宝贵建议请发送至：wangzhenyang@itcast.cn

黑马程序员 itheima.com -编程，始于黑马

Android 课程同步笔记

Alpha 0.01 版

Android-SQLite&ListView&常用控件

1.Android SQLite 数据库（★★★）

SQLite，是一款轻量型的数据库，是遵守 ACID(原子性、一致性、隔离性、持久性)的关联式数据库管理系统，多用于嵌入式开发中。

SQLite 数据库是无类型的，可以向一个 integer 的列中添加一个字符串，但它又支持常见的类型比如: NULL, VARCHAR, TEXT, INTEGER, BLOB, CLOB 等.

Tips：唯一的例外：integer primary key 此字段只能存储 64 位整数。

1.1 Android 中如何管理数据库

在 Android 系统中，提供了一个 SQLiteOpenHelper 抽象类，该类用于对数据库版本进行管理。

该类中常用的方法:

- onCreate 数据库创建时执行(第一次连接获取数据库对象时执行)
- onUpgrade 数据库更新时执行(版本号改变时执行)
- onOpen 数据库每次打开时执行(每次打开数据库时调用，在 onCreate，onUpgrade 方法之后)

Tips：由于 SQLiteOpenHelper 是一个抽象类，所以我们在用的时候可以自定义一个 SQLiteOpenHelper 类的子类来对数据库进行操作。

下面通过一个案例演示 SQLiteOpenHelper 的用法。创建一个 Android 工程。为该工程添加 Android Junit 测试环境（注意：关于 Android Junit 的知识在本人的第二篇笔记中有详细的说明）。

创建一个 PersonOpenHelper 类，继承 SQLiteOpenHelper 抽象类，重写 onCreate 和 onUpgrade 方法。

代码清单如下：

```
public class PersonOpenHelper extends SQLiteOpenHelper {
    /**
     *
     * @param context 上下文对象
     * @param name 数据库名称
     * @param factory 游标结果集工厂，如果需要使用则需要自定义结果集工厂，
null 值代表使用默认结果集工厂
     * @param version 数据库版本号，必须大于等于 1
     */
    public PersonOpenHelper(Context context, String name,
CursorFactory factory, int version) {
        super(context, name, factory, version);
    }
/**
     * 数据库第一次被创建时调用该方法，这里面主要进行对数据库的初始化操作
     */
    public void onCreate(SQLiteDatabase db) {
        // 数据库第一次被创建的时候执行如下 sql 语句创建一个 person 表
        db.execSQL("create table person(id integer primary key
autoincrement, name varchar(20), phone varchar(20), money
integer(20),age integer(10));");
    }
/**
     * 数据库更新的时候调用该方法
     * @param db 当前操作的数据库对象
     * @param oldVersion 老版本号
     * @param enwVersion 新版本号
     */
```

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        // 数据库的版本更新的时候执行
        if (oldVersion == 1 && newVersion == 2) {
            db.execSQL("alter table person add column balance integer");
        }
    }

}
```

创建一个 PersonOpenHelperTest 类，用于测试上面的代码:

```
public class PersonOpenHelperTest extends AndroidTestCase {

    public SQLiteDatabase getDataBase(){
        PersonOpenHelper helper = new PersonOpenHelper(getContext(),
"person.db", null, 1);
        SQLiteDatabase writableDatabase =
helper.getWritableDatabase();
        return writableDatabase;
    }
}
```

执行完上面代码后，通过 DDMS，查看/data/data/com.itheima.sqlite/databases 目录，发现产生了两个文件，person.db 和 person.db-journal。其中第一个文件就是我们的数据库文件。第一次操作数据库时，person.db-journal 文件会被自动创建，该文件是 sqlite 的一个临时的日志文件，主要用于 sqlite 数据库的事务回滚操作了。 但是 Android 系统中将该文件永久的保存在磁盘中，不会被自动清除的，如果没有操作异常或者不需要事务回滚时，此文件的大小为 0。这种机制避免了每次生成和删除 person.db-journal 文件的开销。

1.2 使用 SQLiteDatabase 操作数据库

SQLiteDatabase (★★★)

Android 提供了一个名为 SQLiteDatabase 的类，该类封装了一些操作数据库的 API，使用该类可以完成对数据进行添加(Create)、查询(Retrieve)、更新(Update)和删除(Delete)操作（这些操作简称为 CRUD）。

常用方法：

- execSQL()：可以执行 insert、delete、update 和 CREATE TABLE 之类有更改行为的 SQL 语句。
- rawQuery()：用于执行 select 语句。

游标结果集 Cursor (★★★)

Cursor是结果集游标，用于对结果集进行随机访问。

Cursor与JDBC中的ResultSet作用很相似。

Cursor中维护一个行索引一个列索引，游标中本身没有数据，它只是指向数据库的索引，模拟一个行、列的表结构。其起始位置是在-1的位置上的。

常用方法：

- moveToNext():将游标从当前行移动到下一行，如果已经移过了结果集的最后一行，返回结果为false，否则为true。
- moveToPrevious():用于将游标从当前行移动到上一行，如果已经移过了结果集的第一行，返回值为false，否则为true。

- moveToFirst():用于将游标移动到结果集的第一行，如果结果集为空，返回值为false，否则为true。
- moveToLast():用于将游标移动到结果集的最后一行，如果结果集为空，返回值为false，否则为true。

1.3 执行 SQL 语句操作数据库

执行 SQL 语句来操作数据库有两种方式，拼串和使用占位符”?。使用占位符”?”来执行 SQL 语句能够防止 SQL 注入攻击。

- 拼串方式使用的方法：

  execSQL(String sql)：增、删、改。

  Cursor rawQuery(String sql, String[] selectionArgs): 查询(拼串方式，第二个参数传 null 即可)。

- 占位符”?”使用的方法：

  void execSQL(String sql, Object[] bindArgs)。

  Cursor rawQuery(String sql, String[] selectionArgs)。

Tips：第二个参数对应的是第一个参数中占位符”?”对应的数据数组

下面通过一个案例来演示通过执行 SQL 实现对数据的增删改查操作，这里使用本文档中 1.1 章节中的工程（★★★★）。

在本文档1.1章节中，我们继续使用PersonOpenHelperTest类，在该测试类中添加如下方法：

