Android 真机开发教程

Z Z ： Dason
Q Q ： 623466642
空间: http://hi.baidu.com/dasonn/

一：系统安装与 HelloWorld

【目的】

安装智能手机开发相关软件平台。

【要求】

1、 完成智能手机开发平台安装、以及相关配置

2、 并实现 Hello World

3、 了解项目的基本文件目录结构

【原理】

Eclipse 安装原理，Android 编程方法

【过程】

1、 安装 JAVA JDK

下载网址：http://java.sun.com/javase/downloads/

2、 安装 Eclipse

下载网址：http://www.eclipse.org/downloads/

直接解压拷贝。

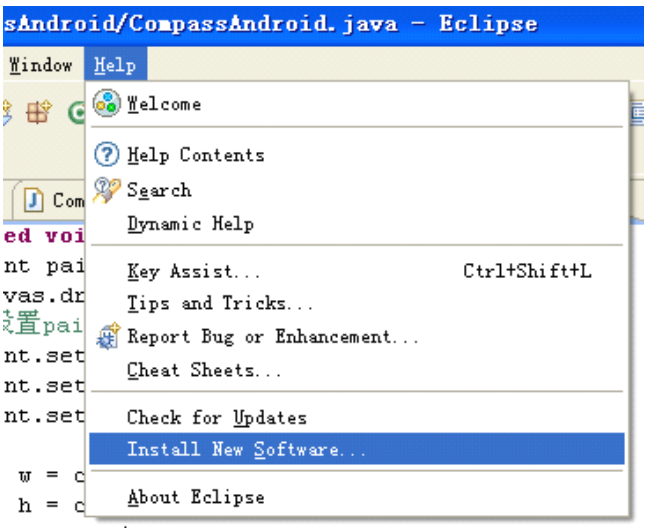
3、 安装 Android

http://developer.android.com 或 http://androidappdocs.appspot.com/index.html

安装 Android 的 SDK。

4、 安装 ADT（Android Development Tools）

http://developer.android.com 或 http://androidappdocs.appspot.com/index.html



5、 安装手机 USB 驱动

http://developer.android.com 或 http://androidappdocs.appspot.com/index.html

也可由系统自行搜索安装，需将手机设置在“应用开发”功能上。如果用模拟器调试，

则可暂时不装。

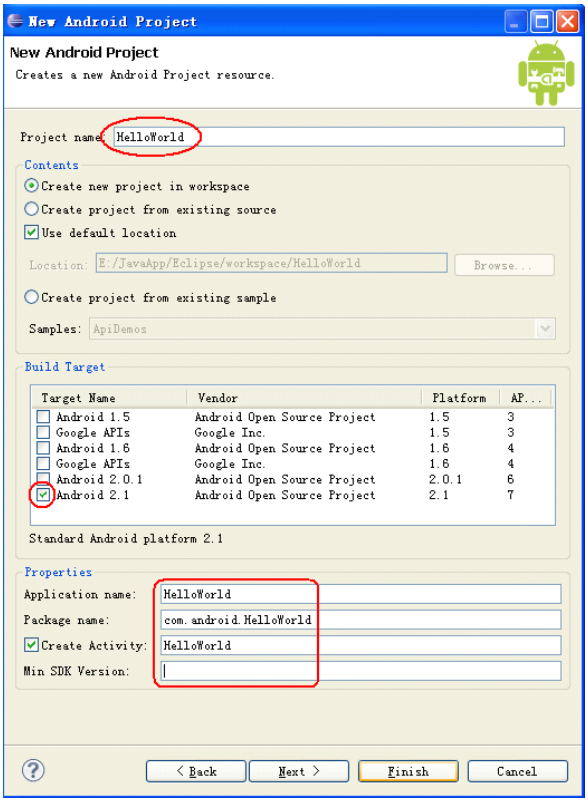
6、 建立新项目，实现 Hello World。

Open Eclipse.

