Mobility Programming

Lecture8:
Data storage + Content Providers

DR. RAMEZ ALKHATIB

Content Provider

Content Provider

- A *content provider makes a specific set of the application's data available to other applications*

  => *Share data to other apps*

- Any app with appropriate permission, can read and write the data.
- Many native databases are available via the content providers, for example Contact Manager
- Common interface for querying the data

About Content Provider

- Content providers expose their data as a simple table on a database model

_ID	NUMBER	NUMBER_KEY	LABEL	NAME	TYPE
13	(425) 555 6677	425 555 6677	Kirkland office	Bully Pulpit	TYPE_WORK
44	(212) 555-1234	212 555 1234	NY apartment	Alan Vain	TYPE_HOME
45	(212) 555-6657	212 555 6657	Downtown office	Alan Vain	TYPE_MOBILE
53	201.555.4433	201 555 4433	Love Nest	Rex Cars	TYPE_HOME

- Every record includes a numeric _ID field that uniquely identifies the record within the table.

About Content Provider

- Content provider exposes a public URI that uniquely identifies its data set:

  content://<Authority>/[(n) path]/[instance identifier]

  - the URI starts with content:// scheme.
  - the Authority is a unique identifier for the content provider (fully-qualified class name)
  - the Authority can be followed by one or more paths (optional) refer to data paths within the content.
  - there can be an instance identifier that refers to a specific data instance.
    - content://media/internal/images - return the list of all internal images.
    - content://com.android.contacts/data/phones - return the list of all contact names.
    - content://contacts/people/45 - return the single result row, the contact with ID=45.

Android Native ContentProvider

- Browser—Read or modify bookmarks, browser history, or web searches.
- CallLog—View or update the call history.
- Contacts—Retrieve, modify, or store the personal contacts. three-tier data model of tables under a ContactsContract object:
  - ContactsContract.Data—Contains all kinds of personal data.
  - ContactsContract.RawContacts—Contains a set of Data objects associated with a single account or person.
  - ContactsContract.Contacts—Contains an aggregate of one or more RawContacts, presumably describing the same person.
- MediaStore—Access audio, video, and images.
- Setting—View and retrieve Bluetooth settings, ring tones, and other device preferences.

Android Native ContentProvider

- Android defines CONTENT_URI constants for all the providers that come with the platform.
  - ContactsContract.CommonDataKinds.Phone.CONTENT_URI
    (content://com.android.contacts/data/phones)
  - Browser.BOOKMARKS_URI
    (content://browser/bookmarks)
  - MediaStore.Video.Media.EXTERNAL_CONTENT_URI
    (content://media/external/video/media)

Classes section under :
http://developer.android.com/reference/android/provider/package-summary.html
(if a provider, CONTENT_URI field exist)

Classes

Constants

String	CONTENT_ITEM_TYPE	*This constant is deprecated. see* `ContactsContract`
String	CONTENT_TYPE	*This constant is deprecated. see* `ContactsContract`
String	DEFAULT_SORT_ORDER	*This constant is deprecated. see* `ContactsContract`
String	PERSON_ID	*This constant is deprecated. see* `ContactsContract`

Inherited Constants [Expand]

▸From interface android.provider.BaseColumns

▸From interface android.provider.Contacts.PeopleColumns

▸From interface android.provider.Contacts.PhonesColumns

Fields

public static final Uri	CONTENT_FILTER_URL	*This field is deprecated. see* `ContactsContract`
public static final Uri	CONTENT_URI	*This field is deprecated. see* `ContactsContract`

Public Methods

final static CharSequence	getDisplayLabel (Context context, int type, CharSequence label, C *This method is deprecated. see* `ContactsContract`
final static CharSequence	getDisplayLabel (Context context, int type, CharSequence label) *This method is deprecated. see* `ContactsContract`

Inherited Methods [Expand]

▸From class java.lang.Object

Querying Native Content Provider

- You need
  - URI
    - ContactsContract.Contacts.CONTENT_URI
  - Names of data fields (result comes in table)
    - ContactsContract.Contacts.DISPLAY_NAME
  - Data types of those fields
    - String
- Remember to modify the manifest file for permissions!

