Content Providers

Data Persistence

Preferences: key/value pairs

SQLite: Use a very small RDBMS that is sandboxed to your app and then expose it to other apps using a content provider

Files: You can read/write files to your raw directory

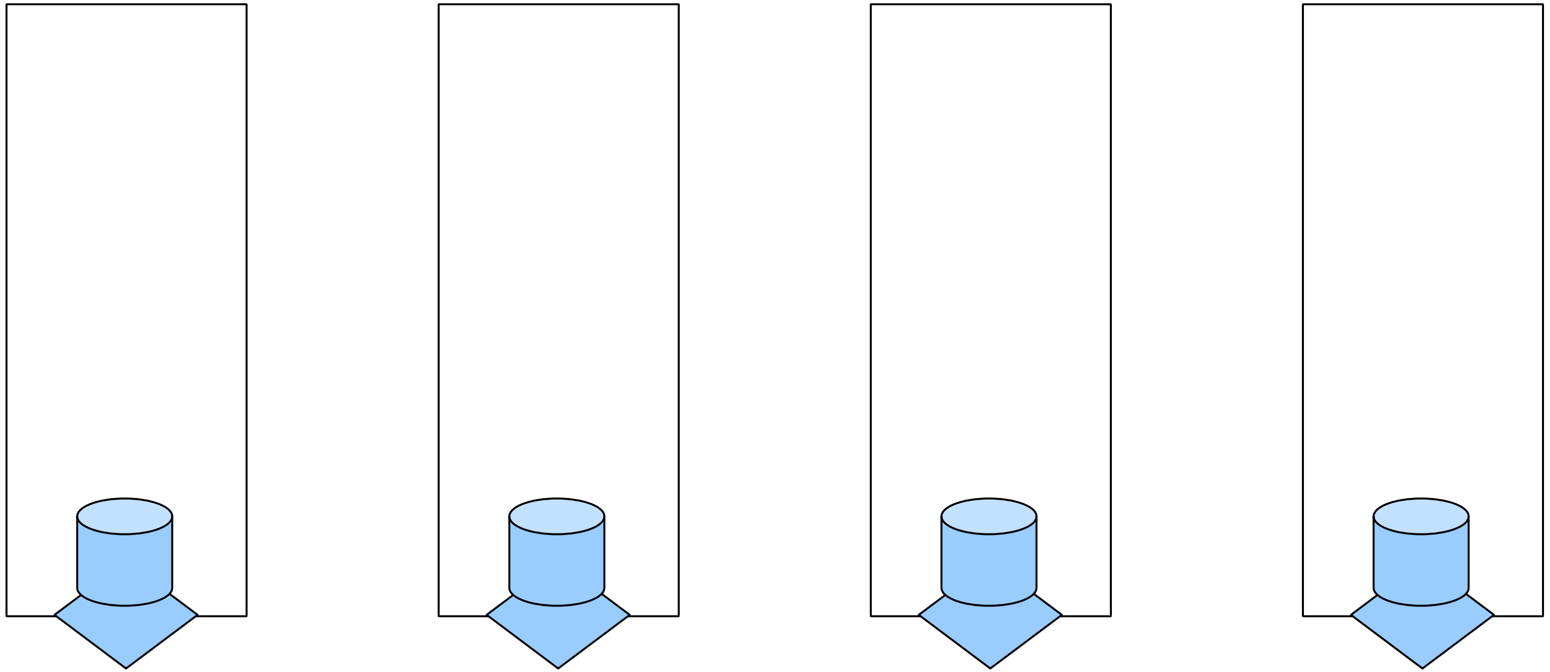
Network: store data on the internet and read/write using any available Internet protocol

What is a Content Provider?

