多个实例中以很低的内存空间运行。各自的虚拟机保护它的应用程序不被其他破坏程序阻碍。

### 与普通 Java 虚拟机的差异

现在,几乎在任何一个桌面计算机上都可以找到的 Java 虚拟机是基于堆栈的虚拟机(VM)。另一方面, DalvikVM 是基于寄存器,由于手机处理器为基于寄存器执行而被优化。基于寄存器的 VM 在程序代价上也允许更快速的执行时间,哪些程序在编译后会更大。





## Android 代码挑战竞赛

Android Code Challenge(ADC)是 Google 为了鼓励社区为 Android 平台创建超酷的应用程序的一个策略，通过奖励提交的前 50 名最有前途的应用程序方式进行挑战竞赛。



图 3 Android 开发者挑战竞赛 Logo

当然，Android 开发者挑战竞赛--有总共 100,000,000 美金作为奖励金额，吸引了更多的软件开发者来创建一系列的正真有用的应用程序。另一边，在 SDK 发布的第一个重要月份中,许多声音表示，选择 Google 不是一个好的主意，因为它会导致少量的代码共享，由于许多人害怕和社区共享他们的主意。

有两个挑战竞赛计划：

- Android 开发者挑战竞赛 I：截止到 2008 八月 14 日。
- Android 开发者挑战竞赛 II：这部分将在 2008 第二季中期首款基于 Android 手机来临时启动。

Android 开发者挑战竞赛 I 中，到八月 14 日前提交的 50 个最有前途的作品将会被授予 25,000 美金奖励来做后期开发。这些被选中的若接下来符合主要认证鉴定并通过，有 10 个 257,000 金额的奖励和 10 个 100,000 金额的奖励。

提交到挑战竞赛的应用程序应该是创新和证明了 Android 平台功能的，像基础定位服务，媒体消费，游戏和社交关系网络等丰富手机体验的。



# 创建 Android 应用程序

本章节我们特别集中在创建 Android 应用程序。

## Android 应用程序剖析

对于一个 Android 应用程序有四种构成部分

- Activity
- Intent Receiver
- Service
- Content Provider



不是每一个的应用程序都需要所有这四个部分，但是你的应用程序将会用这些的组合来编写。

一旦你决定了你应用程序需要哪些组件，你应该在一个名叫 `AndroidManifest.xml` 文件中列出它们。这是一个 XML 文件，在里面声明你应用程序组件和它们的功能和需求是什么。我们将马上讨论，`AndroidManifest.xml` 负责什么作用。

## Activity

Activity 是 Android 四个构建模块中最普通的一个。一个 Activity 通常在你应用程序中是一个单独的屏幕。每一个 Activity 是被实现作为一个继承于 Activity 基类的单独的类。你的 Class 将要显示一个由 View 和响应事件组成的用户界面。多数应用程序由多个屏幕组成。例如：一个文本消息发送程序可能有一个显示要发送信息的联系人列表的屏幕，有一个给选中的联系人编辑短信的屏幕，并且另外的屏幕显示以前信息或是更改设置。每一个这些屏幕应该作为一个 Activity 被实现。移动到另外的屏幕上是通过启动一个新的 Activity 来完成。由于一些原因，一个 Activity 可能会向先前的 Activity 返回一个值——例如：一个让用户选择一个 Photo 的 Activity 将会返回选中的 photo 到 caller 应用程序上。





当一个新的屏幕开启，先前的屏幕被暂停并且压入历史堆栈。用户可以向后导航返回到在历史记录中先前打开的屏幕。当它们遗留不适当时，屏幕也可以选择从历史堆栈中删除。Android 为每一个从 Home 屏幕上运行的每一个应用程序保留历史堆栈。

## Intent 和 Intent Filter



Android 使用一个名叫 `Intent` 的特殊类来让应用程序从一个屏幕移动到另一个屏幕。`Intent` 描述了一个应用程序想要干什么。`Intent` 数据结构两个最重要的部分是：要响应的 `action` 和 `data`。典型的 `action` 值是 `MAIN`（应用程序的入口），`VIEW`、`PICK`、`EDIT`，等等。`Data` 表示成一个统一资源定界符（URI）。例如：要在浏览器中查看一个网站，你将创建一个带有 `VIEW` `action` 的 `Intent` 和设置了网站 `URI` 的 `data`。

```
new Intent(android.content.Intent.VIEW_ACTION,  
           ContentURI.create("http://anddev.org"));
```

有一个名叫 `IntentFilter` 的相关类。当一个 `Intent` 是一个有效地请求来做什么事，一个 `IntentFilter` 是：一个 `Activity`（或 `Intent receiver`，参见下面）能够处理什么样的 `Intent` 的描述。一个为个人设计的有能力显示联系人信息的 `Activity` 应该发布一个当申请表示一个人的数据时，可以说是知道如何处理 `action View` 的 `IntentFilter`。`Activity` 在 `AndroidManifest.xml` 文件中发布它们的 `IntentFilter`。

从一个屏幕到另一个屏幕的导航通过解决 `Intent` 来完成。要向前导航，一个 `Activity` 调用 `startActivity(myIntent)`。系统接下来查找所有应用程序的 `IntentFilter` 并选取一个 `IntentFilter` 最适合 `myIntent` 的 `Activity`。新的 `Activity` 被这个让它运行的 `Intent` 通知，解决 `Intent` 的过程发生在运行时候，即当 `startActivity()` 被调用时。`startActivity()` 提供了两个关键益处：

- `Activity` 可以从其他组件上简单地通过以一个 `Intent` 格式做请求来重用功能
- `Activity` 可以在任何时候被一个新的带有相等的 `IntentFilter` 的 `Activity` 替换



## Intent Receiver

当你想要在你的应用程序中编码来执行对外部事件的响应，你可以使用一个 **IntentReceiver**。例如：当电话响铃，或当数据网络可用时，或当午夜时。虽然 **IntentReceiver** 不显示一个 UI，但它们也许会显示一个通知来提醒用户是否有令人感兴趣的事件发生。**IntentReceiver** 也是在 **AndroidManifest.xml** 中被注册，但你也可以在代码中使用 "**Context.registerReceiver()**" 来注册它们。它的 **IntentReceiver** 被调用时，你的程序不是非得运行的；假如必要时，当一个 **IntentReceiver** 被触发时，系统将会调用你的应用程序。通过使用 "**Context.broadcastIntent()**"，应用程序也能发送它们自己的 "Intent Broadcast" 给别的应用程序。



## Service

一个 **service** 是长期存活并且运行时不带 UI 的编码。这个 **Service** 的好例子是 **Media Player** 从一个播放列表中播放歌曲。在一个 **Media Player** 应用程序中，可能有一个或更多的 **Activity**，这些 **Activity** 允许用户选择歌曲和开始播放歌曲。然而，音乐回放本身不应该被一个 **Activity** 处理，因为用户将期望在导航到一个新的屏幕后音乐保持播放。这种情况中，**Media Player Activity** 应该使用 "**Context.startService()**" 来启动一个服务在后台运行来保持音乐继续播放。接下来系统将保持音乐回放 **Service** 运行，直到这个 **Service** 被停止。（通过阅读 **Android** 应用程序生命周期，你可以学到更多关于先前给出的系统中的 **Service** 的信息。）记住，你可以使用 "**Context.bindService()**" 方法来连接一个 **Service**（如果还没有运行，并开启它）。当连接到一个 **Service**，你可以通过一个 **Service** 显露出来的 **Interface** 来和它通信。对于音乐 **Service**，这可能允许你暂停、倒回，等等。

## Content Provider

应用程序可以把他们的数据存储到文件中，一个 **SQLite** 数据库中，设置中，或是任何其他的有意义结构中。如果你想要你的应用程序的数据和其他应用程序共享，一个 **Content Provider** 是有用的。一个 **Content Provider** 是一个类，它实现了方法的一个标准设置来让其他应用程序存储和恢复被 **Content Provider** 处理的数据类型。






## Android 用户界面 (UI)

Android 中的用户界面可以用两种方式来创建，通过定义 XML-代码 或通过编写 Java-代码。在 XML 中定义 GUI 结构是非常好的，因为正如你从 MVC(Model-Viewer-Control)原理中所知的，UI 应该总是从项目逻辑中分离出来。另外，使一个项目从一个屏幕到另一个更加简单了。

在 XML 中定义一个 UI 是和创建一个普通的 HTML 文件非常相似，这个文件是你有的，换而言之，如此简单的文件：



```
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body> The content of the body element. </body>
</html>
```

和 Android 的 XML-布局一样。每一样结构好并且能够通过树形结构来表达。

```
<?xml version="1.0" encoding="utf-8"?>
  <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  >
    <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Hello World!"
    />
  </LinearLayout>
```



## 屏幕元素层次

Android 应用程序基本功能单元是 **Activity**——一个 `android.app.Activity` 类的物件 (Object)。一个 **Activity** 能够做许多事情，但是通过它自己它不会出现在屏幕上。为了让你的 **Activity** 出现在屏幕上和设计它的 UI，你使用 **View** 和 **ViewGroup**（——表现在 Android 平台上的基本用户界面单元）来工作。

### View



一个 **View** 是一个继承于 `android.view.View` 基类的物件。它是一种数据结构，它的属性为一个特殊的屏幕矩形域存储布局 (**layout**) 和 **Content**。一个 **View** 物件处理它描绘的屏幕区域的尺寸、布局、绘制、焦点改变、翻页和按键或手势。

**View** 类作为一个基类为所有的 **widget**——一系列完整实现的绘制交互屏幕元素的子类服务。**Widget** 处理它们自己的尺寸和绘制，因此，你可以使用它们来更快速地构建你的 UI。可获得的 **widget** 列表包括：（换言之）**TextView**、**EditText**、**Button**、**RadioButton**、**Checkbox**、**ScrollView**，……

