Lesson 12

Android Intents

Victor Matos
Cleveland State University

Notes are based on:

Android Developers
http://developer.android.com/index.html
http://code.google.com/android/reference/android/content/Intent.html

Intents

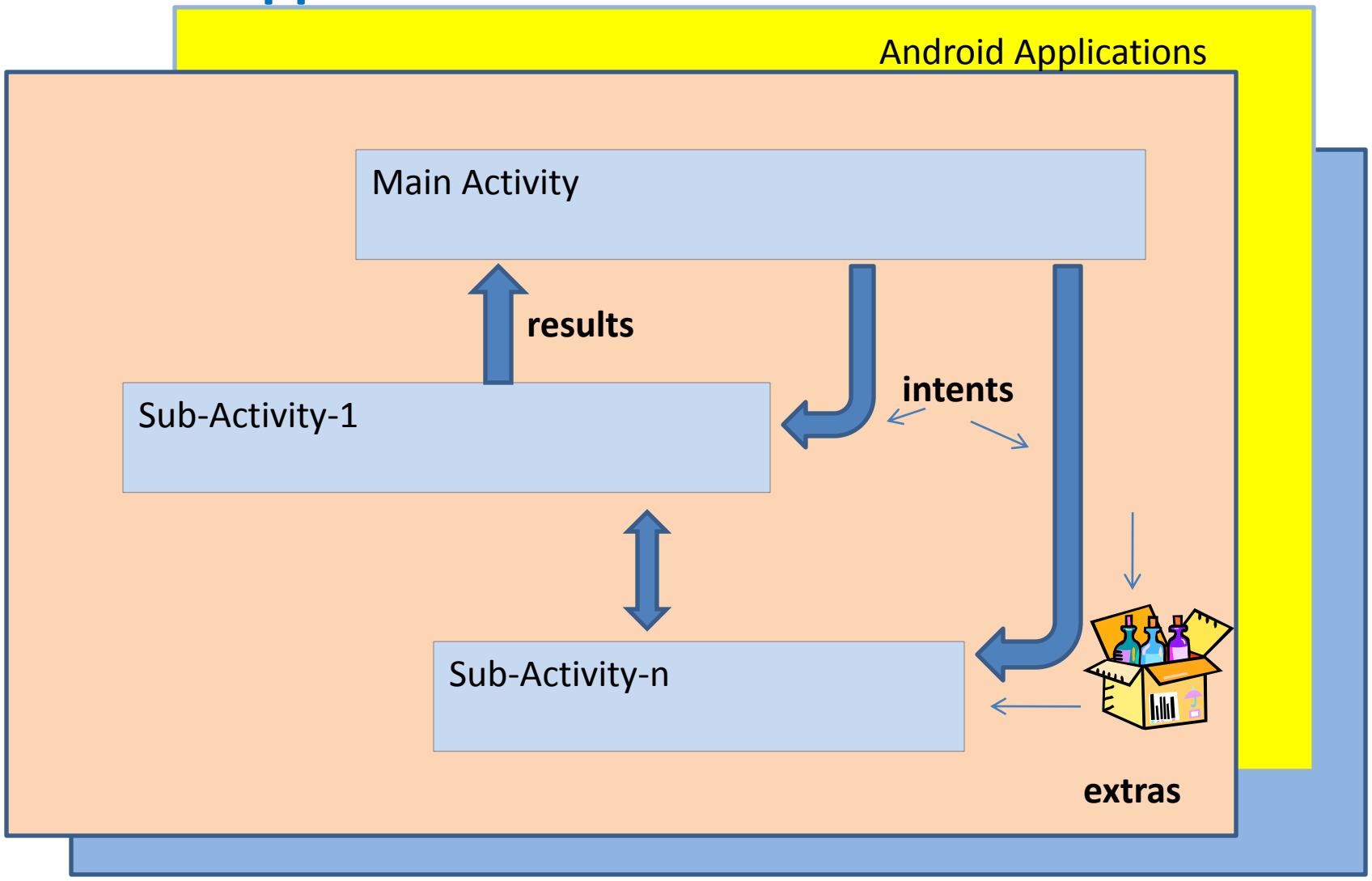
Android Activities

An Android application could include any number of activities.

- An *activity* uses the *setContentView*(...) method to expose (usually) a single UI from which a number of actions could be performed.
- Activities are independent of each other; however they usually cooperate exchanging data and actions.
- Typically, one of the activities is designated as the first one (*main*) that should be presented to the user when the application is launched.
- Moving from one activity to another is accomplished by asking the current activity to execute an ***intent***.
- Activities interact with each other in an **asynchronous** mode.

Intents

Android Applications & Activities



Intents

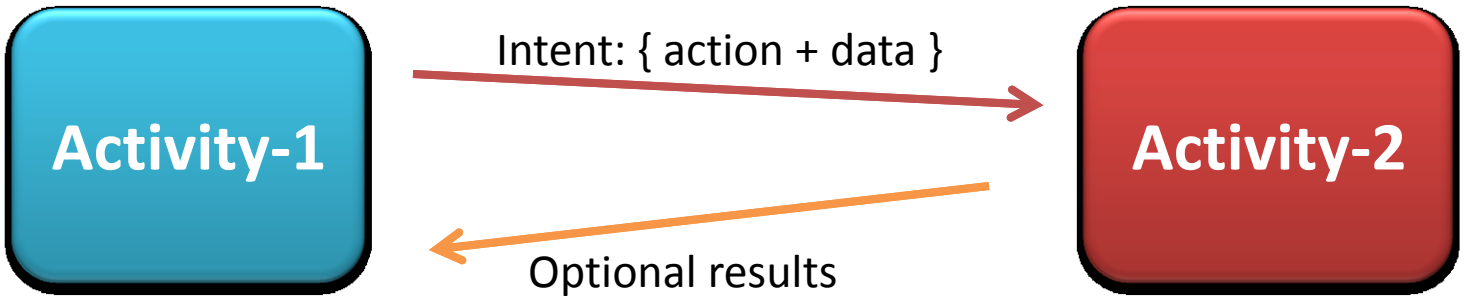
Intents are invoked using the following options

`startActivity (intent)`	launches an *Activity*
`sendBroadcast (intent)`	sends an intent to any interested *BroadcastReceiver* components
`startService(intent)` or `bindService(intent, …)`	communicate with a background Service.

Intents

The main arguments of an Intent are:

1. **Action** The built-in action to be performed, such as `ACTION_VIEW`, `ACTION_EDIT`, `ACTION_MAIN`, … or *user-created-activity*
2. **Data** The primary data to operate on, such as a phone number to be called (expressed as a **Uri**).



Intents

Typically an intent is called as follows:

```
Intent  myActivity = new Intent (action, data);

startActivity (myActivity);
```

Primary data
(as an URI)
tel://
http://
sendto://

Built-in
or
user-created
activity

Intents

Examples of **action/data** pairs are:

ACTION_DIAL ***tel:5551234***
Display the phone dialer with the given number filled in.

ACTION_VIEW ***http://www.google.com***
Show Google page in a browser view.

ACTION_EDIT ***content://contacts/people/2***
Edit information about the contact person whose identifier is "2".

ACTION_VIEW ***content://contacts/people/2***
Used to start an activity to display contact person whose identifier is “2”.

ACTION_VIEW ***content://contacts/*** **people/**
Display a list of people, which the user can browse through.
Selecting a particular person to view would result in a new intent

Intents

Sample of Built-in Standard Actions

List of standard actions that Intents can use for launching activities (usually through *startActivity(Intent).*

ACTION_MAIN	ACTION_ANSWER
ACTION_VIEW	ACTION_INSERT
ACTION_ATTACH_DATA	ACTION_DELETE
ACTION_EDIT	ACTION_RUN
ACTION_PICK	ACTION_SYNC
ACTION_CHOOSER	ACTION_PICK_ACTIVITY
ACTION_GET_CONTENT	**ACTION_SEARCH**
ACTION_DIAL	**ACTION_WEB_SEARCH**
ACTION_CALL	ACTION_FACTORY_TEST
ACTION_SEND	
ACTION_SENDTO	

For a list of actions see:
http://developer.android.com/reference/android/content/Intent.html

Extended List of Predefined Actions

Intents

1. ACTION_AIRPLANE_MODE_CHANGED
2. ACTION_ALL_APPS
3. ACTION_ANSWER
4. ACTION_ATTACH_DATA
5. ACTION_BATTERY_CHANGED
6. ACTION_BATTERY_LOW
7. ACTION_BATTERY_OKAY
8. ACTION_BOOT_COMPLETED
9. ACTION_BUG_REPORT
10. ACTION_CALL
11. ACTION_CALL_BUTTON
12. ACTION_CAMERA_BUTTON
13. ACTION_CHOOSER
14. ACTION_CLOSE_SYSTEM_DIALOGS
15. ACTION_CONFIGURATION_CHANGED
16. ACTION_CREATE_SHORTCUT
17. ACTION_DATE_CHANGED
18. ACTION_DEFAULT
19. ACTION_DELETE
20. ACTION_DEVICE_STORAGE_LOW
21. ACTION_DEVICE_STORAGE_OK
22. ACTION_DIAL
23. ACTION_DOCK_EVENT
24. ACTION_EDIT

Extended List of Predefined Actions

Intents

25. ACTION_EXTERNAL_APPLICATIONS_AVAILABLE
26. ACTION_EXTERNAL_APPLICATIONS_UNAVAILABLE
27. ACTION_FACTORY_TEST
28. ACTION_GET_CONTENT
29. ACTION_GTALK_SERVICE_CONNECTED
30. ACTION_GTALK_SERVICE_DISCONNECTED
31. ACTION_HEADSET_PLUG
32. ACTION_INPUT_METHOD_CHANGED
33. ACTION_INSERT
34. ACTION_INSERT_OR_EDIT
35. ACTION_LOCALE_CHANGED
36. ACTION_MAIN
37. ACTION_MANAGE_PACKAGE_STORAGE
38. ACTION_MEDIA_BAD_REMOVAL
39. ACTION_MEDIA_BUTTON
40. ACTION_MEDIA_CHECKING
41. ACTION_MEDIA_EJECT
42. ACTION_MEDIA_MOUNTED
43. ACTION_MEDIA_NOFS
44. ACTION_MEDIA_REMOVED
45. ACTION_MEDIA_SCANNER_FINISHED
46. ACTION_MEDIA_SCANNER_SCAN_FILE
47. ACTION_MEDIA_SCANNER_STARTED
48. ACTION_MEDIA_SHARED

Extended List of Predefined Actions

Intents

49. ACTION_MEDIA_UNMOUNTABLE
50. ACTION_MEDIA_UNMOUNTED
51. ACTION_MY_PACKAGE_REPLACED
52. ACTION_NEW_OUTGOING_CALL
53. ACTION_PACKAGE_ADDED
54. ACTION_PACKAGE_CHANGED
55. ACTION_PACKAGE_DATA_CLEARED
56. ACTION_PACKAGE_FIRST_LAUNCH
57. ACTION_PACKAGE_INSTALL
58. ACTION_PACKAGE_REMOVED
59. ACTION_PACKAGE_REPLACED
60. ACTION_PACKAGE_RESTARTED
61. ACTION_PASTE
62. ACTION_PICK
63. ACTION_PICK_ACTIVITY
64. ACTION_POWER_CONNECTED
65. ACTION_POWER_DISCONNECTED
66. ACTION_POWER_USAGE_SUMMARY
67. ACTION_PROVIDER_CHANGED
68. ACTION_REBOOT
69. ACTION_RUN
70. ACTION_SCREEN_OFF
71. ACTION_SCREEN_ON
72. ACTION_SEARCH

Extended List of Predefined Actions

Intents

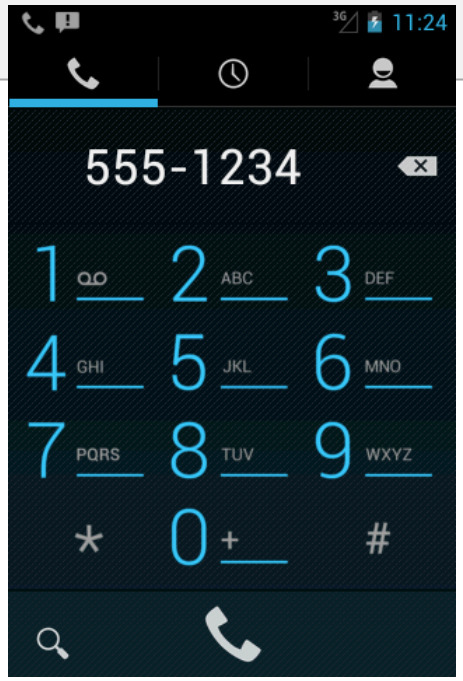
73. ACTION_SEARCH_LONG_PRESS
74. ACTION_SEND
75. ACTION_SENDTO
76. ACTION_SEND_MULTIPLE
77. ACTION_SET_WALLPAPER
78. ACTION_SHUTDOWN
79. ACTION_SYNC
80. ACTION_SYSTEM_TUTORIAL
81. ACTION_TIMEZONE_CHANGED
82. ACTION_TIME_CHANGED
83. ACTION_TIME_TICK
84. ACTION_UID_REMOVED
85. ACTION_UMS_CONNECTED
86. ACTION_UMS_DISCONNECTED
87. ACTION_USER_PRESENT
88. ACTION_VIEW
89. ACTION_VOICE_COMMAND
90. ACTION_WALLPAPER_CHANGED
91. ACTION_WEB_SEARCH

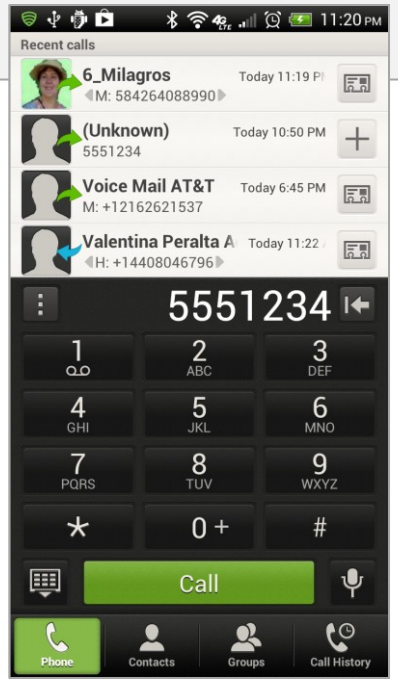
Intents

Example 1

ACTION_DIAL Display the phone dialer with the given number filled in.

```
String myPhoneNumberUri = "tel:555-1234";
Intent myActivity2 = new Intent(Intent.ACTION_DIAL,
                                Uri.parse(myPhoneNumberUri));
startActivity(myActivity2);
```





Images captured from emulator and device respectively

Intents - Secondary Attributes

In addition to the primary *action/data* attributes, there are ***secondary attributes*** that you can also include with an intent, such as: Category, Components, Type, and Extras.

