JAVA环境变量配置：

CLASSPATH：.;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar

JAVA\_HOME:E:\Java\jdk1.8.0\_92

Path:\;E:\Java\jdk1.8.0\_92\bin;E:\Java\jdk1.8.0\_92\jre\bin;

adb 端口检查 netstat -aon|findstr "5037"

根据第一行占用号码 tasklist | findst ”xxxx”

活动的四种启动模式 1、standard 默认模式 无论活动在返回栈中是否已存在，都新建一个实例，并处于栈顶位置。2、singleTop 当活动处于栈顶时，不创建新实例，（会调用onPause和onResume 方法）但不处于栈顶时，无论其是否存在 都创建新实例。3、singleTask 当栈中已存在活动时，将其推至栈顶，并使其上的活动都出栈，没有的话创建新实例。4、singleInstance 为当前活动独立创建一个返回栈，可以实现活动的程序共享。

android:background=*"@android:drawable/editbox\_background"//*使输入框和背景有区分

android:layout\_alignTop=”” //使控件顶部与所选控件对齐，否则可能出现水平位置控制实现但竖直方向上不齐

android:layout\_column=*"3"* 表示单元格列坐标为3，注意当前方坐标不全（如现在没有坐标2时），该单元格会跟在1或0后 ，或干脆就作为首格。同一行同列坐标则依次排序。

*requestWindowFeature(Window.FEATURE\_NO\_TITLE);//*隐藏标题栏

android:layout\_margin=*"5dip"* //设置边距

android:layout\_span="2" //单元格合并

android:stretchColumns="1" //第二行拉伸至布满当前布局

android:scaleType="fitXY"//设置图片相对布局的尺寸改变方式

android:fitSystemWindow=”true” //设置Content不超过StatusBar和NavigationBar

//关于布局单位

px 是像素的意思，即屏幕中可以显示的最小元素单元，我们应用里任何可见的东西都

是由一个个像素点组成的。单独一个像素点非常的微小，肉眼是无法看见的，可是当许许多

多的像素点聚集到一起时，就可以拼接成五彩缤纷的图案。

pt 是磅数的意思，1 磅等于 1/72 英寸，一般 pt 都会作为字体的单位来使用。

dp 是密度无关像素的意思，也被称作 dip，和 px 相比，它在不同密度的屏幕中的显示比

例将保持一致。

sp 是可伸缩像素的意思，它采用了和 dp 同样的设计理念，解决了文字大小的适配问题。

这里有一个新名词需要引起我们的注意，什么叫密度？Android 中的密度就是屏幕每英

寸所包含的像素数，通常以 dpi为单位。比如一个手机屏幕的宽是 2英寸长是 3英寸，如果

它的分辨率是 320\*480 像素，那这个屏幕的密度就是 160dpi，如果它的分辨率是 640\*960，

那这个屏幕的密度就是 320dpi，因此密度值越高的屏幕显示的效果就越精细。

//进度条控件

ProgressBar

style="?android:attr/progressBarStyleHorizontal" //设置进度条为方形

android:max="100"设置进度条最大值

//带进度条的对话框控件

ProgressDialog progressDialog = new ProgressDialog

(MainActivity.this);

progressDialog.setTitle("This is ProgressDialog");

progressDialog.setMessage("Loading...");

progressDialog.setCancelable(true);

progressDialog.show();

//菜单的设置

@Override  
public boolean onCreateOptionsMenu(Menu menu){  
 menu.add("MatTest");  
 return true;  
}  
@Override  
public boolean onOptionsItemSelected(MenuItem item){

Return true;

}

//设置view控件在不触发时自动消失的方法：

//在onTouchEvent或者onUp中延时发出控件消失命令

view.clearAnimation()//预先打断正在进行的动画效果，以免在动画显示过程中触发点击事件导致结果有误

Message msg = handler.obtainMessage(1);

currentTime = System.currentTimeMillis(); //用时间来设置标志位,此时currentTime应该是个全局变量，这样在消息延迟的时间里若有其他的点击事件，可以在handleMessage方法中判断触发事件进程是否需要打断

Bundle bundle = new Bundle();

bundle.putLong("currentTime", currentTime);

msg.setData(bundle);

handler.sendMessageDelayed(msg, 3000);//3000为延时时长

//Handle端的处理过程

private Handler handler = new Handler() {

    public void handleMessage(android.os.Message msg) {

        if (msg.what == 1) {

            if (msg.getData().getLong("currentTime") == currentTime){

                view.startAnimation(disappearAnimation);

