CMPS 121
Content Providers

Luca de Alfaro

Portions of this presentations are based on content taken from the Android Developers Site. The work was created and shared by the Android Open Source Project and used according to terms described in the Creative Commons 2.5 Attribution License.
Content providers

Content providers are used to make data (typically, database data, but also file data) accessible to multiple applications via a well-defined API. Manages permissions as well (usually, declared in the manifest).

Example: a dictionary table

<table>
<thead>
<tr>
<th>word</th>
<th>app id</th>
<th>frequency</th>
<th>locale</th>
<th>_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapreduce</td>
<td>user1</td>
<td>100</td>
<td>en_US</td>
<td>1</td>
</tr>
<tr>
<td>precompiler</td>
<td>user14</td>
<td>200</td>
<td>fr_FR</td>
<td>2</td>
</tr>
<tr>
<td>applet</td>
<td>user2</td>
<td>225</td>
<td>fr_CA</td>
<td>3</td>
</tr>
<tr>
<td>const</td>
<td>user1</td>
<td>255</td>
<td>pt_BR</td>
<td>4</td>
</tr>
<tr>
<td>int</td>
<td>user5</td>
<td>100</td>
<td>en_UK</td>
<td>5</td>
</tr>
</tbody>
</table>
Accessing a content provider

Accessing a provider is done via a `ContentResolver` object, which has methods paralleling a `ContentProvider` object on the provider's side.

Example: the `query()` method:

```java
// Queries the user dictionary and returns results
mCursor = getContentResolver().query(
    UserDictionary.Words.CONTENT_URI,  // The content URI of the words table
    mProjection,  // The columns to return for each row
    mSelectionClause  // Selection criteria
    mSelectionArgs,  // Selection criteria
    mSortOrder);  // The sort order for the returned rows
```

```java
public final Cursor query(Uri uri, String[] projection, String selection,
                          String[] selectionArgs, String sortOrder)
```
Content URIs

Of provider:

```
content://user_dictionary/words
```

You can also create URIs for single items, using an ID:

```
Uri singleUri = ContentUri.withAppendedId(UserDictionary.Words.CONTENT_URI, 4);
```

See the Uri and Uri.Builder classes, as well as ContentUris.
Some methods of ContentResolver

Regular DB methods:

```java
public final Cursor query (Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)

public final Uri insert (Uri url, ContentValues values)

public final int update (Uri uri, ContentValues values, String where, String[] selectionArgs)

public final int delete (Uri url, String where, String[] selectionArgs)
```

Change management:

```java
public void notifyChange (Uri uri, ContentObserver observer)

public final void registerContentObserver (Uri uri, boolean notifyForDescendents, ContentObserver observer)

public final void unregisterContentObserver (ContentObserver observer)

public static Object addStatusChangeListener (int mask, SyncStatusObserver callback)

public static void removeStatusChangeListener (Object handle)
```
// A "projection" defines the columns that will be returned for each row
String[] mProjection =
{
    UserDictionary.Words._ID,       // Contract class constant for the _ID column name
    UserDictionary.Words.WORD,      // Contract class constant for the word column name
    UserDictionary.Words.LOCALE     // Contract class constant for the locale column name
};

// Defines a string to contain the selection clause
String mSelectionClause = null;

// Initializes an array to contain selection arguments
String[] mSelectionArgs = {""};
// Gets a word from the UI
mSearchString = mSearchWord.getText().toString();

// Remember to insert code here to check for invalid or malicious input.

// If the word is the empty string, gets everything
if (TextUtils.isEmpty(mSearchString)) {
    // Setting the selection clause to null will return all words
    mSelectionClause = null;
    mSelectionArgs[0] = "";
}
else {
    // Constructs a selection clause that matches the word that the user entered.
    mSelectionClause = " = ?";

    // Moves the user's input string to the selection arguments.
    mSelectionArgs[0] = mSearchString;
}
Querying - example (cont.)

