**Android学习手记**

ZNing in Neuedu, Qingdao

2016-01-18

目录

[Day 001: Android环境配置与初步接触 1](#_Toc441255251)

[一、Android环境配置 1](#_Toc441255252)

[Step 1: 卸载旧/已安装版本JDK，并删除原有环境变量（非必需） 1](#_Toc441255253)

[Step 2: 根据操作系统情况，安装需要使用的版本JDK 2](#_Toc441255254)

[Step 3: 配置eclipse，为eclipse添加ADT插件： 4](#_Toc441255255)

[Step 4: 下载或离线安装SDK并更新，配置SDK 5](#_Toc441255256)

[Step 5: 使用Intel加速驱动开启计算机的CPU加速 7](#_Toc441255257)

[Step 6: 模拟器的创建 7](#_Toc441255258)

[二、Android初步接触 8](#_Toc441255259)

[Step 1: 创建Android工程 8](#_Toc441255260)

[Step 2: 运行并展示Hello World程序 9](#_Toc441255261)

[Day 002: Android的Activity、Intent、资源使用（一） 11](#_Toc441255262)

[一、Java面向对象的特性 11](#_Toc441255263)

[二、什么是Activity 12](#_Toc441255264)

[三、Activity的生命周期 13](#_Toc441255265)

[四、Intent在不同Activity之间实现跳转 16](#_Toc441255266)

[1、创建Activity并添加事件 16](#_Toc441255267)

[2、Intent在不同Activity之间实现跳转（待补充） 19](#_Toc441255268)

[3、打开新的Activity，传递参数（待补充） 19](#_Toc441255269)

[Day 003: Activity、Intent资源使用（二）与Android UI组件初识 20](#_Toc441255270)

[一、Intent在不同Activity之间实现跳转（续） 20](#_Toc441255271)

[4、返回数据 20](#_Toc441255272)

[二、资源创建与使用 20](#_Toc441255273)

[三、了解各种用户界面的控件的使用方法 22](#_Toc441255274)

[多知一点：关于Java/Android多线程的一些知识 24](#_Toc441255275)

[Day 004: Android UI组件（一）：用户界面基础 27](#_Toc441255276)

[一、CheckBox和RadioButton 27](#_Toc441255277)

[二、Progress进度条（ProgressBar） 27](#_Toc441255278)

[三、Spinner下拉列表 29](#_Toc441255279)

[四、Dialog对话框（AlertDialog） 30](#_Toc441255280)

[多知一点：Android相关小技术内容小结（一） 36](#_Toc441255281)

[1、给控件添加事件的第二种方式（最常用） 36](#_Toc441255282)

[2、在Activity之中显示Toast（吐司） 38](#_Toc441255283)

[3、在Android/Java之中使用正则表达式 38](#_Toc441255284)

[Day 005: Android UI组件（二）：界面布局、样式、适配器的应用 40](#_Toc441255285)

[一、Android界面布局的特点和使用方法 40](#_Toc441255286)

[1、线性布局 41](#_Toc441255287)

[2、框架布局 43](#_Toc441255288)

[3、表格布局（并不完整，而且GridLayout应是网格要独立整理，待定） 45](#_Toc441255289)

[4、网格布局 46](#_Toc441255290)

[5、相对布局 46](#_Toc441255291)

[二、实现欢迎界面跳转 47](#_Toc441255292)

[三、通过网格布局初识服务层 49](#_Toc441255293)

[四、实现动画效果 49](#_Toc441255294)

[多知一点：Android相关小技术内容小结（二） 51](#_Toc441255295)

[Day 006: Android中SQLite数据库应用初识 53](#_Toc441255296)

[一、SQLite数据库介绍 53](#_Toc441255297)

[二、SQLite数据库在Android的应用 53](#_Toc441255298)

# Day 001: Android环境配置与初步接触

## 一、Android环境配置

这里，我们学习一下在Windows环境下进行Android环境配置的相关内容。本教程所使用示例机器为Windows XP Professional SP3，其他Windows版本配置大同小异。

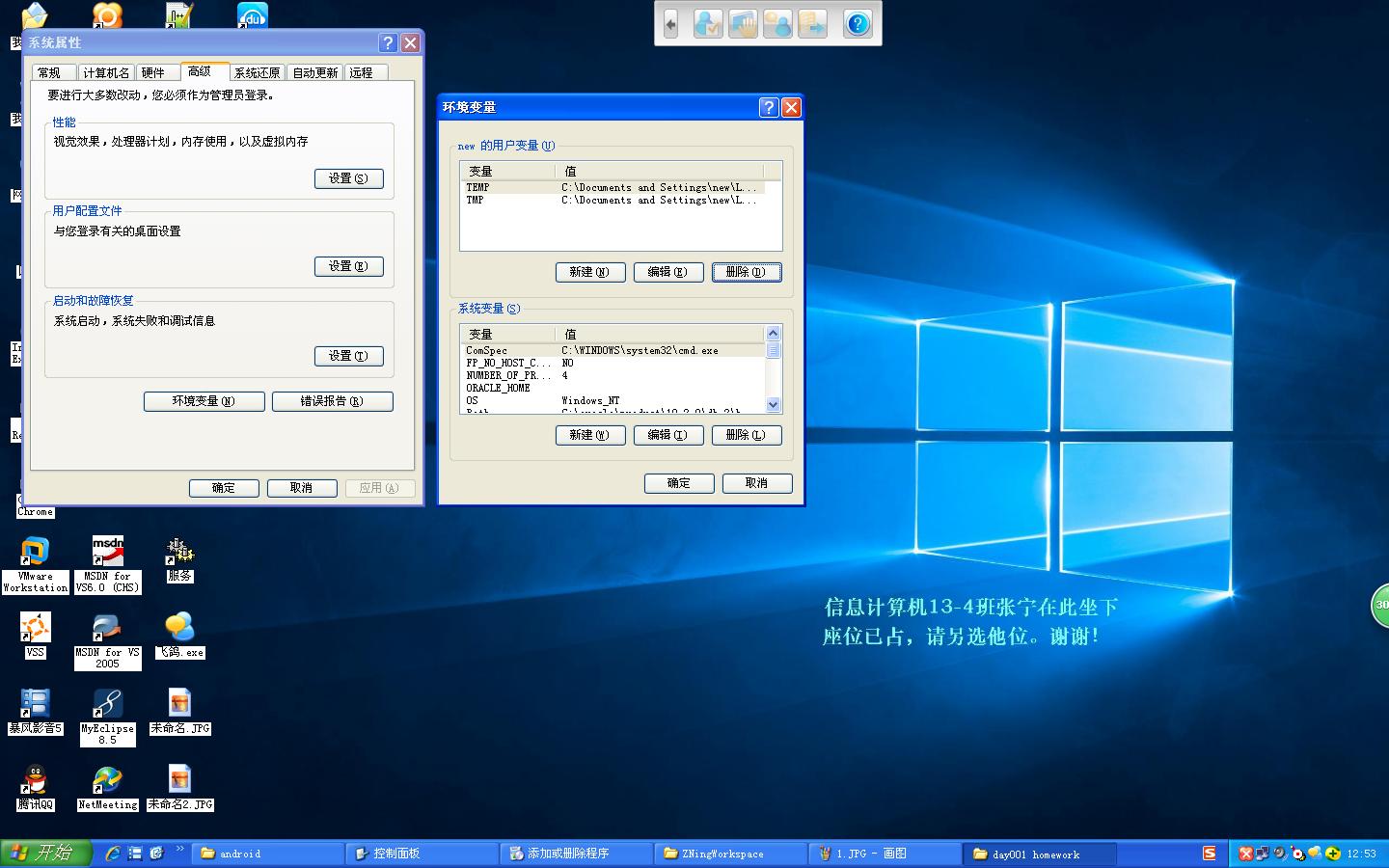
### Step 1: 卸载旧/已安装版本JDK，并删除原有环境变量（非必需）

首先卸载已安装的JDK，可通过“控制面板-添加或删除程序”进行卸载，或者通过第三方软件卸载。此处不再赘述。

卸载完成后，右击“我的电脑”，选择“属性”，打开“系统属性”，选择“高级”-“环境变量”。在打开的对话框中删除JAVA\_HOME及Path中与刚卸载版本JDK相关的配置路径字符串。后连续单击确定即可。



删除前



删除后

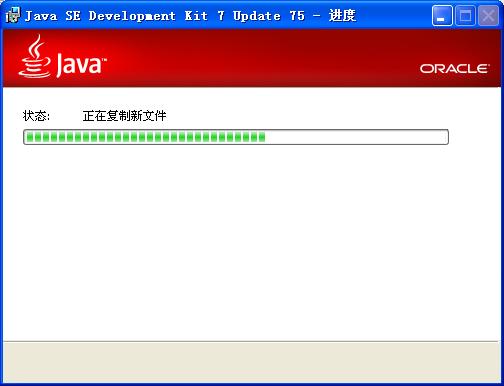
**注意：**如果安装后无法运行eclipse，可能需要重新添加JAVA\_HOME变量及Path、CLASSPATH变量Java环境相关部分。

