**android.app.ActionBar类**

extends Object

早期activity中的toolbar，其中可能显示activityde biaoti 、应用层级导航键和其他项目。

**android.app.ActionBar.setHomeButtonEnabled方法**

public void setHomeButtonEnabled (boolean enabled)

设置actionbar角上的home按钮是否可用。(Note that this is the application home/up affordance on the action bar, not the systemwide home button.)

This defaults to true for packages targeting < API 14. For packages targeting API 14 or greater, the application should call this method to enable interaction with the home/up affordance.

Setting the DISPLAY\_HOME\_AS\_UP display option will automatically enable the home button.

##### Parameters

##### enabled truehome按钮可用，false不可用。

**android.app.ActionBar.setDisplayHomeAsUpEnabled方法**

public abstract void setDisplayHomeAsUpEnabled (boolean showHomeAsUp)

（wpf:此方法可显示home键）

设置home按钮是否显示为”up”启动器。Set this to true if selecting "home" returns up by a single level in your UI rather than back to the top level or front page.

To set several display options at once, see the setDisplayOptions methods.

##### Parameters showHomeAsUp 设置为true来展示给用户选择home键将返回一个层级，而不是返回应用顶级。

##### See Also

setDisplayOptions(int)

setDisplayOptions(int, int)

**android.app.Activity类**

extends ContextThemeWrapper

implements ComponentCallbacks2 KeyEvent.Callback LayoutInflater.Factory2 View.OnCreateContextMenuListener Window.Callback



**android.app.Activity.findViewById方法**

public View findViewById (int id)

找到一个被被在onCreat(Bundle)中处理的XML中id属性定义的视图。

Returns

被找到的视图，否则为null。

**android.app.Activity.getFragmentManager方法**

public FragmentManager getFragmentManager ()

返回与activity联系的fragment相互作用的FragmentManager。

**android.app.Activity.getIntent方法**

public Intent getIntent ()

返回开始这个activity的intent。

**android.app.Activity.getPreference方法**

public SharedPreferences getPreferences (int mode)

检索与此activity联系的内部SharedPreferences对象This simply calls the underlying getSharedPreferences(String, int) method by passing in this activity's class name as the preferences name.

##### Parameters

##### mode 操作模式。使用MODE\_PRIVATE进行默认操作；MODE\_WORLD\_READABLE和MODE\_WORLD\_WRITEABLE来控制权限。

##### Returns

Returns the single SharedPreferences instance that can be used to retrieve and modify the preference values.

**android.app.Activity.getTaskId方法**

public int getTaskId ()

返回这个activity所在任务的标识符。这个标识符在activity的生命周期中保持相同。

Returns

任务标识符，一个不透明的整型。

**android.app.Activity. onActivityResult方法**

protected void onActivityResult (int requestCode, int resultCode, Intent data)

当一个你发动的activity退出时调用，给你开始它时的requestCode、它返回的resultCode和从它得到的任何数据。如果activity没有返回任何结果，或者在运行期间崩溃，resultCode将会明确返回RESULT\_CANCELED。

当你的activity重新开始，你将在onResume()之前立刻接收它。

Parameters

requestCode 先前传递到startActivityForResult()中的整型结果码。

resultCode 从子activity的setResult()返回的整型结果码。

data 一个Intent，它能返回结果数据（变量数据可以附加到Intent的extras）。

See Also

startActivityForResult(Intent, int)

createPendingResult(int, Intent, int)

setResult(int)

**android.app.Activity. onConfigurationChanged方法**

public void onConfigurationChanged (Configuration newConfig)

当antivity运行时设备设置改变时由系统调用。Note that this will only be called if you have selected configurations you would like to handle with the configChanges attribute in your manifest. If any configuration change occurs that is not selected to be reported by that attribute, then instead of reporting it the system will stop and restart the activity (to have it launched with the new configuration).

At the time that this function has been called, your Resources object will have been updated to return resource values matching the new configuration.

##### Parameters

##### newConfig 新的configuration。

**android.app.Activity. onCreate方法**

protected void onCreate (Bundle savedInstanceState)

当activity开始时调用。它应当运行大部分的初始化工作：调用setContentView(int)来填充activity的UI，使用findViewById(int)来编程与UI中装饰物交互，调用managedQuery(android.net.Uri,String,String,String[],String)来检索光标用于数据显示，等等。

你可以在这个函数中调用finish()，以这种方法onDestroy()将立即被调用而不需要调用activity的其他生命过程（onStart(),onResume(),onPause(),etc）。

派生类必须通过父类的实现方法调用此函数。否则将抛出异常。

Parameters

savedInstanceState 如果activity在早先关闭后被重新初始化，那么这个Bundle包含最近由onSaveInstanceState(Bundle)提供的数据。记住：否则它将为空。

See Also

onStart()

onSaveInstanceState(Bundle)

onRestoreInstanceState(Bundle)

onPostCreate(Bundle)

**android.app.Activity. onCreateOptionsMenu方法**

public boolean onCreateOptionsMenu (Menu menu)

初始化activity的标准选项菜单目录。你应该放置你的菜单项目到菜单中。

它只被调用一次，第一次选项菜单被显示。它显示时要更新菜单，查看onPrepareOptionsMenu(Menu)。

默认实现用标准系统菜单项填充。这些被放在CATEGORY\_SYSTEM组中，因此他们将被用应用决定的菜单项正确填充。派生类应到总是通过基础实现调用。

你将安全控制菜单（和所有由它生成的项目），使得对它的修改是被要求的，直到下次onCreateOptionsMenu()被调用。

当你增加菜单项目，你可以执行activity的onOptionsItemSelected(MenuItem)方法来句柄到那。

Parameters

menu 你放置你的项目的的项目菜单。

Returns

你必须为菜单返回true来显示；如果你返回false，它将被关闭。

See Also

onPrepareOptionsMenu(Menu)

onOptionsItemSelected(MenuItem)

**android.app.Activity.onOptionsItemSelected方法**

public boolean onOptionsItemSelected (MenuItem item)

当你的菜单中的一个项目被选择时，这个被调用。默认实现简单地返回false来有一个正常的处理发生（适当调用项目的Runnable或者发送一个消息到它的Handler）。你能使用这个方法处理你想要的任何项目，而不使用其他工具。

派生类应该通过基类调用它，来完成默认菜单处理。

Parameters

item 被选择的菜单项目。

Returns

返回false来允许正常菜单处理运行，true来在这里销毁它。

See Also

onCreateOptionsMenu(Menu)

**android.app.Activity. onPostCreate方法**

protected void onPostCreate (Bundle savedInstanceState)

当activity启动完成时调用 (after onStart() and onRestoreInstanceState(Bundle) have been called). Applications will generally not implement this method; it is intended for system classes to do final initialization after application code has run.

Derived classes must call through to the super class's implementation of this method. If they do not, an exception will be thrown.

##### Parameters

##### savedInstanceState 如果activity在早先关闭后被重新初始化，此Bundle包含最近onSaveIntanceState(Bundle)提供的数据。Note:否则为null。

##### See Also

onCreate(Bundle)

**android.app.Activity. onPostCreate方法**

此方法与onPostCreate(Bundle)相同，但它被调用用来创建persistableMode设置成persistAcrossReboots的activity。

##### Parameters

##### savedInstaceState 最近由onSaveInstaceState(Bundle)提供的数据。

##### PersistentState 此数据来自onSaveInstanceState(Bundle,PersistableBundle)第一次保存的数据。

##### See Also

onCreate(Bundle)

**android.app.Activity. onPrepareOptionsMenu方法**

public boolean onPrepareOptionsMenu (Menu menu)

准备屏幕显示的标准菜单设置。This is called right before the menu is shown, every time it is shown. You can use this method to efficiently enable/disable items or otherwise dynamically modify the contents.

The default implementation updates the system menu items based on the activity's state. Deriving classes should always call through to the base class implementation.

##### Parameters

##### menu 上次显示的或通过onCreateOptionsMenu()创建的操作菜单

##### Returns

You must return true for the menu to be displayed; if you return false it will not be shown.

##### See Also

onCreateOptionsMenu(Menu)

**android.app.Activity. onResume方法**

protected void onResume ()

在nRestoreInstanceState(Bundle),onRestart,或onPause()之后调用，为你的activity来开始与用户交互。这是一个好的地方来开始动作，打开互斥读写设备（比如说相机）。

记住onResume不是一个最好你的activity显示给用户的指示器；一个系统窗口，比如键盘可能在上面。使用onWindowFocusChanged(boolean)来知道确定你的activity显示给用户(例如，开始一个游戏)。

派生类必须通过父类的实现方法调用此函数。否则将抛出异常。

See Also

onRestoreInstanceState(Bundle)

onRestart()

onPostResume()

onPause()

**android.app.Activity. onStart方法**

protected void onStart ()

在onCreate(Bundle)之后调用----或者当activity被停止，但它现在又一次被用户显示，在onRestart()后调用。它后面将跟着onResume()。

派生类必须通过父类的实现方法调用此函数。否则将抛出异常。

See Also

onCreate(Bundle)

onStop()

onResume()

**android.app.Activity. onStop方法**

protected void onStop ()

当你不再显示给用户时调用。你将下一步调用onRestart(),onDestroy()，或什么也不作，取决于用户的动作。

记住这个方法可能永远不会被调用，在低内存情况下，系统没有足够内存来保持你的activity的处理运行,在它的onPause()方法被调用之后。

派生类必须通过父类的实现方法调用此函数。否则将抛出异常。

See Also

onRestart()

onResume()

onSaveInstanceState(Bundle)

onDestroy()

**android.app.Activity.setResult方法**

public final void setResult (int resultCode, Intent data)

截至GINGERBREAD,你在这提供的Intent可以包含Intent.FLAG\_GRANT\_READ\_URI\_PERMISSION或者Intent.FLAG\_GRANT\_WRITE\_URI\_PERMISSION 设置。这将允许Activity接收Intent中特定URI通向的结果。链接将会保留直到Activity结束（它将保留到虚拟机程序关闭或者暂时的破坏），并且将被添加到任何存在的它已经保持的URI允许中。

Parameters

resultCode 传送给源activity的结果代码，经常是RESULT\_CANCELED 或 RESULT\_OK。

data 传给源activity的数据。

See Also

RESULT\_CANCELED

RESULT\_OK

RESULT\_FIRST\_USER

setResult(int)

**android.app.Activity.startActivity方法**

public void startActivity (Intent intent)

与没有特定设定的startActivity(Intent，Bundle)相同。

Parameters

intent 要开始的Intent

Throws

android.content.ActivityNotFoundException

See Also

ERROR(#startActivity(Intent, Bundle)}/{@link #startActivity(Intent, Bundle)})

startActivityForResult(Intent, int)

**android.app.Activity.startActivityForResult方法**

public void startActivityForResult (Intent intent, int requestCode)

与没有选项的startActivityForResult(Intent,int ,Bundle)调用相同。

Parameters

intent 开始的intent。

requestCode 如果>=0,这个代码将在activity退出时在onActivityResult()返回。

Throws

android.content.ActivityNotFoundException

See Also

startActivity(Intent)

**android.app.AlertDialog.Builder类**

extends Object

**android.app.AlertDialog.Builder.AlertDialog.Builder构造方法**

public AlertDialog.Builder (Context context)

这个builder和它创建的AlertDialog的使用一个context的构造方法。

**android.app.AlertDialog.Builder.setPositiveButton方法**

public AlertDialog.Builder setPositiveButton (CharSequence text, DialogInterface.OnClickListener listener)

设置确定按钮按下的监听器。

##### Parameters

##### text 确定按钮上应显示的文字。

##### Listener 应该使用DialogInterface.OnClickListener。

##### Returns

This Builder object to allow for chaining of calls to set methods

**android.app.AlertDialog.Builder.show方法**

public AlertDialog show ()

创建以这个builder提供的参数为支持的AlertDialog并显示它。

**android.app.Application类**

extends ContextWrapper

implements ComponentCallbacks2

我们需要保存的全局应用数据的基类。你可以提供你自己的实现通过在你的AndroidManifest.xml的<application>标签中定义name，它将使当运行你的application/package时，实例化的类被创建

**android.app.Application.onConfigurationChanged方法**

**android.app.Application.onCreate方法**

public void onCreate ()

当应用开始时调用，早于任何activity,service或者receiver对象（包括content提供者）被创建。Implementations should be as quick as possible (for example using lazy initialization of state) since the time spent in this function directly impacts the performance of starting the first activity, service, or receiver in a process. If you override this method, be sure to call super.onCreate().

**android.app.Application.onTerminate方法**

**android.app.Application.onTrimMemory方法**

**android.app.Application.onLowMamory方法**

它在系统低内存时调用，并且运行的程序应该削减内存使用。While the exact point at which this will be called is not defined, generally it will happen when all background process have been killed. That is, before reaching the point of killing processes hosting service and foreground UI that we would like to avoid killing.

You should implement this method to release any caches or other unnecessary resources you may be holding on to. The system will perform a garbage collection for you after returning from this method.

Preferably, you should implement onTrimMemory(int) from ComponentCallbacks2 to incrementally unload your resources based on various levels of memory demands. That API is available for API level 14 and higher, so you should only use this onLowMemory() method as a fallback for older versions, which can be treated the same as onTrimMemory(int) with the TRIM\_MEMORY\_COMPLETE level.

