Content Provider

ContentProvider

A ContentProvider in Android shares data between applications.
Each application usually runs in its own process.
By default, applications can’t access the data and files of other applications.

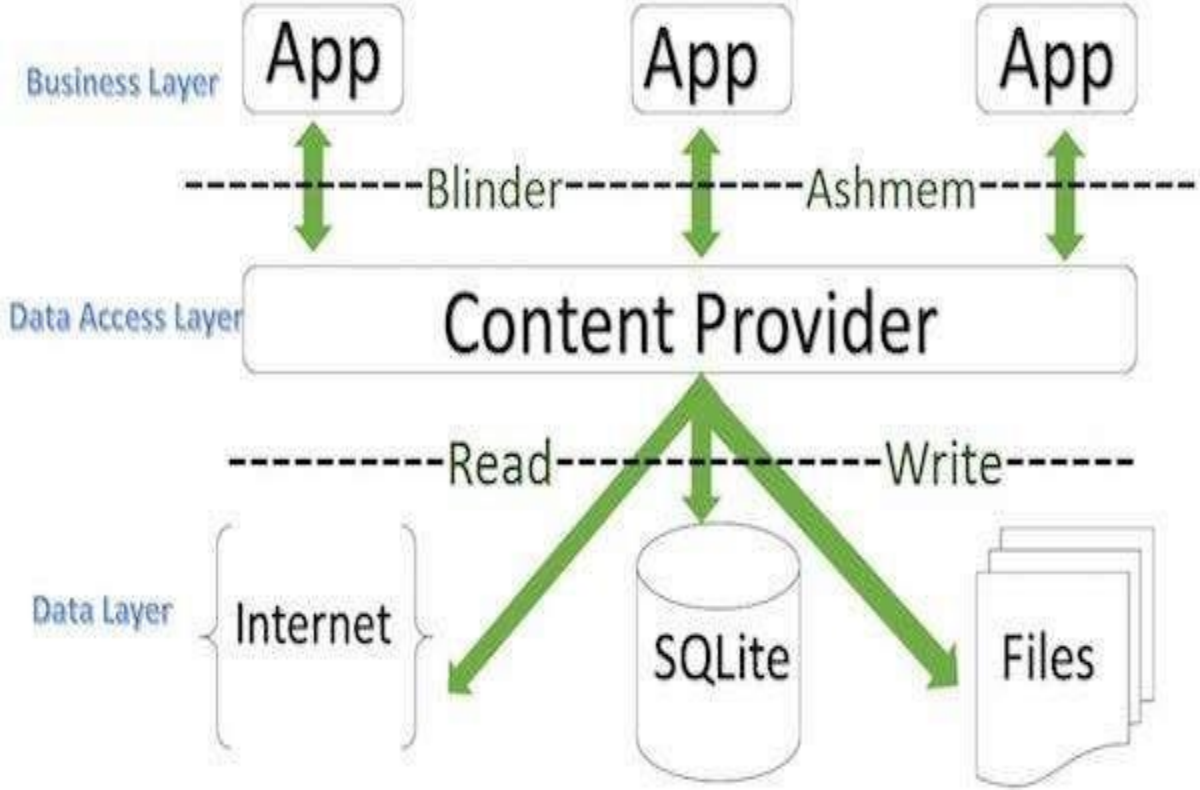
In contrast, with a ContentProvider you can publish and expose a particular data type for other applications to query, add, update, and delete, and those applications don’t need to have any prior knowledge of paths, resources, or who provides the content.

Content Provider

- A content provider manages access to a central repository of data.
- A provider is part of an Android application, which often provides its own UI for working with the data.
- However, content providers are primarily intended to be used by other applications, which access the provider using a provider client object.
- Together, providers and provider clients offer a consistent, standard interface to data that also handles inter-process communication and secure data access.
- Typically you work with content providers in one of two scenarios; you may want to implement code to access an existing content provider in another application, or you may want to create a new content provider in your application to share data with other applications.

Content Provider

- A content provider component supplies data from one application to others on request.
- Such requests are handled by the methods of the ContentResolver class.
- A content provider can use different ways to store its data and the data can be stored in a database, in files, or even over a network.
- In order to share the data, content providers have certain permissions that are used to grant or restrict the rights to other applications to interfere with the data.