### Step 2: 根据操作系统情况，安装需要使用的版本JDK

Windows XP是32位系统，因此这里我们选择32位JDK进行安装。这里我们选用的JDK版本是Java 1.7。



JDK（Java 1.7）安装包运行



JDK（Java 1.7）安装中



JDK（Java 1.7）安装完毕

**注意：**安装时请务必不要出现中文路径，建议按默认路径安装。安装JDK期间需要安装JRE（Java Runtime Environment，Java运行环境），安装环境时请务必与JDK同目录下。

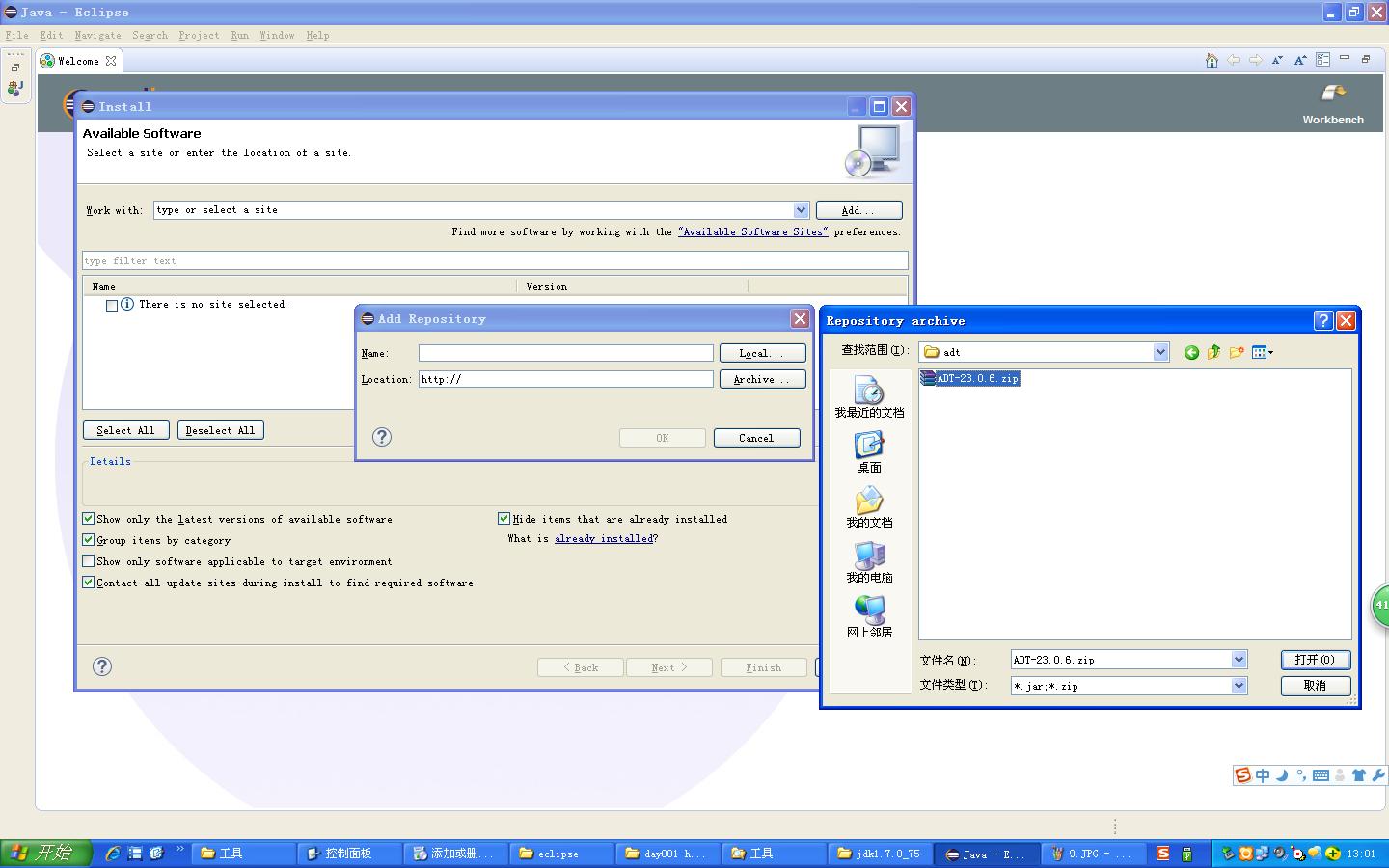


JRE（Java Runtime Environment，Java运行环境）安装

### Step 3: 配置eclipse，为eclipse添加ADT插件：

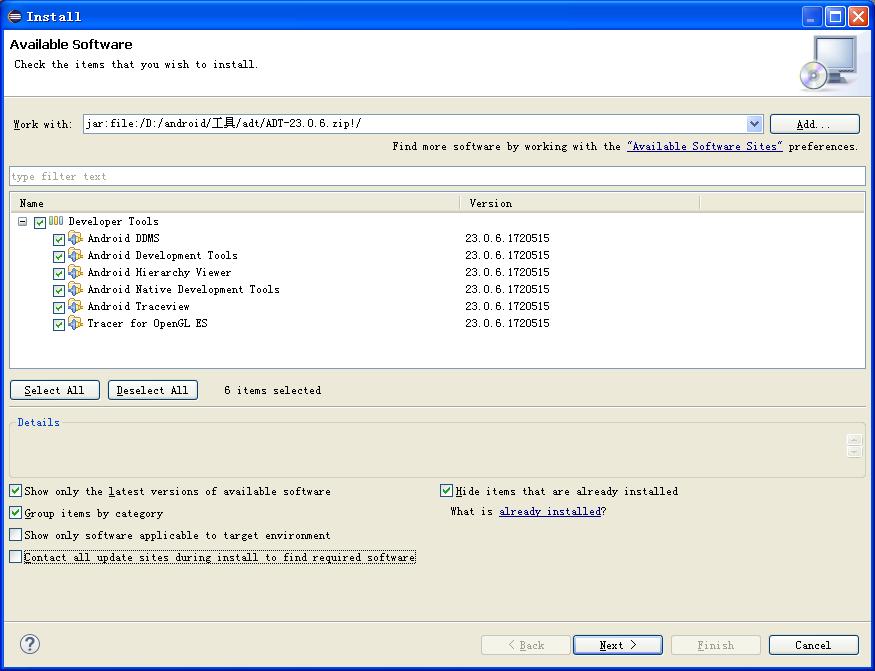
这里我们有多种方法可以进行插件的安装，例如在线安装和离线安装，这里我们演示离线安装方法：

在eclipse工具之中选择Help-Install New Software，在弹出的对话框中，选择“work with”一栏最右边的“Add…”。在弹出的对话框中单击“Archive…”，选择你所下载的ADT插件压缩包，单击OK。

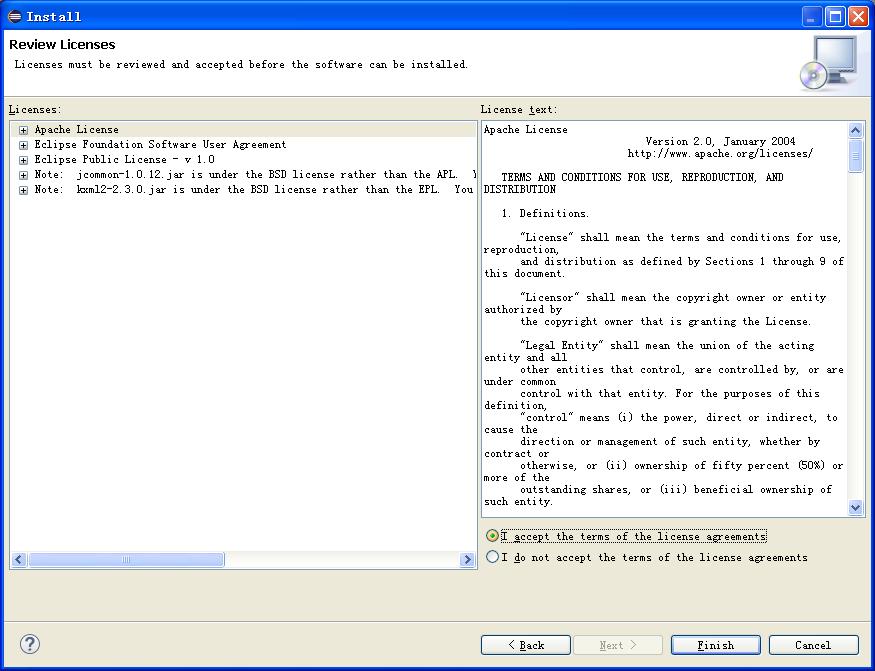


Install New Software安装设置

返回最初的对话框后，取消选中“Contact all update sites during install to find required software”，单击next，下一步选择需要安装的插件内容，默认全选，再下一步同意协议并单击finish完成后，完成插件添加。

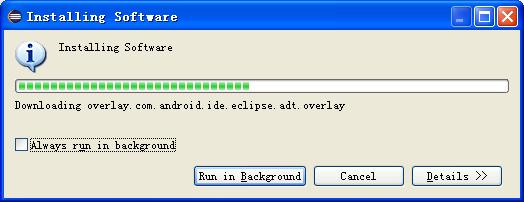


返回界面，需取消选中“Contact all update sites during install to find required software”

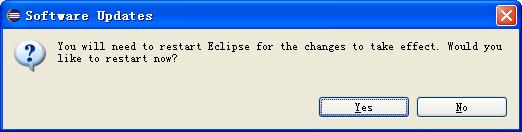


同意协议，单击finish完成

添加完成后需要重启eclipse，按照提示操作即可。



正在加载插件中

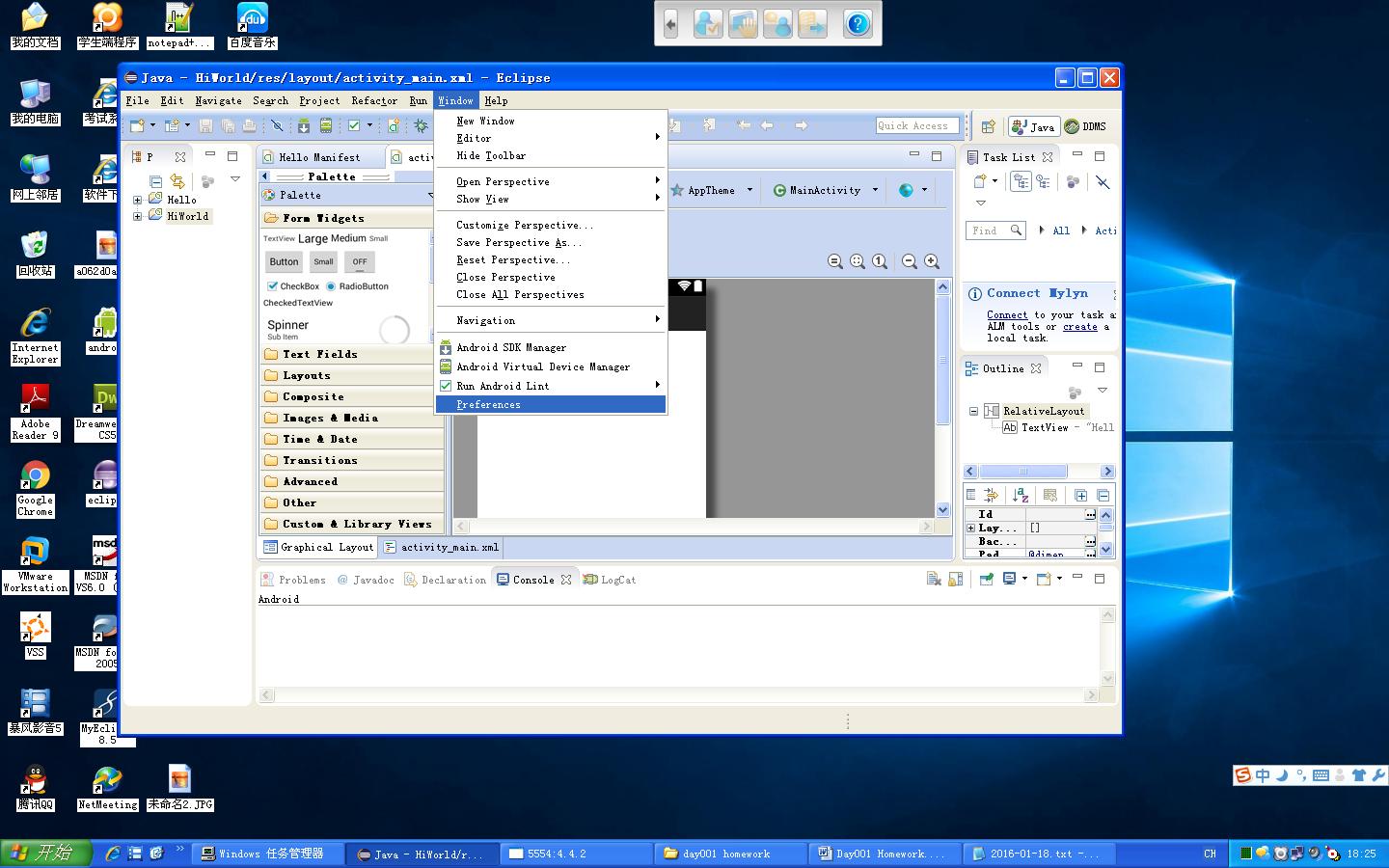


加载插件完成，软件提示重启eclipse，单击Yes自动重启软件

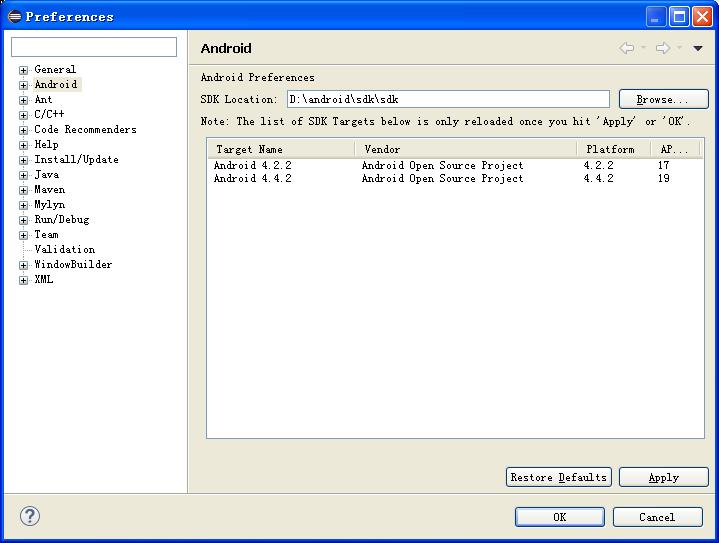
### Step 4: 下载或离线安装SDK并更新，配置SDK

解压SDK文件包放置在合适目录，等待使用即可。

随后在eclipse设置Android SDK环境，在菜单栏选择“Window-Perferences”，在打开的对话框中选择左侧“Android”之后再右侧“SDK Location”一栏选择“Browse…”，在打开的对话框选择SDK所在目录。后单击下面的“Apply”按钮后，单击“OK”即可设置完成。



设置Android SDK环境，在菜单栏选择“Window-Perferences”



