什么是 Android?

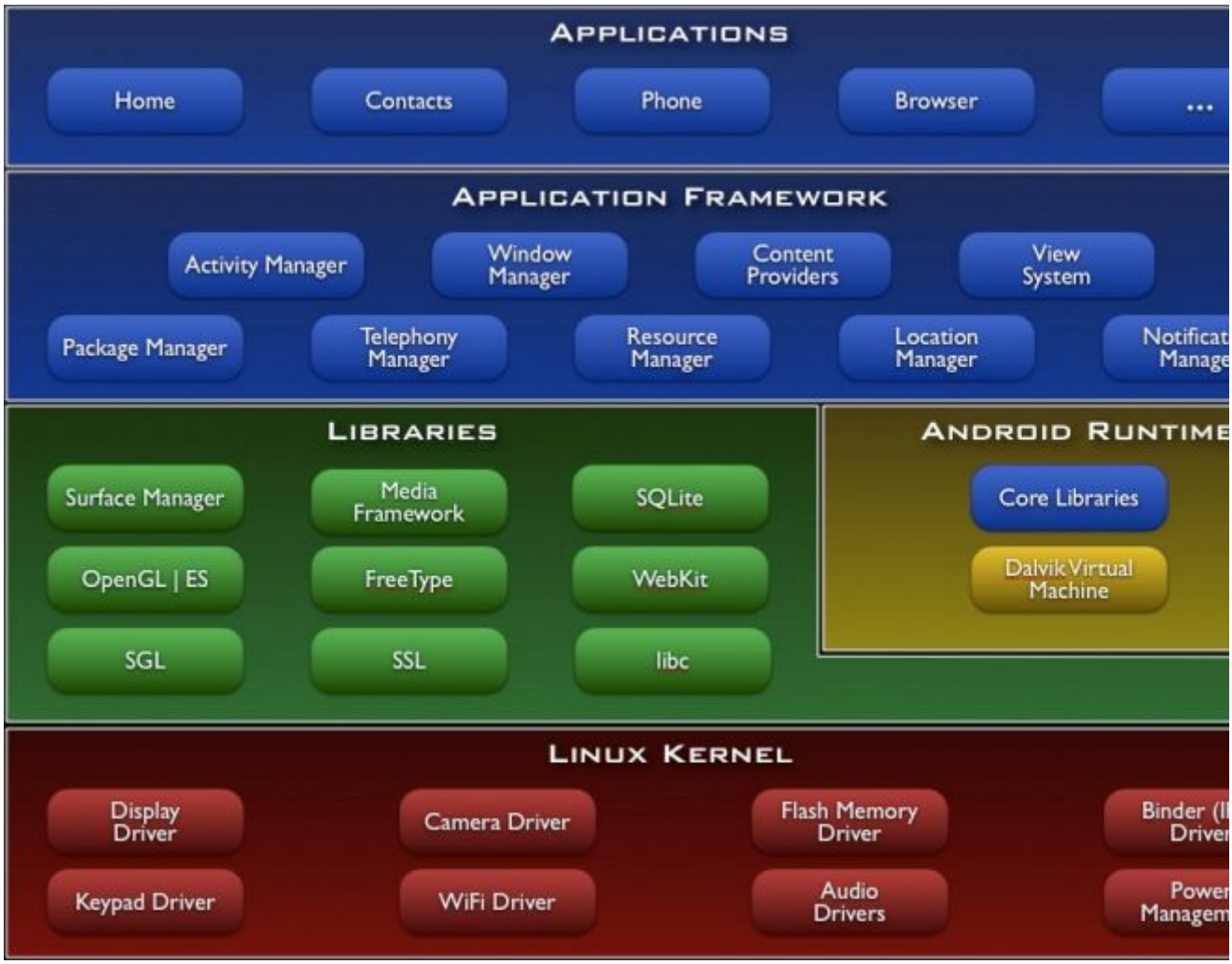
Android 是一个专门针对移动设备的软件集，它包括一个操作系统，中间件和一些重要的应用程序。Beta 版的 Android SDK 提供了在 Android 平台上使用 JaVa 语言进行 Android 应用开发必须的工具和 API 接口。

特性

- **应用程序框架** 支持组件的重用与替换
- **Dalvik 虚拟机** 专为移动设备优化
- **集成的浏览器** 基于开源的 WebKit 引擎
- **优化的图形库** 包括定制的 2D 图形库，3D 图形库基于 OpenGL ES 1.0 （硬件加速可选）
- **SQLite** 用作结构化的数据存储
- **多媒体支持** 包括常见的音频、视频和静态图像格式 （如 MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF）
- **GSM 电话技术** （依赖于硬件）
- **蓝牙 Bluetooth, EDGE, 3G, 和 WiFi** （依赖于硬件）
- **照相机，GPS，指南针，和加速度计（accelerometer）** （依赖于硬件）
- **丰富的开发环境** 包括设备模拟器，调试工具，内存及性能分析图表，和 Eclipse 集成开发环境插件

Android 架构

下图显示的是 Android 操作系统的主要组件。每一部分将会在下面具体描述。



应用程序

Android 会同一系列核心应用程序包一起发布，该应用程序包包括 email 客户端，SMS 短消息程序，日历，地图，浏览器，联系人管理程序等。所有的应用程序都是使用 JAVA 语言编写的。

应用程序框架

开发人员也可以完全访问核心应用程序所使用的 API 框架。该应用程序的架构设计简化了组件的重用；任何一个应用程序都可以发布它的功能块并且任何其它的应用程序都可以使用其所发布的功能块（不过得遵循框架的安全性限制）。同样，该应用程序重用机制也使用户可以方便的替换程序组件。

隐藏在每个应用后面的是一系列的服务和系统，其中包括；

- 丰富而又可扩展的视图（Views），可以用来构建应用程序， 它包括列表（lists），网格（grids），文本框（text boxes），按钮（buttons）， 甚至可嵌入的 web 浏览器。

- 内容提供器（Content Providers）使得应用程序可以访问另一个应用程序的数据（如联系人数据库）， 或者共享它们自己的数据
- 资源管理器（Resource Manager）提供 非代码资源的访问，如本地字符串，图形，和布局文件（ layout files ）。
- 通知管理器 （Notification Manager） 使得应用程序可以在状态栏中显示自定义的提示信息。
- 活动管理器（ Activity Manager） 用来管理应用程序生命周期并提供常用的导航回退功能。

有关更多的细节和怎样从头写一个应用程序，请参考 如何编写一个 Android 应用程序.

程序库

Android 包含一些 C/C++库，这些库能被 Android 系统中不同的组件使用。它们通过 Android 应用程序框架为开发者提供服务。以下是一些核心库：

- **系统 C 库** - 一个从 BSD 继承来的标准 C 系统函数库（ libc ）， 它是专门为基于 embedded linux 的设备定制的。
- **媒体库** - 基于 PacketVideo OpenCORE；该库支持多种常用的音频、视频格式回放和录制，同时支持静态图像文件。编码格式包括 MPEG4, H.264, MP3, AAC, AMR, JPG, PNG 。
- **Surface Manager** - 对显示子系统的管理，并且为多个应用程序提 供了 2D 和 3D 图层的无缝融合。
- **LibWebCore** - 一个最新的 web 浏览器引擎用，支持 Android 浏览器和一个可嵌入的 web 视图。
- **SGL** - 底层的 2D 图形引擎
- **3D libraries** - 基于 OpenGL ES 1.0 APIs 实现；该库可以使用硬件 3D 加速（如果可用）或者使用高度优化的 3D 软加速。
- **FreeType** -位图（bitmap）和矢量（vector）字体显示。
- **SQLite** - 一个对于所有应用程序可用，功能强劲的轻型关系型数据库引擎。

Android 运行库

Android 包括了一个核心库，该核心库提供了 JAVA 编程语言核心库的大多数功能。每一个 Android 应用程序都在它自己的进程中运行，都拥有一个独立的 Dalvik 虚拟 机实例。Dalvik 被设计成一个设备可以同时高效地运行多个虚拟系统。 Dalvik 虚拟机执行（.dex）的 Dalvik 可执行文件，该格式文件针对小内存使用做了 优化。同时虚拟机

是基于寄存器的，所有的类都经由 JAVA 编译器编译，然后通过 SDK 中 的 "dx" 工具转化成.dex 格式由虚拟机执行。
Dalvik 虚拟机依赖于 linux 内核的一些功能，比如线程机制和底层内存管理机制。

Linux 内核

Android 的核心系统服务依赖于 Linux 2.6 内核，如安全性，内存管理，进程管理， 网络协议栈和驱动模型。 Linux 内核也同时作为硬件和软件栈之间的抽象层。

一、开始

安装 SDK

本页内容介绍如何安装 Android SDK 以及如何设置你的开发环境。如果你还没有下载 SDK，你可以点下面的连接进行下载，然后阅读后续文档学习如何安装，配置，并使用该 SDK 来创建 Android 应用程序。

下载 SDK

升级?

如果你已经用更早的版本开发过程，可以跳过本页,并阅读升级 SDK 文档.

系统和软件需求

用 Android sdk 的代码和工具开发 Android 应用程序,你需要适合的开发电脑以及开发环境，如下描述：

所需操作系统:

- Windows XP 或 Vista
- Mac OS X 10.4.8 或更高版本 (仅支持 x86)
- Linux (Linux Ubuntu Dapper Drake 版本已测试)

所需开发环境:

- Eclipse IDE
  - Eclipse 3.3 (Europa), 3.4 (Ganymede)
    - Eclipse JDT 插件 (大多数的 Eclipse IDE 包包含)
    - WST (可选,但 Android 编辑器的功能需要,它被包含在 most Eclipse IDE packages 中）
  - JDK 5 或 JDK 6 (只有 JRE 是不够的)
  - Android Development Tools 插件 (可选)

- 不兼容 GNU Java 编译器(gcj)

- 其他开发环境或 IDE
  - JDK 5 或 JDK 6 (仅有 JRE 不够)
  - Apache Ant 1.6.5 或更高版本（ Linux 和 Mac 环境下）, 1.7 或更高版本（ Windows 环境下）
  - 不兼容 GNU Java 编译器(gcj)

注: 如果你的电脑上已经安装了 jdk，请确定它是上面所列的版本号。另外需要注意有些 linux 版本可能包含有 jdk 1.4 或者 java 的 gnu 编译器，Adroid 开发是不支持在这两种版本

安装 SDK

下载完 SDK 后，把.zip 文件解压到你电脑上合适位置. 默认情况下, SDK 文件被解压到 `android_sdk_<platform>_<release>_<build>`*文件夹*. 这个文件夹包含 `tools/`, `samples/`等.

请注意系统里 SDK 解压后的文件夹的名字和位置 — 当你安装 Android 插件和使用 SDK 工具时，你将需要引用这个文件夹。

你可以添加 SDK `tools` 的文件夹路径到你的环境变量中. 如上所述, `tools/`文件夹位于 SDK 文件夹中。

- Linux 环境下,修改~/.bash_profile 或者 ~/.bashrc 文件. 找到环境变量设置的地方，加入 `tools/`的绝对路径。如果找不到该设置，你需要新添加一行:

  ```
  export PATH=${PATH}:<your_sdk_dir>/tools
  ```

- Mac 环境下,在你的 home 文件夹里面查找.bash_profile，然后和 linux 一样处理。如果之前没有.bash_profile 文件，你可以创建一个新的。
- Windows 环境下, 右击我的电脑, 并选择属性. 在标签页高级, 点击环境变量, 当对话框出现, 在系统变量栏目里双击路径（Path）.并添加 `tools/` 文件夹的完整路径.

添加 `tools` 到你的环境变量里，这样你可以运行 Android Debug Bridge (adb)和其他 tools 下命令，而不需要输入完整路径名。需要说明的是， 如果你升级你的 SDK， 需要将你的相应环境变量更新到新的位置。

安装 Eclipse 插件 (ADT)

如果你要使用 Eclipse IDE 作为开发 Android 应用的环境，你可以安装支持 Android 工程和工具的通用插件 Android Development Tools (ADT). ADT 插件包含强大的扩张，使得创建，运行和调试 Android 更快速，更简单。

如果你*不使用* Eclipse IDE，就不必下载和安装 ADT 插件

下载和安装 ADT 插件，请按照你们各自 Eclipse 版本步骤安装。

Eclipse 3.3 (Europa)

1. 开始 Eclipse，然后选择 **Help** > **Software Updates** > **Find and Install...**.
2. 对话框出现后，选择 **Search for new features to install** 点击 **Next**.
3. 点击 **New Remote Site**.
4. 在对话框中，输入远程站点的名字（如 Android Plugin），输入站点如下:

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

   点击 **OK**.
5. 你可以看到新的站点添加到搜索列表中(并检查)，点击 **Finish**.
6. 在下面的搜索结果对话框，选择复选框 **Android Plugin** > **Developer Tools**. 它将会检查特性: "Android Developer Tools", 和 "Android Editors". Android 编辑器的特性是可选的，但我们推荐安装它，如果你选择安装，需要前面提到的 WST 插件。点击 **Next**.
7. 阅读许可协议，然后选择接受许可协议，点击 **Next**.

Eclipse 3.4 (Ganymede)

1. 启动 Eclipse, 选择 **Help** > **Software Updates...**.
2. 在出现的对话框里，点击标签页 **Available Software**.
3. 点击 **Add Site...**
4. 输入下面的地址:

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

   点击 **OK**.
5. 返回可用软件的视图，你会看到这个插件. 选择下一步到 *Developer Tools* 并点击 **Install...**
6. 在接下来的安装窗口，选中 "Android Developer Tools"和 "Android Editors"。Android 编辑器特性是可选的，但是我们推荐安装它，如果你选择安装，需要前面提到的 WST 插件。点击 **Finish**.
7. 重启 Eclipse。

8. 点击 **Finish**. 9. ADT 插件没有签名，你可以点击 “**Install All**”来安装所有东西。 10. 重启 Eclipse.	

重启之后, **update your Eclipse preferences** 指向 SDK 文件夹:

1. 选择 **Window** > **Preferences...** 来打开属性面板。 (Mac OS X: **Eclipse** > **Preferences**)
2. 从左边面板选择 **Android**。
3. 在主界面上定位 SDK 点击 **Browse...** 然后定位 SDK 文件夹。
4. 点击 **Apply**, 然后点击 **OK**。

ADT 安装疑难解答

I 你如果按照以上步骤下载 ADT 插件有疑问，这里是一些建议：

- 在第四步，尝试改变远程更新地址 URL 为 `http`，而非 `https`.
- 如果你在防火墙保护下（企业防火墙）请确定你的 Eclipse 的代理设置合适。在 Eclipse 3.3/3.4，你可以从主 Eclipse 菜单中配置： **Window** (在 Mac, **Eclipse**) > **Preferences** > **General** > **Network Connections**

如果无法安装下载的 ADT 插件到 Eclipse,按照如下的步骤来从你的电脑下载和安装插件:

1. 下载 ADT 压缩文件 (不解压).
2. 按照默认安装的第一部和第二步(如上).
3. 在 Eclipse 3.3, 点击 **New Archive Site...**.
   In Eclipse 3.4, click **Add Site...**, 然后点击 **Archive...**
4. 浏览和选择已经下载的压缩文件。
5. 从第五步开始完成上面剩下的流程。

更新你的插件，你必须按照这些步骤代替默认的更行说明。

更新 ADT 插件

在某些情况下，你机器上的 SDK 可能和 ADT 插件是兼容的，你可以用以下步骤从 Eclipse 里更新 ADT 插件。

Eclipse 3.3 (Europa)	**Eclipse 3.4 (Ganymede)**

1. 选择 **Help** > **Software Updates** > **Find and Install...**.
2. 选择 **Search for updates of the currently installed features** 并点击 **Finish**.
3. 如果 ADT 可以更新，选择并安装更新。

或：

1. 选择 **Help** > **Software Updates** > **Manage Configuration**.
2. 展开导航树并选择 **Android Development Tools <version>**
3. **Available Tasks** 下选择 **Scan for Updates**。

1. 选择 **Help** > **Software Updates...**
2. 选择标签页 **Installed Software**。
3. 点击 **Update...**
4. 如果 ADT 允许更新,选择它并点击 **Finish**.

安装注意事项

Ubuntu Linux 注意事项

- 如果你需要帮助安装和配置 java 在你的 ubuntu 机器上，下面资源可能对你有帮助:
  - https://help.ubuntu.com/community/Java
  - https://help.ubuntu.com/community/JavaInstallation
- 这里有 java 和 Eclipsed 的安装步骤，先安装 Android SDK 和 ADT 插件.
  1. 如果你你开发机上使用 64 位版本，你需要用 `apt-get` 安装 `ia32-libs` 包

```
apt-get install ia32-libs
```

  2. 下一步,安装 Java:

```
apt-get install sun-java6-bin
```

3. Ubuntu 包管理器现在没有提供 Eclipse 3.3 版本下载，因此我们推荐你从 eclipse.org (http://www.eclipse.org/ downloads/)下载。 建议使用 Java 或 RCP 版本的 Eclipse.
4. 按照前面部分的提供步骤来安装 SDK 和 ADT 插件.

其他版本 Linux 注意事项

- 如果你在安装 Eclipse 的 ADT 插件遇到这个错误:

```
An error occurred during provisioning.
Cannot connect to keystore.
```

```
JKS
```

你的开发环境缺少适合的虚拟机，安装 Sun Java 6 可以解决这个问题，然后你再重新安装 ADT 插件。

- 如果 JDK 已经安装到你的开发电脑上，请确定其版本在这页的顶部列表中已经列出，有些 Linux 包含 jdk1.4 或者 java 的 gnu 编译器，以上二者 Android 不支持。

更新 SDK

本指南将帮助您升级您的开发环境和应用程序到 SDK 的最新版暄 如果你已经应用了以前版本的 Android SDK, 也需要使用本指南。
为了确保您的应用程序能够兼容 android1.0 系统，您需要安装新的 SDK 和用新的 API 移植现有的 android 应用程序，以下各节指导您完成这一进程。

安装新的 SDK

下载 SDK 并解压到一个安全的位置。
解压新的 SDK 以后 ，您应该完成下面的操作。

- 擦除你的模拟器的数据
  由于新的 SDK 版本发布，一些数据格式改变了。因此任何以前保存的模拟器数据必须清除。打开一个控制台/终端和操作 SDK 中的/to ols 目录。启动模拟器瘿 wipe-data 的选项
  Windows: `emulator -wipe-data`
  Mac/Linux: `./emulator -wipe-data`

- 更新您的 PATH 变量（Mac/Linux;可选）
  如果你以前设置 PATH 变量为指向的 SDK tools 目录，那么您必须更新以指向新的 SDK 的。E.g., `.bashrc` or `.bash_profile` file: `export PATH=$PATH:<your_new_sdk_dir>/tools`

更新 ADT Eclipse 插件

如果您在 Eclipse 上使用 ADT 插件开发，请按照下列步骤安装新的插件匹配新的 SDK。

Eclipse 3.3 (Europa)	Eclipse 3.4 (Ganymede)
1. 选择 **Help** > **Software Updates** > **Find and Install...**. 2. 选择 **Search for updates of the currently installed features** 并点击 **Finish**. 3. 如果任何 ADT 有效，选择并安装 4. 重启 Eclipse.	1. 选择 **Help** > **Software Updates...** 2. 选择 the **Installed Software** tab. 3. 点击 **Update...** 4. 如果任何 ADT 有效，选择并点击 Finish 5. 重启 Eclipse。

重新启动之后，更新您的 Eclipse 设置指向 SDK 目录。

1. 选择 **Window** > **Preferences...** to open the Preferences panel. (Mac OSX: **Eclipse** > **Preferences**)
2. 选择 **Android** from the left panel.
3. 对于在主面板中 SDK 的定位，点击 **Browse...** 并找到 SDK 的目录。
4. 点击 **Apply**, 然后 **OK**.

建立应用程序签名

所有应用程序在安装它们之前都必须被签名。ADT 插件和 ant 为基础的开发工具都支持这一要求，它们通过带一个调试 KEY 瘿 apk 文件来签发编译。为了做到这一点，编译工具使用包括在 JDK 的 Keytool 去创造一个 keystore 和带着一个已知的别名和密码一个 key 带着一个已知的别名和密码。如需详细信息，请查阅 签名你的应用程序.
为了支持签签名，你应该首先确认 Keytool 对于 SDK 的编译工具是有效的。在大多数情况下，你可以告诉的 SDK 编译工具如何找到 Keytool，通过设置你的 J AVA_HOME 环境变量设置和一个合适的 JDK。另外，您也可以添加 keytool 的 JDK 版本到您的 PATH 变量
如果你正在开发 Linux 的一个版本，那原本使用的是 GNU 的 JAVA 编译喻，请确保该系统正在使用的 Keytool 的 JDK 版本，而不是使用 gcj,如果 keyt ool 已经在您的路径，

它可能是指向在一个符号链接是/usr/bin/keytool 。在这种情况下，检查符号链接的目标，以确保它指向正确的 Keytool.
如果您使用的 ant 编译你的.apk 文件ضو而不是 ADT，你必须重新产生你的 build.xml 文件。为了做到这一点，请执行下列步骤：

1. 在您的 android 应用程序工程目录中，找到并删除目前的 build.xml 文件
2. 2. 运行 activitycreator ，直接输出到包含您的应用程序项目的文件夹

```
3.  - exec activitycreator
```

```
--out  your.activity.YourActivity
```

运行这种方式ضactivityCreator 不会擦出或创建新的 Java 文件（或 manifest 文件ضو，对于那些已经存在的 activity 和 package。重要的是，package 和 activity 是真实存在的。该工具创建一个新的 build.xml 文件，以及一个新的目录称 libs"中，这个目录将放置第三方 jar 文件，这是你就能够使用 ant 脚本自动处理。

移植您的应用程序

更新过您的 SDK 以后 ，您可能会遇到破损的代码，由于框架和 API 的变化。您需要更新您的代码以匹配变化的 Andriod 的 API。
一种方法是用 Eclipse 打开您的项目和查看你的应用程序中 ADT 的标记错误。从这里，你可以查找对应的变势 变化预览 and API 变化报告.
如果您更新您的代码有其他麻烦，请访问 android 小组讨论 寻求帮助或者求助于其他 android 开发人员.
如果已经修改了一个 ApiDemos 应用程序，并希望移植到新的 SDK 的，请注意您将需要卸载模拟器中预装的 ApiDemos 版本。了解更多信息，或（运行或安装 A piDemos）遇到一个重新安装"的错误，见疑难解答论颧 因为签名错误，我不能在我的 IDE 中安装 ApiDemos 应用程序 来获得解决这个问题的信息。

开发和调试

这一节将介绍在 android 上开发调试应用程序。它将教会我们如何创建，编译，运行以及调试 android 代码。或者，你也可以从 Hello Android tutorial.开始

主要内容

1. **在 eclipse 上开发 android 应用程序**

2. 利用其他 IDE 和工具开发 android 应用程序
3. 给应用程序签名
4. ApiDemo 示例程序用法
5. 调试
6. 设备上的调试和测试设置
7. 顶端调试技巧
8. 编译安装一个 android 应用程序
9. 移除 android 程序
10. Eclipse 技巧

在 eclipse 上开发 Android 应用程序

在用 eclipse IDE 开发 android 应用程序之前，你首先要创建一个 Android 工程，并且建立一个启动配置，在此之后你才可以开始编写，运行，以及调试你的应用程序。
以下章节是假设你已经在 eclipse 环境中安装了 ADT 插件，如果你没有安装，请安装之后再使用以下说明。参考 安装 eclipse 插件（ADT）

创建一个 android 工程

ADT 提供了一个新的工程向导，你可以快速的创建一个新的工程或者在现有代码上创建工程。创建工程的步骤如下：
选择 **File** > **New** > **Project**

1. 选择 **Android** > **Android Project**, 然后按下 **Next**
2. 选择项目内容:
   - 选择 **Create new project in workspace**， 为编码创建一个全新的工程。输入工程名称（project name），基础软件包的名称（the base package name），以及 Activity 类的名称。以创建 stub .java 文件等文件和程序名字。
   - 选择 **Create project from existing source** ，为已有代码创建一个工程。如果你想编译运行 SDK 中提供的示例程序，可以使用这个选项。示例程序的存放在 SDK 的 samples/目录下。
     浏览包含已有代码的目录，点击 ok,如果目录中包含有可用的 android manifest 文件，ADT 将为你填写合适的软件包，activity，和应用程序名称。
3. 按下 **Finish**.

