

移动开发精品系列



# Android SDK 上手指南

51CTO移动开发频道

出品

# 权利声明

Android SDK上手指南

译自 *tuts+ Code Tutorials: Learn Android SDK From Scratch*

译者： 核子可乐、pockry

编校： pockry

出品： 51CTO移动开发频道

本电子书内容著作权归原作者及译者所有。

本电子书由51CTO移动开发频道制作并发行，禁止用于商业目的。

# 编者按

越来越多的人开始尝试Android开发，因为它成本极低，所有的工具基本都能免费获得，并且市场广大，我们开发出的产品可以有很多潜在用户，如果是做着自己用，也很有价值。

网络上Android开发的入门教程已经很多，我们在这里推出的是tuts+上面的最新版本，有理论的讲述也有实践的指导，如果你正打算学习Android开发，这将是一本很好的入门指导。

本书针对的读者是刚开始准备学习Android开发的人，需要有一定的编程经验，但不需要太多，如果了解过Java、学习过HTML足矣。

你可以在[这里](#)看到本书的在线版，在[这里](#)看到它的英文原版。

# 目录

## 第一章 环境需求

让我们从安装Android开发环境开始。

## 第二章 IDE: Eclipse速览

Eclipse原本是Google官方推荐的IDE，对于开发一个Android app来说它足够好用。

## 第三章 IDE: Android Studio速览

Google发布了官方的Android开发IDE: Android Studio，我们有必要对它来一个检视。

## 第四章 应用程序结构

在开始开发前，我们有必要了解一个Android App的结构。

## 第五章 用户界面设计

用户界面是移动App非常重要的部分，我们来学习如何构建用户界面。

## 第六章 用户交互

用户交互逻辑是一个App的核心。

## 第七章 Java应用程序编程

Android程序是用Java语言编写，让我们来快速了解一下Java语言。

## 第八章 应用程序资源

现在让我们来研究一下项目中可能用到的资源类型，包括布局、图片以及数据值。

## 第九章 Manifest文件

Manifest文件非常重要，它指定应用程序包、提供应用组件的形式化描述，此外还负责声明权限、必要的API级别以及链接库等。

## 第十章 应用程序数据

Android有五种方法存储数据，让我们来看看如何来操作这些数据。

## 第十一章 虚拟与物理设备

接下来，我们将一同探索如何在物理及虚拟设备上运行自己的应用程序并与之互动。

## 第十二章 运行与调试

当我们开始着手创建Android应用程序时，需要关注的重点在于运行应用程序并将信息记录到控制台以监控应用的运行活动。

## 第十三章 Activity与生命周期

当大家开始学习如何为Android平台开发应用程序时，Activity当中所涉及的大量状态与回调方法可能会成为很多难题乃至混乱的根源。

## 第十四章 Android组件详解

Android应用程序当中包含四大组件：Activity、Service、Content Provider以及Broadcast Receiver。让我们来看看它们都是什么。

## 第十五章 示例项目

Android SDK示例项目中的应用能够执行种种功能，例如各类用户界面元素、数据管理、交互等，值得大家探索一番。

## 第十六章 应用程序发布

要通过Google Play商店进行应用程序发布，我们需要注意一些必要条件，建议大家认真了解这些内容，并尽可能严格贯彻。

## 第十七章 下一步学习方向

在今天的文章中，我们将把全部注意力集中在可资选择的未来学习对象上。

## 第十八章 知识测试

前面我们已经了解了为Android平台创建应用程序过程中需要涉及的各种基本概念及知识要点。最后请大家接受一份结业测试、看看自己是否掌握了前面提到的各项知识。

# 第一章 环境需求

这是我们系列教程的第一篇，让我们来安装Android的开发环境并且把Android SDK运行起来！

## 介绍

欢迎来到Android SDK入门指南系列文章，如果你想开始开发Android App，这个系列将从头开始教你所须的技能。我们假定你没有任何编程技能，当然，有经验当然更好。

我们将从安装Android开发环境开始，到开发出一个功能完整的应用。Android开发需要一些不同的技能，但如果你每次集中在一个之上，你将拥有开发所需的牢固基础。当开始教程之后，我们将直接进入开发流程，你将直面具体的结果！

## 操作系统需求

Android开发工具能在大多数操作系统上运行，包括Windows XP、Vista、7、8，以及Linux发行版Ubuntu 8.0.4以上的版本，如果是苹果电脑，你需要将系统升级到OS X 10.5.8以上。

如果你使用Linux 64位版本，它需要能运行32位程序。

## 下载并安装JDK6

为了开发Android应用，你需要安装Java开发套件（JDK），你可以下载JDK 6或者以上的版本。

在Oracle Java下载页面，选择Java SE，然后选择合适你操作系统的版本，你也许需要注册一个Oracle账号，不过这是免费的。

## 下载ADT工具包

第一步：

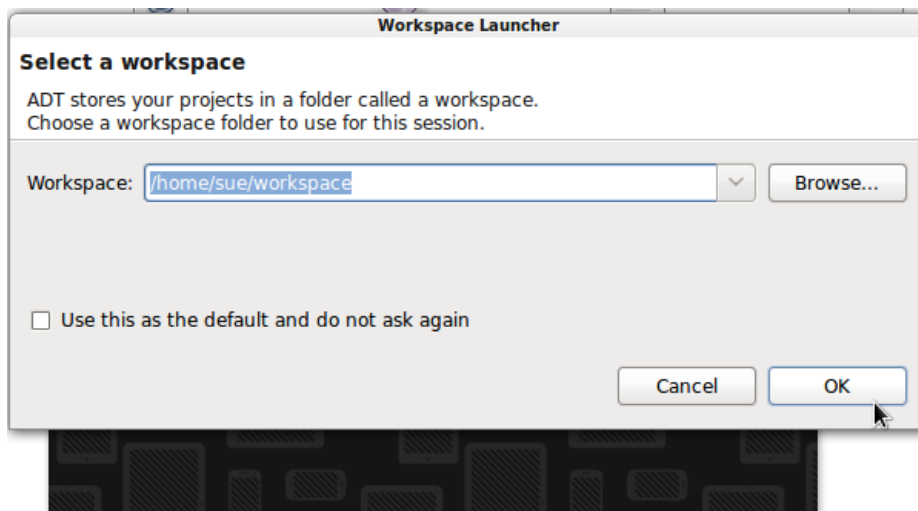
ADT工具包提供了所有Android开发所需要的工具，包括SDK、IDE、ADT插件以

及很多其他的工具。你可以前往Android开发者官网下载。

## 第二步：

如果你选择了Eclipse作为IDE，你需要安装ADT插件。

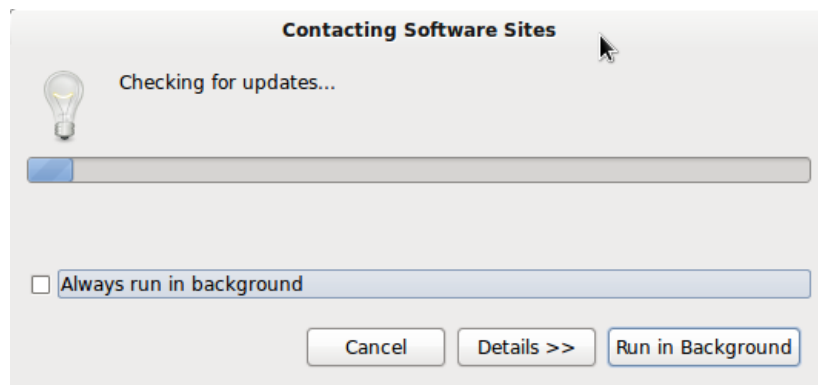
安装完Eclipse后双击打开，它会提示你选择一个工作空间，你的Android应用代码将会储存在这里。在大多数情况下你可以通过Eclipse来操作Android文件，但如果你需要直接与Android文件打交道，你需要记住选择的目录。



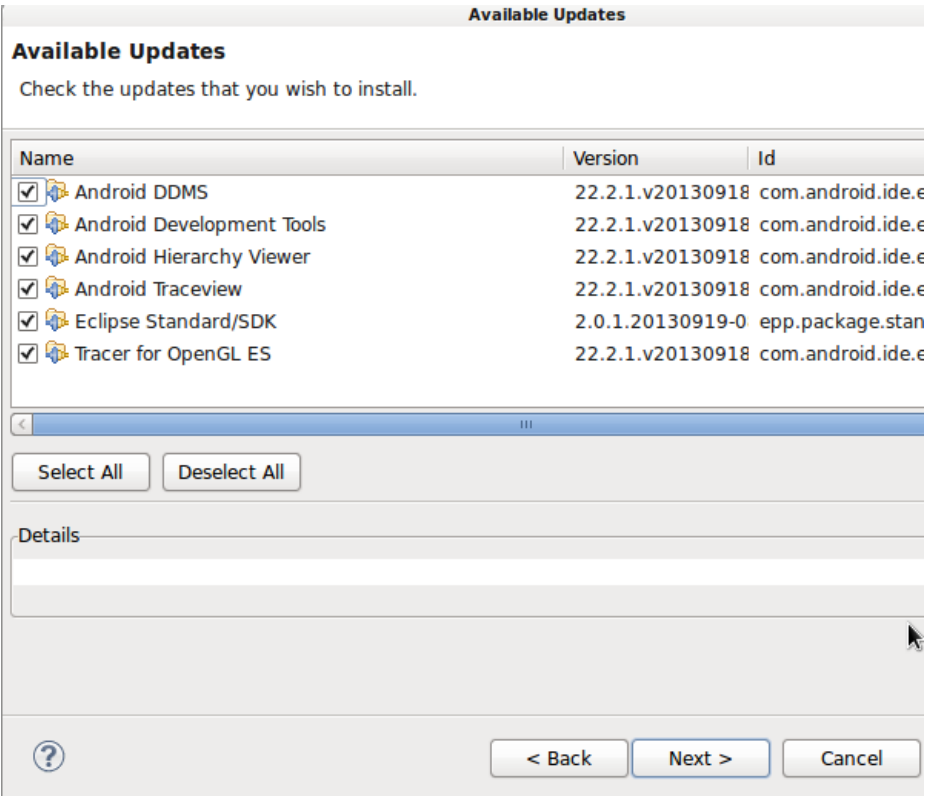
## 下载更新

### 第一步：

你有必要使你的Eclipse与ADT工具保持最新版。打开Eclipse，选择Help - Check for Updates。更新也许会花一些时间，请耐心等待。

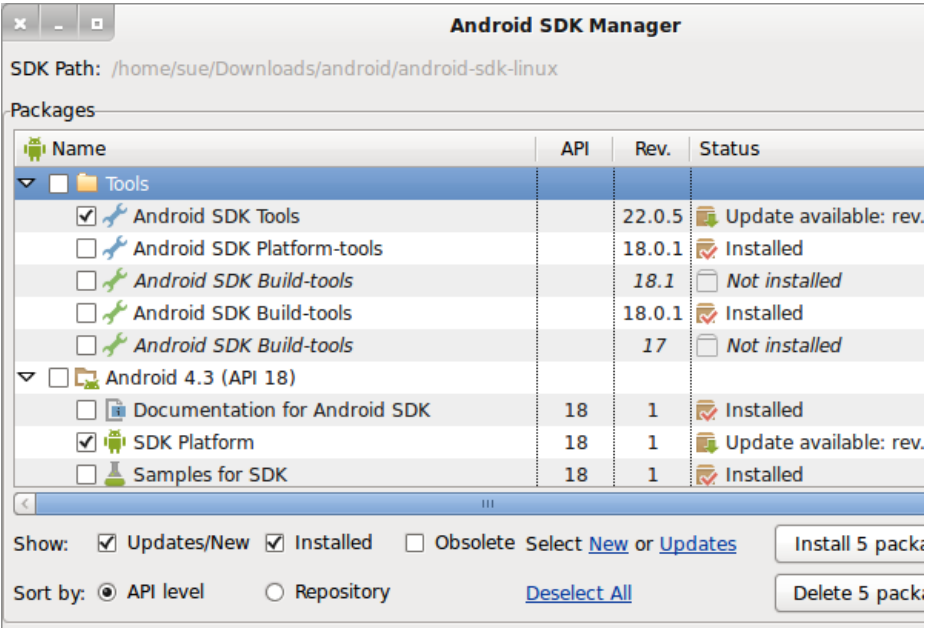


如果有更新，会出现下面的画面，你可以选择需要的更新，点击“下一步”，也许会有一些授权界面出现，选择接受。

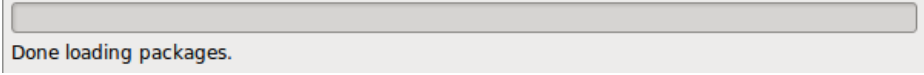


第二步：

打开Android SDK Manager，更新Android SDK和SDK工具。你可以只选择需要的SDK进行更新，进行本教程的学习，安装一个版本的SDK，以及一些工具就够了。







Done loading packages.

### 第三步：

为了保持更新，你可以将ADT工具包更新的URL添加到Eclipse上，让它帮你检查。选择菜单栏Help - Install New Software，如果下拉菜单不包含下面的URL，那么输入后点击添加。

`https://dl-ssl.google.com/android/eclipse/`

## 总结

上面就是Android开发环境的安装，下一篇我们将熟悉我们的IDE：Eclipse 与 Android Studio。

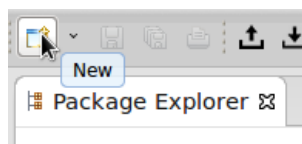
## 第二章 IDE: Eclipse速览

本文我们将对Android开发最常使用的集成开发套件（IDE）Eclipse与专用的Android开发IDE Android Studio做一个亲密接触。让我们先从Eclipse开始。

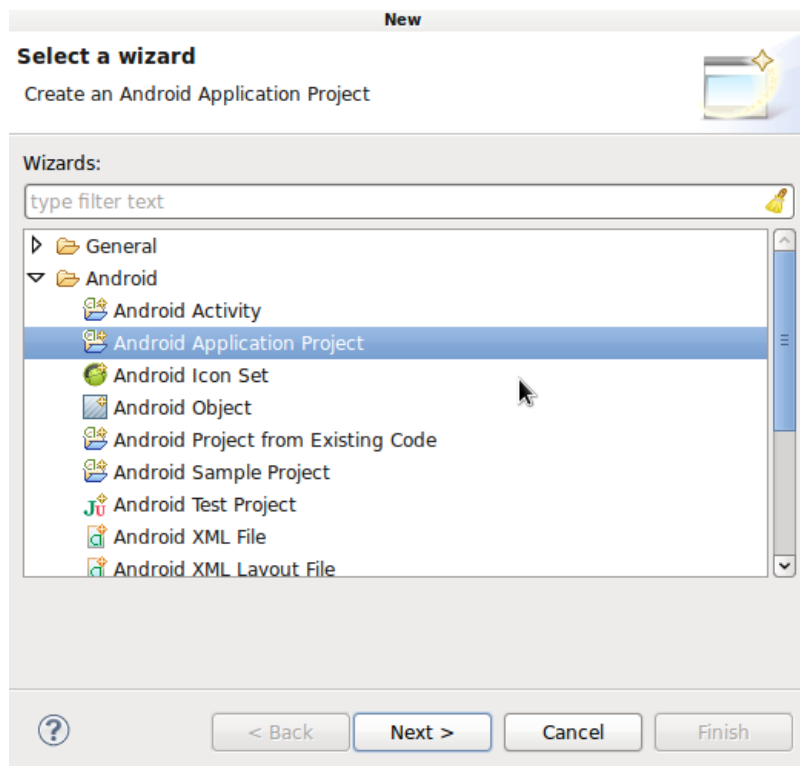
如何来看一个IDE好不好呢？当然是实际用来它来编写一段代码了。我们现在来创建一个Android应用试试。由于我们还没有正式开始Android开发的学习，所以这里的一些细节我们不用过多关注，我们只关心IDE的表现。

### 创建Android工程

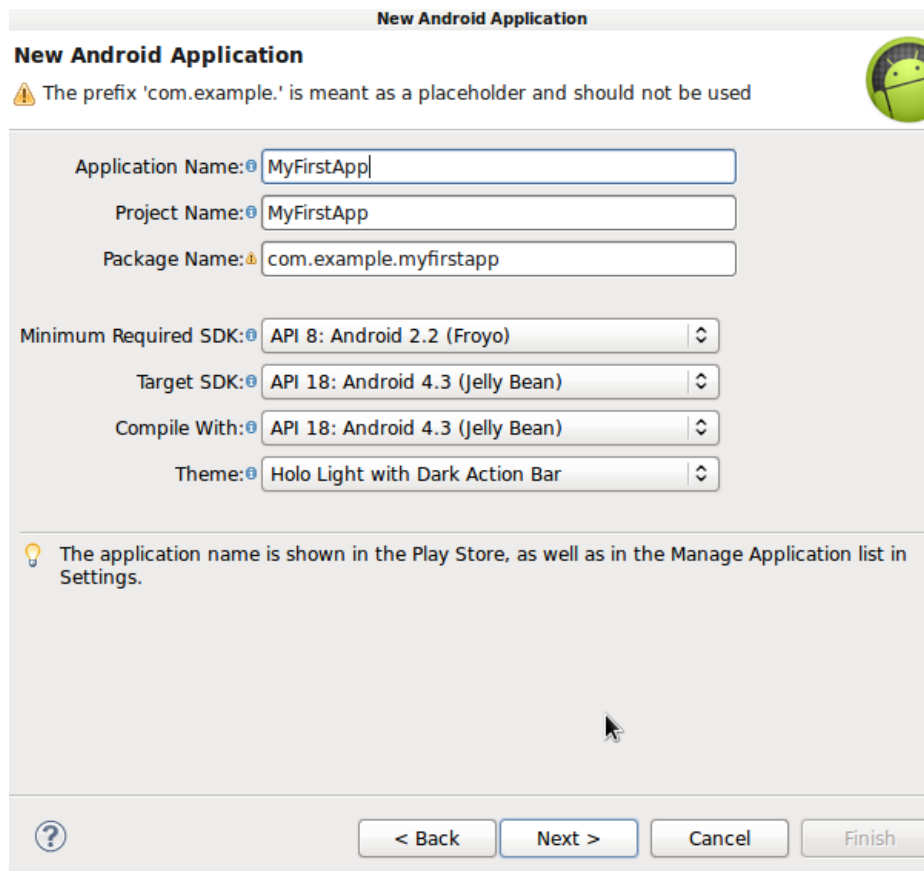
打开Eclipse，点击New，出现一个创建向导。



在向导中展开Android目录，选择Android Application Project，这是创建Android app所必需的一步，选择下一步。



在New Android Application界面，这里会有大量的设置，你点击文本框下面会出现相关提示，你可以按照下图填好。



**New Android Application**

⚠ The prefix 'com.example.' is meant as a placeholder and should not be used

Application Name:

Project Name:

Package Name:

Minimum Required SDK:

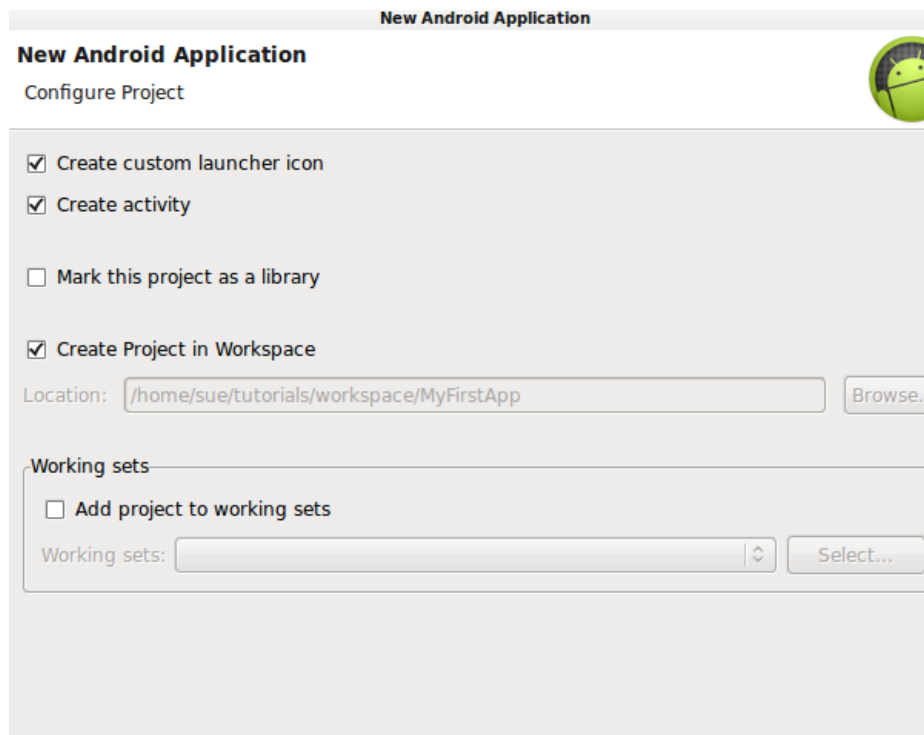
Target SDK:

Compile With:

Theme:

💡 The application name is shown in the Play Store, as well as in the Manage Application list in Settings.

填好之后点击下一步会出现又一个设置界面，一般默认选项即可。



**New Android Application**

Configure Project

☒ Create custom launcher icon

☒ Create activity

☐ Mark this project as a library

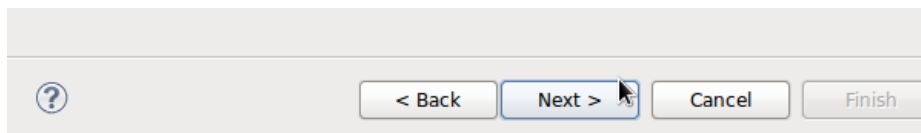
☒ Create Project in Workspace

Location:

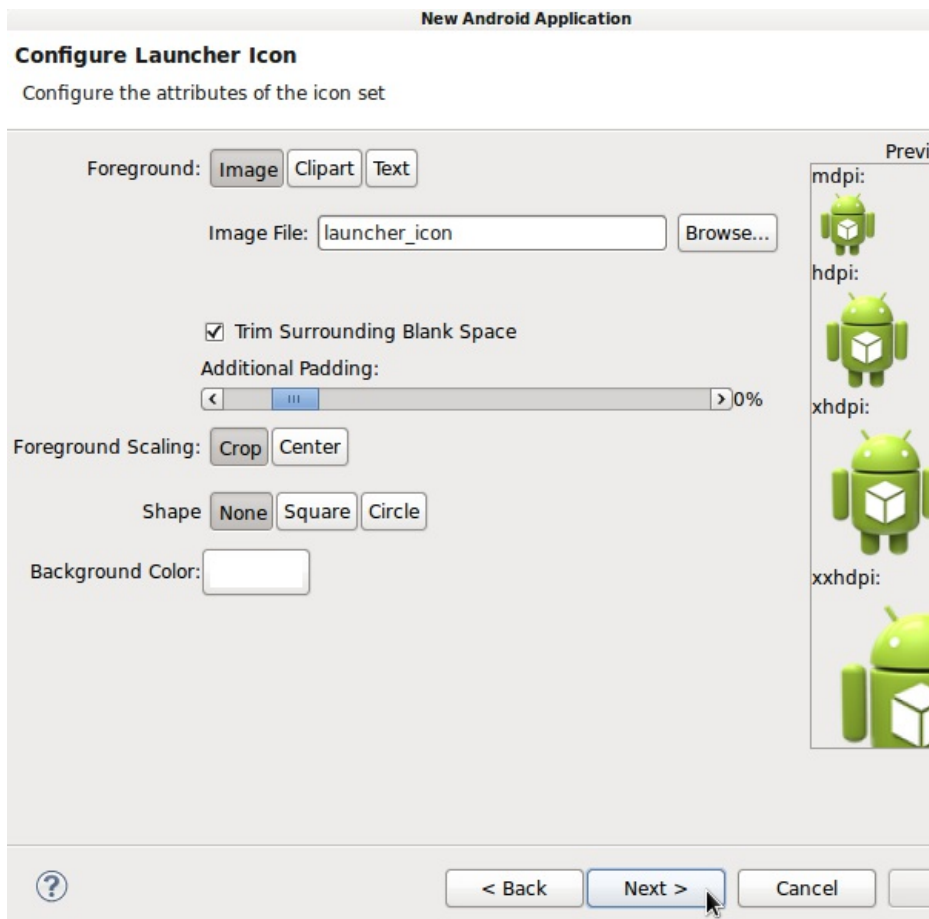
Working sets

☐ Add project to working sets

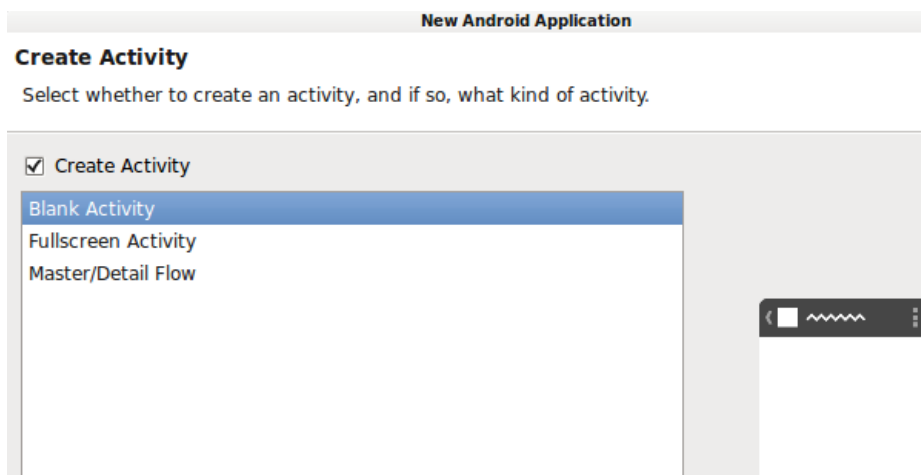
Working sets:

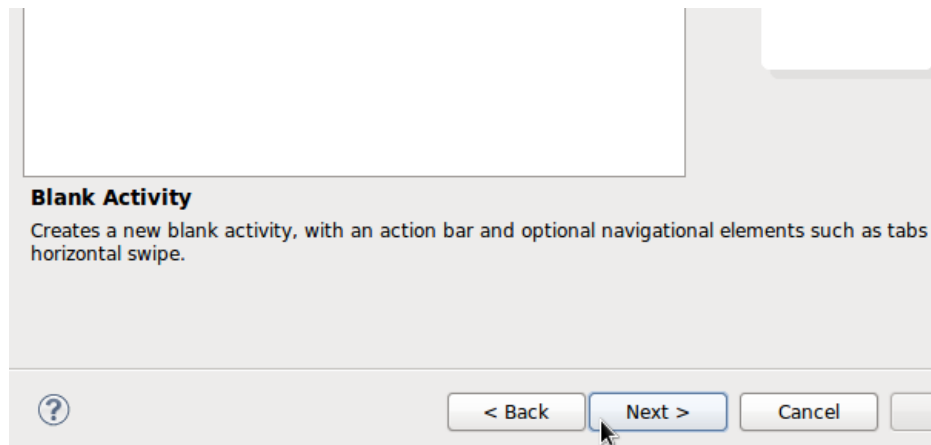


下面你将会看到应用图标的设置，不过现在使用默认的即可，即一个绿色的小机器人。

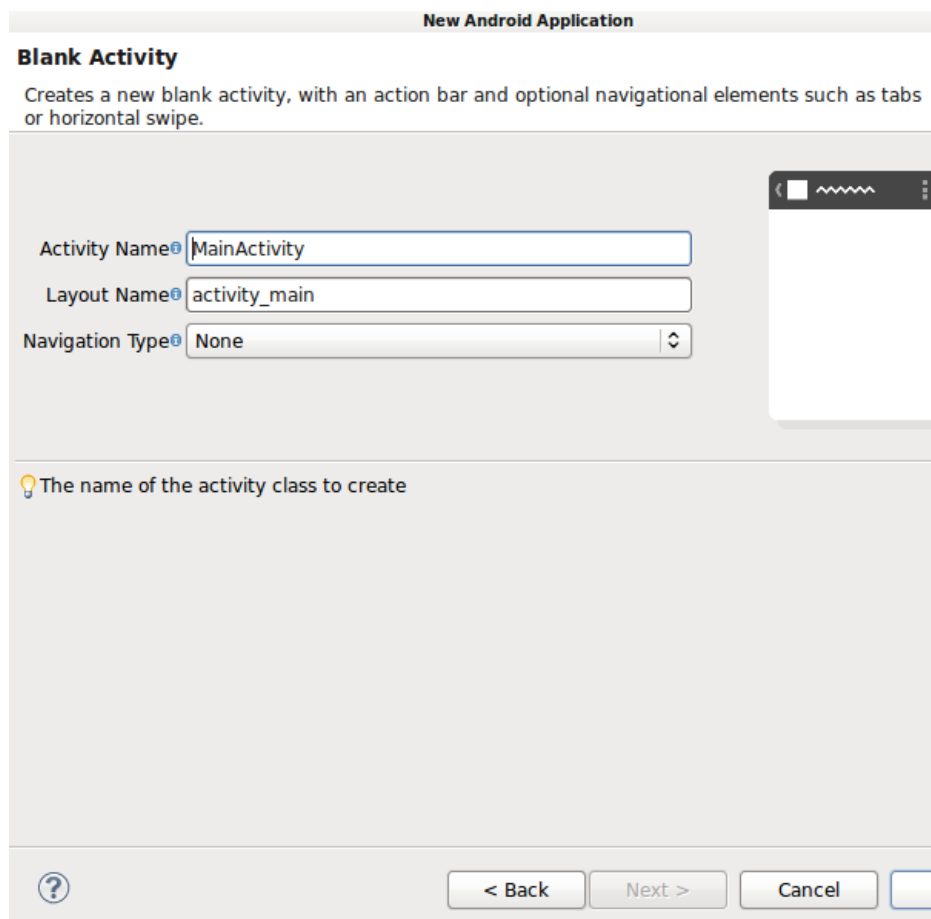


然后下面会让你创建一个Activity，选择默认即可。





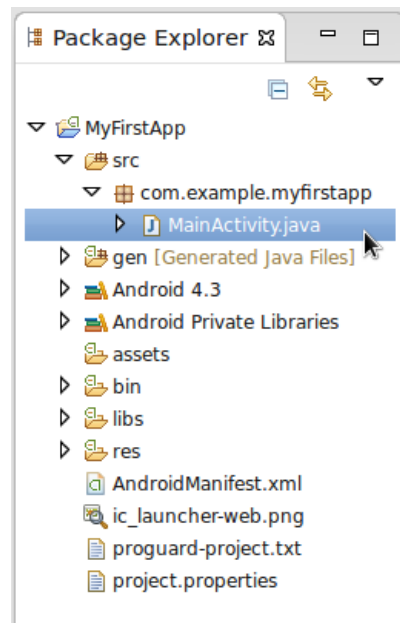
然后会让你确认，这是最后一步，点击“Finish”，你就创建了一个Android工程。



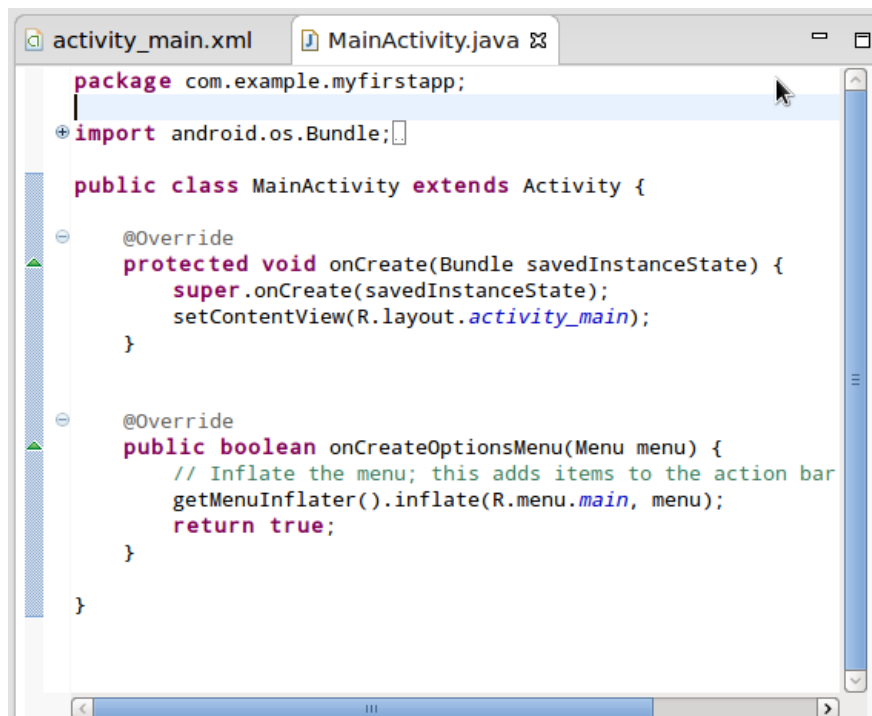
## 使用Eclipse视图

现在我们已经拥有一个Android工程，我们可以看看Eclipse在开发Android应用中是如何表现的。

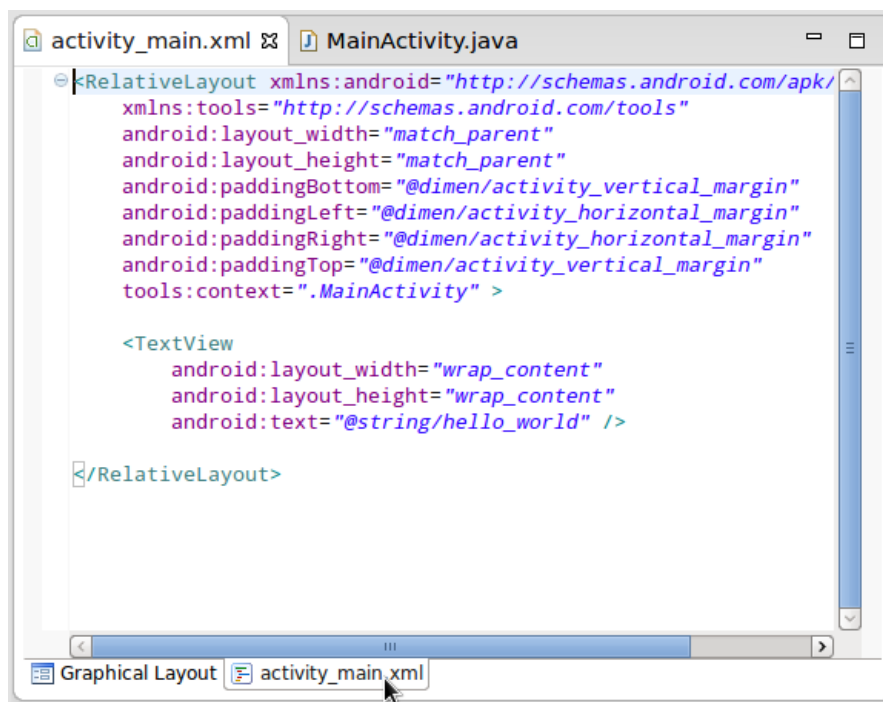
Eclipse的左侧是包管理器（Package Explorer），这里包含着你的工程文件。这里你最应该关注的是src和res目录，以后开发中会用到。



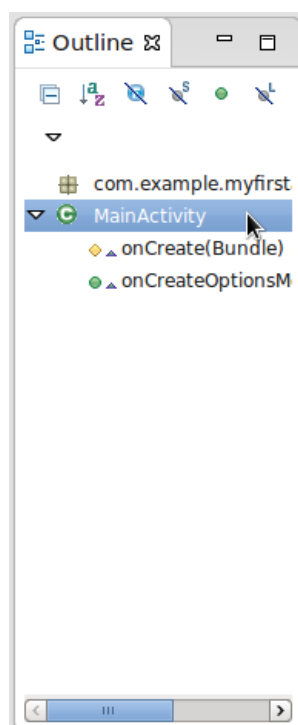
中间是编辑区域，上面显示你打开的所有文件，比如你刚才创建的Activity，你主要在这里进行编程。



