Android Developer Fundamentals

Content Providers

Lesson 11

11.1 Content Providers

Share data with other apps

Contents

- What is a ContentProvider
- App with Content Provider Architecture
- Implementation
  - Contract
  - ContentProvider
  - Manifest Permissions
  - Content Resolver

What is a Content Provider

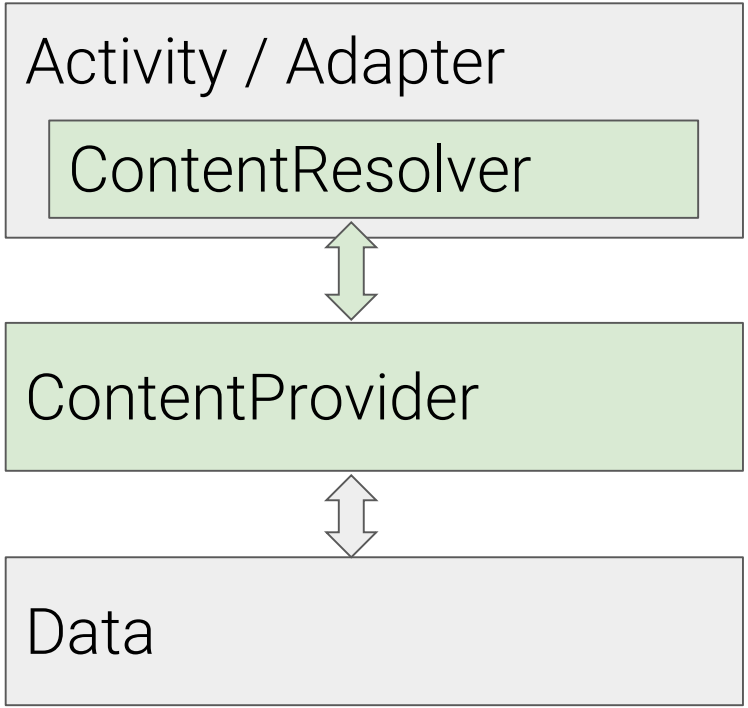
What is a Content Provider?

- A content provider is a component fetches data that the app requests from a repository
- The app doesn't need to know where or how the data is stored, formatted, or accessed

What is a Content Resolver?

- A content resolver is a component that your app uses to send requests to a content provider
- Requests consist of a content URI and an SQL-like query
- The ContentResolver object provides query(), insert(), update(), and delete() methods

How do they work together?



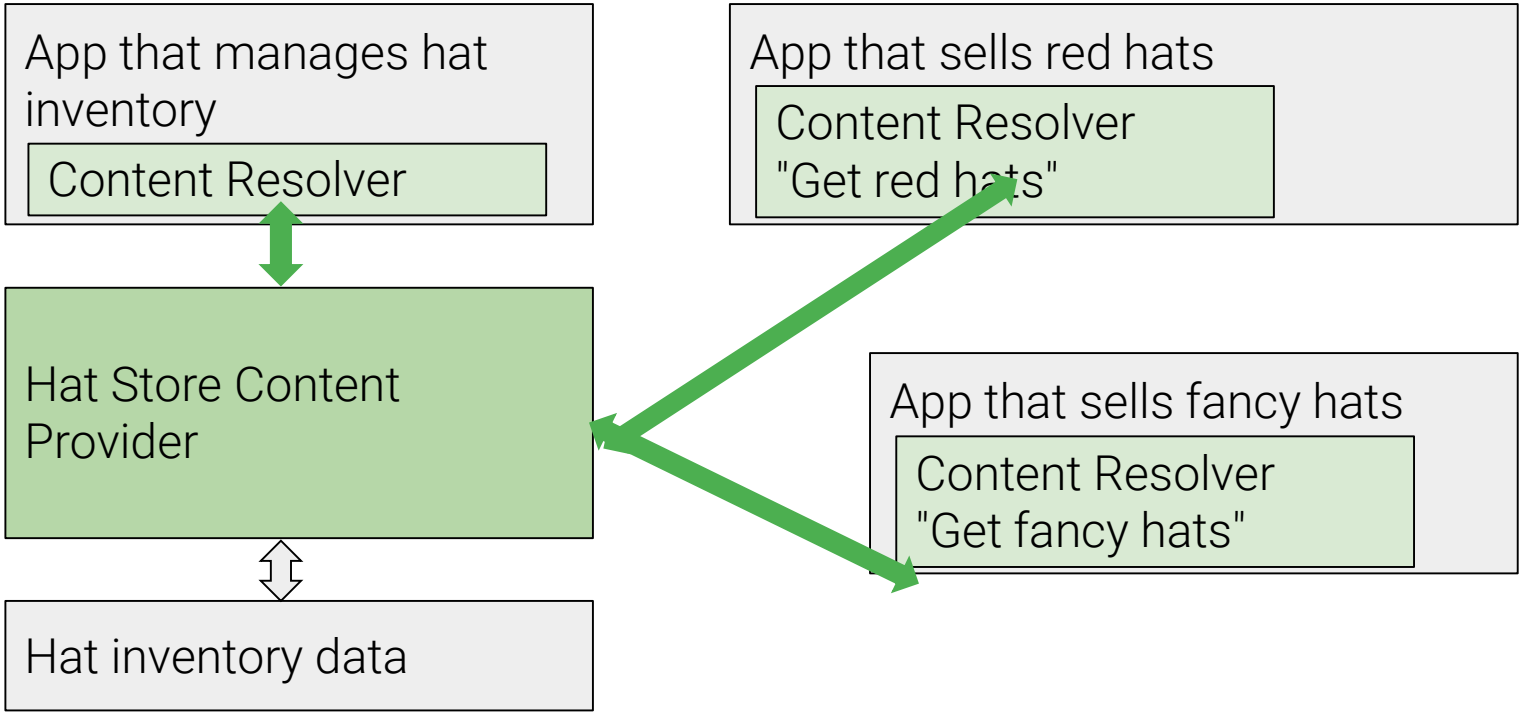
1. Activity/Adapter uses ContentResolver to query ContentProvider
2. ContentProvider gets data
3. ContentResolver returns data as Cursor
4. Activity/Adapter uses data

