# 单元测试：Junit

* 模拟器要打开/真机要连接

## 创建一个类，继承android.test.AndroidTestCase类

|  |
| --- |
| **import** android.test.AndroidTestCase;  **public** **class** TestCase **extends** AndroidTestCase {  **public** **void** test() {  System.*out*.println("case");  }  } |

## 还要在清单文件中，指定指令集和使用的类库

一个测试框架，要测试哪个项目。跟他定义在哪个项目中，是没有关系的

|  |
| --- |
| <manifest  <instrumentation  android:name=*"android.test.InstrumentationTestRunner"*  android:targetPackage=*"com.example.junit"*  ></instrumentation>  <application  <uses-library android:name=*"android.test.runner"*/>  </application>  </manifest> |

# .SQLite

* 在Android中使用的是SQLite数据库，因为它体积小
* SQLite是不检测数据类型的，因为它里面全都是字符串。在创建表示声明好数据类型，是因为方便查看

## 创建数据库

* 数据库是创建在内部存储空间的:data\data\应用包名\databases
* 需要定义一个类，继承android.database.sqlite.SQLiteOpenHelper对象。并且实现它的三个抽象方法
* 在junit测试框架中，测试SQLite数据库创建

## 在数据库被创建时创建表

* 在SQLiteOpenHelper#onCreate()方法中，创建表
* 在SQLite中，主键有个规范的写法：\_id

## 往数据库表中插入数据

* 在TestCase2#insert()，插入数据
* 因为SQLite不会检测数据类型，所以插入时的参数，无论是什么类型的。都会当做字符串插入
* 但是 ，如果在Integer类型的字段中，插入个ABC之类的。插入之后，显示的会是0，

因为SQLite虽然不会检测数据类型，但是也不会让你插入错误的类型

## 在数据库表中删除数据

* 在TestCase2#delete()，删除数据

## 在数据库表中修改数据

* 在TestCase2#update()，修改数据

## 在数据库表中查询数据

* 在TestCase2#select()，查询数据

## 使用api进行增删改查

### 增：在TestCase3#insertApi()，插入数据

### 删：在TestCase3#deleteApi()，删除数据

### 改：在TestCase3#updateApi()，修改数据

### 查：在TestCase3#selectApi()，查询数据

## 事务：在TestCase3#transaction()，进行事务操作

# .把数据库查询出来的数据显示在屏幕上

## 布局：activity\_main.xml

|  |
| --- |
| <ScrollView  xmlns:android=*"http://schemas.android.com/apk/res/android"*  xmlns:tools=*"http://schemas.android.com/tools"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*  >    <LinearLayout  android:id=*"@+id/ll"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*  tools:context=*".MainActivity"*  android:orientation=*"vertical"* >    </LinearLayout>  </ScrollView> |

## 在测试框架中，创建出50条记录到数据库

## 查询出数据并以列表的形式显示在窗口中

# 使用ListView显示数据

* + 用来显示一行一行的条目的
  + MVC结构：
    - M：model模型层，要显示的数据           ————people集合
    - V：view视图层，用户看到的界面          ————ListView
    - C：control控制层，操作数据如何显示     ————adapter对象，一般自己定义个类继承BaseAdapter
* 每一个条目都是一个View对象
* 如果屏幕在没滑动的情况下，只能显示17个条目。那么就只会先创建17个view对象，

滑动只后，上面看不见的view就会被缓存。下面就会创建一个新的view，再显示出来。

当屏幕往回滑动，被缓存好的哪个view就会在调用getview方法的时候被传递过去

* 屏幕上能显示多少个条目，getView方法就会被调用多少次，屏幕向下滑动时，getView会继续被调用，创建更多的View对象显示至屏幕
* 当条目划出屏幕时，系统会把该条目缓存至内存，当该条目再次进入屏幕，系统在重新调用getView时会把缓存的条目作为convertView参数传入，但是传入的条目不一定是之前被缓存的该条目，即系统有可能在调用getView方法获取第一个条目时，传入任意一个条目的缓存

## ListView的优化

* 缓存的优化：如果不存在缓存，就需要填充新的当做条目的布局文件。如果存在缓存，就不需要重新填充，直接拿过来用即可
* FindViewById()的优化：创建一个内部类，一般名为ViewHolder，把用来填充布局文件中的所有组件，当做属性装进ViewHolder中。

在没有缓存的情况下，把holder用布局文件填充成的对象的setTag(obj)方法，把holder装进布局文件填充成的对象中。在有缓存时，

直接拿缓存的布局文件对象来用时，再把holder对象取出来getTag()。这样就可以直接拿到hodler里面的属性来用了。减少了查找组件的操作

## ArrayAdapter

* 只能处理一种类型的数据，所以他可以指定一个泛型

## SimpleAdapter

* 比ArrayAdapter强大些，可以处理多种数据

# 对话框

* + 不能直接new，需要使用android.app.AlertDialog.Builder，构建器来构建

## 确定/取消对话框

* + 确定时的事件侦听器：android.content.DialogInterface.OnClickListener

## 单选对话框

* 事件侦听器：android.content.DialogInterface.OnClickListener

## 多选对话框

* 事件侦听器：android.content.DialogInterface.OnMultiChoiceClickListener;