// Does a query against the table and returns a Cursor object
mCursor = getContentResolver().query(
    UserDictionary.Words.CONTENT_URI,  // The content URI of the words table
    mProjection,  // The columns to return for each row
    mSelectionClause  // Either null, or the word the user entered
    mSelectionArgs,  // Either empty, or the string the user entered
    mSortOrder);  // The sort order for the returned rows

// Some providers return null if an error occurs, others throw an exception
if (null == mCursor) {
    // Error handling code.
}

// If the Cursor is empty, the provider found no matches
} else if (mCursor.getCount() < 1) {
    // Notify the user of no results.
}

} else {
    // Insert code here to do something with the results
}
Displaying query results

// Defines a list of columns to retrieve from the Cursor and load into an output row
String[] mWordListColumns =
{
    UserDictionary.Words.WORD,  // Contract class constant containing the word column name
    UserDictionary.Words.LOCALE  // Contract class constant containing the locale column name
};

// Defines a list of View IDs that will receive the Cursor columns for each row
int[] mWordListItems = { R.id.dictWord, R.id.locale};

// Creates a new SimpleCursorAdapter
mCursorAdapter = new SimpleCursorAdapter(
    getApplicationContext(),  // The application's Context object
    R.layout.wordlistrow,     // A layout in XML for one row in the ListView
    mCursor,                 // The result from the query
    mWordListColumns,        // A string array of column names in the cursor
    mWordListItems,          // An integer array of view IDs in the row layout
    0);                      // Flags (usually none are needed)

// Sets the adapter for the ListView
mWordList.setAdapter(mCursorAdapter);
Getting data from query results

The query returns a **Cursor** object, so you can use the same methods you can use for databases:

```java
// Determine the column index of the column named "word"
int index = mCursor.getColumnIndex(UserDictionary.Words.WORD);

// Only executes if the cursor is valid.
if (mCursor != null) {
    // Moves to the next row in the cursor. Before the first movement in the cursor, the
    // "row pointer" is -1, and if you try to retrieve data at that position you will get an
    // exception.
    while (mCursor.moveToNext()) {
        // Gets the value from the column.
        newWord = mCursor.getString(index);
        // Insert code here to process the retrieved word.
    }
} else {
    // Insert code here to report an error if the cursor is null or the provider threw an exception.
}
```
Inserting data

// Defines a new Uri object that receives the result of the insertion
Uri mNewUri;

// Defines an object to contain the new values to insert
ContentValues mNewValues = new ContentValues();

// Sets the values of each column and inserts the word. The arguments to the "put"
// method are "column name" and "value"
    mNewValues.put(UserDictionary.Words.APP_ID, "example.user");
    mNewValues.put(UserDictionary.Words.LOCALE, "en_US");
    mNewValues.put(UserDictionary.Words.WORD, "insert");
    mNewValues.put(UserDictionary.Words.FREQUENCY, "100");

mNewUri = getContentResolver().insert(
    UserDictionary.Word.CONTENT_URI, // the user dictionary content URI
    mNewValues // the values to insert
);

The URI returned has the format: content://user_dictionary/words/<id_value>
To get the id, you can use ContentUris.ParseId()
Updating data

// Defines an object to contain the updated values
ContentValues mUpdateValues = new ContentValues();

// Defines selection criteria for the rows you want to update
String mSelectionClause = UserDictionary.Words.LOCALE + "LIKE ?";
String[] mSelectionArgs = {"en_%"};

// Defines a variable to contain the number of updated rows
int mRowsUpdated = 0;

// Sets the updated value and updates the selected words.
mUpdateValues.putNull(UserDictionary.Words.LOCALE);

mRowsUpdated = getContentResolver().update(
    UserDictionary.Words.CONTENT_URI, // the user dictionary content URI
    mUpdateValues, // the columns to update
    mSelectionClause, // the column to select on
    mSelectionArgs // the value to compare to
);
Deleting data

// Defines selection criteria for the rows you want to delete
String mSelectionClause = UserDictionary.Words.APP_ID + " LIKE ?";
String[] mSelectionArgs = {"user"};

// Defines a variable to contain the number of rows deleted
int mRowsDeleted = 0;

// Deletes the words that match the selection criteria
mRowsDeleted = getContentResolver().delete(  
    UserDictionary.Words.CONTENT_URI,  // the user dictionary content URI
    mSelectionClause  // the column to select on
    mSelectionArgs   // the value to compare to
);
Example: calendar provider
Calendar: tables

- **CalendarContract.Calendars**: details of a single calendar.
- **CalendarContract.Events**: holds info for each event. There is a **CALENDAR_ID** field that is a foreign key to the _ID field in the Calendars table.
- **CalendarContract.Instances**: start and end times for an event instance (repeated events have multiple instances; regular events just one). **EVENT_ID** is a foreign key to _ID in Events table.
- **CalendarContract.Attendees**: guest information, has a foreign key **EVENT_ID** to _ID in Events table.
- **CalendarContract.Reminders**: similarly.
Permissions