Type

Set an explicit **MIME** data type

- contacts/people
- images/pictures
- images/video
- audio/mp3

MIME - Multipurpose Internet Mail Extensions

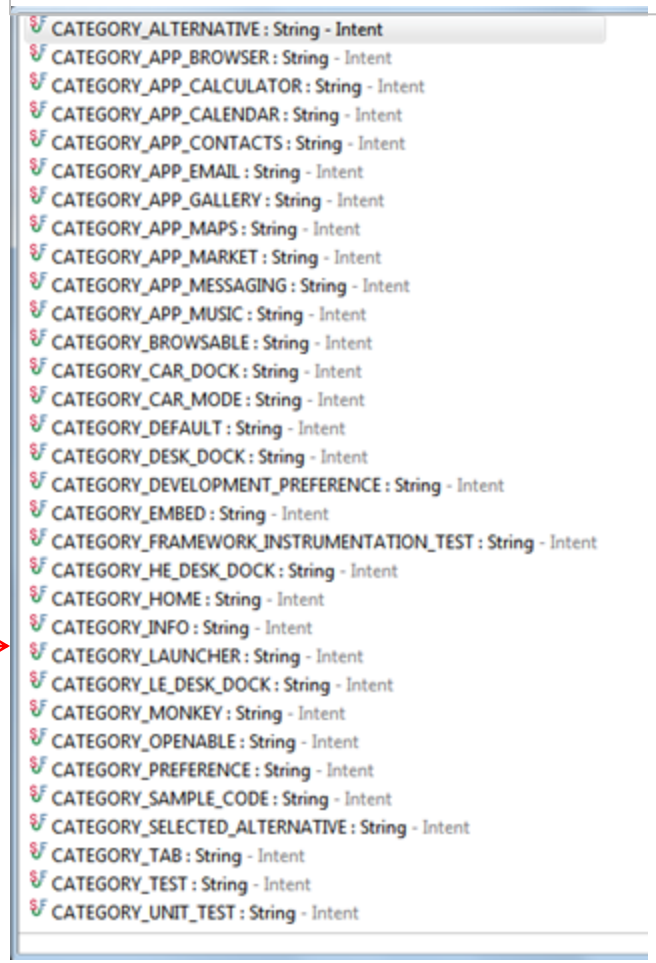
Extras

This is a Bundle of any additional information. Typical methods include:

```
bundle.putInt(key, value)
bundle.getInt(key)
```

Category

additional information about the action to execute



Component

Explicit name of a component class to use for the intent (eg. “MyMethod1”)

Intents

Example 2

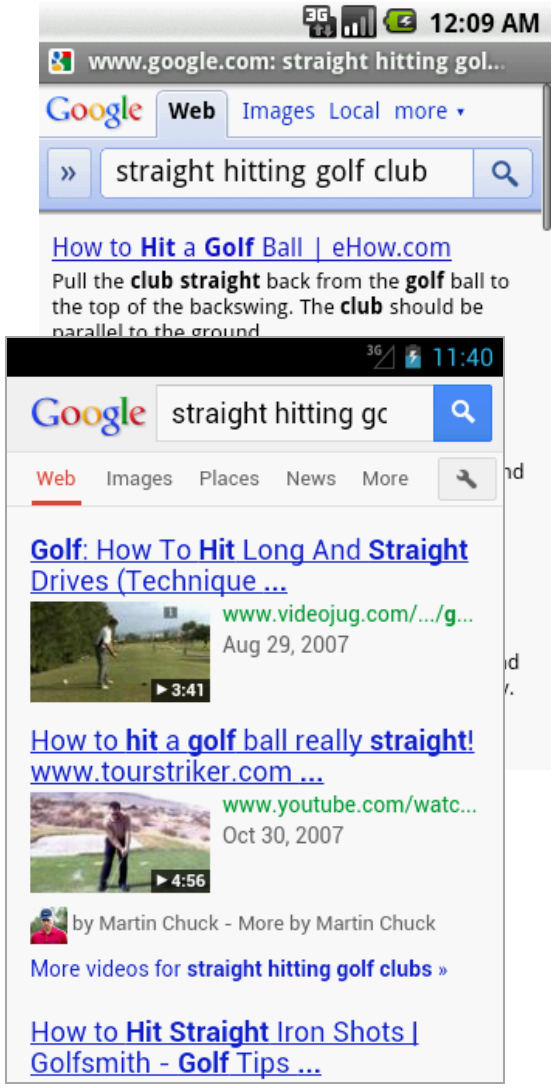
Intents - Secondary Attributes

ACTION_WEB_SEARCH

Passing a string as an ***Extra*** argument for a Google Search. The string is a 'human' query with keywords.

Goal: searching for golf clubs

```
Intent intent = new Intent(
                        Intent.ACTION_WEB_SEARCH);

intent.putExtra(SearchManager.QUERY,
                "straight hitting golf clubs");

startActivity(intent);
```



Secondary data

Apparently the Google answer is 'none'

Intents

Example 3

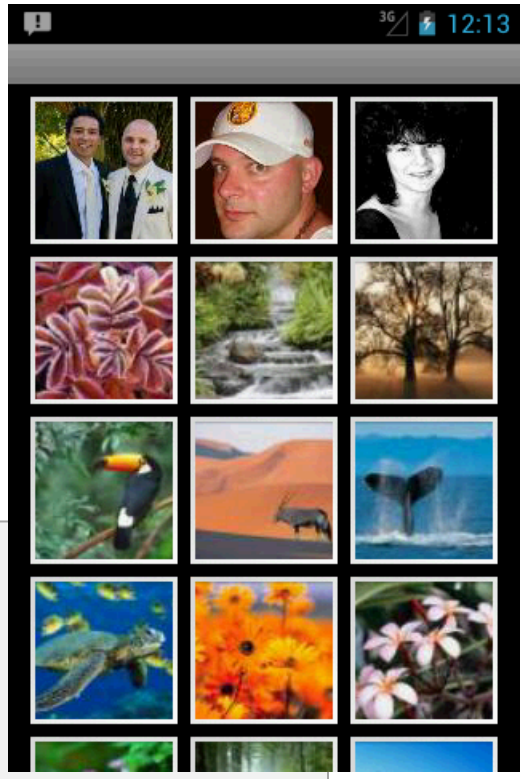
Intents - Secondary Attributes

ACTION_SENDTO

Preparing an SMS. The text is supplied as an **Extra** element. The intent expects such a value to be called “sms_body”

```
Intent intent = new Intent(
                Intent.ACTION_SENDTO,
                Uri.parse("smsto:555-4321"));

intent.putExtra("sms_body",
         "are we playing golf next Sunday?");

startActivity(intent);
```

Intents



Example 4

Intents - Secondary Attributes

ACTION_GET_CONTENT

Displaying the *pictures* contained in the device's external storage. The content to be sought is determined by the MIME type given in `.setType(…)`

```
Intent intent = new Intent();

intent.setType("image/pictures/*");
intent.setAction(Intent.ACTION_GET_CONTENT);

startActivity(intent);
```

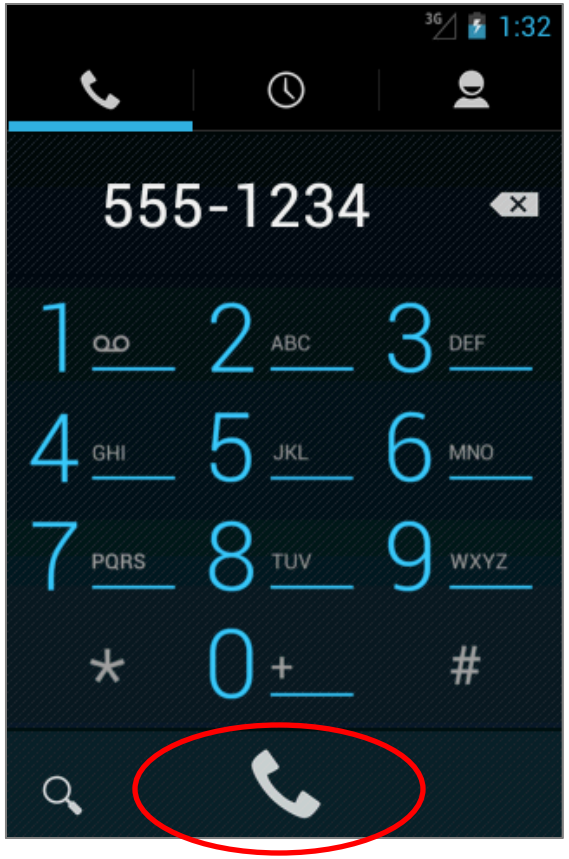
Intents

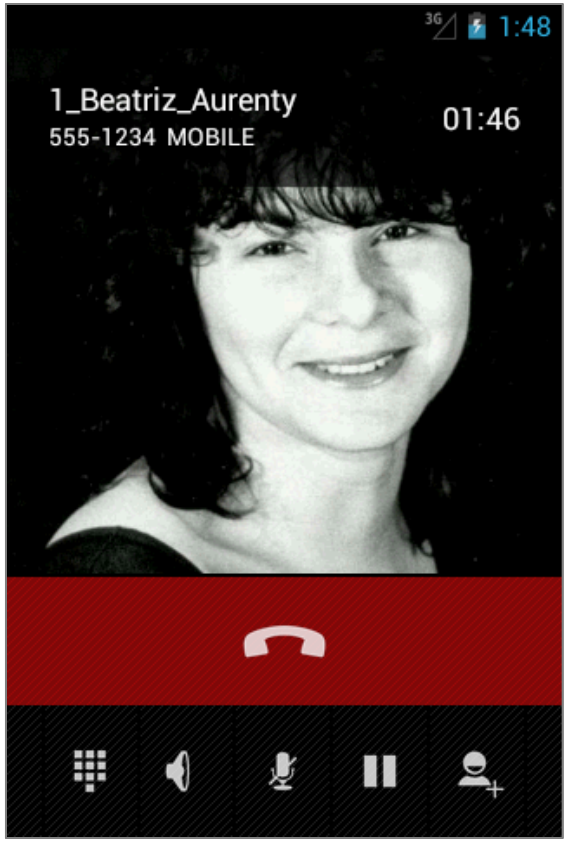
Ex5. A Complete Example: **`IntentDemo1`** displays an interface to accept a phone number and requests (built-in) Activity2 to make the call.


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#FF33B5E5"
        android:text="ACTION_VIEW "
        android:textSize="20sp"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/txtUriString"
        android:layout_width="match_parent"
        android:layout_height="54dp"
        android:hint="Enter URI (eg. tel:555-1234 )"
        android:textSize="18sp" />

    <Button
        android:id="@+id/btnCallActivity2"
        android:layout_width="149dp"
        android:layout_height="wrap_content"
        android:text=" Go "
        android:textStyle="bold" />
</LinearLayout>
```

Intents

Ex5. A Complete Example: **`IntentDemo1`** displays an interface to accept a phone number and requests (built-in) Activity2 to make the call.





Intents

Ex5. A Complete Example: **`IntentDemo1`** displays an interface to accept a phone number and requests (built-in) Activity to make the call.

```
public class IntentDemo1B extends Activity {
   EditText txtUriString;
   Button btnCallActivity2;
   Context context = getApplication();
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      try {
            txtUriString = (EditText) findViewById(R.id.txtUriString);

            btnCallActivity2 = (Button) findViewById(R.id.btnCallActivity2);

            btnCallActivity2.setOnClickListener(new MyClickHandler());

      } catch (Exception e) {
            Toast.makeText(context, e.getMessage(), Toast.LENGTH_LONG).show();
      }
   }// onCreate
```

Intents

Ex5. A Complete Example: **`IntentDemo1`** displays an interface to accept a phone number and requests (built-in) Activity2 to make the call.

```
private class MyClickHandler implements OnClickListener {
  @Override
  public void onClick(View v) {
    try {

        String myData = txtUriString.getText().toString();

        Intent myActivity2 = new Intent(Intent.ACTION_DIAL,
                                        Uri.parse(myData));
        startActivity(myActivity2);


    } catch (Exception e) {
        Toast.makeText(context, e.getMessage(),  Toast.LENGTH_LONG).show();
    }
  }// onClick

}// ClickHandler
```

Intents

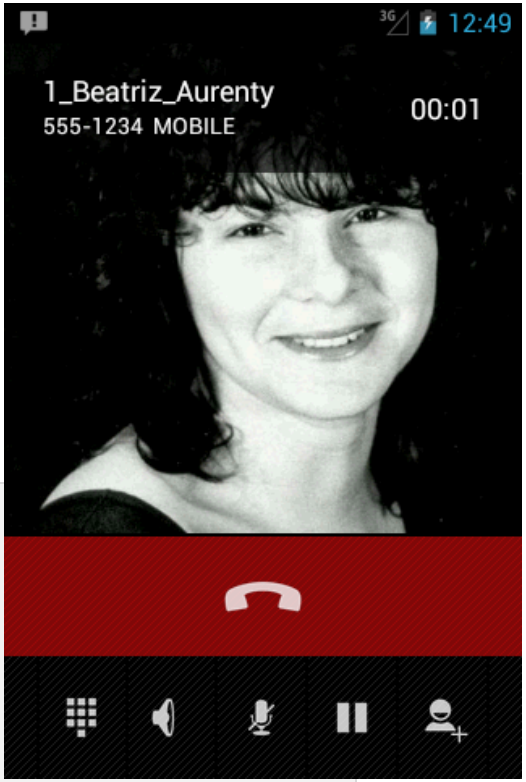
Ex5. A Complete Example: **`IntentDemo1`** displays an interface that accepts from the user a phone number and requests (built-in) Activity2 to make the call.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="csu.matos.intentdemo1b"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />
    <uses-permission android:name="android.permission.CALL_PHONE" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".IntentDemo1B"
            android:label="@string/title_activity_intent_demo1_b" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Intents

Example 6. ACTION_CALL

Placing and immediate phone call

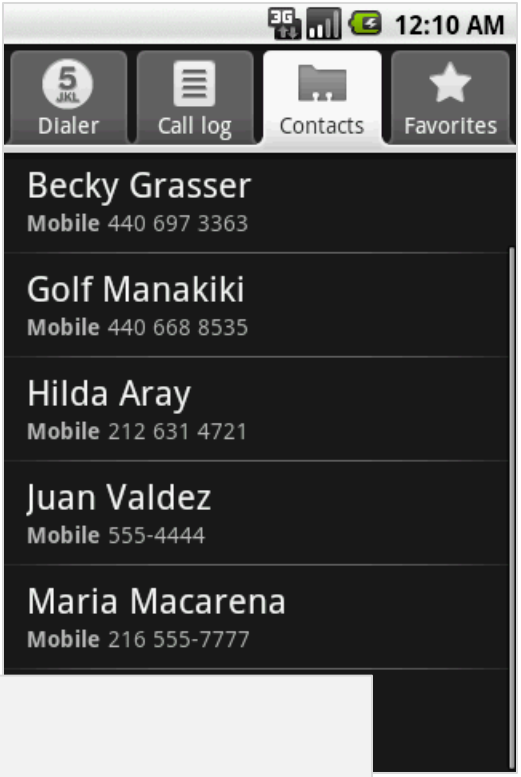



```
String myData = "tel:555-1234";

Intent myActivity2 = new Intent(
                        Intent.ACTION_CALL,
                        Uri.parse(myData));

startActivity(myActivity2);
```

Needs Permission:

```
<uses-permission android:name="android.permission.CALL_PHONE" />
```

Intents



Example 7. ACTION_VIEW

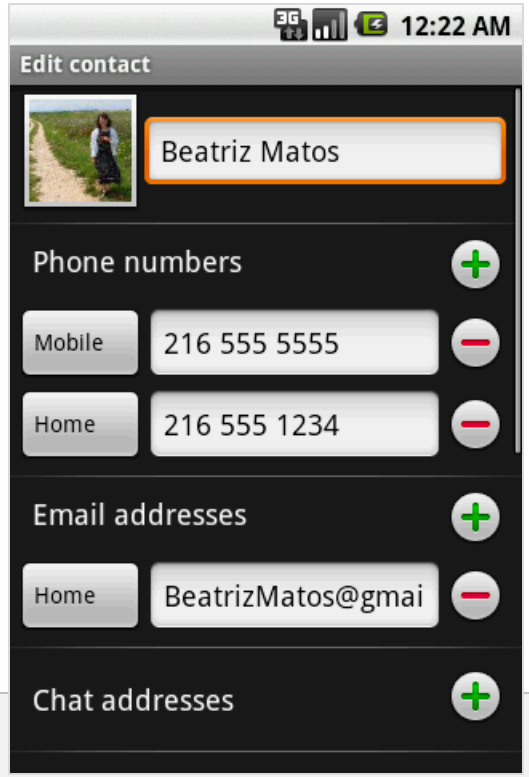
Showing all Contacts stored in your device

```
String myData = "content://contacts/people/";

Intent myActivity2 = new Intent(Intent.ACTION_VIEW,
                                Uri.parse(myData));

startActivity(myActivity2);
```

Intents

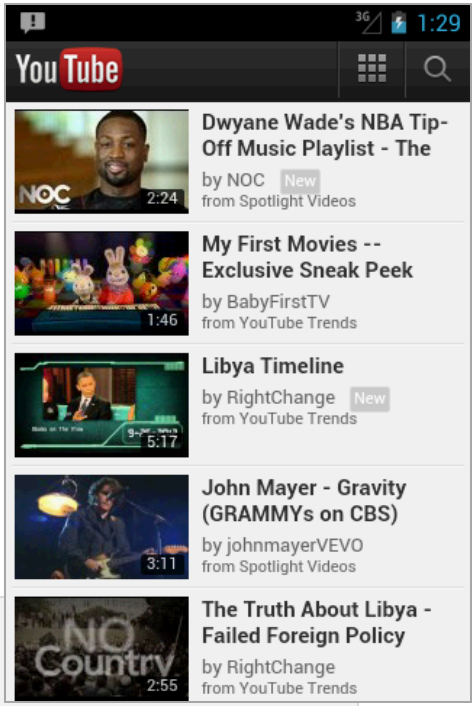
Example 8. ACTION_EDIT

Selecting a particular person (ID 2) from the contact list for editing purposes. Later in this lesson we will learn how to obtain the ID of stored contacts (music tracks, pictures, etc).