```
//插入一条数据
public void insert(){
	//获取数据库对象，该方法在本文档 1.1 章节中已经给出具体代码
	SQLiteDatabase dataBase = getDataBase();
	String sql = "insert into person(name,age,phone)
values(?,?,?)";
	//执行 sql 语句
	dataBase.execSQL(sql,new
Object[]{"lisi","22","13240217764"});
	//关闭数据库
	dataBase.close();
   }
//查询单个数据
public void query(){
	SQLiteDatabase dataBase = getDataBase();
	String sql = "select name,age,phone from person where name=?";
	//执行 rawQuery 查询，返回 Cursor 对象
	Cursor cursor = dataBase.rawQuery(sql , new
String[]{"zhangsan"});
	Person person = new Person();
	//如果游标还有下一个元素，跟我们集合中 Iterator 中 hasNext()方法类
似
	while(cursor.moveToNext()){
	   //获取当前游标的第 0 个元素，元素是从 0 开始的，而不是 1
	   String name = cursor.getString(0);
	   //也可以通过列名来查询该字段在游标中的位置
	   int age = cursor.getInt(cursor.getColumnIndex("age"));
	   String phone = cursor.getString(2);
	   person.setName(name);
	   person.setAge(age);
	   person.setPhone(phone);
	}
	//关闭游标
	cursor.close();
	System.out.println(person);
   }
```

```
//更新数据
public void update(){
      SQLiteDatabase dataBase = getDataBase();
      String sql = "update person set age=? where name=?";
      //将 zhangsan 的年龄修改为 18
      dataBase.execSQL(sql,new String[]{"18","zhangsan"});
      dataBase.close();
   }
//查询所有数据
public void queryAll(){
      SQLiteDatabase dataBase = getDataBase();
      List<Person> persons = new ArrayList<Person>();
      String sql = "select name,age,phone from person";
      Cursor cursor = dataBase.rawQuery(sql , null);
      while(cursor.moveToNext()){
         String name = cursor.getString(0);
         int age = cursor.getInt(1);
         String phone = cursor.getString(2);
         Person p = new Person();
         p.setAge(age);
         p.setName(name);
         p.setPhone(phone);
         persons.add(p);
      }
      //关闭游标
      cursor.close();
      //输出集合中的数据
      for(Person p : persons){
         System.out.println(p);
      }
```

注意：在上面例子各个方法中，为了方便 Android Junit 测试，因此我把数据“写死”在代码里了，真正开发中是不会这么设计的。

1.4 使用 SQLiteDatabase 自带的增删改查方法

SQLiteDatabase 专门提供了对应于添加（insert）、删除（delete）、更新（update）、

查询（query）的操作方法。

这些方法实际上是给那些不太了解 SQL 语法的开发者使用的（对我来讲直接通过 sql 操作数据库反而要简单的多，这两种操作数据库的方式要求都会，开发过程中用什么方法就要看个人习惯和公司有无特殊规定了），对于熟悉 SQL 语法的程序员而言，直接使用 execSQL()和 rawQuery()方法执行 SQL 语句就能完成数据的添加、删除、更新、查询操作。

下面通过代码来演示 SQLiteDatabase 操作数据库的过程，我们这里直接使用本文档 1.1 中工程，只需修改 PersonOpenHelperTest 类中方法即可。

```
//测试添加数据
    public void inset() {
        SQLiteDatabase dataBase = getDataBase();
        ContentValues values = new ContentValues();
        values.put("name", "heima");
        values.put("age", 5);
        values.put("phone", "010-82826816");
        /*
         * 第一个参数 table，代表要将数据插入哪家表 第二个参数
         * nullColumnHack，字符串类型，指明如果某一字段没有值，那么会将该字段的值设为 NULL
         * ，一般给该参数传递 null 就行如果没有特殊要求
         * ，在这里我传递了 phone 字符串，也就是说当我的 ContentValues 中 phone 字段为空的时候系统自动给其值设置为 NULL
         * 第三个参数 ContentValues 类似一个 Map<key,value>的数据结构，key 是表中的字段，value 是值
         */
        dataBase.insert("person", "phone", values);
    }
//测试删除数据

    public void delete() {
        SQLiteDatabase database = getDataBase();
        /*
         * 第一个参数 table，代表要删除的表名
         * 第二个参数 whereClause ，选择的条件选项，如果为 null 则删除表中
```

```
        * 第三个参数 whereArgs  ,如果有条件选项，对应的条件选项的具体参数，没有写 null
        * 删除名字为"heima"的记录
        */
       database.delete("person", "name=?", new String[]{"heima"});
    }
//测试修改数据
    public void update() {
       SQLiteDatabase database = getDataBase();
       ContentValues values = new ContentValues();
       values.put("age", "100");
       /*
        * 第一个参数 table，要更新的表名
        * 第二个参数 ContentValues 设置要修改的字段的新值，没有涉及到的字段则默认不修改
        * 第三、四个参数的含义同方法 delete
        */
       database.update("person", values , "name=?", new String[]{"heima"});
    }
    //测试查询单个数据
    public void query() {
       SQLiteDatabase database = getDataBase();
       /*
        * 第一个参数 table，查询的表名
        * 第二个参数 columns，要查询的字段
        * 第三个参数 selection 过滤字段
        * 第四个参数 selectionArgs 过滤字段的值
        * 第五个参数 groupBy 分组字段，null 代表不分组
        * 第六个参数 having
        * A filter declare which row groups to include in the cursor,
        * if row grouping is being used, formatted as an SQL HAVING clause
        * (excluding the HAVING itself). Passing null will cause all row groups
        * to be included, and is required when row grouping is not being used.
        * 第七个参数 orderBy 排序字段，asc 正序，desc 倒序，null 代表自然顺序
        */
```

