

Android数据存取

版权声明

- 华清远见教育集团版权所有；
- 未经华清远见明确许可，不得为任何目的以任何形式复制或传播此文档的任何部分；
- 本文档包含的信息如有更改，恕不另行通知；
- 华清远见教育集团保留所有权利。

Android数据存取方式

- Android为数据存储提供了多种方式，分别有如下几种：
 - ◆ 文件
 - ◆ SharedPreferences
 - ◆ SQLite数据库
 - ◆ 内容提供者（Content Provider）

将数据保存在文件中

写入文本文件

■FileWriter

- ◆FileWriter写入单位为char。产生对象方式如下：

```
FileWriter fw = new FileWriter( "/sdcard/output.txt", false );
```

- ◆在FileWriter对象参数当中，第一个为文件名，第二个为写入模式是否为append

■BufferedWriter

- ◆使用Buffer机制来做write()时，会先将要写入之文件暂存起来，等到一定的数据量后才写入磁盘，因此可省下不少I/O所造成的负担。

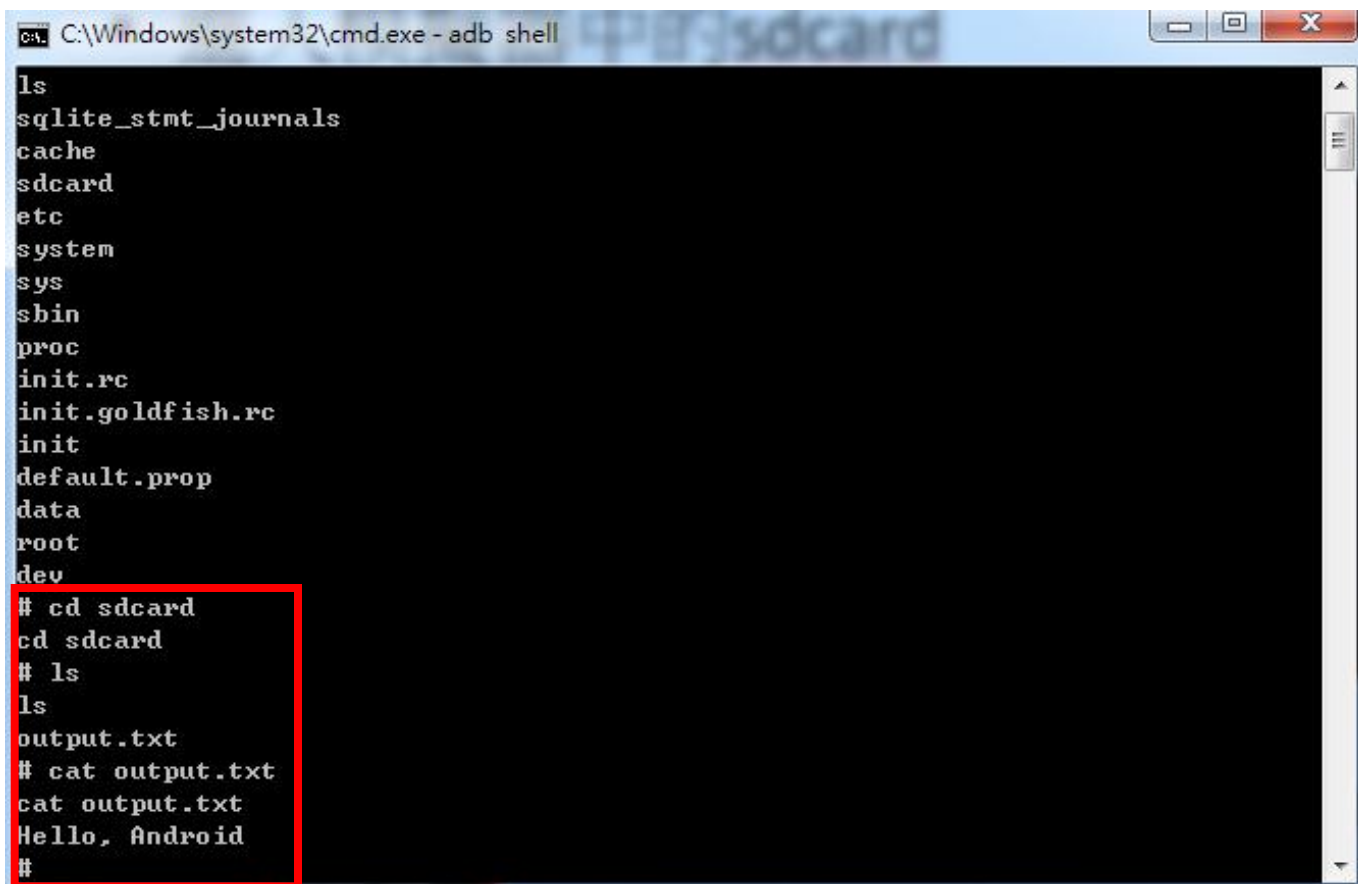
写入文本文件

■BufferedWriter常用方法

Method	功能描述
<code>close()</code>	关闭stream
<code>flush()</code>	清除stream
<code>newLine()</code>	写入换行字符
<code>write(char[] cbuf, int off, int len)</code>	写入长度为len的字符数组
<code>write(int c)</code>	写入一个字符
<code>write(String s, int off, int len)</code>	写入一个长度为len的字符串

写入文本文件

■ 范例结果如下：



The screenshot shows an Android terminal window with the title bar "C:\Windows\system32\cmd.exe - adb shell". The terminal output is as follows:

```
ls
sqlite_stmt_journals
cache
sdcard
etc
system
sys
sbin
proc
init.rc
init.goldfish.rc
init
default.prop
data
root
dev
# cd sdcard
cd sdcard
# ls
ls
output.txt
# cat output.txt
cat output.txt
Hello, Android
#
```

A red rectangular box highlights the commands from "# cd sdcard" to the final prompt "#".

写入文本文件

■程序代码如下：

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    try {  
        //建立FileWriter对象，并将写入位置设定为SD卡中的output.txt  
        FileWriter fw = new FileWriter( "/sdcard/output.txt", false );  
        //建立fw的Output Buffer  
        BufferedWriter bw = new BufferedWriter( fw );  
        bw.write("Hello, Android");  
        bw.newLine();  
        bw.close();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```