选择SDK所在目录后单击下面的“Apply”、“OK”即可设置完成

**Tips: SDK文件夹相关解释与内容**

add-one 目录下的是Google提供地图开发的库函数，支持基于Google Map 的地图开发

add-ons：这里面保存着附加库，比如Google Maps，当然你如果安装了Ophone SDK，这里也会有一些类库在里面。

docs目录下的是Android SDK的帮助文档，通过根目录下的documentation.html文件启动

platforms目录中是不同版本的Android SDK 的库函数、外观样式、程序示例和辅助工具等

tools目录下的是通用的Android开发和调试工具

usb\_driver目录下保了用于amd64和x86平台的USB驱动程序

RELEASE\_NOTES.html是Android SDK的发布说明

market\_licensing：作为Android Market版权保护组件，一般发布付费应用到电子市场可以用它来反盗版

### Step 5: 使用Intel加速驱动开启计算机的CPU加速

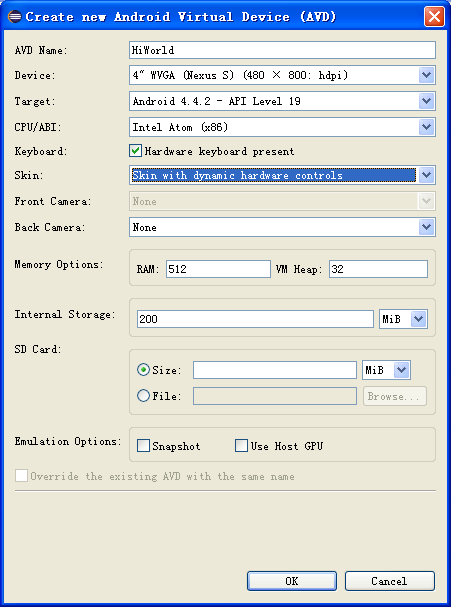
这里，为了更快的使用Android SDK模拟器模拟的安卓设备系统，我们在这里安装Intel (R) Graphics Media Accelerator Driver进行硬件加速。这里软件下载后默认安装即可，此处不再赘述。

**注意：**如果非Intel的CPU，我们可以使用第三方的模拟器，例如BlueStacks等。（BlueStacks安装需要.NET Framework 4.0基本环境支持，请先预装环境。）

### Step 6: 模拟器的创建

启动“Android Virtual Device (AVD) Manager”，单击“Create”按钮，根据本机硬件情况进行相关设置即可。这里我们机子内设置的虚拟设备参数如下图，如有需要可按照下述设置进行虚拟设备的创建。

注意：这里如果尚未安装上一步所提到的“Intel (R) Graphics Media Accelerator Driver” 或者其他进行硬件加速的相关软件的话，在“CPU/ABI”中不要选择“Intel Atom (x86)”。



创建虚拟设备的设置

至此，Android环境配置已基本结束。

## 二、Android初步接触

### Step 1: 创建Android工程

可以创建新Android工程，也可利用已有代码创建Android工程。缺省为创建新Android工程；打开eclipse，选择“File-New-Project”，在打开的对话框中选择“Android Application Project”。

可以使用默认位置存储，也可取消复选框，选择其他位置保存，缺省为使用默认位置D:/Android/workplace/。使用默认位置存储,这里保存位置在： D:/Android/workplace/HelloAndroid

**Tips: Android工程文件夹内相关解释与内容**

src：保存的是所有.Java文件

gen：保存的是编译器自动编译生成的文件

R.Java(Resource，源)：保存的是我们页面中需要的图片，尺寸，颜色，控件的编号。我们开发的时候通过R 来获取相应的资源。

R.dimen.XXX：用于获取dimens.xml文件中的尺寸

R.string.XXX：用于获取strings.xml文件的数据，保存页面固定显示的文本

R.id.XXX：对应的是页面中显示控件的id

R.drawable.XXX：用户查找res下所有以drawable开始的目录中的图片

res：用于保存app中显示的所有资源数据

drawable-XXX：用于保存图片通常会在res下创建drawable目录，存放所有图片

layout：布局，所有页面显示的控件

menu：菜单

values：所有显示的数据

strings.xml：显示的文本

dimens.xml：显示的尺寸

colors.xml：色彩(是一个16进制的数据) e.g. #rrggbb #aarrggbb

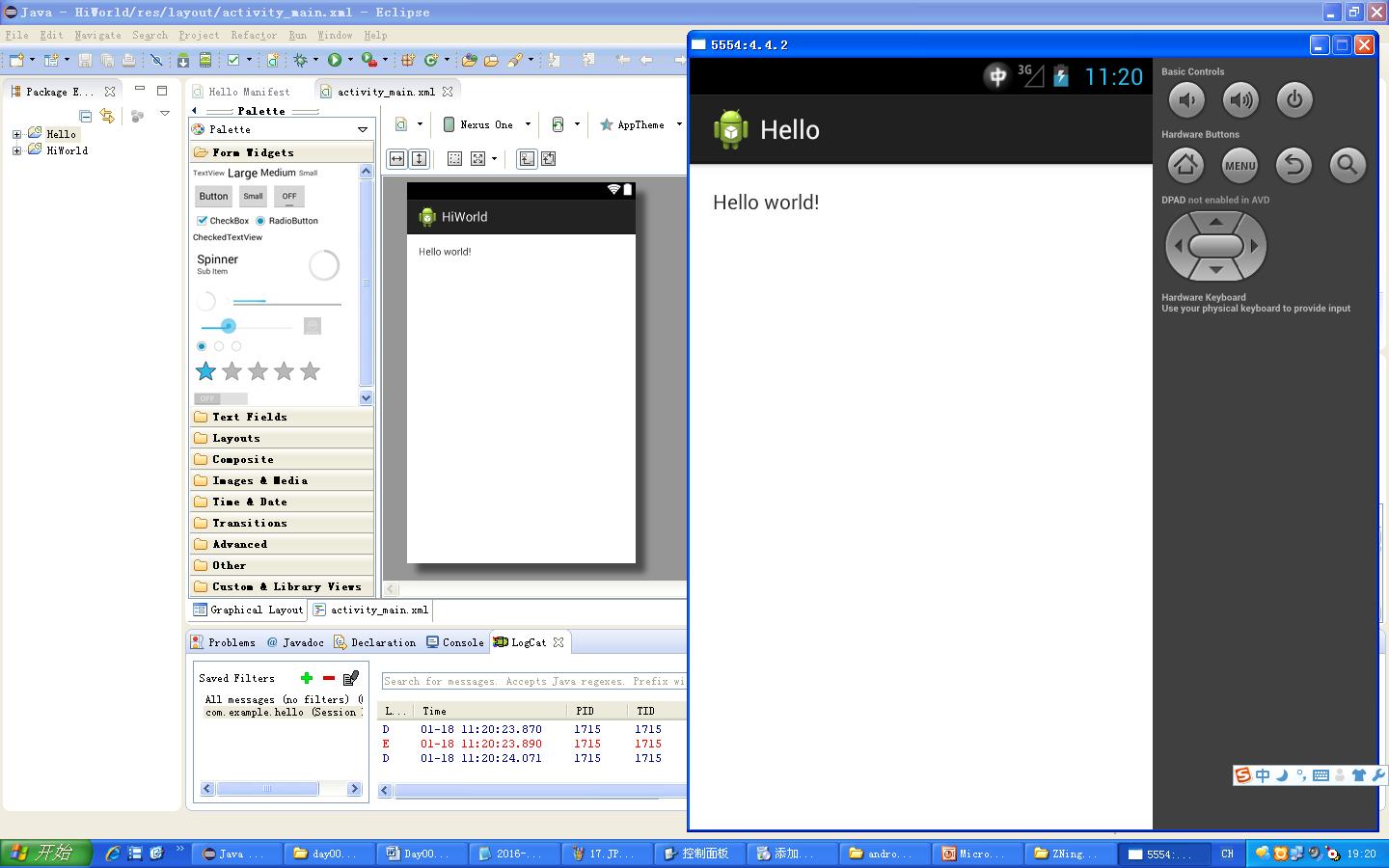
styles.xml：存放样式(用于美化页面)

AndroidManifest.xml：android的清单文件（\*）

**注意：**Android中所有图片等资源命名必须符合Java的命名规范必须以英文字母开始，可以包含数字字母下划线

### Step 2: 运行并展示Hello World程序

这里，因为建立好的程序就是一个Hello World程序。建立好工程后，我们运行即可。首先在AVD中启动模拟器，后在eclipse进行编译，编译成功则在模拟器内显示已建立的Android Hello World程序（如图）：



已建立的Android Hello World工程

# Day 002: Android的Activity、Intent、资源使用（一）

## 一、Java面向对象的特性

面向对象主要有四大特性：封装、抽象、继承和多态。

**封装：**在面向对象语言中，封装特性是由类来体现的，我们将现实生活中的一类实体定义成类，其中包括属性和行为（在Java中就是方法），就好像人类，可以具有name,sex,age等属性，同时也具有eat(),sleep()等行为，我们在行为中实现一定的功能，也可操作属性，这是面向对象的封装特性。

**抽象：**抽象就是将一类实体的共同特性抽象出来，封装在一个抽象类中，所以抽象在面向对象语言是由抽象类来体现的。比如鸟就是一个抽象实体，因为抽象实体并不是一个真正的对象，它的属性还不能完全描述一个对象，所以在语言中体现为抽象类不能实例化。

抽象是忽略一个主题中与当前目标无关的方面，把现实世界中的概念转换为对象,可以是表示数据的VO对象或抽象类，也可以是表示一类行为的接口。有利于从众多变化的状态中抽离出不变的东西。有利于使用继承来构造类的体系，实现多态，使用模板和工厂模式方法来实现对业务过程的抽象。抽象是概念建模和业务流程建模很重要的工具，是面向对象体系的基石，而不是一堆杂乱、混乱、重复、散沙般的、关系错综复杂的类。

**继承：**继承就像是我们现实生活中的父子关系，儿子可以遗传父亲的一些特性，在面向对象语言中，就是一个类可以继承另一个类的一些特性，从而可以代码重用，其实继承体现的是is-a关系，父类同子类在本质上还是一类实体。

继承是一种连结类的层次模型，并且允许和鼓励类的重用，提供了一种明确表示共性的方法。 继承意味着能声明一个类型，而且能继承父类中的所有操作和数据，同事还有机会声明自己的新方法以改进、具体化、代替或者扩展父类中的方法。使用父类为应用领域建立层次模型，提供代码重用并允许定制。

如果需要进行防止继承，我们可以利用final声明类或方法，意味不能被修改。当一个类被声明为final时，只有方法自动被设为final,而字段不会。final有安全、高效等优点。而对于类型转换，Java要求必须在继承体系内进行，并且转换之前应先使用instanceof进行检验。super关键词不是一个对对象的引用，而是指向编译器调用超类方法或调用超类的构造器的超类的专用关键字。

**多态：**多态就是通过传递给父类对象引用不同的子类对象从而表现出不同的行为。一个对象变量可以指向多种实际类型的现象。

## 二、什么是Activity

在认识Activity之前，我们需要认识Android系统中的四大重要组件。在Android系统中，组件是可以调用的基本功能模块，Android应用程序就是由组件组成的。

Android系统有四个重要的组件，分别是Activity、Service、BroadcaseReceiver和ContentProvider。

**Activity：**

Activity是Android程序的呈现层，显示可视化的用户界面，并接收与用户交互所产生的界面事件。Android应用程序可以包含一个或多个Activity，一般在程序启动后会呈现一个Activity，用于提示用户程序已经正常启动。在界面上的表现形式：全屏窗体，非全屏悬浮窗体，对话框。

**Service：**

Service用于没有用户界面，但需要长时间在后台运行的应用

**ContentProvider：**

ContentProvider是Android系统提供的一种标准的共享数据的机制，应用程序可以通过ContentProvider访问其他应用程序的私有数据。私有数据可以是存储在文件系统中的文件，也可以是SQLite中的数据库。Android系统内部也提供一些内置的ContentProvider，能够为应用程序提供重要的数据信息。

**BroadcaseReceiver：**

BroadcaseReceiver是用来接受并响应广播消息的组件，不包含任何用户界面。可以通过启动Activity或者Notification通知用户接收到重要信息。Notification能够通过多种方法提示用户，包括闪动背景灯、震动设备、发出声音或在状态栏上放置一个持久的图标。

