Overview of Android: Key App Components

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

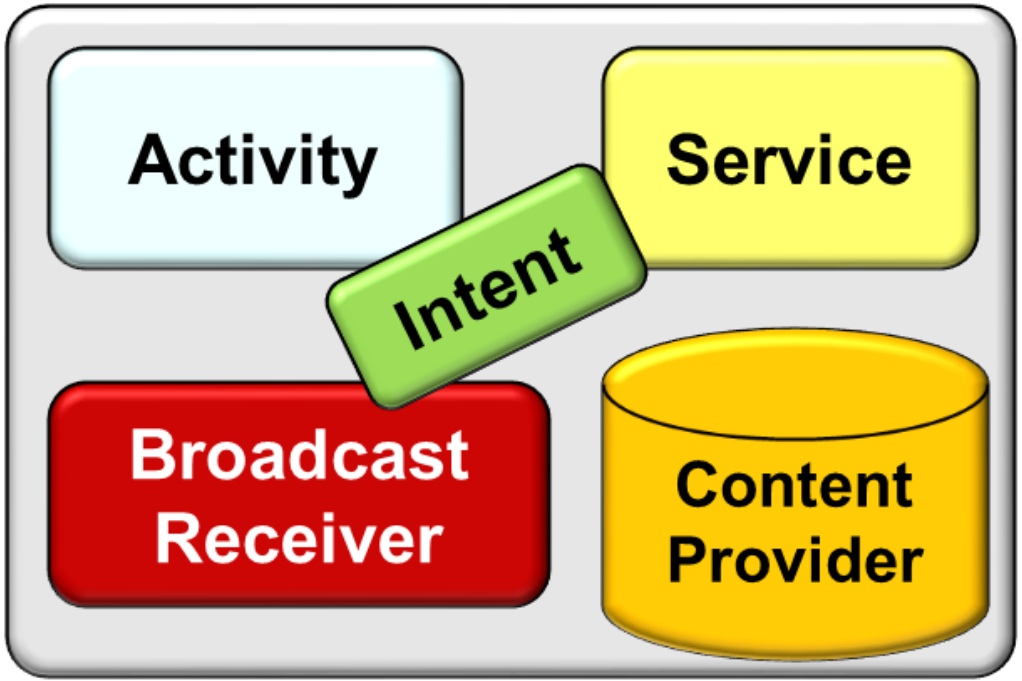
Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

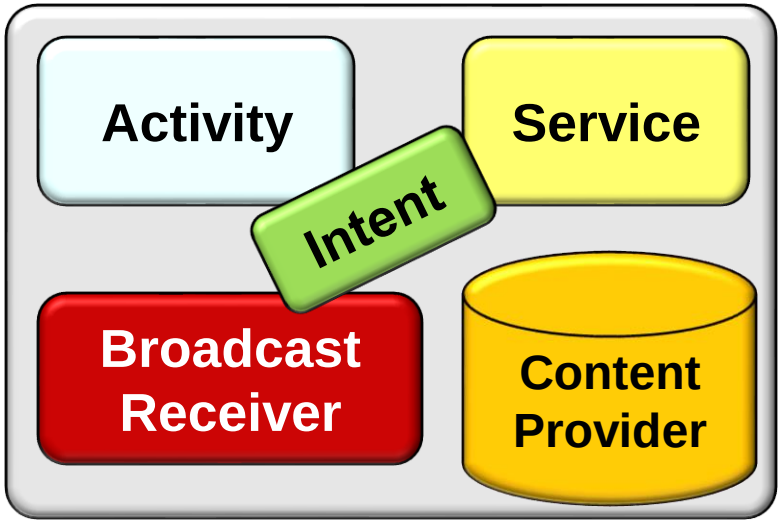
1. Understand key elements in Android's middleware infrastructure
2. Name all the key app components in Android



Overview of Key Android App Components

Overview of Key Android App Components

- App components are essential building blocks of mobile apps that provide various hooks via which Android can effect an app's lifecycle



See developer.android.com/guide/components/fundamentals.html#Components

Overview of Key Android App Components

- **Intent**
  - A message that describes an action to perform or an event that has occurred