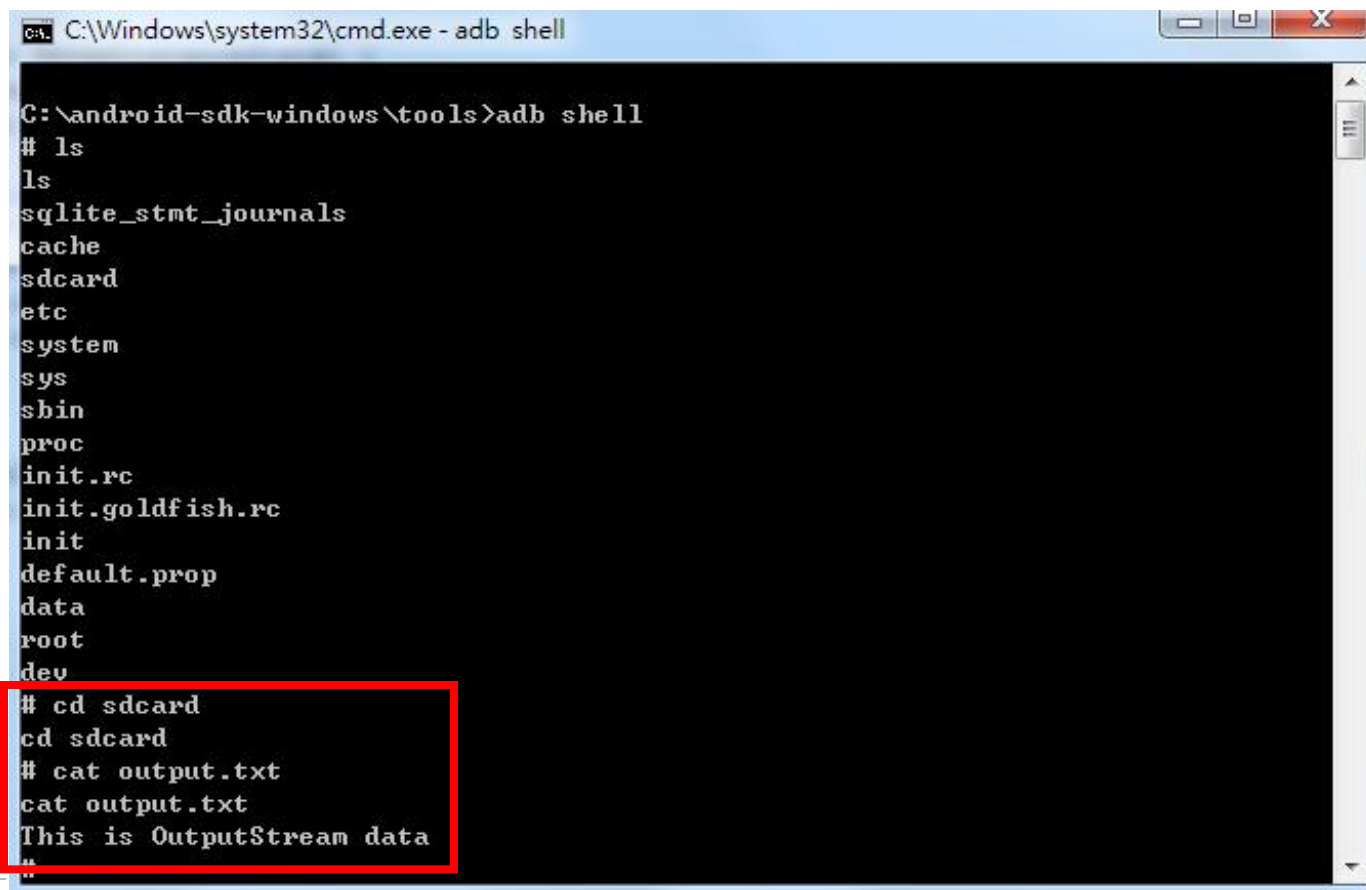

写入文件

■FileOutputStream

- ◆此种方式是以**byte**为单位对文件作存取，故通常在使用这种方式来做文件读写时会一起使用其他**Object**的**OutputStream**，目的是将我们要储存的目标文件自动以**byte**的形式作储存

写入文件

■ 范例结果如下：



```
C:\Windows\system32\cmd.exe - adb shell

C:\android-sdk-windows\tools>adb shell
# ls
ls
sqlite_stmt_journals
cache
sdcard
etc
system
sys
sbin
proc
init.rc
init.goldfish.rc
init
default.prop
data
root
dev
# cd sdcard
cd sdcard
# cat output.txt
cat output.txt
This is OutputStream data
#
```

写入文件

■程序代码如下：

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    try {  
        String data1 = "This is OutputStream data";  
        String data2 = "\n";  
        FileOutputStream output = new FileOutputStream("/sdcard/output.txt");  
        output.write(data1.getBytes());  
        output.write(data2.getBytes());  
        output.close();  
    }  
    catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

读取文本文件

■ FileReader

◆ FileReader读取单位为char。产生对象方式如下：

■ BufferedReader

```
FileReader fr = new FileReader( "/sdcard/output.txt" );
```

```
BufferedReader br = new BufferedReader( fr );
```

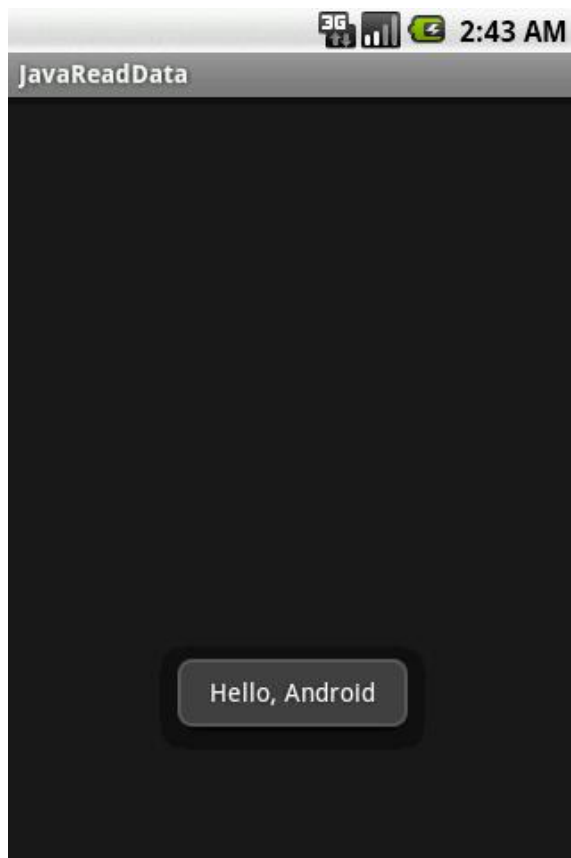
读取文本文件

■BufferedWriter常用Method

方法	功能描述
<code>close()</code>	关闭stream
<code>mark(int readAheadLimit)</code>	标记此stream现在的读取位置
<code>markSupported()</code>	布尔值，看此stream是否支持标记
<code>read()</code>	读取一个字符
<code>read(char[] cbuf, int off, int len)</code>	读取自定义长度字符串至数组中
<code>readLine()</code>	读取一整行
<code>ready()</code>	布尔值，看此stream是否准备好被读取
<code>reset()</code>	重设stream至最近mark的地方

读取文本文件

■ 范例结果如下：



读取文本文件

■程序代码如下：

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    try {  
        //建立FileReader对象，设定读取的文件为SD卡中的output.txt  
        FileReader fr = new FileReader( "/sdcard/output.txt" );  
        //建立fr的Input Buffer  
        BufferedReader br = new BufferedReader( fr );  
        String readData = "";  
        String temp = br.readLine();  
        while( temp != null ) {  
            readData += temp;  
            temp = br.readLine();  
        }  
        Context context = getApplicationContext();  
        int duration = Toast.LENGTH_LONG;  
        Toast toast = Toast.makeText(context, readData, duration);  
        toast.show();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

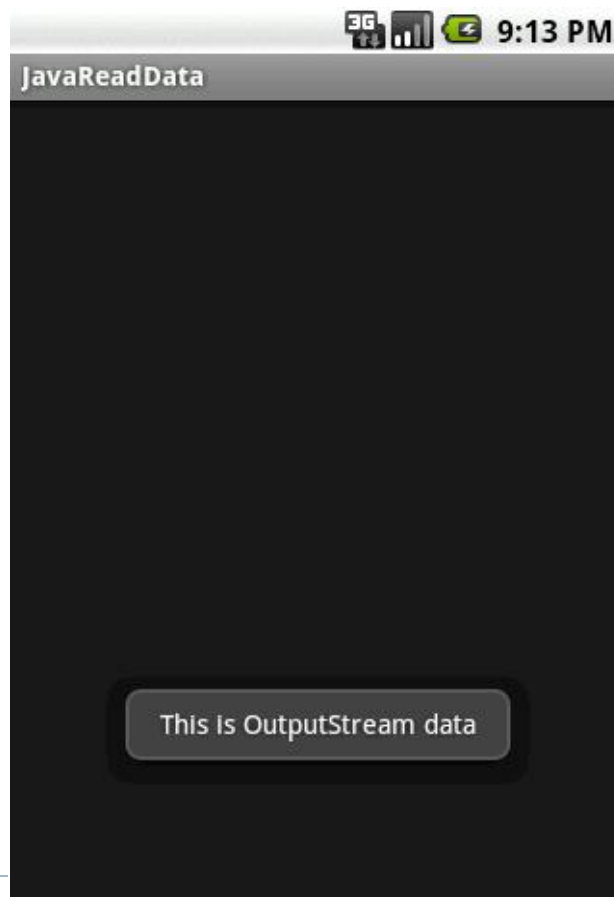
读取文件

■ FileInputStream

- ◆ 此种方式如同FileOutputStream是以byte为单位，故此方法通常也用于不同对象的读取

读取文件

■ 范例结果如下：



读取文件

■程序代码如下：

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    try {  
        FileInputStream input = new FileInputStream("/sdcard/output.txt");  
        String data = "";  
        while (input.available() > 0) {  
            byte [] b = new byte[10];  
            if ( input.read(b) != -1 )  
                data += new String(b);  
            else  
                break;  
        }  
        Context context = getApplicationContext();  
        int duration = Toast.LENGTH_LONG;  
        Toast toast = Toast.makeText(context, data, duration);  
        toast.show();  
    }  
    catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

