Android Content Providers

Content Provider, A data Access Layer abstracting access to different data sources & data types

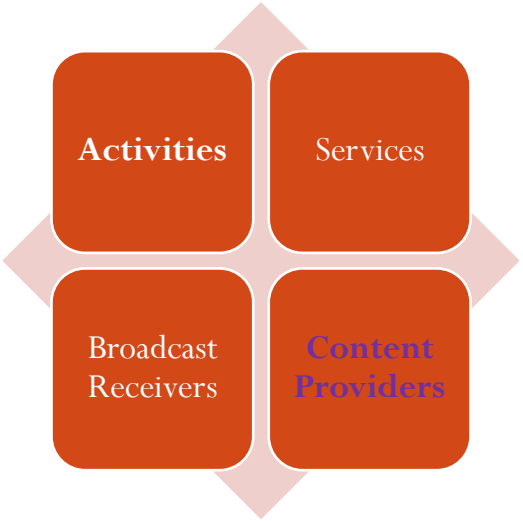
A.R. Kazemi

ContentProvider

- A **ContentProvider** supplies data from one App to other Apps on their request
- You may create a content provider by inheriting from ContentProvider:

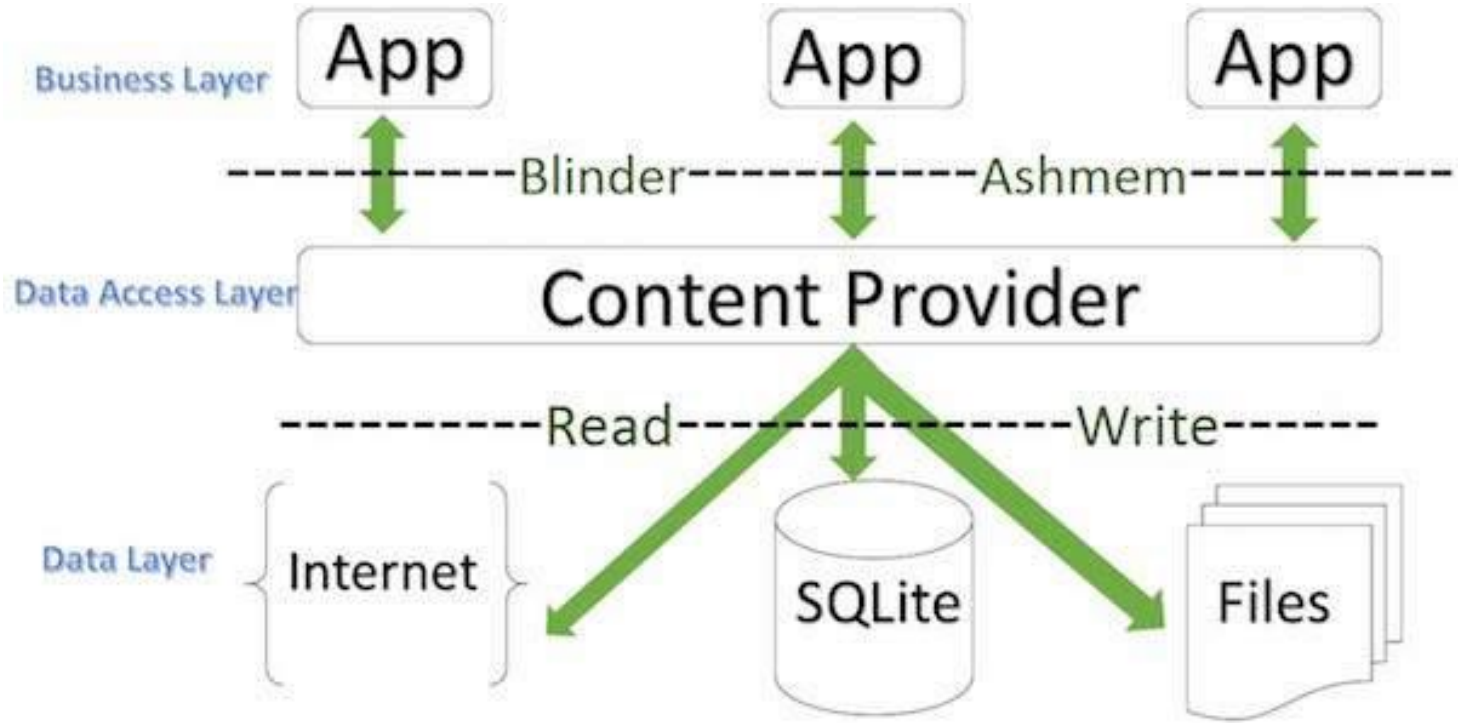
```
class MyLocationProvider extends ContentProvider{ … }
```

