# 单词翻译

|  |  |  |
| --- | --- | --- |
| 单词 | 发音 | 含义 |
| widget | [widʒit] | 窗口小部件 |
| theme | [θi:m] | 主题 |
| sensor |  | 传感器 |
| Morse |  | 摩尔斯式电码 |
| animation |  | 活泼 |
| chronometer |  | 计时器 |
| positive |  | 积极的 |
| negative |  | 消极的 |

# 主要类

## Activity

活动是最基本的Android 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个屏幕显示组成。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活动提供返回值--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。

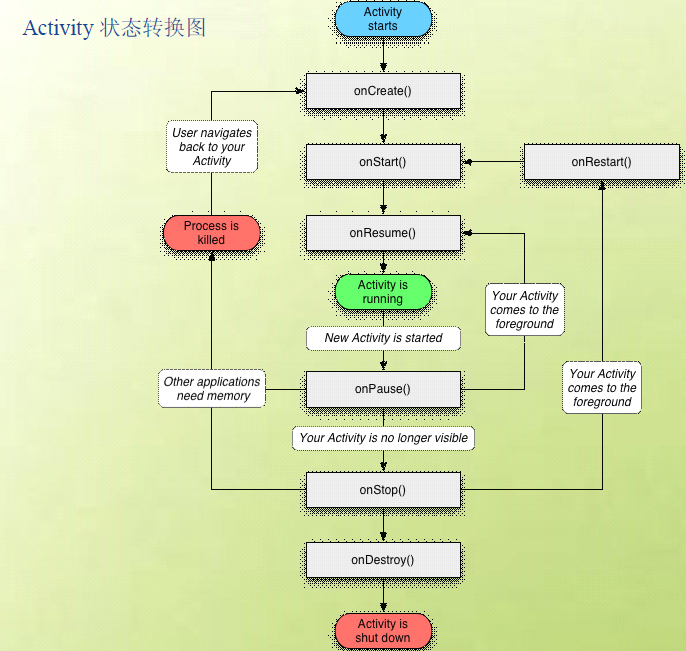
当一个新的屏幕打开后，前一个屏幕将会暂停，并保存在历史堆栈中。用户可以返回到历史堆栈中的前一个屏幕。当屏幕不再使用时，还可以从历史堆栈中删除。默认情况下，Android 将会保留从主屏幕到每一个应用的运行屏幕。

简单理解Activity 代表一个用户所能看到的屏幕，Activity 主要是处理一个应用的整体性工作，例如，监听系统事件(按键事件、触摸屏事件等)、为用户显示指定的View，启动其他Activity 等。所有应用的Activity都继承于android.app.Activity 类，该类是Android 提供的基层类，其他的Activity 继承该父类后，通过Override父类的方法来实现各种功能，这种设计在其他领域也较为常见。

### 生命周期

* 当一个Activity 在屏幕的最上层时（对堆栈的最顶端），它就是属于active 或者running 的状态。
* 如果一个Activity 失去焦点（focus）但还看得到它的画面（比如：一个新的Activity 画面并不是全屏幕或者它是一个半透明的情况），那失去焦点的Activity 则处在paused 的状态。像这个失去焦点的Activity它还是完全活着的，并没有消失。（活着的意思是指，Activity 自己本身所有的状态及数据都还是存在的，也跟窗口管理程序window manager 保持联系着），像这种paused 的Activity，会在一种情况下消失，那就是当系统的内存不够用之时，系统会自动判断，把不重要的Activity 移除。
* 如果一个Activity 被其它的Activity 完全的遮盖住时，它仍然保有全部的状态及数据，但因为它已不再被使用者看见，所以它的画面是被隐藏起来的（画面不需要更新），当系统内存不足时，这种stop 状态的Activity 时最先被系统考虑拿下来释放内存的。
* 当一个Activity 处于pause 或stop 的状态时，系统可以要求Activity 结束（finish）或直接移除（kill）它。当它需要再度呈现在使用者面前时，它必须要能完整的重新启动及回复先前的状态。

### 状态转换图



## Intent

调用Android 专有类Intent 进行架构屏幕之间的切换。Intent 是描述应用想要做什么。Intent 数据结构两个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。

