Android Intents

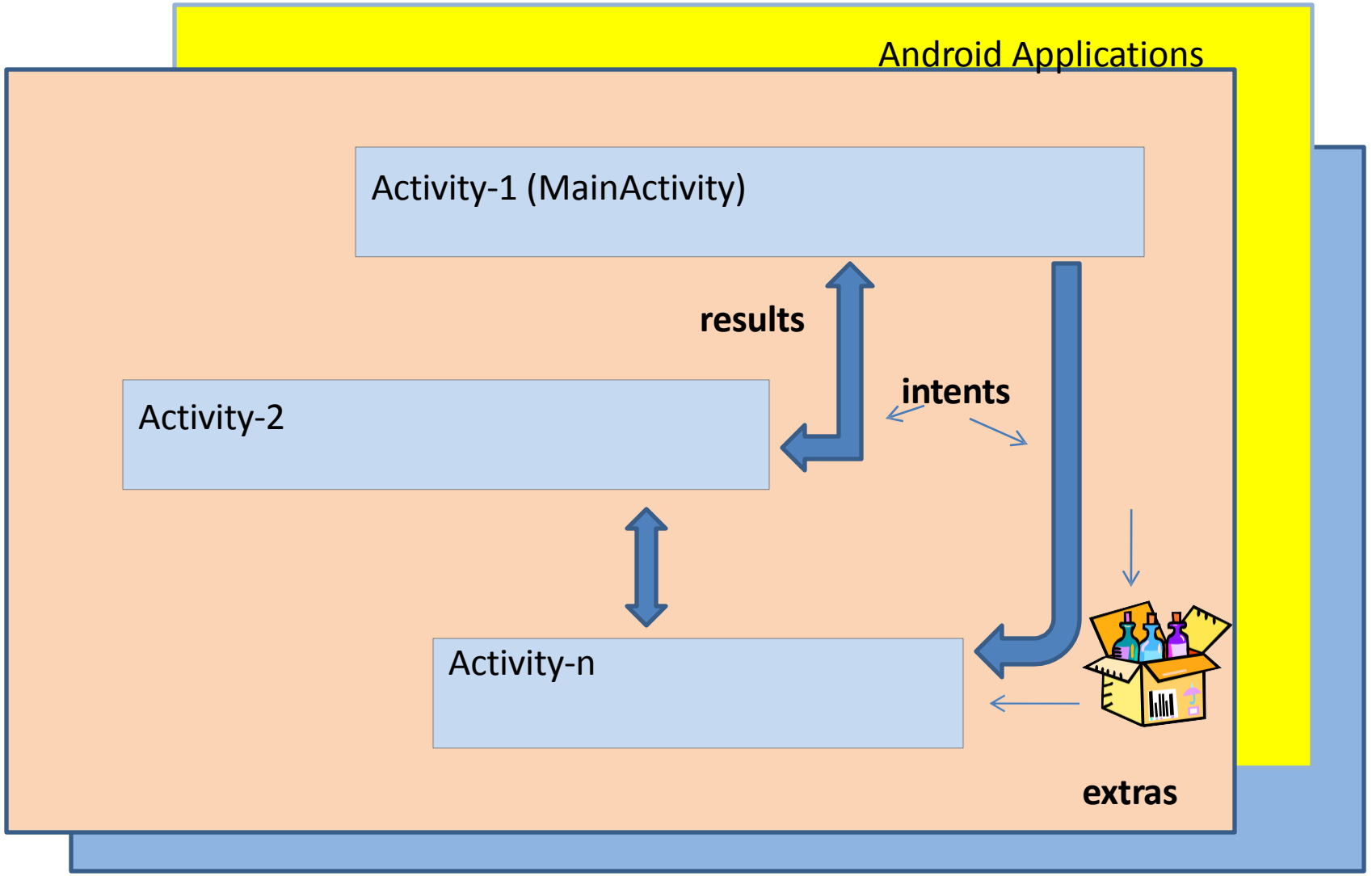
The majority of this lesson is reproduced from work created by Victor Matos, with permission.

Android Intents

Applications, Activities and Intents

- An Android **application** could include any number of **activities**.
- Activities are independent of each other; however they usually cooperate exchanging data and actions.
- Passing control and data from one activity to another is accomplished by asking the current activity to execute an **intent**.

Android Intents



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 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.

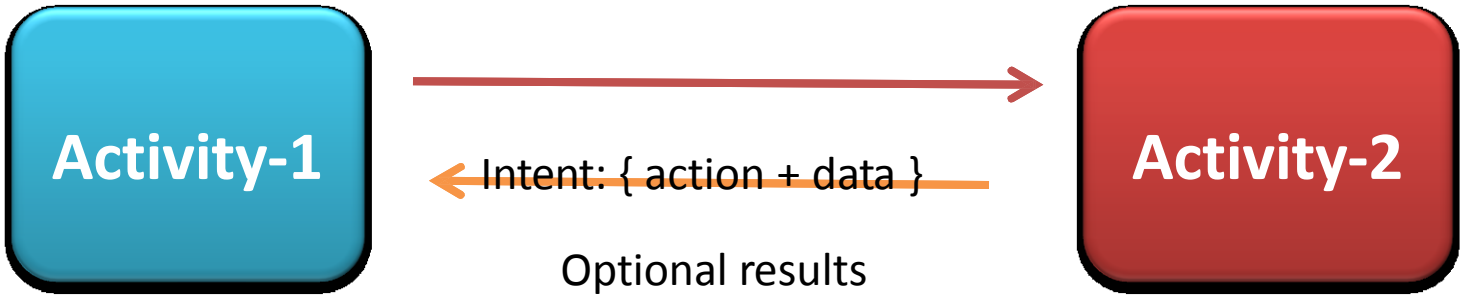
Android Intents

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

Initiating an Intent

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

Android Intents

Examples of action/data pairs:

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

Android Intents

Common Built-in Android Actions

List of common actions that Intents can use for launching built-in 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	

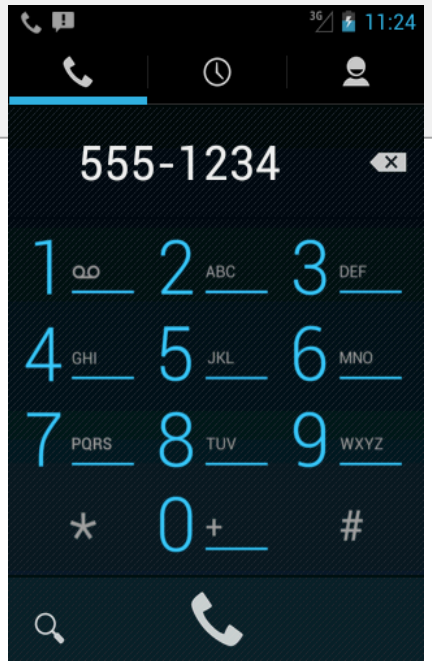
See Appendix A for a detailed list of selected built-in actions.

