实验三

数据存储

2017 年 10 月 15 日

实验介绍

本次实验，我们学习 Android 的数据存储部分。主要包括以下部分：

1. 文件存取
2. Sharedpreferences 文件存取；
3. SQLite 数据库文件存取；

1. 文件存取

学过 java 的同学都知道，新建一个文件和输出流，就能够写文件了，但是 Android 却不一样，因为 Android 是基于 Linux 的，我们在读写文件的时候，还需要加上文件的操作模式。Android 的操作模式如下：

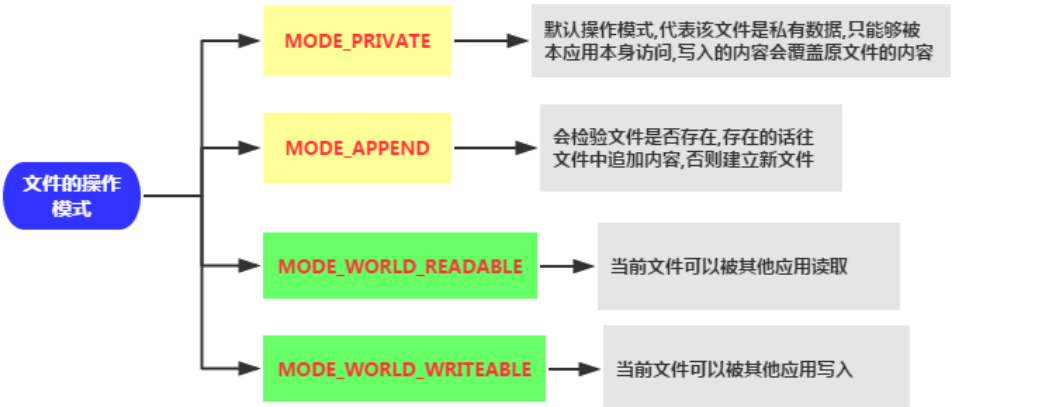
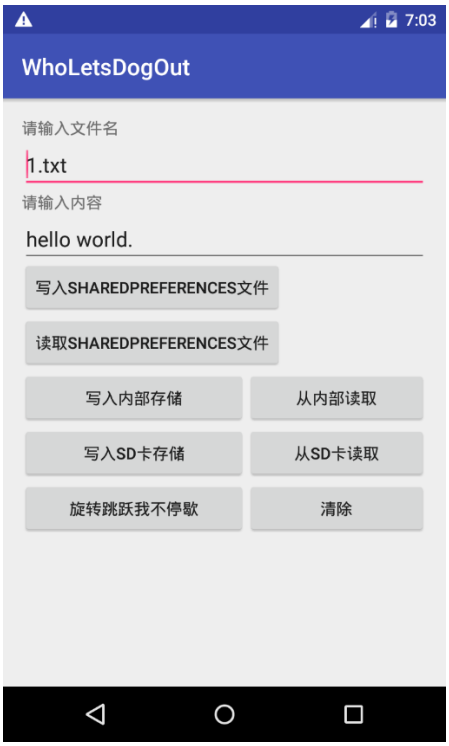


图1：文件的操作模式

Android 的数据存数分为内部存储和外部存储（如 SDCARD 等），本节将详细介绍如何对 Android 的内部存储和外部存储进行数据的读取和写入。此处文件的写入模式以 MODE_PRIVATE 为例。

1.1 内部存储存取

step1. 在 Android Studio 里建立一个空项目，minSdkVersion 选择 16，设置 activity_mai.xml 布局如下：



Step 2.为写入内部存储和从内部读取两个 button 设置监听事件：

```
private Button btnWriteInternal;
private Button btnReadInternal;
```

```
btnReadInternal = (Button)findViewById(R.id.readbutton);
btnWriteInternal = (Button)findViewById(R.id.writebutton);
```

```
btnWriteInternal.setOnClickListener(this);
btnReadInternal.setOnClickListener(this);
```

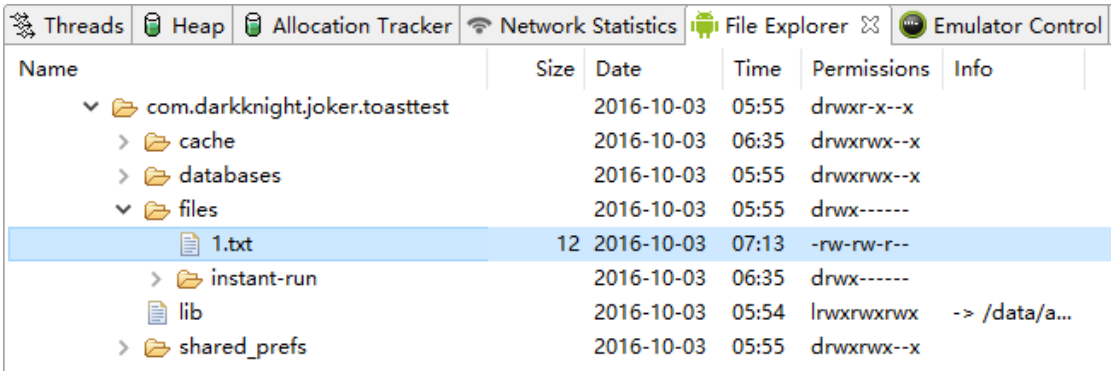
在 onClick(View v)里，分别为读取和写入内部存储添加事件处理方法，获取文件的输出流和输入流，进行读写。

```
case R.id.readbutton:
    FileHelper fh2 = new FileHelper(mContext);
    try{
        String fname = editTextName.getText().toString();
        String fcontent = fh2.read(fname);
        Toast.makeText(mContext,fcontent,Toast.LENGTH_SHORT).show();
    }catch (IOException e){
        e.printStackTrace();
        Toast.makeText(mContext,"数据读取失败",Toast.LENGTH_SHORT).show();
    }
    break;
```