Android 使用了Intent 这个特殊类，实现在屏幕与屏幕之间移动。Intent 类用于描述一个应用将会做什么事。在Intent 的描述结构中，有两个最重要的部分：动作和动作对应的数据。典型的动作类型有：M AIN（activity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。相对于intent 是一个有效的做某事的请求，一个intentfilter 则用于描述一个activity（或者IntentReceiver）能够操作哪些intent。一个activity 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，从一个屏幕导航到另一个屏幕是很简单的。当向前导航时，activity 将会调用startActivity(IntentmyIntent)方法。然后，系统会在所有安装的应用程序中定义的IntentFilter 中查找，找到最匹配myIntent 的Intent 对应的activity。新的activity 接收到myIntent 的通知后，开始运行。当startActivity 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：

A、Activities 能够重复利用从其它组件中以Intent 的形式产生的一个请求；

B、Activities 可以在任何时候被一个具有相同IntentFilter 的新的Activity 取代。

### 使用Intent启动另一个Activity

|  |
| --- |
| Intent intent = **new** Intent();  intent.setClass(MytTestActivity.**this**, NewActivity.**class**);  //MytTestActivity.this本activity的实例，NewActivity.class要打开的acitvity  startActivity(intent); |

### 通过Intent向另一个Activity传参、收参

#### 传参

|  |
| --- |
| Intent intent = **new** Intent();  Bundle bundle = **new** Bundle(); //通过bundle传参数  bundle.putString("userName", "text.text");  intent.putExtras(bundle);  intent.putExtra("userName2", "刘祖刚"); //通过intent直接传参数  intent.setClass(MytTestActivity.**this**, NewActivity.**class**);  startActivity(intent); |

#### 收参

|  |
| --- |
| Bundle bundle = **this**.getIntent().getExtras();//从bundle接收参数  String userName = bundle.getString("userName");  //从intent接收参数  String userName2 = **this**.getIntent().getStringExtra("userName2"); |

### 子UI的结果返回到父UI

#### 父UI

* **打开子UI时的特殊处理**

|  |
| --- |
| button2.setOnClickListener(**new** OnClickListener(){    @Override  **public** **void** onClick(View v) {  Intent intent = **new** Intent();  intent.setClass(MytTestActivity.**this**, NewActivity.**class**);  startActivityForResult(intent, *REQUEST\_ASK*);//关键语句  }  } |

* **子UI结果的处理**

|  |
| --- |
| @Override  **protected** **void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  **super**.onActivityResult(requestCode, resultCode, data); //必须执行  **if**(requestCode == *REQUEST\_ASK*){ // REQUEST\_ASK自定义，用于处理不同业务的区别码  **if**(resultCode == *RESULT\_CANCELED*){ //与子UI中的setResult方法中的resultCode对应  textView1.setText("Cancel\*\*\*\*");  }**else** **if**(resultCode == *RESULT\_OK*){ //与子UI中的setResult方法中的resultCode对应  String userName =data.getStringExtra("userName");//从bundle中获得相应数据  textView1.setText("the name get from the second layout:\n"+userName);  }  }  } |

#### 子UI

|  |
| --- |
| @Override  **public** **void** onClick(View v) {  **if**(v == **this**.okBtn){  String userName = **this**.editText.getText().toString();  intent.putExtra("userName", userName);  **this**.setResult(**this**.*RESULT\_OK*, intent);  //RESULT\_OK与父UI的方法onActivityResult中的resultCode相对应  }**else** **if**(v == **this**.canceleBtn){  **this**.setResult(**this**.*RESULT\_CANCELED*, intent);  //RESULT\_CANCELED与父UI的方法onActivityResult中的resultCode相对应  }  **this**.finish();//  } |

## Intent Filter

当Intent 要求做某件事时，IntentFilter 被用来描述这个Activity 能够做些什么事情。比如一个Activity要能够显示个人联络数据，你就必需要在intentFilter 说明你要如何处理个人联络数据， 并用ACTION\_VIEW 呈现出来。IntentFilter 都会在AndroidManifest.xml 清单里面声明。

## IntentReceiver

当你希望你的应用能够对一个外部的事件(如当电话呼入时，或者数据网络可用时，或者到了晚上时)做出响应，你可以使用一个IntentReceiver。虽然IntentReceiver 在感兴趣的事件发生时，会使用NotificationManager通知用户，但它并不能生成一个UI。IntentReceiver 在AndroidManifest.xml 中注册，但也可以在代码中使用Context.registerReceiver()进行注册。当一个intentreceiver 被触发时，你的应用不必对请求调用intentreceiver，系统会在需要的时候启动你的应用。各种应用还可以通过使用Context.broadcastIntent()将它们自己的intentreceiver 广播给其它应用程序。