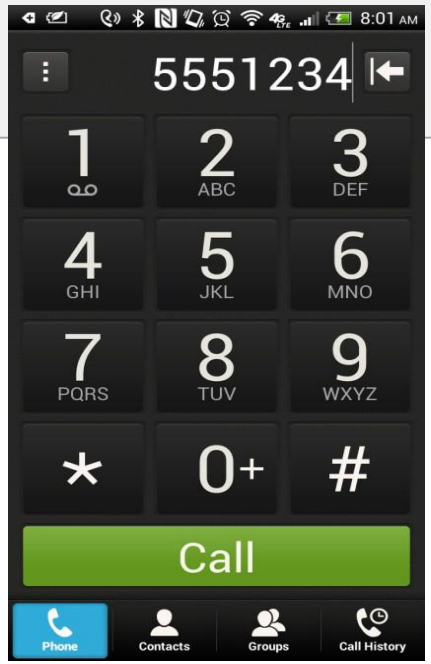
Android Intents

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

Example1B: ACTION_CALL

Placing and immediate phone call

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

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

startActivity(myActivity2);
```

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

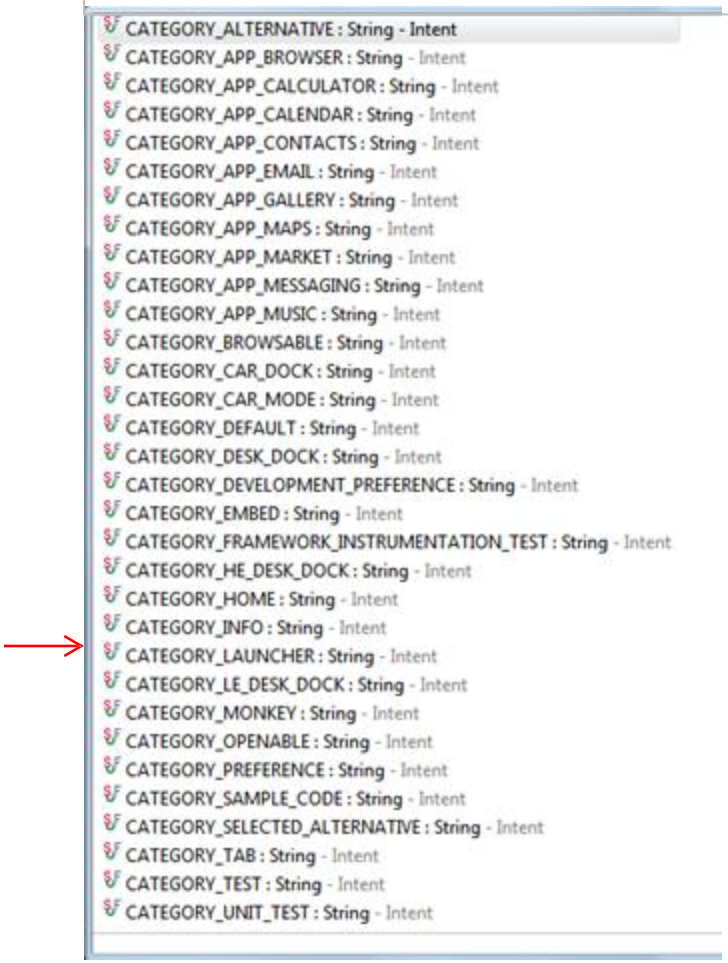
Extras

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”)

Android Intents

Example 2: ACTION_WEB_SEARCH

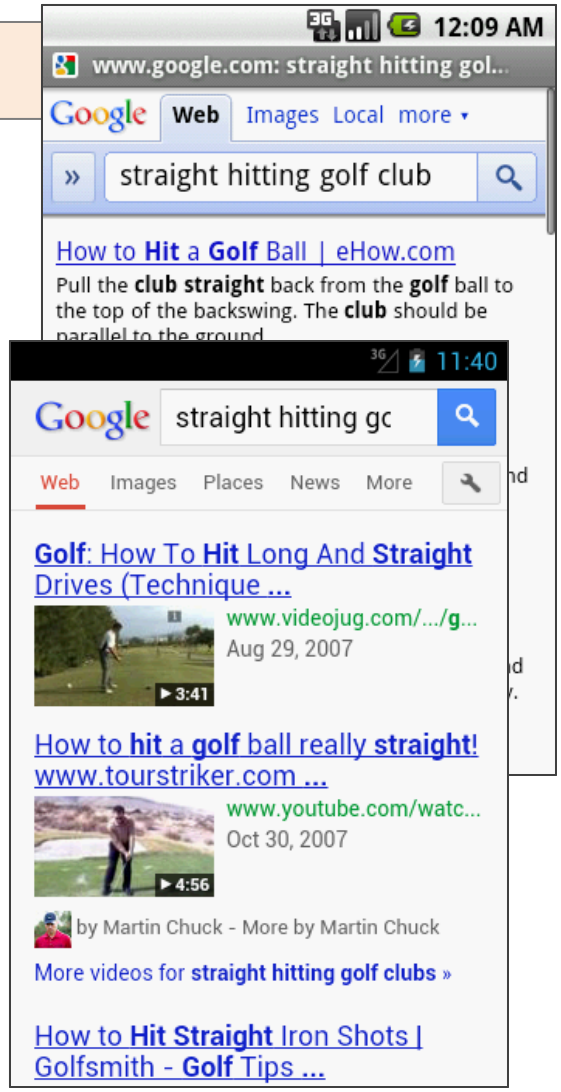
Using Secondary Attributes

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”

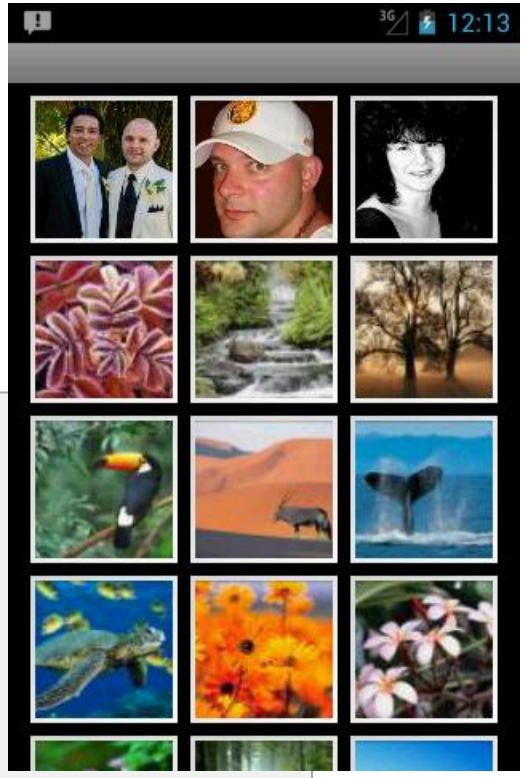
```
Intent intent = new Intent(
                Intent.ACTION_SENDTO,
                Uri.parse("smsto:555-4321"));