- It is one of four majur Android App components



A Standard for Sharing Data

- Hides the differences of different data sources
- Allows Involvement of data owner on data sharing



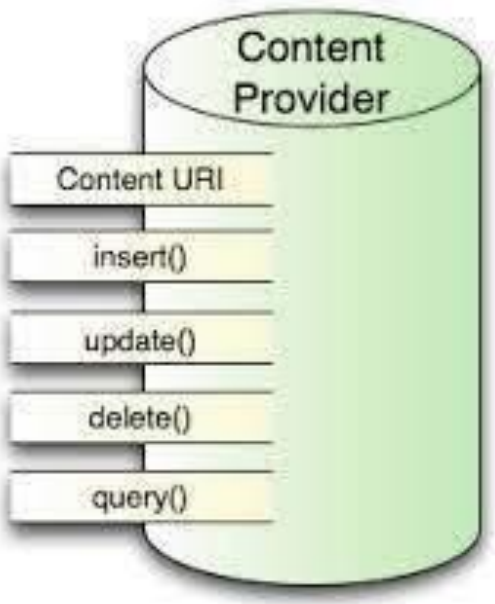
Content URIs

- To query a content provider, you specify the query string in the form of a URI which has following format −
  - <prefix>://<authority>/<data_type>/<id>

Sr.No	Part & Description
1	**prefix:** This is always set to content://
2	**Authority:** This specifies the name of the content provider, for example contacts, browser etc. For third-party content providers, this could be the fully qualified name, such as com.tutorialspoint.statusprovider
3	**data_type:** This indicates the type of data that this particular provider provides. For example, if you are getting all the contacts from the Contacts content provider, then the data path would be people and URI would look like thiscontent://contacts/people
4	**Id:** This specifies the specific record requested. For example, if you are looking for contact number 5 in the Contacts content provider then URI would look like this content://contacts/people/5.

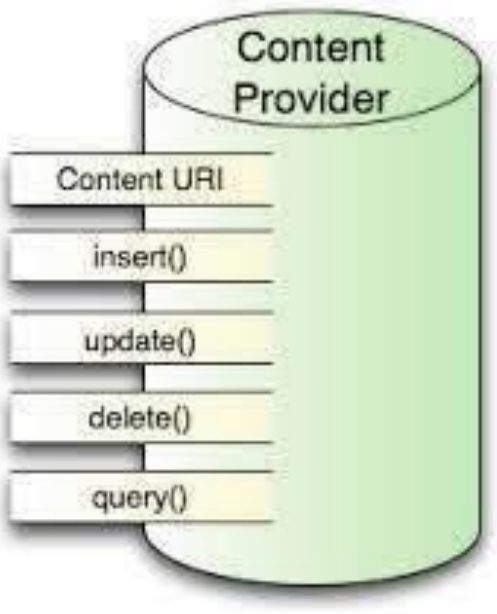
Create Content Provider

- First create class that extends the *ContentProvider* base class.
- Second, define your content provider URI address
  - this will be used to access the content.
- Next create your own database/data storage to keep the content.
  - By Default Android Apps use SQLite database and you needs to override ***onCreate()*** method which will use SQLite Open Helper method to create or open the provider's database.
  - When your application is launched, the *onCreate()* handler of each of its Content Providers is called on the main application thread.
- Next implement ContentProvider queries to perform different data specific operations.
- Finally register your ContentProvider in your manifest file using **<provider>** tag.



