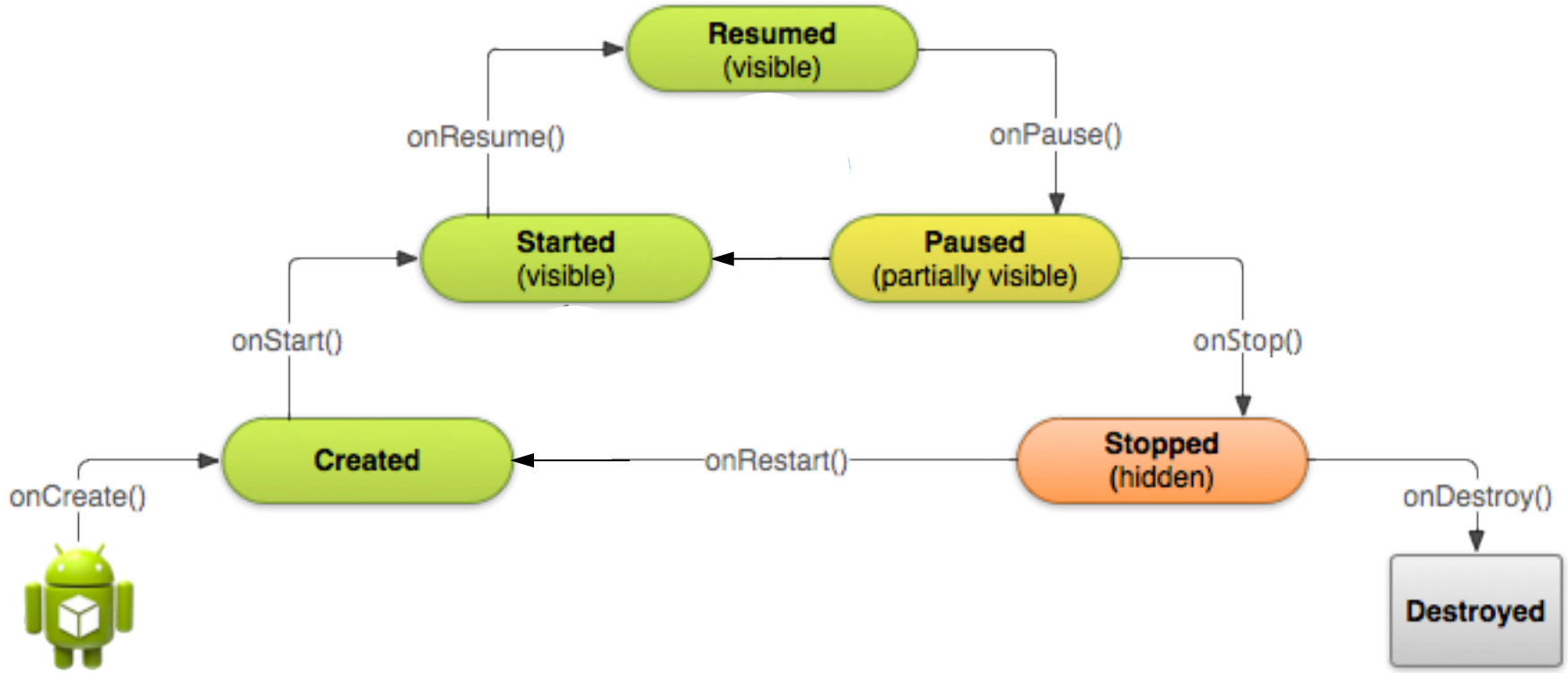
Broadcast Receivers
Content Providers

Activity Lifecycle



Broadcast Receiver

1. A subclass of BroadcastReceiver
   - Need to override onReceive(<context>, <intent>)
   - Can be defined as an anonymous class, or an inner class, or a public class
2. Register the receiver for interested broadcast using Intent Filter
   - Two methods
     - Java
     - XML (Manifest file)

Define BroadcastReceiver

- Anonymous class
  - BroadcastReceiver r = new BrodcastReceier() {
    ```
    @Override
    public void onReceive(Context c, Intent i) {
        <my work to do>
    }
    ```
    }
- Inner class
  - public class MainActivity extends AppCompatActivity {
    ```
    private  MyReceiver r = new MyReceiver();
    public class MyReceiver extends BroadcastRecever {
        @Override
        public void onReceive(..) {..}
    }
    ```
    }
- Public class
  - public class myReceiver extends BroadcastRecever {
    ```
    @Override
    public void onReceive(..) {..}
    ```
    }

Register Receiver

- Java
  - IntentFilter f=new IntentFilter(Intent.<ACTION>);
  - registerReceiver(<receiver>, <filter>) // onResume
  - unregisterReceiver(<receiver>) // onPause
- Manifest
  - Create a public receiver class
  - <receiver android:name=".<receiver class>" android:exported="true">
    <intent-filter>
    <action android:name="android.intent.action.*action*" />
    </intent-filter>
    </receiver>

