http://www.developer.com/http://www.developer.com/ws/android/programming/Working-with-the-Android-Calendar-3850276.htm

Back to article

Working with the Android Calendar

December 14, 2009

Introduction

Android handsets ship with a built-in Calendar application. Third-party applications can leverage the Calendar content provider interface to read user's calendar information and schedule new events in the Calendar. This Calendar directly synchronizes with the user's Google calendar.

Disclaimer

Unfortunately, there is no documentation for integrating with the Calendar application on Android as there is with other applications like the Contacts application. Instead, all information provided in this article was determined via reverse-engineering the Google Calendar content provider. This interface is subject to change and limited features are supported. However, Calendar integration can be a powerful feature for certain types of applications.

The code for this article was tested prior to the Android 2.0 SDK release. Since there are currently no Android 2.0 phones on the market, we cannot yet verify that this code works on the latest devices about to ship. We will post an update if there is a significant change. We did test this code on earlier devices like the T-Mobile G1 (SDK 1.6).

Accessing Calendar Data

In order to add Calendar support to your Android application, you must add the following permissions to your application's `AndroidManifest.xml` file:

```
<uses-permission
  android:name="android.permission.READ_CALENDAR">
  </uses-permission>
<uses-permission
   android:name="android.permission.WRITE_CALENDAR">
   </uses-permission>
```

Registering these permissions allow you to access the user's calendar data without having to deal with any Google login issues. The Calendar application is not installed on the Android emulator so all testing and development must be done with an actual device with the Calendar application installed (using the emulator will fail to launch the appropriate provider). The Calendar application and content provider are available on Android handsets with Google services, such as the T-Mobile G1.

Retrieving a List of the User's Calendars

A user may have any number of named calendars configured within the Calendar application. For example, a user might have a work calendar (for work-related activities), a home calendar (for personal activities), and a holiday calendar (for holidays observed).

The calendars configured by the user are accessed using the content provider interface. In order to retrieve a list of the user's calendars, we need to craft a query for the appropriate Uri for the Calendar content provider, as follows:

```
String[] projection = new String[] { "_id", "name" };
Uri calendars = Uri.parse("content://calendar/calendars");

Cursor managedCursor =
  managedQuery(calendars, projection, null, null, null);
```

Now, this query will return all calendars, including those that are not in active use. In order to get a list of active calendars only, we need to limit our query to only those with the "selected" field set to true:

```
String[] projection = new String[] { "_id", "name" };
Uri calendars = Uri.parse("content://calendar/calendars");

Cursor managedCursor =
   managedQuery(calendars, projection,
   "selected=1", null, null);
```

We have now retrieved a list of calendars. We could iterate through the results as follows:

```
if (managedCursor.moveToFirst()) {
 String calName;
 String calId;
 int nameColumn = managedCursor.getColumnIndex("name");
 int idColumn = managedCursor.getColumnIndex("_id");
 do {
    calName = managedCursor.getString(nameColumn);
```

```
    calId = managedCursor.getString(idColumn);
 } while (managedCursor.moveToNext());
}
```

Once we know which calendar we want to access, we can add a calendar event. Calendar events have a number of important fields, including information such as the event title, time and location as well as settings for how the entry will be displayed in the calendar. The calendar events may be one- time or recurring.

Adding a Single-Occurrence Event to a Calendar

To add an entry to a specific calendar, we need to configure a calendar entry to insert using the `ContentValues` as follows:

```
ContentValues event = new ContentValues();
```

Each event needs to be tied to a specific Calendar, so the first thing you're going to want to set is the identifier of the Calendar to insert this event into:

```
event.put("calendar_id", calId);
```

We then set some of the basic information about the event, including String fields such as the event title, description and location.

```
event.put("title", "Event Title");
event.put("description", "Event Desc");
event.put("eventLocation", "Event Location");
```

There are a number of different options for configuring the time and date of an event.

We can set the event start and end information as follows:

```
long startTime = START_TIME_MS;
long endTime = END_TIME_MS;
event.put("dtstart", startTime);
event.put("dtend", endTime);
```

If we are adding a birthday or holiday, we would set the entry to be an all day event:

```
event.put("allDay", 1);   // 0 for false, 1 for true
```

This information is sufficient for most entries. However, there are a number of other useful calendar entry attributes.

For example, you can set the event status to `tentative (0), confirmed (1) or canceled (2)`:

```
event.put("eventStatus", 1);
```

You can control who can see this event by setting its visibility to default (0), confidential (1), private (2), or public (3):

```
event.put("visibility", 0);
```

You can control whether an event consumes time (can have schedule conflicts) on the calendar by setting its transparency to `opaque (0)` or `transparent (1)`.

```
event.put("transparency", 0);
```

You can control whether an event triggers a reminder alarm as follows:

```
event.put("hasAlarm", 1); // 0 for false, 1 for true
```

Once the calendar event is configured correctly, we're ready to use the `ContentResolver` to insert the new calendar entry into the appropriate Uri for calendar events:

```
Uri eventsUri = Uri.parse("content://calendar/events");
Uri url = getContentResolver().insert(eventsUri, event);
```

The call to the `insert()` method contacts the Calendar content provider and attempts to insert the entry into the appropriate user Calendar. If you navigate to the Calendar application and launch it, you should see your calendar entry in the appropriate Calendar. Since the Calendar syncs, you will also see the Calendar entry online, if you're using the Google Calendar on the web.

Adding a Recurring Event to a Calendar

You can also configure recurring Calendar events. In order to do so, you must add several more fields to the event in the form of a recurrence rule. The rule specification is based upon RFC2445.

Conclusion

Android applications can integrate closely with the user's calendar available with many Android devices. The calendar functionality is accessed via a content provider interface, allowing third-party applications to access calendar information and add new calendar entries.

About the Authors

Shane Conder is a software developer focused on mobile and web technologies. He is currently working at a small mobile software company. With almost two decades of experience in software production, Lauren Darcey specializes in the development of commercial grade mobile applications. Recently, Shane and Lauren coauthored an in-depth programming book entitled Android Wireless Application Development (ISBN: 0321627091), available from Addison-Wesley . They are now working on an entry-level Android book, coming in Spring 2010. They can be reached at androidwirelessdev+a6@gmail.com and via their blog at http://androidbook .blogspot.com