UNIVERSITY OF ONTARIO
INSTITUTE OF TECHNOLOGY

Mobile Devices

Intents

Randy J. Fortier
Randy.Fortier@uoit.ca
@randy_fortier

Outline

- In this section, we'll learn:
  - What are intents?
  - How are intents used?
  - Within an application
  - Between applications
  - What are broadcast receivers?
  - Differences with activities
  - How they communicate with other components, user

UNIVERSITY OF ONTARIO
INSTITUTE OF TECHNOLOGY

Android

Intents

Randy J. Fortier
Randy.Fortier@uoit.ca
@randy_fortier

Intents

- Intents are standardized messages
  - Very useful for communicating between 3rd parties
- Intents can be used to:
  - Switch to a different activity
  - Run an application
  - Open a web browser
  - Send an E-Mail
  - Dial a phone number
  - ...

Intents

- Intents come in two types:
  - Explicit
    - You specify the exact class to receive the intent
  - Implicit
    - You categorize the intent, and let the applications choose to handle it

Launching Activities

1. Create an intent
   - Include information to complete the request
     - e.g. Phone number, URL
2. Start the activity
   - startActivity:
     - Launch the activity as a peer
   - startActivityForResult:
     - Launch the activity as a sub-activity
3. Handle the response of the activity (optional)
   - Sub-activities only

Explicit Intents: Start an Activity

```
Intent intent = new Intent(FirstActivity.this,
                           SecondActivity.class);
startActivity(intent);
```

- Explicit intent constructor arguments:
  - Context (who sent the intent)
  - Recipient class (who receives the intent)
- Explicit intents can also run other component types
  - e.g. services

Explicit Intents: Start a Sub-activity

```
Intent intent = new Intent(this,
                    ChooseColourActivity.class);
int CHOOSE_BG_COLOUR_REQUEST = 17;
startActivityForResult(intent,
                        CHOOSE_BG_COLOUR_REQUEST);
```

- The second argument here is the request code
  - This can be used to have the same activity handle multiple different types of requests
  - e.g. Choosing a background colour versus a colour for another purpose

Explicit Intents: Start a Sub-activity

- The sub-activity sets the result whenever needed:

```
Intent resultIntent = new Intent(Intent.ACTION_PICK);
resultIntent.putExtra("selected", colour);
setResult(Const.RESULT_OK, resultIntent);
finish();
```

- The calling activity retrieves the result via an event:

```
public void onActivityResult(int reqCode,
            int resCode, Intent result) {
  super.onActivityResult(reqCode, resCode, result);
  if (resCode == Const.RESULT_OK)
    String colour = result.getStringExtra("selected");
}
```

Explicit Intents: Start a Service

- Services run in the background for a long time
  - They generally do not have a UI

```
Intent intent = new Intent(this, SomeService.class);
startService(intent);
```

Implicit Intents: Open a Browser

```
Intent intent = new Intent(Intent.ACTION_VIEW,
        Uri.parse("http://www.google.com"));
startActivity(intent);
```

- Implicit intent constructor arguments:
  - Action
    - Used by activities to know if they want to handle it
  - Additional data (in this case, the web address)

Implicit Intents: Send an E-Mail

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("plain/text");
String[] recipients = new String[] {toEmail};
startActivity(Intent.createChooser(intent,
            "How do you want to send this message?"));
```

- This example shows how to handle the situation where two applications can handle the message

Implicit Intents: Send an E-Mail

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("plain/text");
String[] recipients = new String[] {toEmail};
intent.putExtra(Intent.EXTRA_EMAIL, recipients);
intent.putExtra(Intent.EXTRA_SUBJECT, messageSubject);
intent.putExtra(Intent.EXTRA_TEXT, messageBody);
startActivity(Intent.createChooser(intent,
            "How do you want to send this message?"));
```

- This example shows how to put in the contents of the message

Implicit Intents: Dial a Phone Number

```
Intent intent = new Intent(Intent.ACTION_CALL);
intent.setData(Uri.parse("tel:2135551234"));
startActivity(intent);
```

Implicit Intents: Send an SMS Message

```
Intent intent = new Intent(Intent.ACTION_VIEW,
                Uri.parse("sms:2135551234"));
startActivity(intent);
```

Implicit Intents: View a Map

```
Intent intent = new Intent(Intent.ACTION_VIEW,
Uri.parse("google.navigation:q=toronto+on"));
startActivity(intent);
```

