FPT Software

ANDROID TRAINING

LESSON 6

Version 0.1

Agenda

- **Intents**
- **Using Intents to call Activities**
- **Defining Intent Filters**
- **Activity**

Intents

- Intents are asynchronous messages which allow Android components to request functionality from other components of the Android system.
- For example an Activity can send an Intents to the Android system which starts another Activity.

Intents

- Intent Object Structure Is Made Of
  - Component name
  - Action
  - Data
  - Category
  - Extras
  - Flags

Intents

- Component name Field
  - Specifies the name of the component (name of the activity if the component is activity) that should handle the intent
    - >Class name of the target component (for example "com.javapassion.ForwardTargetActivity")
  - Setting component name is optional
    - If it is set, the Intent object is delivered to an instance of the designated class.
    - If it is not set, Android uses other information in the Intent object to locate a suitable target

Intents

- Action Field
  - A string naming the action to be performed
  - The Intent class defines a number of pre-defined action constants, including
    - ACTION_CALL, ACTION_EDIT, ACTION_MAIN, ACTION_SYNC, ACTION_BATTERY_LOW, etc.
  - You can also define your own action strings for activating the components in your application
  - The action largely determines how the rest of the intent is structured - particularly the data and extras fields - much as a method name determines a set of arguments and a return value.

Intents

- Data Field
  - The URI of the data to be acted on and the MIME type of that data.
  - Different actions are paired with different kinds of data specifications.
    - If the action field is ACTION_EDIT, the data field would contain the URI of the document to be displayed for editing.
    - If the action is ACTION_CALL, the data field would be a tel: URI with the number to call.
    - If the action is ACTION_VIEW and the data field is an http: URI, the receiving activity would be called upon to download and display whatever data the URI refers to.

Intents

- Category Field
  - A string containing additional information about the kind of component (activity, service, or broadcast receiver) that should handle the intent.
  - Any number of category descriptions can be placed in an Intent object
  - Android provides a set of predefined categories (We will see them in the following slide)
  - You can define your own categories

Intents

- CATEGORY_BROWSABLE - The target activity can be invoked within the browser to display data referenced by a link — for example, an image or an e-mail message.
- CATEGORY_GADGET - The activity can be embedded inside of another activity that hosts gadgets
- CATEGORY_HOME - This is the home activity, that is the first activity that is displayed when the device boots.
- CATEGORY_LAUNCHER - The activity can be the initial activity of a task and is listed in the top-level application launcher.
- CATEGORY_PREFERENCE - The target activity is a preference panel.
- Many more

Intents

- Extras Field
  - Key-value pairs for additional information that should be delivered to the component handling the intent.
  - Just as some actions are paired with particular kinds of data URIs, some are paired with particular extras.
    - ACTION_TIMEZONE_CHANGED action has a "time-zone" extra that identifies the new time zone
    - ACTION_HEADSET_PLUG action has a "state" extra indicating whether the headset is now plugged in or unplugged , as well as a "name" extra for the type of headset
    - If you were to invent a SHOW_COLOR action, the color value would be set in an extra key-value pair.

Intents

- Flags Field
  - Flags of various sorts.
  - Many instruct the Android system how to launch an activity (for example, which task the activity should belong to) and how to treat it after it's launched (for example, whether it belongs in the list of recent activities).

Intents

- Types of Intents
  - Explicit intents
  - Implicit intents

Explicit Intents

- Explicit Intents explicitly names the component which should be called by the Android system, by using the Java class as identifier.

```
Intent i = new Intent(this, ActivityTwo.class);
i.putExtra("Value1", "This value one for ActivityTwo ");
i.putExtra("Value2", "This value two ActivityTwo");
```

- Explicit Intents are typically used within on application as the classes in an application are controlled by the application developer.

Implicit Intents

- Implicit Intents do not specify the Java class which should be called. They specify the action which should be performed and optionally an URI which should be used for this action.

```
Intent intent = new Intent(Intent.ACTION_VIEW,
    Uri.parse("http://www.google.com"));
```

Data Transfer

- An implicit Intent contains the Action and optional the URI. The receiving component can get this information via the getAction() and getData() methods.
- Explicit and implicit Intents can also contain additional data. This data call be filled by the component which creates the Intent. It can and can get extracted by the component which receives the Intent.

Data Transfer

- Creates the Intent can add data to it via the overloaded putExtra() method
- For example you can trigger all components which have been registered to send some data via the new Intent(Intent.ACTION_SEND). This Intent determines possible receivers via the type. What is send it defined via the putExtra method. You can use any String as key, the following uses the keys which are predefined for the ACTION_SEND intent.

