Intent & Intent Filters

Sang Shin
Michèle Garoche
www.javapassion.com
“Learn with Passion!”

Disclaimer

- Portions of this presentation are modifications based on work created and shared by the Android Open Source Project
  - http://code.google.com/policies.html
- They are used according to terms described in the Creative Commons 2.5 Attribution License
  - http://creativecommons.org/licenses/by/2.5/

Topics

- Intent messaging
- Intent object structure
- Intent resolution
  - Types of Intent
  - Intent filtering
  - Intent filter tests
  - Special case (MAIN action and LAUNCHER category)

Intent Messaging

What is Intent Messaging?

- Three of the core components of an Android application - activities, services, and broadcast receivers - are activated through messages, called intents
  - One activity starts another activity by creating/sending an Intent
- Means of leveraging activities, services, and broadcast receivers of other applications
  - Application #1 can use “Photo display” activity of Application #2 (or system) by creating/sending an intent

What is Intent Messaging?

- Intent messaging is a facility for late run-time binding between components in the same or different applications
  - Enables a flexible and agile architecture: A component that performs a task is selected during runtime
  - One activity can start another activity by creating an intent that says “Display web browser” - Android runtime then selects a best qualified “Display web browser” activity among the possible candidates during runtime through so-called “Intent resolution” (We will talk about this in detail later in this presentation)

What Does Intent Object Contain?

- Information of interest to the target component that receives the intent
  - Action to be taken
  - Data to act on
- Information of interest to the Android system
  - Category of component that should handle the intent
  - Instructions on how to launch a target activity

Intent Object Structure

Intent Object Structure Is Made Of

- Component name
- Action
- Data
- Category
- Extras
- Flags

Component name Field

- Specifies the name of the component (name of the activity if the component is activity) that should handle the intent
  - Class name of the target component (for example "*com.javapassion.ForwardTargetActivity*")
- Setting component name is optional
  - If it is set, the Intent object is delivered to an instance of the designated class.
  - If it is not set, Android uses other information in the Intent object to locate a suitable target - this is called “intent resolution” (We will talk about this later in this presentation)

Action Field

- A string naming the action to be performed
- The *Intent* class defines a number of pre-defined action constants, including
  - *ACTION_CALL, ACTION_EDIT, ACTION_MAIN, ACTION_SYNC, ACTION_BATTERY_LOW,* etc.
- You can also define your own action strings for activating the components in your application
- The action largely determines how the rest of the intent is structured - particularly the *data* and *extras* fields - much as a method name determines a set of arguments and a return value.

Data Field

- The URI of the data to be acted on and the MIME type of that data.
- Different actions are paired with different kinds of data specifications.
  - If the action field is ACTION_EDIT, the data field would contain the URI of the document to be displayed for editing.
  - If the action is ACTION_CALL, the data field would be a *tel:* URI with the number to call.
  - If the action is ACTION_VIEW and the data field is an *http:* URI, the receiving activity would be called upon to download and display whatever data the URI refers to.

Data Field (Cont.)

- Examples of Action/Data Pairs
  - *ACTION_VIEW content://contacts/people/1* -- Display information about the person whose identifier is "1".
  - *ACTION_DIAL content://contacts/people/1* -- Display the phone dialer with the person filled in.
  - *ACTION_VIEW tel:123* -- Display the phone dialer with the given number filled in.
  - *ACTION_DIAL tel:123* -- Dial the phone dialer with the given number filled in.
  - *ACTION_EDIT content://contacts/people/1* -- Edit information about the person whose identifier is "1".
  - *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.

Data Field (Cont.)

