**Intent**

**An intent is an abstract description of an operation to be performed. It can be used with** [**startActivity**](https://developer.android.com/reference/android/content/Context#startActivity(android.content.Intent)) **to launch an** [**Activity**](https://developer.android.com/reference/android/app/Activity)**,** [**broadcastIntent**](https://developer.android.com/reference/android/content/Context#sendBroadcast(android.content.Intent)) **to send it to any interested** [**BroadcastReceiver**](https://developer.android.com/reference/android/content/BroadcastReceiver) **components, and** [**Context.startService(Intent)**](https://developer.android.com/reference/android/content/Context#startService(android.content.Intent)) **or** [**Context.bindService(Intent, BindServiceFlags, Executor, ServiceConnection)**](https://developer.android.com/reference/android/content/Context#bindService(android.content.Intent,%20android.content.Context.BindServiceFlags,%20java.util.concurrent.Executor,%20android.content.ServiceConnection)) **to communicate with a background** [**Service**](https://developer.android.com/reference/android/app/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.**

### **Intent Structure**

**The primary pieces of information in an intent are:**

* **action -- The general action to be performed, such as** [**ACTION\_VIEW**](https://developer.android.com/reference/android/content/Intent#ACTION_VIEW)**,** [**ACTION\_EDIT**](https://developer.android.com/reference/android/content/Intent#ACTION_EDIT)**,** [**ACTION\_MAIN**](https://developer.android.com/reference/android/content/Intent#ACTION_MAIN)**, etc.**
* **data -- The data to operate on, such as a person record in the contacts database, expressed as a** [**Uri**](https://developer.android.com/reference/android/net/Uri)**.**

### **Some examples of action/data pairs are:**

* [ACTION\_VIEW](https://developer.android.com/reference/android/content/Intent#ACTION_VIEW) **content://contacts/people/1 -- Display information about the person whose identifier is "1".**
* [ACTION\_DIAL](https://developer.android.com/reference/android/content/Intent#ACTION_DIAL) **content://contacts/people/1 -- Display the phone dialer with the person filled in.**
* [ACTION\_VIEW](https://developer.android.com/reference/android/content/Intent#ACTION_VIEW) **tel:123 -- Display the phone dialer with the given number filled in. Note how the VIEW action does what is considered the most reasonable thing for a particular URI.**
* [ACTION\_DIAL](https://developer.android.com/reference/android/content/Intent#ACTION_DIAL) **tel:123 -- Display the phone dialer with the given number filled in.**
* [ACTION\_EDIT](https://developer.android.com/reference/android/content/Intent#ACTION_EDIT) **content://contacts/people/1 -- Edit information about the person whose identifier is "1".**
* [ACTION\_VIEW](https://developer.android.com/reference/android/content/Intent#ACTION_VIEW) **content://contacts/people/ -- Display a list of people, which the user can browse through. This example is a typical top-level entry into the Contacts application, showing you the list of people. Selecting a particular person to view would result in a new intent {** [ACTION\_VIEW](https://developer.android.com/reference/android/content/Intent#ACTION_VIEW) **content://contacts/people/N } being used to start an activity to display that person.**

In addition to these primary attributes, there are a number of secondary attributes that you can also include with an intent:

**Activity Explanation:**

* **User Interface (UI): An activity typically corresponds to a single screen with a user interface. It can contain various UI elements such as buttons, text fields, images, etc.**
* **Lifecycle Methods: Activities have a lifecycle that defines different states they go through, from creation to destruction. Developers can override specific methods to perform actions at different points in the lifecycle (e.g., when the activity is created, started, resumed, paused, stopped, or destroyed).**
* **Intent: Activities can be started by the Android system or other activities using an "Intent," which is a messaging object that can carry data between components.**

**The Activity Lifecycle and Fragment Lifecycle are important concepts in Android development. They describe the different states that an activity or fragment goes through during its existence, allowing developers to manage the behavior and appearance of their app's UI effectively. Let's look at each of them separately:**

* **category** -- Gives additional information about the action to execute. For example, [CATEGORY\_LAUNCHER](https://developer.android.com/reference/android/content/Intent#CATEGORY_LAUNCHER) means it should appear in the Launcher as a top-level application, while [CATEGORY\_ALTERNATIVE](https://developer.android.com/reference/android/content/Intent#CATEGORY_ALTERNATIVE) means it should be included in a list of alternative actions the user can perform on a piece of data.
* **type** -- Specifies an explicit type (a MIME type) of the intent data. Normally the type is inferred from the data itself. By setting this attribute, you disable that evaluation and force an explicit type.
* **component** -- Specifies an explicit name of a component class to use for the intent. Normally this is determined by looking at the other information in the intent (the action, data/type, and categories) and matching that with a component that can handle it. If this attribute is set then none of the evaluation is performed, and this component is used exactly as is. By specifying this attribute, all of the other Intent attributes become optional.
* **extras** -- This is a [Bundle](https://developer.android.com/reference/android/os/Bundle) of any additional information. This can be used to provide extended information to the component. For example, if we have a action to send an e-mail message, we could also include extra pieces of data here to supply a subject, body, etc.