```
startActivity(myActivity2);

String myData = ContactsContract.Contacts.CONTENT_URI + "/" + "2";

Intent myActivity2 = new Intent(Intent.ACTION_EDIT,
                                Uri.parse(myData));

startActivity(myActivity2);
```

Intents



Example 9. ACTION_VIEW

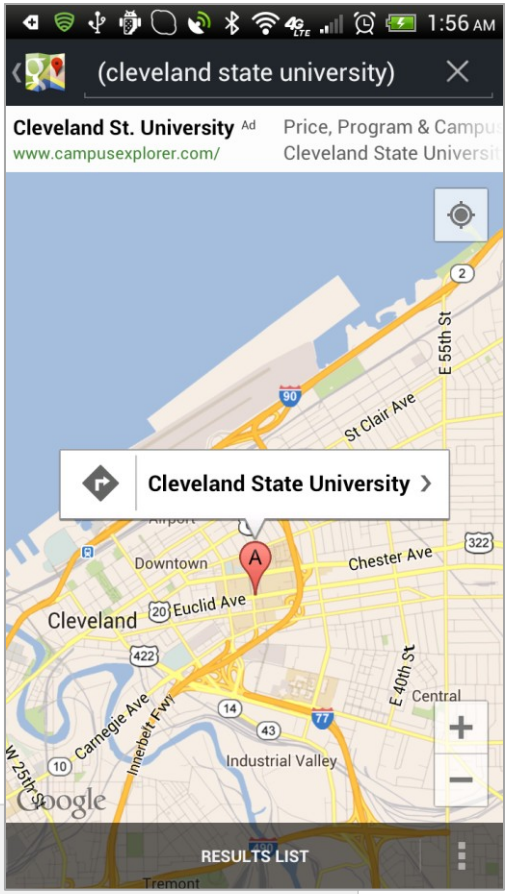
Viewing a web page. The user provides a valid URL pointing to the page.

```
String myUriString = "http://www.youtube.com";

Intent myActivity2 = new Intent(Intent.ACTION_VIEW,
                                Uri.parse(myUriString));
startActivity(myActivity2);
```

Caution. Add to the Manifest a request to use the Internet:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Intents



Example 10. ACTION_VIEW

Geo Mapping an Address / Place

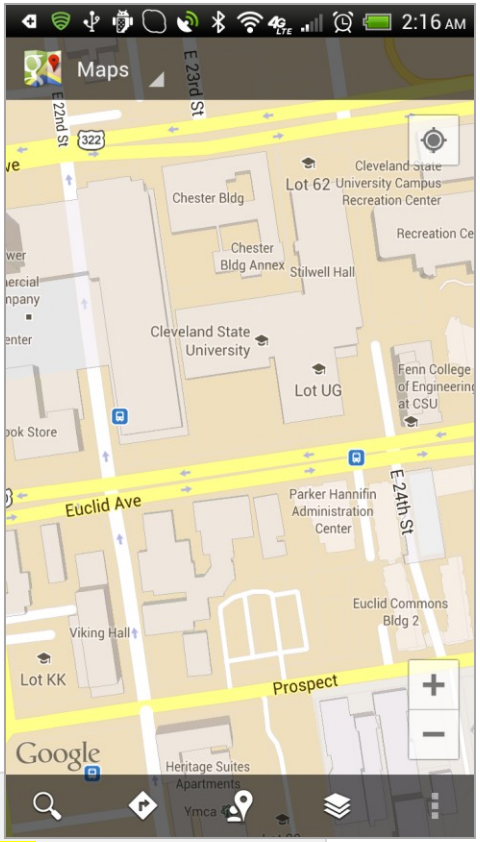
Provide a GeoCode expression holding a street address (or place, such as 'golden gate ca')

```
// (you may get multiple hits)
String thePlace = "Cleveland State University";
Intent intent = new Intent(android.content.Intent.ACTION_VIEW,
                    Uri.parse("geo:0,0?q=(" + thePlace + ")"));
startActivity(intent);
```

Modify the Manifest adding the following requests:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
```

Intents



Example 11. ACTION_VIEW

Geo Mapping Coordinates (latitude, longitude)

Provide a GeoCode holding latitude and longitude
(also an addittional zoom '**?z=xx**' with xx in range 1..23)

```
// map is centered aroung given Lat, Long
String geoCode = "geo:41.5020952,-81.6789717?z=16";
Intent intent = new Intent(Intent.ACTION_VIEW,
                           Uri.parse(geoCode));
startActivity(intent);
```

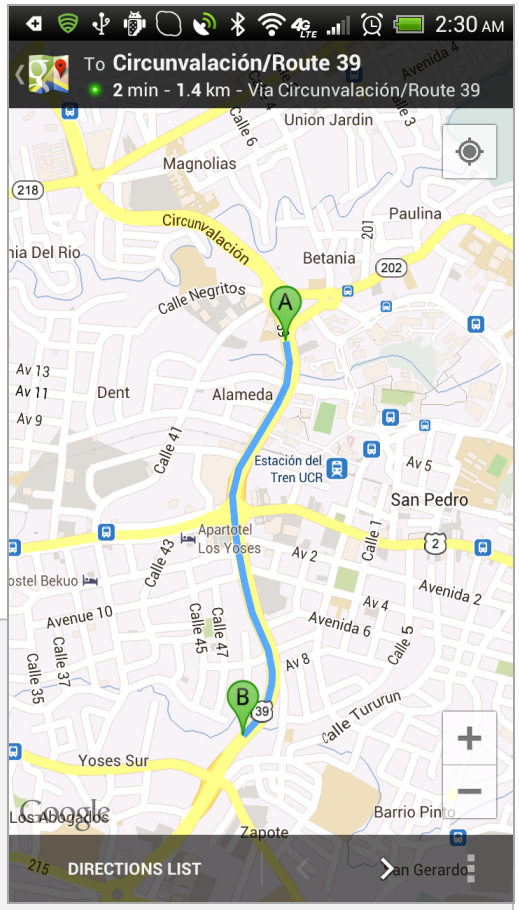
Modify the Manifest adding the following requests:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
```

Intents

Example 12. ACTION_VIEW
Getting driving directions

User provides GeoCodes (latitude,Longitude) for the starting and ending locations




```
Intent intent = new Intent(
        android.content.Intent.ACTION_VIEW,
        Uri.parse("http://maps.google.com/maps?"
        + "saddr=9.938083,-84.054430&"
        + "daddr=9.926392,-84.055964"));
startActivity(intent);
```

Intents

Example 13. ACTION_VIEW

Geo Mapping - Google StreetView

GeoCode Uri structure:

google.streetview:cbll=***latitude,longitude***
&cbp=1,*yaw*,,*pitch*,*zoom*&mz=*mapZoom*

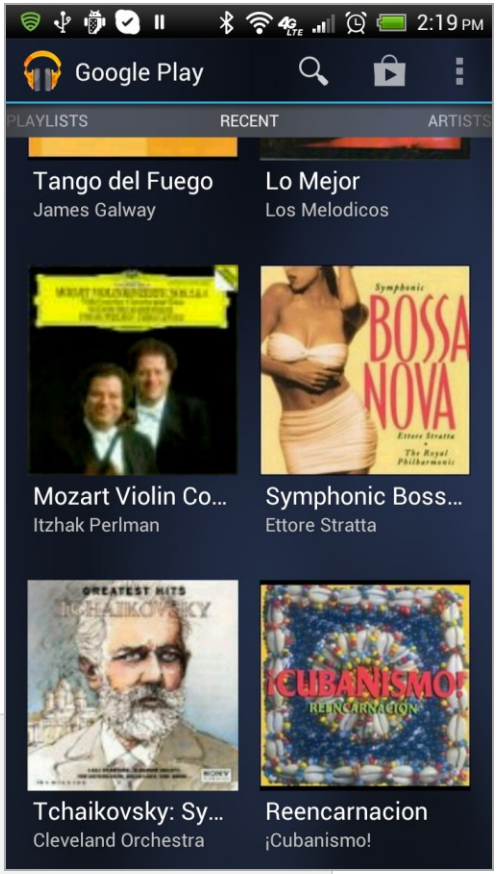
Reference: http://developer.android.com/guide/appendix/g-app-intents.html

```
String geoCode = "google.streetview:"
                + "cbll=41.5020952,-81.6789717&"
                + "cbp=1,270,,45,1&mz=7";
Intent intent = new Intent(Intent.ACTION_VIEW,
                           Uri.parse(geoCode));
startActivity(intent);
```

Modify the Manifest adding the following requests:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
```

Intents



Example 14.

Action Launching the Music Player

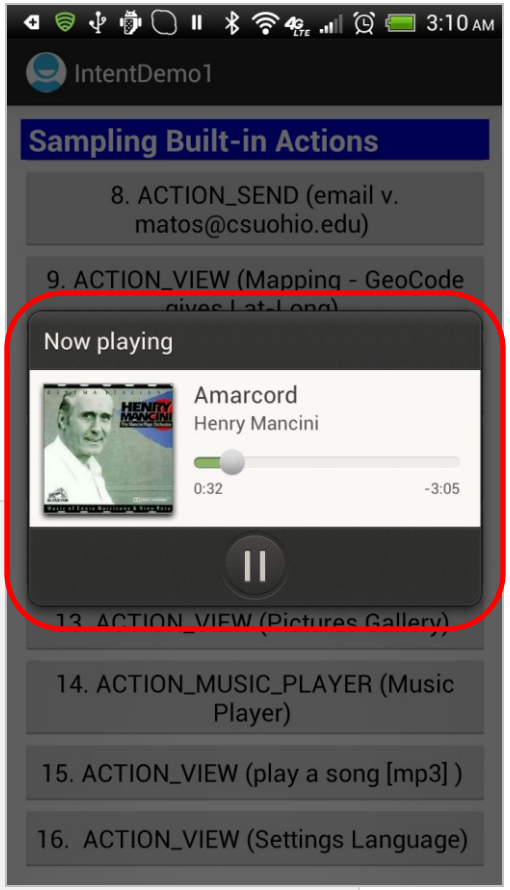
Reference: http://developer.android.com/guide/appendix/g-app-intents.html

```
//launch music player

Intent myActivity2 =
        new Intent("android.intent.action.MUSIC_PLAYER");

startActivity(myActivity2);
```

Note

See Appendix A & B for a list of other "`android.intent.action`…" intents.

Intents



Example 15.
ACTION_VIEW

Playing a song stored in the SD card

Reference: http://developer.android.com/guide/appendix/g-app-intents.html

```
Intent myActivity2 = new Intent(
                              Intent.ACTION_VIEW);
Uri data = Uri.parse("file://"
             + Environment
                .getExternalStorageDirectory()
                .getAbsolutePath()
             + "/Music/Amarcord.mp3");

myActivity2.setDataAndType(data, "audio/mp3";);

startActivity(myActivity2);
```

Add to Manifest:

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

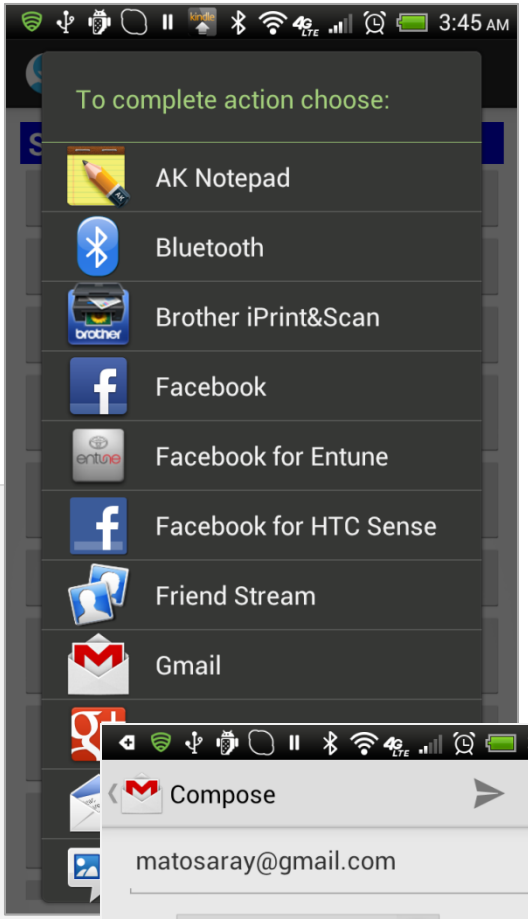
Intents

Example 17. ACTION_SENDTO

Sending Email

Reference: http://developer.android.com/guide/appendix/g-app-intents.html

```
// send email
String emailSender = "chairman@csuohio.edu";
String emailSubject = "Department Meetting";
String emailText = "We will discuss new curriculum "
                 + "on Tue. at 9:00am @ room BU340";
String emailReceiverList[] = {"v.matos@csuohio.edu"};

Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_EMAIL, emailReceiverList);
intent.putExtra(Intent.EXTRA_SUBJECT, emailSubject);
intent.putExtra(Intent.EXTRA_TEXT, emailText);
startActivity(Intent.createChooser(intent,
                    "To complete action choose:"));
```



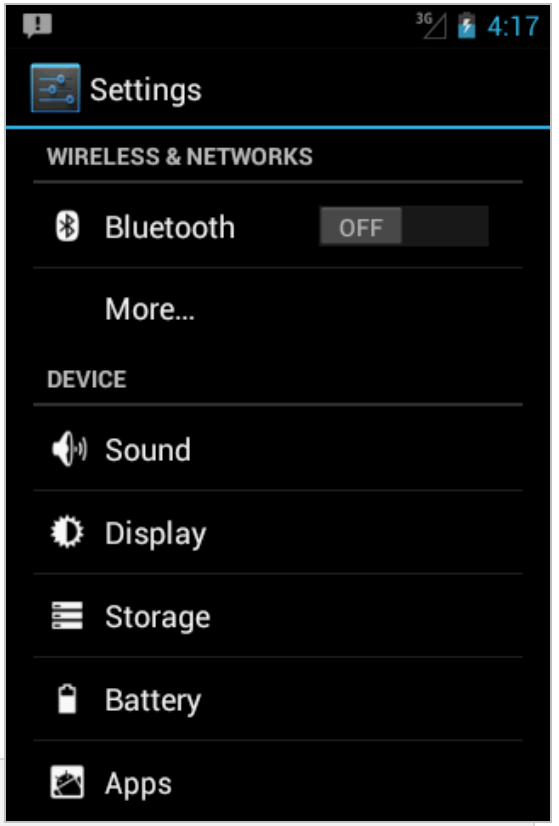
Intents

Example 18.

