



# Lesson 10

## Android Intents

**Victor Matos**  
Cleveland State University

Portions of this page are reproduced from work created and [shared by Google](#) and used according to terms described in the [Creative Commons 3.0 Attribution License](#).

# Android Intents

## Applications, Activities and Intents

- An Android **application** could include any number of **activities**.
- The app's Manifest designates one of the activities as the first one that should be shown to the user when the application is launched (`android.intent.action.MAIN`).
- Usually, each activity is associated to a single screen.
- An *activity* uses the `setContentView(...)` method to show a given UI.
- Activities are independent of each other; however they usually cooperate exchanging data and actions.
- Activities interact with each other in an **asynchronous** mode.
- Passing control and data from one activity to another is accomplished by asking the current activity to execute an **intent**.

# Android Intents

Android  
Applications

App2

App1

Activity-1 (MainActivity)

Activity-2

Activity-n

results

intents

extras



Activities call each other using Intents. An intent may include basic and extra data elements. The called activity may return a result to the caller.

# Android Intents

## Invoking Intents for Execution

**Intents** are roughly equivalent to a procedure call in Java (however the caller does not wait for the subroutine to complete).

Intents are invoked using the following options

<code>startActivity (intent)</code>	launches an activity
<code>sendBroadcast (intent)</code>	sends an intent to any interested BroadcastReceivers
<code>startService(intent)</code> or <code>bindService(intent, ...)</code>	communicates with a background service.

# Android Intents

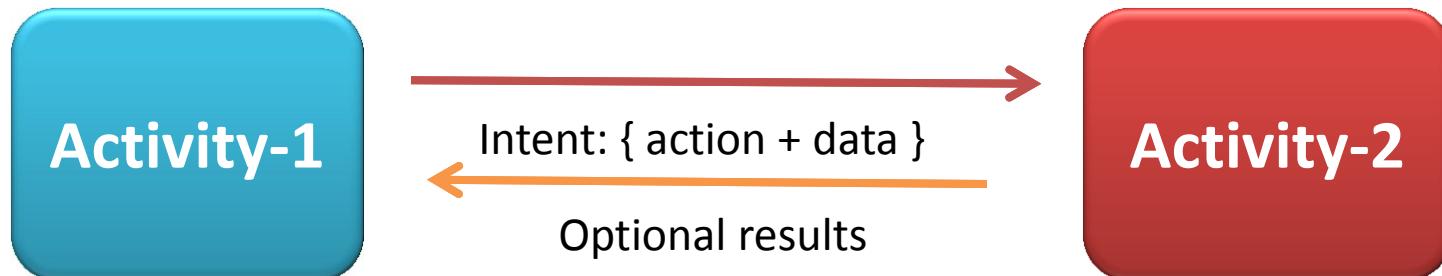
## Parts of an Intent

The two main components of an Intent are:

**Action** The built-in action to be performed, such as

`ACTION_VIEW, ACTION_EDIT, ACTION_CALL, ACTION_SENDTO, ...`  
or a *user-created-activity*

**Data** Basic argument needed by the intent to work. For instance: a phone number to be called , a picture to be shown, a message to be sent, etc.



# Android Intents

## Parts of an Intent

**Data** Data is supplied as an **URI**, i.e. a string whose prefix indicates the composition of the data item. For instance:

tel://,  
http://,  
mailto://,  
file://,  
content://,  
geo:,  
audio/,  
media/,  
vnd.android.cursor.dir

are common URIs used by Android (For a detailed list of all Intents see <http://www.openintents.org/intentsregistry/> )

# Android Intents

## Initiating an Intent

Typically an intent is called as follows:

```
Intent myOtherActivity = new Intent (action, data);  
startActivity (myOtherActivity);
```

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

Built-in or  
user-created activity

# Android Intents

## Examples of action/data pairs:

- |                    |   |
|--------------------|---|
| <b>ACTION_DIAL</b> | <b><i>tel://5551234</i></b> or <b><i>tel:5551234</i></b>  |
|                    | Display the phone dialer with the given number filled in.   |
| <b>ACTION_VIEW</b> | <b><i>http://www.google.com</i></b>   |
|                    | Show Google page in a browser view.   |
| <b>ACTION_EDIT</b> | <b><i>content://contacts/people/2</i></b>   |
|                    | Edit information about the contact person whose identifier is "2".  |
| <b>ACTION_VIEW</b> | <b><i>content://contacts/people/2</i></b>   |
|                    | Used to start an activity to display contact person whose identifier is "2".  |
| <b>ACTION_VIEW</b> | <b><i>content://contacts/people/</i></b>  |
|                    | Display a list of people, which the user can browse through. Selecting a particular person to view would result in a new intent |

# Android Intents

## Common Built-in Android Actions

List of common actions that Intents can use for launching built-in activities  
[usually through *startActivity(Intent)* ]

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

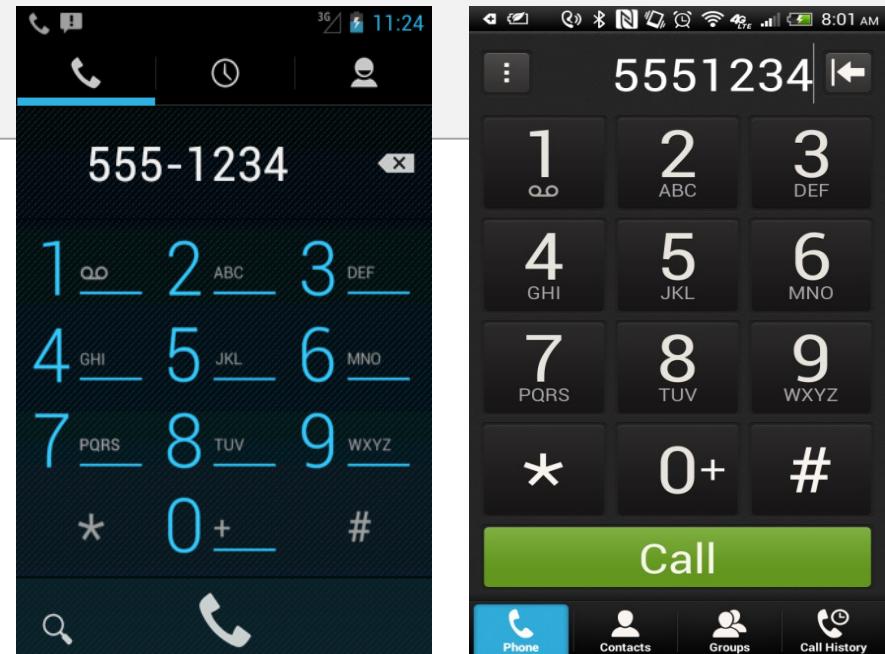
See Appendix A for a detailed list of selected built-in actions.

# Android Intents

## Example 1A: ACTION\_DIAL

**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);
```



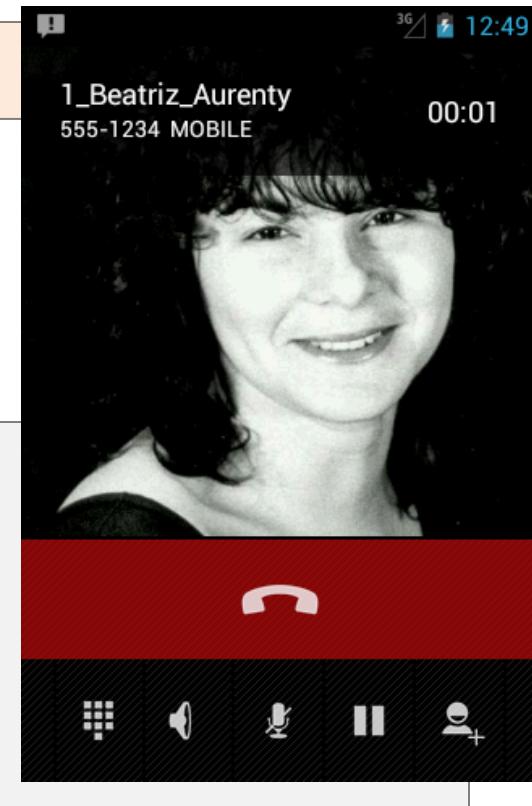
Images captured  
from emulator and  
device respectively

# Android Intents

## Example 1B: ACTION\_CALL

Placing an 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 - 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



```
CATEGORY_ALTERNATIVE : String - Intent  
CATEGORY_APP_BROWSER : String - Intent  
CATEGORY_APP_CALCULATOR : String - Intent  
CATEGORY_APP_CALENDAR : String - Intent  
CATEGORY_APP_CONTACTS : String - Intent  
CATEGORY_APP_EMAIL : String - Intent  
CATEGORY_APP_GALLERY : String - Intent  
CATEGORY_APP_MAPS : String - Intent  
CATEGORY_APP_MARKET : String - Intent  
CATEGORY_APP_MESSAGING : String - Intent  
CATEGORY_APP_MUSIC : String - Intent  
CATEGORY_BROWSABLE : String - Intent  
CATEGORY_CAR_DOCK : String - Intent  
CATEGORY_CAR_MODE : String - Intent  
CATEGORY_DEFAULT : String - Intent  
CATEGORY_DESK_DOCK : String - Intent  
CATEGORY DEVELOPMENT_PREFERENCE : String - Intent  
CATEGORY_EMBED : String - Intent  
CATEGORY_FRAMEWORK_INSTRUMENTATION_TEST : String - Intent  
CATEGORY_HE_DESK_DOCK : String - Intent  
CATEGORY_HOME : String - Intent  
CATEGORY_INFO : String - Intent  
CATEGORY_LAUNCHER : String - Intent  
CATEGORY_LE_DESK_DOCK : String - Intent  
CATEGORY_MONKEY : String - Intent  
CATEGORY_OPENABLE : String - Intent  
CATEGORY_PREFERENCE : String - Intent  
CATEGORY_SAMPLE_CODE : String - Intent  
CATEGORY_SELECTED_ALTERNATIVE : String - Intent  
CATEGORY_TAB : String - Intent  
CATEGORY_TEST : String - Intent  
CATEGORY_UNIT_TEST : String - Intent
```

### Component

Explicit name of a component class to use for the intent (eg. "MyMethod2")

# Android Intents

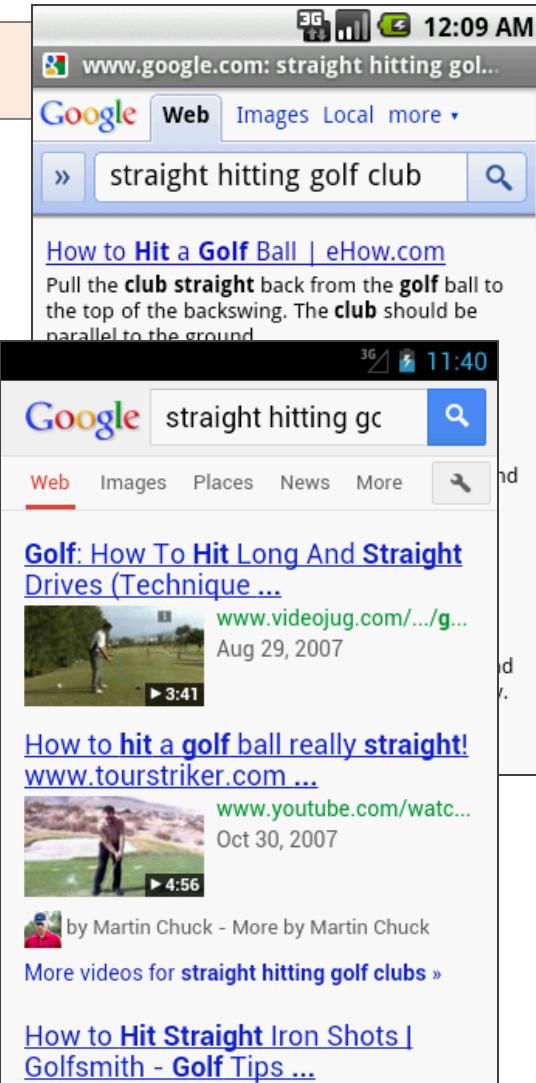
## Example 2: ACTION\_WEB\_SEARCH

### Using Secondary Attributes

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 Google's answer to 'how to hit long and straight' is 'none'

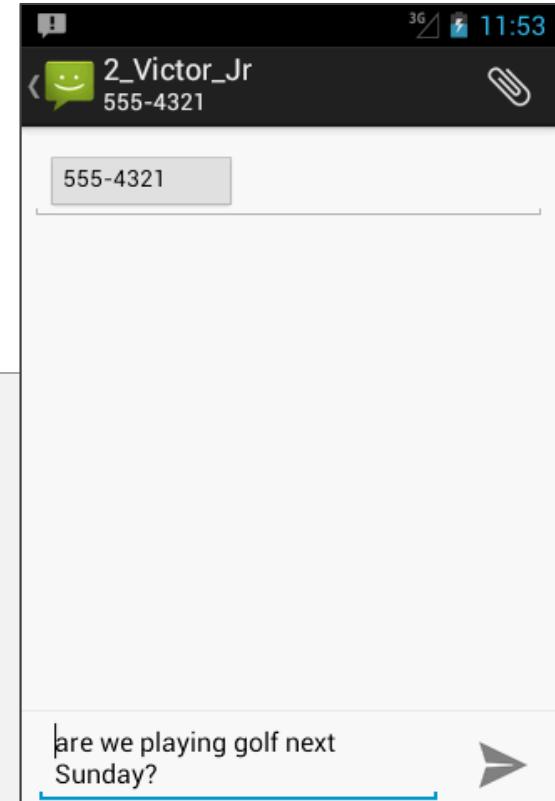
# Android Intents

## Example3: ACTION\_SENDTO

### Using Secondary Attributes

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);
```