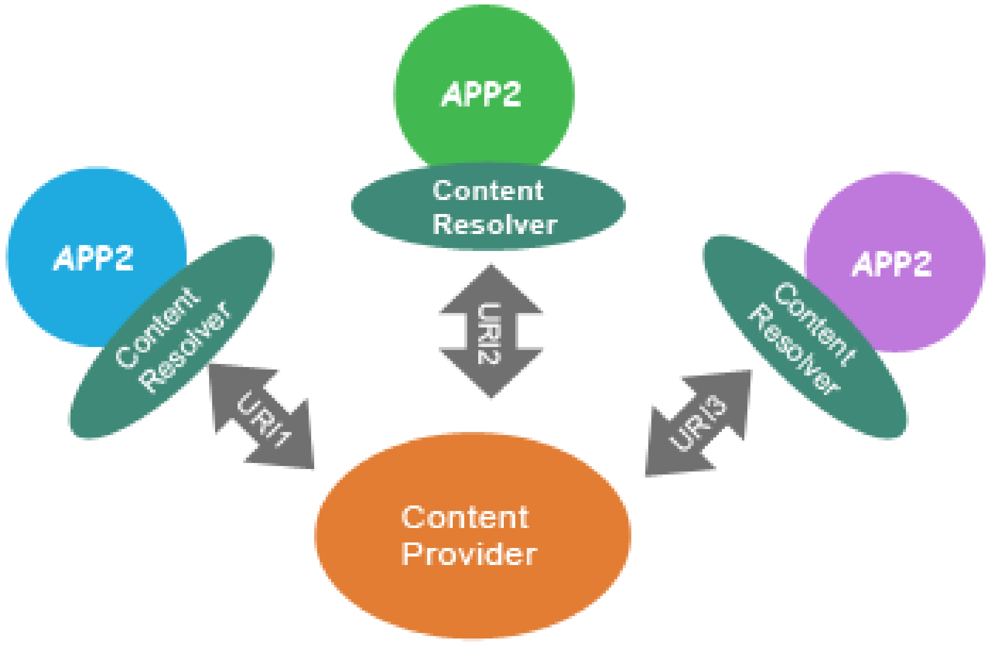
- Mechanism for supplying data to third-party apps
- Data abstraction layer - the data storage implementation behind the content provider **doesn't matter**!

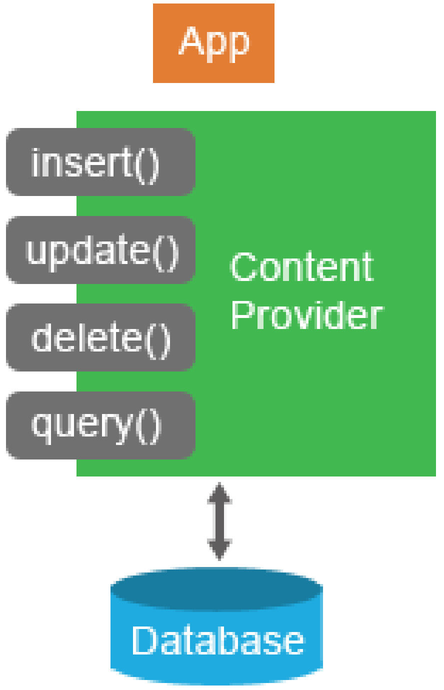
Why Content Providers?

Each application contains databases that can not be accessed outside their application context. To get or expose the data in these DBs, you must use a Content Provider



Content Providers





Content Provider Key Operations

CRUD:

- Creation
- Read
- Update
- Deletion

Content Provider Usage

- Any Uri that begins with content:// represents a resource backed by a content provider.

Anatomy of a Uri

- scheme - always "content"
- authority - name of entire provider
- data type path (optional)
- instance identifier (optional)

$$\underbrace{\texttt{content}}_{\text{scheme}}\texttt{://}\underbrace{\texttt{contacts}}_{\text{authority}}\texttt{/}\underbrace{\texttt{people}}_{\text{data type path}}\texttt{/}\underbrace{\texttt{5}}_{\text{instance identifier}}$$

Uri Class

Convert String to Uri via Uri.parse()

Example:

```
Uri.parse("content://contacts/people");
```

System Content Providers

- Browser - Read or modify bookmarks, browser history, or web searches.
- CallLog - View or update the call history
- Contacts - Retrieve, modify, or store personal contacts.
- MediaStore - Access audio, video, and images
- Settings - View and retrieve Bluetooth settings, ringtones, and other device preferences.