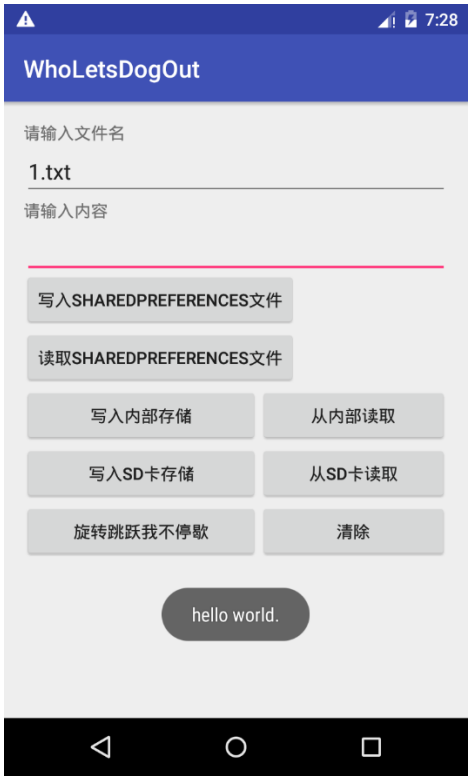
```
case R.id.writebutton:
    FileHelper fh = new FileHelper(mContext);
    String filename = editTextName.getText().toString();
    String fileContent = editTextContent.getText().toString();
    try {
        fh.save(filename, fileContent);
        Toast.makeText(getApplicationContext(), "数据写入成功", Toast.LENGTH_SHORT).show();
    } catch (Exception e) {
        e.printStackTrace();
        Toast.makeText(getApplicationContext(), "数据写入失败", Toast.LENGTH_SHORT).show();
    }
    break;
```

Step 3. 点击写入内部存储 button，写入内部存储：

将 hello world 写入 1.txt 中，该文件位于内部存储中，可以通过 android studio 的 Tools->Android->android device monitor 来查看，结果如下：



