Intents

Dr Tadhg O’Sullivan

Intents

- Intents are message passing mechanisms that works within your application, as well as between applications
- With Intents you can:
  - Declare your intention that an Activity or Service is to be started to perform an action with particular piece of data
  - Broadcast that an event has occurred
  - Explicitly start a particular Service or Activity

Starting Activity via Intent

- Explicit:

```
Intent intent = new Intent(MyActivity.this,
                           MyOtherActivity.class);
startActivity(intent);
```

- Late binding (Implicit):

```
if (somethingWeird && itDoesntLookGood ) {
      Intent intent = new Intent(Intent.ACTION_DIAL,
                                 Uri.parse("tel:5552368"));
      startActivity(intent); }
```

SubActivities

- Activities started with startActivity() are independent and won't provide any feedback when it closes
- Alternatively, you can start activity as a sub-Activity, which returns result to its parent
  - You can call startActivityForResult(), passing it the Intent and a number (unique to the calling activity).
  - The callback method to get a result is onActivityResult().

Implicit Sub-Activity Example

- Using Andorid Camera activity and returning back to your activity

```
Intent intent = new
Intent("android.media.action.IMAGE_CAPTURE");
intent.putExtra(android.provider.MediaStore.EXTRA_OUTPUT,
Uri.fromFile(newFile));

startActivityForResult(intent, 0);
```

SubActivities Cont'd

- What do you get back from SubActivity:
- A result code by calling setResult() .
- Typically RESULT_OK or RESULT_CANCELLED, but this can be customized
- An optional String containing some result data URL
- An optional Bundle containing additional information beyond the result code and data string.

Intent Parameters

- Action:
  - Native Android actions: ACTION_ANSWER, ACTION_CALL, ACTION_DELETE, ACTION_DIAL, ACTION_EDIT, ACTION_INSERT, ACTION_PICK, ACTION_SEARCH, ACTION_SENDTO, ACTION_SEND, ACTION_VIEW, ACTION_WEB_SEARCH
  - For your own actions, use system based on Java package names, *e.g.* 'ie.ucd.comp30150.GHOSTS_FOUND'
  - The most common: ACTION_VIEW
- Category: LAUNCHER, DEFAULT, ...

Intent Parameters Cont'd

- MIME Type
- Component: class that supposed to receive an intent.
- Extras: a Bundle you want to pass along

Intent Routing

- Determining which activity to run:
  - The activity must support the specified action.
  - The activity must support the stated MIME type (if supplied).
  - The activity must support all of the categories named in the intent.

Intent Routing Cont'd

- If there is still uncertainty, the system looks at the default config
- If left unspecified, system prompts a user:
  - Have you noticed that browsers sometimes would give you a choice to open up a map in Google Map or Browser?
  - Or page with embedded podcast to open in Browser or Google Listen?
  - Or if you install a 3rd party dialer, system will ask you too? (never do that unless you really trust the dialer)
- Default choices can be updated in settings :-)

Intent Filters

- All Android components that wish to be notified via intents must declare intent filters
- To do this, you need to add intent-filter elements to your AndroidManifest.xml file.
- Activities, Services, and BroadcastReceivers can be registered as being capable of performing an action with a particular type of data

(Declaring both Activities and Intent Filters are one of the most common errors in Android Development at the beginning)

Intent Filters Cont'd

<manifest xmlns:android="..."
package="comp.Camera.demo">
<uses-feature android:name=*"android.hardware.camera" />*
<application>
<activity android:name=".Now" android:label="Now">
<intentfilter>
<action android:name="android.intent.action.MAIN" />
<category
android:name="android.intent.category.LAUNCHER" />
</intentfilter>
</activity>
</application>
</manifest>

Intent Filters Cont'd

```
<activity android:name=".OurViewActivity">
<intentfilter>
<action android:name="android.intent.action.VIEW" />
<category
android:name="android.intent.category.DEFAULT"
/>
<data
android:mimeType=" text/plain" />
</intentfilter>
</activity>
```

Intent Receivers

- Intent Receivers are disposable objects implementing BroadcastReceiver interface
  <receiver android:name=".MyIntentReceiverClassName" />
- Always active, even if application is killed!
- Used to trigger something in a service rather than activity

  OR
- Used to run different activity depending on on some state

Intent Receivers Cont'd

- Alive only as long as it takes to process onReceive(), then disposed

- Exception: if implemented in longer-living object, *i.e.* Service or Activity. In this case it should be created with *registerReceiver()* in code rather than in Manifest.
  - Do not forget *unregisterReceiver()* afterwards

Questions

- Please ask in the Student Forum