### Viewgroup

一个 **Viewgroup** 是一个 `android.view.ViewGroup` 类物件。顾名思义，一个 **viewgroup** 是一个特殊类型的 **view** 物件，它的功能是用来包含和管理一系列下属 **view** 和其他 **viewgroup**，**Viewgroup** 让你给你的 UI 添加结构和构建起复杂的可以称为一个单独的实体屏幕元素。

**Viewgroup** 类作为一个基类为 **layout**——一系列完整实现的提供普通类型屏幕元素的子类服务。**Layout** 给你一种方式来为一系列 **view** 创建一个结构。





## 树形结构 UI

在 Android 平台上，如下图所示，你使用一个树形 View 和 Viewgroup 节点定义一个 Activity 的 UI。这个树形可以简单或复杂，如你所需来创建它，并且你可以使用 Android 一系列的预定义 Widget 和 Layout 或是你自己创建的 Custom View 类型来构建它。

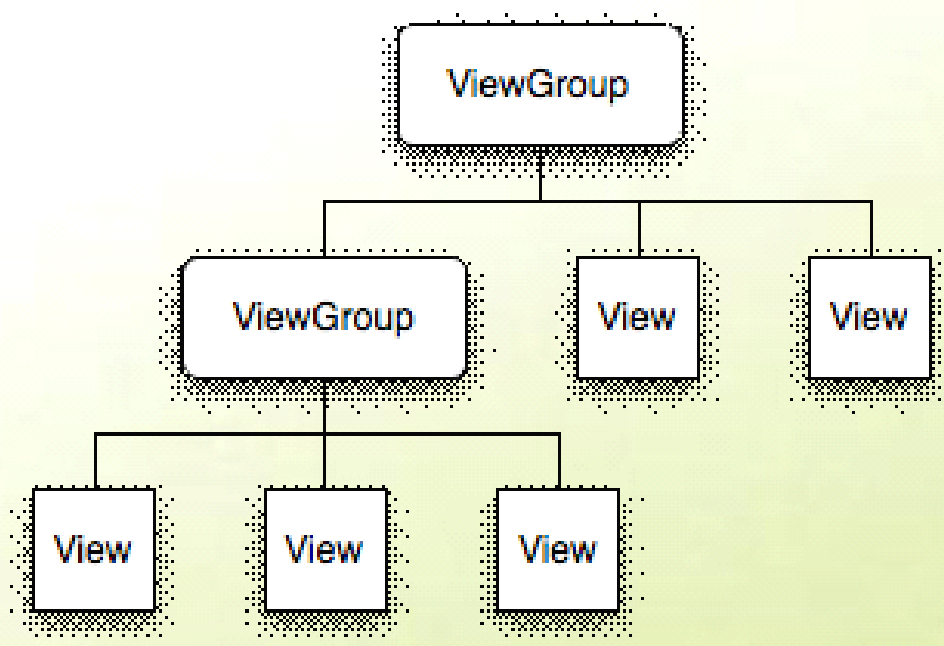


图 4 Android UI——树形结构

为了描述把树和屏幕联系起来，你的 Activity 调用它的 `setContentView()` 方法和传递一个树形根节点物件的引用。一旦 Android 系统拥有这个树形根节点引用物件，它就可以直接使用该节点来工作，使无效、尺寸度量和绘制树。当你的 Activity 变成激活和聚焦，系统通知你的 Activity 并请求根节点度量尺寸并绘制出树。根节点接着请求它的子节点轮流绘制子节点自己，树中每一个 Viewgroup 节点有责任绘制它直系子节点。

如前面陈述，每一个 Viewgroup 有责任度量它的可用空间尺寸，布局它的子节点，并在每个子节点上调用 `draw()` 来它表述自己。子节点可能会在父节点中请求一个尺寸和位置，但是父物件有每一个子 View 放哪里、有多大的最终决定权。



## Android UI 元素与 Swing UI 元素比较

一些正在阅读本文、大概已经用 SwingUI 编过程的开发者会觉得 Android 和 SwingUI 元素有一些相似。

- Activity 在 Android 中差不多和 Swing 中的(J)Frame 相参考
- View 在 Android 中和 Swing 中的(J)Components 相参考
- TextView 在 Android 中和 Swing 中的(J)Lable 相参考
- EditText 在 Android 中和 Swing 中的(J)TextField 相参考
- Button 在 Android 中和 Swing 中的(J)Button 相参考



在 Android 中给一个 View 设置监听器和在 Swing 中几乎相同。

```
//Android
myView.setOnClickListener(new OnClickListener(){ ...
// Swing
myButton.addActionListener(new ActionListener() {...
```





## AndroidManifest.xml

AndroidManifest.xml 是所有 Android 应用程序必须的文件。它位于应用程序的根目录中，并且为你的套件描述全局变量，包括套件向外界暴露的应用程序组件（Activity、Service、等等），你的每个 Activity 和 `co.` 可以处理什么样的数据，和它们如何被运行。



该文件提及到一个重要的事情是它调用 `IntentFilter`。这些 `Filter` (过滤器) 描述了哪里和什么时候 Activity 能够启动。当一个 Activity (或是操作系统) 想要执行一个动作，例如打开网页或是打开一个选择联系人屏幕，它创建一个 `Intent` 对象。这个 `Intent` 对象能够保存许多描述了你要干什么的信息，什么数据需要完成和其他小信息。Android 在一个 `Intent` 对象中使用每个应用程序暴露出来的 `IntentFilter` 比较这个信息，并找出最适合的 Activity 来处理这个数据或是被拨号器指定的动作。假如有超过一个应用程序能够处理那个 `Intent`，Android 就会询问用户喜好哪一个应用程序来处理它。

除了声明你应用程序的 Activity、Content Provider、Service 和 Intent Receiver，你也可以在 AndroidManifest.xml 中指定许可。



## 一般的

一个非常简单的 AndroidManifest.xml 文件像这样的:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.anddev.android.hello_android"
    <application android:icon="@drawable/icon">
        <activity android:name=".Hello_Android"
            android:label="@string/app_name">
            <intent-filter>
                <action
                    android:name="android.intent.action.MAIN" />
                <category
                    android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

- 几乎每一个 AndroidManifest.xml (也包含许多 Android XML 文件) 在它第一个元素中都会包括名字空间声明 (xmlns:android=http://schemas.android.com/apk/res/android)。这样在该文件中做了一种标准的 Android 属性值。这个属性将会用来为该文件中元素支持多数数据。
- 几乎每一个 Manifest 都包括了一个单独的 <application> 标签, 该标签会包含许多描述在本程序中可用的 Application (应用程序)、IntentReceiver, 等等的标签。
- 如果你想要让一个 Activity 通过用户可直接运行, 你将需要让他支持 MAIN 动作和 LAUNCHER category (种类), 它的结果如下所示:

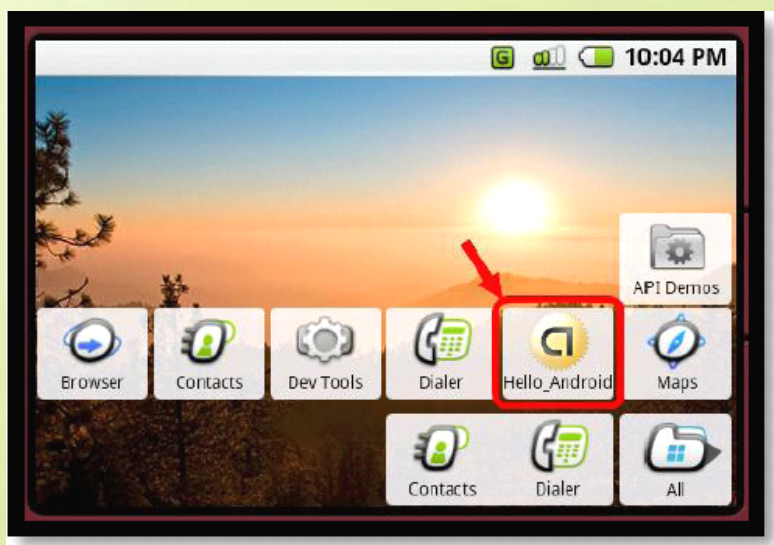


图 5 可直接运行的 Activity

由 [anddev.org](http://anddev.org) 提供





## andbook - Android 编程

以下是一个 AndroidManifest 文件结构的详细列表，描述了所有可用的<tag>标签，每一个附有示例：

### <manifest>

这是每一个 AndroidManifest.xml 的根节点。它包含了指向任何包里面 Activity 外层的套件属性。其他的 Activity-路径将会 c 对的基于它的值。

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.demo.android.bmi2.toast"
```

### <uses-permission>

描述一个为了让你应用程序操作正确（换句话说，当你想要发短信或是使用电话练习簿时）它必须被授予的安全许可。在安装你的应用程序期间，这个许可通过用户授予。数量级:0<sup>+</sup>

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

### <permission>

声明一个安全许可，可以用来限制哪些应用程序可以访问你套件中的组件和特性（或其他的）。数量级:0<sup>+</sup>

### <instrumentation>

声明一个编排组件的代码，它可用来测试组件的功能或是其他套件。参看更多 instrumentation 的详细内容。数量级:0<sup>+</sup>

### <application>

根元素，包含了套件中包含的应用程序级组件的声明。该元素也可以为应用程序包括全局且/或默认属性，例如一个 Label 文本标签，icon，theme 主题，必须的许可，等。数量级:0 或 1