Setting System

Accessing system setting's UI

Reference: http://developer.android.com/reference/android/provider/Settings.html




```
Intent intent = new Intent(
            android.provider.Settings.ACTION_SETTINGS);

startActivity(intent);
```

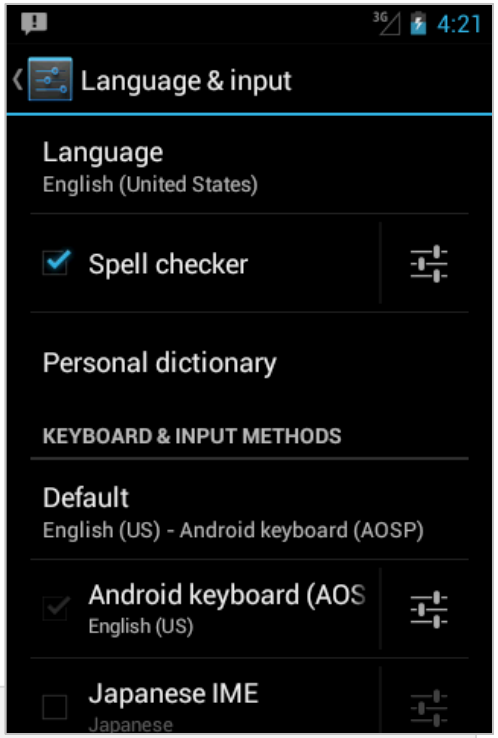
Intents

Example 19.

Setting System Locale:

Language & Keyboard

Reference: http://developer.android.com/reference/android/provider/Settings.html




```
Intent intent = new Intent(Intent.ACTION_MAIN);
intent.setClassName("com.android.settings",
                    "com.android.settings.LanguageSettings");

startActivity(intent);
```

Intents

Starting Activities and Getting Results

The **startActivity(Intent)** method is used to start a new activity, which will be placed at the top of the activity stack. The caller however continues to execute in its own thread.

Sometimes you want to get a result back from the called sub-activity when it ends.

For example, you may start an activity that let the user pick a person from a list of contacts; when it ends, it returns the person that was selected.

Intents

Starting Activities and Getting Results

In order to get results back from the called activity we use the method

startActivityForResult (Intent, requestCodeID **)**

Where *requestCodeID* is an arbitrary value you choose to identify the call (similar to a 'nickname').

The result sent by the sub-activity could be picked up through the listener-like asynchronous method

onActivityResult (requestCodeID, resultCode, Intent **)**

Intents

Starting Activities and Getting Results

- Before an invoked activity exits, it can call `setResult(resultCode)` to return a termination signal back to its parent.

- It is convenient to supply a result code, which can be the standard results
  `Activity.RESULT_CANCELED`,
  `Activity.RESULT_OK`,
  or any custom values.

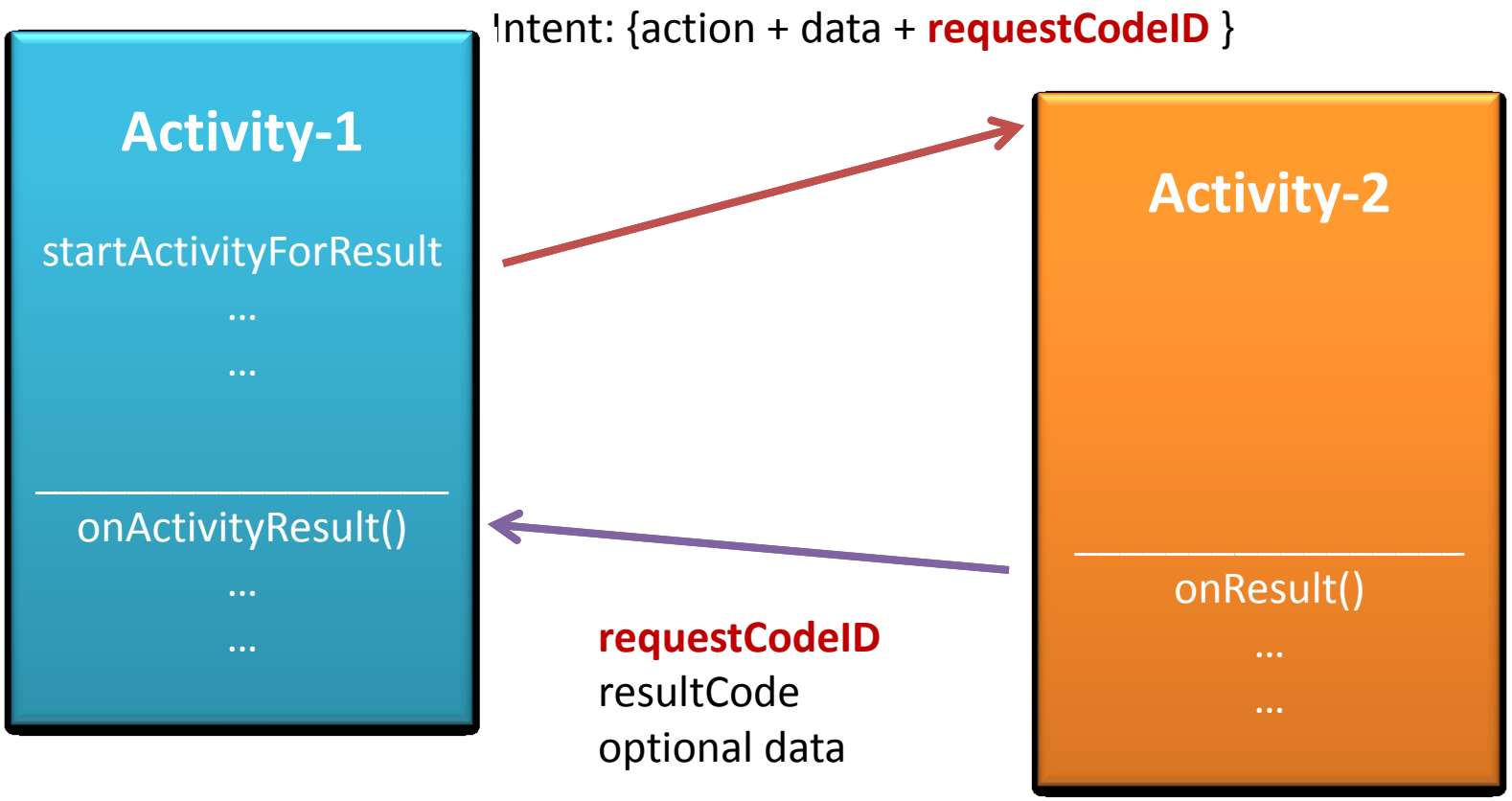
- All of this information can be capture back on the parent's

```
onActivityResult (int requestCodeID,
                  int resultCode,
                  Intent data)
```

- If a child activity fails for any reason (such as crashing), the parent activity will receive a result with the code `RESULT_CANCELED`.

Intents

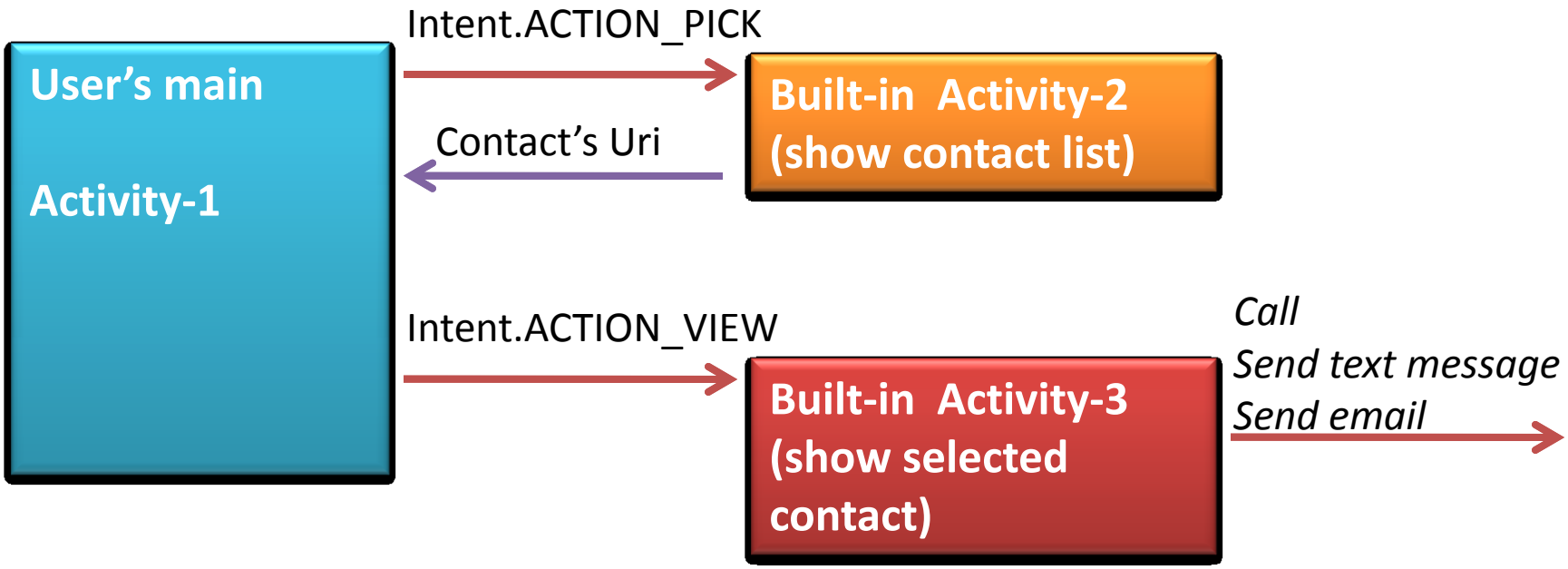
Starting Activities and Getting Results



Intents

Example 20. Let's play golf - Call for a tee-time.

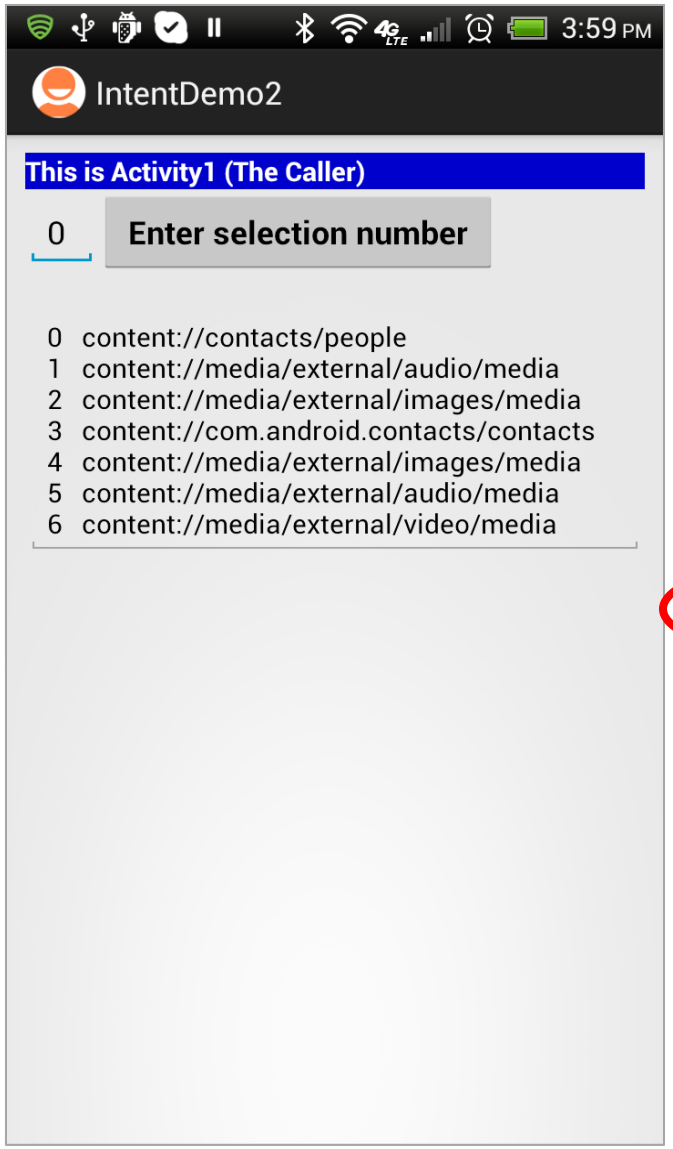
1. Show all contacts and pick a particular one (*Intent.ACTION_PICK*).
2. For a successful interaction the main-activity accepts the returned URI identifying the person we want to call (*content://contacts/people/n*).
3. 'Nicely' show the selected contact's entry allowing calling, texting, emailing actions (Intent.ACTION_VIEW).

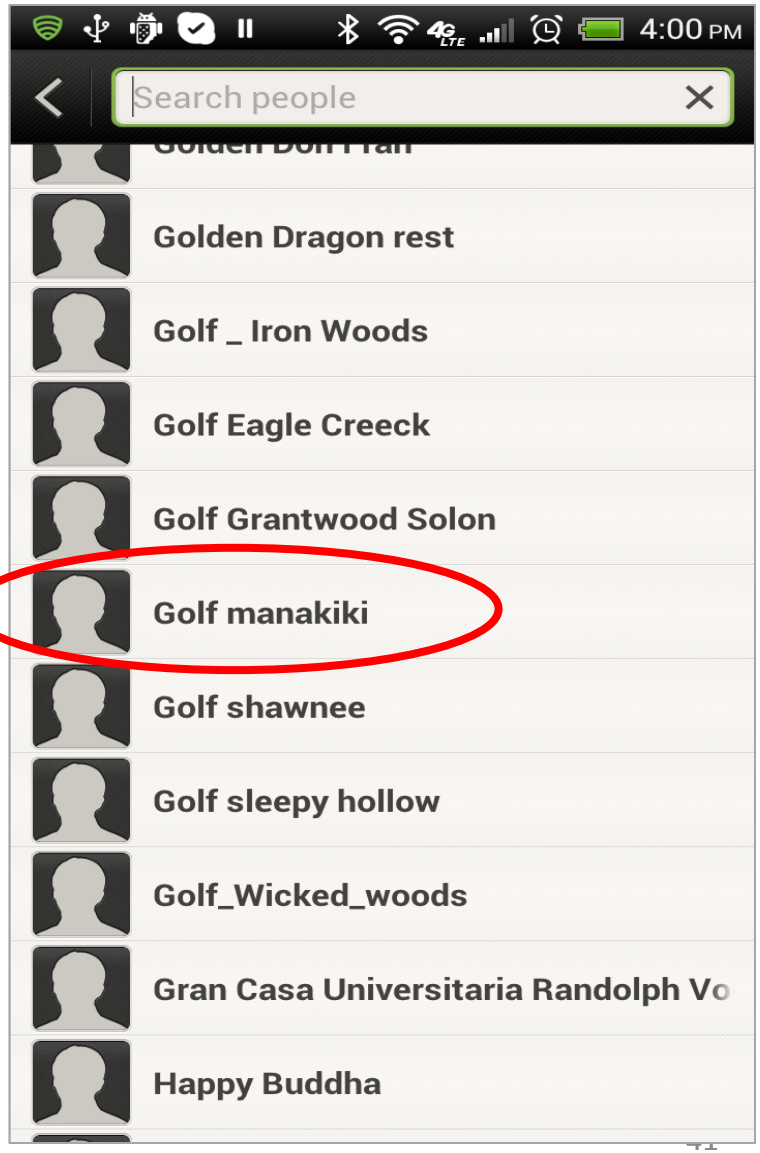


Intents

Cont.