所有Android组件都具有自己的生命周期，是从组件建立到组件销毁的整个过程，在生命周期中，组件会在可见、不可见、活动、非活动等状态中不断变化。

## 三、Activity的生命周期

Activity生命周期指Activity从启动到销毁的过程。其中，Activity表现为四种状态，分别是活动状态、暂停状态、停止状态和非活动状态：

**活动状态：**Activity在用户界面中处于最上层，完全能被用户看到，能够与用户进行交互。

**暂停状态：**Activity在界面上被部分遮挡，该Activity不再处于用户界面的最上层，且不能够与用户进行交互。

**停止状态：**Activity在界面上完全不能被用户看到，也就是说这个Activity被其他Activity全部遮挡。

**非活动状态：**不在以上三种状态中的Activity则处于非活动状态。

需要说明的是，Activity生命周期的几种状态之间可以互相转换。



Activity的四种状态的变换关系图

此外，Activity在Android中的调用形式是以Activity栈来进行活动情况判断的。即Activity的调用遵循“后进先出”的规则。

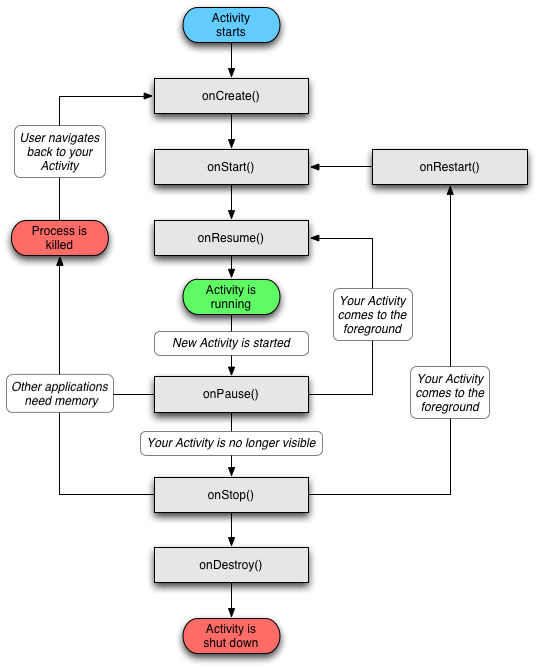


Activity栈示意图

Activity的构造方法是负责创建Activity的一个实例，一般省略，这里概不赘述。而Activity中对于用户操作产生的事件都有明确的方法，这里称为事件的回调函数。回调函数在Activity方法内主要是以下形式的存在：

1. public class MyActivity extends Activity {
2. //负责把布局文件加载到Java中，必需重写（覆盖），生命周期中仅调用一次
3. protected void onCreate(Bundle savedInstanceState);
4. //在Activity呈献给用户的时候调用，生命周期中仅调用一次
5. protected void onStart();
6. //重新开始Activity，生命周期中仅调用一次
7. protected void onRestart();
8. //Activity整个呈现给用户的时候调用的方法，表示Activity已向用户展示
9. protected void onResume();
10. //暂停Activity
11. protected void onPause();
12. //停止Activity，生命周期中仅调用一次
13. protected void onStop();
14. //销毁Activity，生命周期中仅调用一次
15. protected void onDestroy();
16. }

其中，MyActivity即为你所定义的Activity名称。这里我们可以看到，一个Activity的基本的事件回调函数有onCreate()、onStart()、onRestart()、onResume()、onPause()、onStop()、onDestroy()。这些事件的回调函数在一个Android应用中的流程图如下：



事件的回调函数在Android应用中流程图



事件的回调函数在Android应用Activity的生命周期划分

例如，现有A、B两个Activity。由A调用B，其中B以对话框的方式出现，则事件的回调函数调用流程是：onCreate->onStart->onResume->B出现->调用A的onPause->B关闭->调用A的onResume。

若A调用B，其中B以完全覆盖A的方式出现，则事件的回调函数调用流程是：onCreate->onStart->onResume->B出现->调用A的onPause->调用A的onStop-> B关闭->调用A的onRestart->调用A的onStart->调用A的onResume。

再举一个例子，若在A出现的时候，需要读取文本文件中的内容显示给用户，读取文本文件的代码应该在onCreate、onStart、onResume。特别注意的是，onResume推荐适用于经常修改的文本文件的读取代码的放置，因为这样，Activity成为焦点的时候每一次都会执行，如果放置不经常修改的文件，会造成资源浪费。对于不经常修改的文件建议放在onCreate或前两者。

**注意：**在Android开发中，如果使用System.out.println之类的输出方法的话，不会在Console控制台输出，所有内容都在LogCat下输出。

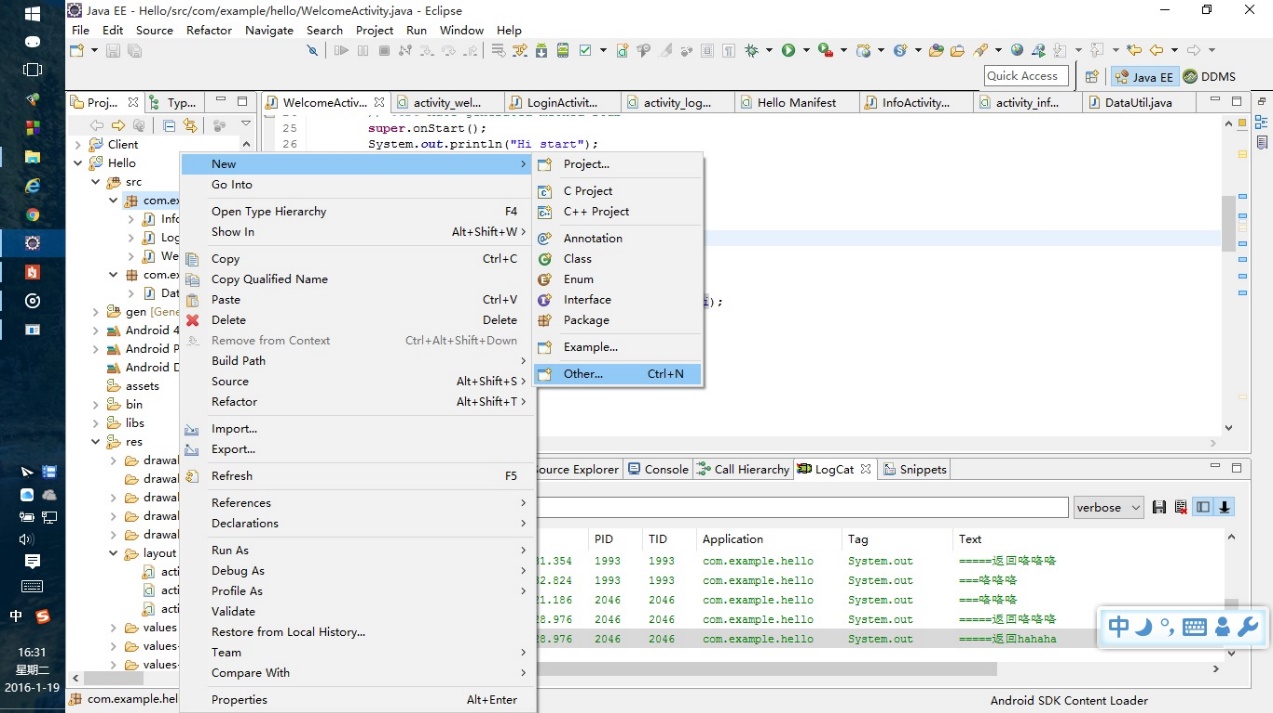
## 四、Intent在不同Activity之间实现跳转

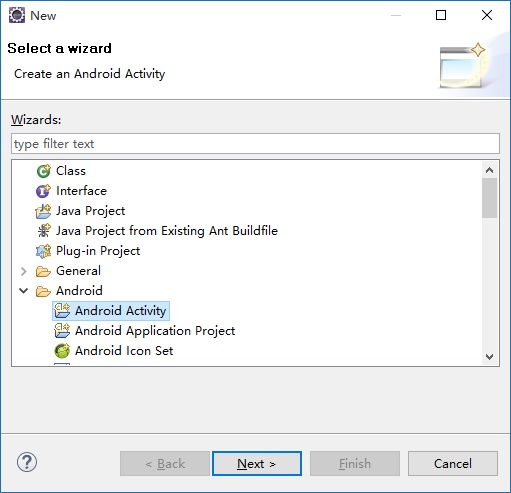
### 1、创建Activity并添加事件

首先，我们需要学习一下如何创建Activity。这里有两种方法可以创建Activity。手动创建和New自动创建。

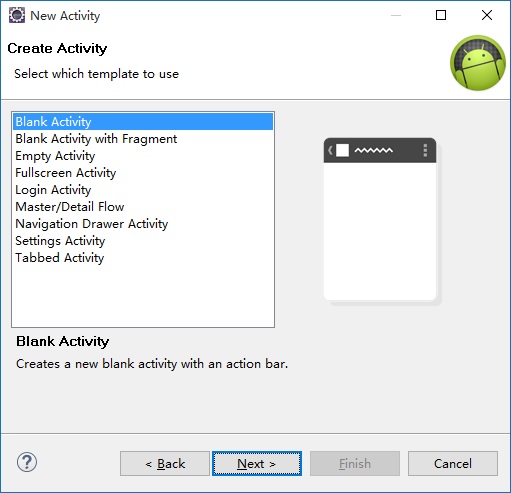
手动创建主要分为“创建布局文件”、“创建类继承Activity”、“重写onCreate()及SetContextView(R.layout.XXX)方法”、“在AndroidManifest文件中配置Activity”等四步。由于比较繁琐这里不再赘述。

New自动创建是通过Android SDK开发已给的模板进行创建即可。在需要创建Activity的包右击，选择“New-Other-Android-Android Activity”后单击“Next >”进行下一步操作，一般我们默认设置的是Empty Activity进行自我排版。单击“Next >”会进行Activity的名称设置。再单击“Next >”将展示需要修改的文件的具体情况，如没有问题可直接单击“Finish”结束即可。

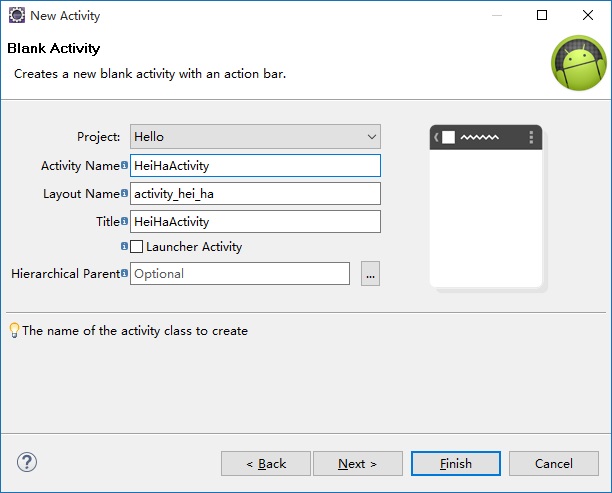




New自动创建选择“New-Other-Android-Android Activity”



设置Empty Activity进行自我排版



进行Activity的名称设置

在布局文件中，给Button控件添加点击事件添加如下代码，即可更改显示效果：android:theme="@android:style/Theme.Dialog"

**Tips: 在改布局文件对应的Java中创建方法，格式固定：**

doXXX：表示用于执行的方法

toXXX：表示用于跳转的方法

public void 方法名(View view){

}

以上方法是给控件添加事件的一种方式。

### 2、Intent在不同Activity之间实现跳转（待补充）

Intent (意图) ，Intent提供了一种通用的消息系统，它允许在你的应用程序与其它的应用程序间传递Intent来执行动作和产生事件，Intent是对一次将要进行的操作的抽象描述。我们使用Intent启动一个Activity、Service、Boradcast等，并进行数据的传递。Intent可以划分成显式意图和隐式意图：

**显式意图：**调用Intent.setComponent()或Intent.setClass()方法指定了组件名或类对象的Intent为显式意图，显式意图明确指定了Intent应该传递给哪个组件。即调用应用本身内部的相关操作。

**隐式意图：**Android系统会根据隐式意图中设置的动作(action)、类别(category)、数据（URI和数据类型）找到最合适的组件来处理这个意图。即调用系统的相关操作。

Intent Filter（过滤器）其实就是用来匹配隐式Intent的，如果Intent Filter定义的动作、类别、数据（URI和数据类型）与Intent匹配，就会使用Intent Filter所在的组件来处理该Intent。想要接收使用startActivity()方法传递的隐式意图的活动必须在它们的意图过滤器中包含"android.intent.category.DEFAULT"。

