Mobile Application Development

Intent

(Professional Android 4 Application Development chapter 5)

Outline

- Overview of an Intent
- Starting Activities, Sub activities, and Services using implicit and explicit Intents.

Intent

- Intents are used as a message-passing mechanism that works both within your application and between applications. OR
- In android, Intent is a messaging object which is used to request an action from another component.
- In android, intents are mainly used to perform the following things.
  - Explicitly start a particular Service or Activity using its class name
  - Start an Activity or Service to perform an action with (or on) a particular piece of data
  - Broadcast that an event has occurred
- There are two types of intents available in android, those are :
  - Implicit Intents
  - Explicit Intents

Intent

- One of the most common uses for Intents is to start new Activities, either explicitly (by specifying the class to load) or implicitly (by requesting that an action be performed on a piece of data).
- In Implicit case the action does not need to be performed by an Activity within the calling application.

Intent

- 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 assocaited 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**.

Intent

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

`startActivity (intent)`	launches an activity
`sendBroadcast (intent)`	sends an intent to any interested BroadcastReceivers
`startService(intent)` or `bindService(intent, …)`	communicates with a background service.

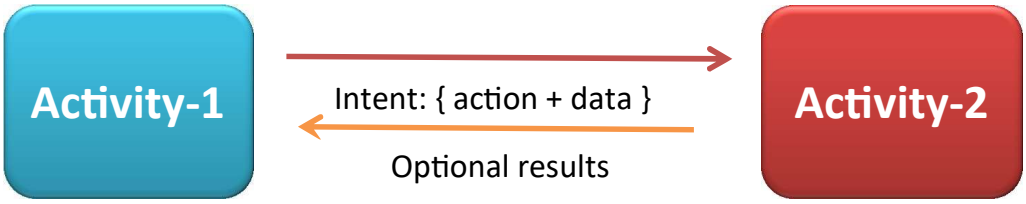
Intent

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.



Intent

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/)

Intent

Initiating an Intent

Typically an intent is called as follows:

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

startActivity (myOtherActivity);
```


Intent

Examples of action/data pairs:

ACTION_DIAL ***tel://5551234*** or ***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

Intent

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.

11

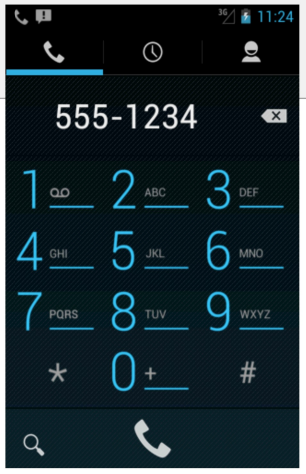
Intent

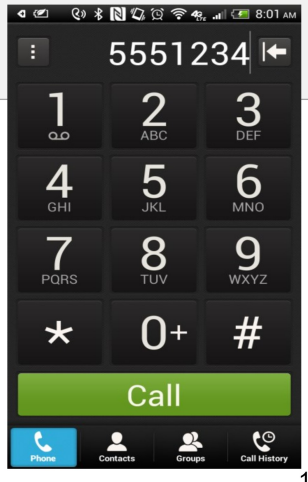
Example: **ACTION_DIAL**

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);
```





12

Intent

- **Implicit Intent**
- When you just have to tell what action you want to perform without worrying which component will perform it, then you can use implicit intent.
- Implicit intents do not name a specific component to perform a particular action, but instead it declares a general action to be performed, which allows any component, even from another app to handle it.
- For example, if you want to show a specific location of the user on a map, you can use an implicit intent to pass the coordinates through the intent and then any other app, which is capable of showing the coordinates on a map will accept that intent.

Intent

- Create an Implicit Intent
- You need to make an Intent object. The constructor of the Implicit Intent's object needs a type of action you want to perform.
- An action is a string that specifies the generic action to be performed. The action largely determines how the rest of the intent is structured, particularly the information that is contained as data and extras in the intent object.

