INTENTS

Roberto Beraldi

Multi activity applications

- So far, we have seen an app is formed by an activity and many fragments
- In some cases, we may need to use more than one activity
- For example, during the app flow one can need to open the browser, or access to the contact list
- Also, a large app can be divided in 2 or more activities
- How do we pass from one activity to another?

Intents

- In adroid, an Activity is alwas started by an Intent
- Even an app starts with the intent mechanism: the main activity of an app is lanched by an intent generated by the launcher (the activity behind the home screen)
- What is an intent? An Intent is an asynchronous message that, in its simplest form carries the destination (or target) activity

Some attributes of an intent

- Action:
  - a string representing the operation the target activity should perform
    - For example: **android.intent.action.MAIN**

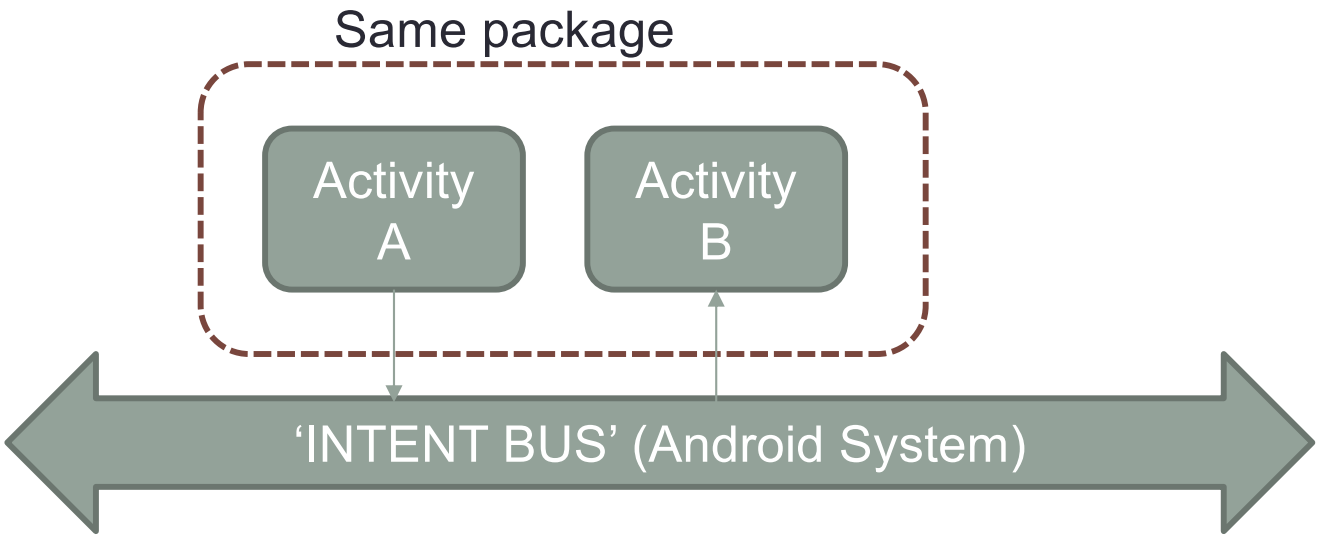
- Category:
  - A string representing additional information about the component that can manage the intent
    - For example: **android.intent.category.LAUNCHER** means it appears on the launcher
    - **android.intent.category.DEFAULT** (means open as full screen)

Orther attributes

- Data:
  - A URI that references data to operate on (*scheme://authority/path*)
    - For example: ***content://contacts/people/1*** → Display information about the person whose identifier is "1"
- Extras
  - Key-value pairs (i.e., a bundle) to carry additional information required to perform the requested action
    - For example, if the action it to send an e-mail message, one could also include extra pieces of data here to supply a subject, body, etc.

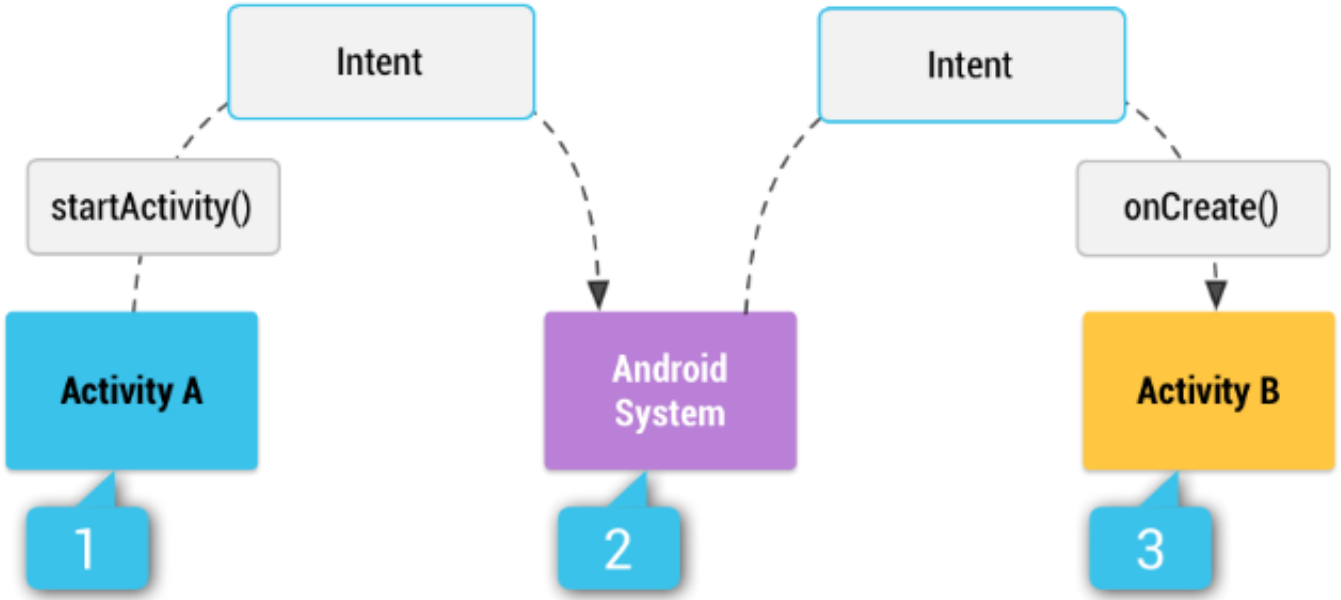
Intent: base case

A creates an Intent destined to B
A starts activity B




```
val i = Intent(this,SecondActivity::class.java)
startActivity(i)
```