```
<application android:icon="@drawable/icon">
```



你可以把 0 替换成下面的每个子元素：

## <activity>

一个 Activity 是应用程序和用户交互的主要事物。当运行一个应用程序时用户看到的首显屏幕是一个 Activity，并且多数用户使用的其他屏幕将会作为独立的 Activity 通过添加 Activity 标签声明被实现。



```
<activity android:name=".Welcome"
          android:label="@string/app_name">
```



**注意：**不论它被展露给世界或只是有意在它自己套件中使用，每一个 Activity 在 Manifest 中必须含有一个<activity>标签。如果一个 Activity 在 Manifest 中没有符合的 Activity 标签，你将不能运行它。

随意地，为了支持新的运行时更新，你可以包含 1 个<intent-filter>元素来声明该 Activity 支持的动作。

## <intent-filter>

声明了一个组件支持什么类型的 Intent。除了多种能在这个元素下被说明的值以外，这里给出的属性也可以为已经描述的动作 action 支持一个单独的 label、icon 和其他信息。

```
<intent-filter>
```

## <action>

一个该组件支持的动作类型。例如：

```
<action android:name="android.intent.action.MAIN" />
```





## <category>

一个该组件支持的 category 类型。例如：

```
<category android:name="android.intent.category.LAUNCHER" />
```

## <data>

一个该组件支持的 MIME 类型，URI 体系、URI 权威或路径。  
你也可以用你的 Activity 来联合 1 + 个 meta-data 片段：

## <meta-data>

添加一个 meta-data 片段到 Activity 中，客户可以通过 ComponentInfo.metaData 来检索。

## <receiver>

一个 IntentReceiver 允许一个应用程序被告知交换数据和发生的动作，即使它现在没有运行。因为有了 Activity 标签，你可以随意地包括 1 + 个 <intent-filter> 元素，这个 <intent-filter> 是该 Receiver 支持的或是 <meta-data> 值，就像 <activity> 一样。

```
<receiver android:name=".SMSReceiver">
```

## <service>

一个 Service 是一个任意时间量内可以在后台运行的组件。由于有了 Activity 标签，你可以随意包含一个或多个 <intent-filter> 元素，这个 <intent-filter> 是该 Service 支持的或是 <meta-data> 值；更多信息参看 Activity 的 <intent-filter> 和 <meta-data> 描述。

## <provider>

一个 ContentProvider 是一个管理持续数据并发布它来让其他应用程序访问。你可以随意附加一个或多个 <meta-data> 值，正如 Activity 的 <meta-data> 描述一样描写。

当然，所有 <tag> 标签不得不使用 </tag> 或是直接地 </> 来结束。



# 资源和魔幻的 R.java 文件

一个项目的资源和 R.java 是非常紧密相关的。

## 资源

资源是被你代码使用和在构建时编译进你的应用程序中的额外的文件（非代码文件）。Android 支持一些不同种类的资源文件，包括 XML、PNG 和 JPEG 文件。XML 文件根据它们描述的不同而有不同的格式。资源比源代码更具体化，XML 文件被编译成二进制码，由于高效的原因，这是快速加载的格式。字符串被压缩进更高效的存储格式中。



## 资源列表

资源类型和它们存放位置：

- 布局文件 `"/res/layout/"`
- 图片 `"/res/drawable/"`
- 卡通动画 `"/res/anim/"`
- 风格、字符串、数组 `"/res/values/"`
  - 名称不是必须严格的像这样：
  - `'arrays.xml'` 定义数组
  - `'colors.xml'` 定义颜色
    - `#RGB` `#ARGB` `#RRGGBB` `#AARRGGBB`
  - `'dimens.xml'` 定义尺寸
  - `'strings.xml'` 定义字符串
  - `'styles.xml'` 定义风格对象
- 未加工文件:mp3/video `"/res/raw/"`





## 在代码中使用资源

在代码中使用资源只是一个知道完整资源 ID 和你资源已被编译成什么类型的问题。这里是相关资源的语法：

```
R.resource_type.resource_name
```

或

```
android.R.resource_type.resource_name
```



`resource_type` 是 `R` 的子类，该子类拥有指定类型的资源。`resources_name` 是在 XML 文件中定义的资源的名称属性，或者是被定义成其他文件格式资源的文件名（除扩展名）。每种类型的资源将会被加载到一个特殊的 `R` 子类中，依赖于该资源的类型。

被你自己的应用程序编译的资源可以不用包套件名（如 `R.resource_type.resource_name` 一样简单）就可以引用。`Android` 包含了一些标准资源，例如：屏幕风格和按钮背景。为了把这些资源引用到代码中，你必须用 `android` 来表述它们，例如：`android.R.drawable.button_background`。



## 资源引用

在一个属性中(或资源中)提供的值也可以是一个其他资源的引用。这经常用在 `layout` 文件中来提供字符串(因此它们可以局部化)和图片(在其他文件中存在的), 而一个引用可以是任何包含颜色和整数的资源类型。

例如, 如果有颜色资源, 我们可以编写一个设置文本颜色大小的 `layout` 文件, 颜色、大小是包含于这些资源中的值:



```
<EditText android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:textColor="@color/opaque_red"
android:text="Hello, World!" />
```

注意这里 '@' 字首的使用来引入一个资源引用--紧跟的文本是一个处于 `@[package:]type/name` 结构中的资源名称。由于这样, 我们不需要指定套件名, 因为我们在我们自己套件包中引用一个资源。要引用一个系统资源, 你应该需要编写:

```
<EditText android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:textColor="@android:color/background_light"
android:text="Hello, World!" />
```

像其他示例一样, 当在一个 `layout` 文件中提供字符串你应该总是使用资源引用, 以便它们能够被局部化:

```
<EditText android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:textColor="@android:color/opaque_red"
android:text="@string/hello_world" />
```





## 可选资源&局部化

可选资源和局部化是一个非常有用的结果，你将会喜爱 Android 这样。期待你会同时设计出一个像风景的 GUI 和好的肖像屏幕定位--那是几乎不可能的。

你可以根据 UI 语言或是设备的硬件配置为你应用程序提供不同资源。记住即使你可以包括不同的字符串、布局和其他资源，SDK 没有显露出方法来让你指定设置哪一个可选资源被使用。Android 为硬件、地域察觉适合的设置，并合适的家在它们。只有用户可以使用设备上的设置面板选择可选语言设置。

为了包含可选资源，使用破折号-分隔的修饰语添加到文件夹名称中，创建平行资源文件夹，指出它实用的配置（语言、屏幕方位、dpi、解析、……）

例如，这有一个工程区分 English 和 German 值资源（这里只是字符串）：

```
MyApp/  
  res/  
    values-en/  
      Strings.xml  
    values-de/  
      strings.xml
```

Android 支持许多修饰类型，每一个带有不同的值。把这些修饰类型加到资源文件夹名的后面，被破折号分割开。你可以添加多个修饰类型到每个文件夹名中，但是他们必须是像它们在这被列出的规则一样出现。例如，为一个完全特定的配置而包含 drawable 资源的一个文件夹应该像这样：

```
MyApp/  
  res/  
    drawable-en-rUS-port-92dpi-finger-keyshidden-12key-dpad-480x320/
```

更典型地，你将是指定一少一个资源被定义的特定配置选项。你可以从完成的列表中丢弃任何一个值，只要留下的值任然处于相同的规则。

