COMP30510 Mobile Application Development

Content Providers

Dr. Abraham Campbell

University College Dublin

Abey.campbell@ucd.ie

Outline

- Content Provider Basics
- Using Content Provider
- Creating Content Provider
- Calendar example

Content Providers

- Manage access to a structured set of data, encapsulate the data, and provide mechanisms for defining data security
- Is the standard interface that connects data in one process with code running in another
- Is the only practical way for applications to exchange data (except for 3rd party services and external SD Card)

Native Android Content Providers

- Lots of Android services are available to your app as content providers (considering you got correct permissions)
  - Browser
  - CallLog
  - Contacts
  - MediaStore
  - Settings
  - UserDictionary

Content Providers: Basics

- Android is fully responsible for the lifecycle of Content Providers
- Internal implementation of a content provider, *i.e. how it actually stores data is up to its* software developer
  - *Remember, there are four different ways to CRUD data in Android!*
- All content providers implement a common interface for CRUDing data

Content Providers: Basics

- Content Providers allow two types of access:

- SQL-like – using the same methods as SQLite
- File-like – OutputStream and InputStream (preferable instead of quering BLOB)

Using Content Providers

- Make sure you have permissions *<usespermission android:name=”android.permission.READ_ USER_DICTIONARY”>*
- Get Reference to a Content Provider with ContentResolver:

```
ContentResolver cr =getContentResolver();
```

- Send your CRUD query...

Using Content Providers: Query

- Query parameters:
  - Uri (from table)
  - Projection (columns)
  - Selection (criteria)
  - SelectionArgs
  - SortOrder

- SQL Comparison:
  - FROM table_name
  - col,col,col
  - WHERE col=value
  - ORDER BY

Content Providers: URI

- Content URI syntax
  - content://authority/path/id

- URI examples
  - content://constants
  - content://contacts/people
  - content://ie.ucd.info/course/30480

Examples:

- Query

  ```
  Cursor mycursor =
  getContentResolver().query(MyProvider.
  CONTENT_URI, columns, selection, args,sortOrder);
  ```

- Query example: return all rows (select * from)

  ```
  Cursor allRows =
  getContentResolver().query(MyProvider.
  CONTENT_URI, null, null, null, null);
  ```

Content Provider File Access Example

```
Uri uri =
getContentResolver().insert(MyProvider.
                                        CONTENT_URI, newValues);
try
{
    OutputStream outStream =
    getContentResolver().openOutputStream(uri);
    sourceBitmap.compress(Bitmap.
    CompressFormat.JPEG, 50, outStream);
}
catch (FileNotFoundException e) { }
```

Do You Need a Content Provider?

- Content Providers are meant to share your data with other applications, but you can use it within your application as well
- You want to offer complex data or files to other applications or allow users to copy complex data into other apps
- However, you don't need a content provider if all you need is to use SQLite database within your project

Creating a Content Provider

- Design the raw storage:

  Files vs “Table-like” data
- Define the authority string and content URI
- Implement ContentProvider class and its methods
- Add sample data, server synchronisation

Deciding the Raw Storage

- If you need to store binary objects (BLOBs), choose data storage options:
  - Internal file system
  - SD card
  - Network
- If you need to store table-like data, *a la* structured, data, use SQLite DB (or network)

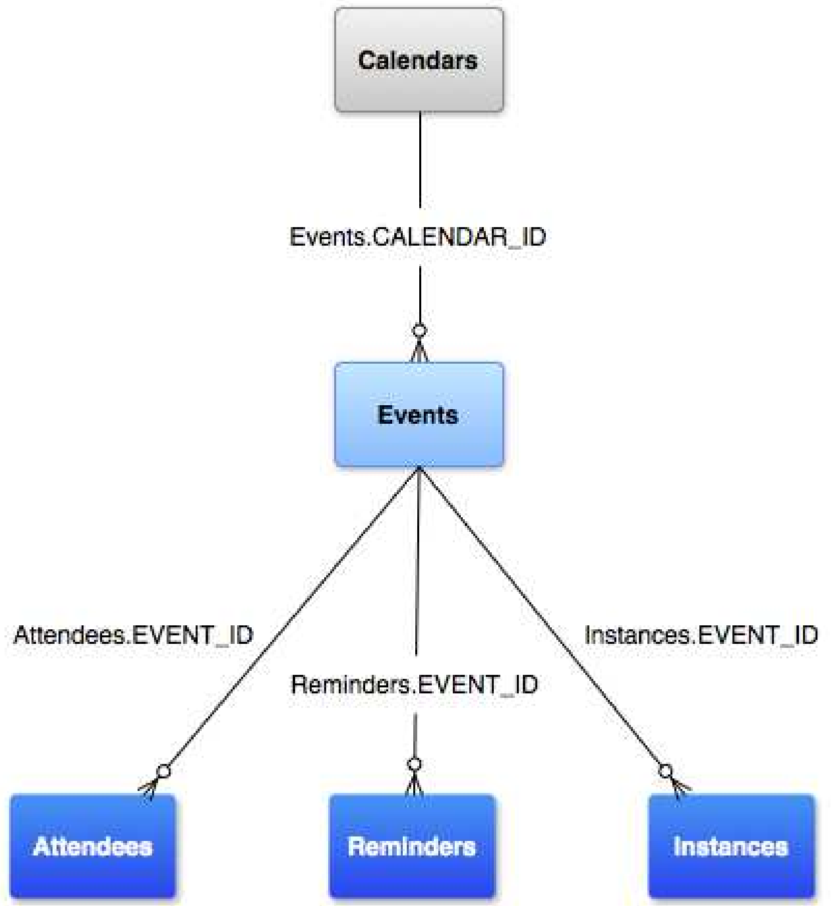
Declaring Content Provider

```
<provider
android:name="ie.ucd.CourseInfoProvider"
android:authorities="ie.ucd.courseinfoprovider" />
...
</provider>
```