**android.app.DatePickerDialog类**

extends AlertDialog  
implements DialogInterface.OnClickListener DatePicker.OnDateChangedListener

一个简单的包含DatePicker的对话框。

**android.app.DatePickerDialog.DatePickerDialog构造方法**

#### public DatePickerDialog (Context context, DatePickerDialog.OnDateSetListener callBack, int year, int monthOfYear, int dayOfMonth)

##### Parameters

##### context 对话框运行在的context。

##### callBack 日期设定时怎样执行。

##### year 对话框中初始年份。

##### monthOfyear 对话框中初始月份。

##### dayOfMonth 对话框中初始日期。

**android.app.Service类**

**android.app.Service.onBind方法**

public abstract IBinder onBind (Intent intent)

返回与service的联系渠道。可能返回null,如果客户没有service的联系。返回的IBinder通常是使用aidl描述的复杂接口。

Note that unlike other application components, calls on to the IBinder interface returned here may not happen on the main thread of the process. More information about the main thread can be found in Processes and Threads.

Parameters

intent 用于绑定接口的Intent，通过Context.bindService传递。Note that any extras that were included with the Intent at that point will not be seen here.

Returns

Return an IBinder through which clients can call on to the service.

**android.app.Service.onCreate方法**

public void onCreate ()

服务第一次额被创建时系统调用。不要直接调用这个方法。

**android.app.Service.onStartCommand方法**

public int onStartCommand (Intent intent, int flags, int startId)

由系统在每次客户明确地通过调用startService(Intent)时调用，提供它传递的参数和一个代表开始请求的独特整型。不要直接调用它。

For backwards compatibility, the default implementation calls onStart(Intent, int) and returns either START\_STICKY or START\_STICKY\_COMPATIBILITY.

If you need your application to run on platform versions prior to API level 5, you can use the following model to handle the older onStart(Intent, int) callback in that case. The handleCommand method is implemented by you as appropriate:

Note that the system calls this on your service's main thread. A service's main thread is the same thread where UI operations take place for Activities running in the same process. You should always avoid stalling the main thread's event loop. When doing long-running operations, network calls, or heavy disk I/O, you should kick off a new thread, or use AsyncTask.

Parameters

intent startService(Intent)传递的Intent。它可能是空，如果服务在它的过程死去之后被重新启动，并且它早先返回任何东西除了START\_STICKY\_COMPATIBILITY。

flags 附加的请求数据。可能是0, START\_FLAG\_REDELIVERY, 或START\_FLAG\_RETRY。

startId 一个独特的整型数，代表开始请求。与stopSelfResult(int)结合使用。

Returns

The return value indicates what semantics the system should use for the service's current started state. It may be one of the constants associated with the START\_CONTINUATION\_MASK bits.

See Also

stopSelfResult(int)

**android.app.Service.onDestroy方法**

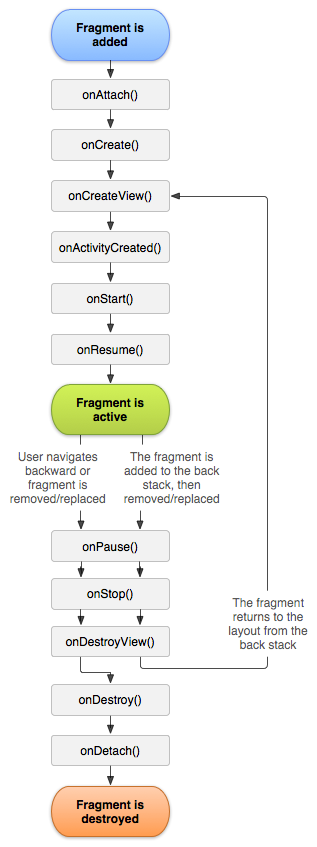
public void onDestroy ()

当告知service不再使用并且被移除时系统调用。 service应该在此处清空任何它保持的资源（threads,registered receivers,etc）。 Upon return, there will be no more calls in to this Service object and it is effectively dead. Do not call this method directly.

**android.app.Fragment类**

extends Object  
implements ComponentCallbacks2 View.OnCreateContextMenuListener

一个Fragment是一个应用的用户接口或能放在一个Activity中表现的一部分。通过FragmentManager与fragment相互作用，它能通过Activity.getFragmentManager()方法和Fragment.getFragmentManager()方法获取。



**android.app.Fragment.getActivity方法**

public final Activity getActivity ()

返回与这个fragment当前联系的Activity。

**android.app.Fragment.getFragmentManager方法**

public final FragmentManager getFragmentManager ()

返回与这个fragment联系的activity相互作用的FragmentManager。Note that this will be non-null slightly beforegetActivity(), during the time from when the fragment is placed in a FragmentTransaction until it is committed and attached to its activity.

If this Fragment is a child of another Fragment, the FragmentManager returned here will be the parent's getChildFragmentManager().

**android.app.Fragment.onCreate方法**

#### public void onCreate (Bundle savedInstanceState)

调用来初始化创建一个fragment。This is called after onAttach(Activity) and before onCreateView(LayoutInflater, ViewGroup, Bundle).

Note that this can be called while the fragment's activity is still in the process of being created. As such, you can not rely on things like the activity's content view hierarchy being initialized at this point. If you want to do work once the activity itself is created, see onActivityCreated(Bundle).

##### Parameters

##### savedInstanceState 如果fragment被从先前保存的状态重新创建，这就是那个状态。

**android.app.Fragment.onCreateView方法**

#### public View onCreateView (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)

调用来使fragment实例化它的用户接口view。This is optional, and non-graphical fragments can return null (which is the default implementation). This will be called between onCreate(Bundle) and onActivityCreated(Bundle).

If you return a View from here, you will later be called in onDestroyView() when the view is being released.

##### Parameters

##### inflater LayoutInfater对象，能被用于inflate任何view在fragment中。

##### Container 如果非空，这是fragment的UI应该被附加到的父view。fragment不应该自己添加view，但这能被用来产生view的LayoutParam。

##### SavedInstanceState 如果非空，这个fragment正在被从先前保存的状态重新创建，这就是这个状态。

##### Returns

返回fragment的UI View，或者为null。

**android.app.Fragment.onDestroy方法**

#### public void onDestroy ()

当fragment不再使用时调用。This is called after onStop() and before onDetach().

**android.app.FragmentManager类**

extends Object

与Activity内部的Fragment对象相互作用的接口。

**android.app.FragmentManager.beginTransaction方法**

public abstract FragmentTransaction beginTransaction ()

开始与这个FragmentManager联系的Fragment的一系列编辑操作。

Note: A fragment transaction can only be created/committed prior to an activity saving its state. If you try to commit a transaction afterActivity.onSaveInstanceState() (and prior to a following Activity.onStart or Activity.onResume(), you will get an error. This is because the framework takes care of saving your current fragments in the state, and if changes are made after the state is saved then they will be lost.

**android.app.FragmentManager.popBackStack方法**

#### public abstract void popBackStack ()

从栈顶出栈状态。这个方法是异步的—它入队请求来出栈，但是这行为将不执行知道应用返回事件循环。

**android.app.FragmentTransation类**

extends Object

对Fragment操作设置的API。

**android.app.FragmentTransation.add方法**

#### public abstract FragmentTransaction add (int containerViewId, Fragment fragment, String tag)

添加一个fragment到fctivity状态。这个fragment可能可选地也在activity的container view中有它的view（如果Fragment.onCreatView返回非null）。

##### Parameters

##### containerViewId 可选的fragment被放置的container的标识符。如果为0，它将不被放置在任何container中。

##### fragment 被增加的fragment。这个fragment必须没有已经被添加到activity中。

##### tag 可选的fragment标志名，来以后用FragmentManager.findFragmentByTag(String)重新检索fragment。

##### Returns

返回同样的FragmentTransation实例。

**android.app.FragmentTransation.addToBackStack方法**

#### public abstract FragmentTransaction addToBackStack (String name)

添加这个transaction到返回栈。This means that the transaction will be remembered after it is committed, and will reverse its operation when later popped off the stack.

##### Parameters

##### name 一个可选的返回栈状态名，或者为null。

**android.app.FragmentTransation.commit方法**

public abstract int commit ()

安排translation的一个承诺（提交）。这个承诺不立即发生；它将在主线程被完成的下一步工作。

A transaction can only be committed with this method prior to its containing activity saving its state. If the commit is attempted after that point, an exception will be thrown. This is because the state after the commit can be lost if the activity needs to be restored from its state. SeecommitAllowingStateLoss() for situations where it may be okay to lose the commit.

##### Returns

如果addToBackStack(String)被调用，返回这个转换的返回栈入口标识符。否则返回一个无用数字。

**android.app.FragmentTransation.replace方法**

#### public abstract FragmentTransaction replace (int containerViewId, Fragment fragment, String tag)

替换一个存在的已经被添加到container中的fragment。This is essentially the same as calling remove(Fragment) for all currently added fragments that were added with the same containerViewId and then add(int, Fragment, String) with the same arguments given here.

##### Parameters

##### containerViewID 被替换的fragment的container的标识符。

##### fragment 新的替换到container的fragment。

##### tag 可选的fragment标志名，来以后用FragmentManager.findFragmentByTag(String)重新检索fragment。

##### Returns

返回相同的FragmentTransaction实例。

**android.app.TimePickerDialog类**

**android.app.TimePickerDialog.TimePickerDialog构造方法**

public TimePickerDialog (Context context, TimePickerDialog.OnTimeSetListener callBack, int hourOfDay, int minute, boolean is24HourView)

##### Parameters

##### context 父亲。

##### callBack 怎样运行。

##### hourOfDay 初始小时。

##### Minute 初始分钟。

##### Is24HourVIew 是否时24小时视图，或者是AM/PM。

**android.content.BroadcastReceiver类**

extends AlertDialog  
implements DialogInterface.OnClickListener TimePicker.OnTimeChangedListener

**android.content.BroadcastTeceiver.abortBroadcast方法**

public final void abortBroadcast ()

设置一个标志来表明这个receiver应该停止当前broadcast；只有当broadcast通过Context.sendOrderdeBroadcast发送时有效。它将阻止其他broadcastreceiver收到此广播。It will still call onReceive(Context, Intent) of the BroadcastReceiver that the caller of Context.sendOrderedBroadcast passed in.

此方法对于未排序的broadcast无效，例如通过Context.sendBroadcast发送的广播。

**android.content.BroadcastTeceiver.onReceive方法**

public abstract void onReceive (Context context,Intent intent)

当BroadcastReceiver接收一个Intent广播时调用这个方法。During this time you can use the other methods on BroadcastReceiver to view/modify the current result values. This method is always called within the main thread of its process, unless you explicitly asked for it to be scheduled on a different thread using registerReceiver(BroadcastReceiver, IntentFilter, String, android.os.Handler). When it runs on the main thread you should never perform long-running operations in it (there is a timeout of 10 seconds that the system allows before considering the receiver to be blocked and a candidate to be killed). You cannot launch a popup dialog in your implementation of onReceive().

If this BroadcastReceiver was launched through a <receiver> tag, then the object is no longer alive after returning from this function. This means you should not perform any operations that return a result to you asynchronously -- in particular, for interacting with services, you should usestartService(Intent) instead of bindService(Intent, ServiceConnection, int). If you wish to interact with a service that is already running, you can usepeekService(Context, Intent).

The Intent filters used in registerReceiver(BroadcastReceiver, IntentFilter) and in application manifests are not guaranteed to be exclusive. They are hints to the operating system about how to find suitable recipients. It is possible for senders to force delivery to specific recipients, bypassing filter resolution. For this reason, onReceive() implementations should respond only to known actions, ignoring any unexpected Intents that they may receive.

##### Parameters

##### context receiver运行的Context。

##### Intent 接收的intent。

**android.content.ComponentName类**

extends Object  
implements Parcelable Cloneable Comparable<ComponentName>

**android.content.ComponentName.ComponentName构造方法**

#### public ComponentName (String pkg, String cls)

创建一个新的component标识符。

##### Parameters

|  |  |
| --- | --- |
| pkg | The name of the package that the component exists in. Can not be null. |
| cls | The name of the class inside of pkg that implements the component. Can not be null. |

**android.content.ContentProvider类**

extends Object  
implements ComponentCallbacks2

ContentProvider是android应用优先创建块之一，给应用提供content。

**android.content.ContentProvider.onCreate方法**

public abstract boolean onCreate ()

执行此方法来在启动时初始化contentprovider。在应用主线程启动时调用所有已注册的contentprovider的此方法。It must not perform lengthy operations, or application startup will be delayed.

You should defer nontrivial initialization (such as opening, upgrading, and scanning databases) until the content provider is used (via query(Uri, String[], String, String[], String), insert(Uri, ContentValues), etc). Deferred initialization keeps application startup fast, avoids unnecessary work if the provider turns out not to be needed, and stops database errors (such as a full disk) from halting application launch.

If you use SQLite, SQLiteOpenHelper is a helpful utility class that makes it easy to manage databases, and will automatically defer opening until first use. If you do use SQLiteOpenHelper, make sure to avoid calling getReadableDatabase() or getWritableDatabase() from this method. (Instead, overrideonOpen(SQLiteDatabase) to initialize the database when it is first opened.)

##### Returns

true if the provider was successfully loaded, false otherwise

**android.content.ContentProvider.insert方法**

public abstract Uri insert (Uri uri, ContentValues values)

执行此方法来执行插入新行命令。As a courtesy, call notifyChange() after inserting. This method can be called from multiple threads, as described in Processes and Threads.

##### Parameters

##### uri 要插入的content://URI。不能为null。

##### Values 添加到数据库的名值对。不能为null。

##### Returns

The URI for the newly inserted item.

**android.content.ContentProvider.query方法**

public Cursor query (Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder, CancellationSignalcancellationSignal)

执行此方法来提交支持取消的客户端查询命令。This method can be called from multiple threads, as described inProcesses and Threads.

If you implement this method then you must also implement the version of query(Uri, String[], String, String[], String) that does not take a cancellation signal to ensure correct operation on older versions of the Android Framework in which the cancellation signal overload was not available.

Parameters

uri 要查询的URI。This will be the full URI sent by the client; if the client is requesting a specific record, the URI will end in a record number that the implementation should parse and add to a WHERE or HAVING clause, specifying that \_id value.

