Day 10: Android Main Concepts: Activities

Today, let’s explore what Activities in Android are –

Activities in Android are what provides screens for the users to interact with. Generally, each screen in Android has a corresponding Activity. An application can have one or more Activities that come together to present the user with various screens that constitute the Application. One among the activities in the application is designated as the “MainLauncher”, typically called MainActivity.

Activities in Xamarin.Android are just classes which inherit from the “Activity” Base class and are decorated with the “Activity” attribute to tell Android more information about it. The Main Launcher Activity is set in this fashion –

|  |
| --- |
| [Activity(Label = "ActivitySamples", MainLauncher = true, Icon = "@drawable/icon")]  public class MainActivity : Activity |

# Activity Life Cycle

Let’s explore all the states that the activity goes through –

* OnCreate
* OnStart
* OnResume
* OnPause
* OnStop
* OnRestart
* OnDestroy

Before we take a more detail look on what happens in each of this states, let’s see the state diagram of Activities Life Cycle –

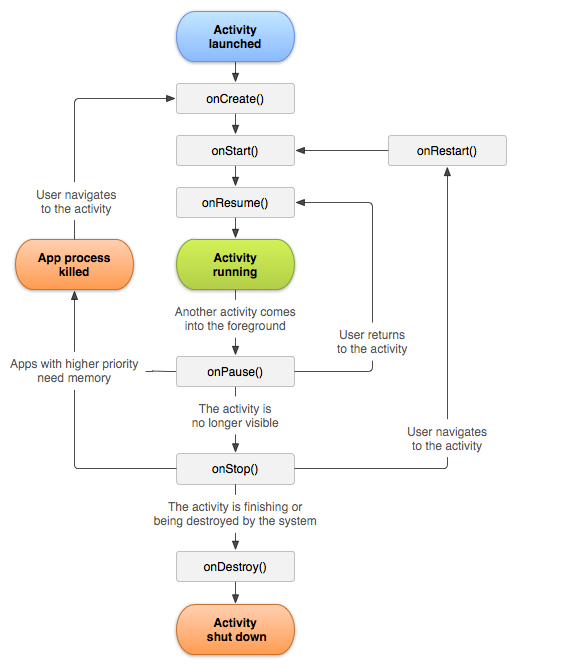


Image Courtesy: <http://developer.android.com/guide/components/activities.html>

The state diagram should give us a pretty good idea of when each of the states of an activity are called. The first state that the Activity will be in is –

## OnCreate

OnCreate is the first method that gets called when an activity is first created. Actions like Creating Views & View Groups, Initialization are generally done in OnCreate Method.

## OnStart

OnStart method follows OnCreate and is generally used to refresh any data in the View and View Groups.

## OnResume

OnResume is one method that is always guaranteed to be called in Android when the Activity becomes Visible. This is when the user will be ready to start using the user interface that goes with the Activity. Another important point about OnResume is that OnResume is responsible for undoing anything that OnPause does.

## OnPause

OnPause gets called when the activity is about to go into background. This method is generally overridden to save any non-persisted information from the user interface.

## OnStop

OnStop is called when the Activity is no longer visible to the user. Activity not being visible to the user can happen in multiple scenarios like another activity is created, or the current activity is destroyed etc.

## OnRestart

OnRestart is called to when bringing back a stopped activity back to life.

## OnDestroy

OnDestroy is called when an activity is destroyed completed. There are scenarios that Android will not call OnDestroy, so don’t count on doing any persistence in OnDestroy.

Let’s take a look at all the state methods that you can override in an Activity real quick in code –

|  |
| --- |
| using Android.App;  using Android.OS;  using Android.Widget;  namespace ActivitySamples  {  [Activity(Label = "ActivitySamples", MainLauncher = true, Icon = "@drawable/icon")]  public class MainActivity : Activity  {  int count = 1;  protected override void OnCreate(Bundle bundle)  {  base.OnCreate(bundle);  // Set our view from the "main" layout resource  SetContentView(Resource.Layout.Main);  // Get our button from the layout resource,  // and attach an event to it  Button button = FindViewById<Button>(Resource.Id.MyButton);  button.Click += delegate { button.Text = string.Format("{0} clicks!", count++); };  }    protected override void OnStart()  {  base.OnStart();  }  protected override void OnResume()  {  base.OnResume();  }  protected override void OnPause()  {  base.OnPause();  }  protected override void OnStop()  {  base.OnStop();  }  protected override void OnRestart()  {  base.OnRestart();  }  protected override void OnDestroy()  {  base.OnDestroy();  }  }  } |

Gist file link: <https://gist.github.com/vkoppaka/081a7d064daef91c72d3>

# Passing data between Activities

Now that we learned all the different states the activity can exist in, let’s learn how to start an activity and how to pass data between two activities.