由于显示意图即在应用内跳转Activity，即

1. Intent intent = new Intent(本Activity.this,跳转目标的.class);

在Activity跳转的时候，可以通过finish();方法关闭上一个Activity。

### 3、打开新的Activity，传递参数（待补充）

Android中操作控件三部曲：

1、声明该控件类型的对象

2、通过FindViewById给对象赋值

3、通过对象中的方法，操作该控件

setText(“ddd”); 用于给控件设置显示的文本

getText().toString(); 获取控件显示的文本

# Day 003: Activity、Intent资源使用（二）与Android UI组件初识

## 一、Intent在不同Activity之间实现跳转（续）

### 4、返回数据

首先，我们通过startActivityForResult(Intent intent, int requestCode)启动，这里requestCode是用于该返回的结果是哪个请求，区分发送请求的对象。紧接着，使用startActivityForResult(Intent intent, int requestCode)方法打开新的Activity，新Activity关闭前需要向前面的Activity返回数据需要使用系统提供的setResult(int resultCode, Intent data)方法实现，通过setResult(resultCode, intent);返回结果，其中resultCode表示返回码，用于判断是否正确返回。这个可以判断是通过应用按钮返回还是系统硬件返回。而intent中封装的是返回的数据。

**注意：**当前Activity必须finish();

最后，显示返回的数据。通过重写onActivityResult(int requestCode, int resultCode, Intent data)方法，处理用户返回的数据。

**Tips: 在Java中编程使用常量代替固定的值可使用**，例如：

public static final int CITY\_REQUEST\_CODE = 666;

且常量通常必须写在接口中。

**Tips：把不同功能的代码，放在不同的package中。**例如：util一般都是工具包、view(activity)一般是视图包存放界面。

## 二、资源创建与使用

所有应用程序都有代码指令和数据构成。Andriod工程中，资源文件和Java类文件时分开存储的。大多数常见的资源类型被存储于XML中。

**Tips：Android资源目录的层级结构：**所有资源存放在res目录中

/res/drawable-\*/ 图形资源文件

/res/layout/ 用户界面资源

/res/values/ 简单数据，字符串字符值等。

/res/values/strings.xml 字符串

/res/values/colors.xml 颜色

/res/values/dimens.xml 尺寸

/res/drawable/ 图像 .png .jpg 或xml定义的图形

/res/anim/xxxxx.xml 补间动画 <set> <apha><scale>…

/res/drawabe/xxx.xml 逐帧动画 <animation-list> <item>

/res/menu/helpmenu.xml 菜单

/res/xml/data.xml xml文件

/res/raw 原始文件 xx.mp3 xx.mp4 xx.txt

/res/layout 布局

/res/values 样式和主题 <style>

在布局文件中需要通过@string/XXX、@color/XXX、@color/XXX、@dimen/XXX调用，在Java类中需要通过getResource().getString(R.string.XXX)、getResource().getColor(R.color.XXX)、getResource().getDimension (R.dimen.XXX)调用，通过@android:drawable/图片名可以访问使用Android系统自带的图片资源。

**使用string字符串资源**

1. <resources>
2. <string name="app\_name">HelloAndroid</string>
3. </resources>

**在应用程序访问字符串资源**

1. String app\_name=getResources().getString(R.string.app\_name);

**使用string字符串数组资源:**

1. <resources>
2. <string-array name="flavors">
3. <item>Java</item>
4. <item>android</item>
5. </string-array>
6. </resources>

**在应用程序访问字符串资源**

1. String flavors[] = getResources().getStringArray(R.array.flavors);

**使用颜色**

Android可以存储RGB颜色值,RGB颜色值以#开头，还可以给出alpha值以控制透明度.

#RGB (#F00 12位颜色 红色)

#ARGB (#F00 12位颜色 红色 alpha为50%红色)

#RRGGBB (#FF00FF 24位颜色 羊红色)

#AARRGGBB (#80FF00FF 12位颜色 红色 alpha为50%羊红色)

**使用尺寸**

Android支持下列所有单位：

px（像素）：屏幕上的点。in（英寸）：长度单位。

mm（毫米）：长度单位。pt（点）：1/72英寸。

dp（与密度无关的像素）：一种基于屏幕密度的抽象单位。在每英寸160点的显示器上，1dp=1px。

dip：与dp相同。

sp（与刻度无关的像素）：与dp类似，但是可以根据用户的字体大小 首选项进行绽放。

为了使用界面能够在现在和将来的显示器类型上正常显示，一般建议始终使用sp作为文字大小的单位，将dip/dp作为其它元素的单位 。另外也可以考虑使用矢量图，而不是位图。

**Tips: px和dp的区别**

如果屏幕密度为160，这时dp、sp和px是一样的。1dp=1sp=1px，但当使用px单位的时候，如果屏幕大小不变（假设还是3.2英寸），而屏幕密度变成了320。那么假如原来控件的宽度设成160px，这时候就会发现，该控件在320密度的屏幕下短了一半。但如果设置成160dp或160sp的话。系统会自动将width属性值设置成320px。

1. <dimen name="fourtpt">10pt</dimen>
2. float myDimen = getResources().getDimension(R.dimen.fourtpt);

**注意：**所有空间尺寸建议采用dp为单位，所有XML固定值都必须写在相应的文件中。

## 三、了解各种用户界面的控件的使用方法

UI控件主要由系统控件与自定义控件。其中系统控件介绍如下：

**Android用户界面框架中的界面元素以一种树型结构组织在一起，称为视图树。**Android系统会依据视图树的结构从上至下绘制每一个界面元素。每个元素负责对自身的绘制，如果元素包含子元素，该元素会通知其下所有子元素进行绘制。



Android用户界面框架（Android UI Framework）采用视图树（View Tree）模型

**视图树由View和ViewGroup构成。**View是界面的最基本的可视单元，存储了屏幕上特定矩形区域内所显示内容的数据结构，并能够实现所占据区域的界面绘制、焦点变化、用户输入和界面事件处理等功能。View也是一个重要的基类，所有在界面上的可见元素都是View的子类。ViewGroup是一种能够承载含多个View的显示单元。ViewGroup功能：一个是承载界面布局，另一个是承载具有原子特性的重构模块。

**单线程用户界面。**在单线程用户界面中，控制器从队列中获取事件和视图在屏幕上绘制用户界面，使用的都是同一个线程。其特点是处理函数具有顺序性，能够降低应用程序的复杂程度，同时也能减低开发的难度。但如果事件处理函数过于复杂，可能会导致用户界面失去响应。因此，在Java进行开发应用的时候，多线程是很好的帮手。

常见的系统控件：TextView、EditView、Button、Menu、RadioGroup、RadioButton、CheckBox、ProgressBar、ListView、TabWidget、SeekBar、ScrollView、GirdView、ImageSwitcher等控件。

## 多知一点：关于Java/Android多线程的一些知识

刚才我们提到，Android UI控件是单线程用户界面，那么单线程作为程序最基本的执行方式会出现由于事件处理函数过于复杂，从而导致用户界面失去响应的情况。因此，这里我们需要引入多线程来进行问题的解决。

关于线程的概念：线程是进程内部的一个执行单元。系统创建好进程后，实际上就启动执行了该进程的主线程，主线程以函数地址形式将程序的启动点提供给操作系统。主线程终止了，进程也就随之终止。多线程可以实现并行处理，避免了某项任务长时间占用CPU时间。

在Java/Android开发中，多线程的使用比例非常多。而线程的撰写和使用也跟MFC大同小异但更加方便。多线程使用主要有两个关键的问题需要解决：

1）线程的创建与调用

2）线程与线程之间的通讯

这里，前者Java已经有相关的封装类为我们做好了这一点，而对于后者，Android通过消息通信机制让线程与线程之间的通信成为可能。以下是创建线程的代码示例（一个倒计时的小应用），相关包引用未列出：

1. //创建一个线程的对象，并指定任务
2. //匿名类
3. Thread t=new Thread(new Runnable(){
4. @Override
5. public void run() {
6. for(int i=10;i>0;i--){
7. //不允许在子线程中访问主线程的内容
8. //我们需要通过消息机制实现线程之间的通信
9. //1、创建的消息的句柄handler，实现消息的发送与处理
10. //2、在子线程中通过handler（Java内的类）。sendMessage向主线程中发消息。
11. //textMsg.setText(String.valueOf(i));
12. System.out.println("==="+i);
13. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
14. //创建消息并发送
15. Message msg = new Message();
16. msg.what = 1;//设置消息的类型，将来用于区分各种消息
17. msg.obj=i;//设置传递的数据
18. handler.sendMessage(msg);
19. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
20. try {
21. Thread.sleep(1000);
22. } catch (InterruptedException e) {
23. // TODO Auto-generated catch block
24. e.printStackTrace();
25. }
26. }
27. Message msg = new Message();
28. msg.what = 2;//设置消息的类型，将来用于区分各种消息
29. handler.sendMessage(msg);
30. }
31. });
32. //开启线程
33. t.start();

上述代码中，Thread.sleep(1000);部分（语句功能：控制CPU执行时间为1秒一次）由于是线程控制，会抛出异常，因此在Eclipse提示下转变为如下代码：（try-catch语句）

1. try {
2. Thread.sleep(1000);
3. } catch (InterruptedException e) {
4. // TODO Auto-generated catch block
5. e.printStackTrace();
6. }

这里，我们可以总结出Java中创建并调用线程的代码框架（即一般方法）：

1. Thread t=new Thread(new Runnable(){
2. @Override
3. public void run() {
4. // TODO Auto-generated method stub
5. //在此添加执行的代码即可
6. }
7. });
8. //开启线程
9. t.start();

线程创建好后，正常情况下，线程之间是无法进行操作的，也不允许在子线程中访问主线程的内容。这里我们就需要创建相关方法进行线程消息通信。这里，Android系统中的android.os.Handler包中给我们带来了一个方便实用的方法，我们称其为句柄（或语柄）。相关调用示例如下：

线程内的消息创建与发送：（详见本附录第一页第一块代码）

1. //创建消息并发送
2. Message msg = new Message();
3. msg.what = 1;//设置消息的类型，将来用于区分各种消息
4. msg.obj=i;//设置传递的数据
5. handler.sendMessage(msg);

这里，通过创建Message对象，并设置msg.what消息类型和msg.obj传递的参数，来对Message对象进行创建（即创建消息）。之后，在Activity继承类之中，我们需要创建handler类型的对象，并覆盖处理消息的方法，用于重写handleMessage方法，处理发送过来的消息。具体示例代码如下：（相关包引用未列出，主要引用android.os.Handler包内容）

1. //创建handler类型的对象，并覆盖处理消息的方法
2. private Handler handler = new Handler(){
3. //重写handleMessage方法用于处理发送过来的消息
4. public void handleMessage(android.os.Message msg){
5. switch (msg.what) {
6. case 1:
7. textMsg.setText(msg.obj.toString());
8. break;
9. case 2:
10. textMsg.setText("Time running out");
11. break;
12. default:
13. break;
14. }
15. };
16. };

这里，我们可以总结出Android中创建Handler对象并进行线程间的消息通信的代码框架（即一般方法）：

1. private Handler handler = new Handler(){
2. public void handleMessage(android.os.Message msg){
3. // TODO Auto-generated method stub
4. //在此添加执行的代码即可
5. }
6. };
7. };

多线程在Android系统中用途广泛，小型应用一般在倒计时、时钟显示、广告轮换或灯箱等显示方面运用频繁。是Android UI控件设计之中的非常令人心悦的方法。不过由于其涉及代码跨度大，操作较复杂，需要勤加练习才可熟练掌握。

# Day 004: Android UI组件（一）：用户界面基础

## 一、CheckBox和RadioButton

CheckBox是一个同时可以选择多个选项的控件，而RadioButton则是仅可以选择一个选项的控件。RadioGroup是RadioButton的承载体，程序运行时不可见，应用程序中可能包含一个或多个RadioGroup。一个RadioGroup包含多个RadioButton，在每个RadioGroup中，用户仅能够选择其中一个RadioButton。

Checkbox应用：密码的显式和隐式

RadioButton设置点击事件监听器的方法：

1. ck1.setOnClickListener(new CheckBox.OnClickListener() {
2. @Override
3. public void onClick(View v) {
4. if(ck1.isChecked()){
5. System.out.println("ck1 check");
6. }else{
7. System.out.println("ck1 nocheck");
8. }
9. }
10. });