Intent

Added in API level 1

Summary: Nested Classes | Constants | Inherited Constants | Fields | Ctors | Methods | Inherited Methods | [Expand All]

```
public class Intent
extends Object implements Parcelable, Cloneable
```

java.lang.Object

↳ android.content.Intent

Known Direct Subclasses

LabeledIntent

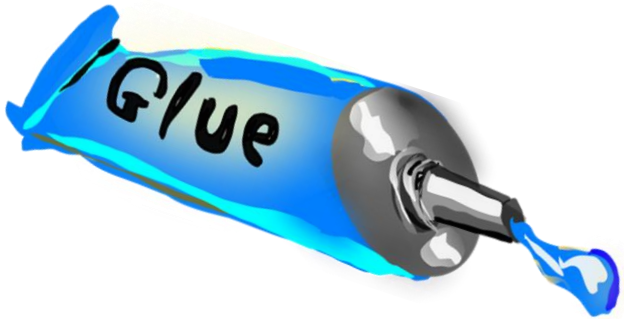
An intent is an abstract description of an operation to be performed. It can be used with `startActivity` to launch an `Activity`, `broadcastIntent` to send it to any interested `BroadcastReceiver` components, and `startService(Intent)` or `bindService(Intent, ServiceConnection, int)` to communicate with a background `Service`.

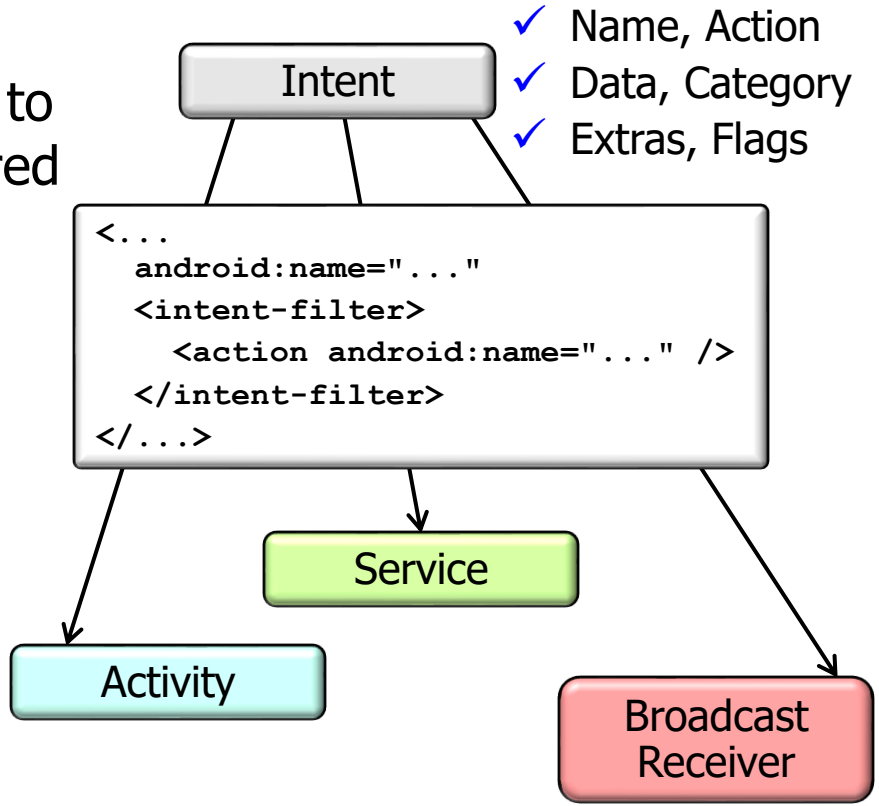
An Intent provides a facility for performing late runtime binding between the code in different applications. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed.