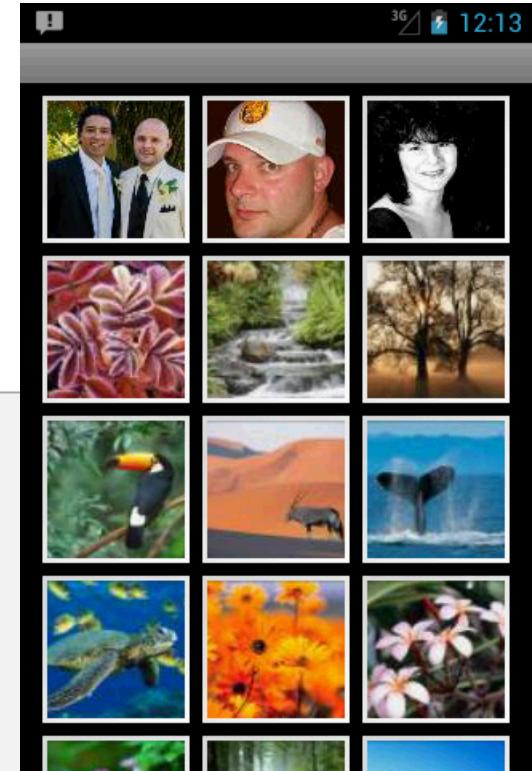
# Android Intents

## Example 4: ACTION\_GET\_CONTENT (Pictures)

### Using Secondary Attributes

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

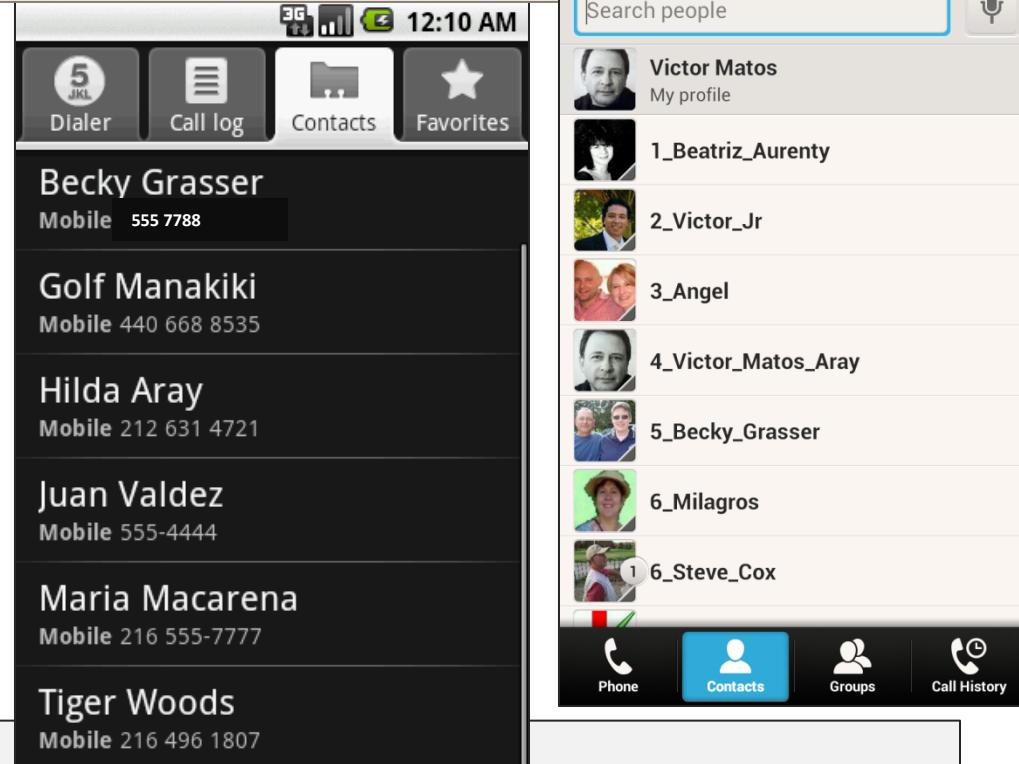
```
Intent intent = new Intent();  
  
intent.setType("image/pictures/*");  
intent.setAction(Intent.ACTION_GET_CONTENT);  
  
startActivity(intent);
```



# Android Intents

## Example 5: ACTION\_VIEW (Contacts)

Showing all Contacts stored in your device



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

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

```
startActivity(myActivity2);
```

# Android Intents

## Example 6: ACTION\_EDIT (Contacts)

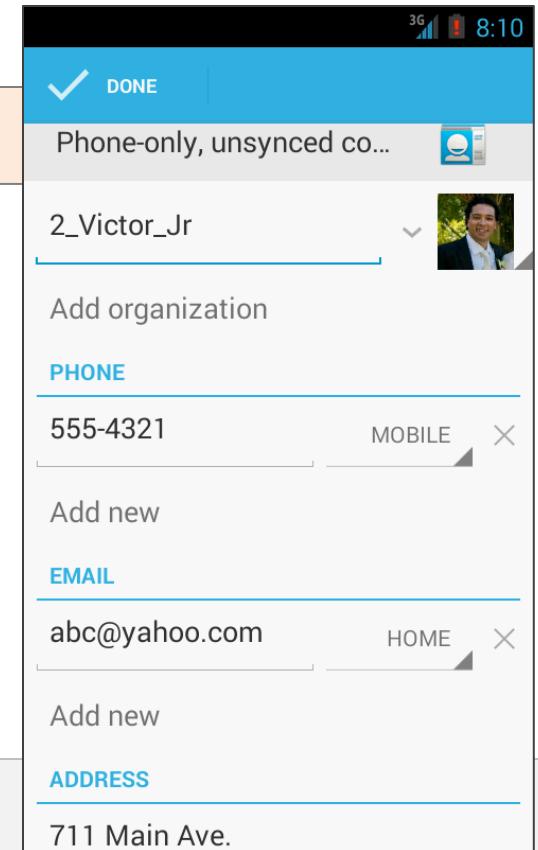
Select a particular person (ID 2) from the contact list for editing purposes.

Later in this lesson we will learn how to obtain the ID of stored contacts (music tracks, pictures, etc).

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

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

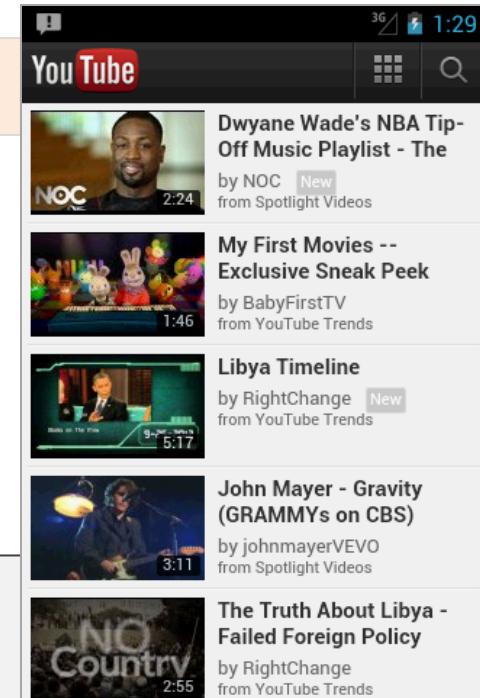
```
startActivity(myActivity2);
```



# Android Intents

## Example 7: ACTION\_VIEW (Web page)

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);
```

Try later with URL:

["http://www.youtube.com/results?search\\_query=ping pong"](http://www.youtube.com/results?search_query=ping+pong)

**Caution.** Must add to the Manifest a request for permission to use the Internet:  
`<uses-permission android:name="android.permission.INTERNET" />`

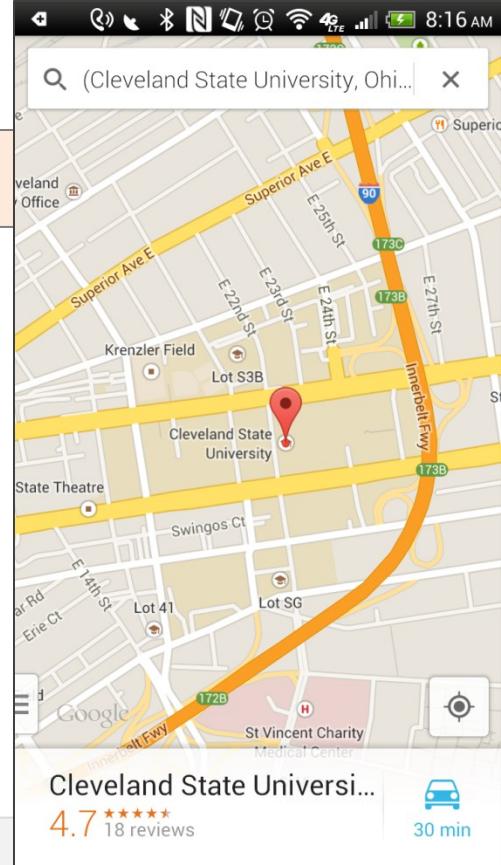
# Android Intents

## Example 8: ACTION\_VIEW (Maps - landmark)

### Geo Mapping an Address / Place

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

```
// (you may get multiple results...)
String thePlace = "Cleveland State University, Ohio";
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:

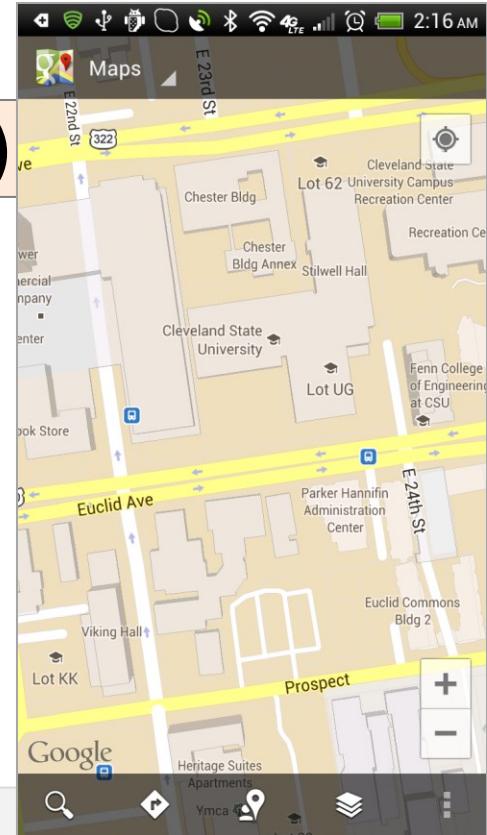
```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
```

# Android Intents

## Example 9: ACTION\_VIEW (Maps - Coordinates)

### Geo Mapping Coordinates (latitude, longitude)

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



```
// map is centered around 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" />
```

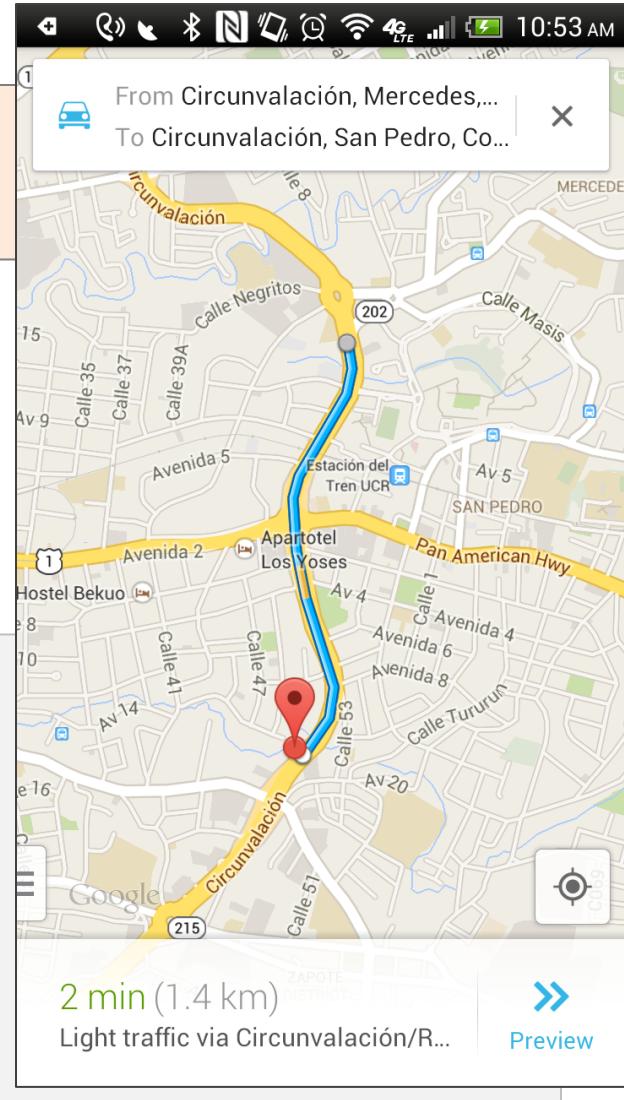
# Android Intents

## Example 10: ACTION\_VIEW (Maps - Directions)

### 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);
```



# Android Intents

## Example 10: ACTION\_VIEW (Maps - StreetView)

GeoCode Uri structure:

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

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

```
String geoCode = "google.streetview:  
    + "cbl=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" />
```

# Android Intents

## Example 12: ACTION\_MUSIC\_PLAYER

### Launching the Music Player

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

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



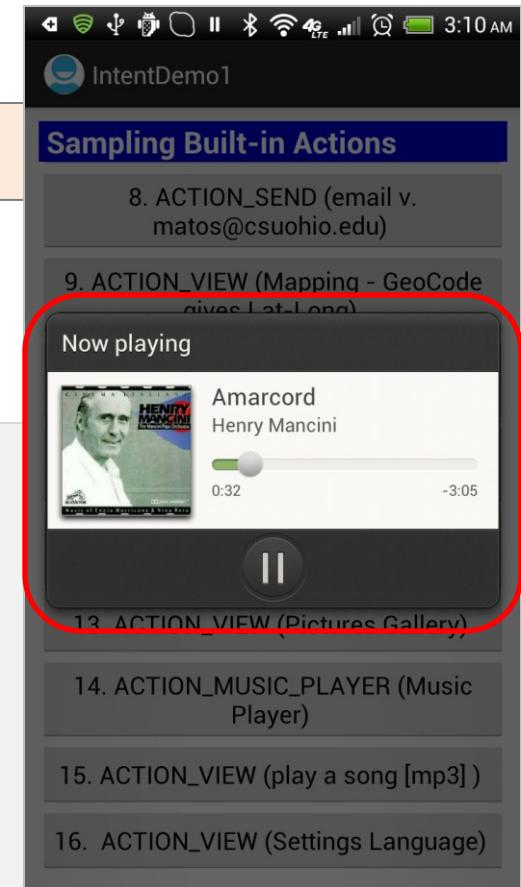
# Android Intents

## Example 13: ACTION\_VIEW (Music)

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"/>
```

# Android Intents

## Example 14: ACTION\_SEND (Email)

### Sending Email

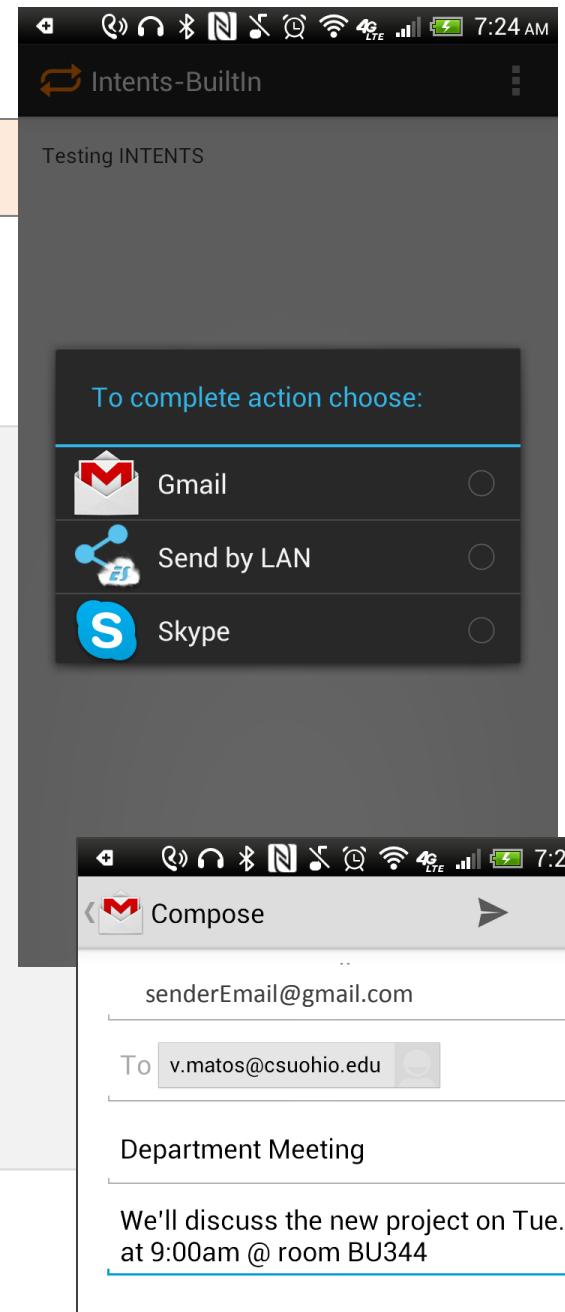
Reference: <http://developer.android.com/guide/appendix/g-app-intents.html>

```
// send email
String emailSubject = "Department Meeting";
String emailText = "We'll discuss the new project "
                  + "on Tue. at 9:00am @ room BU344";
String[] emailReceiverList = {"v.matos@csuohio.edu"};

Intent intent = new Intent(Intent.ACTION_SEND);

intent.setType("vnd.android.cursor.dir/email");
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:"));
```