Intent

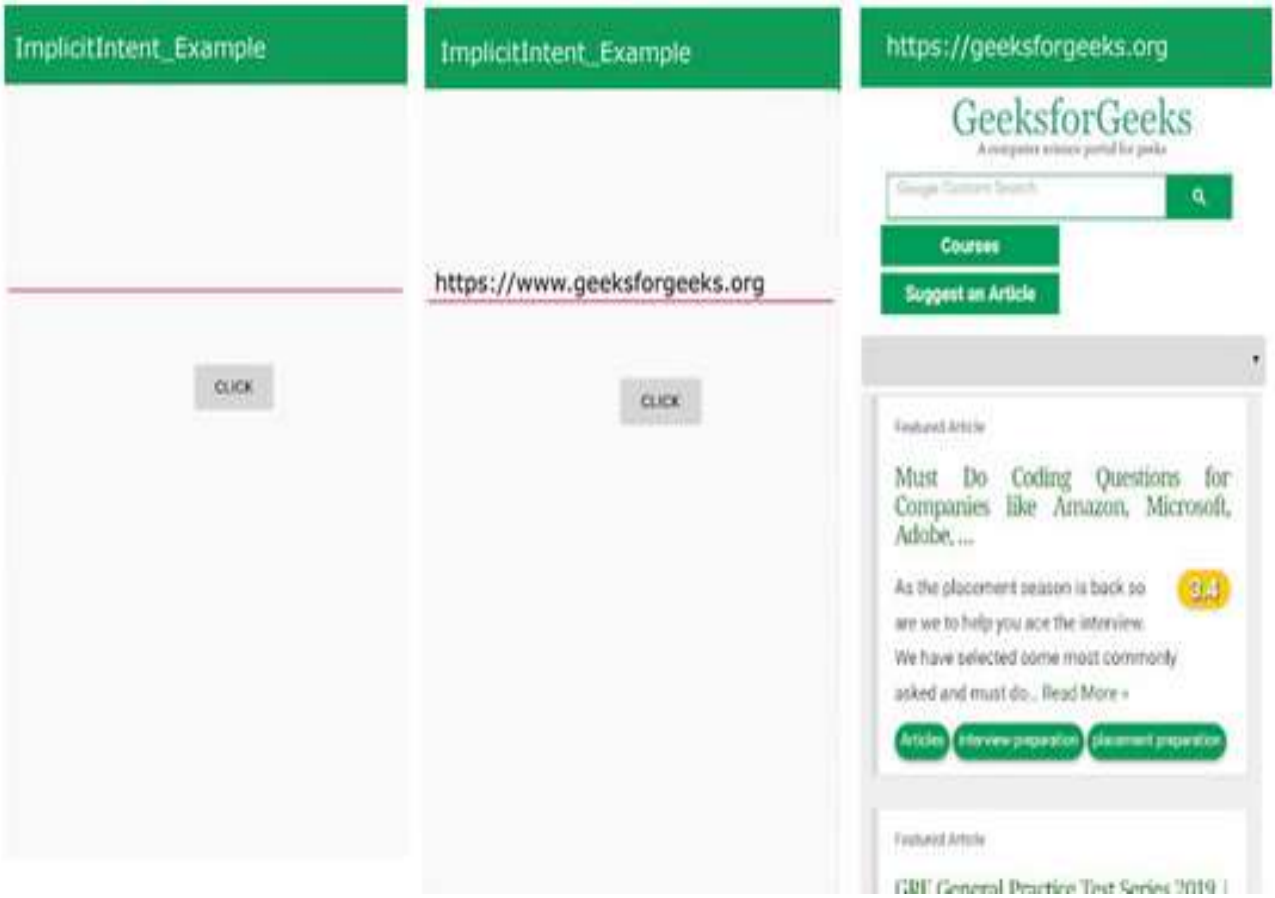
- For example,

ACTION_VIEW: This action is used when you have some information that an activity can show to the user, such as a photo to view in a Gallery app, or an address to view in a Map app.