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(…)`

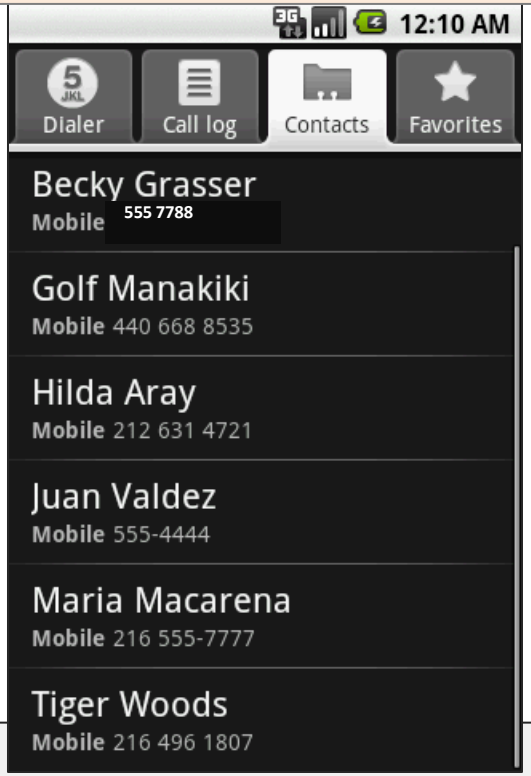


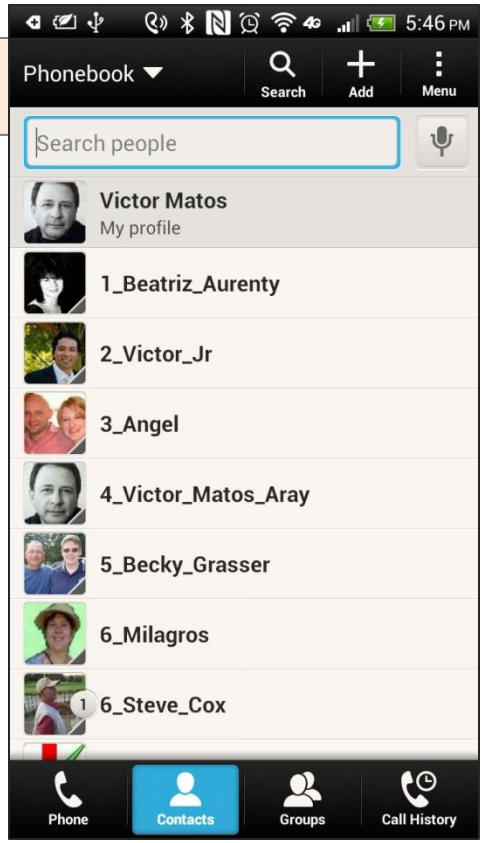

```
Intent intent = new Intent();

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





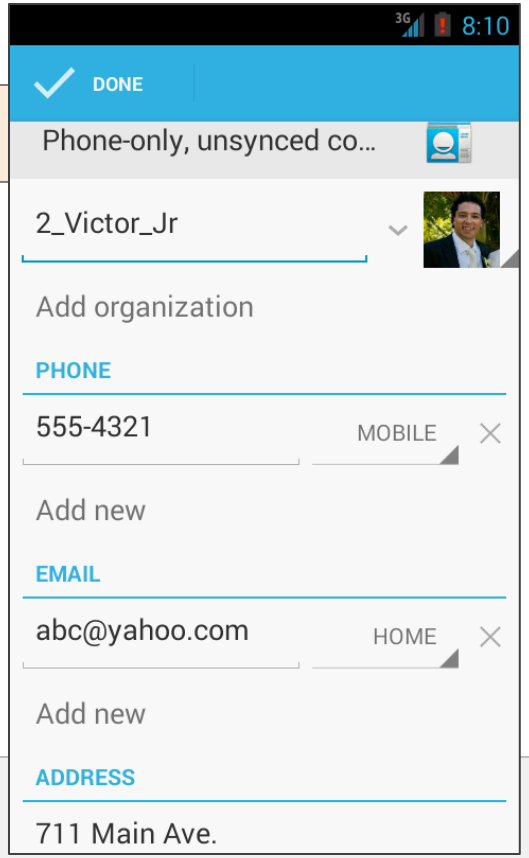

```
String myData = "content://contacts/people/";

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).

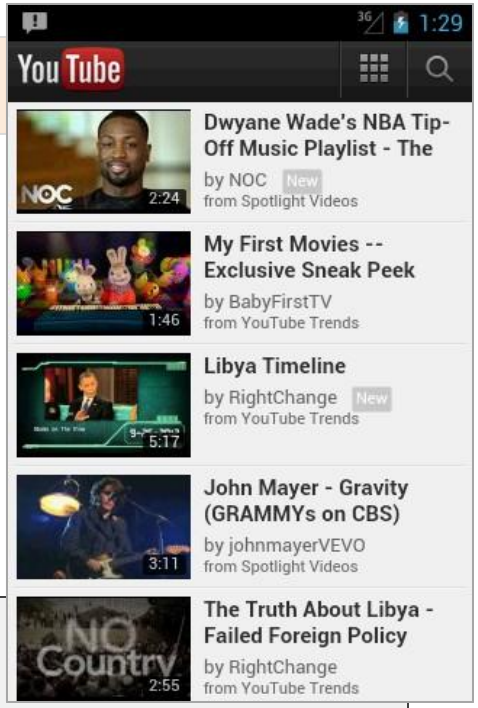



```
String myData = ContactsContract.Contacts
                .CONTENT_URI + "/" + "2";

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);
```

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

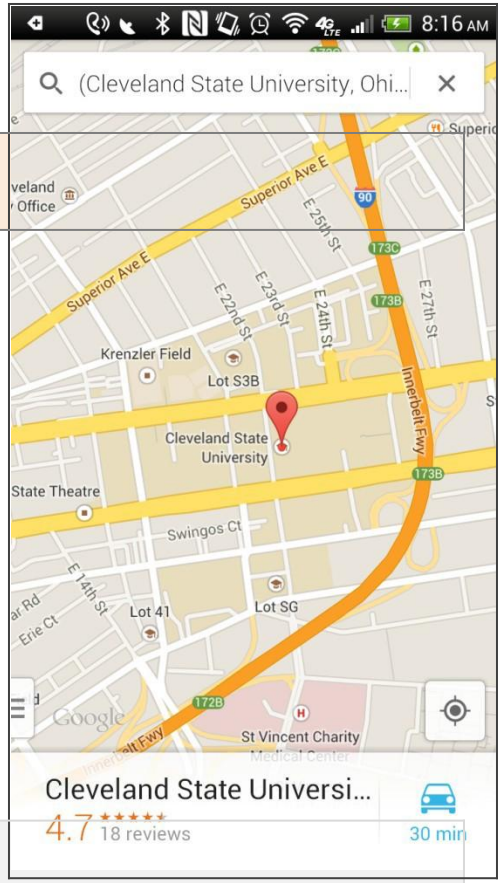
```
<uses-permission android:name="android.permission.INTERNET" />
```

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')