Projection A selection criteria to apply when filtering rows. If null then all rows are included.

##### Selection A selection criteria to apply when filtering rows. If null then all rows are included.

##### SelectionArgs You may include ?s in selection, which will be replaced by the values from selectionArgs, in order that they appear in the selection. The values will be bound as Strings.

##### SortOrder How the rows in the cursor should be sorted. If null then the provider is free to define the sort order.

##### CancellationSignal A signal to cancel the operation in progress, or null if none. If the operation is canceled, then OperationCanceledException will be thrown when the query is executed.

##### Returns

a Cursor or null.

**android.content.ContentResolver类**

extends Object

这个类提供应用到content模型的连接。

**android.content.ContentResolver.insert方法**

public final Uri insert (Uri url, ContentValues values)

在给的URL的表中插入一行。If the content provider supports transactions the insertion will be atomic.

##### Parameters

##### url 要插入的表的url。

##### Values 要插入的新行的格式化的值。键是列名。传入空ContentValues将创建空行。

##### Returns

the URL of the newly created row.

**android.content.ContentValues类**

extends Object  
implements Parcelable

这个类被用来存储一组ContentResolver能处理的值。

**android.content.ContextWrapper类**

extends Context

**android.content.ContextWrapper.bindService方法**

public boolean bindService (Intent service, ServiceConnection conn, int flags)

绑定到一个应用service，如果需要就创建它。 这定义一个依赖在你的应用和service之间。给的conn将接受service对象，当它创建并且被告知它是否死亡或restart。 service将认为只被系统要求，只要context的调用存在。例如，如果这个Context是一个停止的Activity,服务将不被要求继续运行，直到Activity被重新开始。

This function will throw SecurityException if you do not have permission to bind to the given service.

Note: this method can not be called from a BroadcastReceiver component. A pattern you can use to communicate from a BroadcastReceiver to a Service is to call startService(Intent) with the arguments containing the command to be sent, with the service calling its stopSelf(int) method when done executing that command. See the API demo App/Service/Service Start Arguments Controller for an illustration of this. It is okay, however, to use this method from a BroadcastReceiver that has been registered with registerReceiver(BroadcastReceiver, IntentFilter), since the lifetime of this BroadcastReceiver is tied to another object (the one that registered it).

Parameters

service 服务连接的标识符。The Intent may specify either an explicit component name, or a logical description (action, category, etc) to match an IntentFilter published by a service.

conn (wpf:用于与service连接的ServiceConnection对象)接收service开始和停止信息(wpf:监听)。它必须是一个有效的ServiceConnection对象；它必须不为null。

flags Operation options for the binding. May be 0, BIND\_AUTO\_CREATE, BIND\_DEBUG\_UNBIND, BIND\_NOT\_FOREGROUND, BIND\_ABOVE\_CLIENT, BIND\_ALLOW\_OOM\_MANAGEMENT, or BIND\_WAIVE\_PRIORITY.

Returns

If you have successfully bound to the service, true is returned; false is returned if the connection is not made so you will not receive the service object.

**android.content.ContextWrapper.checkCallingOrSelfPermission方法**

public abstract int checkCallingOrSelfPermission (String permission)

检查IPC调用过程或你拥有的是否被授予权限。This is the same as checkCallingPermission(String), except it grants your own permissions if you are not currently processing an IPC. Use with care!

##### Parameters

##### permission 被检查的权限名。

##### Returns

如果调用的pid/uid被授予权限，返回PERMISSION\_GRANTED，否则返回PERMISSION\_DENIED。

##### See Also

checkPermission(String, String)

checkPermission(String, int, int)

checkCallingPermission(String)

**android.content.ContextWrapper.getApplicationContext方法**

**public Context getApplicationContext ()**

得到现在程序的唯一的、全局的Application对象context。这个应该只用于你需要一个声明周期与现在context分开的Context，它的生命与程序而不是组件绑定。

Consider for example how this interacts with registerReceiver(BroadcastReceiver, IntentFilter):

If used from an Activity context, the receiver is being registered within that activity. This means that you are expected to unregister before the activity is done being destroyed; in fact if you do not do so, the framework will clean up your leaked registration as it removes the activity and log an error. Thus, if you use the Activity context to register a receiver that is static (global to the process, not associated with an Activity instance) then that registration will be removed on you at whatever point the activity you used is destroyed.

If used from the Context returned here, the receiver is being registered with the global state associated with your application. Thus it will never be unregistered for you. This is necessary if the receiver is associated with static data, not a particular component. However using the ApplicationContext elsewhere can easily lead to serious leaks if you forget to unregister, unbind, etc.

**android.content.ContextWrapper.getCacheFilesDir方法**

#### public File getExternalCacheDir ()

返回早先外部文件系统的随机目录路径（Environment.getExternalStorageDirectory()的某处），在此处应用可放置其拥有的缓存文件。These files are internal to the applications, and not typically visible to the user as media.

详细信息请查看原文档。

##### Returns

The path of the directory holding application cache files on external storage. Returns null if external storage is not currently mounted so it could not ensure the path exists; you will need to call this method again when it is available.

**android.content.ContextWrapper.getContentReslover方法**

public ContentResolver getContentResolver ()

返回应用包的ContentResolver实例。

**android.content.ContextWrapper.openFileInput方法**

public FileInputStream openFileInput (String name)

打开一个与要读的Context的应用包联系的私有文件。

##### Parameters

##### name 打开的文件名；不能包含层级路径。

##### Returns

The resulting FileInputStream.

##### Throws

|  |  |
| --- | --- |
| FileNotFoundException |  |

**android.content.ContextWrapper.getExternalFilesDir方法**

public File getExternalFilesDir (String type)

返回早先外部文件系统的随机目录路径（Environment.getExternalStorageDirectory()的某处），在此处应用可放置其拥有的持久文件。These files are internal to the applications, and not typically visible to the user as media.

详细信息请查看原文档。

##### Parameters

##### type 返回的文件目录类型。null为根目录，或者是下列Environment包含的子目录值中的一个：DIRECTORY\_MUSIC,DIRECTORY\_PODCASTS,DIRECTORY\_RINGTONES, DIRECTORY\_ALARMS, DIRECTORY\_NOTIFICATIONS,DIRECTORY\_PICTURES, DIRECTORY\_MOVIES.

##### Returns

The path of the directory holding application files on external storage. Returns null if external storage is not currently mounted so it could not ensure the path exists; you will need to call this method again when it is available.

**android.content.ContextWrapper.openFileOutput方法**

public FileOutputStream openFileOutput (String name, int mode)

打开一个与写入的Context的应用包联系的私有文件。如果不存在创建它。

No permissions are required to invoke this method, since it uses internal storage.

##### Parameters

##### name 打开的文件名；不能包含层级路径。

##### Mode 打开模式。使用0或MODE\_PRIVATE进行默认操作；MODE\_APPEND在已存在的文件后添加；MODE\_WORLD\_READABLE和MODE\_WORLD\_WRITEABLE控制权限。

##### Returns

The resulting FileOutputStream.

##### Throws

*FileNotFoundException*

**android.content.ContextWrapper.registerReceiver方法**

public Intent registerReceiver (BroadcastReceiver receiver, IntentFilter filter)

注册一个运行在main activity线程中的BroadcastReceiver。The receiver will be called with any broadcast Intent that matches filter, in the main application thread.

The system may broadcast Intents that are "sticky" -- these stay around after the broadcast as finished, to be sent to any later registrations. If your IntentFilter matches one of these sticky Intents, that Intent will be returned by this function and sent to your receiver as if it had just been broadcast.

There may be multiple sticky Intents that match filter, in which case each of these will be sent to receiver. In this case, only one of these can be returned directly by the function; which of these that is returned is arbitrarily decided by the system.

If you know the Intent your are registering for is sticky, you can supply null for your receiver. In this case, no receiver is registered -- the function simply returns the sticky Intent that matches filter. In the case of multiple matches, the same rules as described above apply.

See BroadcastReceiver for more information on Intent broadcasts.

As of ICE\_CREAM\_SANDWICH, receivers registered with this method will correctly respect the setPackage(String) specified for an Intent being broadcast. Prior to that, it would be ignored and delivered to all matching registered receivers. Be careful if using this for security.

Note: this method cannot be called from a BroadcastReceiver component; that is, from a BroadcastReceiver that is declared in an application's manifest. It is okay, however, to call this method from another BroadcastReceiver that has itself been registered at run time withregisterReceiver(BroadcastReceiver, IntentFilter), since the lifetime of such a registered BroadcastReceiver is tied to the object that registered it.

##### Parameters

##### receiver 句柄到广播的BroadcastReceiver。

##### Filter 选择接收的Intent广播。

##### Returns

The first sticky intent found that matches filter, or null if there are none.

**android.content.ContextWrapper.sendBroadcast方法**

public void sendBroadcast (Intent intent)

广播Intent到所有感兴趣的BroadcastReceivers中。This call is asynchronous; it returns immediately, and you will continue executing while the receivers are run. No results are propagated from receivers and receivers can not abort the broadcast. If you want to allow receivers to propagate results or abort the broadcast, you must send an ordered broadcast using sendOrderedBroadcast(Intent, String).

See BroadcastReceiver for more information on Intent broadcasts.

##### Parameters

##### Intent 要广播的Intent；所有与这个Intent匹配的receiver将会接收到这个广播。

**android.content.ContextWrapper.sendOrderedBroadcast方法**

public void sendOrderedBroadcast (Intent intent, String receiverPermission)

广播给的intent到所有相关联的BroadcastReceiver，一次给他们提供一个以允许更多的receiver来消费broadcost，在它提供给更少的receiver之前。This call is asynchronous; it returns immediately, and you will continue executing while the receivers are run.

See BroadcastReceiver for more information on Intent broadcasts.

##### Parameters

##### intent 广播Intent;所有匹配此intent的receiver将接收到此广播。

##### ReceiverPermission （可选）用String命名一个receiver必须持有来接收广播的许可。如果是null，没有许可要求。

**android.content.ContextWrapper.startService方法**

public ComponentName startService (Intent service)

请求开始一个给的应用service。The Intent should contain either contain the complete class name of a specific service implementation to start or a specific package name to target. If the Intent is less specified, it log a warning about this and which of the multiple matching services it finds and uses will be undefined. If this service is not already running, it will be instantiated and started (creating a process for it if needed); if it is running then it remains running.

Every call to this method will result in a corresponding call to the target service's onStartCommand(Intent, int, int) method, with the intent given here. This provides a convenient way to submit jobs to a service without having to bind and call on to its interface.

Using startService() overrides the default service lifetime that is managed by bindService(Intent, ServiceConnection, int): it requires the service to remain running until stopService(Intent) is called, regardless of whether any clients are connected to it. Note that calls to startService() are not nesting: no matter how many times you call startService(), a single call to stopService(Intent) will stop it.

The system attempts to keep running services around as much as possible. The only time they should be stopped is if the current foreground application is using so many resources that the service needs to be killed. If any errors happen in the service's process, it will automatically be restarted.

This function will throw SecurityException if you do not have permission to start the given service.

Parameters

service Identifies the service to be started. The Intent must be either fully explicit (supplying a component name) or specify a specific package name it is targetted to. Additional values may be included in the Intent extras to supply arguments along with this specific start call.

Returns

If the service is being started or is already running, the ComponentName of the actual service that was started is returned; else if the service does not exist null is returned.

**android.content.ContextWrapper.stopService方法**

public boolean stopService (Intent name)

请求给的应用service停止。 If the service is not running, nothing happens. Otherwise it is stopped. Note that calls to startService() are not counted -- this stops the service no matter how many times it was started.

Note that if a stopped service still has ServiceConnection objects bound to it with the BIND\_AUTO\_CREATE set, it will not be destroyed until all of these bindings are removed. See the Service documentation for more details on a service's lifecycle.

This function will throw SecurityException if you do not have permission to stop the given service.

Parameters

name Description of the service to be stopped. The Intent must be either fully explicit (supplying a component name) or specify a specific package name it is targetted to.

Returns

If there is a service matching the given Intent that is already running, then it is stopped and true is returned; else false is returned.

**android.content.ContextWrapper.unbindService方法**

public void unbindService (ServiceConnection conn)

解除应用service联系。当service重新开始时你将不再接收调用，并且service将允许在任何时间停止。

Parameters

conn 先前通过bindService()传递的连接接口。这个参数必须不为null。

**android.content.ContextWrapper.unbindService方法**

#### public void unregisterReceiver (BroadcastReceiver receiver)

注销一个先前注册的BroadcastReceiver。All filters that have been registered for this BroadcastReceiver will be removed.

##### Parameters

##### receiver 要注销的BroadcastReceiver。

**android.content.Intent类**

**android.content.Intent.Intent构造方法**

public Intent (Context packageContext, Class<?> cls)

为特别的组件创建一个intent。所有其他的领域(action,data,type,class)为null，即使他们能被后面明确的调用修改。它提供了一个便利的方法来创建一个意图执行硬编码的类名的intent，而不是依靠系统来为你找到一个适当的类；查看setComponent(ComponentName)来得到更多它的反响信息。

Parameters

packageContext 一个执行这个类的应用的Content。aapt

cls 一个用于intent的组件类aapt

See Also

setClass(Context, Class)

setComponent(ComponentName)

Intent(String, android.net.Uri, Context, Class)

aapt

**android.content.Intent.Intent构造方法**

public Intent (String action, Uri uri)

用给的action和用于获取数据的URI创建intent。记住action必须是在一个命名空间里，因为intents在系统里被全局使用—例如系统的VIEW action是android.intent.action. VIEW；一个应用的定制的action应当像com.google.app.myapp.CUSTOM\_ACTION。

Note:计划和主机名对应在android框架中是大小写敏感的，不像RFC格式。 As a result, you should always ensure that you write your Uri with these elements using lower case letters, and normalize any Uris you receive from outside of Android to ensure the scheme and host is lower case.因此，以应当一致确保你写的使用小写字母元素的Uri，并且规格化任何你从androi以外接收的uri，确保计划和主机是小写。

Parameters

action Intent内定义的action，例如ACTION\_VIEW。

uri Intent数据URI

**android.content.Intent.getStringExtra方法**

public String getStringExtra (String name)

从intent接收额外数据。

Parameters

name 需求的项目名字。

Returns

先前用putExtra()添加的项目值，如果未发现String值将为空。

See Also

putExtra(String, String)

**android.content.Intent.getStringExtras方法**

public Bundle getExtras ()

从intent恢复一个额外数据映射。

Returns

the map of all extras previously added with putExtra(), or null if none have been added.