## Example 15: Device Settings

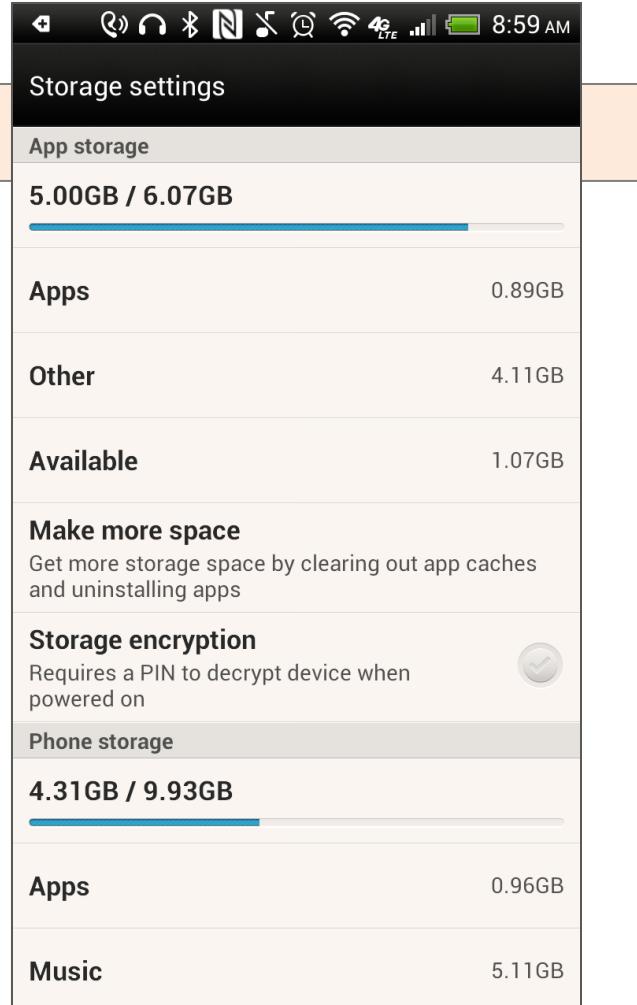
### System Settings

Almost all configurable features of an Android device can be accessed through built-in actions. For example ,an intent using

**android.provider.Settings.XXX**

where **XXX** is as in Appendix A, invokes an app where the corresponding set of parameters defining XXX-settings could be adjusted. For a list of selected built-in actions see Appendix A.

```
startActivity(new Intent(  
        android.provider.Settings.ACTION_INTERNAL_STORAGE_SETTINGS  
));
```



# Android Intents

## Inter-Process Communication.

- A typical Java program runs in a single thread. There, the program calls its methods using a synchronous stop-and-go protocol. Parameters are supplied to a called function, the caller passes control to the sub-routine, and waits for the function to complete. When it finally ends, the caller grabs any returned values , and proceeds with the rest of its work.
- Android apps may include several independent but usually cooperative activities. Each activity works in its own thread with one of them designated as the Main.
- Android uses The **startActivity(Intent)** method to initiate an activity, which will become active and (perhaps) visible; however the caller continues to execute in its own thread.
- The next examples illustrate the basic inter-process communication mechanism used in Android for apps that consists of several collaborative activities. We will see how the calls are made, how input data is supplied to the called activity, and how results are returned to the caller.

# Android Intents

## Starting Activities and Getting Results

- In order for a parent activity to trigger the execution of a child activity, and eventually get results back we use the method

**startActivityForResult ( Intent, requestCodeID )**

Where *requestCodeID* is an arbitrary value you choose to identify the caller (similar to a ‘nickname’ ).

- The results returned by the child-activity (if any) could be asynchronously picked up by a listener method defined in the parent activity

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

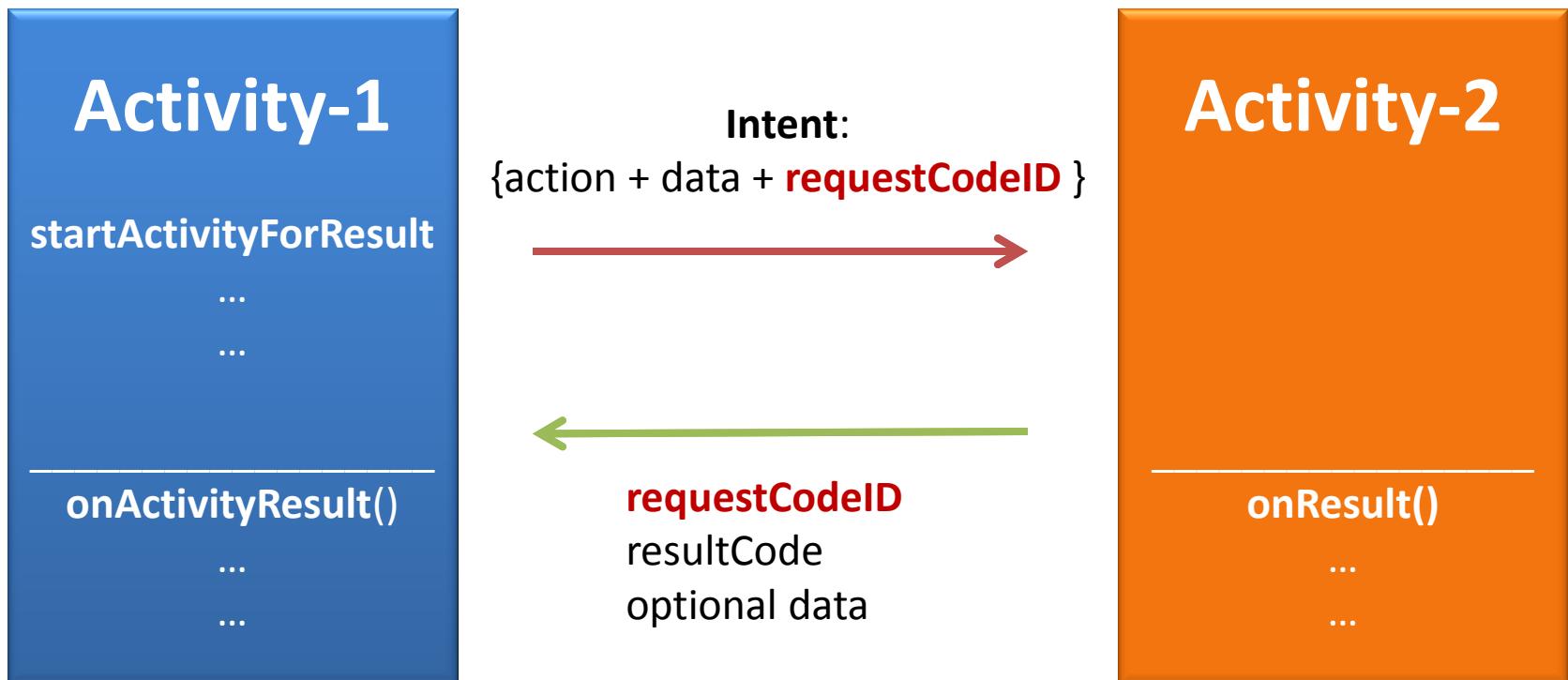
# Android Intents

## Starting Activities and Getting Results

- When the called activity is ready to finish, it could return an optional *resultCode* to the caller to summarize the success of its execution  
`setResult(resultCode)`
- Standard resultCodes include  
`Activity.RESULT_CANCELED` (something bad happened),  
`Activity.RESULT_OK` (a happy ending),  
or any custom values.
- The brief resultCode as well as any additional extra data can be collected back on the parent's using  
`onActivityResult (int requestCodeID,  
                  int resultCode,  
                  Intent data)`

# Android Intents

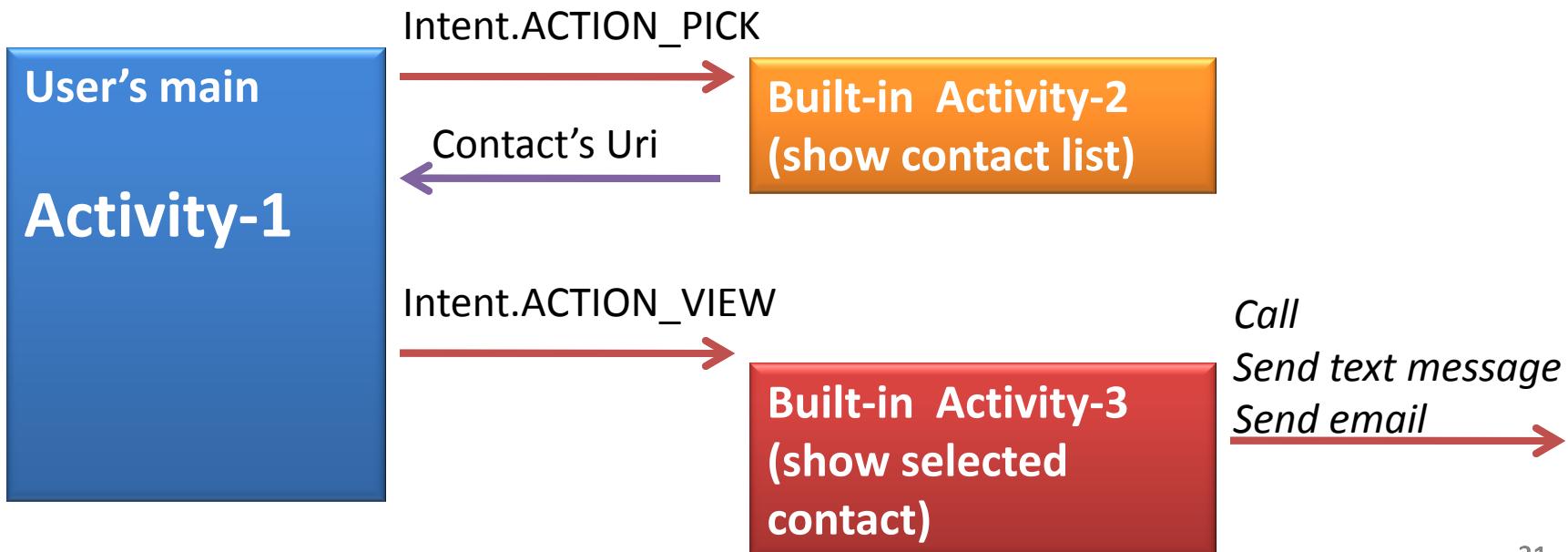
## Starting Activities and Getting Results



# Android Intents

## Example 16. Let's play golf - Call for a tee-time.

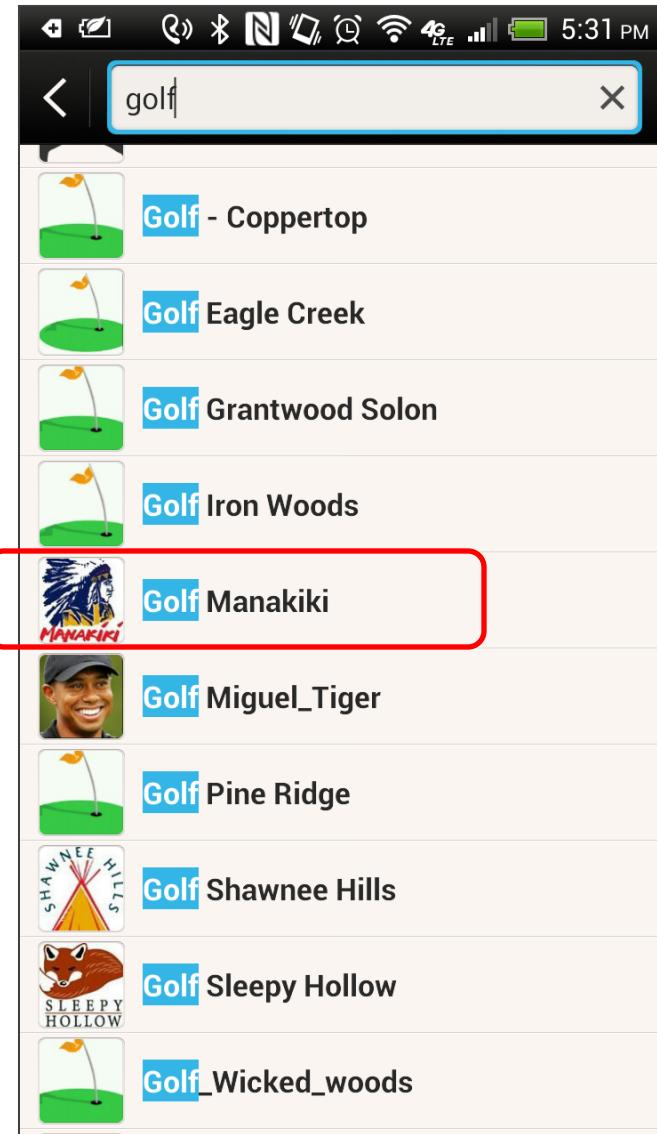
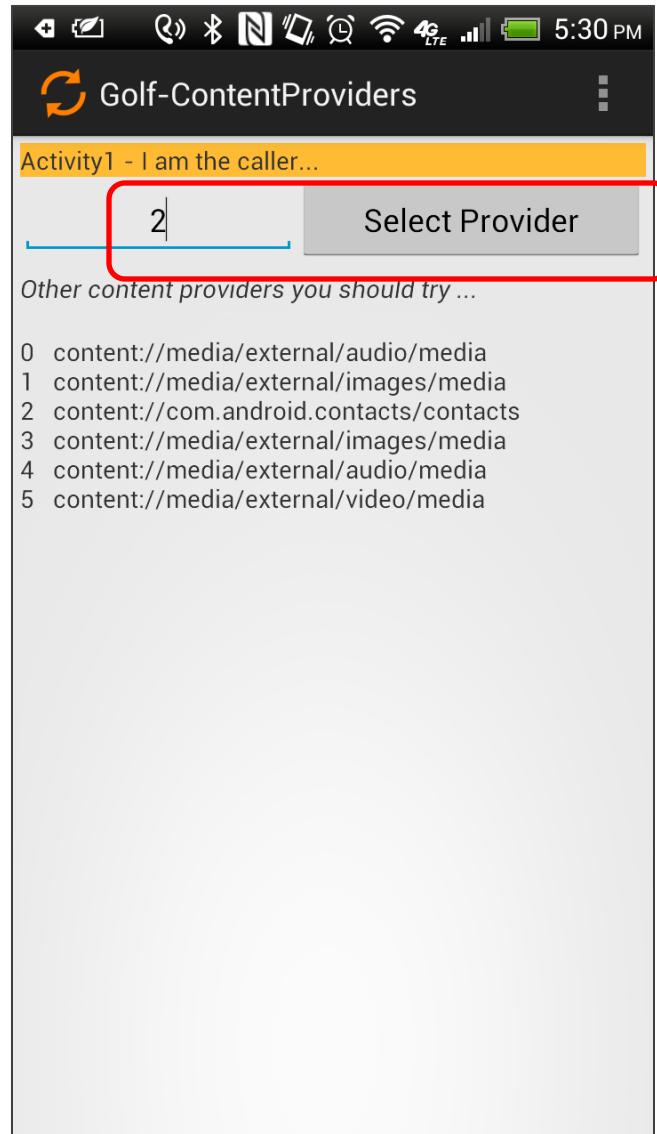
1. Show all our contacts and pick a particular golf course using the ***Intent.ACTION\_PICK*** on the URI :  
`android.provider.ContactsContract.Contacts.CONTENT_URI`
2. Use the returned URI identifying the place we want to call for a tee-time reservation.
3. 'Nicely' show the selected contact's entry allowing calling, texting, emailing actions (use ***Intent.ACTION\_VIEW***).