See developer.android.com/reference/android/content/Intent.html

Overview of Key Android App Components

- **Intent**
  - A message that describes an action to perform or an event that has occurred
    - Android uses intents as "glue" to simplify the integration of apps that reuse existing components





Overview of Key Android App Components

- **Activity**
  - Provides a screen within which users can interact to do a single focused thing

Added in API level 1

Summary: Constants | Inherited Constants | Fields | Ctors | Methods | Protected Methods | Inherited Methods | [Expand All]

Activity

```
public class Activity
extends ContextThemeWrapper implements LayoutInflater.Factory2, Window.Callback,
KeyEvent.Callback, View.OnCreateContextMenuListener, ComponentCallbacks2
```

java.lang.Object
↳ android.content.Context
↳ android.content.ContextWrapper
↳ android.view.ContextThemeWrapper
↳ android.app.Activity

Known Direct Subclasses

AccountAuthenticatorActivity, ActivityGroup, AliasActivity, ExpandableListActivity, FragmentActivity, ListActivity, NativeActivity

Known Indirect Subclasses

ActionBarActivity, AppCompatActivity, LauncherActivity, PreferenceActivity, TabActivity

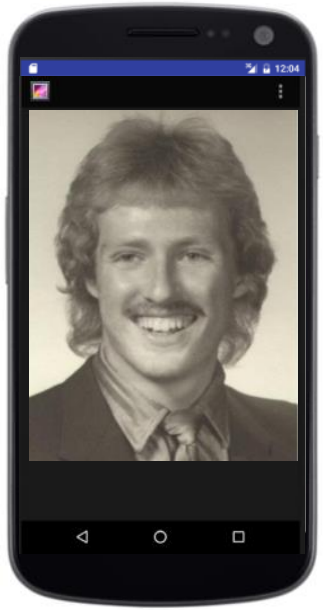
An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with `setContentView(View)`. While activities are often presented to the user as full-screen windows, they can also be used in other ways: as floating windows (via a theme with `windowIsFloating` set) or embedded inside of another activity (using `ActivityGroup`). There are two methods almost all subclasses of Activity will implement:

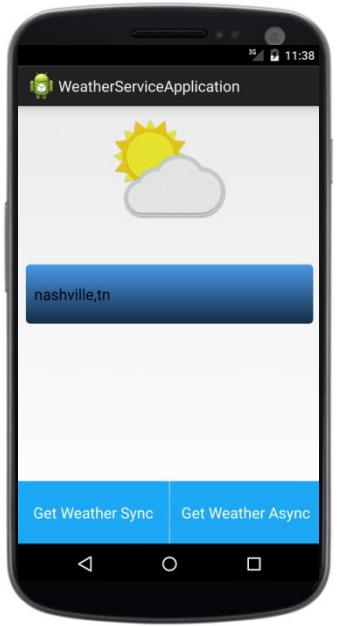
- `onCreate(Bundle)` is where you initialize your activity. Most importantly, here you will usually call `setContentView(int)` with a layout resource defining your UI, and using `findViewById(int)` to retrieve the widgets in that UI that you need to interact with programmatically.
- `onPause()` is where you deal with the user leaving your activity. Most importantly, any changes made by the user should at this point be committed (usually to the `ContentProvider` holding the data).

See developer.android.com/reference/android/app/Activity.html

Overview of Key Android App Components

- **Activity**
  - Provides a screen within which users can interact to do a single focused thing
    - It's the most common type of Android component