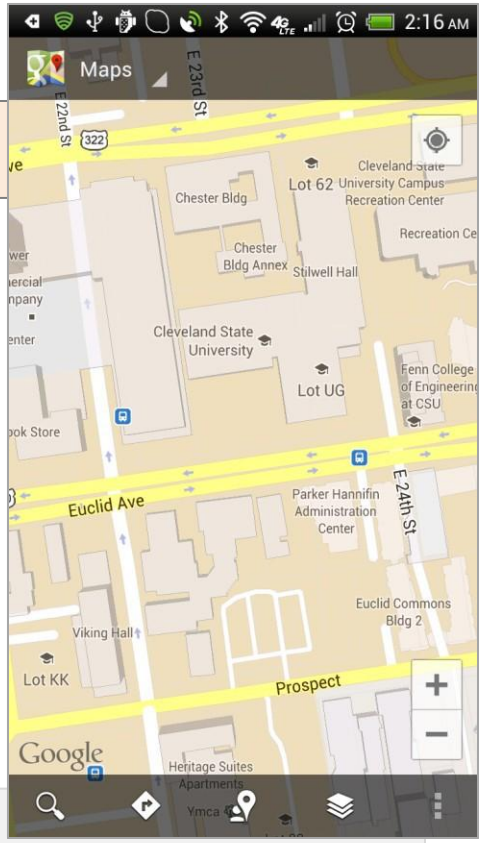

```
// (you may get multiple results...)
String thePlace = "Cleveland State University, Ohio";
Intent intent = new Intent(android.content.Intent.ACTION_VIEW,
                    Uri.parse("geo:0,0?q=(" + thePlace + ")"));
startActivity(intent);
```

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 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" />
```

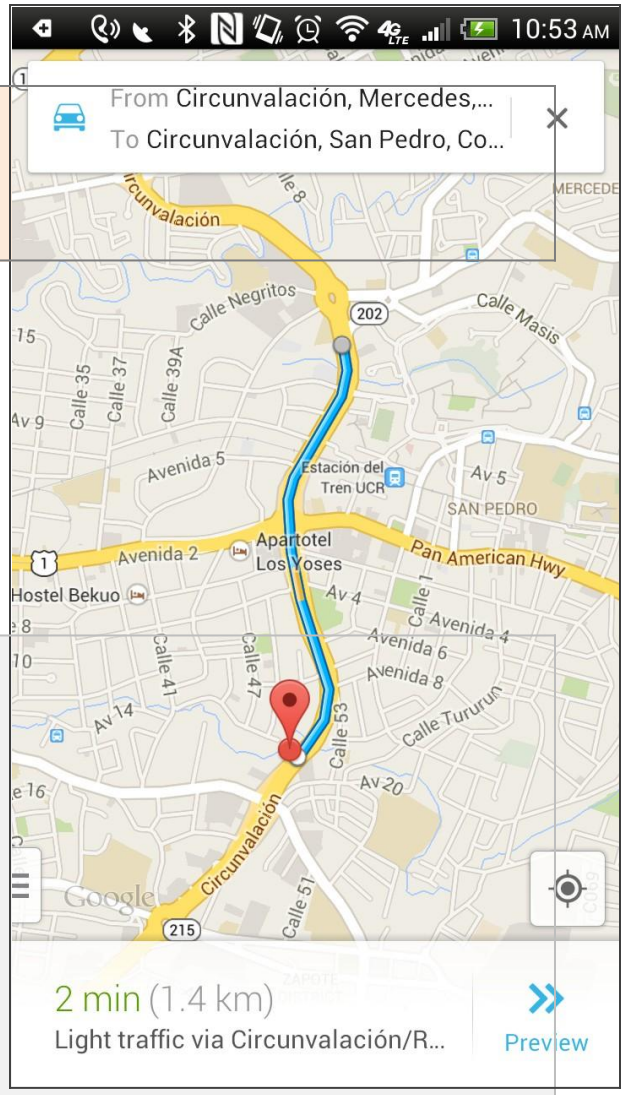
Android Intents

Example 10: ACTION_VIEW (Maps - Directions)

Getting driving directions

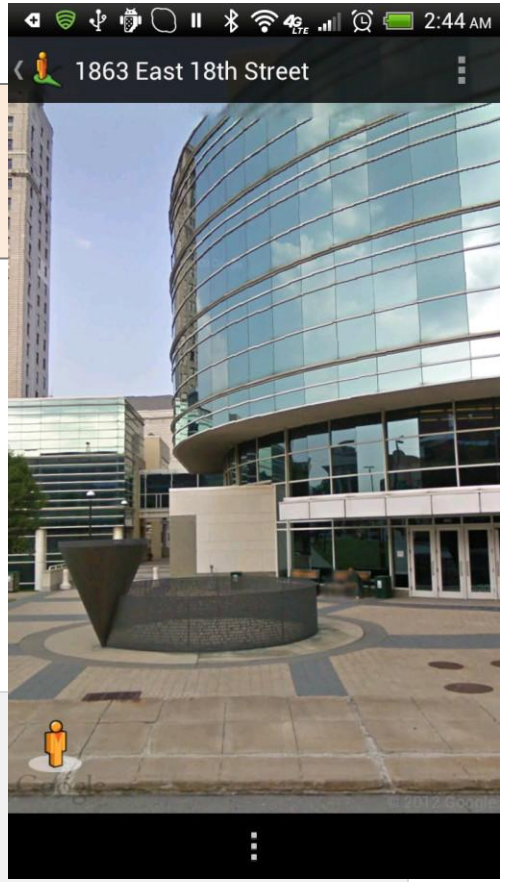
User provides GeoCodes (latitude,Longitude) for the starting and ending locations

```
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:cbll=***latitude,longitude***
&cbp=1,*yaw*,,*pitch*,*zoom*&mz=*mapZoom*

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

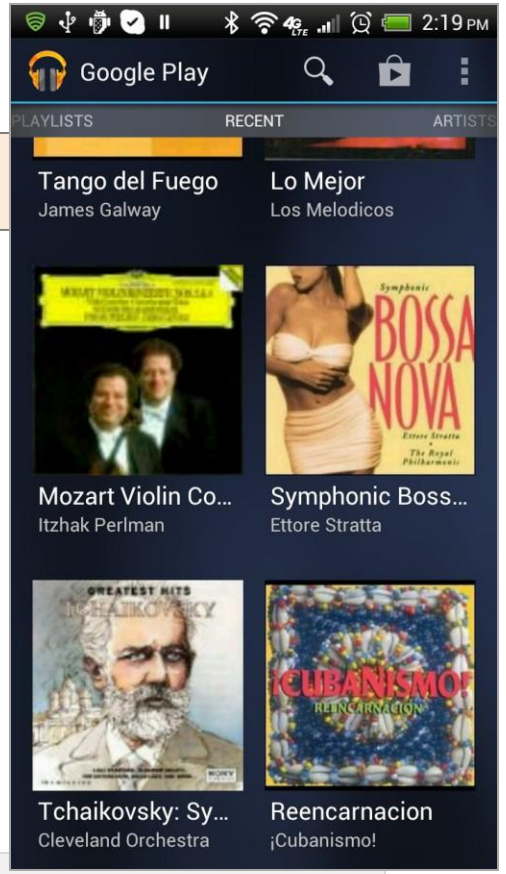
```
String geoCode = "google.streetview:"
               + "cbll=41.5020952,-81.6789717&"
               + "cbp=1,270,,45,1&mz=7";
Intent intent = new Intent(Intent.ACTION_VIEW,
                           Uri.parse(geoCode));
startActivity(intent);
```