```
        Cursor cursor = database.query("person", new
String[]{"name,age,phone"}, "name=?", new String[]{"heima"}, null,
null, null);
            int id = cursor.getInt(0);
            String name = cursor.getString(1);
            String phone = cursor.getString(2);
            System.out.println(id + "_" + name + "_" + phone);
    }
//测试查询所有数据
    public void queryAll(){
        SQLiteDatabase database = getDataBase();
        List<Person> persons = new ArrayList<Person>();
        /*
         * 该方法跟 query()方法是完全一样的，因此参数的含义不在介绍
         */
        Cursor cursor = database.query("person", new
String[]{"name,age,phone"}, null, null, null, null, null);
        while(cursor.moveToNext()){
            Person p = new Person();
            String name = cursor.getString(0);
            int age = cursor.getInt(1);
            String phone = cursor.getString(2);
            p.setName(name);
            p.setAge(age);
            p.setPhone(phone);
            persons.add(p);
        }
    }
```

至此，本小节完！

1.5 SQLite 数据库的事务操作

跟 MySql、Oracle 等常用数据库一样，SQLite 数据库也对事物有较好的支持。

使用方法：

beginTransaction(): 开启一个事务

setTransactionSuccessful():设置成功点

endTransaction()： 结束事务，包括提交和回滚

执行过程：

使用 beginTransaction 开启一个事务，程序执行到 endTransaction 方法时会检查事务的标志是否为成功，如果程序执行到 endTransaction 之前调用了 setTransactionSuccessful 方法设置事务的标志为成功，则提交事务；如果没有调用 setTransactionSuccessful 方法则回滚事务。

那么下面通过一个案例演示 SQLite 操作事务的过程。为了便于直奔主题，我们直接使用本文档 1.4 章节中的 Android 工程中的 PersonOpenHelperTest 类即可。

案例：银行转账，需求：客户 lisi 向 zhangsan 的账户上转了 100 块。

```
public void testTransaction(){
        //这里的最后一个参数（数据库版本号）设置为 2，那么会执行 PersonOpenHelper 类中的 onUpgrade 方法
        PersonOpenHelper helper = new PersonOpenHelper(getContext(), "person", null, 2);
        SQLiteDatabase database = helper.getWritableDatabase();
        try {
            //开启事务
            database.beginTransaction();
            database.execSQL("update person set balance = balance-100 where name=?",new String[]{"lisi"});
            //当把 int a=1/0;放开的时候，发现抛出异常，那么事务就会回滚，上面的扣除 lsii 的 100 元钱不会被真正执行
            //如果把 int a = 1/0;注释掉，才发现事务成功了，lisi 的钱被扣除了 100 元，同时 zhangsan 的钱也多了 100 元。
            int a=1/0;
```

```
            database.execSQL("update person set balance = balance+100 where name=?",new String[]{"zhangsan"});
            //设置事务成功，也就是只有当代码执行到此行，才代表事务已经成功
            database.setTransactionSuccessful();
        } finally{
            //提交事务，如果 setTransactionSuccessful（）方法已经执行，则 beginTransaction（）后的语句执行成功
            //否则，事务回滚到开启事务前的状态
            database.endTransaction();
        }
    }
```

1.6 事务对效率的提高

在批量修改数据的时候，由于事务是在进行事务提交时将要执行的 SQL 操作一次性打开数据库连接执行，其执行速度比逐条执行 SQL 语句的速度快了很多倍。因此当我们开发

中遇到对数据库的批量操作那么，使用事务是提高效率的重要原则。

下面通过一个案例 ,来演示批量操作的情况下 ,使用事务和不使用事务在效率上的差异。

同样的该案例依然使用本文档 1.1 中的 Android 工程。

案例：插入一万条数据到数据库，比较使用事务和不使用事务各自所需的时间。

```
public void testTransactionEfficient(){
	PersonOpenHelper helper = new PersonOpenHelper(getContext(),
"person", null, 2);
	SQLiteDatabase database = helper.getWritableDatabase();
//	------测试不使用事务时插入 1w 条数据耗时--------------------
	long beginTime = System.currentTimeMillis();
	for(int i=0;i<10000;i++){
	   database.execSQL("insert into person(name,age,phone)
values('text'+"+i+","+i+",'"+(1320000+i)+""+"')");
	}
```

```
	long endTime = System.currentTimeMillis();
	System.out.println("不使用事务插入 1w 条数据耗时:
"+(endTime-beginTime)+"毫秒");
//	---------测试使用事务时耗时-----------------------
	beginTime = System.currentTimeMillis();
	database.beginTransaction();
	for(int i=0;i<10000;i++){
	   database.execSQL("insert into person(name,age,phone)
values('text'+"+i+","+i+",'"+(1320000+i)+""+"')");
	}
	database.setTransactionSuccessful();
	database.endTransaction();
	endTime = System.currentTimeMillis();
	System.out.println("使用事务插入 1w 条数据耗时:
"+(endTime-beginTime)+"毫秒");
   }
```

执行上面代码，查看控制台，发现不使用事务耗时 19397 毫秒，使用事务耗时 3404

毫秒，性能差别还是相当的明显。