Use broadcast for intra-App communication

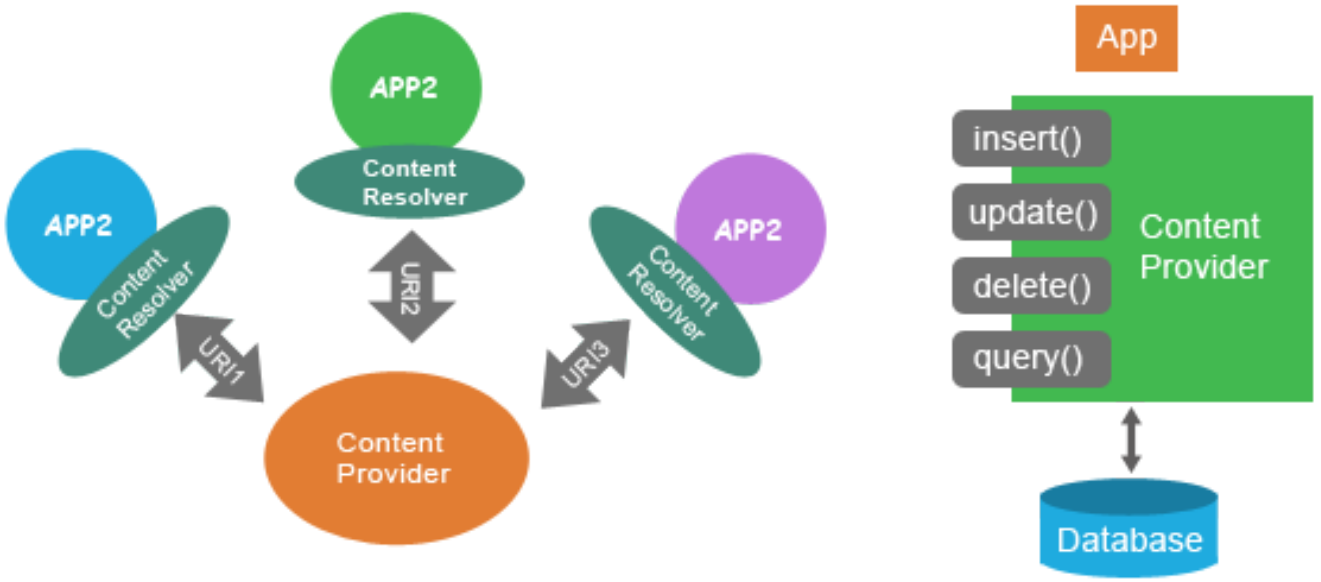
- Send a custom broadcast
  - Intent i= new Intent();
    i.setAction(<myaction>);
    i.putExtra(..);
    sendBroadcast(i);
- Receive a custom broadcast
  - either Java or Manifest
  - use <myaction>

Actions

- <SDK>/platforms/data/broadcast_actions.txt
- android.intent.action.AIRPLANE_MODE
- android.intent.action.BATTERY_LOW
- android.intent.action.TIME_TICK
- android.intent.action.USER_UNLOCKED
- android.hardware.action.NEW_PICTURE
- android.hardware.action.NEW_VIDEO
- android.intent.action.ACTION_POWER_CONNECTED
- android.intent.action.ACTION_POWER_DISCONNECTED
- android.intent.action.DATA_SMS_RECEIVED
- android.intent.action.NEW_OUTGOING_CALL
- android.net.wifi.WIFI_STATE_CHANGED

Content Provider

- Manages a structured data set
- Provide interfaces to connect to data in other process
- Client uses ContentResolver object to interacts with an instance of ContentProvider
- ContentProvider receive request, perform action, returns the result