# Android Intents

## Example 16. Let's play golf - Call for a tee-time.

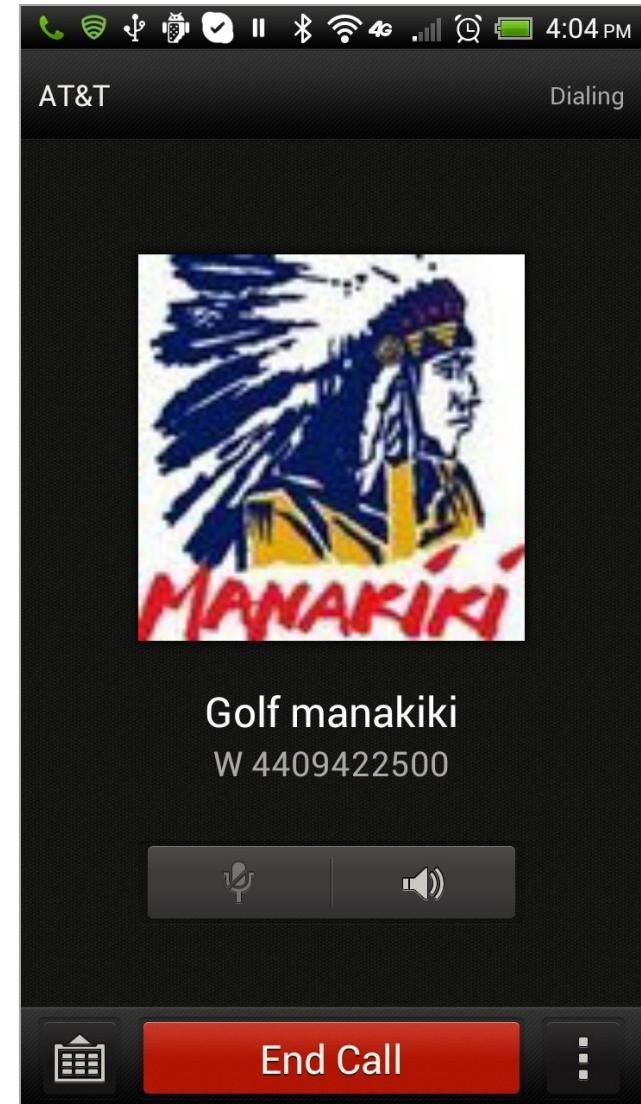
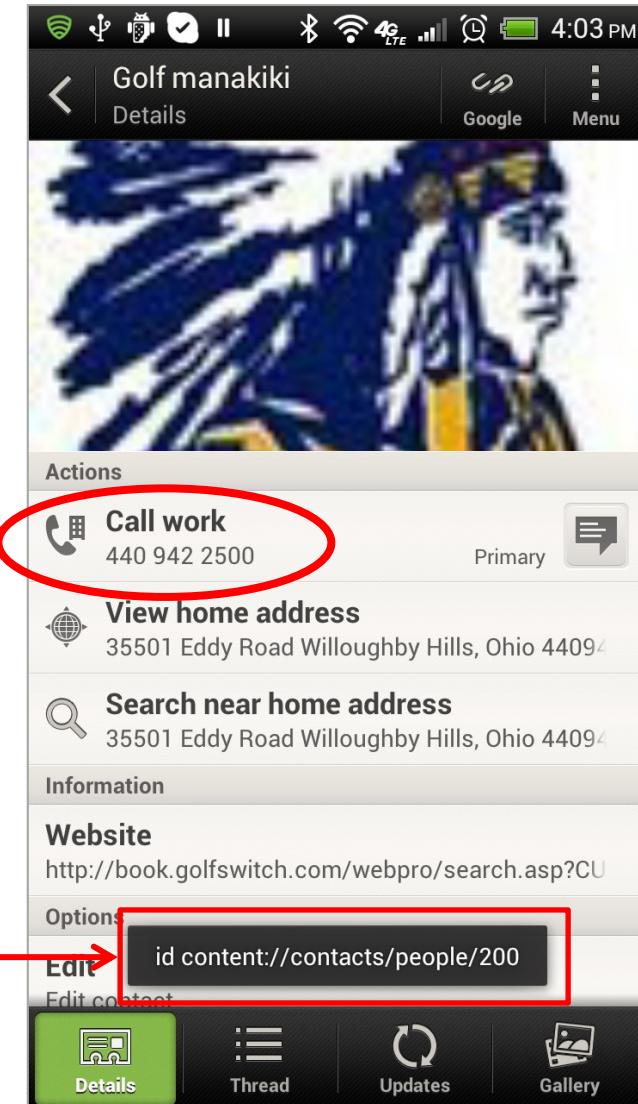
Select Contacts Provider (2)



# Android Intents

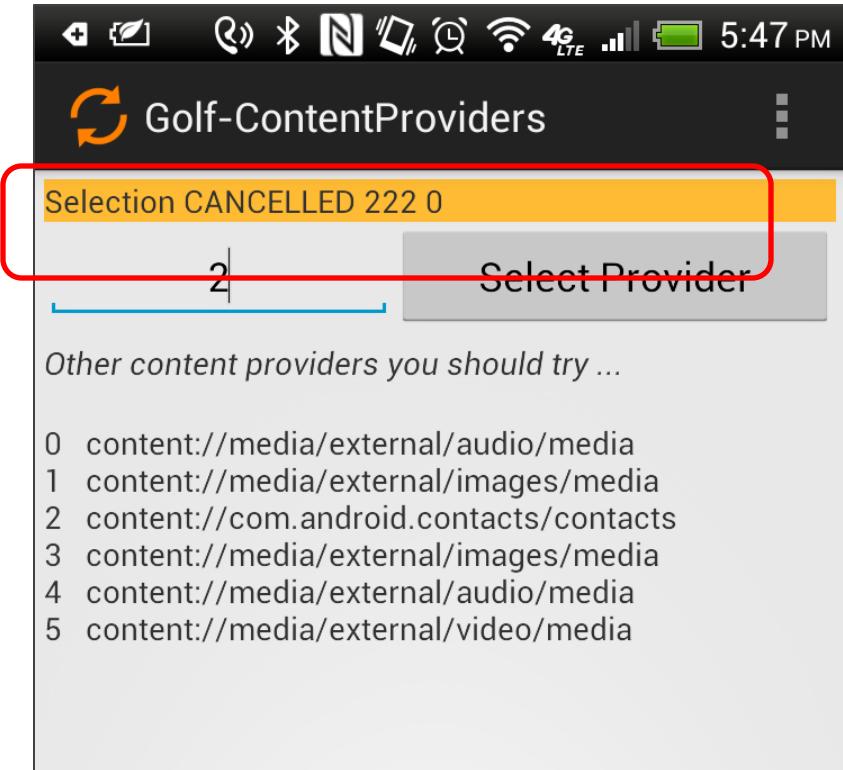
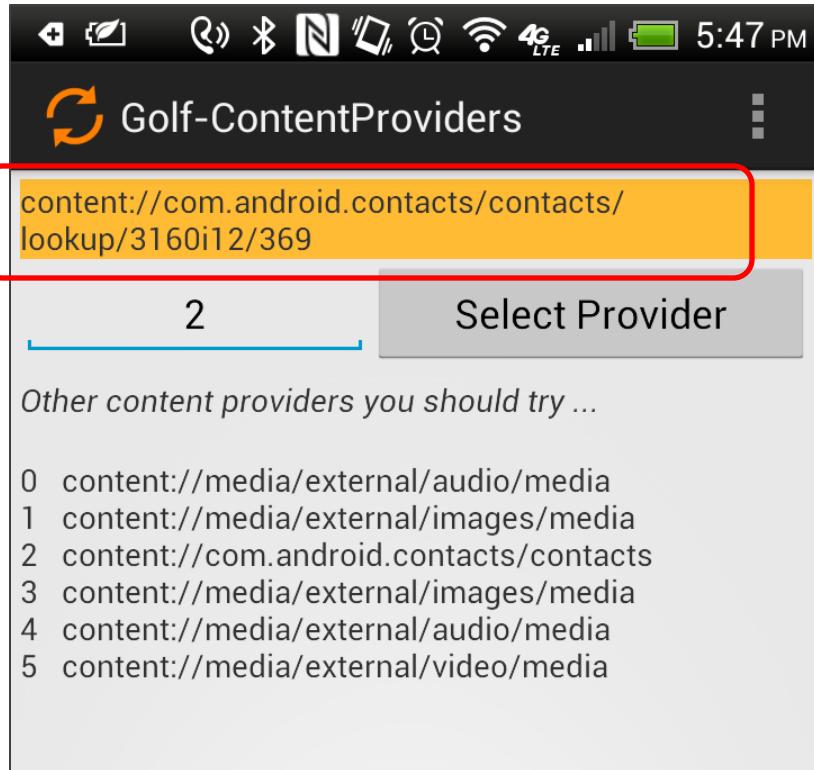
**Example 16.** Let's play golf - Call for a tee-time.

Complete the  
Phone call



# Android Intents

## Example 16. Let's play golf - Call for a tee-time.



The image on the left shows the URI returned from the interaction with the content provider. The path allows a lookup operation targeting the selected golf course. The image on the right shows the result of cancelling the search for data in the contact list.

# Android Intents

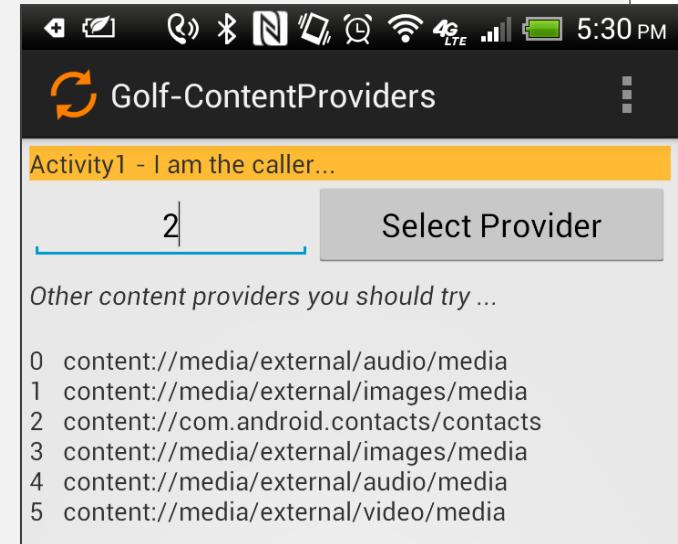
## Example 16. XML Layout – activity\_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="4dp" >

    <TextView
        android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/holo_orange_light"
        android:text="Activity1 - I am the caller..." />

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

        <EditText
            android:id="@+id/txtProviderOption"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:ems="5"
            android:gravity="center_horizontal"
            android:inputType="number"
            android:text="2" />
    
```



# Android Intents

## Example 16. XML Layout – activity\_main.xml

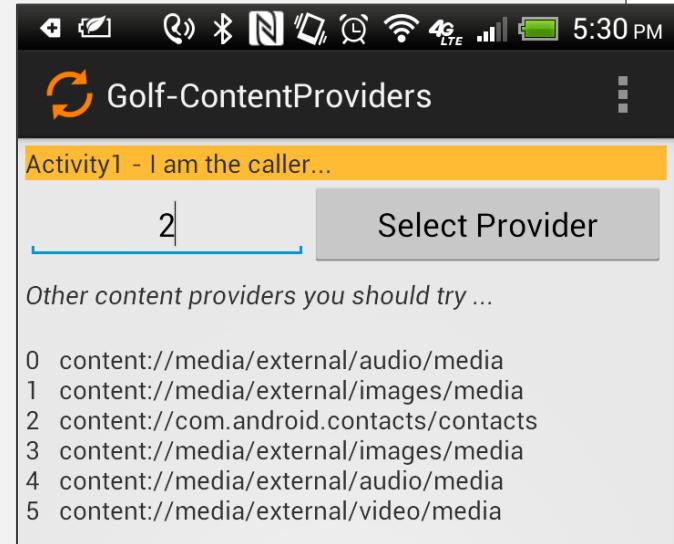
```
<requestFocus />
</EditText>

<Button
    android:id="@+id/btnOption"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="Select Provider" />
</LinearLayout>

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    android:text="Other content providers you should try ..."
    android:textStyle="italic" />

<TextView
    android:id="@+id/txtProviders"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="" />

</LinearLayout>
```



# Android Intents

## Example 16. MainActivity.java

```
public class MainActivity extends Activity {  
    TextView txtMsg;  
    EditText txtProvider;  
    EditText txtExample;  
    Button btnCallActivity2;  
  
    Uri[] uriProvider = {  
        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);  
    }  
}
```

1 →

# Android Intents

## Example 16. MainActivity.java

```
try {

    txtMsg = (TextView) findViewById(R.id.txtMsg);
    txtProvider = (EditText) findViewById(R.id.txtProviderOption);

    // 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(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG)
        .show();
}

}// onCreate
```

# Android Intents

## Example 16. MainActivity.java

```
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(txtProvider.getText().toString());  
  
            ②→ Intent myActivity2 = new Intent( Intent.ACTION_PICK,  
                                              uriProvider[option]);  
  
            ③→ startActivityForResult(myActivity2, 222);  
  
        } catch (Exception e) {  
            Toast.makeText( getBaseContext(), e.getMessage(),  
                           Toast.LENGTH_LONG).show();  
        }  
        // onClick  
    } // ClickHandler
```

# Android Intents

## Example 16. MainActivity.java

```
@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) {  
  
                    ④→ String returnedData = data.getDataString();  
  
                    ⑤→ Toast.makeText(getApplicationContext(), "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();  
                }  
            }  
        }  
    }  
}
```

# Android Intents

## Example 16. MainActivity.java

```
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(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG)
    .show();
}
}// onActivityResult