OnCheckedChangeListener方法：

1. ck1.setOnClickListener(new CheckBox.OnClickListener() {
2. @Override
3. public void onClick(View v) {
4. if(ck1.isChecked()){
5. System.out.println("ck1 check");
6. }else{
7. System.out.println("ck1 nocheck");
8. }
9. }
10. });

RadioButton及RadioGroup

Adroid:checked=”true”

## 二、Progress进度条（ProgressBar）

默认是圆形的，一直会旋转的进度条

1. <ProgressBar android:id="@+id/widget43"
2. android:layout\_width="wrap\_content"
3. android:layout\_height="wrap\_content" android:layout\_gravity="center\_vertical">
4. </ProgressBar>

水平进度条

1. <ProgressBar android:id="@+id/ProgressBar02"
2. android:layout\_width="wrap\_content"
3. android:max="100"
4. android:style="?android:attr/progressBarStyleLarge"
5. android:progress="50"
6. android:secondaryProgress="70"
7. android:visibility="gone"
8. android:layout\_height="wrap\_content"></ProgressBar>

**Progress进度条的相关方法：**

getMax()：返回这个进度条的范围的上限

getProgress()：返回进度

getSecondaryProgress()：返回次要进度

setProgress(5); 设置进度

setSecondaryProgress(10);

setVisibility(int v)：设置该进度条是否可视

View.GONE&View.VISIBLE

1. new Thread(){
2. @Override
3. public void run() {
4. while(i%10==0 && i<=100){
5. pb2.setProgress(i+5);
6. pb2.setSecondaryProgress(i+10);
7. i+=10;
8. try {
9. Thread.sleep(500);
10. } catch (InterruptedException e) {
11. // TODO Auto-generated catch block
12. e.printStackTrace();
13. }
14. }
15. }
16. }.start();

**SeekBar拖动条**

1、拖动条的事件：由于拖动条可以被用户控制。所以需要对其进行事件监听，这就需要实现SeekBar.OnSeekBarChangeListener接口。此接口共需要监听三个事件，分别是：数值改变（onProgressChanged）、开始拖动（onStartTrackingTouch）、停止拖动（onStopTrackingTouch）。

2、 拖动条的主要属性和方法：setMax 设置拖动条的数值、setProgress设置拖动条当前的数值、setSeconddaryProgress设置第二拖动条的数值，即当前拖动条推荐的数值。

3、模拟方式实现手动的动作：Progress进度条应用：延迟加载。

1. seek=(SeekBar)findViewById(R.id.SeekBar01);
2. myTextView=(TextView)findViewById(R.id.TextView01);
3. myTextView2=(TextView)findViewById(R.id.TextView02);
4. seek.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {
5. @Override
6. public void onStopTrackingTouch(SeekBar seekBar) {
7. myTextView2.setText("停止调节");
8. }
9. @Override
10. public void onStartTrackingTouch(SeekBar seekBar) {
11. myTextView2.setText("开始调节");
12. }
13. @Override
14. public void onProgressChanged(SeekBar seekBar, int progress,
15. boolean fromUser) {
16. // TODO Auto-generated method stub
17. myTextView.setText("当前值 为:"+progress);}
18. });
19. }
20. }

## 三、Spinner下拉列表

Spinner是一种能够从多个选项中选一选项的控件，类似于桌面程序的组合框（ComboBox），但没有组合框的下拉菜单，而是使用浮动菜单为用户提供选择。

1. <Spinner
2. android:id="@+id/spinner1"
3. android:layout\_width="wrap\_content"
4. android:layout\_height="wrap\_content“>
5. List<String> age=new ArrayList<String>();
6. for(int i=0;i<10;i++){
7. age.add(i+"");
8. }
9. ArrayAdapter<String> adapter=new ArrayAdapter<String>(this,
10. android.R.layout.simple\_spinner\_dropdown\_item, age);
11. spAge.setAdapter(adapter);

## 四、Dialog对话框（AlertDialog）

对话框是Android系统必不可少的控件，我们进场通过它和户进行交互。Android提供多种创建对话框的方法，而AlertDialog是最方便的方法。

对话框有提醒对话框、选择对话框、列表对话框、单选对话框、多选对话框、自定义对话框等各种种类。

**对话框控件的内部方法：**

**AlertDialog的创建需要它的一个内嵌类Builder**

d.setIcon();

d.setMessage();

d.setItems();

d.setSingleChoiceItems();

d.setMultiChoiceItems();

d.setPositiveButton();

d.setNegativeButton();//取消按钮

d.setNeutralButton();//忽略按钮

d.show();

d.dismiss();

**各种对话框的具体实现示例如下：**

提醒对话框：

1. public class MainActivity extends Activity {
2. private Button btnAlert;//提醒对话框
3. @Override
4. protected void onCreate(Bundle savedInstanceState) {
5. super.onCreate(savedInstanceState);
6. setContentView(R.layout.activity\_main);
7. findView();
8. setListener();
9. }
10. }
11. private void setListener() {
12. // TODO Auto-generated method stub
13. //提醒框
14. this.btnAlert.setOnClickListener(new View.OnClickListener() {
15. @Override
16. public void onClick(View v) {
17. // TODO Auto-generated method stub
18. AlertDialog.Builder alert=new AlertDialog.Builder(MainActivity.this);
19. //alert.setCancelable(false);
20. alert.setTitle("警告");
21. alert.setMessage("你好");
22. alert.setPositiveButton("一探究竟", new DialogInterface.OnClickListener() {
23. @Override
24. public void onClick(DialogInterface dialog, int which) {
25. // TODO Auto-generated method stub
26. finish();
27. }
28. });
29. alert.show();
30. }
31. });
32. }
33. private void findView() {
34. // TODO Auto-generated method stub
35. this.btnAlert=(Button) findViewById(R.id.main\_btn\_alert);
36. }
37. }

选择对话框：

1. public class MainActivity extends Activity {
2. private Button btnChoice;//选择对话框
3. @Override
4. protected void onCreate(Bundle savedInstanceState) {
5. super.onCreate(savedInstanceState);
6. setContentView(R.layout.activity\_main);
7. findView();
8. setListener();
9. }
10. }
11. private void setListener() {
12. // TODO Auto-generated method stub
13. //选择框，通常用于在删除操作的时候提醒用户，以及提交啥的
14. this.btnChoice.setOnClickListener(new View.OnClickListener() {
16. @Override
17. public void onClick(View v) {
18. // TODO Auto-generated method stub
19. AlertDialog.Builder alert=new AlertDialog.Builder(MainActivity.this);
20. alert.setTitle("提示");
21. alert.setMessage("你好");
22. //对话框中每一个按钮都具备关闭对话框的能力
23. alert.setPositiveButton("再探究竟", new DialogInterface.OnClickListener() {
25. @Override
26. public void onClick(DialogInterface dialog, int which) {
27. // TODO Auto-generated method stub
28. Toast.makeText(MainActivity.this, "hi~", Toast.LENGTH\_SHORT).show();
29. }
30. });
31. alert.setNegativeButton("你瞅啥？", null);
32. alert.show();
33. }
34. });
35. }
36. private void findView() {
37. // TODO Auto-generated method stub
38. this.btnChoice=(Button) findViewById(R.id.main\_btn\_choice);
39. }
40. }

列表对话框：

1. public class MainActivity extends Activity {
2. private Button btnList;//列表
3. @Override
4. protected void onCreate(Bundle savedInstanceState) {
5. super.onCreate(savedInstanceState);
6. setContentView(R.layout.activity\_main);
7. findView();
8. setListener();
9. }
10. }
11. private void setListener() {
12. // TODO Auto-generated method stub
13. //列表框
14. this.btnList.setOnClickListener(new View.OnClickListener() {
16. @Override
17. public void onClick(View v) {
18. final String[] cityNames={"济南","青岛","淄博"};
19. // TODO Auto-generated method stub
20. AlertDialog.Builder alert=new AlertDialog.Builder(MainActivity.this);
21. alert.setTitle("选择");
22. alert.setItems(cityNames, new DialogInterface.OnClickListener() {
24. @Override
25. public void onClick(DialogInterface dialog, int which) {
26. // TODO Auto-generated method stub
27. //which表示选择项的下标
28. String cityName=cityNames[which];
29. Toast.makeText(MainActivity.this, cityName, Toast.LENGTH\_SHORT).show();
30. }
31. });
32. alert.setNegativeButton("你瞅啥？", null);
33. alert.show();
34. }
35. });
36. }
37. private void findView() {
38. // TODO Auto-generated method stub
39. this.btnList=(Button) findViewById(R.id.main\_btn\_list);
40. }
41. }

单选对话框：

1. public class MainActivity extends Activity {
2. private Button btnSignal;//单选
3. private String type;
4. @Override
5. protected void onCreate(Bundle savedInstanceState) {
6. super.onCreate(savedInstanceState);
7. setContentView(R.layout.activity\_main);
8. findView();
9. setListener();
10. }
11. }
12. private void setListener() {
13. // TODO Auto-generated method stub
14. //单选
15. this.btnSignal.setOnClickListener(new View.OnClickListener() {
17. @Override
18. public void onClick(View v) {
19. final String[] types=getResources().getStringArray(R.array.types);
20. int index=0;
21. for(int i=0;i<types.length;i++)
22. {
23. //开发过程中必须使用常量与变量比较，不能反过来，即常量放在左边
24. if(types[i].equals(type)){
25. index=i;
26. break;
27. }
28. }
29. // TODO Auto-generated method stub
30. AlertDialog.Builder alert=new AlertDialog.Builder(MainActivity.this);
31. alert.setTitle("单选");
32. alert.setSingleChoiceItems(types, index, new DialogInterface.OnClickListener() {
34. @Override
35. public void onClick(DialogInterface dialog, int which) {
36. // TODO Auto-generated method stub
37. type=types[which];
38. }
39. });
40. alert.setPositiveButton("确定", new DialogInterface.OnClickListener() {
42. @Override
43. public void onClick(DialogInterface dialog, int which) {
44. // TODO Auto-generated method stub
45. btnSignal.setText(type);
46. }
47. });
48. alert.show();
49. }
50. });
51. }
52. private void findView() {
53. // TODO Auto-generated method stub
54. this.btnSignal=(Button) findViewById(R.id.main\_btn\_signal);
55. }
56. }

复选对话框：

（待添加）

自定义对话框：

1. public class MainActivity extends Activity {
2. private Button btnDefine;//自定义
3. @Override
4. protected void onCreate(Bundle savedInstanceState) {
5. super.onCreate(savedInstanceState);
6. setContentView(R.layout.activity\_main);
7. findView();
8. setListener();
9. }
10. }
11. private void setListener() {
12. // TODO Auto-generated method stub
13. //自定义对话框
14. this.btnDefine.setOnClickListener(new View.OnClickListener() {
16. @Override
17. public void onClick(View v) {
18. // TODO Auto-generated method stub
19. //通过Java代码创建Android的控件
20. final EditText editOldPwd = new EditText(MainActivity.this);
21. editOldPwd.setHint("你好~输入好东西");
22. AlertDialog.Builder alert=new AlertDialog.Builder(MainActivity.this);
23. alert.setTitle("自定义 提示");
24. alert.setView(editOldPwd);
25. alert.setPositiveButton("确定", new DialogInterface.OnClickListener() {
27. @Override
28. public void onClick(DialogInterface dialog, int which) {
29. // TODO Auto-generated method stub
30. String pwd=editOldPwd.getText().toString();
31. Toast.makeText(MainActivity.this, pwd, Toast.LENGTH\_SHORT).show();
32. }
33. });
34. alert.setNegativeButton("没了", null);
35. alert.show();
36. }
37. });
38. this.btnMuti.setOnClickListener(new View.OnClickListener() {
40. @Override
41. public void onClick(View v) {
42. // TODO Auto-generated method stub
43. Toast.makeText(MainActivity.this, "仅供展示，不要摸我", Toast.LENGTH\_SHORT).show();
44. }
45. });
46. }
47. private void findView() {
48. // TODO Auto-generated method stub
49. this.btnDefine=(Button) findViewById(R.id.main\_btn\_define);
50. }
51. }

## 多知一点：Android相关小技术内容小结（一）

### 1、给控件添加事件的第二种方式（最常用）

1）创建空间对象

1. ……
2. public class MainActivity extends Activity {
3. private CheckBox checkJava;
4. private CheckBox checkAndroid;
5. private Button btnShow;
7. @Override
8. protected void onCreate(Bundle savedInstanceState) {
9. ……