Built-in Content Providers

Contacts

MediaStore.Audio

MediaStore.Images

MediaStore.Video

Browser

CallLog

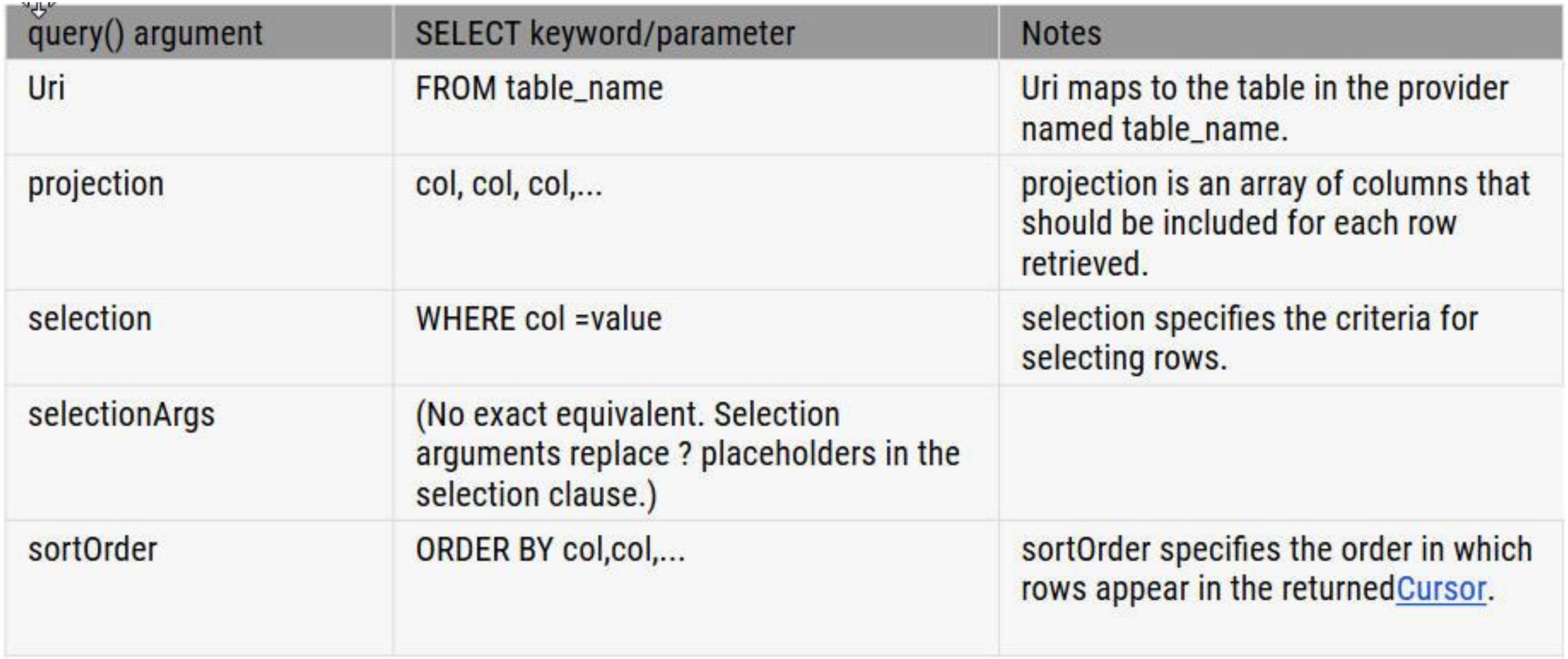
Settings

ContentResolver

Use the ContentResolver class to perform CRUD operations on a ContentProvider:

- Create: `insert(uri, contentValues)`
- Read: `query(uri, projection, selection, selectionArgs, sortOrder)`
- Update: `update(uri, contentValues, where, selectionArgs)`
- Delete: `delete(uri, where, selectionArgs)`

Query Parameters



query() argument	SELECT keyword/parameter	Notes
Uri	FROM table_name	Uri maps to the table in the provider named table_name.
projection	col, col, col,...	projection is an array of columns that should be included for each row retrieved.
selection	WHERE col =value	selection specifies the criteria for selecting rows.
selectionArgs	(No exact equivalent. Selection arguments replace ? placeholders in the selection clause.)	
sortOrder	ORDER BY col,col,...	sortOrder specifies the order in which rows appear in the returnedCursor.

ContentResolver - Query Example

```
// Example: Print all names and phone numbers in the contacts content provider,
// Content provider is content://com.android.contacts/data/phones
// Remember to add the following permission:
// <uses-permission android:name="android.permission.READ_CONTACTS" />

ContentResolver cr = getContentResolver();
Cursor cursor = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
        null, null, null, null);
int displayNameColumn = cursor.getColumnIndex(
        ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME);
int phoneNumberColumn = cursor.getColumnIndex(
        ContactsContract.CommonDataKinds.Phone.NUMBER);
while (cursor.moveToNext()) {
    Log.d("TAG", cursor.getString(displayNameColumn) + ": " +
            cursor.getString(phoneNumberColumn));
}
cursor.close();
```

ContentResolver - Insert Example

```
// Prank example: Insert a call log entry for a psychic hotline.
// Remember to add the following permissions:
// <uses-permission android:name="android.permission.WRITE_CALL_LOG" />
// <uses-permission android:name="android.permission.READ_CALL_LOG" />
ContentValues values = new ContentValues();
values.put(CallLog.Calls.NUMBER, "1-900-PSYCHIC");
values.put(CallLog.Calls.DATE, System.currentTimeMillis());
values.put(CallLog.Calls.TYPE, CallLog.Calls.OUTGOING_TYPE);
values.put(CallLog.Calls.NEW, 1);
Log.d("TAG", "Inserting call log");
getContentResolver().insert(CallLog.Calls.CONTENT_URI, values);
```

Creating a Content Provider

Why create a content provider?

Only create a content provider if:

- You want to offer complex data or files to other applications.
- You want to allow users to copy complex data from your app into other apps.
- You want to provide custom search suggestions using the search framework.

Data Storage Options

- SQLite database
- File-system APIs
- Network-based storage

Creating a Content Provider

Steps:

1. Create class that extends ContentProvider
2. In AndroidManifest.xml, add provider entry

```
<provider android:name="ZooContentProvider"
          android:authorities="zoo" >
</provider>
```

Using the Custom Provider

```
// Create
getContentResolver().insert(Uri.parse("content://zoo/animals"), new ContentValues());

// Read
getContentResolver().query(Uri.parse("content://zoo/animals"), null, null, null, null);

// Update
getContentResolver().update(Uri.parse("content://zoo/animals/5"), new ContentValues(),
null, null);

// Delete
getContentResolver().delete(Uri.parse("content://zoo/animals"), null, null);
```

Security

```
<!-- Declare the permission -->
<permission android:name="myapp.permission.ACCESS_ZOO"
    android:label="Zoo Access"
    android:description="Provide access to information about animals in zoo"
    android:protectionLevel="normal" />

<!-- Protect the content provider with the permission -->
<provider android:name=".MyProvider"
          android:authorities="zoo"
          android:permission="myapp.permission.ACCESS_ZOO"/>
```

Limitations

- No onDestroy() companion to onCreate()
  - If you open a database, you can't close it.
- Interface may be too simple for rich backend
  - e.g., GROUP BY not supported in interface
- Public by default
  - Can be accessed by other third-party processes
  - To make private:
    - Use permissions or set android:exported="false"
    - Set targetSdkVersion to 17 or higher(API Level 17 changes default behavior)