Intent: base case behind the scene

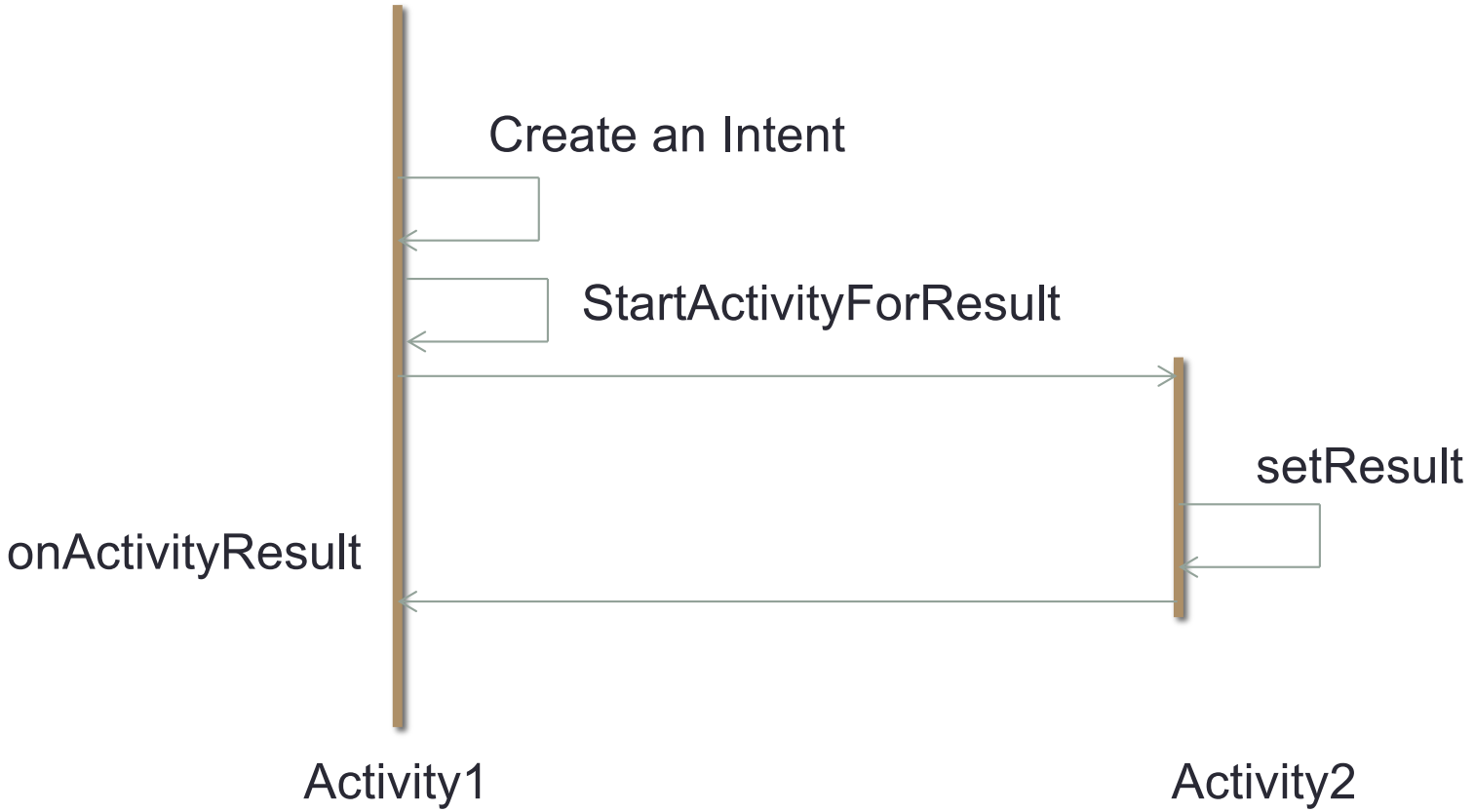


1. A Creates the intent and calls startActivity with a reference to B
2. The ‘android system’ checks if such an Activity is installed
3. The ‘android system’ calls the B’s onCreate method

Starting an activity and getting results

- An activity A can call an activity B and then get a result back from B
- This interaction is asynchronous
- The calling activity A will not wait
- The called activity will issue **setResult** method call
- This causes the **onActivityResult** method of the calling activity to be executed

Starting an activity and getting results



```kotlin
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val intent = Intent( packageContext: this,SecondActivity::class.java)
        startActivity(intent)
        startActivityForResult(intent, requestCode: 0)

    }

    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
    }

}
```

Starting an activity of another package

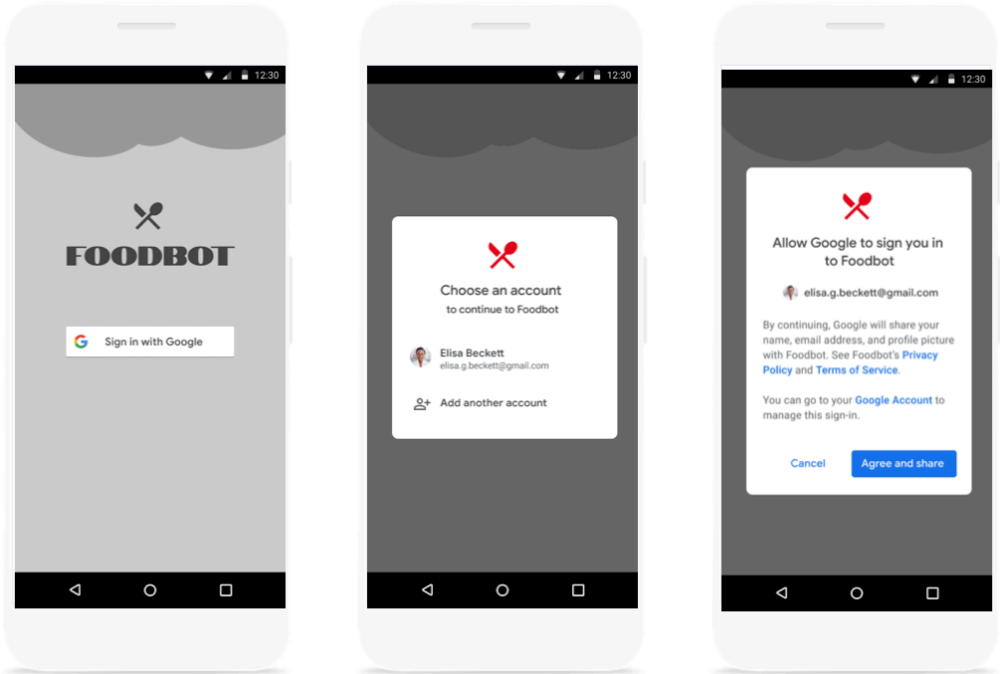
- Activity A can call and Activity B of a kwown pacakge by adding the package name to the intent