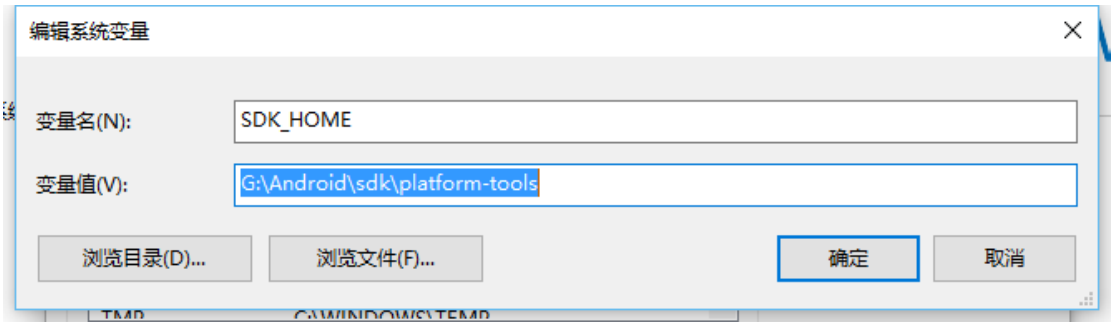
step4，我们也可以点击从内部读取来查看 1.txt 中的文件内容：

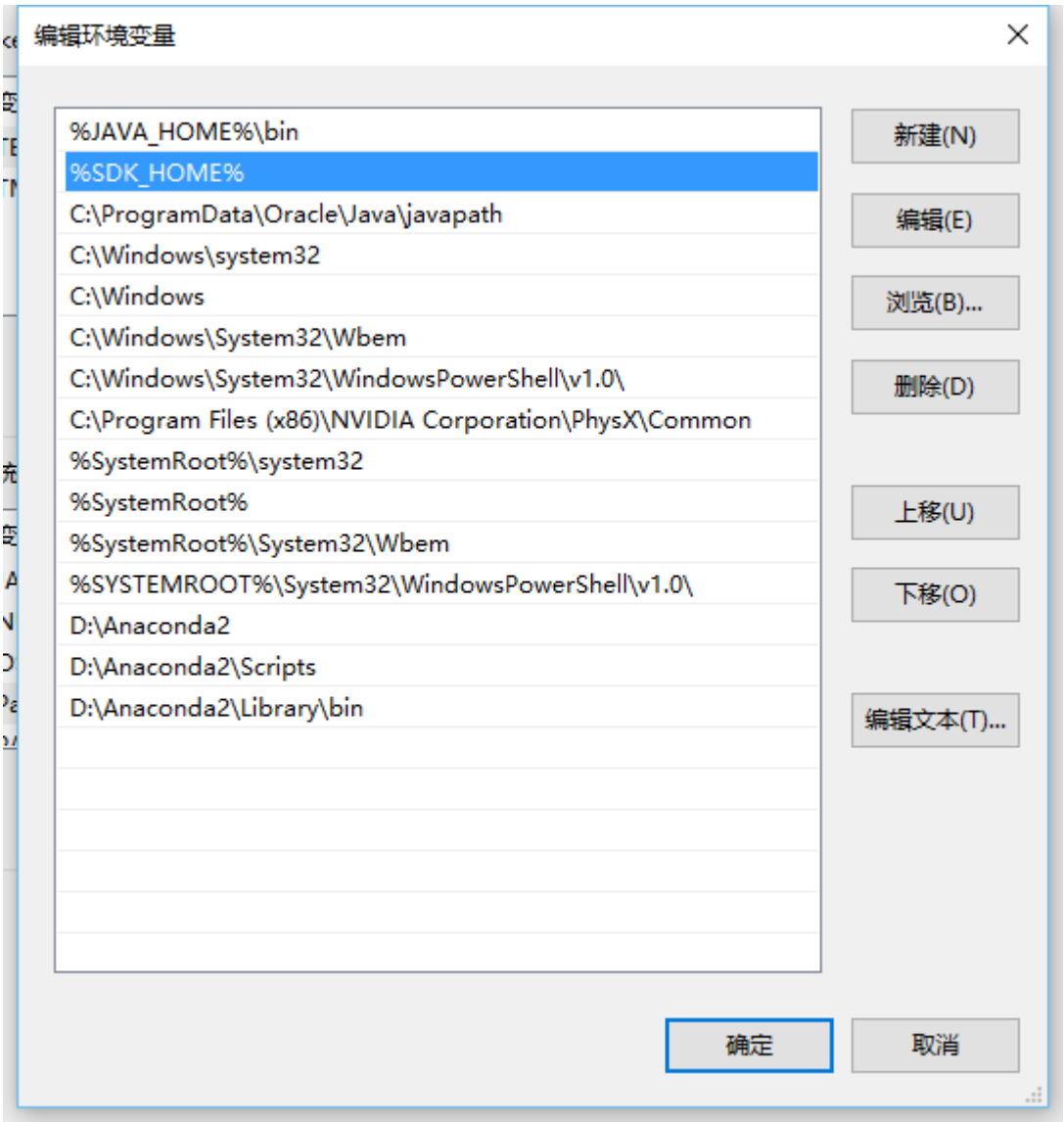


但是还有更为方便的方法，即通过 adb 工具来查看。

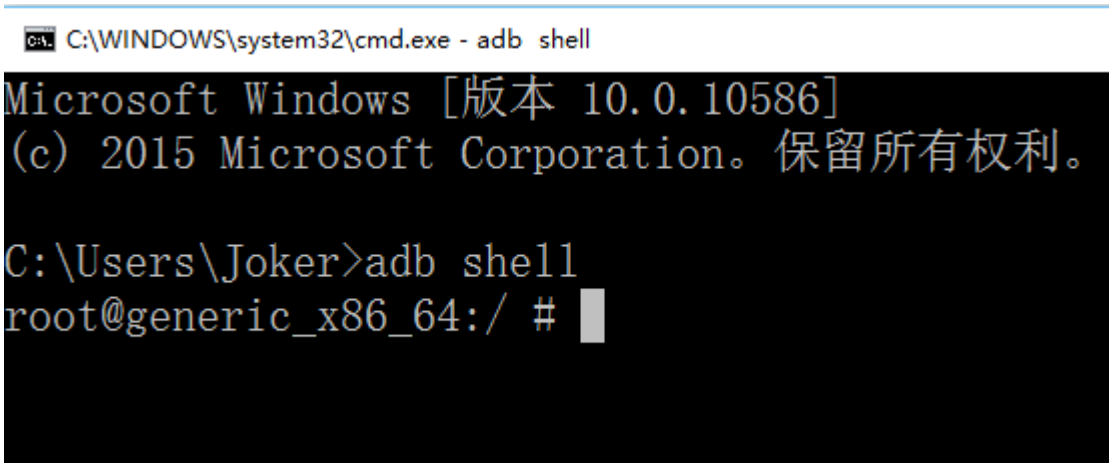
1.2 配置 adb 工具

1） 在 windows 下设置环境便令：SDK_HOME，在 path 里添加%SDK_HOME%，如下：





2） cmd+R 打开命令行，键入 adb shell，便可像操作 linux 系统一般进行操作：

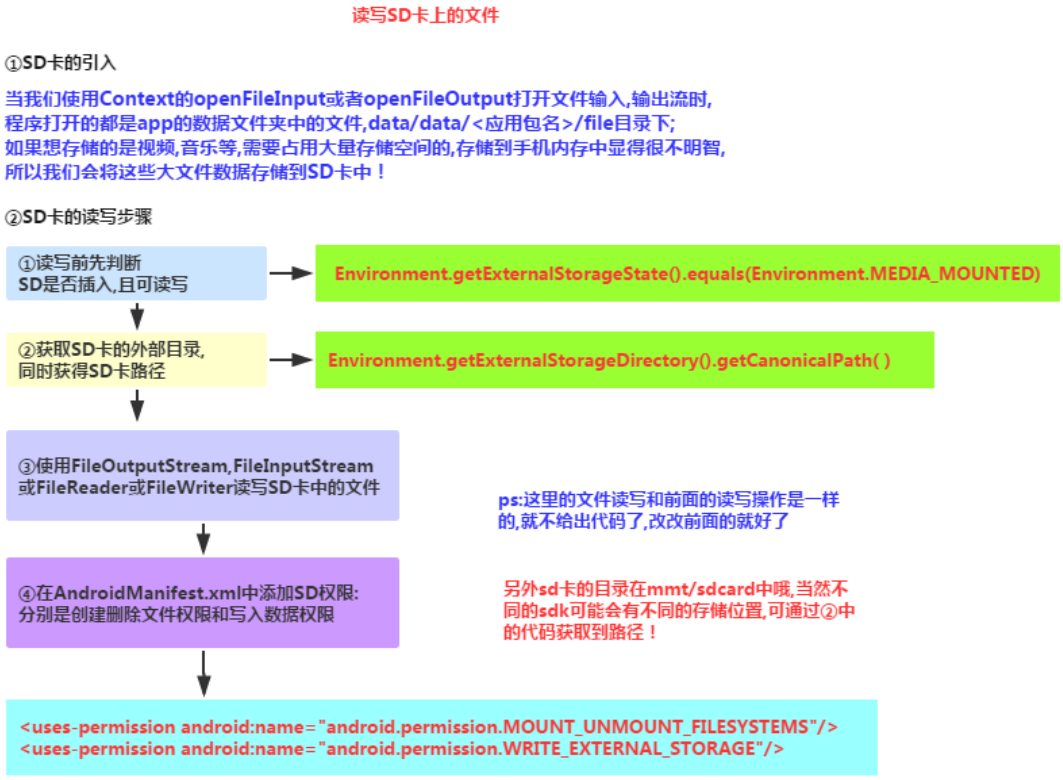


3） 键入 cd /data/data/YOUR APPLICATION ID/files 进入内部存储文件保存目录，ls 可以查看该目录下的所有文件，可以发现已经生成了前文中的 1.txt，cat 输出内容查看。

4） 键入 exit 退出 adb shell

1.3 外部存储存取

这里的外部存储，一般指 sdcard 存储。



Android 在 4.4 之后对外部存储的限制严格，需要在 AndroidManifest.xml 里申请权限：

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

在 6.0 上，需要在设置->应用->权限里手动打开存储权限，否则无法写入。

Step1.大致步骤与 1.1 相同，在进行写入和读取时候，需要显式地判断外部存储是否已经挂载和是否已经具备读写权限，之后再获取输入流和输出流进行读写；

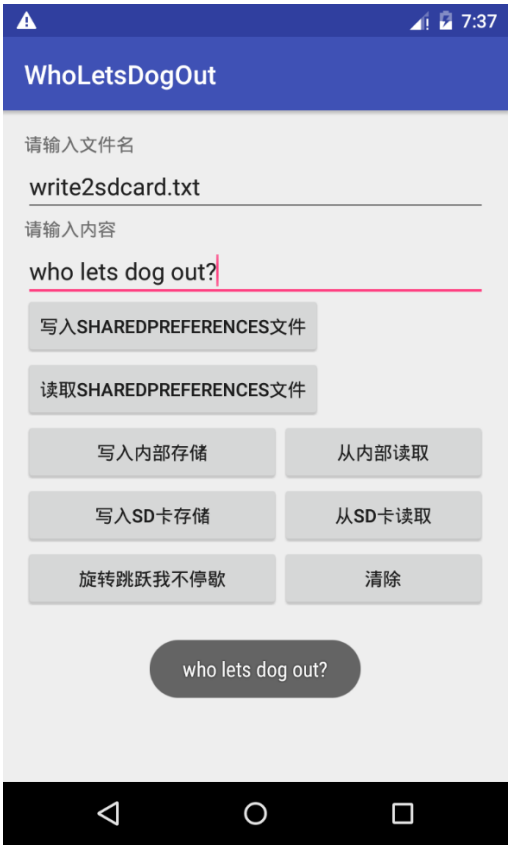
```
//如果手机已插入sd卡,且app具有读写sd卡的权限
if (Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)) {
    filename = Environment.getExternalStorageDirectory().getCanonicalPath() + "/" + filename;
    FileOutputStream output = new FileOutputStream(filename);
    output.write(filecontent.getBytes());
    //将String字符串以字节流的形式写入到输出流中
    output.close();