Data Transfer

```
Intent sharingIntent = new Intent(Intent.ACTION_SEND);
sharingIntent.setType("text/plain");
sharingIntent.putExtra(android.content.Intent.EXTRA_TEXT,
    "News for you!");
// createChooser is a convenience method to create
// an Chooser Intent with a Title
startActivity(Intent.createChooser(sharingIntent,"Share this
    using"));
```

Data Transfer

- The component which receives the Intent can use the getIntent().getExtras() method call to get the extra data.

```
Bundle extras = getIntent().getExtras();
if (extras == null) { return; }
 // Get data via the key
String value1 = extras.getString(Intent.EXTRA_TEXT);
if (value1 != null) {
// Do something with the data
 }
```

Calling Activities

- To start an Activity use the method startActivity(Intent). This method is defined on the Context object and available in every Activity object.

Calling Sub-Activities for result data

- If you need some information from the called Activity use 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);
}
```

Calling Sub-Activities for result data

- 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 Intent. This is done in the finish() method.

Calling Sub-Activities for result data

```
@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();
}
```

Calling Sub-Activities for result data

- Once the Sub-Activity finished, the onActivityResult() method in the calling Activity will be 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();
            }
      }
}
```

Intent Filters

- What are Intent Filters?
  - Filters informs the system which implicit intents a component can handle
  - If a component does not have any intent filters, it can receive only explicit intents.
  - A component with filters can receive both explicit and implicit intents.
  - A component has separate filters for each job it can do

Intent Filters

- IntentFilters are typically defined via the AndroidManifest.xml file. For BroadcastReceiver it is also possible to define them in coding. An IntentFilters is defined by its category, action and data filters. It can also contain additional metadata.

Intent Filters

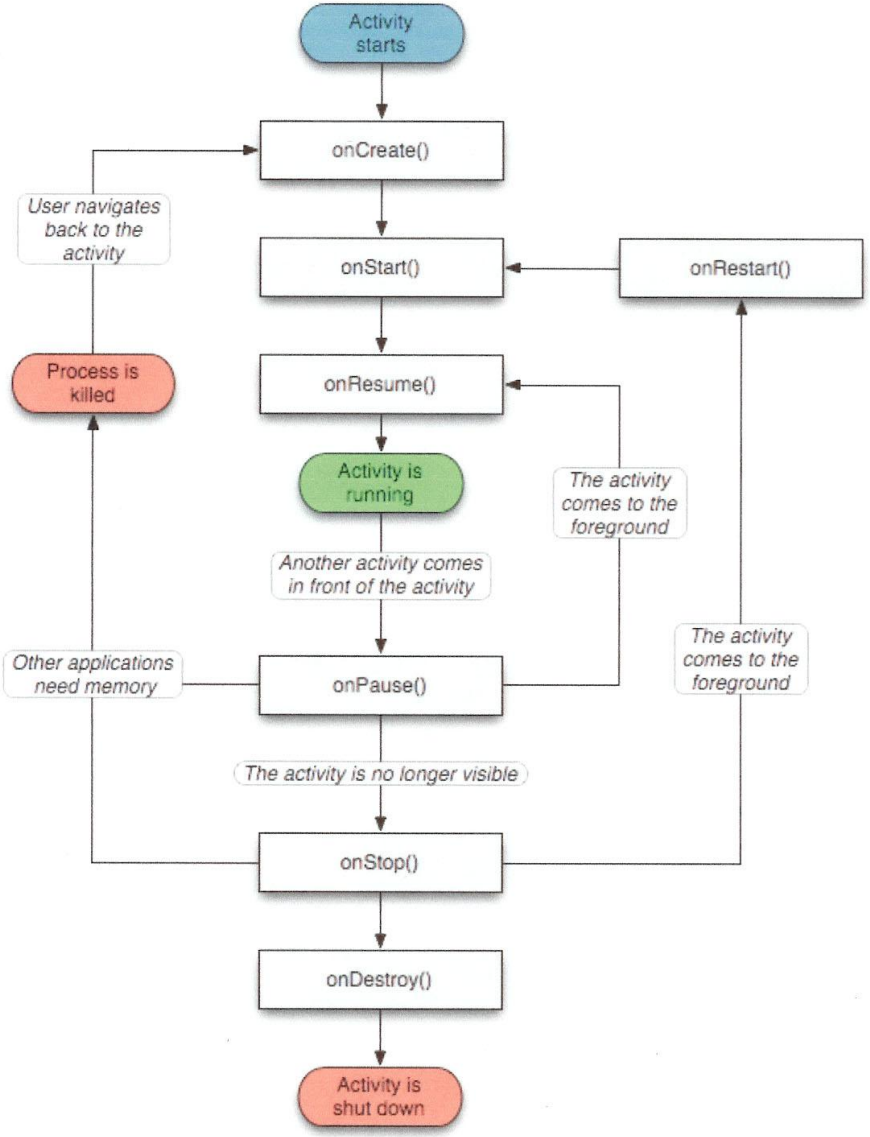
- An implicit Intent object are tested against an intent filters (of target components) in three areas
  - Action
  - Category
  - Data (both URI and data type)
- To be delivered to the target component that owns the filter, it must pass all three tests
- If an intent can pass through the filters of more than one activity or service, the user may be asked which component to activate.
- If no target can be found, an exception is raised

