ContentProviders

Programming the Android Platform

ContentProvider

- Databases for reading & writing data
- Support typical database operations
  - e.g., query, insert, update & delete

ContentResolver

- Manages & supports ContentProviders
  - Enables ContentProviders to be used across multiple applications
  - Provides additional services such as change notification
- Use Context.getContentResolver() to access

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

Android ContentProviders

- ContentProviders manage data for
  - Browser – bookmarks, history
  - Call log- telephone usage
  - Contacts – contact data
  - Media – media database
  - UserDictionary – database for predictive spelling
  - Many more

Data Model

- Content providers stored logically as database tables
  - e.g., artists table in MediaProvider

artist_id	artist_key	artist
13	hashcode1	Lady Gaga
44	hashcode2	Frank Sinatra
45	hashcode1	Elvis Presley
53	hashcode4	Barbara Streisand

Content URIs

- ContentProvider identify data sets through URIs,
  - e.g., *content://authority/path/id*
- content - data managed by a content provider
- authority – id for the content provider
- path – 0 or more segments indicating the type of data to be accessed
- id – specific record being requested

Example: Contacts

- Uri for searching contacts database

ContactsContract.Contacts.CONTENT_URI =
"content://com.android.contacts/contacts/"

Querying

- Use ContentResolver.query() to retrieve data
  - Returns a Cursor instance for accessing results
  - A Cursor is an iterator over a result set

```
Cursor query(
        Uri uri,                    // ContentProvider Uri
        String[] projection         // Columns to retrieve
        String selection            // SQL selection pattern
        String[] selectionArgs      // SQL pattern args
        String sortOrder            // Sort order
)
```

Querying (cont.)

```
public class ContactsListExample extends ListActivity {
  public void onCreate(Bundle savedInstanceState) {
    ContentResolver cr = getContentResolver();
    Cursor c = cr.query(
       ContactsContract.Contacts.CONTENT_URI,
       new String[] { ContactsContract.Contacts.DISPLAY_NAME },
       null,   null,   null);
    …
  }
…
```

Querying (cont.)

```
public class ContactsListExample extends ListActivity {
  public void onCreate(Bundle savedInstanceState) {
  ...
  String columns[] = new String[] {
    ContactsContract.Contacts._ID,
    ContactsContract.Contacts.DISPLAY_NAME,
    ContactsContract.Contacts.STARRED };
  ...
  ContentResolver cr = getContentResolver();
  Cursor c = cr.query(
    ContactsContract.Contacts.CONTENT_URI, columns,
    ContactsContract.Contacts.STARRED + "= 0", null, null);
  ...
```

Cursor

- Provides access to query results
- Some useful methods
  - boolean moveToFirst()
  - boolean moveToNext()
  - int getColumnIndex(String columnName)
  - String getString(int columnIndex)

Cursor (cont.)

```
public class ContactsListExample extends ListActivity {
  public void onCreate(Bundle savedInstanceState) {
  Cursor c = // issue query
  List<String> contacts = new ArrayList<String>();
  if (c.moveToFirst()) {
   do {
      contacts.add(c.getString(c.getColumnIndex(
                        ContactsContract.Contacts.DISPLAY_NAME)));
    } while (c.moveToNext());
   …
```

Putting it Together (Example 1)

```
public class ContactsListExample extends ListActivity {
  public void onCreate(Bundle savedInstanceState) {
    ...
    ContentResolver cr = getContentResolver();
    Cursor c = // issue query
    List<String> contacts = new ArrayList<String>();
    if (c.moveToFirst()) {
      do {
        // add data to contacts variable
      } while (c.moveToNext());
    }
    ...
```

Putting it Together (Example 1)

```
...
// populate list view widget
  ArrayAdapter<String> adapter =
    new ArrayAdapter<String>(this, R.layout.list_item, contacts);
  setListAdapter(adapter);
  }
}
```

Putting it Together (Example 2)

```
public class ContactsListExample extends ListActivity {
  public void onCreate(Bundle savedInstanceState) {
    ...
    // columns to retrieve
    String columns[] = new String[] {
                        ContactsContract.Contacts._ID,
                        ContactsContract.Contacts.DISPLAY_NAME,
                        ContactsContract.Contacts.STARRED };
    // columns to display
    String disp [] = new String[] {
                            ContactsContract.Contacts.DISPLAY_NAME};
   // layout for columns to display
    int[] colResIds = new int[] { R.id.name };
    ...
```

Putting it Together (cont.)

```
  ...
  ContentResolver cr = getContentResolver();
  Cursor c = cr.query(
       ContactsContract.Contacts.CONTENT_URI, columns,
       ContactsContract.Contacts.STARRED + "= 0”, null, null);

  setListAdapter(new SimpleCursorAdapter(
          this, R.layout.list_layout,  c,colsToDisplay, colResIds));
  }
}
```