Eclipse还会自动的打开布局文件，即“activity\_main.xml”，这里你可以选择可视化的布局界面，或者保持代码界面。

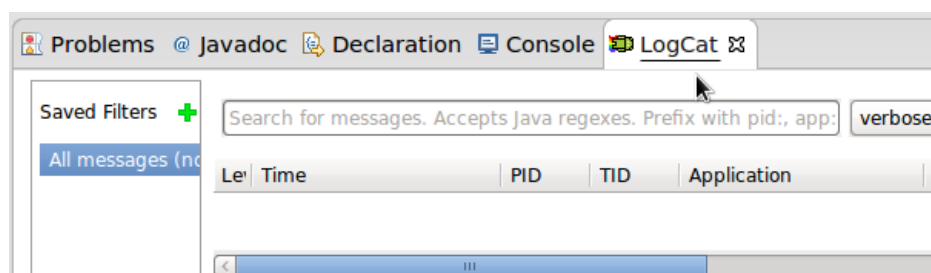
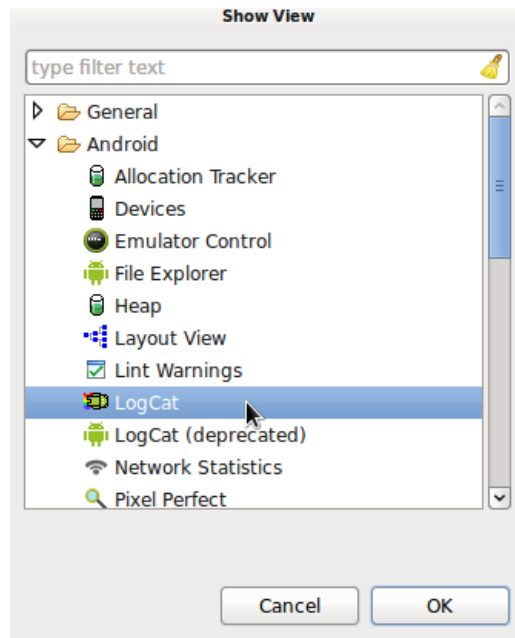


在编辑区域的右边，是文件的轮廓（outline）视图，它会显示文件里的结构，使用它可以快速的在打开的文件里导航。



在Eclipse的底部是另一些视图，包括Problems、Javadoc以及声明窗口。开发者通常使用这些区域来输出错误信息和调试结果。Android提供了非常好的log输出工具logcat，让我们来打开它。

选择菜单Window - Show View - Other，在弹出框里，展开Android，选择Logcat并点击OK。这个工具可以使应用程序在运行时输出信息，这个将帮助我们开发。

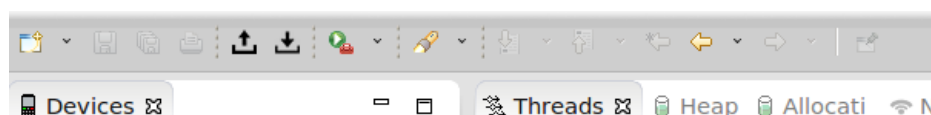


## 使用Eclipse透视图（Perspective）

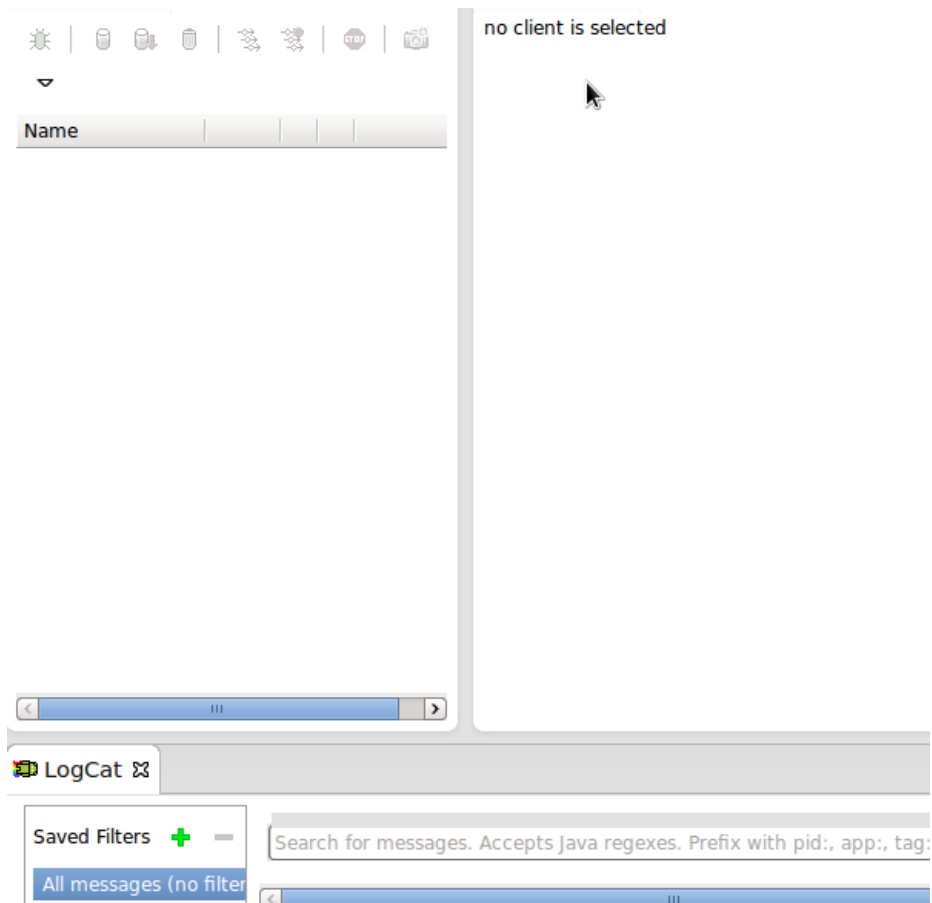
Eclipse的透视图是非常有用的工具，它能记忆你的Eclipse界面的视图和布局，提供某一情景下的最佳开发环境。

在Android开发中用的最多的是Java透视图，但当调试Android应用的时候你会发现DDMS透视图也非常有用。下面让我们来打开它。

选择菜单Window - Open Perspective，然后在清单中选择DDMS。







Eclipse会记忆你最近打开的透视图，通过点击这些快捷按钮，你可以迅速的在这些透视图切换。

## 总结

现在我们已经熟悉了Eclipse，将Eclipse的全貌记在心里将有助于你开始开发一个Android应用。下一章我们将会讲Android Studio。

## 第三章 IDE: Android Studio速览

Android Studio是Google官方提供的IDE，它是基于IntelliJ IDEA开发而来，用来替代Eclipse。不过目前它还属于早期版本，目前的版本是0.4.2，每个3个月发布一个版本，最近的版本由2014年1月发布。

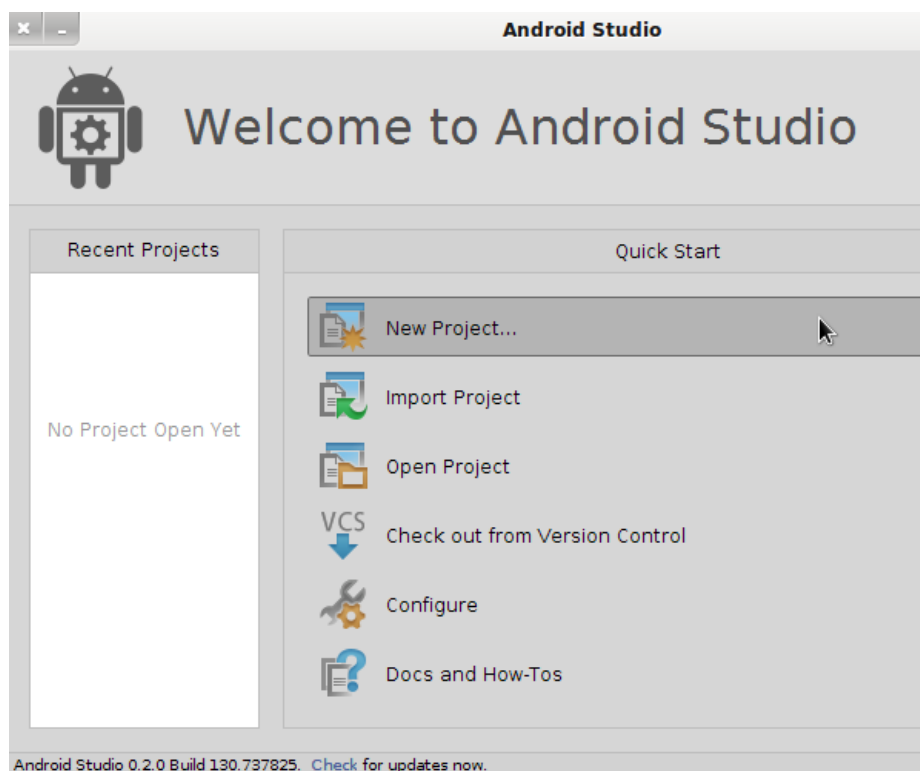
Android Studio包括了所有开发Android app所需要的工具，但是它并不成熟，所以如果需要稳定还是推荐使用Eclipse。不过Android Studio为我们带来了许多新的特性，让我们来看看这个令人期待的未来之星。

### 安装

你可以在Android开发者官网上下载对应的版本。

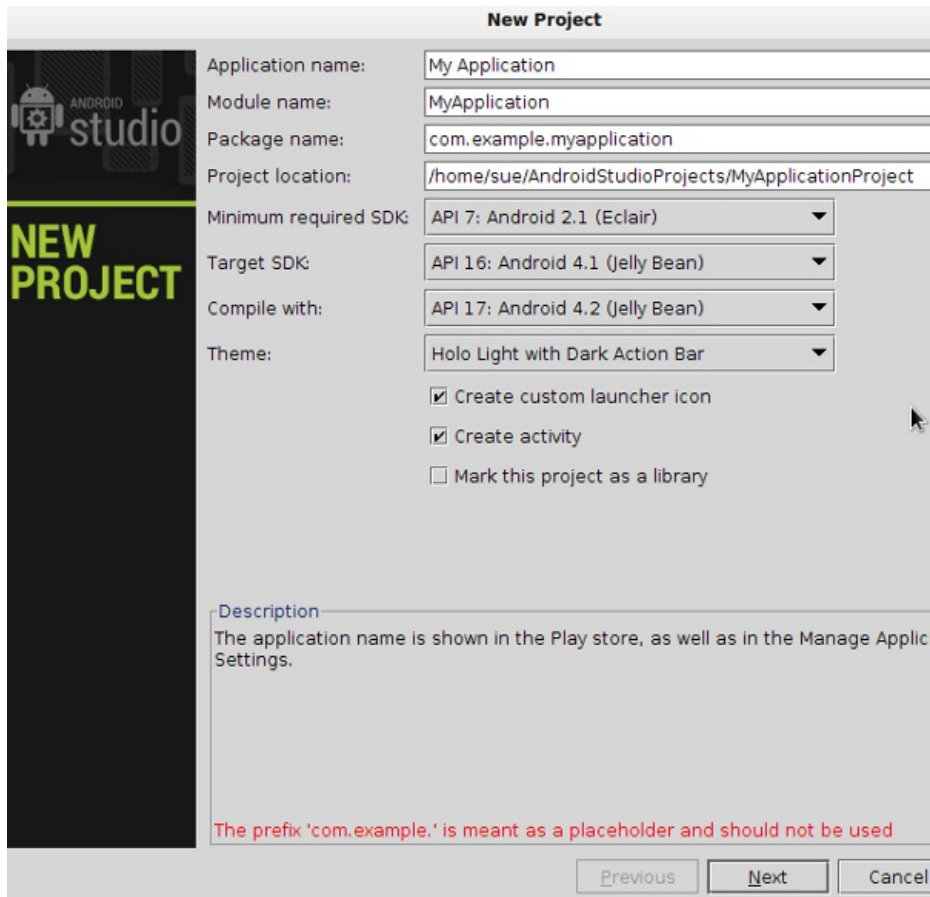
### 创建工程

打开Android Studio（下面缩写为AS），在欢迎界面有一些选项，选择 New Project。



你可以看到创建界面和Eclipse很相似，你可以全部选择默认，然后点击下一

步。

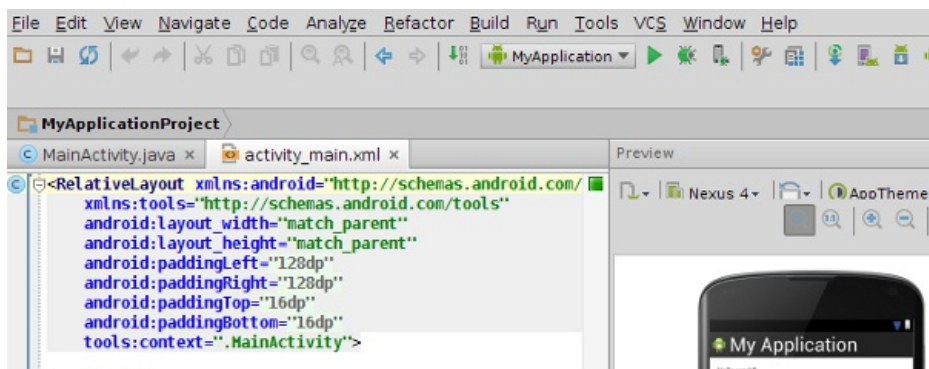


一路下一步，这些和Eclipse很类似。

创建向导结束后，AS会创建一个AndroidStudioProjects文件夹，所有的Android工程文件都在这里。

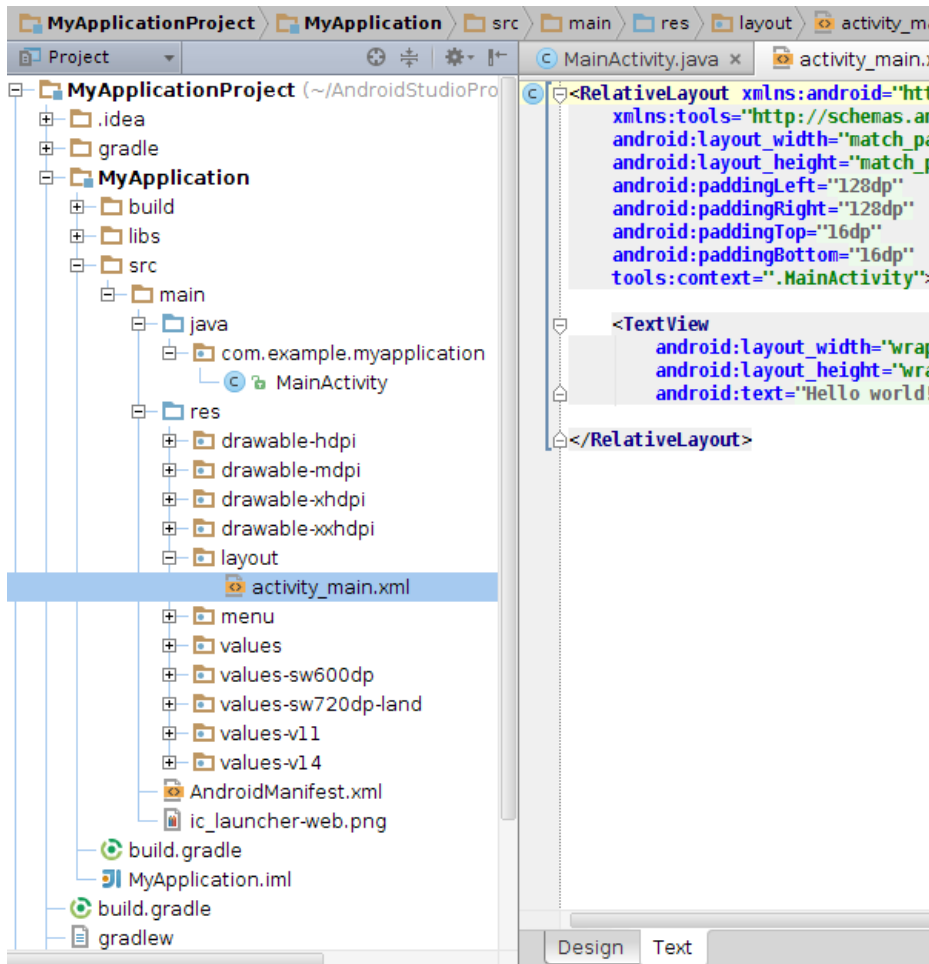
## 开发环境

当创建工程完成后，AS会默认打开Activity以待编辑，并且同时打开一个虚拟设备界面，将应用显示在上面。

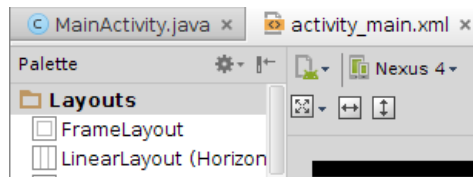


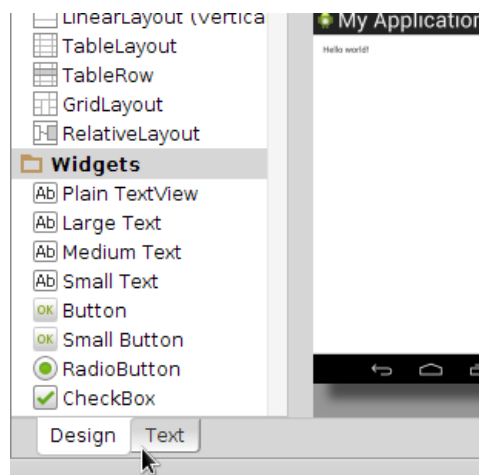


在Eclipse中我们会会有一个包浏览器（Package Explorer），在AS也有类似的界面，只是不是默认显示，双击项目名即可打开。



在布局界面你可以同时看到组件树以及可视化界面。

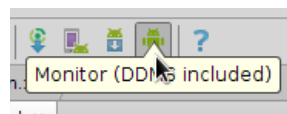




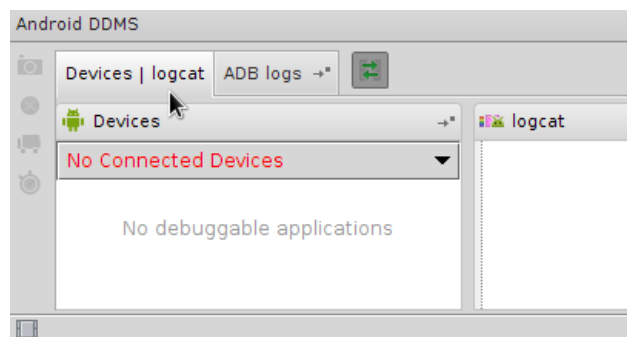
AS和Eclipse的一个不同点是，AS里你需要操作的文件基本都在src目录下面，因为AS是基于Gradle的，不过这并不影响你在两种IDE中切换使用。

## 工程的交互

AS的一个优点是它很多的工具都有按钮可以直接使用，如AVD Manager、SDK Manager、调试工具等。



和Eclipse一样，AS也提供了很多视图以供和工程进行交互。其中你会发现一个叫“Android”的视图非常有用，它提供了虚拟和真实的设备的信息，还包括Logcat输出的信息。



## 总结

本文简单介绍了Android Studio的界面，但它包括更多好用的特性，能让开发变得更加简单。这是因为它专门为开发Android应用而设计，不像Eclipse需要支持更多的开发场景。当你用惯了Eclipse之后，也许你会尝试着使用一下Android Studio。

现在的开发环境已经准备就绪了，在下一个章节里我们将会来看一下Android app的结构。

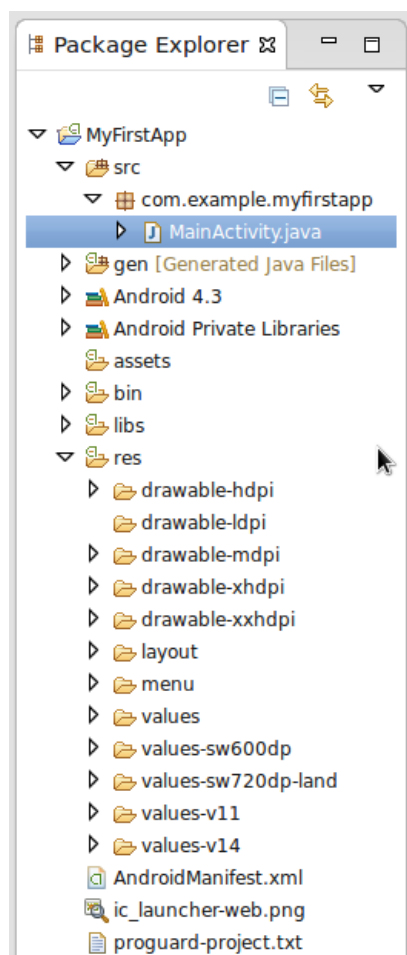
# 第四章 应用程序结构

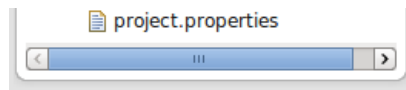
本教程将主要以探索与了解为主要目的，但后续的系列文章则将进一步带大家深入学习如何创建用户界面、响应用户交互操作以及利用Java编排应用逻辑。我们将专注于大家刚刚开始接触Android开发时最常遇到的项目内容，但也会同时涉及一部分已经存在于应用结构当中的其它一些元素。在今天的文章中，我们不会对这些额外元素进行深入探讨。总而言之，了解关于Android应用的基础创建知识，这就是我们今天要完成的任务。

## 1. 源文件

### 第一步

打开Eclipse并在Package Explorer当中查找我们已经创建完成的项目。在“src”文件夹里，大家应该会看到设置项目所命名的项目包。包中应该包含我们的Activity类文件，这也是要在编辑器中打开的内容。源文件夹保存着我们在开发Android应用时所要用到的全部Java文件。





每当我们创建一个项目时，都会创建一个用于容纳各Java类文件的包。一款应用程序可能拥有不止一个包，而且每个包当中也可能容纳着多个类文件。这些类文件中的处理代码能够将我们的应用呈现给用户、响应用户的交互操作并执行任何的必要处理。从实质上看，类文件是在根据面向对象概念模型划分与应用程序相关的代码。

我们将在后续文章中进一步讨论关于Java的概念以及对应实践。在今天的教程内，大家只需理解一个Java应用会将各类处理任务拆分成一定数量的对象。每个对象都由一个类声明来定义，这在应用程序中通常是一个独立的文件，不过也可以被嵌套在其它类文件当中。一个对象基本上就是一大段代码，其中承载着与应用程序相关的某项功能的一部分。类文件中的代码能够引用应用程序中的其它类或者应用程序中的其它包。

在大家着手进行应用程序开发时，首先需要向源文件夹中的包中添加Java类。一个向用户提供用户界面的典型Android应用将拥有至少一个Activity文件，应用中的不同屏幕显示内容还要用到更多Activity类。其它一些类型的应用，例如工具程序或者服务，则采用不同的结构。大家最好首先关注Activity UI这种类型的应用程序，并在熟练掌握之后再接触其它应用类型。

## 第二步

现在查看新应用中的Activity类文件。我们会在本系列教程的后续文章中进一步探讨Activity代码，因此目前大家不用太过关注细节。今天我们主要面向应用中的主Activity，它会在应用启动后同时开始生效。大家的应用也可能会启动其它一些用于用户交互的Activity。在我们创建自己的项目时，Eclipse会对应用进行设置并将主Activity作为主类——它在项目清单当中也将被作为主Activity进行显示，我们稍后会看到。

在主Activity类当中，大家会看到onCreate方法，其中包含的代码将在Activity被创建——也就是应用程序启动时开始执行。在该方法中，大家会看到以下代码行：

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



在我们启动项目之后，这一行的内容用于指定我们所创建的布局文件，告诉Android将其作为内容视图。这意味着无论布局文件中包含什么样的内容，都将在该Activity显示在屏幕上时呈现给用户。

我们将在稍后进一步探讨相关话题，目前暂时需要将注意力集中在“`R.layout.activity_main`”语法上。这就是我们的Java代码引用应用程序资源的方式。我们将利用类似的语法通过资源的ID值对其进行引用，例如图片及数据值等资源也可以通过这种方式实现引用。其中的“R”代表应用资源，后面的部分则用于指定保存在“`res/layout`”目录下的条目类型——在这里就是布局。这些资源最终要根据其名称进行识别——对于示例中的布局，使用的就是文件名。由此推断，我们要使用的语法就成了“`R.type.name`”。在我们开始编程之后，各位就会开始使用该语法。

在本系列的后续文章中，我们将向Activity类文件中添加代码以实现用户交互。现在打开应用中的“`res`”文件夹，大家会在其中找到多个子文件夹。这些文件夹是由Eclipse与ADT在我们启用新Android项目后默认创建而成的，不过我们可能还需要为不同类型的资源添加其它一些目录。

## 2. 布局资源

正如我们已经看到，项目创建后所生成的布局文件会保存在“`res/layout`”文件夹中。如果某款应用拥有多个Activity屏幕，那么一般会为每个屏幕保留一个独立的布局文件。大家可能还会将布局文件用于个别UI组件。当大家为Activity创建类文件时，需要如上所述利用`setContentView`进行布局设置。除此之外，大家也可以通过Java代码进行布局设置——这算是种备选方案。在我们的示例中，布局设置是在应用执行时动态生成的。不过利用XML的优势在于，我们可以在界面设计工作中直观感受布局方案的视觉效果。

在应用程序的主布局文件当中（现在应该已经用编辑器打开了），大家会看到XML结构。如果各位之前没有接触过XML也不必担心，我们会在后续文章中进一步讨论这些基础知识。就目前来说，大家只需了解：XML是一种标记语言，类似于HTML——如果之前接触过Web开发的话。XML文件利用树状结构作为数据模型。通常来说，一个布局文件拥有一个根布局元素，并将其作为特定布局类型模型——其中所包含的用于UI条目的子元素则包括按钮、图片及文本等。

## 3. 可绘制资源

大家在资源目录下应该会看到多个在名称中包含“drawable”字样的文件夹，这些文件夹用于保存应用程序所使用的图片文件。这些图片文件可以是我们在Eclipse之外所准备的数字图片文件，格式包括PNG或者JPEG等。或者，大家也可以通过XML代码来描述形状、颜色以及外观，从而定义特定可绘制资源。一旦我们在drawable文件夹中创建了文件，就可以在应用布局文件或者Java代码中进行引用。这样一来，之前准备好的视觉元素就能用于应用UI了。

资源目录中会保留针对每一种尺寸的drawable文件夹。这些尺寸是各类运行Android系统的设备在像素密度方面的通用型分类依据。具体类别分为低、中、高、超高与超超高密度四种。只需从对应类型中作出选择，我们就可以在对应尺寸的帮助下轻松简化多屏幕密度的支持过程。这意味着当我们在项目中包含图片文件时，可以将其放置在不同的文件夹当中，并通过裁剪提供满足各种密度方案的版本。

## 4. 数据资源

在“res”目录中，我们会看到一些标题中带有“values”字样的文件夹。这些文件夹用于容纳大家希望在应用程序中所使用的数据值。这些值可以包含文本字符串以及数字。包含XML文件的值文件夹会列出其中的一项或者多项值。每份列表都包含一个名称以及内容中的值。应用中的其它文件，例如Java类或者布局文件，能够通过这些名称为引用这些值。在典型用例中，我们能够需要通过这些保存在文本字符串的值在UI元素当中显示内容——例如按钮。

应用程序中的不同值文件，允许大家针对特定屏幕尺寸及API级别对值进行修改。如果同样的值足以应对多种设备，则可以被直接保存在“Values”文件夹内。

## 5. Manifest文件

在查看应用程序中的主文件夹时，大家一定会发现项目的Manifest文件。通过双击即可利用编辑器将其打开。接下来，我们会看到一个显示其内容的视图。点击编辑器窗口底部的“AndroidManifest.xml”标签来查看其XML代码。这个文件将应用程序的各个方面定义成统一整体。Eclipse与ADT会在我们创建应用的同时，在清单中创建特定元素，具体创建方式取决于大家在项目创建过程中的设置。大家可以手动向清单中添加其它元素，例如添加其它Activity。



我们将运行其中的一部分主元素，旨在理解Manifest文件的作用，不过还有其它多种元素可以被包含其中。在Manifest文件中所列举的新应用项目元素当中，我们将看到uses-sdk元素，我们利用它表示最小及目标API级别。

Application元素中包含指向启动机制与应用程序名称的属性。在application元素中还存在着一个activity元素，会在应用程序开始运行时通过intent-filter元素作为主Activity启动。当我们向应用中添加新的Activity时，则会为每个相关元素添加新的activity元素。

大家可能还需要向Manifest中添加其它元素，其中包括uses-permission元素，用于罗列应用所要求的权限——用户会在安装应用之前观看到该列表。权限中包含多种操作条目，例如通过互联网获取数据、写入存储或者访问设备上的其它功能——如相机。清单还会列举应用程序所能支持的设备类型以及其它一些应用程序组件（例如后台服务）。

## 6. 其它文件

讲到这里，我们已经谈到了大家需要了解的Android应用程序项目结构中的各大主要方面。随着对Android开发的学习，大家将在今后经常与这些内容打交道。通过Eclipse，我们还会看到项目中包含的其它一些文件及目录，不过就目前来说基本都可以直接忽略。

正如在前面看到的，大家可以利用“R.”语法实现资源引用。Eclipse以及管

理系统的ADT都会引用应用中来自Java的资源。当大家在项目中对这些资源进行添加或者编辑时，Eclipse会将对应内容写入“R.java”文件，从而帮助我们利用“R.”进行资源引用。当大家开始处理自己的Java文件，会在引用时看到Eclipse弹出的提示信息——这种机制能简化对应用资源的管理工作。

“R.java”文件被保存在“gen”文件夹中。请注意：千万不要尝试直接编辑这个文件，它会在我们编辑项目资源时自动生成。系统会通过为应用中的每项资源分配惟一整数ID的形式管理这一过程。

提示:当大家开始尝试Android应用程序开发时，可能会在使用R时遇到问题。如果Eclipse显示任何与R相关的错误信息，特别是“R无法被解析为一个变量”，则需要检查类文件的起始内容，看看其中是否存在“R”的导入语句，例如“import android.R;”。如果找到了对应内容，特别是在已经将代码复制并粘贴到文件中后，请删除这一导入语句。如果遇到其它与R相关的提示，请确保资源文件当中不存在错误。如果问题仍然存在，尝试利用“Project”，“Clean”清理项目。当一切努力皆告失败时，试着重新启动Eclipse。

## 总结

在本篇文章中，我们了解了关于Android项目结构的基础知识。大家可以再花点时间随意查看项目中的其它文件及文件夹，借此了解项目的整体结构。在接下来的后续教程中，我们将在应用中创建用户界面元素并处理用户交互操作。我们还会探讨关于Java编程的基本特性，借此进一步提升自己对Android开发项目的理解。

# 第五章 用户界面设计

在本篇教程中我们将为应用程序项目添加布局方案，在这方面XML与Eclipse ADT接口将成为工作中的得力助手——不过在后面两节中还会用到一部分Java开发知识。XML与Java在Android平台的开发工作当中可谓无处不在，如果大家对二者还缺乏基本的了解，请尽快想办法补补课。对于刚刚入门的读者朋友来说，本文所介绍的要点将成为各位日后开发工作的重要基础。

## 1. XML基础知识

在我们开始讨论布局之前，先来梳理作为标记语言的XML的基础知识。如果大家对于XML已经很熟悉，可以直接跳过本节。XML是一种用于保存数据值的语言。XML文件在多个领域发挥作用。它们在某些项目中的功能与数据库非常相近，而且通常被作为网页的输出机制。如果大家之前曾经使用过HTML，应该会对XML的基本功能感到熟悉。

在XML中，数据值被保存在元素当中。单一元素通常包含一个开始标记与一个结束标记，如下所示：

```
<product>Onion</product>
```

如大家所见，开始标记与结束标记几乎完全一样，惟一的区别在于结束标记中多了一个“/”符号。在上面的例子中，数据值也就是元素内容，即文本字符串“Onion”。开始标记也可以容纳与数据项目相信的其它属性信息，如下所示：

```
<product type="vegetable">Onion</product>
```

每项属性都有一个名称与一个值，其中值就是引号内的部分。元素中还可以包含其它元素：

```
<section name="food">  
  <product type="vegetable">Onion</product>  
  <product type="fruit">Banana</product>  
</section>
```

在这种结构中，我们将section元素称为主元素、products元素则被称为子元素。两个子元素之间属于“兄弟关系”。在XML文档当中，必须存在一个root元素作为主元素，或者被称为“嵌套”。这就构成了一种tree结构，其中子元素作为自主元素延伸出去的分支。如果某个子元素之下还包含其它子元素，那么它本身同时也具有主元素属性。

大家还会遇到另一种自结束元素，其中开始与结束标记并非独立存在：

```
<order number="12345" customerID="a4d45s"/>
```

其中元素末尾的“/”符号代表结束。

我们在Android平台上所使用的全部资源文件都要用到XML标记，其中包括布局文件、可绘制元素、数据值以及Manifest。

## 2. Android布局

### 第一步

当大家在安装了ADT的Eclipse IDE当中使用XML时，输入过程中显示的相关背景提示能让编码过程变得更轻松一些。在编辑器中打开新应用中的主布局文件，确保XML编辑标签已经被选中，这样我们就能直接对代码进行编辑了。我们首先要处理的是用于主屏幕的布局方案，用户在启动应用之后最先看到的就是它。Eclipse会提供一套基础布局，供我们进行个性化修改：

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

```

    rgin"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>

```

如大家所见，根元素是一项布局元素，在上面的示例中为RelativeLayout。Android当中还提供其它几种布局类型，我们可以将一种布局嵌套到另一种当中。这里的根布局元素拥有几项额外属性且与布局效果密切相关，例如宽度、高度以及边距等等。布局元素当中的TextView允许开发人员显示一条文本字符串。TextView是View的一种，View属于可见及交互性元素，用以构成我们的应用程序UI。因此，应用程序中的每套分屏方案都要选择一种View，并在其中包含一种或者多种布局机制。在Android系统中，这些布局被称为ViewGroup对象，每个 ViewGroup内包含一套或者多套View。

## 第二步

为了专注于一套布局的基础创建工作，我们要把主布局文件中的现有内容全部删掉，这样才能从零开始着手设计。正如我们之前所提到，大家可以利用Java代码创建自己的布局或者View，不过Android上的多种工具允许开发者利用XML设计自己的应用UI——这样各位就可以在创建元素的同时直接观察设计效果了。在某些实例中，大家可能见过单纯通过Java代码创建一些或者全部UI的做法，但现实情况下大部分创建工作还是要由XML完成的。这种做法还能保证应用程序逻辑与显示元素彼此独立。

```

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

    <!-- views go here -->

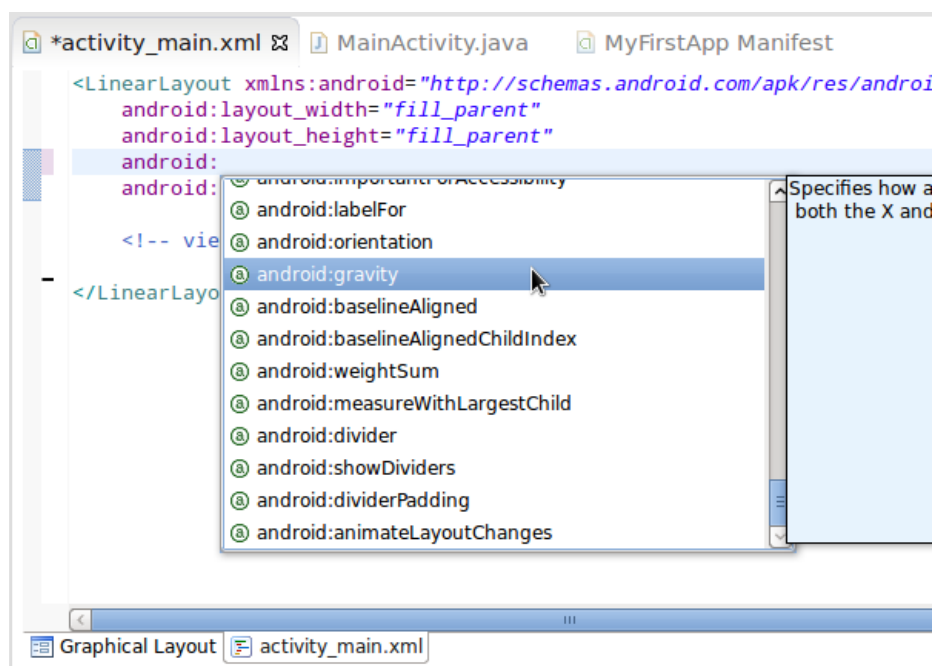
</LinearLayout>