缓存文件

- 缓存文件（**Cache File**）临时文件，在系统存储空间不足时，会被系统清除
 - ◆ **File Context.getCacheDir()**：返回一个针对特定应用的绝对路径，可以用于临时文件读写
 - ◆ 当应用被卸载时，其对应的临时文件会被删除
- 得到这个表示**Cache File**的文件路径后，就可以像其他文件操作一样对其进行操作了。

使用SharedPreferences

SharedPreferences

- Android平台提供了一个SharedPreferences类，它是一个轻量级的存储类，以“键-值”对的方式存储数据，特别适合用于保存软件配置参数。
- 使用SharedPreferences保存数据，其实质是用xml文件存放数据，文件存放在
/data/data/<package name>/shared_prefs目录下。

SharedPreferences

■使用方法

◆`SharedPreferences sharedPreferences = getSharedPreferences("init", Context.MODE_PRIVATE);`

◆`Editor editor = sharedPreferences.edit();`//获取编辑器

◆`editor.putString("username", "zhangsan");`

◆`editor.putString("password", "12345");`

◆`editor.putInt("age", 30);`

◆`editor.commit();`//提交修改

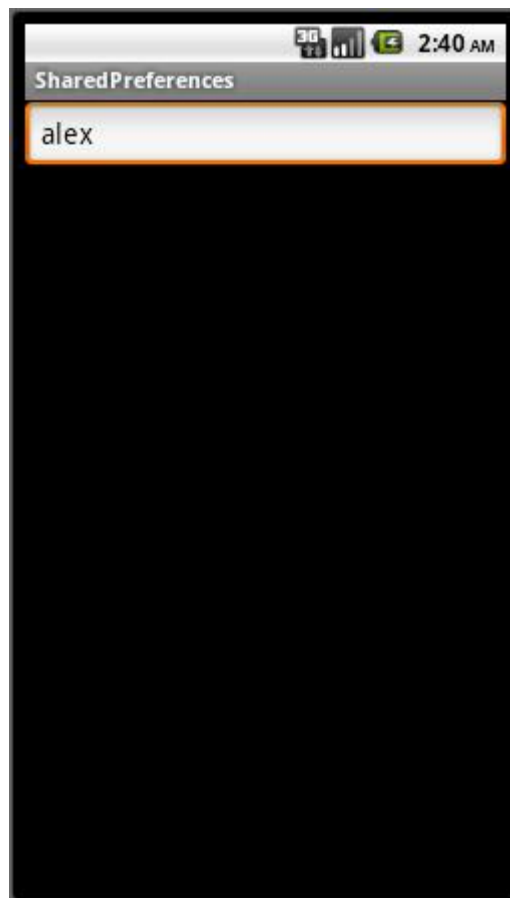
■如果希望SharedPreferences所使用的xml文件能被其他应用读和写，可以指定Context.MODE_WORLD_READABLE和Context.MODE_WORLD_WRITEABLE权限。

■Activity还提供了另一个getPreferences(mode)方法操作SharedPreferences，这个方法默认使用当前类不带包名的类名作为文件的名称。

SharedPreferences

- 使用getXXX(key,defaultValue)方法来从中读取数据，如果对应key的数据不存在，则使用defaultValue

SharedPreferences – 示例



SharedPreferences – 示例

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    edit = (EditText) findViewById(R.id.editText1);

    // 传入Preferences文件名和打开模式
    sp = getSharedPreferences("init", this.MODE_PRIVATE);
    // 试图从配置文件读取数据
    String username = sp.getString("username", null);
    // 如果有数据，则显示到输入框
    if (username != null) {
        // 将EditText的内容设置为上一次退出时保存在Preferences文件中的字符串
        edit.setText(username);
    } else {
        edit.setText("请输入用户名");
    }
}
```

SharedPreferences – 示例

```
public void onDestroy() {  
    // 在onDestroy()方法中，实现了将当前  
    // EditText中的字符串存储到Preferences文件  
    Editor editor = sp.edit();  
    editor.putString("username",  
        String.valueOf(edit.getText()));  
    editor.commit();  
    super.onDestroy();  
}
```

读取其他应用的SharedPreferences

■ 1. 创建一个要获取的那个应用的上下文

◆ Context context =

```
createPackageContext(packageName, Context.CONTEXT_IGNORE_SECURITY);
```

➤ packageName: 要访问的应用的包名

➤ 后面的那个参数，有 CONTEXT_INCLUDE_CODE和
CONTEXT_IGNORE_SECURITY两个选项。

CONTEXT_INCLUDE_CODE的意思是包括代码，也就是说可以执行这个包里面的代码。CONTEXT_IGNORE_SECURITY的意思是忽略安全警告，如果不加这个标志的话，有些功能是用不了的，会出现安全警告。

◆ 只能访问到其他应用中

MODE_WORLD_READABLE/MODE_WORLD_WRITABLE模式的
SharedPreferences

读取其他应用的SharedPreferences

■ 2. 获得SharedPreferences

◆ SharedPreferences share =
context.getSharedPreferences(name, Context.MODE_W
ORLD_READABLE+Context.MODE_WORLD_WRITEA
BLE);

■ 3. 对其进行读写操作

读取其他应用的SharedPreferences



读取其他应用中的SharedPreferences

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    try {
        context = createPackageContext("cn.com.farsight",
        Context.CONTEXT_IGNORE_SECURITY);
    } catch (NameNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    if (context != null) {
        SharedPreferences share =
        context.getSharedPreferences("init",
        Context.MODE_WORLD_READABLE + Context.MODE_WORLD_WRITEABLE);
        TextView tv = (TextView) findViewById(R.id.textView1);
        tv.setText(share.getString("username", "No value"));
    }
}
```

把数据保存到SQLite

SQLite特点

■SQLite 特点:

- ◆ 不需要一个额外的系统来运行整个数据库系统
 - ◆ 写入或是读取数据都是直接链接到文件中
- ◆ 由于都以文件形式存在，所以可以将此数据随意在大部分平台下使用
 - ◆ 支持大部分SQL92的语法
 - ◆ 运行数据库操作时所占用的资源较小



SQL语法简介

■ CREATE TABLE

- ◆ Create table用来建立表格，而表格分为列（row）和栏（column），而表格当中的数据可以有不同的数据类型

```
CREATE TABLE 数据表名称 (
    字段一      数据类型,
    字段二      数据类型,
    字段三      数据类型,
    .           .
    .           .
    .           .
    栏位N数据类型 ) ;
```

SQLite 3的数据类型

- SQLite3中，内部只支持以下几种数据存储类（Storage Class）：
 - ◆ NULL：空值
 - ◆ INTEGER：有符号整数，根据值的大小存储在1/2/3/4/6/8字节的空间中
 - ◆ REAL：浮点数，存储在8个字节的空间
 - ◆ TEXT：文本字符串，以数据库的编码方式存储文本
 - ◆ BLOB：二进制字节数据

SQLite 3的数据类型

- 数据存储类比其他数据库的数据类型更加通用。例如，**INTEGER**存储类，包含了6中不同长度的**integer**数据类型，但从数据库读取到内存中的时候，他们都被转成通用数据类型，即8字节的**integer**，从这个角度说，存储类和数据类型并无区别。
- 除了**Integer**类型的主键字段外，其他的字段都可以存储任何存储类型的数据

SQLite 3的数据类型

- Boolean类型数据，将会被保存在一位的Integer存储类
- 日期和时间类型数据，使用内置的日期时间函数，转换成TEXT、INTEGER、REAL的存储类存储：
 - ◆ TEXT: YYYY-MM-DD HH:MM:SS.SSS格式
 - ◆ REAL: 保存的是儒略日（Julian Day），从公元前4714年11月24日格林威治时间的中午算起的日期数
 - ◆ INTEGER: 从1970年1月1日以来的秒数