Example: Whatsup

- val sendIntent = Intent()
- sendIntent.*action* = Intent.*ACTION_SEND*
- sendIntent.putExtra(Intent.*EXTRA_TEXT*, "Hiii")
- sendIntent.*type* = "text/plain”
- sendIntent.setPackage("com.whatsapp")
- startActivity(sendIntent)

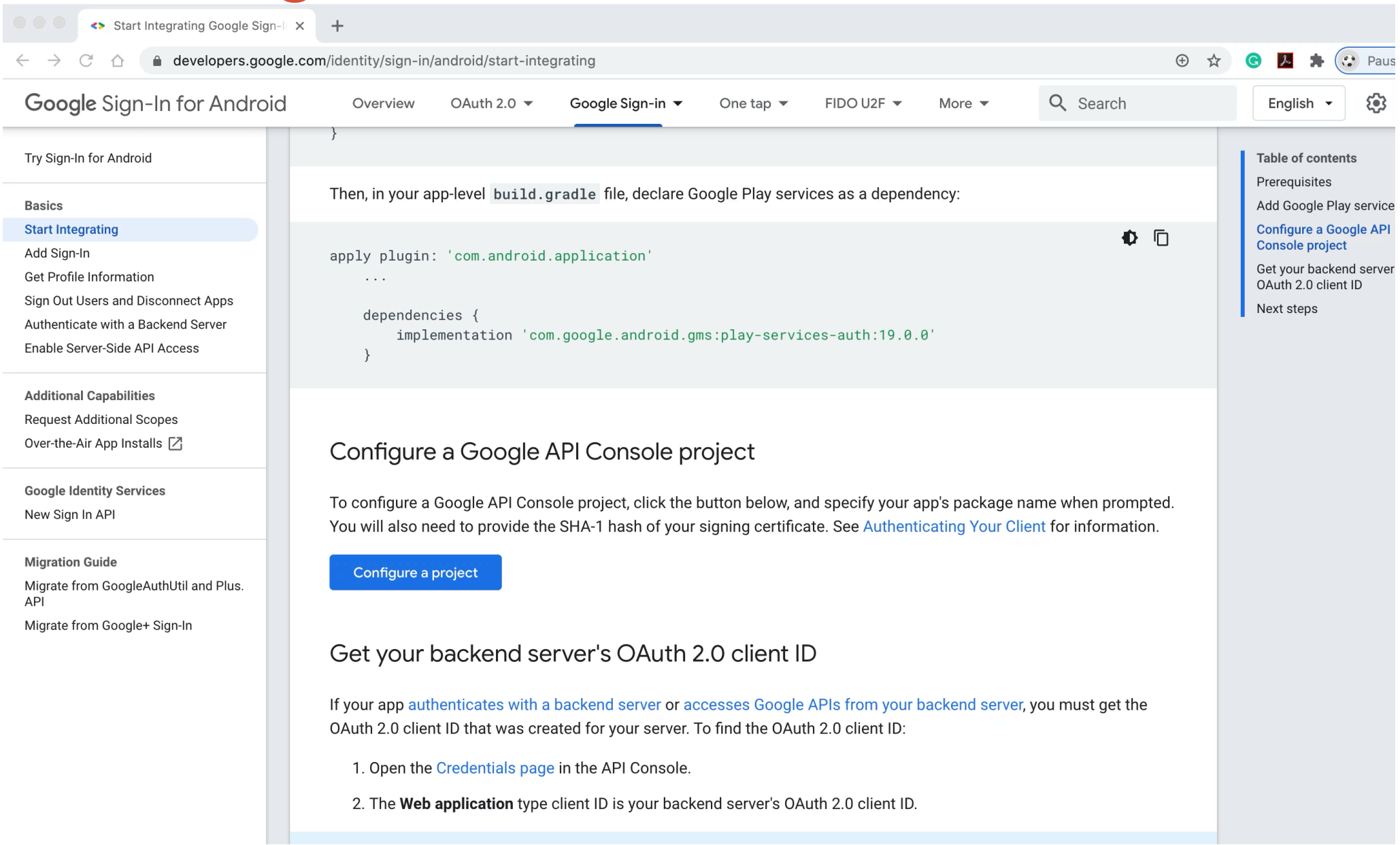
Another example: Google sign-in button

- Google Identity Services (GIS) is a new set of APIs that provides users easy and secure sign-in and sign-up, in an easy-to-implement package for developers.



https://developers.google.com/identity/sign-in/android/sign-in-identity

Adding the button



Calling an unknown activity

- Sometimes the package name is unknown or there can be more packages that can perform the same action

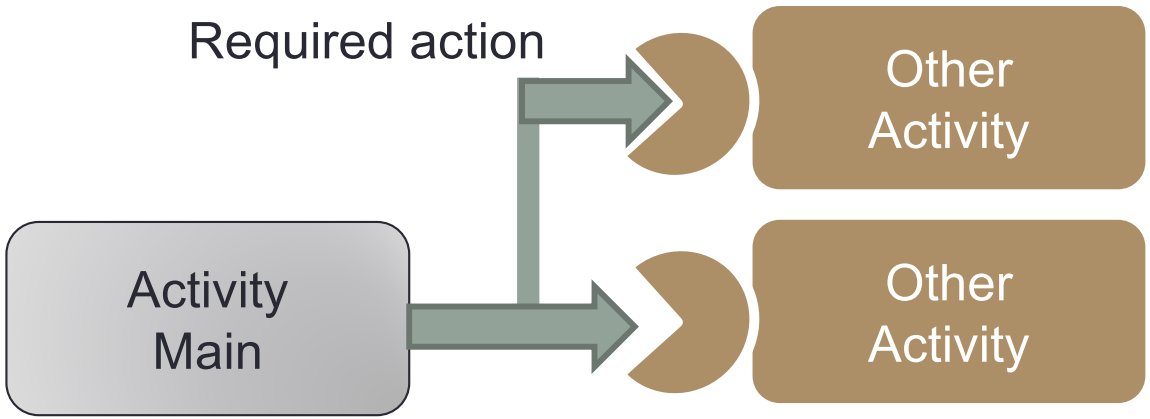
- For example, if more than one browser is installed the app can leave to the user the decision about which one to use