What is it good for?

- Securely make data available to other apps
- Manage access permissions to app data

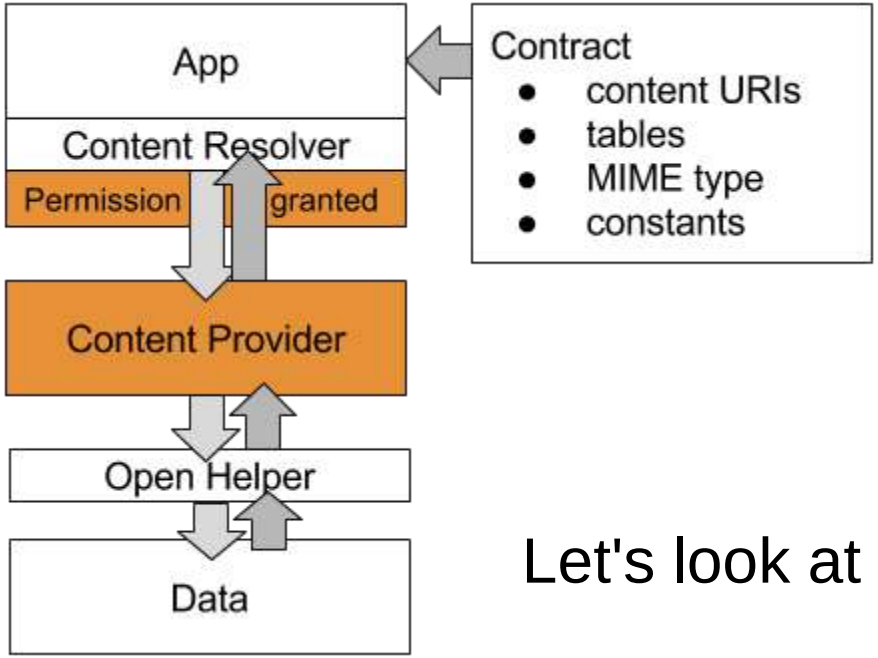
- Store data or develop backend independently from UI
- Standardized way of accessing data
- Required to work with CursorLoaders