SQLite 3数据类型

- 虽然SQLite3内部只支持少数的存储类，但实际上，sqlite3也接受如下的数据类型：
 - ◆ Smallint: 16 位的整数。
 - ◆ integer : 32 位的整数。
 - ◆ decimal(p,s): p 精确值和 s 大小的十进位整数，精确值p是指全部有几个数(digits)大小值，s是指小数点后有几位数。如果没有特别指定，则系统会设为 p=5; s=0 。
 - ◆ float : 32位的实数。
 - ◆ double : 64位的实数。
 - ◆ char(n) : n 长度的字串，n不能超过 254。

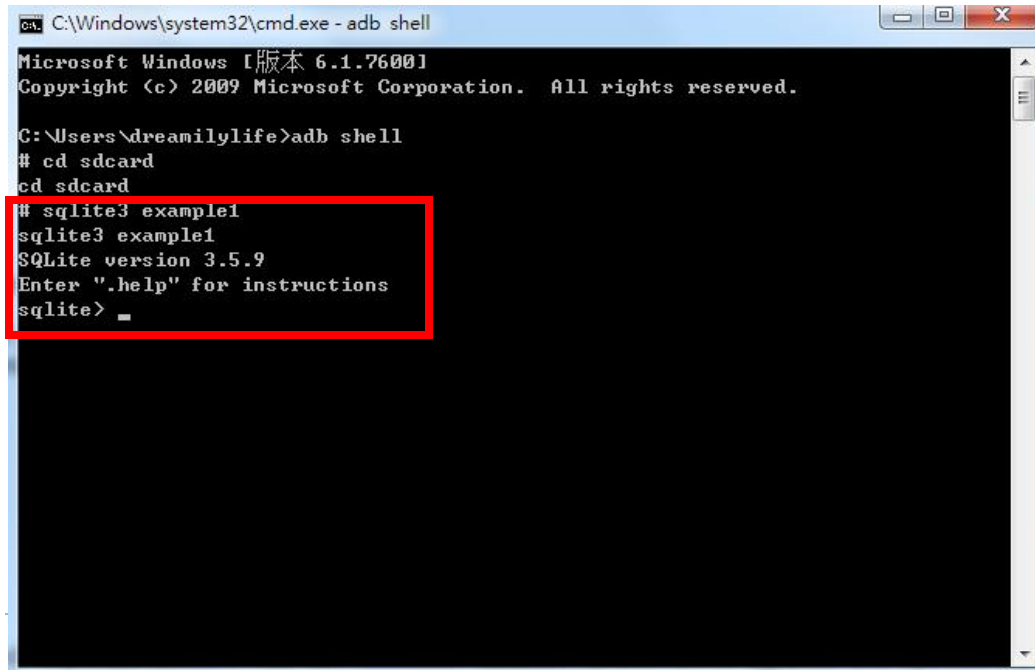
SQLite 3数据类型

- ◆ **varchar(n)** : 长度不固定且其最大长度为 **n** 的字符串, **n** 不能超过 **4000**。
- ◆ **graphic(n)** 和 **char(n)** 一样, 不过其单位是两个字节 **double-bytes**, **n** 不能超过 **127**。这个类型是为了支持两个字节长度的字体, 例如中文字。
- ◆ **vargraphic(n)** : 可变长度且其最大长度为 **n** 的双字节字符串, **n** 不能超过 **2000**
- ◆ **date** : 包含了 年份、月份、日期。
- ◆ **time** : 包含了 小时、分钟、秒。
- ◆ **Timestamp**: 包含了 年、月、日、时、分、秒、千分之一秒。

SQLite环境介绍

- 使用adb工具进入仿真器中，接着在仿真器当中使用sqlite3这个工具，使用方法为：

sqlite3 example1



```
C:\Windows\system32\cmd.exe - adb shell
Microsoft Windows [版本 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\dreamilylife>adb shell
# cd sdcard
cd sdcard
# sqlite3 example1
sqlite3 example1
SQLite version 3.5.9
Enter ".help" for instructions
sqlite> _
```

SQL语法简介

■CREATE TABLE 范例

```
CREATE TABLE STUDENTINFO (  
    ID    INTEGER NOT NULL,  
    NAME  CHAR(20)  NOT NULL,  
    PHONE CHAR(20) ,  
    CLASS CHAR(50) ,  
    PRIMARY KEY(ID) );
```


SQL语法简介

■CREATETABLE 范例结果:

```

C:\Windows\system32\cmd.exe - adb shell
Enter ".help" for instructions
sqlite> CREATE TABLE STUDENTINFO (
  ID      INTEGER      NOT NULL,
  NAME    CHAR(20)     NOT NULL,
  PHONE   CHAR(20) ,
  CLASS   CHAR(50) ,
  PRIMARY KEY(ID) );
CREATE TABLE STUDENTINFO (
  ...> ID      INTEGER      NOT NULL,
  NAME    CHAR(20)     NOT NULL,
  PHONE   CHAR(20) ,
  CLASS   CHAR(50) ,
  ...> ...> ...> ...> PRIMARY KEY(ID) );
sqlite> .tables
.tables
STUDENTINFO
sqlite> .schema
.schema
CREATE TABLE STUDENTINFO (
  ID      INTEGER      NOT NULL,
  NAME    CHAR(20)     NOT NULL,
  PHONE   CHAR(20) ,
  CLASS   CHAR(50) ,
  PRIMARY KEY(ID) );
sqlite>
    
```