## Service

**一个Service 是一段长生命周期的，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activity，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activity，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activity 会使用Context.startService()来启动一个service，从而可以在后台保持音乐的播放。同时，系统也将保持这个service 一直执行，直到这个service 运行结束。另外，我们还可以通过使用Context.bindService()方法，连接到一个service 上（如果这个service 还没有运行将启动它）。当连接到一个service 之后，我们还可以service 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。

## Content Provider

Android 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，内容提供器就可以发挥作用了。因为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。

数据是应用的核心。在Android 中，默认使用鼎鼎大名的SQLite 作为系统DB。但是在Android 中，使用方法有点小小的不一样。在Android 中每一个应用都运行在各自的进程中，当你的应用需要访问其他应用的数据时，也就需要数据在不同的虚拟机之间传递，这样的情况操作起来可能有些困难(正常情况下，你不能读取其他的应用的db 文件)，ContentProvider 正是用来解决在不同的应用包之间共享数据的工具。

1. 所有被一个Android 应用程序创建的偏好设置，文件和数据库都是私有的。
2. 为了和其他应用程序共享数据，应用程序不得不创建一个Content Provider
3. 要回索其他应用程序的数据，它自己的Content Provider 必须被调用
4. Android 本地Content Provider 包括：

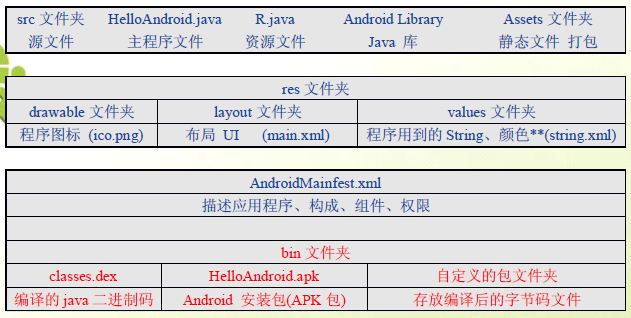
CallLog：地址和接收到的电话信息

Contact.People.Phones：存储电话号码

Setting.System：系统设置和偏好设置

等等

# 项目结构



# 视图控件

## EditText

## Button

### 增加点击事件

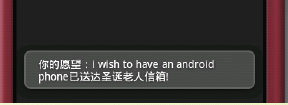
|  |
| --- |
| button\_Login.setOnClickListener(**new** OnClickListener(){  @Override  **public void** onClick(View v) {  show\_Login\_TextView.setText(getNameEditText.getText()+"欢迎您进入");  }  }); |

## TextView

## Toast（提示信息）

|  |
| --- |
| Toast.*makeText*( MytTestActivity.**this**, editText.getText().toString(), Toast.*LENGTH\_LONG*).show(); |

例如：



## AlertDialog.Builder

|  |
| --- |
| /\*新建一个AlertDialog.Builder对象\*/  AlertDialog.Builder my\_ADialog = **new** AlertDialog.Builder(MytTestActivity.**this**);  /\*设置标题\*/  my\_ADialog.setTitle("Android 提示");  /\*设置显示消息\*/  my\_ADialog.setMessage("AlertDialog.Builder提示对话框消息!!");  /\*显示\*/  my\_ADialog.show(); |



## CheckBox

checkBox.isChecked()

## RadioGroup、RadioButton

|  |
| --- |
| /\*给单RadioGroup添加状态改变监听器\*/  radioGroup.setOnCheckedChangeListener(**new** RadioGroup.OnCheckedChangeListener(){  @Override  **public** **void** onCheckedChanged(RadioGroup group, **int** checkedId) {  // **TODO** Auto-generated method stub  **if**(boy\_RadioButton.isChecked()){  answer\_TextView.setText(R.string.iam\_Boy);  }**else**{  answer\_TextView.setText(R.string.iamGirl);  }  }  });  radioButton.isChecked(); |

## Spinner（自定义下拉菜单）

|  |
| --- |
| /\* 自定义下拉菜单中的值 \*/  String[] countriesStr = { " 北京市", " 上海市", " 天津市", " 重庆市" };  ArrayAdapter<String> adapter = **new** ArrayAdapter<String>(**this**, android.R.layout.*simple\_spinner\_item*, countriesStr);  /\* 定义自定义下拉菜单样式 \*/  adapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);  //android.R.layout.simple\_spinner\_dropdown\_item  //android.R.layout.simple\_spinner\_item  /\* 将ArrayAdapter 添加Spinner 对象中 \*/  spinner.setAdapter(adapter); |