ADT 插件会根据你的工程类型创建合适的文件和文件夹，如下:

- src/ 包含 stub .java Activity 文件的文件夹.

- res/ 资源文件夹.
- AndroidManifest.xml 工程清单.

创建一个启动项

能够在 eclipse 上运行调试应用程序之前，你必须为它创建一个启动项。启动项指定哪个工程将被启动，哪个 activity 开始工作，以及使用哪些模拟器选项等。
按照以下步骤为 Eclipse 版本的应用程序创建合适的启动项：

1. 打开启动项管理工具。
    - 在 Eclipse 3.3 (Europa)的版本中，酌情选择 **Run** > **Open Run Dialog...** or **Run** > **Open Debug Dialog...** 。
    - 在 Eclipse 3.4 (Ganymede)版本中，酌情选择 **Run** > **Run Configurations...** or **Run** > **Debug Configurations...** 。
2. 在左边的工程类型列表选择 Android Application 选择，双击（或者点击右键选择 new）,创建一个新的启动项。
3. 输入启动项名称。
4. 在 Android 标签中，浏览要开始的工程和 Activity 。
5. 在 Target 标签中，设置想要显示的屏幕及网络属性，以及其他任何模拟器启动选项。
6. 你可以在 Common 标签中设置更多的选项.
7. 按下 Apply 保存启动配置，或者按下 Run 或 Debug（）。

运行和调试应用程序

一旦你设定了工程和工程启动配置，你就可以按照以下的说明运行和调试应用程序了。
从 eclipse 主菜单，根据情况选择 Run>Run 或者 Run>Debug，开始运行或者调试活动启动项。
注意，这里活动启动项是在运行配置管理中最最近一次选中的那个。它不一定就是在 Eclipse Navigation 面板中选择的程序（如果有的话）
设置和修改活动启动项，可以使用启动项管理工具。如何获得启动项管理工具可以参考创建一个启动项
运行或调试应用程序将触发以下动作:

- 启动模拟器，如果他还没有开始运行。
- 编译工程， 如果在上次编译的基础上修改过代码，将重新编译。在模拟器上安装应用程序。
- Run 选项，开始运行程序。

- **Debug** 在"Wait for debugger "模式下启动程序,然后打开调试窗口并将 Eclipse Java 调试器和程序关联。

利用其他 IDEs 和工具开发 Android 应用程序

通常我们使用安装有 ADT 插件的 eclipse Eclipse with the ADT plugin.来开发 Android 程序，这个插件将编辑，build 和调试功能集成到 IDE 上。
然而，如果你想在其他的 IDE 上开发程序，例如 IntelliJ，或者使用没有 ADT 插件的 eclipse 也可以。SDK 提供了安装，编译，调试应用程序所需要的工具。

创建一个 android 工程

Android SDK 包含一个 activityCreator 的程序，它将为工程产生多个 stub 文件和一个 build 文件。你可以用这个程序创建一个新的 Android 工程或者在现有代码上创建工程，如 SDK 中包含的例子。对于 Linux 和 Mac 系统，SDK 提供 activityCreator.py,一个 Python 脚本，Windows 上则是 activityCreator.bat 一个批处理脚本。无论是哪种平台，用法是一样的。
按以下步骤运行 activityCreator 创建 Android 工程:

1. 在命令行下，切换到 SDK 下的 tools/目录下，为你的工程文件新建一个目录。如果你是在现有代码上创建工程，切换到程序的根目录下。
2. 运行 activityCreator。在命令行下，你必须指定完全合格的类名作为参数。如果你是创建一个全新的工程，这个类代表的与它同名的 stub 类和脚本文件。如果是在现有代码上创建工程，必须指定软件包中其中一个 Activity 类的名称。 命令选项的脚本包括：
    - `--out <folder>` 设定输出目录。默认情况下输出目录为当前目录。如果你想为工程文件创建一个新的目录，可以使用这个选项来指向它。
    - `--ide intellij`, 在一个新的项目中生成 IntelliJ IDEA 工程文件。

这里有个例子：

```
~/android_linux_sdk/tools $ ./activityCreator.py --out myproject
your.package.name.ActivityName

package: your.package.name

out_dir: myproject

activity_name: ActivityName
```

```
~/android_linux_sdk/tools $
```

activityCreator 脚本生成以下文件和目录（但是不能重写已有文件）：

- `AndroidManifest.xml` 程序的清单文件，同时为工程指定 Activity 类。
- `build.xml` 一个 Ant 文件，用来编译/打包应用程序。
- `src/your/package/name/ActivityName.java` 你指定的输入 Activity 类。
- *`your_activity`*`.iml, `*`your_activity`*`.ipr, `*`your_activity`*`.iws` [*only with the `-ide intelliJ` flag*] intelliJ 工程文件
- `res/` 资源目录.
- `src/` 源代码目录.
- `bin/` build 脚本的输出目录.

现在你可以将开发文件夹移到任何地方，但是记住，必须使用 tool/文件夹下的 adb 程序将文件发送到模拟器上。因此你需要在你工作环境和 tools/文件夹之间活动。
当然你需要避免移动 SDK 目录，因为它将打断编译脚本。（再重新 build 之前需要手动更新 SDK 的映射路径）

编译 android 应用程序

使用 activityCreator 生成的 Ant 文件 build.xml 来编译程序

1. 如果你没有，你可以通过 Apache Ant home page 得到 Ant 文件。安装它，并确定它在你的可执行文件路径下。
2. 呼叫 Ant 之前，你需声明 JAVA_HOME 环境变量，并将它设置为 JDK 的安装路径。

   > 注意：在 windows 上，JDK 默认的安装路径为"Program Files"，这个路径将会引起 Ant 失败，因为路径中间有空格。解决这个问题，你可以像这样指定环境变量 JAVA_HOME：`JAVA_HOME=c:\Prora~1\Java\` 然而简单的解决方法是将 JDK 安装在没有空格的目录下。例如：`c:\java\jdk1.6.0_02`.

3. 如果你还没有这么准备好，按照上面创建一个新的工程的介绍建立一个工程。

4. 现在你可以为你的工程运行 Ant 编译文件，只需在 build.xml 同文件夹下输入 ant 即可。每次修改原文件或是资源，都需要重新运行 ant，它将把最新版的应用程序打包以便 deploy.

运行 Android 程序

运行一个编译好的程序,你需要用 adb 工具将.apk 文件加载到模拟器的/data/app/目录下，用法如下面介绍。

1. 启动模拟器（命令行下运行 sdk 目录下的/tools/emulator）。
2. 模拟器切换到主画面（最好不要在程序运行的时候向模拟器安装程序，可以按 home 键离开应用程序）。
3. 运行 adb，安装 myproject/bin./<appname>.apk 文件。例如，安装 Lunar Lander 示例，命令行下，切换到 SDK 目录下的/sample/LunarLander 子目录下，输入../../tools/adb install bin/LunarLander.apk
4. 在模拟器中，打开可执行程序列表，卷动屏幕，选中并启动你的应用程序。

> 注意：当你第一次安装一个 Activity 时，你可能需要在启动项显示之前，或者其它程序调用它之前重新启动模拟器。因为软件包管理工具通常只有在模拟器启动时才能完全的审查 manifests。

为程序附加调试器

这一节我们介绍如何在屏幕上显示调试信息（例如 CPU 使用率），以及如何将 IDE 和模拟器上运行的程序关联起来。
使用 eclipse 插件可以自动的生成调试器。但你也可以通过配置 IDES 来监听调试端口得到调试信息。

1. 启动 **Dalvik Debug Monitor Server (DDMS) 工具**，它在 IDE 和模拟器之间扮演着端口转换服务的角色。？
2. 设置模拟器调试配置选项。例如，等到调试信息被加载后才启动应用程序。注意，很多调试选项无需 DDMS 也可以使用，例如模拟器上显示 CPU 的使用效率，或者屏幕的刷新频率。
3. 配置 IDE，使得调试时 IDE 与 8700 端口关联 .how to set up Eclipse to debug your project. 包含以下信息。

配置 IDE 附加调试端口

DDMS 将为每一个虚拟机分配一个特殊的调试端口，这个端口在模拟器上可以找到。你必须将你的 IDE 与此端口（虚拟机上信息栏中有列出这些端口）关联或者是默认的端口 8700。这样可以使 IDE 连接到模拟器上程序列表中的任一个程序。
你的 IDE 需要能够关联模拟器上正在运行的程序，显示它的线程，并允许你挂起它，检查它的状态，设置断点。如果你在开发设置面板选择了“等待调试”，应用程序将等到 Eclipse 连接后才运行，所以你需要在连接之前设置断点。
修改正在调试的程序，或者在当前程序运行时选择“等待调试”将引起系统杀死这个应用程序。如果你的程序处于一种坏的状态，你可以使用方式杀死它，方法很简单，只需要设置和钩掉复选框。

应用程序签名

Android 系统要求所有的程序经过数字签名才能安装，如果没有可用的数字签名，系统将不许安装运行此程序。不管是模拟器还是真实设备，只要是 android 系统，这都适用。鉴于此原因，在设备或者是模拟器上运行调试程序之前，你必须为你的应用程序设置数字签名。
理解 android 程序签名的重要几点：：

- 所有的程序都必须签名，没有被签名的程序，系统将不能安装。
- 你可使用自签署证书签署你的应用程序，必须是无凭证授权是的。
- 系统仅仅会在安装的时候测试签名证书的有效期，如果应用程序的签名是在安装之后才到期，那么应用程序仍然可以正常启用。
- 你可以使用标准工具-Keytool and Jarsigner-生成密钥，来签名应用程序的.apk 文件。

Android SDK 工具可以帮助你在调试时给应用程序签名。ADT 插件和 Ant 编译工具都提供了两种签名模式-debug 模式和 release 模式

- debug 模式下，编译工具使用 JDK 中的通用程序 Keytool 通过已知方法和密码创建秘锁和密钥。每次编译的时候，工具使用 debug 密钥签名应用程序的.apk 文件。因为密码是已知的，工具不需要在每次编译的时候提示你输入密锁和密钥。
- 当你的应用程序已经准备 release 了，你可以在 release 模式下编译。release 模式下，工具编译时不会将.apk 文件签名。你需要用 Keytool 生成密钥和密锁，再用 JDK 中的 Jarsigner 工具给.apk 文件签名。

签名基本设置

为了支持生成密锁和密钥，你首先要确定 Keytool 在 SDK 编译工具中是有效的。在很多情况下，你可以设置 JAVA_HOME 环境变量，告诉 SDK 如何找到 Keytool，或者你可以在 PATH 变量中添加 Keytool 的 JDK 版本。

如果你是在 linux 版本中开发，原本是来自 Java Gnu 编译器，请确定系统用的是 Keytool 版本的 JDK，而不是 gcj 版本的。如果 Keytool 已经在 PATH 中，它将指向符号连接 /usr/bin/keytool。这种情况下，核实符号连接的目标是指向 JDK 下的 Keytool

Eclipse/ADT 中的签名

如果你是在 Eclipse 下开发，并已经按照上面所介绍的安装了 Keytool，默认情况下是可以在 debug 模式下签名的。当你运行调试程序的时候 ADK 将给.apk 文件签名，并安装到模拟器上。这部分不需要特殊的动作，ADT 已经进入 Keytool
在 release 模式下编译程序，在 Package 面版上按 project 右键，选择 Android Tools>Export Application Package.或者你可以点击 Manifest Editor， overview 页面上的“Exporting the unsigned .apk”连接 ，导出未签名 apk 文件。保存.apk 文件之后，用 Jarsigner 及你自己的密钥给 apk 文件签名 ，如果没有密钥， 你可以用 Keystore 创建密钥和密锁。如果已经有一个密钥了，如公共密钥，就可以给.apk 文件签名了。

Ant 签名

如果用 Ant 编译.apk 文件，假设你使用最新版的 SDK 中包含的 activitycreator 工具生成 build.xml 文件，默认情况下可以使用 debug 签名模式。当你运行 Ant 对 build.xml 编译程序，build 脚本将生成密锁和密钥并签名.apk 文件。这部分不需要做其它特殊的动作。
release 模式下编译程序，你需要做的是在 Ant 命令中指定编译目标“release”。例如，如果是在 bulid.xml 所在目录下运行 ant,输入以下命令：
ant release
build 脚本编译程序时并没有签名。编译完.apk 文件后，你需要用 Jarsigner 和你自己的密钥给.apk 文件签名。如果没有密钥， 你可以用 Keystore 创建密钥和密锁。如果已经有一个密钥了，如公共密钥，你就可以给.apk 文件签名了。

调试证书期限

自签名证书用于程序的 debug 模式下（默认情况下是 Eclipse/ADT 和 Ant builds）,自它创建时间起有一年的期限。
当证书到期时，将会有编译错误。 And 下错误显示如下：

```
debug:

[echo] Packaging bin/samples-debug.apk, and signing it with a debug
key...
```

```
[exec] Debug Certificate expired on 8/4/08 3:43 PM
```

在 Eclipse/ADT 下，你可以看到类似的错误。
解决这个问题的简单方法是删除 debug.keystore 文件。Linux/Mac OSX 下这个文件保存在~/.android 下，windows XP 下，文件保存在 `C:\Documents and Settings\<user>\Local Settings\Application Data\Android`。windows Vista 下文件保存在 `C:\Users\<user>\AppData\Local\Android`。

下次编译时，编译工具将生成新的密锁和密钥。
注意：如果你的开发设备使用的是 non-Gregorian locale,编译工具经常错误的生成一个过期的调试证书，因此编译的时候你会得到错误提示。 对于解决信息，请参见疑难解答专题 I can't compile my app because the build tools generated an expired debug certificate.

使用 ApiDemo 示例应用程序

Android SDK 包含了一套示例程序，他们验证了许多功能以及 API 的用法。ApiDemos 软件包被提前安装在模拟器中，所以你可以启动模拟器，在主画面的应用程序抽屉里打开它。
你也可以在<SDK>/samples/ApiDemos 中找到源码，可用看看它，学习 Demo 的实现方法。
如果你愿意，你还可以将 ApiDemo 的示例程序作为一个工程加载进来，修改并在模拟器上运行。然而，在这之前你首先要卸载之前已经安装的 ApiDemos。如果你没有移除之前安装的版本而直接在开发环境中运行或修改 ApiDemos,将会有安装错误。
关于如何卸载和重装 ApiDemo，可以参考 I can't install ApiDemos apps in my IDE because of a signing error.这样你就可以在你的开发环境中工作了。

调试

Android 有相当广泛的一套工具帮助你调试你的应用程序：

- **DDMS** -一个生动的程序，它支持端口转换（因此你可以在 IDE 中给你的代码下端点），支持抓取模拟器屏幕，线程和堆栈信息，以及许多其他功能。你还可以运行 logcat 重新获得 Log 信息。点击此连接查看更多信息。
- **logcat**- 转储系统信息，这些信息包括，模拟器抛出错误时堆栈的运行过程以及日志信息。运行 logcat,点击此连接。

```
- ...
```

```
- I/MemoryDealer( 763): MemoryDealer (this=0x54bda0): Creating 2621440 bytes heap at 0x438db000
- I/Logger( 1858): getView() requesting item number 0
- I/Logger( 1858): getView() requesting item number 1
- I/Logger( 1858): getView() requesting item number 2
- D/ActivityManager( 763): Stopping: HistoryRecord{409dbb20 com.android.home.AllApps}
```

```
...
```

- **Android Log** - 输出模拟器上 log 文件信息日志类。如果你在 DDMS 上运行了 logcat，你可以实时阅读这些信息。在你的代码中添加 logging 方法的调用。使用 log 类，你可以根据你想获得信息的重要程度不同调用 Log.v(verbose),Log.d()(debug),Log.i()(information),Log.w()(warning)或者 Log.e(error).来分派 log 信息 `Log.i("MyActivity", "MyClass.getView() — Requesting item number " + position)` 你可以用 logcat 阅读这些信息。
- **Traceview** - Android 可以将函数的调用情况以及调用时间保存到一个 log 文件中，你可以用图形阅读器 Traceview 查看详细内容。更多信息查看这个连接下的主题
- **Eclipse plugin** -Eclipse 插件整合了相当数量的工具（ADB，DDMS，logcat output，以及其它功能），点击此连接查看更多信息。
- **Debug and Test Device Settings** -Android 揭示了很多有用的设定，例如 CPU 使用率和 帧速率，参看下面的 Debug and Test Settings on the Emulator

Also, see the Troubleshooting section of the doc to figure out why your application isn't appearing on the emulator, or why it's not starting.
此外，参看疑难解答这一节文档，以找出您的应用程序为什么没有出现在模拟器上，或为什么不开始。

设备上的调试和测试设置

Android 允许你设置多个设定以便你测试和调试程序。获得模拟器的开发设置，可以选择 Dev Tools>Development Settings。按照以下选项将打开开发设置页（或其中之一）：

- **Debug app** 选择要被调试的程序，你不需要设置这个来关联调试器，但是这个变量有两个作用：
  - 防止 Android 在调试的断点处长时间停留时抛出错误。
  - 允许你选择 Wait for Debugger 选项来暂停程序启动，直到调试器被关联上（如下介绍）
- **Wait for debugger** 阻塞程序加载直到关联上调试器。这样你可以在 onCreate()中设置端点，这在调试 Activity 的启动进程时很重要。当你改变这个选项，任何当前运行的程序实例将被杀死。为选中此框，你必须如上面介绍的选择一个调试程序。这和在代码中添加 waitForDebugger()是一样的。
- **Immediately destroy activities** 告诉系统只要 activity 停止了就销毁它。 (犹如 Android 必须回收内存). 这个在测试 onSaveInstanceState(Bundle) / onCreate(android.os.Bundle)代码路径 时非常有用，否则将难以生效.选择这个选项可能带来很多问题，因为他们没有保存程序的状态。
- **Show screen updates** 选中这个选项时，屏幕上任何被重绘的矩形区域会闪现粉红色。这对于发现屏幕不必要的绘图很有用。
- **Show CPU usage** 在屏幕顶端显示一个 CPU 进度，显示 CPU 的使用情况。上面红色栏显示总的 CPU 使用率，下方绿色栏显示目前画面的 CPU 使用时间。注意：一旦打开次功能就不能关掉，除非重新启动模拟器。？？？
- **Show background** 没有 activity 屏幕显示时显示背景面板，这个通常在调试的时候才会发生。

模拟器重起后这些设置仍被记忆。

顶端调试技巧

快速堆栈转储

从模拟器上获得堆转储，你可以登录 adb shell，用"ps"命令找到你想要的进程，然后用"kill-3"，堆栈使用轨迹将显示在 log 文件中。

在模拟器屏幕上显示有用信息

设备可以显示一些有用信息，例如 CPU 使用率，以及高亮显示重绘区域。可以在开发设定窗口打开和关闭这些功能。Setting debug and test configurations on the emulator.中有详细介绍。

你可以通过 Dalvik Debug Monitor Service 工具获得转储状态信息。请参考 adb 中介绍的 dumpsys and dumpstate

获得模拟器中应用程序状态信息(dumpsys)

你可以通过 Dalvik Debug Monitor Service 工具获得 dumpsys 信息。参考 adb 中介绍的 dumpsys and dumpstate 。

获得无线连接信息

你可以通过 Dalvik Debug Monitor Service 工具获得无线连接信息。在 Device 菜单中选择"Dump radio state"

记录跟踪数据

你可以在 activity 中通过调用 android.os.Debug.startMethodTracing()来记录函数的调用以及其它跟踪数据。详细的参考 Running the Traceview Debugging Program 。

记录无线数据

默认情况下系统不记录无线数据（数据很多）。然而，你可以用下面的命令记录无线数据：

```
adb shell

logcat -b radio
```

运行 adb

Android 有 adb 工具，他提供了许多功能，包括移动和同步文件到模拟器上，改变端口，在模拟器上运行 UNIX shell。 详见 Using adb。

获得模拟器屏幕截图

Dalvik Debug Monitor Server (DDMS)可以抓取模拟器屏幕截图。

使用调试帮助类

Android 为方便使用提供了调试帮助类，例如 util.Log 和 Debug

编译安装 Anroid 应用程序

Android 要求专门的编译工具可以正确的编译资源文件和应用程序的其他部分，因此，你必须为你的应用程序建立一个专门的编译环境。

专门 Android 编译器编译步骤包括，编译 XML 和其他资源文件并创建合适的输出格式。

编译好的 Android 应用程序是一个.apk 压缩文件，它含有.dex 文件，资源文件，原 data 文件，以及其他文件。你可以通过 scratch,或者源文件构造一个合适的 Android 工程。

Android 目前不支持的在本地代码上开发第三方应用程序。

比较推荐的 Andriod 应用程序开发方法是 use Eclipse with the Android plugin，它支持编译，运行，调试 Android 应用程序。

如果你还有其他 IDE，Android provides tools for other IDEs 可以编译运行 Android 应用程序，但是他们不是很完整。

移出一个 Android 应用程序

移出一个安装在模拟器上的应用程序，你需要执行 adbrun adb 删除.apk 文件。.apk 文件是在安装的时候发送到模拟器上的。使用 adb shell 进入设备的 shell，切换到 data/app 目录下，用 rm 命令删除 apk 文件 ： rm your_app.apk。用法在连接中介绍。

Eclipse 技巧

在 Eclipse 上执行任意 java 代码

在 Eclipse 上，当程序停在断点处时你可以执行任意代码。例如，在一个含有“zip”字符串参数的函数中，你可以获得软件包信息，调用类方法。你也可以执行任意静态方法：

如，输入 `android.os.Debug.startMethodTracing()` ，启动 dmTrace。

打开代码执行窗口，主菜单中选择 **Window>Show View>Display**，打开显示窗口，一个简单的文本编辑器。输入你的代码，高亮显示文字，单击'J'图标（或者 CTRL + SHIFT + D）运行代码。代码在被选线程的上下文中运行， 而这个线程必须是停在断点处或者单步停止点。（如果你手动挂去线程，你必须单步执行。线程停在 Object.wait()是没有用的）。
如果你目前是停在断点，你可以简单的按下（CTRL + SHIFT + D）高亮并执行一段代码。
你可以高亮同一选中区域的文字，通过按下 ALT +SHIFT + 向上/向下箭头来改变所选区域的大小
下面是一些例子，输入内容和 eclipse 显示窗口的回应信息。

Input	Response
`zip`	`(java.lang.String) /work/device/out/linux-x86-debug/android/app/a`
`zip.endsWith(".zip")`	`(boolean) true`
`zip.endsWith(".jar")`	`(boolean) false`

你也可以利用剪贴板在不调试时插入执行代码。在 eclipse 文档中查找"scrapbook"相关。

手动运行 DDMS

虽然推荐用 ADT 插件调试程序，但是你也可以手动运行 DDMS，配置 Eclipse 以便在 8700 端口上调试程序（注意：首先确定你启动了 DDMS）。

增加 JUnit 测试类

在 Eclipse/ADT ,你可以在程序中添加 JUnit 测试类，然而，测试运行正常之前你需要设置专门的 JUnit 配置，
关于如何设置 JUnit 配置的详细细节，参看请参见疑难解答专题 I can't run a Junit test class in Eclipse。

Hello, Android!

