Intents and Intent Filters

Alexander Nelson
September 18, 2017

University of Arkansas - Department of Computer Science and Computer Engineering
Intents
What are intents?

- Asynchronous messages
- Use to request an action
Intents

What are they used for?

1. Starting an activity
2. Starting a service
3. Delivering a broadcast
Starting an Activity

Start new instance of an Activity by passing intent to `startActivity()`

Intent describes the activity to start and any necessary data

Can call `startActivityForResult()` if you need a result

Receives a result in the form of a separate Intent object in `onActivityResult()`
Starting a Service

Started Service:
Pass an Intent to `startService()`

Bound Service:
Pass an Intent to `bindService()`

Intent describes the service to start and any necessary data
Deliver a broadcast to other apps by passing Intent to:

- `sendBroadcast()`
- `sendOrderedBroadcast()`
Two types of Intents

1. Explicit Intents
2. Implicit Intents
Explicit Intents

Specify the component to start by name

- Fully qualified class name

Commonly used to start a component within your own application
Do not name a specific component!
Declare a general **action**
Allows a component from another application to handle it

e.g. Show user location on a map, can use implicit intent to request a mapping app (Google, OpenStreetMap, etc...)
Using Intents

Example use of an Intent to start an Activity

1. startActivity()
2. Android System
3. onCreate()
How does the component know to start?

**Intent Filters**

Android matches against filters to determine implicit Intents

e.g.

```xml
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```
Explicit intents require the package and the class of the component to access.

E.g.

```java
// Executed in an Activity, so 'this' is the Context
// The fileUrl is a string URL, such as "http://www.example.com/image.png"
Intent downloadIntent = new Intent(this, DownloadService.class);
downloadIntent.setData(Uri.parse(fileUrl));
startService(downloadIntent);
```
Implicit intents instead specify an **Action** to be taken

```java
// Create the text message with a string
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(sendIntent);
}
```
Implicit Intents

It is possible that no applications can handle an implicit Intent. An app may also be unavailable because of permissions.

`resolveActivity(Intent)` – Function to check if an Intent can be handled.

If `resolveActivity(Intent)` returns null:

- Intent cannot be handled.
- Disable the feature which issues the Intent object.
// Create the text message with a string
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(sendIntent);
}
App Chooser

If only one Application can handle the Intent, it is launched immediately.

If multiple Applications can handle an Implicit Intent, the App Chooser is displayed.
Once chosen, this acts as the default action for that Intent.
A user may want the option of choosing every time e.g. `ACTION_SEND` is used for sharing, and may go to different social media accounts.

To show the chooser, create Intent using `createChooser()` Pass that Intent to `startActivity()` instead
Forcing App Chooser

```java
Intent sendIntent = new Intent(Intent.ACTION_SEND);
...

// Always use string resources for UI text.
// This says something like "Share this photo with"
String title = getResources().getString(R.string.chooser_title);
// Create intent to show the chooser dialog
Intent chooser = Intent.createChooser(sendIntent, title);

// Verify the original intent will resolve to at least one activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(chooser);
}
```
Building an Intent
Intent Object

Intent object carries information to the Android system so that it can determine what to start.

Contains:

- Component Name
- Action
- Data
- Category
- Extras
- Flags
This is an optional component of the Intent object

If included, the Intent is **Explicit**
If included, the Intent is delivered *only* to the specified component

If not included, the Intent is **implicit**
If not included, the Intent is delivered to the component determined by the system
Services should **always** specify the component name.
Otherwise, there is no way to be certain that a service will respond.
The Component Name field is populated by a ComponentName object.
This is specified by the fully qualified class name of the target.
E.g.,