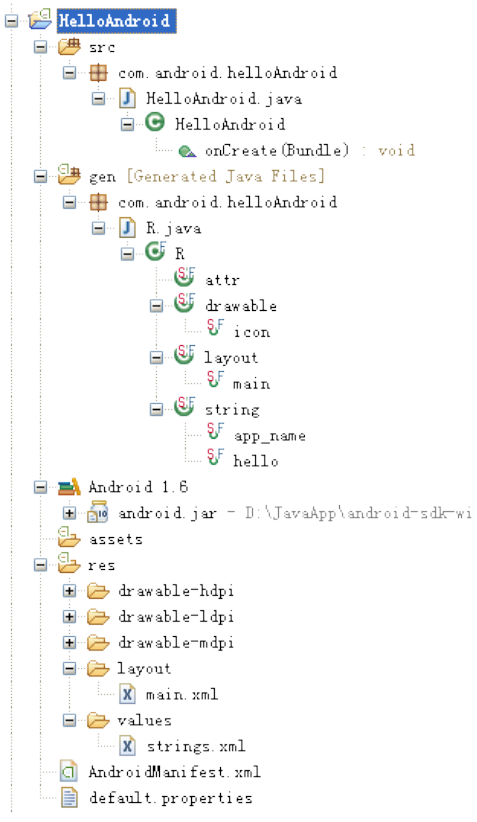
Click the menu File -> New -> Project.

Expand the Android folder and select Android Project.