//disappeaAnimation是自定义的消失动画，需要在该动画播放完之后监听语句里将空间状态设为不可见

}

//使特定界面在全局显示的方式——在后台服务中设置一个全局化的悬浮窗口

mWindowManager=(WindowManager)getApplication().getSystemService(getApplication().***WINDOW\_SERVICE***);//获取窗口管理权限

//设置窗口参数

mParams=**new** WindowManager.LayoutParams();

mParams.type=LayoutParams.***TYPE\_SYSTEM\_OVERLAY***;//最重要的一步：设置窗口模式，该模式下窗口将在所有界面中存在且无法获取焦点，即成为一个固定背景 该模式需要配置权限”android.permission.SYSTEM\_ALERT\_WINDOW”;

mParams.format=PixelFormat.***RGBA\_8888***;//设置窗口色彩风格

mParams.flags=LayoutParams.***FLAG\_NOT\_FOCUSABLE***;//设置不可聚焦

mParams.alpha=0.3f;//设置窗口透明度

mParams.gravity=Gravity.***LEFT***|Gravity.***TOP***;//设置窗口位置

//设置位置的相对起点终点

mParams.x=0;

mParams.y=0;

mParams.width=WindowManager.LayoutParams.***MATCH\_PARENT***;

mParams.height =WindowManager.LayoutParams.***MATCH\_PARENT***;

//导入准备好的特定view

LayoutInflater inflater=LayoutInflater.*from*(getApplication());

//这里有一个小问题，即ImageButton这种view并不能完全铺满整个平面，虽然一般都不太会这样做，但是仍要注意 mFloatLayout=(LinearLayout)inflater.inflate(R.layout.***float\_layout***,**null**);

mWindowManager.addView(mFloatLayout, mParams);//设置其为窗口view

//为了提升控件与使用者的交互式体验，需要进行的一些小的处理

view.setFocusable(true);view.requestFocus()//为当前空间获取焦点，可用于EditText框中内容需要输出时。

<receiver android:name=*".MyBroadcastReceiver"*>

<intent-filter >

<action android:name=*"com.example.broadcasttest.MY\_BROADCAST"*/>

</intent-filter>

</receiver>//广播注册方式

//要在广播中创建新活动 需要对活动的Flag进行设置 创建对话框也需要设置特定类型

intent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

alertDialog.getWindow().setType(WindowManager.LayoutParams.TYPE\_SYSTE

M\_ALERT);

//文件可持续化相关

SharedPreferences.Editor editor =Context.getSharedPreferences("data", Context.MODE\_PRIVATE).edit(); //创建一个SharePreference.Editor 对象，来向文件写数据 MODE\_PRIVATE表示其只接受单线程操作。

editor.putString("name", "Tom"); //采用键值对形式存储数据，类型在put后声明，保存方式为xml格式。

取值时只要创建一个SharedPreference对象就可以使用getString等方法获得存入的值，参数为键和不存在该键时的默认值。

//提交数据时有commit和apply两种方式，差别在于前者有返回值，但是写入磁盘时需要等待未完成的任务，后者无返回值且异步提交，可以加快速度，但易覆盖数据。

//用IO流存储与读取文件 需要抛出异常 并在finally中close相应的缓冲区

FileInputStream in=openFileInput("data");

BufferedReader reader=**new** BufferedReader(**new** InputStreamReader(in));

String line="";

**while**((line=reader.readLine())!=**null**){

StringBuilder content.append(line);}

FileOutputStream out = openFileOutput("data", Context.MODE\_PRIVATE); //PRIVATE模式下后面写入的数据会覆盖前面的数据，当MODE为APPEN时才会进行追加操作。

BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(out));

writer.write(inputText);

数据库创建时可借助SQLiteOpenHelper,其传入的参数中包括Context,数据库名称,cursor对象和版本号。调用getReadableDatabase()或者getWriteableDatabase()函数时，若相应名称的数据库不存在，则会根据名称新建一个数据库，并返回一个可操作的SQLiteDatabase对象（前者可用于空间已满的时候，此时返回的数据库对象为只读，后者更多地用于对数据库进行操作，除了空间已满时，两者一般情况下返回的对象是一致的）。

//数据库建表语句 primary key表示设为主键 autoincrement表示自增长，text为文本类型，real为浮点型 varchar(?)为?位字符型

public static final String CREATE\_BOOK = "create table book ("

+ "id integer primary key autoincrement, "

+ "author text, "

+ "price real, "

+ "pages integer, "

+ "name text)";

//在android函数中的建表操作

@Override

public void onCreate(SQLiteDatabase db) {

db.execSQL(CREATE\_BOOK);