**android.content.Intent.setAction方法**

public Intent setAction (String action)

设置启动的action。

##### Parameters

##### action 一个action名，例如ACTION\_VIEW。应用特定action应该包含包名。（wpf:用于显示data的默认action：android.intent.action.VIEW）

##### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

##### See Also

getAction()

**android.content.Intent.setComponent方法**

public Intent setComponent (ComponentName component)

（通常是可选的）明确地设置intent句柄到的组件。If left with the default value of null, the system will determine the appropriate class to use based on the other fields (action, data, type, categories) in the Intent. If this class is defined, the specified class will always be used regardless of the other fields. You should only set this value when you know you absolutely want a specific class to be used; otherwise it is better to let the system find the appropriate class so that you will respect the installed applications and user preferences.

Parameters

component intent句柄到的组件名，或者是null来让系统为你找到一个。

Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

See Also

setClass(Context, Class)

setClassName(Context, String)

setClassName(String, String)

getComponent()

resolveActivity(PackageManager)

**android.content.Intent.setData方法**

public Intent setData (Uri data)

设置这个intent操作的data。This method automatically clears any type that was previously set by setType(String) or setTypeAndNormalize(String).

Note: scheme matching in the Android framework is case-sensitive, unlike the formal RFC. As a result, you should always write your Uri with a lower case scheme, or use normalizeScheme() or setDataAndNormalize(Uri) to ensure that the scheme is converted to lower case.

##### Parameters

##### data 这个intent目标的数据Uri。

##### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

##### See Also

getData()

setDataAndNormalize(Uri)

normalizeScheme()

**android.content.Intent.putExtre方法**

public Intent putExtra (String name, String value)

增加额外数据到intent。The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".名字必须包括包前缀，例如appcom.android.contacts应该使用名字像”com.android.contacts.ShowAll”。

Parameters

name 带有包前缀的额外数据名。

value 字符串数据。

Returns

Returns the same Intent object, for chaining multiple calls into a single statement.返回一个相同的Intent对象，用于在单一声明中链接混合调用。

See Also

putExtras(Intent)

removeExtra(String)

getStringExtra(String)

**android.content.Intent.putExtres方法**

public Intent putExtras (Bundle extras)

为intent增加一个额外的数据设置。 The keys must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

Parameters

extras The Bundle of extras to add to this intent.Returns

See Also

putExtra(String, Bundle)

removeExtra(String)

**android.content.IntentFilter类**

extends Object  
implements Parcelable

描述匹配的Intent值的结构。

**android.content.IntentFilter.IntentFilter构造方法**

#### public IntentFilter (String action)

新的IntentFilter过滤器匹配一个不带有数据的单一的action。If no data characteristics are subsequently specified, then the filter will only match intents that contain no data.

##### Parameters

|  |  |
| --- | --- |
| *action* | The action to match, i.e. Intent.ACTION\_MAIN. |

**android.content.ServiceConnection接口**

(wpf:实现本接口用于跟service绑定)

android.content.ServiceConnection.onServiceConnected方法

public abstract void onServiceConnected (ComponentName name, IBinder service)

当到一个Service的连接建立时调用，包含与communication交流的渠道IBinder。

Parameters

name 被连接的具体组件名。

service service的交流渠道IBinder,你现在可以调用打开。

**android.content.ServiceConnection.onServiceDisconnected方法**

public abstract void onServiceDisconnected (ComponentName name)

当一个到Service的连接丢失时调用。代表是当程序主机service被清除或杀死。这不移除ServiceConnection本身—这与service的绑定将继续进行，并且当service下一次运行，你将接收一个调用到onServiceConnected(ComponentName,IBinder)。

**android.content.res.AssetManager类**

extends Object  
implements AutoCloseable

提供了通向应用原始asset文件的途径；查看Resources类中大多数应用想要检索资源数据的方式。

**android.content.res.AssetManager.open方法**

public final InputStream open (String fileName)

使用ACCESS\_STREAMING模式打开一个asset。 This provides access to files that have been bundled with an application as assets -- that is, files placed in to the "assets" directory.

##### Parameters

##### fileName 打开的asset文件名。这个名字可以分层。

##### Throws

IOException

##### See Also

open(String, int)

list(String)

**android.content.res.Resources类**

**android.content.res.Resources.getAssets方法**

public final AssetManager getAssets ()

返回这些资源的底层AssetsManager存储。

**android.content.res.Resources.getText方法**

public CharSequence getText (int id)

返回与特定资源ID关联的string值。 The returned object will be a String if this is a plain string; it will be some other type of CharSequence if it is styled.

Parameters

想要的资源的标识符，由aapt工具生成。This integer encodes the package, type, and resource entry. The value 0 is an invalid identifier.

Returns

CharSequence The string data associated with the resource, plus possibly styled text information.

Throws

Resources.NotFoundException Throws NotFoundException if the given ID does not exist.

**android.content.res.Resources.openRawResource方法**

public InputStream openRawResource (int id)

打开一个数据流来读取raw资源。This can only be used with resources whose value is the name of an asset files -- that is, it can be used to open drawable, sound, and raw resources; it will fail on string and color resources.

##### Parameters

##### id 打开资源的标识符，由appt工具生成。

##### Returns

InputStream Access to the resource data.

##### Throws

*Resources.NotFoundException Throws NotFoundException if the given ID does not exist.*

**android.content.SharedPreference接口**

getSharedPreferences(String,int)返回的接口，用来读取和修改首选项数据。

**android.content.SharedPreference.edit方法**

public abstract SharedPreferences.Editor edit ()

创建一个首选项的新Editor，通过它你可以修改首选项中的数据并且提交这些改变到SharedPreferences对象。

Note that you must call commit() to have any changes you perform in the Editor actually show up in the SharedPreferences.

##### Returns

Returns a new instance of the SharedPreferences.Editor interface, allowing you to modify the values in this SharedPreferences object.

**android.content.SharedPreference.Editor接口**

使用此接口修改SharedPreferenced对象的值。

**android.content.SharedPreference.Editor.commit方法**

public abstract boolean commit ()

从Editor提交首选项修改到它编辑的SharedPreferences对象。这自动进行要求的修改，替换SharedPreferences中现在的项。

Note that when two editors are modifying preferences at the same time, the last one to call commit wins.

If you don't care about the return value and you're using this from your application's main thread, consider using apply() instead.

##### Returns

##### 如果新值成功写入到持续的存储中，返回true。

**android.database.Cursor接口**

##### implements Closeable

##### 这个接口提供对数据库查询返回的值的随机读写。

**android.database.Cursor.getColumnIndex方法**

#### public abstract int getColumnIndex (String columnName)

返回从0开始的给的列名的索引序号，如果不存在返回-1。 If you expect the column to exist usegetColumnIndexOrThrow(String) instead, which will make the error more clear.

##### Parameters

##### columnName 目标列的名称。

##### Returns

the zero-based column index for the given column name, or -1 if the column name does not exist.

##### See Also

getColumnIndexOrThrow(String)

**android.database.Cursor.moveToPosition方法**

#### public abstract boolean moveToPosition (int position)

移动cursor到任意的位置。The valid range of values is -1 <= position <= count.

This method will return true if the request destination was reachable, otherwise, it returns false.

##### Parameters

##### position 移动到的从0开始的位置。

##### Returns

whether the requested move fully succeeded.

**android.database.sqlite.SQLiteDatabase类**

##### extends SQLiteClosable

##### 提供方法来操作SQLite数据库。

**android.database.sqlite.SQLiteDatabase.execSQl方法**

public void execSQL (String sql)

执行一条非SELECT或其他返回数据的SQL语句。

It has no means to return any data (such as the number of affected rows). Instead, you're encouraged to use insert(String, String, ContentValues),update(String, ContentValues, String, String[]), et al, when possible.

When using enableWriteAheadLogging(), journal\_mode is automatically managed by this class. So, do not set journal\_mode using "PRAGMA journal\_mode' enableWriteAheadLogging()

##### Parameters

##### sql 要执行的SQL语句。通过分号分开多条语句是不支持的。

##### Throws

|  |  |
| --- | --- |
| SQLException | if the SQL string is invalid |

**android.database.sqlite.SQLiteDatabase.insert方法**

#### public long insert (String table, String nullColumnHack, ContentValues values)

在数据库中插入一行的方便方法。

##### Parameters

##### table 要插入行的表名。

##### MullColumnHack 可选的，可以是null。SQL doesn't allow inserting a completely empty row without naming at least one column name. If your provided values is empty, no column names are known and an empty row can't be inserted. If not set to null, the nullColumnHackparameter provides the name of nullable column name to explicitly insert a NULL into in the case where your values is empty.

##### Values 它包含了一行的列值。其中的键应该是列名，值应该是列值。

##### Returns

the row ID of the newly inserted row, or -1 if an error occurred

**android.database.sqlite.SQLiteDatabase.query方法**

#### public Cursor query (String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having,String orderBy)

查询表，返回一个带有结果的Cursor。

##### Parameters

##### table 用来查询的表名。

##### Columns 返回的列的列表。传入null将返回所有列，不鼓励从存储中读取不用的数据。

##### Selection 一个过滤器来决定返回哪些行，格式为SQL中的WHERE语句。传入null将返回所有行。

##### SelectiongArgs 为了它们能出现在selection中，你可以在selection参数中包含?，它将被selection中的值代替。这些值将被绑定为String。

##### GroupBy 一个过滤器来决定行如何分组，格式为SQL中的GROUP BY语句。传入null不分组。

##### Having 一个过滤器来决定哪些组包含在cursor中，格式为SQL中的HAVING语句。传入null将导致所有的组被包含，并且当groupby未使用时传入此值。

##### OrderBy 怎样排列行，格式为SQL中的ORDER BY语句。传入null使用默认排序，这可能是不排序。

**android.database.sqlite.SQLiteOpenhelper类**

extends Object

一个操作数据库创建和版本更新的帮助类。

**android.database.sqlite.SQLiteOpenhelper.SQLiteOpenHelper构造方法**

public SQLiteOpenHelper (Context context, String name, SQLiteDatabase.CursorFactory factory, int version)

创建一个对象来创建、打开、操作数据库 This method always returns very quickly. The database is not actually created or opened until one of getWritableDatabase() or getReadableDatabase() is called.

##### Parameters

##### context 用来打开或创建数据库。

##### Name 数据库的名字，或者是null来创建内存中的数据库。

##### Factory 用来创建cursor对象，或null为默认。

##### Version 数据库版本（从1开始）；如果数据库是旧的，onUpgrade方法将被用来升级数据库；如果数据库是更新的onDowngrade方法将用来变旧数据库。

**android.database.sqlite.SQLiteOpenhelper.getReadableDatabase方法**

#### public SQLiteDatabase getReadableDatabase ()

创建或打开一个数据库。This will be the same object returned by getWritableDatabase() unless some problem, such as a full disk, requires the database to be opened read-only. In that case, a read-only database object will be returned. If the problem is fixed, a future call to getWritableDatabase()may succeed, in which case the read-only database object will be closed and the read/write object will be returned in the future.

Like getWritableDatabase(), this method may take a long time to return, so you should not call it from the application main thread, including fromContentProvider.onCreate().

##### Returns

##### 一个数据库对象，直到getWritableDatabase()或close()被调用有效。

Throws

SQLiteException if the database cannot be opened

**android.database.sqlite.SQLiteOpenhelper.getWritableDatabase方法**

#### public SQLiteDatabase getWritableDatabase ()

创建或打开一个用来读写的数据库。The first time this is called, the database will be opened andonCreate(SQLiteDatabase), onUpgrade(SQLiteDatabase, int, int) and/or onOpen(SQLiteDatabase) will be called.

Once opened successfully, the database is cached, so you can call this method every time you need to write to the database. (Make sure to call close()when you no longer need the database.) Errors such as bad permissions or a full disk may cause this method to fail, but future attempts may succeed if the problem is fixed.

Database upgrade may take a long time, you should not call this method from the application main thread, including from ContentProvider.onCreate().

##### Returns

##### 一个读写数据库对象，直到close()方法被调用有效。

Throws

SQLiteException if the database cannot be opened for writing

**android.database.sqlite.SQLiteOpenhelper.onCreate方法**

public abstract void onCreate (SQLiteDatabase db)

数据库第一次创建时调用。This is where the creation of tables and the initial population of the tables should happen.

##### Parameters

##### db 数据库。

**android.os.AsyncTask<Params,Progress,Result>类**

##### AsyncTask使UI线程简便易用。这个类允许后台处理操作并且返回结果到UI线程，而不使用多线程和消息句柄。

**android.os.AsyncTask.AsyncTask<Params,Progress,Result>构造方法**

#### public AsyncTask ()

创建一个新的asynchronoustask。这个构造方法必须在主线程中调用。

**android.os.AsyncTask.doInBackground构造方法**

#### protected abstract Result doInBackground (Params... params)

重写此方法来执行一个后台线程。特定的参数是同够任务调用的execute(Params...)方法中传入的。这个方法能调用publishProgress(Progress...)来更新UI线程。

##### Parameters

##### Params 任务的参数

##### Returns

A result, defined by the subclass of this task.