}// IntentDemo2
```

6 →

# Android Intents

## Example 16. MainActivity.java

### Comments

1. The app offers a list of content providers to choose from (Contacts can be reached from: **android.provider.ContactsContract.Contacts.CONTENT\_URI**)
2. An intent object is assembled combining **ACTION\_PICK** with the chosen URI.
3. The caller app identifies itself with the requestCode **222**, starts the intent and waits for ACTION\_PICK to send results back to it.
4. The app's listener **onActivityResult** verifies that a resultCode was sent back to itself (222). Then it checks that ACTION\_PICK satisfactorily completed its work and returned an **Activity.RESULT\_OK** completion code.
5. The URI string coming back from ACTION\_PICK can be obtained from the returned intent with **data.getDataString()**.
6. The previous result URI that can be passed to other actions, such as ACTION\_VIEW. This will complete the user's request.

# Android Intents

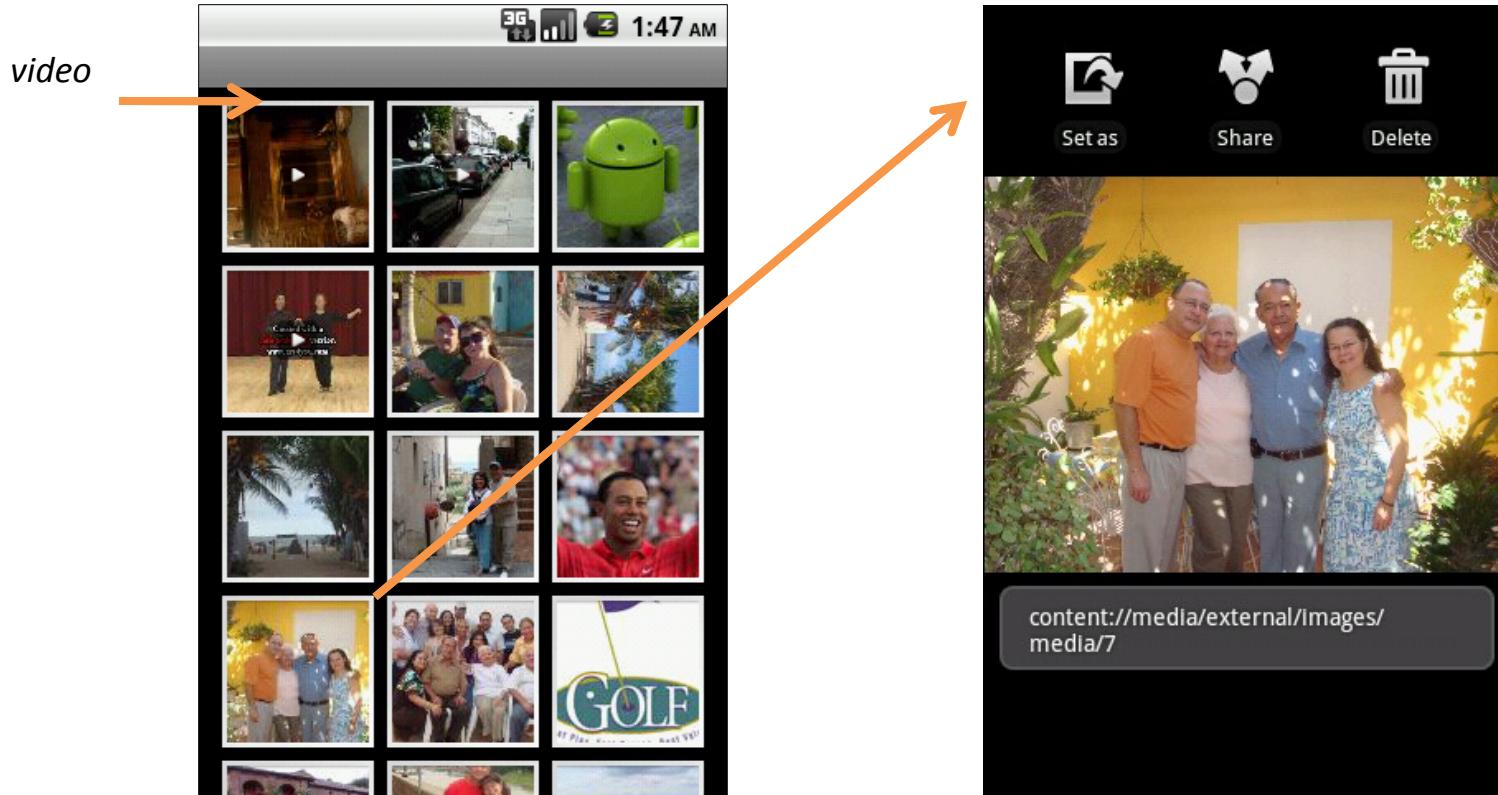
## Example 16. Screenshots

Showing Pictures and Video - For this example we selected:

`ACTION_PICK & content://media/external/images/media`

followed by

`ACTION_VIEW & content://media/external/images/media/7`



# Android Intents

## Using BUNDLES to Pass Data

- A **Bundle** is an Android data-exchange mechanism used for efficient interprocess communications on either in-process or cross-process calls.
- A **Bundle** is conceptually similar to a Java HashMap. It associates a string key to a parcelable (exchangeable) data element. Data could be either primitive data types or object-references. Bundles are functionally equivalent to a 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");
```

# Android Intents

## Android Intents and Bundles - Calling a Receiver

A single Bundle could contain an unlimited number of <key,value> items. They offer an elegant solution to Android IPC exchanges; observe it is sufficient to attach a single extra bundle to an intent for two interacting activities to move any amount of data.

### Activity1: Sender

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

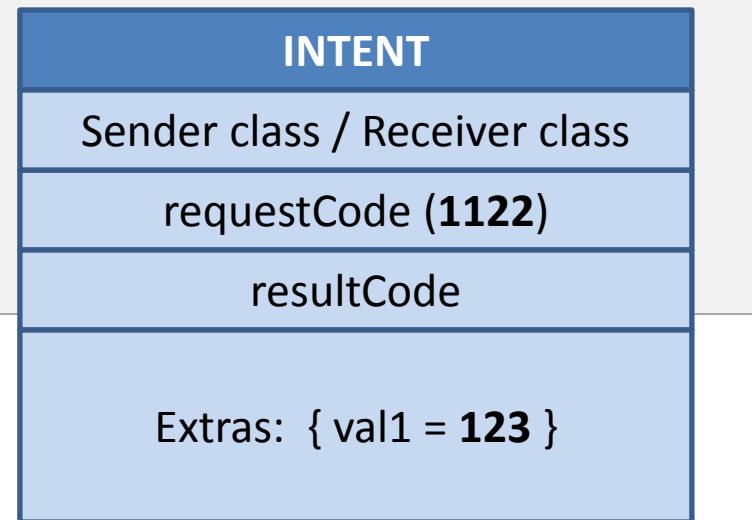
Bundle myBundle1 = new Bundle();

myBundle1.putInt ("val1", 123);

myIntentA1A2.putExtras(myBundle1);

startActivityForResult(myIntentA1A2, 1122);
```

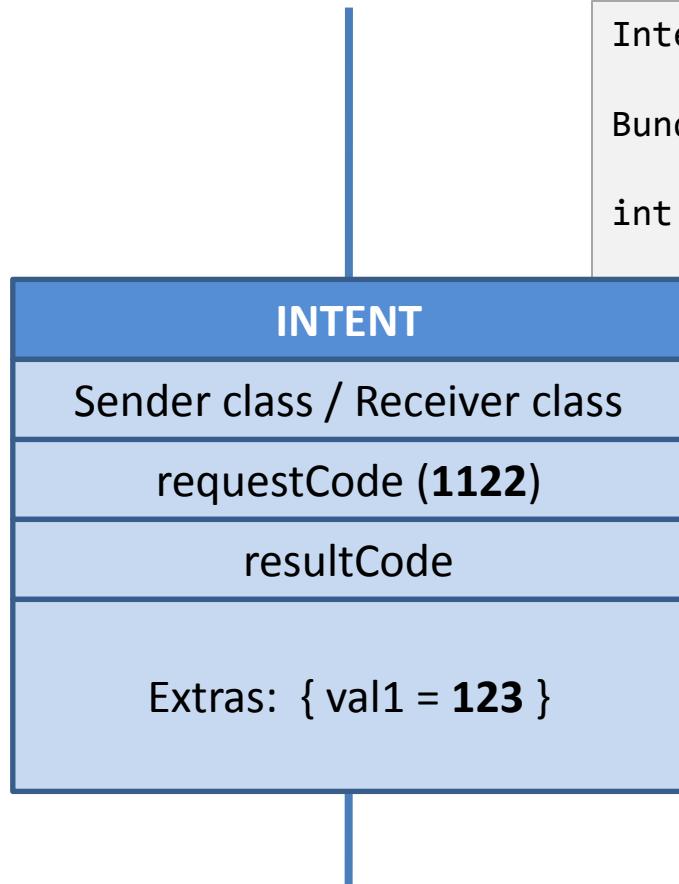
### Activity2: Receiver



# Android Intents

## Android Intents and Bundles - Receiver is awoken

### Activity1: Sender



### Activity2: Receiver

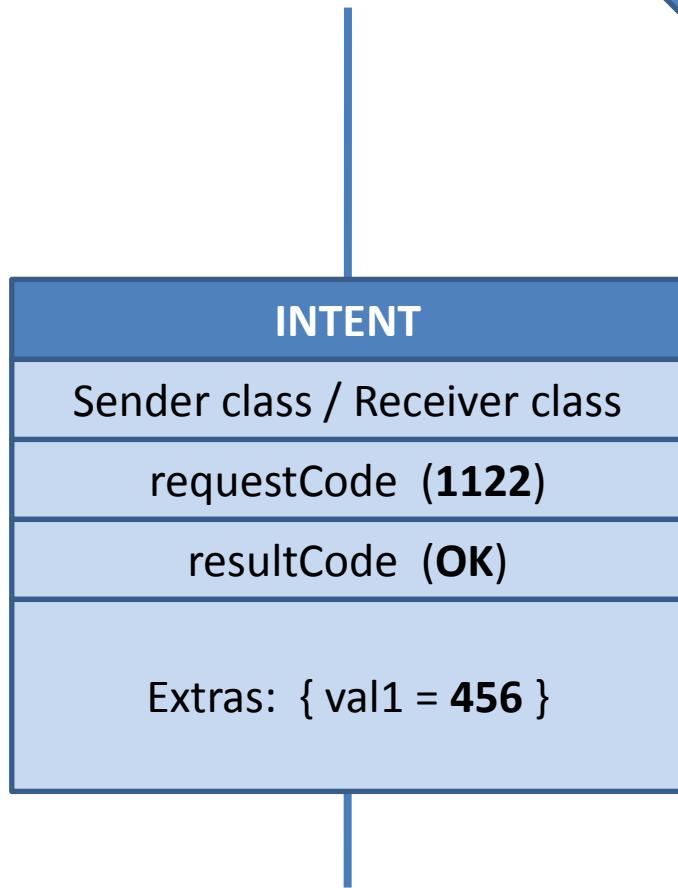
```
Intent myCallerIntent = getIntent();  
  
Bundle myBundle = myCallerIntent.getExtras();  
  
int val1 = myBundle.getInt("val1");
```



# Android Intents

## Android Intents and Bundles - Receiver Returns Results

### Activity1: Sender



### Activity2: Receiver

```
myBundle.putString("val1", 456 );  
myCallerIntent.putExtras(myBundle);  
  
setResult(Activity.RESULT_OK,  
myCallerIntent);
```



# Android Intents

## Common Bundle Methods



### **.clear()**

Removes all elements from the mapping of this Bundle.

### **.clone()**

Clones the current Bundle.

### **.containsKey(String key)**

Returns true if the given key is contained in the mapping of this Bundle.

### **.putIntArray(String key, int[] value)**

### **.getIntArray(String key)**

Inserts/replaces /retrieves an int array value into the mapping of this Bundle

### **.putString(String key, String value)**

### **.getString(String key)**

Inserts /replaces/retrieves a String value into the mapping of this Bundle

### **.putStringArray(String key, String[] value)**

### **.getStringArray(String key)**

Inserts /replaces/retrieves a String array value into the mapping of this Bundle

### **.putStringArrayList(String key, ArrayList<String> value)**

Inserts/replaces an ArrayList value into the mapping of this Bundle.

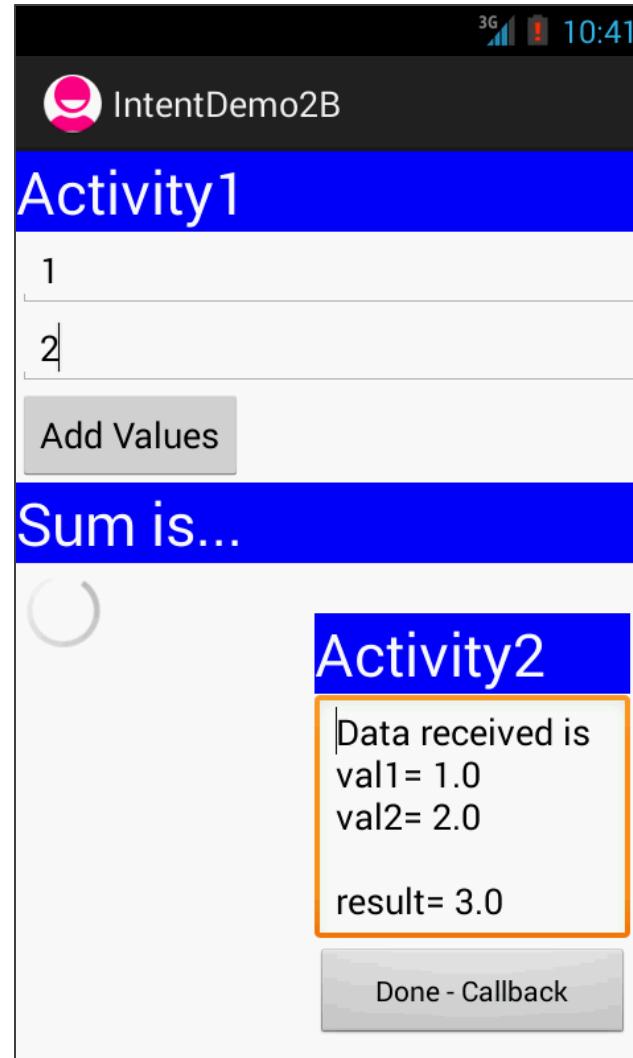
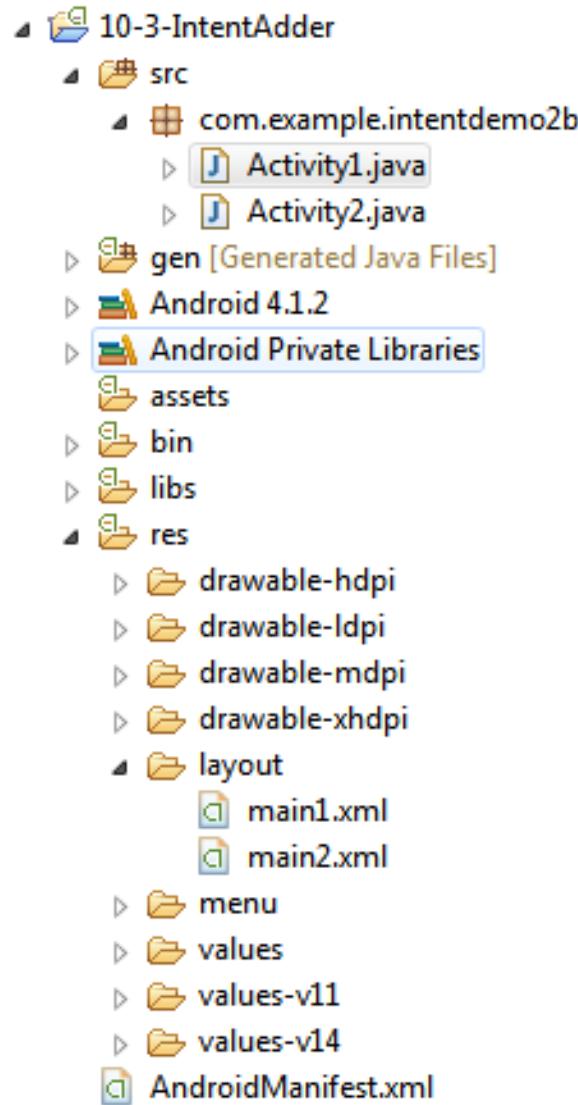
### **.remove(String key)**

Removes any entry with the given key from the mapping of this Bundle.

### **.size()**

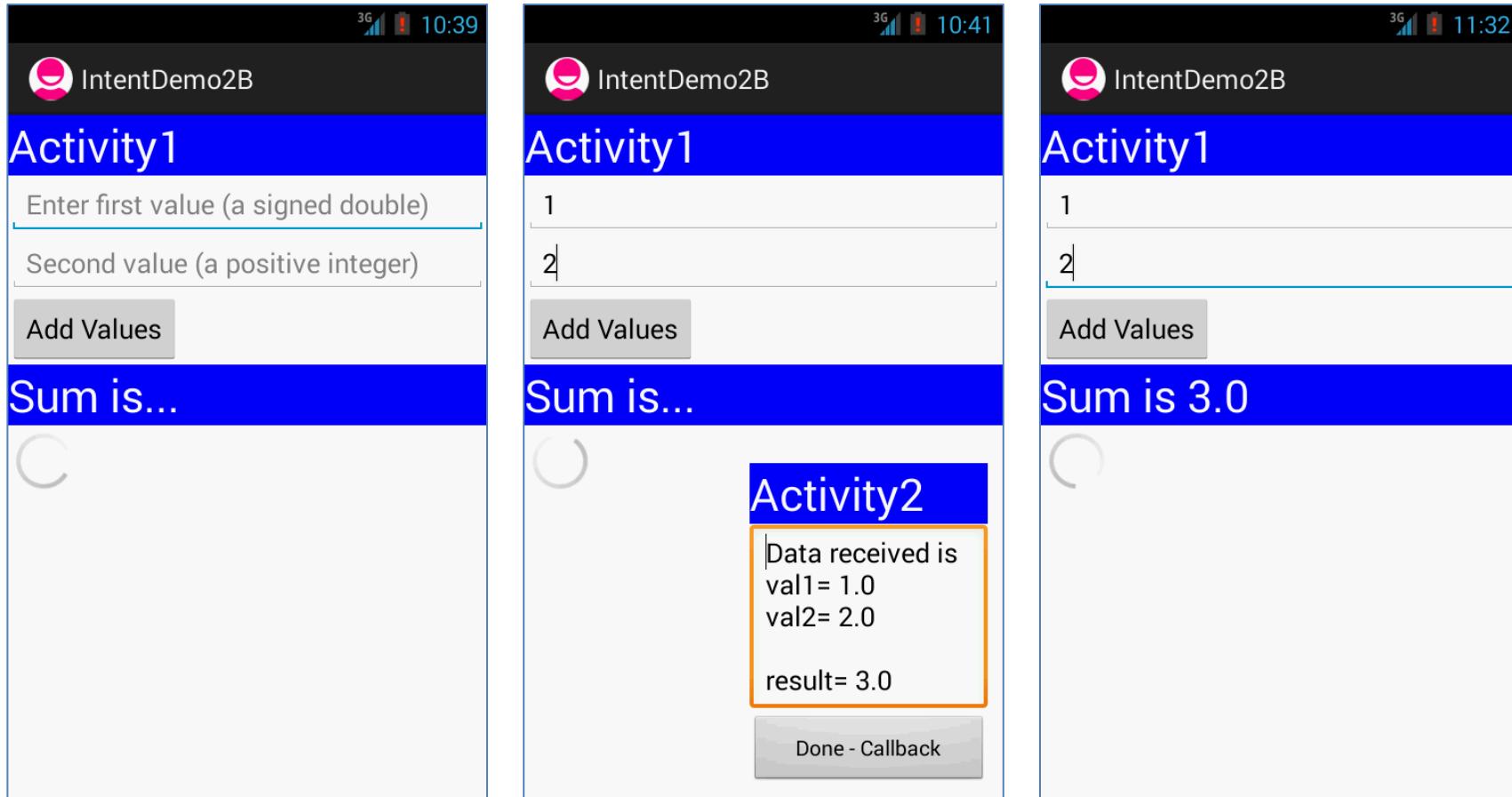
Returns the number of mappings contained in this Bundle.

## Example 17. XML Layout – activity\_main.xml



In this example **Activity1** passes two numbers to **Activity2** which will add them up and return the result. All data is passed back and forth in a Bundle.

## Example 17. Screenshots



Data values collected by Activity1 are placed into a bundle and sent to Activity2. Observe that progress bar in Activity1 remains active however this process has no focus (its button and textboxes do not respond) Activity2 is visible and has focus.

Activity1 receives the result of the operation carried out by Activity2.

# Android Intents

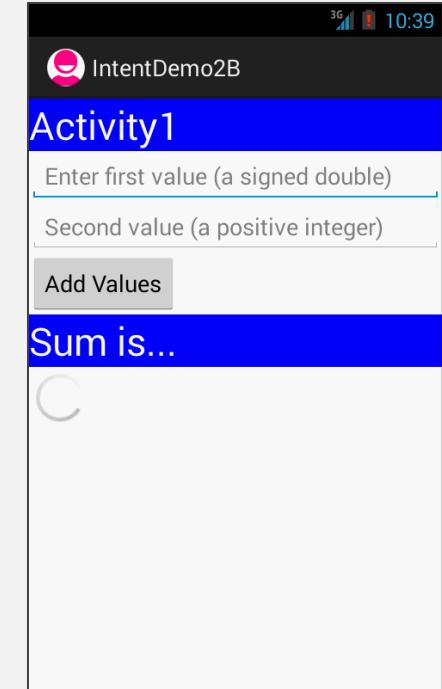
## Example 17. XML Layout – main1.xml (Activity1)