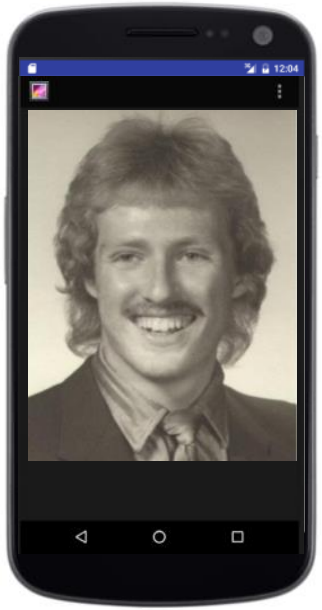


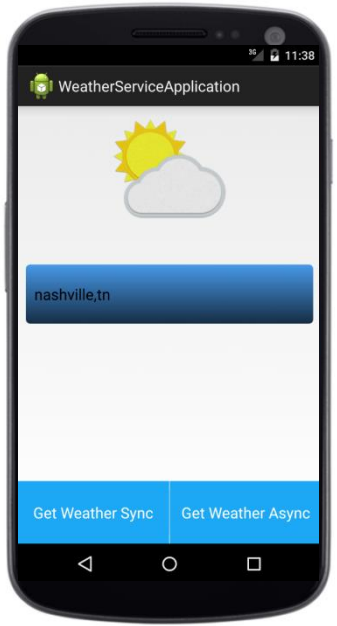


See developer.android.com/guide/components/activities.html

Overview of Key Android App Components

- **Activity**
  - Provides a screen within which users can interact to do a single focused thing
    - It's the most common type of Android component

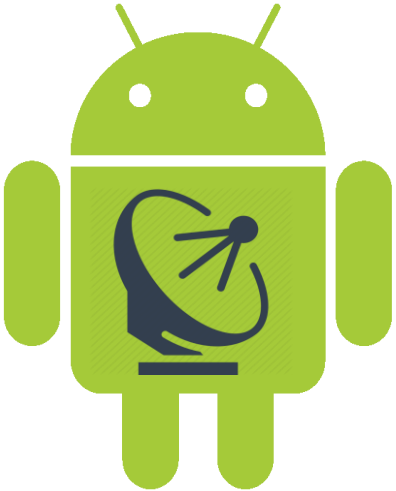




Activities are at the heart of all Android apps

Overview of Key Android App Components

- **Broadcast Receiver**
  - Event handler that can respond to system-wide broadcasts



BroadcastReceiver

Added in API level 1

Summary: Nested Classes | Ctors | Methods | Inherited Methods | [Expand All]

```
public abstract class BroadcastReceiver
extends Object
```

java.lang.Object
↳ android.content.BroadcastReceiver

Known Direct Subclasses

AppWidgetProvider, DeviceAdminReceiver, MediaButtonReceiver, RestrictionsReceiver, WakefulBroadcastReceiver

Base class for code that will receive intents sent by sendBroadcast().

If you don't need to send broadcasts across applications, consider using this class with `LocalBroadcastManager` instead of the more general facilities described below. This will give you a much more efficient implementation (no cross-process communication needed) and allow you to avoid thinking about any security issues related to other applications being able to receive or send your broadcasts.

You can either dynamically register an instance of this class with `Context.registerReceiver()` or statically publish an implementation through the `<receiver>` tag in your `AndroidManifest.xml`.

Note: If registering a receiver in your `Activity.onResume()` implementation, you should unregister it in `Activity.onPause()`. (You won't receive intents when paused, and this will cut down on unnecessary system overhead). Do not unregister in `Activity.onSaveInstanceState()`, because this won't be called if the user moves back in the history stack.

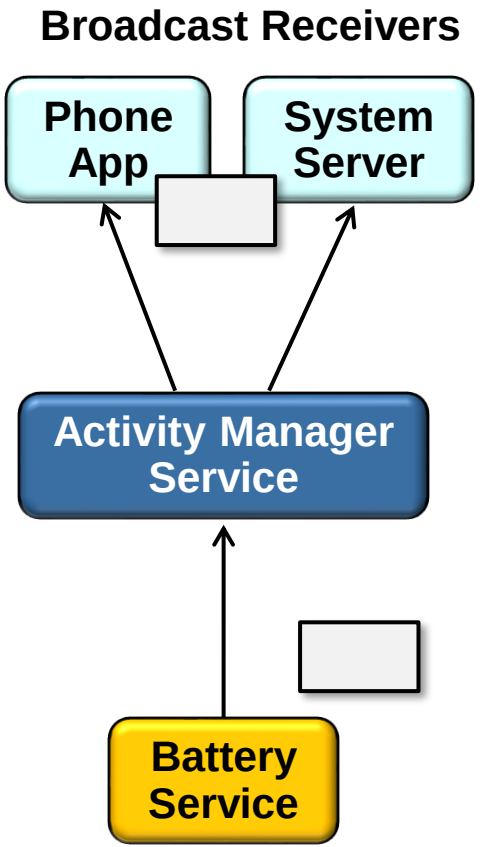
There are two major classes of broadcasts that can be received:

- **Normal broadcasts** (sent with `Context.sendBroadcast`) are completely asynchronous. All receivers of the broadcast are run in an undefined order, often at the same time. This is more efficient, but means that receivers cannot use the result or abort APIs included here.
- **Ordered broadcasts** (sent with `Context.sendOrderedBroadcast`) are delivered to one receiver at a time. As each receiver executes in turn, it can propagate a result to the next receiver, or it can completely abort the broadcast so that it won't be passed to other receivers. The order receivers run in can be controlled with the `android:priority` attribute of the matching intent-filter; receivers with the same priority will be run in an arbitrary order.

See developer.android.com/reference/android/content/BroadcastReceiver.html

Overview of Key Android App Components

- **Broadcast Receiver**
  - Event handler that can respond to system-wide broadcasts
    - Activities & services in 1+ apps can interact by broadcasting intents handled by 0+ broadcast receivers



Overview of Key Android App Components

- **Service**
  - Runs in background to perform long-running operations or access remote resources

public abstract class

Summary: Constants | Inherited Constants | Ctors | Methods | Protected Methods | Inherited Methods | [Expand All]
Added in API level 1

Service

extends ContextWrapper
implements ComponentCallbacks2

java.lang.Object
↳android.content.Context
↳android.content.ContextWrapper
↳android.app.Service

▸ Known Direct Subclasses
AbstractInputMethodService, AccessibilityService, DreamService, HostApduService, IntentService, MediaRouteProviderService, NotificationCompatSideChannelService, NotificationListenerService, OffHostApduService, PrintService, RecognitionService, RemoteViewsService, SettingInjectorService, SpellCheckerService, TextToSpeechService, VpnService, WallpaperService

▸ Known Indirect Subclasses
InputMethodService

Class Overview

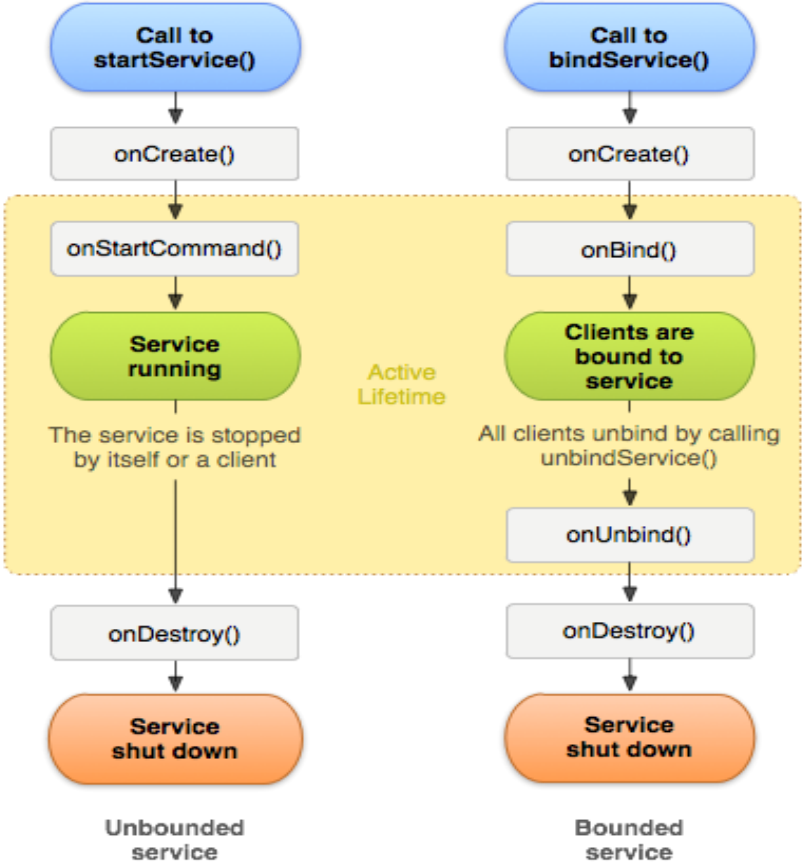
A Service is an application component representing either an application's desire to perform a longer-running operation while not interacting with the user or to supply functionality for other applications to use. Each service class must have a corresponding `<service>` declaration in its package's `AndroidManifest.xml`. Services can be started with `Context.startService()` and `Context.bindService()`.