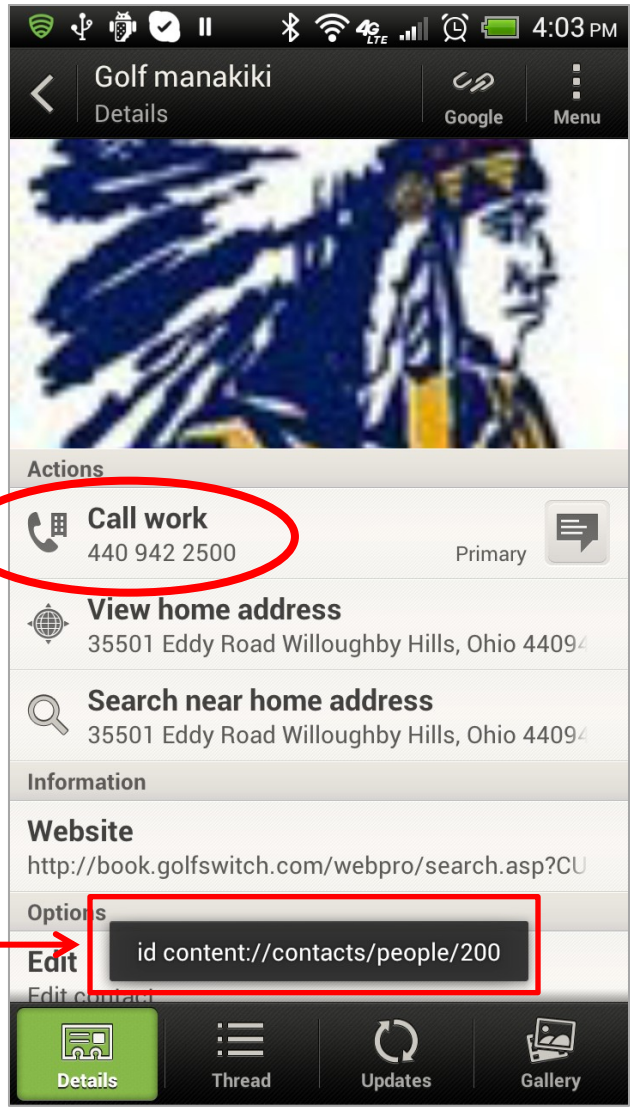
Example 20.
Let's play golf
Call for a tee-time.

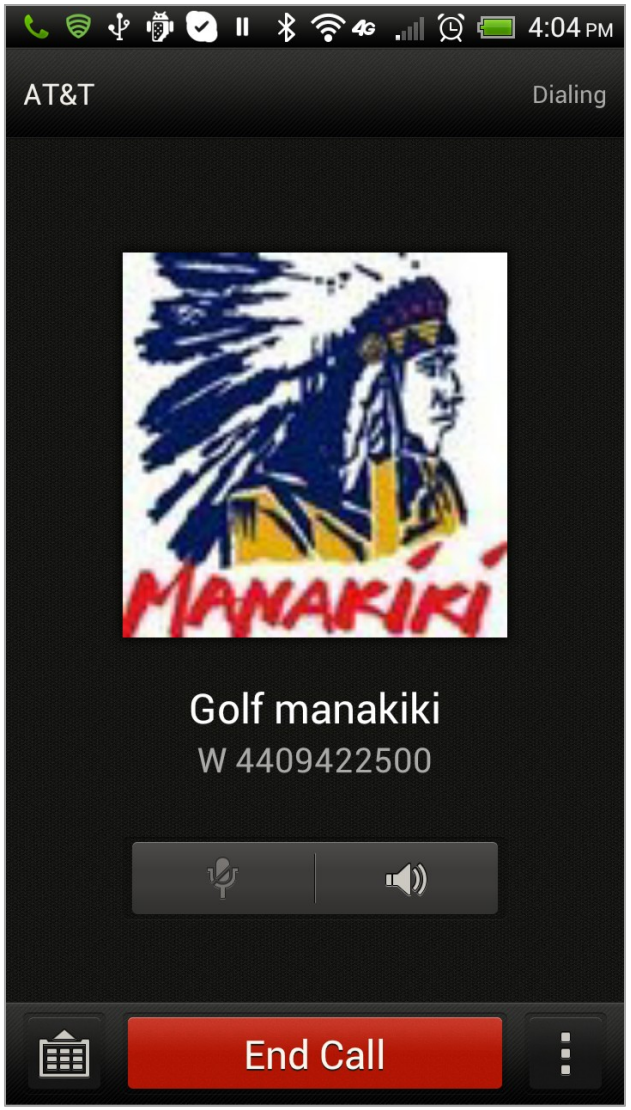




Intents

Example 20.
Let's play golf
Call for a tee-time.

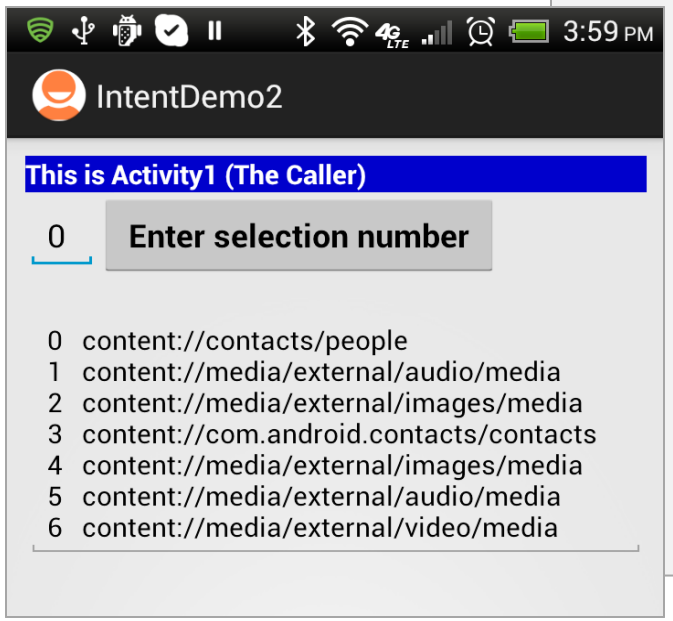




Intents

Example 20.
Calling a sub-activity, receiving results.

main.xml
1 of 2

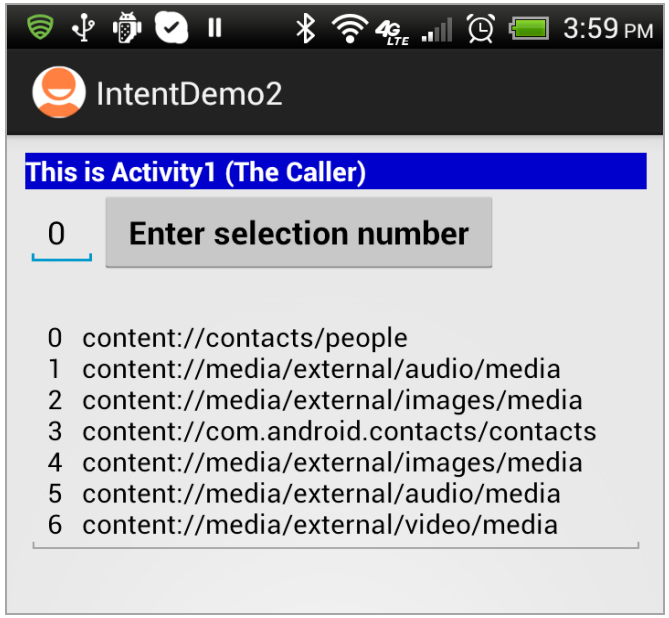



```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:padding="10dp" >

        <TextView
            android:id="@+id/txtMsg"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#ff0000cc"
            android:text="This is Activity1 (The Caller)"
            android:textColor="@android:color/primary_text_dark"
            android:textSize="15sp"
            android:textStyle="bold" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent" >
```

Intents

Example 20.
Calling a sub-activity, receiving results.

main.xml
1 of 2




```
            <EditText
                android:id="@+id/txtUriOption"
                android:layout_width="40dp"
                android:layout_height="wrap_content"
                android:inputType="number"
                android:text="0"
                android:textSize="18sp" />

            <Button
                android:id="@+id/btnOption"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text=" Enter selection number "
                android:textStyle="bold" />
        </LinearLayout>

        <EditText
            android:id="@+id/txtExamples"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text=""
            android:textSize="15dp" />

    </LinearLayout>

</ScrollView>
```

Intents

Example 20.

Let's play golf - Call for a tee-time. Calling a sub-activity, receiving results.

```
public class IntentDemo2 extends Activity {
  TextView txtMsg;
  EditText txtUri;
  EditText txtExample;
  Button btnCallActivity2;

  Uri[] uriProvider = {
      Uri.parse("content://contacts/people"),
      Uri.parse("content://media/external/audio/media"),
      Uri.parse("content://media/external/images/media"),
      android.provider.ContactsContract.Contacts.CONTENT_URI,
      android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
      android.provider.MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,
      android.provider.MediaStore.Video.Media.EXTERNAL_CONTENT_URI
  };

  @SuppressLint("NewApi")
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
```

Intents

Example 20.

Let's play golf - Call for a tee-time. Calling a sub-activity, receiving results.

```
try {

    txtMsg = (TextView) findViewById(R.id.txtMsg);
    txtUri = (EditText) findViewById(R.id.txtUriOption);

    // show some examples of built-in content providers
    txtExample = (EditText) findViewById(R.id.txtExamples);
    for (int i=0; i<uriProvider.length; i++)
      txtExample.append( "\n" + i + "   "
                         + uriProvider[i].toString());

    btnCallActivity2 = (Button) findViewById(R.id.btnOption);
    btnCallActivity2.setOnClickListener(new ClickHandler());

  } catch (Exception e) {
    Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_LONG)
        .show();
  }

 }// onCreate
```

Intents

Example 20.

Let's play golf - Call for a tee-time. Calling a sub-activity, receiving results.

```
private class ClickHandler implements OnClickListener {
  @Override
  public void onClick(View v) {
    try {
      // start myActivity2.  Tell it that my nickname is 222

      int option = Integer.parseInt(txtUri.getText().toString());

      Intent myActivity2 = new Intent( Intent.ACTION_PICK,
                                       uriProvider[option]);

      startActivityForResult(myActivity2, 222);

    } catch (Exception e) {
      Toast.makeText( getBaseContext(), e.getMessage(),
                      Toast.LENGTH_LONG).show();
    }
  }// onClick
}// ClickHandler
```

Intents

Example 20.

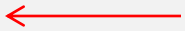
Let's play golf - Call for a tee-time. Calling a sub-activity, receiving results.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
  super.onActivityResult(requestCode, resultCode, data);
  try {
    // use requestCode to find out who is talking to us
    switch (requestCode) {
    case (222): {
      // 222 is our friendly contact-picker activity
      if (resultCode == Activity.RESULT_OK) {

        String returnedData = data.getDataString();

        Toast.makeText(getApplication(), "id " + returnedData, 1).show();

        // it will return an URI that looks like:
        // content://contacts/people/n
        // where n is the selected contact's ID
        txtMsg.setText(returnedData.toString());

        // show a 'nice' screen with the selected contact
        Toast.makeText( getApplication(), "Nice UI for\n "
                        + returnedData, 1).show();
```

Intents

Example 20.

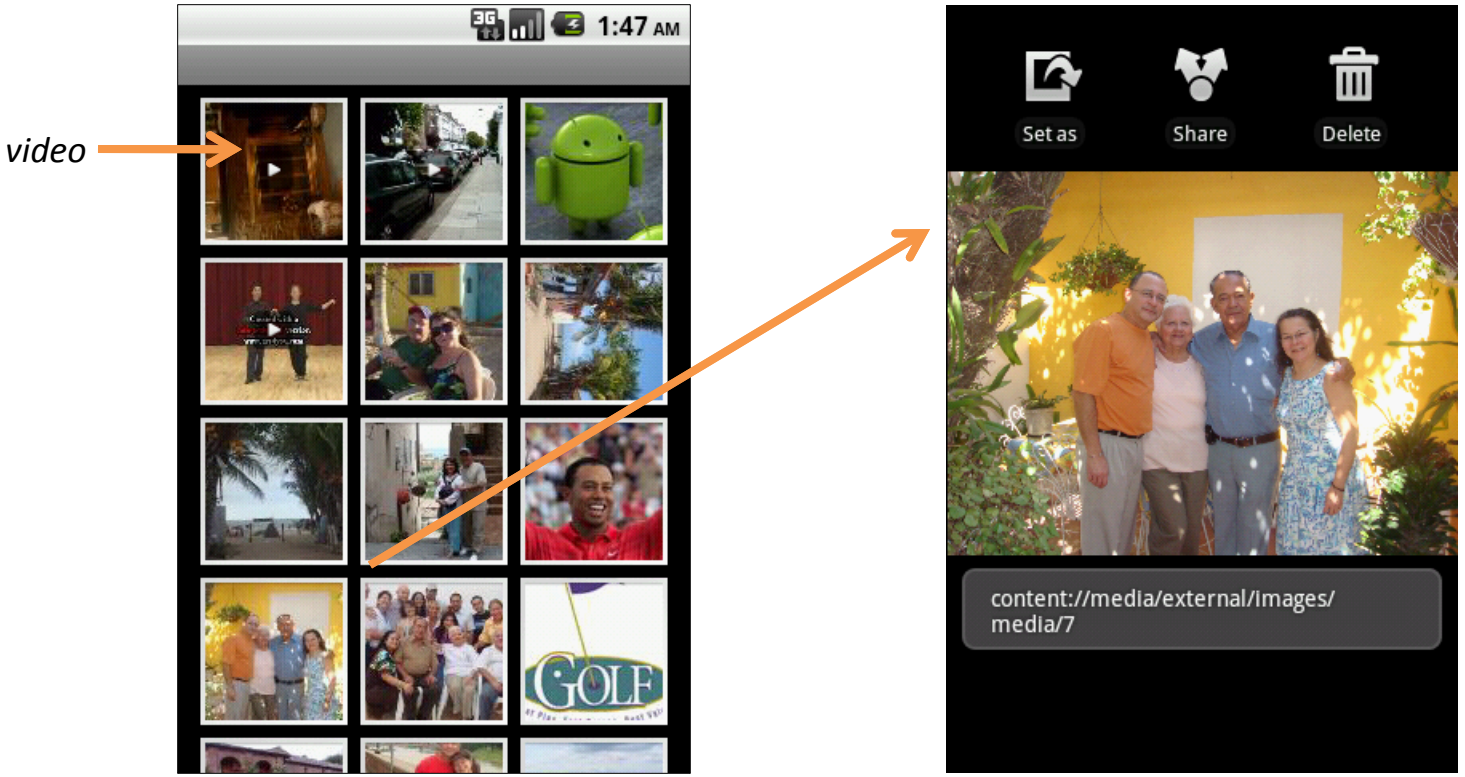
Let's play golf - Call for a tee-time. Calling a sub-activity, receiving results.