##### See Also

onPreExecute()

onPostExecute(Result)

publishProgress(Progress...)

**android.os.AsyncTask.execute方法**

#### public final AsyncTask<Params, Progress, Result> execute (Params... params)

用特定的参数执行task。这个task返回它本身(this)，因此调用者能保持它的引用。

Note: this function schedules the task on a queue for a single background thread or pool of threads depending on the platform version. When first introduced, AsyncTasks were executed serially on a single background thread. Starting with DONUT, this was changed to a pool of threads allowing multiple tasks to operate in parallel. Starting HONEYCOMB, tasks are back to being executed on a single thread to avoid common application errors caused by parallel execution. If you truly want parallel execution, you can use the executeOnExecutor(Executor, Params...) version of this method withTHREAD\_POOL\_EXECUTOR; however, see commentary there for warnings on its use.

This method must be invoked on the UI thread.

##### Parameters

##### Params task的参数。

##### Returns

This instance of AsyncTask.

##### Throws

*IllegalStateException If* getStatus() *returns either* RUNNING *or* FINISHED*.*

##### See Also

executeOnExecutor(java.util.concurrent.Executor, Object[])

execute(Runnable)

**android.os.AsyncTask.onPostExecute方法**

#### protected void onPostExecute (Result result)

Runs on the UI thread after doInBackground(Params...). The specified result is the value returned by doInBackground(Params...).

This method won't be invoked if the task was cancelled.

##### Parameters

##### Result The result of the operation computed by doInBackground(Params...).

##### See Also

onPreExecute()

doInBackground(Params...)

onCancelled(Object)

**android.os.AsyncTask.onPreExecute方法**

#### protected void onPreExecute ()

Runs on the UI thread before doInBackground(Params...).

##### See Also

onPostExecute(Result)

doInBackground(Params...)

**android.os.AsyncTask.onProgressUpdate方法**

#### protected void onProgressUpdate (Progress... values)

当UI线程中的publishProgress(Progress...)方法被引用时运行。 The specified values are the values passed to publishProgress(Progress...).

##### Parameters

##### Values 调用程序中的值。

##### See Also

publishProgress(Progress...)

doInBackground(Params...)

**android.os.AsyncTask.publishProgress方法**

#### protected final void publishProgress (Progress... values)

This method can be invoked from doInBackground(Params...) to publish updates on the UI thread while the background computation is still running. Each call to this method will trigger the execution of onProgressUpdate(Progress...) on the UI thread.onProgressUpdate(Progress...) will not be called if the task has been canceled.

##### Parameters

##### Values 更新UI需要的处理值。

##### See Also

onProgressUpdate(Progress...)

doInBackground(Params...)

**android.os.BaseBundle类**

extends Object

**android.os.BaseBundle方法**

public void putString (String key, String value)

在Bundle的映射插入一个String值，代替给的key的任何存在的值。key或者value都有可能是null。

Parameters

key a String, or null

value a String, or null

**android.os.Binder类**

extends Object

implements IBinder

**android.os.Handler类**

extends Object

一个Handler允许你发送和处理与线程的MessageQueue联系的Message和Runnable对象。

**android.os.Handler.handleMessage方法**

public void handleMessage (Message msg)

子类必须实现它来接收消息。

**android.os.Message类**

extends Object

implements Parcelable

**android.os.Message.getData方法**

public Bundle getData ()

获得一个与此事件联系的任意数据的Bundle，如果需要懒散地创建它。通过调用setData(Bundle)设置这个值。Note that when transferring data across processes via Messenger, you will need to set your ClassLoader on the Bundle via Bundle.setClassLoader() so that it can instantiate your objects when you retrieve them.记住当你通过Messenger在程序中传递数据，你将需要通过Bundle.setClassLoader()在Bundle设置你的ClassLoader，从而当你重新获得他们时它能示例你的对象。

See Also

peekData()

setData(Bundle)

**android.os.Message.setData方法**

public void setData (Bundle data)

Sets a Bundle of arbitrary data values. Use arg1 and arg2 members as a lower cost way to send a few simple integer values, if you can.设置一个任意数据值的Bundle。使用arg1和arg2成员作为低消耗方式来发送一个简单的整型值，如果你能的话。

See Also

getData()

peekData()

**android.os.Parcelable接口**

**android.os.Parcelable.describeContents方法**

public abstract int describeContents ()

描述包含在Parcelable的已编组的表示中的特别对象

Returns

一个位掩码，指示被Parcelable编组的特殊对象设置。

**android.os.Parcelable.writeToParcel方法**

public abstract void writeToParcel (Parcel dest, int flags)

序列化此对象到Parcel中

Parameters

dest 应该被写入到的Parcel。

flags Additional flags about how the object should be written. May be 0 or PARCELABLE\_WRITE\_RETURN\_VALUE.

**android.os.Parcelable.Creator接口**

android.os.Parcelable.Creator<T>

**android.os.Parcelable.Creator.createFromParcel方法**

public abstract T createFromParcel (Parcel source)

创建一个新的Parcelable类实例，从给的Parcel实例化它，Parcel被Parcelable.writeToParcel()预先写好数据。

Parameters

source 从中读取object的数据的Parcel。

Returns

返回一个新的Parcelable类实例。

**android.os.Parcelable.Creator.newArray方法**

public abstract T[] newArray (int size)

创建一个Parcelable类的新数组。

Parameters

size 数组大小。

Returns

Returns返回一个Parcelable类的数组，包含每一个初始化为null的输入。

**android.os.Parcelable.describeContents方法**

public abstract void writeToParcel (Parcel dest, int flags)

添加对象到parcel中。

Parameters

dest 应该被写进的Parcel对象。

flags 添加对象应如何被写进的标志。 May be 0 or PARCELABLE\_WRITE\_RETURN\_VALUE.

**android.os.Parcelable.Environment类**

extends Object

提供通向环境变量。

**android.os.Parcelable.Environment类**

public static File getExternalStorageDirectory ()

返回先前的外部存储目录（wpf：sdcard目录）。

详细信息请查看原文档。

**android.preference.PreferenceActivity类**

extends ListActivity  
implements PreferenceFragment.OnPreferenceStartFragmentCallback

这是activity类的基类来展示给用户首选项的层级。

**android.preference.PreferenceActivity.addPreferencesFromResource方法**

public void addPreferencesFromResource (int preferencesResId)

This method was deprecated in API level 11.This function is not relevant for a modern fragment-based PreferenceActivity.

解析给的XML资源并且将preference层级添加到现在的preference层级。

Parameters

preferencesResId 解析的XML资源ID。

**android.preference.PreferenceActivity.getPreferenceManager方法**

#### public PreferenceManager getPreferenceManager ()

#### This method was deprecated in API level 11.此方法与现在的与现在以fragment为基础的PreferenceActivity不相关。

#### 返回此activity使用的PreferenceManager。

#### Returns

The PreferenceManager.

**android.preference.PreferenceManager类**

extends Object

用来帮助从XML或activity创建Preference层级。

**android.preference.PreferenceManager.findPreference方法**

#### public Preference findPreference (CharSequence key)

通过键找到一个Preference。

##### Parameters

##### key 要检索的preference的键。

##### Returns

The Preference with the key, or null.

##### See Also

findPreference(CharSequence)

**android.support.v4.app.ActionBarDrawerToggle类**

extends Object  
implements DrawerLayout.DrawerListener

这个类提供了一个方便的方法来联系DrawerLayout和ActionBar以实现导航抽屉的推荐设计。

这个类已过时，请使用support-v7-appcompat中的ActionBarDrawerToggle。

**android.support.v4.app.ActionBarDrawerToggle.ActionBarDrawerToggle构造方法**

public ActionBarDrawerToggle (Activity activity, DrawerLayout drawerLayout, int drawerImageRes, int openDrawerContentDescRes, int closeDrawerContentDescRes)

构造一个新的ActionBarDrawerToggle。

The given Activity will be linked to the specified DrawerLayout. The provided drawer indicator drawable will animate slightly off-screen as the drawer is opened, indicating that in the open state the drawer will move off-screen when pressed and in the closed state the drawer will move on-screen when pressed.

String resources must be provided to describe the open/close drawer actions for accessibility services.

##### Parameters

##### activity drawer属于的Activity。

##### DrawerLayout 链接到给的activity的actionBar的DrawerLayout。

##### DrawerImageRes 作为drawer指示器的drawable资源。

##### OpenDrawerContentDescRes 一个字符串资源描述”open drawer”action用于可访问性。

##### CloseDrawerContentDescRes 一个字符串资源描述”close drawer”action用于可访问性。

**android.support.v4.app.FragmentActivity类**

extends Activity

**android.support.v4.app.FragmentActivity.getSupportFragmentManager方法**

public FragmentManager getSupportFragmentManager ()

返回与activity联系的fragment相互作用的FragmentManager。

**android.support.v4.app.FragmentPagerAdapter类**

extends PagerAdapter

PagerAdapter类的继承，代表在fragment manager中持续保持的fragment的每个页面，只要用户能返回到页面。

**android.support.v4.widget.DrawerLayout类**

extends ViewGroup

DrawerLayout是一个顶级窗口内容容器，它能表现出抽屉视图，能从窗口中拖出。

**android.support.v4.widget.DrawerLayout.closeDrawer方法**

public void closeDrawer (View drawerView)

关闭特定抽屉视图，通过其中的第二个视图。

##### Parameters

##### drawerView 关闭的抽屉（第二个视图）。

**android.support.v4.widget.DrawerLayout.isDrawerOpen方法**

#### public boolean isDrawerOpen (View drawer)

、检查给的drawer是否处于打开状态To be considered "open" the drawer must have settled into its fully visible state. To check for partial visibility use isDrawerVisible(android.view.View).

##### Parameters

##### drawer 检查的Drawer视图（DrawerLyaout中第二个视图）。

##### Returns

true if the given drawer view is in an open state

##### See Also

isDrawerVisible(android.view.View)

**android.support.v7.app.ActionBarDrawerToggle类**

extends Object  
implements DrawerLayout.DrawerListener

这个类提供了一个方便的方法来联系DrawerLayout和ActionBar以实现导航抽屉的推荐设计。

**android.support.v7.app.ActionBarDrawerToggle.ActionBarDrawerToggle构造方法**

public ActionBarDrawerToggle (Activity activity, DrawerLayout drawerLayout, Toolbar toolbar, int openDrawerContentDescRes, int closeDrawerContentDescRes)

构造一个新的ActionBarDrawerToggle。

The given Activity will be linked to the specified DrawerLayout and the Toolbar's navigation icon will be set to a custom drawable. Using this constructor will set Toolbar's navigation click listener to toggle the drawer when it is clicked.

This drawable shows a Hamburger icon when drawer is closed and an arrow when drawer is open. It animates between these two states as the drawer opens.

String resources must be provided to describe the open/close drawer actions for accessibility services.

Please use ActionBarDrawerToggle(Activity, DrawerLayout, int, int) if you are setting the Toolbar as the ActionBar of your activity.

##### Parameters

##### activity drawer属于的Activity。

##### DrawerLayout 链接到给的activity的actionBar的DrawerLayout。

##### Toolbar 如果你的应哟能依赖Toolbar，这个Toolbar。

##### OpenDrawerContentDescRes 一个字符串资源描述”open drawer”action用于可访问性。

##### CloseDrawerContentDescRes 一个字符串资源描述”close drawer”action用于可访问性。

**android.support.v7.app.ActionBarDrawerToggle.onConfigurationChanged方法**

#### public void onConfigurationChanged (Configuration newConfig)

此方法应总是被调用，在你的Activity的onConfigurationChanged方法中。

##### Parameters

##### newConfig 新的configuration。

**android.support.v7.app.ActionBarDrawerToggle.onDrawerClosed方法**

DrawerLayout.DrawerListener的回调方法。 If you do not use your ActionBarDrawerToggle instance directly as your DrawerLayout's listener, you should call through to this method from your own listener object.

##### Parameters

##### drawerView 现在关闭的Drawer view。

**android.support.v7.app.ActionBarDrawerToggle.onDrawerOpened方法**

public void onDrawerOpened (View drawerView)

DrawerLayout.DrawerListener的回调方法。If you do not use your ActionBarDrawerToggle instance directly as your DrawerLayout's listener, you should call through to this method from your own listener object.

##### Parameters

##### drawerView 现在打开的Drawer view。

**android.support.v7.app.ActionBarDrawerToggle.onOptionsItemSelected方法**

public booleanonOptionsItemSelected (MenuItem item)

这个方法应该被Activity的onOptionsItemSelected调用。如果返回true，你的onOptionsItemSelected将返回true并且跳过以后的处理。

##### Parameters

##### item 代表选择的menu项目的MenuItem实例。

##### Returns

true if the event was handled and further processing should not occur

**android.support.v7.app.ActionBarDrawerToggle.syncState方法**

public void syncState ()

与链接的DrawerLayout同步drawer indicator/affordance（wpf:ActionDrawerToggle）状态。

This should be called from your Activity's onPostCreate method to synchronize after the DrawerLayout's instance state has been restored, and any other time when the state may have diverged in such a way that the ActionBarDrawerToggle was not notified. (For example, if you stop forwarding appropriate drawer events for a period of time.)

**android.support.v7.widget.RecyclerVIew.Adapter抽象类**

extends Object

Adapter类的基类。

Adatpter从一个\_设置显示在RecyclerView中的view的特定应用数据\_提供一个绑定。

**android.support.v7.widget.RecyclerVIew.Adapter.getItemCount方法**

public abstract int getItemCount ()

返回adapter持有的数据设置中的项目总数。

##### Returns

The total number of items in this adapter.