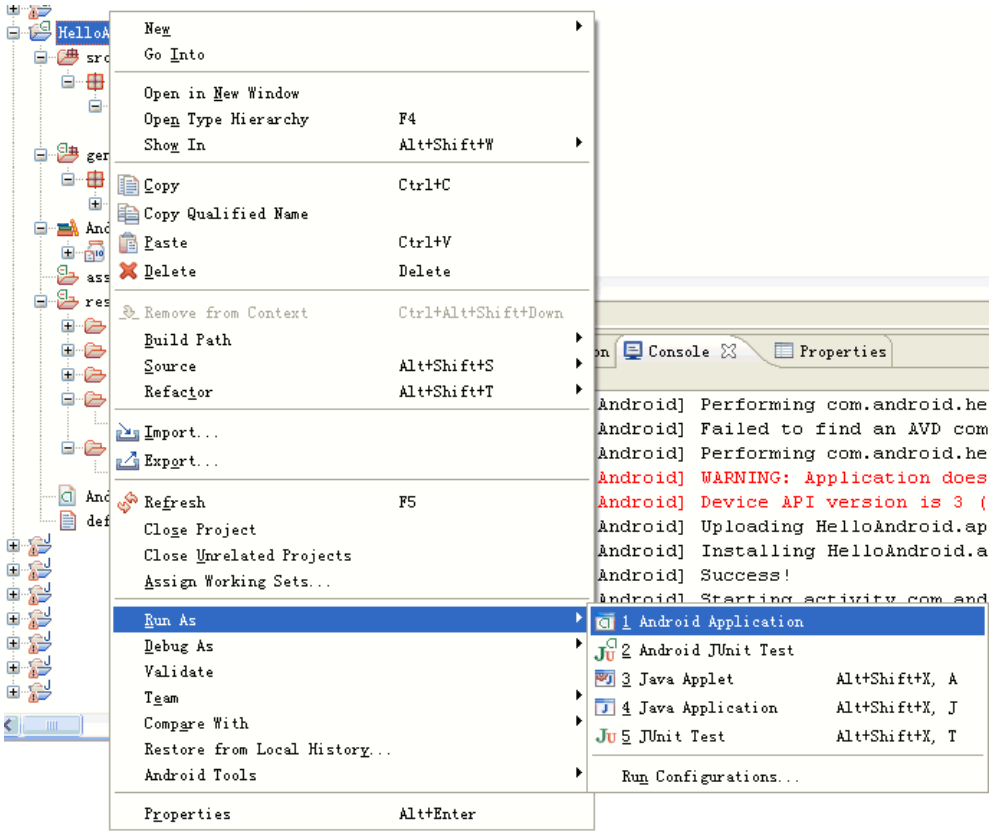
Name the project HelloWorld



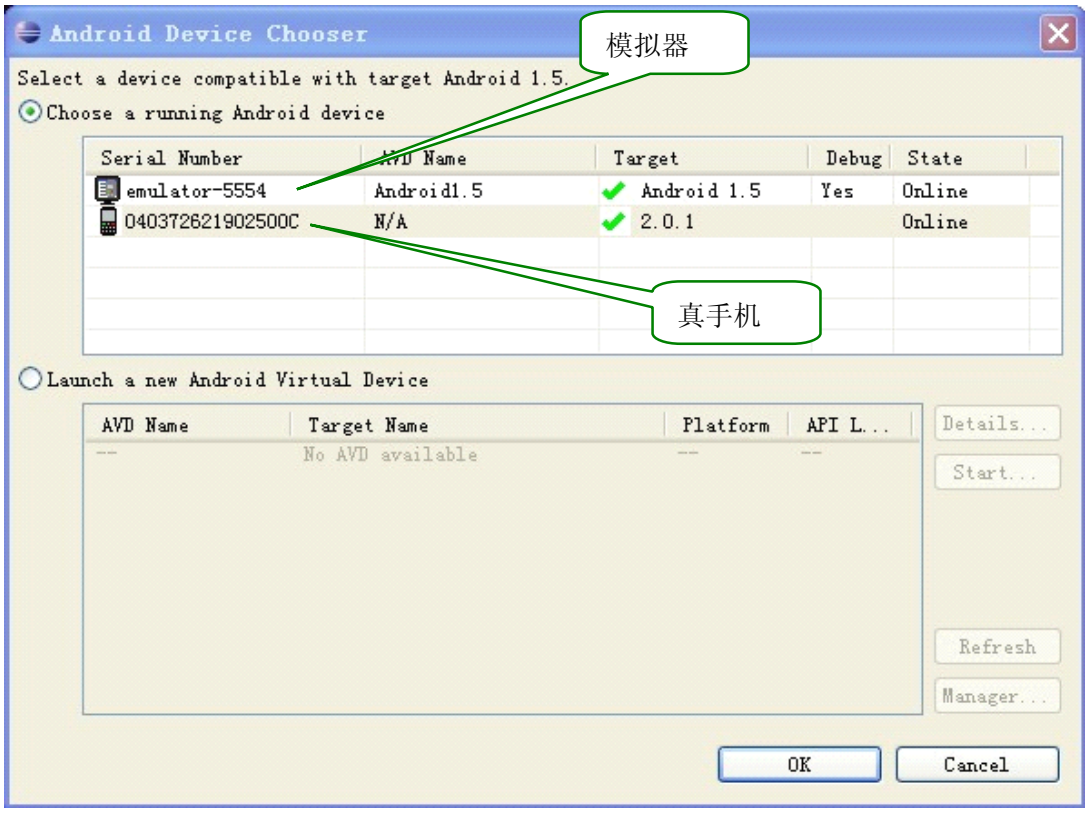
得到的文件结构如下：



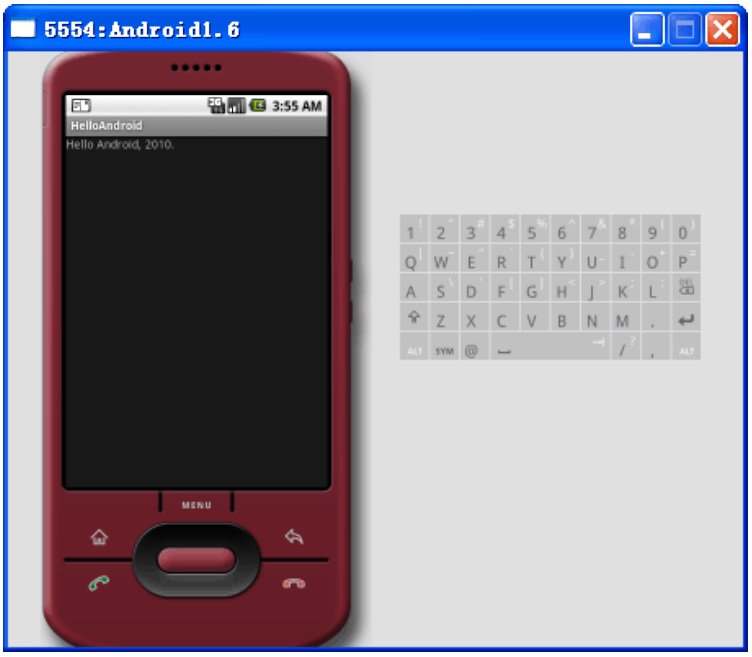
运行：



选运行的设备，可以是模拟器，也可以是真机(如果已经连接好真实手机的话)：



模拟器运行：



真实手机调试：

在 Android 的应用开发中，通常使用的是 java 语言开发，除了需要熟悉 JAVA 语言基础知识之外，还需要了解 Android 提供扩展的 java 功能。