    //关闭输出流
} else
    Toast.makeText(context, "SD卡不存在或者不可读写", Toast.LENGTH_LONG).show();
```

同样我们可以采用 android device monitor 里的 ddms 或者用 adb shell 来进行查看，这里我们采用 ddms：

storage		2016-10-03	05:51	drwxr-x--x
sdcard		2016-10-03	07:39	drwxrwx--x
Alarms		2016-10-03	05:52	drwxrwx---
Android		2016-10-03	05:53	drwxrwx--x
DCIM		2016-10-03	05:56	drwxrwx---
Download		2016-10-03	05:52	drwxrwx---
LOST.DIR		2016-10-03	05:52	drwxrwx---
Movies		2016-10-03	05:52	drwxrwx---
Music		2016-10-03	05:52	drwxrwx---
Notifications		2016-10-03	05:52	drwxrwx---
Pictures		2016-10-03	05:52	drwxrwx---
Podcasts		2016-10-03	05:52	drwxrwx---
Ringtones		2016-10-03	05:52	drwxrwx---
write2sdcard.txt	17	2016-10-03	07:37	-rwxrwx---

可以看到，write2sdcard.txt 文件已经被保存在 /storage/sdcard 里了。如果采用 ddms 查看的同学，可以发现，整个文档结构里有很多指向该 sdcard 的链接。

Step2.同样我们可以点击读取数据 button，结果如下：



1.4sharedpreferences 文件的读写

SharedPreferences 即用户偏好参数，当我们的应用想要保存用户的一些偏好参数，比如是否自动登陆，是否记住账号密码,是否在 Wifi 下才能 联网等相关信息,如果使用数据库的话,显得有点大材小用了！我们把上面这些配置信息称为用户的偏好 设置，就是用户偏好的设置，而这些配置信息通常是保存在特定的文件中！比如 windows 使用 ini 文件， 而 J2SE 中使用 properties 属性文件与 xml 文件来保存软件的配置信息;而在 Android 中我们通常使用 一个轻量级的存储类——SharedPreferences 来保存用户偏好的参数！