- When matching an intent to a component that is capable of handling the data, it's often important to know the type of data (its MIME type) in addition to its URI.
  - For example, a component that is displays image data should not be called upon to play an audio file.
- In many cases, the data type can be inferred from the URI — particularly *content:* URIs, which indicate that the data is located on the device and controlled by a content provider. But the type can also be explicitly set in the Intent object.
  - (We will cover content providers later)

Category Field

- A string containing additional information about the kind of component (activity, service, or broadcast receiver) that should handle the intent.
- Any number of category descriptions can be placed in an Intent object
- Android provides a set of predefined categories (We will see them in the following slide)
- You can define your own categories

Pre-defined Categories (by Android)

- CATEGORY_BROWSABLE - The target activity can be invoked within the browser to display data referenced by a link — for example, an image or an e-mail message.
- CATEGORY_GADGET - The activity can be embedded inside of another activity that hosts gadgets
- CATEGORY_HOME - This is the home activity, that is the first activity that is displayed when the device boots.
- CATEGORY_LAUNCHER - The activity can be the initial activity of a task and is listed in the top-level application launcher.
- CATEGORY_PREFERENCE - The target activity is a preference panel.
- Many more

Extras Field

- Key-value pairs for additional information that should be delivered to the component handling the intent.
- Just as some actions are paired with particular kinds of data URIs, some are paired with particular extras.
  - ACTION_TIMEZONE_CHANGED action has a "time-zone" extra that identifies the new time zone
  - ACTION_HEADSET_PLUG action has a "state" extra indicating whether the headset is now plugged in or unplugged , as well as a "name" extra for the type of headset
  - If you were to invent a SHOW_COLOR action, the color value would be set in an extra key-value pair.

Flags Field

- Flags of various sorts.
- Many instruct the Android system how to launch an activity (for example, which task the activity should belong to) and how to treat it after it's launched (for example, whether it belongs in the list of recent activities).

Intent Resolution

What is Intent Resolution?

- Android's scheme of determining which target component should be selected for handling a passed intent

Intent Resolution:
Types of Intents

Types of Intents

- Explicit intents
  - Designate the target component by its class (the component name field is set by the class name)
  - Since component names (class name of the target activity, for example) would generally not be known to developers of other applications, explicit intents are typically used for application-internal messages — such as an activity starting a subordinate service or launching a sister activity.
- Implicit intents
  - Do not name a target (the component name field is blank).
  - Implicit intents are often used to activate components in other applications.

Intent Resolution Schemes

- "Explicit resolution" for Explicit intents
  - Android delivers an explicit intent to an instance of the designated target class.
  - Nothing in the Intent object other than the component name matters for determining which component should get the intent.
- "Implicit" resolution for Implicit intents
  - In the absence of a designated target, the Android must find the best component (or components) to handle the intent
  - By comparing the contents of the Intent object to intent filters, structures associated with the possible target components that can potentially receive intents - intent filters are specified for each component in the *AndroidManifest.xml* file

Intent Resolution (for Implicit Intents): Intent Filters

What are Intent Filters?

- Filters describe and advertise the capabilities of a target component
  - Informs the system which implicit intents a component can handle
- If a component does not have any intent filters, it can receive only explicit intents.
- A component with filters can receive both explicit and implicit intents.
- A component has separate filters for each job it can do
  - NoteEditor activity of the sample Note Pad application has two filters — one for starting up with a specific note that the user can view or edit, and another for starting with a new, blank note

Where are Filters Specified?

- Filters are specified for each component in the *AndroidManifest.xml* file

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" ...>
  <application android:icon="@drawable/app_notes" ... >

      <activity android:name="NoteEditor"
              android:theme="@android:style/Theme.Light"
              android:label="@string/title_note" >
        <intent-filter android:label="@string/resolve_edit">
            <action android:name="android.intent.action.VIEW" />
            <action android:name="android.intent.action.EDIT" />
            <action android:name="com.android.notepad.action.EDIT_NOTE" />
            <category android:name="android.intent.category.DEFAULT" />
            <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
        </intent-filter>
        <intent-filter>
            <action android:name="android.intent.action.INSERT" />
            <category android:name="android.intent.category.DEFAULT" />
            <data android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
        </intent-filter>
      </activity>