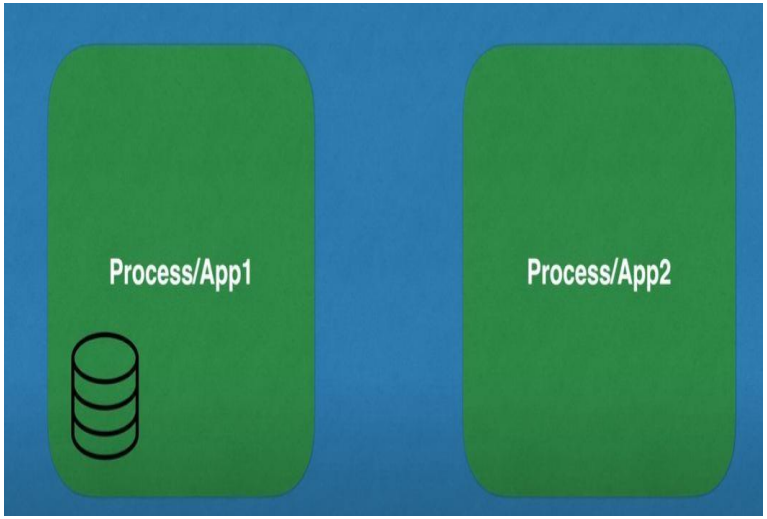


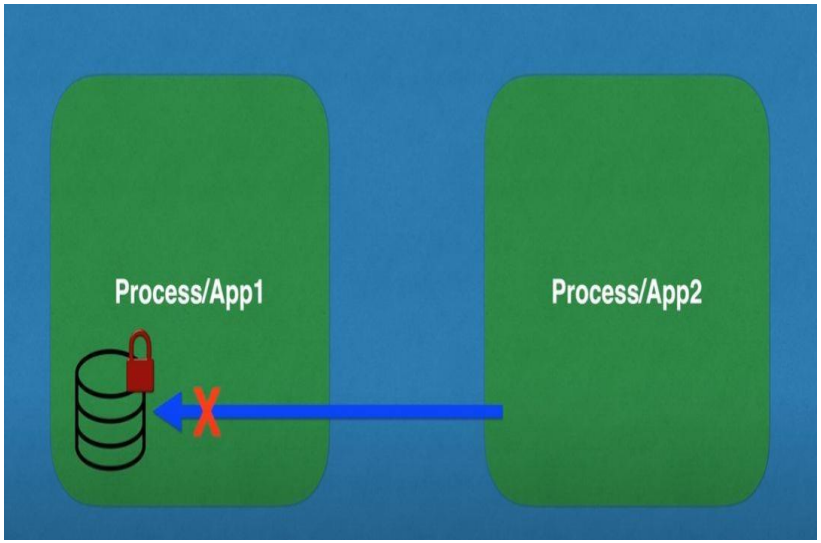
Content providers let you centralize content in one place and have many different applications access it as needed.