}

//删表操作

db.execSQL("drop table if exists Book");

//数据库的增删改查

ContentValues作为数据的容器，以put()方法通过键值对的方式存入对象中，然后作为参数传入SQLiteDatabase的insert()方法或update()方法中，表示要增或者改的数据。删和改两种方法传入的参数中还有类似一个给出条件形式，另一个参数表示条件具体值的样式，下面为三种方法的示例语句。values.put("name", "The Da Vinci Code");//ContentValue对象的数据添加

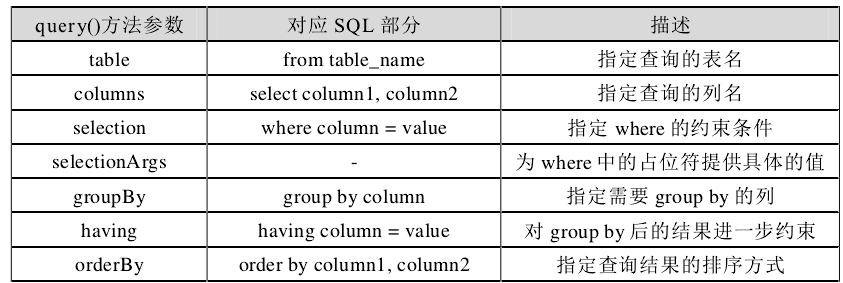
db.insert("Book", null, values);//增加数据

db.update("Book", values, "name = ?", new String[] { "The Da

Vinci Code" }); //改数据

db.delete("Book", "pages > ?", new String[] { "500" });//删数据

而query()方法相对来说较为复杂，其最少包含7个参数，表示意义如下



若要简单起见的话，可以只填表名，其他参数全部设为null，即遍历所有表中行和列，内容存在Cursor对象中返回，而Cursor对象通过getColumnIndex()方式取出相应值，如：

Cursor cursor = db.query("Book", null, null, null, null, null, null);

if (cursor.moveToFirst()) {

do{

String name = cursor.getString(cursor. getColumnIndex("name"));

}while(cursor.moveToNext());

}cursor.close();

//数据库高级用法

事务：事务开启后，除非事务内所有程序全部执行完毕到事务结束，否则已执行语句并不会生效，可以保证数据的修改是同步的，不会因为异常造成数据损失。

db.beginTransaction(); // 开启事务

db.setTransactionSuccessful(); // 事务已经执行成功

db.endTransaction(); // 结束事务

关于数据库更新：可以在onUpgrade()函数中写一个switch语句，根据旧版本的版本号做相应的新版本更新，这样可以防止新数据无法写入，并且不会影响原有的有用数据，影响用户体验。

例：

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

switch (oldVersion) {

case 1:

db.execSQL(CREATE\_CATEGORY);

case 2:

db.execSQL("alter table Book add column category\_id integer");

default:

}

} //注意各项case中没有break符号，这样一来由低版本越版本更新时前面版本的更新都会被执行。

//数据库文件command命令行查看

进入数据库文件所在地，输入sqlite3 文件名 即进入操作模式

输入.tables可以查看当前文件内的表，.scheme可以查看各个表的建表类型，select\*from 表名;可以查看表内内容 Window下chcp 65001命令可以改变字符集为UTF-8,否则查找到的表中文可能为乱码。

//CotentProvider 的注册写法，用于应用间提供数据内容

<provider

android:name=*"com.example.databasetest.DatabaseProvider"*

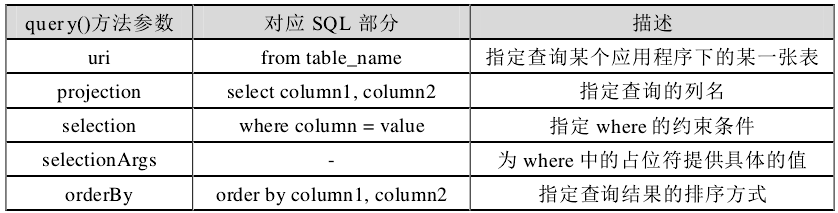
android:authorities=*"com.example.databasetest.provider"*

android:exported = *"true"* >

</provider>

实际程序进行数据处理的时候 用的是ContentResolver，该类对象可通过Context的getContentResolver()方法得到，对数据增删改查时其相当于SQLite中的SQLiteDatabase对象，参数也基本一致，但不再以表名为参数而以uri为参数，其中具体的数据数值仍然用ContentValue来存放。