SharedPreferences 也是使用 xml 文件, 然后类似于 Map 集合,使用键-值的形式来存储数据;我们只需要调用 SharedPreferences 的 getXxx(name), 就可以根据键获得对应的值！使用起来很方便！

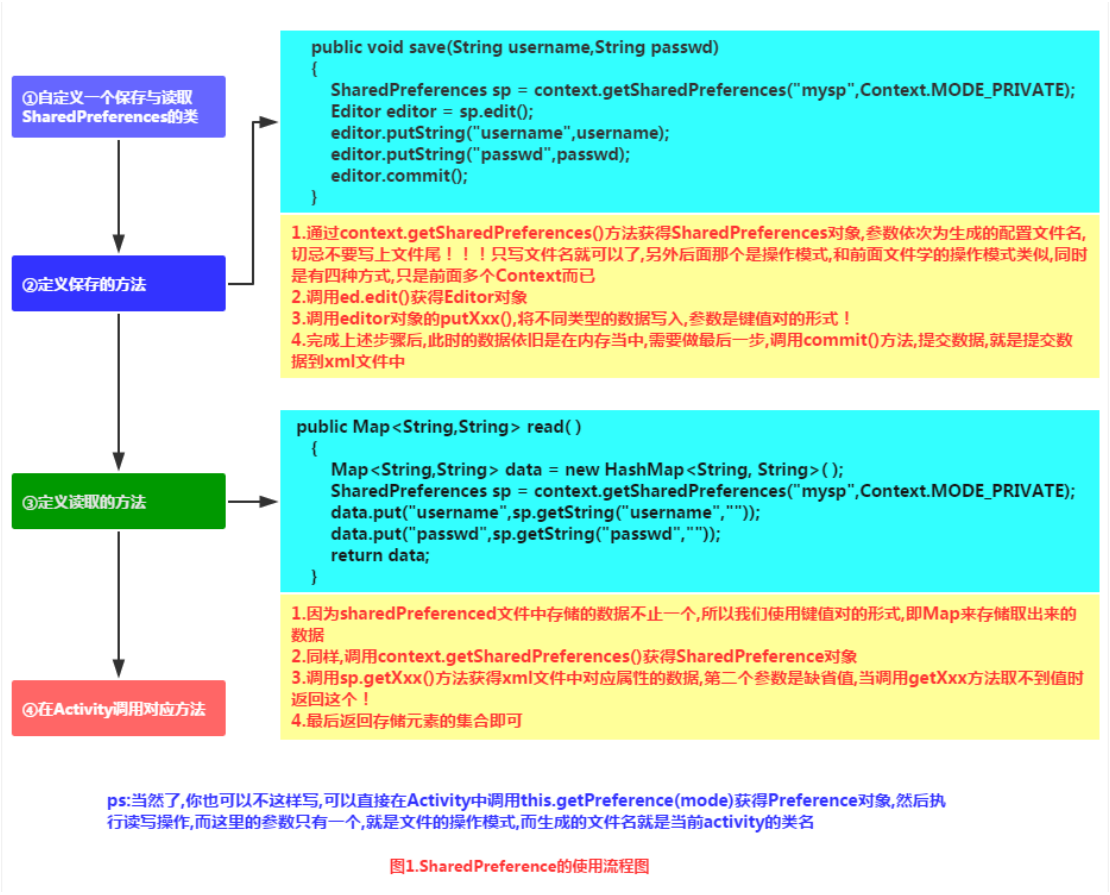


图1.SharedPreference的使用流程图

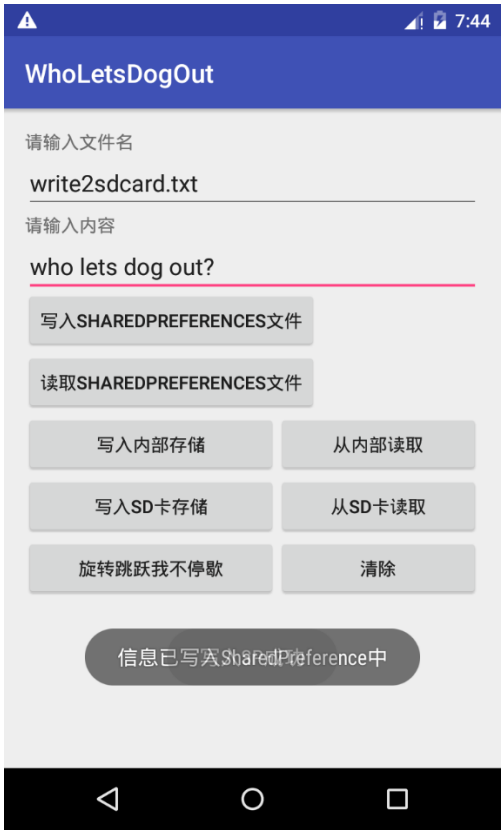
Step1. 类似于内部存储和外部存储读写中，建立两个 button：写入 sharedpreferences 文件和读取 sharedpreferences 文件；