如果countriesStr是List，则可以通过下面代码，向下拉菜单中加元素。

|  |
| --- |
| adapter.add(addString);  /\*获取刚刚添加进的字符串位置\*/  **int** post = adapter.getPosition(addString);  /\*设置刚刚添加进的下拉菜单内容被选中\*/  spinner.setSelection(post); |

## AutoCompleteTextView，MultiAutoCompleteTextView（自动完成输入框）

* **布局代码**

|  |
| --- |
| <AutoCompleteTextView  android:id=*"@+id/AutoCompleteTextView\_input"*  android:layout\_width=*"275px"*  android:layout\_height=*"wrap\_content"*  android:layout\_x=*"23px"*  android:layout\_y=*"98px"*  android:text=*""*  android:textSize=*"18sp"* >  </AutoCompleteTextView> |

* **程序代码**

|  |
| --- |
| String[] normalString = **new** String[] { "Android", "Android Blog",  "Android Market", "Android SDK", "Android AVD", "BlackBerry",  "BlackBerry JDE", "Symbian", "Symbian Carbide", "Java 2ME",  "Java FX", "Java 2EE", "Java 2SE", "Mobile", "Motorola", "Nokia",  "Sun", "Nokia Symbian", "Nokia forum", "WindowsMobile", "Broncho","Windows XP", "Google", "Google Android ", "Google 浏览器", "IBM", "MicroSoft", "Java", "C++", "C", "C#", "J#", "VB" };  /\* 从XML中获取UI元素对象 \*/  autoCompleteTextView = (AutoCompleteTextView) findViewById(R.id.*AutoCompleteTextView\_input*);  /\* 实现一个适配器对象，用来给自动完成输入框添加自动装入的内容 \*/  ArrayAdapter arrayAdapter = **new** ArrayAdapter<String>(**this**, android.R.layout.*simple\_dropdown\_item\_1line*, normalString);  /\* 给自动完成输入框添加内容适配器 \*/  autoCompleteTextView.setAdapter(arrayAdapter); |



## TabActivity

## AlertDialog(各种类型的对话框)

App/Dialog

com.example.android.apis.app.AlertDialogSamples

### 注意

**必须通过重写方法protected Dialog onCreateDialog(int id) {}来实现以下功能，并且必须通过this.showDialog(1);方法调用。**

### 确定、取消、短信息

|  |
| --- |
| new AlertDialog.Builder(AlertDialogSamples.this)  .setIcon(R.drawable.alert\_dialog\_icon)  .setTitle(R.string.alert\_dialog\_two\_buttons\_title)  .setPositiveButton(R.string.alert\_dialog\_ok, new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int whichButton) {  /\* User clicked OK so do some stuff \*/  }  })  .setNegativeButton(R.string.alert\_dialog\_cancel, new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int whichButton) {  /\* User clicked Cancel so do some stuff \*/  }  })  .create(); |

### 确定、取消、长信息

|  |
| --- |
| new AlertDialog.Builder(AlertDialogSamples.this)  .setIcon(R.drawable.alert\_dialog\_icon)  .setTitle(R.string.alert\_dialog\_two\_buttons\_msg)  .setMessage(R.string.alert\_dialog\_two\_buttons2\_msg)  .setPositiveButton(R.string.alert\_dialog\_ok, new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int whichButton) {  /\* User clicked OK so do some stuff \*/  }  })  .setNeutralButton(R.string.alert\_dialog\_something, new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int whichButton) {  /\* User clicked Something so do some stuff \*/  }  })  .setNegativeButton(R.string.alert\_dialog\_cancel, new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int whichButton) {  /\* User clicked Cancel so do some stuff \*/  }  })  .create(); |

### 列表

|  |
| --- |
| new AlertDialog.Builder(AlertDialogSamples.this)  .setTitle(R.string.select\_dialog)  .setItems(R.array.select\_dialog\_items, new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int which) {  /\* User clicked so do some stuff \*/  String[] items = getResources().getStringArray(R.array.select\_dialog\_items);  new AlertDialog.Builder(AlertDialogSamples.this)  .setMessage("You selected: " + which + " , " + items[which])  .show();  }  })  .create(); |