```

LinearLayout会沿横向或者纵向显示我们打算使用的View。在以上示例中显示

方向为垂直，因此每个View都会沿屏幕下方依次排列。如果采取横向布局，那么各个View将由左至右依次排列。如果使用“layout width”与“layout height”两种属性（在Android当中，它们往往被称为布局参数），那么布局会被拉伸至横向与纵向的最大长度。

在“layout height”声明行之后再添加一条新行，通过键入“android:”准备开始输入属性。当大家输入对应内容，Eclipse就会提供一套与该属性相关的列表。大家可以继续输入内容以缩小属性列表，也可以直接在列表中用鼠标进行点选。现在我们选择“android: gravity”属性。



键入“center\_horizontal”作为gravity值，这样其中包含的元素就会以X轴为中心加以显示：

```
android:gravity="center_horizontal"
```

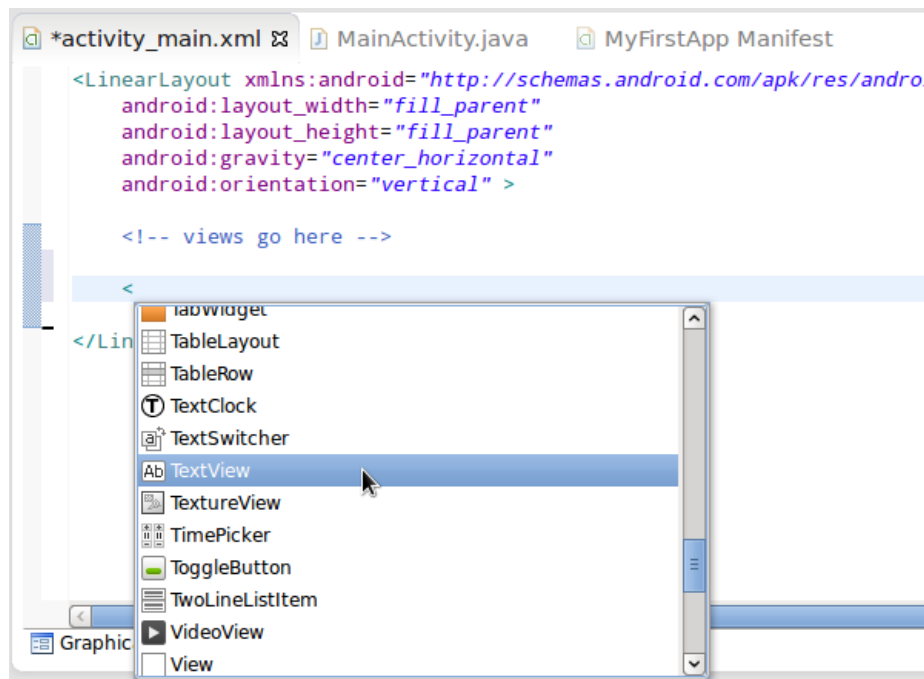
这种方式适用于布局中的一切元素。我们可以添加其它几种额外显示属性，例如填充、边距以及背景等。不过在今天的文章中，我们先从最简单的项目入手。

### 3. 添加View

#### 第一步



正面我们开始向布局中添加View。所谓View，是指UI当中的可见元素。让我们首先添加一些文本内容和一个按钮。进入LinearLayout元素（在开始性结束标记之间），输入“<”之后Eclipse就会提示大家与属性相关的可用元素列表。



在列表中选择TextView。请注意，与大部分View一样，这是一种自结束元素。为TextView设置两种属性，分别为layout width与layout height（键入‘android: ’并选择对应提示）：

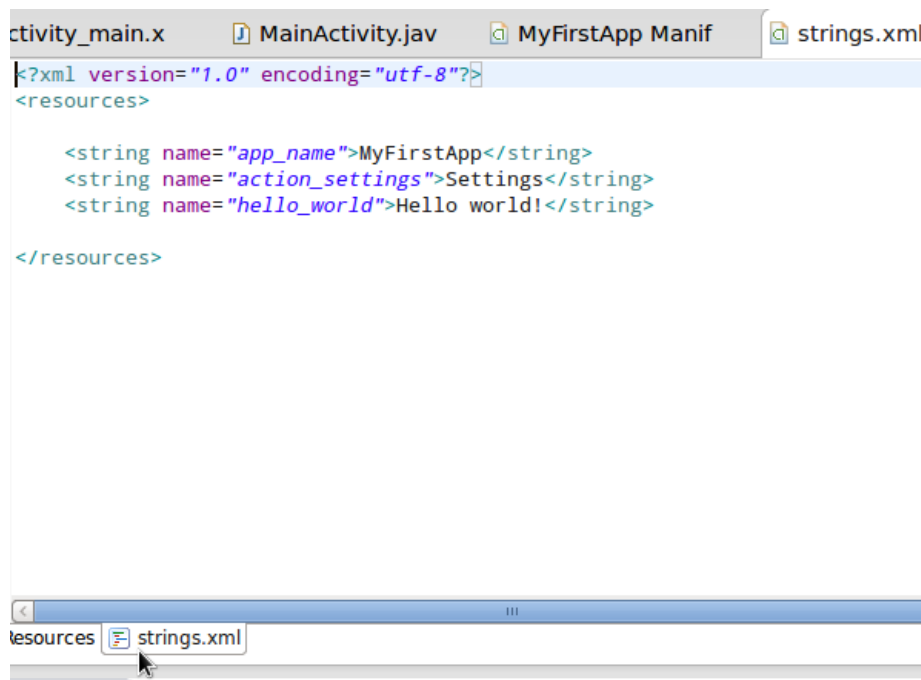
```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

通过“wrap\_content”，我们可以保证View的宽度足以容纳其显示内容——这就避免了像布局那样以填充方式显示元素。现在再为TextView添加另一项属性，这一次通过列举文本字符串实现显示功能：

```
    android:text="Hello there"
```

在保存文件之后，大家会看到Eclipse显示出一条警告消息。如果将鼠标悬停在消息之上，编辑器的边框处将显示该文本——这部分内容也会同时显示在Problem视图当中。警告内容为“Hardcoded string……should use @string resource（硬编码字符串……应使用@string资源）。”系统推荐的做法是将

每一个文本字符串值保存为一项值资源，而不应将其直接包含在布局 XML 当中。尽管从起步阶段来看这样的处理方式既麻烦又毫无意义，但一旦养成良好习惯、大家会在今后的工作中逐渐发现其在大型项目中的价值。通过 Package Explorer 找出 “res/values/strings.xml” 文件并打开，切换到 “strings.xml” 标签并对代码进行编辑。



可以看到，Eclipse已经添加了几条字符串。要另行添加，只需为其设定名称与值：

```
<string name="hello">Hello there</string>
```

这意味着如果大家需要在应用程序UI当中不止一次使用同一条字符串，而且稍后又需要对其进行修改，则只需在一处做出变更即可。保存字符串文件并切换到布局文件。将TextView的 “text” 属性引用到值文件的对应字符串中：

```
android:text="@string/hello"
```

我们通过在字符串名称前加上 “@string” 的方式告知Android工具需要在哪里寻找字符串资源。这样一来，警告信息就不会再出现了。Eclipse通常会在我们编码的过程中发出这些提醒，从而通知我们当前存在的错误或者警示问题。大家可以选择遵循或者忽略警告信息的内容，但对于错误则 必须加以调整，否则应用程序将无法正常工作。

## 第二步

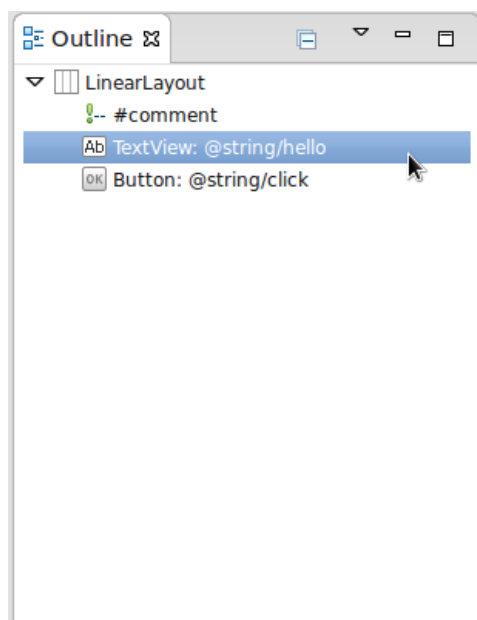
在TextView之后添加一个Button:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/click" />
```

在我们的示例中，Button使用的属性与TextView相同。不过在其它情况下，它可能会使用更多属性，而且一般来说不同视图需要配合不同属性。按钮上显示的是“text”属性值。将这条字符串同之前一样添加到我们的“res/values/strings.xml”文件当中：

```
<string name="click">Click Me!</string>
```

在接下来的教程中，我们将处理按钮的点击效果。切换到布局文件，查看编辑器右侧的Outline视图——它显示的是另一套指向文件元素的界面。双击列出的项目以跳转到对应代码位置。大家也可以展开或者折叠主元素。当布局变得更加复杂时，这种处理方式就变得非常实用。

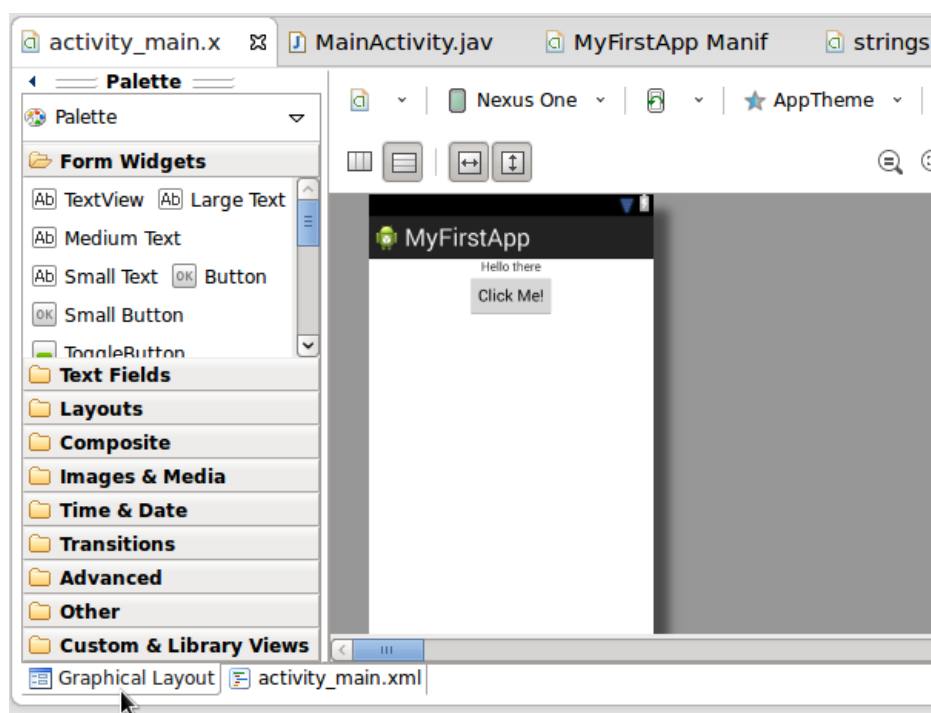


提示：要整理Eclipse编辑中所打开的全部文件，我们只需按下“Ctrl+A”对其进行全选，然后按下“Ctrl+I”即可。

## 4. Graphical Layout

### 第一步

确保我们的布局文件已经正确保存，然后切换到Graphical Layout标签。



大家可以看到自己所设计的布局已经能够直接查看。界面左侧的Palette区域允许我们选择UI组件并将其拖动到布局当中。不过我们应该首先使用XML，直至对基本框架拥有初步概念。XML能帮助我们控制细节设计，所以即使在使用图形化工具的时候，我们也可能需要对XML结果进行编辑。

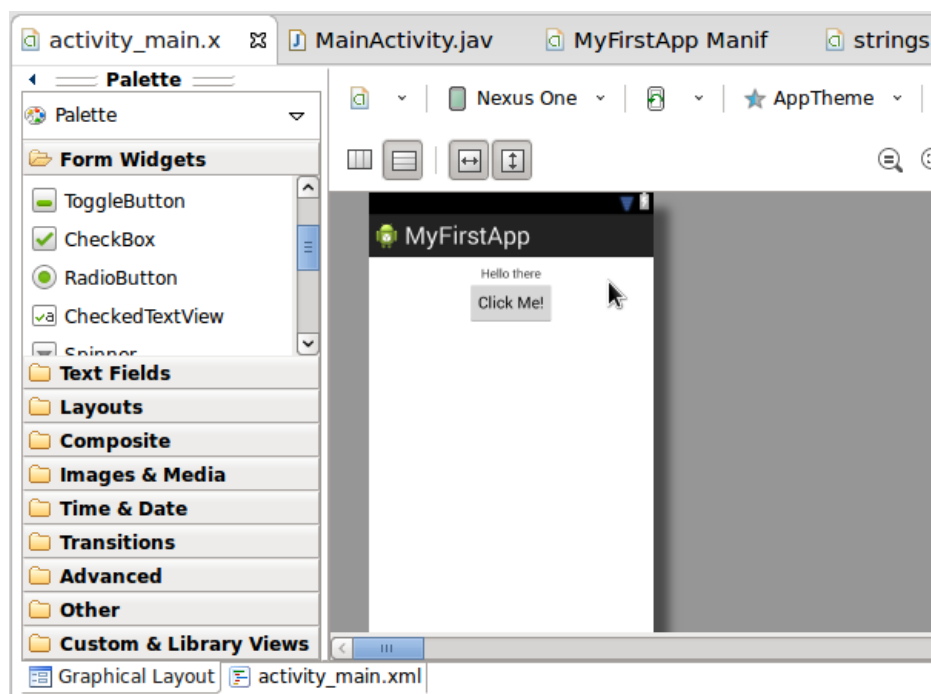
在Graphical Layout视图上方是一套下拉清单，我们可以从中选择用于查看布局效果的设备类型，其中也提供切换显示方向及缩放效果的工具。大家需要在设计布局的过程中不断利用Graphical Layout对效果加以控制。另外，这里也提供其它一些值得尝试的布局元素与设置。

### 第二步

大家可能已经注意到，在这一次的布局设计当中可见元素的显示位置与屏幕上边缘靠得比较近。下面就来解决这个问题。切换到XML编辑标签并向LinearLayout当中添加边距属性：

```
android:layout_margin="10dp"
```

我们使用“dp”来设置像素的独立密度，这样设计就会让像素密度自动与用户设备相匹配。保存文件并切换到Graphical Layout以查看实际效果。



在我们进行布局设计时，Graphical Layout是一款非常实用的参考工具，但只能起到引导的效果。要了解我们的布局在应用程序运行时以怎样的方式显示、又能实现怎样的功能，大家需要将其载入虚拟或者物理设备进行实际验证。我们会在后续文章中进一步讨论这个话题。

## 5. 选项

大家可以在应用程序屏幕中包含各类布局类型以及View，但其基本处理方式都是一致的。我们前面所使用的是LinearLayout，但还有其它多种方案可供选择，其中比较常见的有RelativeLayout、FrameLayout、AbsoluteLayout以及GridLayout。大家可以在LinearLayout Palette当中找到这些类型，建议各位放松心态、在自己的View中任意选择并观察其显示效果。当添加来自Graphical Layout工具的元素时，请务必切换到XML以观察新元素的加入会产生什么样的标记代码。

Android平台针对多种常见需求提供View方案，例如单选按钮、复选框以及文本输入区等。这些方案能够大大节约我们需要手动执行的功能数量；但如果

各位需要使用非自带UI元素，则需要创建一个自定义View类。一般来说，最好是在没有其它选择时再这样处理，毕竟标准化UI元素在用户设备上的表现更为可靠，同时也能节约开发及测试的时间。

## 总结

在今天的教程中，我们讨论了Android平台上用户界面布局的基本设计流程，但并未做深层次挖掘。在本系列文章的下一部分，我们将尝试在应用程序添加用户交互元素、检测并响应按钮点击。接下来，我们将着眼于同Android开发关系最密切的Java相关概念，并进一步探讨应用程序开发过程中所涉及的要素及实践方式。

# 第六章 用户交互

在这篇教程中，我们将对之前所添加的Button元素进行设置以实现对用户点击的检测与响应。为了达成这一目标，我们需要在应用程序的主 Activity类中略微涉及Java编程内容。如果大家在Java开发方面的经验不太丰富也没必要担心，只要按步骤进行即可完成学习。我们将在本系列的 下一篇文章中深入探讨Java语法，从而保证大家了解初步Android开发任务中所必需的编程语言知识。

大家可以在Android当中以多种不同方式实现用户交互。我们将学习两种最为典型的处理方案，从而实现应用按钮对用户点击的感应——两种方案都会用到一点XML代码以及Java实施流程。Android当中包含几种不同的交互UI元素，足以感应来自用户的各类输入操作。输入操作的处理方式必须与 UI项相匹配，但整个过程仍然大体相同。我们将以一个按钮为起点开始探索Android平台上的用户交互，因为按钮无疑是最简单也最常用的界面元素。

## 1. 用户交互基础

在进一步探讨细节之前，我要首先为刚刚接触应用程序开发工作的朋友们解释几项UI概念。为了实现应用交互，我们需要利用特定元素检测用户的交互操作。看过上一篇文章的朋友一定还记得，Android中存在View，而在今天的示例中具体是指Button。要实现交互，我们首先需要“监听”用户的操作。虽然Android主要运行在搭载触控屏幕的移动设备上，但大家仍然可以在计算机上利用编程语言处理交互开发。举例来说，在后面提到“点击”的部分，我们指的是利用鼠标点击或者用手指触摸/点触对应位置。

用户与应用程序的交互方式是多种多样的。他们可以点触、划动以及“长按”对应项目。当这些操作活动发生时，我们将其称为一个“事件”。因此，我们需要通过设置让应用程序监听特定UI项目上是否发生了特定事件。在今天的示例中，我们需要监听针对Button的点击（或者点触/触摸）操作。

我们需要监听并响应这类用户事件。要做到这一点，我们将向Java Activity类中添加代码以实现对接按钮点击的监听与响应。只要按钮上出现点击事件，这部分代码就会开始执行。虽然其它类型的用户交互会涉及不同的方法 代码以及多种多样的事件类型，但其基本过程都是相通的。

## 2. 识别UI元素

### 第一步

为了指明用户交互具体指向哪个View，我们需要在应用程序当中识别出每个交互性View。在文章列举的范例中，我们只讨论一个View——但大家 在今后实际进行应用开发时，可能会用到多种不同类型的交互性View。为了让它们彼此之间有条不紊地运作，我们需要为每个View设置一个用于识别的独特 ID属性，并将其应用于整个应用程序。首先在Eclipse中打开我们的主布局文件并切换到XML编辑标签。接下来找到我们为Button元素添加的代码，利用以下语法为其分配一个ID：

```
android:id="@+id/myButton"
```

我们需要为Android布局中所使用的每一个元素分配ID属性，从而帮助自己顺利识别每个View元素。请注意以上代码中的“@+id”语法。这会提示Android工具在项目资源“R.java”文件中创建一个新ID，并为其指定一个在应用程序内独一无二的文本字符串，也就是“myButton”。在应用中XML布局代码的其余部分乃至其它XML与Java文件内，我们将使用这一名称来指定Button View。而后保存当前布局文件。

### 第二步

打开应用程序中的主Activity文件。我们将向其中添加一点点Java代码，但大家不用为自己令人捉急的Java水平而担忧，只要理解其中与处理用户交互相关的大致流程即可。如果各位朋友原先从未接触过Java，请继续关注我们的下一篇教程，到时候回头再看就会发现现在的内容其实非常简单。我们要在Activity类中创建一个变量来引用Button View。在类声明开头、起始内容之后：

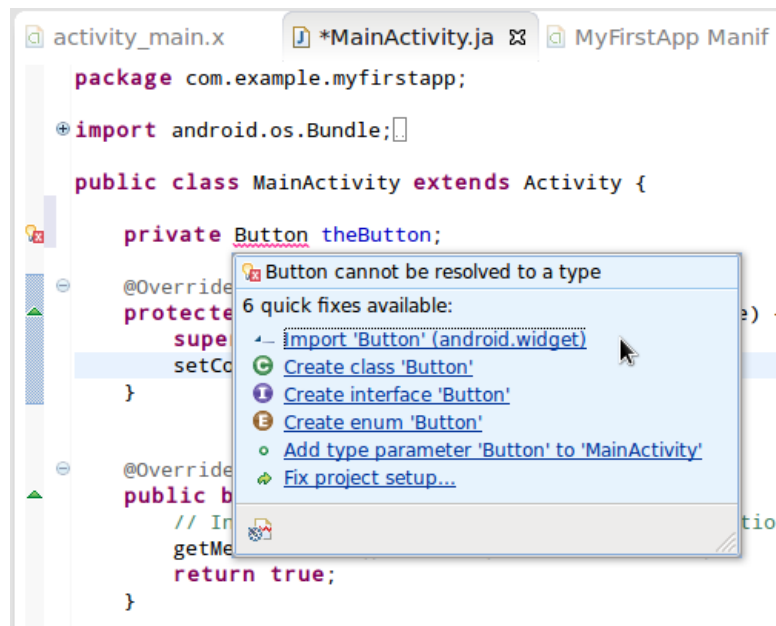
```
public class MainActivity extends Activity {
```

添加变量声明：

```
private Button theButton;
```



我们的声明包含视觉特性（下一次再详加说明）、变量类型以及变量名称。Eclipse可能会在“Button”文本部分加注下划线并提示“Button不能被解析为一个类型”。由于我们使用的是由Android平台所提供的Button类型，所以必须将其导入至类文件当中。将鼠标悬停在“Button”文本上方，Eclipse将为我们显示出一套建议列表。在其中选择“Import ‘Button’ (android.widget)” 。这样类文件顶部就会出现一个可以自由展开与收起的导入声明列表。



### 第三步

现在我们可以布局当中取回指向Button View的引用，并将该引用保存在我们所创建的变量当中。在我的Activity onCreate方法中，紧接着以下代码进行布局设置：

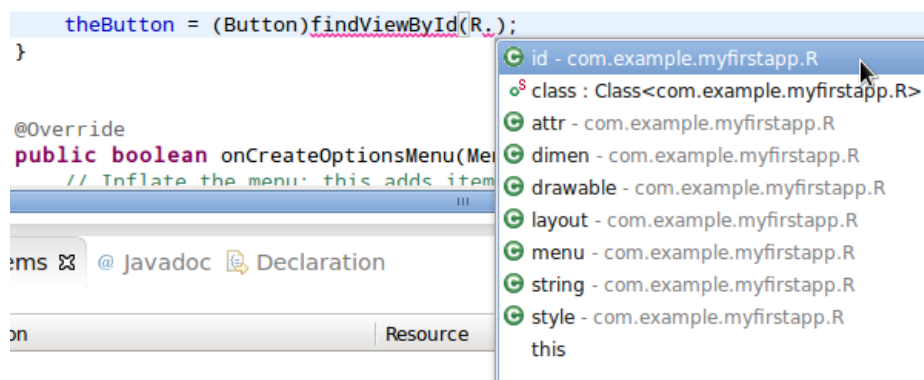
```
setContentView(R.layout.activity_main);
```

如下所示输入一行新代码以取回Button：

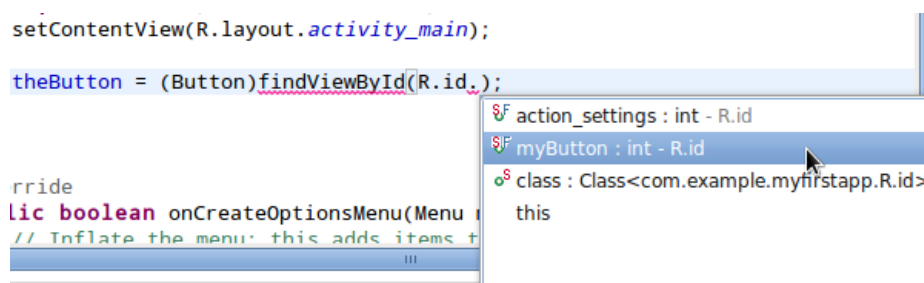
```
theButton = (Button)findViewById();
```

在“findViewById()”的括号中输入“R.”——Eclipse会为我们提供资源类型提示列表。在其中选择“id”。

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



输入另一个句号“.”——Eclipse会显示现有ID值列表。目前我们只添加了一个ID值，选择我们为Button设置的ID名称——也就是“myButton”。



大家会定期利用这种方法在Java代码当中实现资源引用。现在我们应该拥有以下代码行：

```
theButton = (Button) findViewById(R.id.myButton);
```

这条声明将Button View引用分配到了我们刚刚创建的新变量当中，旨在利用其ID实现View识别。

### 3. 监听事件

#### 第一步

在我们要求时，Android系统只会在View上检测事件。因此我们需要为View分配一个监听器。分配监听器也有几种不同的途径，不过我们还是先从最简便的入手：由Activity类自身进行点击的监听与响应。在类的开头按以下内容对声明行进行扩展：

```
public class MainActivity extends Activity implemen
```

```
ts OnClickListener {
```

与前面提到的情况一样，Eclipse这次又会对“OnClickListener”类型提出警告。老办法，鼠标悬停在错误内容上方并根据需求进行导入——选择“Import ‘OnClickListener’ (android.view.View)”。在这里，大家可以看到Eclipse如何帮助我们管理项目中的各组成部分。现在它又显示出另一条错误信息，提示我们需要实施一种方法。先不管它，这个问题放到后面解决。

代码中“implements OnClickListener”部分是指该Activity类将采用一套特定接口。下一次我们会更深入地探讨其具体细节——它从本质上意味着该类将提供一类特殊功能，在我们的例子中该功能允许大家处理点击操作。

## 第二步

回到Activity onCreate方法。在我们通过ID将Button View引用分配给变量的代码行下面，添加新的代码行：

```
theButton.setOnClickListener(this);
```

这一行命令应用程序监听Button上的点击操作。括号中的“this”指定处理点击操作的对象。在本文的示例中，该对象指代Activity类运行实例本身。

## 4. 响应事件

### 第一步

现在我们已经能够响应按钮点击了。在类onCreate方法的最后加入右括号：

```
theButton.setOnClickListener(this);
```

添加以下方法概要：

```
public void onClick(View v){  
    //respond to click  
}
```

再次进行导入流程，在“View”上方悬停并选

择“Import ‘View’ (android.view)” 。由于我们已经命令该类监控按钮上的点击，因此当点击操作发生时，方法就将开始执行（其内容，或者说“方法本体”，将被放置在两个大括号之间）。其中的“View v”是该方法的一项参数，代表该方法将作为指向被点击View的引用处理，这样我们就能对其进行识别。

## 第二步

在onClick方法当中，我们首先需要检查被点击的是哪个View。我们只设置了一种点击监听机制，但应用程序之后可能需要处理多种View点击。在方法本体当中，检查已经通过的View参数是否就是我们引用到变量当中的按钮：

```
if(v.getId()==theButton.getId()){  
    //the button was clicked  
}
```

这是一个条件语句（之后我们再来详细讨论其具体结构），用于检查被点击的View是否拥有与我们的变量相同的ID。如果这部分内容得到执行，我们就能断定确实是经过设置的按钮受到点击。如果交互元素只有这一个，那么测试过程似乎没什么必要；但大家可以想象当应用中的可点击元素越来越多时，我们必然要在onClick执行时判断被触发的到底是哪一个。

## 第三步

在onClick中的if条件部分，我们可以对按钮点击操作做出响应。响应取决于该按钮在实际应用中的作用，不过在此次示例中，我们只是为了演示整个过程。添加以下代码：

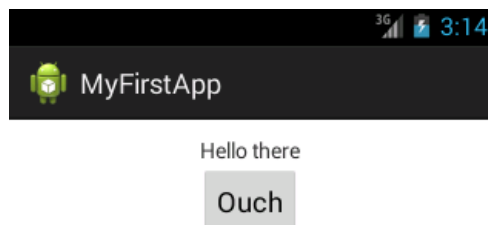
```
theButton.setText("Ouch");
```

这里我们只需简单在按钮被点击后改变其上显示的文本内容。现在大家的onClick方法应该如下所示：

```
public void onClick(View v){  
    //respond to click  
    if(v.getId()==theButton.getId()){  
        //the button was clicked  
        theButton.setText("Ouch");  
    }  
}
```

```
}
```

下图为我们在虚拟设备上点击该按钮后的效果。以后我们将讨论如何让应用程序运行在物理及虚拟设备之上，但现在大家只需观察响应结果即可。



## 5. 替代方案与选项

### 第一步

我们已经演示了一种在Android上处理按钮点击的方法，但方法绝不止这一种。值得关注的另一种替代方案是将下列属于添加到XML布局中的Button内：

```
android:onClick="buttonClicked"
```

上述代码会在按钮被点击后指定需要执行的方法名称。对应方法应该被添加到显示在布局中的Activity类当中。这样一来，我们就不必向Activity 类中加入大量代码，包括创建Button变量、在其中保存View引用、实施OnClickListener或者为该按钮设置一个专门的点击监听类。在本 次示例中，我们可以通过添加以下代码（使用同样的代码以实现一致的操作效果）来取代向类中添加onClick方法：

```
public void buttonClicked(View v) {  
    Button theButton = (Button)v;  
    theButton.setText("Ouch");  
}
```

尽管这种方法看起来似乎更简单，但它利用Java让引用指向布局元素的过程值得认真关注——大家在今后的应用程序开发过程中会经常用到。另外，如果大

家的布局当中包含多个可点击项目，可能会更倾向于在同一个方法内处理所有点击事件——在这种情况下，文章前面提到的方案会更为理想。

除了文章中提到的两套方案，我们还可以通过其它多种途径实现View上的点击处理任务，但其它办法要更复杂一些，不太适合作为新手教学来使用。

## 第二步

在这篇教程中，我们了解了如何利用最基本的方式处理Android系统中的按钮点击事件。该平台还针对不同View类型提供一系列其它用户事件的处理能力，包括长按、按键以及触摸等等。感兴趣的朋友可以参阅Android开发者指南，从中了解自己在未来的项目开发工作中可能接触到的各类事件处理任务。

## 总结

在这一部分当中，我们探讨了如何遵循基本流程实现Android UI中对用户点击按钮的响应。今天涉及到的内容与整套Android用户交互机制相比只能算是九牛一毛，但大家应该能够从这种通用型方法中领会主干、从而指导自己在未来的项目中拿出符合用户输入习惯的开发成果。在本系列的其它教程中，我们将了解Java语言中最为本质的主要特性，从而在Android开发的学习当中取得一个又一个辉煌的胜利。

# 第七章 Java应用程序编程

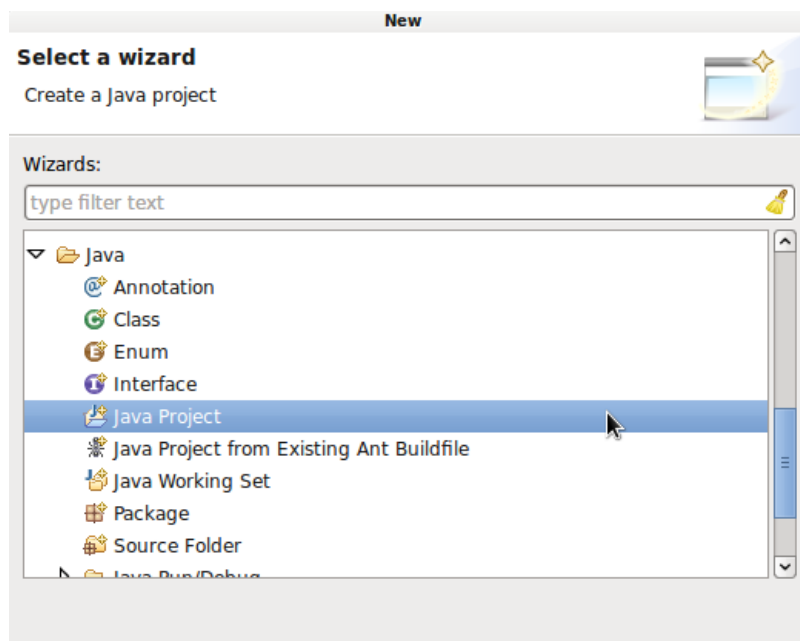
如果大家已经对Java非常熟悉，那么不妨直接忽略这部分内容。如果大家的技巧还存在局限或者对Java这种语言只闻其名，那么本文将为各位解答很多在Android开发当中经常遇到的问题。需要注意的是，这篇文章并不能作为Java起步教程来阅读，最多只能算是基础知识汇总。如果对Java毫无了解，大家还需要参考其它一些额外的Java学习资料。

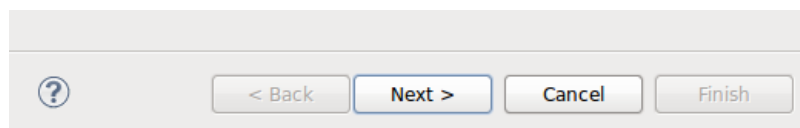
在这份教程中，我们不会过度深入细节，不过如果大家对于某些概念还不大清楚，请点击[此处](#)参阅甲骨文Java指南。这是一份非常优秀的Java语言指导材料，非常适合初学者。如果在刚刚开始阅读时发现本教程提到的一些内容有些陌生，也请大家千万不要惊慌。只要真正着手开始在Android项目中进行尝试，各位很快就能理解本文表达的内容。

## 1. Java语法

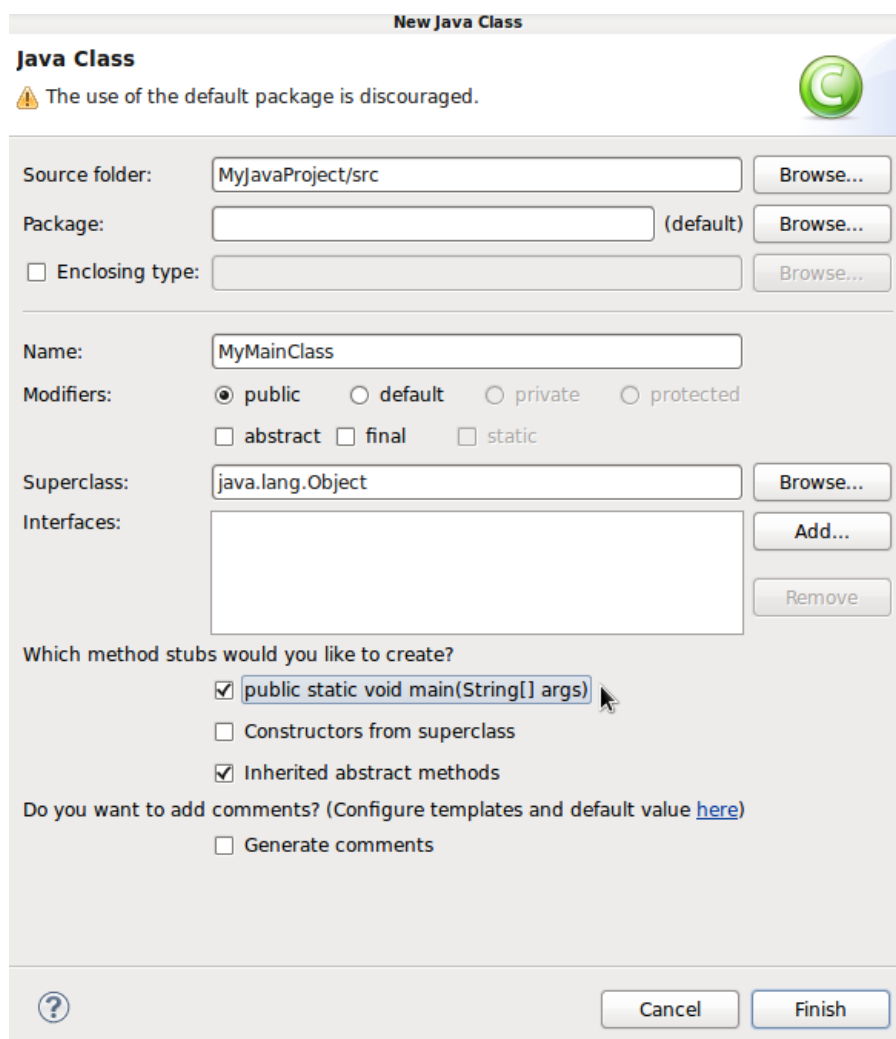
### 第一步

大家已经在我们的Android项目中见识过一部分Java语法了，但为了清楚起见，让我们再从另一个项目重新学习。这一次我们不再使用Android项目，而直接换成Java项目——这样大家就能更轻松地感受到我们所使用的结构。打开Eclipse，点击“New”按钮。在弹出的导航窗口中下滚到Java文件夹中并将其打开，选择“Java Project”然后单击下一步。