注：ContentResolver的query方法参数意义



//Uri的表示

1. \*：表示匹配任意长度的任意字符

2. #：表示匹配任意长度的数字

所以，一个能够匹配任意表的内容 URI格式就可以写成：

content://com.example.app.provider/\*

协议声明 权限 路径

而一个能够匹配 table1表中任意一行数据的内容 URI格式就可以写成：

content://com.example.app.provider/table1/#

//data标签的写法 其表达方式其实就是一个Uri

<data>标签中主要可以配置以下内容。

1. android:scheme

用于指定数据的协议部分，如上例中的 http部分。

2. android:host

用于指定数据的主机名部分，如上例中的 [www.baidu.com](http://www.baidu.com/)部分。

3. android:port

用于指定数据的端口部分，一般紧随在主机名之后。

4. android:path

用于指定主机名和端口之后的部分，如一段网址中跟在域名之后的内容。

5. android:mimeType

用于指定可以处理的数据类型，允许使用通配符的方式进行指定。

1. 如果data的URI和datatype为空，则 filter 的URI和type也必须为空，才能匹配成功2. 如果data的URI不为空，但是datatype为空，则 filter 必须定义URI并匹配成功，且type为空，才能匹配成功3. 如果data的URI为空，但是datatype不为空，则 filter 必须URI为空，定义type并匹配成功，才能匹配成功4. 如果data的URI和data都不为空，则 filter 的URI和type都必须定义并匹配成功，才能匹配成功。

对于URI部分，有一个特殊处理，就是即使filter没有定义URI，content和file两种URI也作为既存的URI存在。

只有<data>标签中指定的内容和 Intent 中携带的 Data 完全一致时，当前活动才能够响应

该 Intent。

//通知的创建方法

NotificationManagermanager=(NotificationManager)getSystemService(***NOTIFICATION\_SERVICE***);

Notification notification=**new** Notification.Builder(MainActivity.**this**)

.setContentTitle("This is title")

.setContentText("Not much")

.setSmallIcon(R.drawable.***ic\_launcher***)

.setLargeIcon(**null**)

.setContentIntent(pendingIntent).build();

manager.notify(1,notification);//启动一个通知，Id为1

startForeground(1,notification);//启动一个前台服务的写法

PendingIntent pendingIntent=PendingIntent.*getActivity*(MainActivity.**this**,0,intent,PendingIntent.***FLAG\_CANCEL\_CURRENT***);//pendingIntent 为推送消息的点击事件页面，含于intent中

//音频和视频的播放

分别借助MediaPlayer 和VideoView，两者都需要准备和初始化动作 ，并在关闭程序时需要调用释放资源的函数，前者为release() 后者为suspend()

//关于Git版本控制工具

建议打开Git Bash使用命令行模式操作

初始命令：git config --global user.name/user.email values //设置发布者名称或邮箱

到项目文件夹底部 输入git init //创建一个空的git代码仓库

输入 ls -al //查看该项目的所有git操作

输入 git add file/document/. //将文件或文件夹添加进欲提交内容里 “.”为全部添加

输入 git commit -m note //提交已添加的内容，并对此次提交行为提供注释

在项目根目录 输入 touch .gitignore //创建名为.gitignore的文本文件

编辑该文本文件 如输入 /bin （/r/n） /gen 则用git .命令提交全部文件时，它会按行读取.gitignore中的数据，并忽略其注明的文件夹

输入 git status //可以查看提交代码后 原代码文件是否遭到修改

输入 git diff //可以看到所有文件的修改内容 若要看单个文件的修改内容，可以在其后加上绝对文件路径

输入 git reset HEAD （absolute path）//可以取消该绝对路径下文件的add

输入 git checkout //可以撤销未被add的文件修改 后加修改文件的绝对路径

输入 git log //可以查看提交记录 后加提交ID -l可查看单条记录 再加-p可查看修改的内容

输入 git branch （分支名）//可以开辟分支 用于对原有版本的维护和修改

输入 git branch -a//可以查看现有分支

输入 git checkout （分支名）//可以从当前分支跳到新分支

输入 git merge （分支名）//可以合并该版本修改信息

输入 git branch -D (分支名)//可以删除该分支

输入 git clone （远程版本库地址） //可以从远程版本库获得代码

输入 git push origin （分支名）// 可以将修改过的远程版本库获得的代码上传至远程代码库指定分支

输入 git fetch origin（分支名）//可以同步远程版本库的代码 存储于名为origin/（分支名）的分支上

输入 git pull origin（分支名）//可以同步远程版本库的代码并且与本地分支合并