Loaders

•Loaders are used for asynchronous loading of data for an Activity. Application should call Loader API from the main thread.
•The Loader (or subclasses of Loader) executes their functionality in a separate thread and delivers the results to the main thread.
•Loaders provide listener interface that any Activity can implement to receive callbacks from Loader.
•Loaders use AsyncTask for their internal operation

CursorLoaders

- CursorLoader queries the Content Resolver in the background thread to avoid and returns the loaded cursor in callback to the Activity or Fragment.
- CursorLoader handles the life cycle of the cursor. Developer should never call close() on the returned cursor.
- CursorLoader automatically deliver updated cursor to the Activity or Fragment in callback when the content of the content provider changes. In other words, Loader monitor the source of its data and there is no need to query for updated data again.

CursorLoaders

2 benefits of CursorLoaders:

The query is handled on a background thread for you (courtesy of being build on AsyncTaskLoader) so large data queries do not block the UI. This is something the docs recommended you do for yourself when using a plain Cursor, but now it's done under the hood.

CursorLoader is auto-updating. In addition to performing the initial query, the CursorLoader registers a ContentObserver with the dataset you requested and calls forceLoad() on itself when the data set changes. This results in you getting async callbacks anytime the data changes in order to update the view.

Option 1: Accessing data with a Content Provider using CursorLoader

```
CursorLoader cursor Loader =
        new CursorLoader(
                Context context, Uri uri, String[ ] projection,String selection,
                String[ ] selectionArgs, String sortOrder)
```

***context* =** associated context

***uri* =** Content Provider URI

projection =which columns to return

selection = SQL Where clause with "WHERE"

selectionArgs =Arguments for selection

sortOrder = SQL ORDER BY clause

Option1: Example Accessing Content Provider data with CursorLoader

```
import android.content.CursorLoader;
// INSIDE Activity Class ********
@Override
public void onCreate(Bundle savedInstances) {
    //*** other code*****
    Uri  allContacts = Uri.parse(“content://contacts/people”);
    CursorLoader  cursorLoader =  new CursorLoader( this,
                                        allContacts,  //URI of content provider
                                            null,    //means return all columns
                                            null,       // WHERE clause-- won't specify.
                                            null,     // no where clause, so no arguments
                                            null);  //no order by specified
    //get data from Content Provider, populates Cursor c with result set
   Cursor c = cursorLoader.loadInBackground();   //LOADS in background, no blocking
```

Option1: Example Accessing Content Provider data with CursorLoader

This example:

display results using a SimpleCursorAdapter

```
import android.widget.SimpleCursorAdapter;
import android.database.Cursor;
import android.widget.CursorAdapter;
import android.provider.ContactsContract;  //built in contacts content provi


// INSIDE Activity Class ********
@Override
public void onCreate(Bundle savedInstances) {
    //*** other code SEE PREVIOUS SLIDE***** results in c (Cursor instance)
   //info will display from ContentProvider results in Cursor c
   String[]  columns = new String[] {ContactsContract.Contacts.DISPLAY_NAME,
                                      ContactsContract.Contacts._ID};
   int[]  views = new int[] {R.id.contactName,   R.id.contactID};
   adapter = new SimpleCursorAdapter(this, R.layout.main, c, columns, views
                          CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
```

CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER Means this adapter registered to be informed when change in content provider

Option1: Example-SimpleCursorAdapter

An Adapter used to represent Cursor (data result set)

Used to populate a related View → example …. android.app.ListActivity as one possibility

```
ListActivityInstance.setListAdapter(SimpleCursorAdapter_Instance)
```

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

context = context where the ListView associated with this

layout = resource identifier of a layout file that defines the views for this list item. The layout file should include at least those named views defined in "to"

c = database cursor.

from =list of column names representing the data to bind to the UI. Can be null if the cursor is not available yet.

to = views that should display column in the "from" parameter. These should all be TextViews. The first N views in this list are given the values of the first N columns in the from parameter.

flags = Flags used to determine the behavior of the adapter, as per CursorAdapter(Context, Cursor, int).