```
        Intent myAct3 = new Intent( Intent.ACTION_VIEW,
                                    Uri.parse(returnedData) );
        startActivity(myAct3);

      } else {
        // user pressed the BACK button to end called activity
        txtMsg.setText("Selection CANCELLED " + requestCode + " "
                       + resultCode);
      }
      break;
    }
    }// switch
  } catch (Exception e) {
    Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_LONG)
        .show();
  }
}// onActivityResult

}// IntentDemo2
```

Intents

Example20. Showing Pictures and Video - Calling a sub-activity, receiving results.
For this example we selected option *content://media/external/images/media*

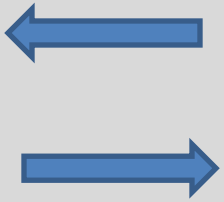


Intents

Android Bundles

- Android applications may include multiple inter-active activities.
- The Android **Bundle** container is a simple mechanism used to pass data between co-operating activities.
- A **Bundle** is a type-safe MAP collection of **<name, value>** pairs.
- There is a set of ***putXXX*** and ***getXXX*** methods to store and retrieve (single and array) values of primitive data types from/to the bundles. For example

```
Bundle myBundle = new Bundle();
myBundle.putDouble ("var1", 3.1415);
...
Double v1 = myBundle.getDouble("var1");
```

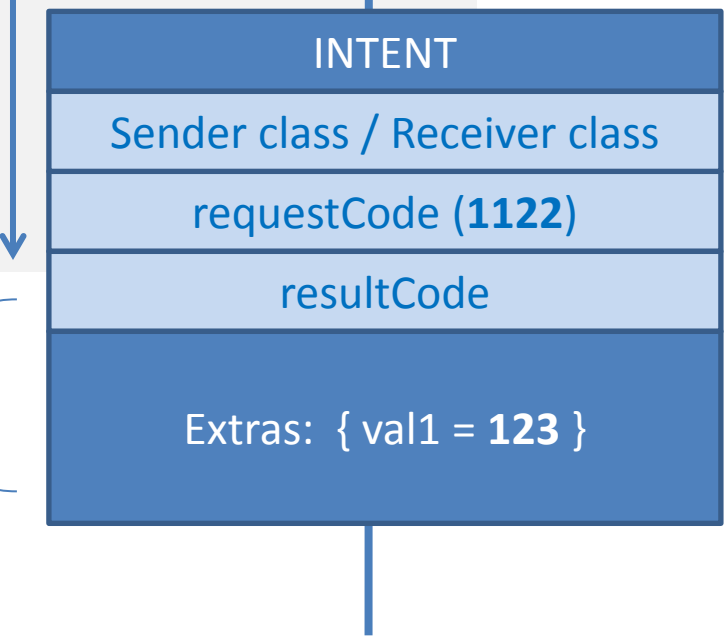


Intents

Android Intents & Bundles

Activity1: Sender → **Activity2: Receiver**

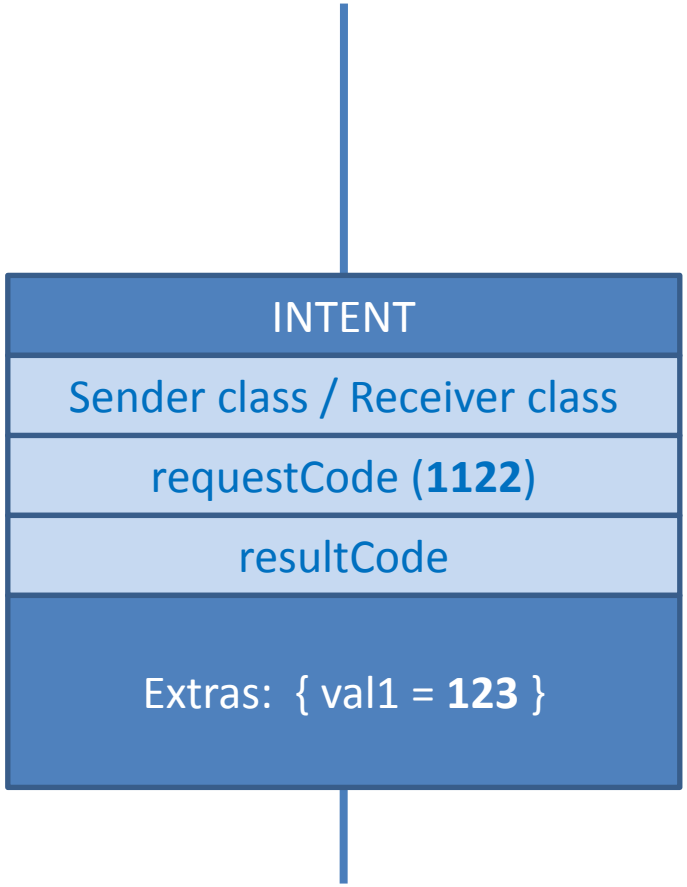
```
Intent myIntentA1A2 = new Intent  (Activity1.this,
                                   Activity2.class);

Bundle myBundle1 = new Bundle();

myBundle1.putInt ("val1", 123);

myIntentA1A2.putExtras(myBundle1);

startActivityForResult(myIntentA1A2, 1122);
```



Intents

Android Intents & Bundles

Activity1: Sender



Activity2: Receiver

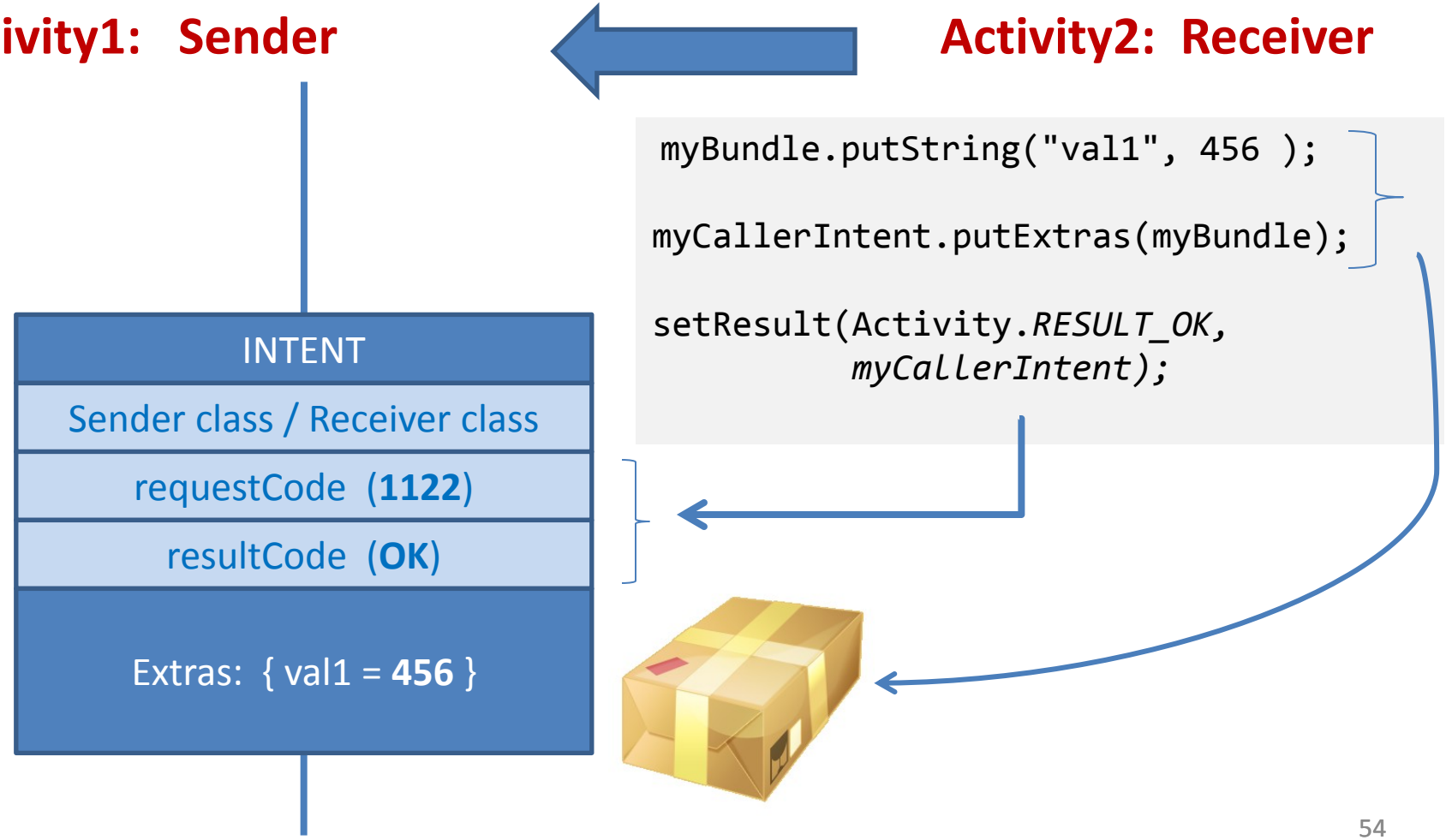
```
Intent myCallerIntent2 = getIntent();

Bundle myBundle = myCallerIntent.getExtras();

int val1 = myBundle.getInt("val1");
```

Intents

Android Intents & Bundles

Activity1: Sender

Activity2: Receiver



Intents

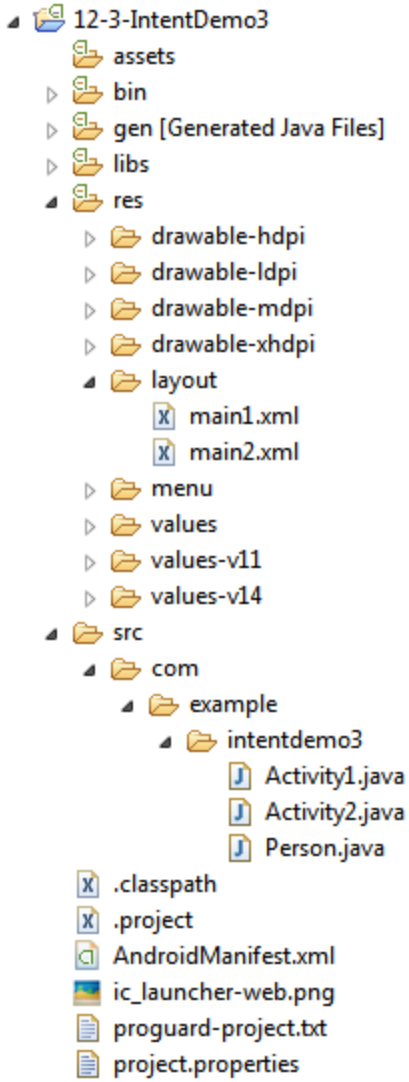
Android Bundles

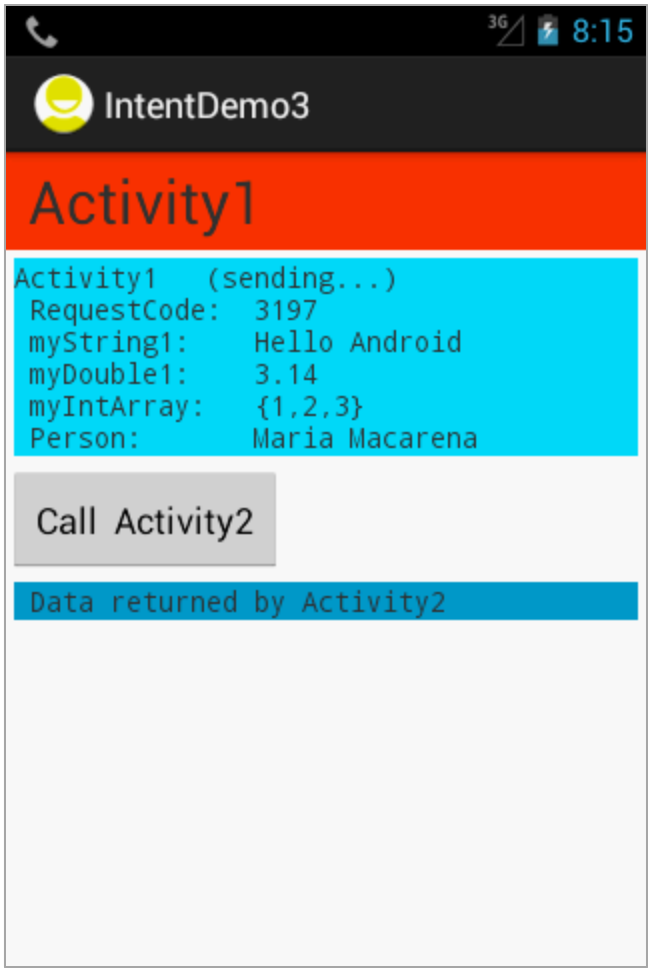
Available at: http://developer.android.com/reference/android/os/Bundle

Example of Public Methods	
void	clear() Removes all elements from the mapping of this Bundle.
Object	clone() Clones the current Bundle.
boolean	containsKey(String key) Returns true if the given key is contained in the mapping of this Bundle.
void	putIntArray(String key, int[] value) Inserts an int array value into the mapping of this Bundle, replacing any existing value for the given key.
void	putString(String key, String value) Inserts a String value into the mapping of this Bundle, replacing any existing value for the given key.
void	putStringArray(String key, String[] value) Inserts a String array value into the mapping of this Bundle, replacing any existing value for the given key.
void	putStringArrayList(String key, ArrayList<String> value) Inserts an ArrayList value into the mapping of this Bundle, replacing any existing value for the given key.
void	remove(String key) Removes any entry with the given key from the mapping of this Bundle.
int	size() Returns the number of mappings contained in this Bundle.

Intents

Example 21. Exchanging data between two activities.

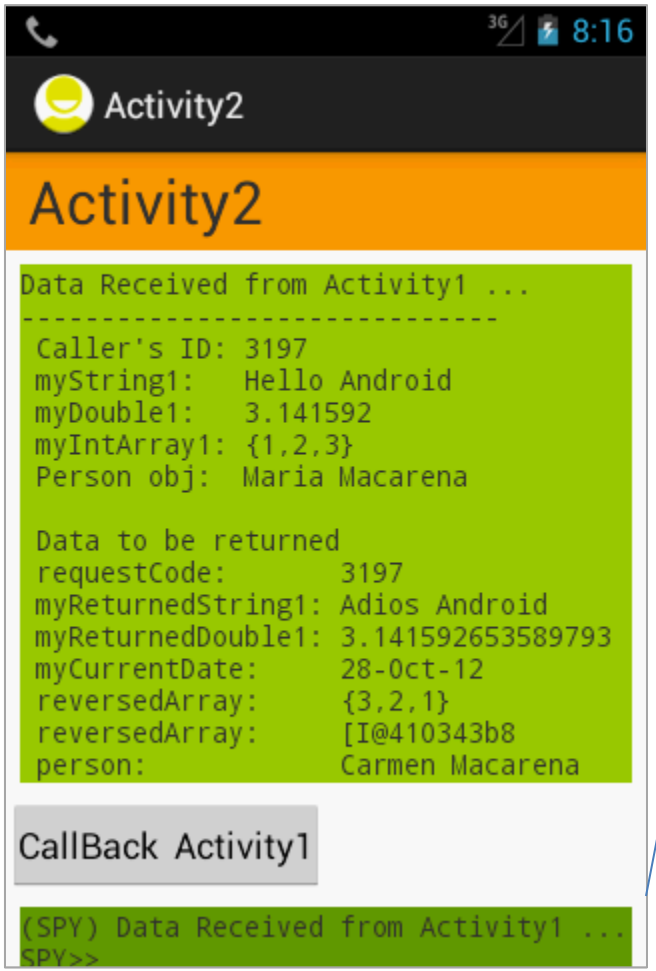




Send all of these items to **Activity2** and wait for

Intents

Example 21. Exchanging data between two activities.



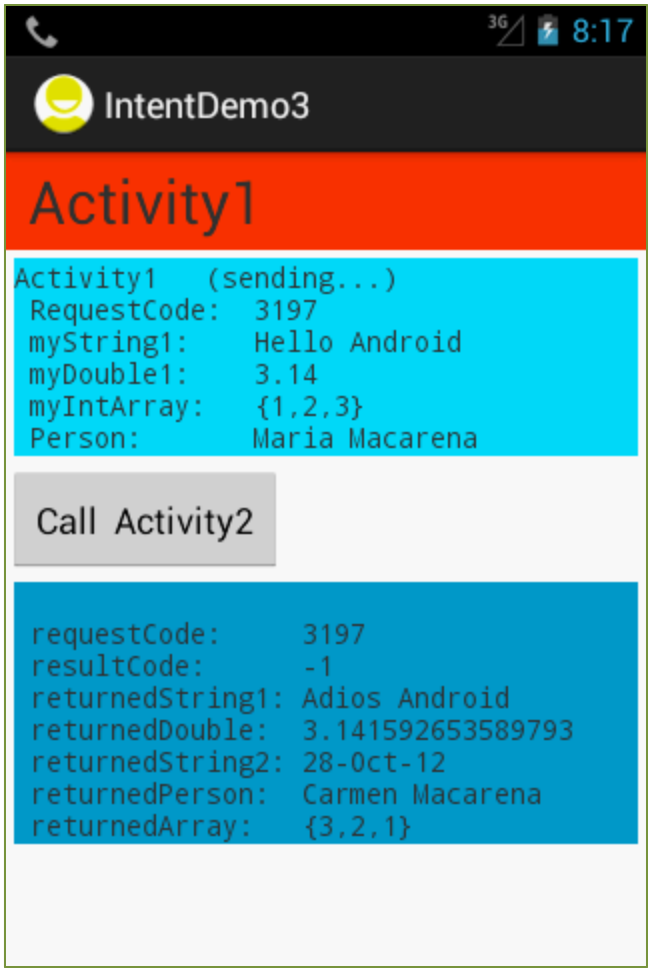
Echo data received
Do local operations (reverse array, change Person's name, return PI and current date)

Explore Bundle to obtain: keyName, keyValue, and keyType of each arriving item

Intents

Example 21. Exchanging data between two activities.

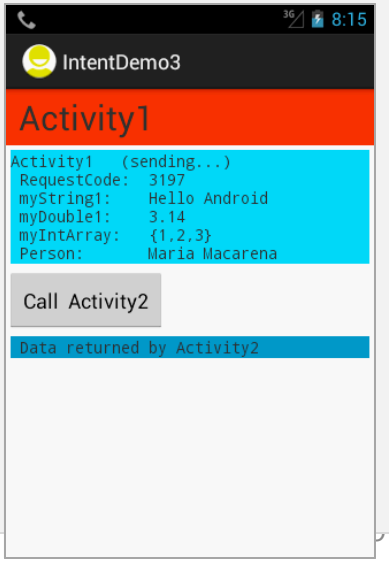
Returned values sent back from **Activity2**



Intents

Example 21. Exchanging data between two activities – Layout: **main1.xml**

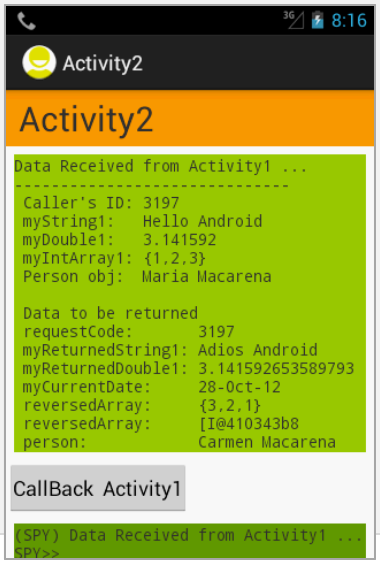
```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#ffff3300"
            android:padding="4sp"
            android:text=" Activity1 "
            android:textSize="30sp"  />