Many apps can use one content provider



App with Content Provider

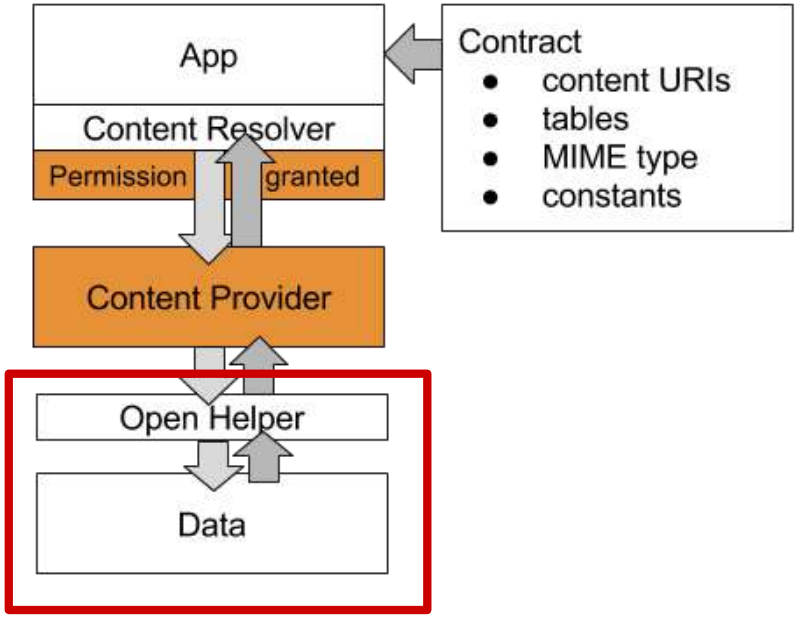
Components of a Content Provider App



Let's look at each of these...

Data Repository

- Data, commonly in SQLite Database but can be any backend
- OpenHelper if you use SQLite database

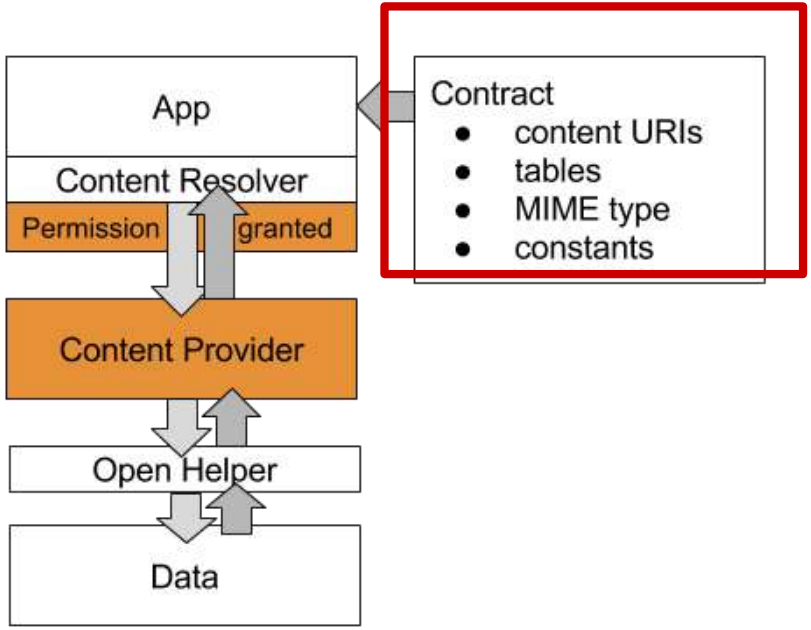


Contract

Public class that exposes information about the content provider to other apps

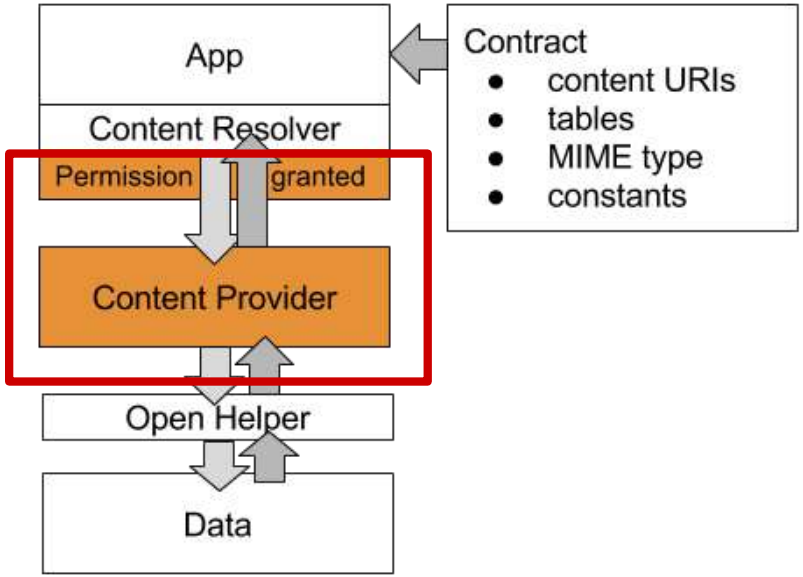
Contract specifies

- URIs to query data
- Table structure of data
- MIME type of returned data
- Constants