Modify the Manifest adding the following requests:

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

Android Intents

Example 12: ACTION_MUSIC_PLAYER



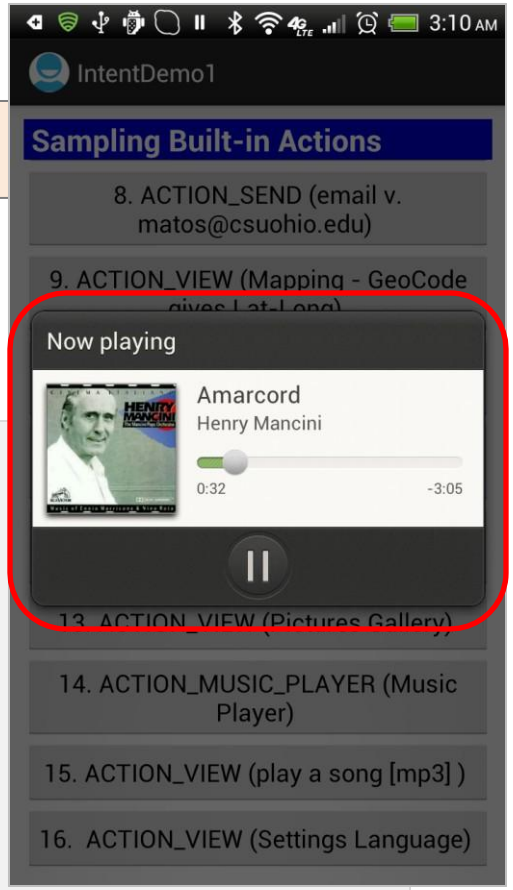
Launching the Music Player

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"/>
```

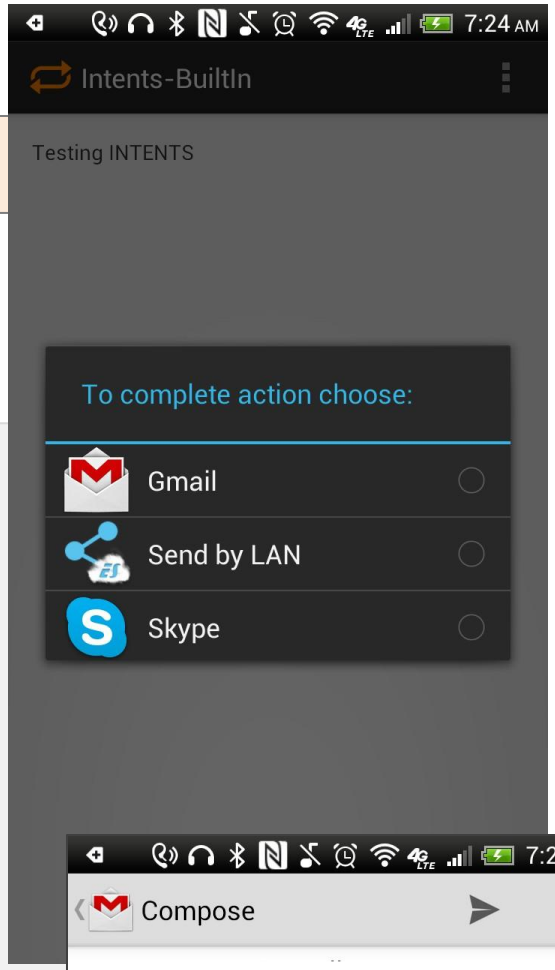
Android Intents

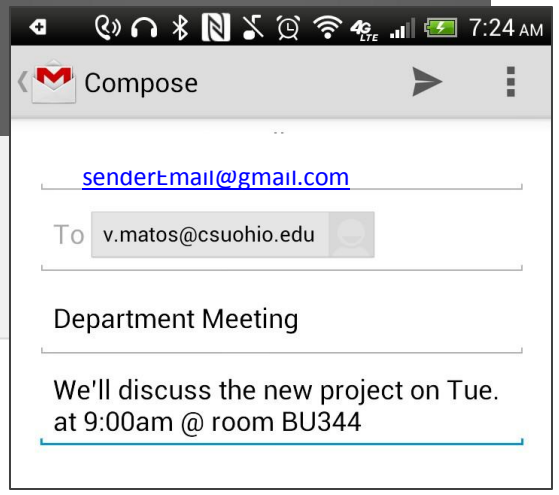
Example 14: ACTION_SEND (Email)

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:"));
```