      ...
```

Intent Resolution (for Implicit Intents): Intent Filter Tests

How Android System Perform Intent Resolution (for Implicit Intents)?

- An implicit Intent object are tested against an intent filters (of target components) in three areas
  - Action
  - Category
  - Data (both URI and data type)
- To be delivered to the target component that owns the filter, it must pass all three tests
- If an intent can pass through the filters of more than one activity or service, the user may be asked which component to activate.
- If no target can be found, an exception is raised

Action Test

- To pass this test, the action specified in the Intent object must match one of the actions listed in the filter.
- If the Intent object or the filter does not specify an action, the results are as follows:
  - If a filter does not specify any action, there is nothing for an intent to match, so all intents fail the test. No intents can get through the filter.
  - On the other hand, an Intent object that doesn't specify an action automatically passes the test — as long as the filter contains at least one action.

Category Test

- For an intent to pass the category test, every category in the Intent object must match a category in the filter.
  - The filter can list additional categories, but it cannot omit any that are in the intent
  - The categories of the filter should be the super-set of the categories of the Intent object
- Special case
  - An Intent object with no categories should always pass this test, regardless of what's in the filter

Category Test - android.intent.category.DEFAULT

- Android treats all implicit intents passed to startActivity() as if they contained at least one category: "android.intent.category.DEFAULT" (the CATEGORY_DEFAULT constant).
- Therefore, activities that are willing to receive implicit intents must include "android.intent.category.DEFAULT" in their intent filters.

Data Test

- Example

```
<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 data type (MIME media type) and URI.

Data Test - URI

- For each part of the URI, there are separate parts - scheme, host, port, and path

  scheme://host:port/path

- Example

  content://com.example.project:200/folder/subfolder/etc

  The scheme is "content", the host is "com.example.project", the port is "200", and the path is "folder/subfolder/etc".

  The host and port together constitute the URI authority

- When the URI in an Intent object is compared to a URI specification in a filter, it's compared only to the parts of the URI actually mentioned in the filter

Data Test - Mime media type

- It's more common in filters than a URI.
- Both the Intent object and the filter can use a "*" wildcard for the subtype field — for example, "text/*" or "audio/*" — indicating any subtype matches.

Data Test - Testing Rules

- An Intent object that contains both a URI and a data type (or a data type can be inferred from the URI) passes the data type part of the test only if its type matches a type listed in the filter.
- It passes the URI part of the test either if its URI matches a URI in the filter or if it has a *content:* or *file:* URI and the filter does not specify a URI.
  - In other words, a component is presumed to support *content:* and *file:* data if its filter lists only a *data* type.
  - Components are able to get local data from a file or content provider
  - Their filters can list just a data type and do not need to explicitly name the *content:* and *file:* schemes

Data Test - Common Case #1

- The example below tells Android that the component can get image data from a content provider and display it:

  <data android:mimeType="image/" />*

- Since most available data is dispensed by content providers, filters that specify a data type but not a URI are perhaps the most common.

Data Test - Common Case #2

- Another common configuration is filters with just a scheme and a data type.
- For example, a <data> element like the following tells Android that the component can get video data from the network and display it:

  <data android:scheme="http" android:type="video/" />*

- Usage examle
  - Consider what the browser application does when the user follows a link on a web page.
  - It first tries to display the data (as it could if the link was to an HTML page). If it can't display the data, it puts together an implicit intent with the scheme and data type and tries to start an activity that can do the job. If there are no takers, it asks the download manager to download the data. That puts it under the control of a content provider, so a potentially larger pool of activities (those with filters that just name a data type) can respond.

Intent Resolution:
Special Use Cases

.MAIN Action & .LAUNCHER Category

- Activities that can initiate applications have filters with "*android.intent.action.MAIN*" specified as the action
  - This is a way an application gets started fresh, without a reference to any particular data.
- If they are to be represented in the application launcher, they also specify the "*android.intent.category.LAUNCHER*" category:

```
<intent-filter . . . >
    <action android:name="code android.intent.action.MAIN" />
    <category android:name="code android.intent.category.LAUNCHER" />
</intent-filter>
```

.MAIN Action & .Launcher Category

- The Android system populates the application launcher, the top-level screen that shows the applications that are available for the user to launch, by finding all the activities with intent filters that specify the "android.intent.action.MAIN" action and "android.intent.category.LAUNCHER" category. It then displays the icons and labels of those activities in the launcher.
- The Android system discovers the home screen by looking for the activity with "android.intent.category.HOME" in its filter

Thank you!

Check JavaPassion.com Codecamps!
http://www.javapassion.com/codecamps
"Learn with Passion!"