```
Text
不使用事务插入1w条数据耗时：19397毫秒
使用事务插入1w条数据耗时：3404毫秒
```

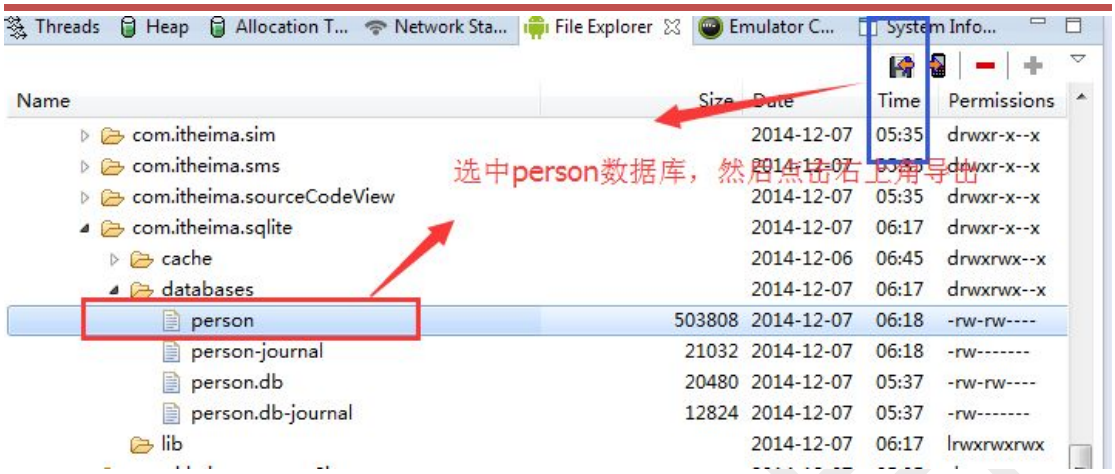
1.7 查看 SQLite 数据库

查看 SQLite 数据库有多种方法，上面章节的演示过程其实就是通过 Android API 的方式查看。那么下面要介绍的是非常常用的两种方式：1、通过 SQLite Expert 工具 2、通过 Android sqlite3 工具。

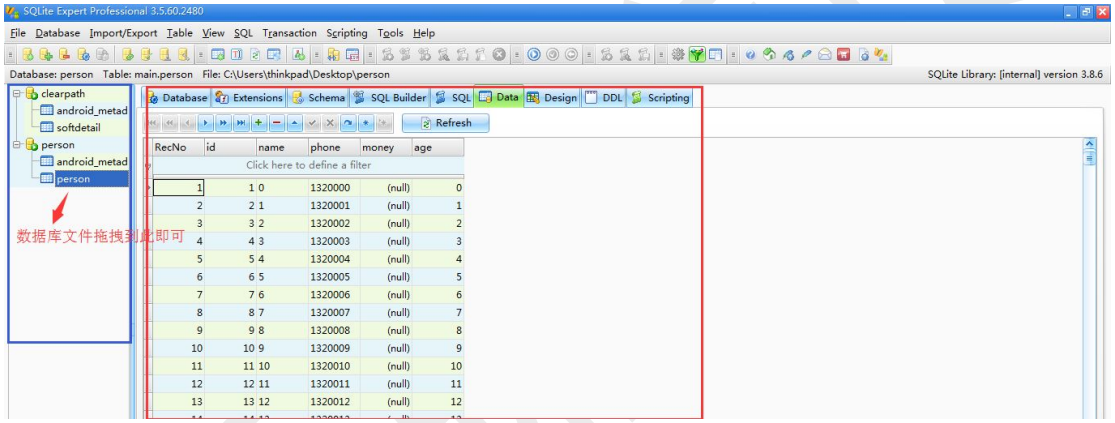
- 通过 SQLite Expert 工具

SQLite Expert（http://www.sqliteexpert.com/）是一款强大的 SQLite 数据库管理工具。从官网上下载的版本只能免费试用 30 天，试用期过后如果继续使用需要支付一定的费用购买许可。该工具的安装过程很简单，在这里就不再演示。只给大家演示如果利用该工具打开我们在上一个章节中生成的测试数据。

在 DDMS 视图中打开/data/data/com.itheima.sqlite/databases/person 文件，点击右上角的导出按钮，然后在弹出的文件对话框中选择需要保存的位置，然后点击确定即可将模拟器中的数据库文件导出到本地。



然后打开 SQLite Expert 软件，将 person 数据拖拽到如下图的左侧区域即可。



通过 Android sqlite3 工具

Android 提供了一个 sqlite3.exe 程序，位于 sdk 的 tools 目录下，用于操作 SQLite 数据库，其常用命令为：

1、sqlite3 数据库名称：进入数据库操作模式　　eg: sqlite3 contacts.db

2、tables：查看所有的表　　eg: .tables

3、schema：查看查看库中所有表的 DDL 语句　eg: .schema

4、help　：　查看帮助　　eg: .help

5、headers on/off　：显示表头，默认 off　　eg: headers on

6、mode list|column|insert|line|tabs|tcl|csv：改变输出格式

eg: .mode column

7、nullValue： NULL 空值数据显示问题 eg: .nullValue NULL

8、dump 表名 ： 生成形成表的 SQL 脚本 eq: .dump person

9、dump ： 生成整个数据库的 SQL 脚本 eq: .dump

9、exit ： 退出 sqlite 操作模式 eq: .exit

操作步骤：

1. 在命令行界面使用 adb shell 命令进入 linux 内核
2. 使用 cd 命令进入数据库所在目录（数据库的路径为”/data/data/应用包名/databases/数据库”）
3. 使用”sqlite3 数据库名”进入数据库操作模式
4. 直接使用各种命令操作数据库

2.ListView 控件（★★★★）

ListView 是我们 Android 中最重要的控件之一,是用于对数据进行列表展示的控件。

Tips：ListView 的特点。

- 屏幕上可以展示几个控件, ListView 就初始化几个,节省内存，防止内存溢出。
- 通过使用 convertView 对创建的视图对象进行复用 ,ListView 始终保持创建的对。

象个数为: 屏幕显示的条目的个数 + 1。

- ListView 自带 ScrollView 的功能，可以实现界面滚动。
- ListView 控件的设计遵循 MVC 设计模式：

  mode 数据模型(数据) ：要被显示到 ListView 上的数据集合

  view 视图(展示数据) ： ListView

  controller 控制层(把数据展示到空间上)：适配器 Adapter

下面我们依然通过案例来演示 ListView 的用法。

2.1 读取数据库的数据并显示到 ListView 上

在这里依然使用本文档 1.1 章节中的工程。在此工程上添加布局文件（2 个，一个是 Activity 对应的布局文件，另外一个是用于显示用户的一条记录，指定 ListView 数据项的展示样式）。

ListView 展示样式效果如下： 

。

注意：**布局文件的名字必须全部都是小写字母！**

1 创建 ListView 展示样式布局文件，文件名为 listview_item.xml