Android 重要包的描述

android.app	封装了 Android 应用程序全局模型的高级类。
android.content	包含用于在设备上访问和发布数据的类。

android.database	包含了用于浏览内容提供源返回数据的类。
android.database.sqlite	包含了 SQLLite 数据库管理类，应用程序可以利用这些类来管理其私有数据库。
android.graphics	允许你直接在屏幕上绘图的绘图工具，比如画布、颜色过滤器、点和矩形等。
android.graphics.drawable	提供了用于管理多种可视界面元素的类，这些可视界面元素仅用于显示，例如 bitmap 和 gradient。
android.graphics.glutils	提供了大量能够在 Android 设备上使用 OpenGL 嵌入式系统版(OpenGL ES)绘图的类.
android.hardware	提供对硬件设备的支持，这些硬件设备不一定会出现在每一个 Android 设备上。
android.location	定义 Android 定位和相关服务的类。
android.media	定位，视频，音频 和相关的服务
android.net	用于网络连接的类，功能比 ava.net.* 强大。
android.opengl	提供 OpenGL（高性能图形算法行业标准）工具。 3D 加速等
android.os	提供设备上基础的操作系统服务、信息传递和进程间通信。
android.provider	提供用于方便地访问 Android 支持的内容提供源的类。
android.sax	一个可以方便地编写高效、健壮的 SAX handler 的框架。
android.speech.recognition	提供用于语音识别的类。
android.telephony	提供了用于拨打、接收以及监听电话和电话状态的工具。
android.telephony.gsm	提供了用于从 GSM 电话上控制或读取数据的类。
android.text	提供了用于在屏幕上绘制或跟踪文本和文本跨度的类。
android.text.method	提供了用于监听或修改键盘输入的类。
android.text.style	提供了用于预览或修改视图对象中文本跨度形式的类。
android.util	提供了通用的工具方法，例如日期/时间操作、64 位编码解码器、字符串数组互换方法和与 XML 相关的方法。
android.view	提供了用于处理屏幕布局和用户交互的基本 UI 类。
android.view.animation	提供了动画处理的类
android.webkit	提供了浏览网页的工具。
android.widget	widget 包包含了用在应用程序屏幕上的 UI 元素(绝大部分可视)。

文件格式描述：

Android 的相关文件类型：

Java---应用程序源文件

Android 本身相当一部分是由 java 编写而成，而且 android 应用必须使用 java 开发

class---java 编译后的目标文件：

是由 java 虚拟机编译而成一个字节码文件，在之前我们用所学的 j2ee 以及 j2se 它是一个可执行文件，但是在 Android 当中它只是一个目标文件即过渡文件

dex---Android 平台可执行文件：

Android 自己提供了一个虚拟机（Dalvik），这种虚拟机执行的并非 java 字节码，而是另一种字节码：dex 格式的字节码，在 JVM 将 java 文件编译成 Class 文件后，再次通过 Android

平台工具将此 Class 文件转换成 dex 字节码

apk 文件---Android 上的安装文件

Apk 是 Android 安装包的扩展名，一个 Android 安装包包含了与某个 Android 应用程序相关的所有文件，apk 文件将 androidManifest.xml 文件，应用程序代码（dex 文件）资源文件和其他文件打成一个压缩包，一个工程只能打进一个 apk 文件（有点类似 exe 文件）。

二：界面设计：控件与布局

【目的】

Android 编程基础，UI 设计。

【要求】

1、 了解 Android 编程原理

2、 掌握界面控件设计

3、 掌握控件的事件处理编程

【原理】

UI 设计原理

【过程】

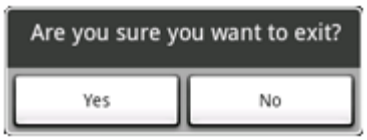
1、 了解各种控件的基本功能

各种控件：

Menu

TextView、EditText、

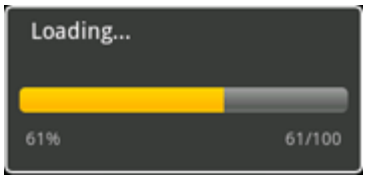
Button



Radio button

List

ProgressBar;