**android.support.v7.widget.RecyclerVIew.Adapter.onBindViewHolder方法**

public abstract void onBindViewHolder (VH holder, int position)

通过RecyclerView调用来在特定位置显示数据。这个方法应该更新view项目的内容来反映给定位置的项目。

Note that unlike ListView, RecyclerView will not call this method again if the position of the item changes in the data set unless the item itself is invalidated or the new position cannot be determined. For this reason, you should only use the position parameter while acquiring the related data item inside this method and should not keep a copy of it. If you need the position of an item later on (e.g. in a click listener), use getAdapterPosition() which will have the updated adapter position.

##### Parameters

##### holder 应该被更新的ViewHolder来代表在在数据设置中的给定位置的项目的内容。

##### position 带有adapter数据设置的项目位置。

**android.support.v7.widget.RecyclerVIew.Adapter.onCreateViewHolder方法**

public abstract VH onCreateViewHolder (ViewGroup parent, int viewType)

当RecyclerVIew需要一个新的\_代表特定项目的给定类型\_的RecyclerView.ViewHolder时调用。

这个新的ViewHolder必须用一个新的View构造，这个View能代表给定类型的项目。You can either create a new View manually or inflate it from an XML layout file.

The new ViewHolder will be used to display items of the adapter using onBindViewHolder(ViewHolder, int). Since it will be re-used to display different items in the data set, it is a good idea to cache references to sub views of the View to avoid unnecessary findViewById(int) calls.

##### Parameters

##### parent 新View在绑定到adapter后将被添加到的ViewGroup。

##### ViewTpye 新View的view类型。

##### Returns

一个新的ViewHolder保有给的view类型的View。

See Also

getItemViewType(int)

onBindViewHolder(ViewHolder, int)

**android.support.v7.app.ActionBarActivity类**

**android.support.v7**.**ActionBarActivity.** **getMenuInflater方法**

public MenuInflater getMenuInflater ()

根据上下文返回MenuInflater值。

**android.support.v7.app**.**ActionBarActivity.** **onCreate方法**

protected void onCreate (Bundle savedInstanceState)

初始化所有片段和装载程序的平台

Parameters

savedInstanceState 如果activity正在被重新初始化在先前被关闭之后，那么这个Bundle包含最近在onSaveInstaceState(Bundle)中提供的数据。

Note: 否则它将为空。

**android.support.v7.app**.**ActionBarActivity.setContentView方法**

public void setContentView (int layoutResID)

从布局资源中设置activity目录。资源将被使用，添加所有的顶级视图到activity中

Parameters

layoutResID 被使用的资源id。

**android.support.v7.app.AppCompatActivity类**

extends FragmentActivity  
implements AppCompatCallback TaskStackBuilder.SupportParentable ActionBarDrawerToggle.DelegateProvider

**android.support.v7.app.AppCompatActivity.getSupportActionBar方法**

public ActionBar getSupportActionBar ()

Support library version of getActionBar().

检索此activity的ActionBar引用。

##### Returns

The Activity's ActionBar, or null if it does not have one.

**android.support.v7.app.AppCompatActivity.supportInvalidateOptionsMenu方法**

public void supportInvalidateOptionsMenu ()

Support library version of invalidateOptionsMenu().

作为activity的menu设置。这将导致下次需要菜单时有关菜单的设置全部更新，通过调用onCreateOptionsMenu和onPrepareOptionsMenu

**android.support.v7.widget.GridLayoutManager类**

extends LinearLayoutManager

**android.support.v7.widget.GridLayoutManager.GridLayoutManager构造方法**

public GridLayoutManager (Context context, int spanCount, int orientation, boolean reverseLayout)

##### Parameters

##### context 当前的context，将被用来链接到资源。

##### spanCont 表格中行或列的数量。

##### orentation Layout走向，应该是HORIZONTAL或VERTICAL。

##### reverseLayout 是否反转，当设置为true，layout从尾部开始。

**android.support.v7.widget.LinearLayoutManager类**

extends LinearLayoutManager

**android.support.v7.widget.LinearLayoutManager.LinearLayoutManager构造方法**

public LinearLayoutManager (Context context, int orientation, boolean reverseLayout)

##### Parameters

##### context 当前的context，将被用来链接到资源。

##### orentation Layout走向，应该是HORIZONTAL或VERTICAL。

##### reverseLayout 是否反转，当设置为true，layout从尾部开始。

**android.support.v7.widget.RecyclerView类**

extends ViewGroup  
implements ScrollingView

一个灵活的view用来在大的数局设置中提供一个限制的窗口。

**android.support.v7.widget.RecyclerView.setAdapter方法**

public void setAdapter (Adapter adapter)

设置一个新的适配器来提供子view需求。（wpf:adapter中控制(添加等)view）

当适配器改变，所有存在的view被回收回池。如果池里只有一个适配器，它将被清空。

##### Parameters

##### adapter 设置的新的适配器，如果为null不设置适配器。

##### See Also

swapAdapter(Adapter, boolean)

**android.support.v7.widget.RecyclerView.setLayoutManager方法**

public void setLayoutManager (RecyclerView.LayoutManager layout)

设置RecyclerView将使用的RecyclerView.LayoutManager。

In contrast to other adapter-backed views such as ListView or GridView, RecyclerView allows client code to provide custom layout arrangements for child views. These arrangements are controlled by the RecyclerView.LayoutManager. A LayoutManager must be provided for RecyclerView to function.

Several default strategies are provided for common uses such as lists and grids.

##### Parameters

##### layout 使用的LayoutManager。

**android.support.v7.widget.RecyclerView.LayoutManager类**

extends Object

一个LayoutManager用来在RecyclerView中测量和放置view项目，并且recyclerview项目不再可见时决定策略。

（wpf:有实现类LinearLayoutManager/GridLyoutManager，使用此类可实现类似ListView的作用)

**android.util.ArrayList<T>类**

extends AbstractList<E>  
implements Cloneable Serializable RandomAccess

ArrayList是一个List的实现，依赖数组。所有可选的操作，包括添加、移除、替换等方法被提供。

**android.util.Log类**

**android.util.Log.d方法**

#### public static int d (String tag, String msg)

发送一个DEBUG日志信息。

##### Parameters

##### tag 用来标识一个日志信息资源。它通常用来标识日志出现的class或activity。

##### msg 日志信息。

**android.util.Log.e方法**

**android.util.Log.i方法**

**android.util.Log.v方法**

**android.util.Log.w方法**

**android.view.ContextThemeWrapper类**

extends ContextWrapper

**android.view.ContextThemeWrapper.getResources方法**

public Resources getResources ()

返回一个你的应用包的资源实例。

**android.view.LayoutInflater类**

extends Object

实例化一个layout XML文件到它相应的View对象中。它从不被直接使用。而是，使用getLayoutInflater()或getSystemService(String)来重新检索一个已经链接到当前context和正确配置到你运行的设备的标准LayoutInflate实例。

**android.view.LayoutInflater.from方法**

public static LayoutInflater from (Context context)

从给的context中获得LayoutInflater。

**android.view.LayoutInflater.inflate方法**

public View inflate (int resource, ViewGroup root)

从特定的xml资源中解析一个新的view层次。如果有错误，抛出InflateException。

##### Parameters

##### resource 要装载的XML layout ID。（例如，R.layout.main\_page）

##### root 可选的view来做子层次的父亲。

##### Returns

解析的层次的根View。如果根被提供，这是根View；否则它是解析的XML文件的根，

**android.view.LayoutInflater.inflate方法**

public View inflate (int resource, ViewGroup root, boolean attachToRoot)

从特定的xml资源中解析一个新的view层次。如果有错误，抛出InflateException。

##### Parameters

##### resource 要装载的XML layout ID。（例如，R.layout.main\_page）

##### root 可选的view来做子层次的父亲（如果attachToRoot为true），或者否则简化一个为返回层次的根提够一个LayoutParams值设置的对象（如果attachToRoot位false）。

##### AttachToRoot inflate的层次是否应该被匹配到根参数？如果是false，根只被用来为XML中根view创建LayoutParames的正确子类

##### Returns

inflate的层次的根View。如果根被提供并且attachToRoot为true，这是根；否则它是inflate的XMLfile的根。

**android.view.** **MenuInflater类**

**android.view. MenuInflater.** **inflate方法**

void inflate(int menuRes, Menu menu)

从定义的XML资源中形成一个菜单层级。

**android.view.MenuItem接口**

**android.view.MenuItem.getItemId方法**

public abstract int getItemId ()

返回菜单项目的标识符。标识符在菜单创建后不能改变。

Returns

菜单项目的标识符

**android.view.View. OnClickListener接口**

**android.view.View. OnClickListener.onClick方法**

public abstract void onClick (View v)

当视图被点击时调用

Parameters

v 被点击的视图。

**android.view.View类**

**android.view.View.getId()**

public int getId ()

Returns this view's identifier.返回view的标识符（wpf：xml中定义）。

Related XML Attributes

android:id

Returns

a positive integer used to identify the view or NO\_ID if the view has no ID

See Also

setId(int)

findViewById(int)

**android.view.View.setOnClickListener方法**

public void setOnClickListener (View.OnClickListener l)

注册一个回调函数来调用，当视图被点击。如果视图为not clickable，它将变为clickable。

Parameters

l 将要运行的回调

See Also

setClickable(boolean)

**android.webkit.AdapterView.OnItemClickListener接口**

**android.webkit.AdapterView.OnItemClickListener.onItemClick方法**

public abstract void onItemClick (AdapterView<?> parent, View view, int position, long id)

当view中的项目被点击时调用的回调方法。

Implementers can call getItemAtPosition(position) if they need to access the data associated with the selected item.

##### Parameters

##### parent 点击发生时的AdapterView。

##### view AdapterView中的被点击的view。（这是adapter提供的view）。

##### position view在adapter中的位置。

##### id 点击的项目的行id。

**android.webkit.AdapterView.OnItemSelectedListener接口**

**android.webkit.AdapterView.OnItemSelectedListener.onItemSelected方法**

public abstract void onItemSelected (AdapterView<?> parent, View view, int position, long id)

当view中的项目被选择时调用的回调方法。This callback is invoked only when the newly selected position is different from the previously selected position or if there was no selected item.

Impelmenters can call getItemAtPosition(position) if they need to access the data associated with the selected item.

Parameters

##### parent 选择发生时的AdapterView。

##### view AdapterView中的被点击的view。

##### position view在adapter中的位置。

##### id 选择的项目的行id。

**android.webkit.ArrayAdapter<T>类**

extends BaseAdapter  
implements Filterable

一个被任意对象数组支持的实在的BaseAdapter。

**android.webkit.ArrayAdapter.ArrayAdapter<T>构造方法**

public ArrayAdapter (Context context, int resource, int textViewResourceId, T[] objects)

构造函数。

##### Parameters

##### context 当前的context。

##### resource 包含一个当实例化view时使用的layout文件的资源ID。

##### textViewResourceId 在layout资源中TextView的id来被填充。

##### object 在ListView中代表的对象。

**android.webkit.ListView类**

extends AbsListView

一个展示垂直滚动项目列表的view。项目来自与这个view联系的ListAdapter（wpf：或其他Adapter，例如ArrayAdapter\siroleCursorAdapter）。

**android.webkit.ListView.setAdapter方法**

public void setAdapter (ListAdapter adapter)

设置此ListView中的数据。The adapter passed to this method may be wrapped by a WrapperListAdapter, depending on the ListView features currently in use. For instance, adding headers and/or footers will cause the adapter to be wrapped.

##### Parameters

##### adapter 这个ListAdapter负责保持数据返回到这个列表，并且产生一个视图来代表数据设置选项。

##### See Also

getAdapter()

**android.webkit.WebView类**

**android.webkit.WebView.loadUrl方法**

public void loadUrl (String url)

加载给的URL。

##### Parameters

##### url 要加载的URL。

**android.widget.CursorAdapter抽象类**

##### extends BaseAdapter implements Filterable

##### 将Cursor的数据暴露给ListView的Adapter。

**android.widget.SimpleCursorAdapter.changeCursor方法**

public void changeCursor (Cursor cursor)

将底层的cursor替换为一个新cursor。如果没有存在的cursor它将被关闭。

##### Parameters

##### cursor 使用的新cursor。

**android.widget.SimpleCursorAdapter.getCursor方法**

public Cursor getCursor ()

返回Adapter中的cursor。

##### Returns

the cursor.

**android.widget.EditText类**

**android.widget.EditText.getText方法**

public Editable getText ()

返回TextView正显示的文本。如果setText()被带有BufferType.SPANNABLE 或BufferType.EDITABLE参数调用，你能将这个方法的返回值分别传到Spannable或Editanle。记住：返回值和内容不能被修改。如果你想修改，应该先复制一个。

**android.widget.ImageView类**

extends View

**android.widget.ImageView.ImageView构造方法**

public ImageView (Context context)

**android.widget.ImageView.setImageResource方法**

public void setImageResource (int resId)

设置一个drawable作为这个ImageView的内容。

This does Bitmap reading and decoding on the UI thread, which can cause a latency hiccup. If that's a concern, consider usingsetImageDrawable(android.graphics.drawable.Drawable) or setImageBitmap(android.graphics.Bitmap) and BitmapFactory instead.