```
<LinearLayout
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"
   android:orientation="horizontal"
    >
    <ImageView_
        android:layout_height="60dp"
        android:layout_width="60dp"
        android:src="@drawable/ic_launcher"
        />
    <LinearLayout
        android:layout_height="match_parent"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:orientation="vertical"
        >
```

```
<TextView
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:text="用户名"
            android:id="@+id/tv_username"
            />
        <TextView
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:text="年龄"
            android:id="@+id/tv_age"
            />
        <TextView
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:text="电话"
            android:id="@+id/tv_phone"
            />
    </LinearLayout>

</LinearLayout>
```

2 创建业务类操作数据库，在该工程中新创建 PersonDao

```
//省略文件头部信息
public class PersonDao {
    private PersonOpenHelper helper;
    public PersonDao(Context context){
        helper = new PersonOpenHelper(context, "person", null, 2);
    }
    public List<Person> queryAll(){
        List<Person> persons = new ArrayList<Person>();
        SQLiteDatabase database = helper.getReadableDatabase();
        Cursor cursor = database.rawQuery("select name,age,phone from
person",null);
        while(cursor.moveToNext()){
            Person p = new Person();
            String name = cursor.getString(0);
            int age = cursor.getInt(1);
            String phone = cursor.getString(2);
            p.setName(name);
            p.setAge(age);
            p.setPhone(phone);
            persons.add(p);
        }
        cursor.close();
        database.close();
        return persons;
    }
}
```

3 修改 main_activity.xml 布局文件

```
<LinearLayout
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"
    tools:context=".MainActivity"
   android:orientation="vertical"
    >
    <TextView
        android:gravity="center_horizontal"
```

```
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:text="将数据显示在 ListView 中"
        />
    <ListView
        android:id="@+id/lv"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        ></ListView>
</LinearLayout>
```

4 使用并修改该工程默认的 Activity 类，MainActivity。该类的主要业务功能有：

（1）调用 Dao 获取数据库的全部数据

（2）获取 ListView 控件的实例

（3）自定义适配器，继承 BaseAdapter，重写 getCount 以及 getView 方法

Int getCount()：用于获取要展示的数据的总条数，即 ListView 的总长度

View getView(int position,View convertView,ViewGroup parent)

position:当前要显示的项在 ListView 的索引

convertView：缓存对象，ListView 中创建对象的个数=屏幕显示的条目数+1，当滑动屏幕时，被隐藏的项会作为缓存对象，作为 getView 的参数传递进来。只需修改此缓存对象的内容，直接使用即可，而不需要再重新 new 一个新的对象出来，节省了内存，防止内存溢出。

```
public class MainActivity extends Activity {
    private ListView lv;
    private List<Person> persons;
    private PersonDao dao;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        lv = (ListView) findViewById(R.id.lv);
        dao = new PersonDao(this);
        persons = dao.queryAll();
        lv.setAdapter(new MyAdapter());

    }
    private class MyAdapter extends BaseAdapter{

        @Override
        public int getCount() {
            return persons.size();
        }

        @Override
        public Object getItem(int position) {
            return null;
        }
@Override
        public long getItemId(int position) {
            return 0;
        }

        @Override
        public View getView(int position, View convertView, ViewGroup
parent) {
            View view;
            if (convertView!=null) {
                view = convertView;
            }else {
                view = View.inflate(MainActivity.this,
R.layout.listview_item, null);
            }
```

```
            TextView tv_username = (TextView)
view.findViewById(R.id.tv_username);
            TextView tv_age = (TextView)
view.findViewById(R.id.tv_age);
            TextView tv_phone = (TextView)
view.findViewById(R.id.tv_phone);
            Person person = persons.get(position);
            tv_age.setText("年龄: "+person.getAge()+"");
            tv_phone.setText("电话: "+person.getPhone());
            tv_username.setText("姓名: "+person.getName());
            return view;
        }

    }
}
```

至此，代码和布局文件清单均已经完成，运行程序后效果图如下。

2.2 常见的适配器 Adapter

ListView 中使用的适配器有： BaseAdapter、ArrayAdapter、SimpleAdapter。在 2.1 章节中我们演示了 BaseAdapter 的用法。下面我们将通过两个案例分别介绍 ArrayAdapter 和 SimpleAdapter。

2.3 ArrayAdapter

ArrayAdapter 不仅可以用于显示简单的文本，也可以显示样式和内容丰富的对象。在这里只演示其最简单的使用方法。为了方便演示，新创建一个 Android 工程，工程名字就叫 Adapter，使用该工程默认的布局文件和 Activity 类。

布局清单如下：

```
<LinearLayout
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"
    android:orientation="vertical"
    tools:context=".MainActivity" >
   <TextView
       android:layout_height="wrap_content"
       android:layout_width="wrap_content"
       android:layout_gravity="center"
       android:text="演示ArrayAdapter 的使用"
       />
    <ListView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/lv"
        ></ListView>
</LinearLayout>
```

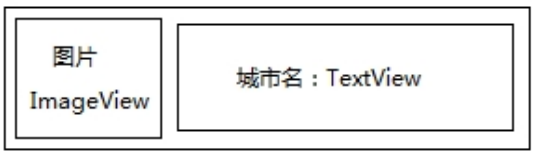
MainActivity 代码清单如下：

```
package com.itheima.adapter;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;


public class MainActivity extends Activity {

    private ListView lv;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        lv = (ListView) findViewById(R.id.lv);
        //创建一个数组
        String[] cities = new String[]{"北京","上海","广州","深圳","杭
州","珠海","武汉","郑州"};
        /**
         * 创建一个 ArrayAdapter 对象
         * 第一个参数是 Context
         * 第二个参数是 ArrayAdapter 的自身布局文件，这里使用 Android 系统
默认提供的
         * 第三个参数是数组对象
         */
        ArrayAdapter<String> myAdapter = new
ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
cities);
        lv.setAdapter(myAdapter);
    }