```
com.example.ExampleActivity
```
Action

A string that specifies the generic action to perform
e.g. View, Pick, etc...
These *can* be user defined
More often specified by constants within the Intent class (or other framework)
Example Actions

ACTION_VIEW

Use this action in an intent with `startActivity()` when you have some information that an activity can show to the user, such as a photo to view in a gallery app, or an address to view in a map app.

ACTION_SEND

Also known as the `share` intent, you should use this in an intent with `startActivity()` when you have some data that the user can share through another app, such as an email app or social sharing app.
Actions and Broadcasts

Actions defined in Intents aimed at Activities, Services, and Content Providers define the Action to be taken.

Broadcasts instead define the field by what Action took place! Reports the action to any Broadcast Receiver listening for that event.
A URI object which references data to be acted on
The type of data may be included, and is highly dependent on the Action

Data Type should be included, especially with implicit Intents

- Allow system to determine best application to handle
setData() – Used to set the URI for the data
setType() – Used to set the MIME type of the data
setDataAndType() – Used to set both

If setting both, use the setDataAndType() as the others nullify the opposite field
String containing information about the kind of component which should handle the Intent

Intents can include 0 or more Categories

• Most do not require a category
Example Categories

**CATEGORY_BROWSABLE**

The target activity allows itself to be started by a web browser to display data referenced by a link, such as an image or an e-mail message.

**CATEGORY_LAUNCHER**

The activity is the initial activity of a task and is listed in the system's application launcher.
Extras

Intent objects may need to carry additional information

`putExtra()` – Family of methods which allow inclusion of key/value pairs
When using **ACTION_SEND**:

- To – **EXTRA_EMAIL**
- Subject – **EXTRA_SUBJECT**

To declare your own, make sure to include the package as part of the string:

```java
static final String CALCULATOR_ACTION =
  "com.ahnelson.calculator.DIVIDE";
```
Flags – Metadata for the Intent

Flags may:

- Tell system how to launch activity
- Tell system how to treat after launch
Flags

FLAG_GRANT_READ_URI_PERMISSION,
FLAG_GRANT_WRITE_URI_PERMISSION,
FLAG_FROM_BACKGROUND,
FLAG_DEBUG_LOG_RESOLUTION,
FLAG_EXCLUDE_STOPPED_PACKAGES,
FLAG_INCLUDE_STOPPED_PACKAGES,
FLAG_GRANT_PERSISTABLE_URI_PERMISSION,
FLAG_GRANT_PREFIX_URI_PERMISSION,
FLAG_RECEIVER_REGISTERED_ONLY,
FLAG_RECEIVER_REPLACE_PENDING,
FLAG_RECEIVER_FOREGROUND,
FLAG_RECEIVER_NO_ABORT,
FLAG_ACTIVITY_CLEAR_TOP,

FLAG_ACTIVITY_FORWARD_RESULT,
FLAG_ACTIVITY_PREVIOUS_IS_TOP,
FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS,
FLAG_ACTIVITY_BROUGHT_TO_FRONT,
FLAG_RECEIVER_VISIBLE_TO_INSTANT_APP
FLAG_ACTIVITY_LAUNCHED_FROM_HISTORY,
FLAG_ACTIVITY_NEW_DOCUMENT,
FLAG_ACTIVITY_NO_USER_ACTION,
FLAG_ACTIVITY_REORDER_TO_FRONT,
FLAG_ACTIVITY_NO_ANIMATION,
FLAG_ACTIVITY_CLEAR_TASK,
FLAG_ACTIVITY_TASK_ON_HOME,
FLAG_ACTIVITY_REMAIN_IN_RECENTS,
FLAG_ACTIVITY_LAUNCH_ADJACENT
Receiving Intents
Intent Filters advertise which **Implicit** Intents the Application can receive.

Created in the Android Manifest file using: `<intent-filter>` elements nested in components e.g. `<activity>` element)
Intent-Filter

Specifies type of Intent to receive based on:

- Action
- Data
- Category
<action> – Declares which Intent actions are accepted

- Accepts based on the android:name attribute
- Value must be literal string (not class constants)
<data> – Declares type of data which can be accepted

One or more attributes to specify parts of the data URI as well as the MIME type

URI Parts:

- Scheme
- Host
- Port
- Path
Declares the Intent categories which are accepted
Accepted in the android:name attribute
**Value must be literal string value of action**

You **must** include the CATEGORY_DEFAULT category to receive implicit intents
You may include other categories for broader scope
<activity android:name="ShareActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
For greater scope, you can combine multiple instances of actions, data, and category in the filter.

Your component will need to meet any combination of those filters.

As long as the Intent matches on each a name, a category, and a data, then it will accept the Intent.
Multiple Filters

You may also include multiple intent-filters in a single component. If you don’t want to handle all combinations, just certain ones. If any one intent-filter matches a name, data, and category, it will accept the Intent.
Unless necessary, a service should not declare an Intent filter

Services should be called by Explicit Intents

- That way it can’t be called by an external application
- Often, you may use your activity to call the service if needed in another application
<activity android:name="MainActivity">
   <!-- This activity is the main entry, should appear in app launcher -->
   <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
</activity>

<activity android:name="ShareActivity">
   <!-- This activity handles "SEND" actions with text data -->
   <intent-filter>
      <action android:name="android.intent.action.SEND"/>
      <category android:name="android.intent.category.DEFAULT"/>
      <data android:mimeType="text/plain"/>
   </intent-filter>

   <!-- This activity also handles "SEND" and "SEND_MULTIPLE" with media data -->
   <intent-filter>
      <action android:name="android.intent.action.SEND"/>
      <action android:name="android.intent.action.SEND_MULTIPLE"/>
      <category android:name="android.intent.category.DEFAULT"/>
      <data android:mimeType="application/vnd.google.panorama360+jpg"/>
      <data android:mimeType="image/*"/>
      <data android:mimeType="video/*"/>
   </intent-filter>
</activity>
<activity android:name="MainActivity">
  <!-- This activity is the main entry, should appear in app launcher -->
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>

<activity android:name="ShareActivity">
  <!-- This activity handles "SEND" actions with text data -->
  <intent-filter>
    <action android:name="android.intent.action.SEND" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="text/plain" />
  </intent-filter>
  <!-- This activity also handles "SEND" and "SEND_MULTIPLE" with media data -->
  <intent-filter>
    <action android:name="android.intent.action.SEND" />
    <action android:name="android.intent.action.SEND_MULTIPLE" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="application/vnd.google.panorama360+jpg" />
    <data android:mimeType="image/*" />
    <data android:mimeType="video/*" />
  </intent-filter>
</activity>
Pending Intents

**PendingIntent** – Object which wraps an Intent object

- Grants permissions to foreign applications to use the contained Intent as if it were executed from your application
When and Why use PendingIntents?

Major use cases:

- Intent to be executed when a user performs an action with your Notification
- Intent to be executed when user interacts with your App Widget
- Intent to be executed at a specified future time

In each of these cases, your application does not execute the intent In order, the NotificationManager, the Home Screen app, and the AlarmManager
Pending Intent Methods

When using a pending Intent, execution not handled by startActivity(), startService(), etc...

Specific methods for PendingIntents

- PendingIntent.getActivity()
- PendingIntent.getService()
- PendingIntent.getBroadcast()
Pending Intent - Notification Example

// prepare intent which is triggered if the notification is selected

Intent intent = new Intent(this, NotificationReceiver.class);
// use System.currentTimeMillis() to have a unique ID for the pending intent
PendingIntent pIntent = PendingIntent.getActivity(this, 0, intent, 0);

// build notification
// the addAction re-use the same intent to keep the example short
Notification n = new Notification.Builder(this)
    .setContentTitle("New mail from test@gmail.com")
    .setContentText("Subject")
    .setSmallIcon(R.drawable.icon)
    .setContentIntent(pIntent)
    .setAutoCancel(true)
    .addAction(R.drawable.icon, "Call", pIntent)
    .addAction(R.drawable.icon, "Work", pIntent)
    .addAction(R.drawable.icon, "And more", pIntent).build();

NotificationManager notificationManager =
    (NotificationManager) getSystemService(NOTIFICATION_SERVICE);

notificationManager.notify(0, n);

http://www.vogella.com/tutorials/AndroidNotifications/article.html
On receiving an implicit Intent, the system attempts to determine the best activity by comparing it to Intent Filters Matches on a three-part test on aspects:

- Action
- Data
- Category
An Intent-filter may specify zero or more `<action>` elements. If a filter has zero actions, an Intent cannot match. If an Intent has zero actions, it passes the Action test as long as a filter has at least one action.