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
  <uses-sdk android:minSdkVersion="14" />
  <uses-permission android:name="android.permission.READ_CALENDAR" />
  <uses-permission android:name="android.permission.WRITE_CALENDAR" />
  ...
</manifest>
Querying a calendar

// Projection array. Creating indices for this array instead of doing
// dynamic lookups improves performance.
public static final String[] EVENT_PROJECTION = new String[] {
    Calendars._ID, // 0
    Calendars.ACCOUNT_NAME, // 1
    CalendarsCALENDAR_DISPLAY_NAME // 2
};

// The indices for the projection array above.
private static final int PROJECTION_ID_INDEX = 0;
private static final int PROJECTION_ACCOUNT_NAME_INDEX = 1;
private static final int PROJECTION_DISPLAY_NAME_INDEX = 2;

// Run query
Cursor cur = null;
ContentResolver cr = getContentResolver();
Uri uri = Calendars.CONTENT_URI;
String selection = "((" + Calendars.ACCOUNT_NAME + " = ?) AND (" + Calendars.ACCOUNT_TYPE + " = ?))";
String[] selectionArgs = new String[] {"sampleuser@gmail.com", "com.google"};
// Submit the query and get a Cursor object back.
cur = cr.query(uri, EVENT_PROJECTION, selection, selectionArgs, null);
// Querying a calendar (cont.)

// Use the cursor to step through the returned records
while (cur.moveToNext()) {
    long calID = 0;
    String displayName = null;
    String accountName = null;

    // Get the field values
    calID = cur.getLong(PROJECTION_ID_INDEX);
    displayName = cur.getString(PROJECTION_DISPLAY_NAME_INDEX);
    accountName = cur.getString(PROJECTION_ACCOUNT_NAME_INDEX);

    // Do something with the values...

    ...
}

...
Adding events

```java
long calID = 3;
long startMillis = 0;
long endMillis = 0;
Calendar beginTime = Calendar.getInstance();
beginTime.set(2012, 9, 14, 7, 30);
startMillis = beginTime.getTimeInMillis();
Calendar endTime = Calendar.getInstance();
endTime.set(2012, 9, 14, 8, 45);
endMillis = endTime.getTimeInMillis();

ContentResolver cr = getContentResolver();
ContentValues values = new ContentValues();
values.put(Events.DTSTART, startMillis);
values.put(Events.DTEND, endMillis);
values.put(Events.TITLE, "Jazzercise");
values.put(Events.DESCRIPTION, "Group workout");
values.put(Events.CALENDAR_ID, calID);
values.put(Events.EVENT_TIMEZONE, "America/Los_Angeles");
Uri uri = cr.insert(Events.CONTENT_URI, values);

// get the event ID that is the last element in the Uri
long eventId = Long.parseLong(uri.getLastPathSegment());
```
private static final String DEBUG_TAG = "MyActivity";
...
long eventID = 188;
...
ContentResolver cr = getContentResolver();
ContentValues values = new ContentValues();
Uri updateUri = null;
// The new title for the event
values.put(Events.TITLE, "Kickboxing");
myUri = ContentUris.withAppendedId(Events.CONTENT_URI, eventID);
int rows = getContentResolver().update(updateUri, values, null, null);
Log.i(DEBUG_TAG, "Rows updated: " + rows);

This is only an example: the best practice is to use intents to add or update events.
private static final String DEBUG_TAG = "MyActivity";
public static final String[] INSTANCE_PROJECTION = new String[] {
    Instances.EVENT_ID, // 0
    Instances.BEGIN,   // 1
    Instances.TITLE    // 2
};

// The indices for the projection array above.
private static final int PROJECTION_ID_INDEX = 0;
private static final int PROJECTION_BEGIN_INDEX = 1;
private static final int PROJECTION_TITLE_INDEX = 2;
Querying the instance table (cont.)