SQL语法简介

■ALTER TABLE

- ◆Alter table用来更改Table，如新增、删除、更改字段属性或名称，更改指令如下表：

功能描述	功能描述
ADD域名 数据类型	新增字段
RENAME TO新的Table名称	更改Table名称

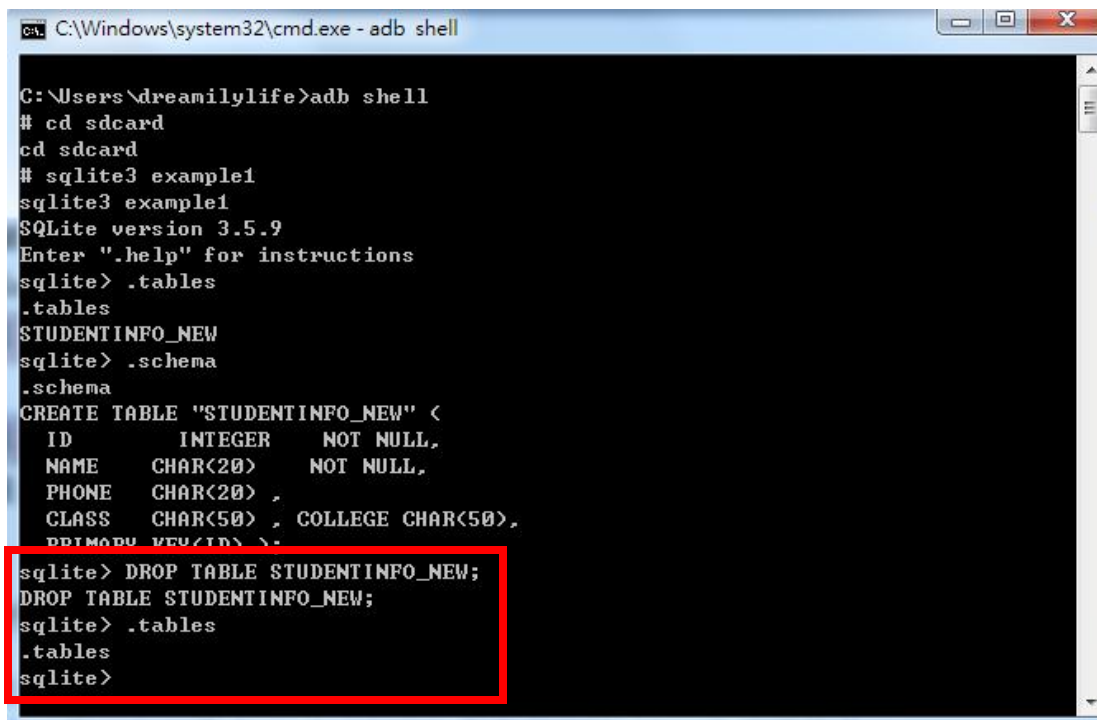
ALTER TABLE数据表名称 指令

SQL语法简介

■ DROP TABLE

◆ Drop table指令为删除一个表格，其用法如下：

DROP TABLE表格名称



```

C:\Windows\system32\cmd.exe - adb shell

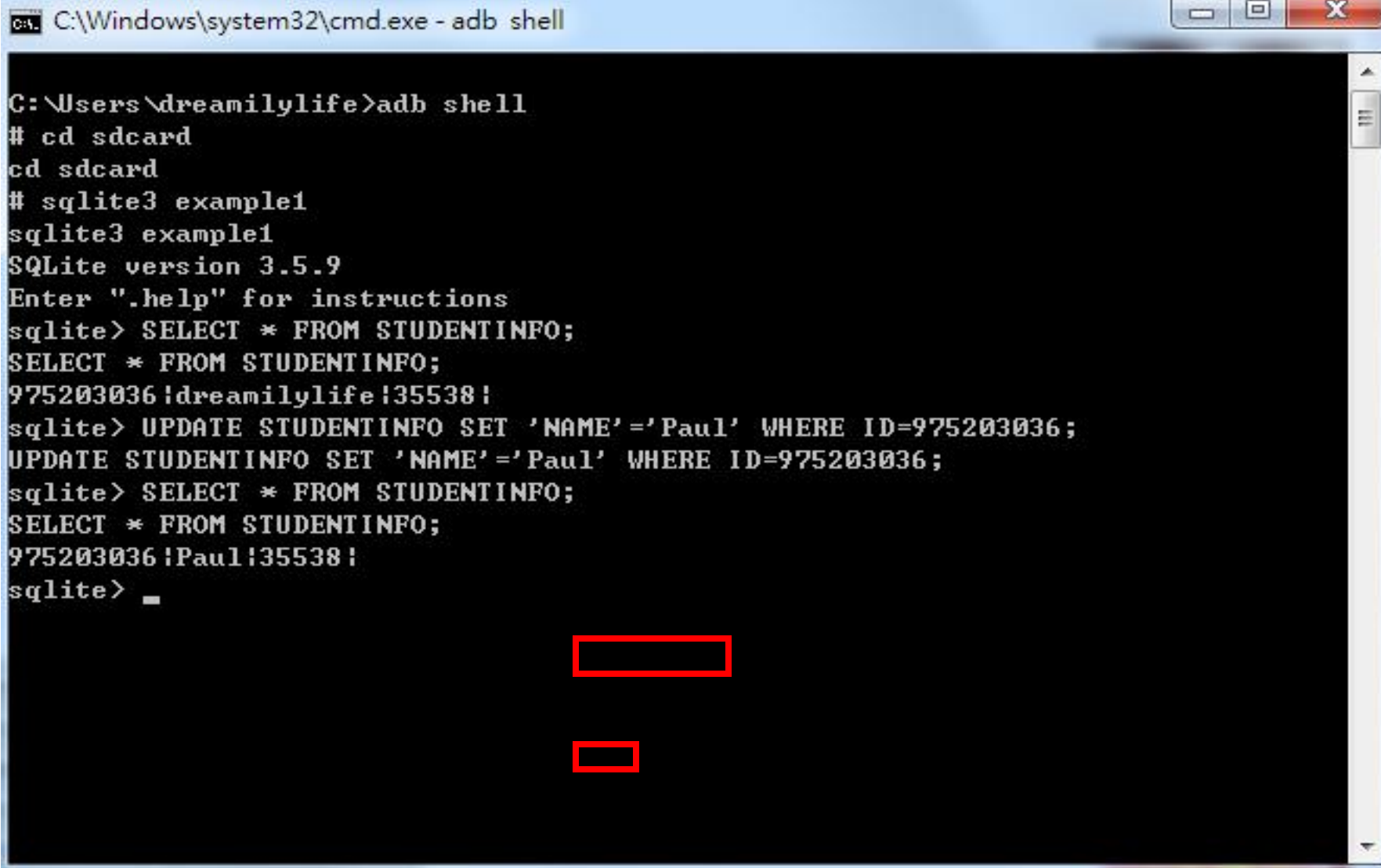
C:\Users\dreamilylife>adb shell
# cd sdcard
cd sdcard
# sqlite3 example1
sqlite3 example1
SQLite version 3.5.9
Enter ".help" for instructions
sqlite> .tables
.tables
STUDENTINFO_NEW
sqlite> .schema
.schema
CREATE TABLE "STUDENTINFO_NEW" (
  ID      INTEGER      NOT NULL,
  NAME    CHAR(20)     NOT NULL,
  PHONE   CHAR(20)     ,
  CLASS   CHAR(50)     , COLLEGE CHAR(50),
  PRIMARY KEY(ID) );
sqlite> DROP TABLE STUDENTINFO_NEW;
DROP TABLE STUDENTINFO_NEW;
sqlite> .tables
.tables
sqlite>
  
```

C:\Windows\system32\cmd.exe - adb shell

```
C:\Users\dreamilylife>adb shell
# cd sdcard
cd sdcard
# sqlite3 example1
sqlite3 example1
SQLite version 3.5.9
Enter ".help" for instructions
sqlite> INSERT INTO STUDENTINFO ( 'ID', 'NAME', 'PHONE', 'CLASS' ) VALUES ( '975203036', 'dreamilylife', '35538', '' );
INSERT INTO STUDENTINFO ( 'ID', 'NAME', 'PHONE', 'CLASS' ) VALUES ( '975203036', 'dreamilylife', '35538', '' );
sqlite> _
```

```
C:\Users\dreamilylife>adb shell
# cd sdcard
cd sdcard
# sqlite3 example1
sqlite3 example1
SQLite version 3.5.9
Enter ".help" for instructions
sqlite> INSERT INTO STUDENTINFO ( 'ID', 'NAME', 'PHONE', 'CLASS' ) VALUES ( '975
203036', 'dreamilylife', '35538', '' );
INSERT INTO STUDENTINFO ( 'ID', 'NAME', 'PHONE', 'CLASS' ) VALUES ( '975203036',
'dreamilylife', '35538', '' );
sqlite> SELECT * FROM STUDENTINFO;
SELECT * FROM STUDENTINFO;
975203036|dreamilylife|35538|
sqlite> _
```

SQL语法简介



```
C:\Windows\system32\cmd.exe - adb shell

C:\Users\dreamilylife>adb shell
# cd sdcard
cd sdcard
# sqlite3 example1
sqlite3 example1
SQLite version 3.5.9
Enter ".help" for instructions
sqlite> SELECT * FROM STUDENTINFO;
SELECT * FROM STUDENTINFO;
975203036!dreamilylife!35538!
sqlite> UPDATE STUDENTINFO SET 'NAME'='Paul' WHERE ID=975203036;
UPDATE STUDENTINFO SET 'NAME'='Paul' WHERE ID=975203036;
sqlite> SELECT * FROM STUDENTINFO;
SELECT * FROM STUDENTINFO;
975203036!Paul!35538!
sqlite> _
```

SQL 语句综合

C:\Windows\system32\cmd.exe - adb shell

```
C:\Users\dreamilylife>adb shell
# cd sdcard
cd sdcard
# sqlite3 example1
sqlite3 example1
SQLite version 3.5.9
Enter ".help" for instructions
sqlite> SELECT * FROM STUDENTINFO;
SELECT * FROM STUDENTINFO;
975203036;Paul;35538;
sqlite> DELETE FROM STUDENTINFO WHERE ID=975203036;
DELETE FROM STUDENTINFO WHERE ID=975203036;
sqlite> SELECT * FROM STUDENTINFO;
SELECT * FROM STUDENTINFO;
sqlite> _
```



SQLite3常用函数

■ 算术函数

- ◆ `abs(X)` 返回给定数字表达式的绝对值。
- ◆ `max(X,Y[,...])` 返回表达式的最大值。
- ◆ `min(X,Y[,...])` 返回表达式的最小值。
- ◆ `random(*)` 返回随机数。
- ◆ `round(X[,Y])` 返回数字表达式并四舍五入为指定的长度或精度。