ContentProvide methods

- **onCreate()**
  - This method is called when the provider is started.
- **query()**
  - This method receives a request from a client. The result is returned as a Cursor object.
- **insert()**
  - This method inserts a new record into the content provider.
- **delete()**
  - This method deletes an existing record from the content provider.
- **update()**
  - This method updates an existing record from the content provider.
- **getType()**
  - This method returns the MIME type of the data at the given URI.



Example StudentProvider

```
public class StudentsProvider extends ContentProvider {
   static final String PROVIDER_NAME = "com.example.MyApplication.StudentsProv
   static final String URL = "content://" + PROVIDER_NAME + "/students";
   static final Uri CONTENT_URI = Uri.parse(URL);

   static final String _ID = "_id";
   static final String NAME = "name";
   static final String GRADE = "grade";

   private static HashMap<String, String> STUDENTS_PROJECTION_MAP;

   static final int STUDENTS = 1;
   static final int STUDENT_ID = 2;

   static final UriMatcher uriMatcher;
   static{
      uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
      uriMatcher.addURI(PROVIDER_NAME, "students", STUDENTS);
      uriMatcher.addURI(PROVIDER_NAME, "students/#", STUDENT_ID);
   }
```

Example StudentProvider ...

```
/**
    * Database specific constant declarations
*/

private SQLiteDatabase db;
static final String DATABASE_NAME = "College";
static final String STUDENTS_TABLE_NAME = "students";
static final int DATABASE_VERSION = 1;
static final String CREATE_DB_TABLE =
    " CREATE TABLE " + STUDENTS_TABLE_NAME +
        " (_id INTEGER PRIMARY KEY AUTOINCREMENT, " +
        " name TEXT NOT NULL, " +
        " grade TEXT NOT NULL);";
```

Example StudentProvider ...

```
/**
    * Helper class that actually creates and manages
    * the provider's underlying data repository.
*/

private static class DatabaseHelper extends SQLiteOpenHelper {
    DatabaseHelper(Context context){
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_DB_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
        db.execSQL("DROP TABLE IF EXISTS " +  STUDENTS_TABLE_NAME);
        onCreate(db);
    }
}
```

Example StudentProvider ...

```
@Override
public boolean onCreate() {
    Context context = getContext();
    DatabaseHelper dbHelper = new DatabaseHelper(context);

    /**
     * Create a write able database which will trigger its
     * creation if it doesn't already exist.
     */

    db = dbHelper.getWritableDatabase();
    return (db == null)? false:true;
}
```

Example StudentProvider ...

```
@Override
public Uri insert(Uri uri, ContentValues values) {
    /**
        * Add a new student record
    */
    long rowID = db.insert(    STUDENTS_TABLE_NAME, "", values);

    /**
        * If record is added successfully
    */
    if (rowID > 0) {
        Uri _uri = ContentUris.withAppendedId(CONTENT_URI, rowID);
        getContext().getContentResolver().notifyChange(_uri, null);
        return _uri;
    }

    throw new SQLException("Failed to add a record into " + uri);
}
```

Example StudentProvider …

```
@Override
public Cursor query(Uri uri, String[] projection,
   String selection,String[] selectionArgs, String sortOrder) {
   SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
   qb.setTables(STUDENTS_TABLE_NAME);

   switch (uriMatcher.match(uri)) {
      case STUDENTS:
         qb.setProjectionMap(STUDENTS_PROJECTION_MAP);
      break;
      case STUDENT_ID:
         qb.appendWhere( _ID + "=" + uri.getPathSegments().get(1));
      break;
      default:
   }
   if (sortOrder == null || sortOrder == ""){
      /** By default sort on student names */
      sortOrder = NAME;
   }
   Cursor c = qb.query(db,   projection,     selection,
      selectionArgs,null, null, sortOrder);
   /** register to watch a content URI for changes  */
   c.setNotificationUri(getContext().getContentResolver(), uri);
   return c;
}
```