Note that services, like other application objects, run in the main thread of their hosting process. This means that, if your service is going to do any CPU intensive (such as MP3 playback) or blocking (such as networking) operations, it should spawn its own thread in which to do that work. More information on this can be found in Processes and Threads. The `IntentService` class is available as a standard implementation of Service that has its own thread where it schedules its work to be done.

See developer.android.com/reference/android/app/Service.html

Overview of Key Android App Components

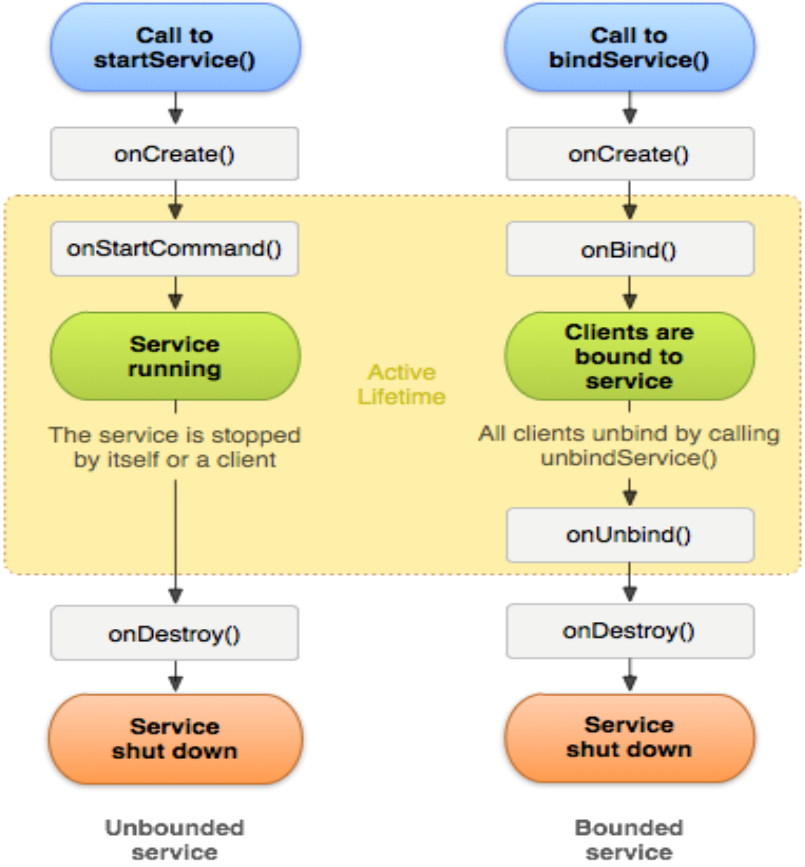
- **Service**
  - Runs in background to perform long-running operations or access remote resources
    - A bound service has a lifecycle that depends on its creating component(s)