}
```

运行该工程，效果图如下：

2.4 SimpleAdapter

SimpleAdapter 可以实现比 ArrayAdapter 复杂一点的布局。使用 SimpleAdapter 的数据用一般都是 HashMap 构成的 List，List 的每一节对应 ListView 的每一行。HashMap 的每个键值数据映射到布局文件中对应 id 的组件上。因为系统没有对应的布局文件可用，我们可以自己定义一个布局文件。在本文档中我们用 TextView 和 ImageView 组合来进行布局以演示 SimpleAdapter 的用法。布局效果如下图：



该布局，采用 LinearLayout 水平布局。为了直奔主题，我们直接修改本文档 2.4 中的 MainActivity 类即可，使用默认的 main_activity.xml 布局文件，另外需要创建上图的布局文件，该布局文件清单如下：

```
<LinearLayout
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"
    android:orientation="horizontal"
    tools:context=".MainActivity" >
   <ImageView
       android:layout_height="60dp"
       android:layout_width="60dp"
       android:id="@+id/iv_icon"
       />
    <LinearLayout
        android:orientation="horizontal"
       android:layout_height="match_parent"
       android:layout_width="0dp"
       android:layout_weight="1"
       android:id="@+id/ll"
       android:gravity="center"
       >
        <TextView
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
             android:id="@+id/tv_text"
            />
    </LinearLayout>

</LinearLayout>
```

MainActivity 类代码清单：


```
public class MainActivity extends Activity {
    private ListView lv;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        lv = (ListView) findViewById(R.id.lv);
        List<Map<String, Object>> list = new ArrayList<Map<String,
Object>>();
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("icon", R.drawable.ic_launcher);
        map.put("text", "北京");
        list.add(map);
        map = new HashMap<String, Object>();
        map.put("icon", R.drawable.ic_launcher);
        map.put("text", "上海");
        list.add(map);
        map = new HashMap<String, Object>();
        map.put("icon", R.drawable.ic_launcher);
        map.put("text", "广州");
        list.add(map);
        map = new HashMap<String, Object>();
        map.put("icon", R.drawable.ic_launcher);
        map.put("text", "深圳");
        list.add(map);
        map = new HashMap<String, Object>();
        map.put("icon", R.drawable.ic_launcher);
        map.put("text", "杭州");
        list.add(map);
        /**
         * 创建一个 SimpleAdapter 对象
         * 第一个参数 Context 上下文
         * 第二个参数 data 要显示的数据集合
         * 第三个参数 id 指定一个作为 ListView 的子条目的布局文件
         * 第四个参数 String[] 定义得 Map 中 key 组成的数组
         * 第五个参数 int[] 控件的 id 组成的数组，必须与第四个参数的 key 一
一对应
         */
```

```
SimpleAdapter myAdapter = new SimpleAdapter(this, list,
R.layout.listview_item, new String[] { "icon", "text" }, new int[]
{ R.id.iv_icon, R.id.tv_text });
        lv.setAdapter(myAdapter);
    }
}
```

运行该工程效果如下图：

3.Android 中的对话框（★★★）

Android 中常用的对话框有通知对话框、列表对话框、单选对话框、多选对话框以及进度对话框。其中，通知对话框、列表对话框、单选以及多选对话框由 AlertDialog.Builder 创建，进度对话框由 ProgressDialog 创建。

常用方法：

- setIcon 设置对话框标题栏左侧的那个图标
- setTitle 设置对话框标题栏的提示信息
- setMessage 设置对话框主体部分的提示信息
- setPositiveButton 设置确定按钮
- setNegativeButton 设置取消按钮
- setCancelable 设置对话框在点击返回键时是否会关闭
- show 显示对话框

下面将通过案例分别介绍这五种对话框的用法。

Tips：为了方便演示，在这里需要创建一个新工程，工程名叫《对话框》。该工程的主界面有五个按钮，这五个按钮绑定了分别调用其对应对话框的函数，点击不同的按钮在其对应的函数里展示不同的对话框。因此下面的操作默认都是在该工程中实现的。

该工程的布局文件清单如下：

```
<LinearLayout
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"
    android:orientation="vertical"
    tools:context=".MainActivity" >
    <TextView
        android:layout_gravity="center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="演示五种常见对话框" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="通知对话框"
        android:onClick="notifyDialog"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="列表对话框"
        android:onClick="listDialog"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="单选对话框"
        android:onClick="singleDialog"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="多选对话框"
        android:onClick="multiDialog"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="进度对话框"
        android:onClick="progressDialog"
        />
</LinearLayout>
```

该工程运行效果图如下：

3.1 通知对话框

通知对话框使用 Builder 创建，一般都会有个确认和取消按钮。当通知对话框提示的信息是要求用户必须观看且必须做出确定或者取消的选择的时候，需要设置 setCancelable 属性为 false(默认 true),以防止用户直接使用返回键关闭对话框。

notifyDialog()方法代码清单：

```
public void notifyDialog(View v){
        Builder builder = new Builder(this);
        builder.setIcon(R.drawable.ic_launcher);
        builder.setTitle("通知");
        builder.setMessage("您看到的是通知对话框！");
        //设置对话框点击返回键不关闭
        builder.setCancelable(false);
        //设置确定按钮的点击事件
        builder.setPositiveButton("确定", new OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(MainActivity.this, "您点击了确定按钮！",
0).show();
            }
        });
        //设置取消按钮的点击事件
        builder.setNegativeButton("取消", new OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(MainActivity.this, "您点击了取消按钮！",
0).show();
            }
        });
        //将对话框显示出来
        builder.show();
    }
```

运行效果图：

3.2 列表对话框

列表对话框使用 Builder 来创建，只需调用 Builder 对象的 setItems 方法设置要展示的列表项即可。