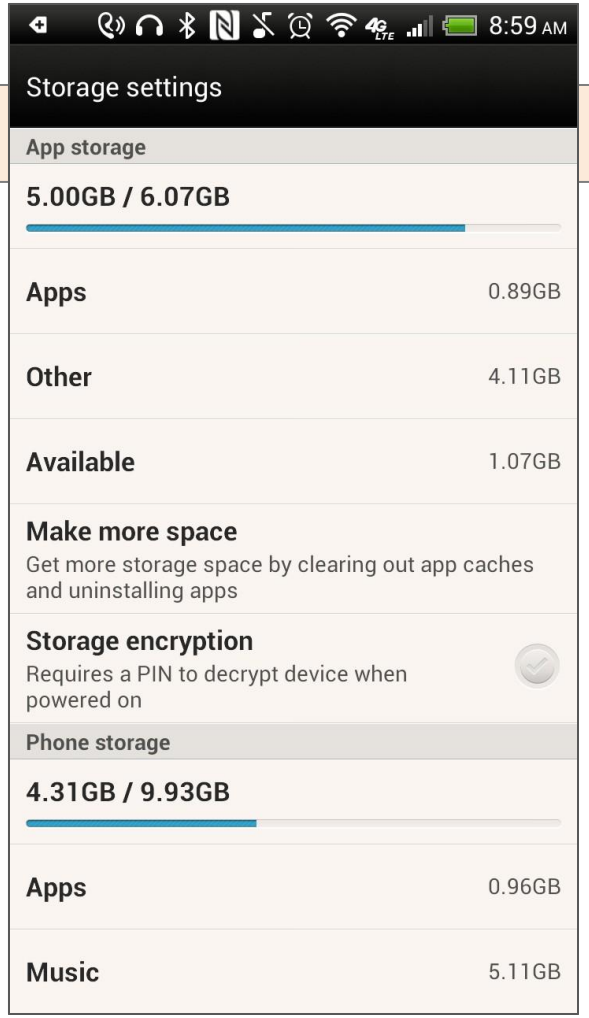
Example 15: Device Settings

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
             ));
```

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

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

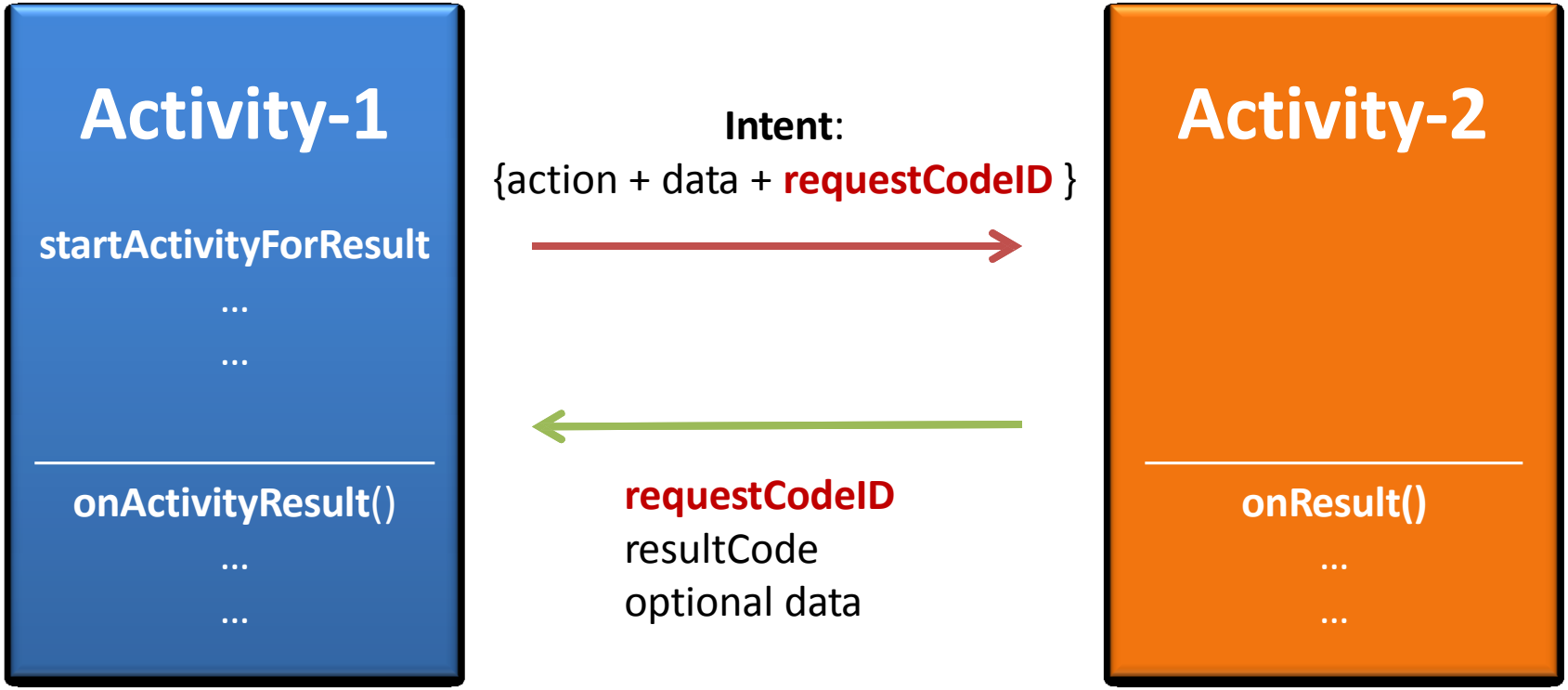
  `Activity.RESULT_CANCELED` (something bad happened),
  `Activity.RESULT_OK` (a happy ending),
  or any custom values.

- 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



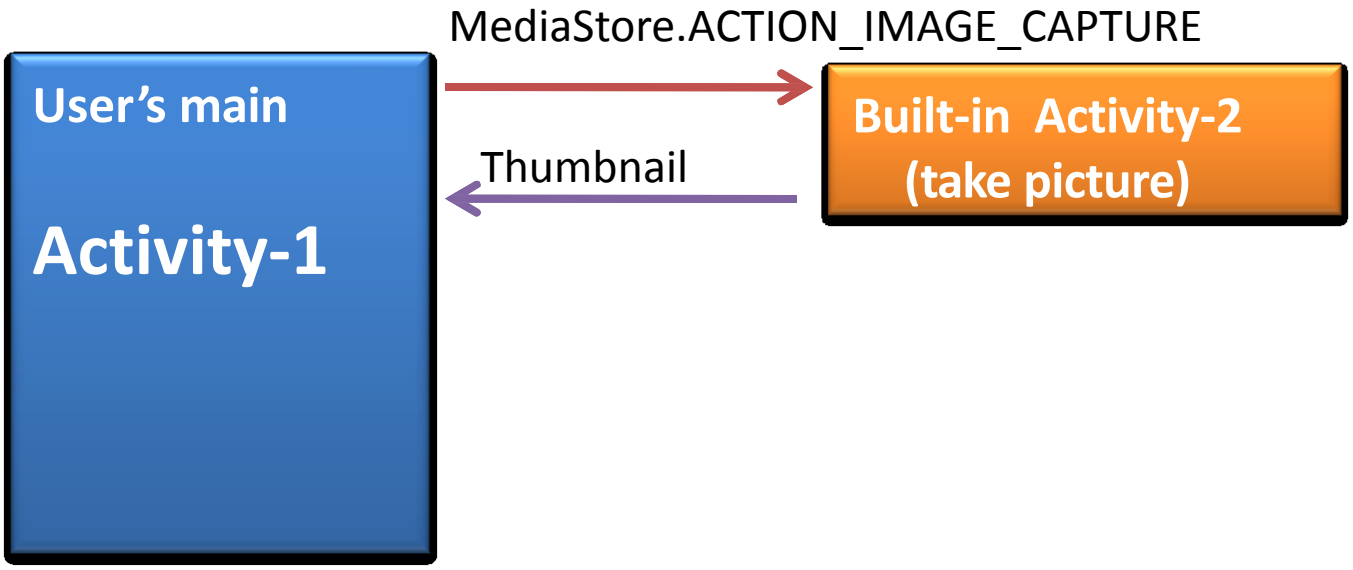
Android Intents

Example 16. Simple camera App

1. Launch the camera activity with MediaStore.ACTION_IMAGE_CAPTURE action.

2. Use the returned thumbnail to display.

Source: http://developer.android.com/training/camera/photobasics.html



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 → **Activity2: Receiver**

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

Bundle myBundle1 = new Bundle();

myBundle1.putInt ("val1", 123);

myIntentA1A2.putExtras(myBundle1);

startActivityForResult(myIntentA1A2, 1122);
```

INTENT
Sender class / Receiver class
requestCode (**1122**)
resultCode
Extras: { val1 = **123** }