SQLite3常用函数

■ 字符处理函数

- ◆ `length(X)` 返回给定字符串表达式的字符个数。
- ◆ `lower(X)` 将大写字符数据转换为小写字符数据后返回字符表达式。
- ◆ `upper(X)` 返回将小写字符数据转换为大写的字符表达式。
- ◆ `substr(X,Y,Z)` 返回表达式的一部分。

SQLite3常用函数

■集合函数

- ◆avg(X) 返回组中值的平均值。
- ◆count(X) 返回组中项目的数量。
- ◆max(X) 返回组中值的最大值。
- ◆min(X) 返回组中值的最小值。
- ◆sum(X) 返回表达式中所有值的和。

SQLite 3 日期时间函数

■ **datetime()**: 产生日期和时间

■ **date()**: 产生日期

■ **time()**: 产生时间

■ **strftime()**: 对以上三个函数产生的日期和时间进行格式化

◆ 在时间/日期函数里可以使用如下格式的字符串作为参数:

YYYY-MM-DD

YYYY-MM-DD HH:MM

YYYY-MM-DD HH:MM:SS

YYYY-MM-DD HH:MM:SS.SSS

HH:MM

HH:MM:SS

HH:MM:SS.SSS

now——产生现在的时间。

SQLite 3日期时间函数

- **strftime()**函数可以把YYYY-MM-DD HH:MM:SS格式的日期字符串转换成其它形式的字符串。
- **strftime()**的语法是**strftime(格式, 日期/时间, 修正符, 修正符, ...)**
- 它可以用以下的符号对日期和时间进行格式化:
 - ◆ %d 月份, 01-31
 - ◆ %f 小数形式的秒, SS.SSS
 - ◆ %H 小时, 00-23

SQLite3 日期时间函数

- ◆ %j 算出某一天是该年的第几天, 001-366
 - ◆ %m 月份, 00-12
 - ◆ %M 分钟, 00-59
- ◆ %s 从1970年1月1日到现在的秒数
 - ◆ %S 秒, 00-59
- ◆ %w 星期, 0-6 (0是星期天)
- ◆ %W 算出某一天属于该年的第几周, 01-53
 - ◆ %Y 年, YYYY
 - ◆ %% 百分号

SQLite3常用函数

■其他函数

- ◆typeof(X) 返回数据的类型。
- ◆last_insert_rowid() 返回最后插入的数据的ID。
 - ◆sqlite_version(*) 返回SQLite的版本。
- ◆change_count() 返回受上一语句影响的行数。
 - ◆last_statement_change_count()

Android中和SQLite操作相关的类

■ SQLiteDatabase

- ◆ `openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flag)`: 打开指定路径的数据库文件，传入CursorFactory对象用于创建查询时返回的Cursor对象。参数flag为打开模式，可以是：
OPEN_READWRITE、OPEN_READONLY、
CREATE_IF_NECESSARY等
- ◆ `openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)`: 相当于openDatabase方法以flag为CREATE_IF_NECESSARY方式打开数据库
- ◆ `create(SQLiteDatabase.CursorFactory factory)`: 创建一个内存数据库，主要应用于数据处理速度要求高的场合

Android中和SQLite操作相关的类

- ◆ `void execSQL(String sql, Object[] bindArgs)`: 执行增删改查外的SQL语句, `Object`数组为绑定的参数
- ◆ `void execSQL(String sql)`: 执行除select等有返回数据的SQL语句
- ◆ `update(String table, ContentValues values, String whereClause, String[] whereArgs)`
- ◆ `delete(String table, String whereClause, String[] whereArgs)`
- ◆ `insert(String table, String nullColumnHack, ContentValues values)`
- ◆ `execSQL(String sql)/execSQL(String sql, Object[] bindArgs)`: 执行除Select等有返回值的语句
- ◆ `close()`

Android中和SQLite操作相关的类

- ◆ Cursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)
- ◆ Cursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)
- ◆ Cursor query(boolean distinct, String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)

Android中和SQLite操作相关的类

◆ Cursor

queryWithFactory(SQLiteDatabase.CursorFactory
cursorFactory, boolean distinct, String table, String[]
columns, String selection, String[] selectionArgs, String
groupBy, String having, String orderBy, String limit)

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

◆ Cursor

rawQueryWithFactory(SQLiteDatabase.CursorFactory
cursorFactory, String sql, String[] selectionArgs, String
editTable)

Android中和SQLite操作相关的类

■ContentValues:

- ◆ 一个用于保存名-值对的类，用于保存数据库表中的字段和其对应的值
 - ◆ put(name,value)
 - ◆ get(name)
 - ◆ getAsXXX(name)如
getAsByte(name)/getAsBoolean(name)等
 - ◆ clear()清除数据

Android中和SQLite操作相关的类

■SQLiteOpenHelper

- ◆这是一个比较常用于对数据库进行操作的类。我们只需要继承这个类，并且实现其中的2个方法：
 - onCreate(SQLiteDatabase db): 该方法在数据库第一次被创建时调用，一般将表的创建工作放在该方法中执行
 - onUpgrade(SQLiteDatabase db,int oldVersion,int newVersion): 在打开数据库版本不一致时调用
- ◆通过调用SQLiteOpenHelper的getWritableDatabase()或者getReadableDatabase()方法来打开一个数据库

Android中和SQLite操作相关的类

■Cursor: 用于表示数据集的游标

◆常用Cursor方法如下表

类型	方法	功能描述
void	close()	关闭Cursor并释放其资源
int	getColumnCount()	回传Column数量
int	getColumnIndex(String columnName)	回传输入之字段的字段索引值
String	getColumnName(int columnIndex)	回传对应之索引值的域名
String[]	getColumnNames()	回传域名字符串数组
int	getCount()	回传数据列数
int	getPosition()	回传现在数据在哪一列
String	getString(int columnIndex)	将此字段数据以String回传
double	getDouble(int columnIndex)	将此字段数据以Double回传
float	getFloat(int columnIndex)	将此字段数据以Float回传
int	getInt(int columnIndex)	将此字段数据以int回传
long	getLong(int columnIndex)	将此字段数据以Long回传

Android中和SQLite操作相关的类

■常用Cursor如下表

类型	方法	功能描述
boolean	isFirst()	看目前Cursor位置是否在最前面
boolean	isLast()	看目前Cursor位置是否在最后面
boolean	isNull(int columnIndex)	看指定域值是否为Null
boolean	move(int offset)	移动Cursor位置至指定的offset
boolean	moveToFirst()	移动Cursor位置到最前面
boolean	moveToLast()	移动Cursor位置到最后面
boolean	moveToNext()	移动Cursor到下一个位置
boolean	moveToPrevious()	移动Cursor到前一个位置
boolean	moveToPosition(int position)	移动Cursor到绝对位置
boolean	requery()	重新做Query

Activity中对Cursor的管理

- 我们一般需要在应用周期内对Cursor进行管理，例如在暂停或者停止的时候，调用**deactivated**让其暂时“休眠”，释放部分资源，而在应用程序重新开始的时候，调用**requery()**对其进行刷新激活
 - 在不需要的时候，使用**close()**关闭
 - 可以将这个工作委托给Activity来完成，调用Activity的**startManagingCursor ()**来让Activity开始管理Cursor，**stopManagingCursor()**结束管理

SQLite的事务处理

- 一项事务从**beginTransaction()**方法开始，且包含在**try/catch**中。
- 如果操作成功，则调用**setTransactionSuccessful()**方法提交更改，而如果没有调用这个方法，则不会被提交。
 - 最后，调用**endTransaction()**来终止事务
- 可以使用**inTransaction()**方法来判断是否在事务中

SQLite开发- 案例

```
class MyDatabaseHelper extends SQLiteOpenHelper {
    public String DATABASE_TABLE = "STUDENTINFO";
    public final String DB_CREATE_TABLE = "CREATE TABLE " +
        DATABASE_TABLE
        + " ( " + "ID          INTEGER          NOT NULL, "
        + "NAME      CHAR(20)      NOT NULL, " + "PHONE  CHAR(20), "
        + "CLASS     CHAR(50), " + "PRIMARY KEY(ID) );";

    public MyDatabaseHelper(Context context, String name,
        CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(DB_CREATE_TABLE);
    }
}
```