2、 了解布局 Layout 的应用

各种控件通过布局，确定在屏幕上显示的方式，与相互位置关系。有设计一个良好的要机界面，必须了解相关的布局，选择合适的布局安排各个控件。

多种 Layout：

AbsoluteLayout

FrameLayout

GridView

LinearLayout

ListLayout

RadioGroup

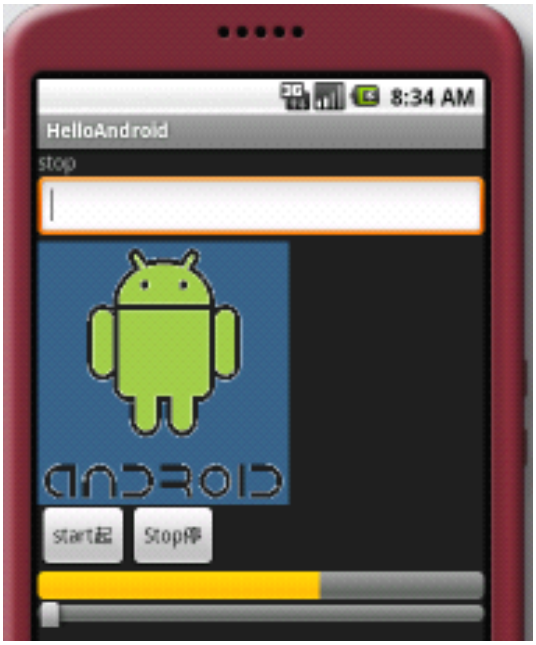
TableLayout

……

3、 利用布局安排各种控件，设计良好用户界面

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView android:id="@+id/TextView01"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
<EditText android:id="@+id/EditText01"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
<ImageView android:id="@+id/ImageView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/adr"
    />
<LinearLayout android:id="@+id/LinearLayout01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
<Button android:id="@+id/Button01"
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:text="@string/btn_name"
    />
<Button android:id="@+id/Button02"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/stp_name"
    />
</LinearLayout>
<ProgressBar android:id="@+id/progressbar01"
    android:layout_width="fill_parent"
    android:layout_height="20px"
    style="?android:attr/progressBarStyleHorizontal"
    />
<SeekBar android:id="@+id/seekbar01"
    android:layout_width="fill_parent"
    android:layout_height="20px"
    style="?android:attr/progressBarStyleHorizontal"
    />
</LinearLayout>
```




17:13
HelloAndroid
Hello Android, 2010.

三：图形绘制与 OpenGL ES

【目的】

在屏幕绘制各种图形，了解 OpenGL

【要求】

1、 了解在屏幕绘图方法

2、 了解 OpenGL

【原理】

【过程】

1、 绘制直线、园、曲线等各种图形

2、 显示字符

3、 利用 OpenGL ES 编程方法

Android 中的图形系统采用 Client/Server 架构。Server （即 SurfaceFlinger)主要由 c++代码编写而成。Client 端代码分为两部分，一部分是由 Java 提供的供应用使用的 api，另一部分则是由 c++写成的底层实现。

Android 图形系统中通过 surface 为 view 创建一个 Canvas 对象，管理 view 在 surface 上的绘图操作。View 及其子类（如 TextView, Button）要画在 surface 上。

OpenGL ES (OpenGL for Embedded Systems) 是一个针对嵌入式应用的，免费的，支持全功能 2D、3D 的跨平台 API（OpenGL® ES is a royalty-free, cross-platform API for full-function 2D and 3D graphics on embedded systems - including consoles, phones, appliances and vehicles)。目前主要由 3 个版本，1.0，1.1，2.0。

OpenGL ES 1.0 是以 OpenGL 1.3 规范为基础的，OpenGL ES 1.1 是以 OpenGL 1.5 规范为基础的，1.1 完全兼容 1.0。OpenGL ES 2.0 则是参照 OpenGL 2.0 规范定义的。简单的来说，OpenGL ES 是 OpenGL 针对嵌入式应用的简化版，也就是 android 使用的标准。OpenGL ES 1.1 强调 api 的硬件加速，OpenGL ES 2.0 更强调 3D 能力。

OpenGL ES 1.1 和 OpenGL ES 2.0 之间的关系并不是旧版本和新版本之间的差别，而是一个针对相对低端的应用，一个针对高级应用，OpenGL 官方的 roadmap 也是将这两个版本并行发展的。2.X 并不能百分百兼容 1.X。

Android 现在支持 1.X 和 2.X。OpenGL ES 是专为内嵌和移动设备设计的一个 2D/3D 轻量级图形库，它基于 OpenGL API 设计，是 OpenGL 三维图形 API 的子集。Android 里有三个与 OpenGL 有关的包：

android.opengl

javax.microedition.khronos.egl

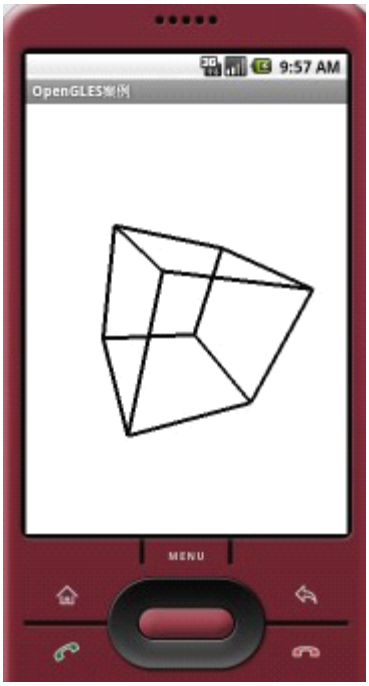
javax.microedition.khronos.opengles

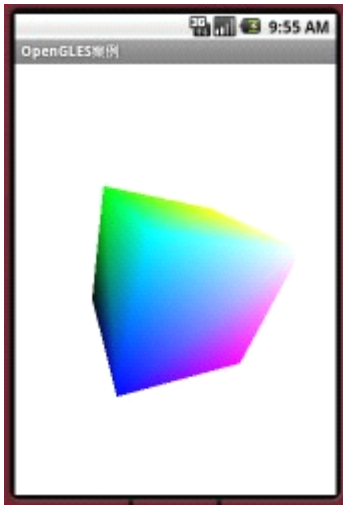
```
    public void onDrawFrame(GL10 gl) {
        //一般的opengl程序，首先要做的就是清屏
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT |
GL10.GL_DEPTH_BUFFER_BIT);
```