作为开发者，你的第一感觉是你拿到这个开发框架写“Hello, World!”程序时的难易程度给你留下的。 当然，在 Android 里，这个非常容易，下面我给你演示:

- 创建一个工程
- 创建 UI
- 运行代码: Hello, Android

下面的章节将详细描述

- 使用 XML 构建 UI
- 调试工程
- 不使用 Eclipse 创建一个工程

让我们开始把

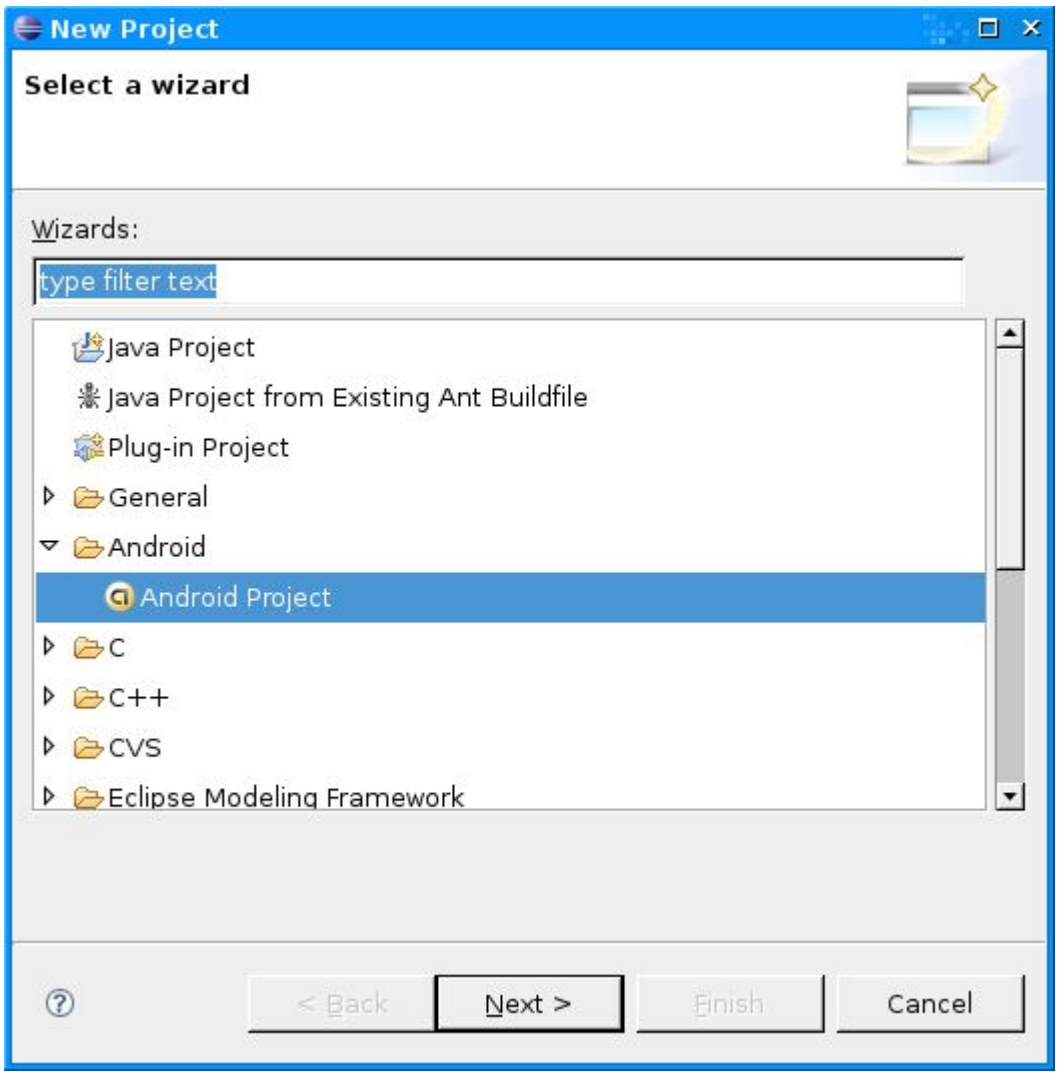
创建工程

创建一个尽可能简单的工程，Eclipse 的插件可以使 Android 的开发变得简单。
你需要一台装有 Eclipse IDE (参考系统和软件需求)，你还需要安装 Android Eclipse 插件 (ADT)。 如果你都准备好了，继续看这里。
首先，你需要对如何创建“Hello, World!”有个大概的了解：

1. 在菜单中 **File > New > Project** 中创建一个新的 Android 项目。
2. 在创建 Android 项目的对话框里填写项目的详细信息。
3. 编辑自动生成代码的模板去显示一些输出。

这样就可以了，下一步，我们将详细讲解每一步。

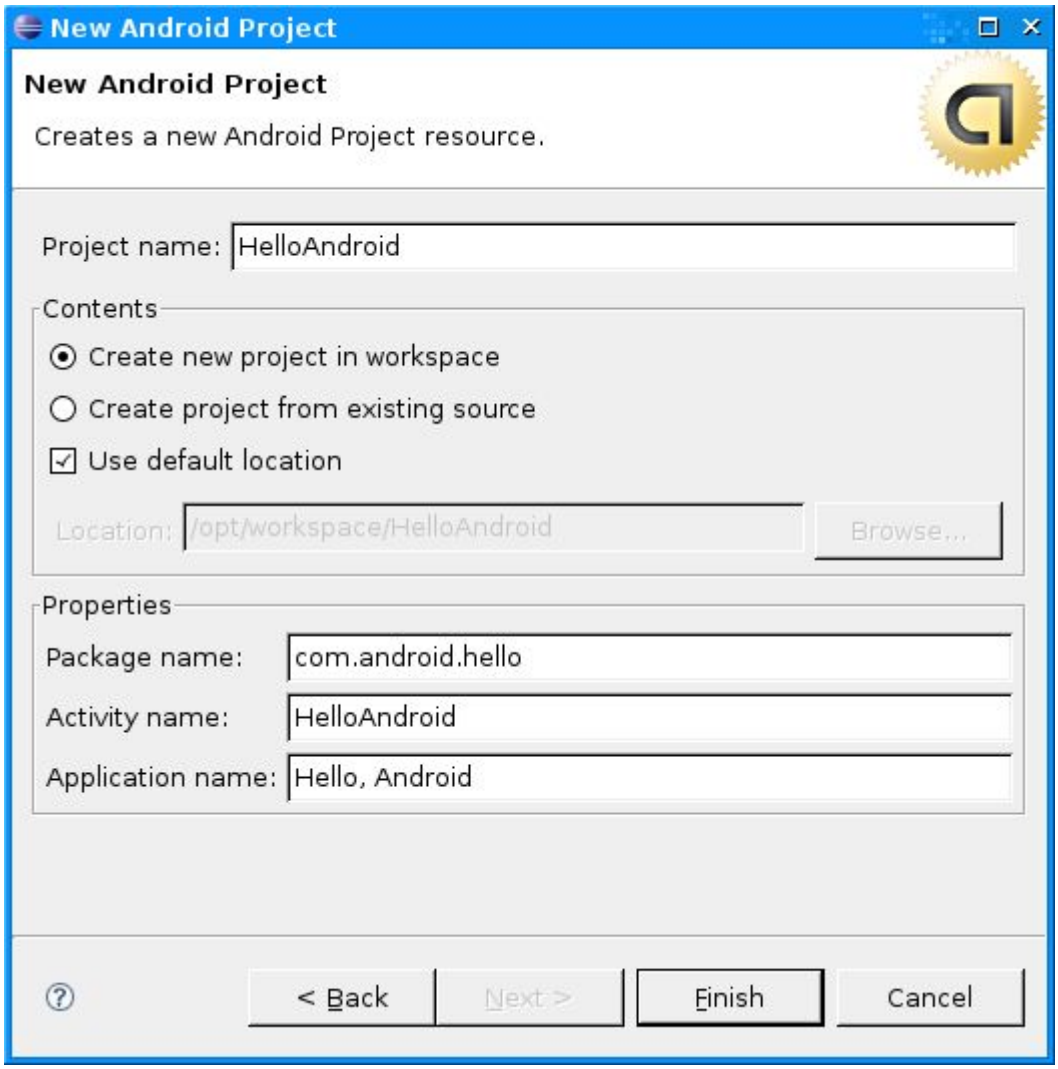
1. **创建一个新的 Android 项目**
   打开 Eclipse，选择 File > New > Project 如果 Android 的 Eclipse 的插件正确安装了，弹出的对话框里就会有一项“Android”，这一项有唯一的子项 “Android Project”。



选中**“Android Project”**，点下一步。

2. 填写工程的详细信息

下一步需要你填写项目的详细信息，这里有个例子：



每一项具体的意思：

工程名	你想把工程保存在你机器上哪个目录里
包名	>包的命名空间（需要遵循 java 编程语言的命名规则）,你的所有代码都会在这个命名空间下。这也会生成包的名称从而活动自动生成。你使用的这个包的名字必须和其他安装在你机器上的包名字不冲突，所以，使用一个标准规则的包名字是非常重要的。如上例，我们使用包的名字为“com.android”，但你需要使用一个不同的类型。
“活动”名	这是插件为你自动生成类的名字。它也会是 Android 活动类的一个子类。一个活动仅仅是一个包含一些功能并能执行的类。如果它选择，可以创建用户界面,但这不是必须的。

程序名	这是最后生成应用程序的名字。

可选框里**"Use default location"** 允许你选择其他目录保存项目产生的其它文件。

3. **编辑自动生成代码**
   运行插件后，会自动生成一个类“HelloAndroid ”(可以在程序包里 HelloAndroid > src > com.android.hello 找到)。 像这样:

```
public class HelloAndroid extends Activity {

    /** 活动第一次被创建后调用。 **/

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);

    }

}
```

现在，你可以执行程序了。不过我们还可以进一步研究下，这样我们能更好得理解程序。所以，下一步我们可以改一些代码

建立 UI

看下面我们修改的代码，你可以在你的 HelloAndroid.java 文件里做相同的修改，我们来逐行分析:

```
package com.android.hello;


import android.app.Activity;

import android.os.Bundle;

import android.widget.TextView;
```

```
public class HelloAndroid extends Activity {

   /** 活动第一次被创建后调用 */

   @Override

   public void onCreate(Bundle savedInstanceState) {

      super.onCreate(savedInstanceState);

      TextView tv = new TextView(this);

      tv.setText("Hello, Android");

      setContentView(tv);

   }

}
```

小提示: 如果你忘记引入 TextView 的包，可以尝试 **Ctrl-Shift-O** (如果是 Mac 系统 **Cmd-Shift-O**)。 这是 Eclipse 管理应用的快捷方式-它会显示没有找到的包然后自动为你加上。

在 Android 里，用户接口由一些称之为视图的不同层次的类组成。 一个视图就是一个简单的对象。如单选框，动画控件，一个文本框（我们的例子里的），我们称处理文本的这样一个子视图就叫 TextView。这里教你如何创建 TextView。
这里教你如何创建 TextView:

```
TextView tv = new TextView(this);
```

TextView 构造器就是 Android 上下文的实例，这个上下文仅仅是指向系统的一个句柄，它提供像资源处理之类的服务。包含一些进入资料库以及参数选择的入口。这个活动也是继承上下文。 HelloAndroid 类是活动的一个子类，它也是一个上下文，我们能通过 `this` 操作 TextView。

创建 TextView 后，加入需要显示的内容：

```
tv.setText("Hello, Android");
```

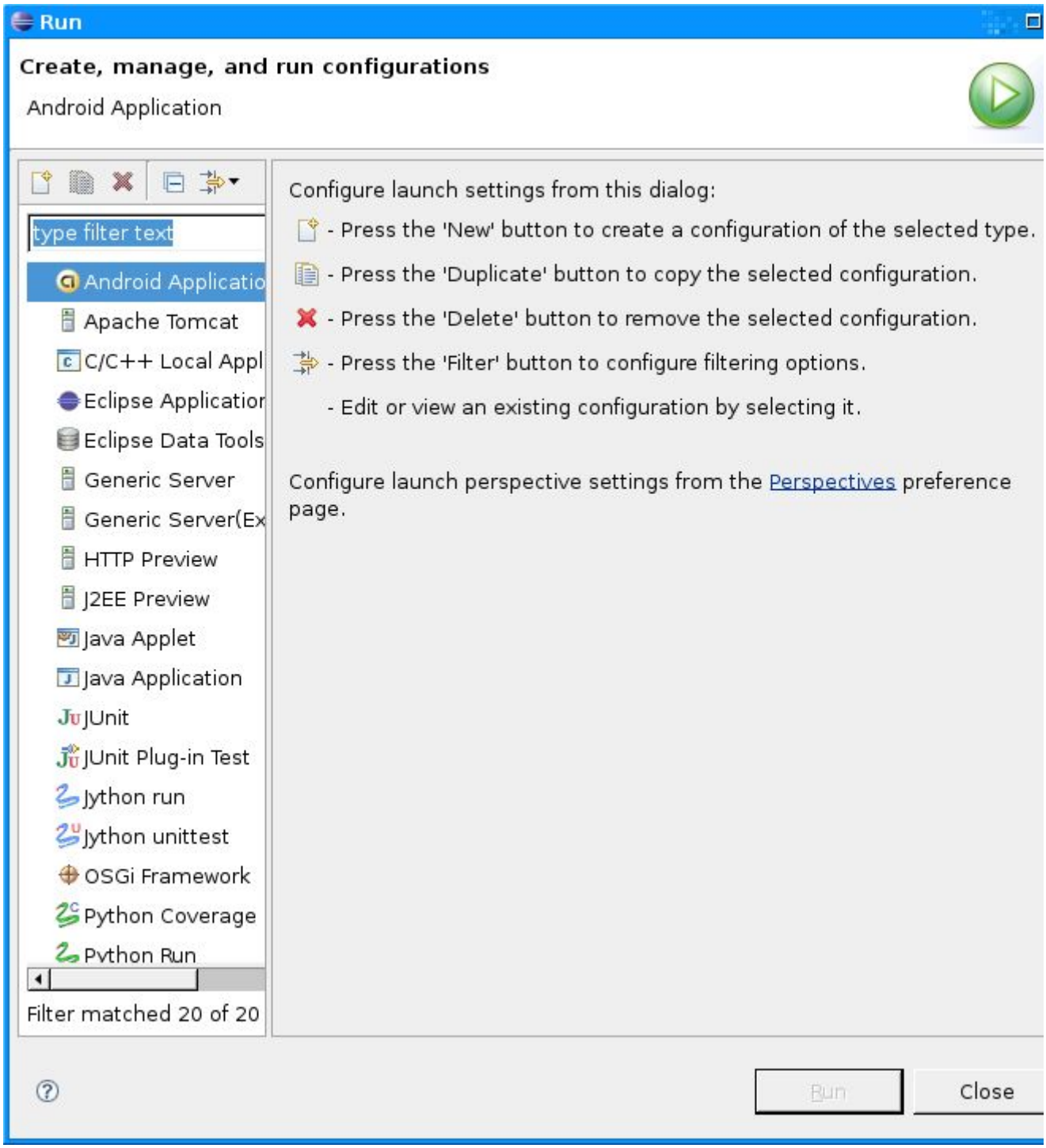
这里很正常。
我们创建一个 TextView，然后告诉它显示的内容。最后一步就是让 TextView 在屏幕上显示出来，像这样：

```
setContentView(tv);
```

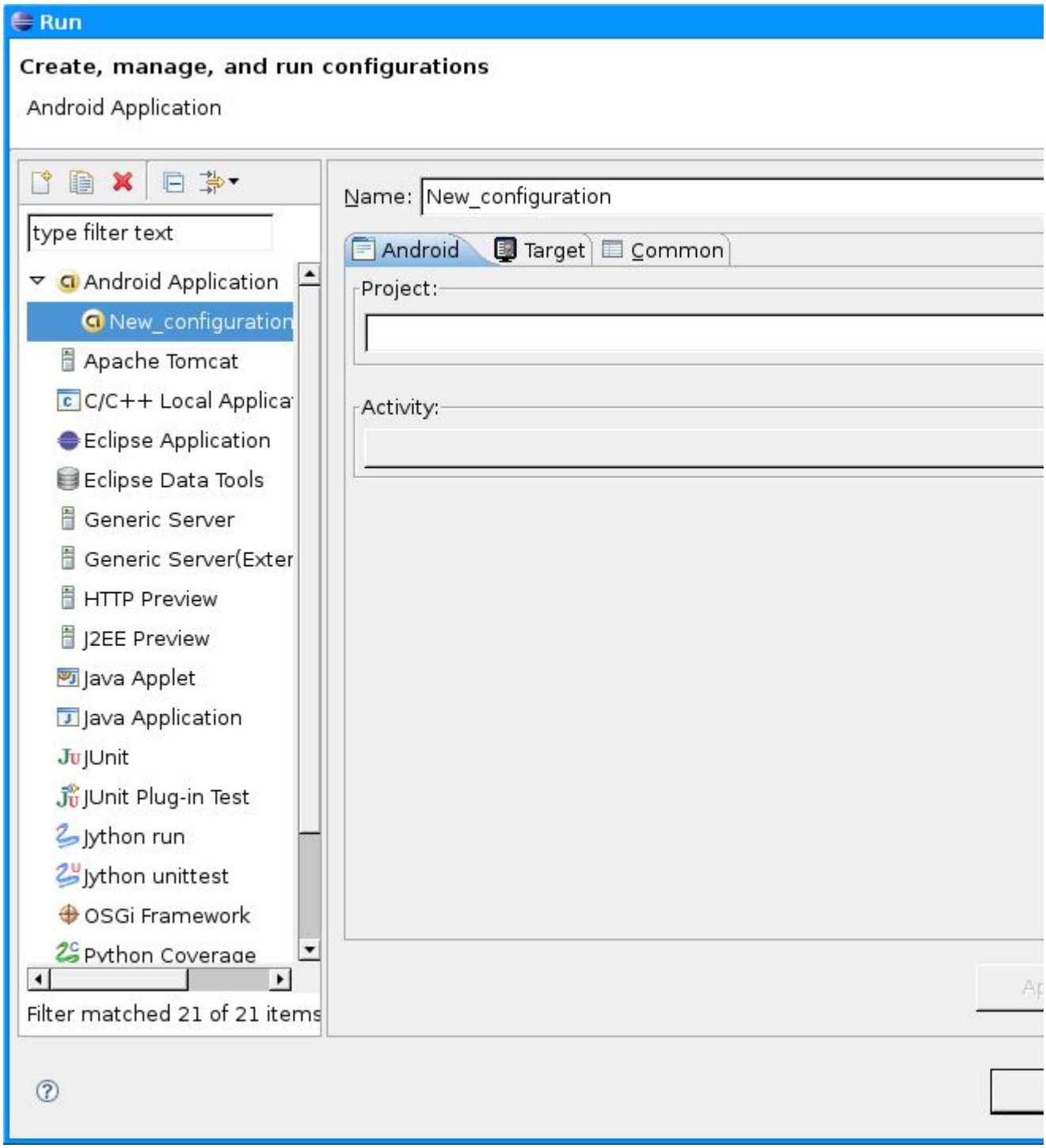
活动里 setContentView（）的方法表明哪个视图需要在当前 UI 上被操作。如果一个活动不能调用这个方法，那么当前就没有界面系统显示为一个空白屏幕。我们仅仅想显示一些文本，所以我们将刚才创建的 TextView 连接上
这就是 Android 平台里的“Hello，World”，当然，我们可以看下运行情况。

执行代码：Hello，Android

Eclipse 的插件使得你的程序很容易运行。选择 **Run>->Open Run Dialog** 菜单。（Eclipse3.4 版本中，菜单为 Run->Run Configurations）可以看到这样的对话框

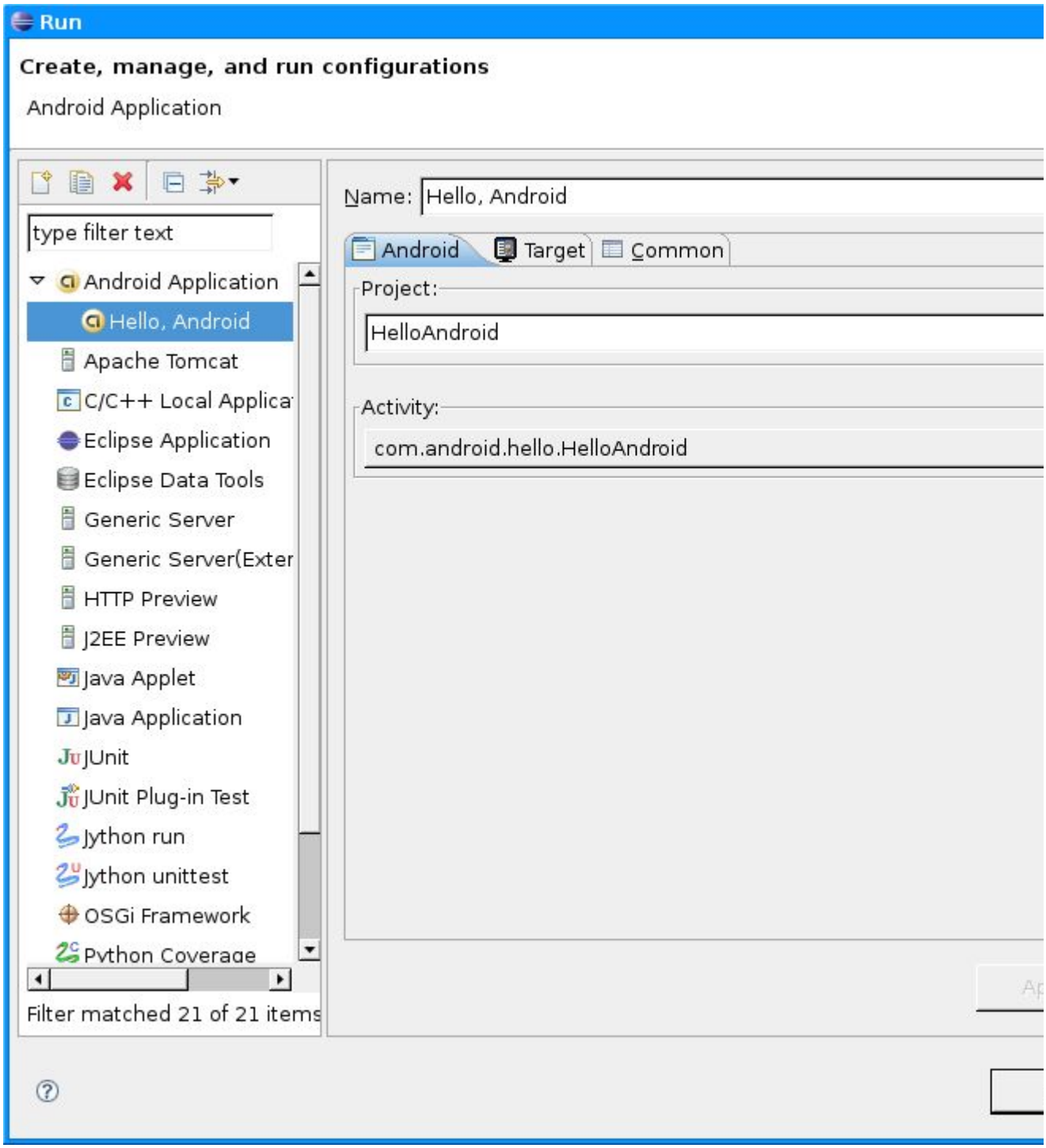


下一步，选择“Android Application”，点击在左上角（按钮像一张纸上有个“+”号）或者双击“Android Application”。 有个新的选项“New_configuration”。