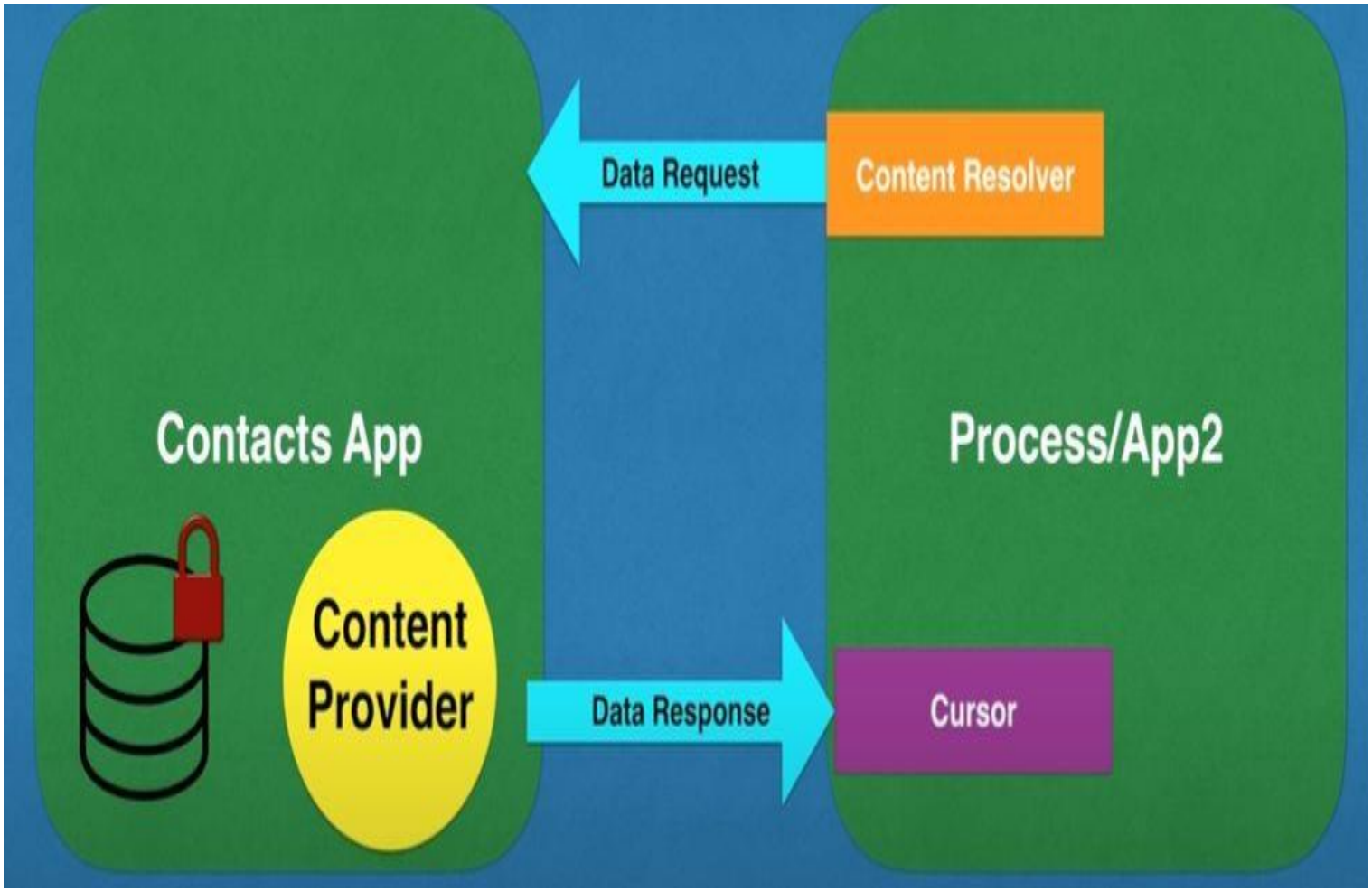
A content provider can be used to manage access to a variety of data storage sources, including both structured data, such as a SQLite relational database, or unstructured data such as image files.

A content provider behaves very much like a database where you can query it, edit its content, as well as add or delete content using insert(), update(), delete(), and query() methods.

In most cases this data is stored in an **SQlite** database.