Example

Main.xml ---interface for app’s main Activity (a ListActivity)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

<ListView
    android:id="@+id/android:list"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:stackFromBottom="false"
    android:transcriptMode="normal" />

<TextView
    android:id="@+id/contactName"
    android:textStyle="bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<TextView
    android:id="@+id/contactID"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

Example --- main Activity Class

```
public class ProviderActivity extends ListActivity {
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
       Uri allContacts = ContactsContract.Contacts.CONTENT_URI;
        String[] projection = new String[]
                {ContactsContract.Contacts._ID,
                 ContactsContract.Contacts.DISPLAY_NAME,
                 ContactsContract.Contacts.HAS_PHONE_NUMBER};


        Cursor c;
        CursorLoader cursorLoader = new CursorLoader( this,
            allContacts,   projection,
                    ContactsContract.Contacts.DISPLAY_NAME + "
LIKE ?“,  new String[] {"%Lee"},
                    ContactsContract.Contacts.DISPLAY_NAME + "
ASC");
```

```
 c = cursorLoader.loadInBackground();
 String[] columns = new String[] {
    ContactsContract.Contacts.DISPLAY_NAME,
    ContactsContract.Contacts._ID};


int[] views = new int[] {R.id.contactName, R.id.contactID};


 SimpleCursorAdapter adapter;
adapter = new SimpleCursorAdapter( this, R.layout.main,
                c, columns, views,
 CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);


 this.setListAdapter(adapter);


 PrintContacts(c);
}
```

Example --- main Activity Class

```
private void PrintContacts(Cursor c)
    {
        if (c.moveToFirst()) {
            do{
            String contactID = c.getString(c.getColumnIndex(
                   ContactsContract.Contacts._ID));
            String contactDisplayName =
                  c.getString(c.getColumnIndex(
                     ContactsContract.Contacts.DISPLAY_NAME));
                Log.v("Content Providers", contactID + ", " +
                   contactDisplayName);
            } while (c.moveToNext());
        }
    }

}
```

<<< UTILITY Method to print out the result set returned in the Cursor instance c that contains all the Contacts.

Example ---some explanation

Predefined Query String Constants

Uri allContacts = ContactsContract.Contacts.*CONTENT_URI*

SAME AS

Uri allContacts = Uri.parse(“content://contacts/people”);

Example ---some explanation

Following is like saying give me all the contacts with the columns ID, DISPLAY_NAME and HAS_PHONE_NUMBER where the DISPLAY_NAME is Like Lee and orderby DISPLAY_NAME in Ascending order (ASC)

```
String[] projection = new String[]
            {ContactsContract.Contacts._ID,
             ContactsContract.Contacts.DISPLAY_NAME,
             ContactsContract.Contacts.HAS_PHONE_NUMBER};


    Cursor c;
    CursorLoader cursorLoader = new CursorLoader( this,
      allContacts,   projection,
          ContactsContract.Contacts.DISPLAY_NAME + " LIKE ?“,  new String[] {"%Lee"},
          ContactsContract.Contacts.DISPLAY_NAME + " ASC");
```

Predefined Query Strings

Predefined Query String Constants

Uri allContacts = ContactsContract.Contacts.*CONTENT_URI*

SAME AS

Uri allContacts = Uri.parse(“content://contacts/people”);

Other examples

Browser.BOOKMARKS_URI

Browser.SEARCHES_URI

CallLog.CONTENT_URI

MediaStore.Images.Media.INTERNAL_CONTENT_URI

MediaStore.Images.Media.EXPERNAL_CONTENT_URI

Settings.CONTENT_URI

Questions?