将名字改得更形象一点，如“Hello，Android”，然后按 Browse 按钮选择你的项目，（如果你 Eclipse 里有多个 Android 项目需要打开，确定要选择正确）插件会会自动扫描你项目里的活动子类，然后在“活动”的下拉菜单里加载。如果你的“Hello，Android”项目只有一个，它将被设置为默认项目，然后你可以继续。

点击“Apply”按钮，这里有个例子:



这样就可以了，点击“Run”按钮，Android 的模拟器启动。一启动你的程序就会出现，当一切 OK，你可以看到:

这就是 Android 的“Hello, World”，是不是非常简单呢？下一章节我们将提供更详细的信息。当你接触更多的 Android 时，你会发现它非常有价值。

使用 XML 构建 UI

你刚刚完成的“Hello, World”的例子使用的是我们称为“可编程”的 UI 层，意思是你通过编写代码来组建 UI 层。当你开发了足够多的 UI 程序，你会发现一些很糟糕的现象： 一些小的变化需要你做大量的代码改动。你常常会忘记将 View 连接起来，这样会导致一些错误，浪费你很多时间去调试代码。

这就是 Android 为什么提供一个可变化的 UI 开发模块：基于 XML 的文件。最简单解释这个概念就是演示个例子。这里有个 XML 的文件，它能达到和你刚才完成代码同样的效果:

```
<?xml version="1。0" encoding="utf-8"?>

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

```
  android:layout_width="fill_parent"

  android:layout_height="fill_parent"

  android:text="Hello, Android"/>
```

通常 Android 里 XML 文件结构是很简单的。只是一些标记的树形集合，每个标记就是一个视图类。在这个例子中，它就是一个简单的 TextView 元素的树，你可以在 XML 文件里使用任何扩展类的名字作为你的标记，这也包括你在你的自己的代码里定义的名字。这个结构使得你能使用简单结构和语法快速的组建 UI，这种模型就像网站的开发模型，你能够将 UI 和程序逻辑分开，单独获取或者填充数据。

在这个例子里，有 4 个 XML 属性，下面是属性的大概意思:

属性	描述
`xmlns:android`	这是一个 XML 命名空间，告诉 Android 开发工具你准备使用 Android 命名空间里的一些通用属性。在所有 Android XML 设计文件中最外层的标记必须使用这个树形。
`android:layout_width`	这个属性定义了这个视图需要占用的屏幕宽度。在这个例子中，我们仅有的一个视图可以占用整个屏幕，那就是“fill_parent”的意思。
`android:layout_height`	这个和“layout_width”差不多，表示占用屏幕的高度。
`android:text`	这个设置文本显示内容，在这个例子里，我们使用“Hello，Android”。

这就是 XML 的布局，你需要把这个文件放在什么位置？ 放在你的工程/res/layout 下就可以。“res”是“resource”的简称，这个目录包含了所有应用程序需要的非代码部分。比如图片、字符串、XML 文件。

Eclipse 插件为你创建了这些 XML 文件中的一个。在我们上面的范例，我们根本没有使用过它。在包的管理器里，展开目录/res/layout,编辑 main.xml 文件，替换上面的文本然后保存修改。

在从代码目录里打开 R.java 文件，你可以看到他们像这样:

```
public final class R {
```

```
    public static final class attr {
    };
    public static final class drawable {
        public static final int icon=0x7f020000;
    };
    public static final class layout {
        public static final int main=0x7f030000;
    };
    public static final class string {
        public static final int app_name=0x7f040000;
    };
};
```

R.java 是文件中所有资源的索引界定值定义。你在代码中使用这个类，就像在你的项目里使用一个简洁的方法表示你的资源。在 Eclipse 这样的 IDE 工具里，这个方式对于代码自动完成功能还是非常有效的，因为这能让你快速得定位你要寻找的东西。

有个重要点需要注意的是有个内部类“main”,是“layout”的成员类。Eclipse 插件提醒你加了一个新的 XML 文件，然后生成 R.java 文件，当你加入其他资源到你的工程里，你可以看到 R.java 在同步更改。

最后要做的就是使用你最新版本的XML修改你的HelloAndroid代码去代替以前的编码。这个有例子说明重新写过的类的模型，你可以看到，代码变得非常简单:

```
package com.android.hello;


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


public class HelloAndroid extends Activity {
```

```
    /** Called when the activity is first created */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

当你做这些修改的时，你只需要拷贝、复制。在那个 R 的类里你可以使用代码自动完成功能，你会发现那确实很有帮助。
现在你已经完成了这些修改，继续执行你的程序-你需要做的就是按下绿色的“Run”按钮，或者在菜单中选择 Run->Run History->Hello,Android。 你可以看到… 哇哦，和之前看到的一样！这点表明两种不同的构建方式都能达到同样的结果。
还有很多方法去创建 XML 文件，在你想做之前。 读一下 实现用户接口 可以获得更多信息。

调试工程

Android 对 Eclipse 开发的插件和 Eclipse 的调试工具结合得也很好。 为了演示一下，我们在代码里加入一个 bug。你可以像下面的代码一样修改你的代码:

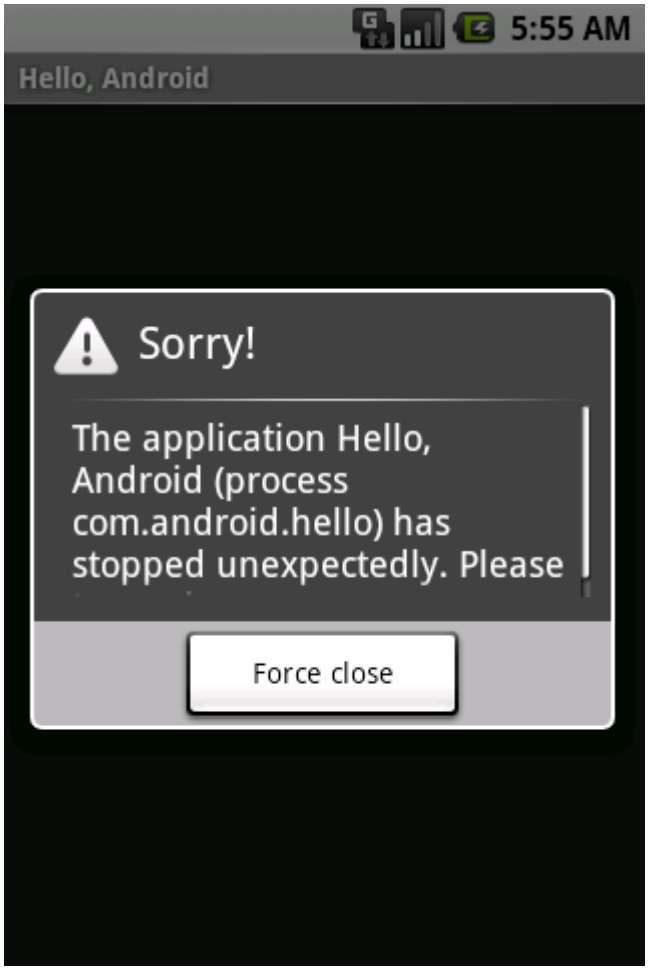
```
package com.android.hello;


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


public class HelloAndroid extends Activity {
    /** 活动第一次被创建后调用 */
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Object o = null;
        o.toString();
        setContentView(R.layout.main);
    }
}
```

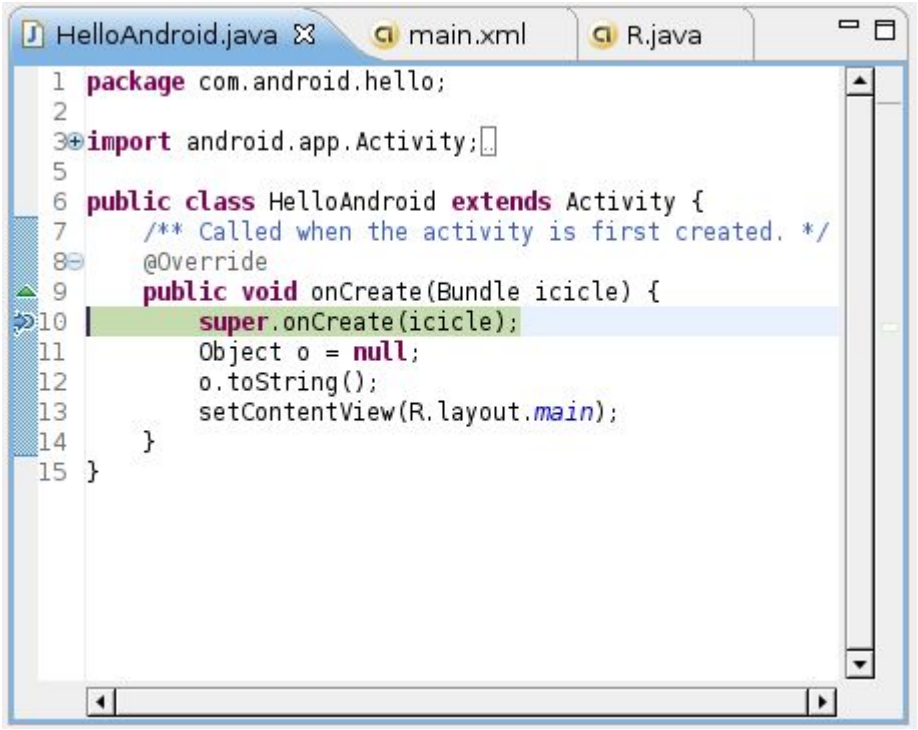
这个修改仅仅引入了一个 NullPointerException 到你的代码里。 如果你再次运行程序，最后你会看到:



按“Force Quit”终止程序并关掉模拟器窗口。

为了寻找错误的原因，设置一个断点在 `Object o = null;` (在代码行旁边的标记栏双击)；然后从菜单选择 Run->Debug History->Hello,Android 进入调试模式。你的应用

程序会在模拟器里重新启动，但这次会在你设的断点处停住。你可以在Eclipse’s Debug Perspective 里单步执行代码，就像你调试其他程序一样。



不使用 **Eclipse** 创建工程

如果你不使用 Eclipse（比如你喜欢其他 IDE 工具，或者仅仅使用文本编辑器或者命令行工具），那么 Eclipse 的插件对你没有作用。别担心-它不会因为你不使用 Eclipse 而使用失去任何功能。

Android 对 Eclipse 的插件仅仅是 Android SDK 外围的一个工具。(这些工具如： 模拟器，aapt, adb, ddms 等等都在 其他文档) 因此，将这些工具和其他工具如“ant”编译文件结合也是有可能的。

Android SDK 里包含一个 Python 的脚本“activitycreator.py”,它可以为你的项目创建所有的代码以及目录。就像 ant 中的“build.xml”文件。这允许你使用命令行编译你的工程或者使用你自己的 IDE 工具集成。

例如，就像我们刚使用 Eclipse 创建的 HelloAndroid 项目，你可以使用命令:

```
activitycreator.py --out HelloAndroid
com.android.hello.HelloAndroid
```

编译工程的时候，你运行“ant”命令，当命令成功执行后，将在“bin”目录里产生一个“HelloAndroid.apk”的文件，“apk”文件是 Android 的一个包，你可以使用“adb”工具安装或者执行。

如果想获得更多信息，请阅读网站提供的替他文档。

Android 应用程序构成

一般情况 Android 应用程序是由以下四种组件构造而成的：

- 活动
- 广播接收器
- 服务
- 内容提供器

需要注意的是，并不是每个 Andorid 应用程序都必须构建这 4 个组件，有些可能由这些组件的组合而成。

一旦你确定了你的应用程序中需要的组件，那么你就应该在 AndroidManifest.xml 中列出他们。 这是一个 XML 配置文件，它用于定义应用程序中需要的组件、组件的功能及必要条件等。这个文件是必须的。 详情参见 Android manifest file documentation

四种组件说明如下：

活动

活动是最基本的 Andorid 应用程序组件，应用程序中，一个活动通常就是一个单独的屏幕。每一个活动都被实现为一个独立的类，并且从活动基类中继承而来，活动类将会显示由视图控件组成的用户接口，并对事件做出响应。 大多数的应用是由多屏幕显示组成。例如，一个文本信息的应用也许有一个显示发送消息的联系人列表屏幕， 第二个屏幕用来写文本消息和选择收件人， 再来一个屏幕查看消息历史或者消息设置操作等。这里每一个这样的屏幕就是一个活动，很容易实现从一个屏幕到一个新的屏幕并且完成新的活动。在某些情况下当前的屏幕也许需要向上一个屏幕动提供返回值--比如让用户从手机中挑选一张照片返回通讯录做为电话拨入者的头像。

当打开一个新的屏幕时，之前一个屏幕会被置为暂停状态并且压入历史堆栈中。用户可以通过回退操回到以前打开过的屏幕。我们可以选择性的移除一些没有必要保留的屏幕，因为 Android 会把每个从桌面打开的程序保留在堆栈中。

Intent 和 Intent Filters

调用 Android 专有类 Intent 进行构屏幕之间的切换。 Intent 是描述应用想要做什么。Intent 数据结构两最重要的部分是动作和动作对应的数据。典型的动作类型有：MAIN（活动的门户）、VIEW、PICK、EDIT 等。而动作对应的数据则以 URI 的形式进行表示。例如：要查看某一个人的联系方式，你需要创建一个动作类型为 VIEW 的 intent，以及一个表示这个人的 URI。

与之有关系的一个类叫 IntentFilter。当 intent 被要求做某事的时候，intent filter 用于描述一个活动（或者 BroadcastReceiver,看下面）能够操作哪些 intent。一个活动如果要

显示一个人的联系方式时，需要声明一个 IntentFilter，这个 IntentFilter 要知道怎么去处理 VIEW 动作和表示一个人的 URI。 IntentFilter 需要在 AndroidManifest.xml 中定义。通过解析各种 intent，从一个屏幕切换到另一个屏幕是很简单的。当向前导航时，活动将会调用 `startActivity(myIntent)`方法。然后，系统会在所有安装的应用程序定义的 IntentFilter 中查找，找到最匹配 `myIntent` 的 Intent 对应的活动。新的活动接收到 myIntent 的通知后，开始运行。当 start 活动方法被调用将触发解析 myIntent 的动作，这个机制提供了两个关键好处：

- 活动能够重复利用从其它组件中以 Intent 的形式产生的一个请求
- 活动可以在任何时候被一个具有相同 IntentFilter 的新的活动取代

广播接收器

你可以使用 BroadcastReceiver 来让你的应用对一个外部的事件做出响应。比如：当电话呼入时，数据网络可用时，或者到了晚上时。BroadcastReceivers 不能显示 UI，它只能通过 NotificationManager 来通知用户这些有趣的事情发生了。BroadcastReceivers 既可以在 AndroidManifest.xml 中注册，也可以在代码中使用 `Context.registerReceiver()`进行注册。但这些有趣的事情发生时，你的应用不必对请求调用 BroadcastReceivers，系统会在需要的时候启动你的应用，并在必要情况下触发 BroadcastReceivers。各种应用还可以通过使用 `Context.sendBroadcast()` 将它们自己的 intent broadcasts 广播给其它应用程序。

服务

一个服务是具有一段较长生命周期且没有用户界面的程序。比较好的一个例子就是一个正在从播放列表中播放歌曲的媒体播放器。在一个媒体播放器的应用中，应该会有多个活动，让使用者可以选择歌曲并播放歌曲。 然而，音乐重放这个功能并没有对应的活动，因为使用者当然会认为在导航到其它屏幕时音乐应该还在播放的。在这个例子中，媒体播放器这个活动会使用 `Context.startService()` 来启动一个服务，从而可以在后台保持音乐的播放。同时，系统也将保持这个服务一直执行，直到这个 service 运行结束。(你可以通过阅读 Life Cycle of an Android Application 获取更多关于服务的介绍). 另外，我们还可以通过使用 `Context.bindService()` 方法，连接到一个服务上（如果这个服务还没有运行将启动它）。当连接到一个服务之后，我们还可以通过服务提供的接口与它进行通讯。拿媒体播放器这个例子来说，我们还可以进行暂停、重播等操作。

内容提供器

应用程序能够将它们的数据保存到文件、SQLite 数据库中，甚至是任何有效的设备中。当你想将你的应用数据与其它的应用共享时，内容提供其将会很有用。一个内容提供器类实现了一组标准的方法，从而能够让其它的应用保存或读取此内容提供器处理的各种数据类型。 更详细的内容提供器资料，可以参考附带文档中的 Accessing Content Providers。

教程：一个记事本应用程序范例

本教程通过手把手教你的方式，讲解如何利用 Android 框架和诸多工具建立自己的手机应用。从一个预先配置好的工程文件开始，该教程通过一个简单记事本应用程序完整的开发过程，并辅以贯穿始终的详尽例子，指导你如何搭建工程、组织应用逻辑以及 UI，乃至接下来的编译及运行可执行程序等等。

该教程将这个记事本应用的开发过程视作一组练习(见如下)，每一个练习都由若干步骤组成。你可以亦步亦趋地完成每个练习步骤，逐步建立并完善自己的应用程序。这些练习提供了你实现此应用所需的——细到每一步骤的——具体范例代码。

当你完成此教程后，一个具有实际功能的 Android 应用就从你手上诞生了，并且你对 Android 应用开发中的一些极为重要的概念也会有更加深刻的理解。若你想为你这个简单的记事本应用添加更多复杂功能的话，你可以用另一方式实现的记事本程序比照你的练习代码，具体可参看 Sample Code 文档部分。

本教程目标读者

该教程主要是面向有一定经验，尤其是那些具备一定 Java 编程语言知识的开发者。如果你之前从未写过一个 Java 应用程序的话，仍可以使用此教程，只是学习进度稍稍慢一点罢了。

本教程假定你已熟悉了一些基本的 Android 应用概念和术语。如果你对这些还不够熟稔的话，你得将 Overview of an Android Application 好好温故一下，才能继续下面的学习。

同时需注意的时，该教程的集成开发环境是预装 Android 插件的 Eclipse。如果你不用 Eclipse，仍可做下面的这些练习和建立应用，但你届时将不得不面对一些涉及 Eclipse 的步骤在非 Eclipse IDE 中如何实现的问题。

练习前的准备

有关此教程中涉及搭建 Android 应用程序工程的相关信息已在 Installing the SDK 和 Hello Android 这两份文档中做了详细说明。这两份文档详细地解释了为建立 Android 应用，搭建自己的开发环境所必需的知识。在你开始学习本教程之前，两份文档、SDK 安装以及开发环境搭建你得确保都已万事俱备，只欠东风了。

为此节训练所需的准备工作： project exercises archive (.zip) 2.解压至你本地的某一目录下 3.打开 NotepadCodeLab 文件夹

1. 下载 project exercises archive (.zip)
2. 解压至你本地的某一目录下
3. 打开 NotepadCodeLab 文件夹

NotepadCodeLab 这一文件夹下共有六个工程文件，具体分别是：Notepadv1, Notepadv2, Notepadv3, Notepadv1Solution, Notepadv2Solution 和 Notepadv3Solution。其中诸如 Notepadv#的工程目录是每个练习的起点，而 Notepadv#Solution 则是对应的答案。如果你在某一练习中遇到问题，你可以通过将你本地工程的代码与对应的答案对比一下来自我解答。

练习

下表列出了本教程的一些练习，并描述每一练习所涵盖的开发训练内容。每一练习都是基于你已完成之前的任一练习这一前提来展开的。

练习 1	一个简单记事本应用程序的开发过程起始于此。构建一个可以添加新便笺，但不能编辑的简单便笺列表。演示 ListActivity 的基本方面以及如何创建和控制菜单项，以及如何用一个 SQLite 数据库存储便笺内容。
练习 2	为记事本应用程序添加第二个 Activity。演示如何构造一个新 Activity 至 Android 操作系统，在不同 Activity 之间传输数据，使用更多高级的屏幕布局，并演示如何通过调用 startActivityForResult()这个 API 来激活另一窗口并返回一个结果。
练习 3	为应用程序添加对 Event 生命周期的控制，以得到应用程序在整个生命周期中的状态。
额外学习	演示如何使用 Eclipse 的调试器及通过调试器查看 Event 产生后的整个生命周期。此节非必读，但仍极力推荐一读。

其它资源及延伸学习

- 对本教程未涉及的一些轻微但更广博的概念请参阅 Common Android Tasks.

- Android SDK 已包含了适合进一步学习，功能完整的绝佳范例。具体范例可以在你下载的 SDK 中的 samples 目录中找到。
- 尽管本教程与 SDK samples 目录下的完整记事本应用程序并非完全一致，但还是脱胎于此。当你学习本教程时，强烈建议你细细揣摩 SDK Sample 目录下的记事本应用程序。其中包含了一些较之个人学习应用程序更为有趣的附加信息，诸如：
  - 建立一个有条纹的自定义便笺列表
  - 创建一个自定义的便笺文本编辑窗口，该窗口覆写 draw() 方法，使其看起来像一个真正的有下划线的便笺记事本
  - ContentProvider()方法的一个完整实现
  - 撤销和放弃编辑操作而非自动保存该编辑操作的结果

开发工具

Android SDK 包括各种各样的定制工具,可以帮助你在 Android 平台上开发移动应用程序。其中最重要的工具是 Android 模拟器和 Eclipse 的 Android 开发工具插件，但 SDK 也包含了各种在模拟器上用于调试，打包和安装的工具

Android 模拟器（Android Emulator ）

它是在你的计算机上运行的一个虚拟移动设备。你可以使用模拟器来在一个实际的 Android 运行环境下设计，调试和测试你的应用程序。

层级观察器 （Hierarchy Viewer New! ）

层级观察器工具允许你调试和优化你的用户界面。它用可视的方法把你的视图（view）的布局层次展现出来，此外还给当前界面提供了一个具有像素栅格(grid)的放大镜观察器，这样你就可以正确地布局了。

Draw 9-patch New!