SQLite开发- 案例

```
public class Main extends Activity {  
    private final String DATABASE_NAME = "school";  
    private SQLiteDatabase db;  
    private ListView lv;  
    private ArrayList<Map<String, Object>> data = new  
        ArrayList<Map<String, Object>>();  
    private HashMap<String, Object> mymap;
```

```
@Override
```

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    //setContentView(R.layout.main);  
    lv = new ListView(this);
```

```
MyDatabaseHelper myDBHelper = new MyDatabaseHelper(this,  
    DATABASE_NAME,  
    null, 3);
```

```
db = myDBHelper.getWritableDatabase();
```

```
// 初始化数据
```

```
initData("tom", "09393", "1106");
```

```
// initData("jerry", "843293", "1107");
```

```
// initData("harry", "232342", "1108");
```

SQLite开发—案例

```
// 获取STUDENTINFO数据表中的数据
String sql = "SELECT * FROM STUDENTINFO;";

Cursor result = db.rawQuery(sql, null);

    data.clear();
//将数据放到Adapter中
while (!result.isLast()) {
    result.moveToNext();
    mymap = new HashMap<String, Object>();
    for (int i = 0; i < result.getColumnCount(); i++) {
        mymap.put(result洗getColumnName(i), result.getString(i));
    }
    data.add(mymap);
}
SimpleAdapter adapter = new SimpleAdapter(this, data,
R.layout.listcontent, new String[] { "ID", "NAME", "PHONE",
"CLASS" }, new int[] { R.id.textView1, R.id.textView2,
R.id.textView3, R.id.textView4 });
```

SQLite开发—案例

```
lv.setAdapter(adapter);  
setContentView(lv);  
  
result.close();  
db.close();  
//自己在ListView上加上上下文菜单，并且做删除/修改的处理  
  
}
```

练习

■使用SQLite实现自己的通讯录

◆表格结构如下

字段名称	数据类型	说明	字段名称	数据类型	说明
_id	Integer	记录编号	Name	Varchar	姓名
Phone	Varchar	固定电话	Mobile	Varchar	手机号码
Email	Varchar	邮箱地址	Post	Varchar	邮编
Address	Varchar	通信地址	Comp	Varchar	公司名称

Content Provider



Content Provider

■什么是Content provider?

- ◆Content Provider 是Android应用程序的四大组成部分之一
- ◆是Android中的跨应用访问数据机制
- 为何需要content provider?
- ◆Android中每一个app的资源是私有的
- ◆app通过content provider和其他app共享私有数据

Content Provider

- **ContentProvider**可提供一个接口给所有应用程序来分享数据，而分享数据的基本格式是利用**URI**来当成传递的媒介：

```
<scheme> : //<authority path-abempty> "?"<query> #<fragment>
```

- 而**ContentProvider**的scheme为「**content://**」，而在**CONTENT_URI**中的各种**URI**都有ID，所以在向**ContentProvider**指定取得某个ID的资料，如下：

```
content : //.../35
```

- 在此可利用**ContentUris**中的**withAppendedId**方法来帮**URI**加入ID：

```
Uri myUri = ContentUris.withAppendedId( Uri contentUri, long id );
```

- 或是利用**Uri**的**withAppendedPath**方法加入ID

```
withAppendedPath(Uri baseUri, String pathSegment);
```


ContentProvider

■URI:

- ◆A: 标准前缀，用来说明一个Content Provider控制这些数据，对于自定义ContentProvider来说，就是“content://”，我们无法改变；
- ◆B: URI的标识，它定义了是哪个Content Provider提供这些数据。对于第三方应用程序，为了保证URI标识的唯一性，它必须是一个完整的、小写的包名+类名。这个标识在<provider> 元素的 authorities 属性中说明：

```
<provider name=".TransportationProvider"
authorities="com.example.transportationprovider" ... >
```

content://com.example.transportationprovider/trains/122

A B C D

ContentProvider

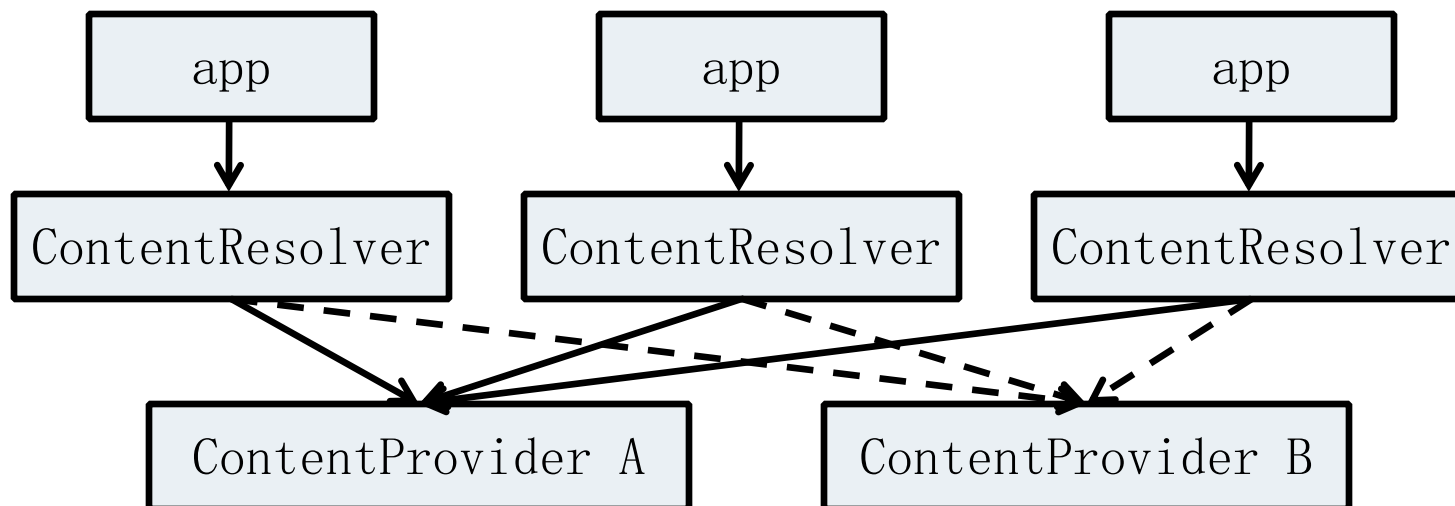
- ◆C: 路径, Content Provider使用这些路径来确定当前需要的是什么样的数据, URI中可能不包括路径, 也可能包括多个;
- ◆D: 如果URI中包含, 表示需要获取的记录的ID; 如果没有ID, 就表示返回全部;

由于URI通常比较长, 而且有时候容易出错, 且难以理解。所以, 在Android当中定义了一些辅助类, 并且定义了一些常量来代替这些长字符串, 例如:

`People.CONTENT_URI`

Content Provider

■ 如何使用content provider



Content Provider

■ URI定位资源

◆ content://contacts/people

◆ content://call_log

■ 类似关系数据库的访问方式

```
delete(Uri url, String where, String[] selectionArgs)
insert(Uri url, ContentValues values)
query(Uri uri, String[] projection, String selection,
      String[] selectionArgs, String sortOrder)
update(Uri uri, ContentValues values, String where,
```

■ 以一组数据为String[] selectionArgs, 返回Uri

_ID	NUMBER	NUMBER_KEY	LABEL	NAME	TYPE
13	(425) 555 6677	425 555 6677	Kirkland office	Bully Pulpit	TYPE_WORK

Content Provider

在AndroidManifest.xml中声明一个provider

```
<provider ...>...</provider>
```

创建MyProvider类，继承自ContentProvider。
定义该provider提供的数据集的URI和字段名

在MyProvider中实现ContentProvider的6个
abstract method: query, insert, update,
delete, getType, onCreate

在其他app中使用ContentResolver通过URI访
问MyProvider提供的数据

Content Provider

ContentProvider的Manifest配置