        <TextView
            android:id="@+id/txtTop"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="4dip"
            android:background=
                      "@color/holo_blue_bright"
            android:text=
               "Data to be sent to SubActivity:"
            android:typeface="monospace"  />

        <Button
            android:id="@+id/btnCallActivity2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="15dp"
            android:text="Call  Activity2"  />

        <TextView
            android:id="@+id/txtReturnedValues"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="4dip"
            android:background="@color/holo_blue_dark"
            android:text=" Data returned by Activity2"
            android:typeface="monospace"  />

    </LinearLayout>

</ScrollView>
```



Intents

Example 21. Exchanging data between two activities – Layout: **main2.xml**

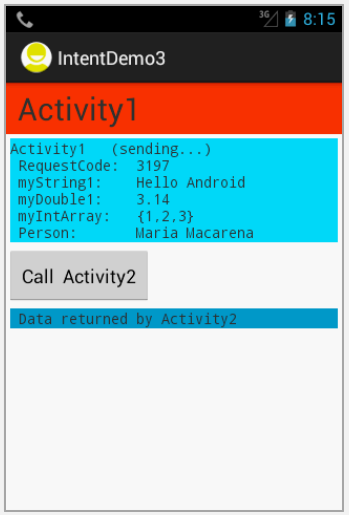
```
<?xml version="1.0" encoding="utf-8"?>

<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#ffff9900"
            android:padding="4sp"
            android:text=" Activity2"
            android:textSize="30sp" />

        <TextView
            android:id="@+id/txtIncomingData"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="7dip"
            android:background=
                          "@color/holo_green_light"
            android:text=
                  "Data Received from Activity1 ..."
            android:typeface="monospace" />

        <Button
            android:id="@+id/btnCallActivity1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="6sp"
            android:text="CallBack  Activity1" />

        <TextView
            android:id="@+id/spyBox"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="7dip"
            android:background="@color/holo_green_dark"
            android:text=
               "(SPY) Data Received from Activity1 ..."
            android:typeface="monospace" />
    </LinearLayout>

</ScrollView>
```



Intents

Example 21. Exchanging data between two activities – **Activity1** 1/5

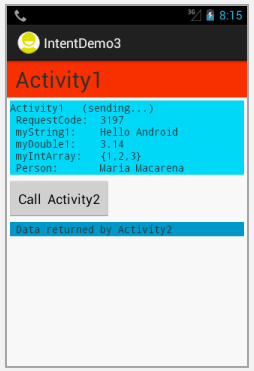
```
public class Activity1 extends Activity {
   TextView txtTop;
   TextView txtReturnedValues;
   Button btnCallActivity2;
   // arbitrary interprocess communication ID (just a nickname!)
   private final int IPC_ID = (int) (10001 * Math.random());

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      try {
         setContentView(R.layout.main1);
         txtTop = (TextView) findViewById(R.id.txtTop);
         txtReturnedValues = (TextView) findViewById(R.id.txtReturnedValues);
         btnCallActivity2 = (Button) findViewById(R.id.btnCallActivity2);
         btnCallActivity2.setOnClickListener(new Clicker1());
         // for demonstration purposes- show in top textBox
         txtTop.setText("Activity1   (sending...) "
               + "\n RequestCode:  " + IPC_ID
               + "\n myString1:    Hello Android"
               + "\n myDouble1:    3.14 "
               + "\n myIntArray:   {1,2,3} "
               + "\n Person:       Maria Macarena");
      } catch (Exception e) {
         Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_LONG).show();
      }
   }// onCreate
```



Intents

Example 21. Exchanging data between two activities – **Activity1** **2/5**

```
private class Clicker1 implements OnClickListener {
   public void onClick(View v) {
      try {
         // create an Intent to talk to Activity2
         Intent myIntentA1A2 = new Intent(Activity1.this,  Activity2.class);

         // prepare a Bundle and add the data pieces to be sent
         Bundle myData = new Bundle();
         myData.putInt("myRequestCode", IPC_ID);
         myData.putString("myString1", "Hello Android");
         myData.putDouble("myDouble1", 3.141592);
         int [] myLittleArray = { 1, 2, 3 };
         myData.putIntArray("myIntArray1", myLittleArray);

         // creating an object and passing it into the bundle
         Person p1 = new Person("Maria", "Macarena");
         myData.putSerializable("person", p1);

         // bind the Bundle and the Intent that talks to Activity2
         myIntentA1A2.putExtras(myData);

         // call Activity2 and wait for results
         startActivityForResult(myIntentA1A2, IPC_ID);

      } catch (Exception e) {
         Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_LONG).show();
      }
   }// onClick
}// Clicker1
```



Intents

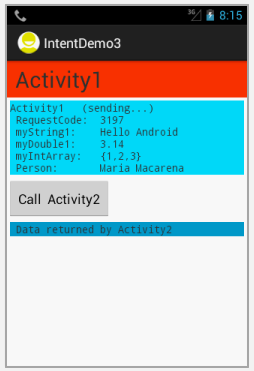
Example 21. Exchanging data between two activities – **Activity1** **3/5**

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
  super.onActivityResult(requestCode, resultCode, data);
  try {
    // check that these results are for me
    if (IPC_ID == requestCode) {

      // Activity2 is over - see what happened
      if (resultCode == Activity.RESULT_OK) {

        // good - we have some data sent back from Activity2
        Bundle myReturnedData = data.getExtras();
        String myReturnedString1 = myReturnedData
            .getString("myReturnedString1");
        Double myReturnedDouble1 = myReturnedData
            .getDouble("myReturnedDouble1");
        String myReturnedDate = myReturnedData
            .getString("myCurrentDate");
        Person myReturnedPerson = (Person) myReturnedData
            .getSerializable("person");

        int[]  myReturnedReversedArray = myReturnedData
                    .getIntArray("myReversedArray");
```

Intents

Example 21. Exchanging data between two activities – **Activity1** **4/5**

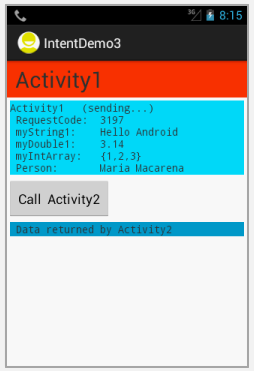
```
        // display in the bottom label
        txtReturnedValues.setText(
            "\n requestCode:     " + requestCode
          + "\n resultCode:      " + resultCode
          + "\n returnedString1: " + myReturnedString1
          + "\n returnedDouble:  " + Double.toString(myReturnedDouble1)
          + "\n returnedString2: " + myReturnedDate
          + "\n returnedPerson:  " + myReturnedPerson.getFullName()
          + "\n returnedArray:   "
          + Activity1.myConvertArray2String(myReturnedReversedArray));
      } else {
        // user pressed the BACK button
        txtTop.setText("Selection CANCELLED!");
      }

    }

  } catch (Exception e) {
    Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_LONG)
        .show();
  }// try
}// onActivityResult
```



Intents

Example 21. Exchanging data between two activities – **Activity1** **5/5**

```
   static String myConvertArray2String(int[] intArray ) {

      if ( intArray == null)
         return "NULL";

      String array2Str =  "{" + Integer.toString( intArray[0] );

      for (int i=1; i<intArray.length;  i++) {

         array2Str = array2Str + "," + Integer.toString( intArray[i] );
      }
      return array2Str + "}";
   }

}// AndroIntent1
```



Intents

Example 21. Exchanging data between two activities – **Activity2** 1/6


```
public class Activity2 extends Activity {
    TextView txtIncomingData;
    TextView spyBox;
    Button   btnCallActivity1;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main2);
        //bind UI variables to Java code
        txtIncomingData = (TextView)findViewById(R.id.txtIncomingData);
        spyBox = (TextView)findViewById(R.id.spyBox);
        btnCallActivity1 = (Button)findViewById(R.id.btnCallActivity1);
        btnCallActivity1.setOnClickListener(new Clicker1());

        // create a local Intent handler – we have been called!
        Intent myCallerIntentHandler = getIntent();

        // grab the data package with all the pieces sent to us
        Bundle myBundle = myCallerIntentHandler.getExtras();

        // extract the individual data parts from the bundle
        // observe you know the individual keyNames
        int    paramInt = myBundle.getInt("myRequestCode");
        String paramString = myBundle.getString("myString1");
        double paramDouble = myBundle.getDouble("myDouble1");
        int[]  paramArray = myBundle.getIntArray("myIntArray1");
        Person paramPerson = (Person) myBundle.getSerializable("person");
        String personName = paramPerson.getFullName();
```

Intents

Example 21. Exchanging data between two activities – **Activity2** **2/6**

```
//for debugging purposes - show arriving data
    txtIncomingData.append("\n------------------------------ "
       + "\n Caller's ID: " + paramInt
       + "\n myString1:    " + paramString
       + "\n myDouble1:    " + Double.toString(paramDouble)
       + "\n myIntArray1: " + Activity1.myConvertArray2String(paramArray)
       + "\n Person obj:   " + paramPerson.getFullName()
         );

 // next method assumes you do not know the data-items keyNames
    String spyData = extractDataFromBundle( myBundle );
    spyBox.append(spyData);

    // ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    // do here something with the extracted data. For example,
    // reverse the values stored in the incoming integer array

    //int[] intReversedArray = myIntReverseArray( paramArray );
    int[] intReversedArray = myIntReverseArray( paramArray );
    String strReversedArray = Activity1.myConvertArray2String(intReversedArray);
    myBundle.putIntArray("myReversedArray", intReversedArray);

    // change the person's firstName
    paramPerson.setFirstName("Carmen" );
    myBundle.putSerializable("person", paramPerson);
```

Intents

Example 21. Exchanging data between two activities – **Activity2** **3/6**


```
        // Returning Results.
        // Go back to myActivity1 with some new data made/change here.
        myBundle.putString("myReturnedString1", "Adios Android");
        myBundle.putDouble("myReturnedDouble1", Math.PI);
        SimpleDateFormat formatter = new SimpleDateFormat("dd-MMM-yy");
        String now = formatter.format(new Date());
        myBundle.putString("myCurrentDate", now );

        myCallerIntentHandler.putExtras(myBundle);

        // just debugging - show returning data
        txtIncomingData.append("\n\n Data to be returned  "
             + "\n requestCode:       " + paramInt
             + "\n myReturnedString1: " + myBundle.getString("myReturnedString1")
             + "\n myReturnedDouble1: " + myBundle.getDouble("myReturnedDouble1")
             + "\n myCurrentDate:     " + myBundle.getString("myCurrentDate")
             + "\n reversedArray:     " + strReversedArray
             + "\n reversedArray:     " + myBundle.getIntArray("myReversedArray")
             + "\n person:            " + ((Person) myBundle
                                              .getSerializable("person"))
                                              .getFullName()  );
        // all done!
        setResult(Activity.RESULT_OK, myCallerIntentHandler);

    }//onCreate
```

Intents

Example 21. Exchanging data between two activities – **Activity2** **4/6**

```
 private class Clicker1 implements OnClickListener {
   public void onClick(View v) {
     //clear Activity2 screen so Activity1 could be seen
     finish();
   }//onClick
 }//Clicker1

// ////////////////////////////////////////////////////////
 private int[] myIntReverseArray( int[] theArray ) {
   int n = theArray.length;

   int[] reversedArray =  new int[n];
        for (int i=0; i< theArray.length; i++ ) {
         reversedArray[i] = theArray[n -i -1];
        }
        return reversedArray;
 }
```