Draw 9-patch 工具允许你使用所见即所得(WYSIWYG)的编辑器轻松地创建 NinePatch 图形。它也可以预览经过拉伸的图像，高亮显示内容区域。

Eclipse IDE Android 开发工具插件（Android Development Tools Plugin for the Eclipse IDE）

ADT 插件大大扩展了 Eclipse 集成环境功能，使得生成和调试你的 Android 应用程序既容易又迅速。如果你使用 Eclipse，ADT 插件可以让你难以置信地加快开发 Android 应用程序的 速度。

- 你可以从 Eclipse IDE 内部访问其它 Android 开发工具。例如，ADT 可以让你直接从 Eclipse 访问 DDMS 工具的很多功能—屏幕截图，管理端口转发（port-forwarding）,设置断点，观察线程和进程信息。

- 它提供了一个新的项目向导（New Project Wizard），帮助你快速生成和建立起新 Android 应用程序所需的最基本的文件。
- 它使得构建 Android 应用程序的过程变得自动化以及简单易行。
- 它提供了一个 android 代码编辑器，可以帮助你为 Android manifest 和资源文件编写有效的 XML。

有关 ADT 插件的更多详细信息，包括安装指令，可参考 Installing the ADT Plugin for Eclipse。如果你想看一个用法范例的屏幕截图，可参考 Hello Android。

Dalvik 调试监视器服务（Dalvik Debug Monitor Service (ddms)）

这个工具集成了 Dalvik（为 Android 平台定制的虚拟机（VM）），能够让你在模拟器或者设备上管理进程并协助调试。你可以使用它杀死进程，选择某个特定的进程来调试，产生跟踪数据，观察堆（heap）和线程信息，截取模拟器或设备的屏幕画面，还有更多的功能。

Android 调试桥（Android Debug Bridge (adb) ）

Adb 工具可以让你在模拟器或设备上安装应用程序的.apk 文件，并从命令行访问模拟器或设备。你也可以用它把 Android 模拟器或设备上的应用程序代码和一个标准的调试器连接在一起。

Android Asset Packaging Tool (aapt)

Aapt 工具可以让你创建包含 Android 应用程序二进制文件和资源文件的.apk 文件。

Android 接口描述语言（Android Interface Description Language (aidl)）

可以让你生成进程间的接口的代码，诸如 service 可能使用的接口。

sqlite3

这个工具能够让你方便地访问 SQLite 数据文件。这些数据文件是由 Android 应用程序创建并使用的。

Traceview

这个工具可以将你的 Android 应用程序产生的跟踪日志（trace log）转换为图形化的分析视图。

mksdcard

帮助你创建磁盘映像（disk image），你可以在模拟器环境下使用磁盘映像来模拟外部存储卡（例如 SD 卡）。

dx

Dx gongju 将.class 字节码（bytecode）转换为 Android 字节码（保存在.dex 文件中）。

UI/Application Exerciser Monkey

Monkey 是在模拟器上或设备上运行的一个小程序，它能够产生为随机的用户事件流，例如点击(click)，触摸(touch)，挥手（gestures），还有一系列的系统级事件。你可以使用 Monkey 来给你正在开发的程序做随机的，但可重复的压力测试 。

activitycreator

一个可以产生 Ant build 文件的脚本，你可以使用它编译你的 android 应用程序。如果你正在 Eclipse 上开发，并使用 ADT 插件，你不必使用这个脚本。

Android 应用程序模块：应用，任务，进程，和线程

在大多数操作系统里，存在独立的一个 1 对 1 的可执行文件（如 Windows 里的 exe 文件）， 它可以产生进程，并能和界面图标、应用进行用户交互。但在 Android 里，这是不固定的，理解将这些分散的部分如何进行组合是非常重要的。

由于 Android 这种可灵活变通的，在实现一个应用不同部分时你需要理解一些基础技术：

- 一个 **android 包**（简称 **.apk**），里面包含应用程序的代码以及资源。这是一个应用发布，用户能下载并安装他们设备上的文件。
- 一个 **任务** ，通常用户能当它为一个“应用程序”来启动：通常在桌面上会有一个图标可以来启动任务，这是一个上层的应用，可以将你的任务切换到前台来。
- 一个 **进程** 是一个底层的代码运行级别的核心进程。通常.apk 包里所有代码运行在一个进程里，一个进程对于一个.apk 包；然而，进程 标签常用来改变代码运行的位置，可以是 全部的.apk 包 或者是独立的 活动，接收器，服务，或者 提供器组件。

任务

记住关键的一点：当用户看到的“应用”，无论实际是如何处理的，它都是一个任务。如果你仅仅通过一些活动来创建一个.apk 包，其中有一个肯定是上层入口（通过动作的 intent-filter 以及分类 `android.intent.category.LAUNCHER`），然后你的.apk 包就创建了一个单独任务，无论你启动哪个活动都会是这个任务的一部分。

一个任务，从使用者的观点，他是一个应用程序；对开发者来讲，它是贯穿活动着任务的一个或者多个视图，或者一个活动栈。当设置 Intent.FLAG_ACTIVITY_NEW_TASK

标志启动一个活动意图时，任务就被创建了；这个意图被用作任务的根用途，定义区分哪个任务。如果活动启动时没有这个标记将被运行在同一个任务里（除非你的活动以特殊模式被启动，这个后面会讨论）。如果你使用 FLAG_ACTIVITY_NEW_TASK 标记并且这个意图的任务已经启动，任务将被切换到前台而不是重新加载。
FLAG_ACTIVITY_NEW_TASK 必须小心使用：在用户看来，一个新的应用程序由此启动。如果这不是你期望的，你想要创建一个新的任务。另外，如果用户需要从桌面退出到他原来的地方然后使用同样的意图打开一个新的任务，你需要使用新的任务标记。否则，如果用户在你刚启动的任务里按桌面（HOME）键，而不是退出（BACK）键，你的任务以及任务的活动将被放在桌面程序的后面，没有办法再切换过去。

任务亲和力（**Affinities**）

一些情况下 Android 需要知道哪个任务的活动附属于一个特殊的任务，即使该任务还没有被启动。这通过任务亲和力来完成，它为任务中一个或多个可能要运行的活动提供一个独一无二的静态名字。默认为活动命名的任务亲和力的名字，就是实现该活动.apk 包的名字。这提供一种通用的特性，对用户来说，所有在.apk 包里的活动都是单一应用的一部分。
当不带 Intent.FLAG_ACTIVITY_NEW_TASK 标记启动一个新的活动，任务亲和力对新启动的活动将没有影响作用：它将一直运行在它启动的那个任务里。然而，如果使用 NEW_TASK 标记，亲和力会检测已经存在的任务是否具有相同的亲和力。如果是，该任务会被切换到前台，新的活动会在任务的最上面被启动。
你可以在你的表现文件里的应用程序标签里为.apk 包里所有的活动设置你自己的任务亲和力，当然也可以为单独的活动设置标签。这里有些例子演示如何使用：

- 如果你的.apk 包里包含多个用户可启动的上层应用程序，那么你可能想要为每个活动分配不同的亲和力。这里有一个不错的协定，你可以将不同的名字字串加上冒号附加在.apk 包名字的后面 。 例如，"com.android.contacts"的亲和力命名可以是"com.android.contacts:Dialer" and "com.android.contacts:ContactsList"。

- 如果你想替换一个通知，快捷键，或者其它能从外部启动的应用程序的内部活动，你需要在你想替换的活动里明确的设置任务亲和力(taskAffinity)。例如，如果你想替换联系人详细信息浏览界面（用户可以直接操作或者通过快捷方式调用），你需要设置任务亲和力(taskAffinity)为“com.android.contacts”。

启动模式以及启动标记

你控制活动和任务通信的最主要的方法是通过设置启动模式的属性以及意图相应的标记。这两个参数能以不同的组合来共同控制活动的启动结果，这在相应的文档里有描述。这里我们只描述一些通用的用法以及几种不同的组合方式。

你最通常使用的模式是 `singleTop`（除了默认为 `standard` 模式）。这不会对任务产生什么影响；仅仅是防止在栈顶多次启动同一个活动。

`singleTask` 模式对任务有一些影响：它能使得活动总是在新的任务里被打开（或者将已经打开的任务切换到前台来）。使用这个模式需要加倍小心该进程是如何和系统其他部分交互的，它可能影响所有的活动。这个模式最好被用于应用程序入口活动的标记中。（支持 MAIN 活动和 LAUNCHER 分类）。

`singleInstance` 启动模式更加特殊，该模式只能当整个应用只有一个活动时使用。

有一种情况你会经常遇到，其它实体（如搜索管理器 SearchManager 或者 通知管理器 NotificationManager）会启动你的活动。这种情况下，你需要使用 Intent.FLAG_ACTIVITY_NEW_TASK 标记，因为活动在任务（这个应用/任务还没有被启动）之外被启动。就像之前描述的一样， 这种情况下标准特性就是当前和任务和新的活动的亲和性匹配的任务将会切换到前台，然后在最顶端启动一个新的活动。当然，你也可以实现其它类型的特性。

一个常用的做法就是将 Intent.FLAG_ACTIVITY_CLEAR_TOP 和 NEW_TASK 一起使用。这样做，如果你的任务已经处于运行中，任务将会被切换到前台来，在栈里的所有的活动除了根活动，都将被清空，根活动的 onNewIntent(Intent) 方法传入意图参数后被调用。当使用这种方法的时候 `singleTop` 或者 `singleTask` 启动模式经常被使用，这样当前实例会被置入一个新的意图，而不是销毁原先的任务然后启动一个新的实例。

另外你可以使用的一个方法是设置活动的任务亲和力为空字串（表示没有亲和力），然后设置 finishOnBackground 属性。 如果你想让用户给你提供一个单独的活动描述的通知，倒不如返回到应用的任务里，这个比较管用。要指定这个属性，不管用户使用 BACK 还是 HOME，活动都会结束；如果这个属性没有指定，按 HOME 键将会导致活动以及任务还留在系统里，并且没有办法返回到该任务里。

请确保阅读过文档启动模式属性（launchMode attribute） 以及 意图标记（Intent flags） ，关注这些选项的详细信息。

进程

在 Android 中，进程是应用程序的完整实现，而不是用户通常了解的那样。他们主要用途很简单：

- 提高稳定性和安全性，将不信任或者不稳定的代码移动到其他进程。
- 可将多个.apk 包运行在同一个进程里减少系统开销。
- 帮助系统管理资源，将重要的代码放在一个单独的进程里，这样就可以单独销毁应用程序的其他部分。

像前面描述的一样，进程的属性被用来控制那些有特殊应用组件运行的进程。注意这个属性不能违反系统安全： 如果两个.apk 包不能共享同一个用户 ID，却试图运行在通一个进程里，这种情况是不被允许的，事实上系统将会创建两个不同的进程。

请查看安全相关文档以获取更多关于安全限制方面的信息。

线程

每个进程包含一个或多个线程。多数情况下，Android 避免在进程里创建多余的线程，除非它创建它自己的线程，我们应保持应用程序的单线程性。 一个重要的结论就是所有呼叫实例, 广播接收器, 以及 服务的实例都是由这个进程里运行的主线程创建的。
注意新的线程**不是**为活动，广播接收器，服务或者内容提供器实例创建：这些应用程序的组件在进程里被实例化（除非另有说明，都在同一个进程处理），实际上是进程的主线程。这说明当系统调用时这些组件（包括服务）不需要进程远距离或者封锁操作（就像网络呼叫或者计算循环），因为这将阻止进程中的所有其他组件。你可以使用标准的线程 类或者 Android 的 HandlerThread 类去对其它线程执行远程操作。
这里有一些关于创建线程规则的例外：

- 呼叫 IBinder 或者 IBinder 实现的接口，如果该呼叫来自其他进程，你可以通过线程发送的 IBinder 或者本地进程中的线程池呼叫它们，从进程的主线程呼叫是不可以的。特殊情况下，,呼叫一个服务 的 IBinder 可以这样处理。（虽然在服务里呼叫方法在主线程里已经完成。）这意味着 IBinder 接口的实现必须要有一种线程安全的方法，这样任意线程才能同时访问它。
- 呼叫由正在被调用的线程或者主线程以及 IBinder 派发的内容提供器 的主方法。被指定的方法在内容提供器的类里有记录。这意味着实现这些方法必须要有一种线程安全的模式，这样任意其它线程同时可以访问它。
- 呼叫视图以及由视图里正在运行的线程组成的子类。通常情况下，这会被作为进程的主线程，如果你创建一个线程并显示一个窗口，那么继承的窗口视图将从那个线程里启动。

Android 应用程序的生命周期

在大多数情况下，每个 Android 应用程序都运行在自己的 Linux 进程中。当应用程序的某些代码需要运行时，这个进程就被创建并一直运行下去，直到系统认为该进程不再有用为止。然后系统将回收进程占用的内存以便分配给其它的应用程序。
应用程序的开发人员必须理解不同的应用程序组件（尤其是 Activity, Service, 和 BroadcastReceiver）是如何影响应用程序进程生命周期的，这是很重要的一件事情。**不正确地使用这些组件可能会导致系统杀死正在执行重要任务的应用程序进程。**
一个常见的进程生命周期 bug 的例子是 BroadcastReceiver， 当 BroadcastReceiver 在 BroadcastReceiver.onReceive()方法中接收到一个 Intent 时，它会启动一个线程，然后返回。一旦它返回，系统将认为 BroadcastReceiver 不再处于活动状态，因而 BroadcastReceiver 所在的进程也就不再有用了（除非该进程中还有其它的组件处于活动状态）。因此，系统可能会在任意时刻杀死进程以回收内存。这样做的话，进程中创建（spawned）出的那个线程也将被终止。对这个问题的解决方法是从 BroadcastReceiver 启动一个服务，让系统知道进程中还有处于活动状态的工作。为了

决定在内存不足时让系统杀死哪个进程，Android 根据每个进程中运行的组件以及组件的状态把进程放入一个”重要性分级(importance hierarchy)”中。进程的类型包括（按重要程度排序）：

1. 前台（**foreground**）进程，与用户当前正在做的事情密切相关。不同的应用程序组件能够通过不同的方法使它的宿主进程移到前台。当下面任何一个条件满足时，可以考虑将进程移到前台：
    1. 进程正在屏幕的最前端运行一个与用户交互的 Activity（它的 onResume() 方法被调用）
    2. 进程有一正在运行的 BroadcastReceiver （它的 BroadcastReceiver.onReceive()方法正在执行）
    3. 进程有一个 Service，并且在 Service 的某个回调函数（Service.onCreate(), Service.onStart(), 或 Service.onDestroy()）内有正在执行的代码。
1. 可见（**visible**）进程，它有一个可以被用户从屏幕上看到的 Activity，但不在前台（它的 onPause()方法被调用）。举例来说，如果前台的 Activity 是一个对话框，以前的 Activity 隐藏在对话框之后，就可能出现这种进程。这样的进程特别重要，一般不允许被杀死，除非为了保证前台进程的运行不得不这样做。
2. 服务（**service**）进程，有一个已经用 startService() 方法启动的 Service。虽然这些进程用户无法直接看到，但它们做的事情却是用户所关心的（例如后台 MP3 回放或后台网络数据的上传下载）。因此，系统将一直运行这些进程除非内存不足以维持所有的前台进程和可见进程。
3. 后台（**background**）进程， 拥有一个当前用户看不到的 Activity（它的 onStop() 方法被调用）。这些进程对用户体验没有直接的影响。如果它们正确执行了 Activity 生命期（详细信息可参考 Activity），系统可以在任意时刻杀死进程来回收内存，并提供给前面三种类型的进程使用。系统中通常有很多个这样的进程在运行，因此要将这些进程保存在 LRU 列表中，以确保当内存不足时用户最近看到的进程最后一个被杀掉。
4. 空（**empty**）进程，不包含任何处于活动状态的应用程序组件。保留这种进程的唯一原因是，当下次应用程序的某个组件需要运行时，不需要重新创建进程，这样可以提高启动速度。

系统将以进程中当前处于活动状态组件的重要程度为基础对进程进行分类。请参考 Activity, Service 和 BroadcastReceiver 文档来获得有关这些组件在进程整个生命期中是如何起作用的详细信息。每个进程类别的文档详细描述了它们是怎样影响应用程序整个生命周期的。进程的优先级可能也会根据该进程与其它进程的依赖关系而增长。例如，如果进程 A 通过在进程 B 中设置 Context.BIND_AUTO_CREATE 标记或使用 ContentProvider 被绑定到一个服务（Service），那么进程 B 在分类时至少要被看成与进程 A 同等重要。

二、开发应用程序

Implementing a User Interface

This section describes the basics of how to implement the user interface of an Android screen. It covers the basic elements that make up a screen, how to define a screen in XML and load it in your code, and various other tasks you'll need to handle for your user interface.

Topics

- Hierarchy of Screen Elements
- Common Layout Objects
- Working with AdapterViews (Binding to Data)
- Designing Your Screen in XML
- Hooking into a Screen Element
- Listening for UI Notifications
- Applying a Theme to Your Application
- UI Elements and Concepts Glossary

Android 应用构成

Android 应用是由各种各样的组件来构成。 这些组件大部分都是松散连接的，准确 的说你可以把它们看成组件的联合而非是一个单一的应用。

通常，这些组件运行在同一个系统进程里面。你也可以在这个进程里面创建多个线程（这是很常见的），如果必要你也可以创建独立的子进程。不过这种情况是非常少见的，因为 Android 尽力使代码进程间透明。

以下部分是很重要的 Android APIs：

AndroidManifest.xml

AndroidManifest.xml 是系统的控制文件，它告诉系统如何处理你所创建的所有顶层组件(尤其是 activities,服务,Intent 接收器和后面描述的内容管理器)。举例来说，控制文件就是把你的活动（Activities）要接收的 Intents 连接在一起的“胶水”。

活动（Activities）

活动（Activity）就是一个有生命周期的对象。 一个 Activity 就是完成某些工作的代码块， 如必要的话，这部分工作还可能包括对用户 UI 界面的显示。不过这不是必须的，有些活 动从不显示 UI 界面。典型地，你将会指定你的应用程序中的一个活动为整个程序的入口点。

视图（Views）

视图(Views)可以将其自身绘制到屏幕上。Android 的用户界面由一系列的视图树 （trees of views）构成。接口都是由一组以树的形式出现的视图组成的。开发者可 以通过创建一个新的视图的方法来使用自定义的图形处理技术(比如开发游戏，或者是 使用了不常用的用户图形(UI)窗口界面(widget))。

Intents

Intents 是一个简单的消息对象，它表示程序想做某事的“意图”（intention）。比如如果你的应用程序想要显示一个网页,那么它通过创建一个 Intent 实例并将其传递给 系统来表示意图浏览这个 URI。系统将定位于知道如何能处理这一 Intent 的代码（在当 前情况下就是浏览器），并运行之。Intents 也可以用于广播系统范围内的有效事件 (例如通知事件)。

服务（Services）

服务是运行在后台的一段代码。它可以运行在它自己的进程,也可以运行在其他应用程 序进程的上下文（context）里面，这取决于自身的需要.。其它的组件可以绑定到一个服 务（Service）上面，通过远程过程调用（RPC）来调用这个方法。例如媒体播放器的服务， 当用户退出媒体选择用户界面，她仍然希望音乐依然可以继续播放，这时就是由服务 （service）来保证当用户界面关闭时音乐继续播放的。

通知（Notifications）

通知将以小图标的形式呈现在状态栏里，用户通过与图标的交互式操来接收消息。最常见 的通知包括短信息，通话记录，语音邮件，但是应用程序也可以创建它们自己的通知事件。 我们推荐采用通知事件实现提醒用户的注意。

内容管理器（ContentProviders）

内容管理器（ContentProvider）提供对设备上数据进行访问的数据仓库。典型的例子就 是使用内容管理器来访问联系人列表。你的应用程序也可以使用其它程序通过内容管理器提 供的数据，同时你也可以定义你自己的内容管理器来向其它应用提供数据访问服务。

存、取、提供数据

典型的桌面操作系统一般能提供一种通用的文件系统，所有应用程序都能储存和读文件，并且其他应用程序也能访问该文件（可能需要一些访问控制设置）. Android 使用不同的方式：在平台上，所有应用程序的数据（包括文件），对该应用程序是私有的。当然，Android 也提供一种标准方法将自己的私有数据提供给其他应用程序访问。这一章节讲了很多方法，描述应用如何存取数据，以及将数据提供给其他程序访问，当然，你也可以向其他应用程序请求并获得它的数据。

Android 提供下面的方式来存取数据：

参数选择

使用一个轻量级机制来存取基本数据类型的数据对，这是典型应用程序参数的存储模式。

文件

你可以将你的文件存储在设备上或者其他移动媒介上，默认情况下，其他应用程序是不能访问这些文件的。

数据库

Android 有直接 SQLite 数据库的 API。应用程序可以创建以及使用 SQLite 数据库。 每个包创建的数据库都是私有的。

数据提供

数据提供是应用程序的一个可选组件，它可以提供读/写应用程序私有数据的方法。内容提供组件实现了一种标准请求数据的语法，和一种标准处理返回值的机制。Android 提供很多标准数据的提供方式，例如私有联系人。

网络

不要忘记，我们还可以使用网络存取数据。

Android 的安全与权限

Android 是一个多进程系统，每一个应用程序（和系统的组成部分）都运行在自己的进程中。在应用程序和系统间的安全通过标准的 Linux 设备在进程级被执行，例如被分配给应用程序的用户和组 ID。额外的细粒度安全特性通过“许可”机制来提供，该机制能够对一个指定进程可实现的特定操作进行约束。

内容

安全结构

应用程序签名

用户标识和文件访问

权限命名

权限的声明和支持

在 AndroidManifest.xml 文件中支持权限

发送广播时支持权限

其它权限的支持

URI 权限

安全结构

Android 安全学中的一个重要的设计点是在默认情况下应用程序没有权限执行对其它应用程序、操作系统或用户有害的操作。这些操作包括读/写用户的隐私数据（例如联系方式或 e-mail），读/写其它应用程序的文件，执行网络访问，保持设备活动，等等。
应用程序的进程是一个安全的沙箱。它不能干扰其它应用程序，除非在它需要添加原有沙箱不能提供的功能时明确声明*权限*。这些权限请求能够被不同方式的操作所处理，特别的要基于证书和用户的提示被自动的允许或禁止。权限的请求在那个应用程序中通过一个应用程序被声明为静态的，所以在此之后在安装时或没有改变时它们会预先知道。

应用程序签名

所有的 Android 应用程序（.apk 文件）必须通过一个证书的签名，此证书的私钥必须被开发者所掌握。这个证书的标识是应用程序的作者。这个证书*不*需要通过证书组织的签署：Android 应用程序对于使用自签署的证书是完全允许的和特别的。这个证书仅仅被用于与应用程序建立信任关系，不是为了大规模的控制应用程序可否被安装。最重要的方面是通过确定能够访问原始签名权限和能够共享用户 ID 的签名来影响安全。

用户标识和文件访问

安装在设备中的每一个Android包文件(.apk)都会被分配给一个属于自己的统一的Linux用户 ID，并且为它创建一个沙箱以防止影响其它应用程序（或者其它应用程序影响它）。用户 ID 在应用程序安装到设备中时被分配，并且在这个设备中保持它的永久性。
因为安全执行发生在进程级，所以一些不同包中的代码在相同进程中不能正常的运行，自从他们需要以不同 Linux 用户身份运行时。你可以使用每一个包中的

`AndroidManifest.xml` 文件中的 manifest 标签属性 sharedUserId 拥有它们分配的

相同用户 ID。通过这样做，两个包被视为相同的应用程序的安全问题被解决了，注意为了保持安全，仅有相同签名（和请求相同 sharedUserId 标签）的两个应用程序签名将会给相同的用户 ID。
应用创建的任何文件都会被赋予应用的用户标识，并且，正常情况下不能被其它包访问。当你通过 getSharedPreferences(String, int), openFileOutput(String, int) 或者 openOrCreateDatabase(String, int, SQLiteDatabase.CursorFactory)创建一个新文件时，你可以同时或分别使用 MODE_WORLD_READABLE 和 MODE_WORLD_WRITEABLE 标志允许其它包读/写此文件。当设置了这些标志时，这个文件仍然属于你的应用程序，但是它的全局读、写和读写权限已经设置所以其它任何应用程序可以看到它。