Implicit intents and filters

- **Implicit** : the intent just specifies the *action* the component should provide, not the target

- Any activity that declared the same action in the **intent-filters** tag in the manifest file can be started

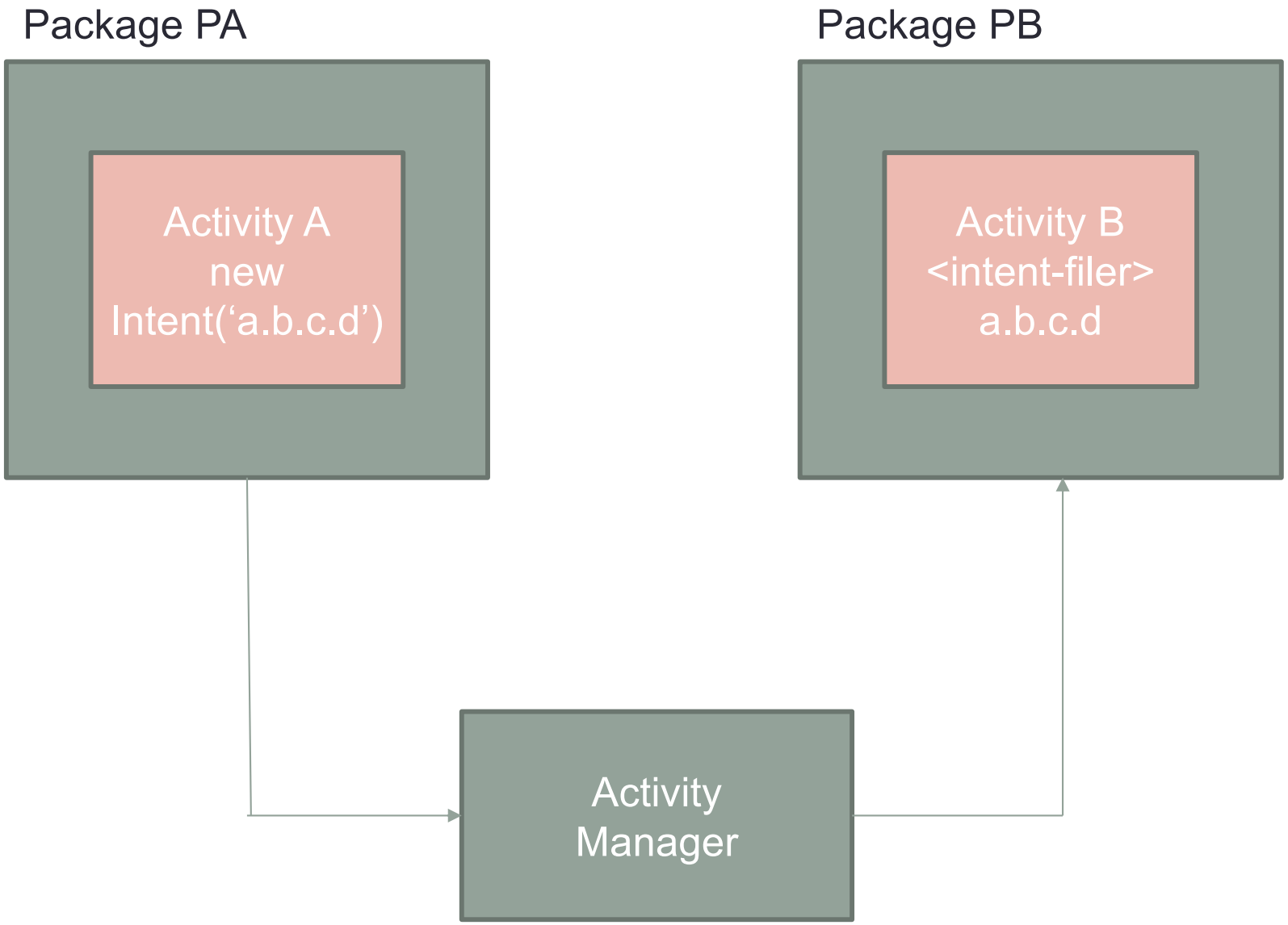
Implicit intents



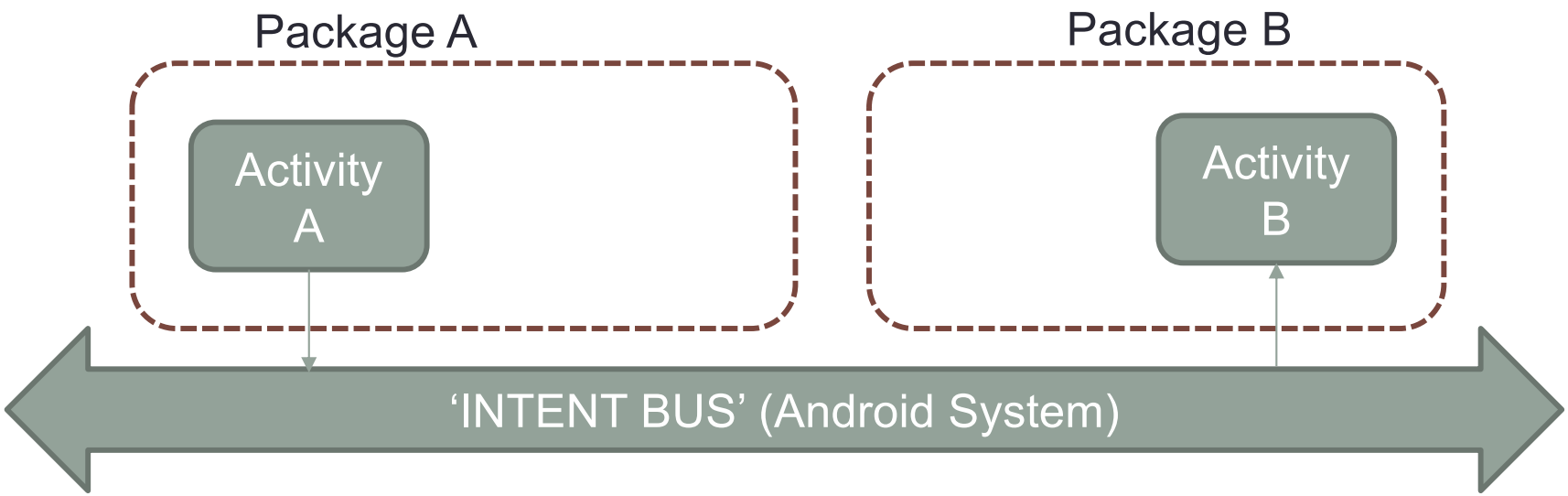
- The Intent doesn’t specify the Activity to start, but only an “Action”
- There are several predefined actions in the system to choose from