Life Cycle

- Each application runs in its own process.
- Each activity of an app is run in the apps process
- Processes are started and stopped as needed to run an apps components.
- Processes may be killed to reclaim needed resources.
- Killed apps may be restored to their last state when requested by the user

Activity

- Most management of the life cycle is done automatically by the system via the activity stack.
- The activity class has the following method callbacks to help you manage the app:
  - onCreate()
  - onStart()
  - onResume()
  - onPause()
  - onStop()
  - onRestart()
  - onDestroy()


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

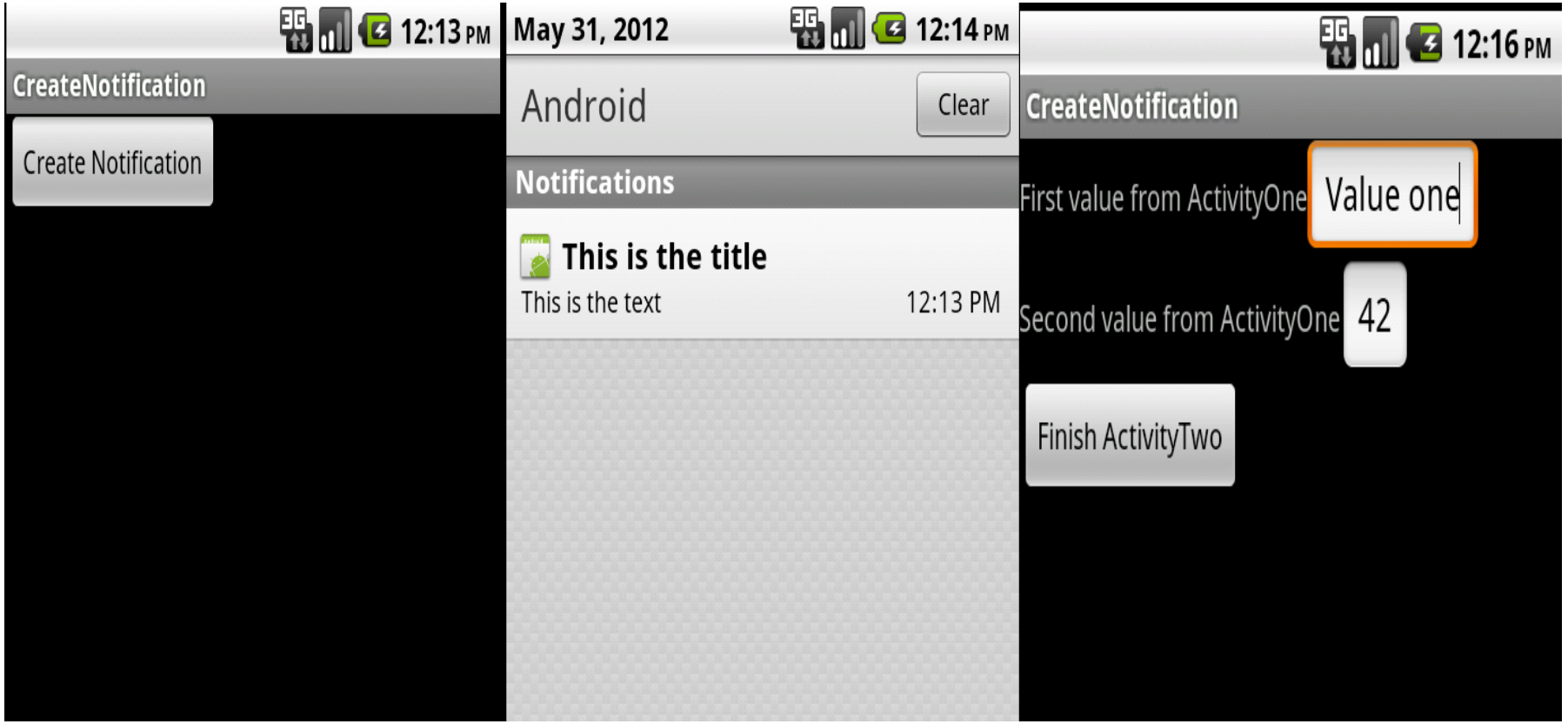
Activity

- There are three "phases" of the activity life cycle: Start- Up, Normal Execution and Shutdown.
  - In the Start - Up phase, the system calls onCreate(), onStart/ onRestart() , possibly onRestoreInstanceState() , and onResume().
  - During the Normal Execution phase, it is common for onSaveInstanceState() and onPause() to be called.
  - In the Shutdown phase, onStop() and onDestroy() may be called.

Homework

- Create Notification
- Click on Notification to start active with extra data

Homework



Thank you!