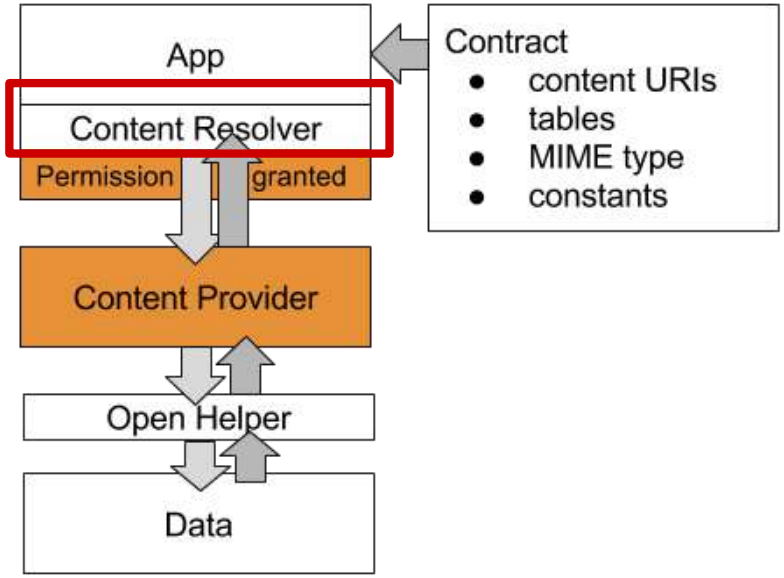
Content Provider

- Public secure interface
- Permissions control access
- Subclass of ContentProvider
- query(), insert(), delete(), update() API



Content Resolver

- Send requests to content provider and return response



Implementation

Implementing a Content Provider

- Data—Commonly an SQLite database
- Write contract—Information about the content provider
- Subclass ContentProvider and implement methods
- Get data using ContentResolver
- Set permissions in Android Manifest

Why SQLite Database?

- Represents data in tables
- Supports same operations as content provider
- Returns requested data as cursor
- OpenHelper class to simplify management

Contract

What's in a Contract?

- Public class that documents content provider API
- Called Contract by convention
- Contains
  - Content URIs and URI scheme to query data
  - Table and column names for returned data
  - MIME types to help process returned data
  - Shared constants to make life easier

Content URI

- `http://` and `file://` are URIs for web pages and files
- Content URI is path to data and uses `content://`

E.g. request all the entries in the "words" table

```
content://com.android.example.wordcontentprovider.provider/words
```

General form of a URI

```
scheme://authority/path/id
```

- **scheme** is always `content://` for content URIs
- **authority** represents the domain, and for content providers customarily ends in `.provider`
- **path** is the path to the data
- **id** uniquely identifies the data set to search

URI Scheme

By convention, provide constants for

- `AUTHORITY`—Domain
- `CONTENT_PATH`—Path to the data
- `CONTENT_URI`—URI to one set of data

URI Scheme in Code

```
public static final String AUTHORITY =
"com.android.example.minimalistcontentprovider.provider";

public static final String CONTENT_PATH =  "words";

public static final Uri CONTENT_URI = Uri.parse("content://" +
AUTHORITY + "/" + CONTENT_PATH);
```

Table definitions

```
public static final String DATABASE_NAME = "wordlist";

public static abstract class WordList implements BaseColumns {
    public static final String WORD_LIST_TABLE = "word_entries";
    // Column names...
    public static final String KEY_ID = "_id";
    public static final String KEY_WORD = "word";
}
```

MIME Type

- Format of returned data
- text/html for web pages, application/json for JSON data
- App calls `getType()` to get MIME type from provider
- Use Android's vendor-specific format for your content providers MIME type

Android's vendor-specific MIME Type

`type.subtype/provider-specific-part`

- Type: **`vnd`**
- Subtype
  - If URI pattern is for a single row: **`android.cursor.item/`**
  - If URI pattern is for more than one row: **`android.cursor.dir/`**
- Provider-specific part: **`vnd.<name>.<type>`**
  - <name>: globally unique, such as company or package name
  - <type> unique to corresponding URI pattern, such as table name

MIME Type Example

Multiple words

```
vnd.android.cursor.dir/vnd.com.example.provider.words
```

One word

```
vnd.android.cursor.item/vnd.com.example.provider.words
```

MIME Type code in Contract

```
static final String SINGLE_RECORD_MIME_TYPE =
    "vnd.android.cursor.item/vnd.com.example.provider.words";

static final String MULTIPLE_RECORDS_MIME_TYPE =
        "vnd.android.cursor.item/vnd.com.example.provider.words";
```

getType()

```
@Override
public String getType(Uri uri) {
    switch (sUriMatcher.match(uri)) {
        case URI_ALL_ITEMS_CODE:
            return MULTIPLE_RECORDS_MIME_TYPE;
        case URI_ONE_ITEM_CODE:
            return SINGLE_RECORD_MIME_TYPE;
        default:
            return null;
    }
}
```