To retrieve the contacts:

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

Query

```
import android.provider.Contacts.People;
import android.content.ContentUris;
import android.net.Uri;
import android.database.Cursor;

// Use the ContentUris method to produce the
//          base URI for the contact with _ID == 23.
//  → ”content://com.android.contacts/people/23”
Uri myPerson = ContentUris.withAppendedId(People.CONTENT_URI, 23);

// Alternatively, use the Uri method to produce the base URI.
// It takes a string rather than an integer.
//  → ”content://com.android.contacts/people/23”
Uri myPerson = Uri.withAppendedPath(People.CONTENT_URI, "23");

// Then query for this specific record:
Cursor cur = managedQuery(myPerson, null, null, null, null);
```

Query

```
import android.provider.Contacts.People;
import android.database.Cursor;

// Form an array specifying which columns to return.
String[] projection = new String[] {
                          People._ID,
                          People._COUNT,
                          People.NAME,
                          People.NUMBER
                      };

// Get the base URI for the People table in the Contacts content provider.
Uri contacts =  People.CONTENT_URI;

// Make the query.
Cursor managedCursor = managedQuery(contacts,
                       projection, // Which columns to return
                       null,       // Which rows to return (all rows)
                       null,       // Selection arguments (none)
                       // Put the results in ascending order by name
                       People.NAME + " ASC");
```

Insert/Update/Delete

```
private void insertRecords(String name, String phoneNo) {
        ContentValues values = new ContentValues();
        values.put(People.NAME, name); Uri uri =
        getContentResolver().insert(People.CONTENT_URI, values);
 }
```

```
private void updateRecord(int recNo, String name) {
        //appending recNo, record to be updated
        Uri uri = ContentUris.withAppendedId(People.CONTENT_URI, recNo);
        ContentValues values = new ContentValues();
        values.put(People.NAME, name);
        getContentResolver().update(uri, values, null, null);
  }
```

```
private void deleteRecords() {
        Uri uri = People.CONTENT_URI;
        getContentResolver().delete(uri, null, null);
 }
```

Implementing your own Content Provider

1. Set up a system for storing the data
   - e.g., SQLite using SQLiteOpenHelper
2. Extend ContentProvider class to provide access:
   - query()—Allows third-party applications to retrieve content.
   - insert()—Allows third-party applications to insert content.
   - update()—Allows third-party applications to update content.
   - delete()—Allows third-party applications to delete content.
   - getType()—Allows third-party applications to read each of URI structures supported.
   - onCreate()—Creates a database instance to help retrieve the content.
3. Declare the Content Provider in manifest