权限命名

一个基本的 Android 应用程序没有与其相关联的权限，意味着它不能做任何影响用户体验或设备中的数据的有害操作。要利用这个设备的保护特性，在你的应用程序需要时，你必须在 `AndroidManifest.xml` 文件中包含一个或更多的`<uses-permission>` 标签来声明此权限。
例如：需要监听来自 SMS 消息的应用程序将要指定如下内容：

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

    package="com.android.app.myapp" >



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



</manifest>
```

在安装应用程序时，通过包安装器应用程序要通过权限请求的许可，使建立在与应用程序签名的核对下声明对于用户的那些权限和影响。在应用运行期间对用户不做检查：它要么在安装时被授予特定的许可，并且使用想用的特性；要么不被授予许可，并且使得一切使用特性的尝试失败而不提示用户。
例如，sendBroadcast(Intent) 方法就是当数据被发送给到每个接收器时检查许可的，在方法调用返回之后，因此当许可失败时你不会收到一个异常。然而，几乎在所有例子中，许可失败都会被打印到系统日志中。通常，多次的许可错误会产生抛回至应用程序的 SecurityException 异常。
Android 系统提供的许可可以在 Manifest.permission 中找到。每个引用也可以定义和 Enforce 它自己的许可，因此这不是全面的所有可能的列表。
在程序操作期间，个别权限在一些地方可能被强制：

- 在系统接到呼叫的时候，预防一个应用程序去执行特定的函数。
- 在启动 Activity 时，防止一个应用启动其它应用程序的 Activities。
- 发送和接收 Intent 广播时，控制谁能接收你的广播或者谁能发送广播给你。
- 在一个内容提供器上访问和操作时。
- 绑定或开始一个服务时。

权限的声明和支持

为了执行你自己的权限，你必须首先在你的 AndroidManifest.xml 中使用一个或多个<permission> 标签声明它们。

例如，一个应用程序想用控制谁能启动一个 activities，它可以为声明一个做这个操作的许可，如下:

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

    package="com.me.app.myapp" >


    <permission
android:name="com.me.app.myapp.permission.DEADLY_ACTIVITY"

        android:label="@string/permlab_deadlyActivity"

        android:description="@string/permdesc_deadlyActivity"


android:permissionGroup="android.permission-group.COST_MONEY"

        android:protectionLevel="dangerous" />


</manifest>
```

<protectionLevel> 属性是必需的，告诉系统用户应如何处理应用程序接到请求此权限的通知，或者在这个文档中对这个权限的许可的描述。

<permissionGroup> 属性是可选的，仅仅用于帮助系统为用户显示权限。通常，你要设置这些，向一个标准的系统组（列在 android.Manifest.permission_group 中），或者在更多的情况下要自定义。它更偏向于使用一个已经存在的组，做为简化的权限用户界面显示给用户。

注意：应该为每个权限提供标签(label) 和描述(description)。当用户浏览权限列表时，它们可以为用户展示字符资源，如(android:label) 或者一个许可的详细信息(android:description) 。标签(label)比较短，用几个词来描述该权限保护的关键功能。描述(description)应该是一组句子，用于描述获得权限的用户可以做什么。我们

写描述的习惯是两句话，第一句声明权限，第二句警告用户如果应用许可该权限时，会发生什么不好的事情。
下面是一个 CALL_PHONE 权限的标签和描述的例子：

```
   <string name="permlab_callPhone">directly call phone
numbers</string>

   <string name="permdesc_callPhone">Allows the application to
call

      phone numbers without your intervention. Malicious
applications may

      cause unexpected calls on your phone bill. Note that this
does not

      allow the application to call emergency numbers.</string>
```

你可以在系统中通过 shell 命令 `adb shell pm list permissions` 查看权限当前定义。特别，'-s' 操作以简单粗略的方式为使用者显示权限：

```
$ adb shell pm list permissions -s

All Permissions:



Network communication: view Wi-Fi state, create Bluetooth
connections, full

Internet access, view network state



Your location: access extra location provider commands, fine (GPS)
location,

mock location sources for testing, coarse (network-based) location
```

```
Services that cost you money: send SMS messages, directly call phone
numbers


...
```

在 AndroidManifest.xml 文件中支持权限

通过 `AndroidManifest.xml` 文件可以设置高级权限，以限制访问系统的所有组件或者使用应用程序。所有的这些请求都包含在你所需要的组件中的 android:permission 属性，命名这个权限可以控制访问此组件。

Activity 权限 (使用 <activity> 标签) 限制能够启动与 **Activity** 权限相关联的组件或应用程序。此权限在 Context.startActivity() 和 Activity.startActivityForResult() 期间要经过检查；如果调用者没有请求权限，那么会为调用抛出一个安全异常 (SecurityException)。

Service 权限(应用 <service> 标签)限制启动、绑定或启动和绑定关联服务的组件或应用程序。此权限在 Context.startService(), Context.stopService() 和 Context.bindService() 期间要经过检查；如果调用者没有请求权限，那么会为调用抛出一个安全异常(SecurityException)。

BroadcastReceiver 权限(应用 <receiver> 标签)限制能够为相关联的接收者发送广播的组件或应用程序。在 Context.sendBroadcast() 返回后此权限将被检查，同时系统设法将广播递送至相关接收者。因此，权限失败将会导致抛回给调用者一个异常；它将不能递送到目的地。在相同方式下，可以使 Context.registerReceiver() 支持一个权限，使其控制能够递送广播至已登记节目接收者的组件或应用程序。其它的，当调用 Context.sendBroadcast() 以限制能够被允许接收广播的广播接收者对象一个权限（见下文）。

ContentProvider 权限(使用 <provider> 标签)用于限制能够访问 ContentProvider 中的数据的组件或应用程序。(Content providers 有一个重要的附加安全设施可用于它们调用被描述后的 URI 权限。) 不同于其它组件，它有两个不相连系的权限属性要设置：android:readPermission 用于限制能够读取提供器的组件或应用程序，android:writePermission 用于限制能够写入提供器的组件或应用程序。注意，如果一个提供者的读写权限受保护，意思是你只能从提供器中读，而没有写权限。当你首次收回提供者（如果你没有任何权限，将会抛出一个 SecurityException 异常），那么权限要被检查，并且做为你在这个提供者上的执行操作。 使用 ContentResolver.query() 请求获取读权限; 使用 ContentResolver.insert(), ContentResolver.update() 和 ContentResolver.delete() 请求获取写权限。在所有这些情况下，一个 SecurityException 异常从一个调用者那里抛出时不会存储请求权限结果。

发送广播时支持权限

当发送一个广播时你能总指定一个请求权限，此权限除了权限执行外，其它能发送 Intent 到一个已注册的 BroadcastReceiver 的权限均可以。 通过调用 Context.sendBroadcast() 及一些权限字符串，为了接收你的广播，你请求一个接收器应用程序必须持有那个权限。
注意，接收者和广播者都能够请求一个权限。当这样的事发生了，对于 Intent 来说，这两个权限检查都必须通过，为了交付到共同的目的地。

其它权限支持

任意一个好的粒度权限都能够在一些调用者的一个服务中被执行。 和 Context.checkCallingPermission() method. 方法一起被完成。调用并产生一个需要的权限字符串，它将返回一个整型，以确定当前调用进程是否被许可。注意，仅仅当你执行的调用进入到其它进程的时候这些才会被使用，通常，通过 IDL 接口从一个服务发布，或从一些其它方式通知其它进程。
这有许多其它有益的方式去检查权限。如果你有一个其它进程的 PID，你可以使用上下文的方法 Context.checkPermission(String, int, int) 检查权限违反 PID。 如果你有一个其它应用程序的包名，你可以使用直接的包管理器方法 PackageManager.checkPermission(String, String) 去查看特定的包是否被指定的权限所许可。

URI 权限

迄今为止，在与内容提供器共同使用时，标准权限系统描述通常是不充分的。一个内容提供器要保护它自己及读和写权限，当为了它们产生作用，它的直接客户端总是需要手动的对其它应用程序指定 URI。一个典型的例子是邮件应用程序中的附件。访问邮件的权限应该被保护，因为这是敏感用户数据。可以，如果一个网址图片附件提供给一个图片查看器，那个图片查看器将没有权限打开这个附件，因为它没有原因去拥有一个权限从而不能访问所有的电子邮件。
对于这个问题的解决是通过网址权限：当开始一个活动或对一个活动返回一个结果，调用者可通过设置 Intent.FLAG_GRANT_READ_URI_PERMISSION 和 Intent.FLAG_GRANT_WRITE_URI_PERMISSION 中的一个或者两个。允许接收活动权限访问在 Intent 中特定的数据地址，不论它有权限访问数据在内容提供器相应的 Intent 中。
这种机制允许一个公用的功能性模型使用户相互交互（打开一个附件，从一个列表中选择一个联系人，等等）驱动 ad-hoc 在优粒度权限的许可下。这可能是一个主要设备，应用程序为了减少这个权限而需要，仅仅直接关系到它们的行为。

这优粒度 URI 权限的许可工作，然而，请求一些协作和内容提供者保持那些 URI。强烈推荐内容提供者实现这种设备，并且通过 android:grantUriPermissions 属性或者 <grant-uri-permissions> 标签声明支持它。
更多信息可以在 Context.grantUriPermission(), Context.revokeUriPermission(), 和 Context.checkUriPermission() 方法中找到。

资源管理和多国版本

资源是外部文件（不含代码的文件），它被代码使用并在编译时编入应用程序。Android 支持不同类型的资源文件，包括 XML，PNG 以及 JPEG 文件 XML 文件根据描述的不同有不同格式。这份文档描述可以支持什么样的文件，语法，以及各种格式.
源代码以及 XML 文件将资源打包并编译进二进制文件，这种模式能使得资源更快得被加载。字符串也同样被压缩成更高效的模式。由于这些原因， Android 平台上存在不同的资源类型.
这篇文档包含下列章节:

- 资源
  - 创建资源
  - 使用资源
    - 在代码中使用资源
    - 资源引用
    - 主题属性相关
    - 使用系统资源
  - 可变资源
  - 资源相关
  - 术语
- 国际化(I18N)

和资源引用一起，这是一份完全技术性的文档，它包含了很多资源相关的信息。如果仅仅是使用 Android 平台不需要完全了解这份文档，当你需要时，再去了解它。

资源

这个主题包含了一个资源相关的术语列表，以及如何使用资源的例子。如果想看一个完整的资源类型查询，请查看 资源.
Android 资源系统能跟踪所有非代码相关的应用程序。你可以使用 资源 类来访问应用程序的资源，资源的实例通常和应用程序联系在一起， 你可以通过 Context.getResources()来访问。
应用程序的资源在编译时就被编译到应用程序二进制代码里。为了使用某个资源，你需要将它在代码目录结构里放正确，然后编译。作为编译过程的一 部分，产生的资源代

号你可以在源代码里使用 -- 这允许编译器验证你的程序代码和你定义的资源是否相符。
下面的章节教你如何在应用程序代码里使用资源类.

创建资源

Android 支持字符串，图片以及很多其他类型的资源。每个对象语法、格式以及它们存储位置的支持，都是取决于不同类型的对象? 通常，你可以通过三种类型的文件来创建资源：XML 文件（除位图以及原数据文件），位图文件（对于图片）以及原始数据（其它类型，例如声音文件，等等。）。事实上，有两种不同类型的 XML 文件，一种是编译到包里的，另外一种是通过 aapt 来产生的资源文件， 这里有一张包含所有资源类型，文件格式，文件描述以及所有 XML 文件的详细信息的列表。

在项目里，你可以在子目录 `res/`下创建和存储资源文件。Android 有一个资源编译工具（aapt），它可以编译在这个 目录下所有的子目录中的资源，这里有个各种资源的列表。你可以从 资源引用 这里看到各种类型的对象，包含其语法以及格式。

路径	资源类型
`res/anim/`	XML 文件被编译进 逐帧动画 或 补间动画 的对象
`res/drawable/`	.png, .9.png, .jpg files 这些类型的文件被编译进下列这些图表资源列表 为了获得这些资源的类型，使用 `Resource.getDrawable(id)` • 位图文件 • 9-patches (可改变尺寸的图像)
`res/layout/`	可编译成屏幕布局的 XML 文件 (或者屏幕的一部分). 查看 布局
`res/values/`	可编译成多种类型资源的文件 **注意:** 不像其他 res/ 文件夹,它能容纳任何数量的文件，但只是描述其创建而不是资源本身. XML 的元素类型可以决定这些资源在 R.class 里什么位置被替换 . 文件可以被命名为任何名字，文件夹里有一些典型的文件(一般约定文件以定义的元素类型后面部分为文件名）：：

	• **arrays.xml** 定义数组 • **colors.xml** 定义 颜色 和 颜色字串数值. 你可以使用 `Resources.getDrawable()` 以及 `Resources.getColor(), respectively,` 取得这些资源. • **dimens.xml** 定义 尺寸数据 . 使用 `Resources.getDimension()` 取得这些资源。 • **strings.xml** 定义字符串 数值 (使用 `Resources.getString` 或 `Resources.getText()` 取得资源，（后者更好一点）`getText()` 能取到在用户界面上显示的文本框里的文本。 • **styles.xml** 定义类型 对象。
`res/xml/`	任何 XML 文件可以进行编译，并能在运行时调用 Resources.getXML() 显示 XML 原文件。
`res/raw/`	这里的任何文件都将直接被复制到设备上。编译产品时，这些数据不会被编译，它们被直接加入到程序包里。 为了在程序中使用这些资源，你可以调用 Resources.openRawResource() , 参数为 ID： R.raw.*somefilename*.

资源最终会被编译成 APK 文件，Android 创建一个包装类，命名为 R，这样你能做你的代码里使用这些资源类。根据资源路径和文件名的不同，R 包含很多子类。

全局资源

- 一些资源类允许你定义颜色。它能接受多种网络类型的值 -- 你可以写成 #RGB, #ARGB, #RRGGBB, #AARRGGBB 这样 16 进制常数都可以。
- 所有的颜色都可以设置一个阿尔法值，开始的两个 16 进制数指定为透明。 0 在阿尔法值里意味着透明。当然，默认值是不透明的。

使用资源

这个章节描述如果使用你创建的资源，它包含以下内容：

- 在代码里使用资源类 - 如何在代码中调用资源并实例化。

- 从其他资源类里引用 - 可以从其他资源类里引用. 这样可以复用通用资源。
- 不同设置使用可变资源 - 可以按照机器要求的不同语言以及设置装载不同的资源。

编译时，Android 产生一个叫 R 的类，它指向你程序中所有的资源。这个类包含很多子类。每一种都是 Android 支持的，同时，编译后会产生一个资源文件。每个类提供一个或多个编译后资源的标识符，你可以在代码中使用。下面是个源代码的文件，里面包含了字符串，布局文件（全屏或者部分屏幕），以及图像资源。

注意： 这个 R 类是自动产生的，你不能手动编写。当资源变化的时候它会自动更新。

```
package com.android.samples;

public final class R {

    public static final class string {

        public static final int greeting=0x0204000e;

        public static final int start_button_text=0x02040001;

        public static final int submit_button_text=0x02040008;

        public static final int main_screen_title=0x0204000a;

    };

    public static final class layout {

        public static final int start_screen=0x02070000;

        public static final int new_user_pane=0x02070001;

        public static final int select_user_list=0x02070002;


    };

    public static final class drawable {

        public static final int company_logo=0x02020005;

        public static final int smiling_cat=0x02020006;
```

```
        public static final int yellow_fade_background=0x02020007;

        public static final int stretch_button_1=0x02020008;


    };
};
```

在代码中使用资源

只要知道资源的 ID 以及你编译进目标文件的资源类型就可以在代码里使用它来。下面是一些语法：

`R.resource_type.resource_name`

或者

`android.R.resource_type.resource_name`

`resource_type` 是 R 子类的一种类型。 `resource_name` 是定义在 XML 文件里的资源名或者为其他文件类型定义的资源文件（没有后缀）名。每种类型的资源会被加入到一个特定的 R 的子类中；为了学习哪种 R 的子类里有你编译的资源类型，参考资源引用 文档。被编译进应用程序的资源不需要包的名字就可以直接被访问到（像这样：`R.resource_type.resource_name`）. Android 包含一些标准资源，如屏幕的类型，按钮的背景。要使用这些代码，你需要包含 `android`, 如 `android.R.drawable.button_background`.

这里有一些好的和糟糕的例子说明如何在代码里使用编译后的资源：

```
// 从画图资源类里装载一个当前屏幕背景。

this.getWindow().setBackgroundDrawableResource(R.drawable.my_background_image);



// 错误！ 将一个资源 ID 装入一个需要字符串的方法中

this.getWindow().setTitle(R.string.main_title);
```

```
//正确！需要从资源封装类里取得标题。

this.getWindow().setTitle(Resources.getText(R.string.main_titl
e));



// 从当前屏幕中装载布局数据。

setContentView(R.layout.main_screen);



//从 ViewFlipper 对象中设置动画中一帧 。

mFlipper.setInAnimation(AnimationUtils.loadAnimation(this,

       R.anim.hyperspace_in));



// 在 TextView 对象中设置文本内容。

TextView msgTextView = (TextView)findViewByID(R.id.msg);

msgTextView.setText(R.string.hello_message);
```

资源引用

一个在属性（或者资源）里提供的数值可以被指向一个具体的资源。这常常被使用在布局文件中用于字符串(可以被本地化) 以及图片（存在于其他文件中的），通过一个引用可以是包括颜色和整数的任何资源类型。
例如，如果有 颜色资源，我们可以将文本的颜色值写在布局文件中，颜色值可以从资源文件里取得：

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

<EditText id="text"
```

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

    android:layout_width="fill_parent"
android:layout_height="fill_parent"

    android:textColor="@color/opaque_red"

    android:text="Hello, World!" />
```

注意这里使用‘@’的前缀是说明资源引用 -- 后面的文本是资源的名字 `@[package:]type/name`. 这里我们不需要指定包，因为我们在我们自己的包里引用资源。为了指定一个系统资源，你需要这样写：

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

<EditText id="text"

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

    android:layout_width="fill_parent"
android:layout_height="fill_parent"

    android:textColor="@android:color/opaque_red"

    android:text="Hello, World!" />
```

另外一个例子，当你在布局文件里使用字符串，你必须做资源引用，这样字符串才能被使用：

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

<EditText id="text"

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

    android:layout_width="fill_parent"
android:layout_height="fill_parent"

    android:textColor="@android:color/opaque_red"

    android:text="@string/hello_world" />
```

这段代码也能被用来创建资源间引用。例如，我们能这样创建图像资源：

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

<resources>

    <drawable
id="my_background">@android:drawable/theme2_background</drawab
le>

</resources>
```

主题属性引用

另一种资源数值允许你引用当前主题属性值。这种属性引用只能被用于特殊的资源类以及 XML 属性中；它允许你根据现在主题风格将你定制的 UI 变得更标准化，而不用使用大量的具体数值。
这里有个例子，我们能在布局文件中将文本颜色设置为基本系统主题中定义好的标准颜色：

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

<EditText id="text"

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

    android:layout_width="fill_parent"
android:layout_height="fill_parent"

    android:textColor="?android:textDisabledColor"

    android:text="@string/hello_world" />
```

注意除来我们将前缀'?'代替了'@',其他非常像资源引用。当你使用这个标记，系统会自动查找你提供的属性的名字 -- 资源工具知道肯定会有资源属性相符合，你不需要详细指定（`?android:attr/android:textDisabledColor`).

使用资源标识符到主题里去寻找相应的数据而不是直接使用原数据，其语法和'@'模式是一样的： `?[namespace:]type/name` 这里的 type 是可选择的.

使用系统资源

许多系统资源应用程序是可以使用的。这样的资源定义在"android.R"的类里。 例如，你可以使用下面的代码在屏幕上显示一个标准的应用程序图标：

```
public class MyActivity extends Activity

{

    public void onStart()

    {

        requestScreenFeatures(FEATURE_BADGE_IMAGE);


        super.onStart();


        setBadgeResource(android.R.drawable.sym_def_app_icon);

    }

}
```

用相似的方法，这段代码能将你的屏幕变成系统定义的标准的“绿色背景”：

```
public class MyActivity extends Activity

{

    public void onStart()

    {

        super.onStart();


        setTheme(android.R.style.Theme_Black);

    }
```

```
}
```

对于不同的语言和设置支持不同的资源

你可以根据产品界面语言以及硬件配置设置不同的资源。注意，虽然你可以包含不同的字串，布局以及其他资源，但开发包（SDK）不会给你显式的方法去指定不同的资源去加载。Android 检测你的硬件以及位置信息选择合适的设置去加载。用户可以到设备上的设置界面去选择不同的语言。

要包含不同的资源，在同一目录下创建并行的文件夹，在每个文件夹后加上合适的名字，这个名字能表明一些配置信息（如语言，原始屏幕等等）。例如，这里的项目字符串文件一个是英文版的，另一个是法文版的：

```
MyApp/
    res/
        values-en/
            strings.xml
        values-fr/
            strings.xml
```

Android 支持不同类型的修饰语，并可以加多条在文件夹名的后面， 修饰语之间以破折号分开。例如：一个绘图资源类指定全部配置名称命名会像这样：

```
MyApp/
    res/

drawable-en-rUS-port-160dpi-finger-keysexposed-qwerty-dpad-480x320/
```

更典型的，你可以仅仅指定部分特定的配置选项。只要保证所有的数值都是按顺序排列：

```
MyApp/
    res/
```

```
drawable-en-rUS-finger/

drawable-port/

drawable-port-160dpi/

drawable-qwerty/
```

修饰语	值
语言	两个小写字母 ISO 639-1。例如: `en`, `fr`, `es`
地区	两个大写字母加上一个小写字母'r' ISO 3166-1-alpha-2。 例如: `rUS`, `rFR`, `rES`
屏幕方向	`port`, `land`, `square`？
屏幕像素	`92dpi`, `108dpi`, 等等。
触摸屏类型	`notouch`, `stylus`, `finger`
键盘是否有效	`keysexposed`, `keyshidden`
基本文本输入模式	`nokeys`, `qwerty`, `12key`
无触摸屏的主要 导航模式	`notouch`, `dpad`, `trackball`, `wheel`
屏幕分辨率	`320x240`, `640x480`, 等等。大分辨率需要开始指定。

这个列表不包含一些特殊的参数，如载体，商标，设备/硬件，制造商。任何应用程序需要知道的信息都在资源修饰语里有说明。

这里有一些通用的关于资源目录的命名指导：

- 各个变量用破折号分开 (每个基本的目录名后跟一个破折号)
- 变量大小写敏感（其大小写法必须始终一致）
  例如，
  - 一个 `drawable` 的目录必须命名为 `drawable-port`，而不是 `drawable-PORT`。
  - 你不能有两个目录命名为 `drawable-port` 以及 `drawable-PORT`，甚至故意将"port" 和 "PORT"指为不同的参数也不可以。
- 一个式子里同一个类型修饰语中只有一个值是有效的（你不能指定像这样 `drawable-rEN-rFR/`)
- 你可以指定多个参数去定义不同的配置，但是参数必须是上面表格里的。例如，`drawable-en-rUS-land` 意思在 US-English 的机器里载入风景视图。
- Android 会寻找最适合当前配置的目录，这会在下面描述
- 表格里所列的参数是用来打破平衡以防止多重路径限制。 (看下面的例子)
- 所有目录，无论是限制的，还是不限制的，只要在 `res/` 目录下.一些目录是不能嵌套的（这样 `res/drawable/drawable-en` 是不可以的)
- 所有的资源在被代码引用中最好都使用简单的、不加修饰的名字，如果一个资源这样命名：
  `MyApp/res/drawable-port-92dp/myimage.png`
  它将这样被引用：
  `R.drawable.myimage` (code)
  `@drawable/myimage` (XML)

Android 如何找到最合适的目录

Android 将会挑出哪些基本资源文件在运行时会被使用，这依靠当前的配置。 选择过程如下：

1. 删去一些和当前设备配置不符合的资源。例如，如果屏幕的像素是 108dpi，这可以删除 `MyApp/res/drawable-port-92dpi/`.

```
2.  MyApp/res/drawable/myimage.png
3.  MyApp/res/drawable-en/myimage.png
```

```
4.  MyApp/res/drawable-port/myimage.png
5.  ~~MyApp/res/drawable-port-92dpi/myimage.png~~
```

6. 挑出一些最经常符合配置的资源。例如，如果我们的地区是 en-GB, 方向是 port，那我们有两个符合配置的选项： `MyApp/res/drawable-en/` 和 `MyApp/res/drawable-port/`. 这个目录 `MyApp/res/drawable/` 可以被删除了，因为当另外一个有一次匹配正确，而它没有。

```
7.  ~~MyApp/res/drawable/myimage.png~~
8.  MyApp/res/drawable-en/myimage.png
9.  MyApp/res/drawable-port/myimage.png
```

10. 根据配置的优先级选取最终适合的文件，它们按顺利被排列在上面的表格里。更确切得说，语言匹配比方位匹配更重要, 所以我们可以通过选择语言文件来平衡, MyApp/res/drawable-en/.

```
11. MyApp/res/drawable-en/myimage.png
12. ~~MyApp/res/drawable-port/myimage.png~~
```

术语

资源系统将一系列分散内容集合在一起形成最终的完整的资源功能，去帮助我们了解整个系统。这里有一些核心概念以及组件的概要说明，你在开发中将可能使用到：

最终文件: 应用程序的独立的数据包。这包含所有从 java 程序编译成的目标文件，图像 (例如 PNG 图片)， XML 文件等等。这些文件以一种特定的方式组织在一起，在程序打包最后时，它们被打包成一个独立的 ZIP 文件。

aapt: Android 最终文件打包工具。这个工具产生最终程序的 ZIP 文件。除了将最终的元数据文件打包在一起，它也解析资源定义到最终的二进制数据里。

资源表:aapt 工具产生的特殊的文件，描述了所有在程序/包里的资源。这个文件可以通过资源类来访问；它不能直接和应用程序接触。

资源: 资源表里一条记录描述的是单一的命名值。大体上, 资源分成两种：基本的和包装的.

