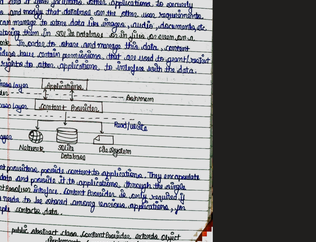
Unit 04: deepankar

24 December 2022 00:12

Content Provider: How ContentProvider works, steps to create content provider, URI, CRUD.

Content Provider

In android, content provider is one very important component that serves the purpose of sharing data across the applications. The role of content provider is the centralized system to have a central repository in which data of applications are stored and it also facilitates other applications to securely access and modify that database on the other user requirements. It can be used to store data like images, audio, documents etc. by storing them in SQLite database. It allows to share or save a network. In order to share and manage this data, content provider have certain guidelines that are used to grant/restrict the rights to other applications to interface with the data.



Content provider provides content to applications. They encapsulate the data and provide it to applications. Through the single Content Provider Interface. Content Provider is easily accessible if data needs to be shared among various applications, for example contacts data.

```
public abstract class ContentProvider extends Object
    Implements Component Callbacks2
```

Content URI (Uniform Resource Identifier)

Content URI is the key concept of content Provider. To access the data from a content provider, URI is used as a query arg.

Structure of a Content URI

```
content://authority/optional path/optional id
```

Details of different Parts of content URI

1. content:// → mandatory part of URI as it represents that the given URI is a Content URI.
2. authority → specifies the name of content provider, like contacts, browser. This part must be unique for each content provider.
3. optional Path → specifies the type of data provided by the content provider.
4. optional id → → numerical value needed to access any particular record.

Content Resolver

A content provider is a component in android that provides an abstract layer of accessing data from another content provider. Android provides to use you to access data across different applications and components and provides a standard interface for performing CRUD (Create, Read, Update, Delete) operations on data.

Content Resolver are useful when you want to access data to other applications or when you want to abstract data access across layer and make it accessible by other components. They are implemented as a collection of Content Resolver class and support CRUD operations through their methods such as "insert", "update", "delete and query".

To use a content provider, you can obtain a "Content Resolver" object using the getContentResolver() and use methods of ContentResolver to perform operations on Content provider.

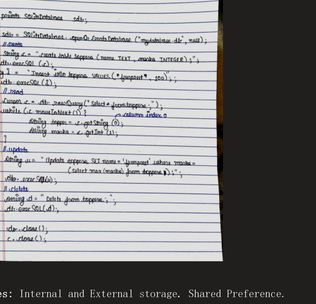
Content Resolver uses Uniform Resource Identifier 'URI' (Uniform Resource Identifier) which consists of an authority and a path. The authority identifies the content provider and path specifies the kind of resource in Content provider.

Data form CRUD operations. Content provider also supports in file based support of data like images, audio & videos.

CRUD in Android

CRUD is an acronym that stands for Create, Read, Update and Delete and refers to the four basic operations that are typically used to manipulate data in databases or other data stores. In Android, you can use various APIs and tools to perform CRUD operations on a local SQLite database or remote databases accessed over the network. Some of the APIs and tools that you can use for CRUD operations in Android are:

1. SQLiteDatabase class → This methods of this class such as "insert", "update", "delete" and "query" are used for performing CRUD operations on local SQLite database. This class is often used as a wrapper around the low-level SQLite API.
2. ContentResolver class → This class is a component that provides an abstract layer for accessing data from a content provider, on a device and supports CRUD operations through its methods.



Files: Internal and External storage. Shared Preference.

Shared Preference

In android, shared preference are a way of storing small amount of primitive data in key/value pairs, such as settings, preferences, and flags. Shared preferences are persisted between app launches and are private to the app by default. Although they can be made accessible to other apps if needed. Shared preferences are commonly used to store user preferences and other small pieces of data that need to be accessed quickly and easily. The shared preference class provides methods for reading and writing the key/value pairs, and for registering listeners to be notified of changes to the shared preferences.

```
import SharedPreferences.*;
sp = getSharedPreferences("myPref", MODE_PRIVATE);
editor.putInt("key", 123);
editor.putString("name", "John");
editor.apply();
int val = sp.getInt("key", 0);
```

↳ default value, if key is not found

Storage: Introduction to SQLite Database, SQLiteOpenHelper class, and Cursor.

SQLite Database

SQLite is a lightweight, self-contained database management system that is widely used in Android and other platforms. It is an open source relational database that stores data in binary files, on a device, and supports standard SQL commands for data manipulation and queries.

SQLiteOpenHelper

The SQLiteOpenHelper class is a helper class that is provided by Android framework to make it easier to work with SQLite databases in Android. It provides several methods that can be used to create, upgrade and manage a SQLite database and helps to ensure that the database is properly initialized and configured.

Cursor

A cursor is an interface in Android that represents a result set from a database query. It provides methods for iterating over the rows of the result set and for accessing the data in the columns of each row. Cursors are often used in conjunction with SQLiteOpenHelper class and SQLiteDatabase class to manage data from a database.

Database Creation

```
Database myDb = getDatabasePath("Contacts", MODE_PRIVATE);
```

Database Transition

```
myDb.execSQL("Create table if not exists TWINE (id INT);");
myDb.execSQL("Insert Into TWINE values (1);");
```

Database Querying

We can retrieve everything from the database using the object of the Cursor class. The moveNext() method of this class will return a resultset with the cursor pointing to the rows.

```
Cursor resultSet = myDb.rawQuery("Select * from TWINE");
resultSet.moveToFirst();
String Name = resultSet.getString(0);
```

Database Helper Class

For managing all the operations related to the database, we helper class has been given and is called SQLiteOpenHelper. It automatically manages the creation and upgradation of the database.

```
public class DbHelper extends SQLiteOpenHelper {
}
```

Creating an Android Provider

1. Create a class for querying database when the ContentProvider file resides and this class must extend the ContentProvider class.
2. To access the content, define a content provider URI address
3. Create a directory to store the application data.
4. Implement five core abstract methods of the ContentProvider class
   1. query() → retrieve cursor. This accepts the query parameters passed as arguments
   2. insert() → insert new row in database, returns URI of the inserted row
   3. update() → update rows in database, returns no of rows updated
   4. delete() → delete rows from database, returns no of rows deleted
   5. getType() → return MIME (Multipurpose Internet Mail Extension) data to given content URI
   6. onCreate() → initialization & initialization of content provider.
      → initialize system calls
5. Register the content provider. In AndroidManifest.xml file using <provider> tag.

Content URI

A content URI is a URI that identifies data in a provider. Content URI includes the authority, name of the content provider (e.g. contacts) and a name that points to a table (path). When you call a client method to access a table in a provider, the content URI for that table is one of this arguments.

```
content://com.example.project/users
```

The ContentResolver uses the path part of the content URI to choose the table to access