Deleting Data

- Use ContentResolver.delete() to delete data

```
public final int delete (
        Uri url,                    // content Uri
        String where,               // SQL selection pattern
        String[] selectionArgs      // SQL pattern args
)
```

Deleting Data (cont.)

```
public class ContactsListDisplayActivity extends ListActivity {
  …
  private void deleteContact(String name) {
    getContentResolver().delete(
     ContactsContract.RawContacts.CONTENT_URI,
     ContactsContract.Contacts.DISPLAY_NAME + "=?",
     new String[] {name});
  }
  private void deleteAllContacts() {
    getContentResolver().delete(
            ContactsContract.RawContacts.CONTENT_URI, null, null);
  }
  …
}
```

Inserting Data

```
public class ContactsListDisplayActivity extends ListActivity {
  …
  private void insertContact(String name) {
    ArrayList<ContentProviderOperation> ops =
                    new  ArrayList<ContentProviderOperation>();
   …
```

Inserting Data (cont.)

```
// create new RawContact
  ops.add(ContentProviderOperation
    .newInsert(RawContacts.CONTENT_URI)
    .withValue(RawContacts.ACCOUNT_TYPE, "com.google")
    .withValue(RawContacts.ACCOUNT_NAME,
                                        "adam.a.porter@gmail.com")

    .build());
// add new RawContact
  ops.add(ContentProviderOperation.newInsert(Data.CONTENT_URI)
   .withValueBackReference(Data.RAW_CONTACT_ID,0)
   .withValue(Data.MIMETYPE,StructuredName.CONTENT_ITEM_TYPE)
   .withValue(StructuredName.DISPLAY_NAME, name)
   .build());
```

Inserting Data (cont.)

```
  ...
    try {
      getContentResolver()
          .applyBatch(ContactsContract.AUTHORITY, ops);
    } catch (RemoteException e) {
    } catch (OperationApplicationException e) {
    }
  ...
}
```

Creating a ContentProvider

- Implement a storage system for the data
- Implement a ContentProvider subclass
- Declare content provider in manifest

MyContentProvider

```
public class MyContentProvider extends ContentProvider {

  public static final Uri CONTENT_URI = Uri.parse(
    "content://course.examples.ContentProviders.myContentProvider/");
  public static final String _ID = "id", DATA= "data";

  private static final String[] columns = new String[] { _ID, DATA};
  private static final Map<Integer, DataRecord> db =
                                          new HashMap<Integer, DataRecord>();
  private static final String contentTypeSingle =
                        "vnd.android.cursor.item/myContentProvider.data.text";
  private static final String contentTypeMultiple =
                         "vnd.android.cursor.dir/myContentProvider.data.text";
```

MyContentProvider (cont.)

```
public synchronized int delete(
                    Uri uri, String selection, String[] selectionArgs) {
String requestIdString = uri.getLastPathSegment();
  if (null == requestIdString) {
    for (DataRecord dr : db.values()) { db.remove(dr.get_id()); }
  } else {
    Integer requestId = Integer.parseInt(requestIdString);
    if (db.containsKey(requestId)) { db.remove(db.get(requestId));
    }
  }
  return // # of records deleted;
}
```

MyContentProvider (cont.)

```
public synchronized Uri insert(Uri uri, ContentValues values) {
  if (values.containsKey(Data)) {
    DataRecord tmp =
                    new DataRecord(values.getAsString(Data));
    db.put(tmp.get_id(), tmp);
    return Uri.parse(CONTENT_URI +
                                    String.valueOf(tmp.get_id()));
  }
  return null;
}
```

MyContentProvider (cont.)

```
public synchronized Cursor query(
                    Uri uri, String[] projection, String selection,
                              String[] selectionArgs, String sortOrder) {
  String requestIdString = uri.getLastPathSegment();
  MatrixCursor cursor = new MatrixCursor(columns);
  if (null == requestIdString) {
    for (DataRecord dr : db.values()) {
      cursor.addRow(new Object[] {dr.get_id(), dr.get_data()});
    }
  }
...
```

MyContentProvider (cont.)

```
...
else {
    Integer requestId = Integer.parseInt(requestIdString);
    if (db.containsKey(requestId)) {
      DataRecord dr = db.get(requestId);
      cursor.addRow(new Object[] {dr.get_id(), dr.get_data() });
    }
  }
  return cursor;
}
```

MyContentProvider (cont.)

```
<manifest ...
   package="course.examples.ContentProviders.myContentProvider"  ..."> 
   <application... >
      <activity android:name=".CustomContactProviderDemo" ...>
         ...
      </activity>
      <provider android:name=".MyContentProvider"
                     android:authorities=
                        "course.examples.ContentProviders.myContentProvider">
      </provider>
   </application>
 <uses-sdk android:minSdkVersion="10" />
</manifest>
```

Lab Assignment

Source Code Examples

- ContentProviderCustom
- ContentProviderExample
- ContentProviderWithInsertionDeletion
- ContentProviderWithSimpleCursorAdapter