```
        //紧接着设置模型视图矩阵
        gl.glMatrixMode(GL10.GL_MODELVIEW);
        gl.glLoadIdentity();//清空矩阵
        GLU.gluLookAt(gl, 0, 0, 3, 0, 0, 0, 0, 1, 0);//视点变换，将相机位置设置为(0, 0, 3)，同时指向(0, 0, 0)点

        //设置模型位置旋转及缩放信息
        gl.glTranslatef(0.0f, 0.0f, -1.0f);//将模型位置设置为(0, 0, -1)
        float angle = 30.0f;
        gl.glRotatef(angle, 0, 1, 0);//绕模型自身Y轴旋转30度
        gl.glRotatef(angle, 1, 0, 0);//绕模型自身X轴旋转30度
        gl.glScalef(1.2f, 1.2f, 1.2f);//设置三方向的缩放系数

        //设置颜色
        gl.glColor4f(0.0f, 0.0f, 0.0f, 1.0f);
        //渲染立方体
        mCube.draw(gl, gl.GL_TRIANGLES);
        //mCube.draw(gl, gl.GL_LINES);
    }
```





四：网络访问与服务

【目的】

掌握 Android 网络访问方法

【要求】

1、 了解手机 WEB 网站访问编程

2、 通过网络进行数据访问

3、 了解数据库使用

【原理】

利用 Android 网络访问协议

【过程】

1、 访问 WEB 网站，通过 HttpResponse 类，读入网络数据。

Android SDK 网络包：

包	描述
android.net	Android 网络访问 socket。该包包括 URI 类，不仅仅是传统的联网方面。
android.net.http	处理 Android 有关 Http 协议类。
android.net.wifi	在 Android 平台上管理有关 WiFi(802.11 无线 Ethernet)应用的类。
android.telephony.gsm	包含用于管理和发送 SMS(文本)消息的类。

读入 WEB 数据例程：

```
        HttpClient client = new DefaultHttpClient();
        HttpGet get = new HttpGet(url);
        HttpResponse response = client.execute(get);
        HttpEntity entity = response.getEntity();
        //尝试读取entity的长度，返回-1表示长度未知
        long length = entity.getContentLength();
        InputStream is = entity.getContent();
        String s = null;
        if (is != null) {
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            byte[] buf = new byte[512];
            int ch = -1;
            int count = 0;
            while ((ch = is.read(buf)) != -1) {
                baos.write(buf, 0, ch);
                count += ch;
                //如果长度已知，可以通过taskProgress()通知监听者任务执行的比例
                if (length > 0) {
                    listener.taskProgress(this, count, length);
                }
                //为了更好的演示进度，让线程休眠100ms
```

```
                Thread.sleep(100);
            }
            Log.e("HttpTask", "length=" + baos.toByteArray().length);
            //返回内容
            s = new String(baos.toByteArray());
        }
        return s;
```

读入 www.google.com 网站数据：

五：硬件访问与传感器

【目的】

通过底层 API 访问手机硬件及手机上的各种传感器

【要求】

1、 获取手机上电话、短信等各种功能的编程

2、 了解手机上各种传感器的功能与使用方法

【原理】

利用手机本身的功能与相关传感器的使用

【过程】

1、 了解程序使用手机电话功能的方法

短信收发：

◆发送短信只需要几行代码，如下：

```
SmsManager sms = SmsManager.getDefault();
PendingIntent pi = PendingIntent.getBroadcast(this, 0, new Intent(), 0);
sms.sendTextMessage(phoneNumber, null, MsgStr, pi, null);
```

其中参数 phoneNumber 和 MsgStr 均是 String 类型，表示接收方的电话号码和短信内容

◆接收短信主要是继承 BroadcaseReceiver 类 ，覆盖 onReceive 函数：