2）控件对象绑定页面控件

在onCreate方法中写的：

1. //以下方法的调用顺序不能更改
2. findView();//负责实现控件的绑定
3. setListener();负责给控件设置监听程序

在Activity方法内进行写的：

1. /\*\*
2. \* 负责实现控件的绑定
3. \*/
4. private void findView() {
5. this.btnShow=(Button) findViewById(R.id.main\_btn\_show);
6. this.checkAndroid=(CheckBox) findViewById(R.id.main\_check\_android);
7. this.checkJava=(CheckBox) findViewById(R.id.main\_check\_Java);
9. }

3）给控件设置相应的事件监听器

在Activity方法内进行写的：

1. /\*\*
2. \* 负责给控件设置监听程序
3. \*/
4. private void setListener() {
5. //给按钮设置 点击的监听
6. this.btnShow.setOnClickListener(new View.OnClickListener() {
7. @Override
8. public void onClick(View v) {
9. //添加处理程序
10. String like="";
11. if(checkJava.isChecked()){
12. like+="Java,";
13. }
14. if(checkAndroid.isChecked()){
15. like+="android";
16. }
17. //利用系统提供的工具完成相关的验证
18. if(TextUtils.isEmpty(like)){
19. like="=======";
20. }
21. Toast.makeText(MainActivity.this, like, Toast.LENGTH\_SHORT)
22. .show();
23. }
24. });

4）在监听程序中完成需要的功能（见上述代码）

### 2、在Activity之中显示Toast（吐司）

1. Toast.makeText(MainActivity.this, sex,Toast.LENGTH\_SHORT).show();

Toast.makeText方法的形参表参数为：当前Activity，需要显示的信息，显示信息的秒数（.show();方法是如果需要Toast显示则进行写入）。而显示信息的秒数，这里Android提供了两个固定参数进行设置，一个是Toast.LENGTH\_SHORT另一个是Toast.LENGTH\_LONG，前者为1秒显示后者为3秒显示。

### 3、在Android/Java之中使用正则表达式

在Android/Java之中使用正则表达式示例：（异常简单）

1. // Java中使用正则表达式
2. // 1.创建表达式,6-12为之间，以字母开始，可以包含数字字母
3. String p="[a-zA-Z][a-zA-Z0-9]{5,11}";
4. // 2.使用表达式
5. boolean flag=Pattern.matches(p, str);
6. if(!flag){
7. //你的数据格式有误！
8. }

**正则表达式中关键的表示字段：**

[]:表示其中的任意一个字符，例如[a-z],[0-9]分别表示小写字母和阿拉伯数字。

{}:表示出现的次数，例如a{3}表示接下来a在字符串中出现三次。{m}表示正好出现m次，{m,n}表示出现最少 m 次最多 n次，{m,}表示最少m次没有上限。“.”在正则中是特殊字符，表示除了回车以外的任意一个字符。增加“\”转义即可。

e.g. 若要用正则表达式验证身份证，则需要用 [0-9]{17}[0-9Xx] 表示。

若要用正则表达式验证手机号码，则需要用 1[34578] [0-9] {9} 表示。

若要用正则表达式验证中文，则需要用 [\u4e00-\u9fa5] 表示。

若要用正则表达式验证用户名和密码，则需要用[a-zA-Z0-9\_]{6,12}表示。

若要用正则表达式验证邮箱，则需要用[a-zA-Z0-9\_][a-zA-Z0-9\_]{0,}@ [a-zA-Z0-9\_]{1,}\.com(.cn){0,1}表示。( [a-zA-Z0-9\_]{1,}@ [a-zA-Z0-9\_]{1,}\.com(.cn){0,1}也应该可以 )

**在Android之中使用正则表达式进行邮箱验证示例：**

onCreate方法中声明findView();与setListener();，Activity继承类中声明private EditText editEmail;与private Button buttonTest;对象。

setListener()方法：

1. private void setListener() {
2. //XXX
3. this.buttonTest.setOnClickListener(new View.OnClickListener() {
4. @Override
5. public void onClick(View v) {
6. String email=editEmail.getText().toString();
7. String emailPattern=
8. "[a-zA-Z0-9][a-z0-9A-Z]{0,}@[0-9a-zA-Z]{1,}\\.com";
9. boolean flag=Pattern.matches(emailPattern, email);
10. if(!flag){
11. editEmail.setError("邮箱格式不正确");
12. // editEmail.setText("");
13. // 高亮显示，选择用户输入的内容
14. editEmail.selectAll();
15. }
16. }
17. });

findView()方法：

1. private void findView() {
2. this.editEmail=(EditText) findViewById(R.id.main\_edit\_email);
3. this.buttonTest=(Button) findViewById(R.id.main\_button\_test);
4. }

# Day 005: Android UI组件（二）：界面布局、样式、适配器的应用

## 一、Android界面布局的特点和使用方法

界面布局（Layout）是用户界面结构的描述，定义了界面中所有的元素、结构和相互关系。声明Android程序的界面布局有两种方法：使用XML文件描述界面布局、在程序运行时动态添加或修改界面布局。用户既可以独立使用任何一种声明界面布局的方式，也可以同时使用两种方式。

在Android应用开发中，我们使用XML文件声明界面布局，对于这种生命界面布局的方式，有将程序的表现层和控制层分离，以及在后期修改用户界面时，无需更改程序的源代码的特点。同时用户还能够通过可视化工具直接看到所设计的用户界面，有利于加快界面设计的过程，并且为界面设计与开发带来极大的便利性。

Android界面布局（Layout）中有FrameLayout（框架布局）、LinearLayout（线性布局）、RelativeLayout（关联布局）、TableLayout/GridLayout/GridView（网状或表格布局）四种，其简介分别如下：

FrameLayout – 用于显示一堆子视图控件。多视图控件可以添加到这个布局中。它可以用来在相同的屏幕空间展示多个控件。

LinearLayout – 用于在单行或单列中显示子视图控件。这对于创建表单来说是非常方便的布局方法。

RelativeLayout – 用于相对彼此地显示子视图控件。比如，你可以设置一个控件相对另一个控件“在上方”或“在下方”或“在左边”或“在右边”。你也可以相对于父级元素的边界来放置子视图控件。

TableLayout/GridLayout/GridView – 用于将子视图控件组织到行或列。对于表格的每一行，单个视图控件使用TableRow布局视图被添加到表格的每一行。

这里我们详细介绍常用的线性布局和框架布局，简略介绍网状布局，对于其他布局方式，我们这里不进行介绍。（RelativeLayout是默认创建的模式，经常使用，可自行理解）

### 1、线性布局

线性布局（LinearLayout）是一种重要的界面布局中，也是经常使用到的一种界面布局。在线性布局中，所有的子元素都按照垂直或水平的顺序在界面上排列：如果垂直排列，则每行仅包含一个界面元素；如果水平排列，则每列仅包含一个界面元素。

**线性布局的重要特性和属性：**

1. android:background 设置整个布局画面的背景。
2. android:orientation=“horizontal” 子元素的排列队形，是横向排列，还是纵向排列。
3. android:gravity=“bottom” 子元素在布局中的缺省对齐方式。
4. android:padding 设置子元素与布局边缘之间的空白。
5. layout\_weight 用于给一个线性布局中的诸多视图的重要度赋值权重。
6. 所有的视图都有一个layout\_weight值，默认为零，意思是需要显示多大的视图就占据多大的屏幕空间。
7. 若赋一个高于零的值，则将父视图中的可用空间分割，分割大小具体取决于每一个视图的layout\_weight值以及该值在当前屏幕布局的整体 layout\_weight值和在其它视图屏幕布局的layout\_weight值中所占的比率而定。

**线性布局示例：（实现效果：线性布局实现底部菜单块，并设置显隐藏）**

1. <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2. xmlns:tools="http://schemas.android.com/tools"
3. android:layout\_width="match\_parent"
4. android:layout\_height="match\_parent"
5. tools:context="${relativePackage}.${activityClass}" >
6. <LinearLayout
7. android:id="@+id/main\_linear\_content"
8. android:layout\_width="match\_parent"
9. android:layout\_height="match\_parent"
10. android:orientation="vertical">
11. <LinearLayout
12. android:id="@+id/main\_linear\_top"
13. android:layout\_width="match\_parent"
14. android:layout\_height="wrap\_content"
15. android:background="#ff00ff"
16. android:orientation="vertical"
17. android:layout\_weight="1">
18. <Button
19. android:id="@+id/main\_button\_show"
20. android:layout\_width="wrap\_content"
21. android:layout\_height="wrap\_content"
22. android:text="显示"/>
23. <Button
24. android:id="@+id/main\_button\_hidden"
25. android:layout\_width="wrap\_content"
26. android:layout\_height="wrap\_content"
27. android:text="隐藏" />
28. </LinearLayout>
29. <LinearLayout
30. android:id="@+id/main\_linear\_bottom"
31. android:layout\_width="match\_parent"
32. android:layout\_height="wrap\_content"
33. android:orientation="horizontal"
34. android:visibility="gone">
35. <TextView
36. android:id="@+id/main\_text\_msg"
37. android:layout\_width="wrap\_content"
38. android:layout\_height="wrap\_content"
39. android:text="消息"
40. android:drawableTop="@android:drawable/ic\_dialog\_map"
41. android:layout\_weight="1"
42. android:gravity="center"/>
43. <TextView
44. android:id="@+id/main\_text\_info"
45. android:layout\_width="wrap\_content"
46. android:layout\_height="wrap\_content"
47. android:text="我"
48. android:drawableTop="@android:drawable/ic\_dialog\_map"
49. android:layout\_weight="1"
50. android:gravity="center"/>
51. <TextView
52. android:id="@+id/main\_text\_add"
53. android:layout\_width="wrap\_content"
54. android:layout\_height="wrap\_content"
55. android:text="添加"
56. android:drawableTop="@android:drawable/ic\_dialog\_map"
57. android:layout\_weight="1"
58. android:gravity="center"/>
59. </LinearLayout>
60. </LinearLayout>
61. </RelativeLayout>

### 2、框架布局

框架布局（FrameLayout）是最简单的界面布局，是用来存放一个元素的空白空间，且子元素的位置是不能够指定的，只能够放置在空白空间的左上角。如果有多个子元素，后放置的子元素将遮挡先放置的子元素。

使用Android SDK中提供的层级观察器（Hierarchy Viewer）进一步分析界面布局。层级观察器能够对用户界面进行分析和调试，并以图形化的方式展示树形结构的界面布局。

**框架布局的重要特性和属性：**

android:gravity：android:gravity属性是对该view 内容的限定.比如一个button 上面的text. 你可以设置该text 在view的靠左，靠右等位置.以button为例，android:gravity="right"则button上面的文字靠右。

android:layout\_gravity：android:layout\_gravity是用来设置该view相对与起父view 的位置.比如一个button 在linearlayout里，你想把该button放在靠左、靠右等位置就可以通过该属性设置.以button为例，android:layout\_gravity="right"则button靠右。

android:layout\_alignParentRigh：使当前控件的右端和父控件的右端对齐。这里属性值只能为true或false，默认false。

**框架布局示例：（实现效果：框架布局实现抽奖转盘）**

XML布局文件：

1. <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2. xmlns:tools="http://schemas.android.com/tools"
3. android:layout\_width="match\_parent"
4. android:layout\_height="match\_parent"
5. tools:context="${relativePackage}.${activityClass}"
6. android:background="@drawable/bg">
7. <FrameLayout
8. android:layout\_width="match\_parent"
9. android:layout\_height="match\_parent">
10. <RelativeLayout
11. android:layout\_width="match\_parent"
12. android:layout\_height="match\_parent">
13. <ImageView
14. android:id="@+id/main\_image\_rotate"
15. android:layout\_width="wrap\_content"
16. android:layout\_height="wrap\_content"
17. android:src="@drawable/rotate"
18. android:layout\_centerInParent="true"/>
19. </RelativeLayout>
20. <RelativeLayout
21. android:layout\_width="match\_parent"
22. android:layout\_height="match\_parent">
23. <ImageView
24. android:id="@+id/main\_image\_btn"
25. android:layout\_width="wrap\_content"
26. android:layout\_height="wrap\_content"
27. android:src="@drawable/bt"
28. android:layout\_centerInParent="true"
29. android:maxWidth="140dp"
30. android:maxHeight="140dp"
31. android:adjustViewBounds="true"
32. android:paddingBottom="30dp"/>
33. </RelativeLayout>
34. </FrameLayout>
35. </RelativeLayout>