// Specify the date range you want to search for recurring
// event instances
Calendar beginTime = Calendar.getInstance();
beginTime.set(2011, 9, 23, 8, 0);
long startMillis = beginTime.getTimeInMillis();
Calendar endTime = Calendar.getInstance();
endTime.set(2011, 10, 24, 8, 0);
long endMillis = endTime.getTimeInMillis();
Querying the instance table (cont.)

```java
Cursor cur = null;
ContentResolver cr = getContentResolver();

// The ID of the recurring event whose instances you are searching
// for in the Instances table
String selection = Instances.EVENT_ID + " = ?";
String[] selectionArgs = new String[] {"207"};

// Construct the query with the desired date range.
Uri.Builder builder = Instances.CONTENT_URI.buildUpon();
ContentUris.appendId(builder, startMillis);
ContentUris.appendId(builder, endMillis);

// Submit the query
cur = cr.query(builder.build(),
               INSTANCE_PROJECTION,
               selection,
               selectionArgs,
               null);
```
while (cur.moveToNext()) {
    String title = null;
    long eventID = 0;
    long beginVal = 0;

    // Get the field values
    eventID = cur.getLong(PROJECTION_ID_INDEX);
    beginVal = cur.getLong(PROJECTION_BEGIN_INDEX);
    title = cur.getString(PROJECTION_TITLE_INDEX);

    // Do something with the values.
    Log.i(DEBUG_TAG, "Event: "+ title);
    Calendar calendar = Calendar.getInstance();
    calendar.setTimeInMillis(beginVal);
    DateFormat formatter = new SimpleDateFormat("MM/dd/yyyy");
    Log.i(DEBUG_TAG, "Date: "+ formatter.format(calendar.getTime()));
}
Accessing data via intents

1. Your application sends an intent containing the action `ACTION_PICK` and the "contacts" MIME type `CONTENT_ITEM_TYPE`, using the method `startActivityForResult()`.

2. Because this intent matches the intent filter for the People app's "selection" activity, the activity will come to the foreground.

3. In the selection activity, the user selects a contact to update. When this happens, the selection activity calls `setResult(resultcode, intent)` to set up an intent to give back to your application. The intent contains the content URI of the contact the user selected, and the "extras" flags `FLAG_GRANT_READ_URI_PERMISSION`. These flags grant URI permission to your app to read data for the contact pointed to by the content URI. The selection activity then calls `finish()` to return control to your application.

4. Your activity returns to the foreground, and the system calls your activity's `onActivityResult()` method. This method receives the result intent created by the selection activity in the People app.

5. With the content URI from the result intent, you can read the contact's data from the Contacts Provider, even though you didn't request permanent read access permission to the provider in your manifest. You can then get the contact's birthday information or his or her email address and then send the e-greeting.
Loaders (since Android 3.0)

Loaders are used to load data asynchronously. To use them, we need:

- An [Activity](#) or [Fragment](#).
- An instance of the [LoaderManager](#).
- A [CursorLoader](#) to load data backed by a [ContentProvider](#). Alternatively, you can implement your own subclass of [Loader](#) or [AsyncTaskLoader](#) to load data from some other source.
- An implementation for [LoaderManager.LoaderCallbacks](#). This is where you create new loaders and manage your references to existing loaders.
- A way of displaying the loader's data, such as a [SimpleCursorAdapter](#).
- A data source, such as a [ContentProvider](#), when using a [CursorLoader](#).
// Prepare the loader. Either re-connect with an existing one,  
// or start a new one.
getLoaderManager().initLoader(0, null, this);

- **0**: the unique id of the loader.
- **this**: the current activity

The call will either re-use or create a loader. If a loader is created, this results in a call to **onCreateLoader()**. Callbacks:

- **onCreateLoader()**
- **onLoadFinished()**
- **onLoaderReset()**
Creating a loader

// If non-null, this is the current filter the user has provided.
String mCurFilter;
...

