Mobile Programming

[week 04]

[Event Handling]

Hilmy A.T

hilmi.tawakal@gmail.com

Introduction

- Events are a useful way to collect data about a user's interaction with interactive components of your app, like button presses or screen touch etc.
- The Android framework maintains an event queue into which events are placed as they occur and then each event is removed from the queue on a first-in, first-out (FIFO) basis.
- You can capture these events in your program and take appropriate action as per requirements.

Introduction

Android Event Management Concepts:

- Events Listeners:

  The Event Listener is the object that receives notification when an event happens.

- Event Listeners Registration:

  Event Registration is the process by which an Event Handler gets registered with an Event Listener.

- Event Handlers

  The method that actually handles the event.

Event Listener & Event Handlers

Event Handler	Event Listener & Description
onClick()	**OnClickListener()** This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. You will use onClick() event handler to handle such event.
onLongClick()	**OnLongClickListener()** This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. for one or more seconds. You will use onLongClick() event handler to handle such event.
onFocusChange()	**OnFocusChangeListener()** This is called when the widget looses its focus ie. user goes away from the view item. You will use onFocusChange() event handler to handle such event.
onKey()	**OnFocusChangeListener()** This is called when the user is focused on the item and presses or releases a hardware key on the device. You will use onKey() event handler to handle such event.
onTouch()	**OnTouchListener()** This is called when the user presses the key, releases the key, or any movement gesture on the screen. You will use onTouch() event handler to handle such event.
onMenuItemClick()	**OnMenuItemClickListener()** This is called when the user selects a menu item. You will use onMenuItemClick() event handler to handle such event.

Event Listeners Registration

- Event Registration is the process by which an Event Handler gets registered with an Event Listener.
- The registered Event Handler is called when the Event Listener fires the event.
- There are several ways to registering event listener for any event. 3 top ways are:
  - Using an anonymous inner class
  - Activity class implement the Listener interace
  - Use layout files (*.xml) to specify event handler directly

Using an Anonymous Inner Class

```
// -- register click event with first button ---
sButton.setOnClickListener(new View.OnClickListener() {
   public void onClick(View v) {
       // --- find the text view --
       TextView txtView = (TextView) findViewById(R.id.text_id);
       // -- change text size --
       txtView.setTextSize(14);
   }
});
```

Using an Anonymous Inner Class

- This approach will be useful if each class is applied to a single control only and have advantage to pass arguments to event handler.
- Event handler methods can access private data of Activity. No reference is needed to call to Activity.
- But if you applied the handler to more than one control, you would have to cut and paste the code for the handler and it makes the code harder to maintain

Implement Listener Interface

```
public class MainActivity extends Activity implements OnClickListener {
```

```
// -- register click event with first button ---
sButton.setOnClickListener(this);
```

```
//--- Implement the OnClickListener callback
public void onClick(View v) {
   if(v.getId() == R.id.button_s)
   {
        // --- find the text view --
        TextView txtView = (TextView) findViewById(R.id.text_id);
        // -- change text size --
        txtView.setTextSize(14);
        return;
   }
   if(v.getId() == R.id.button_l)
   {
        // --- find the text view --
        TextView txtView = (TextView) findViewById(R.id.text_id);
        // -- change text size --
        txtView.setTextSize(24);
        return;
   }
}
```

Implement Listener Interface

- This approach is fine if your application has only a single control of that Listener type.
- Otherwise you will have to do further programming to check which control has generated event.
- Second you cannot pass arguments to the Listener so, again, works poorly for multiple controls.

Registration Using Layout File

- Layout File:

```
<Button
android:id="@+id/button_s"
android:layout_height="wrap_content"
android:layout_width="match_parent"
android:text="@string/button_small"
android:onClick="doSmall"/>
```

- Activity Class:

```
//--- Implement the event handler for the first button.
public void doSmall(View v)  {
    // --- find the text view --
    TextView txtView = (TextView) findViewById(R.id.text_id);
    // -- change text size --
    txtView.setTextSize(14);
    return;
}
```

Registration Using Layout File

- The event handler method must have a void return type and take a View as an argument. The method name is arbitrary, and the main class need not implement any particular interface.
- This approach does not allow you to pass arguments to Listener → difficult to know which method is the handler for which control until they look into activity_main.xml file.
- Can not handle any other event except click event.

End

Question?