资源标识符: 在资源表里所有的资源都被唯一的整数标识着。 所有的代码中（资源描述，XML 文件，Java 源代码）你可以直接使用符号名代替真实的整数数值。

基本资源: 所有基本资源都可以被写成一个简单的字串，使用一定的格式可以描述资源系统里各种不同的基本类型： 整数，颜色，字串，其他资源的引用，等等。像图片以及 XML 描述文件这些复杂资源，被以基本字串资源储存，它们的值就是相关最终数据文件的路径。

包装资源: 有一种特殊类型的资源，不是简单的字符串，而是有一个随意的名字/数值配对列表。每个数值可以对应它本身的资源标识，每个值可以持相同类型的字符串格式的数据作为一个正常的资源。包装资源支持继承：一个包里的数据能从其他包里继承，有选择地替换或者扩展能产生你自己需要的内容。

种类: 资源种类是对于不同需求的资源标识符而言的。例如，绘制资源类常常实例化绘制类的对象，所以这些包含颜色以及指向图片或 XML 文件的字符串路径数据是原始数据。其它常见资源类型是字符串（本地化字符串），颜色（基本颜色），布局（一个指向 XML 文件的字串路径，它描述的是一个用户界面）以及风格（一个描述用户接口属性的包装资源）。还有一个标准的“attr”资源类型，它定义了命名包装数据以及 XML 属性的资源标识符。

风格: 包含包装资源类型的名字常常用来描述一系列用户接口属性。例如，一个 TextView 的类可能会有一个描述界面风格的类来定义文本大小，颜色以及对齐方式。在一个界面布局的 XML 文件中，可以使用“风格” 属性来确定整体界面风格，它的值就是风格资源的名字。

风格类: 这里将详述一些属性资源类。其实数据不会被放在资源表本身，通常在源代码里它以常量的形式出现，这也可以使你在风格类或者 XML 的标签属性里方便找到它的值。例如，Android 平台里定义了一个“视图”的风格类，它包含所有标准视图的属性：画图区域，可视区域，背景等。这个视图被使用时，它就会借助风格类去从 XML 文件取得数据并将其载入到实例中。

配置: 对许多特殊的资源标识符，根据当前的配置，可以有多种不同的值。配置包括地区（语言和国家），屏幕方向，屏幕分辨率，等等。当前的配置用来选择当资源表载入时哪个资源值生效。

主题: 一个标准类型的资源能为一个特殊的上下文提供全局的属性值。例如，当应用工程师写一个活动时，他能选择一个标准的主题去使用，白色的或者黑色的；这个类型能提供很多信息，如屏幕背景图片/颜色，默认文本颜色，按钮类型，文本编辑框类型，文本大小，等。当布置一个资源布局时，控件（文本颜色，选中后颜色，背景）的大部分设置值取自当前主题；如果需要，布局中的风格以及属性也可以从主题的属性中获得。

覆盖层: 资源表不能定义新类型的资源，但是你可以在其他表里替换资源值。就像配置值，这可以在装载时候进行；它能加入新的配置值（例如，改变字串到新的位置），替换现有值（例如，将标准的白色背景替换成"Hello Kitty"的背景图片），修改资源包（例如修改主题的字体大小。白色主题字体大小为 18pt）。这实际上允许用户选择设备不同的外表，或者下载新的外表文件。

资源引用

资源引用 这份文档提供了不同类型资源的详细列表，并提供了如何在 Java 代码中使用资源以及如何引用资源的描述。

国际化和本地化

即将完成: 国际化和本地化是非常关键的，但现在的 SDK 还没有完全支持好。当 SDK 成熟时， 这个章节会包含 Android 平台国际化和本地化的相关信息。 那时，外部字串以及良好的结构将会使得创建和使用资源变得更省事。

三、开发工具箱

Android 设计哲学

即使平台之间有很大的不同，但是如何利用 API 创建应用程序的学习过程是大同小异的。一般来说，有两个步骤：首先，应该知道怎么用 API 实现你的功能。其次，要了解平台间的细微差别。换句话说，首先你应该学会如何创建应用程序（了解应用程序的基本结构等），然后就要学会根据具体情况实现这个应用程序。

相比而言，第二阶段（学习使用正确的方法来实现应用程序）通常需要很长一段时间，在这个过程中你会不断地写代码，犯错误，然后从错误中吸取教训。显然，这不是一个有效的学习方法，本小节和下面的一些连接针对这向你伸出援助之手，教你怎么学习创建你的应用程序。

在此之前，先讲一个要点：成功的应用程序往往提供一个突出的用户体验。当 Android 团队构建了一个有着健壮核心的系统时，大多数的用户体验将来源于用户和应用程序之间的的交互。因此，我们鼓励你们花时间去构建应用程序优秀的用户体验。

显著的用户体验体现在三个核心特征上：*1*、*快速*；*2*、*响应*；*3*、*无缝*。当然，自从计算机出现以后，每一个平台都曾经有过类似的三种性质。尽管如此，每个平台实现这些特性的方式也有所不同；下面将会简单的介绍在 Android 平台下面你的应用程序将如何达到这些要求。

快速（Fast）

Android 程序执行应该是很快的。当然，准确来说它的程序应该执行的很有*效率*（有效率才会快）。在目前的计算机世界里哟这样一个倾向：假设摩尔定律能够最终解决我们所有的问题。当这种倾向遇到嵌入式应用程序的时候，摩尔定律就变得有些复杂了。

与桌面和服务应用程序不一样，摩尔定律在移动设备应用程序上面不是真正适用的。摩尔定律实际上是关于电子晶体管集成密度的，它原本的含义是：随着时间的流逝，在给定的电路尺寸上面可以集成更多的电路。对于桌面和服务应用陈旭来说，它的含义是：你可以打包更多的“速度”在一个大小差不多的芯片上面，速度的提高，其他相应的一些性能也会显著的提高。对以像手机这样的嵌入式应用程序来说，相反的，使用摩尔定律

是为了使芯片变得更小。这样，随着芯片密度的提高，相同功能的芯片会变得越来越小，功耗会越来越低，从而使手机做的越来越小，电池的持续的时间越来越长。这样就导致手持嵌入式设备相对于桌面系统来说正在以一种比较慢二实际的速度在增涨。因而，对于嵌入式设备来说，摩尔定律就意味着更多的功能和更少的功耗，速度只是次要的。
这就是我们要写出高效代码的原因：你不能天真的认为电话在速度上面的增涨速度和桌面、服务应用程序是一样的。一般来说，高效的代码意味着最小的内存占用，意味着紧凑的风格，意味着避免了因为某种语言和编码习惯对性能的影响。我们用面向对象这个概念来理解，大部分这样的工作都是在*方法*层面上，与实际的代码，循环等等相类似。在 “编写高效 Android” 一文中，我们会对此做详细的介绍。

响应（Responsive）

我们有可能能够编写赢得世界上所有的性能测试的代码，但是用户使用起来会感到很恼火，这是因为应用程序没有足够的*响应性*——让人感觉反映迟钝，在关键的时刻失灵，或者处理输入太慢。在 Android 平台下，那些响应性不够的应用程序会经常弹出 "Application Not Responding" (ANR)这样的致命消息。
通常，这会在应用程序不能响应用户的输入的情况下发生。例如，你的应用程序在一些 I/O 操作上（如网络接口调用）阻塞，这是主线程将不会处理用户的输入事件，一段时间之后应用系统就会认为你的程序挂起了，就会给一个选项给用户询问是否结束它。同样的，如果你的应用程序花费很多时间去构建内存中的一个结构、或者计算游戏的下一步，这时系统同样会认为程序已经挂起。当碰到上面情况的时候，要确保计算的高效性，但是即使是最高效的代码也需要花费时间。
在上面两个例子中，问题的解决方案是建立一个子线程来处理大部分的工作，这样就能保证你的主线程（响应用户界面事件）一直运行，这样防止系统认为你的程序已经僵化。因为这种线程的实现一般在*“类”（CLASS）*这个层次上，你可以把响应当成是类的问题来处理（这里，可以和*方法*层次描述的基本性能相比较）。
这里只是一个简单的介绍，在 “构建响应 Android 应用程序” 一文中对于应用程序的响应性有详细的介绍。

无缝性（Seamless）

即使是你的应用程序执行很快，并且具有很高的响应性，它仍然有可能让用户苦恼。一个常见的例子是后台进程(比如 Android 的 Service 和 BroadcastReceiver)对某些事件可能会突然弹出一个 UI 响应。这似乎是无关紧要的，并且一般开发者会认为这这是正常的，因为他们花费了戴亮时间去测试和使用自己的应用程序。可是，Android 应用程序模型的构建是能够允许用户在不同的应用程序之间进行流畅的切换。这就意味着，当你的后台进程实际上弹出那个 UI 的时候，用户可能正在系统的其他部分中，做一些其他的事情，如在接电话。想像一下，如果 SMS 服务每次都会在文本消息传入时弹出一

个对话框，这很快就会使用户崩溃。这就是为什么 Android 标准对于这些事件使用的是通知（Notifications）机制；这使用户能够自己控制。
这仅仅是一个例子，相似的例子数不胜数。比如，如果 Activities 没有正确的实现 onPause()方法和其他生命周期方法，这将会导致数据丢失。或者如果你的应用程序有意的暴露数据给其他应用程序使用，你应该使用一个 ContentProvider，而不是用一个路人皆可见的未加工过的文件或者数据库。
这些例子有一个共同的特点，他们都涉及到程序与程序或则程序与系统之间的交互。系统被设计为将多个应用程序视为一种松耦合组件的联合，而不是大块的代码黑盒。这就允许作为开发人员的你将整个系统看作是一个这些组件的大联合。这允许你干净地封装，无缝地和其他应用程序结合，因而你能设计自己喜欢的程序。
这使一种*组件*层次的概念（与性能和响应的*类*层次和*方法*层次相对应）。至于怎样编写无缝性能很高的代码， “与系统相结合” 一文中将会对此做出介绍，提供代码提示和最佳实例。

构建自定义组件

Android 中，你的应用程序程序与 View 类组件有着一种固定的联系，例如按钮（Button）、文本框（TextView），可编辑文本框（EditText），列表框（ListView），复选框（CheckBox），单选框（RadioButton），滚动条（Gallery），微调器（Spinner），等等，还有一些比较先进的有着特殊用途的 View 组件，例如 AutoCompleteTextView, ImageSwitcher 和 TextSwitcher。除此之外，种类繁多的像 线性布局（LinearLayout），框架布局（FrameLayout），这样的布局组件（Layout）也被认为是 View 组件，他们是从 View 类派生过来的。
你的应用程序就是这些控制组件和布局组件以某种方式结合显示在屏幕上，一般来说这些组件对你来说基本够用，但是你也应该知道你是可以通过类继承创建属于自己的组件，一般可以继承像 View、Layouts（布局组件）这样的组件，甚至可以是一些比较高级的控制类组件。下面我们说一下为什么要继承：

- 你可以为实现某种功能创建一个完全自定义风格的组件，例如用二维的图形创建控制组件实现声音的控制，就像电子控制一样。
- 你可以把几种组件结合形成一个新的组件，你的组件可能同时包含 ComboBox（一个能输入的文本列表）和 dual-pane selector control（左右两个 List 窗口，你可以分配窗口每一项的从属关系）等等。
- 你可以创建自己的布局组件（Layout）。SDK 中的布局组件已经提供了一系列的选项让你打造属于自己的应用程序，但是高级的开发人员会发现根据现有的 Layout 组件开发新的 Layout 组件是很有必要的，甚至是完全从底层开发新的组件。
- 你可以覆盖一个现有组件的显示或功能。例如，改变 EditText（可编辑文本）组件在屏幕上的显示方式（可以参考 Notepad 的例子，里面教你如何创建一个下划线的显示页面）。

- 你可以捕获像按键按下这样的事件，以一些通用的方法来处理这些事件（一个游戏的例子）。

为了实现某种目标你可能很有必要扩展一个已经存在的 View 组件，下面我们结合一些例子教你如何去做。

内容：

基本方法（The Basic Approach ）

完全自定义组件（Fully Customized Components ）

定制组件的例子（Customized Component Example ）

组件的混合（或者控制类的混合） （Compound Components (or Compound Controls) ）

修改现有组件（Tweaking an Existing Component ）

小结（Go Forth and Componentize ）

基本方法（The Basic Approach ）

下面的一些步骤都比较概括，教你如何创建自己的组件：

1. 让你的类（Class）继承一个现有的 View 类或 View 的子类。
2. 重载父类的一些方法：需要重载的父类方法一般以‘on’开头，如 onDraw(), onMeasure()和 onKeyDown()等等。
    - 这个在 Activity 或则 ListActivity 派生中同样适用，你需要重载一些生命周期函数和一些其他功能性的 HOOK 函数。
3. 使用你的继承类：一旦你的继承类创建完成，你可以在基类能够使用的地方使用你的继承类，但完成功能就是你自己编写的了。

继承类能够定义在 activities 里面，这样你能够方便的调用，但是这并不是必要的（或许在你的应用程序中你希望创建一个所有人都可以使用的组件）。

完全自定义组件（Fully Customized Components）

完全自定义组件的方法可以创建一些用于显示的图形组件（graphical components），也许是一个像电压表的图形计量器，或者想卡拉 OK 里面显示歌词的小球随着音乐滚动。无论那种方式，你也不能单纯的利用组件的结合完成，无论你怎么结合这些现有的组件。

幸运的是，你可以以你自己的要求轻松地创建完全属于自己的组件，你会发现不够用的只是你的想象力、屏幕的尺寸和处理器的性能（记住你的应用程序最后只会在那些性能低于桌面电脑的平台上面运行）。
下面简单介绍如何打造完全自定义的组件：

1. 最为通用的 VIEW 类的父类毫无疑问是 View 类，因此，最开始你要创建一个基于此类的一个子类。
2. 你可以写一个构造函数从 XML 文件中提取属性和参数，当然你也可以自己定义这些属性和参数（也许是图形计量器的颜色和尺寸，或者是指针的宽度和幅度等等）
3. 你可能有必要写自己的事件监听器，属性的访问和修改函数和一些组件本身的功能上的代码。
4. 如果你希望组件能够显示什么东西，你很有可能会重载 `onMeasure()` 函数，因而你就不得不重载 `onDraw()` 函数。当两个函数都用默认的，那么 `onDraw()` 函数将不会做任何事情，并且默认的 `onMeasure()` 函数自动的设置了一个 100x100 —的尺寸，这个尺寸可能并不是你想要的。
5. 其他有必要重载的 `on...` 系列函数都需要重新写一次。

`onDraw()`和 `onMeasure()`

`onDraw()`函数将会传给你一个 Canvas 对象，通过它你可以在二维图形上做任何事情，包括其他的一些标准和通用的组件、文本的格式，任何你可以想到的东西都可以通过它实现。
注意: 这里不包括三维图形如果你想使用三维的图形，你应该把你的父类由 View 改为 SurfaceView 类，并且用一个单独的线程。可以参考 GLSurfaceViewActivity 的例子。

`onMeasure()` 函数有点棘手，因为这个函数是体现组件和容器交互的关键部分，*onMeasure*（）应该重载，让它能够有效而准确的表现它所包含部分的测量值。这就有点复杂了，因为我们不但要考虑父类的限制（通过 *onMeasure*（）传过来的），同时我们应该知道一旦测量宽度和高度出来后，就要立即调用 *setMeasuredDimension()* 方法。

概括的来讲，执行 `onMeasure()`函数分为一下几个阶段：

1. 重载的 `onMeasure`（）方法会被调用，高度和宽度参数同时也会涉及到（`widthMeasureSpec` 和 `heighMeasureSpec` 两个参数都是整数类型），同

时你应该考虑你产品的尺寸限制。这里详细的内容可以参考 View.onMeasure(int, int) （这个连接内容详细的解释了整个 measurement 操作）。

2. 你的组件要通过 `onMeasure()` 计算得到必要的 measurement 长度和宽度从而来显示你的组件，它应该与规格保持一致，尽管它可以实现一些规格以外的功能（在这个例子里，父类能够选择做什么，包括剪切、滑动、提交异常或者用不同的参数又一次调用 `onMeasure()` 函数）。

3. 一旦高度和宽度计算出来之后，必须调用 `setMeasuredDimension(int width, int height)`，否则就会导致异常。

一个自定义组件的例子（A Customized Component Example）

在 API Demos 中的 CustomView 提供了以一个自定义组件的例子，这个自定义组件在 LabelView 类中定义。
LabelView 例子涉及到了自定义组件的方方面面：

- 首先让自定义组件从 View 类中派生出来。
- 编写带参数的构造函数（参数可以来源于 XML 文件）。这里面的一些处理都已经在 View 父类中完成，但是任然有些 Labelview 使用的自定义组件特有的新的参数需要处理。
- 一些标准的 Public 函数，例如 `setText()`, `setTextSize()`, `setTextColor()`
- 重载 `onMeasure()` 方法来确定组件的尺寸（注意：在 LabelView 中是通过一个私有函数 `measureWidth()`来实现的）
- 重载 `onDraw()`函数把 Lable 显示在提供的 canvas 上。

在例子中，你可以通过 custom_view_1.xml 看到自定义组件 LabelView 的用法。在 XML 文件中特别要注意的是 `android:` 和 `app:` 两个参数的混合运用，`app:` 参数表示应用程序中被认为是 LabelView 组件的个体，这些也会作为资源在 R 类中定义。

组件混合技术 Compound Components (or Compound Controls)

如果你不想创建一个完全自定义的组件，而是由几个现有组件的组合产生的新的组件，那么混合组件技术就更加适合。简单的来说，这样把几个现有的组件融合到一个逻辑组合里面可以封装成一个新的组件。例如，一个 Combo Box 组件可以看作是是一个 EditText 和一个带有弹出列表的 Button 组件的混合体。如果你点击按钮为列表选择一项，

在 Android 中，其实还有其他的两个 View 类可以做到类似的效果： Spinner 和 AutoCompleteTextView,，但是 Combo Box 作为一个例子更容易让人理解。

下面简单的介绍如何创建组合组件：

1. 一般从 Layout 类开始，创建一个 Layout 类的派生类。也许在 Combo box 我们会选择水平方向的 LinearLayout 作为父类。记住，其他的 Layout 类是可以嵌套到里面的，因此混合组件可以是任何组件的混合。注意，正如 Activity 一样，你既可以使用外部 XML 文件来声明你的组件，也可以嵌套在代码中。
2. 在新的混合组件的构造函数中，首先，调用所有的父类的构造函数，传入对应的参数。然后可以设置你的混合组件的其他的一些方面，在哪创建 EditText 组件，又在哪创建 PopupList 组件。注意：你同时也可以在 XML 文件中引入一些自己的属性和参数，这些属性和参数也可以被你的混合组件所使用。
3. 你也可以创建时间监听器去监听新组件中 View 类触发的事件，例如，对 List 选项单击事件的监听，你必须在此时间发生后更新你 EditText 的值。
4. 你可能创建自己的一些属性，带有访问和修改方法。例如，允许设置 EditText 初始值并且提供访问它的方法。
5. 在 Layout 的派生类中，你没有必要去重载 `onDraw()` 和 `onMeasure()` 方法，因为 Layout 会有比较好的默认处理。但是，如果你觉得有必要你也可以重载它。
6. 你也可能重载一些 `on` 系列函数，例如通过 `onKeyDown()`的重载，你可以通过按某个键去选择列表中的对应的值。

总之，把 Layout 类作为基类有下面几个优点：

- 正如 activity 一样，你也可以通过 XML 文件去声明你的新组件，或者你也可以在代码中嵌套。
- `onDraw()` 函数和 `onMeasure()`函数是没有必要重载的，两个函数已经做得很好了。
- 你可以很快的创建你的混合组件，并且可以像单一组件那样使用。

混合组件的例子（Examples of Compound Controls）

In the API Demos project 在 API Demos 工程中，有两个 List 类的例子——Example 4 和 Example 6，里面的 SpeechView 组件是从 LinearLayout 类派生过来，实现显示演讲显示功能，对应的原代码是 `List4.java` 和 `List6.java`。

调整现有组件（Tweaking an Existing Component）

在某些情况下，你可能有更简单的方法去创建你的组件。如果你应经有了一个非常类似的组件，你所要做的只是简单的从这个组件派生出你的组件，重在其中一些有必要修改的方法。通过完全自定义组件的方法你也可以同样的实现，但通过冲 View 派生产生新的组件，你可以简单获取一些已经存在的处理机制，这些很可能是你所想要的，而没有必要从头开始。

例如，在 SDK 中有一个 NotePad 的例子（NotePad application ）。该例子演示了很多 Android 平台实用的细节，例如你会学到从 EditView 派生出能够自动换行的记事本。这还不是一个完美的例子，因为相比早期的版本来说，这些 API 已经感变了很多，但它确实说明了一些问题。

如果你还未查看该程序，现在你就可以在 Eclipse 中导入记事本例程（或仅通过提供的链接查看相应的源代码）。特别是查看 NoteEditor.java 中的 `MyEditText` 的定义。