public Loader<Cursor> onCreateLoader(int id, Bundle args) {
    // This is called when a new Loader needs to be created. This
    // sample only has one Loader, so we don't care about the ID.
    // First, pick the base URI to use depending on whether we are
    // currently filtering.
    Uri baseUri;
    if (mCurFilter != null) {
        baseUri = Uri.withAppendedPath(Contacts.CONTENT_FILTER_URI,
                                   Uri.encode(mCurFilter));
    } else {
        baseUri = Contacts.CONTENT_URI;
    }
}
Creating a loader (cont.)

```java
// Now create and return a CursorLoader that will take care of
// creating a Cursor for the data being displayed.
String select = "((" + Contacts.DISPLAY_NAME + " NOTNULL) AND (" +
                   Contacts.HAS_PHONE_NUMBER + "+1) AND (" +
                   Contacts.DISPLAY_NAME + "+ != " )");
return new CursorLoader(getActivity(), baseUri,
                          CONTACTS_SUMMARY_PROJECTION, select, null,
                          Contacts.DISPLAY_NAME + " COLLATE LOCALIZED ASC");
```
A full loader example

```java
public static class CursorLoaderListFragment extends ListFragment implements OnQueryTextListener, LoaderManager.LoaderCallbacks<Cursor> {
    // This is the Adapter being used to display the list's data.
    SimpleCursorAdapter mAdapter;

    // If non-null, this is the current filter the user has provided.
    String mCurFilter;
```
A full loader example (cont.)

```java
@Override
public void onActivityCreated(Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);

    // Give some text to display if there is no data. In a real
    // application this would come from a resource.
    setEmptyText("No phone numbers");

    // We have a menu item to show in action bar.
    setHasOptionsMenu(true);

    // Create an empty adapter we will use to display the loaded data.
    mAdapter = new SimpleCursorAdapter(getActivity(),
        android.R.layout.simple_list_item_2, null,
        new String[] { Contacts.DISPLAY_NAME, ContactsCONTACT_STATUS },
        new int[] { android.R.id.text1, android.R.id.text2 }, 0);
    setListAdapter(mAdapter);

    // Prepare the loader. Either re-connect with an existing one,
    // or start a new one.
    getLoaderManager().initLoader(0, null, this);
}
```
A full loader example (cont.)

```java
@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    // Place an action bar item for searching.
    MenuItem item = menu.add("Search");
    item.setIcon(android.R.drawable.ic_menu_search);
    item setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM);
    SearchView sv = new SearchView(getActivity());
    sv.setOnQueryTextListener(this);
    item.setActionView(sv);
}

public boolean onQueryTextChange(String newText) {
    // Called when the action bar search text has changed. Update
    // the search filter, and restart the loader to do a new query
    // with this filter.
    mCurFilter = !TextUtils.isEmpty(newText) ? newText : null;
    getLoaderManager().restartLoader(0, null, this);
    return true;
}
```
A full loader example (cont.)

```java
@Override public boolean onQueryTextSubmit(String query) {
    // Don't care about this.
    return true;
}

@Override public void onListItemClick(ListView l, View v, int position, long id) {
    // Insert desired behavior here.
    Log.i("FragmentComplexList", "Item clicked: " + id);
}

// These are the Contacts rows that we will retrieve.
static final String[] CONTACTS_SUMMARY_PROJECTION = new String[] {
    Contacts._ID,
    Contacts.DISPLAY_NAME,
    Contacts.CONTACT_STATUS,
    Contacts.CONTACT_PRESENCE,
    Contacts.PHOTO_ID,
    Contacts.LOOKUP_KEY,
};
```
public Loader<Cursor> onCreateLoader(int id, Bundle args) {
    // This is called when a new Loader needs to be created. This
    // sample only has one Loader, so we don't care about the ID.
    // First, pick the base URI to use depending on whether we are
    // currently filtering.
    Uri baseUri;
    if (mCurFilter != null) {
        baseUri = Uri.withAppendedPath(Contacts.CONTENT_FILTER_URI,
                                         Uri.encode(mCurFilter));
    } else {
        baseUri = Contacts.CONTENT_URI;
    }

    // Now create and return a CursorLoader that will take care of
    // creating a Cursor for the data being displayed.
    String select = "((" + Contacts.DISPLAY_NAME + " NOTNULL) AND (" +
                          + Contacts.HAS_PHONE_NUMBER + "=1) AND (" +
                          + Contacts.DISPLAY_NAME + " != ")")";
    return new CursorLoader(getActivity(), baseUri,
                              CONTACTS_SUMMARY_PROJECTION, select, null,
                              Contacts.DISPLAY_NAME + " COLLATE LOCALIZED ASC");
}
A full loader example (cont.)

```java
public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
    // Swap the new cursor in. (The framework will take care of closing the
    // old cursor once we return.)
    mAdapter.swapCursor(data);
}

public void onLoaderReset(Loader<Cursor> loader) {
    // This is called when the last Cursor provided to onLoadFinished()
    // above is about to be closed. We need to make sure we are no
    // longer using it.
    mAdapter.swapCursor(null);
}
```