```
<provider android:authorities= "list"  
    android:enabled=[ "true" | "false" ]  
    android:exported=[ "true" | "false" ]  
    android:grantUriPermissions=[ "true" | "false" ]  
    android:icon= "drawable resource"  
    android:initOrder= "integer"  
    android:label= "string resource"  
    android:multiprocess=[ "true" | "false" ]  
    android:name= "string"  
    android:permission= "string"  
    android:process= "string"  
    android:readPermission= "string"  
    android:syncable=[ "true" | "false" ]  
    android:writePermission= "string" >  
</provider>
```

Content Resolver

Content Resolver

- 在前面提到的**ContentProvider**可将**Content**分享至不同的应用程序之中，而**ContentResolver**则是一个标准的方式来取得**ContentProvider**所提供的数据，也是用来修改数据的方法，但若牵涉到写入或修改数据的话，则要看目标的**ContentProvider**是否允许用户对数据做存取的动作，若无此权限，则**ContentResolver**方法会失败。
- ContentProvider**通常会使用**URI**的方式来当作分享数据的识别，故在**ContentResolver**要读取某一**ContentProvider**数据时则必须将要读取的**URI**当成参数来使用。

Content Resolver

- 在本范例中，我们利用系统内建**Phones.CONTENT_URI**来得到电话中的联系人信息并将其列表，范例程序代码如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Uri uri = ContactsContract.Contacts.CONTENT_URI;
    Uri uriDetail = ContactsContract.Data.CONTENT_URI;
    ContentResolver resolver = this.getContentResolver();
    Cursor cursor = resolver.query(uriDetail, new
String[]{ContactsContract.Data.RAW_CONTACT_ID, ContactsContract.Data.DATA2},
ContactsContract.Data._ID+"=?", new String[]{"1"}, null);
    StringBuffer detail = new StringBuffer();
    String s = ContactsContract.Data.DATA2;
    System.out.println(cursor.getColumnCount());
    System.out.println(cursor.getCount());
    while(!cursor.isLast()){
        cursor.moveToNext();
        int idx = cursor.getColumnIndex(s);
        detail.append(cursor.getString(idx));
    }
}
```

Content Resolver

■ContentResolver的查询参数如下表：

参数	功能描述
uri	欲获得资料之content://
projection	指定要回传哪些字段，若值为null代表全部回传
selection	指定要回传哪些列，若值为null代表全部回传
selectionArgs	可在回传条件中加入「？」，此数组会依序取代未知的值
sortOrder	设定回传结果的排序方式

Content Resolver

■ContentResolver 的数据处理方法如下表：

方法	描述
delete(Uri url, String where, String[] selectionArgs)	删除选中的数据
insert(Uri url, ContentValues values)	插入数据
query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)	查询数据
update(Uri uri, ContentValues values, String where, String[] selectionArgs)	更新数据

Content Resolver



Content Resolver

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Uri uri =
    Uri.parse("content://com.android.contacts/contacts");
    Cursor c = getContentResolver().query(uri,
    null, null, // query(Phones.CONTENT_URI, null, null,
    null, null);
    startManagingCursor(c);
    ListAdapter adapter = new SimpleCursorAdapter(this,
    android.R.layout.simple_list_item_2, c, new String[] {
    ContactsContract.Contacts._ID,
    ContactsContract.Contacts.DISPLAY_NAME}, new int[] {
    android.R.id.text1, android.R.id.text2 });
    setListAdapter(adapter);
}
```

Content Resolver

- 本例中，需要在AndroidManifest.xml中加入访问通讯录的权限：

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

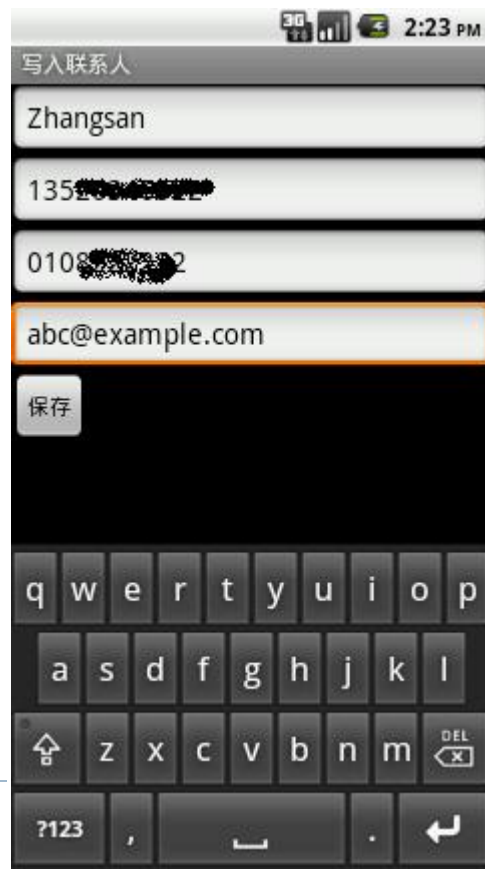
ContentResolver

■Android通讯录相关类:

- ◆ ContactsContract.RawContacts
 - ◆ ContactsContract.Data
 - ◆ ContactsContract.CommonDataKinds
- ◆ ContactsContract.CommonDataKinds.Phone
- ◆ ContactsContract.CommonDataKinds.Email
 - ◆ ContactsContract.CommonDataKinds.Im
- ◆ ContactsContract.CommonDataKinds.StructuredName

Content Resolver

■上面范例是取得ContentProvider的数据，而接着介绍如何通过ContentResolver来新增数据，要达到此目的则需使用ContentValues，此范例新增一笔联系人数据至通讯录中



Content Resolver

```
//首先向RawContacts.CONTENT_URI执行一个空值插入，目的是获取系统返回的
rawContactId
ContentValues cv = new ContentValues();
Uri uri = resolver.insert(RawContacts.CONTENT_URI, cv);
long rawContactId = ContentUris.parseId(uri);
//插入姓名
cv.clear();
cv.put(Data.RAW_CONTACT_ID, rawContactId);
cv.put(Data.MIMETYPE, StructuredName.CONTENT_ITEM_TYPE);
cv.put(StructuredName.PHONETIC_FAMILY_NAME, name);
resolver.insert(RawContacts.CONTENT_URI, cv);

//插入手机号码
cv.clear();
cv.put(Data.RAW_CONTACT_ID, rawContactId);
cv.put(Data.MIMETYPE,
ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE);
//电话类型
cv.put(ContactsContract.CommonDataKinds.Phone.TYPE,
ContactsContract.CommonDataKinds.Phone.TYPE_MOBILE);
//电话号码
cv.put(ContactsContract.CommonDataKinds.Phone.NUMBER, phone);
resolver.insert(RawContacts.CONTENT_URI, cv);
```

ContentResolver

■ 读取媒体库

◆ 使用的content uri是:

`MediaStore.Audio.Media.EXTERNAL_CONTENT_URI`

◆ 读取的字段名称封装在`MediaStore.Audio.Media`的常量中，如：`MediaStore.Audio.Media.TITLE`等

媒体库内容
媒体库内容
媒体库内容
媒体库内容
媒体库内容
媒体库内容

Audio files: 2
Columns: 2
标题: qingge
长度: 260 seconds
标题: 红豆
长度: 236 seconds

ContentResolver

```
try {
    String[] requestedColumns =
{ MediaStore.Audio.Media.TITLE, MediaStore.Audio.Media.DURATION };
    Cursor cur =
managedQuery(MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,
requestedColumns, null, null, null);
    Log.d(DEBUG_TAG, "Audio files: " + cur.getCount());
    Log.d(DEBUG_TAG, "Columns: " + cur.getColumnCount());
    // String[] columns = cur.getColumnNames();
    int name =
cur.getColumnIndex(MediaStore.Audio.Media.TITLE);
    int size =
cur.getColumnIndex(MediaStore.Audio.Media.DURATION);
    cur.moveToFirst();
    while (!cur.isAfterLast()) {
        Log.d(DEBUG_TAG, "标题: " + cur.getString(name));
        Log.d(DEBUG_TAG, "长度: " + cur.getInt(size) / 1000
+ " seconds");
        cur.moveToNext();
    }
}
```

Q&A