Constants

- Constants that are used by multiple classes in an app
- Convenience constants for client use
- Encapsulate parameters whose values might change as constants, so that if the content provider changes, the clients don't break

ContentProvider

Extend ContentProvider

```
public class WordListContentProvider
                        extends ContentProvider {}
```

Implement methods

- query(), insert(), delete(), and update() methods
- interact with the data backend …
- … such as an Open Helper

Example insert()

```
/**
     * Inserts one record.
     *
     * @return URI for the newly created entry.
     */
    @Override
    public Uri insert(Uri uri, ContentValues values) {
        long id = mDB.insert(values);
        return Uri.parse(CONTENT_URI + "/" + id);
    }
```

Manifest Permissions

Set Permissions!

- By default, with no permissions set explicitly, any other app can access a content provider for reading and writing
- Set read or write permissions in AndroidManifest
- Use unique tags that include package name

Provider in Android Manifest

```
<provider
android:name=".WordListContentProvider"
android:authorities=
    "com.android.example.wordlistsqlwithcontentprovider.provider"
android:exported="true" />
```

Permissions inside <provider>

```
android:readPermission=
    "com.android.example.wordlistsqlwithcontentprovider.PERMISSION"

android:writePermission=
    "com.android.example.wordlistsqlwithcontentprovider.PERMISSION"
```

Client permissions

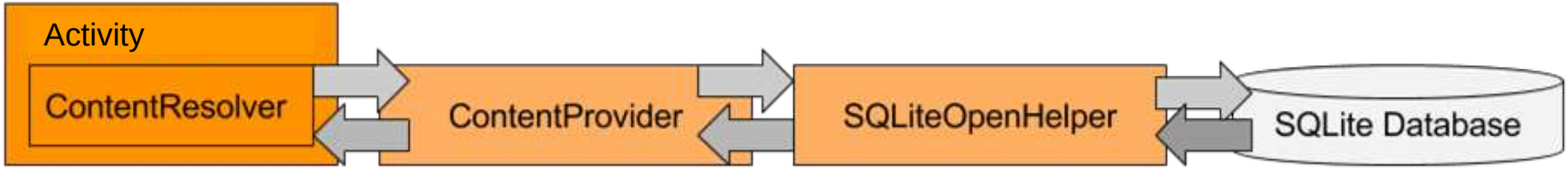
- Granted by the user

```
<uses-permission android:name =
 "com.android.example.wordlistsqlwithcontentprovider.PERMISSION"/>
```

Content Resolver

Content Resolver

- Must use a ContentResolver to send requests as queries to the content provider
- Data is returned in a Cursor object, as rows and columns



Content Resolver methods

- ContentResolver.query()
- ContentResolver.insert()
- ContentResolver.delete()
- ContentResolver.update()

getContentResolver.query()

```
public Cursor query(
    Uri uri,
    String[] projection,
    String selection,
    String[] selectionArgs,
    String sortOrder){ … // implementation }
```

Calling getContentResolver.query()

```
Cursor cursor = getContentResolver().query(
    Uri.parse(queryUri), projection, selectionClause,
    selectionArgs, sortOrder);
```

Query parameters (recap)

```
uri: String queryUri = Contract.CONTENT_URI.toString();

projection: String[] projection =
                    new String[] {Contract.CONTENT_PATH};

selection: String where = KEY_WORD + " LIKE ?";

selectionArgs: String[]whereArgs = new String[]{searchString};

sortOrder: > null for default, ASC / DESC
```

Recap

Summary of implementing content provider

- Data, for example, in a database
- A way for accessing the backend, for example, through an open helper
- Declare content provider in Android Manifest and set permissions
- Extend ContentProvider and implement query(), insert(), delete(), update(), count(), and getType() methods
- Create public Contract class to expose URI scheme, table names, MIME type, and important constants to other classes and apps
- Use a ContentResolver to send requests to content provider
    - Process data returned as cursor

Practicals Info

- Minimalist Content Provider to show you mechanics
- Add a content provider to the WordList app for a more realistic example
- Create a separate client app that accesses the content provider of the WordList app

Learn more

- Uniform Resource Identifiers or URIs
- MIME type
- MatrixCursor and Cursors
- Content Providers

Videos

- Android Application Architecture
- Android Application Architecture: The Next Billion Users

What's Next?

- Concept Chapter
- 11.1 C Content Providers
- Practicals
  - 11.1A P Minimalist Content Provider
  - 11.1B P Add a Content Provider to WorldListSQL
  - 11.1C P Sharing Content with Other Apps

END