```
<?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="#ff0000ff"
        android:text="Activity1"
        android:textColor="#ffffffff"
        android:textSize="30sp" />

    <EditText
        android:id="@+id/EditText01"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter first value (a signed double)"
        android:inputType="numberDecimal|numberSigned/number" />

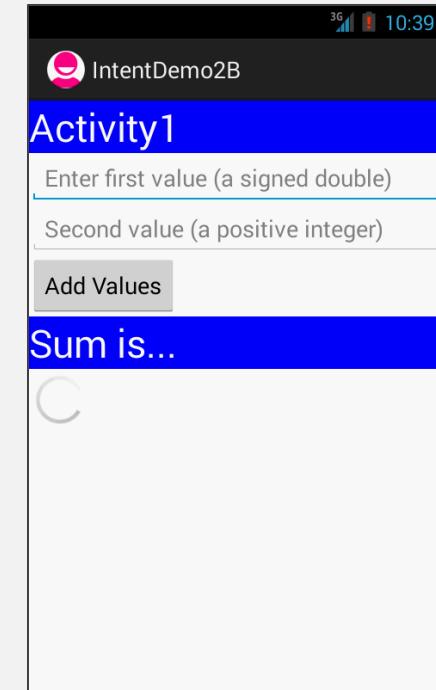
    <EditText
        android:id="@+id/EditText02"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```



# Android Intents

## Example 17. XML Layout – main1.xml (Activity1)

```
        android:hint="Second value (a positive integer)"  
        android:inputType="number" />  
  
<Button  
        android:id="@+id/btnAdd"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Add Values" />  
  
<TextView  
        android:id="@+id/txtResult"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:background="#ff0000ff"  
        android:text="Sum is..."  
        android:textColor="#ffffffff"  
        android:textSize="30sp" />  
  
① → <ProgressBar  
        android:id="@+id/progressBar1"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content" />  
  
</LinearLayout>
```



# Android Intents

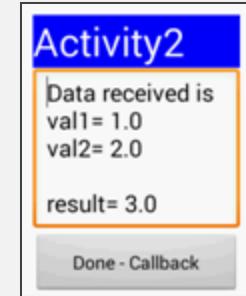
## Example 17. XML Layout – main2.xml (Activity2)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="right|bottom"
    android:layout_margin="10dp"
    android:background="@android:color/transparent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff0000ff"
        android:text="Activity2"
        android:textColor="#ffffffff"
        android:textSize="30sp" />

    <EditText
        android:id="@+id/etDataReceived"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Data received..." />

    <Button
        android:id="@+id/btnDone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Done - Callback" />
</LinearLayout>
```



# Android Intents

## Example 17. Activity1.java

```
package com.example.intentdemo2b;
// Multi-Activity Application
// Activity1: collects two data items from the user's UI, places
// them into a Bundle, and calls Activity2
// Activity2: accepts two data items, adds them, returns result

public class Activity1 extends Activity {
    EditText txtValue1;
    EditText txtValue2;
    TextView txtResult;
    Button btnAdd;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main1);
        txtValue1 = (EditText) findViewById(R.id.EditText01);
        txtValue2 = (EditText) findViewById(R.id.EditText02);
        txtResult = (TextView) findViewById(R.id.txtResult);

        btnAdd = (Button) findViewById(R.id.btnAdd);
        btnAdd.setOnClickListener(new OnClickListener() {
```

# Android Intents

## Example 17. Activity1.java

cont. 1

```
@Override  
public void onClick(View v) {  
    // get values from the UI  
    Double v1 = Double.parseDouble(txtValue1.getText().toString());  
    Double v2 = Double.parseDouble(txtValue2.getText().toString());  
  
    // create intent to call Activity2  
    Intent myIntentA1A2 = new Intent (Activity1.this,  
                                    Activity2.class);  
    // create a Bundle (MAP) container to ship data  
    Bundle myDataBundle = new Bundle();  
  
    // add <key,value> data items to the container  
    myDataBundle.putDouble("val1", v1);  
    myDataBundle.putDouble("val2", v2);  
  
    // attach the container to the intent  
    myIntentA1A2.putExtras(myDataBundle);  
  
    // call Activity2, tell your local listener to wait a  
    // response sent to a listener known as 101  
    startActivityForResult(myIntentA1A2, 101);  
  
}  
});  
}//onCreate
```

③ →

④ →

# Android Intents

## Example 17. Activity1.java

cont. 2

```
// local listener receives callbacks from other activities
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    super.onActivityResult(requestCode, resultCode, data);

    try  {
        if ((requestCode == 101 ) && (resultCode == Activity.RESULT_OK)){
            Bundle myResultBundle = data.getExtras();
            Double myResult = myResultBundle.getDouble("vresult");
            txtResult.setText("Sum is " + myResult);
        }
    }
    catch (Exception e) {
        txtResult.setText("Problems - " + requestCode + " " + resultCode);
    }
}//onActivityResult

}//Activity1
```

5 →

# Android Intents

## Example 17. Activity2.java

```
public class Activity2 extends Activity implements OnClickListener{  
    EditText dataReceived;  
    Button btnDone;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main2);  
        dataReceived = (EditText) findViewById(R.id.etDataReceived);  
        btnDone = (Button) findViewById(R.id.btnDone);  
        btnDone.setOnClickListener(this);  
  
        // pick call made to Activity2 via Intent  
        Intent myLocalIntent = getIntent();  
  
        // look into the bundle sent to Activity2 for data items  
        Bundle myBundle = myLocalIntent.getExtras();  
        Double v1 = myBundle.getDouble("val1");  
        Double v2 = myBundle.getDouble("val2");  
  
        // operate on the input data  
        Double vResult = v1 + v2;
```

6 →

# Android Intents

## Example 17. Activity2.java

cont 1.

```
// for illustration purposes. show data received & result  
dataReceived.setText("Data received is \n"  
    + "val1= " + v1 + "\nval2= " + v2  
    + "\n\nresult= " + vResult);  
  
// add to the bundle the computed result  
myBundle.putDouble("vresult", vResult);  
  
// attach updated bumble to invoking intent  
myLocalIntent.putExtras(myBundle);  
  
// return sending an OK signal to calling activity  
setResult(Activity.RESULT_OK, myLocalIntent);  
  
// experiment: remove comment  
// finish();  
  
}//onCreate  
  
@Override  
public void onClick(View v) {  
    // close current screen - terminate Activity2  
    finish();  
}  
}
```

7 →

8 →

# Android Intents

## Example 17. Manifest

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.intentdemo2b"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="18" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".Activity1"
            android:label="@string/title_activity_intent_demo2_b" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:name=".Activity2"
            android:theme="@android:style/Theme.Translucent.NoTitleBar" >
        </activity>

    </application>

</manifest>
```



9 →

# Android Intents

## Example 17.

### Comments

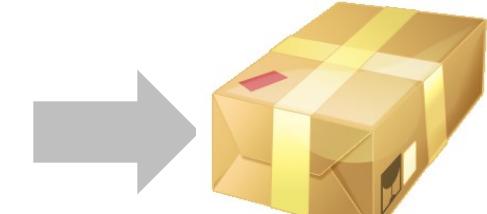
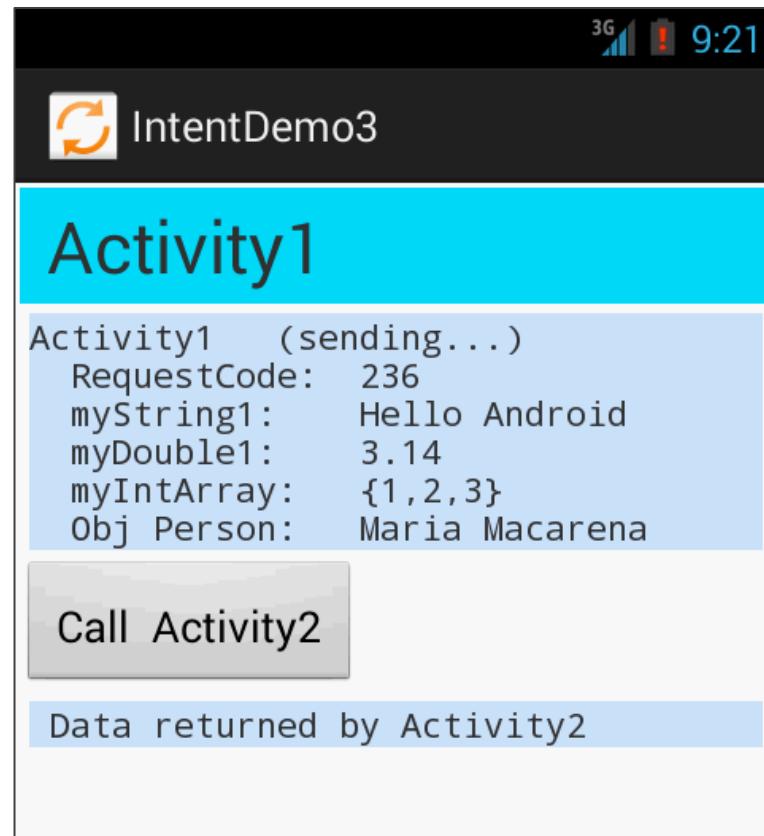
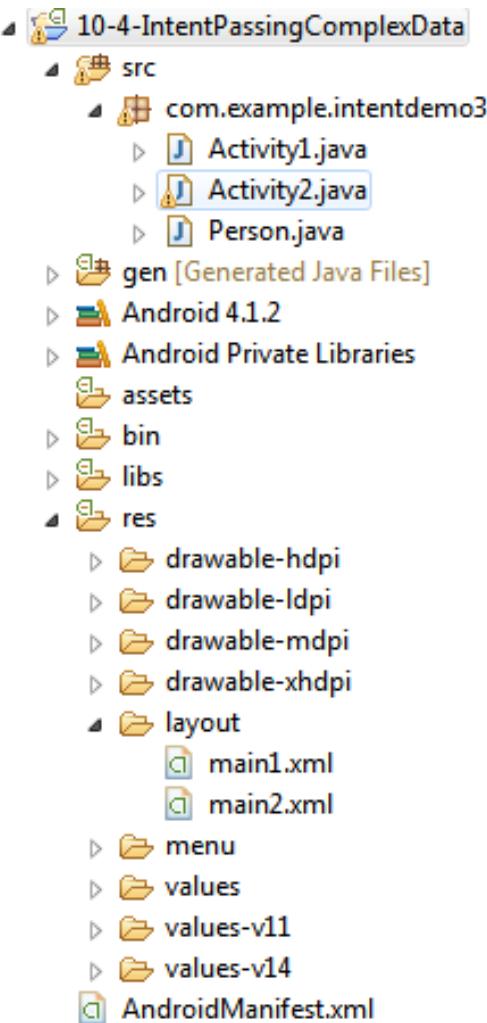
1. The continuous rotation of this circular progress bar will visibly indicate the working state of Activity1.
2. Activity2 has a small layout and a transparent background. When displayed it will be partially super-imposed on top of Activity1's screen.
3. Activity1 prepares an Intent to invoke Activity2. The statement  
`myIntentA1A2.putExtra(myDataBundle)` attaches the bundle to the intent.
4. `startActivityForResult(...)` passes the intent and its id 101 to Activity2.
5. The listener in Activity1 waits for the result wrapped into an extra bundle. When it arrives it is extracted and displayed to the user.
6. Activity2 issues `.getIntent()` to grab the incoming intent and the extra bundle it carries. The two numeric variables are combined to produce a result (`vResult`).
7. Activity2 issues `.putDouble(...)` to store the result into the outgoing bundle.
8. Activity2 releases the outgoing bundle together with a `RESULT_OK` flag.
9. The manifest defines both activities and applies a `translucent, NoTitleBar` theme to Activity2.