```
public void listDialog(View v){

	Builder builder = new Builder(this);
	builder.setIcon(R.drawable.ic_launcher);
	builder.setTitle("请选择要去的城市");
	final String[] cities = new String[]{"北京","上海","广州","深圳","杭州"};
	builder.setItems(cities, new OnClickListener() {
		/*
		 * 第一个参数代表对话框对象
		 * 第二个参数是点击对象的索引
		 */
		@Override
		public void onClick(DialogInterface dialog, int which) {
			String city = cities[which];
			Toast.makeText(MainActivity.this, "您选择的是: "+city, 0).show();
		}
	});
	builder.show();
   }
```

执行上面的代码，运行效果如下图：

3.3 单选对话框

单选对话框使用 Builder 来创建，只需调用 Builder 对象的 setSingleChoiceItems 方法,设置要展示的列表项即可。

setSingleChoiceItems(CharSequence[] items,//要展示的单选选项列表

int checkedItem,//默认选中项的索引

final OnClickListener listener)//选中选项触发的点击事件

```
public void singleDialog(View v){
	Builder builder = new Builder(this);
	builder.setIcon(R.drawable.ic_launcher);
	builder.setTitle("请选择您要学习的编程语言");
	final String[] languages = new
String[]{"Java","C/C++","iOS",".Net","PHP"};
	/*
	 * 第一个参数 显示的可选项
	 * 第二参数 默认选中的索引
	 * 第三个参数 点击事件监听器
	 */
	builder.setSingleChoiceItems(languages, 0, new
OnClickListener() {

		@Override
		public void onClick(DialogInterface dialog, int which) {
			String language = languages[which];
			Toast.makeText(MainActivity.this, "您选择的编程语言是:
"+language, 0).show();
		}
	});
	//设置确定按钮的点击事件
		builder.setPositiveButton("确定", new OnClickListener() {
			@Override
	public void onClick(DialogInterface dialog, int which) {
				Toast.makeText(MainActivity.this, "您点击了确定按
钮！", 0).show();
			}
		});
		//设置取消按钮的点击事件
	builder.setNegativeButton("取消", new OnClickListener() {
			@Override
	public void onClick(DialogInterface dialog, int which) {
				Toast.makeText(MainActivity.this, "您点击了取消按
钮！", 0).show();
			}
		});
	builder.show();
   }
```

Tips：该种类型的对话框在点击选项的时候 Toast 即输出您点击的条目，但是此时该对话框并没有自动消息，需要我们点击确定和取消才会消失。而列表对话框在我们选种任何一个条目后对话框自动就消失。

3.4 多选对话框

多选对话框使用 Builder 来创建，只需调用 Builder 对象的 setMultiChoiceItems 方法，设置要展示的列表项即可。

setMultiChoiceItems(CharSequence[] items, //要展示的多选列表项

boolean[] checkedItems, //对应每个列表项的选中状态

final OnMultiChoiceClickListener listener)//点击事件

```
public void multiDialog(View v) {
      Builder builder = new Builder(this);
      builder.setIcon(R.drawable.ic_launcher);
      builder.setTitle("请选择您的兴趣爱好");
      final String[] hobbies = new String[] { "逛淘宝", "看杂志", "
看电影", "游泳", "徒步旅游" };
      /*
       * 第一个参数 显示的可选项 第二参数 默认选中的索引 第三个参数 点击
事件监听器
       */
      builder.setMultiChoiceItems(hobbies, new
boolean[]{false,false,false,false,true}, new
OnMultiChoiceClickListener() {

         @Override
   public void onClick(DialogInterface dialog, int which, boolean
isChecked) {
            String hobby = hobbies[which];
            //如果选择了某一项
            if (isChecked) {
               Toast.makeText(MainActivity.this, "您选择了："+hobby,
0).show();
            }else {//取消某一项
               Toast.makeText(MainActivity.this, "您取消了："+hobby,
0).show();
            }
         }
      });
      // 不设置点击事件的监听，仅仅退出当前对话框
      builder.setPositiveButton("确定", null);
      builder.show();
   }
```

运行上面代码效果如下图：

3.5 进度对话框

进度对话框不同于之前几种对话框，它是由 ProgressDialog 对象来创建的，而且进度对话框内部使用了消息机制 Handler 来进行处理，所以它可以直接在子线程中进行修改，无需再单独设置 Handler 来修改 UI。

```
public void progressDialog(View v) {
      final ProgressDialog dialog = new ProgressDialog(this);
      dialog.setIcon(R.drawable.ic_launcher);
      dialog.setTitle("进度对话框");
      dialog.setMessage("玩命下载中...");
      // 设置对话框的样式为水平
      dialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
      //给进度条设置最大值
      dialog.setMax(100);
      //显示对话框
      dialog.show();
      new Thread(new Runnable() {

         @Override
         public void run() {
            while (true) {
               //休眠 500ms
               SystemClock.sleep(100);
               //进度条每次增加一个单位
               dialog.incrementProgressBy(1);
               //进度条到头时退出
               if (dialog.getMax() == dialog.getProgress()) {
                  dialog.dismiss();
                  Toast.makeText(MainActivity.this, "下载完成！",
0).show();
               }
            }
         }
      }).start();
   }
```

运行上面代码效果如下图：

至此，常见的对话框都已经演示完毕！

4. Android 中几个常用控件（★★★）

在本章节中我们将用介绍 ProgressBar、Spinner、AutoCompleteTextView、MultiAutoCompleteTextView 等共四个控件的基本用法。这些控件在我们以后的工作中经常会用到的，因此这里也要求我们必须掌握这部分知识。

Tips：为了便于演示我们新创建一个项目，项目名《常用控件》。使用默认的布局文

件和默认的 Activity 类。

布局文件如下：

```
<LinearLayout
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"
    android:orientation="vertical"
    tools:context=".MainActivity" >

    <TextView
        android:layout_gravity="center_horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="演示常用控件的使用" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="进度条 ProgressBar"
        />
    <!-- 进度条控件
       android:max 进度条条总刻度
       android:progress 默认进度
    -->
     <ProgressBar
       style="@android:style/Widget.ProgressBar.Horizontal"
       android:layout_height="wrap_content"
       android:layout_width="match_parent"
       android:max="100"
       android:progress="20"
       android:id="@+id/progressBar"
       />
       <TextView
        android:layout_marginTop="30dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="下拉列表框 Spinner"
        />
```