下面有几点要注意的地方：

1. **声明（The Definition）**
   这个类是通过下面一行代码来定义的：
   ```
   public static class MyEditText extends EditText
   ```
   - 它是定义在 `NoteEditor activity` 类里面的，但是它是共有的（public），因此如果有必要，它可以通过 `NoteEditor.MyEditText` 从 `NoteEditor` 外面来调用。
   - 它是 `static` 类（静态类），意味着不会出现所谓的通过父类访问数据的“虚态方法”， 这样就使该类成为一个可以不严重依赖 `NoteEditor` 的单独类。对于不需要从外部类访问的内联类的创建，这是一个很清晰地思路，保证所产生的类很小，并且允许它可以被其他的类方便的调用。
   - 它是 `EditText` 类的扩展，它是我们选择的用来自定义的父类。当我们完成以后，新的类就可以作为一个普通的 `EditText` 来使用。

2. **类的初始化**

一般来说，父类是首先调用的。进一步来说，这不是一个默认的构造函数，而是一个带参数的构造函数。因为 EditText 是使用从 XML 布局文件提取出来的参数进行创建，因此我们的构造函数也要取出参数并且将这些参数传递给父类。

3. **方法重载**

在本例中，仅对 onDraw()一个方法进行重载。但你可以很容易地为你的定制组件重载其他需要的方法。

对于记事本例子来说，通过重载 onDraw()方法我们可以在 EidtView 的画布（canvas）上绘制蓝色的线条（canvas 类是通过重写的 onDraw()方法传递）。

该函数快要结束时要调用 super.onDraw()函数。父类的方法是应该调用，但是在这个例子里面，我们是在我们划好了蓝线之后调用的。

4. **使用定制组件**

现在，我们已经有自己定制的组件了，但是应该怎样使用它呢？在记事本例子中，定制的组件直接在预定义的布局文件中使用，让我们看一看 res/layout 目录中的 note_editor.xml 文件。

```
<view
xmlns:android="http://schemas.android.com/apk/res/android"
  class="com.android.notepad.NoteEditor$MyEditText"
  id="@+id/note"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:background="@android:drawable/empty"
  android:padding="10dip"
  android:scrollbars="vertical"
  android:fadingEdge="vertical" />
```

- 该自定义组件在 XML 中是作为一个一般的 View 类来创建的，并且是通过全路径包来描述的。注意这里内联类是通过 NoteEditor$MyEditText 来表示的，这是 Java 编程中引用内联类的标准方法。

- 在定义中的其他属性和参数将传递给定制组件的构造函数，然后才传到 EditText 构造函数中，因此这些参数也是你使用 EditText 组件的参数。注意，这里你也可以增加你自己的参数，我们将在下面讨论这个问题。

这就是你全部需要做的，诚然这是一个简单的例子。但问题的关键是：你的需求有多复杂，那么你的自定义组件就有多么复杂。

一个更为复杂的组件可能需要重载更多的 on 系列函数，并且还要很多特有的函数来充分实现自定义组件的功能。唯一的限制就是你的想象力和你需要组件去执行什么工作。

现在开始你的组件化之旅吧

如你所见，Android 提供了一种精巧而又强大的组件模型，让你尽可能的完成你的工作。从简单的组件调整到组件混合，甚至完全自定义组件，灵活的运用这些技术，你应该可以得到一个完全符合你外观要求的的 Android 程序

Android 平台的可选 API

Android 适用于各种各样的手机，从最低端直到最高端的智能手机。核心的 Android API 在每部手机上都可使用，但任然有一些 API 接口有一些特别的适用范围：这就是所谓的“可选 API”。

这些 API 之所以是“可选的”，主要是因为一个手持设备并不一定要完全支持这类 API，甚至于完全不支持。例如，一个手持设备可能没有 GPS 或 Wi-FI 的硬件。在这个条件下，这类功能的 API 任然存在，但不会以相同的方式来工作。例如 Location API 任然在没有 GPS 的设备上存在，但极有可能完全没有安装功能提供者，意味着这类 API 就不能有效地使用。

你的应用应该无障碍地运行或连接在一个可能不支持你 API 的设备，因为你的设备上有这些上层接口（the classes）。当然执行起来可能什么也不会做，或者抛出一个异常。每个 API 会做些什么我们可以参考这些 API 的说明文档，你应该编写你的程序来适当的处理这类问题。

Wi-Fi API

Wi-Fi API 为应用程序提供了一种与那些带有 Wi-FI 网络接口的底层无线堆栈相互交流的手段。几乎所有的请求设备信息都是可利用的，包括网络的连接速度、IP 地址、当前状态等等，还有一些其他可用网络的信息。一些可用的交互操作包括扫描、添加、保存、结束和发起连接。

Wi-Fi API 在 android.net.wifi 包中。

定位服务（Location-Based Services）

定位服务允许软件获取手机当前的位置信息。这包括从全球定位系统卫星上获取地理位置，但相关信息不限于此。例如，未来其他定位系统可能会运营，届时，对其相应的 API 接口也会加入到系统中。
定位服务的 API 在 android.location 包中。
点击这里查看更多有关 Android 定位 API 的信息。

多媒体 API（Media APIs）

多媒体 API 主要用于播放媒体文件。这同时包括对音频（如播放 MP3 或其他音乐文件以及游戏声音效果等）和视频（如播放从网上下载的视频）的支持，并支持"播放 URI 地址"（Note:URI 即是统一资源识别地址）模式－在网络上直接播放的流媒体。技术上来说，多媒体 API 并不是“可选的”，因为它总是要用到。但是不同的硬件环境上面可能有不同的编解码的硬件机制，因而它又是“可选的”。
多媒体 API 在 android.media 包中。
点击这里查看更多有关 Android 多媒体 API 的信息。

基于 OpenGL 的 3D 图形（3D Graphics with OpenGL）

Android 的主要用户接口框架是一个典型的面向控件的类继承系统。但不要让表面的情况迷惑了你，因为在它下面是一种非常快的 2D 和 3 D 组合的图形引擎，并且支持硬件加速。用来访问平台 3D 功能的 API 接口是 OpenGL ES API。和多媒体 API 一样，OpenGL 也不是严格意义上的“可选”，因为这些 API 会总是存在并且实现那些固定的功能。但是，一些设备可能有硬件加速环节，使用它的时候就会影响你的应用程序的表现。
OpenGL 的 API 在 android.opengl 中可以看到。
点击这里查看 OpenGL API 的介绍。

四、参考资料

浩瀚的参考资料，不懂就查。

五、API DEMO

The API Demos include sample code for many aspects of the Android APIs, from screen layout to Intent resolution.

App

Activity

Hello World

Demonstrates a basic screen activity.

Code:
HelloWorld.java

Layout:
hello_world.xml

Save & Restore State

Demonstrates how an activity should save state when it is paused.

Persistent State

Demonstrates how you can save and restore preferences, which are stored even after the user closes the application.

Receive Result

Demonstrates how an activity screen can return a result to the activity that opened it.

Forwarding

Demonstrates opening a new activity and removing the current activity from the history stack, so that when the user later presses BACK they will not see the intermediate activity.

Redirection

Demonstrates how to save data to preferences and use it to determine which activity to open next.

Translucent

Demonstrates how to make an activity with a transparent background.

TranslucentBlur

Demonstrates how to make an activity with a transparent background with a special effect (blur).

Service

Local Service Controller

Starts and stops the service class LocalService that runs in the same process as the activity, to demonstrate a service's lifecycle when using {@link android.content.Context#startService Context.startService} and {@link android.content.Context#stopService Context.stopService}.

Local Service Binding

Demonstrates binding to a service class LocalService that runs in the same process as the activity, to demonstrate using the {@link android.content.Context#bindService Context.bindService} and {@link android.content.Context#unbindService Context.unindService} methods with a service. This also shows how you can simplify working with a service when you know it will only run in your own process.

Remote Service Controller

Demonstrates starting a service in a separate process, by assigning `android:process=":remote"` to the service in the AndroidManifest.xml file.

Remote Service Binding

Demonstrates binding to a remote service, similar to the Local Service Binding sample, but illustrating the additional work (defining aidl interfaces) needed to interact with a service in another process. Also shows how a service can publish multiple interfaces and implement callbacks to its clients.

Service Start Arguments Controller

Demonstrates how you can use a Service as a job queue, where you submit jobs to it with {@link android.content.Context#startService Context.startService} instead of binding to the service. Such a service automatically stops itself once all jobs have been processed. This can be a very convenient way to interact with a service when you do not need a result back from it.

Code:
ServiceStartArgumentsController.java

ServiceStartArguments.java

Layout:
service_start_arguments_controller.xml

Alarm

Alarm Controller

Demonstrates two ways you can schedule alarms: a one-shot alarm that will happen once at a given time, and a repeating alarm that will happen first at a given time and then continually trigger at regular intervals after that.

Code:
AlarmController.java

OneShotAlarm.java

RepeatingAlarm.java

Layout:
alarm_controller.xml

Alarm Service

Demonstrates how you can schedule an alarm that causes a service to be started. This is useful when you want to schedule alarms that initiate long-running operations, such as retrieving recent e-mails.

Code:
AlarmService.java

AlarmService_Service.java

Layout:
alarm_service.xml

Notification

NotifyWithText

Demonstrates popup notifications of varying length.

IncomingMessage

Demonstrates sending persistent and transient notifications, with a View object in the notification. It also demonstrated inflating a View object from an XML layout resource.

Search

SearchInvoke

Demonstrates various ways in which activities can launch the Search UI.

SearchQueryResults

Demonstrates an activity that receives Search intents and handles them.

SearchSuggestionSampleProvider

Demonstrates how to configure and use the built-in "recent queries" suggestion provider.

Files

- AdvancedPreferences.java
- AlarmController.java
- AlarmService.java
- AlarmService_Service.java
- AlertDialogSamples.java
- ContactsFilter.java
- ContactsFilterInstrumentation.java
- ContactsSelectInstrumentation.java
- CustomDialogActivity.java
- CustomTitle.java
- DefaultValues.java
- DialogActivity.java
- ForwardTarget.java
- Forwarding.java
- HelloWorld.java
- IncomingMessage.java
- IncomingMessageView.java
- Intents.java
- LauncherShortcuts.java
- LaunchingPreferences.java
- LocalSample.java
- LocalSampleInstrumentation.java
- LocalService.java
- LocalServiceBinding.java
- LocalServiceController.java
- MenuInflateFromXml.java
- MyPreference.java
- NotificationDisplay.java
- NotifyWithText.java

- NotifyingController.java
- NotifyingService.java
- OneShotAlarm.java
- PersistentState.java
- PreferenceDependencies.java
- PreferencesFromCode.java
- PreferencesFromXml.java
- ReceiveResult.java
- RedirectEnter.java
- RedirectGetter.java
- RedirectMain.java
- RemoteService.java
- RemoteServiceBinding.java
- RemoteServiceController.java
- ReorderFour.java
- ReorderOnLaunch.java
- ReorderThree.java
- ReorderTwo.java
- RepeatingAlarm.java
- SaveRestoreState.java
- SearchInvoke.java
- SearchQueryResults.java
- SearchSuggestionSampleProvider.java
- SendResult.java
- ServiceStartArguments.java
- ServiceStartArgumentsController.java
- StatusBarNotifications.java
- TranslucentActivity.java
- TranslucentBlurActivity.java
- VoiceRecognition.java

Content

Resources

Styled Text

Demonstrates loading styled text (bold, italic) defined in a resource file.

Resources

Demonstrates loading styled strings from a resource file, and extracting the raw text.

Files

- ReadAsset.java
- ResourcesSample.java

- StyledText.java

View

RelativeLayout

1. Vertical

Demonstrates a simple relative layout.

2. Simple Form

Demonstrates a more complex relative layout to create a form.

LinearLayout

1. Vertical

Demonstrates a simple LinearLayout, with child width set to WRAP_CONTENT.

2. Vertical (Fill Screen)

Demonstrates a simple LinearLayout, with child width set to FILL_PARENT.

3. Vertical (Padded)

Demonstrates a LinearLayout where one of the elements can expand to fill any remaining screen space (weight=1).

4. Horizontal

Demonstrates a horizontal LinearLayout, plus an expanding column.

5. Simple Form

Demonstrates nested layouts to create a user form.

6. Uniform Size

LinearLayout which uses a combination of wrap_content on itself and fill_parent on its children to get every item to be the same width.

7. Fill Parent

Demonstrates a horizontal linear layout with equally sized columns. Some columns force their height to match the parent.

8. Gravity

Demonstrates a simple linear layout with menu options demonstrating horizontal and vertical gravity options.

9. Layout Weight

Demonstrates how the layout_weight attribute can shrink an element too big to fit on screen.

ScrollView

1. Short

Demonstrates scrolling screen with buttons altermating with a text view.

2. Long

Demonstrates a longer scrolling screen similar to ScrollView1.

TableLayout

1. Basic

Demonstrates a basic TableLayout with identical children.

2. Empty Cells

Demonstrates a TableLayout with column-spanning rows and different child objects.

3. Long Content

Rows have different number of columns and content doesn't fit on screen: column 4 of row 2 shrinks all of the other columns

4. Stretchable

Demonstrates a TableLayout with a stretchable column.

5. Spanning and Stretchable

Demonstrates a complex TableLayout with spanning columns and stretchable columns to create a menu-like layout.

6. More Spanning and Stretchable

Similar to example 5, but with an additional "checked" column.

7. Column Collapse

Similar to example 6, but now with buttons on the bottom of the screen that enable you dynamically hide or show columns.

8. Toggle Stretch

Demonstrates toggling the "stretch" value on a column to fill the screen width.

9. Toggle Shrink

Demonstrates toggling the "shrink" value on a column to make an over-wide table shrink to fit the screen size.

10. Simple Form

Demonstrates using a table to design a user form.

11. Gravity

Demonstrates the use of advanced gravity attributes, such as *center_horizontal* and *right|bottom* to align cell contents in a table.

12. Various Widths

Demonstrates the use of elements of various widths in a table.

Baseline

Demonstrates the use of the *android:layout_alignBaseline* XML attribute in various page layouts.

1. Top

Demonstrates the default baseline alignment in a simple LinearLayout with items at the top of

the screen.

2. Bottom

Demonstrates the default baseline alignment in a simple LinearLayout with items at the bottom of the screen.

3. Center

Demonstrates the default baseline alignment in a simple LinearLayout with items in the center of the screen.

4. Everywhere

Demonstrates the default baseline alignment in a complex LinearLayout.

6. Multi-line

Demonstrates a baseline alignment with a multiline field.

7. Relative

Demonstrates baseline alignment in a RelativeLayout.

BaselineNested1

Demonstrates baseline aligning specific elements in three parallel vertical LinearLayout objects.

BaselineNested2

Demonstrates baseline aligning specific elements in three mixed vertical and horizontal LinearLayout objects.

BaselineNested3

Demonstrates baseline alignment within nested LinearLayout objects.

Radio Group

Radio Group

Demonstrates using radio buttons and capturing the selected item.

ScrollBars

1. Basic

Demonstrates a scrollable LinearLayout object.

2. Fancy

Demonstrates a scrollable LinearLayout object with a custom thumb slider image.

Visibility

Visibility

Demonstrates toggling the visibility of a View object between visible, invisible, and gone.

Lists

1. Array

Demonstrates binding a ListAdapter to a string array as a data source, and displaying the elements on the screen.

2. Cursor (People)

Demonstrates binding results from a database query to a field in a template.

3. Cursor (Phones)

Demonstrates binding multiple columns from a database query to fields in a template.

4. ListAdapter

Demonstrates implementing a custom ListAdapter to return View objects laid out in a custom manner.

5. Separators

Demonstrates implementing a custom ListAdapter that includes separators between some items.

6. ListAdapter Collapsed

Demonstrates another custom list adapter with that returns expandible items.

7. Cursor (Phones)

Demonstrates a list adapter where data comes from a Cursor object.

8. Photos

Demonstrates a list activity that uses a custom ListAdapter, setting the view for an empty item, and also how to customize the layout of a ListActivity.

Custom

CustomView

Demonstrates implementing a custom view subclass.

ImageButton

ImageButton

Demonstrates an ImageButton: a button with an arbitrary graphic on it.

Date Widgets

1. Dialog

Demonstrates the DatePickerDialog and TimePickerDialog picker dialogs.

2. Inline

Demonstrates using a TimePicker directly in a layout without using a confirmation button or dialog.

Gallery

1. Icons

Demonstrates implementing a Gallery widget and extending GalleryAdapter to create a custom class to serve out source images to the widget.

2. People

Demonstrates populating a Gallery with images from the contacts photos.

Spinner

Spinner

Demonstrates populating two Spinner widgets with values.

Grid

1. Icon Grid

Demonstrates populating a GridView widget with a list of applications using a custom ListAdapter object.

2. Photo Grid

Demonstrates populating a GridView widget with images using a custom ListAdapter object.

ImageSwitcher

ImageSwitcher

Demonstrates using the ImageSwitcher widget with a custom Adapter.

TextSwitcher

TextSwitcher

Demonstrates using the TextSwitcher widget.

Animation

1. Shake

Demonstrates a simple tweened animation (android.view.animation.Animation).

2. Push

Demonstrates a variety of transformations (android.view.animation.Animation), including fading, motion, and rotation.

Controls

1. Theme White

Demonstrates a variety of common form type widgets, such as check boxes and radio buttons using the white theme.

2. Theme Dark

Demonstrates a variety of common form type widgets, such as check boxes and radio buttons using the dark theme.

Auto Complete

1. Screen Top

Demonstrates the use of AutoCompleteTextView, an autocomplete dropdown box below a text box, with data taken from an array.

2. Screen Bottom

Demonstrates an autocomplete box above a text box.

3. Scroll

Demonstrates an autocomplete text box in the midst of a vertical list.

4. Contacts

Demonstrates an autocomplete text box that gets its content from a database query.

5. Contacts with Hint

Demonstates an autocomplete text box that understands the * wildcard.

Progress Bar

1. Incremental

Demonstrates large and small rotating progress indicators that can be incremented or decremented in units.

2. Smooth

Demonstrates large and small continuously rotating progress indicators used to indicate a generic "busy" message.

3. Dialogs

Demonstrates a ProgressDialog, a popup dialog that hosts a progress bar. This example demonstrates both determinate and indeterminate progress indicators.

4. In Title Bar

Demonstrates an Activity screen with a progress indicator loaded by setting the WindowPolicy's progress indicator feature.

Focus

1. Vertical

Demonstrates how to block selection of a specific screen element.

2. Horizontal

Demonstrates how to change the order of which screen element is selected when the user presses arrow keys.

3. Circular

Another version of Focus2.

Files

- Animation1.java
- Animation2.java
- AutoComplete1.java
- AutoComplete2.java
- AutoComplete3.java
- AutoComplete4.java
- AutoComplete5.java
- AutoComplete6.java
- Baseline1.java
- Baseline2.java
- Baseline3.java
- Baseline4.java
- Baseline6.java
- Baseline7.java
- BaselineNested1.java

- BaselineNested2.java
- BaselineNested3.java
- Buttons1.java
- ChronometerDemo.java
- Controls1.java
- Controls2.java
- CustomView1.java
- DateWidgets1.java
- DateWidgets2.java
- ExpandableList1.java
- ExpandableList2.java
- ExpandableList3.java
- Focus1.java
- Focus2.java
- Focus3.java
- Gallery1.java
- Gallery2.java
- Grid1.java
- Grid2.java
- ImageButton1.java
- ImageSwitcher1.java
- ImageView1.java
- InternalSelectionFocus.java
- InternalSelectionScroll.java
- InternalSelectionView.java
- LabelView.java
- LayoutAnimation1.java
- LayoutAnimation2.java
- LayoutAnimation3.java
- LayoutAnimation4.java
- LayoutAnimation5.java
- LayoutAnimation6.java
- LayoutAnimation7.java
- LinearLayout1.java
- LinearLayout10.java
- LinearLayout2.java
- LinearLayout3.java
- LinearLayout4.java
- LinearLayout5.java
- LinearLayout6.java
- LinearLayout7.java
- LinearLayout8.java
- LinearLayout9.java
- List1.java
- List10.java
- List11.java

- List12.java
- List13.java
- List14.java
- List2.java
- List3.java
- List4.java
- List5.java
- List6.java
- List7.java
- List8.java
- List9.java
- ProgressBar1.java
- ProgressBar2.java
- ProgressBar3.java
- ProgressBar4.java
- RadioGroup1.java
- RatingBar1.java
- RelativeLayout1.java
- RelativeLayout2.java
- ScrollBar1.java
- ScrollBar2.java
- ScrollBar3.java
- ScrollView1.java
- ScrollView2.java
- SeekBar1.java
- Spinner1.java
- TableLayout1.java
- TableLayout10.java
- TableLayout11.java
- TableLayout12.java
- TableLayout2.java
- TableLayout3.java
- TableLayout4.java
- TableLayout5.java
- TableLayout6.java
- TableLayout7.java
- TableLayout8.java
- TableLayout9.java
- Tabs1.java
- Tabs2.java
- Tabs3.java
- TextSwitcher1.java
- Visibility1.java
- WebView1.java

Graphics

Drawable

ShapeDrawable

Demonstrates creating Drawables in XML.

OpenGL|ES

CameraPreview

Demonstrates capturing the image stream from the camera, drawing to a surface (extending SurfaceView) on a separate thread (extending Thread).

GL SurfaceView

Demonstrates how to perform OpenGL rendering in to a SurfaceView.

Code:
GLSurfaceViewActivity.java

Layout:
hello_world.xml

PolyToPoly

Demonstrates calling the {@link android.graphics.Matrix.html#setPolyToPoly(float[],int,float[],int,int)} method to translate coordinates on a canvas to a new perspective (used to simulate perspective).

DrawPoints

Demonstrates using the {@link android.graphics.Paint} and {@link android.graphics.Canvas} objects to draw random points on the screen, with different colors and strokes.

PathEffects

Demonstrates the use of {@link android.graphics.Path} and various {@link android.graphics.PathEffect} subclasses.

SurfaceView Overlay

Shows how you can place overlays on top of a SurfaceView.

Code:
SurfaceViewOverlay.java, GLSurfaceView.java

Layout:
surface_view_overlay.xml

TouchPaint

Demonstrates the handling of touch screen events to implement a simple painting app.

Subdirectories

- kube/
- spritetext/

Files

- AlphaBitmap.java
- AnimateDrawable.java
- AnimateDrawables.java
- Arcs.java
- BitmapDecode.java
- BitmapMesh.java
- BitmapPixels.java
- CameraPreview.java
- Clipping.java
- ColorMatrixSample.java
- ColorPickerDialog.java
- Compass.java
- CreateBitmap.java
- Cube.java
- CubeRenderer.java
- DrawPoints.java
- FingerPaint.java
- GLSurfaceViewActivity.java
- GradientDrawable1.java
- GraphicsActivity.java
- Layers.java
- MeasureText.java
- PathEffects.java
- PathFillTypes.java
- Patterns.java
- PictureLayout.java
- Pictures.java
- PolyToPoly.java
- ProxyDrawable.java
- Regions.java
- RoundRects.java
- ScaleToFit.java
- SensorTest.java
- ShapeDrawable1.java
- SurfaceViewOverlay.java
- Sweep.java
- TextAlign.java
- TouchPaint.java
- TouchRotateActivity.java
- TranslucentGLSurfaceViewActivity.java
- TriangleActivity.java
- TriangleRenderer.java
- Typefaces.java
- UnicodeChart.java

- Vertices.java
- Xfermodes.java

Text

Files

- Link.java
- LogTextBox.java
- LogTextBox1.java
- Marquee.java