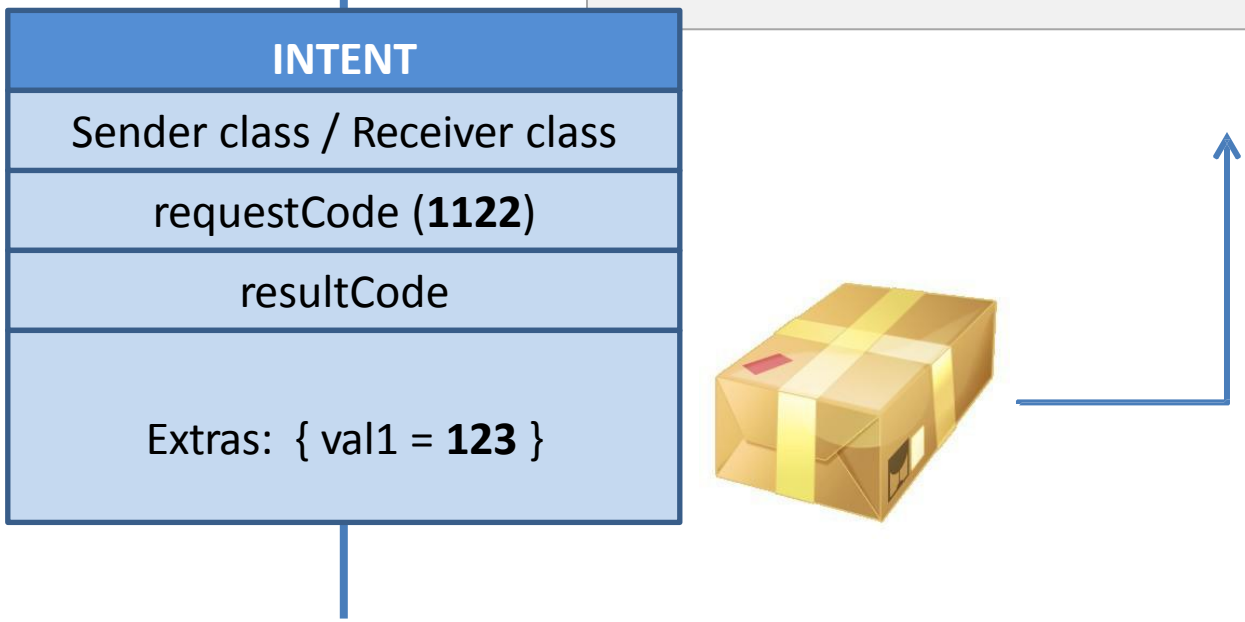
Android Intents

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);
```

INTENT
Sender class / Receiver class
requestCode (**1122**)
resultCode (**OK**)
Extras: { val1 = **456** }

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.

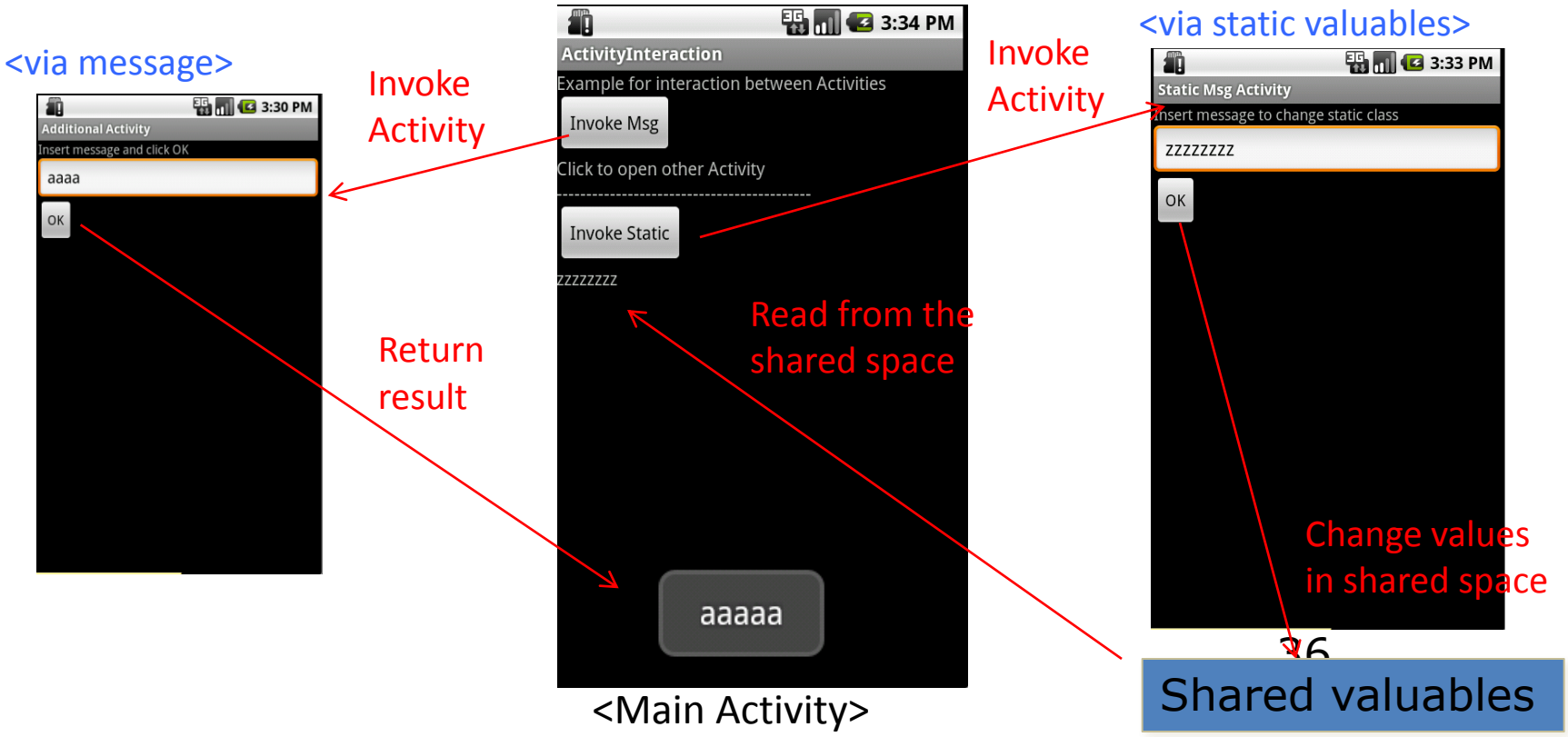
Reference: http://developer.android.com/reference/android/os/Bundle.html

Android Intents

Example

Goal

Create an application that has 3 Activities: a main activity and two sub-activities
Introduce two different methods for inter-activity communication: message return and static variable



Overview

Create two new sub-activities (in addition to the main activity)

- Create the two classes (.java)
- Create the two layouts (.xml)

Invoke the two new sub-activities

Have the sub-activities return results

Read the results of the sub-activities from the main activity

Invoke Sub-activities from the Main Activity

```
private static final int INTENT_GET_MSG      = 1;