Extend Content Provider

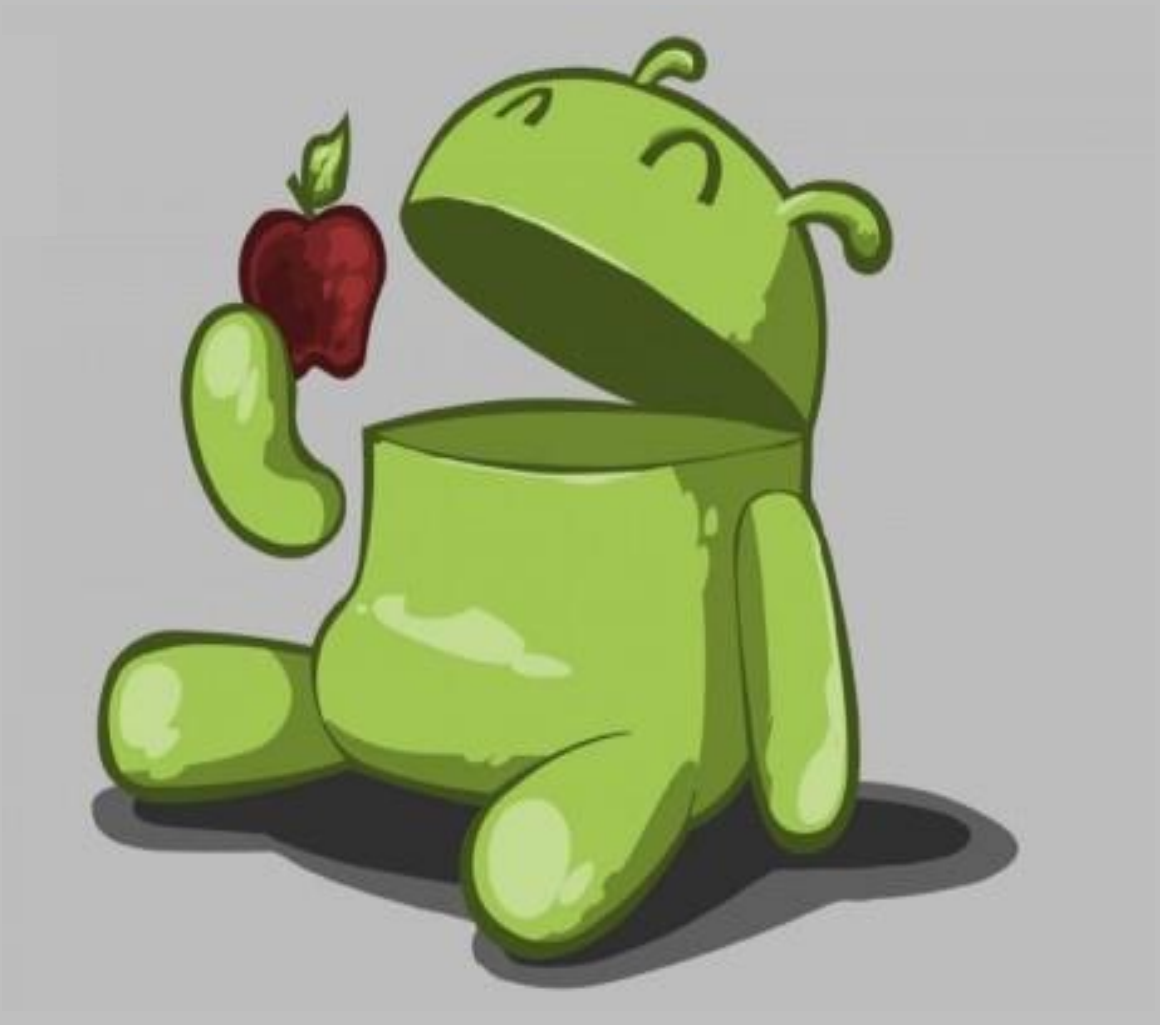
```
public class MyContentProvider extends ContentProvider {
        public static final String AUTHORITY = "edu.odu.phonenumber";
        public static final Uri CONTENT_URI =
                        Uri.parse("content://"+ AUTHORITY + "/phones");
        private MySQLDatabase mDB;
        private static final int PHONES = 1;
        private static final int PHONES_ID = 2;
        private static final UriMatcher uriMatcher;
        static{
              uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
              uriMatcher.addURI(AUTHORITY, “phones", PHONES);
              uriMatcher.addURI(AUTHORITY, “phones/#", PHONES_ID);
        }

        public boolean onCreate() {
              mDB = new MySQLDatabase(getContext());
              return true;
        }
…
}
```

Extend Content Provider

```
public class MyContentProvider extends ContentProvider {
        ...
        public Cursor query(Uri uri, String[] projection, String selection,
                            String[] selectionArgs, String sortOrder) {
            Cursor c=null;
            SQLiteDatabase db = mDB.getWritableDatabase();
            switch (uriMatcher.match(uri)) {
                case PHONES:            //get all phones records
                    ...
                    break;
                case PHONES_ID:         //get a particular phone record
                    …
                    break;
             }
             …
        }

        public String getType(Uri uri) {...}
        public int delete(Uri uri, String selection, String[] selectionArgs) {...}
        public Uri insert(Uri uri, ContentValues values) {...}
        public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {...}
 }
```

Manifest

```
<application android:icon="@drawable/icon" android:label="@string/app_name">
        …

      <provider android:name=".MyContentProvider"
            android:authorities=" edu.odu.phonenumber"></provider>

</application>
```



Questions?