Implementing Content Provider

- Extend ContentProvider
- Use **onCreate()** to initialise your storage
- Override **insert()**, **delete()**, **update()**, and **query()** methods
- From Android 3.0 on , you can use a Loader class to help monitor any changes in a content provider.
- A loader class will also keep the last loaders cursor after configurations changes , thus stopping the need for repeating a query after simple changes like a rotation.

Calendar example from Android Dev notes



Calendar API

Table (Class)	Description
CalendarContract.Calendars	This table holds the calendar-specific information. Each row in this table contains the details for a single calendar, such as the name, color, sync information, and so on.
CalendarContract.Events	This table holds the event-specific information. Each row in this table has the information for a single event—for example, event title, location, start time, end time, and so on. The event can occur one-time or can recur multiple times. Attendees, reminders, and extended properties are stored in separate tables. They each have an EVENT_ID that references the _ID in the Events table.
CalendarContract.Instances	This table holds the start and end time for each occurrence of an event. Each row in this table represents a single event occurrence. For one-time events there is a 1:1 mapping of instances to events. For recurring events, multiple rows are automatically generated that correspond to multiple occurrences of that event.
CalendarContract.Attendees	This table holds the event attendee (guest) information. Each row represents a single guest of an event. It specifies the type of guest and the guest's attendance response for the event.
CalendarContract.Reminders	This table holds the alert/notification data. Each row represents a single alert for an event. An event can have multiple reminders. The maximum number of reminders per event is specified in MAX_REMINDERS, which is set by the sync adapter that owns the given calendar. Reminders are specified in minutes before the event and have a method that determines how the user will be alerted.

User Permissions

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"...>
   <uses-sdk android:minSdkVersion="15" />
   <uses-permission android:name="android.permission.READ_CALENDAR" />
   <uses-permission android:name="android.permission.WRITE_CALENDAR" />
   ...
</manifest>
```

Calendars Table

Constant	Description
NAME	The name of the calendar.
CALENDAR_DISPLAY_NAME	The name of this calendar that is displayed to the user.
VISIBLE	A boolean indicating whether the calendar is selected to be displayed. A value of 0 indicates that events associated with this calendar should not be shown. A value of 1 indicates that events associated with this calendar should be shown. This value affects the generation of rows in the CalendarContract.Instances table.
SYNC_EVENTS	A boolean indicating whether the calendar should be synced and have its events stored on the device. A value of 0 says do not sync this calendar or store its events on the device. A value of 1 says sync events for this calendar and store its events on the device.

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
    Calendars.CALENDAR_DISPLAY_NAME,         // 2
    Calendars.OWNER_ACCOUNT                  // 3
};

// 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;
private static final int PROJECTION_OWNER_ACCOUNT_INDEX = 3;
```

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

Querying a Calendar (Cursor Object)

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

    // Do something with the values...

   ...
}
```

Adding data

- You can add data using Intents or directly adding .
- Normally you would use intents

Intents that can be used

Intent Extra	Description
Events.TITLE	Name for the event.
CalendarContract.EXTRA_EVENT_BEGIN_TIME	Event begin time in milliseconds from the epoch.
CalendarContract.EXTRA_EVENT_END_TIME	Event end time in milliseconds from the epoch.
CalendarContract.EXTRA_EVENT_ALL_DAY	A boolean that indicates that an event is all day. Value can be true or false.
Events.EVENT_LOCATION	Location of the event.
Events.DESCRIPTION	Event description.
Intent.EXTRA_EMAIL	Email addresses of those to invite as a comma-separated list.
Events.RRULE	The recurrence rule for the event.
Events.ACCESS_LEVEL	Whether the event is private or public.
Events.AVAILABILITY	If this event counts as busy time or is free time that can be scheduled over.

Intent example

```
Calendar beginTime = Calendar.getInstance();
beginTime.set(2015, 0, 19, 7, 30);
Calendar endTime = Calendar.getInstance();
endTime.set(2015, 0, 19, 8, 30);
Intent intent = new Intent(Intent.ACTION_INSERT)
        .setData(Events.CONTENT_URI)
        .putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME,
beginTime.getTimeInMillis())
        .putExtra(CalendarContract.EXTRA_EVENT_END_TIME,
endTime.getTimeInMillis())
        .putExtra(Events.TITLE, "Yoga")
        .putExtra(Events.DESCRIPTION, "Group class")
        .putExtra(Events.EVENT_LOCATION, "The gym")
        .putExtra(Events.AVAILABILITY, Events.AVAILABILITY_BUSY)
        .putExtra(Intent.EXTRA_EMAIL,
"rowan@example.com,trevor@example.com");
startActivity(intent);
```