##### Related XML Attributes

##### android:src

##### Parameters

##### resId drawable的资源标识符。

**android.widget.SimpleCursorAdapter类**

##### extends ResourceCursorAdapter

##### 一个简单的Adapter来从cursor匹配列到XML文件中的TextView或ImageView。

**android.widget.SimpleCursorAdapter.SimpleCursorAdapter构造方法**

public SimpleCursorAdapter (Context context, int layout, Cursor c, String[] from, int[] to, int flags)

标准构造方法。

##### Parameters

##### context 与SimpleListItenFactory联系的ListView运行的context。

##### Layout 定义列表项view的layout文件资源标识符。layout文件中至少应包含to参数中定义的名称。

##### C 数据库cursor。如果cursor暂未或取，可以是null。

##### From 代表数据的列的名字。如果cursor暂未或取，可以是null。

##### To 显示from参数中列的视图名。这些都应该是TextView。from参数中的第n个值放入列表中的第n个视图中。如果cursor暂未或取，可以是null。

##### Flags 决定adapter的行为，按照CursorAdapter(Context,Cursor,int)方法。

**android.widget.Spinner类**

extends AbsSpinner  
implements DialogInterface.OnClickListener

一次显示一个孩子的view，并且让用户从它们之中选择。

**android.widget.Spinner.setAdapter方法**

#### public void setAdapter (SpinnerAdapter adapter)

设置用来提供在Spinner中选择的数据的适配器。

Note that Spinner overrides getViewTypeCount() on the Adapter associated with this view. Calling getItemViewType(int) on the object returned fromgetAdapter() will always return 0. Calling getViewTypeCount() will always return 1. On API LOLLIPOP and above, attempting to set an adapter with more than one view type will throw an IllegalArgumentException.

##### Parameters

##### adapter 要设置的adapter。

##### Throws

|  |  |
| --- | --- |
| *IllegalArgumentException* | if the adapter has more than one view type |

##### See Also

setAdapter(SpinnerAdapter)

**android.widget.TextView类**

extends View

implements ViewTreeObserver.OnPreDrawListener

**android.widget.TextView.TextView构造方法**

public TextView (Context context)

**android.widget.TextView.setText方法**

public final void setText (CharSequence text)

设置TextView的字符串值。TextView不接收HTML格式，你可以使用XML资源文件中的字符串。To style your strings, attach android.text.style.\* objects to a SpannableString, or see the Available Resource Types documentation for an example of setting formatted text in the XML resource file.要样式化你的字符串，附加android.text.style.\*对象到一个SpannableString,或者查看Avaliable Resource Types文档中的在XML资源中设置样式的例子。

Related XML Attributes

android:text

**android.widget.Toast类**

extends Object

toast是一个包含用户快速小信息的视图。

**android.widget.Toast.makeText方法**

public static Toast makeText (Context context, CharSequence text, int duration)

产生一个只包含一条信息的标准toast。

##### Parameters

##### context 使用的context。通常是你的Application或Activity对象。

##### Text 显示的信息。可以是格式化的信息。

##### Duration 信息显示多长时间。可以是LENGTH\_SHORT或LENGTH\_LONG。

**android.widget.Toast.show方法**

public void show ()

显示view特定时间。

**Java.net.HttpURLConnection类**

extends URLConnection

一个用于HTTP(RFC2616)的URLConnection用来发送和接收web数据。

**Java.net.HttpURLConnection.setRequestMethod方法**

#### public void setRequestMethod (String method)

设置将要发送到远程HTTP服务器的请求命令。This method can only be called before the connection is made.

##### Parameters

##### Method 此字符串代表使用的方法。

##### Throws

ProtocolException if this is called after connected, or the method is not supported by this HTTP implementation.

##### See Also

getRequestMethod()

method

**Java.net.URL类**

extends Objectimplements Serializable

一个格式化的资源位置，代表RFC1738标准定义的网络资源位置。

**Java.net.URL.URL构造方法**

public URL (String spec)

通过传入的spec创建一个新的URL实例。

##### Throws

*MalformedURLException if* spec *could not be parsed as a URL.*

***Java.net.URL.openConnection*方法**

#### public URLConnection openConnection ()

返回连接到代表资源的URL的新的Connection。

##### Throws

*IOException if an error occurs while opening the connection.*

**Java.net.URLConnection类**

extends Object

用于读或写的URL连接。用于http连接，查看HttpURLConnection来查看http特定特性。

**Java.net.URLConnection.getInputStream方法**

#### public InputStream getInputStream ()

返回一个InputStream用于从URLConnection指向的资源中读取数据。 It throws an UnknownServiceException by default. This method must be overridden by its subclasses.

##### Returns

the InputStream to read data from.

##### Throws

*IOException if no InputStream could be created.*

**Java.net.URLConnection.setDoOutput方法**

*public void setDoOutput (boolean newValue)*

设置标签代表此*URLConnection*是否允许*output*。*It cannot be set after the connection is established.*

##### *Parameters*

##### *NewValue* 设置的新的标签值。

##### *Throws*

##### See Also

*doOutput*

**Application**

**android:name** 绑定的用于存储数据的实现application接口的类名。程序包运行时此类将实例化。此选项是非必须的。

**Activity**

**android:exported**  是否可以被其他应用访问 true/false

**android:launchMode** 启动Activity的模式 standard/singleTop实例处于栈顶不创建实例，不处于栈顶创建新实例/singleTask只创建一个实例/singleInstance每个实例存在于一个任务栈中，创建独立的任务栈

label标注Activity的标签

**mata-data 额外名值对数据**

**android:name** 名称

**android:value** 值

**intent-filter Intent过滤器**

**android:priority**  设置优先级，数字越大优先级越高

**action 启动Intent**

**android:name** action名称（通常用于Intent构造）

**cetegory activity的行为种类**

**android:name**种类名称android.intent.category.DEFAULT默认/android.intent.category.BROWSABLE可被浏览器安全打开/android.intent.category.LAUNCHER设置为顶级启动器

**data 匹配的数据类型**

**android:scheme** 定义一个句柄到的URI协议，按照IntentFilter.addDataScheme()

**permission 声明权限**

**android:name**  权限名称

**uses-permission 请求应用必须被授予的权限**

**android:name** 权限名称

**android:background** 背景颜色 #xxxxxx

**android:choiceMode** listview视图的选择模式 singleChoice/multipleChoice

**android:dialogTitle** prefernecescreen的edittextpreference生成对话框的标题

**android:dialogMessage** prefernecescreen的edittextpreference生成对话框的内容

**android:divider** listview视图的分割颜色或图片 #xxxxxx/@xxxxxx

**android:dividerHeight** 分割线宽度 ndp

**android:entries** prefernecescreen的listpreference链接一个数组的键 @array/

**android:entryValues** prefernecescreen的listpreference链接一个数组的值 @array/

**android:gravity** 空间中内容的位置 left/right/center

**android:hint** edittext未输入内容时提示

**android:icon** menu项图标 @xxx/xxx

**android:id**  ID @+id/xxx

**android:key** prefernecescreen中标签的键。

**android:layout\_gravity** 视图的滑动方向 start自左向右/end自右向左

**android:layout\_height** 高度 match\_parent匹配父容器/wrap\_content根据内容确定

**android:layout\_margin** 边距 ndp

**android:layout\_marginTop** 上边距 ndp

**android:layout\_weight** 分割父容器的比例

**android:layout\_width** 宽度 match\_parent匹配父容器/wrap\_content根据内容确定

**android:orientation** 走向 horizontal水平/vertical垂直

**android:src**  内容路径 @xxx/xxx

**android:summary** 摘要，标题下的文字

**android:summaryOn** 某功能开启时的摘要

**android:summaryOff** 某功能关闭时的摘要

**android:tag** 额外数据

**android:text**  文字内容

**android:textSize** 文字大小 nsp

**android:theme** 使用的主题

**android:title** 标题，文字说明

**android:visibility**  是否可见 visible可见/invisible不可见

**app:showAsAction** menu中项目是否显示（此属性在Android2.1之前要指明命名空间：xmlns:yourapp=”http://schemas.android.com/apk/res-auto”） always/never/ifRoom有位置才显示/withText与文字一起显示

**3.1认识Android中的Activity组件**

**3.1.2Activity绑定自定义视图**

通过setContentView(资源)方法指定R.layout中的Activity视图。

**3.1.3启动另一个Activity**

使用startActivity(Intent)方法启动另一个Activity。

**3.2Activity生命周期**

**3.2.1认识Activity声明周期**

见android.app.Activity。

**3.2.3在Activity跳转过程中的生命周期**

**3.3在Activity之间传递参数**

**3.3.1传递简单数据**

使用Intent.putExtra(数据名，数据内容)方法传递数据；使用Intent.getxxxExtra(数据名)方法获取数据。

使用getIntent()方法可以获得打开activity的Intent。

**3.3.2传递数据包Bundle**

使用Bundle.putxxx(数据名，数据)方法存储数据到Bundle，使用Bundle.getxxx(数据名，「不存在时的默认值」)方法获取数据。

使用Intent.putExtras(数据)方法传递一个数据，使用Intent.putExtras()方法获取数据。

**3.3.3传递值对象**

传递自定义数据类型：

1）实现Serializable接口，系统自动进行序列化；

2）实现Parcelable接口：Parcelable.describeContents();Parcelable.writeToParcel(Parcel dest,int flags)，在其中使用dest.writexxx(数据)实现序列化。实现Creator<T>接口并创建其静态常量对象：createFromParcel(Parcel source)，在其中使用source.readxxx()方法，重新构成T对象并返回;newArray(int size)，返回T[size]。若有多个同样类型数据，可以使用Bundle。

**3.3.4获取Activity的返回参数**

使用startActivityForResult(Intent,请求码)方法打开Activity。重写onActivityResult(int requestCode,int resultCode,Intent data)方法，其中data是setResult方法中传递的数据。

被打开的Activity中，使用setResult(返回码,Intent)返回值。

可以使用finish()方法结束Activity。

**3.4Android中Activity中启动模式**

**3.4.1标准启动模式**

在Manifest中的activity标签下android:launchMode设置启动模式。

可使用getTaskID()方法获取任务ID。

**3.4.2SingleTop启动模式**

**3.4.3SingleTask与SingleInstance模式**

**3.5在 Android 中 Intent 的概念及应用**

**3.5.1显式Intent**

**3.5.2隐式Intent**

在Manifest中的Activity.Intent-filter.action中设置action值，使用Intent(action字符串)构造Intent,并使用startActivity(Intent)打开Activity。通常在类中添加ACTION常量。

可以使用此方法打开其他应用的Activity。

**3.5.3Intent过滤器相关选项**

如果多个Activity拥有相同的action,可以使用在Manifest中的Activity.Intent-filter.data中android:scheme等等加以区分，使用Intent(action字符串，uri)构造Intent。

**3.5.4通过浏览器链接启动本地Activity**

在Manifest中的Activity.Intent-filter.cetegory设置Activity的行为方式。

在Manifest中的Activity.Intent-filter.data中设置地址。

**3.6Android 中 Context 的理解及使用**

**3.6.1Context的作用**

通过Context获取全局信息。

可使用getResources().getxxx()方法获取全局信息。

**3.6.2Application的用途**

使用继承Application的类存储全局共享数据，Application继承自Context。

使用getApplicationContext()方法获得上述类中的数据。

**3.6.3Application生命周期**

生命周期包括：onCreate()，启动应用时执行，并且先于Activity的onCreate()，可将初始化工作放于其中;onTerminate(),onLowMemory(),onTrimMemory(),onConfigurationChange()。

**3.7认识Android Service**

**3.7.1使用Service**

使用startService(Intent)方法启动Service；使用stopService(Intent)停止服务。

**3.7.2绑定Service**

使用bindService(Intent,服务帧听连接,创建方式)绑定服务； 使用unbindService(服务帧听连接)解除绑定服务。

服务帧听链接是实现ServiceConnection的类，需要重写：onServiceConnected(ComponentName name,Ibinder service)方法，此方法在服务绑定成功时执行；onServiceDisconnected(Component name)方法。

**3.7.3Service生命周期**

生命周期包括：onCreate();onDestroy();onStartCommand()。

**3.8Android中Service通信**

**3.8.1启动Service并传递数据**

通过打开Service的Intent传递数据。

**3.8.2绑定Service进行通信（上）**

服务监听类中onServiceConnected(ComponentName name,Ibinder service)方法的第二个参数是服务类中onBind(Intent)方法的返回值。可以返回自定义的服务类的内部类使其继承Binder，这样就可以在监听类中访问和修改服务类。

**3.8.3绑定Service进行通信（下）**

可以在服务类中添加返回本服务this的方法，通过上述Binder对象调用此方法就可以得到服务this,进而对服务进行访问和修改。

可以在服务类中添加内部回调接口，并创建此接口对象和get/set方法，在服务类中调用此对象的方法，但该方法的实现在服务类外进行（通常是在set时）。

其他线程修改UI线程资源时，不能直接修改，要用发送消息的方式。定义Handler对象，重写其handleMessage方法用于接收消息，在其中完成对UI线程的修改；使用Handleer.sendMessage(Message)方法发送消息。

**3.9Android中AIDL的理解与使用**

**3.9.1跨应用启动Service**

使用Intent.setComponent(Component)初始化Intent，其中Component构造方法为Component(要打开的服务所在包名，服务类名或全路径)。

**3.9.2跨应用绑定Service**

创建AIDL，就可以在服务类的onBind方法中直接返回AIDL的stub并实现basicType方法。

**3.9.3跨应用绑定Service并进行通信**

在AIDL中添加方法(通常在服务类中创建AIDL对象时实现)，以访问和修改服务类。

打开服务的类和服务被打开的类必须有相同的AIDL定义并且包名相同。

AIDL转换成Binder时不能直接进行强制类型转换，要使用AIDL接口.Stub.asInterface(AIDL转换成的Binder对象）方法。

**3.10Android 广播接收器 BroadcastReceiver**

**3.10.1使用BroadcastReceiver**

使用sendBroadcast(Intent)方法发送消息；广播接收类中使用onReceive(Context context,Intent intent)方法接收广播。

**3.10.2动态注册和注销BroadcastReceiver**

在AndroidManifest删除receiver标签，若不注册接收器发送将无效。

使用registerReceiver(待注册receiver，IntentFilter)方法注册接收器；使用unregisterReceiver(receiver)方法注销接收器。IntentFilter和发送的Intent要使用隐式构造。