Intents

Example 21. Exchanging data between two activities – **Activity2** 5/6

```
// ////////////////////////////////////////////////////////////////
private String extractDataFromBundle(Bundle myBundle) {
    // ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    // What if I don't know the key names?
    // what types are in the bundle?. This fragment shows
    // how to use bundle methods to extract its data.
    // SOME ANDROID TYPES INCLUDE:
    // class [I (array integers)
    // class [J (array long)
    // class [D (array doubles)
    // class [F (array floats)
    // class java.lang.xxx   (where xxx= Integer, Double, ...)
    // ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    // Remember, the Bundle is a set of <keyName, keyValue> pairs
    String spy = "\nSPY>>\n";

    Set<String> myKeyNames = myBundle.keySet();  //get all keyNames

    for (String keyName : myKeyNames){

     Serializable keyValue =  myBundle.getSerializable(keyName);

     String keyType = keyValue.getClass().toString();
     if (keyType.equals("class java.lang.Integer")){
       keyValue = Integer.parseInt(keyValue.toString());
     }
```

Intents

Example 21. Exchanging data between two activities – **Activity2** 6/6

```
        else if (keyType.equals("class java.lang.Double")){
          keyValue = Double.parseDouble(keyValue.toString());
        }
        else if (keyType.equals("class java.lang.Float")){
          keyValue = Float.parseFloat(keyValue.toString());
        }
        else if (keyType.equals("class [I")){
          int[] arrint = myBundle.getIntArray(keyName);
          int n = arrint.length;
          keyValue = arrint[n-1]; // show only the last!
        }
        else {
           keyValue = (String)keyValue.toString();
        }
        spy  +=  "\n\nkeyName..." + keyName
             +   "  \nKeyValue.." + keyValue
             +   "  \nKeyType..." + keyType ;
      }

      return spy;

  }//extractDataFromBundle

}//Activity2
```

Intents

Example 21. Exchanging data between two activities – **Activity2** **6/6**

```
public class Person implements Serializable {

   private static final long serialVersionUID = 1L;
   private String firstName;
   private String lastName;

   public Person(String firstName, String lastName) {
      super();
      this.firstName = firstName;
      this.lastName = lastName;
   }

   public String getFullName() {
      return  firstName + " " + lastName;
   }

   public void setFirstName(String firstName) {
      this.firstName = firstName;
   }
}
```

Intents

Example 21. Exchanging data between two activities – **Manifest**

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.intentdemo3"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="15" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".Activity1"
            android:label="IntentDemo3" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".Activity2"
            android:label="Activity2" >
        </activity>
    </application>
</manifest>
```

Intents

Questions ?

Intents

Appendix A. Built-In Intents 1/4

A list of built-in actions is in the file `…/android-sdk/platforms/platform-xx/data`

android.app.action.ADD_DEVICE_ADMIN
android.app.action.SET_NEW_PASSWORD
android.app.action.START_ENCRYPTION
android.bluetooth.adapter.action.REQUEST_DISCOVERABLE
android.bluetooth.adapter.action.REQUEST_ENABLE

android.intent.action.ALL_APPS
android.intent.action.ANSWER
android.intent.action.APP_ERROR
android.intent.action.ASSIST
android.intent.action.ATTACH_DATA
android.intent.action.BUG_REPORT
android.intent.action.CALL
android.intent.action.CALL_BUTTON
android.intent.action.CHOOSER
android.intent.action.CREATE_LIVE_FOLDER
android.intent.action.CREATE_SHORTCUT
android.intent.action.DELETE
android.intent.action.DIAL
android.intent.action.EDIT
android.intent.action.EVENT_REMINDER
android.intent.action.GET_CONTENT

Intents

Appendix A. Built-In Intents 2/4

A list of built-in actions is in the file `…/android-sdk/platforms/platform-xx/data`

android.intent.action.INSERT
android.intent.action.INSERT_OR_EDIT
android.intent.action.INSTALL_PACKAGE
android.intent.action.MAIN
android.intent.action.MANAGE_NETWORK_USAGE
android.intent.action.MEDIA_SEARCH
android.intent.action.MUSIC_PLAYER
android.intent.action.PASTE
android.intent.action.PICK
android.intent.action.PICK_ACTIVITY
android.intent.action.POWER_USAGE_SUMMARY
android.intent.action.RINGTONE_PICKER
android.intent.action.RUN
android.intent.action.SEARCH
android.intent.action.SEARCH_LONG_PRESS
android.intent.action.SEND
android.intent.action.SENDTO
android.intent.action.SEND_MULTIPLE
android.intent.action.SET_ALARM
android.intent.action.SET_WALLPAPER
android.intent.action.SYNC
android.intent.action.SYSTEM_TUTORIAL

Intents

Appendix A. Built-In Intents 3/4

A list of built-in actions is in the file `…/android-sdk/platforms/platform-xx/data`

android.intent.action.UNINSTALL_PACKAGE
android.intent.action.VIEW
android.intent.action.VOICE_COMMAND
android.intent.action.WEB_SEARCH

android.media.action.DISPLAY_AUDIO_EFFECT_CONTROL_PANEL
android.net.wifi.PICK_WIFI_NETWORK
android.nfc.action.NDEF_DISCOVERED
android.nfc.action.TAG_DISCOVERED
android.nfc.action.TECH_DISCOVERED
android.provider.calendar.action.HANDLE_CUSTOM_EVENT
android.search.action.SEARCH_SETTINGS

android.settings.ACCESSIBILITY_SETTINGS
android.settings.ADD_ACCOUNT_SETTINGS
android.settings.AIRPLANE_MODE_SETTINGS
android.settings.APN_SETTINGS
android.settings.APPLICATION_DETAILS_SETTINGS
android.settings.APPLICATION_DEVELOPMENT_SETTINGS
android.settings.APPLICATION_SETTINGS
android.settings.BLUETOOTH_SETTINGS

Intents

Appendix A. Built-In Intents 4/4

A list of built-in actions is in the file `…/android-sdk/platforms/platform-xx/data`

android.settings.DATE_SETTINGS
android.settings.DEVICE_INFO_SETTINGS
android.settings.DISPLAY_SETTINGS
android.settings.INPUT_METHOD_SETTINGS
android.settings.INPUT_METHOD_SUBTYPE_SETTINGS
android.settings.INTERNAL_STORAGE_SETTINGS
android.settings.LOCALE_SETTINGS
android.settings.LOCATION_SOURCE_SETTINGS
android.settings.MANAGE_ALL_APPLICATIONS_SETTINGS
android.settings.MANAGE_APPLICATIONS_SETTINGS
android.settings.MEMORY_CARD_SETTINGS
android.settings.NETWORK_OPERATOR_SETTINGS
android.settings.NFCSHARING_SETTINGS
android.settings.NFC_SETTINGS
android.settings.PRIVACY_SETTINGS
android.settings.QUICK_LAUNCH_SETTINGS
android.settings.SECURITY_SETTINGS
android.settings.SETTINGS
android.settings.SOUND_SETTINGS
android.settings.SYNC_SETTINGS
android.settings.USER_DICTIONARY_SETTINGS
android.settings.WIFI_IP_SETTINGS
android.settings.WIFI_SETTINGS
android.settings.WIRELESS_SETTINGS

android.speech.tts.engine.CHECK_TTS_DATA
android.speech.tts.engine.INSTALL_TTS_DATA

Intents

Appendix B. Built-In Broadcast Intents 1/6

A list of built-in actions is in the file `…/android-sdk/platforms/platform-xx/data`

android.app.action.ACTION_PASSWORD_CHANGED
android.app.action.ACTION_PASSWORD_EXPIRING
android.app.action.ACTION_PASSWORD_FAILED
android.app.action.ACTION_PASSWORD_SUCCEEDED
android.app.action.DEVICE_ADMIN_DISABLED
android.app.action.DEVICE_ADMIN_DISABLE_REQUESTED
android.app.action.DEVICE_ADMIN_ENABLED

android.bluetooth.a2dp.profile.action.CONNECTION_STATE_CHANGED
android.bluetooth.a2dp.profile.action.PLAYING_STATE_CHANGED
android.bluetooth.adapter.action.CONNECTION_STATE_CHANGED
android.bluetooth.adapter.action.DISCOVERY_FINISHED
android.bluetooth.adapter.action.DISCOVERY_STARTED
android.bluetooth.adapter.action.LOCAL_NAME_CHANGED
android.bluetooth.adapter.action.SCAN_MODE_CHANGED
android.bluetooth.adapter.action.STATE_CHANGED
android.bluetooth.device.action.ACL_CONNECTED
android.bluetooth.device.action.ACL_DISCONNECTED
android.bluetooth.device.action.ACL_DISCONNECT_REQUESTED
android.bluetooth.device.action.BOND_STATE_CHANGED
android.bluetooth.device.action.CLASS_CHANGED
android.bluetooth.device.action.FOUND

Intents

Appendix B. Built-In Broadcast Intents 2/6

A list of built-in actions is in the file `…/android-sdk/platforms/platform-xx/data`

android.bluetooth.device.action.NAME_CHANGED
android.bluetooth.device.action.UUID
android.bluetooth.devicepicker.action.DEVICE_SELECTED
android.bluetooth.devicepicker.action.LAUNCH
android.bluetooth.headset.action.VENDOR_SPECIFIC_HEADSET_EVENT
android.bluetooth.headset.profile.action.AUDIO_STATE_CHANGED
android.bluetooth.headset.profile.action.CONNECTION_STATE_CHANGED
android.bluetooth.input.profile.action.CONNECTION_STATE_CHANGED
android.bluetooth.pan.profile.action.CONNECTION_STATE_CHANGED

android.hardware.action.NEW_PICTURE
android.hardware.action.NEW_VIDEO
android.hardware.input.action.QUERY_KEYBOARD_LAYOUTS

android.intent.action.ACTION_POWER_CONNECTED
android.intent.action.ACTION_POWER_DISCONNECTED
android.intent.action.ACTION_SHUTDOWN
android.intent.action.AIRPLANE_MODE
android.intent.action.BATTERY_CHANGED
android.intent.action.BATTERY_LOW
android.intent.action.BATTERY_OKAY
android.intent.action.BOOT_COMPLETED
android.intent.action.CAMERA_BUTTON
android.intent.action.CONFIGURATION_CHANGED

Intents

Appendix B. Built-In Broadcast Intents 3/6

A list of built-in actions is in the file `…/android-sdk/platforms/platform-xx/data`

android.intent.action.DATA_SMS_RECEIVED
android.intent.action.DATE_CHANGED
android.intent.action.DEVICE_STORAGE_LOW
android.intent.action.DEVICE_STORAGE_OK
android.intent.action.DOCK_EVENT
android.intent.action.EXTERNAL_APPLICATIONS_AVAILABLE
android.intent.action.EXTERNAL_APPLICATIONS_UNAVAILABLE
android.intent.action.FETCH_VOICEMAIL
android.intent.action.GTALK_CONNECTED
android.intent.action.GTALK_DISCONNECTED
android.intent.action.HEADSET_PLUG
android.intent.action.INPUT_METHOD_CHANGED
android.intent.action.LOCALE_CHANGED
android.intent.action.MANAGE_PACKAGE_STORAGE
android.intent.action.MEDIA_BAD_REMOVAL
android.intent.action.MEDIA_BUTTON
android.intent.action.MEDIA_CHECKING
android.intent.action.MEDIA_EJECT
android.intent.action.MEDIA_MOUNTED
android.intent.action.MEDIA_NOFS
android.intent.action.MEDIA_REMOVED
android.intent.action.MEDIA_SCANNER_FINISHED

Intents

Appendix B. Built-In Broadcast Intents 4/6

A list of built-in actions is in the file `…/android-sdk/platforms/platform-xx/data`

android.intent.action.MEDIA_SCANNER_SCAN_FILE
android.intent.action.MEDIA_SCANNER_STARTED
android.intent.action.MEDIA_SHARED
android.intent.action.MEDIA_UNMOUNTABLE
android.intent.action.MEDIA_UNMOUNTED
android.intent.action.MY_PACKAGE_REPLACED
android.intent.action.NEW_OUTGOING_CALL
android.intent.action.NEW_VOICEMAIL
android.intent.action.PACKAGE_ADDED
android.intent.action.PACKAGE_CHANGED
android.intent.action.PACKAGE_DATA_CLEARED
android.intent.action.PACKAGE_FIRST_LAUNCH
android.intent.action.PACKAGE_FULLY_REMOVED
android.intent.action.PACKAGE_INSTALL
android.intent.action.PACKAGE_NEEDS_VERIFICATION
android.intent.action.PACKAGE_REMOVED
android.intent.action.PACKAGE_REPLACED
android.intent.action.PACKAGE_RESTARTED
android.intent.action.PHONE_STATE
android.intent.action.PROVIDER_CHANGED
android.intent.action.PROXY_CHANGE
android.intent.action.REBOOT

Intents

Appendix B. Built-In Broadcast Intents 5/6

A list of built-in actions is in the file `…/android-sdk/platforms/platform-xx/data`

android.intent.action.SCREEN_OFF
android.intent.action.SCREEN_ON
android.intent.action.TIMEZONE_CHANGED
android.intent.action.TIME_SET
android.intent.action.TIME_TICK
android.intent.action.UID_REMOVED
android.intent.action.USER_PRESENT
android.intent.action.WALLPAPER_CHANGED

android.media.ACTION_SCO_AUDIO_STATE_UPDATED
android.media.AUDIO_BECOMING_NOISY
android.media.RINGER_MODE_CHANGED
android.media.SCO_AUDIO_STATE_CHANGED
android.media.VIBRATE_SETTING_CHANGED
android.media.action.CLOSE_AUDIO_EFFECT_CONTROL_SESSION
android.media.action.OPEN_AUDIO_EFFECT_CONTROL_SESSION

android.net.conn.BACKGROUND_DATA_SETTING_CHANGED
android.net.nsd.STATE_CHANGED

Intents

Appendix B. Built-In Broadcast Intents 6/6

A list of built-in actions is in the file `…/android-sdk/platforms/platform-xx/data`

android.net.wifi.NETWORK_IDS_CHANGED
android.net.wifi.RSSI_CHANGED
android.net.wifi.SCAN_RESULTS
android.net.wifi.STATE_CHANGE
android.net.wifi.WIFI_STATE_CHANGED
android.net.wifi.p2p.CONNECTION_STATE_CHANGE
android.net.wifi.p2p.DISCOVERY_STATE_CHANGE
android.net.wifi.p2p.PEERS_CHANGED
android.net.wifi.p2p.STATE_CHANGED
android.net.wifi.p2p.THIS_DEVICE_CHANGED
android.net.wifi.supplicant.CONNECTION_CHANGE
android.net.wifi.supplicant.STATE_CHANGE
android.provider.Telephony.SIM_FULL
android.provider.Telephony.SMS_CB_RECEIVED
android.provider.Telephony.SMS_EMERGENCY_CB_RECEIVED
android.provider.Telephony.SMS_RECEIVED
android.provider.Telephony.SMS_REJECTED
android.provider.Telephony.SMS_SERVICE_CATEGORY_PROGRAM_DATA_RECEIVED
android.provider.Telephony.WAP_PUSH_RECEIVED
android.speech.tts.TTS_QUEUE_PROCESSING_COMPLETED
android.speech.tts.engine.TTS_DATA_INSTALLED