Implicit Intents: View a Video

- This is not ideal, but it does work:

```
Intent intent = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://www.youtube.com/watch?v=...");
startActivity(intent);
```

UNIVERSITY OF ONTARIO
INSTITUTE OF TECHNOLOGY

Android

Broadcast Receivers

Randy J. Fortier
Randy.Fortier@uoit.ca
@randy_fortier

Broadcast Receiver

- Broadcast receivers are controllers for a specific purpose, to receive intents
  - Specifically, intents that have been broadcast
- Broadcast receivers do **not** perform significant calculation or data storage/retrieval
  - Usually, the broadcast receiver would pass along the information to another component (e.g. Activity)
  - The runtime of a single receive is limited to 5s

Broadcast Intents

- Broadcast events:
  - Are frequently generated by the Android OS
  - Can be generated by your application
- System intents:
  - Powering off
  - Screen unlocked
  - Battery low
  - Headphones plugged in
  - Screen orientation changed
  - ...

Broadcasting Intents

- Broadcast your own intents:

```
Intent intent =
            new Intent("ca.uoit.DOWNLOAD_COMPLETE");
intent.putExtra("application", appName);
sendBroadcast(intent);
```

Broadcasting Sticky Intents

- Sticky intents don't disappear until a receiver is available to handle them:

```
Intent intent = new Intent("ca.uoit.DOWNLOAD_COMPLETE");
intent.putExtra("application", appName);
sendStickyBroadcast(intent);
```

Broadcast Receiver Example

```
public class HeadphoneNotifier
             extends BroadcastReceiver {
  @Override
  public void onReceive(Context context,
                        Intent intent) {
    // do something, but what?
  }
}
```

Registering a Broadcast Receiver

- In the manifest:

```
<receiver android:name=".HeadphoneNotifier">
  <intent-filter>
    <action android:name="android.intent.ACTION_HEADSET_PLUG" />
  </intent-filter>
</receiver>
```

- From an Activity:

```
IntentFilter filter =
                    new IntentFilter(Intent.ACTION_HEADSET_PLUG);
HeadphoneNotifier receiver = new HeadphoneNotifier(...);
registerReceiver(receiver, filter);
```

Toasts

- Toasts are like popup notifications

```
public class HeadphoneNotifier
             extends BroadcastReceiver {
  @Override
  public void onReceive(Context context,
                        Intent intent) {
    Toast.makeText(getBaseContext(),
                   "Headphones plugged/unplugged!",
                   Toast.LENGTH_SHORT).show();
  }
}
```

Notifications

- Notifications are like system messages

```
public class HeadphoneNotifier
             extends BroadcastReceiver {
  public static int HEADPHONE_PLUGGED = 201;

  @Override
  public void onReceive(Context context,
                        Intent intent) {
        int icon = R.drawable.icon;
        long now = System.currentTimeMillis();
        Notification note = new Notification(icon, message, now);
        note.setLatestEventInfo(context, message, message, null);

        NotificationManager notificationManager;
        notificationManager = (NotificationManager)
          context.getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.notify(HEADPHONE_PLUGGED, note);
  }
}
```

Notifications

- You can also launch an activity if the user clicks on a notification

```
Intent prefsIntent = new Intent(context, PrefsActivity.class);
PendingIntent prefsPending =
          PendingIntent.getActivity(context, 0, prefsIntent, 0);


int icon = R.drawable.icon;
long now = System.currentTimeMillis();
Notification note = new Notification(icon, message, now);
note.setLatestEventInfo(context, message, message, prefsPending);

NotificationManager notificationManager;
notificationManager = (NotificationManager)
  context.getSystemService(Context.NOTIFICATION_SERVICE);
notificationManager.notify(HEADPHONE_PLUGGED, note);
```

UNIVERSITY OF ONTARIO
INSTITUTE OF TECHNOLOGY

Android

Demonstration

Randy J. Fortier
Randy.Fortier@uoit.ca
@randy_fortier

Wrap-Up

- In this section, we learned:
  - How to listen for intents
  - How to send and broadcast intents
  - How to initiate another activity
  - How to initiate a sub-activity
  - How to write broadcast receivers
  - How to register broadcast receivers
  - How to output messages to the user (from broadcast receivers, and other components)