```xml
<intent-filter>
    <action android:name="android.intent.action.EDIT" />
    <action android:name="android.intent.action.VIEW" />
    ...
</intent-filter>
```

Test passes if Intent action matches at least one intent-filter action element.
Intent filter can specify zero or more `<category>` elements. To pass test, **every** category in the Intent must match a category in the filter. However, an Intent need not have every category in the filter.

```xml
<intent-filter>
  <category android:name="android.intent.category.DEFAULT" />
  <category android:name="android.intent.category.BROWSABLE" />
  ...
</intent-filter>
```

CATEGORY_DEFAULT is appended to all Intent objects. Intent-filter must define the default category to receive Implicit Intents.
An intent filter may declare zero or more `<data>` elements

```xml
<intent-filter>
    <data android:mimeType="video/mpeg" android:scheme="http" ... />
    <data android:mimeType="audio/mpeg" android:scheme="http" ... />
    ...
</intent-filter>
```

Each data element can specify a URI structure and a data type.

The URI is specified by the parts of the URI as separate attributes

```xml
<scheme>://<host>:<port>/<path>
```
Example: content://com.ahnelson.calculator:200/folder/subfolder

- **Scheme** – content
- **Host** – com.ahnelson.calculator
- **Port** – 200
- **Path** – folder/subfolder
Each attribute is optional, however:

- If scheme is not specified, host is ignored
- If host is not specified, port is ignored
- If scheme and host are unspecified, path is ignored
When URI from Intent is compared, it only compares to parts of URI included in the filter
i.e.

- If filter specifies only a scheme, all URIs with that scheme match
- If filter specifies a scheme and authority, but no path, all URIs with same scheme and authority pass
- If filter specifies scheme, authority, and path, filter must match all

Path specification can contain a wildcard as an asterisk (*)
Data Test

The Data test compares the URI and the MIME type elements pursuant to the following:

- Intent which contains no URI or MIME type passes only if filter does not specify any URIs or MIME types.
- Intent which contains a URI but no MIME type passes the test only if URI matches and the filter does not specify a MIME type.
- Intent that contains a MIME type, but no URI passes only if filter lists same MIME type and does not specify a URI format.
- Intent that contains both a URI and a MIME passes if matches the MIME type in the filter and either:
  - Matches a URI in the filter
  - If it has a “content:” or “file:” URI scheme
A filter is assumed to support the “content:” and “file:” URI schemes if only a MIME type is specified.

It is assumed that components can read local data from content providers and the file system.
Example Data Attribute

Intent-filter for fetching video data from the network:

```
<intent-filter>
    <data android:scheme="http" android:type="video/*" />
    ...
</intent-filter>
```
Filters are used to discover which components can handle an Intent and to discover attributes of components on the device.

Example:
All activities with intent filters which specify ACTION_MAIN action and CATEGORY_LAUNCHER populate in the Home app.
Your application may also use Intent Matching using PackageManager methods

- `queryIntentActivities()`
- `queryIntentActivityOptions()`
- `resolveActivity()`
- `queryIntentServices()`
- `resolveService()`
- `queryBroadcastReceivers()`
- `queryIntentContentProviders()`
- `resolveContentProvider()`