输入“MyJavaProject”作为项目名称并点击“Finish”。Eclipse接下来会在工作区内创建我们的新项目。在Package Explorer当中，打开新项目文件夹，右键点击“src”并选择“New”、然后选“Class”。这时在Name框中输入“MyMainClass”。接着勾选旁边的复选项“public static void main”最后点击“Finish”。



Eclipse会创建出类并在编辑器中打开。大家不必过多关注项目结构或者类中的现有内容，因为我们的Android项目所使用的结构与Java项目并不相同。各位可以利用这个项目来磨练自己的Java编码技能，在这里代码的运行与测试都要比Android应用简便得多，而且我们也能更多地关注 Java语法本身。

我们在类文件中看到的“public static void main”行就是主方法。无论方法的具体内容是什么，它都会应用程序运行时加以执行。方法的内容就是显



示在“`public static void main(String[] args)`”后面大括号里的部分。Eclipse可能还生成了一个“to do”行——直接无视就好。在其后创建新行，我们就从这里开始添加自己的代码。

## 第二步

在Java当中，一条变量可以保存一个数据值，例如文本字符串或者数字。当我们在Java中创建或者“声明”一个变量时，需要指定其中的数据类型并为其命名。输入以下代码：

```
int myNum;
```

这一行声明了一个整数变量。我们可以通过以下代码行声明一个变量并为其分配一个值：

```
int myNum = 5;
```

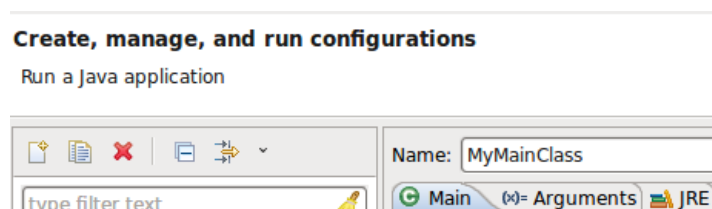
现在我们可以通过名称引用这条变量了。添加以下代码行，从而将变量值写入到输出控制台：

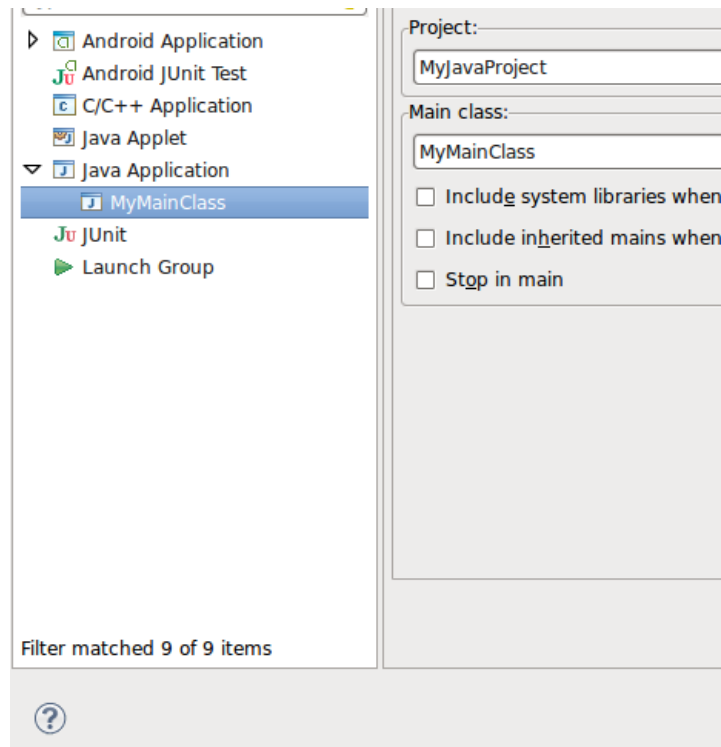
```
System.out.println(myNum);
```

大家一般不会在自己的Android应用中以这种方式向系统输出写入结果，而是用LogCat视图取而代之。不过通过这种输出写入方式，我们能够更为便捷地对Java代码进行测试。

## 第三步

现在让我们运行应用。运行过程与Android应用存在些许不同，但我们会在稍后继续进行说明。选择“Run”，而后选择“Run Configurations”。在弹出的列表左侧选择“Java Application”并点击上方的“New launch configuration”。如果这是我们的惟一个Java应用，Eclipse会自动选择运行刚刚创建完成的小小成果。

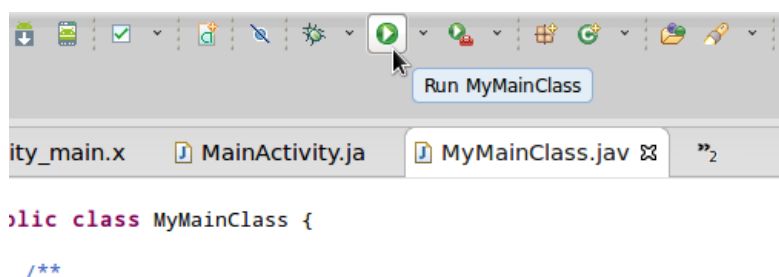




现在点击“Run”来运行我们的应用程序。大家会看到，编辑器下方的控制台视图中将显示出数字“5”。大家可以利用这种方式在学习过程中对Java代码进行测试。



大家现在可以通过工具栏中的“Run”按钮随时运行上一次启动过的项目。



## 第四步

无论何时，只要是在Java中进行变量声明，我们都会使用相同的语法。为了在以后的编程工作中为变量分配不同的值，我们可以通过名称对其进行引用：

```
myNum = 3;
```

上述代码会覆盖掉现有值。在Java中存在着很多不同的变量类型。其中int属于被引用的基本类型，此外还有一些其它数字类型；char用于字符值，而boolean则用于保存真假值。对象的类型也分许多种；关于对象的话题，我们放在以后进行讨论。对于大家来说，最熟悉的基本对象类型应该要数String了，它的作用是保存一条文本字符串：

```
String myName = "Sue";
```

文本字符串值要用引号括起来。大家可以在正面的例子中看到它的使用方法：

```
System.out.println("number: " + myNum);
```

添加上述代码并运行，控制台会显示“number: ”再加上变量值。

## 第五步

在上面我们看到了赋值运算符“=”——正面我们再来看其它一些常见运算符：

```
//add  
myNum = 5+6;  
//subtract  
myNum = 7-3;  
//multiply  
myNum = 3*2;
```

```
//divide
myNum = 10/5;
//remainder
myNum = 10%6;
//increment (add one)
myNum++;
//decrement (subtract one)
myNum--;
```

运算符既可以被用在变量当中，也可以作为硬编码数字（如上所示）：

```
int myNum = 5;
int myOtherNum = 4;
int total = myNum+myOtherNum;//9
```

## 第六步

作为Android基础内容的另一种Java结构就是注释。大家可以通过以下两种方式添加注释：

```
//this is a single line comment

/* This is a multiline comment
 * stretching across lines
 * to give more information
 */
```

最重要的是养成编写代码的同时添加注释的好习惯，这一方面便于我们自己日后查看，另外也能让其他合作者了解我们的编码意图。

## 2. 控制结构

### 第一步

我们向主方法中添加的代码会在Java应用程序运行时同时执行。而在运行我们所创建的Android应用程序时，主Activity中onCreate方法的代码会同时执行。这些方法中的所有代码行都会按从上到下的顺序依次执行，不过执行的流程并不总是线性的。Java当中有很多控制结构，正面我们就从条件开始了解其中最常见的几种。条件语句一般用于进行测试从而确定执行流程。在Java当中，最简单的条件结构就是if语句：

```
if (myNum>3)
    System.out.println("number is greater than 3");
```

这项测试的目的在于检查变量的值是否大于3。如果确实大于3，那么字符串将被写入输出结果。如果小于等于3，则不向输出结果写入任何内容、继续执行程序中的下一行。条件测试会“返回”一个真假值。真与假都属于boolean值。我们也可以向其中添加else，这样其内容只会在返回假值时才执行：

```
if (myNum>3)
    System.out.println("number is greater than 3");
else
    System.out.println("number is not greater than
3");
```

在我们的示例中，else语句会在值等于或者小于3时执行。尝试在代码的整数变量中添加不同的值，看看条件测试结果会发生哪些变化：

```
if (myNum>10)
    System.out.println("number is greater than 10")
;
else if (myNum>7)
    System.out.println("number is greater than 7");
else if (myNum>3)
    System.out.println("number is greater than 3");
else
    System.out.println("number is 3 or less");
```

只有在流程中的每一次测试中都返回假值时，所有测试行才会被彻底执行一遍。因此对于大部分数字来说，只会输出一条字符串。如果有必要，大家可以把多条else if语句串点起来。大家还可以利用if语句与一个或者多个else if相结合，而不必每一次都在之后单独设置else。

下面我们测试一个数字是否大于另一个。尝试使用以下变量：

```
if (myNum<10)
    System.out.println("number less than 10");
if (myNum==10)
    System.out.println("number equals 10");
if (myNum!=10)
    System.out.println("number is not equal to 10")
```

```
;
if(myNum>=10)
    System.out.println("number either greater than
or equal to 10");
if(myNum<=10)
    System.out.println("number either less than or
equal to 10");
```

大家也可以利用包含字符串的变量类型进行类似的测试。要同时进行多项测试，可以利用以下语法：

```
if(myNum>=10 && myNum<=50)
    System.out.println("number is between 10 and 50
");
```

其中的“&&”是作为“and”运算符存在的，意思是整条语句只有在两项测试都返回真值时才被判定为真。而“or”运算符将在两条测试中任意一条返回真值时判定为真：

```
if(myNum<0 || myNum!=-1)
    System.out.println("number is less than 0 or no
t equal to -1");
```

为了将代码组成代码块，我们可以使用大括号——两个括号之间的所有代码都会在测试返回真值时执行：

```
if(myNum<10)
{
    System.out.println("number less than 10");
    myNum=10;
}
```

这些括号能够在循环、方法以及类中实现代码分组。

## 第二步

接下来让我们看看循环。下面的for循环会进行十次遍历，意味着它的内容将执行十次：

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

```
        System.out.println(i);  
    }
```

在for循环中的第一个表达式旨在将一个整数型计数器变量初始化为零。第二个表达式则是条件测试，检查该变量的值是否小于10。如果返回的是真值，则循环内容得到执行；如果返回的是假值，则中止循环。一旦循环当中的内容开始执行，第三个表达式就同时执行，即递增计数器。

另一种循环while所使用的语法稍有区别。以下代码就是我们利用while来实现上面的for循环的相同执行效果：

```
int i=0;  
while(i<10){  
    System.out.println(i);  
    i++;  
}
```

循环当中可以容纳多行代码，其中包括其它循环。

### 第三步

我们已经接触了主方法与Android的onCreate方法。下面让我们一起学习如何创建自己的方法。将以下方法放置在主方法的右括号之后：

```
public static void doSomething(){  
    System.out.println("something");  
}
```

该方法被定义为public，这意味着项目中的所有类都可以调用其进程。如果它的属性为“private”，则代表只供同一个类内部进行访问（也就是‘visibility’）。一般来说，大家不会在自己的第一个Android应用中包含“static”修饰符，因此忽略掉它即可。而“void”代表着返回类型。在我们的示例中，该方法不会返回任何值。为了执行该方法，我们需要在主方法中添加一项调用：

```
doSomething();
```

运行应用程序并查看其功能——改变方法以返回一个值：

```
public static int doSomething() {  
    return 5;  
}
```

改变方法调用并再次运行：

```
System.out.println(doSomething());
```

返回的值会被写出。方法还可以接收参数：

```
public static int doSomething(int firstNum, int secondNum) {  
    return firstNum*secondNum;  
}
```

在调用该方法时，大家必须符合正确的参数类型与数字：

```
System.out.println(doSomething(3, 5));
```

方法能够将应用程序进程拆分为逻辑块。如果大家需要多次执行同一项任务，那么它们的作用将非常重要；我们可以简单在方法中进行定义，然后在需要时随时调用。如果各位需要改变处理流程，也只需在方法代码中进行修改。

## 3. 类与对象

### 第一步

我们已经了解了方法如何被用于重新使用代码并将其拆分成逻辑部分。类与对象则能够在更大的范围内实现此类功能。大家可以将应用中的任务划分成不同对象，其中每个对象都由它所归属的类为其定义一系列职责。这类似于用一种方法负责一个特定功能区域，不过一个对象可以拥有多个方法而且能够保存数据值。

想象我们正在创建一款游戏——大家可以创建一个专门用来处理用户详细信息的类。在Package Explorer中选择我们的应用程序包，右键点击并选择“New”而后是“Class”。输入“GameUser”作为类名称，确保main method stub复选框没有被勾选，然后点击“Finish”。Eclipse会打开这个类文件，在初始状态下其中只包含它的类声明概要：



```
public class GameUser {  
    //class content  
}
```

大家所添加的所有内容都应该位于两个大括号之间（除非大家添加导入语句，这部分内容将位于最前方）。我们的Android应用会识别出罗列于文件开头的包名称。当然这里我们使用的是默认包，所以前面并没有列出其它内容。

## 第二步

在这个类当中添加以下变量：

```
private String playerName;  
private int score;
```

这些被称为“实例变量”，因为它们被定义为我们所创建的类中的实例。在它们之后添加一个构造方法，它会在该类中的某个对象被创建后开始执行：

```
public GameUser(String userName, int userScore){  
    playerName=userName;  
    score=userScore;  
}
```

这里的构造永远与类使用同样的名称，而且可能要求也可能不要求使用参数。该构造通常应该向实例变量分配值，一般是通过参数来实现。

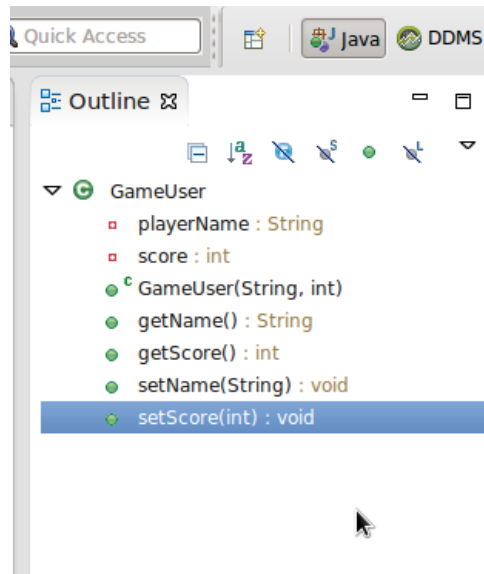
## 第三步

类也可以定义方法。将以下典型集合添加到构造之后：

```
public String getName() {return playerName;}  
public int getScore() {return score;}  
public void setName(String newName) {playerName=new  
Name;}  
public void setScore(int newScore) {score=newScore;  
}
```

这些被称为get与set方法，或者叫getter与setter，因为它们会利用接收及发送实例变量值的能力将外部代码添加到类中来。查看Eclipse中的Outline视

图，理解它如何帮助实现导航类内容。



#### 第四步

保存我们新建的类文件。回到主类当中，为新类在主方法中创建一个对象：

```
GameUser aUser = new GameUser("Jim", 0);
```

我们符合构造当中的参数要求——以上代码中的“new”关键字将使构造开始执行。现在我们可以使用这个类实例，通过调用其方法访问其中的数据：

```
System.out.println(aUser.getScore());  
aUser.setScore(5);  
System.out.println(aUser.getScore());
```

运行程序以查看调用对象上的public方法之后，值产生了什么样的变化。大家可以创建多个对象实例，并对它们进行分别管理：

```
GameUser anotherUser = new GameUser("Jane", 5);
```

## 4. 继承与界面

#### 第一步

我们已经了解了如何通过创建对象实例来使类定义一系列职责。它的效果不仅作用于我们所创建的类本身，同时也作用于其它我们能够使用的现有Java及

Android类。除此之外，在创建这些平台类实例的同时，大家还可以利用继承对其加以扩展。在继承机制的帮助下，我们可以创建出一个继承现有类功能、同时又拥有自己运行流程的类。在我们所创建的第一个Android项目中，主Activity类就是一个很好的例子。

现在打开Android项目中的这个类。在类声明的开头，大家会看到“extends Activity”。这意味着该类属于Android Activity类中的一个子类。这里的Activity类用于使Android系统处理向用户呈现的屏幕内容，而各方法则用于不同变量状态下的屏幕内容（创建、暂停与消除等）。通过向Android Activity类声明中的定义方法添加代码并在必要时增加额外方法的方式，我们能够更专注于实现应用程序的独特风格。

这是我们经常会在Android上使用的模式，用于为应用程序的常见需要扩展定义类。大家可以用自己的类适当对其加以补充。

## 第二步

再来看Activity类中的起始行。请记住，我们添加了“implements OnClickListener”来处理UI中的按钮点击操作。这将通过引用被实施在界面当中。界面类似于一个我们利用“extends”继承而来的类，只不过界面声明只需简单罗列方法概述。大家需要对每一项概述进行方法实施。因此当我们实施OnClickListener时，需要委托该类提供一个onClick方法——正如我们在之前的Android项目中所做。因此界面类似于一项协定。在继承机制的辅助下，扩展类能够继承由类声明所提供的、用于实现超类（即经过扩展的类）的方法实施。如果需要，大家可以覆盖这些实施内容。

## 总结

在今天的教程中，我们简要介绍了一些Java语法方面的基本知识。当然，还有很多其它关于Java的结构与概念需要了解。如果大家此前没有接触过Java，又希望保证自己能拥有足以顺利应对Android开发工作的必要知识，请务必点击此处认真阅读甲骨文公司发布的Java指南。其中需要认真学习的主题包括数组与交换语句。在本系列的后续文章中，我们将探讨一些大家最常用到的Android类。而在下一章节中，我们则开始探索Android应用项目中的资源。

# 第八章 应用程序资源

在系列教程中的最新一篇里，我们将研究大家最可能在第一个开发项目中涉及到的资源类型。项目资源当中包含布局、图片以及数据值，这些都是应用需要使用的元素。当我们创建一个新项目时，项目目录下会自动生成多个用于容纳通用资源类型的文件夹。如果需要，大家还可以添加更多文件夹以扩展资源类型数量。

大家可以通过Package Explorer浏览“res”文件夹当中的内容，这些就是我们之前在创建项目时所使用的资源。打开文件夹、看看里面都藏着哪些宝贝。大家还可以在资源目录下添加更多新文件夹，也可以在各文件夹中添加新文件，或者是直接使用现有文件（例如我们在前几篇系列文章中所使用的布局与字符值文件）。

## 1. 备用资源

在我们正式开始之前，首先指出一点注意事项——大家可以将自己的Android资源划分为两大类：一种是能够被跨设备使用的资源，另一种则是针对设备特定子集的资源。大家可以在现有项目结构中发现二者的实例。在Eclipse Package Explorer当中，查看“res”目录。请记住，不同的可绘制文件夹对应着特定设备屏幕像素密度。在今天的文章中，我们打算使用非特定可绘制文件（即能够跨设备使用的资源）。

大家可以通过类别限定的方式为每种资源类型添加备用目录。在Eclipse当中，这代表着“drawable-hdpi”、“drawable-xhdpi”等不同类型的Android平台支持多种用户设备分类方式，其中包括屏幕尺寸、像素密度、API级别、语言以及区域等等。任何在名称中不包含类型限定的资源类型文件夹都能够实现跨设备使用。大家并不一定需要为所有资源类型都设置类别限定文件夹，但当我们针对不同设备进行应用测试时，可能需要在不同配置之间做出一些细微调整。

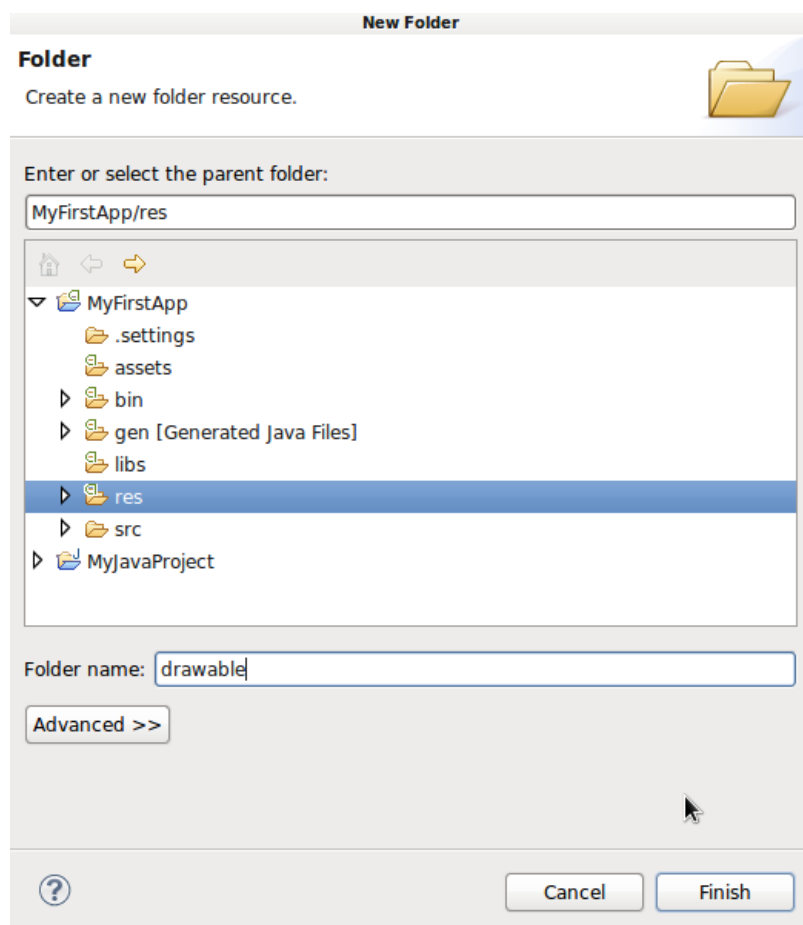
## 2. 可绘制资源

### 第一步

我们都知道，Eclipse会创建多个可绘制文件夹，每一个都针对一种特定的密

度桶。可绘制文件夹当中包含我们在应用程序中所使用的任何图片。大家可以在Eclipse之外准备一些数字格式的图片，例如JPEG、PNG以及GIF。大家还可以利用XML代码定义可绘制资源。下面我们就着手尝试，并将其添加到主布局当中。

尽管大家应该尝试针对各种特定像素密度创建可绘制资源，但由于本教程的篇幅所限，今天我们姑且使用适应所有设备的单一可绘制方案。在Eclipse Package Explorer当中选择“res”文件夹，选择“File”或者右键点击该文件夹->选择“New”->“Folder”来创建一个新文件夹。将该文件夹命名为“drawable”然后点击“Finish”完成创建。



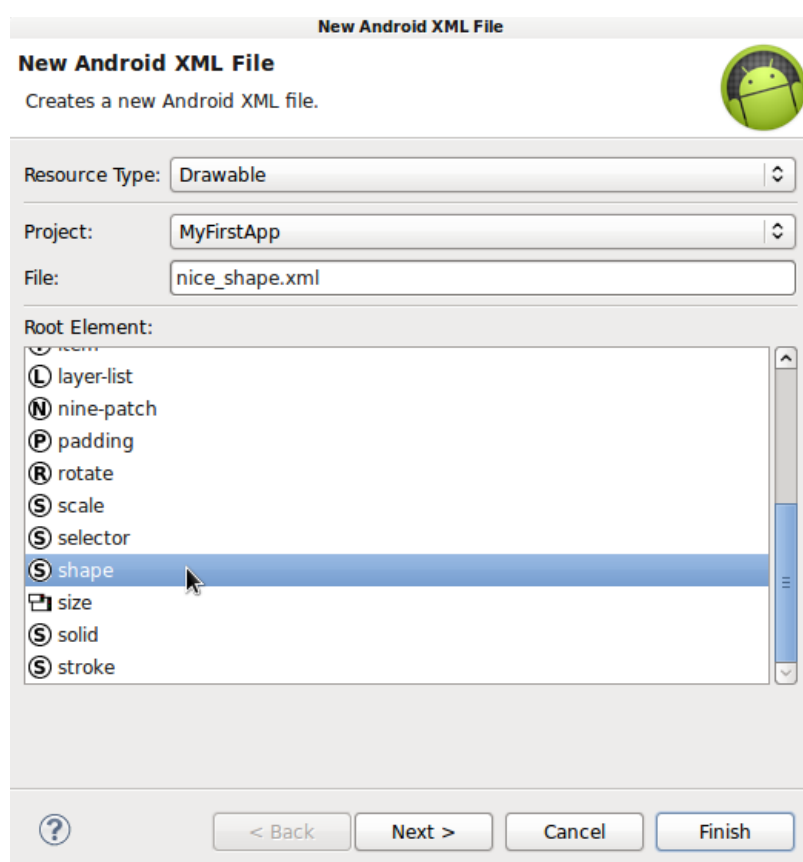
无论何时，只要我们需要在项目当中创建新文件夹，都可以遵循以上流程。

## 第二步

现在我们新建的可绘制文件夹应该已经与其它原有可绘制文件夹一道，显示在Package Explorer当中了。正如我们之前所提到，如果某个文件夹并不针对特定设备子集（即由像素密度分类或者API级别加以定义），那么大家完全可以随意置换其中的资源并将其用于任何用户设备。因此，无论我们向新的可绘制

文件夹当中添加什么内容，其都会显示在所有用户设备之上。对于大部分可绘制资源来说，我们最需要注意的就是其不同像素密度版本；不过为了简单起见，我们将在今天的文章中使用这个新文件夹。

在Package Explorer当中选中我们的新建可绘制文件夹、右键点击或者选择“File”、而后选择“New”以及“Android XML File”，从而在文件夹中创建一个新文件。这时Eclipse会弹出新文件的创建导航。Android支持多种不同的可绘制文件类型。今天我们要创建的是一个图形可绘制文件，并利用其中所包含的不同图形及外观实现标记的目的。大家可以点击此处，通过开发者指南了解其它可绘制类型。



在顶部的下拉列表中，我们可以选择资源的具体类型——由于我们是在可绘制文件夹中创建新文件的，所以Eclipse会自动选中“drawable”。接下来是项目下拉列表，同时应该自动填入我们所选择的项目。然后是在文本输入框内为文件命名——输入“nice\_shape.xml”。输入框下方是我们可以选择的根元素列表。下滚并选择“shape”，因为我们打算定义的是图形可绘制资源。最后点击“Finish”，Eclipse会创建新文件并在编辑器中打开。

### 第三步

在图形可绘制对象中，大家可以选择一系列通用图形类别，其中包括矩形、椭圆、线条和圆环。选择了图形类别之后，我们就可以对其具体属性加以定义，例如实心或者渐变颜色、边角、填充、尺寸以及笔触等。我们通过以下代码编辑根shape元素，从而获得一个矩形：

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >

</shape>
```

大家接下来可以通过在根shape元素中添加其它元素来定义图形属性。首先定义一个梯度：

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >

    <gradient
        android:angle="270"
        android:centerColor="#FFFFFF00"
        android:endColor="#FF0000FF"
        android:startColor="#FFFF0000"
        android:type="linear" />

</shape>
```

我们需要定义梯度的类型、角度外加起点、终点以及中央的颜色。在gradient元素完成后，我们再添加一些圆角：

```
<corners android:radius="10dp" />
```

下面添加笔触：

```
<stroke
    android:width="2dp"
    android:color="#FF339966" />
```

当我们在编辑器中输入内容时，会看到Eclipse提示的可用元素与属性类型。在完成本教程之后，大家可以用一段时间来尝试它们对效果的影响。我们将在

下一步骤中将创建好的图形用在UI当中。现在保存可绘制文件。

提示：要在应用程序内使用Eclipse之外所准备的数字图片文件，大家只需直接将其复制到工作区目录下对应的可绘制文件夹当中即可。在文件向资源文件夹的复制过程结束后，大家可能需要刷新Eclipse视图——即在Package Explorer中选择对应项目，右键点击或者选择“File”，然后选择“Refresh”。这样我们就能在应用程序代码中引用这些图片文件了。

### 3. 布局资源

#### 第一步

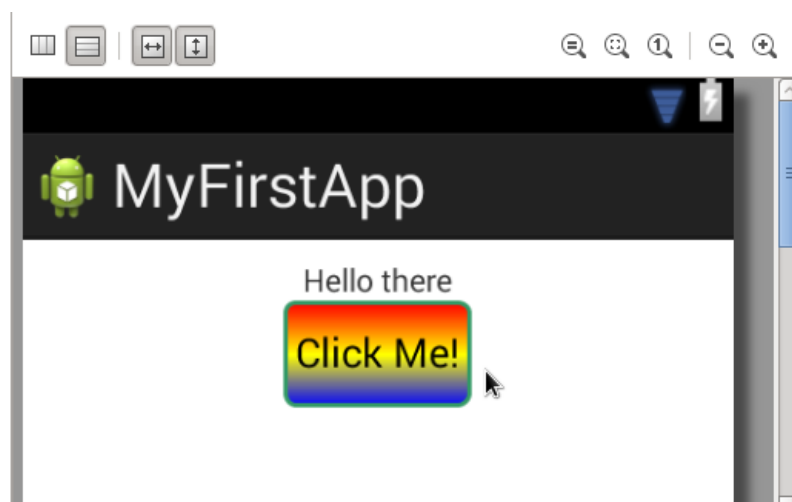
我们先回顾一下之前文章在设计应用程序用户界面时，其中所涉及的布局知识。先来看布局文件与可绘制交互。我们可以在布局当中将可绘制资源作为View或者特定View的背景加以显示。首先列出我们已经创建完成的图形可绘制资源，并将其作为现有View的背景。打开应用程序的主布局文件，将图形可绘制资源作为我们之前添加的按钮的背景图案。为Button元素添加以下属性：

```
android:background="@drawable/nice_shape"
```

我们利用资源类型与名称（即我们为可绘制资源设定的文件名）将其引用到布局当中。请注意，这与我们之前用于引用字符串值的语法形式是一样的。保存并切换到Graphical Layout标签，查看图形是否已经成为按钮背景。大家可能会注意到，按钮还需要一点填充调整。现在切换回XML编辑模式并为Button元素添加填充属性：

```
android:padding="5dp"
```

重新切换回Graphical Layout标签并查看实际效果。







## 第二步

现在让我们在专用View中使用图形可绘制资源。将以下代码添加到布局中的Button元素内：

```
<ImageView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_margin="10dp"
    android:src="@drawable/nice_shape" />
```

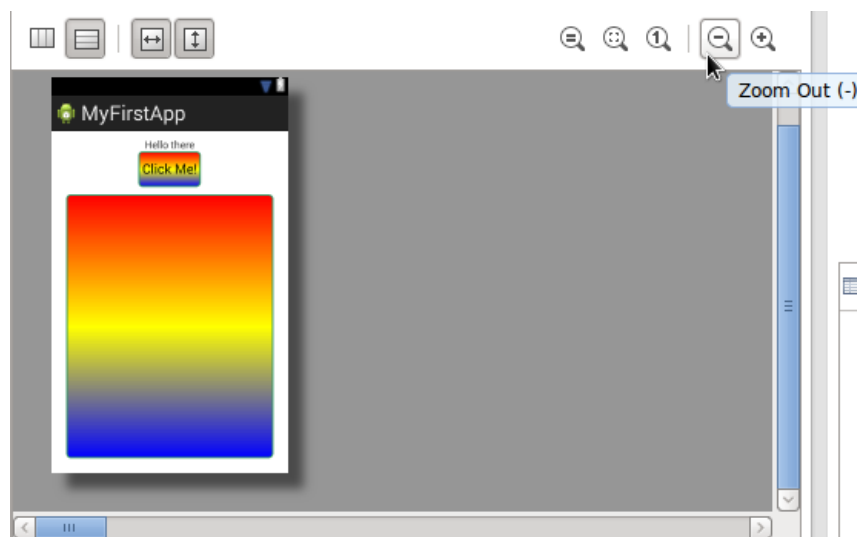
我们设置ImageView来填满除了边界之外的空间。大家也可以选择为其设置固定长度与宽度来达到同样的效果。请注意，Eclipse会显示警告，因为我们还没有为其添加内容描述属性。现在开始着手进行，打开我们的“res/values”字符串XML文件并添加以下内容：

```
<string name="pic">Picture</string>
```

现在大家可以将该字符串添加到布局文件下的ImageView当中：

```
android:contentDescription="@string/pic"
```

如大家所见，资源的使用需要在项目中的不同文件之间来回切换，同时使用标准语法模式实现对不同资源条目的引用。现在再次切换以查看图形预览效果。



大家可以使用图中所示的控件进行放大与缩小。

## 4. 其它资源类型

到目前为止，我们已经在应用程序当中使用了三种资源类型：布局、可绘制与值。其实可以在应用中使用的资源类型还有很多，只需通过前面介绍的方法加以引用即可。正如我们在本系列之前的文章中所提到，大家还可以利用以下语法引用Java文件中的资源：

```
//R.type.name  
R.string.pic//example
```

让我们简要总结一下将来可能会用在应用程序当中的其它资源类型。在之前的文章和本文的前面几个章节，我们已经使用了可绘制与布局资源。字符串值也出现在布局文件当中。现在在Package Explorer当中打开“values”文件夹，除了字符串文件之外，Eclipse通常还会添加一个尺寸文件和一个样式文件。在样式文件内，大家可以定义外观属性从而与应用程序的UI风格保持一致。而在“dimens”文件中，大家可以定义应用程序所使用的尺寸值。

如我们之前所提到，大家可以通过限定机制为特定设备属性创建备用资源类型文件夹。如大家所见，Eclipse会针对特定API级别创建值文件夹，但我们也可以利用其它一些限定手段实现对特定设备的支持。举例来说，大家可能希望在我们所添加的ImageView当中使用固定宽高尺寸，从而使显示大小与设备屏幕尺寸完美契合。为了实现这一点，大家可以添加值文件夹及其中的尺寸文件来匹配各种尺寸或者像素密度桶（例如‘-small’、‘-large’、‘-hdpi’、‘-mdpi’等等）。通过在每个文件当中包含尺寸值，并在相同的值名称之下使用不同数字，Android系统将自动选择最适合用户设备的方案。

大家可能需要用到的其它资源类型还包括数字、菜单、动画以及颜色值。

Eclipse通常会在我们创建一个应用程序之后为其创建一个菜单文件夹，现在请大家打开该文件夹看看其中的内容。要定义XML动画，大家可以向“res”目录中添加一个“anim”或者“animator”文件夹，也可以直接将动画文件添加到可绘制文件夹当中——具体方式取决于我们所使用的实际动画类型。