Activity动作文件：

1. private ImageView imageRotate;
2. private ImageView imageBtn;
3. private void setListener() {
4. this.imageBtn.setOnClickListener(new View.OnClickListener() {
6. @Override
7. public void onClick(View v) {
8. RotateAnimation rotate=
9. new RotateAnimation(0, 3630,
10. Animation.RELATIVE\_TO\_SELF, 0.5f,
11. Animation.RELATIVE\_TO\_SELF, 0.5f);
12. rotate.setDuration(5000);
13. rotate.setFillAfter(true);
15. imageRotate.startAnimation(rotate);
17. }
18. });
20. }
21. private void findView() {
22. this.imageRotate=(ImageView) findViewById(R.id.main\_image\_rotate);
23. this.imageBtn=(ImageView) findViewById(R.id.main\_image\_btn);
25. }

### 3、表格布局（并不完整，而且GridLayout应是网格要独立整理，待定）

表格布局（TableLayout/GridLayout/GridView）也是一种常用的界面布局，它将屏幕划分网格，通过指定行和列可以将界面元素添加的网格中，网格的边界对用户是不可见的。

表格布局还支持嵌套，可以将另一个表格布局放置在前一个表格布局的网格中，也可以在表格布局中添加其他界面布局，例如线性布局、相对布局等等。

**建立表格布局要注意以下几点：**

1. 向界面中添加一个线性布局，无需修改布局的属性值。其中，Id属性为TableLayout01，Layout width和Layout height属性都为wrap\_content。
2. 向TableLayout01中添加两个TableRow。TableRow代表一个单独的行，每行被划分为几个小的单元，单元中可以添加一个界面控件。其中，Id属性分别为TableRow01和TableRow02，Layout width和Layout height属性都为wrap\_content。

**表格布局的重要的属性：**

android:collapse="1”：隐藏该TableLayout里的TableRow的列1，即第2列（从0开始计算），若有多列要隐藏，用“，”隔开。

android:stretchColumns=“0,1,2”：设置列0、1、2为可伸展列。

android:shrinkColumns=“1,2”：设置列1、2为可收缩列。

### 4、网格布局

gridLayout、gridView:

1.创建gridView中每一个项，内容的布局文件

2.创建填充gridView的数据，格式

List<Map<String,String>

List<HashMap<String,String>>

List：表示要填充的所有数据

Map<String,String> 表示每一项中显示的内容

3.创建适配器，转换数据

ArrayAdapter：只能处理简单的数据 ，数组，或者简单的List 集合

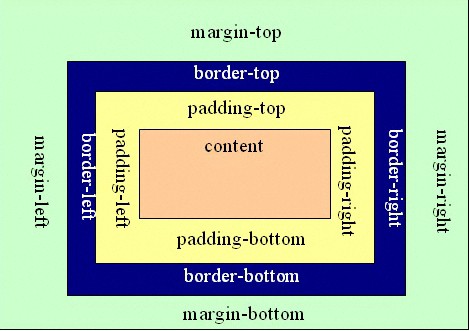
SimpleAdapter：用于把自己的布局文件绑定到控件中

4.把适配器设置给我们的控件

### 5、相对布局

相对布局（RelativeLayout）是一种非常灵活的布局方式，能够通过指定界面元素与其他元素的相对位置关系，确定界面中所有元素的布局位置。

特点：能够最大程度保证在各种屏幕类型的手机上正确显示界面布局。



相对布局（RelativeLayout）示意图

**相对布局（RelativeLayout）中重要的属性：**

**属性值为true或false**

android:layout\_centerHrizontal 水平居中

android:layout\_centerVertical 垂直居中

android:layout\_centerInparent 相对于父元素完全居中

android:layout\_alignParentBottom 贴紧父元素的下边缘

android:layout\_alignParentLeft 贴紧父元素的左边缘

android:layout\_alignParentRight 贴紧父元素的右边缘

android:layout\_alignParentTop 贴紧父元素的上边缘

android:layout\_alignWithParentIfMissing 如果对应的兄弟元素找不到的话就以父元素做参照物

**属性值必须为id的引用名“@id/id-name”**

android:layout\_below      在某元素的下方

android:layout\_above      在某元素的的上方

android:layout\_toLeftOf   在某元素的左边

android:layout\_toRightOf  在某元素的右边

android:layout\_alignTop   本元素的上边缘和某元素的的上边缘对齐

android:layout\_alignLeft  本元素的左边缘和某元素的的左边缘对齐

android:layout\_alignBottom 本元素的下边缘和某元素的的下边缘对齐

android:layout\_alignRight  本元素的右边缘和某元素的的右边缘对齐

**属性值为具体的像素值，如30dip，40px**

android:layout\_marginBottom 离某元素底边缘的距离

android:layout\_marginLeft 离某元素左边缘的距离

android:layout\_marginRight 离某元素右边缘的距离

android:layout\_marginTop 离某元素上边缘的距离

## 二、实现欢迎界面跳转

二话不说放代码，这里有两种方法。但是不管是哪种方法，都是通过线程进行控制关闭然后进行跳转Activity的。

**老师讲的：**

1. public class WelcomeActivity extends Activity {
2. private Handler handler=new Handler(){
3. public void handleMessage(android.os.Message msg) {
4. switch (msg.what) {
5. case 1:
6. Intent intent=new Intent();
7. intent.setClass(WelcomeActivity.this, MainActivity.class);
8. startActivity(intent);
9. finish();
10. //在startActivity或者finish之后，实现该方法
11. overridePendingTransition(R.anim.anim\_in, R.anim.anim\_out);
12. break;
13. default:
14. break;
15. }
16. };
17. };
18. @Override
19. protected void onCreate(Bundle savedInstanceState) {
20. super.onCreate(savedInstanceState);
21. setContentView(R.layout.activity\_welcome);
23. Thread thread=new Thread(new Runnable() {
25. @Override
26. public void run() {
27. try {
28. Thread.sleep(3000);
29. } catch (InterruptedException e) {
30. // TODO Auto-generated catch block
31. e.printStackTrace();
32. }
33. //使用消息机制
34. handler.sendEmptyMessage(1);
35. }
36. });
37. thread.start();
38. }
39. }

**实际使用的：**

1. public class StartActivity extends Activity {
2. @Override
3. protected void onCreate(Bundle savedInstanceState) {
4. super.onCreate(savedInstanceState);
5. setContentView(R.layout.activity\_start);
6. Handler x = new Handler();
7. x.postDelayed(new splashhandler(), 2000);
8. }
9. class splashhandler implements Runnable{
10. public void run() {
11. startActivity(new Intent(StartActivity.this,LoginActivity.class));
12. StartActivity.this.finish();
13. }
14. }
15. }

## 三、通过网格布局初识服务层

Service服务层，实现数据的封装。

（待写）

## 四、实现动画效果

实现动画，有两种方式：1）通过Java代码实现2）通过xml文件配置。在Java中使用xml文件，动画的文件通常放在res/anim目录下，也可以放在res/drawable目录下（目录名不能错）。

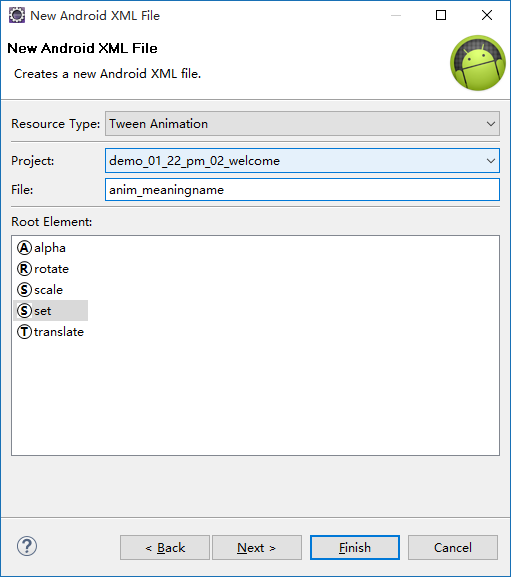
首先介绍一个方式：

MainActivity.java文件（findView();方法在onCreate方法中已创建）

1. protected void onResume() {
2. // TODO Auto-generated method stub
3. super.onResume();
4. Animation anims=
5. AnimationUtils.loadAnimation(MainActivity.this,
6. R.anim.anim\_image);
7. image.startAnimation(anims);
8. }
9. private void findView() {
10. this.image=(ImageView) findViewById(R.id.main\_image);
11. }

下面介绍第二种方式：

在res下信件anim文件夹（文件夹名不得修改，不能随意其名，否则Android开发ADT不认），之后新建Android XML File，这时他会提示如下内容：



在anim文件夹下New Android XML File对话框

其中，alpha表示透明度，rotate表示旋转，scale表示缩放，set表示动画集合，translate表示移动。

**这里，我们来展示一个动画集合的例子：**

anim\_image.xml文件

1. <set xmlns:android="http://schemas.android.com/apk/res/android">
2. <scale
3. android:fromXScale="1"
4. android:toXScale="4"
5. android:fromYScale="1"
6. android:toYScale="4"
7. android:pivotX="50%"
8. android:pivotY="50%"
9. android:duration="3000"
10. android:repeatCount="3"
11. android:repeatMode="reverse"></scale>
12. <rotate
13. android:fromDegrees="0"
14. android:toDegrees="10200"
15. android:duration="3000"
16. android:pivotX="50%"
17. android:pivotY="50%"/>
18. <alpha
19. android:fromAlpha="0"
20. android:toAlpha="1"
21. android:duration="5000"
22. android:startOffset="3000"></alpha>
23. </set>

## 多知一点：Android相关小技术内容小结（二）

1、在Android开发中所有空间包括布局都必须提供id属性。

2、layout\_weight表示该控件占父容器的多少分之几，指的是可分配空间。如果该XML文件中只有一个控件带layout\_weight，那么它将占有所有可分配空间。

3、不管页面多么复杂，也都是由一个个局部组成的。每一个局部，都是一个布局。

4、android:visibility 表示控件的隐藏域显示

="gone" 表示该控件消失，位置都没有了

="invisible" 表示控件隐藏，位置还在

="visible" 表示显示

5、在布局文件中出现的属性，在Java代码中都有相应的方法与之对应。

6、在android布局文件中如果有多个布局使用相同的内容（例如底部菜单之类的），把相同的布局单独创建一个布局文件。通过<include />把该布局文件应用到对应的位置。

7、在培训期间经常使用的事件：

1）View.onClickListener //用于给控件添加点击事件

2）DialogInterface.onClickListener：//给对话框的按钮添加的事件

3）AdapterView.onItemClickListener: //给复杂控件中Item添加事件

# Day 006: Android UI组件（三）：ListView界面布局与SQLite数据库应用初识

## 一、列表视图（ListView）

### 1、ListView

按钮设置

自动登录

## 二、SQLite数据库介绍

SQLite，是一款轻型的数据库，是遵守ACID的关系型数据库管理系统，它包含在一个相对小的C库中。它是D.RichardHipp建立的公有领域项目。它的设计目标是嵌入式的，而且目前已经在很多嵌入式产品中使用了它，它占用资源非常的低，在嵌入式设备中，可能只需要几百K的内存就够了。它能够支持Windows/Linux/Unix等等主流的操作系统，同时能够跟很多程序语言相结合，比如 Tcl、C#、PHP、Java等，还有ODBC接口，同样比起Mysql、PostgreSQL这两款开源的世界著名数据库管理系统来讲，它的处理速度比他们都快。SQLite第一个Alpha版本诞生于2000年5月。 至2015年已经有15个年头，SQLite也迎来了一个版本 SQLite 3已经发布。

不像常见的客户-服务器范例，SQLite引擎不是个程序与之通信的独立进程，而是连接到程序中成为它的一个主要部分。所以主要的通信协议是在编程语言内的直接API调用。这在消耗总量、延迟时间和整体简单性上有积极的作用。整个数据库(定义、表、索引和数据本身)都在宿主主机上存储在一个单一的文件中。它的简单的设计是通过在开始一个事务的时候锁定整个数据文件而完成的。

## 三、SQLite数据库在Android的应用