Step2. 添加 button 的监听事件和处理方法：

对于写入 button：

```
case R.id.wspbutton:
    try {
        SharedPreferencesHelper sph = new SharedPreferencesHelper(mContext, "Test_SharedPreferencesFile");
        sph.addKeyValuePair("key1", "value1");
        sph.addKeyValuePair("key2", "value2");
        sph.addKeyValuePair(editTextName.getText().toString(),editTextContent.getText().toString());
        sph.save();
        Toast.makeText(mContext, "写入SP成功", Toast.LENGTH_SHORT).show();
    }catch (Exception e){
        e.printStackTrace();
        Log.i("Lin","Write SharedPreferences error "+ e.toString());
        Toast.makeText(mContext,"数据写入失败",Toast.LENGTH_LONG).show();
    }
    break;
```

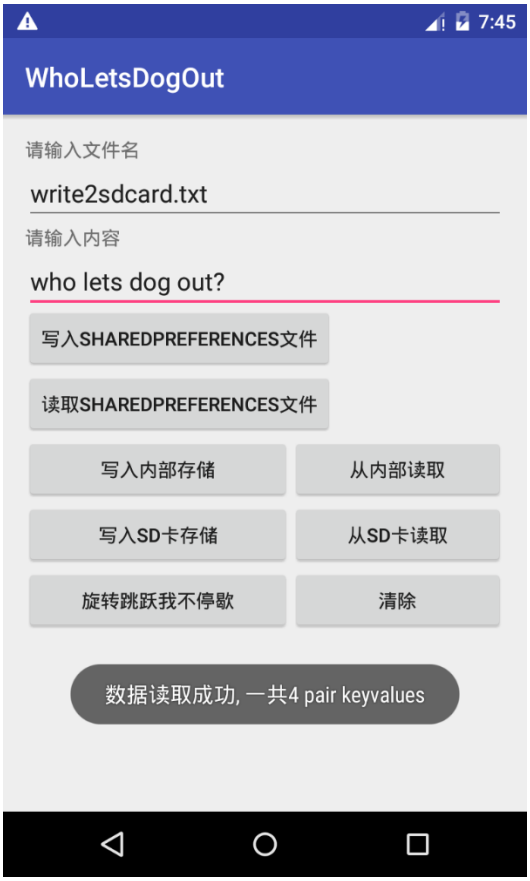
其中 SharedPreferncesHelper 是我们自定义的一个类，可以参见源码；



Step3 读取 sharedpreferences 文件内容：

```
case R.id.rspbutton:
    try {
        int cnt = 0;
        SharedPreferencesHelper sph2 = new SharedPreferencesHelper(mContext, "Test_SharedPreferencesFile");
        for (Map.Entry<String, String> entry : sph2.read().entrySet()) {
            cnt+=1;
            Log.i("Lin", entry.getKey() + " : " + entry.getValue());
        }
        Toast.makeText(mContext,"数据读取成功, 一共"+cnt+" pair keyvalues",Toast.LENGTH_SHORT).show();
        Toast.makeText(mContext,editTextName.getText().toString()+
                " : "+
                sph2.getSPValueByKey(editTextName.getText().toString()),
                Toast.LENGTH_SHORT).show();
    }catch (Exception e){
        e.printStackTrace();
        Log.i("Lin","Read from SharedPreferences error: "+e.toString());
        Toast.makeText(mContext,"数据读取失败",Toast.LENGTH_SHORT).show();
    }
    break;
```

其中，我们统计了 sharedpreferences 中键值对的个数，以及对应内容。如图所示：



2. sqlite 数据库

SQLite 数据库，和其他的 SQL 数据库不同， 我们并不需要在手机上另外安装一个数据库软件，Android 系统已经集成了这个数据库，我们无需像使用其他数据库软件(Oracle，MSSQL，MySql 等)又要安装，然后完成相关配置，又要改端口之类的！

2.1 基本概念

2.1.1 SQLite 是什么？为什么要用 SQLite？SQLite 有什么特点？

答：下面请听我们娓娓道来：

①SQLite 是一个轻量级的关系型数据库，运算速度快，占用资源少，很适合在移动设备上使用， 不仅支持标准 SQL 语法，还遵循 ACID(数据库事务)原则，无需账号，使用起来非常方便！

②前面我们学习了使用文件与 SharedPreference 来保存数据,但是在很多情况下， 文件并不一定是有效的,如多线程并发访问是相关的；app 要处理可能变化的复杂数据结构等等！ 比如银行的存钱与取钱！使用前两者就会显得很无力或者繁琐，数据库的出现可以解决这种问题， 而 Android 又给我们提供了这样一个轻量级的 SQLite，为何不用？

③SQLite 支持五种数据类型:NULL,INTEGER,REAL(浮点数),TEXT(字符串文本)和 BLOB(二进制对象) 虽然只有五种,但是对于 varchar,char 等其他数据类型都是可以保存的;因为 SQLite 有个最大的特点: 你可以各种数据类型的数据保存到任何字段中而不用关心字段声明的数据类型是什么,比如你 可以在 Integer 类型的字段中存放字符串,当然除了声明为主键 INTEGER PRIMARY KEY 的字段只能够存储 64 位整数！ 另外， SQLite 在解析 CREATE TABLE 语句时， 会忽略 CREATE TABLE 语句中跟在字段名后面的数据类型信息。如下面语句会忽略 name 字段的类型信息：

CREATE TABLE person (personid integer primary key autoincrement, name varchar(20))

小结下特点：

SQlite 通过文件来保存数据库，一个文件就是一个数据库，数据库中又包含多个表格，表格里又有 多条记录，每个记录由多个字段构成，每个字段有对应的值，每个值我们可以指定类型，也可以不指定 类型(主键除外)