Example StudentProvider ...

```
@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
    int count = 0;
    switch (uriMatcher.match(uri)){
        case STUDENTS:
            count = db.delete(STUDENTS_TABLE_NAME, selection, selectionArgs);
        break;

        case STUDENT_ID:
            String id = uri.getPathSegments().get(1);
            count = db.delete( STUDENTS_TABLE_NAME, _ID +  " = " + id +
                (!TextUtils.isEmpty(selection) ? "
                AND (" + selection + ')' : ""), selectionArgs);
            break;
        default:
            throw new IllegalArgumentException("Unknown URI " + uri);
    }

    getContext().getContentResolver().notifyChange(uri, null);
    return count;
}
```

Example StudentProvider ...

```
@Override
public int update(Uri uri, ContentValues values,
    String selection, String[] selectionArgs) {
    int count = 0;
    switch (uriMatcher.match(uri)) {
        case STUDENTS:
            count = db.update(STUDENTS_TABLE_NAME, values, selection, selectionArgs);
        break;

        case STUDENT_ID:
            count = db.update(STUDENTS_TABLE_NAME, values,
                _ID + " = " + uri.getPathSegments().get(1) +
                (!TextUtils.isEmpty(selection) ? "
                AND (" +selection + ')' : ""), selectionArgs);
            break;
        default:
            throw new IllegalArgumentException("Unknown URI " + uri );
    }

    getContext().getContentResolver().notifyChange(uri, null);
    return count;
}
```

Example StudentProvider ...

```
   @Override
   public String getType(Uri uri) {
      switch (uriMatcher.match(uri)){
         /**
            * Get all student records
         */
         case STUDENTS:
            return "vnd.android.cursor.dir/vnd.example.students";
         /**
            * Get a particular student
         */
         case STUDENT_ID:
            return "vnd.android.cursor.item/vnd.example.students";
         default:
            throw new IllegalArgumentException("Unsupported URI: " + uri);
      }
   }
}
```

Example StudentProvider

Declare in Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.MyApplication">

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">
         <activity android:name=".MainActivity">
            <intent-filter>
               <action android:name="android.intent.action.MAIN" />
               <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
         </activity>

      <provider android:name="StudentsProvider"
         android:authorities="com.example.MyApplication.StudentsProvider"/>
   </application>
</manifest>
```

Example StudentProvider

Use it in Activity

```
public class MainActivity extends Activity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
   public void onClickAddName(View view) {
      // Add a new student record
      ContentValues values = new ContentValues();
      values.put(StudentsProvider.NAME,
         ((EditText)findViewById(R.id.editText2)).getText().toString());

      values.put(StudentsProvider.GRADE,
         ((EditText)findViewById(R.id.editText3)).getText().toString());

      Uri uri = getContentResolver().insert(
         StudentsProvider.CONTENT_URI, values);

      Toast.makeText(getBaseContext(),
         uri.toString(), Toast.LENGTH_LONG).show();
   }
```

Example StudentProvider

Use it in Activity …

```
   public void onClickRetrieveStudents(View view) {
      // Retrieve student records
      String URL = "content://com.example.MyApplication.StudentsProvider";

      Uri students = Uri.parse(URL);
      Cursor c = managedQuery(students, null, null, null, "name");

      if (c.moveToFirst()) {
         do{
            Toast.makeText(this,
               c.getString(c.getColumnIndex(StudentsProvider._ID)) +
                  ", " +  c.getString(c.getColumnIndex( StudentsProvider.NAME)) +
                     ", " + c.getString(c.getColumnIndex( StudentsProvider.GRADE)),
            Toast.LENGTH_SHORT).show();
         } while (c.moveToNext());
      }
   }
}
```