- A user can define its own action as well
- Intents declare their ability to perform actions in the manifest file

Starting an activity of another package



Big view


Package A
Package B
Activity A
Activity B
'INTENT BUS' (Android System)

Example: taking a picture

When there is just one activity that is able of managing the action,
The activity is executed immediately

```
val i = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
startActivityForResult(i, 1)
```

Intents and actions

- User can define new actions

- An app may be not allowed to generate some specific action as they are reserved to the system

Example of system defined actions

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

Example

- In this example, the system shows all the installed application that declares to be able to respond to the MAIN action

```
intent.setAction(Intent.ACTION_MAIN);
startActivity(intent);
```

Example

- Select a contact from the contact list
- Show the contact ID on the screen and view the details

```
val i = Intent()
i.action=Intent.ACTION_PICK
i.data= Uri.parse("content://contacts/people")
startActivity(i)
```

```
Intent intent = new Intent();
intent.setAction(Intent.ACTION_PICK);
intent.setData(Uri.parse("content://contacts/people/"));
startActivityForResult(intent,1);
```

```
protected void onActivityResult(int requestCode,int resultCode,Intent data) {

    if ((requestCode==1)&&(resultCode==Activity.RESULT_OK))
    {

        String selectedContact = data.getDataString();
        Toast.makeText(this, "Contact number:"+selectedContact, 1).show();
        startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(selectedContact)));

    }

}
```

Example of action/data pairs

ACTION_DIAL ***tel:123***
Display the phone dialer with the given number filled in.

ACTION_VIEW ***http://www.google.com***
Show Google page in a browser view. Note how the VIEW action does what is considered the most reasonable thing for a particular URI.

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

ACTION_VIEW ***content://contacts/people/2***
Used to start an activity to display 2-*nd* person.

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

Example: placing a call

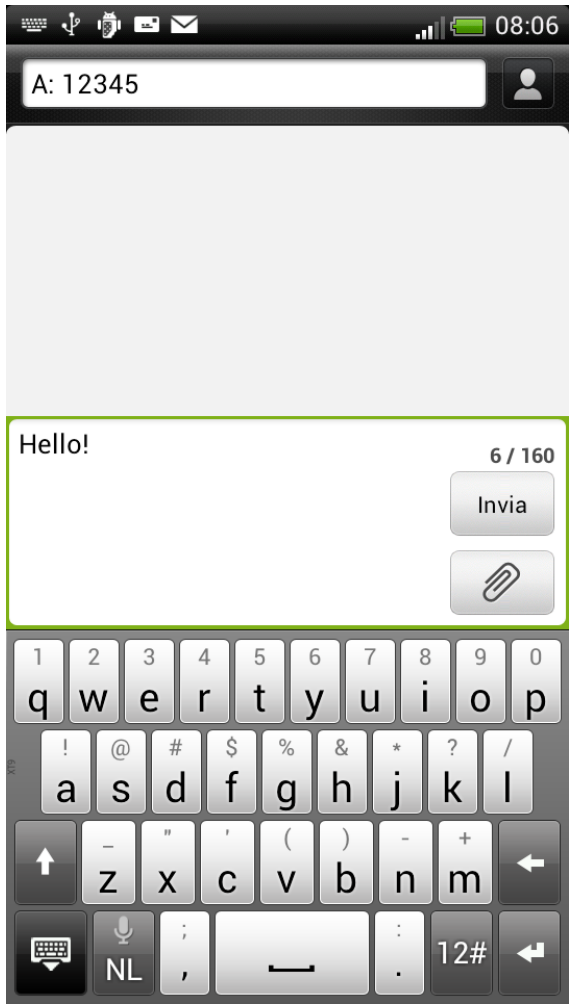
```
Intent intent = new Intent();
intent.setAction(Intent.ACTION_DIAL);
intent.setData(Uri.parse("tel:1234"));
startActivity(intent);
```

Same as

```
Intent intent = new Intent(Intent.ACTION_DIAL,Uri.parse("tel:1234"));
```

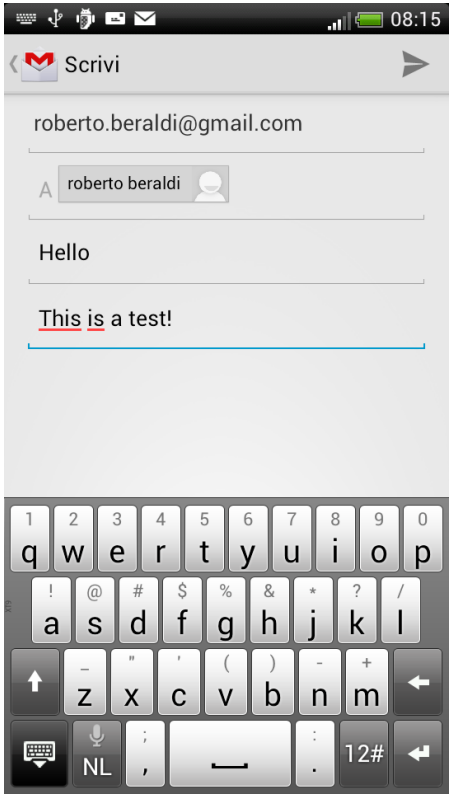
Example: sending sms

```
intent.setAction(Intent.ACTION_SENDTO);
intent.setData(Uri.parse("sms:12345"));
intent.putExtra("sms_body","Hello!");
```



