Agenda 455/555 9.17.18 Mon.

* Chapter read: 10,39,40 (pgs. 90, 391,395)
* Intents
* Logs
* File IO, Access Privileges! For file processing, aka photos, data, etc.
* Demo on intents
* Lab 2 breakdown / Demo on the seekbar

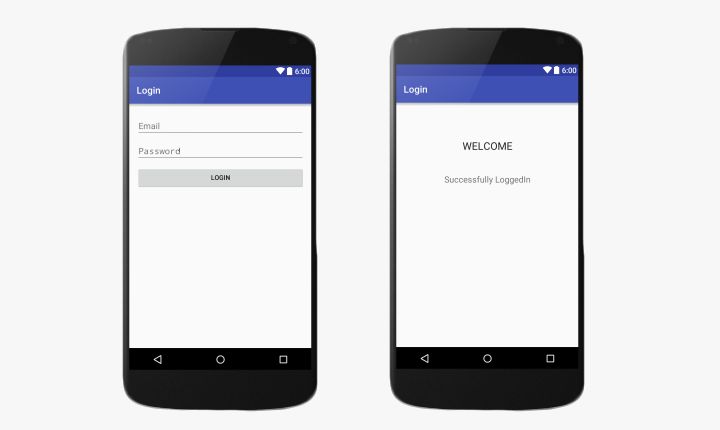
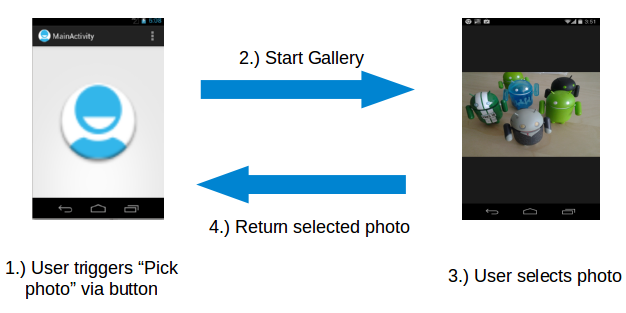
* **Intents**

Android application components can connect to other Android applications. This connection is based on a task description represented by an Intent object.

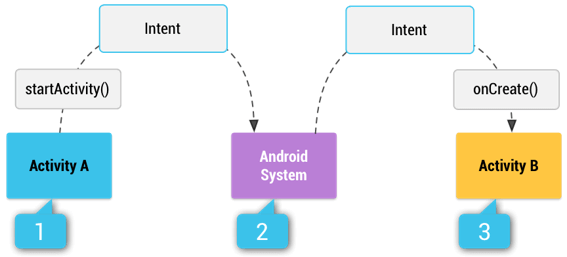
*Intents* are asynchronous messages which allow application components to request functionality from other Android components. Intents allow you to interact with components from the same applications as well as with components contributed by other applications. For example, an activity can start an external activity for taking a picture.

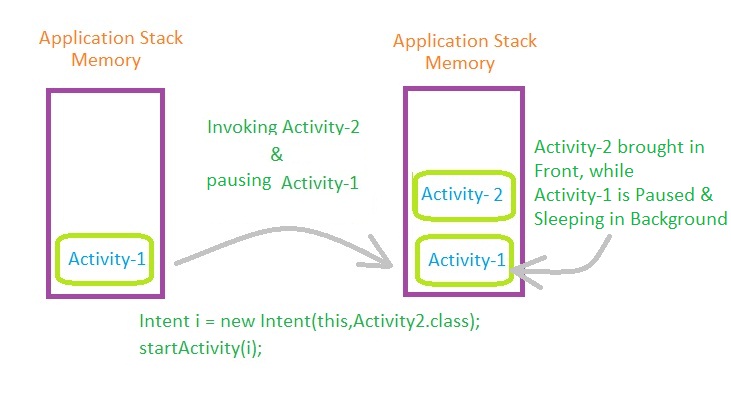
Intents are objects of the **android.content.Intent** type. Your code can send them to the Android system defining the components you are targeting. For example, via the startActivity() method you can define that the intent should be used to start an activity.

An intent can contain data via a **Bundle**. This data can be used by the receiving component.



Explicit Intent





Explicit vs. Implicit Explained

Android supports explicit and implicit intents. An application can define the target component directly in the intent (explicit intent) or ask the Android system to evaluate registered components based on the intent data(implicit intents).

Explicit intents explicitly define the component which should be called by the Android system, by using the Java class as identifier. Explicit intents are typically used within an application as the classes in an application are controlled by the application developer. The following shows how to create an explicit intent and send it to the Android system to start an activity.