****

**Activity Lifecycle:**

### **onCreate()**

**You must implement this callback, which fires when the system creates your activity. Your implementation should initialize the essential components of your activity: For example, your app should create views and bind data to lists here. Most importantly, this is where you must call [setContentView()](https://developer.android.com/reference/android/app/Activity" \l "setContentView(android.view.View)) to define the layout for the activity's user interface.**

**When [onCreate()](https://developer.android.com/reference/android/app/Activity" \l "onCreate(android.os.Bundle)) finishes, the next callback is always [onStart()](https://developer.android.com/reference/android/app/Activity" \l "onStart()).**

### **onStart()**

**As [onCreate()](https://developer.android.com/reference/android/app/Activity" \l "onCreate(android.os.Bundle)) exits, the activity enters the Started state, and the activity becomes visible to the user. This callback contains what amounts to the activity’s final preparations for coming to the foreground and becoming interactive.**

### **onResume()**

**The system invokes this callback just before the activity starts interacting with the user. At this point, the activity is at the top of the activity stack, and captures all user input. Most of an app’s core functionality is implemented in the [onResume()](https://developer.android.com/reference/android/app/Activity" \l "onResume()) method.**

**The [onPause()](https://developer.android.com/reference/android/app/Activity" \l "onPause()) callback always follows [onResume()](https://developer.android.com/reference/android/app/Activity" \l "onResume()).**

### **onPause()**

**The system calls [onPause()](https://developer.android.com/reference/android/app/Activity" \l "onPause()) when the activity loses focus and enters a Paused state. This state occurs when, for example, the user taps the Back or Recents button. When the system calls [onPause()](https://developer.android.com/reference/android/app/Activity" \l "onPause()) for your activity, it technically means your activity is still partially visible, but most often is an indication that the user is leaving the activity, and the activity will soon enter the Stopped or Resumed state.**

**An activity in the Paused state may continue to update the UI if the user is expecting the UI to update. Examples of such an activity include one showing a navigation map screen or a media player playing. Even if such activities lose focus, the user expects their UI to continue updating.**

**You should not use [onPause()](https://developer.android.com/reference/android/app/Activity" \l "onPause()) to save application or user data, make network calls, or execute database transactions.**

**Once [onPause()](https://developer.android.com/reference/android/app/Activity" \l "onPause()) finishes executing, the next callback is either [onStop()](https://developer.android.com/reference/android/app/Activity" \l "onStop()) or [onResume()](https://developer.android.com/reference/android/app/Activity" \l "onResume()), depending on what happens after the activity enters the Paused state.**

### **onStop()**

**The system calls [onStop()](https://developer.android.com/reference/android/app/Activity" \l "onStop()) when the activity is no longer visible to the user. This may happen because the activity is being destroyed, a new activity is starting, or an existing activity is entering a Resumed state and is covering the stopped activity. In all of these cases, the stopped activity is no longer visible at all.**

**The next callback that the system calls is either [onRestart()](https://developer.android.com/reference/android/app/Activity" \l "onRestart()), if the activity is coming back to interact with the user, or by [onDestroy()](https://developer.android.com/reference/android/app/Activity" \l "onDestroy()) if this activity is completely terminating.**

### **onRestart()**

**The system invokes this callback when an activity in the Stopped state is about to restart. [onRestart()](https://developer.android.com/reference/android/app/Activity" \l "onRestart()) restores the state of the activity from the time that it was stopped.**

**This callback is always followed by [onStart()](https://developer.android.com/reference/android/app/Activity" \l "onStart()).**

### **onDestroy()**

**The system invokes this callback before an activity is destroyed.**

**This callback is the final one that the activity receives. [onDestroy()](https://developer.android.com/reference/android/app/Activity" \l "onDestroy()) is usually implemented to ensure that all of an activity’s resources are released when the activity, or the process containing it, is destroyed.**