Process/App1
Process/App2

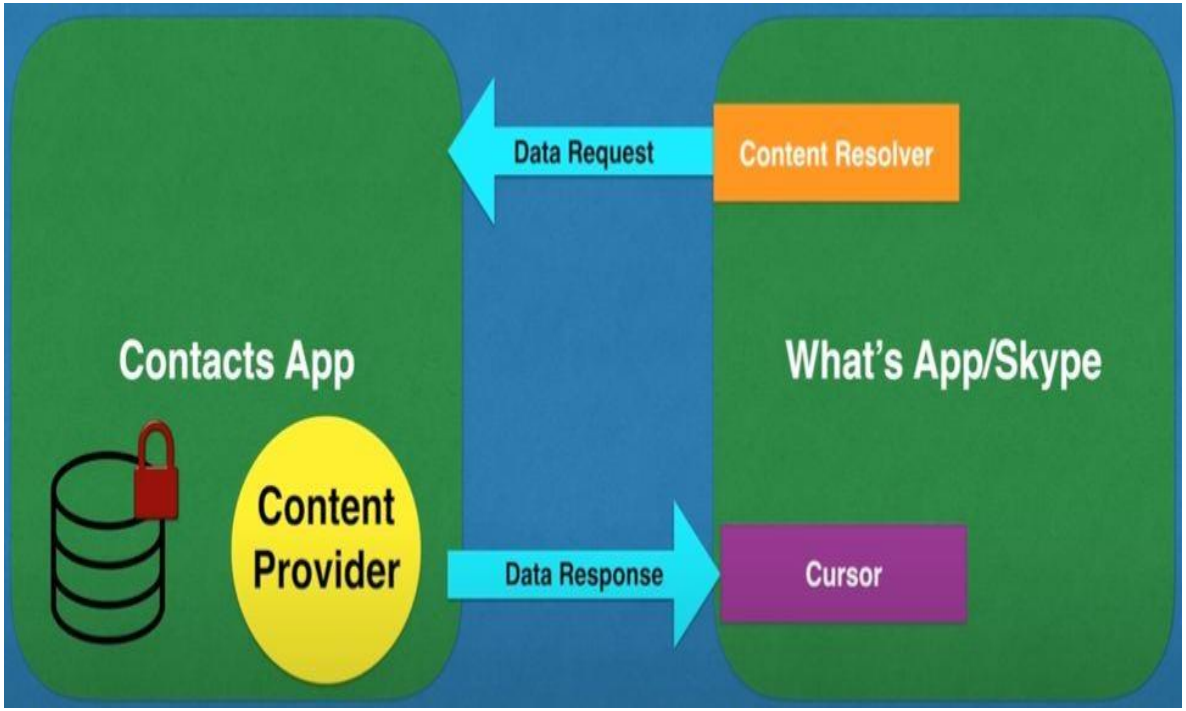

Process/App1
Process/App2
X



The ContentResolver object sends requests (like create, read, update, and delete) to the **ContentProvider** as a client.

After receiving a request, ContentProvider process it and returns the desired result.

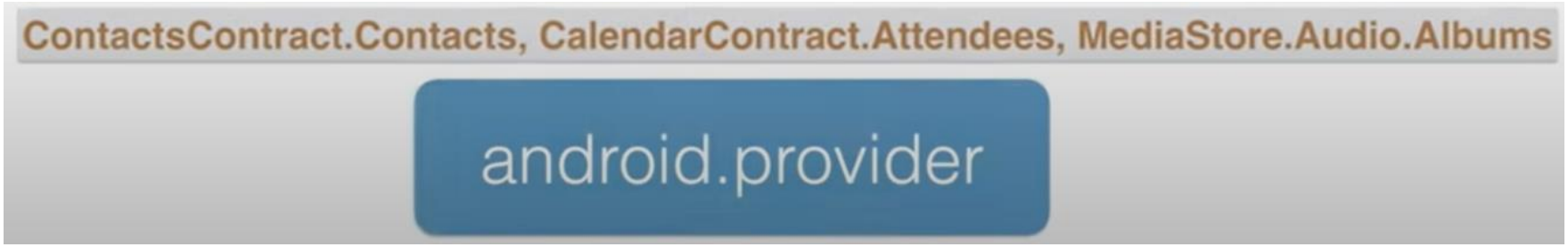
Example of ContentProvider



Content URIs

Content URI(Uniform Resource Identifier) is the key concept of Content providers.
To access the data from a content provider, URI is used as a query string.

Example



The Android framework includes content providers that manage data such as audio, video, images, and personal contact information.
You can see some of them listed in the reference documentation for the android.provider package.

- Create an android application that reads all contacts stored in device using content provider.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.contentproviderdemo">
<uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.WRITE_CONTACTS"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ContentProviderDemo">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="20dp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scrollbars="vertical" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Get Contacts"
        android:onClick="GetContact"/>

</LinearLayout>
```

MainActivity.java

```java
package com.example.contentproviderdemo;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import android.Manifest;
import android.annotation.SuppressLint;
import android.content.ContentResolver;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.text.method.ScrollingMovementMethod;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    TextView tv;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tv=findViewById(R.id.textview);
    }
    @SuppressLint("Range")
    public void GetContact(View view) {
        if(ContextCompat.checkSelfPermission(this, Manifest.permission.READ_CONTACTS)!=
                PackageManager.PERMISSION_GRANTED)
        {
            ActivityCompat.requestPermissions(this,new String[]
{Manifest.permission.READ_CONTACTS},10);
        }
        ContentResolver contentResolver=getContentResolver();
        Uri uri= ContactsContract.CommonDataKinds.Phone.CONTENT_URI;

        Cursor cursor=contentResolver.query(uri,null,null,null,null);
```

```
if(cursor.getCount()>0)
        {
            while (cursor.moveToNext())
            {
                String contactName;
                contactName =
cursor.getString(cursor.getColumnIndex(ContactsContract.CommonDataKind
s.Phone.DISPLAY_NAME));
                String
contactNumber=cursor.getString(cursor.getColumnIndex(ContactsContract.
CommonDataKinds.Phone.NUMBER));
                //Log.i("Contact",contactName+contactNumber);
                tv.append("Name - "+contactName + "     Number -
"+contactNumber+"\n");
            }
           tv.setMovementMethod(new ScrollingMovementMethod());
        }
        else Toast.makeText(this, "No contacts in device",
Toast.LENGTH_SHORT).show();
    }
}
```

ContactsContract.CommonDataKinds.Phone

- public static final class
- A datakind representing a telephone number.

CONTENT_URI

- field
- public static final Uri CONTENT_URI
- The content:// style URI for all data records of the CONTENT_ITEM_TYPE MIME type, combined with the associated raw contact and aggregate contact data.
- CONTENT_ITEM_TYPE - MIME type used when storing this in data table