## Starting an Activity

We looked at this while we are working with Intents, but let’s recap –

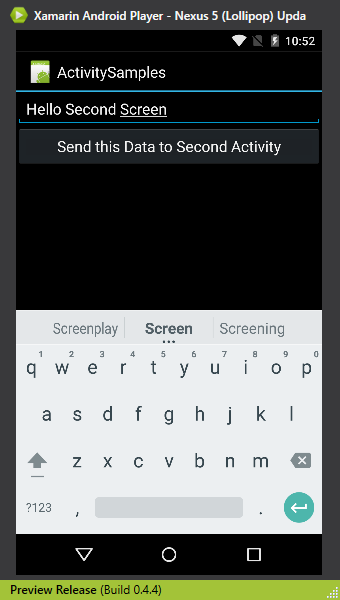
All we need to do to start a new activity in the app is to declare an explicit intent with the activities fully qualified class name and call StartActivity with the intent –

|  |
| --- |
| void button\_Click(object sender, System.EventArgs e)  {  var intent = new Intent(this, typeof (DetailsActivity));  StartActivity(intent);  } |

## Passing data

Now let’s take a look at how we can pass data to the DetailsActivity –

I modified the MainActivity slightly to have a simple EditText that we can use to capture the text we need to send to DetailsActivity.



Here the AXML –

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="vertical"  android:layout\_width="fill\_parent"  android:layout\_height="fill\_parent">  <EditText  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:id="@+id/MessageEditText" />  <Button  android:id="@+id/MyButton"  android:layout\_width="fill\_parent"  android:layout\_height="wrap\_content"  android:text="@string/Hello" />  </LinearLayout> |

Gist file link: <https://gist.github.com/vkoppaka/75025022fc9b75a54e6a>

Here is the Activity code –

|  |
| --- |
| using System;  using Android.App;  using Android.Content;  using Android.OS;  using Android.Widget;  namespace ActivitySamples  {  [Activity(Label = "ActivitySamples", MainLauncher = true, Icon = "@drawable/icon")]  public class MainActivity : Activity  {  protected override void OnCreate(Bundle bundle)  {  base.OnCreate(bundle);  // Set our view from the "main" layout resource  SetContentView(Resource.Layout.Main);  // Get our button from the layout resource,  // and attach an event to it  Button button = FindViewById<Button>(Resource.Id.MyButton);  button.Click += button\_Click;  }  void button\_Click(object sender, EventArgs e)  {  var messageEditText = this.FindViewById<EditText>(Resource.Id.MessageEditText);  var intent = new Intent(this, typeof (DetailsActivity));    intent.PutExtra("TitleToShow", messageEditText.Text);  StartActivity(intent);  }    protected override void OnStart()  {  base.OnStart();  }  protected override void OnResume()  {  base.OnResume();  }  protected override void OnPause()  {  base.OnPause();  }  protected override void OnStop()  {  base.OnStop();  }  protected override void OnRestart()  {  base.OnRestart();  }  protected override void OnDestroy()  {  base.OnDestroy();  }  }  } |

Gist file link: <https://gist.github.com/vkoppaka/f77f78f81880db9beee6>

So as you can see from the highlighted code above, we use intent.PutExtra method to send any extra information to the Second activity, in our sample, I am sending text from the EditText we placed on the Layout.

And to read the passed data in our DetailsActivity –

|  |
| --- |
| using Android.App;  using Android.OS;  using Android.Widget;  namespace ActivitySamples  {  [Activity(Label = "DetailsActivity")]  public class DetailsActivity : Activity  {  protected override void OnCreate(Bundle bundle)  {  base.OnCreate(bundle);  // Create your application here  SetContentView(Resource.Layout.Details);  var titleToShow = Intent.GetStringExtra("TitleToShow");  var titleTextView = this.FindViewById<TextView>(Resource.Id.TitleTextView);  if (string.IsNullOrWhiteSpace(titleToShow))  {  titleToShow = "Didn't receive anything from First screen";  }  titleTextView.Text = titleToShow;  }  }  } |

Gist file link: <https://gist.github.com/vkoppaka/b42b74e55c1f71460106>

We call, Intent.GetStringExtra method in the OnCreate method.

The PutExtra method has bunch of overloads that can take any sought of content, we just used a simple string. Similar there are many Get\*Extra methods that are used to get the data out.

That’s it for today, tomorrow, we learn about Services.

Venkata