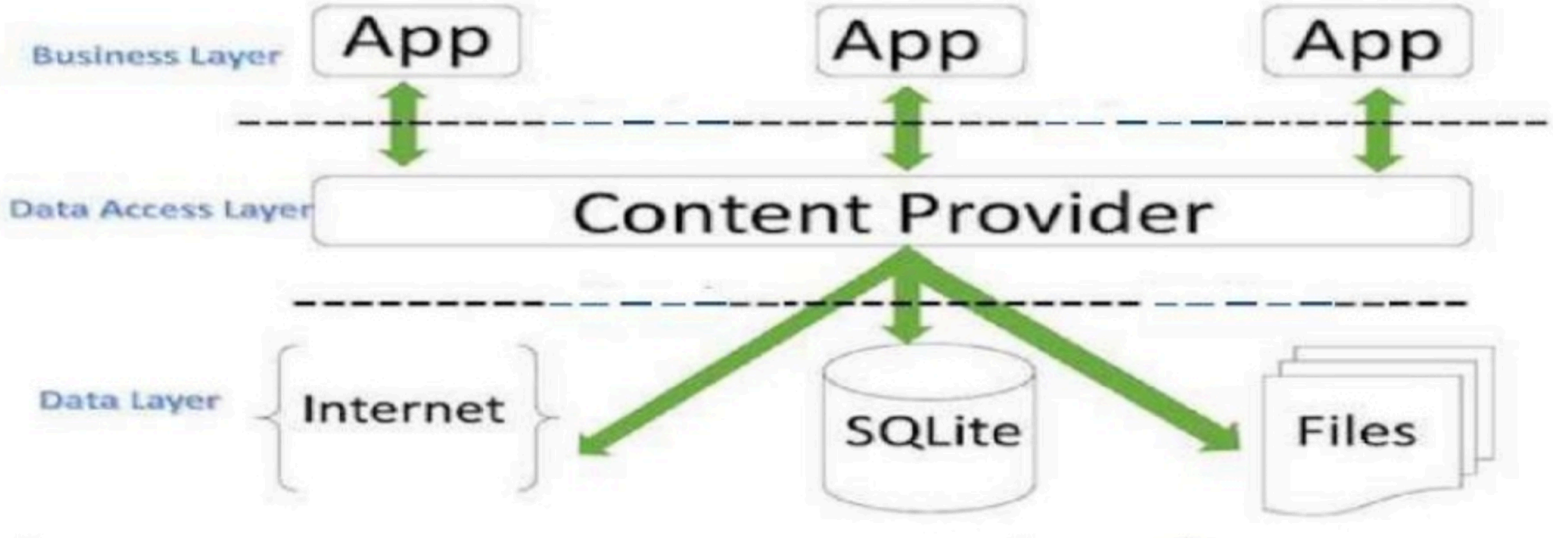


Default Content Providers

- Contacts – Contact details
- Call log – All call history
- Media Store – Audio/Video details
- Browser – Bookmarks, history, etc.
- Settings – All phone settings like Wi-Fi, Bluetooth, Security, etc.

Default Content Providers

Content Provider	Intended Data
Browser	Browser bookmarks, browser history, etc.
CallLog	Missed calls, call details, etc.
Contacts	Contact details
MediaStore	Media files such as audio, video and images
Settings	Device settings and preferences



Accessing Contacts & SMS

- Get Content Provider
  - getContentResolver()
- Get Contacts with cursor
  - Cursor c = getContentResolver().query(<uri>, nullx4)
  - c.getString(c.getColumnIndex(<col-name>))
- Iteration
  - if(<cur>) while (<cur>.moveToNext()) {..};
- Contacts
  - URI = ContactsContract.Contacts.CONTENT_URI, or content://com.android.contacts/contacts
  - <col-name>: ContactsContract.PhoneLookup.DISPLAY_NAME
  - permission: android.permission.READ_CONTACTS
- SMS
  - URI = Telephony.Sms.CONTENT_URI (content://sms)
  - <col-name>: Telephony.Sms.BODY
  - Only after KITKAT
    - android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODE.KITKAT
  - permission: android.permission.READ_SMS

Storages

- Shared Preferences
  - Store private primitive data in key-value pairs.
- Internal Storage
  - Store private data on the device memory.
- External Storage
  - Store public data on the shared external storage.
- SQLite Databases
  - Store structured data in a private database.
- Network Connection
  - Store data on the web with your own network server.

Shared Preferences

- Used to store simple state information to keep across runs
  - ex: Keep App settings/preferences, keep user input values
- (key, value)
- Saved in XLM file
- Persistent
- Stored in App's sandbox
- Not for sensitive information
- SharedPreference instance is automatically created in every APp

SharedPreferences

- Get an existing SharedPreference, or create one otherwise
  - SharedPreferences <pref> = getSharedPreferences(<pref_name>, <mode>)
    - <mode>=MODE_{PRIVATE|APPEND|..}
- pref.getString(<key>, <default>)
  - Get the value for <key>, or get <default> if not exists yet
- Save a key/value
  - SharedPreferences.Editor ed = <pref>.edit();
    ed.put{String|Int|Bool|..} (<key>, <val>)
    ed.commit();
- Check pref XML file (use Android Device Monitor)
  - /data/data/<pkg>/shared_prefs/<pref_name>.xml

SQLite

- Create a database (or open if exists)
  - *SQLiteDatabase db = openOrCreateDatabase(<db_name>, <mode>, <cursor>)*
- Run SQL
  - *db.execSQL(<sql>)*
  - *http://www.w3schools.com/sql/*
- Run SQL with cursor
  - *Cursor <cur> = db.rawQuery(<SQL>, null)*
  - *<cur>.moveToFirst()*
  - *<cur>.getString(<cur>.getColumnIndex(<col-name>))*
  - *do {..} while(<cur>.moveToNext());*
- Create a table
  - *create table if not exists <table-name> (<colname> <type>, ..)*
- Insert values
  - *insert into <table> values*
- Close the database
  - *db.close*
- Stored in
  - /data/data/<pkg>/databases/<db-name>