Example: sending an email

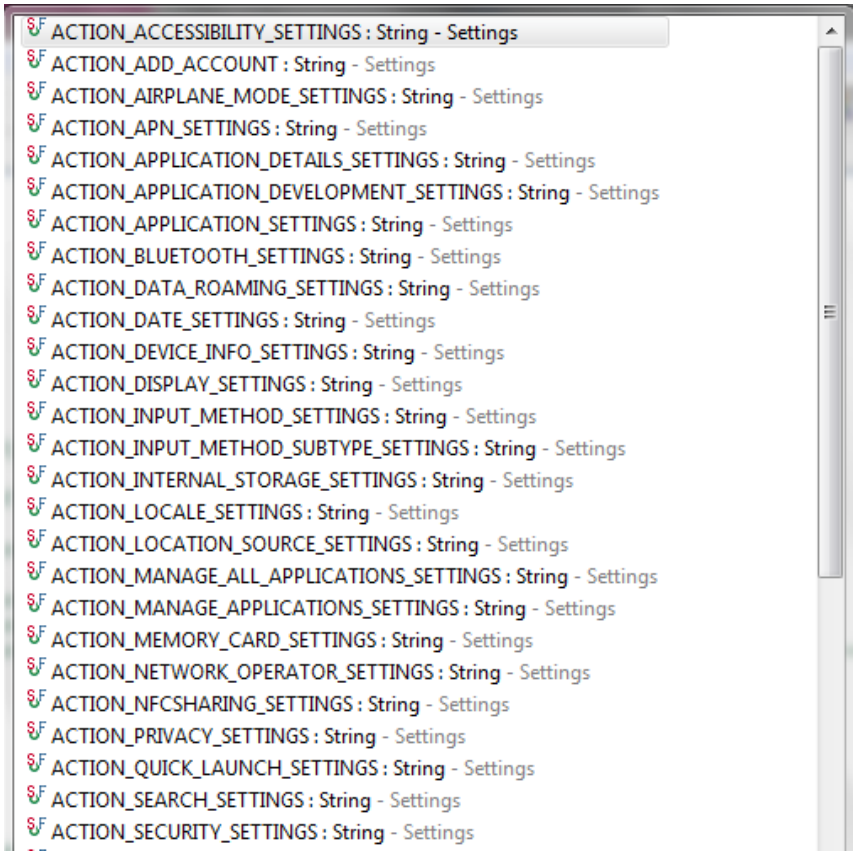
```
intent.setAction(Intent.ACTION_SENDTO);
intent.setData(Uri.parse("mailto:beraldi@dis.uniroma1.it"));
intent.putExtra(Intent.EXTRA_SUBJECT,"Hello");
intent.putExtra(Intent.EXTRA_TEXT,"This is a test!");
```



- The are two activities in the device that can perform the action
- The user needs to select one
- Can set the choice as the default

Another example: showing settings

```
intent.setAction(android.provider.Settings.ACTION_WIFI_IP_SETTINGS);
```



Passing data to the new activity

- When an Activity A has to pass some data to anohter activity it needs to set extra fields in the intent object

Passing data via a bundle

```
Intent intent = new Intent(MainActivity.this,Activity2.class);
Bundle bundle = new Bundle();
bundle.putDouble("temperature", 21.3);
int [] ia = {1,2,3};
bundle.putIntArray("array",ia);
bundle.putInt("int", 123);
intent.putExtras(bundle);
startActivityForResult(intent,2);
```

```
public class Activity2 extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        Intent intent = getIntent();
        setResult(Activity.RESULT_OK,intent);
        finish();
    }
}
```

Activity2

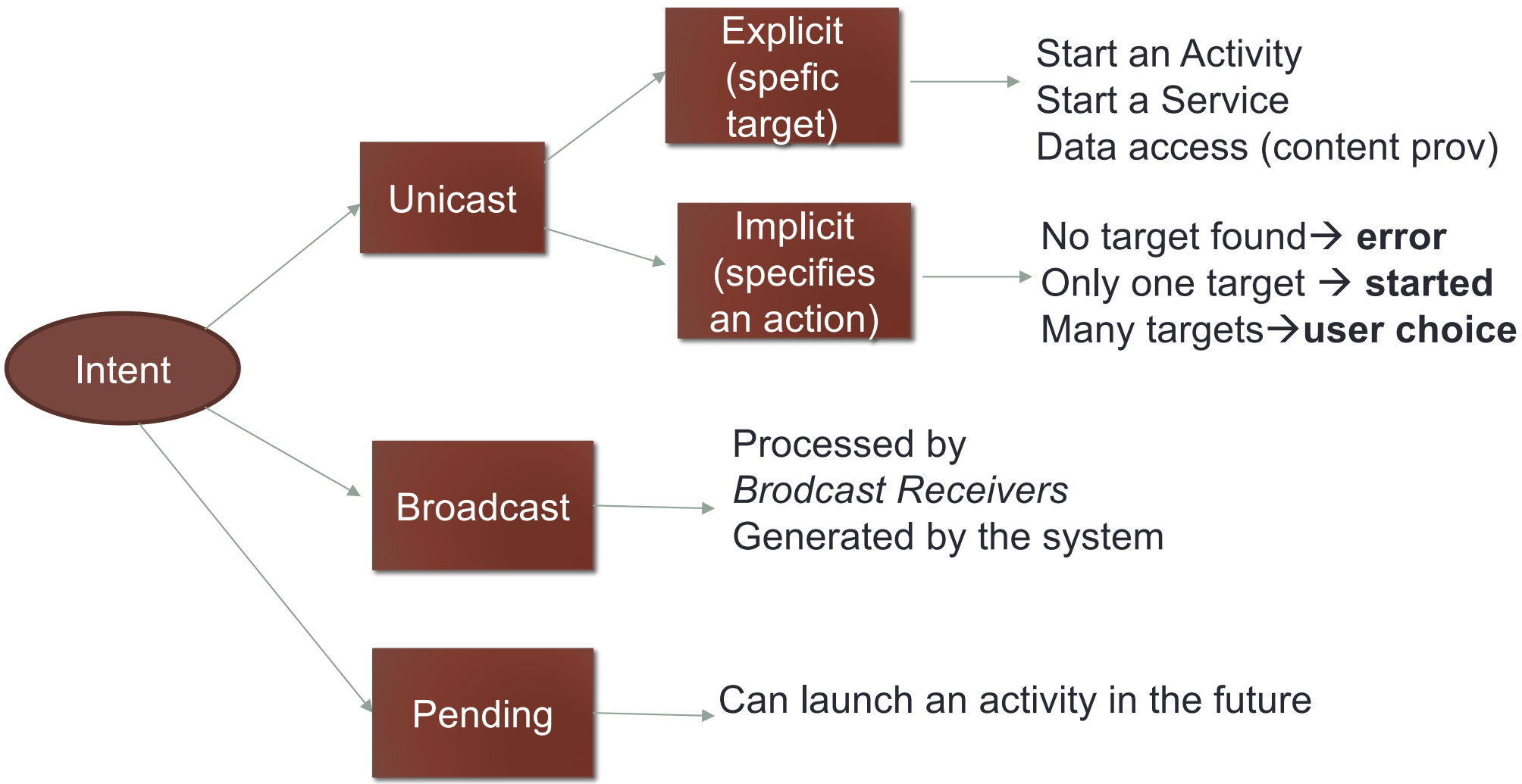
```
protected void onActivityResult(int requestCode,int resultCode,Intent data) {


    if ((requestCode==2)&&(resultCode==Activity.RESULT_OK))
    {
        Bundle b = data.getExtras();
        Toast.makeText(this, "ok:"+b.getInt("int")+b.getIntArray("array"), 1).show();

    }
}
```