See developer.android.com/guide/components/services.html

Overview of Key Android App Components

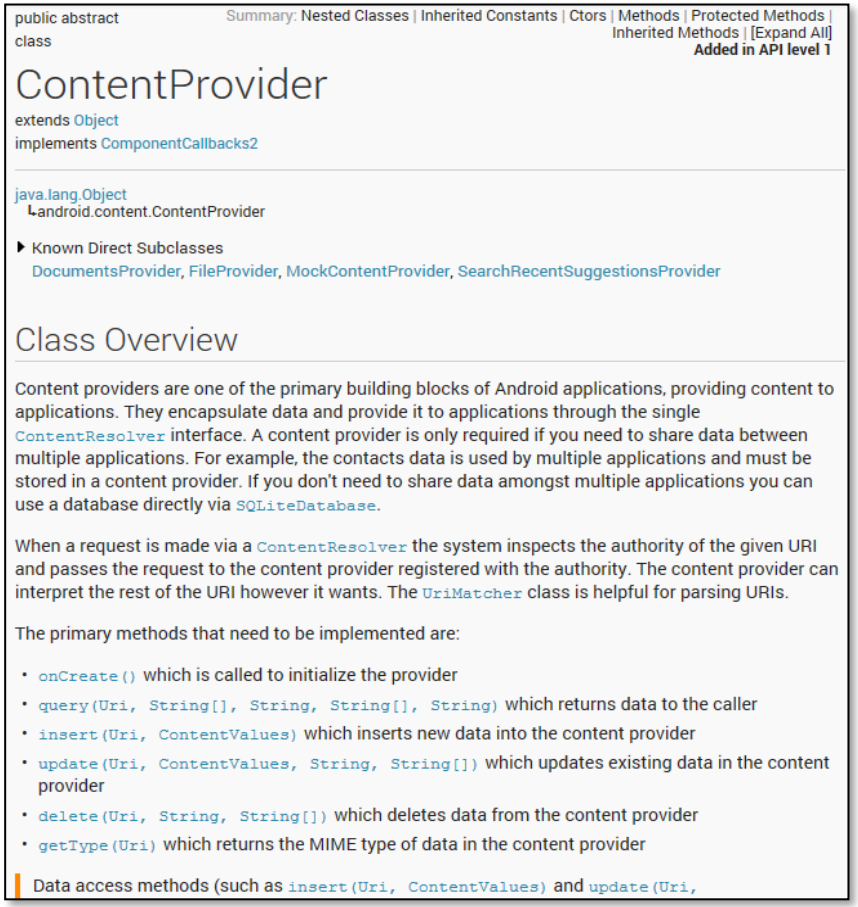
- **Service**
  - Runs in background to perform long-running operations or access remote resources
    - A bound service has a lifecycle that depends on its creating component(s)
    - A started service's lifecycle doesn't depend on its creating component



Started services are deprecated in later versions of Android

Overview of Key Android App Components

- **Content Provider**
  - Manages access to structured data & provides data security mechanisms



public abstract class

Summary: Nested Classes | Inherited Constants | Ctors | Methods | Protected Methods | Inherited Methods | [Expand All]
Added in API level 1

ContentProvider

extends Object
implements ComponentCallbacks2

java.lang.Object
↳android.content.ContentProvider

▸ Known Direct Subclasses
DocumentsProvider, FileProvider, MockContentProvider, SearchRecentSuggestionsProvider

Class Overview

Content providers are one of the primary building blocks of Android applications, providing content to applications. They encapsulate data and provide it to applications through the single `ContentResolver` interface. A content provider is only required if you need to share data between multiple applications. For example, the contacts data is used by multiple applications and must be stored in a content provider. If you don't need to share data amongst multiple applications you can use a database directly via `SQLiteDatabase`.

When a request is made via a `ContentResolver` the system inspects the authority of the given URI and passes the request to the content provider registered with the authority. The content provider can interpret the rest of the URI however it wants. The `UriMatcher` class is helpful for parsing URIs.

The primary methods that need to be implemented are:

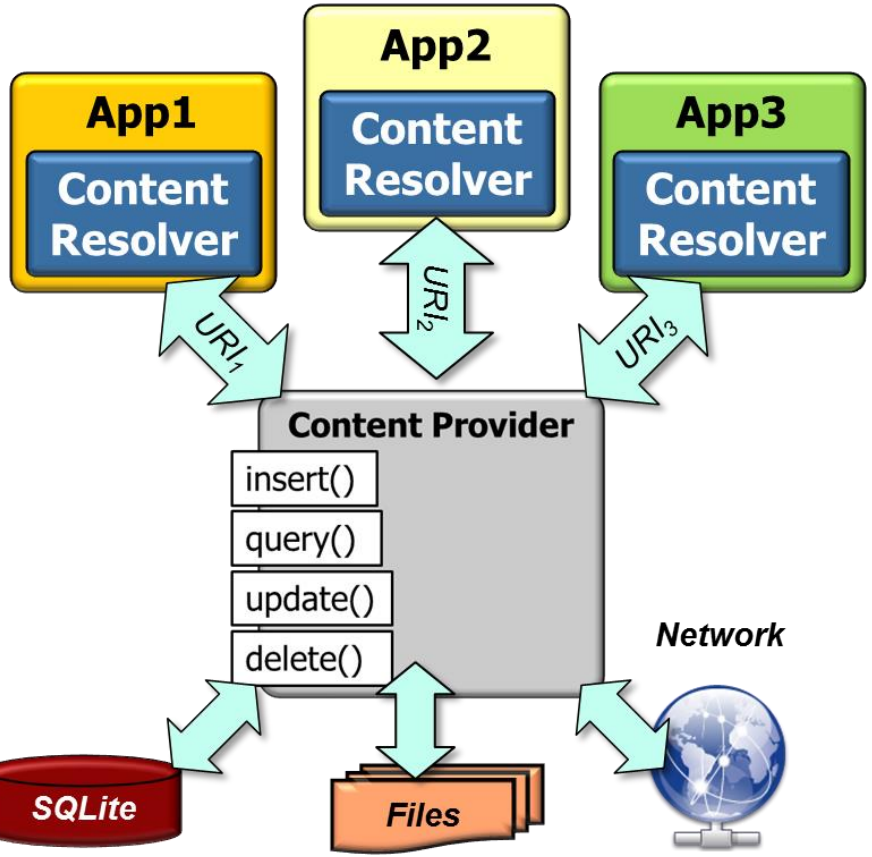
- `onCreate()` which is called to initialize the provider
- `query(Uri, String[], String, String[], String)` which returns data to the caller
- `insert(Uri, ContentValues)` which inserts new data into the content provider
- `update(Uri, ContentValues, String, String[])` which updates existing data in the content provider
- `delete(Uri, String, String[])` which deletes data from the content provider
- `getType(Uri)` which returns the MIME type of data in the content provider

Data access methods (such as `insert(Uri, ContentValues)` and `update(Uri,`

See developer.android.com/reference/android/content/ContentProvider.html

Overview of Key Android App Components

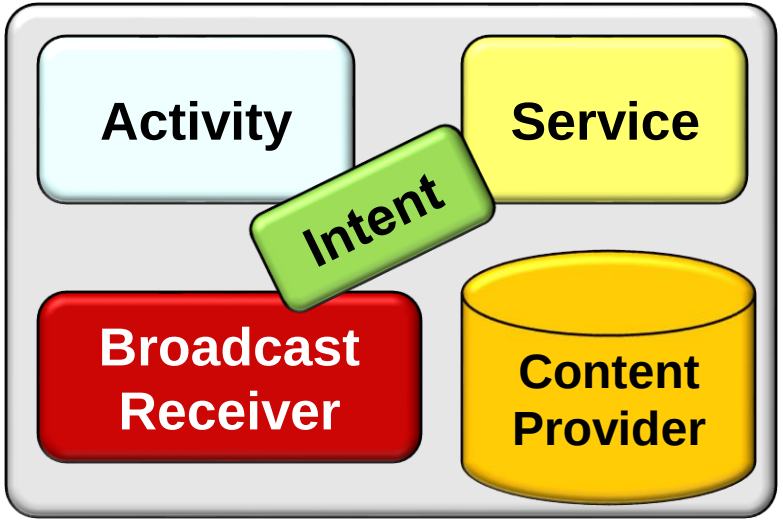
- **Content Provider**
  - Manages access to structured data & provides data security mechanisms
    - Content Providers/Resolvers can make an app's data available to other apps



See developer.android.com/guide/topics/providers/content-providers.html

Overview of Key Android App Components

- We cover intents, activities, & broadcast receivers in MOOC 2 & services & content providers in MOOC 3 of the *Android App Development* specialization



See www.coursera.org/specializations/android-app-development

End of Overview of Android: Key App Components