如果大家希望在应用程序UI中使用一组颜色，则可以在保存于值目录下的文件中对color元素进行定义。每个color元素都可以包含一个HEX值与一个名称属性，这样我们就可以在其它文件当中引用这些颜色了。至于那些无法被归于任

何一种Android定义类别的XML资源，大家可以将其保存在“res”目录下的“xml”文件夹当中。

要对Android当中的全部资源类型拥有透彻了解，大家可以查看开发者指南当中的“资源类型”与“更多资源类型”两个章节。虽然在学习的起步阶段最好只接触比较浅显的内容，但这些资料仍然值得大家认真阅读，从而为今后的开发工作奠定良好的知识基础。

提示：当大家参阅Android项目实例或者开发者指南时，经常会在其中发现一些经常被资源使用的标准文件名。不过文件名本身其实可以随意选择——只要大家使用正确的文件夹名称与元素，这样应用程序代码就可以通过识别系统访问所有资源。除此之外，坚持使用传统文件名可以让我们的应用程序更加清晰且便于理解，特别是对于值文件夹来说。

## 总结

在今天的文章中，我们了解了Android系统当中关于应用程序资源的基础知识。不过正如之前所提到，还有很多未知领域等等等着我们去探索。对于大家的第一款应用，各位只需要使用相对简单的方案帮助自己习惯资源的使用方法即可。但随着我们在应用开发方面的不断成长，大家应该尝试思考各类用户设备在运行我们应用时可能面临的情况，并为其提供必要的额外资源。在本系列教程的下一部分中，我们将一同了解项目的Manifest（清单）文件。

# 第九章 Manifest文件

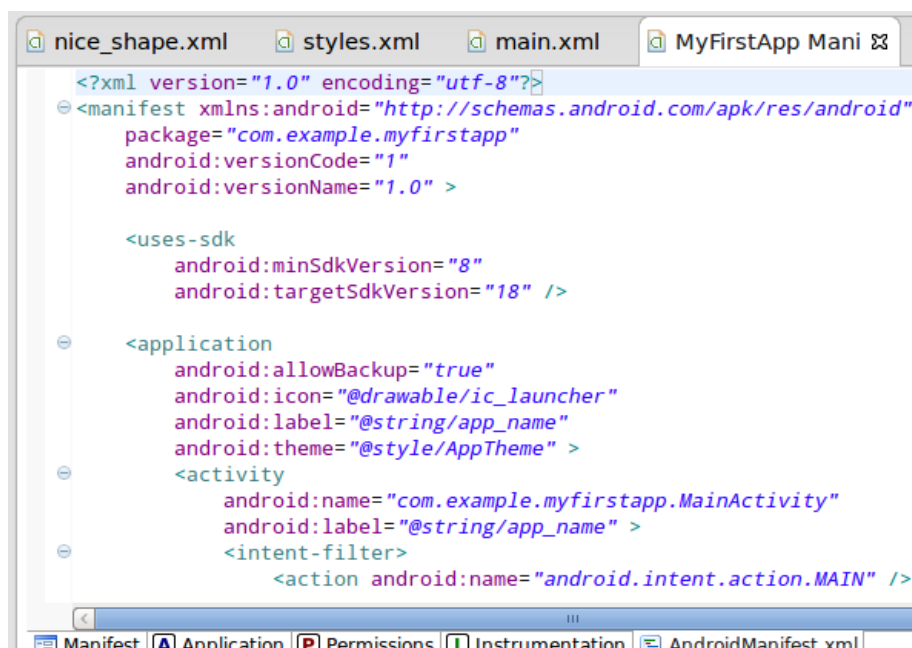
到目前为止，我们已经熟悉了Android项目中的各个组成部分，包括其资源。在今天的文章中，我们将以项目Manifest文件作为核心内容。

对于一个项目来说，Manifest既可以很简单、也可以很复杂，其具体情况要视应用程序而定。我们首先看看Manifest中那些在所有应用中都不可或缺的组成部分，再进一步探索未来在项目开发中可能涉及的备选组成部分。

Manifest文件当中可以包含众多元素与属性，我们不妨点击[此处](#)在Android开发者指南当中找到关于它们的详细信息。Manifest拥有几大主要作用：它指定应用程序包、提供应用组件的形式化描述，此外还负责声明权限、必要的API级别以及链接库等。我们目前只讨论能够在Manifest中列出的最为基础的元素与属性，但大家也要知道其中完全可以容纳更多元素、并在所涵盖的元素范围之外使用更多附加属性。

## 1. Manifest元素

在Eclipse当中打开我们的项目Manifest文件——大家总能在项目的根目录下找到这份Manifest。正如前面所提到，大家可以通过多种方式查看Manifest内容。在底部的编辑器区域中，大家可以看到Manifest、应用、权限、工具以及XML代码等多个标签。现在快速浏览这些标签——我们需要使用XML代码，所以请切换到“AndroidManifest.xml”标签。





Manifest文件中所显示的元素是由Eclipse在我们创建项目的同时生成的。但这些只够满足简单应用的需求，在大多数情况下、我们还需要在创建项目时向Manifest中添加更多元素。文件中的根元素为manifest元素：

```
<manifest xmlns:android="http://schemas.android.com
/apk/res/android"
    package="com.example.myfirstapp"
    android:versionCode="1"
    android:versionName="1.0" >
</manifest>
```

Eclipse会在项目创建时将大家所选择的包名称作为manifest元素的属性。版本代码与名称初始分别为1与1.0。当大家将应用程序提交到Play商店中并进行后续次级版本更新时，需要为每一次更新分配一个更新数字。版本名称是用户们在Play商店中所看到的应用程序的实际名称，所以大家可以随意使用自己喜欢的数字来表示。用户们是无法看到版本代码的，而且新版本的数字必须高于旧版本——不过每一次递增的幅度并不固定。如果大家尝试向Google Play软件商店上传新的应用程序版本，但其版本代码并未高于之前版本的代码，那么Play商店将拒绝这一上传操作。

## 2. Uses-SDK元素

我们在manifest元素当中首先见到的应该是uses-sdk元素：

```
<uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="18" />
```

这一元素负责定义最低必要API级别以及大家在测试项目时所设定的目标级别。我们在创建应用程序时就需要选择这些相关值。如果需要，大家也可以在项目创建完成后，通过修改Manifest文件内容进行属性变更。举例来说，大家可能发现自己需要使用某些当前选定API级别无法实现的平台功能，这时就需要通过Manifest文件作出调整了。如果大家改变了SDK版本，Eclipse会重新建立整个项目。

如果用户设备所运行的API级别低于项目的最低要求，则无法下载并安装我们

的应用程序。列出目标API级别代表着我们已经对当前应用版本进行过测试。为保证应用产品的可靠性，大家应该在尽可能多的API级别之下进行应用程序测试。

### 3. Application元素

Eclipse还会向我们的Manifest中插入application元素。该元素中包含多种子元素，我们将在稍后逐一讨论。现在先来看看打开标签后的内容：

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

需要注意的几种主要属性分别为图标、标签和主题条目。该元素还能够承载多种附加属性。图标属性代表应用可绘制资源中的某个文件。在默认情况下项目会直接使用Android图标，但大家可以添加自己的图标文件并将其引用到这里。标签图标引用的同样是来自应用程序资源的字符串。打开名称中包含“res/values”字符串XML文件，大家会看到被引用的字符串，内容如下所示：

```
<string name="app_name">MyFirstApp</string>
```

这应该是一条可读字符串，因为它会显示在用户界面中的多个位置，包括紧靠着启动图标的设备菜单。大家通常需要变更上述字符串内容，例如在表述中加入空格——即“My First App”。

回到Manifest文件当中，请注意application元素的属性。它同样引用一种资源，大家可以在“res/values”类型的XML文件中找到其具体引用关系，马上去看看吧。接下来切换回Manifest标签。如果大家稍后决定定义自己的应用程序风格，则可以在主题属性当中对其加以引用。

### 4. Activity元素

在application元素当中，大家会看到一项activity元素——它对应着我们在项目开发中所创建的Activity类。在activity元素中包含有多种子元素，我们

稍后再详加讨论。现在先来看看打开后的标签内容：

```
<activity
    android:name="com.example.myfirstapp.MainActivity"
    android:label="@string/app_name" >
```

名称属性利用应用程序包中所限定的路径引用对应类。标签允许我们控制Activity启用时、窗口标题中的显示内容。在默认情况下，窗口标题往往就是应用程序名称，因此大家一般不需要再进行额外调整。不过随着应用程序复杂性的提高，大家将向项目中添加更多Activity——每一个对应UI中的一套屏幕显示方案。每一次向应用程序中添加新的Activity类时，大家都需要在application元素中添加一个对应子元素，如下所示：

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

如大家所见，我们并不总是需要使用完整的应用包名称。上面展示的简写形式也能正常起效，只要Activity类（名称为‘About’）仍然处于manifest元素所定义的应用包当中即可。Activity元素能够容纳多种属性，从而决定用户如何与其进行交互。

## 5. 意图过滤器

在主activity元素当中，大家会看到一个intent-filter元素：

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

这里的Intent Filter用于描述主activity能够响应哪些“意图”。在Android系统中，所谓意图是指在Activity启动时向其传递的数据对象。当大家在自己的应用程序中启用一个又一个Activity时，就需要使用到意图机制了——不过意图也可以在不同应用之间进行传递。

针对主activity的意图过滤器代表前者应该在应用程序运行时被一并启用。意

图是通过action元素实现这一效果的，也就是上述代码中的“MAIN”操作。另外，category元素旨在通过分类名称描述意图过滤器，在我们的示例中就是“LAUNCHER”。这两种元素相结合意味着我们的应用程序应该利用Activity作为其主入口点，而且该入口点将在应用程序运行时一并启动。

意图过滤器当中可以包含多种子元素，其中包括数据规范。如果大家希望了解更多信息，可以点击[此处](#)查看Android开发者指南中的“意图与意图管理器”章节。在activity元素中，除了intent-filter之外还可以包含其它子元素——例如metadata，其作用是定义一个数据条目的名值对。

## 6. 用户权限

我们已经了解了创建项目时Eclipse当中所包含的所有Manifest元素，现在再来看看今后开发复杂应用时需要涉及的其它元素。某些应用程序需要判断用户是否有权执行特定操作或者查看特定内容，这种与权限相关的功能要靠uses-permission元素来实现。当用户在Play软件商店中查看我们的应用时，该平台会在应用下载前显示使用全部功能所必需的权限。如果用户选择继续，则需要接受权限控制提示，而后应用才能正常运行。在Manifest当中必须强制要求的权限包括使用内部数据、写入外部存储以及访问摄像头等设备功能。我们通过以下示例代码演示这一概念：

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

应用程序还可以强制要求其它几种潜在权限，请大家点击[此处](#)查看API引用说明中的“Manifest.permission”部分。

## 7. 用户设备

有这样一类Manifest元素可以被用于描述应用程序运行所必需的硬件及软件功能，其中包括uses-configuration元素。在该元素中，大家可以为导航、键盘以及触摸屏选项指定相关要求。在uses-feature元素中，大家可以通过功能名称与布尔标记列举关于硬件或者软件的单一功能要求。这些功能包括蓝牙与摄像头选项，例如闪存、位置检测以及传感器。而supports-screens元素则允许大家为应用程序定义所支持的屏幕尺寸，所指定的元素可同时涉及尺寸与像素密度。



## 8. 其它元素

我们前面所探讨的元素主要围绕着自己的初始应用展开，但其它不同类型的应用中也有一些元素值得关注。我们为标准应用所创建的初始项目在启动后只涉及用户菜单并为使用者提供一个或者多个Activity屏幕。然而其它类型的应用还可能包括功能部件以及持续运行的后台进程——旨在处理数据访问或者接收系统通知。

应用程序通常会利用Android系统中的Service类来处理后台进程，这就要求我们在Manifest当中添加service元素——与Activity类似，service元素与Service类之间一一对应。Android应用中的内容提供者负责管理对数据源的访问，具体内容被列入provider元素当中。最后，Manifest中的receiver元素旨在帮助应用接收来自其它应用或者操作系统本身的意图。

## 结论

到这里，我们已经阐述了Android应用的Manifest文件中，最为基础的各项功能。当大家创建应用程序时，Eclipse都会同时向Manifest文件中添加初始项目必需的主要元素。随着在应用中引入更多功能，我们需要不断向Manifest内增加新内容以确保应用程序整体能够顺利起效。如果大家在开发过程中的实机或者虚拟设备测试出现了问题，很可能是因为在Manifest中缺少某些必要元素。在下一篇系列教程中，我们将一同了解Android应用如何存储及访问数据。

# 第十章 应用程序数据

我们已经熟悉了Android应用程序的结构与基本组成元素，其中包括资源、清单与用户界面。在着手进行Android平台的功能性应用开发之后，大家肯定需要保存这样或者那样的数据信息。Android平台提供多种选项，用于打理应用程序中的数据存储任务，而这正是今天这篇文章要讨论的核心内容。

从广义上讲，Android应用中的数据存储选项共有五种主要类型：将数据保存在应用的共享偏好当中、保存在内部存储（专属于应用本身）当中、保存在外部存储（向设备公开）当中、保存在数据库当中以及保存在可通过设备互联网连接访问的Web资源当中。受篇幅所限，我们无法详细对这些选项作出论述，但会对每种方案的基础特性加以概括、从而帮助大家在需要使用持久化数据时理清存储问题的解决思路。

## 1. 共享偏好

### 第一步

共享偏好允许大家以键-值对的形式保存基本数据类型。应用程序的共享偏好文件通常被视为最简单的数据存储选项，但从本质上说它对于存储对象提出了一定程度的限制。大家可以通过它存储基本类型数字（如整数、长数以及浮点数字）、布尔值以及文本字符串。我们需要为自己保存的每个数值分配一个名称，从而在应用程序运行时据此对其进行检索。由于大家很可能在自己创建的第一款应用中就用到共享偏好，因此我们人把它作为讲解的重点、以更为详尽的方式（相较于其它选项）进行表述，从而帮助各位巩固必要知识。

大家可以在自己的主Activity类中尝试这些代码，并在稍后运行本系列教程的应用示例时对其加以测试。在理想情况下，共享偏好应该可以符合应用程序中的用户配置选项，如同选择外观设置一样。大家应该还记得，我们曾经创建过一个简单的按钮，用户点击它之后屏幕上会显示出“Ouch”文本内容。现在让我们假设自己希望用户在点击一次之后，该按钮上会持续显示“Ouch”字样，且该状态在应用程序运行过程中始终保持不变。这意味着按钮上的初始文本仅在用户首次点击操作之前存在。

让我们为应用程序添加共享偏好内容。在该类的起始位置、onCreate方法之前，我们为共享偏好选择一个名称：

```
public static final String MY_APP_PREFS = "MyAppPrefs"
```

利用“public static”修饰符，我们可以访问处于应用内任何类中的这项变量，因此我们只需要将偏好名称字符串保存在这里即可。我们使用大写是因为该变量属于常数，“final”修饰符也是因此而存在。每一次检索或者在应用程序偏好当中设置数据条目时，大家都必须使用同样的名称。

## 第二步

现在我们来编写共享偏好内容。在我们的onClick方法中、按钮“Ouch”文本设置部分的下方，尝试通过名称取回这条共享偏好：

```
SharedPreferences thePrefs = getSharedPreferences(MY_APP_PREFS, 0);
```

大家需要为“android.content.SharedPreferences”类添加一条导入。将鼠标悬停在“SharedPreferences”文本上方，并利用Eclipse提示完成导入。第一项参数是我们所定义的偏好名称，第二项则是我们作为默认选项的基本模式。

现在我们需要为共享偏好指定一套编辑器，从而实现对其中数值的设定：

```
SharedPreferences.Editor prefsEd = thePrefs.edit();
```

现在我们可以向共享偏好当中写入值了：

```
prefsEd.putBoolean("btnPressed", true);
```

这里我们使用了布尔类型，因为当前状态只分为两种——用户已经或者尚未按下按钮。编辑器提供多种不同类型，我们可以从中选择以保存这套共享偏好，其中每种方法都拥有自己的名称与值参数。最后，我们需要提交编辑结果：

```
prefsEd.commit();
```

## 第三步

现在让我们利用已经保存的值来检测用户运行应用程序后，按钮应该显示什么

样的内容。在onCreate中的现有代码之后添加共享偏好：

```
SharedPreferences thePrefs = getSharedPreferences(MY_APP_PREFS, 0);
```

这一次我们不必使用编辑器，因为我们只需要获取一个值：

```
boolean pressed = thePrefs.getBoolean("btnPressed", false);
```

现在我们利用已经设置过的名称检索该值，并读取变量中的结果。如果该值尚未被设置，返回的则为第二项参数，也就是默认值——代表否定含义。现在让我们使用该值：

```
if(pressed) theButton.setText("Ouch");
```

如果用户在应用程序运行之后按下该按钮，则按钮直接显示“Ouch”字样。在本系列的后续文章当中，大家会看到我们在应用运行中进行这一操作的情况。这个简单的例子很好地诠释了共享偏好的使用过程。大家会发现，共享偏好在帮助应用程序通过外观及使用感受迎合用户喜好方面具有重要的作用。

## 2. 私有内部文件

### 第一步

大家可以将文件保存在用户设备的内部以及外部存储当中。如果将文件保存在内部存储中，Android系统会将其视为专属于当前应用的私有数据。这类文件基本上属于应用程序的组成部分，我们无法在应用程序之外直接对其进行访问。再有，如果应用程序被移除、这些文件也会同时被清空。

大家可以利用以下输出例程在内存存储中创建一个文件：

```
FileOutputStream fileOut = openFileOutput("my_file", Context.MODE_PRIVATE);
```

大家需要为“java.io.FileOutputStream”类进行导入添加。我们提供了文件名称与模式，选择私有模式意味着该文件将只能被该应用程序所使用。如果大家现在就把这部分代码加入到Activity当中，例如onClick方法中，Eclipse将

弹出错误提示。这是因为当我们进行输入/输出操作时，应用程序可能遭遇一些需要应对的错误。如果大家的输入/输出操作无法解决这类错误，Eclipse就会提示异常状况、应用程序也会中止运行。为了保证应用程序在这种情况下仍能正常运行，我们需要将自己的输入/输出代码封装在try代码块当中：

```
try{
    FileOutputStream fileOut = openFileOutput("my_file", Context.MODE_PRIVATE);
}
catch(IOException ioe){
    Log.e("APP_TAG", "IO Exception", ioe);
}
```

如果输入/输出操作导致异常，那么catch块中的上述代码就会付诸执行，从而将错误信息写入到日志当中。大家今后会经常用到应用程序中的Log类（导入‘android.util.Log’），它会记录代码执行时所发生的具体情况。我们可以为字符串标签定义一个类变量，也就是上述代码中的第一条参数。这样一旦出现错误，大家就可以在Android LogCat中查看异常信息了。

## 第二步

现在回到try块，在创建了文件输出例程之后，大家可以尝试将以下代码写入文件：

```
String fileContent = "my data file content"
fileOut.write(fileContent.getBytes());
```

在将所有必要内容写入数据文件之后，利用以下代码作为结尾：

```
fileOut.close();
```

## 第三步

当大家需要检索内部文件中的内容时，可以通过以下流程实现：

```
try{
    FileInputStream fileIn = openFileInput("my_file");
    //read the file
```

```

    }
    catch(IOException ioe){
        Log.e("APP_TAG", "IO Exception", ioe);
    }

```

在try块当中，利用利用缓冲读取器读取文件内容：

```

InputStreamReader streamIn = new InputStreamReader(
    fileIn);
BufferedReader fileRead = new BufferedReader(stream
In);
StringBuilder fileBuild = new StringBuilder("");
String fileLine=fileRead.readLine();
while(fileLine!=null){
    fileBuild.append(fileLine+"\n");
    fileLine=fileRead.readLine();
}
String fileText = fileBuild.toString();
streamIn.close();

```

大家不要被其中所涉及的大量不同对象所吓倒，这其实属于标准的Java输入/输出操作。其中的while循环会在文件中的每一行执行一次。在执行完成后，“fileText”变量将把文件内容保存为字符串、以备我们直接使用。

### 3. 公共外部文件

#### 第一步

只要用户设备支持，我们的应用程序也可以将文件保存在外部存储当中。外部存储种类繁多，包括SD卡、其它便携式介质或者用户无法移除但被系统认定为外部类型的内存存储机制。当我们将文件保存在外部存储中时，其内容将完全公开、大家也无法以任何方式阻止用户或者其它应用对其进行访问。

在我们尝试将数据保存在外部存储中之前，必须首先检查对应存储机制是否可用——尽量避免意外状况绝对是种好习惯：

```

String extStorageState = Environment.getExternalStorageState();

```

系统会将信息以字符串的形式返回，大家可以对其进行分析、并与

Environment类中的外部存储状态字段加以比对：

```
if(Environment.MEDIA_MOUNTED.equals(extStorageState)) {
    //ok to go ahead and read/ write to external storage
}
else if(Environment.MEDIA_MOUNTED_READ_ONLY.equals(extStorageState)) {
    //can only read
}
else {
    //cannot read or write
}
```

即使设备上确实存在外部存储，我们也不能先入为主地假定应用可以向其写入数据。

## 第二步

在证实了我们确实能够向外部存储写入数据之后，大家接下来需要检索目录以指定文件保存的位置。以下应用程序设置内容指向八级及更高API：

```
File myFile = new File(getExternalFilesDir(null), "MyFile.txt");
```

这样大家就可以对该文件进行写入与读取了。不过也别忘了在项目的清单文件中添加以下仅限：

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

随着我们开发的应用程序变得愈发复杂，大家可能希望将自己保存得到的文件与其它应用共享。在这种情况下，大家可以使用公共目录下的各类通用条目，例如图片以及音乐文件。

## 4. 数据库

随着我们的应用程序所涉及的复杂结构数据越来越多，共享偏好或者内部/外部文件可能已经无法满足实际需求，这时候大家就应该考虑使用数据库方案

了。Android支持开发人员在应用程序内部创建并访问SQLite数据库。在我们创建一套数据库时，其将作为私有组件服务单纯服务于相关应用程序。

在Android应用中利用SQLite数据库的方法多种多样，推荐大家使用扩展SQLiteOpenHelper的类来实现这方面需求。在该类当中，我们需要定义数据库属性、创建各种类变量（包括我们所定义的数据库列表名称及其SQL创建字符串），具体代码如下所示：

```
private static final String NOTE_TABLE_CREATE =
    "CREATE TABLE Note (noteID INTEGER PRIMARY KEY
    AUTOINCREMENT, " +
    "noteTxt TEXT);";
```

这里所举的例子只涉及一套非常简单的表格，其中包含两列，一列内容为ID、另一列内容为文本；两列都用于记录用户注释信息。在SQLiteOpenHelper类当中，大家可以重写onCreate方法来创建自己的数据库。在应用程序的其它部分当中，例如Activity类中，大家可以通过SQLiteOpenHelper实现对数据库的访问，并利用WritableDatabase方法插入新记录、利用getReadableDatabase方法来查询现有记录，而后将结果显示在应用程序UI当中。

在对查询结果进行迭代时，我们的应用程序将使用Cursor类——该类会依次引用结果集中的每一行内容。

## 5. 互联网数据

很多应用都会使用互联网数据资源，而且某些应用甚至基本是由一套界面与大量Web数据源所构成。大家可以利用用户设备上的互联网连接来存储并检索来自Web的数据，只要网络连接有效、这一机制就能正常运作。为了实现这一目标，我们需要在自己的清单文件中添加“android.permission.INTERNET”权限。

如果我们希望自己的应用能够从互联网中获取数据，则必须保证这一流程脱离应用主UI线程。利用AsyncTask，大家可以通过后台进程的方式从Web源获取数据、在数据下载完成后将结果写入UI、最后让UI正常执行自身功能。

大家还可以将一个内部AsyncTask类添加到Activity类当中，并在需要获取数据的时候在该Activity中创建一个AsyncTask实例。通过在AsyncTask中引入doInBackground与onPostExecute两种方法，大家可以检索Activity中所获取



到的数据并将其写入用户界面。

获取Web数据在应用开发工作当中属于中等难度的任务，大家最好在熟练掌握了Android开发知识之后再尝试。不过大家可能很快就会发现，这样的数据获取机制对不少应用都非常适合，因为这能有效利用用户设备的连接资源。Java与Android都提供相关工具，用于处理返回的结构化数据——例如JSON feed。

## 结论

在今天的文章中，我们基本了解了开发Android应用程序时需要接触到的数据存储方案。无论大家最终选择哪种方案，都应该以实际需求作为参考标准，因为不同的方案只适合特定需求。在本系列教程的下一篇当中，我们将共同探讨如何将物理设备与已安装的Eclipse相连、同时学习如何创建虚拟设备。在此之后，我们还将探索如何让应用程序运行在这两种类型的设备之上。顺便向大家报告，再有两篇文章本系列教程就将彻底结束；在最后一篇文章中，我们将研究通用类以及Android Activity生命周期，从而帮助大家做好开发应用程序的一切准备。

# 第十一章 虚拟与物理设备

在之前的文章里，大家已经了解了Android项目当中的基本元素、接触了用户界面的设计以及数据存储方案。接下来，我们将一同探索如何在物理及虚拟设备上运行自己的应用程序并与之互动。在系列文章的下一篇中，我们将分步讲解如何让应用程序运行在物理设备及模拟器当中。而在今天的教程里，我将带大家先来学习如何在Eclipse中设置物理与虚拟设备。

当大家开发将要公开发布的Android应用程序时，必须提前在物理实机与模拟器中对自己的产品进行测试。在模拟器方面，我们可以通过配置让虚拟设备拥有各种硬件及软件功能。虽然这样处理的效果不可能像真正在各种物理设备上那么可靠，但也足以帮助我们了解手头设备与外部可能接触的Android设备之间的差异。某些特定硬件与软件功能在模拟器中无法实现，但大家仍然可以在其中测试自己第一款应用程序中的大部分功能。

## 1. 硬件设备

### 第一步

当大家开发Android应用程序时，应该首先关注成果在物理硬件设备上的运行情况。除了能够切实为我们带来应用程序外观、使用感受以及用户功能之外，硬件设备也是我们测试特定功能的唯一途径，例如通话。如果大家在开发过程中手边正好有一台硬件设备，也可以通过配置让模拟器拥有与其完全一致的硬件及软件功能，从而做到边开发边运行测试。

让我们首先将硬件设备与Eclipse相连。连接Android设备与计算机的就是大家都很熟悉的USB接口啦。我们可能需要在设备上启用USB调试，具体步骤为打开设备的设置屏幕、选择“开发者选项”、“等级设置”或者“应用程序”，然后选择“开发”。接着勾选USB调试项目。如果大家使用的设备上运行着Android 4.2或者更高版本，则可能需要通过设置让开发者选项正常显示。打开“关于手机”，然后在列表中重复多次（七次）点击“内部版本号”，最后返回之前的屏幕。

提示：大家可能还需要让自己的系统在Android设备接入时对其进行自动检测；没错，大多数情况下这一检测会默认进行，但我们还要需要防范万一。如果大家的Windows系统检测不到对应设备，请[下载](#)并安装USB驱动程序。如果大家使用的是Linux并在检测设备时遇到了麻烦，则可能需要使用udev文件并查看其中列出的设备制造商。如果仍然搞不

定，请 [点击此处](#) 查看Android官方开发者指南中的对应说明。

## 第二步

一旦系统检测到了我们接入的Android设备，大家就可以在Eclipse中与其进行交互了。在下一篇教程中，我们将详细讲解如何处理这些工作，请大家安心期待。现在，我们只需要切换至DDMS视图。在Eclipse当中，选择“窗口”、“打开视图”然后选择“DDMS”。大家应该会在屏幕左侧的设备视图中看到自己接入的设备。另外，大家还会看到LogCat视图开始弹出消息，提示对该设备的处理正在进行。

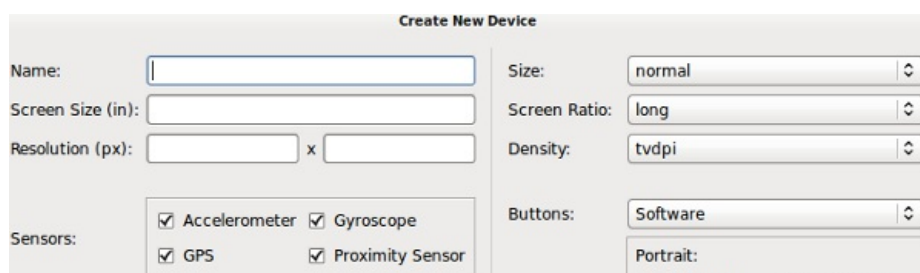
接下来请大家花点时间，在设备视图中选定自己的设备后、认真通过文件浏览视图查看其中的内容。我们将看到设备上保存的各文件及文件夹。如大家所见，这些视图允许我们任意使用接入的设备，包括在上面运行、测试以及调试自己的应用程序。大家还可以使用设备视图中的各个按钮，其中包括截屏按钮——当万事就绪之后，各位可能希望截取几个精彩瞬间作为应用的介绍素材。

## 2. 虚拟设备

### 第一步

现在让我们看看如何创建虚拟设备。在Eclipse中，选择“窗口”并点击“Android虚拟设备管理器（简称AVD管理器）”以将其开启。AVD管理器当中将显示两个选项卡，一个用于显示我们创建并启动的虚拟设备、另一个则用于管理可重新使用的设备定义。在未来的开发过程中，大家可能需要通过配置创建自己的AVD，从而有针对性地对应用的某种功能加以测试；但作为初期学习，我们会发现直接使用现有设备定义显然更快更方便。现在切换到“设备定义”选项卡当中。

大家会在AVD管理器当中看到一份设备定义清单，通过这种方式，我们可以保存设备配置、以备今后重复使用。大家也可以通过点击“新设备”按钮创建自己的设备定义。点击之后，我们会看到如下图所示的界面。



Cameras: ☐ Front ☒ Rear

Input: ☐ Keyboard ☒ No Nav ☐ DPad ☐ Trackball

RAM:  MiB

Device States:

- ☒ Enabled ☒ Navigation
- Landscape: ☒ Enabled ☒ Navigation
- Portrait with keyboard: ☒ Enabled ☒ Navigation
- Landscape with keyboard: ☒ Enabled ☒ Navigation

☐ Override the existing device with the same name

⚠ Please enter a name for the device.

Cancel Create Device

在这里，大家对虚拟设备的硬件和软件等进行全方位配置，其中包括屏幕尺寸、分辨率、传感器、摄像头、输入方式、像素密度以及按钮等。在创建了新设备之后，配置会显示在现有定义列表当中。大家应该尝试为自己配置的定义起个有意义的名称，这样我们才能更容易地在列表中将其找到。不过现在我们姑且使用已有定义来测试手中的应用，点击“取消”退出当前界面。

## 第二步

大家可以通过两种方式使用现有设备定义：直接复制当前设备定义并对属性进行修改，或者直接根据现有定义创建一个虚拟设备实例。在列表中选择一种设备并点击“Clone”（克隆）。

Clone Device

Name:

Screen Size (in):

Resolution (px):  x

Sensors: ☒ Accelerometer ☒ Gyroscope ☒ GPS ☐ Proximity Sensor

Cameras: ☒ Front ☐ Rear

Input: ☐ Keyboard ☒ No Nav ☐ DPad ☐ Trackball

RAM:  MiB

Device States:

- ☒ Enabled ☐ Navigation
- Landscape: ☒ Enabled ☐ Navigation
- Portrait with keyboard: ☒ Enabled ☒ Navigation
- Landscape with keyboard: ☒ Enabled ☒ Navigation

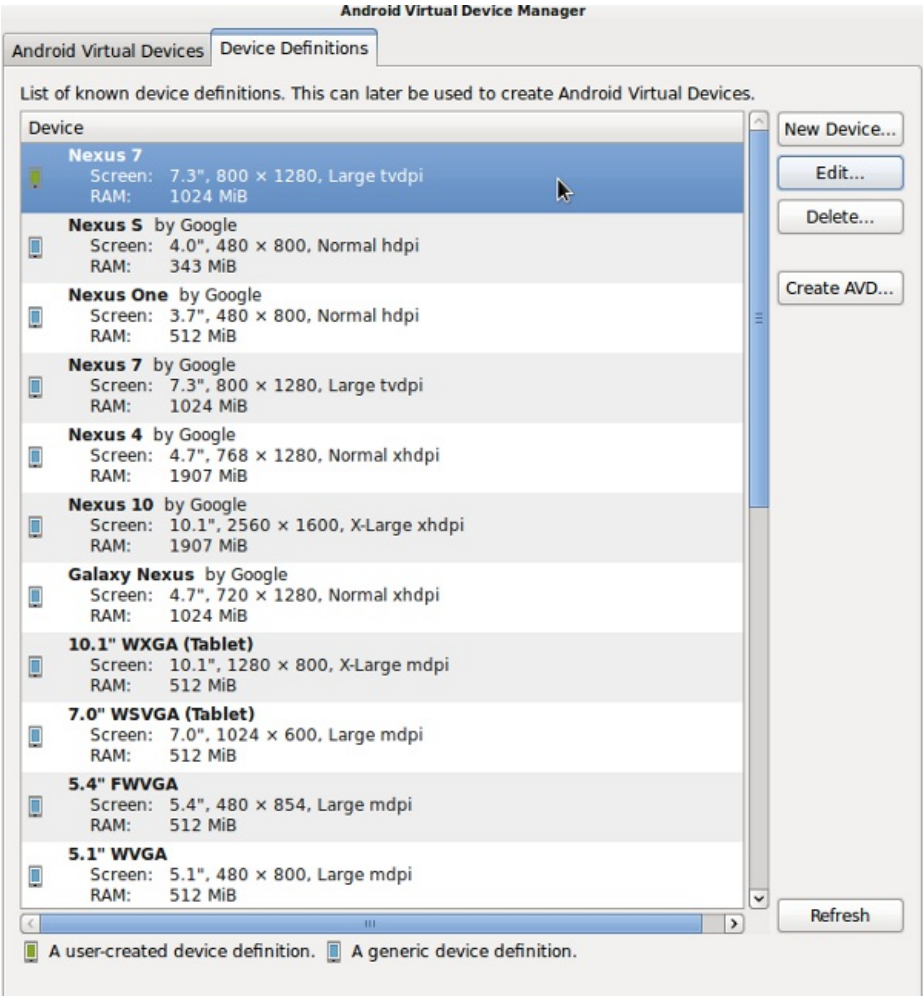
☒ Override the existing device with the same name

⚠ Only user created devices are editable.  
A clone of it will be created under the "User" category.