```
MyApp/  
  res/  
    drawable-en-rUS-finger/  
    drawable-port/  
    drawable-port-160dpi/  
    drawable-qwerty/
```

Android 将会在运行期间选择一个在资源文件下面的哪一个值最合适，这依赖于当前设备配置。



## 魔幻的 R.java



一个工程的 R.java 是一个自动生成文件，它索引了所有你项目的资源。你在你的源代码中作为一种快捷方式使用这个类来索引你已经包括在你项目中的资源。这是显著强劲的带有代码自动完成的 IDE 特色，这种 IDE 如：Eclipse，因为他让你快速地和交互地找出你在寻找的指定引用。

此外，你要使用的资源确实存在，你就获得了编译时安全。





## HelloWorld --Android 方式

在这本第一手教程中你将学习如何使用一个 XML 布局来创建一个 Android 应用程序。记住，使用 XML-Layouts 是非常好的。

最终的现实结果像这样：

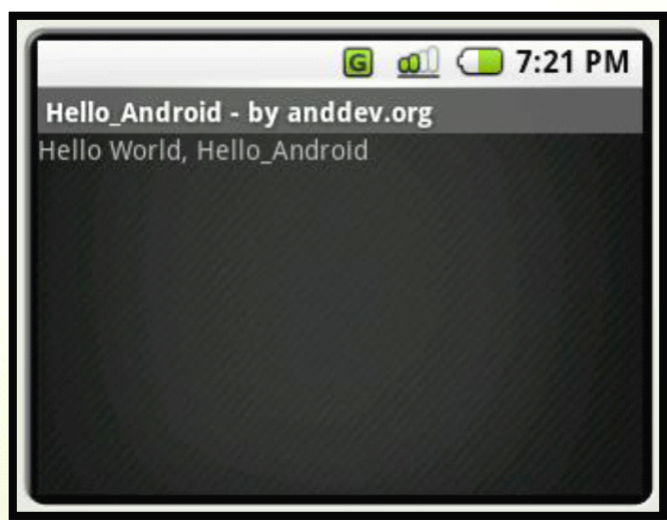


图 6 第一个 Android 应用程序--最终显示(SDK-version m5)

在所有之前，你需要下载和安装 Android SDK……



## 安装 Android SDK

最新 Android SDK --Windows、Mac 和 Linux 版本总可以从以下 URI 获得:

<http://code.google.com/android/download.html>

它只需要加载并解压到你合适的地方。



## Android 开发工具(ADT)

Android 提供了一个叫"ADT"的 Eclipse 插件来让开发和调试更简单。

ADT 提供了 LogCat 简单访问, Android-Manifest(应用程序配置)/Resource-Editor(资源编辑)/File(文件)/Thread(线程)和 Heap Control(堆栈控制)、打进来的电话/短信 仿真, 等等。--从 SDK version m5 , 所有的多种模拟器都同时得到证实。





## 安装 Eclipse 插件(ADT)

要下载和安装 ADT 插件，跟着 Google 提供给开发者的步骤：

1. 启动 Eclipse，接着选择"Help">"Software Updates">"Find and Install"
2. 在出现的对话框中，选择"Search for new features to install" 点击"New"
3. 点击"New Remote Site"
4. 在结果对话框中，为远程站点键入一个名称(如：Android Plugin)并键入这个 URL 作为它的 URL：  
<https://dl-ssl.google.com/android/eclipse/>  
点击 OK。
5. 现在你应该看到新站点加载到搜索列表中(并选中)，点击 "Finish"。
6. 在后来搜索结果对话框中，选择"Android Plugin">"Developer Tools"复选框。这将会选择"Android Developer Tools"和"Android Editors"特性。Android Editors 特性是可选的，但是推荐。如果你选择安装它，你需要在本文早先提及到的 WST 插件。  
现在点击"New"
7. 阅读许可协议,选择"Accept terms of the license agreement",点击"Next"
8. 点击"Finish"
9. ADT 插件没有被签名；总之，你可以点击"Install All"接受安装
10. 重启 Eclipse
11. 在重启后，更新你的 Eclipse 配置来指向 SDK 目录：
  - a.选择"Window">"Preferences" 来打开配置面板。(Mac OS X:"Eclipse">"Preferences")
  - b.从左边面板中选择"Android"
  - c.对于主面板中"SDK Location"，点击"Browse"并选中 SDK 目录位置
  - d.点击"Apply",接着点"OK"

## 更新 ADT 插件

更新 ADT 插件遵循更新一个普通 Eclipse 插件的标准步骤：

- 1.选择"Help">"Software Updates">"Find and install"
- 2.选择"Search for updates of the currently installed features"并点击"Finish"
- 3.如果任何 ADT 更新可用，选中并安装。



## 安装完毕，我们来写点代码

在我们能够开始编写代码前，我们明显地不得不创建一个新的 Android 项目。

### 创建一个新的 Android 项目

1. 需要为每一个 Android 应用程序做的第一件事是创建一个新的 Android 项目。要这样，简单地打开 Eclipse 中的 Package 浏览器，在一些空白处右键并选择：

New >Project .....

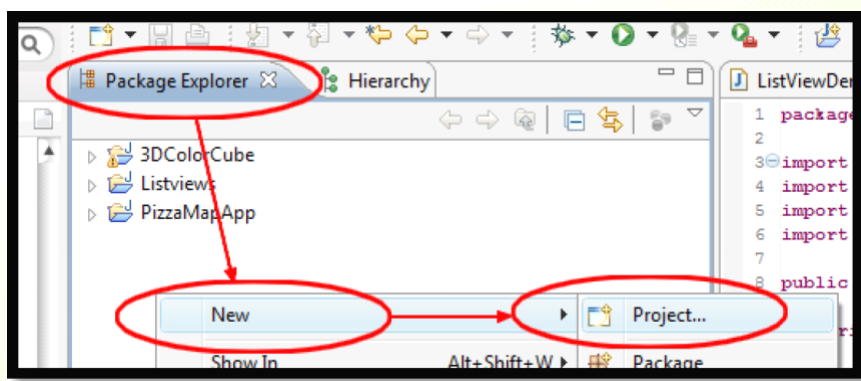


图 7 第一个 Android 应用程序-步骤 1

2. 选择: Android > Android Project

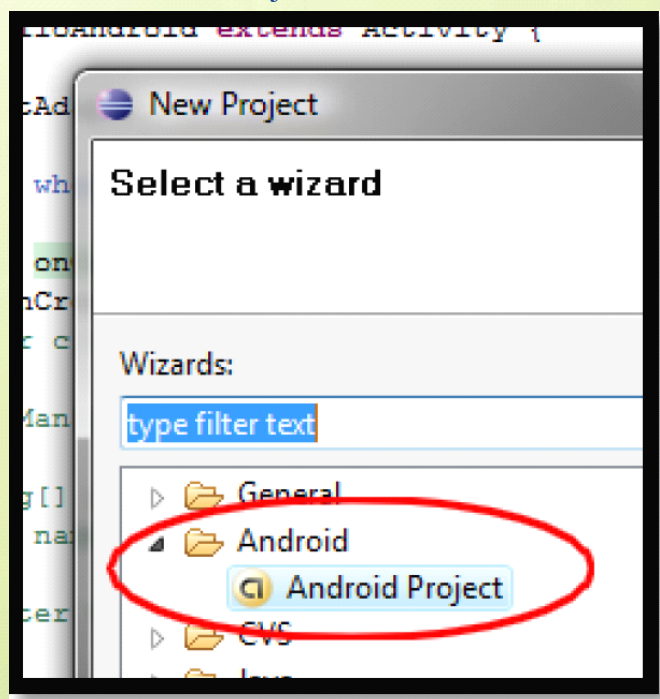


图 8 第一个 Android 应用程序-步骤 2





## andbook - Android 编程

### 3. 用适合你应用程序意图的值来填写表单

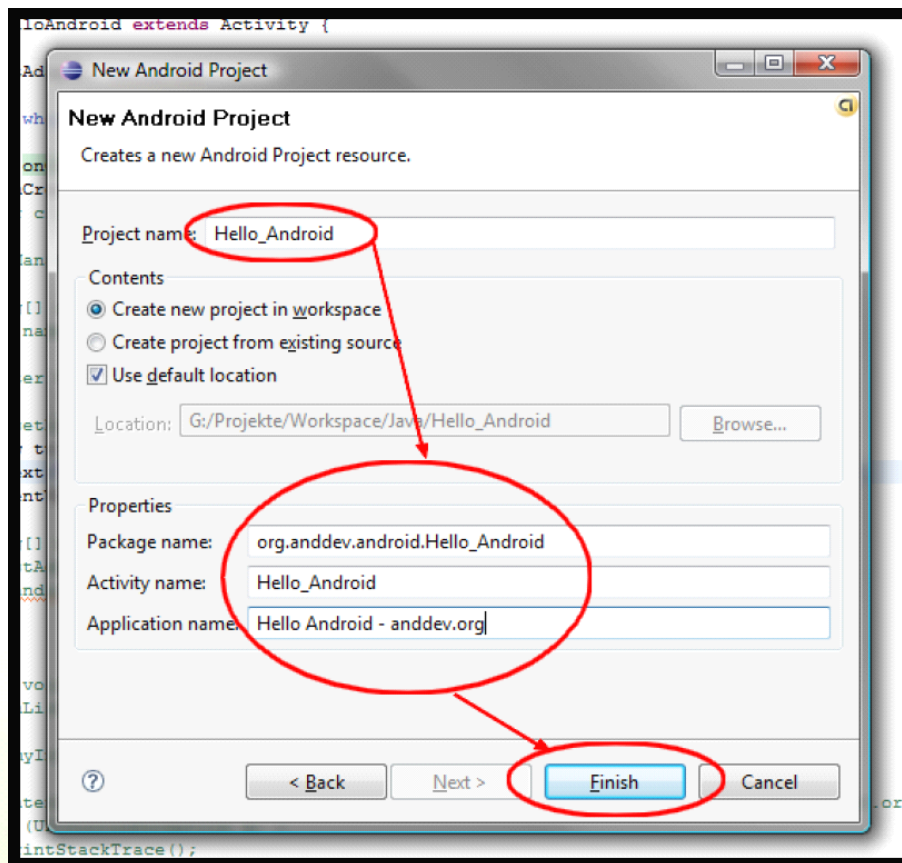


图 9 第一个 Android 应用程序-步骤 3

### 4. 这是你第一个 Android- 应用程序的所有文件(别惊慌, 它们中大多数是资源文件)

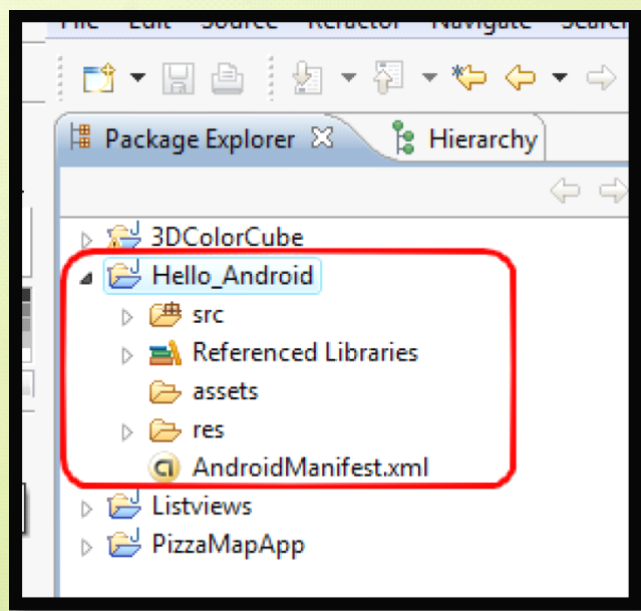


图 10 第一个 Android 应用程序-步骤 4




## 哈哈，所有那些文件可以干什么？

当你现在已经创建类你第一个 Android 项目，你将看到一串文件出现在新项目中。

### Main Activity

当你导航到：`"/src/your_package_Structure/Hello_Android.java"`  
你将看到一些像下面的默认代码：

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

我们可以立即  我们的应用程序但是让我也阐释一下其他文件。

### XML-Layout(main.xml)

ADT 为你创建了这个非常基本的 Activity。正如我们知道的，Activity 在某方面就像 Swing 中的 JFrame。在 Android 中，我们继承自 Activity 并需要重写一个单独的方法，叫做"onCreate()". 在此方法中，我们不得不调用，换句话说，setContentView(R.layout.main)。让我们的 Activity 使用也被 ADT 创建的 main.xml 文件。

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

我们有一个"全屏"垂直的线性布局 LinearLayout，它包含一个显示预设字符串的单个 TextView





## AndroidManifest.xml

我们看看 AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.anddev.android.Hello_Android"
    android:versionCode="1"
    android:versionName="1.0">
  <application
    android:icon="@drawable/icon"
    android:label="@string/app_name">
    <activity android:name=".Hello_Android"
      android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category
          android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
      </activity>
    </application>
    <uses-sdk android:minSdkVersion="2" />
  </manifest>
```