Pending intents and broadcast intents

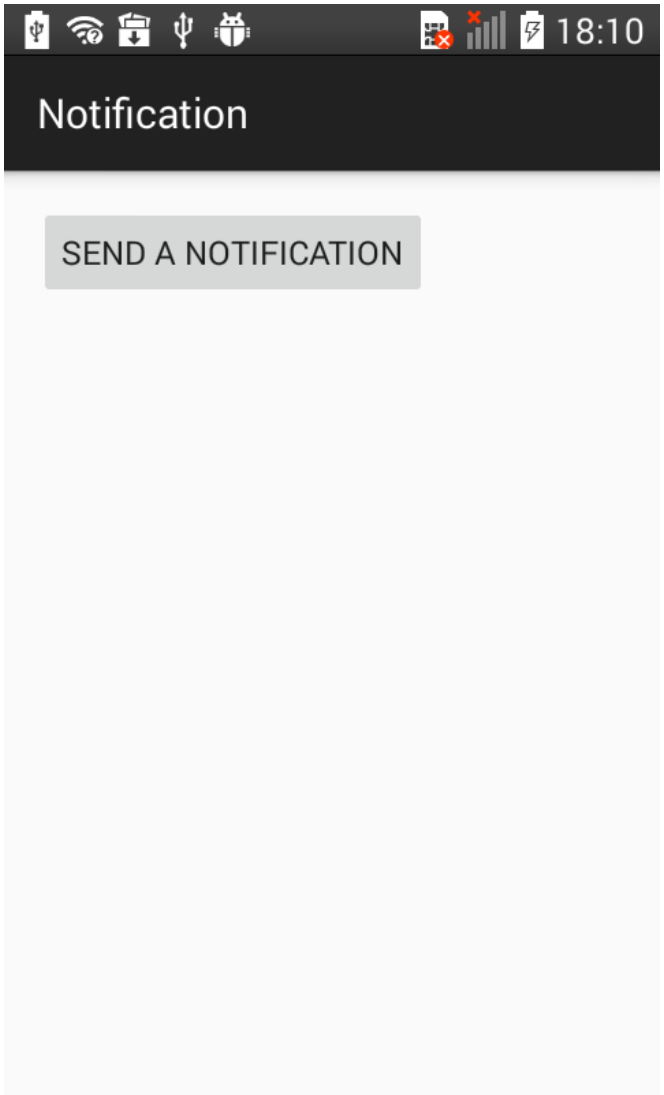
- An intent can also be **pending**, meaning that an activity will be activated in the future (e.g., notification)

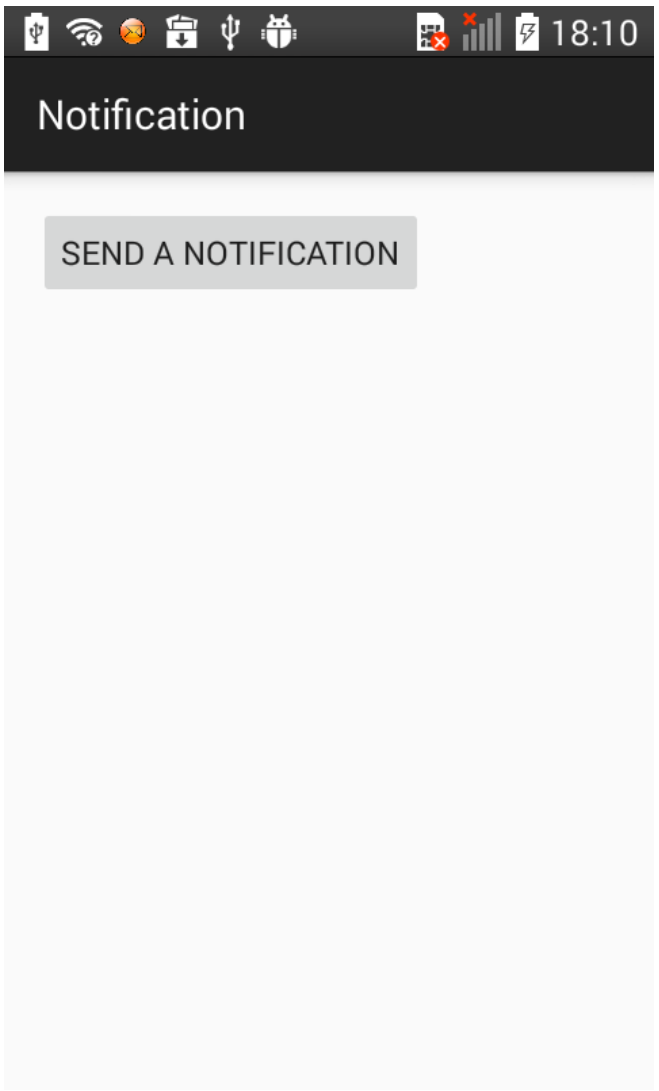
- It can also be **broadcast** when it announces something to all (see broadcast receivers)