PS：对了，Android 内置的 SQLite 是 SQLite 3 版本的~

2.1.2 几个相关的类：

嘿嘿，学习一些新东西的时候，最不喜欢的莫过于遇到一些新名词，是吧，我们先来说下几个 我们在使用数据库时用到的三个类：

SQLiteOpenHelper：抽象类，我们通过继承该类，然后重写数据库创建以及更新的方法， 我们还可以通过该类的对象获得数据库实例，或者关闭数据库！

SQLiteDatabase：数据库访问类：我们可以通过该类的对象来对数据库做一些增删改查的操作

Cursor：游标，有点类似于 JDBC 里的 resultset，结果集！可以简单理解为指向数据库中某 一个记录的指针！

2.2 使用 SQLiteOpenHelper 类创建数据库与版本管理

对于涉及数据库的 app,我们不可能手动地去给他创建数据库文件,所以需要在第一次启用 app 的时候就创建好数据库表;而当我们的应用进行升级需要修改数据库表的结构时,这个时候就需要 对数据库表进行更新了;对于这两个操作,安卓给我们提供了 SQLiteOpenHelper 的两个方法, onCreate( )与 onUpgrade( )来实现

方法解析：

onCreate(database):首次使用软件时生成数据库表

onUpgrade(database,oldVersion,newVersion):在数据库的版本发生变化时会被调用， 一般在软件升级时才需改变版本号，而数据库的版本是由程序员控制的，假设数据库现在的版本是 1，由于业务的变更，修改了数据库表结构，这时候就需要升级软件，升级软件时希望 更新用户手机里的数据库表结构，为了实现这一目的，可以把原来的数据库版本设置为 2 或者其他与旧版本号不同的数字即可！

2.3 代码步骤示例

Step1. 创建新的一个 activity，在首界面中加入一个 button 进行跳转：

```
case R.id.jumpToSQliteActivity:
    startActivity(new Intent(MainActivity.this, SQLiteActivity.class));
    Toast.makeText(mContext,"旋转~跳跃~",Toast.LENGTH_SHORT).show();
    break;
```

Step2. 我们在该示例中，主要过程是建立一个 student 表，表项有自增的 id，name 和 score 项，另外，为了下一步的存储二进制文件的示例，我们还建立了一个 blob 类型的项：

```
class MyDBOpenHelper extends SQLiteOpenHelper {
    public MyDBOpenHelper(Context context, String name,
                          SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        String sqlstr = "CREATE TABLE student(stuId INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "name VARCHAR(20), "+
                "score INTEGER, " +
                "img BLOB)";
        db.execSQL(sqlstr);
        Log.i("SQLite",sqlstr);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE student");
        db.execSQL("CREATE TABLE student(stuId INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "name VARCHAR(20), " +
                "score INTEGER");
    }
}
```

Step3.点击 插入数据，则插入一条编辑框里对应的内容：

```
case R.id.btn_insert:
    ContentValues values1 = new ContentValues();
    String name1 = nameEditView.getText().toString();
    int score1 = Integer.parseInt(scoreEditView.getText().toString());
    values1.put("name",name1);
    values1.put("score", score1);
    db.insert("student",null,values1);
    Toast.makeText(mContext,"插入数据完毕",Toast.LENGTH_SHORT).show();
    break;
```

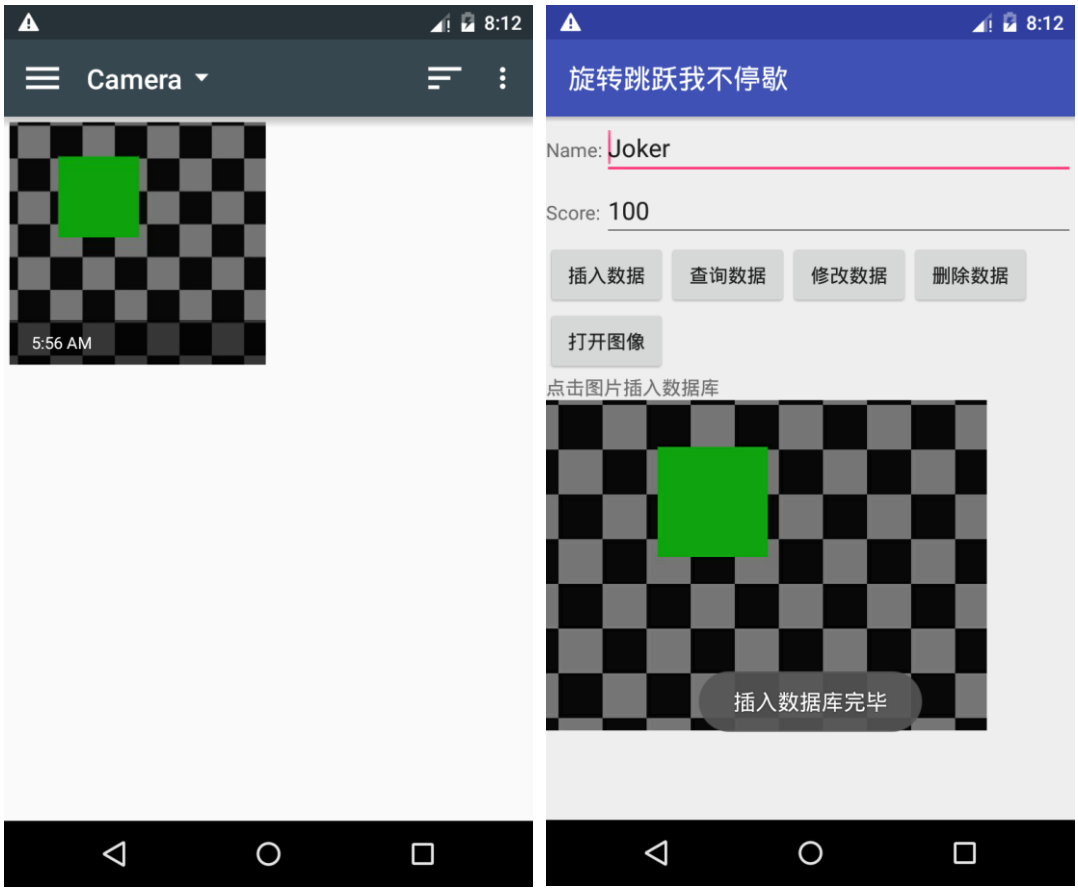
Step4.插入数据之后，便可以点击查询数据进行查询：

```
case R.id.btn_query:
    StringBuilder sb = new StringBuilder();
    Cursor cursor = db.query("student",null,null,null,null,null,null);
    if(cursor.moveToFirst()){
        do{
            int stuId = cursor.getInt(cursor.getColumnIndex("stuId"));
            String name = cursor.getString(cursor.getColumnIndex("name"));
            int score = cursor.getInt(cursor.getColumnIndex("score"));
            sb.append("stuId: "+stuId+": "+name+": "+score+"\n");
        }while (cursor.moveToNext());
    }
    cursor.close();
    Toast.makeText(mContext,sb.toString(),Toast.LENGTH_SHORT).show();
    break;
```