Code logic for intents

**# Start the activity connect to the**

**# specified class**

**Intent i = new Intent(this, ActivityTwo.class);**

**startActivity(i);**

-Single variables passed-

**NEWER API’S!**

**SET DATA IN ONE ACTIVITY**

**Intent i = new Intent(packageContext:LoginActivity1.this, MainActivity.class);  
i.putExtra(name:"key", uName);  
i.putExtra(name:"key2", value:"success!");  
startActivity(i);**

**RETRIEVE DATA IN ANOTHER ACTIVITY**

**Intent intent = getIntent();**

**//get the attached extras from the intent  
//we should use the same key as we used to attach the data.  
String uName = intent.getStringExtra("key");**

**//if you have used any other type of data, you should use the  
//particular getExtra method to extract the data from Intent**

**txtUname.setText("Logged in as " + uName); //show resulting data**

Getting to the Web anyone?

For example, the following tells the Android system to view a webpage. All installed web browsers should be registered to the corresponding intent data via an intent filter.

**Intent i = new Intent(Intent.ACTION\_VIEW, Uri.parse("http://www.iit.edu"));**

**startActivity(i);**

Bundle object passed (send/retrieve all data)

**Bundle extras = getIntent().getExtras();**

**if (extras == null) {**

**return;**

**}**

***// get data via the key***

**String value1 = extras.getString(Intent.SOME\_TEXT);**

**if (value1 != null) {**

***// do something with the data***

**}**

***// This is the callback for the started sub-activities***

**@Override**

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

**if (requestCode == SUB\_ACTIVITY\_CREATE\_USER && resultCode ==**

**Activity.RESULT\_OK) {**

**Bundle extras = data.getExtras();**

**if (extras != null) {**

**String name = extras.getString(User.USER\_NAME);**

**boolean gender = extras.getBoolean(User.USER\_GENDER);**

**user = new User(name, gender);**

**updateUserInterface();**

**}**

**}**

**}**

Check your manifest for added intents!! Which intent will fire first?

**<activity  
 android:name=".LoginActivity1"  
 android:label="@string/title\_activity\_login1"  
 android:theme="@style/AppTheme.NoActionBar">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
<activity android:name=".MainActivity">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
</activity>**

* **File IO**

**try {  
 File file = new File(getFilesDir(), "myFile.txt");  
  
 FileOutputStream fileOutputStream = new FileOutputStream(file);  
  
 OutputStreamWriter outputStreamWriter = new OutputStreamWriter(fileOutputStream);  
  
 BufferedWriter bufferedWriter = new BufferedWriter(outputStreamWriter);  
  
 bufferedWriter.write(uName);  
  
 bufferedWriter.close();  
 }  
 catch(Exception e){}**

* **Lab 2** – continues with lab 1 and introduces a Checkbox, SeekBar, listview with an ArrayAdapter and a Viewstub.

View controls

**ViewStub** - A ViewStub is an invisible, zero-sized View that can be used to lazily *inflate*

layout resources at runtime.

<https://developer.android.com/reference/android/view/ViewStub.html>

**Seekbar** - A SeekBar is an extension of ProgressBar that adds a draggable thumb.

<https://developer.android.com/reference/android/widget/SeekBar.html>

**Listview** -  is a view group that displays a list of scrollable items.

<https://developer.android.com/guide/topics/ui/layout/listview.html>

**Arrayadapter** widget - A concrete BaseAdapter that is backed by an array of arbitrary objects.

<https://developer.android.com/reference/android/widget/ArrayAdapter.html>

Simple\_list\_item - It tells the **listview** what layout to use for the individual rows.

<https://developer.android.com/reference/android/R.layout.html>

<http://stackoverflow.com/questions/6079344/what-is-android-r-layout-simple-list-item-1>

Event Listeners/Handling,

SeekBar.[OnSeekBarChangeListener](https://developer.android.com/reference/android/widget/SeekBar.OnSeekBarChangeListener) is a ***callback*** that notifies clients when the progress level has changed for the seek bar control by touch gestures, arrow keys/trackballs, etc. and has three possible abstract methods that need to be overriden for implementation. Two of the methods are onStartTrackingTouch and onProgressChanged. Can you name the third needed overriden method to implement?

abstract void **onProgressChanged(**SeekBar seekBar, int progress, boolean fromUser)

Need: Notification that the progress level has changed.

abstract void **onStartTrackingTouch(**SeekBar seekBar)

Need: Notification that the user has started a touch gesture.

abstract void **onStopTrackingTouch**(SeekBar seekBar)

Need: Notification that the user has finished a touch gesture.