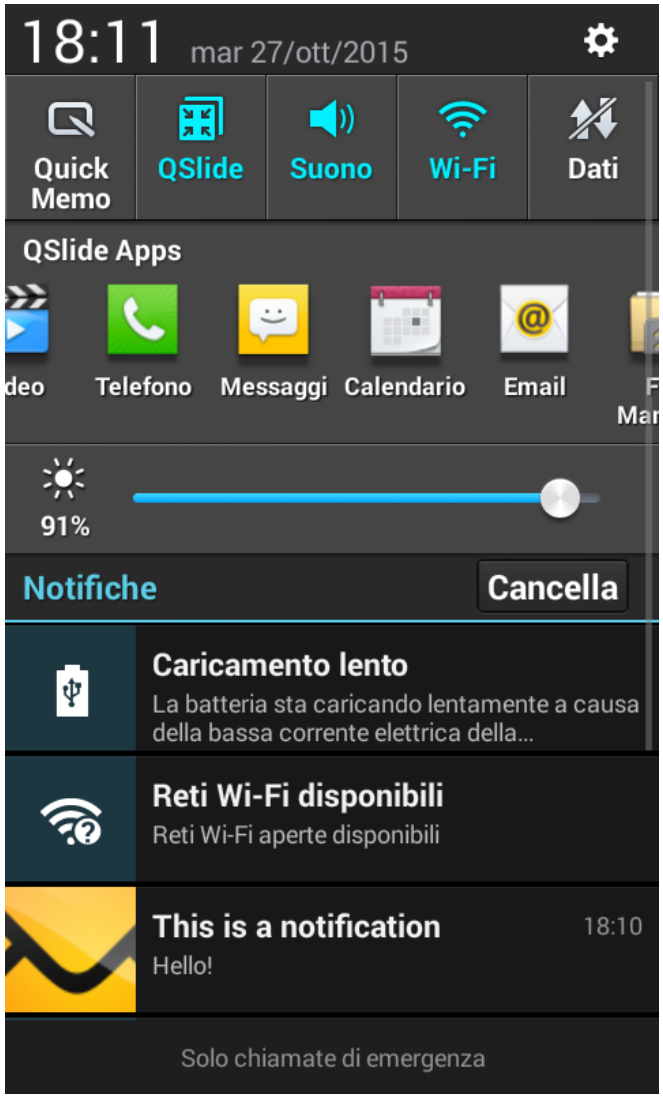
Intents: classification



Pendind intents: Notification







Broadcast intents

```
ACTION_TIME_TICK
ACTION_TIME_CHANGED
ACTION_TIMEZONE_CHANGED
ACTION_BOOT_COMPLETED
ACTION_PACKAGE_ADDED
ACTION_PACKAGE_CHANGED
ACTION_PACKAGE_REMOVED
ACTION_PACKAGE_RESTARTED
ACTION_PACKAGE_DATA_CLEARED
ACTION_UID_REMOVED
ACTION_BATTERY_CHANGED
ACTION_POWER_CONNECTED
ACTION_POWER_DISCONNECTED
ACTION_SHUTDOWN
```

The action specifies that an event is occurred

Broadcast Receiver

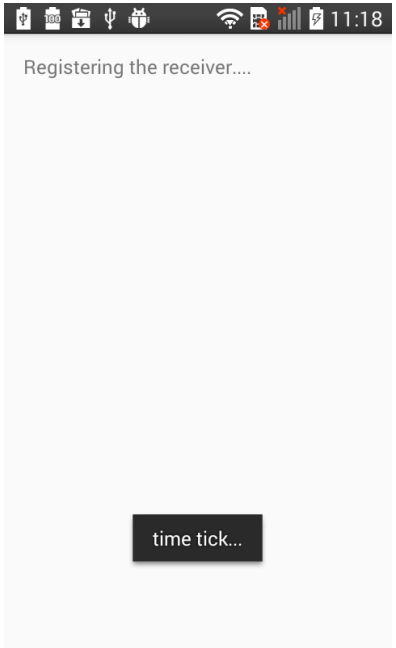
- It’s a software component that reacts to system-wide events (in the form bcast action)
- A receiver has to register to specific bcast intents
- When the intent occurs the *onReceive* method in the receiver is executed

Registering Broadcast receivers

- Registration to receive bcast intent can be done
- Statically (through XML, e.g., <receiver> tag)
- Dynamically (from an activity). Called in the UI thread..
- Statically registered receivers reamin dormant and respond to the intent - Only a feew can be registered this way (like for BOOT_COMPLETED event)
- Dynamically registered event are alive as long as the registering activity is alive
- Subscription to some events can only be done dynamically (e.g., TIME_TICK – this is to avoid battery drain)

BroadcastReceiver

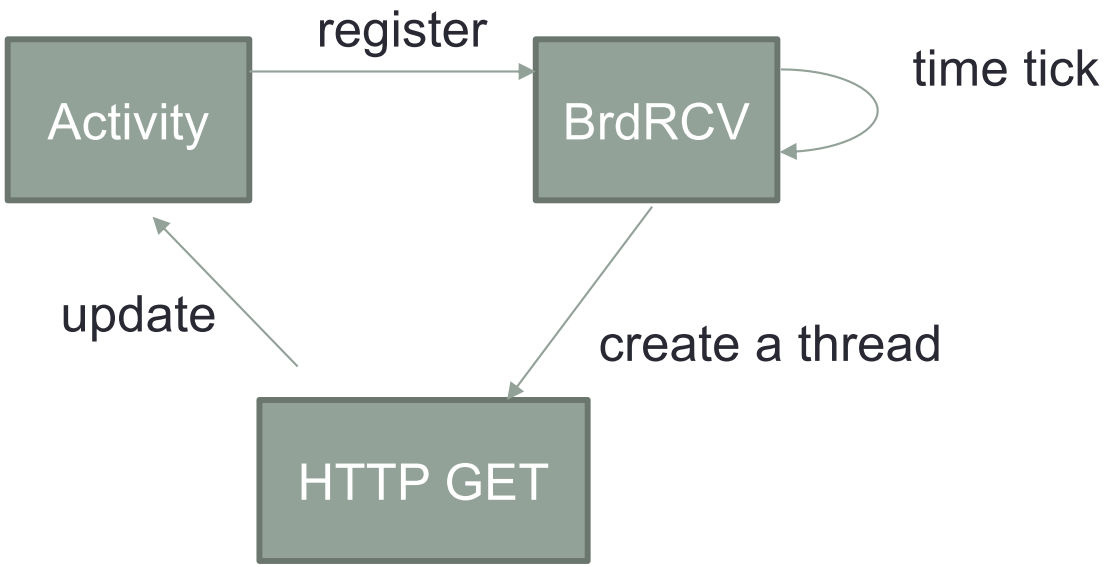
- Register to the TIME_TICK event
- Warning: the registration can only be done dynamically from the code. Also, for security reason it cannot be generated (sendBroadcast(…))



when the time changes a toast is displayed
Useful for example to perform polling…

Example

- Poll an API periodically every minute and update the UI
- For example, openwheather



QUESTIONS?