Step5.也可以根据 name 来查找对应的 student 记录，并更新其 score 值：

```
case R.id.btn_update:
    ContentValues values2 = new ContentValues();
    values2.put("name", nameEditView.getText().toString());
    values2.put("score", Integer.parseInt(scoreEditView.getText().toString()));
    db.update("student",values2,"name = ?",new String[]{nameEditView.getText().toString()});
    break;
```

Step6.点击打开图像，选择一张图片，然后显示在屏幕上，点击该图片，将其存储在 sqlite 数据库中：



Step7.这回我们通过 adb shell 进行查看，

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.10586]
(c) 2015 Microsoft Corporation。保留所有权利。

C:\Users\Joker>adb shell
 /data/data/com.darkknight.joker.toasttest/databases/
root@generic_x86_64:/data/data/com.darkknight.joker.toasttest/databases #
pwd
/data/data/com.darkknight.joker.toasttest/databases
root@generic_x86_64:/data/data/com.darkknight.joker.toasttest/databases #
ls
test.db
test.db-journal
root@generic_x86_64:/data/data/com.darkknight.joker.toasttest/databases #
sqlite3 tes
test.db         test.db-journal
sqlite3 test.db
SQLite version 3.8.6.1 2015-05-21 17:24:32
Enter ".help" for usage hints.
sqlite> .table
android_metadata  student
sqlite> select * from student;
4|Joker|100|
5|Joker|100|PNG

6|Joker|100|PNG


sqlite> .exit
root@generic_x86_64:/data/data/com.darkknight.joker.toasttest/databases #
exit
```

可以看到，select * from student 里，有 PNG 格式图片，可为什么是 PNG 格式呢？是因为我们在存储到 sqlite 数据库的时候，显式地将其压缩成 png 格式：

```
case R.id.imgview:
    Toast.makeText(mContext,"插入数据库完毕",Toast.LENGTH_SHORT).show();
    try{
        ByteArrayOutputStream outs = new ByteArrayOutputStream();
        ((BitmapDrawable)(imgView.getDrawable())).getBitmap().compress(
                Bitmap.CompressFormat.PNG,100,outs
        );
        Object[] args = new Object[]{
                nameEditView.getText().toString(),
                Integer.parseInt(scoreEditView.getText().toString()),
                outs.toByteArray()
        };
        db.execSQL("INSERT INTO student(name,score,img) values(?,?,?)",args);
        outs.close();
    }catch (Exception e){
        e.printStackTrace();
        Log.e(TAG,e.toString());
    }
    break;
```

那打开图片选择的界面是怎么做的呢？也很简单，通过一个 intent action：

Intent.ACTION_OPEN_DOCUMENT 即可打开 documentUI，进行选择：

```
case R.id.btn_openImg:
    Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);
    intent.addCategory(Intent.CATEGORY_OPENABLE);
    intent.setType("image/*");
    startActivityForResult(intent, READ_REQUEST_CODE);
    break;
```

另外，startActivityForResult 方法，定义了一个方法，使得当选择完图片之后，能够根据 request code 来调用回调函数，进行不同的操作，在本处，是将其内容写入到 imageview 里，具体的存储到 sqlite 的操作，是 imageview 的监听事件完成的：

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent resultData){
    if(requestCode == READ_REQUEST_CODE && resultCode == Activity.RESULT_OK){
        if(resultData!=null){
            Uri uri = resultData.getData();
            Log.i(TAG,uri.toString());
            AsyncTask<Uri, Void, Bitmap> imageLoadAsyncTask = new AsyncTask<Uri, Void, Bitmap>()
            {
                @Override
                protected Bitmap doInBackground(Uri... uris) {
                    dumpImageMetaData(uris[0]);
                    return getBitmapFromUri(uris[0]);
                }

                @Override
                protected void onPostExecute(Bitmap bitmap) {
                    txtView.setText("点击图片插入数据库");
                    imgView.setImageBitmap(bitmap);
                }
            };
            imageLoadAsyncTask.execute(uri);
        }
    }
}
```

到此，我们整个实验 3 便结束了。