# Android Intents

## Example 18. Exchanging data between two activities in the same app.

In this example the caller **Activity1** sends a variety of arguments to **Activity2** including simple types, arrays, and serialized objects. Activity2 applies changes on the input data and returns new values.

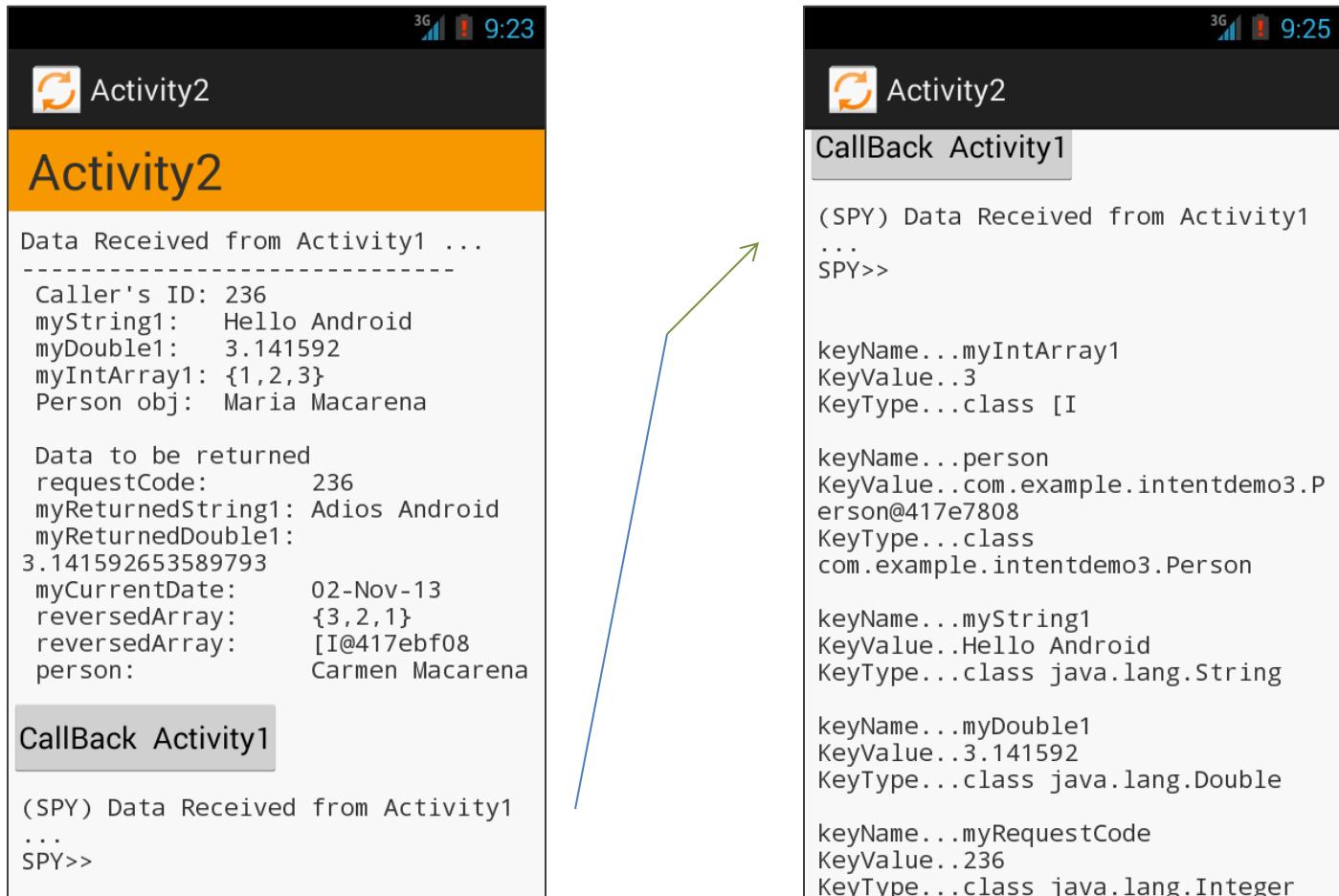
*Both activities are part of the same app.*



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

# Android Intents

## Example 18. Exchanging data between two activities in the same app.



Echo data received

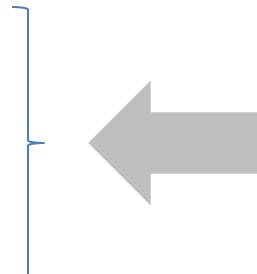
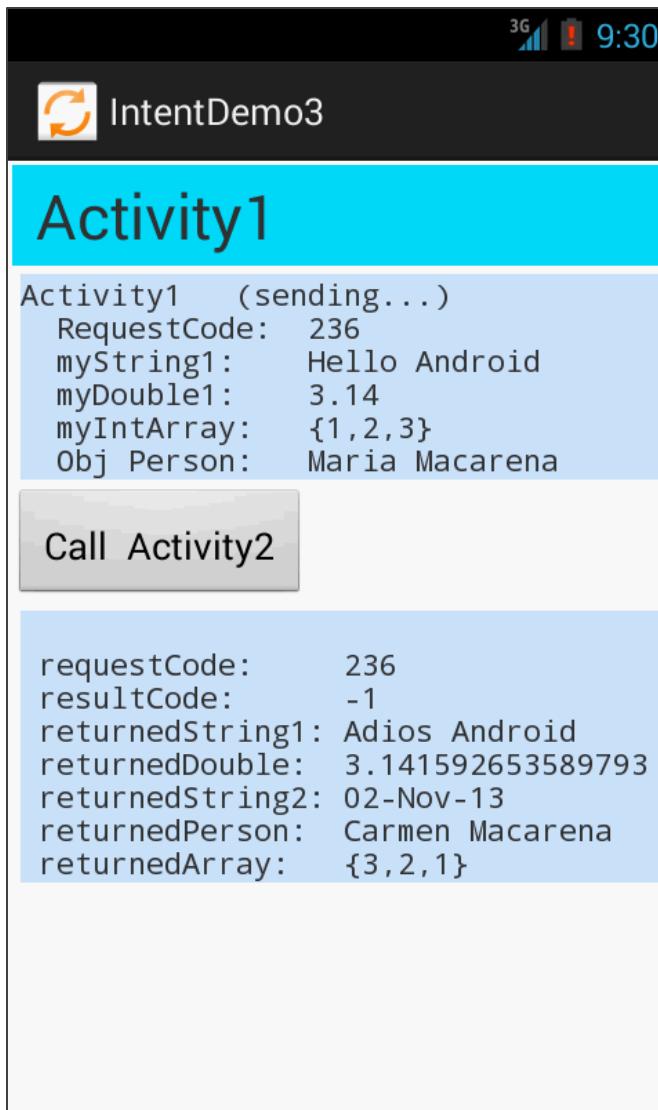
Do local operations (reverse array, change Person's name, return PI and current date)

Explore Bundle to obtain:

keyName, keyValue, and keyType  
of each arriving item

# Android Intents

## Example 18. Exchanging data between two activities in the same app.



Returned values  
sent back from  
**Activity2**

# Android Intents

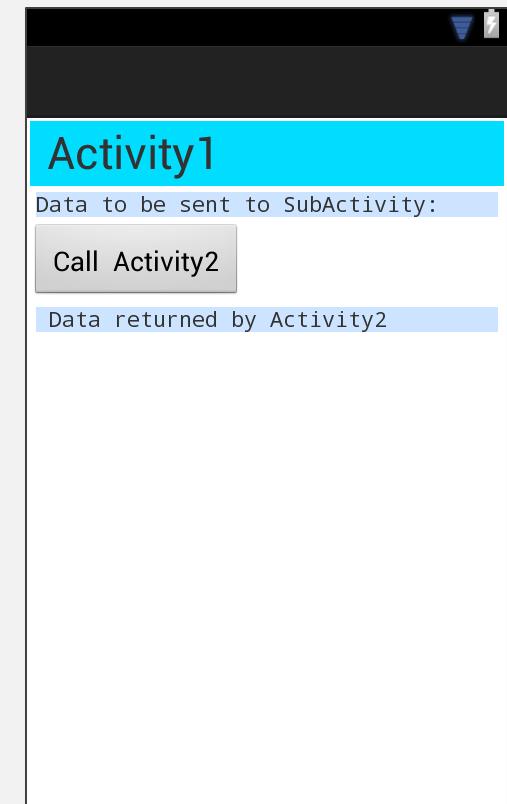
## Example 18. XML Layout – main1.xml (Activity1)

```
<?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"
    android:layout_margin="2dp" >

    <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="@color/holo_blue_bright"
            android:padding="4sp"
            android:text=" Activity1 "
            android:textSize="30sp" />

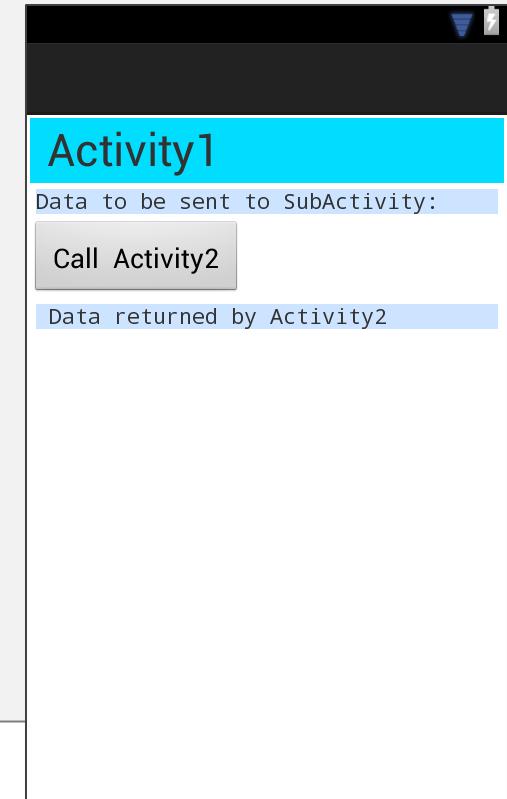
        <TextView
            android:id="@+id/txtTop"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="4dip"
            android:background="#330077ff"
            android:text="Data to be sent to SubActivity:"
            android:typeface="monospace" />
    
```



# Android Intents

## Example 18. XML Layout – main1.xml (Activity1)

```
<Button  
    android:id="@+id	btnCallActivity2"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:background="@android:drawable/btn_default"  
    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="#330077ff"  
    android:text=" Data returned by Activity2"  
    android:typeface="monospace" />  
/</LinearLayout>  
  
</ScrollView>
```



# Android Intents

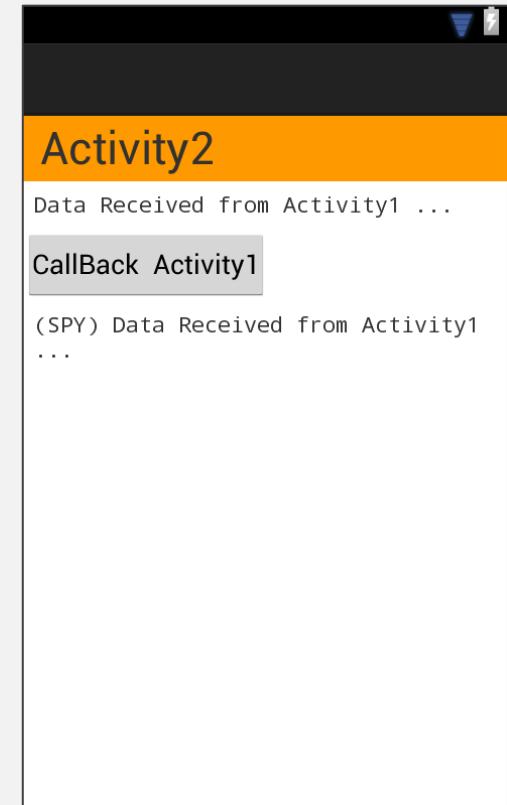
## Example 18. XML Layout – main2.xml (Activity2)

```
<?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:background="#ffffffff"
        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" />

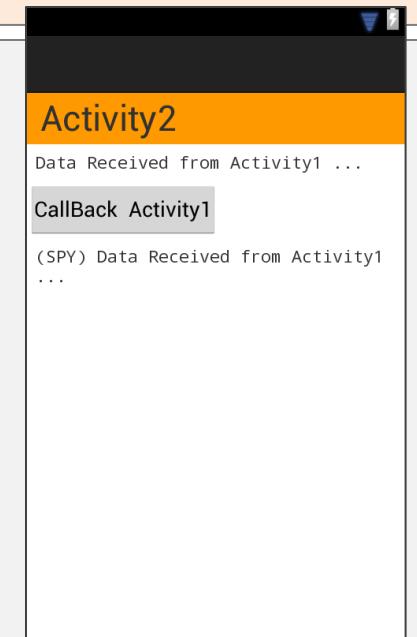
        <TextView
            android:id="@+id/txtIncomingData"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="7dip"
            android:text="Data Received from Activity1 ... "
            android:typeface="monospace" />
 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



# Android Intents

## Example 18. XML Layout – main2.xml (Activity2)

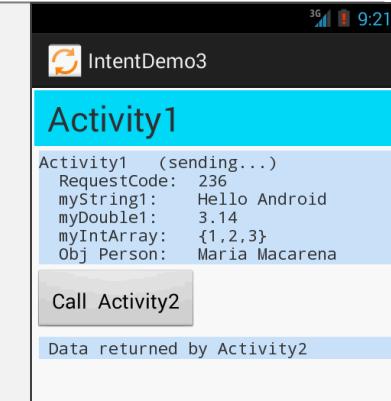
```
<Button  
    android:id="@+id	btnCallBackActivity1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:padding="6sp"  
    android:text="CallBack Activity1" />  
  
<TextView  
    android:id="@+id/spyBox"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_margin="7dip"  
  
    android:text="(SPY) Data Received from Activity1 ..."  
    android:typeface="monospace" />  
/<LinearLayout>  
  
</ScrollView>
```



# Android Intents

## Example 18. Activity1.java (This is the Caller)

```
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(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG).show();  
        }  
    } // onCreate
```



# Android Intents

## Example 18. Activity1.java cont 1.

```
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);

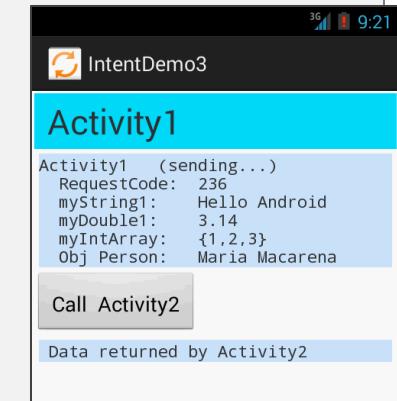
            // prepare a Bundle and add the data pieces to be sent
            Bundle myData = new Bundle();
            myData.putInt("myrequestCode", IPC_ID);
            myData.putString("myString1", "Hello Android");
            myData.putDouble("myDouble1", 3.141592);
            int [] myLittleArray = { 1, 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(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG).show();
        }
    }
} // onClick
} // Clicker1
```



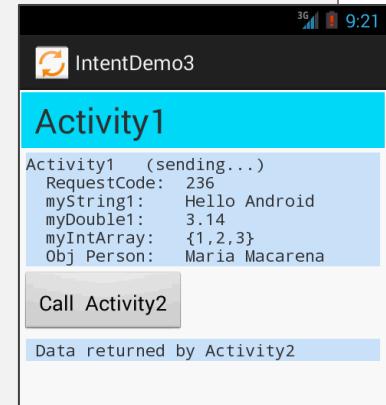
1 →

2 →

# Android Intents

## Example 18. Activity1.java cont 2.