我们拆分开它

每一个 xml 文件以下面一行开始, 它定义了 xml-version 和 xml-文本编码类型。只是把它复制粘贴到每一个新文件中。

```
<?xml version="1.0" encoding="utf-8"?>
```

如我们所知, 每一个 xml 文件中最外层的标签应该包含这个属性:

```
xmlns:android="http://schemas.android.com/apk/res/android"
```

因为它让各种类型的普通 Android 属性在该文件中可用。

```
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.anddev.android.Hello_Android">
</manifest>
```

`<application>` 标签是包含所有包含在套件中的应用程序级别的组件声明的根元素。该元素也可以包括应用程序的全局和、或者默认属性, 例如一个 `label` 标签, `icon`、`theme` 主题、需求许可, 等等。



这里我们将只是定义 icon，通过"@-引用"到一个放在"/res/drawable"下的图片。

```
<application
    android:icon="@drawable/icon">
</application>
```

在<application> 标签中，我们需要定义所有需要通过 Intent 来启动的 Activity/IntentReceiver。在本例中我们只有一个简单的 Activity。

```
<activity
    android:name=".Hello_Android"
    android:label="@string/app_name">
    .....
</activity>
```

你大概已经认识到来自<manifest> 标签的 Package- 属性和来自<activity> 标签的 android:name 属性一起总是产生到 Activity 被描述的完成的包路径中。

最内层的标签是<intent-filter> 标签，它定义了哪个 Intent 我们想要监听。这里我们想要 Hello\_Android Activity 可以在模拟器中通过点击图标（在<application> 标签中定义了的）来运行。

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category
        android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```



注意：Android 应用程序调试过程像一个普通 Java 应用程序一样严密。





## 运行你第一个应用程序

现在我们需要创建一个"配置"。在 Eclipse 上菜单中打开"运行"下拉列表"并选择"Open Run Dialog"

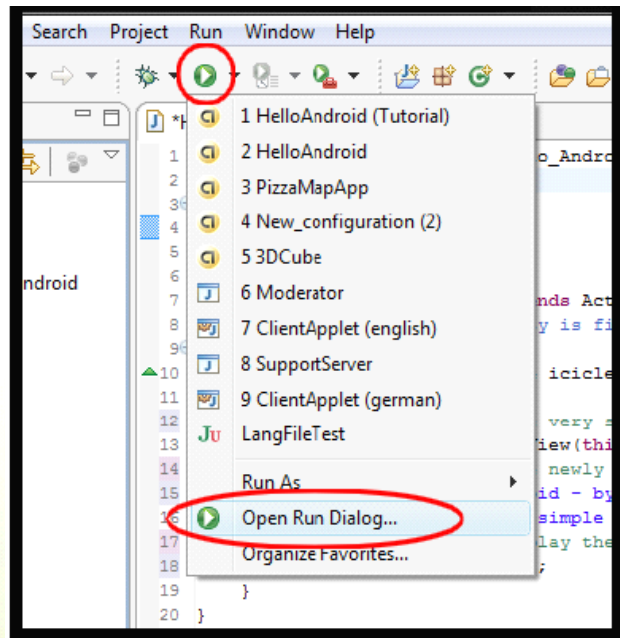


图 11 打开 运行-对话框

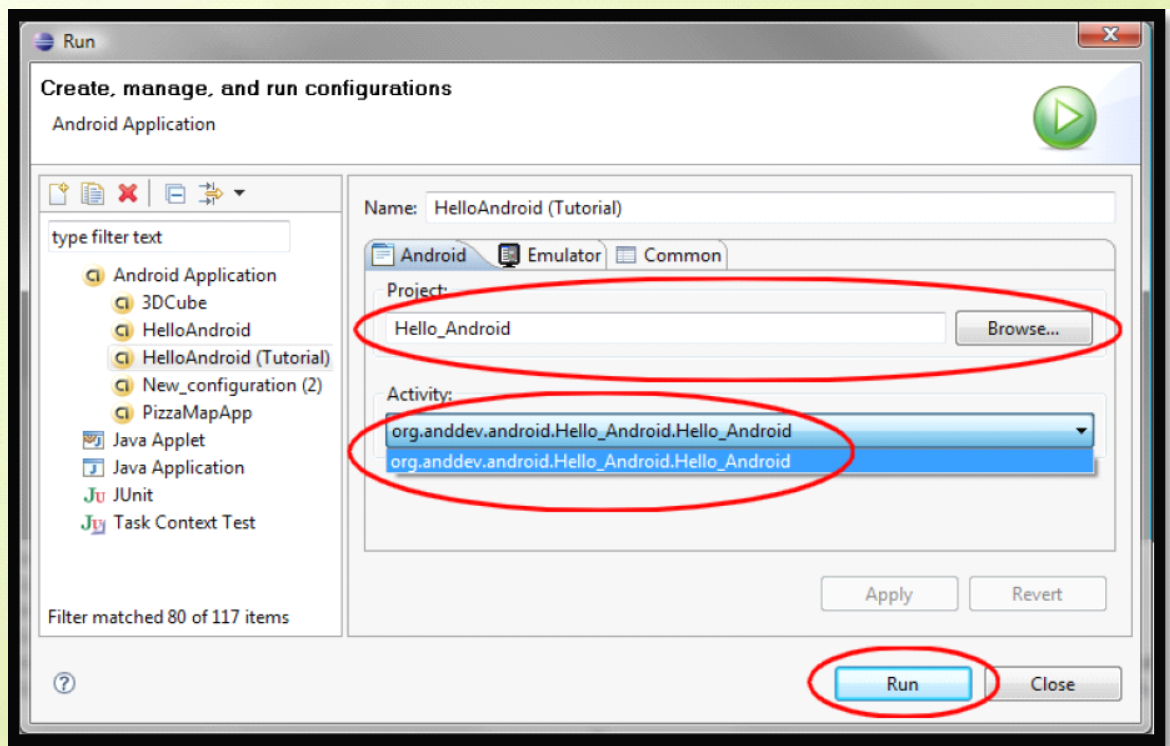


图 12 创建 运行配置



从现在开始，你可以通过点击  来运行你的应用程序。完成后你将看到这个：

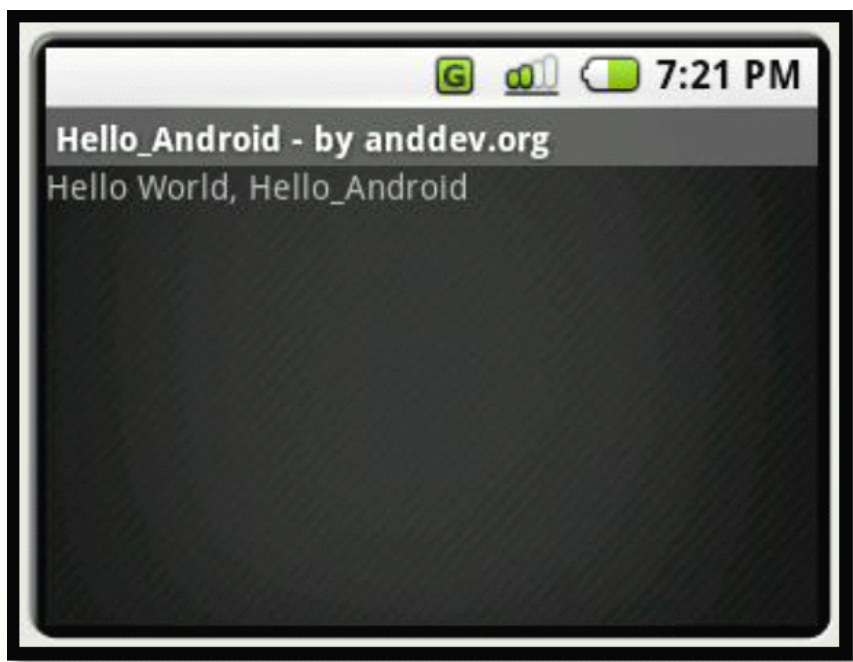


图 13 第一个 Android 应用程序-结果(SDK version m5)