R.array.select\_dialog\_items：数组或list

### 单选

|  |
| --- |
| new AlertDialog.Builder(AlertDialogSamples.this)  .setIcon(R.drawable.alert\_dialog\_icon)  .setTitle(R.string.alert\_dialog\_single\_choice)  .setSingleChoiceItems(R.array.select\_dialog\_items2, 0, new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int whichButton) {  /\* User clicked on a radio button do some stuff \*/  }  })  .setPositiveButton(R.string.alert\_dialog\_ok, new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int whichButton) {  /\* User clicked Yes so do some stuff \*/  }  })  .setNegativeButton(R.string.alert\_dialog\_cancel, new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int whichButton) {  /\* User clicked No so do some stuff \*/  }  })  .create(); |

### 多选

|  |
| --- |
| new AlertDialog.Builder(AlertDialogSamples.this)  .setIcon(R.drawable.ic\_popup\_reminder)  .setTitle(R.string.alert\_dialog\_multi\_choice)  .setMultiChoiceItems(R.array.select\_dialog\_items3,  new boolean[]{false, true, false, true, false, false, false},  new DialogInterface.OnMultiChoiceClickListener() {  public void onClick(DialogInterface dialog, int whichButton,  boolean isChecked) {  /\* User clicked on a check box do some stuff \*/  }  })  .setPositiveButton(R.string.alert\_dialog\_ok, new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int whichButton) {  /\* User clicked Yes so do some stuff \*/  }  })  .setNegativeButton(R.string.alert\_dialog\_cancel, new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int whichButton) {  /\* User clicked No so do some stuff \*/  }  })  .create(); |

### 其它布局（UI）

|  |
| --- |
| LayoutInflater factory = LayoutInflater.from(this); //获取工厂类  final View textEntryView = factory.inflate(R.layout.alert\_dialog\_text\_entry, null); //通过工厂类获取布局  new AlertDialog.Builder(AlertDialogSamples.this)  .setIcon(R.drawable.alert\_dialog\_icon)  .setTitle(R.string.alert\_dialog\_text\_entry)  .setView(textEntryView)  .setPositiveButton(R.string.alert\_dialog\_ok, new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int whichButton) {  /\* User clicked OK so do some stuff \*/  }  })  .setNegativeButton(R.string.alert\_dialog\_cancel, new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int whichButton) {  /\* User clicked cancel so do some stuff \*/  }  })  .create(); |

# 文件读写

## 读文件

* Context.openFileInput(String name)打开一个与应用程序联系的私有文件输入流
* 当文件不存在时抛出FileNotFoundException 异常

|  |
| --- |
| FileInputStream in = **this**.openFileInput("test2.txt");//打开文件"test2.txt"  ……  in.close();//关闭输入流 |

## 写文件

* Context.openFileOutput(String name,int mode)开启一个与应用程序联系的私有文件输出流
* 当文件不存在时该文件将被创建
* 文件输出流可以在添加模式中打开，这意味新的数据将被添加到文件的末尾

|  |
| --- |
| FileOutputStream out = **this**.openFileOutput("test2.txt",*MODE\_APPEND*);  //打开文件"test2.txt"进行写操作、使用MODE\_APPEND 在添加模式中打开文件  out.write(“……”);  out.close();//关闭输出流 |

## 读取静态文件

* 要打开打包在应用程序中的静态文件，使用Resources.openRawResource(R.raw.mydatafile)
* 该文件必须放在文件夹res/raw/中

|  |
| --- |
| InputStream in = **this**.getResources().openRawResource(R.raw.my);  … //获得Context资源  in.close();//关闭输入流 |

# 数据库操作（SQLite）

## 创建数据库

* Context.createDatabase(String name,int version ,int mode,CursorFactory factory)创建一个新的数据库并返回一个SQLiteDatabase 对象
* 假如数据库不能被创建，则抛出FileNotFoundException 异常
* 新创建SQLite 数据库方法

|  |
| --- |
| SQLiteDatabase mydataBase=SQLiteDatabase.*create*(**new** CursorFactory(){  //创建一个数据库  //工厂类，一个可选工厂类，当查询时调用来实例化一个光标  @Override  **public** Cursor newCursor(SQLiteDatabase db,  SQLiteCursorDriver masterQuery, String editTable,  SQLiteQuery query) {  **return null**;  }  }); |

|  |
| --- |
| //创建新的数据库，名称myDatabase，模式MODE\_PRIVATE，鼠标工厂  //工厂类，一个可选工厂类，当查询时调用来实例化一个光标  SQLiteDatabase myDataBase = this.openOrCreateDatabase("myDataBase.db",  MODE\_PRIVATE, new CursorFactory(){  @Override  public Cursor newCursor(SQLiteDatabase db,  SQLiteCursorDriver masterQuery, String editTable,  SQLiteQuery query) {  return null;  }  }); |