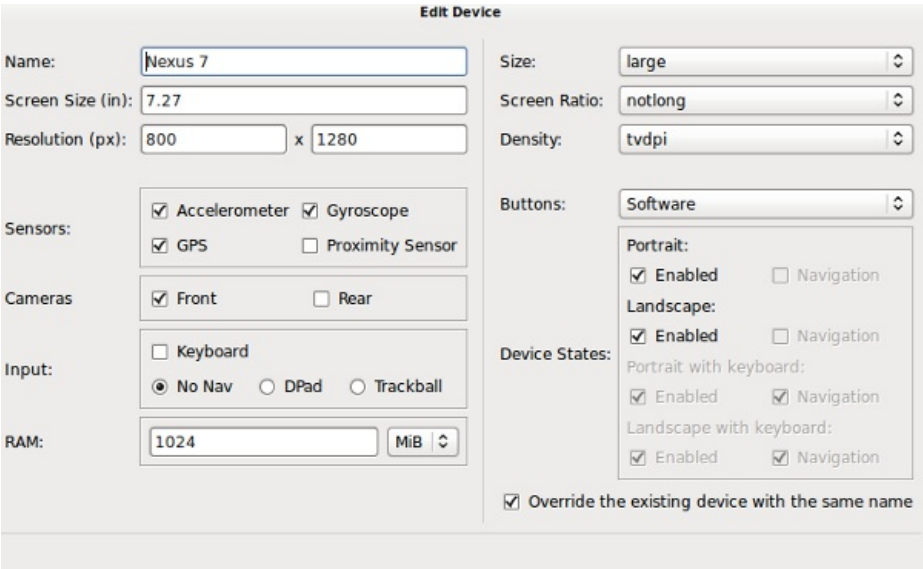
Cancel Clone Device

克隆设备窗口中的各输入框将被自动填充为与所选定义匹配的内容。现有设备的配置与我们的要求基本一致，因此所有设备属性都可以保留下来，直接点

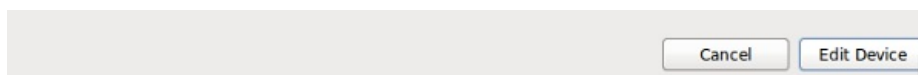
击“克隆设备”即可。这样列表中就会出现一套设备定义副本。



如大家所见，我们可以一目了然地通过Android设备定义与用户设备定义的颜色区别来判断哪些是默认方案、哪些是定制方案。大家只能对用户定义进行编辑，因此在列表中选择刚刚创建好的克隆设备并点击“Edit”（编辑）。



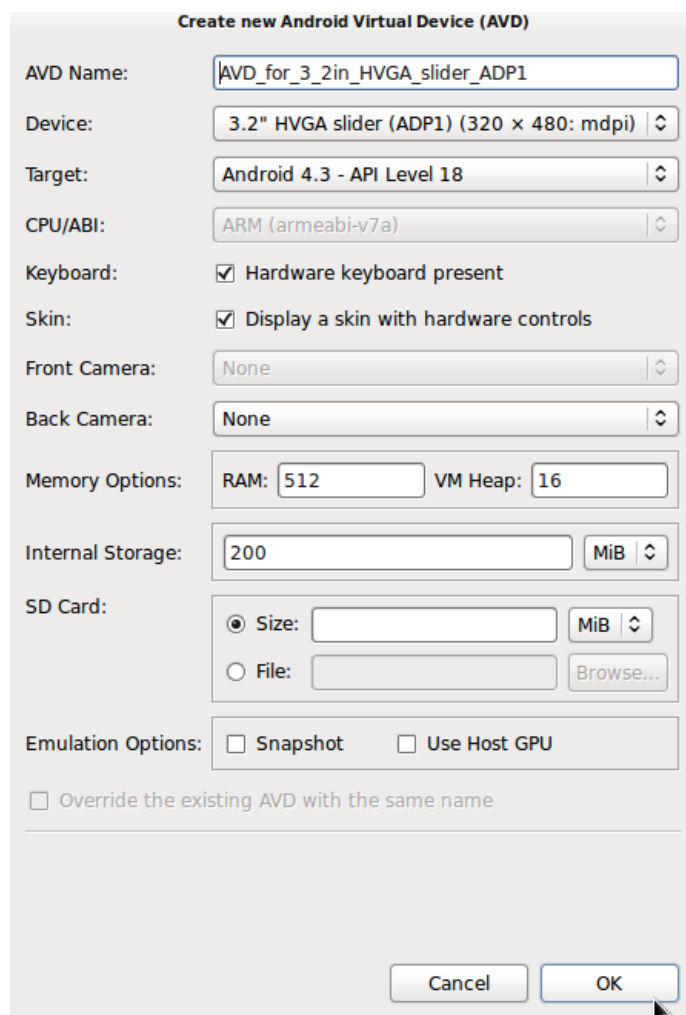




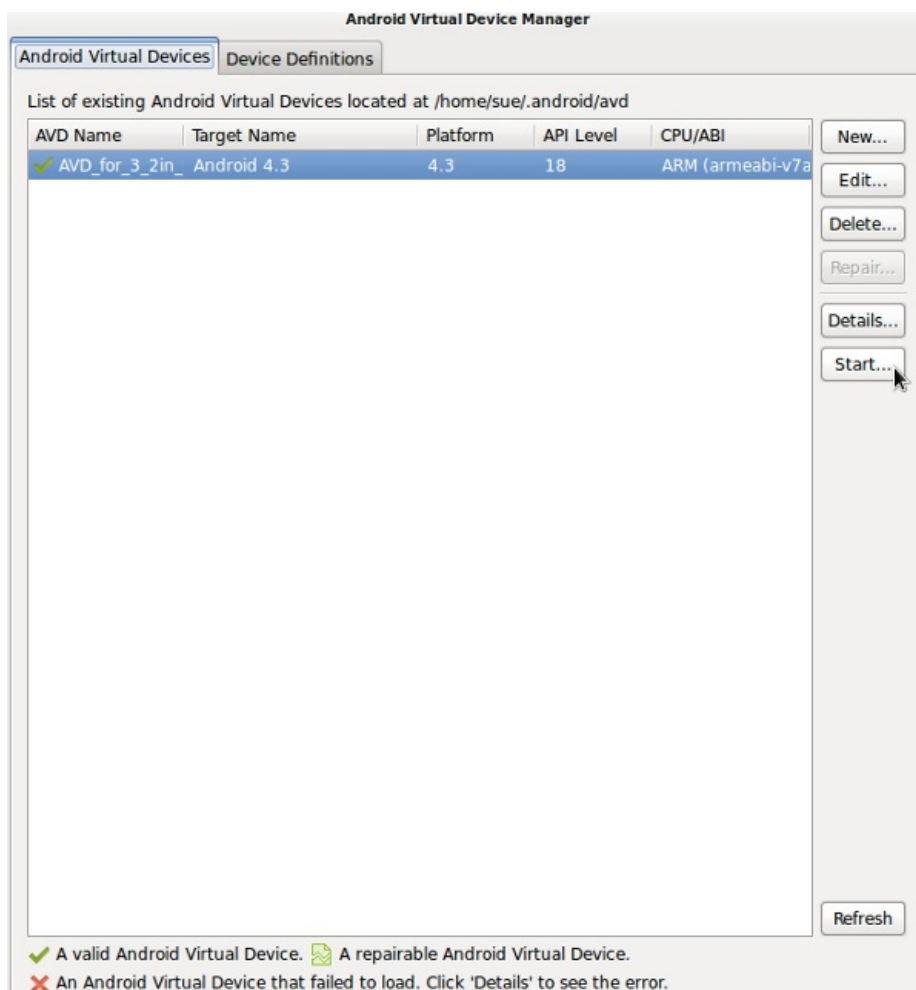
现在我们可以对设备的各项属性进行编辑了，完成后点击“编辑设备”以应用配置方案。现在大家可以根据列表中的设备定义创建AVD实例了。

### 第三步

只要根据现在设备定义创建AVD并将其运行在模拟器当中，大家就可以看到设备定义之一开始起效。从列表中选择一项设备定义并点击“创建AVD”。在窗口中，我们可以对实例进行配置，或者直接保留定义中的所有设定。关于AVD选项列表的具体情况，大家可以 [点击此处](#) 查看Android开发者指南中的相关说明。现在点击“OK”以创建AVD。



Eclipse会切换回Android虚拟设备选项卡，这时我们的新设备就会显示在其中。在选定了新AVD之后，点击“Start”即可将其投入运行。

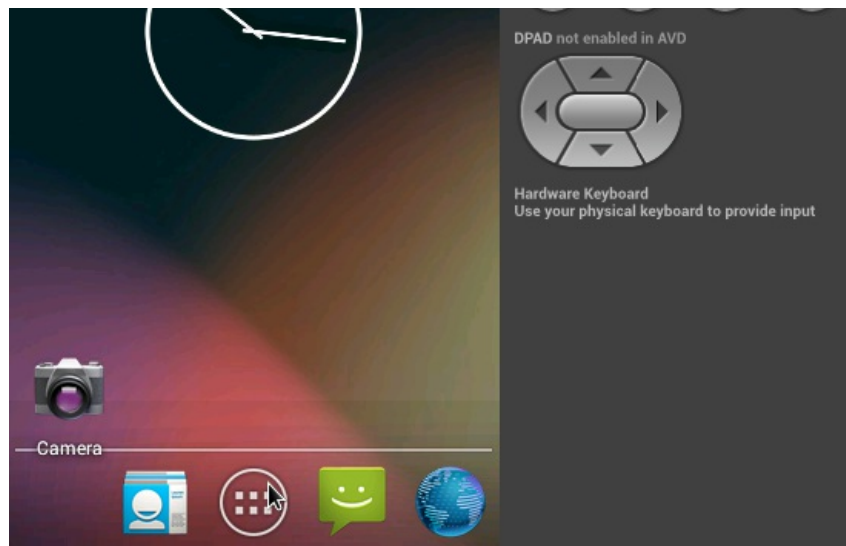


点击弹窗中的“Launch”，Eclipse将开始运行搭载着我们自定义AVD的模拟器。该设备可能需要几分钟才能完成启动，在设备已经开始运行后，大家就可以关闭AVD管理器了。

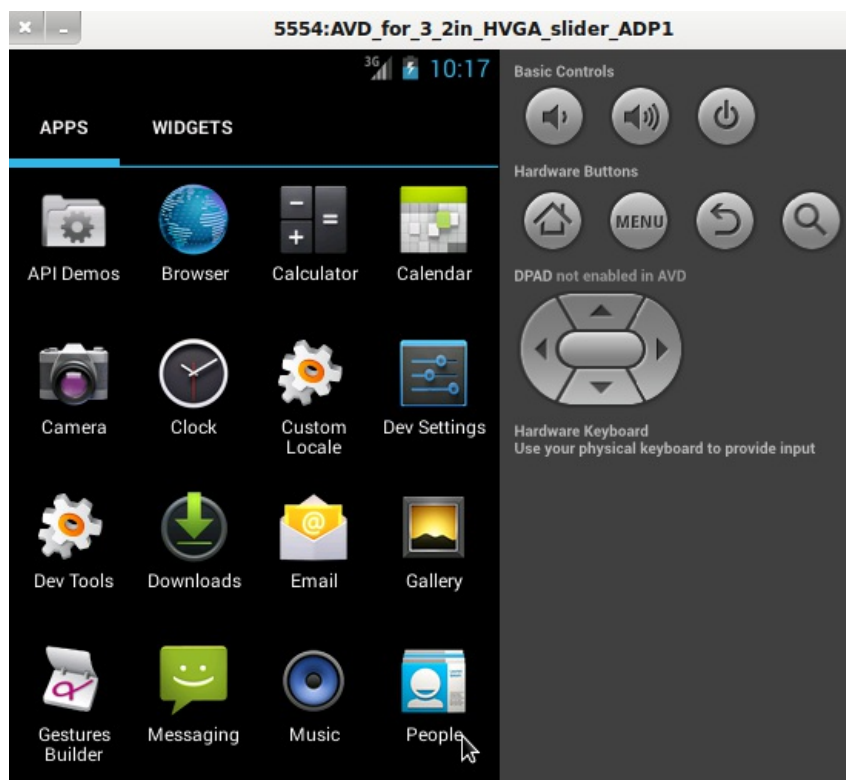
#### 第四步

在适当条件下，模拟器会显示设备硬件所控制的虚拟版本。大家可以通过鼠标点击与其进行交互。除此之外，模拟器还支持多种键盘快捷键组合，习惯之后能大大简化我们的日常操作——例如设备上的“Home”键对应键盘上的“Home”键。大家可以 [点击此处](#) 查看Android开发者指南中所罗列的模拟器快捷键清单。

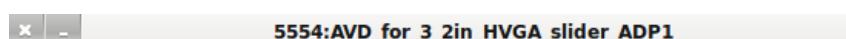




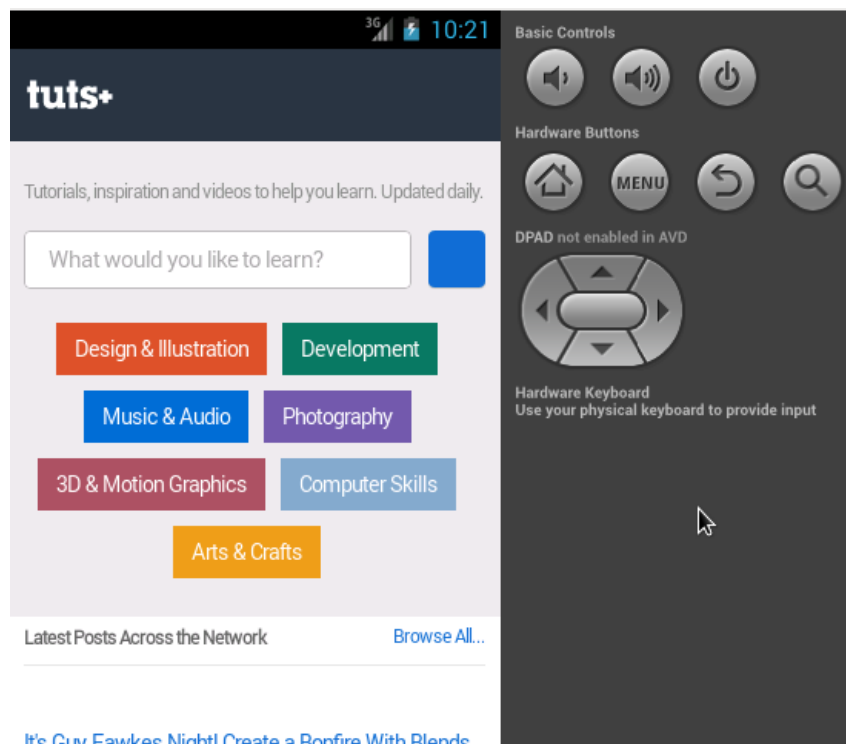
根据大家所创建的AVD，我们可能需要按下F2或者点击“OK”来解除锁屏状态。如大家所见，模拟器会显示出如物理设备一样的效果。现在请大家花点时间探索一下虚拟设备，查看一下应用程序菜单、再启动几个应用试试。



正面启动虚拟设备中的浏览器应用。点击导航栏并输入要访问的网址。大家可以使用计算机键盘作为输入设备。模拟器会自动使用任何计算机上可用的互联网连接，这样我们就可以在测试应用程序时评估其Web连接功能。

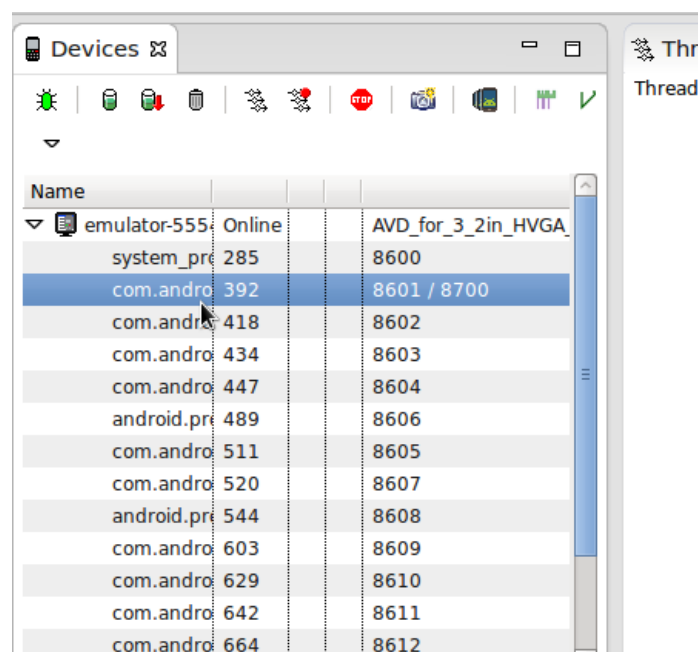


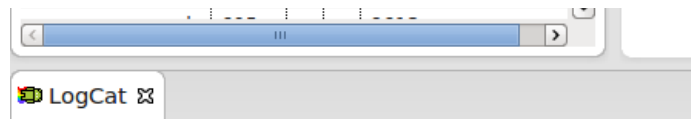




## 第五步

现在请保持AVD的运行状态，切换回Eclipse并再次打开DDMS视图。大家将在设备视图当中看到自己的虚拟设备已经出现在列表内，它旁边还会显示已经接入的物理设备。Eclipse将显示一份当前设备上运行着的所有进程的清单。选中某个进程后，我们就可以利用设备视图中的多个功能按钮对其进行处理。现在请大家花点时间认真观察这一界面，它们将在未来成为我们调试应用的好帮手。





## 总结

现在我们已经熟悉了在Eclipse中使用硬件或者模拟设备的整个流程。在下一篇教程中，我们将正式开始让应用程序运行在来自Eclipse的设备当中。在后续文章中，我们还会探讨Android通用组件与Activity周期，帮助大家明确未来学习的方向。在大家开发了几款应用程序之后，也可能会愿意回头再温习本系列教程。希望这几篇文章能成为朋友们在应用程序创建及设备交互过程中的指引与参考。

# 第十二章 运行与调试

在上一篇文章中，我们了解了如何将硬件与虚拟设备同Eclipse进行对接。而在今天的指南里，我们将探索如何通过Eclipse在物理设备及Android虚拟设备（简称AVD）上进行应用程序运行与调试。

首先需要强调一点：我们在之前文章中创建出的应用程序暂时还没什么实际用处，但我们可以通过它来体验应用的运行流程，并以此为基础介绍一些非常重要的Eclipse ADT实用程序。随着大家开发水平的提高，未来的新应用必然会变得更先进也更复杂，到那时我们现在所介绍的调试工作将扮演极为重要的角色。在完成了今天的指南后，大家可能希望花点时间对自己创建的应用作出调整，而后尝试将其运行在物理或者虚拟设备上。掌握了这种方法，大家就可以在开发过程中定期将应用程序半成品运行在设备之上，从而实现边开发边调试的理想效果。

## 1. 运行

### 第一步

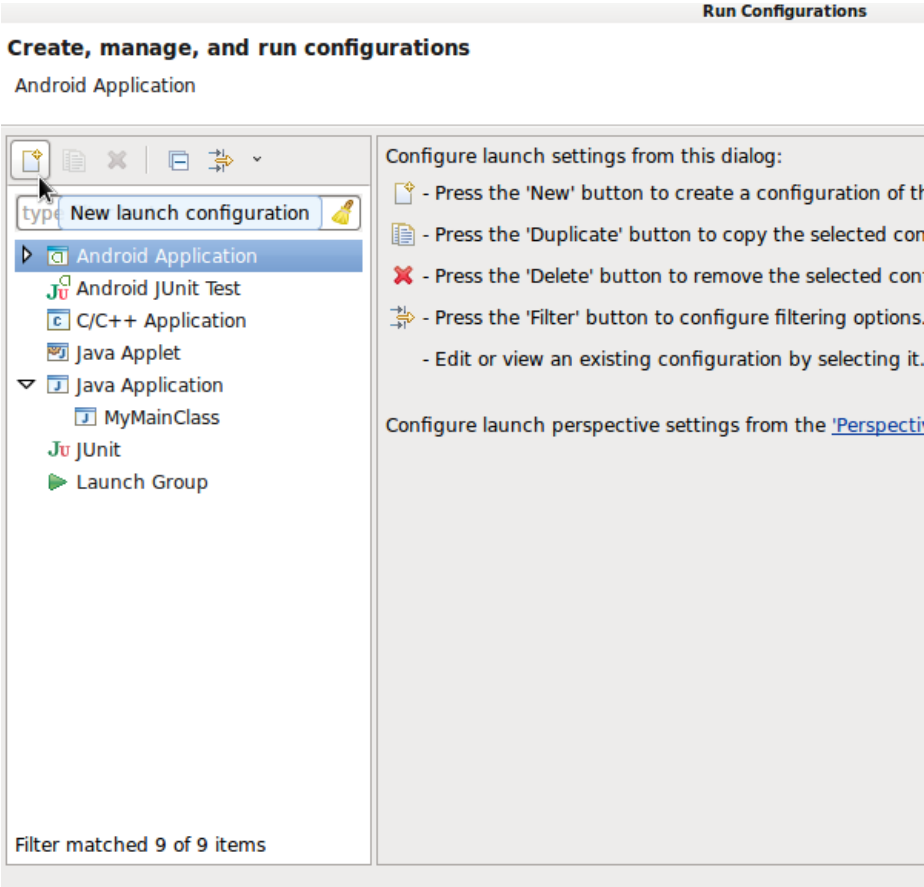
当我们在虚拟或者物理设备上编译并运行自己的Android应用时，Eclipse会处理大部分必要的细节工作。Eclipse以及ADT会为我们的应用创建一个APK文件，同时将其安装在我们所使用的设备上。APK文件也就是用户们从Google Play商店中所下载的应用文件格式。不过大家还需要进行额外一些步骤来进行应用程序的发布，这些内容我们将在之后的教程中一一说明。总之，Eclipse会首先建立一个用于调试的APK文件，我们可以通过这套IDE将其直接运行在设备上。

相信大家已经通过上一篇指南文章了解了如何利用相关技术启动自己的AVD或者将硬件设备与计算机相连。我们将让应用程序运行在接入的设备之上。无论是虚拟还是物理设备，都必须满足我们在应用程序清单当中所指定的最低API级别，否则将无法正常运行应用对象。

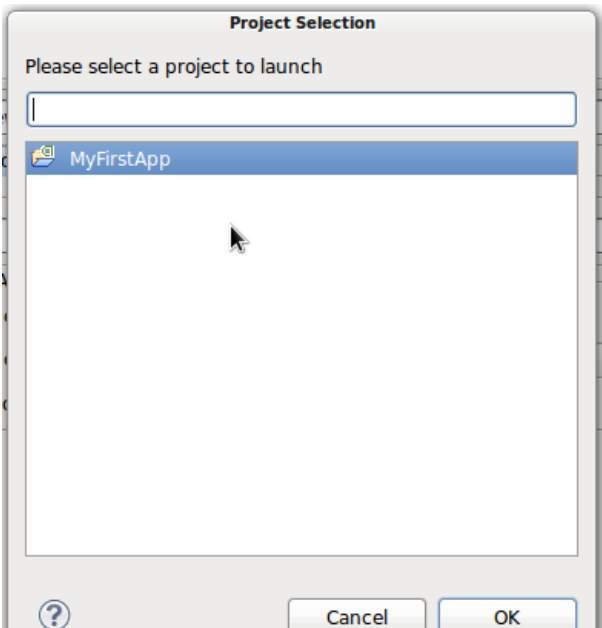
### 第二步

现在我们可以首先为自己的应用程序创建一套运行配置方案。在Eclipse当中，选择“运行”而后选择“运行配置”。在运行配置窗口左侧，大家将看到

可以运行的应用程序类型清单。在其中选择“Android应用程序”然后点击上方  
的新建按钮。

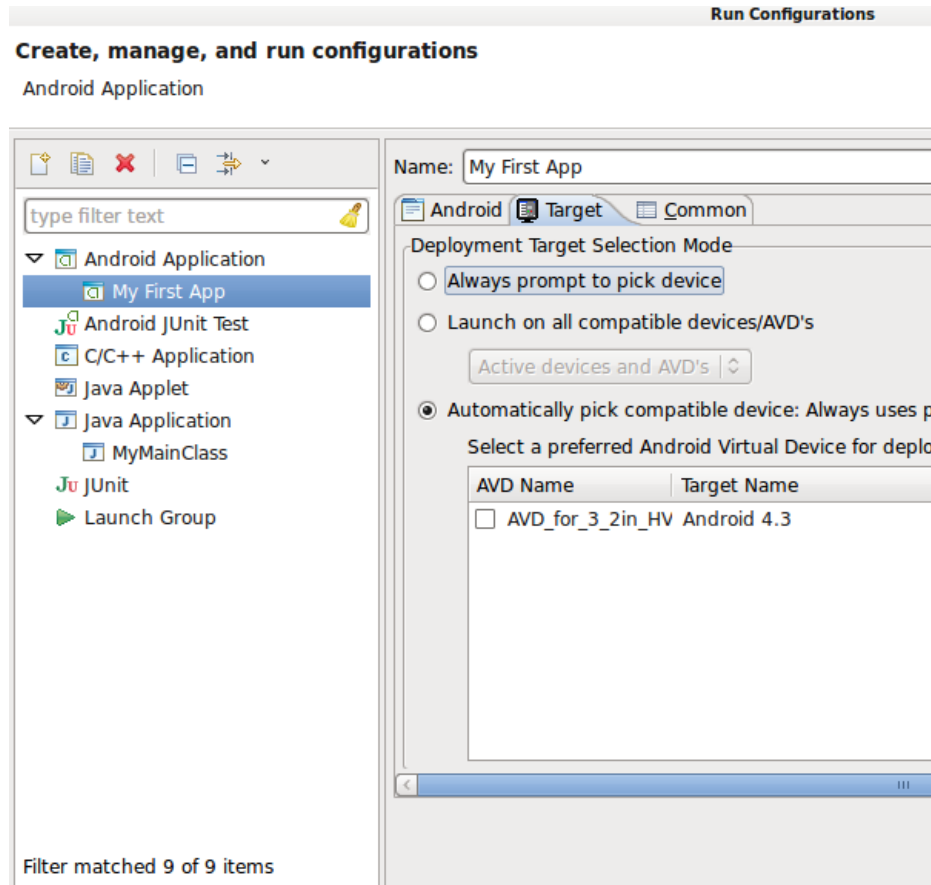


在打开的新配置项目当中，在名称栏内输入一个名称以替代现有文本。请大家  
选择一个清晰的名称，让自己能够明确区分不同应用程序。现在点击“浏  
览”按钮，在其中选择自己的应用程序项目，而后点击“OK”。





点击“目标”选项卡。大家可以让Eclipse自动选择要启动应用程序的设备，也可以将其设置为每次运行应用程序时都提醒用户手动选择。这套选项只适用于同时有多种设备接入Eclipse的情况，例如一台硬件设备与一套AVD。



请注意，我们的应用程序现在已经被列举在运行配置清单的Android应用程序当中。当一切准备就绪之后，大家可以从这里进行应用程序启动——相信在未来的实际工作中，各位的工作区内还将包含更多应用。

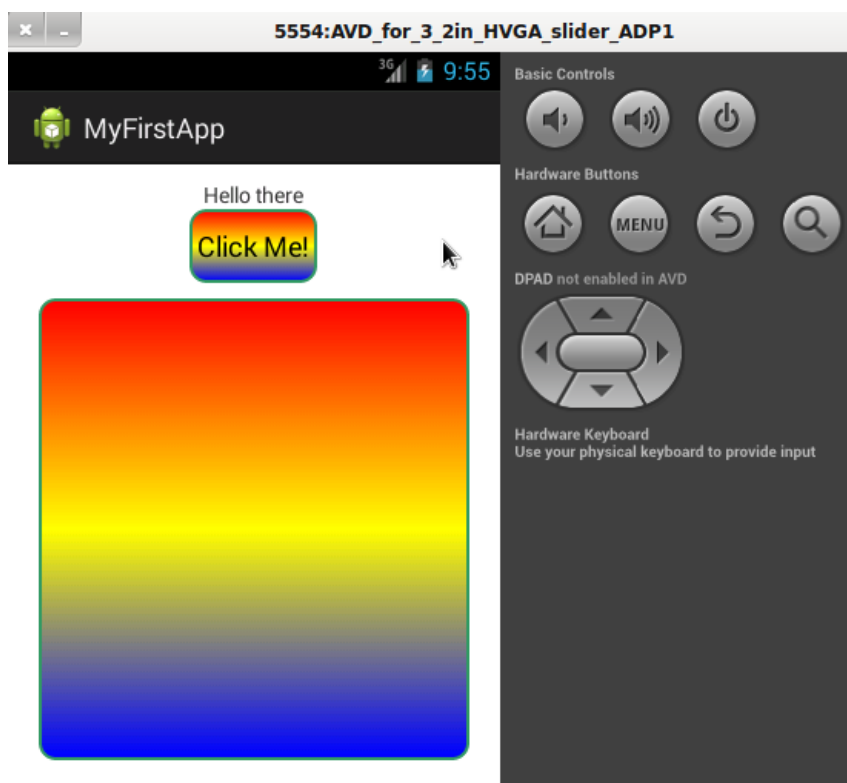
点击“运行”。如果大家在配置中设定了每次运行前提示用户选择一种设备，那么Eclipse这时就将提供对应选项。选择当前要使用的物理或者虚拟设备。如果我们没有接入设备或者运行AVD，但又在设定中要求Eclipse自动选择设备，则系统会启动一套适用于当前情况的AVD。大家也可以通过Eclipse工具栏中的“运行”按钮来启动上一次应用程序所使用的运行环境，这就省去了每一次打开运行配置窗口的麻烦。

提示：在应用程序运行配置的“目标”选项卡中，如果大家向下滚动则会看到多种模拟器选

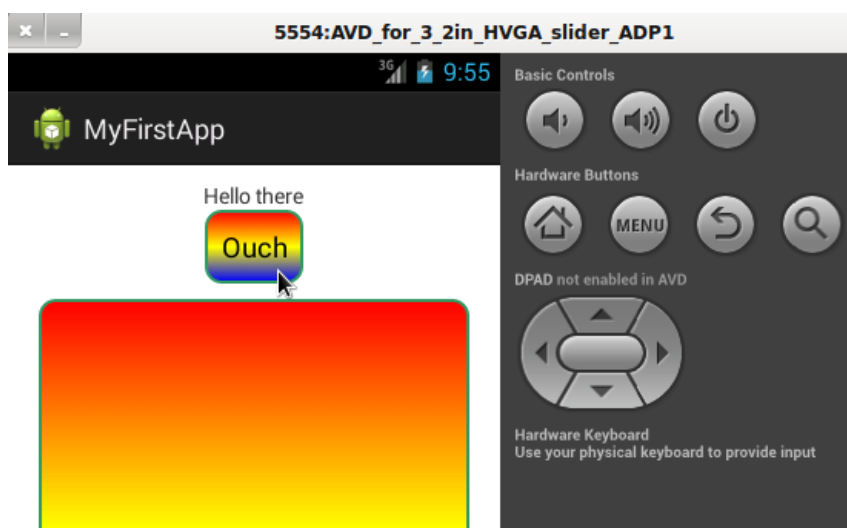
项，其中还包含命令行区域。大家可以点击此处查看使用AVD过程中可能需要的各种命令行参数。

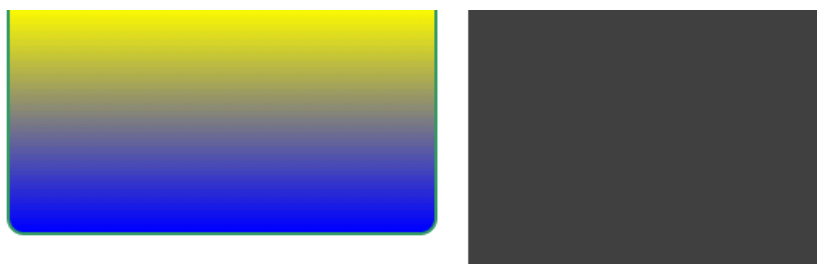
### 第三步

当我们运行自己的应用程序时，Eclipse会将APK复制到目标设备当中、进行安装并启动主Activity。



大家应该还记得我们当初在按钮上建立的这套基本用户交互机制：点击按钮来改变显示文本内容（在AVD当中使用鼠标模拟点击操作，在实机上则使用手指进行触控）。





## 第四步

在大家开发自己的应用程序时，很可能需要重复将当前成果加以进行、编辑内容然后再次运行。在这种情况下，Log将成为我们使用频率最高的主要工具之一。在Java文件当中，我们可以编写输出至LogCat的相关信息来帮助自己更顺畅地完成开发与调试工作。在我们的主Activity类中，将以下实例变量添加到原有类内容之前：

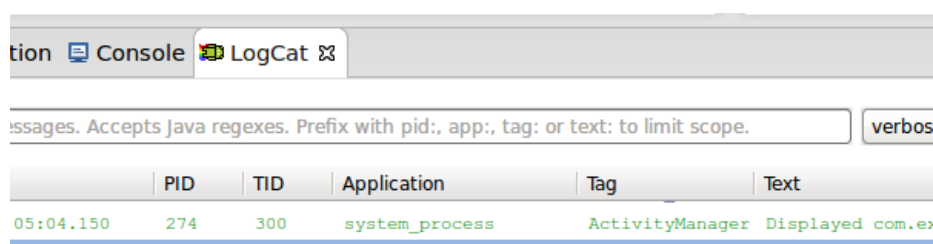
```
private final String LOG_TAG = "MainActivity";
```

这是一条标记常量，我们通常利用它来编写日志输出信息；通过类名称，我们可以更明确地看到当前日志信息来自哪个类。在onClick方法中，将以下代码添加到按钮文本设定部分之前：

```
Log.v(LOG_TAG, "button clicked");
```

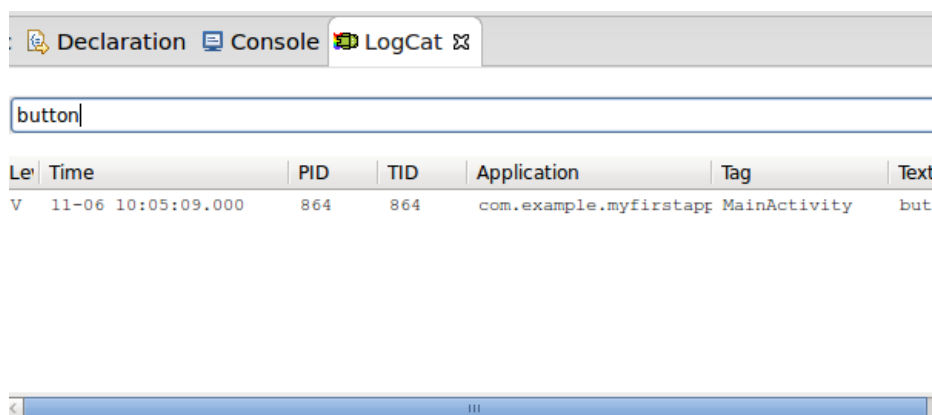
大家需要将“android.util.Log”导入到自己的类当中。在向Log中写入内容时，我们可以从多种方法中作出选择，从而表达与目的相符的对应信息。在上述代码中，我们用v来指代详细（verbose）。大家还可以用d来指代调试信息（debug message）、i指代信息（information）、w指代警告（warning）并用e指代错误（error）。

现在保存我们的文件并通过“运行”按钮再次运行应用程序。请大家确保自己已经在Eclipse当中打开了LogCat View。再次点击设备或者AVD上的UI按钮。现在向下滚动LogCat View直至我们找到对应的运行信息。



05:09.000	864	864	com.example.myfirstapp	MainActivity	Button clicked
05:09.020	274	337	system_process	SoundPool	error loading /s
05:09.020	274	337	system_process	AudioService	Soundpool could
05:09.020	274	337	system_process	SoundPool	error loading /s
05:09.020	274	337	system_process	AudioService	Soundpool could

正如大家所见，以上彩色信息反映每种情况下Log方法的实际运行情况。我们可以通过在文本框中输入内容的方式搜索信息，这一点在显示信息量较大的时候非常有用——例如使用硬件设备运行应用程序时。



提示：如果大家发现自己在使用LogCat View时Eclipse停止响应或者崩溃，则需要在“窗口”——>“偏好设置”——>“Android”——>“LogCat”中设置LogCat信息的最大缓冲数量。如果这样的调整仍然无法解决问题，请进一步降低缓冲数量并再试一次。

## 2. 测试

我们在今天的文章当中不会讨论太多测试方面的细节，因为这是一项非常重要也相当重复的工作、将成为大家未来Android学习过程中的主要课题之一。在准备好进行测试之后，大家可以在Eclipse当中创建一个测试专用项目。Android开发者指南当中专门提供了“[测试基本原理](#)”与“[Activity测试指南](#)”两个章节，从深层次讲解了Android平台上的测试知识。测试工具全部以JUnit为基础，作为汇聚专有Android测试资源并加以扩展的解决方案，JUnit专为Android开发工作而生。如果大家已经熟悉了对Java代码的测试工作，那么应该会更清楚自己需要在Android应用测试当中做些什么。