在第一次部署后，你也可以看到你的应用程序被列入快速-菜单中，用默认图标来显示。

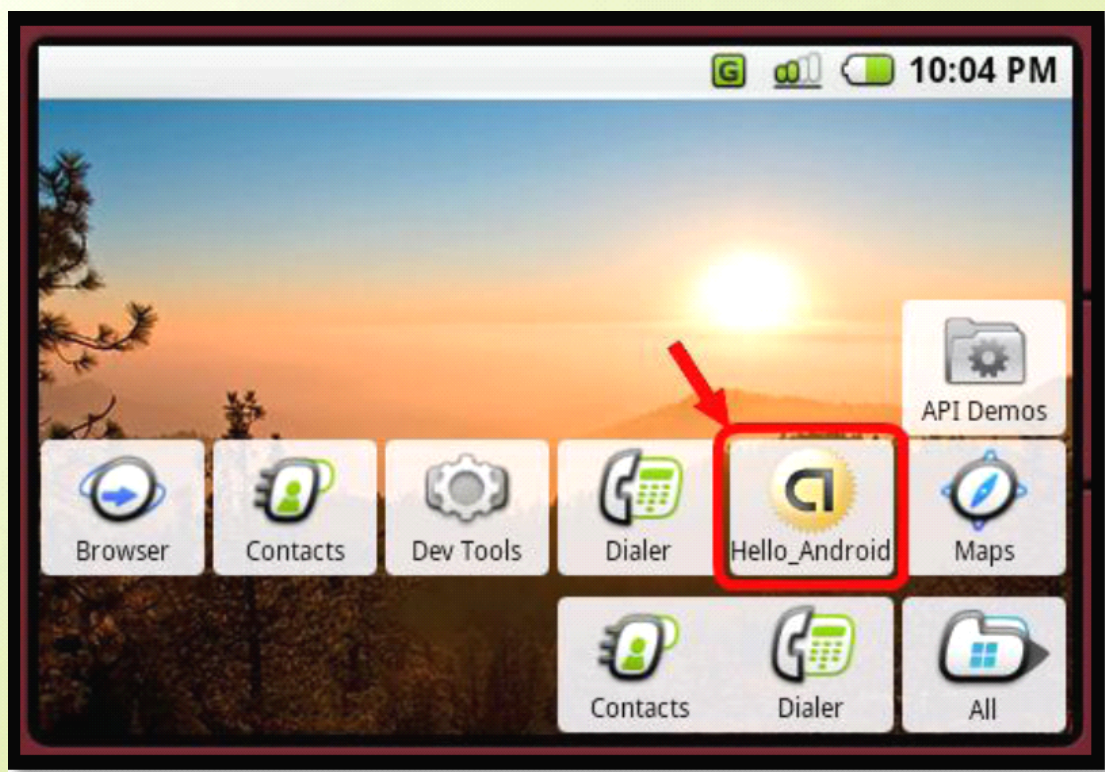


图 14 第一个 Android 应用程序-添加到快速菜单(SDK version m5)





## UI--Java 方式

我们用 XML 代码做的同样的事可以通过编写一些 Java 代码行来实现。记住我们在 xml 例子中如何把我们的 `main.xml` 设置成为 `ContentView`。我们通过进行以下步骤来完成：

```
/* Make this application use
 * * the main.xml-layout-file. */
this.setContentView(R.layout.main);
```



`Activity.setContentview()`也接受一个 `View` 作为参数。我们将使用该方法来设置一个简单的 `TextView` 作为我们的 `ContentView`。

```
package org.anddev.android.Hello_Android;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class Hello_Android extends Activity {
    /** Called when the activity is first created. */
    @Override public void onCreate(Bundle icle) {
        super.onCreate(icle);
        /* We want to view some very simple text,
         * so we need a TextView associated with this Activity. */
        TextView tv = new TextView(this);
        /* Put some text to the newly created TextView */
        tv.setText("Hello Android - by: anddev.org \n"
                   + "This is soooo simple =D ");
        /* Tell our Activity to display that TextView */
        this.setContentView(tv);
    }
}
```



## System.out.println(.....)?

在 Android 中调试不能用 System.out.println(.....)来做, 因为如我们所知: Android 不是运行在一个普通虚拟机上, 而是在 DalvikVM 上的模拟硬件中运行。(老实说, 这可以进行, 但是明确地不是你的选择)

但是别担心, Android 提供了许多强劲的调试特性--LogCat



## LogCat

LogCat 是 DDMS(Dalvik 调试监视器服务)的一部分。它提供了一个收集和查看系统调试输出的机制。来自不同应用程序的 Log 和系统部分被收集到可以查看和过滤的 LogCat 中。

如果你看不到 A, 那就进行 B 方式

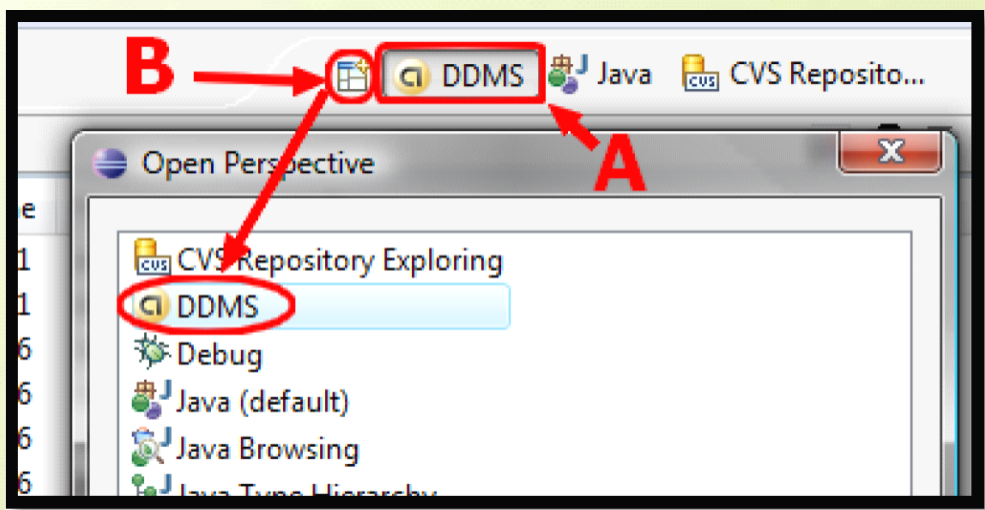


图 15 打开 DDMS 视图





## 使用 LogCat

每一个 Android Log 信息有一个 tag 标签和一个跟它联系的优先级。

Log 信息的 tag 是一个指示了信息从那个系统组件发出来(例如视图系统的"View")的短字符串。

Priority 是以下字符值的一个，从最低到最高的优先级:

- V ---详细信息(最低优先级)
- D ---调试
- I ---信息
- W ---警告
- E ---错误(最高优先级)

当我们正使用 Eclipse 时,我们能够通过点击 LogCat 视图中的"V- D - I - W - E"按钮(你可以在下面看到)以优先级来简单的过滤。由于那个特性,你将会喜欢上 ADT 插件,因为在全部系统生成的输入中,要找到没有被过滤的任何信息都是十分困难的。

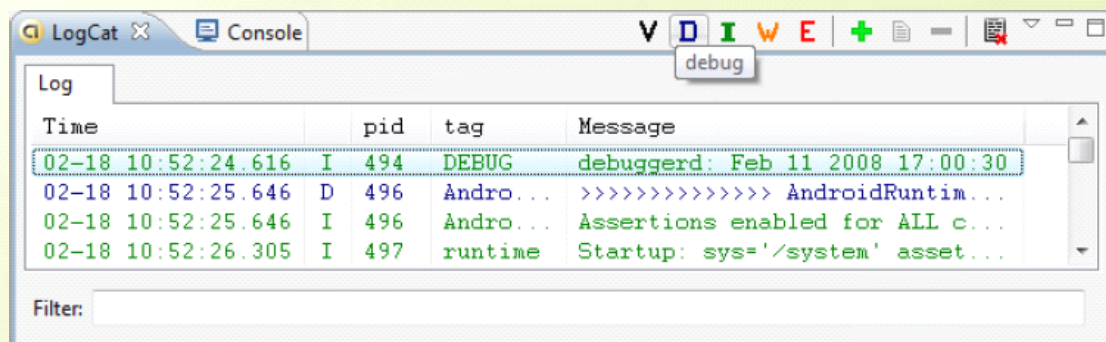


图 16 LogCat

在代码中用法十分简单。你需要做一个单独的导入:

```
import android.util.Log;
```

接着你就可以使用调试陈述, 像这样:

```
Log.d("DEBUGTAG", "My debug-message.");
```

当一些危险性事件发生时, 你就可以通过 Throwable/Exception 到 Log.e(……), 得到打印的错误 StackTrace 输入到 LogCat 中。

```
try{
    throw new Exception();
}catch (Exception e){
    Log.e("DEBUGTAG", "Error occurred", e);
}
```



# 使用 Intent

如我们所知，需要被做的事件用 **Intent** 意图目的来表达。**Intent** 换句话说被用来开启其他 **Activity**。

## 启动(子)Activity



在一个应用程序生命期中一个基本的事情是比 **HelloWorld** 应用程序更精密的，就是启动其他 **Activity**，尤其是子 **Activity**。让我们假设以下情况：

我们想要一个像输入框这样的。用户可以在该输入框中写一个他想要在 **Google** 上查询的关键字。

因此我们将像我们以前已经做过的一样来创建一个 **Android** 项目。第一件事就是添加一个我们将叫做 **"MySecondActivity"** 的第二个 **Activity**。

一开始，代码框架像这样：

```
package org.anddev.andbook.startingsubactivities;
import android.app.Activity;
import android.os.Bundle;
public class MySecondActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle icle) {
        super.onCreate(icle);
        setContentView(R.layout.main);
    }
}
```