ACTION_SEND: This action is used when you have some data that the user can share through another app, such as an Email app or some Social Networking app.

Intent

- **Intent i = new Intent(Intent.ACTION_VIEW);**
- You need to provide some data for the action to be performed. Data is typically expressed as a URI(Uniform Resource Identifier) which provides data to the other app so that any other app which is capable of handling the URI data can perform the desired action.
- For example, if you want to open a website through your app, you can pass the Uri data using setData() method as follows:
- **i.setData(Uri.parse("**http://www.google.co.in**"));**
- Call startActivity() method in the end with the intent object as the parameter.
- **startActivity(i);**



Intent

- **Explicit Intent:**
- When you explicitly define which Android component should be opened on some user action, then you use explicit intents.
- You generally use an explicit intent to start a new component in your own app, because you know which exact activity or service you want to start.
- For example, you can start a new activity in response to a user action or start a service to download a file in the background.

Intent

- **Explicit Intent:**
- Create an Explicit Intent
- You need to make an Intent object. The constructor of the Explicit Intent's object needs two parameters as follows:
- Context c: This represents the object of the Activity from where you are calling the intent.
- Java file name: This represents the name of the java file of the Activity you want to open.

Note: You need to mention the java file name with .class extension

Intent i = new Intent(this, MyJavaFile.class);

Intent

- **Explicit Intent:**
- Call startActivity() method and pass the intent's object as the parameter. This method navigates to the java file mentioned in the Intent's object.
- **startActivity(i);**
- If you need to pass some information or data to the new Activity you are calling, you can do this by calling putExtra() method before the startActivity() method. This method accepts key-value pair as its parameter.

i.putExtra("key1", "I am value1");

i.putExtra("key2", "I am value2");

startActivity(i);

Intent

- **Starting and Retrieving result data from a sub-activity**
- An activity can be closed via the back button on the phone. In this case the finish() method is performed. If the activity was started with the startActivity(Intent) method call, the caller requires no result or feedback from the activity which now is closed.
- If you start the activity with the startActivityForResult() method call, you expect feedback from the sub-activity. Once the sub-activity ends, the onActivityResult() method on the sub-activity is called and you can perform actions based on the result.
- In the startActivityForResult() method call you can specify a result code to determine which activity you started. This result code is returned to you.

Intent

Intent resolution by the Android system

One activity is started

Intent

Intent + resultCode provided by called activity

onActivityResult(requestCode, resultCode, intent)

RequestCode provided by Android to identify which activity type was started

Intent

- The sub-activity uses the finish() method to create a new intent and to put data into it. It also sets a result via the setResult() method call.
- The following example code demonstrates how to trigger an intent with the startActivityForResult() method.

```
public void onClick(View view) {
   Intent i = new Intent(this, ActivityTwo.class);
   i.putExtra("Value1", "This value one for ActivityTwo ");
   i.putExtra("Value2", "This value two ActivityTwo");
   // set the request code to any code you like,
   // you can identify the callback via this code
   startActivityForResult(i, REQUEST_CODE);
}
```


Intent

- If you use the startActivityForResult() method, then the started activity is called a sub-activity.
- If the sub-activity is finished, it can send data back to its caller via an Intent. This is done in the finish() method.

```
@Override
public void finish() {
   // Prepare data intent
   Intent data = new Intent();
   data.putExtra("returnKey1", "Swinging on a star. ");
   data.putExtra("returnKey2", "You could be better then you are. ");
   // Activity finished ok, return the data
   setResult(RESULT_OK, data);
   super.finish();
}
```

Intent

- Once the sub-activity finishes, the onActivityResult() method in the calling activity is called.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
   if (resultCode == RESULT_OK && requestCode == REQUEST_CODE) {
      if (data.hasExtra("returnKey1")) {
         Toast.makeText(this, data.getExtras().getString("returnKey1"),
            Toast.LENGTH_SHORT).show();
      }
   }
}
```