```
@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");  
            }  
        }  
    } catch (Exception e) {  
        Log.e("IntentDemo3", "Error in Activity1", e);  
    }  
}
```



③ →

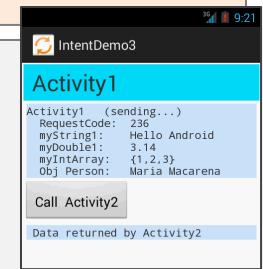
# Android Intents

## Example 18. Activity1.java cont 3.

```
// 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.myConvertArrayToString(myReturnedReversedArray));
} else {
    // user pressed the BACK button
    txtTop.setText("Selection CANCELLED!");
}

}

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



# Android Intents

## Example 18. Activity1.java cont 4.

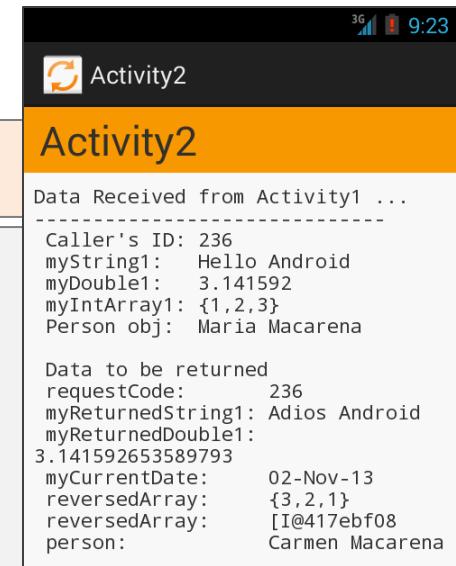
```
static String myConvertArrayToString(int[] intArray) {  
  
    if (intArray == null)  
        return "NULL";  
  
    String array2Str = "{" + Integer.toString( intArray[0] );  
  
    for (int i=1; i<intArray.length; i++) {  
  
        array2Str = array2Str + "," + Integer.toString( intArray[i] );  
    }  
    return array2Str + "}";  
}  
  
}// AndroIntent1
```



# Android Intents

## Example 18. Activity2.java (This is the Receiver)

```
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();
```



④

⑤

# Android Intents

## Example 18. Activity2.java cont 1.

```
//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);
```



# Android Intents

## **Example 18. Activity2.java cont 2.**



# Android Intents

## Example 18. Activity2.java cont 3.

```
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;
}
```



# Android Intents

## Example 18. Activity2.java cont 4.

```
// ++++++  
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());  
        }  
    }  
}
```



8 →

# Android Intents

## Example 18. Activity2.java cont 5.

```
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
```



# Android Intents

## Example 18. Activity2.java cont 6.

```
public class Person implements Serializable {  
    9→  
        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 getFirstName() {  
            return firstName;  
        }  
  
        public void setFirstName(String value) {  
            this.firstName = value;  
        }  
  
        public String getFullName() {  
            return firstName + " " + lastName;  
        }  
}
```

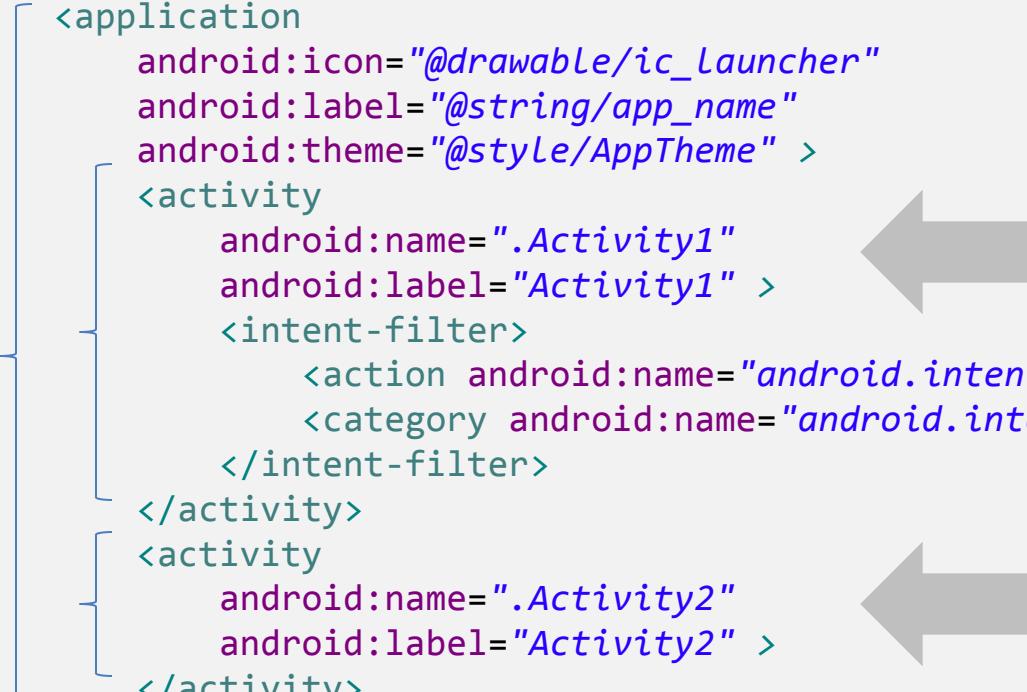


# Android Intents

## Example 18. 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="17" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".Activity1"
            android:label="Activity1" >
            <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>
```



# Android Intents

## Example 18.

### Comments

1. Various data items are inserted into an outgoing bundle, including simple types, an array, and a serialized object.
2. Activity1 invokes Activity2 and waits for its results.
3. The listener in Activity1 issues the statement `data.getExtras()` to get the returning bundle. Data items are extracted and displayed.
4. Activity2 uses `getIntent()` to capture the incoming intent request.
5. The arriving bundle is picked up and its data items are extracted. Observe the use of `.getSerializable(...)` to grab the object argument.
6. After operating on the input data, Activity2 packs all the outgoing extra items in a bundle and attaches it to the original intent.
7. Activity2 finishes with a `RESULT_OK` termination code.
8. The method `extractDataFromBundle` is used to get all the `<key,value>` pairs packed in a bundle.
9. This fragment defines a simple `Person` class, with is private data members, constructor, accessors, and custom method.

# Android Intents

**Questions ?**

# Android Intents

## Appendix A. Built-In Intent Actions

A complete list of built-in, broadcast, service actions, categories, and features for a particular SDK can be found in the folders:

.../android-sdk/platforms/platform-YYY/data/

### android.app.action.

ACTION\_PASSWORD\_CHANGED  
ACTION\_PASSWORD\_EXPIRING  
ACTION\_PASSWORD\_FAILED  
ACTION\_PASSWORD\_SUCCEEDED  
ADD\_DEVICE\_ADMIN  
DEVICE\_ADMIN\_DISABLE\_REQUESTED  
DEVICE\_ADMIN\_DISABLED  
DEVICE\_ADMIN\_ENABLED  
SET\_NEW\_PASSWORD  
START\_ENCRYPTION

### android.bluetooth.a2dp.profile.action.

CONNECTION\_STATE\_CHANGED  
PLAYING\_STATE\_CHANGED

### android.bluetooth.adapter.action.

CONNECTION\_STATE\_CHANGED  
DISCOVERY\_FINISHED  
DISCOVERY\_STARTED

LOCAL\_NAME\_CHANGED  
REQUEST\_DISCOVERABLE  
REQUEST\_ENABLE  
SCAN\_MODE\_CHANGED  
STATE\_CHANGED

### android.bluetooth.device.action.

ACL\_CONNECTED  
ACL\_DISCONNECT\_REQUESTED  
ACL\_DISCONNECTED  
BOND\_STATE\_CHANGED  
CLASS\_CHANGED  
FOUND  
NAME\_CHANGED  
UUID

### android.bluetooth.devicepicker.action.

DEVICE\_SELECTED  
LAUNCH

# Android Intents

## Appendix A. Built-In Intent Actions cont. 1

### **android.bluetooth.headset.**

action.VENDOR\_SPECIFIC\_HEADSET\_EVENT  
profile.action.AUDIO\_STATE\_CHANGED  
profile.action.CONNECTION\_STATE\_CHANGED

ATTACH\_DATA  
BATTERY\_CHANGED  
BATTERY\_LOW  
BATTERY\_OKAY  
BOOT\_COMPLETED  
BUG\_REPORT  
CALL  
CALL\_BUTTON  
CAMERA\_BUTTON  
CHOOSER  
CONFIGURATION\_CHANGED  
CREATE\_LIVE\_FOLDER  
CREATE\_SHORTCUT  
DATE\_CHANGED  
DELETE  
DEVICE\_STORAGE\_LOW  
DEVICE\_STORAGE\_OK  
DIAL  
DOCK\_EVENT  
DREAMING\_STARTED  
DREAMING\_STOPPED  
EDIT

### **android.hardware.action.**

NEW\_PICTURE  
NEW\_VIDEO  
input.action.QUERY\_KEYBOARD\_LAYOUTS

### **android.intent.action.**

ACTION\_POWER\_CONNECTED  
ACTION\_POWER\_DISCONNECTED  
ACTION\_SHUTDOWN  
AIRPLANE\_MODE  
ALL\_APPS  
ANSWER  
APP\_ERROR  
ASSIST

# Android Intents

## Appendix A. Built-In Intent Actions cont. 2

### `android.intent.action.`

EVENT\_REMINDER  
EXTERNAL\_APPLICATIONS\_AVAILABLE  
EXTERNAL\_APPLICATIONS\_UNAVAILABLE  
FETCH\_VOICEMAIL  
GET\_CONTENT  
GTALK\_CONNECTED  
GTALK\_DISCONNECTED  
HEADSET\_PLUG  
INPUT\_METHOD\_CHANGED  
INSERT  
INSERT\_OR\_EDIT  
INSTALL\_PACKAGE  
LOCALE\_CHANGED  
MAIN  
MANAGE\_NETWORK\_USAGE  
MANAGE\_PACKAGE\_STORAGE  
MEDIA\_BAD\_REMOVAL  
MEDIA\_BUTTON  
MEDIA\_CHECKING

MEDIA\_EJECT  
MEDIA\_MOUNTED  
MEDIA\_NOFS  
MEDIA\_REMOVED  
MEDIA\_SCANNER\_FINISHED  
MEDIA\_SCANNER\_SCAN\_FILE  
MEDIA\_SCANNER\_STARTED  
MEDIA\_SEARCH  
MEDIA\_SHARED  
MEDIA\_UNMOUNTABLE  
MEDIA\_UNMOUNTED  
MUSIC\_PLAYER  
MY\_PACKAGE\_REPLACED  
NEW\_OUTGOING\_CALL  
NEW\_VOICEMAIL  
PACKAGE\_ADDED  
PACKAGE\_CHANGED  
PACKAGE\_DATA\_CLEARED  
PACKAGE\_FIRST\_LAUNCH  
PACKAGE\_FULLY\_REMOVED

# Android Intents

## Appendix A. Built-In Intent Actions cont. 3

### **android.intent.action.**

PACKAGE\_INSTALL  
PACKAGE\_NEEDS\_VERIFICATION  
PACKAGE\_REMOVED  
PACKAGE\_REPLACE  
PACKAGE\_RESTARTED  
PACKAGE\_VERIFIED  
PASTE  
PHONE\_STATE  
PICK  
PICK\_ACTIVITY  
POWER\_USAGE\_SUMMARY  
PROVIDER\_CHANGED  
PROXY\_CHANGE  
REBOOT  
RESPOND\_VIA\_MESSAGE  
RINGTONE\_PICKER  
RUN  
SCREEN\_OFF  
SCREEN\_ON  
SEARCH

SEARCH\_LONG\_PRESS  
SEND  
SEND\_MULTIPLE  
SENDTO  
SET\_ALARM  
SET\_WALLPAPER  
SYNC  
SYSTEM\_TUTORIAL  
TIME\_SET  
TIME\_TICK  
TIMEZONE\_CHANGED  
UID\_REMOVED  
UNINSTALL\_PACKAGE  
USER\_PRESENT  
VIEW  
VOICE\_COMMAND  
WALLPAPER\_CHANGED  
WEB\_SEARCH

# Android Intents

## Appendix A. Built-In Intent Actions cont. 4

### android.media.

action.CLOSE\_AUDIO\_EFFECT\_CONTROL\_SESSION  
action.DISPLAY\_AUDIO\_EFFECT\_CONTROL\_PANEL  
action.OPEN\_AUDIO\_EFFECT\_CONTROL\_SESSION  
ACTION\_SCO\_AUDIO\_STATE\_UPDATED  
AUDIO\_BECOMING\_NOISY  
RINGER\_MODE\_CHANGED  
SCO\_AUDIO\_STATE\_CHANGED  
VIBRATE\_SETTING\_CHANGED

### android.net.

conn.BACKGROUND\_DATA\_SETTING\_CHANGED  
conn.CONNECTIVITY\_CHANGE  
nsd.STATE\_CHANGED  
wifi.action.REQUEST\_SCAN\_ALWAYS\_AVAILABLE  
wifi.NETWORK\_IDS\_CHANGED  
wifi.p2p.CONNECTION\_STATE\_CHANGE  
wifi.p2p.DISCOVERY\_STATE\_CHANGE  
wifi.p2p.PEERS\_CHANGED  
wifi.p2p.STATE\_CHANGED

wifi.p2p.THIS\_DEVICE\_CHANGED

wifi.PICK\_WIFI\_NETWORK

wifi.RSSI\_CHANGED

wifi.SCAN\_RESULTS

wifi.STATE\_CHANGE

wifi.suplicant.CONNECTION\_CHANGE

wifi.suplicant.STATE\_CHANGE

wifi.WIFI\_STATE\_CHANGED

### android.nfc.action.

ADAPTER\_STATE\_CHANGED  
NDEF\_DISCOVERED  
TAG\_DISCOVERED  
TECH\_DISCOVERED

# Android Intents

## Appendix A. Built-In Intent Actions cont. 5

### **android.settings.**

ACCESSIBILITY\_SETTINGS  
ADD\_ACCOUNT\_SETTINGS  
AIRPLANE\_MODE\_SETTINGS  
APN\_SETTINGS  
APPLICATION\_DETAILS\_SETTINGS  
APPLICATION DEVELOPMENT SETTINGS  
APPLICATION\_SETTINGS  
BLUETOOTH\_SETTINGS  
DATA\_ROAMING\_SETTINGS  
DATE\_SETTINGS  
DEVICE\_INFO\_SETTINGS  
DISPLAY\_SETTINGS  
DREAM\_SETTINGS  
INPUT\_METHOD\_SETTINGS  
INPUT\_METHOD\_SUBTYPE\_SETTINGS  
INTERNAL\_STORAGE\_SETTINGS  
LOCALE\_SETTINGS  
LOCATION\_SOURCE\_SETTINGS  
MANAGE\_ALL\_APPLICATIONS\_SETTINGS  
MANAGE\_APPLICATIONS\_SETTINGS

MEMORY\_CARD\_SETTINGS  
NETWORK\_OPERATOR\_SETTINGS  
NFC\_SETTINGS  
NFC\_SHARING\_SETTINGS  
PRIVACY\_SETTINGS  
QUICK\_LAUNCH\_SETTINGS  
SECURITY\_SETTINGS  
SETTINGS  
SOUND\_SETTINGS  
SYNC\_SETTINGS  
USER\_DICTIONARY\_SETTINGS  
WIFI\_IP\_SETTINGS  
WIFI\_SETTINGS  
WIRELESS\_SETTINGS

### **android.speech.tts.**

engine.CHECK\_TTS\_DATA  
engine.GET\_SAMPLE\_TEXT  
engine.INSTALL\_TTS\_DATA  
engine.TTS\_DATA\_INSTALLED  
TTS\_QUEUE\_PROCESSING\_COMPLETED