## andbook - Android 编程

我们现在将添加一个 Button 到第一个 Activity 中。我们修改 main.xml 来完成，当然，不通过选择 Java UI 来完成。

浏览"/res/layout/main.xml" 并且你将看到和下面相似的代码：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Hello World, StartingSubactivities" />
</LinearLayout>
```

注意：也许 Eclipse 用它自己的 xml-编辑器（对我们没有任何用处的编辑器）打开这个 main.xml。从 SDK-version m5 开始，ADT 插件提供了一个带有高亮洁语法的 Resource-Editor。

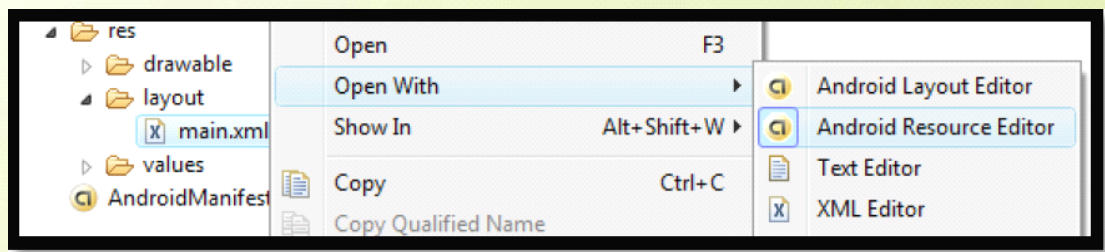


图 17 打开 Resource-Editor

如果我们现在启动该应用程序，它像这样的：

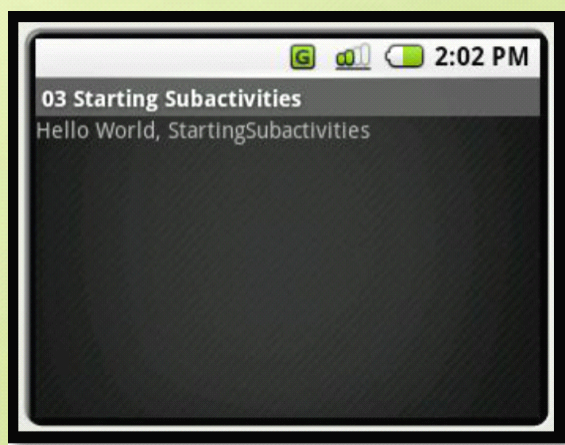




图 18 默认布局(SDK -version m5)

当我们想要在我们应用程序中有一个 **Button** 时，我们将不得不加一些 xml 代码:

```
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Open Search-Dialog" />
```

如你可以想象的, "fill\_parent" 让一个 **View** 使用所有父 **View** 提供的空间, "wrap\_content" 只是使用它需要正确地显示自己内容的布局空间。

因此我们的 **button** 将宽度填满整个屏幕, 包装我们键入的 **android:text** 属性 "Open Search-Dialog"。

我们的应用程序现在像这样:

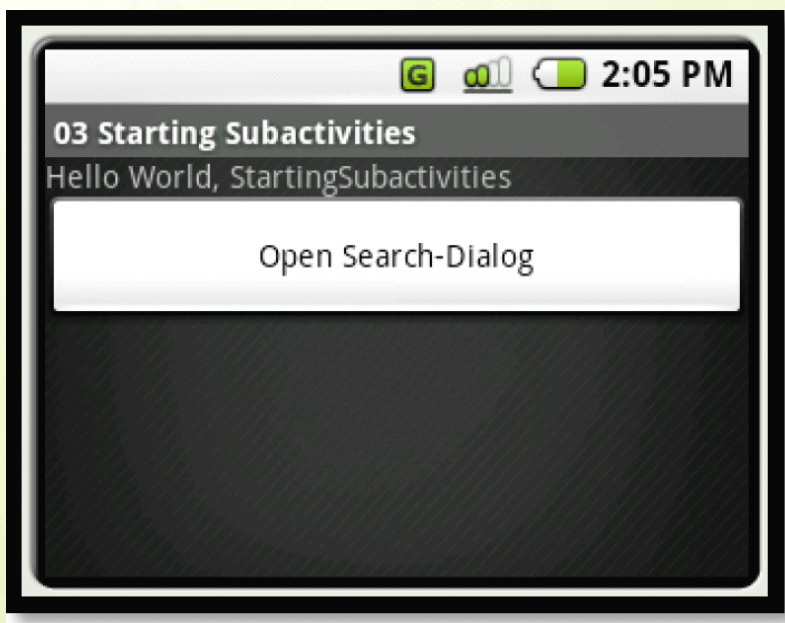


图 19 带有按钮的布局(SDK -version m5)

当我们现在点击该按钮--当然, 没有事件发生。我们不得不申请一个监听器, 是特殊的一个 **OnClickListener**, 来处理在这个按钮上的点击事件。

但是等等……我们怎样在 **Java** 代码中引用到那个按钮?





## 在 Java 代码中找到 XML 定义的 View

为了找到一个在 XML 被定义的 View，我们第一不得不添加那个 View 的一单行定义到 XML 定义中，一个 `android:id` 属性。在此例我们将给它 `"btn_open_search"` 这个 id:

```
<Button
    android:id="@+id/btn_open_search"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Open Search-Dialog" />
```

在 Java 代码中找到那个 View 同样简单。在一个 Activity 中你可以使用 `"findViewById(int resourceID)"` 方法，使用已经在 XML 中定义的 `android:id` 来得到一个 View 的引用。

同样其他继承于 View 的类也可以这样做，换句话说，就是 EditText、TextView、MapView 等等……



但是 `"findViewById(int resourceID)"` 只能被用在 View 上面。这些 View 是放进布局中的。而这个 Layout 布局被 Activity 使用 `setContentView()` 来加载的。

回到我们例子中，我们添加了以下代码到 `onCreate()`，在 `setContentView()` 的右边：

```
/* Find the Button from our XML-layout. */
Button b = (Button) this.findViewById(R.id.btn_open_search);
```

如果 Eclipse 不能找到 Button 类，只是敲击 `"Ctrl+Shift+O"` 即可，它将会组织导入和自动添加(本例中):

```
import android.widget.Button;
```





## 为 View 处理点击

我们现在记得我们曾想让我们的 Button 按钮可点击。因此我们简单的设置一个未命名的 OnClickListener 到我们的 Button 上。

```
b.setOnClickListener(new OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        // TODO Auto-generated method stub  
        // Place code to handle Button-Click here.  
    }  
});
```

注意: Eclipse 可能不会自己不会认识到以下的导入:

```
import android.view.View.OnClickListener;
```

现在我们将添加一个 Intent 到 onClick 事件中。该事件会启动我们的子 Activity:

```
/* Create an Intent to start *  
 * MySecondActivity. */  
Intent i =  
    new Intent( StartingSubactivities.this, MySecondActivity.class);  
/* Send intent to the OS to make  
 * * it aware that we want to start  
 * * MySecondActivity as a SubActivity. */  
startSubActivity(i, 0x1337);
```

startSubActivity()第二个参数可以是任何唯一整数。他将在后面有用(当我们将替换它向我们的 Activity 声明为 final 时),当我们想要从 SubActivity 中返回一个结果时。

以上方法已经不用了,在新的 SDK version1.5 中,使用以下方法:

```
/* Create an Intent to start * MySecondActivity. */  
Intent i = new Intent( NextActivity.this, MySecondActivity.class);  
/* Send intent to the OS to make  
 * * it aware that we want to start  
 * * MySecondActivity as a SubActivity. */  
startActivity(i);
```

startSubActivity()方法已经不用了。要从 SubActivity 中返回一个结果,使用以下两个方法:

```
startActivityResult(intent, requestCode);  
startActivityFromChild(child, intent, requestCode);
```





## andbook - Android 编程

如果我们现在就运行我们的代码并且点击我们的按钮，我们将会收到以下错误信息：

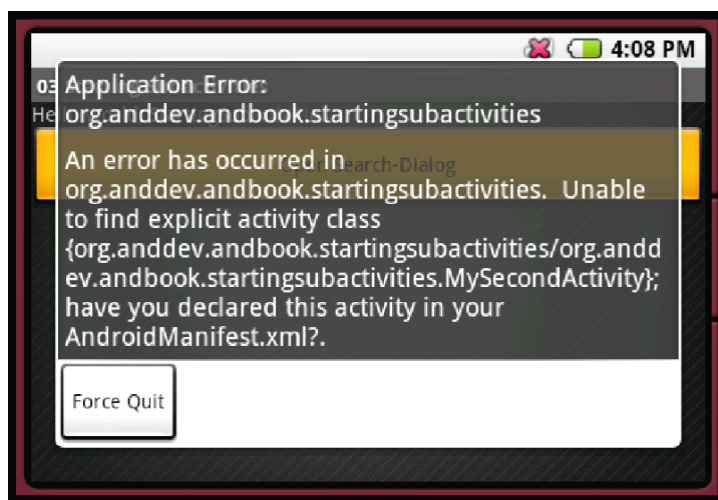


图 20 Activity 没有在 AndroidManifest.xml 中定义



注意：显示的信息中一些十分有用，阅读它们。

我们也需要在 `AndroidManifest.xml` 文件中定义我们的 `"MySecondActivity"`。只要在第一个 `</activity>` 标签后面编写：

```
<activity
    android:name=".MySecondActivity"
    android:label="@string/app_name">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category
            android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

这时我们不为 `<action>` 选择 `"MAIN"`，并且也不为 `<category>` 选择 `"LAUNCHER"`，因为不需要让 `"MySecondActivity"` 从我们应用程序外部运行。

现在 `"MySecondActivity"` 可以通过点击该按钮来达到了。但是他也引用到我们的 `main.xml`：

```
setContentView(R.layout.main);
```



因此我们不得不为"MySecondActivity"创建一个额外的 Layout. 该 Layout 经包含一个叫做 EditText 的(在 Swing 术语中叫做 TextField)和其他返回到我们"主"-Activity 的 Button:

```
<EditText
    android:id="@+id/et_keyword"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

<Button
    android:id="@+id/btn_return"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Submit" />
```