```
package com.android.TinySMS;

import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.gsm.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class TinySMS extends Activity {
    public static final String SMS_ACTION= "com.android.TinySMS.RESULT";
//  private TextView  message;
    private Button    snd;
    private EditText  tel;
    private EditText  txt;
    private SentReceiver receiver = new SentReceiver();

    private class SentReceiver extends BroadcastReceiver {
```

```
    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals(SMS_ACTION)) {
        int code = getResultCode();
        //短消息发送成功
        if(code == Activity.RESULT_OK)
            Toast.makeText(TinySMS.this, R.string.msg_sent,
                    Toast.LENGTH_SHORT).show();
        }
    }
    };

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    tel = (EditText) findViewById(R.id.EditText01);
    tel.setText("5554");  //模拟器之间互发短信
    txt = (EditText) findViewById(R.id.EditText02);
    txt.setText("我用自己的程序试试发短信。");
    snd = (Button) findViewById(R.id.Button01);

    snd.setOnClickListener(new View.OnClickListener() {
        public void onClick(View arg0) {
        String phoneNo = tel.getText().toString();
        String message = txt.getText().toString();
        if (phoneNo.length()>0 && message.length()>0){
            sendSMS(phoneNo, message);
        } else {
            Toast.makeText(TinySMS.this,
                "请重新输入电话号码和短信内容",
                Toast.LENGTH_LONG).show();
        }
        }
    });
    }

    private void sendSMS(String address, String content)
    {
    SmsManager manager = SmsManager.getDefault();
    Intent i = new Intent(SMS_ACTION);
    //生成PendingIntent，当消息发送完成，接收到广播
    PendingIntent sentIntent = PendingIntent.getBroadcast(
```

```
            this,
            0,
            i,
            PendingIntent.FLAG_ONE_SHOT);
    manager.sendTextMessage(
        address,
        null,
        content,
        sentIntent,
        null);
    }
}
```

如果要收发短信，还需在 AndroidManifest.xml 中声明权限：

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

2、手机上有多种传感器，可以对这些传感器进行编程。

Android SDK 中提供的面向硬件的特性

特性	描述
android.hardware.Camera	相机交互的类，可以截取照片、获取预览屏幕的图像，修改理相机操作的参数。
android.hardware.SensorManager	允许访问 Android 平台传感器的类。并非所有配备 Android 的设备都支持 SensorManager 中的所有传感器。
android.hardware.SensorListener	在传感器值实时更改时，希望接收更新的类要实现的接口。用以监视

	硬件中一个或多个可用传感器。
android.media.MediaRecorder	用于录制媒体的类。
android.FaceDetector	人脸进行基本识别类。
android.os.*	可以与操作环境交互的包，包括电源管理、文件查看器、处理器和消息类。

android.hardware.SensorManager 包含几个常量，这表示 Android 传感器系统的不同方面，包括：

传感器类型：方向、加速表、光线、磁场、临近性、温度等。采样率最快、游戏、普通、用户界面。

当应用程序请求特定的采样率时，其实只是对传感器子系统的一个提示，或者一个建议。不保证特定的采样率可用。准确性高、低、中、不可靠。

SensorListener 接口是传感器应用程序的中心。它包括两个必需方法：

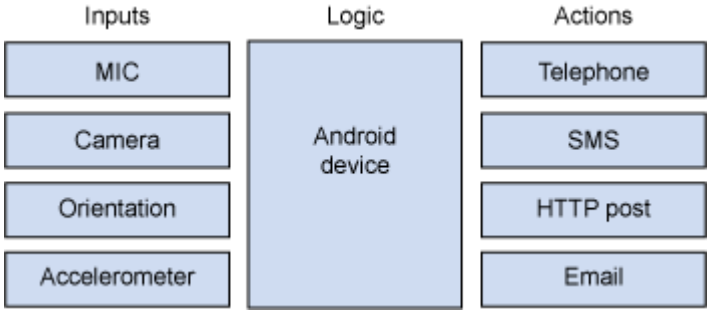
onSensorChanged(int sensor,float values[]) 方法在传感器值更改时调用。该方法只对受此应用程序监视的传感器调用。该方法的参数包括：

- 一个整数，指示更改的传感器；
- 一个浮点值数组，表示传感器数据本身。有些传感器只提供一个数据值，另一些则提供三个浮点值。方向和加速表传感器都提供三个数据值。

当传感器的准确性更改时，将调用 onAccuracyChanged(int sensor,int accuracy) 方法。参数包括两个整数：一个表示传感器，另一个表示该传感器新的准确值。

要与传感器交互，应用程序必须注册以侦听与一个或多个传感器相关的活动。注册使用 SensorManager 类的 registerListener 方法完成。

并非所有支持 Android 的设备都支持 SDK 中定义的所有传感器。



以 Android 为中心的传感器系统

相机拍摄：