```
    <!-- 下拉列表框控件
        android:entries 字符串数组常量
     -->
        <Spinner
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:entries="@array/cities"
            android:id="@+id/spinner"
            />

        <TextView
           android:layout_marginTop="30dp"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="自动提示文本框AutoCompleteTextView"
         />
        <!-- 自动提示文本框控件
         android:completionThreshold 自动提示门限 默认是2，这里改成1 也
就是输入一个字符即可提示
     -->
        <AutoCompleteTextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/auto_tv"
            android:completionThreshold="1"
            />
        <TextView
         android:layout_marginTop="30dp"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="多个自动提示文本框MultiAutoCompleteTextView"
         />
        <!-- 自动提示文本框控件
         android:completionThreshold 自动提示门限 默认是2，这里改成1 也
就是输入一个字符即可提示
         -->
      <MultiAutoCompleteTextView
          android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:id="@+id/multi_tv"
        android:completionThreshold="1"
        />

</LinearLayout>
```

布局文件完！

Tips：在上面布局文件清单中 Spinner 控件有个比较特殊的属性需要引入外部文件。

android:entries="@array/cities"。该属性值是一个布局文件，需要在工程中 res/values/strings.xml 中添加。下面不是 strings.xml 的文件清单，其中黄色背景是新添加的字符串常量，作为 Spinner 的备选项。

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

    <string name="app_name">常用控件</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string-array name="cities">
        <item>北京</item>
        <item>上海</item>
        <item>广州</item>
        <item>深圳</item>
    </string-array>

</resources>
```

MainActivity 代码清单如下：


```
package com.itheima.commonWidget;

import android.app.Activity;
import android.os.Bundle;
import android.os.SystemClock;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.MultiAutoCompleteTextView;
import android.widget.ProgressBar;
import android.widget.Spinner;
import android.widget.Toast;

public class MainActivity extends Activity {
    private ProgressBar progressBar;
    private Spinner spinner;
    private AutoCompleteTextView auto_tv;
    private MultiAutoCompleteTextView multi_tv;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        /*
         * 初始化控件实例
         */
        progressBar = (ProgressBar) findViewById(R.id.progressBar);
        auto_tv = (AutoCompleteTextView) findViewById(R.id.auto_tv);
        multi_tv = (MultiAutoCompleteTextView)
findViewById(R.id.multi_tv);
        spinner = (Spinner) findViewById(R.id.spinner);
        /**
         * 给 spinner 绑定选中事件，当选中某一项时触发该事件
         *
         */
```

```
spinner.setOnItemSelectedListener(new OnItemSelectedListener() {
            /*
             * 某个条目被选中后调用此函数
             * 第三个参数 position 代表选中条目的索引
             */
        @Override
        public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                /*
                 * 调用 Spinner 的 getItemAtPosition(int position)方法，获
取选中的对象
                 * 强制转换为 String 类型
                 */
                String city = (String)
spinner.getItemAtPosition(position);
            Toast.makeText(MainActivity.this, "您选择的是： " + city,
0).show();
            }
            /*
             * 什么都没有选择时调用次函数
             */
        @Override
        public void onNothingSelected(AdapterView<?> parent) {
                Toast.makeText(MainActivity.this, "您什么都没选！ ",
0).show();
            }
        });
        //调用本类中自定义的私有方法显示进度条控件
        showProgressBar();
        //创建字符串数组
        String[] books = new String[]{"JavaScript 网页开发","Android 源
码分析","深入理解 JVM","Android-阳哥随堂笔记","Android-阳哥面试宝典
","JavaEE 入门"};
        //创建 ArrayAdapter，在本文档的 ListView 控件中介绍过，因此不再重
复介绍
        ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,
books);
```

```
//给自动文本提示框控件设置 Adapter
        auto_tv.setAdapter(adapter);
        //给多文本提示框控件设置 Adapter
        multi_tv.setAdapter(adapter);
        //给多文本提示框设置分隔符，这里使用逗号作为分隔符
        multi_tv.setTokenizer(new
MultiAutoCompleteTextView.CommaTokenizer());
    }
    /*
     * 显示进度条控件
     */
     void showProgressBar() {
       new Thread(new Runnable() {

           @Override
           public void run() {
              while (true) {
                 for (int i = 0; i < 100; i++) {
                    //设置进度条的进度
                    progressBar.setProgress(i + 1);
                    SystemClock.sleep(100);
                 }
              }
           }
       }).start();
    }

}
```

代码清单完毕！

4.1 进度条 ProgressBar

进度条，与进度对话框有着类似的功能，也可以直接在子线程中控制其进度的改变。不同的是进度条一般嵌入在我们自定义的布局文件中，而进度对话框则单独占据一个布局（该

布局系统自带提供）。

4.2 下拉列表框 Spinner

Spinner 是一个列表选择框，会在用户选择后，展示一个列表供用户进行选择。Spinner 是 ViewGroup 的间接子类，它和其他的 Android 控件一样，数据需要使用 Adapter 进行封装。

4.3 自动提示文本框 AutoCompleteTextView

AutoCompleteTextView 和 EditText 组件类似，都可以输入文本。

AutoCompleteTextView 组件可以和一个字符串数组或 List 对象绑定，当用户输入一个及以上字符时，系统将在 AutoCompleteTextView 组件下方列出字符串数组中所有以输入字符开头的字符串（只能提示一次）。这一点和 www.baidu.com、www.google.com 的搜索框非常相似，当输入某一个要查找的字符串时，搜索框就会列出以这个字符串开头的最热门的搜索字符串列表。

4.4 多个自动提示文本框 MultiAutoCompleteTextView

功能类似 AutoCompleteTextView,只不过 AutoCompleteTextView 只能提示一次，再遇到短信多人发送的情况时，只提示一次的 AutoCompleteTextView 使用不是很合适，所以就有了 MultiAutoCompleteTextView。

调用 MultiAutoCompleteTextView.setTokenizer 方法为此控件设置一个值与值之间的分隔符即可，其余设置于 AutoCompleteTextView 类似。

至此该文档全部完毕！