## 3. 调试

### 第一步

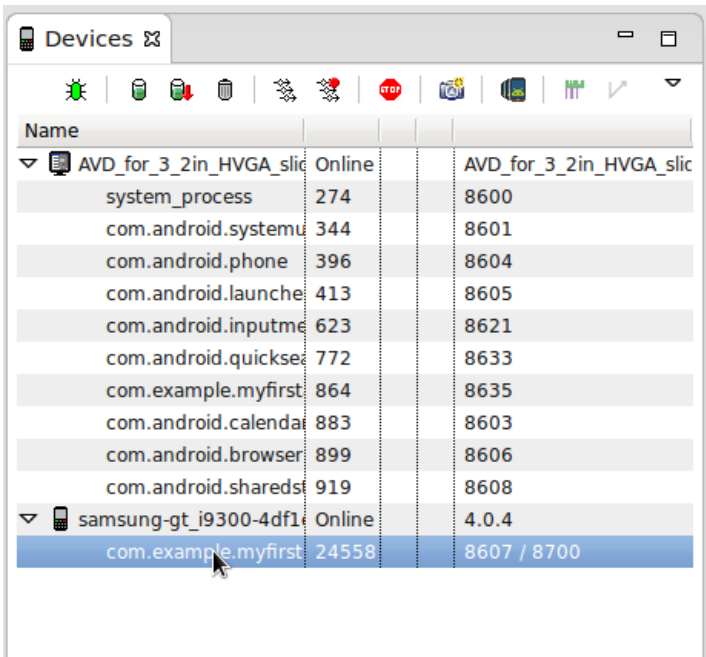


尽管大家并不需要马上对应用程序进行调试，不过我还是要通过今天的文章为大家初步介绍DDMS（即Dalvik调试监控服务器）以及这些工具能为开发流程带来的实际价值。我们已经了解了Eclipse当中的一种调试View，也就是LogCat；不过另外几种同样值得大家认真掌握。

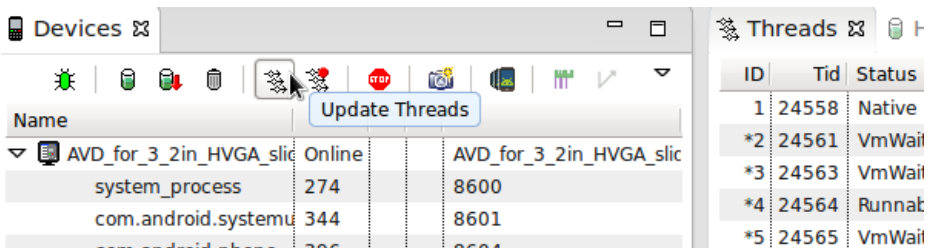
在运行在或者连接着Eclipse的虚拟设备或者硬件设备上，我们可以通过选择“窗口”、“打开视图”、“DDMS”的方式开启DDMS视图。在这里，大家能够利用Eclipse右上方的按钮在DDMS视图与Java视图之间进行切换。

## 第二步

让我们就DDMS视图展开探讨。上一次我们已经了解过设备视图（其中会显示一份清单，罗列所有已经连接的设备外加运行其中的进程链接）。选择一项进程后即可访问设备视图中的各项功能。下面通过名称从列表找出我们自己的按钮。



在选中对应进程之后，点击设备视图中的“Update Threads”按钮以开启Threads View。



com.android.phone	396		8604
com.android.launcher	413		8605
com.android.inputmethod	623		8621
com.android.quicksearch	772		8633
com.example.myfirst	864		8635
com.android.calendar	883		8603
com.android.browser	899		8606
com.android.sharedstorage	919		8608
▼ samsung-gt_i9300-4df1e	Online		4.0.4
com.example.myfirst	24558		8607 / 8700

\*6 24566 Wait

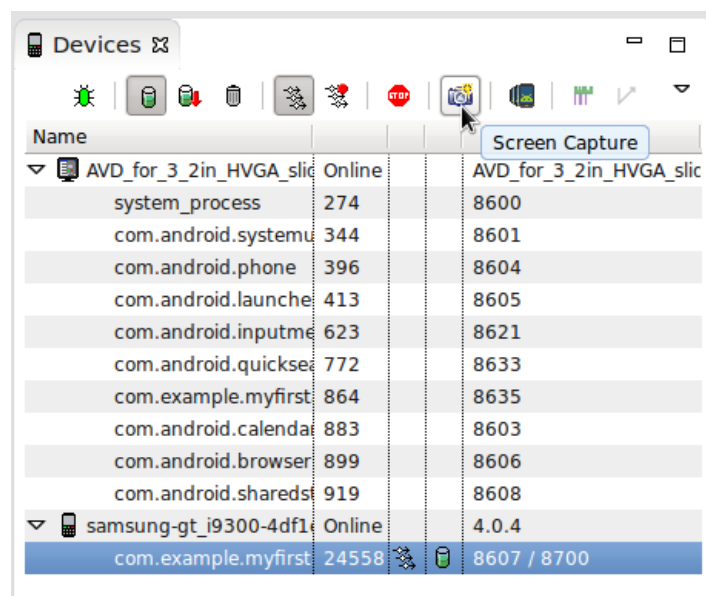
\*7 24567 Wait

\*8 24568 Wait

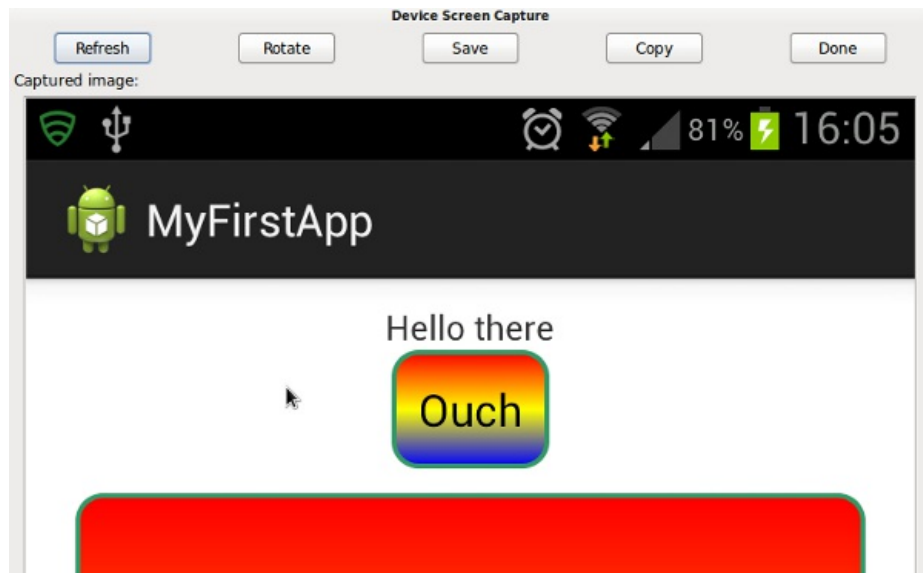
Refresh

“Update Heap”按钮对于Heap View也将起到同样的作用。在未来的开发工作当中，大家还会根据需要了解其它视图的作用，其中一些只与执行特定任务类型的应用程序相关。不过现在我们不考虑太多，只把时间用在DDMS当中值得关注的內容身上。

模拟器视图提供针对任何运行中虚拟设备实例的控制选项，大家可以在设备列表中选择一套AVD并打开其模拟器控制选项卡来查看相关内容。另外，需要注意的是我们可以在设备视图当中停止当前运行着的进程、调试运行中进程并强制执行垃圾回收。在我们结束对DDMS的说明之前，应用程序一定正运行在硬件或者虚拟设备之上，请在设备视图当中选择对应设备并点击“截图”按钮。



这时新窗口中将出现当前设备显示内容的截图结果。在这里，大家可以保存或者复制该图片，并将其用于应用程序商店中的宣传资料当中。受篇幅所限，我们对调试工作的说明比较简略，大家可以[点击此处](#)查看Android开发者指南当中关于Android应用程序调试的内容。



## 总结

当我们开始着手创建Android应用程序时，需要关注的重点在于运行应用程序并将信息记录到控制台以监控应用的运行活动。不过随着项目复杂程度的提升，大家往往会在应用程序发布前的准备阶段用到大量Eclipse当中不可或缺的Android调试与测试工具。在本系列教程的下一篇文章中，我们将介绍Android Activity生命周期，只有掌握了这方面知识、大家才能真正开始创建功能全面的应用程序。

# 第十三章 Activity与生命周期

Activity生命周期并不仅仅在用户运行应用程序之后才开始生效，事实上它也影响着用户切出以及切回应用时得到的不同反馈。当我们开发一款应用时，首先需要牢记一点：用户会经常在执行过程中、在我们的应用与其它应用之间频繁切换。取决于用户的操作方式，同一款应用程序有时在前台运行、有时则在后台运行。大家必须保证自己的应用能够就会这类情况，并在此类切换过程中及时保存并恢复数据。再次提醒各位，这一过程对于某些特定应用程序略有不同——例如功能性组件。

## 1. 回调方法

### 第一步

为了控制Activity处于不同状态下时应用程序的运行方式，例如当用户切出或者切回应用，大家可以选择多种处理方法。这类方法也就是 Activity生命周期回调方法。Android系统会在我们的Activity进入某种特定状态后调用这些方法，从而通过一系列步骤确保我们的应用程序能够继续起效、不至于丢失数据而且在用户不与之交互时不会使用非必要性资源。每一种回调方法都会让我们的应用进入一种可能的状态。

如果大家之前曾经接触过Java应用程序的编程工作，那么应该已经发现Android应用程序的启动遵循另一种方式。与Java应用直接使用主方法不同，Android在启动后会首先执行主Activity类中的onCreate方法。请记住，我们已经在清单中将该类指定为主启动Activity。Activity会首先回调onCreate方法，相当于重复用户启动应用程序后的流程。这时候onCreate方法会使应用程序进入Created状态。

开发者指南当中通过[示意图](#)以直观方式介绍了生命周期、回调方法以及状态的概念。其中onResume方法负责提供Resumed状态，这时我们的应用程序可以接受用户的直接操作。其它各类回调方法都以onResume为核心，即将应用程序引导至Resumed状态或者从该状态脱离、启动该状态或者将其停止。

对于大部分应用程序来说，我们只需要使用一部分回调方法，但最起码要用到onCreate。虽然使用频率不高，但了解全部回调及状态的作用将帮助我们了解自己的应用程序在运行及停止运行时，Android系统会受到怎样的影响。一

般情况下，大家需要保证用户能够在任何操作过程切换出去之后、都能顺利恢复到之前的运行状态；如果他们通过导航选择前进或者后退，应用则需保存全部必要数据并释放不必要占用的硬件资源。

## 第二步

我们的应用程序可能处于以下五种状态，分别为：Created、Started、Resumed、Paused以及Stopped。另有七种回调方法能够让应用进入或者脱离上述状态，它们分别是：onCreate、onStart、onRestart、onResume、onPause、onStop以及onDestroy。这些方法能够让我们的应用程序在可能的状态之间进行切换，而且某些情况下切换速度会很快。通常来说，大家可以认为自己的应用程序始终处于resumed、paused或者stopped这三种状态之下，因为其它状态都是暂时性的。

当我们的应用程序正处于运行当中且用户与之进行操作交互，这时的应用状态为Resumed；当另一个Activity处于前台但仅仅使我们的应用被部分隐藏时，这时的应用状态为Paused——在这种状态下用户无法再与应用进行交互。当我们的应用完全处于后台之下，而且用户既无法操作、也无法观看到它时，其状态即为Stopped。在这种状态下Activity会保留之前的所有数据，但无法加以执行。

## 2. 进入Resumed状态

如我们所知，主Activity会在应用程序启动时开始运行，onCreate方法也将执行、从而让我们准备该类所需要的Activity UI以及全部数据条目。我们创建的大部分应用当中都包含不只一个Activity，其它Activity会在用户与应用程序进行操作交互时启动。大家可以利用以下代码通过Intent类启动另一个非主Activity：

```
Intent aboutIntent = new Intent(this, About.class);
startActivity(aboutIntent);
```

这代表着应用程序包中另一个名为“About”的Activity类。大家可以通过选择自己的源码包而后选择“文件”、“新建”、“类”的方式在Eclipse当中创建一个新Activity，而后将该Android Activity类选定为超级类。请记住，每一个Activity都必须在我们的应用程序清单当中列出。大家还可以利用

Intent类实现不同 Activity之间的数据转移。

当一个Activity处于运行当中时， onCreate方法也在同时执行， 因此除了把其它Activity类列入清单之外、大家也能够以与主 Activity类似的方式在应用程序当中处理这些类。我们也可以为每个Activity创建一个布局文件， 并通过设置让其使用与主Activity同样 的技术机制。

在某个Activity的onCreate方法开始执行之后， onStart与onResume两个方法也将开始执行， 从而使该Activity处于Resumed状态、并在后续执行过程中根据情况转换为Created以及Started状态。

我们的Activity可以通过不只一种方式进入Resumed状态， 应用程序启动只是其中最基本的途径。如果Activity处于Paused或 者Stopped状态， 则应用程序切换至当前之后该Activity将直接进入前台运行模式， 且无需重复调用 onCreate方法。如果大家的应用从 Paused状态切换回Resumed状态， 则Activity的onResume方法将开始执行。如果该应用由Stopped状态切换回运行状态， 则执 行onRestart方法、而后依次为onStart与onResume方法。

### 3. 进入Destroyed状态

#### 第一步

当我们的应用程序处于退出或者隐藏状态下， 则Resumed就会转变为Destroyed。这时候， onPause方法会将应用的Activity 由运行时的Resumed状态转换为Paused状态。在onPause当中， 大家应当停止任何需要占用资源的任务， 例如动画播放、传感器数据处理以及广 播接收等等。如果onPause正在执行， 那么onStop也可以开始执行， 因为用户此时通常已经通过导航退出了我们的应用程序。大家还可以利用 onPause方法进行数据保存——虽然通常来说数据保存工作由onStop方法来负责最为妥当。

正如我们之前曾经提到， 大家的Activity能够通过onResume方法从Paused状态重新回归至Resumed状态。这意味着我们可以利 用onResume来恢复任何我们之前在onPause当中停止或者发布过的内容。不过大家还需要记住一点， onResume在其它情况下也会付诸执行， 例如在应用程序启动时。

#### 第二步

在onPause之后， 如果应用程序进入Stopped状态， 那么onStop也将开始执行。

在这种情况下，onRestart、onStart以及onResume等方法仍然能够使应用程序重新回到Resumed状态。在onStop中，大家应当尽可能压缩只在必要数据的操作量，例如向数据库中写入内容。请大家确保在onStop当中囊括了所有应用程序所使用的资源，从而避免该应用在被彻底关闭之后导致内存溢出问题。

这套系统会在应用程序从resumed状态切换至stopped状态后保存特定数据，例如视图中需要显示的内容。当某个Activity从Stopped状态恢复到Resumed状态时，onRestart、onStart以及onResume方法都会开始执行。不过onStart与onResume的执行情况有所不同——例如在应用程序启动之时。而onRestart方法只会在应用程序从Stopped状态恢复至前台之后才会执行，这样大家就能利用它来恢复任何保存在onStop当中的运行内容。

提示：当大家从一个Activity之下启动另一个Activity时，前者会进入Stopped状态。如果用户随后利用后退按钮再次由后者返回先前的Activity时，那么前者的onRestart方法就会开始执行。

### 第三步

如果大家的应用程序即将彻底关闭，例如我们的当前Activity被从系统当中移除，则onDestroy方法会开始执行。尽管这是在我们的Activity完全消失之前执行的最后一个方法，大家仍然不应该简单地将所有内容一股脑清除。事实上，我们需要利用onStop或者onPause来处理结束工作。当然也有例外情况，如果应用程序的后台进程仍然处于运行状态，那么这时候大家应该在onDestroy当中将其停止。

在onDestroy执行之后，如果用户通过导航返回应用程序Activity，则对应onCreate方法将再次被启动。一般情况下，大家可以假设onPause与onStop会在onDestroy之前执行。不过如果大家明确调用finish方法来结束一个Activity，则只有onDestroy会被执行。

在多数情况下，我们并不需要为应用程序当中的生命周期回调问题投入过多精力，因为大家完全可以利用onCreate方法的参数实现数据保留效果。在Activity onCreate方法当中，Bundle参数负责如前所述自动进行视图信息保存。不过大家也可以利用该对象保存更多数据内容，例如记录用户与应用程序之间的交互所产生的变量更新。要实现这一目标，大家可以在Activity类当中使用onSaveInstanceState方法，完成数据键值对的编写之后、我们就可能在onCreate当中将其恢复。



提示：当用户改变设备显示模式时，也就是在纵向及横向模式间进行切换，我们的Activity实际上会经历重新创建、onCreate也会被再次执行。这一过程被我们称为配置变化。在这种情况下，系统会假设大家需要重新创建Activity，例如大家在每种显示模式下使用不同的布局方案。不过在多数情况下，大家可能不希望系统照此办理。为了避免我们的Activity在显示模式转换时发生重新创建，大家可以从两种解决方式中作出选择：向清单内的Activity添加“android:configChanges”属性，或者调整我们的Activity结构、利用我们在配置变量时所保留的Fragments。

## 总结

当大家开始学习如何为Android平台开发应用程序时，Activity当中所涉及的大量状态与回调方法可能会成为很多难题乃至混乱的根源。然而在大多数情况下，我们只需要采用最低数量的方法以确保自己的应用程序有能力提供用户所预期的功能与效果。在本系列教程的下一篇当中，我们将共同了解部分常用Android类，大家很可能会在自己的第一款应用当中与它们打交道。在此之后，我们将着眼于Android代码示例、需要了解的应用程序发布知识以及其它一些关于今后进一步学习的建议。



# 第十四章 Android组件详解

到目前为止，我们已经一同学习了Android应用程序中的结构与典型元素，其中包括用户界面元素以及数据存储。利用当下已经掌握的知识，大家完全可以着手创建自己的Android应用。不过在实际操作之前，我们还要梳理一遍部分常用Android组件——这也正是今天这篇文章的主要内容。在本系列的下一篇文章中，我们将探讨SDK示例代码。

Android应用程序当中包含四大组件：Activity、Service、Content Provider 以及Broadcast Receiver。只要大家创建或者使用其中的任何一种，就必须将对应元素添加到项目Manifest当中。我们之前已经跟Activity打了不少交道，因此在今天的文章中我就不再浪费篇幅加以介绍了。现在让我们把注意力集中在另外三种主要应用程序组件身上。需要强调的是，我还将介绍大家在应用程序当中最可能用到的其它一些资源，其中包括fragment以及action bar。

## 1. Service

在Android系统当中，一项Service就相当于一个后台进程。Service通常被用于那些正在进行或者需要持续很长一段时间的进程。事实上Service并不具备用户界面，因此它们通常需要与其它组件结合起来以实现功效，例如与Activity联手。最典型的例子就是，在应用程序当中Activity会在用户操作的同时启动一项Service，这项Service也许会将数据上传至Web资源当中。用户可以继续与该Activity进行交互，但与此同时Service的运作却不受影响——因为它的执行一直在后台完成。

提示：如果大家希望执行获取互联网数据之类的后台进程，其中也不一定非要使用Service类。根据应用实际需求的不同，大家可能更适合在自己的Activity中利用内部AsyncTask类来解决问题。它能让后台进程与用户界面彻底分离，但我们的Activity仍然可以接收来自AsyncTask的运作结果并将其上更新至用户界面当中。

要将一项Service组件添加到Manifest当中，我们需要利用以下语法及其元素取代应用程序元素：

```
<service android:name=".ServiceClassName" />
```

大家可以在Eclipse当中创建一个Service类，其过程与之前介绍过的Activity一样、只不过这次是把Service选择为超级类。Service与Activity组件的区别

我们之前已经谈到过，二者之间存在多种重大差异。如果我们以Activity为起点启动一项Service，而用户又通过导航从该Activity切换到了其它应用程序处，那么该Service仍将继续运行。因此，Service拥有的生命周期与我们所熟知的Activity完全不同，大家需要将这一点牢记于心、从而确保自己的应用程序能够有效运转。

其它应用程序组件可以与Service绑定，并向其请求并接收数据。如果一项绑定Service正处于运行当中，那么所有与之相绑定的组件停止之后它也将同时进入停止状态。尽管Service与应用程序的用户界面并无关联，但随Activity启动的Service会与之运行在同一线程当中。然而，如果大家的Service需要使用大量处理资源，那么我们最好为其创建一个独立的运行线程。请大家 [点击此处](#) 查看更多来自Android开发者指南中关于Service的说明。

## 2. Content Provider

另一种组件Content Provider用于管理数据集。这里指的数据集既可以单纯归属于我们的应用程序，也可以与其它应用所共享、能够对数据进行查询及修改。如果大家为自己的应用程序创建了一个用于管理数据的Content Provider，那么我们的UI组件（例如Activity）就能够使用该Content Provider，通常是利用Content Resolver类来实现与数据的交互。当被其它应用程序使用时，该Content Provider则通过标准方法来访问数据、从而与数据库等结构化数据集实现交互。

如果大家对于关系类数据库已经非常熟悉，那么应该非常了解Content Provider所使用的数据访问方法。数据会被Content Provider提交给一系列包含行与列的表格，其中每行（或者每条记录）中的列都包含一个单独的数据值。因此，处理通过Content Provider返回的数据与处理数据库查询结果非常相似。

虽然有时候我们可能会创建一个专门的Content Provider程序，但在最初着手开发自己的第一款应用时、大家基本上还是会直接使用由其他开发者或者Android系统本身所提供的Content Provider——例如设备日程表或者联系人。Content Provider能够定义客户端应用所需要的权限，从而实现内容使用。为了在应用程序当中使用Content Provider，大家需要在Manifest当中为其添加相应的权限。

提示：如果大家只是需要一套应用程序当中的结构化数据源，那么我们通常并不需要单

独创建Content Provider。大家可以创建一套只适用于我们应用程序本身的SQLite数据库，而完全不需要使用Content Provider类。我们需要创建Content Provider的惟一情况在于，大家希望从我们的应用程序当中将结构化数据复制到其它应用处，或者大家希望使用搜索框架。

### 3. Broadcast Receiver

Android系统能够发出多种应用程序能够响应的广播信息。大家也可以开发出应用程序来发出这些广播，但这种可能性与监听现有广播相比要低得多，至少对我们的第一款应用来说是这样。系统通知当中包含关于设备硬件的各类信息，例如电池电量、屏幕关闭以及充电器是否接入插座等等。

为了接收来自Android系统的广播通知，我们的应用程序需要使用Broadcast Receiver。举个典型的例子，电池电量工具会在电池电量发生实际变化时更新显示结果。在这种情况下，大家可以将Service类与Broadcast Receiver配合使用，从而保证应用程序始终在后台监听通知内容。

Android系统将广播通知称为intent，它能够被用于启动Activity。作为系统执行的操作，intent可以实现Activity启动或者发出通知。要使用Broadcast Receiver，我们的应用程序必须在Manifest中对其作出声明；此外还有一套备选intent filter，用于指示我们想要接收的操作：

```
<receiver android:name=".YourReceiver">
  <intent-filter>
    <action android:name="android.intent.action.BATTERY
      _LOW" />
  </intent-filter>
</receiver>
```

通过以上代码，我们应用中的Broadcast Receiver就能够接收“battery low（电池电量低）”这一intent。请注意，我们无法通过在Manifest中声明action这种方式接收全部系统通知。在某些情况下，大家必须完成注册来实现通知接收，例如在Java当中使用BATTERY\_CHANGED action。

### 4. 其它类

正如大家所见，Android组件的设计初衷在于为不同应用程序提供彼此之间的交互效果。正如广播通知可用于系统之上的任何应用，Android还提供其它一系列action、帮助我们在应用程序中完成各类常见任务——例如拨打电话号

码。同样，大家也可以使用由其他开发人员所提供的功能，从而快速实现预定处理流程、节约代码开发量并帮助自己将更多精力集中在应用程序的独特之处上。当我们启动一项intent时，可以通过设置使其向启动中的Activity返回一个结果——即使所启动的intent并非我们应用程序的组成部分。这就使我们的应用能够在要求操作完成之后继续运行。

其它值得关注的Android组件还包括fragment以及action bar。下面就让我们简单对二者进行探讨：

## Fragment

比起单纯利用Activity以及布局配置来定义应用程序中各个屏幕下的用户界面，使用fragment的话效率可能会更高一些。在fragment的帮助下，大家可以将自己的用户界面拆分为逻辑部分，甚至在应用程序的多个屏幕之间重复使用同样的部分。这样一来，我们不仅能够节约大量花在实现同样视觉/交互元素上的重复劳动，同时也实现了修改一点即完成对整款应用变更的效果。

Fragment在Android系统中属于Activity中的组成部分，因此每个fragment都要与其所在的Activity生命周期相适应。

## Action Bar

在应用程序开发过程中，action bar也是我们经常需要用到的关键性用户界面元素。它的作用在于为我们的应用程序提供足以作用于整套Android系统的用户界面，这也使它成为该平台用户们最熟悉的元素。一般来说，action bar当中所显示的条目包括用户在应用程序当中所处的位置以及应用程序各个部分之间的导航系统。要在Activity当中使用action bar，大家需要保证自己的类当中包含ActionBarActivity类，并在Manifest当中将AppCompatActivity应用在该Activity当中。

## 总结

具体使用哪一种Android类及组件取决于大家的应用程序到底要执行哪些任务。不过经过文章的论述，相信应该能够帮助大家对类与组件的类型及数量有所了解，并根据实际情况作出正确选择。对于特定特性或者功能来说，我们往往很难决定具体该使用哪款组件或者类，因此请大家在判断之前确保自己对它们的作用拥有清晰的了解。在接下来的教程当中，我们将探讨Android示例代码以及应用程序的发布流程。



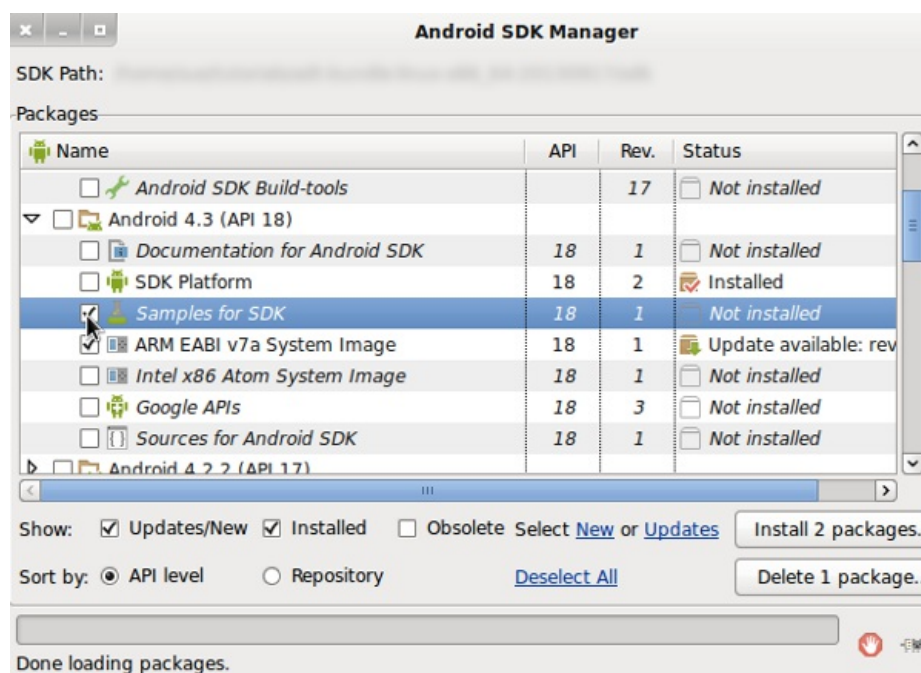
# 第十五章 示例项目

Android SDK示例项目中的应用能够执行种种功能，例如各类用户界面元素、数据管理、交互、媒体及连接使用说明等。即使大家不打算在自己的开发过程中用到示例所包含的某些特定应用类型，其中的大部分功能仍然适用于其它不同类型的应用。总而言之，这些示例资源值得大家探索一番。

## 1. 安装

### 第一步

要在Eclipse当中直接使用Android示例项目，大家首先需要确保自己已经将其正确安装到位。在“Android SDK Manager”当中选择“Window”，在拉下来的软件包列表当中大家会在每个API级别的文件夹当中找到不同示例——选择最新的一个然后将其展开。如果其中的“Samples for SDK”尚未安装，现在就将其选中并安装。

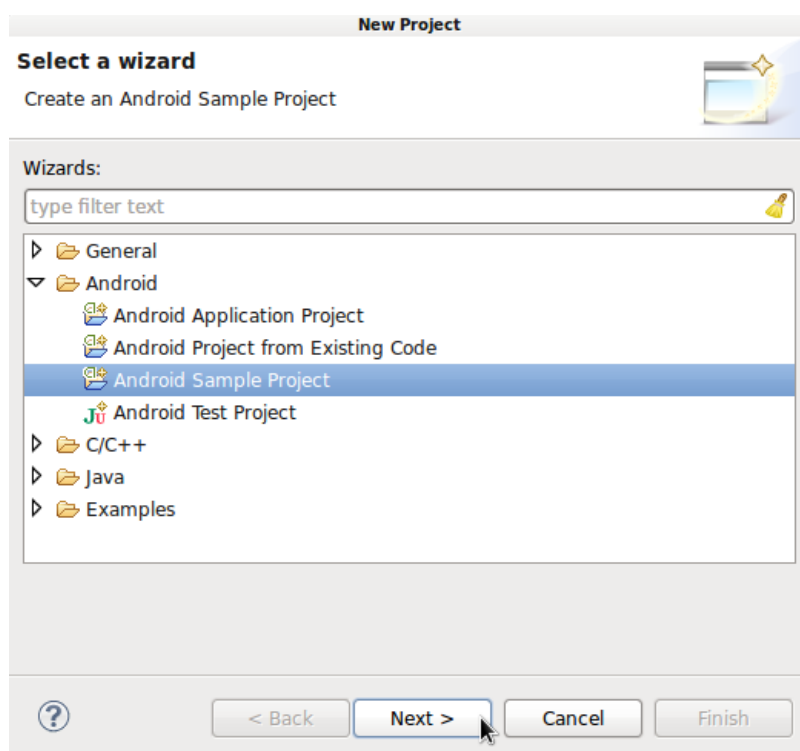


示例项目安装完成之后，大家就可以在Eclipse当中直接找到它们了。请记住，我们在本系列教程的前几篇文章中谈到过如何利用SDK Manager保持Android SDK工具处于最新状态，因此请各位在当前拥有可更新内容时马上进行安装。

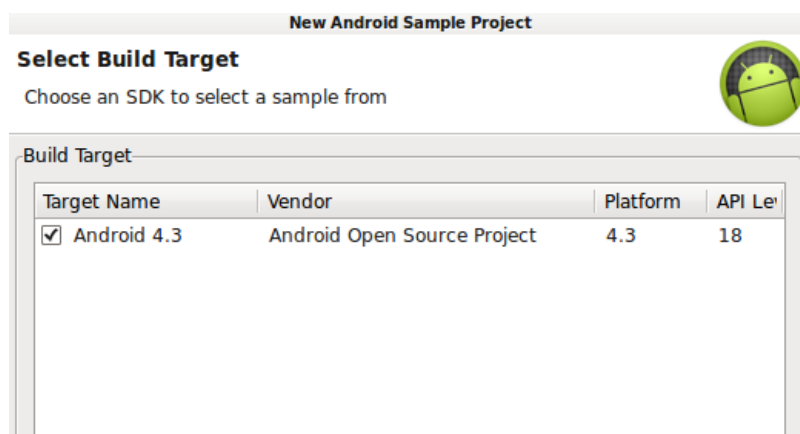
## 2. 创建示例项目

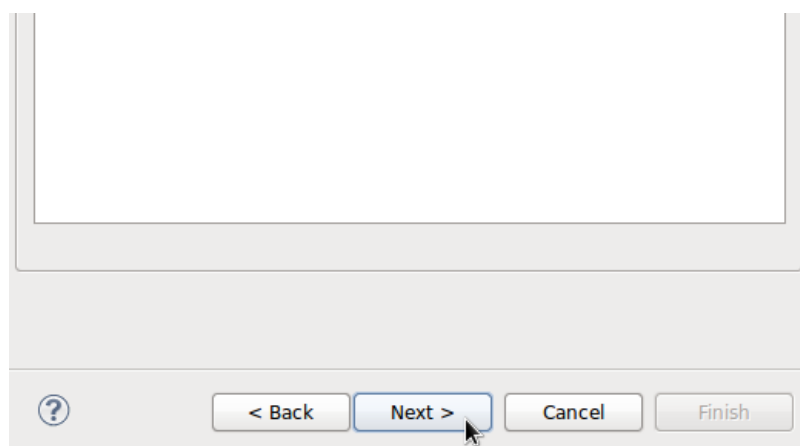
### 第一步

大家可以在Eclipse当中创建一个示例项目，从而查看该示例中的代码并在物理或者虚拟设备上加以运行。这样我们就能通过复制和粘贴来借用示例当中所涉及的算法，或者通过解读与学习把握概念、今后用在自己的应用程序当中。要在Eclipse当中创建示例项目，需依次选择“File”、“New”然后是“Project”。展开Android文件夹之后，选择“Android Sample Project”然后点击“Next”。

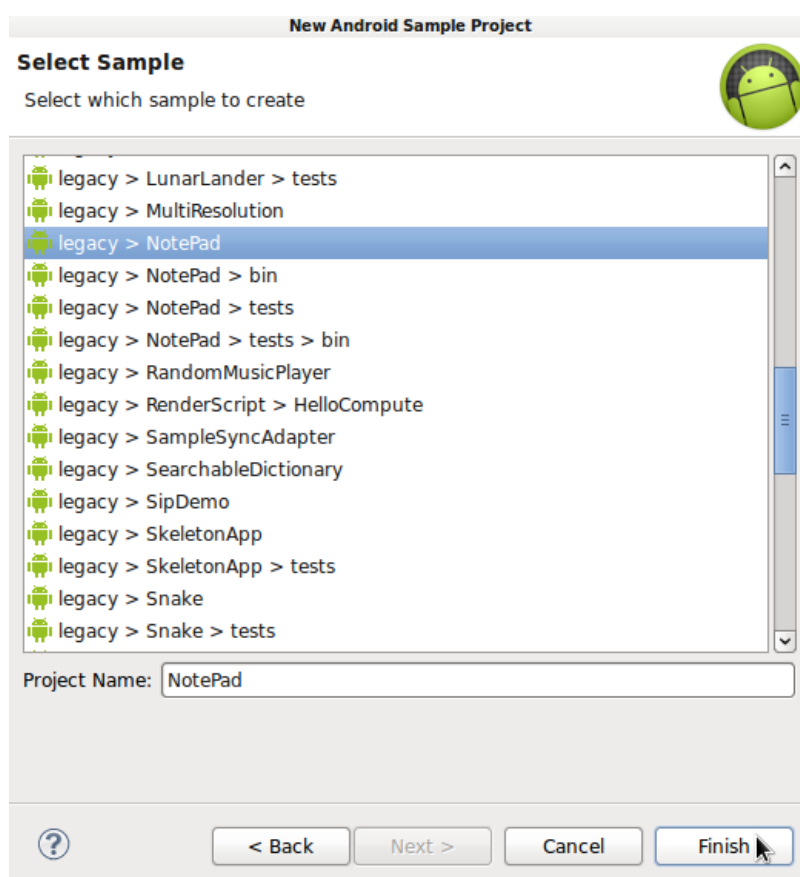


选择一个创建目标并点击“Next”。



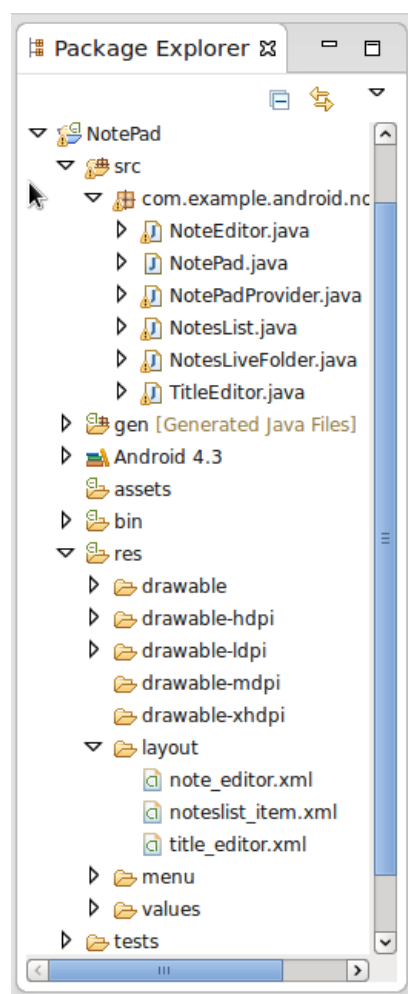


现在大家可以从示例列表当中作出选择，其中包括早期遗留示例——大部分属于全功能应用，包含我们可能在首次应用开发工作中所需要的处理流程类型。大家可以在业余时间慢慢研究这些示例，但作为初次接触、我们这里先选择“Notepad”示例并点击“Finish”。