```
package com.android.cameraAndroid;

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

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

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

```
import android.graphics.PixelFormat;
import android.hardware.Camera;
import android.hardware.Camera.Parameters;
import android.hardware.Camera.PictureCallback;
import android.hardware.Camera.ShutterCallback;
import android.media.AudioManager;
import android.media.ToneGenerator;
import android.net.Uri;
import android.os.Environment;
import android.os.StatFs;
import android.view.Menu;
import android.view.MenuItem;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public class CameraAndroid extends Activity {

    private CameraPreview preview;
    private Camera camera;
    private ToneGenerator tone;
    private static final int OPTION_SNAPSHOT = 0;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        preview = new CameraPreview(this);
        setContentView(preview);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int itemId = item.getItemId();
        switch(itemId){
        case OPTION_SNAPSHOT:
            //拍摄照片
            camera.takePicture(shutterCallback, null, jpegCallback);
            break;
        }
        return true;
    }
    //返回照片的 JPEG 格式的数据
    private PictureCallback jpegCallback = new PictureCallback(){

        public void onPictureTaken(byte[] data, Camera camera) {
```

```
            Parameters ps = camera.getParameters();
            if(ps.getPictureFormat() == PixelFormat.JPEG){
                //存储拍照获得的图片
                String path = save(data);
                //将图片交给 Image 程序处理
                Uri uri = Uri.fromFile(new File(path));
                Intent intent = new Intent();
                intent.setAction("android.intent.action.VIEW");
                intent.setDataAndType(uri, "image/jpeg");
                startActivity(intent);
            }
        }
    };

    //快门按下的时候 onShutter()被回调
    private ShutterCallback shutterCallback = new ShutterCallback(){
        public void onShutter() {
            if(tone == null)
                //发出提示用户的声音
                tone = new ToneGenerator(AudioManager.STREAM_MUSIC,
                        ToneGenerator.MAX_VOLUME);
            tone.startTone(ToneGenerator.TONE_PROP_BEEP2);
        }
    };

    private String save(byte[] data){
        String path = "/sdcard/"+System.currentTimeMillis()+".jpg";
        try {
            //判断 SD 卡上是否有足够的空间
            String storage = Environment.getExternalStorageDirectory().toString();
            StatFs fs = new StatFs(storage);
            long available = fs.getAvailableBlocks()*fs.getBlockSize();
            if(available<data.length){
                //空间不足直接返回空
                return null;
            }
            File file = new File(path);
            if(!file.exists())
                //创建文件
                file.createNewFile();
            FileOutputStream fos = new FileOutputStream(file);
            fos.write(data);
            fos.close();
        } catch (Exception e) {
```

```
            e.printStackTrace();
            return null;
        }
        return path;
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        menu.add(0, OPTION_SNAPSHOT, 0, R.string.snapshot);
        return super.onCreateOptionsMenu(menu);
    }

    class CameraPreview extends SurfaceView implements SurfaceHolder.Callback {
        SurfaceHolder mHolder;

        public CameraPreview(Context context) {
            super(context);
            mHolder = getHolder();
            mHolder.addCallback(this);
            mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
        }
        //Sureface 创建的时候，此方法被调用
        public void surfaceCreated(SurfaceHolder holder) {
            //打开摄像头，获得 Camera 对象
            camera = Camera.open();
            try {
                //设置显示
                camera.setPreviewDisplay(holder);
            } catch (IOException exception) {
                camera.release();
                camera = null;
            }
        }

        //Surface 销毁的时候，此方法被调用
        public void surfaceDestroyed(SurfaceHolder holder) {
            camera.stopPreview();
            //释放 Camera
            camera.release();
            camera = null;
        }

        public void surfaceChanged(SurfaceHolder holder, int format, int w,
                int h) {
```

```
                //已经获得 Surface 的 width 和 height，设置 Camera 的参数
                Camera.Parameters parameters = camera.getParameters();
                parameters.setPreviewSize(w, h);
                camera.setParameters(parameters);
                //开始预览
                camera.startPreview();
            }
        }
}
```

拍摄模拟：