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

    staticMsgTextView = (TextView)findViewById(R.id.textViewStaticMsg);

    invokeButton = (Button)findViewById(R.id.invokeButton);
    invokeButton.setOnClickListener(new OnClickListener() {
      public void onClick(View v) {
            Intent msgActivityIntent = new Intent(ActivityInteractionActivity.this, AdditionalActivity.class);
        startActivityForResult(msgActivityIntent, INTENT_GET_MSG);
      }
    });

    invokeStaticButton = (Button)findViewById(R.id.invokeStaticButton);
    invokeStaticButton.setOnClickListener(new OnClickListener() {
      public void onClick(View v) {
            Intent staticActivityIntent = new Intent(ActivityInteractionActivity.this, StaticMsgActivity.class);
            startActivity(staticActivityIntent);
      }
    });
  }
```

Name of your main activity class

for interaction via message return

for interaction via static valuables

Interaction via message return

```
- Invoke button clicked !
-startActivityForResult(msgActivityIntent,
                                INTENT_GET_MSG);
```

```
- OK button clicked !
- Intent intent = new Intent();
- intent.putExtra(RETURN_MSG, msg);
-setResult(Activity.RESULT_OK, intent);
-finish();
```

Build an intent
to return the result message

```
public void onActivityResult() {
        :
    case INTENT_GET_MSG:
          String returnMsg = data.getExtras()

.getString(AdditionalActivity.RETURN_MSG);
        :
  }
```

Receive the message

<Main Activity>

<Sub Activity>

Interaction via message return

Sub-activity for message return
When ok button clicked, return to the main Activity with result message via intent.

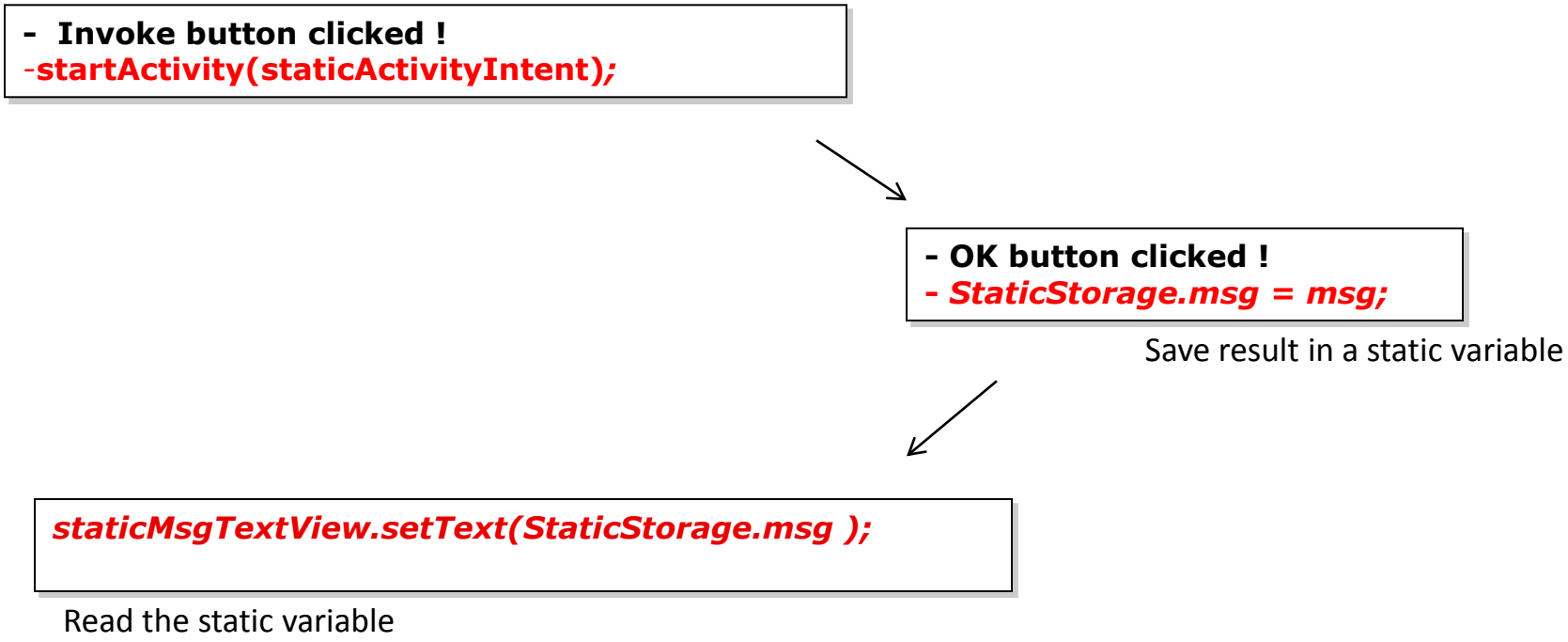
```
public Static String RETURN_MSG = "return_msg";

@Override
    public void onCreate(Bundle savedInstanceState) {
        :
        okButton.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                String msg = editText.getText().toString();

                Intent intent = new Intent();
                intent.putExtra(RETURN_MSG, msg);

                // Set result and finish this Activity
                setResult(Activity.RESULT_OK, intent);
                // The ActivityResult is propagated back to main activity
                // via onActivityResult().
                finish();
            }
        });
    }
```

Interaction via message return

Main Activity can receive the result message from sub-activity by overriding `onActivityResult()`

```
 @Override
    public void onActivityResult(int requestCode, int resultCode, Intent
data) {
        switch (requestCode) {

        case INTENT_GET_MSG:
            if (resultCode == Activity.RESULT_OK ) {
                 String returnMsg = data.getExtras()
                .getString(AdditionalActivity.RETURN_MSG);
                 Toast.makeText(this, returnMsg ,
Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(this, "Error !! ", Toast.LENGTH_SHORT).show();
            }
            break;
        }// end switch
    }
```

Interaction via static variables



Interaction via static variables

By defining a public static variable, both Activities can share the static variables. Create a new class we call StaticStorage (File>New>Class) – same package as the other java files.

```
package com.example.android;

public class StaticStorage {
        public static String msg = null;
}
```

Interaction via static variables

Sub-activity using static variables
When ok button clicked, save the result message in the static variable(s) shared between Activities.

```
@Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.static_msg_activity);

        editText = (EditText)findViewById(R.id.editText);
        okButton = (Button)findViewById(R.id.okButton);

        okButton.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                    String msg = editText.getText().toString();
                    StaticStorage.msg = msg; // Save result in a static variable
                    finish();
            }
        });
    }
```

Interaction via static variables

Main Activity access the static variable in `onResume()`

```
@Override
   public void onResume() {
        super.onResume();
        if( StaticStorage.msg != null && staticMsgTextView != null){
                staticMsgTextView.setText( StaticStorage.msg );
        }
   }
```

AndroidManifest.xml

Add two sub-activities in AndroidManifest.xml

```
<activity android:name=".AdditionalActivity"
          android:label="@string/app_additional_activity">
</activity>

<activity android:name=".StaticMsgActivity"
          android:label="@string/app_static_msg_activity">
</activity>
```

strings.xml

Add activity names used in AndroidManifest.xml

```
<string name="app_additional_activity">Additional Activity</string>
<string name="app_static_msg_activity">Static Msg Activity</string>
```

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

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
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 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_REPLACED
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.supplicant.CONNECTION_CHANGE
wifi.supplicant.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
NFCSHARING_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