## 删除数据库

* Context.deleteDatabase(String name)删除指定名称的数据库
* 假如数据库成功删除则返回true，失败则为false(例如数据库不存在)

|  |
| --- |
| //删除指定名称的数据库  **this**.deleteDatabase("myDatabase.db"); |

## 打开数据库

* Context.openDatabase(String file,CursorFactory factory) 打开一个存在的数据库并返回一个SQLiteDatabase 对象
* 如果数据库不存在则抛出FileNotFoundException 异常

|  |
| --- |
| //创建一个名为：myDataBase的数据库，后缀为.db  SQLiteDatabase my\_DataBase=**this**.openOrCreateDatabase("myDateBase.db",  *MODE\_PRIVATE*, **null**);  my\_DataBase.close();//不要忘记关闭数据库 |

## 非查询SQL指令

* SQLiteDatabase.execSQL(String sql)可以用来执行非查询SQL 指令，这些指令没有结果。包括：CREATE TABLE / DROP TABLE / INSERT 等等

### 创建表

|  |
| --- |
| //创建一个名为"test"并带两个参数的表  my\_DataBase.execSQL("CREATE TABLE test (\_id INTEGER PRIMARY KEY,  someNumber INTERGER);"); |

### 插入数据

|  |
| --- |
| //在数据库中插入一个元组  my\_DataBase.execSQL("INSERT INTO test (\_id,someNumber) values(1,8);"); |

### 删除表

|  |
| --- |
| //删除表  my\_DataBase.execSQL("DROP TABLE test"); |

## 查询SQL指令-游标Cursors

* Android 使用游标(Cursors)来导航浏览查询结果；
* 游标(Cursors)被android.database.Cursor 对象来描述；
* 一个游标(Cursors)是一个简单的指针，它从查询结果的一个元组跳到下一个元组(或是前一个或是第一个或是……)；
* 游标(Cursors)在它定位位置的那一刻返回元组数据；

|  |
| --- |
| //为了创建一个Cursor(游标)，必须执行一个查询，要么通过SQL使用rawQuery()方法  //或是更精心设计的方法，像query()方法  Cursor cur=my\_DataBase.rawQuery("SELECT \* FORM test", **null**);  **if**(cur!=**null**){//游标不为空  //返回给定名称的列的基于0开始的index，如果该属性列不存在则返回-1  //通过它们的index来检索属性值  **int** numColumn=cur.getColumnIndex("someNumber");  **if**(cur.moveToFirst()){  //cur.moveToFirst()让游标指向第一行，如果游标指向第一行，则返回true  **do** {  **int** num=cur.getInt(numColumn);//获得当前行该属性的值  /\*Cursor提供了不同的方法来回索不同的数据类型  例如getInt(int index)/getString(int index)等等\*/  /\*做一些事情\*/  } **while** (cur.moveToNext());  /\*游标移动到下一行，如果游标已经通过了结果集中的最后，  即没有行可以移动时，则返回false\*/  //其他可能移动的是previous() 和first()方法  }  } |

# 常用代码片段

## 取得屏幕大小

|  |
| --- |
| DisplayMetrics displaysMetrics=**new** DisplayMetrics();  //DisplayMetrics 一个描述普通显示信息的结构，例如显示大小、密度、字体尺寸  getWindowManager().getDefaultDisplay().getMetrics(displaysMetrics);  //getManager()获取显示定制窗口的管理器。  //获取默认显示Display对象  //通过Display 对象的数据来初始化一个DisplayMetrics 对象  displaysMetrics.widthPixels; //宽  displaysMetrics.heightPixels; //高 |

## 隐藏标题

**注：在Activity.setContentView();之前调用此方法。**

|  |
| --- |
| **private** **void** HideTitle() {  requestWindowFeature(Window.*FEATURE\_NO\_TITLE*);  } |

## 隐藏状态栏

**注：在Activity.setContentView();之前调用此方法。**

|  |
| --- |
| **private** **void** HideStatusBar() {  //定义全屏参数  **int** flag = WindowManager.LayoutParams.*FLAG\_FULLSCREEN*;  //获得窗口对象  Window myWindow=**this**.getWindow();  //设置Flag标识  myWindow.setFlags(flag,flag);  } |