当然, 两个都需要一个我们在 Java 代码中使用到的 android:id。现在我们可以修改"MySecondActivity"的 setContentView 成:

```
package zyf.myTest;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class StartingSubactivities extends Activity
    implements OnClickListener {

    private Button open;
    private Intent toIntent;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        open=(Button) findViewById(R.id.btn_open_search);
        open.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        if(v.getId()==R.id.btn_open_search){
            toIntent = new Intent();
            toIntent.setClass(StartingSubactivities.this,
                                MySecondActivity.class);
            startActivity(toIntent);
        }
    }
}
```





## 从 SubActivity 返回值

从 SubActivity 返回数值给调用者也非常简单：（灰色部分）

```
package zyf.myTest;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
public class MySecondActivity extends Activity
                                implements OnClickListener {

    private EditText getText;
    private Button back;
    private String getString;
    private Intent backIntent;
    private Bundle backBundle;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.second);
        getText=(EditText) findViewById(R.id.et_keyword);
        back=(Button) findViewById(R.id.btn_return);
        back.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        if(v.getId()==R.id.btn_return){
            getString=getText.getText().toString();
            backIntent=new Intent();
            backBundle=new Bundle();
            backBundle.putString("GET_TEXT", getString);
            backIntent.putExtras(backBundle);
            setResult(RESULT_OK, backIntent);
            finish();
        }
    }
}
```





你可以额外地通过一个叫做 **Bundle** 的来返回给调用者(或多或少是一个 **HashMap**)，但是我马上会告诉你更多它的信息。

明显地调用者不得不响应 **SubActivity** 指定返回的行为。为了达到那样，我们不得不重载一个单独的来自于 **Activity** 类的方法。它叫做 **onActivityResult()**:

```
@Override
protected void onActivityResult (int requestCode,
                                int resultCode, Intent data) {
    // TODO Auto-generated method stub
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode==this.requestCode) {
        if (resultCode==RESULT_CANCELED) {
            setTitle ("Result Canceled");
        } else {
            String getStringFrom=data.getStringExtra ("GET_TEXT");
            setTitle (getStringFrom);
        }
    }
}
```

你可能认识到第一个参数叫做 **requestCode**-- 是的它和我们早期传给

```
startActivityForResult (toIntent, requestCode);
```

的 **requestCode** 相同。因此，假如我们有更多的 **SubActivity**，我们能使用 **requestCode** 来区分哪一个 **SubActivity** 返回。

贴士：Eclipse 提供一个你将喜爱的非常有用的方法，尤其是如果你正在继承一个基类时并在寻找重载的方法：

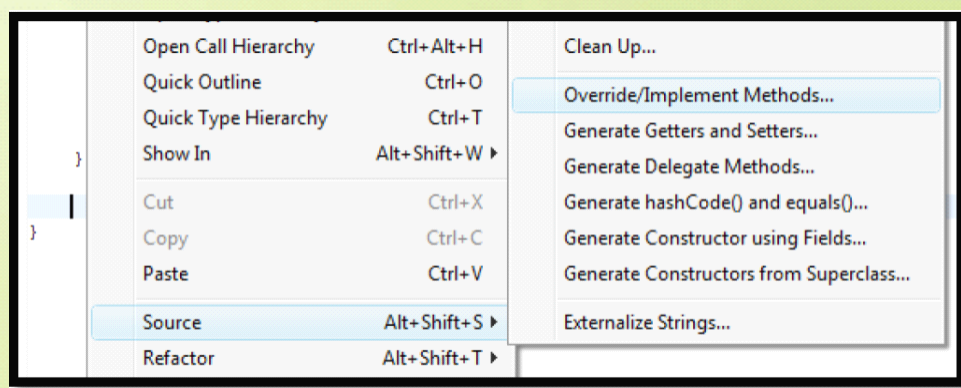


图 21 使用 Eclipse 来找到重载的方法





因此，我们现在可以简单的选择 `requestCode` 和创建其他的 `Intent` 来做 Google 搜索。

```
switch (requestCode) {  
    case MYSECONDACTIVITY_REQUESTCODE :  
        /*  
         * Create a new Intent to  
         * show the Google-Search Page *  
         * with the keyword returned.  
         */  
        Intent webIntent =  
            new Intent (Intent.ACTION_VIEW,  
                Uri.parse ("http://www.google.com/search?q=" + getStringFrom));  
        startActivity (webIntent);  
        break;  
}
```

当你这时大概已经认识到这个了，我们将用其他方式创建 `Intent`。上次我们像这样说："我们想要开启 `XYZ.class`"，但是这次我们描述了我们想要得到。我们想要查看一个 `URI`(统一资源定界符)，它可以通过使用 `Uri.fromParts(".....")`来构造。

我们的应用程序现在能开启一个(子)`Activity`，抓取它的返回结果和用一个 `Intent` 来运行浏览器。

但是如何来传递数据给一个(子)`Activity`？



## 给(子)Activity 传递数据

再次，传递数据给(子)Activity 不是难于控制的。你可以使用叫做 **Bundle** 的从来从一个 Activity 携带信息到另一个 Activity。这个 **Bundle** 或多或少是个普通的 **HashMap**，它只可以携带普通的数据类型。只是记住上一章中我们在哪里启动我们的子 Activity：

```
/* Create an Intent to start * MySecondActivity. */
Intent i = new Intent( NextActivity.this, MySecondActivity.class );
/* Send intent to the OS to make
* * it aware that we want to start
* * MySecondActivity as a SubActivity. */
startActivity(i);
```

在这两行的右边，我们将放入下面的代码：(灰色部分)

```
toIntent = new Intent();
toIntent.setClass(NextActivity.this, MySecondActivity.class);
Bundle passBundle=new Bundle();
passBundle.putString( "MY_DEFAULTSTRING_ID", "anddev.org" );
toIntent.putExtras( passBundle );
startActivity( toIntent );
```

**MY\_DEFAULTSTRING\_ID** 可以是任意你想象中的识别字符串。也有个 **Intent.putExtra(……,……)** 方法，此方法每次调用只携带一个信息，我们也应该用它。

现在我们不得不从我们子 Activity 中抽取那个信息。通过调用 **getIntent()**，每个 Activity 可以访问原开启该 Activity 的 **Intent**。如果有一个 **Bundle** 绑定到这个 **Intent** 上，我们可以通过使用 **getIntent().getExtras()** 来抓取到该 **Bundle**。在本例中，我们将填充 **EditText**，它里面用户会键入带有我们通过 **Intent** 传过去的 **DefaultString** 的关键词：

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.second);
    getText=(EditText) findViewById(R.id.et_keyword);
    back=(Button) findViewById(R.id.btn_return);
    back.setOnClickListener(this);
    Bundle getBundle=getIntent().getExtras();
    String default_keyword=
        getBundle.getString( "MY_DEFAULTSTRING_ID" );
    getText.setText( default_keyword );
}
```





让我们重新回顾一下我们在这里学到了些什么：

- 使用 XML 布局创建一个简单的应用程序
- 用基于 Java 的布局来创建
- 运行 Android 应用程序
- 使用 Logcat 来调试
- 使用 Intent 来开启(子)Activity
- 找到在 XML 中定义的 View、在 Java 代码中使用它们
- 为 View 处理点击事件
- 从 SubActivity 返回值
- 使用 Bundle 传递数据给 SubActivity

现在还不糟糕





## 重要 Layout 和 ViewGroup

Android 提供了一些更多的 Layout 而不是只有我们在这里使用过的 LinearLayout。除了 LinearLayout 以外,你将在这里更进一步了解最重要的 Layout:

- ◆ RelativeLayout(View 被放在相对其他 View 的位置)
- ◆ TableLayout(像在 HTML 中一样使用表格)

额外地我下面将会向你介绍真正经常使用的 ViewGroup:

- ♣ GridView(和 TableLayout 相似,支持"Adapters"适配器)
- ♣ ScrollView(如果你超过了屏幕尺寸,你将需要滚屏)
- ♣ TabHost(在制表中显示内容)



## RelativeLayout

根据它的名称,当使用 RelativeLayout 时,你可以定义 View 相对于其他"相邻 View"的位置。





## 参考书目

1. Open Hanset Alliance. Open Hanset Alliance. [Online]

<http://www.openhandsetalliance.com/>

2. GoogleGroups. Android Discussion Groups. [Online]

<http://code.google.com/android/groups.html>



3. Alexa Webstats. The Web Information Company. alexa.com. [Online]

<http://www.alexa.com>

4. Gramlich, Nicolas. Android Development Community | Android Tutoria ls. [Online]

<http://anddev.org>

5. Hobbs, Zach. Hello Android. [Online]

<http://helloandroid.com/>

6. Nguyen, Vincent. Android Community. [Online]

<http://androidcommunity.com/>

7. Srinivas, Davanum. Show me the code! [Online]

<http://davanum.wordpress.com/>



## 缺失章节

## 社区

从第一天开始，一组社区就从 Internet 深处成立了。提到(依赖于 Alexa-stats(3))的最重要的社区网站是 [anddev.org](http://anddev.org)(4),它是转为 Google Androi 设置的最新的一个可视化的教程、社区平台。知名的也有 [helloandroid.com](http://helloandroid.com) 社区平台(5)和 [androidcommunity.com](http://androidcommunity.com)(6)。最后的但明确地不是最新的提及到的是 [davanum.wordpress.com](http://davanum.wordpress.com)-Blog(7),它们提供了一些非常简单的代码，是许多开发者找到好的入门资源的地方。

一些站点更多、一些更少有指望，因此，观察者迟早会把他们指定到以上更重要的站点清单中。

## 翻译&学习完毕

时间：2009 年 5 月 9 日 凌晨 3: 09

译者：张劲锋

职业：学生

专业：软件工程

邮箱：[zyf19870302@126.com](mailto:zyf19870302@126.com)

谢谢！！



אנדרואיד  
developers



אנדרואיד