Eclipse会在我们的工作区当中利用示例代码创建应用，过程与创建我们自己开发的应用一样。完成后，大家应该可以在自己的Package Explorer当中找到Notepad应用了。展开该应用的文件夹，我们还可以进一步查看其中的内容。

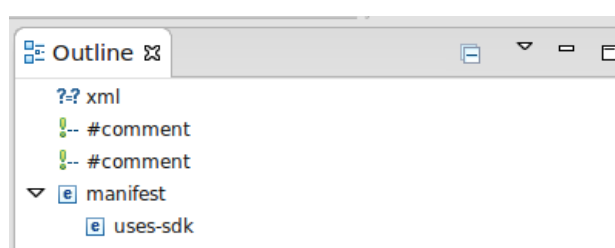


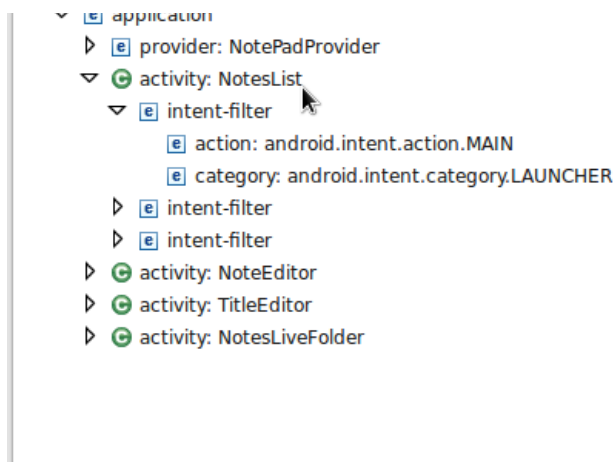


## 第二步

大家可以花点时间看看示例项目当中所包含的各个文件，其中包括Manifest、source code以及resource files，例如布局、可绘制对象、值以及菜单等。Notepad应用对于初学者来说算是很好的资源，大家可以通过它了解Android平台上很多常见的功能类型。现在让我们打开Manifest文件并切换到XML选项卡。

大家不要被示例文件当中复杂的文件结构所吓倒。如果各位在Eclipse View当中查看这些内容，特别是采用Outline View，肯定会看到它们与我们在前几篇文章中开发过的应用拥有同样的整体结构。



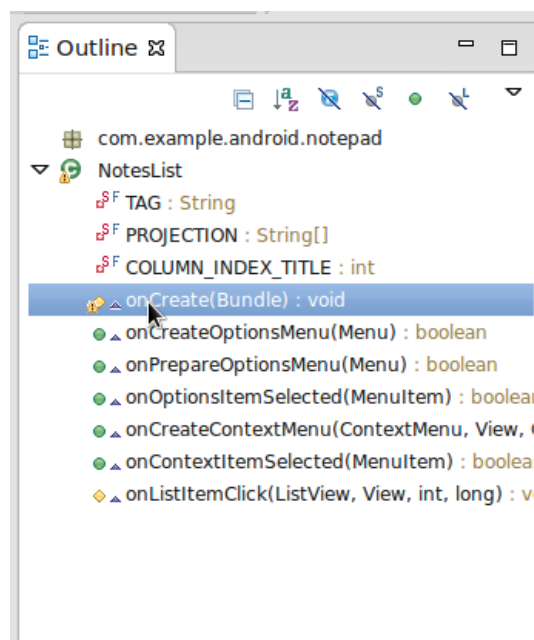


源文件当中通常包含大量代码注释，因此如果大家不能确定特定代码段的具体含义、完全可以从注释当中得到解答。在Outline View当中展开元素，我们会看到NotesList Activity将以应用程序main launcher Activity的形式被详细列出。由于这是在系统上进入应用程序的主要入口，因此我们可以将其作为很好的应用源代码探索起点。

### 第三步

在花时间浏览过其它Manifest内容之后，打开Notepad应用src文件夹下的NoteList Activity。

如大家所见，所有源代码都拥有良好的注释，不仅该类的作用拥有详尽说明、其中每个方法以及各方法的组成部分也经过了解释。下面请在Outline View当中查看onCreate方法。



大家可以很直观地发现，onCreate方法中的代码与之前我们所看到的ListActivity只存在细微的差别。我们发现列表当中的每个条目都会引用同一种布局，也就是noteslist item。打开应用程序的res布局文件夹并找到该布局。大家可以在该布局条目的标记与图形视图之间进行切换，这一点我们之前也已经说明过了。

大家可以利用这种通用型方式探索所有示例应用，即浏览各种组成元素、以逻辑方式遵循引用关系逐步查看各个源文件。

## 第四步

大家可以在物理设备或者Android虚拟设备（简称AVD）上运行这些示例应用，具体方式与运行自己开发的应用程序完全相同。让我们再次捋清思路：接入一台设备或者启动AVD，而后进入Run Configurations。在其中选择Android Application，点击New按钮并利用Browse按钮定位要运行的示例应用。如大家所示，运行示例应用的过程与运行自己创建的应用并无二致。

如果大家在开发应用程序时用到了与示例应用当中类似的功能，则可以将示例应用内容打开并安置在自己的应用旁边，从而以此为参考开发自己的算法。Notepad示例这类应用程序当中包含多种非常典型的处理过程，例如通过Content Provider处理数据——我们可以在NotePadProvider类当中看到这部分代码。示例应用的设计初衷在于向开发者展示如何在Android平台上完成特定任务，因此大家在遵循《Android开发者指南》的说明时这些示例能够起到很好的辅助作用。

提示：当我们使用Android SDK当中的早期遗留示例时，会发现Eclipse显示关于废弃代码的警告信息。这是因为早期遗留示例是针对早期SDK版本创建的。如果大家希望使用被Eclipse列为“不推荐”的任何功能，请首先查看新版本Android SDK中是否提供更新更有效的替代方案。

## 第五步

请大家随意探索SDK示例中的代码内容，很多示例内容的含义都非常清晰。不过有时候其内容功能与示例名称之间的关系可能没什么直接联系。其中最值得关注的示例要数API Demos，大家可以按前面介绍的方法在Notepad应用列表中找到它。

先利用API Demos示例启动一个示例项目，如前文所述将其在Package Explorer当中展开，然后打开src文件夹。该应用会被拆分为多个包，各自对应不同的功能类型。我们可以从这里看到并学到很多实用的知识。打开其中一个包，查看它所包含的各个类。与其它示例代码一样，这部分代码同样拥有详尽而明确的注释，能够帮助大家理解每个类及方法的实际作用。API Demos应用当中包含图形、动画、媒体、安全以及可访问性几大功能。

### 3. 示例的其它使用方法

在Eclipse当中创建示例项目是最简单也最实用的学习方法，能够帮助大家透彻掌握Android SDK所提示的说明性代码内容。不过我们还可以通过其它方式来充分利用SDK示例代码。大家可以在自己的计算机上从ADT Bundle目录中找到各个示例项目的具体文件。在其中的sdk文件夹内，大家会看到一个名为samples的文件夹。我们在各个平台层面上已经安装过的示例都能在这里找到，而且它们都拥有自己的专有文件夹。在该文件夹中，大家还会看到被划分为不同示例类型的文件夹。我们可以通过这种方式浏览、打开并与源代码文件进行交互。

在Android开发者网站的[Samples](#)部分，大家可以下载并查看该平台上的各种示例代码。这些示例旨在与利用Gradle创建的Android Studio相协作。大家可以下载完整的项目并将其导入至Android Studio当中，从而在该IDE下与这些内容进行交互或者将应用运行在Android设备之上。

## 总结

Android示例当中包含大量功能。尽管其中一部分早期遗留内容现在看起来有点过时，但作为学习材料仍然具备良好的价值。如果大家发现自己打算开发的功能在示例项目中已经存在，那正好能省下大量的开发时间——只需对SDK给出的内容稍加改动即可。在本系列的下一篇文章中，我们将探讨在应用程序开发并测试结束之后，该如何将成果发布出去。

# 第十六章 应用程序发布

在今天的文章中，我们将重点探讨通过Google Play软件商店进行应用程序发布，不过如果愿意，大家也可以选择其它一些发布途径。要通过Google Play商店进行应用程序发布，我们需要注意一些必要条件；不过如果通过其它途径发布则有可能无需考虑这些前提。但我个人建议大家认真了解这些内容，并尽可能在任何发布方式当中都严格贯彻这些最佳实践方案。

## 1. 准备工作

在我们考虑发布一款应用程序之前，首先要完成的就是全面的调试与测试工作，其中包括确保应用能够在各类不同配置的设备上正常运行。此外，大家还应该注意其它一些前提性事项。首先，如果我们的Java代码当中包含任何日志声明或者其它用于输出调试信息的调用内容，请务必在发布之前将其从应用内移除。

如果大家的清单文件当中包含`android:debuggable`属性集，则需要在发布应用程序之前将其移除。我们的清单版本属性还需要经过合理配置，我将在后续内容中进一步解释该话题。大家要确保自己的应用程序资源正确包含在软件包文件夹里，例如`drawables`等指向配置信息的媒体条目。如果大家的应用程序需要使用数据库之类资源，则必须确保其经过合理调整。

如果大家的应用程序在运行中需要使用某种程度的权限，则必须在`manifest`当中利用`uses-permission`元素将其添加进来。为了顺利实现应用发布，大家还需要在`manifest`应用程序元素中设置应用图标以及标签属性。我们在清单当中列出的大部分条目都将与Google Play软件商店的应用列表内容相对应。

## 2. 版本管理

我们在之前的文章当中已经提到过，大家需要为自己的应用程序设置一个版本号并为其命名。说起版本号，这部分信息应该被包含在根`manifest`元素中`manifest`下的`android:versionCode`与`android:versionName`属性当中。

其中`versionCode`属性应该为一个整数，且每一个应用程序新版本分配到的数字都需要比前一个更大。从逻辑角度讲，大家应该以1为初始版本号，并在每一次推出新版本时逐渐递增该数值；不过大家也可以随意选择自己想要的数

字，只要比上一个版本数值更大即可。终端用户无法看到应用程序的版本代码值，该数值仅用于在发布过程中衡量应用程序的当前版本号是否比原先已经安装的版本更新。

不过versionName属性则不同，它是一个可被终端用户查看的字符串。版本名称并不需要一定与版本代码相匹配，但从逻辑上讲其同样应该遵循递进关系。举例来说，从1.0开始，接下来应该是1.1，当我们发布的新内容更新幅度较大时则将其提升为2.0。换句话说，版本名称应该能够帮助终端用户理解应用版本的先后顺序。如果大家有计划为自己的应用程序发布多个版本，则最好花点时间来考虑怎样的版本名称最适合反映自己的升级进程。

### 3. 签名

#### 第一步

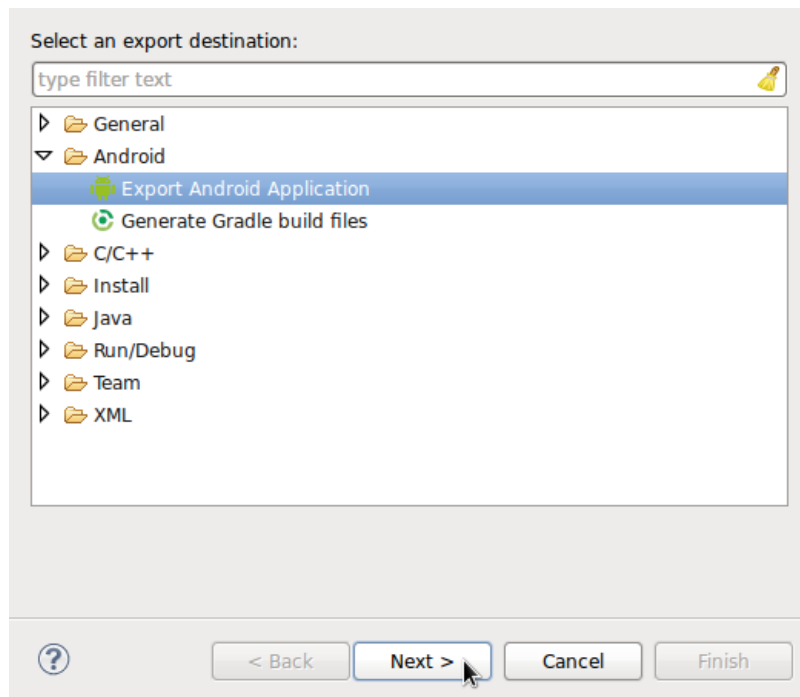
要在Android系统上安装一款应用程序，该应用必须利用具备私有密钥的证书进行签名验证。在我们的开发过程中，Eclipse与Android SDK会自动利用调试密钥完成应用程序的签名过程，但大家无法通过这个调试密钥进行应用程序发布。在Android应用程序的创建过程中，系统会选择debug或者release两种模式之一进行创建。在release模式下，大家需要利用自己的私有密钥完成应用程序签名。

大家也可以利用keytool程序为自己的应用程序生成一个密钥，我们可以在Java Development Kit（或者简称为JDK）当中找到该程序。各位可以点击[此处](#)查看keytool说明文档以了解更多详细情况。在为我们的私有密钥创建了keystore之后，大家即可选择alias name以及password，从而在日后进行应用程序签名时加以使用。

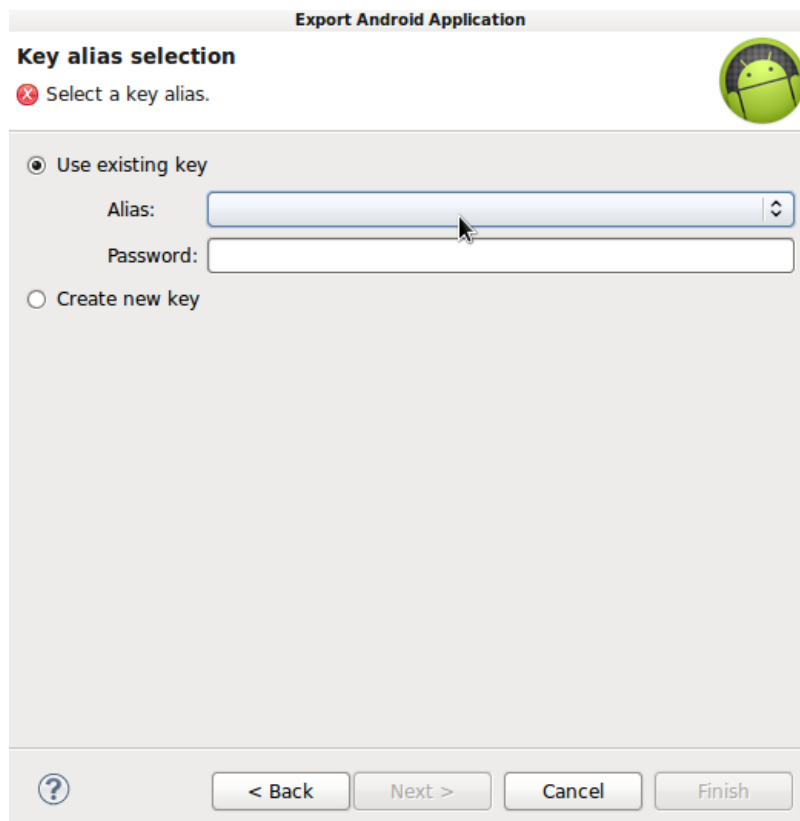
#### 第二步

当我们已经准备好了应用程序签名所必需的key/keystore之后，正面要做的就是为其创建一个发布版本。在Eclipse当中，通过Package Explorer选中自己的应用程序项目，右键点击该项目或者选择“File”、“Export”。展开其Android文件夹后，选择“Export Android Application”并点击“Next”。





接下来，Eclipse将突出显示创建过程中所遇到的全部错误信息，我们需要将其一一解决才能继续进行下一步。如果整个过程没有出现错误，大家可以直接点击“Next”以继续。在Keystore Selection容器中，浏览至我们的keystore文件并为其输入密码。接下来，从菜单中选择我们为密钥指定的alias并输入自己设定的密码内容。当一切准备就绪之后，点击“Next”进入下一步。





选择一个位置并为我们应用程序的APK文件指定一个名称。该APK文件也就是我们将要上传到Google Play商店中的文件，用户在安装过程中需要将其下载到自己的设备上并加以运行。Eclipse会利用正确的密钥与验证机制处理应用程序签名。在点击“Finish”之后，该APK文件就会出现在大家所选定的保存位置。现在我们应该已经可以将该APK文件复制到Android设备上了。在复制工作完成之后，利用文件管理器应用选择该APK文件，并依据说明进行安装。只要应用程序签名一切正常，则系统应该能够顺利安装该应用、供大家在设备上运行所发布版本的应用程序了。

请确保我们的发布密钥受到严格保护，因为只有使用同一套密钥、大家才能在为同一软件包发布更新内容。如果大家利用其它不同的验证机制处理应用程序的更新版本，则必须使用与原先不同的软件包名称。

提示：在发布特定应用程序时，大家可能需要执行额外一些步骤。举例来说，需要利用谷歌地图库的应用程序要求我们为Maps API设定一个专门的发布密钥。

## 4. 发布

在我们创建好了一个完整版APK文件之后，接下来要做的就是将应用程序发布到Google Play软件商店当中。除了应用程序本身，我们还需要为发布过程准备一些额外资源——其中一部分最好能提前准备就绪。首先，应用程序的销售宣传图片以及功能描述是必不可少的，大家需要将这部分信息添加到应用程序的列表当中。我们还需要为应用程序在软件商店中的介绍与下载界面设计说明内容，其中包括应用程序的定价（如果这是一款收费应用的话）、应用程序的内购项目以及语言设定等等。

要着手进行发布流程，我们首先需要登录自己的谷歌账户并导航至[Developer Console](#)。点击“Add New Application”并在弹出的窗口中选择自己应用程序的默认语言、输入应用程序名称。从这里开始，大家就要用到之前准备好的应用程序清单并上传我们刚刚创建好的APK文件。

相信大家已经注意到了，我们应用程序的Google Play清单会要求填写大量细节信息，而且整个发布过程也需要一定时间。在每款应用的主清单部分，大家可以添加一部分图形内容——例如应用图标、应用截图、视频介绍、应用程序描述、功能类别、内容评级以及开发者的联系方式等。如果各位对这些信息在Google Play中的显示效果还没啥概念，请直接参照商店中其它已经摆上货架



的其它应用程序。

在应用程序清单中的Pricing and Distribution部分，大家需要指定自己的应用是否需要收费或者可以免费提供给用户。请注意，免费应用程序无法被重新修改为收费应用。不过大家可以修改收费应用的价格或者将收费应用修改为免费。我们也可以在免费应用当中设置内购机制。如果大家的应用程序适用于多个国家，Google Play会自动将我们设定的价格换算为其它货币单位。请大家花点时间仔细查看应用程序清单当中的其它内容，例如应用内产品以及API等。

如果大家已经在Google Play上发布一个或者多个应用程序，则可以在Developer Console当中查看各应用的当前状态。这套控制台为开发人员提供了非常详尽的报告内容。我们可以对清单的统计部分进行配置，从而更准确地把握其Android版本、应用版本、国家、设备以及语言设定的运作情况。其它统计内容还包括安装与营收报告、详细的崩溃情况报告、评分以及用户评论等。相比之下，苹果的App Store还允许开发人员回复用户评价并直接与应用程序用户进行对话。当应用程序被摆上Google Play商店的货架之后，大家就可以通过分享应用清单的链接来达到宣传的目的。

最后，在着手进行应用程序发布流程之后，请认真阅读 [Launch Checklist](#) 中的内容。

## 总结

如果大家刚刚开始接触Android平台，那么离发布自己的第一款应用恐怕还有很多长的路要走。不过了解整个发布流程对于新手开发者来说仍然非常重要。应用程序的首次发布可能需要花上不少时间，但一旦填写了初始信息及描述之后，未来的更新上传将变得很快。在本系列教程的下一部分，我们将为大家提供一些进一步学习Android开发的建议性方向。最后，我们将共同面对一次小小的结业考试——通过试题验证大家是否已经真正掌握了到目前为止介绍过的知识。

# 第十七章 下一步学习方向

通过之前的文章，我们了解了各类开发工具、探索了应用程序项目中所包含的基本要素、学习用户界面设计、交互性、资源以及数据的使用机制，另外我们也全面追踪了应用程序运行的整个过程。到目前为止，我们已经介绍过的知识足以帮助大家从非常理想的起点开始进行Android应用程序开发，但Android所提供的发挥空间还远不止这些、因此开发中的可能性也几乎没有穷尽。因此，我们需要为自己的下一步学习选择明确的方向。在今天的文章中，我们将把全部注意力集中在可资选择的未来学习对象上。在本系列教程的下一篇文章——也就是最后一篇文章里，我们将通过一项测验回顾曾经了解过的各项知识。

通过阅读本系列教程，大家应该已经在创建Android应用时拥有多种可以选择的潜在发展方向。当然，如果大家已经规划好了具体的开发项目，那么由此带来的实际需求已经足以指导各位明确未来的学习路线。不过Android平台所提供的元素极为丰富，而且其中很多应用元素都普遍适用于我们将要接触的大多数应用。在今天的文章中，我们将近距离观察一部分实用性最高的主题，并以此为起点建立体系完整的Android开发技能。

## 1. 数据

### 第一步

大家可以利用我们之前创建好的示例应用项目尝试下面将要介绍的各种Android开发技巧。请注意，我们原先是把数据保存在shared preferences当中，因此大家可以选择将数据保存在文件内。我们要做的是尝试让用户利用EditText视图以及功能按钮实现数据的提交与保存。将用户数据写入到一个文件当中，让应用程序在启动时读取其内容并把结果显示在TextView视图当中。通过这种方式，大家可以实现持久性数据的保存，从而在应用程序下一次运行时加以使用。

### 第二步

另一项重要技能在于从远程资源，例如Web服务，当中获取数据。为了实现这一目标，我们需要在Activity类当中添加一个内部 AsyncTask类，并借此进行远程数据获取。在获取到数据之后，还需将其显示在应用程序当中。此外，如

果应用程序获取到的是XML、JSON或者其它 一些常见数据格式，大家可能还希望对其加以解析或者进行格式调整。参阅谷歌官方提供的“[连接到网络](#)”部分的说明，其中提供的示例代码足以作为理想的起步素材。

### 第三步

另一种主要数据存储选项同样适用于很多应用程序的实际需求，这就是使用SQLite数据库。大家可以通过创建一个SQLiteOpenHelper 类来尝试这种方式，在这里我们需要定义该数据库架构，例如表和列。在应用程序的其它部分，我们可以要求SQLiteOpenHelper类将数据写入到数据库 中并利用游标实现内容读取，从而将结果显示在应用程序的用户界面当中。再次提醒大家，我们可能需要将用户输入的数据保存在数据库当中，并在应用程序下一次 启动时显示结果。除了向数据库中插入记录并对其进行查询之外，大家还可以尝试更新并删除记录。作为初次上手的起点，我们不妨先从阅读谷歌官方提供的“[在SQLite数据库中保存数据](#)”说明开始。

## 2. 应用屏幕

### 第一步

在已经创建完成的应用程序当中，我们只在用户界面当中使用了一个操作屏幕。接下来，大家可以通过在应用中创建第二个activity类来增加屏幕数量，即顺序打开File、New、Class菜单选项。今天的任务是在第一个屏幕——也就是主activity当中添加一个按钮，并在用户点触该按钮时启动第二个activity。举例来说，如果我们开发的是一款游戏应用，那么该按钮所显示的文本内容可以是“游戏说明”，而第二个activity的名称则 可以被设为How.java。为第二个activity创建一个布局文件，其中包含一个将文本内容保存在res/values字符串XML文件中的 TextView。如果我们使用的示例应用如上所述是一款游戏的话，那么第二个activity中应该包含关于游戏操作与进行方式的信息。应用程序中的大部分信息显示都可以遵循这种简单的处理方式。

在第一个activity的onClick处理程序中，大家可以利用 intents 启动“游戏说明” activity。一旦我们在应用程序中包含了 第二个activity，则需要利用putExtra方法向其提交来自第一个activity的数据。大家也可以尝试反过来让第二个activity执行 某些任务，并把获得的结果提供给第一个activity。后者需要通过onActivityResult方法来接收结果数据。请大家[点击此处](#)查看谷歌

官方提供的“Activity类引用”说明以了解更多与此相关的知识。

## 第二步

作为下一个步骤，大家可以尝试在自己的activity当中包含多个不同用户界面视图。大家还会注意到，不同类型的视图需要通过不同的方式加以实现。在大家熟练掌握了不同视图类型的使用方法之后，接下来可以尝试使用fragment——这样我们就能够在应用程序的不同部分中重复使用用户界面的各个组成元素了。

另一种实用性极高的用户界面组件要数list视图。在list视图中，屏幕会显示一份包含多种条目的列表。系统会利用adapter将来自数据源的信息填充到list视图当中，也就是实现由数据到视图的映射。大家可以利用ListActivity来取代标准的Activity类。在list activity当中，大家可以对方法集进行重写以响应用户与列表内条目的交互操作。作为学习list视图的开端，大家可以首先认真阅读谷歌官方提供的[list视图功能示例](#)。

一般来说，应用程序在外观与使用体验上最好能够与Android系统本身保持一致。只要可能，大家应该尽量利用标准化Android用户界面元素而非创建自己的定制组件。从这一理由出发，大家可能希望了解更多关于[Action Bar](#)的知识，并始终提醒自己在设计用户界面以及应用程序的导航模式时遵循上述结论。

## 3. 多媒体

大家可以在自己创建的Android应用程序当中使用多种媒体类型，例如音频与视频、动画以及通过设备摄像头所捕捉到的图片乃至视频等。Android系统提供一系列标准方法，大家可以借此访问设备资源（例如摄像头）并实现特定使用需求（例如进行视频记录）。请大家查看《Android开发者指南》中的“[多媒体指南](#)”一节以获取更多信息。

正如我们在本系列教程中所提到，大家可以在Android应用程序当中利用XML创建视觉元素、从而获得属于自己的可绘制组件。大家也可以利用XML来定义动画，并通过代码控制动画的播放效果。在Android平台上，我们可以利用动画为用户界面添加各种动态效果——系统支持的效果数量繁多，其中包括淡入淡出、旋转、翻转以及其它各种过渡类型。大家点击此处参阅谷歌官方提供的“[添加动画](#)”说明。

## 4. 与其它应用程序交互

### 第一步

在Android平台上进行应用程序开发的优势之一在于，我们可以充分利用平台所提供的现有资源，其中包括其它应用程序。正如我在本系列教程的前几篇文章中所提到，大家可以在不同应用程序之间实现数据共享并在自己的应用中使用这些共享数据。大家还可以允许用户利用电子邮件、即时消息以及社交网络等方式在我们的应用当中共享内容。最简单也最可靠的数据共享途径就是利用send意图。当我们的应用程序启动一条send活动时，操作系统会为用户提供一份应用程序列表、其中罗列了用户可以将内容发送至哪些目标处。请大家在着手尝试之前认真阅读谷歌官方提供的“[向其它应用发送简单数据](#)”说明，其中还包含几个示例。

### 第二步

除了使用send之外，我们还可以利用其它多种方式实现从应用程序内部启用Android资源，因此请大家在自己的项目多多进行尝试。举例来说，大家可以利用dial来拨出电话号码、利用view在浏览器中查看见面或者使用地图应用中的位置信息。大家还可以从设备的内置传感器处获取数据，从而使自己的应用程序能够捕捉并处理与位置及周边环境相关的数据。在多数情况下，我们可以通过在action启动结束时在应用程序与Android环境之间建立起对话，并向应用程序返回信息时触发其它action。利用这种方式，我们的应用程序就可以充分利用Android设备所提供的移动特性。

## 5. 资源

### 第一步

我们已经讨论了未来学习的几大主要潜在方向，但事实上大家所接触的仍然只是Android平台颇为表面化的浅层次知识。这套平台仍然在不断变化，而可行性名单也会变得越来越长。针对常见任务的推荐性技术也会定期变更，因此如果大家希望能一直开发出高品质的Android应用程序，请记住不断学习才是最重要的职业习惯。

下面我再为大家推荐一些实用性很高的学习资源：



访问[Android开发者博客](#)来获取关于这套平台的最新及未来发展趋势。该博客还经常提供一些功能性代码示例，可以作为很好的学习素材。

[Vogella Android指南](#)是网络上现有的最具Android学习资源之一。Vogella网站通常会提供一些完整的技术示例项目，旨在对官方开发者指南当中所涉及的专业知识进行详细讲解。

[Google+上的Android开发者交流平台](#)经常发布大量公告与Android开发讨论话题，同样值得大家多加关注。

正如其它开发平台一样，大家会在学习Android的过程中发现更多无法确定的问题，其中大部分都需要通过网络搜索加以解决。正如大家所了解，很多常见问题都能在[Stack Overflow](#)上找到答案——如果各位还不是该网站的常客，请马上去逛逛吧。

## 结论

在本系列教程当中，我的目标是帮助大家学习关于Android开发的基础知识。如果各位过去曾经学习过编程或者开发技能，肯定清楚本系列教程还远远不是探索的终点。如果大家有意继续为Android平台开发应用，则需要熟练运用本教程所介绍的知识并将其作为Android工具箱中的财富好好加以保管。作为本系列教程的结尾，我将在下一篇文章中提供一项测试、看看大家有没有真正掌握之前提到的内容——请做好准备吧！

# 第十八章 知识测试

## 教程说明

完成时间:十五分钟

执行难度:简单

前面我们已经了解了为Android平台创建应用程序过程中需要涉及的各种基本概念及知识要点。一路走来，我们探讨了关于Android开发的各方面内容，其中包括Java开发、XML使用、用户界面设计、项目结构、数据存储以及发布流程等。为了检验我们的学习效果，在今天的文章中请大家接受一份结业测试、看看自己是否掌握了前面提到的各项知识。

## 问题一

我们的Java类被保存在以下哪个Android应用程序目录之下？

1. res
2. layout
3. src
4. values

## 问题二

我们不会在项目清单文件中执行以下哪项内容？

1. 在应用程序当中声明activity。
2. 设定最低API支持级别。
3. 定义按钮被点击后执行何种事件。
4. 列出应用程序运行所需要的权限。

## 问题三

为了在Java当中利用“@+id/how”语法检索XML中某个视图集的id，我们应该使用以下哪条语句？

1. R.how
2. R.view.how
3. findViewById(how)
4. R.id.how

## 问题四

我们应该使用以下哪条语句在XML当中设定TextView所显示的文本字符串？

1. android:text='@string/info'
2. android:string='info'
3. android:text='@text/info'
4. android:value='@string/info'

## 问题五

以下哪一种才是我们用于定义用户点击某个按钮时所执行事件的标准方法？

1. onClickListener
2. onViewClick
3. onClick
4. onButtonClick

## 问题六

我们需要将以下哪种XML属性添加到视图当中，从而指定用户进行点击时所执行的方法？

1. android:onClick
2. android:click
3. android:clickListener
4. android:clicked

## 问题七

我们需要使用以下哪条语句在ImageView当中设置一个可绘制显示图形？

1. android:img='@drawable/my\_shape'
2. android:shape='@drawable/my\_shape'
3. android:drawable='@drawable/my\_shape'
4. android:src='@drawable/my\_shape'

## 问题八

我们需要将以下哪种activity元素包含在清单当中，从而在应用程序从设备菜单中启动时执行该activity？

1. 包含在某个属性当中的应用程序名称。
2. 主要及启动器属性。
3. 主action以及启动器类型元素。
4. 主类型与启动器action元素。

## 问题九



我们需要在哪个元素当中声明应用程序在清单中所要求的权限？

1. permission
2. request-permission
3. permission-required
4. uses-permission

## 问题十

应用程序的Shared Preferences是用来干什么的？

1. 保存原始数据项的键值对。
2. 在表当中以行和列的方式保存结构化数据。
3. 检索互联网数据。
4. 将数据保存在用户设备上的外部文件中。

## 问题十一

应用程序在读取并写入文件时，我们需要如何处理I/O错误？

1. 仔细检查文件名字符串。
2. 将我们的I/O代码放置在一个独立的类当中。
3. 尝试并获取与I/O代码相关的数据块。
4. 向用户输出警告信息。

## 问题十二

在尝试向外部存储机制进行写入之前，我们的应用程序不需要执行以下哪个步骤？

1. 检查外部存储机制是否可用。
2. 检查外部存储机制的写入访问。
3. 使用清单内用于向外部存储写入操作的权限。
4. 使用警告对话框，要求用户为数据写入提供权限。

## 问题十三

在从互联网源获取数据时，我们需要坚持做到以下哪一点？

1. 使用一个service类来获取数据。
2. 使用一个单独的进程、而不要利用用户界面进程进行数据获取。
3. 在主activity类中的一个方法内获取数据。
4. 将检索数据保存在SQLite数据库当中。

## 问题十四

以下哪种说法存在错误？

1. 即使是在启动某service的activity停止运行之后、该service仍将继续处于运行状态。
2. 除非用户利用后退按钮进行退出操作，否则activity将始终处于运行状态。
3. 某个绑定service在任何与之相绑定的组件停止运行后、也将一同停止运行。
4. 当某个activity的指向发生变化时、其在默认情况下将进行重新创建。

## 问题十五

要在某个activity当中启用另一个activity，我们需要使用以下哪种类？

1. Intent
2. Thread
3. View
4. Service

## 问题十六

当一款应用程序启动并处于resumed状态时，以下哪种回调方法不会执行？

1. onCreate
2. onPause
3. onStart
4. onResume

## 问题十七

当用户在暂停之后重新返回我们的应用程序时，以下哪种回调方法会付诸执行？

1. onRestart
2. onResume
3. onStart
4. onCreate

## 问题十八

我们需要利用当种方法将状态数据保存在activity的onCreate与onRestoreInstanceState方法当中、以备未来访问？

1. onDestroy
2. onSaveInstanceState
3. onStateChange
4. onSaveState

## 问题十九

哪个类允许我们定义可重复使用的用户界面部分？

1. Fragment
2. Service
3. Activity
4. View

## 问题二十

在向Google Play发布应用程序时，我们不需要进行以下哪个步骤？

1. 在清单当中包含应用程序的版本与名称。
2. 利用release key进行APK签名。
3. 为应用程序选择内容分级以及产品定价。
4. 为应用程序创建一段视频介绍。

正确答案：

1-5、CCDAC；

6-10、ADCDA；

11-15、CDBBA；

16-20、BBBAD。

# 后记

现在你已经学习了Android开发的基础知识，如想进一步学习，可前往[51CTO移动开发频道](#)，那里将有更多的更全面的内容。

本电子书由pockry维护，如果发现错误，可发邮件至 [pockry@outlook.com](mailto:pockry@outlook.com)，感谢您的反馈。