**3.10.3BroadcastReceiver的优先级**

对于拥有相同action的receiver，可以使用intent-filter中android:priority属性指定优先级。在接收器类中使用abortBroadcast()方法后面的接收器将不再接收广播。

**3.11Android日志系统**

发送日志的方法包括Log.e(TAG,日志内容);Log.w;Log.i;Log.d;Log.v。

**3.12Android权限系统**

**3.12.1请求权限实例**

在AndroidManifest中使用uses-permission标签请求权限。

**3.12.2为代码添加权限检查**

在AndroidManifest中使用permission标签声明权限。

使用Context.checkCallingOrSelfPermission(权限名)检查是否拥有权限。

**3.12.3为基本组件添加权限检查**

在AndroidManifest的activity标签中的android:permission属性设置组件所需要的权限，此权限只在通过其他应用启动此组件时才需要。

**4.1Android Fragment**

**4.1.1使用Fragment**

使用getFragmentManager()/getSupportFragmentManager得到FragmentManager；使用beginTransaction()方法得到FragmentTransaction；使用FragmentTraction.add(容器id,fragment对象)/replace(容器id,fragment对象)等方法操作fragment；操作完成后要使用FragmentTraction.connmt()方法提交。

调用FragmentTraction.addToBackStack(String)方法可以使fragment支持后退键；调用FragmentManager.popStack()方法可返回上一层。

**4.1.2Fragment的生命周期**

详见Fragment类附图。主要有onCreate(...),onCreaterView(...);onDestroy(...)。

**4.1.3带侧边栏的Activity**

Activity的侧边栏实际是一个Fragment，点击侧边栏显示的也是另外的Fragment。

在侧边栏的Fragment中有回调接口，在Activity实现此接口，执行打开其他Fragment等操作。

**4.1.4Tabbed Activity**

TabbedActivity实际是viewpager控件添加了FragmentPagerAdapter，tabbedActivity中各页面实际是fragment。

自动创建时：在继承FragmentPagerAdapter的类中的getItem(tab序号)方法中执行打开fragment等操作。

**4.2Android基本布局**

**4.2.2用代码控制子对象**

可以在代码中用LinearLayout类、Button类等类及其方法动态控制布局。

在LinearLayout（也可能是RelativeLayout等其他布局）中定义有LinearLayout的属性，LinearLayout.LayoutParams中定义有其他控件属性。可以先生成LinearLayout.LayoutParema的对象，再在添加控件时使用此对象。

**4.3Android RecyclerView**

**4.3.1使用RecyclerView**

定义RecyclerView对象，使用：RecyclerView.setLayoutManager(LayoutManager)添加布局，参数可使用LinearLayoutManager对象;RecyclerView.setAdapter(Adapter)设置显示的view。

Adapter对象中，添加ViewHolder的继承类，并分别实现：onCreateViewHolder(....)方法，创建ViewHolder继承类对象；onBindViewHolder(...)方法，对不同序号的对象分别进行修改；getItemCount()，其返回值是创建ViewHolder的总数。

**4.3.2使用资源文件自定义列表项**

Adapter继承类中，onBindViewHold(ViewGroup,viewType)可以返回自定义的视图。方法为：使用LayoutInflater.from(Context)方法获得LayoutInflater对象；再使用此对象的inflate(id,...)方法解析视图作为ViewHolder构造方法的参数。其中Context使用ViewGroup.getContext方法获取。

**4.3.3RecyclerView的布局样式**

RecyclerView.setLayoutManager(LayoutManager)的参数，可以使用LinearLayoutManager，也可以使用GridLayoutManager。

**4.4Android常用控件**

**4.4.1下拉列表**

Spinner控件。使用Spinner.setAdapter(SpinnerAdapter/ArrayAdapter)设置列表中的项。使用Spinner.setOnItemSelectedListener(...)设置事件监听器。

**4.4.2日期选择器**

DataPickerDialog控件。使用DataPickerDialog(context,DatePickerDialog.OnDateSetListener,三个日期整型)构造方法，其中第二个参数是事件监听接口，在其中实现onDataSet方法。使用DataPickerDialog.show()方法显示对话框。

**4.4.3时间选择器**

TimePickerDialog控件。使用TimePickerDialog(context,TimePickerDialog.OnTimeSetListener,两个时间整型，是否二十四时制)构造方法，其中第二个参数是事件监听接口，在其中实现onTimeSet方法。使用DataPickerDialog.show()方法显示对话框。

**4.4.4单项选择**

在RadioGroup中放置RadioButton。使用RadioButton.isChecked()方法判断是否选中。

**4.4.5多项选择**

CheckBox控件。

**4.5Android用户界面优化AndroidSlidingMenu菜单栏程序设计开发**

**4.5.4SlidingMenu程序配置**

在eclipse中直接导入项目即可。

在android studio中，更改setting.gradle；将两个库添加为主Module的依赖项；更改两个库的依赖项。暂未使用成功。

**4.5.5SlidingMenu程序开发**

使用SlidingMenu类及其方法即可。

**4.12Android侧滑菜单之DrawerLayout的使用**

**4.12.2创建抽屉布据**

主activity的布局设为android.support.v4.widget.DrawerLyaout，其中包含两个视图，第一个是内容视图，第二个是抽屉视图。主内容视图宽度要match\_parent；抽屉视图宽度不要铺满屏幕。使用layout\_gravity属性设置滑动方向，start为从左向右。

**4.12.3初始化导航列表**

DrawerLayout中的第二个视图ListView，直接调用其setAdapter(adapter)方法设置adapter即可，通常是ArrayAdapter或SiroleCurrsoAdapter。

调用ListView.setOnItemClickListener(AdapterView.onClickListener)方法设置监听点击事件。

DrawerLayout.closeDrawer(view)方法可以关闭抽屉，通常在点击后调用。

**4.12.4监听抽屉的打开关闭事件**

使用DrawerLayout.setDrawerListener(ActionBarDraeerToggle)方法设定抽屉打开关闭监听。其中ActionBarDrawerToggle类继承了DrawerLayout.DrawerListener接口，在其中重写onDrawerOpened(view)和onDrawerClosed(view)方法，在两个方法中可以进行对菜单栏的操作，例如更改标题等。

ActionBarDrawerToggle在supportv4和supportv7中是不同的，对应对菜单栏的操作方法也是不同的。

**4.12.5点击图标开闭抽屉**

使用ActionBar的：setDisPlayHomeAsUpEnable (boolean)方法使菜单home键可见；setHomeButtonEnabled (true)方法使菜单home键可用。在Activity的onOptionsItemItemSelected (menuitem)方法中调用ActionBarDrawerToggle .onOptionsItemSelected(item)方法在点击home键时跳过其他项的处理。

google建议重写Activity的onPostCreate(...)方法和onConfigurationChanged(...)方法，并在其中分别调用ActionBarDrawerToogle的syncState()方法和onConfigurationChanged(...)方法。

**4.13Android中ActionBar的使用**

**4.13.2启用ActionBar**

在Android3.0以上版本时，使用Theme.Holo主题。

在Android3.0以下版本时，要依赖appcompat-v7包，使用Theme.AppCompat.Light.DarkActionBar主题，并使Activity继承ActionBarActivity类。

**4.13.3在ActionBar上添加按钮**

在menu的xml文件中添加item标签即可添加菜单按钮。重写Activity中：onCreateOptionsMenu(menu)方法创建menu；onOptionsItemSelected(menuitem)监听点击菜单项事件。

菜单左上角home键使用方法见4.12.5。其系统提供标识符为R.id.home。

**4.13.4自定义ActionBar样式**

**4.13.5自动隐藏ActionBar**

**4.13.6ActionProvider的使用**

**4.13.7ActionBarSherlock的使用**

**6.1Android文件读写操作**

**6.1.2读取Assets中的文件数据**

Assets位于src/main目录。Assets用于存放资源文件。使用getResources()方法得到Resources对象；使用其 getAssets()方法得到AssetsManager对象；使用其open(文件名)方法打开文件。

**6.1.3读取raw目录中的文件数据**

raw位于src/main/res目录。raw文件夹下资源会自动生成资源标识符。使用Resource.openRawResource(资源标识符)打开raw文件。

**6.1.4读写内部存储的文件数据**

读写系统data/data/包名/files文件夹下的文件。使用openFileOutput(文件名，模式)打开读文件；使用openFileInput(文件名)打开写文件。

可使用Context.getFilesDir()得到File类。

**6.1.5读取外部存储的文件数据**

android4.0以后不再允许读取外部文件；android4.0之前要添加WRITE\_EXTERNAL\_ STORAGE权限和READ\_EXTERNAL\_STORAGE权限。

直接使用路径创建File类即可。使用Environment.getExernalStorageDirectory()方法可得到storage/sdcard路径

对于android4.0以后，可以使用Context.getExternalFilesDir(...)与Context.getCacheDir(...)方法来获取系统分配外部文件，此用法与内部文件相近。

**6.2Android读写首选项**

**6.2.1SharedPreference简介**

SharedPreference读写的是data/data/包名/shared\_prefs文件夹下的xml格式文件，每个数据存储为一对键值对。

**6.2.2SharedPreference数据的读写操作**

使用getPreferences()方法得到activity对应的SharedPreferences对象；使用SharedPreference的一系列get方法可读取数据。

使用SharedPreferences.edit()方法得到SharedPreferences.Editor对象，再通过该对象的一系列put方法写入键值。

**6.2.3PreferenceActivity使用方法**

创建的activity类要继承PreferenceActivity类，并使用addPreferencesFromResource(xmlid)方法将preference XML资源添加到preference层级。使用getPreferenceManager()方法得到PreferenceManager对象，使用此对象的findPreference(key)方法得到各Preference对象，使用这些对象进行操作。

主要的Preference有CheckBoxPreference、ListPreference、EditTextPreference。分别方法有：isCHecked()；getEntry()、getValue()；getText()。

**6.3使用SQLite数据库存储数据**

**6.3.1SQLite数据库的读取和写入**

需要有一个类继承SQLiteOpenHelper类，重写这个类的构造方法和onCreate(...) ，onUpgrade(...)方法。并通过这个类的getWritableDatabase()和getReadableDatabase()方法得到SQLiteDatabase类对象对数据库进行读写。

SQLiteDatabase有insert(...)、query(...)等方法。insert方法传入的是ContentValue类对象，query返回的是Cursor类对象。

**6.3.2通过界面操作数据库**

使用CursorAdapter/SimpleCursorAdapter可以将cursor中的数据直接传递给listview中的一项来显示，使用这两个类数据库中必须有\_id这一列。使用其changerCursor(cursor)方法可改变cursor；使用getCursor()方法可得到cursor。

listview可以直接添加事件监听器。使用AlertDialog.Builder来可以设置并弹出对话框。

**7.1异步任务处理**

**7.1.2AsyncTask的使用方法**

创建AsyncTask类对象，重写其:doInBackground(params)，来在后台操作，其参数是execute(pragma)中传入的值;重写onPostExecute(result),onPreExecute(),onProgressUpdate(value)方法，分别在后台操作结束、开始和publishProgress(value)方法执行时与UI线程交互。要调用AsynTask.execute(pragma)方法使后台操作开始执行。

**7.2基于HTTP的通信技术**

**7.2.1使用Http的Get方式读取网络数据**

创建URL对象。使用URL.openConnection()方法得到URLConnection类对象。URLConnect类中的一系列方法可得到字节流。

**7.2.2使用Http的Post方式与网络交互通信**

使用URLHttpURLConnection类对象进行操作。使用此类的setRequestMethod(...)方法可设置读取http资源的方式。其他同URLConnection。

在写入之前，应使用setDoOutput(bool)方法设置写权限。写入格式一般为：键=值&。

**7.2.3使用HttpClient进行Get方式通信**

创建一个HttpClient类对象，其可初始化为DefaultHttpClient类对象。创建HttpGet类对象，构造方法为HttpGet(url);使用HttpClient.execute(HttpGet)方法得到HttpResponse类对象；使用HttpResponse.getEntity()方法可得到HttpEntity类对象；使用EntityUtils.toString(Entity)方法将其转化为字符串。

**7.2.4使用HttpClient进行Post方式通信**

使用HttpPost类对象，其构造方法为HttpPost(url)；使用HttpPost.setEntity(Entity)方法设置Post内容，通常为使用UrlEncodedFormEntity(list)方法转化的list<BasicNameValuePair>键值对。

其他同Get方式。

**7.5Google开源库-Volley的应用**

**7.5.2使用Volley实现JSON字符串请求**

创建RequestQueue类对象，可以使用Volley.newRequestQueue(Context)方法得到。使用RequestQueue.add(JsonObjectRequest)方法提交请求。

JsonObjectRequest类的构造方法：第一个参数是请求方式；第二个参数是URL；第三个参数是POST时的JSONObject；第四五个参数分别是请求成功和失败时的事件监听，成功的事件监听中参数为返回的JSONObject值。

**7.5.3使用Volley实现异步加载图片**

创建ImageLoader类对象，其构造方法第一个参数为RequestQueue类对象，见7.5.2；第二个参数为ImageLoader.ImageCache类对象。在ImageCache类对象创建时，之前要有一个LruCache类对象，重写ImageCache的getBitmap(...)和putBitmap(...)方法，并在两个方法中操作LruCache类对象。

使用ImageLoader.get(url,listener)方法提交指令，其中第一个参数为图片url；第二个参数为获取图片的事件监听，可使用ImageLoader.getImageListener(imageview,resource,resource)方法获取。

**7.5.4使用NetWorkImageView完成图片加载**

在xml使用com.android.volley.toolbox.NetworkImageView标签。前半部分操作与7.5.3第一段相同。使用NetworkImageView.setImageUrl(url,ImageLoader)方法将NetworkImageView设置显示指定url的图片。