Day 17: Fragments in Xamarin.Android

Today we will take a look at a concept that is heavily used in Android world called Fragments.

Fragments in Android are modular and self-contained pieces that have their own lifecycle, layout and behavior. Fragments are generally smaller than Activity and are often used for dynamic UI. A group of Fragments generally make up an Activity (screen) in Android.

Fragments were first introduced in Android HoneyComb (3.0) so your application must be targeting at least API Level 11 to make use of Fragments in Android. Fragments in Xamarin.Android are just classes which inherit from the “Fragment” Base class.

|  |
| --- |
| public class MyFragment : Fragment  {  } |

And Fragments can be added to any layout using the **<Fragment>** AXML element –

|  |
| --- |
| <?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">  <fragment  class="FragmentSamples.MyFragment"  android:id="@+id/titles\_fragment"  android:layout\_weight="1"  android:layout\_width="0px"  android:layout\_height="match\_parent" />  </LinearLayout> |

Gist file link: <https://gist.github.com/vkoppaka/6f249251b4ca38f4ad28>

# Fragment Life Cycle:

Let’s explore all the states that a fragment goes through –

* OnAttach
* OnCreate
* OnCreateView
* OnActivityCreated
* OnStart
* OnPause
* OnStop
* OnDestroy

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

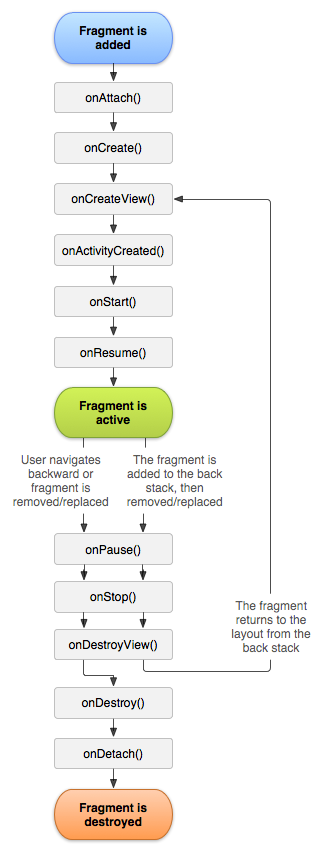


Image courtesy: <http://developer.android.com/guide/components/fragments.html>

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

## OnAttach

This is where a fragment is associated with an activity. Keep in mind that when this method is called, your Fragment, and or, Activity is not fully initialized and you will receive a reference to the activity that its associated with.

## OnCreate

OnCreate gets called when a fragment is first created. Actions like Creating Views & View Groups, Initialization are generally done in OnCreate Method.

## OnCreateView

OnCreateView gets called when Android is ready draw fragment user interface. To draw UI for the fragment we must return a View Component from this method.

## OnActivityCreated

OnActivityCreated gets called when the Activity that the fragment is associated with gets created. This is where we can use our handy-dandy FindViewById<T> methods to find views in Activity.

## OnStart

OnStart method is generally used to refresh any data in the View and View Groups, this method gets called when the Fragment is first visible.

## OnPause

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

## OnStop

OnStop is called when the Fragment is no longer visible to the user and is being stopped.

## OnDestroy

OnDestroy is called when a Fragment is destroyed completely. There are scenario when 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 a Fragment real quick in code –

|  |
| --- |
| using Android.App;  using Android.OS;  using Android.Views;  namespace FragmentSamples  {  public class MyFragment : Fragment  {  public override void OnAttach(Activity activity)  {  base.OnAttach(activity);  }  public override void OnCreate(Bundle savedInstanceState)  {  base.OnCreate(savedInstanceState);  }  public override View OnCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)  {  return base.OnCreateView(inflater, container, savedInstanceState);  }  public override void OnActivityCreated(Bundle savedInstanceState)  {  base.OnActivityCreated(savedInstanceState);  }  public override void OnStart()  {  base.OnStart();  }  public override void OnPause()  {  base.OnPause();  }  public override void OnStop()  {  base.OnStop();  }  public override void OnDestroy()  {  base.OnDestroy();  }  }  } |

Gist file link: <https://gist.github.com/vkoppaka/0f88b07387af023032eb>

# Simple Example

Let’s build a simple example on using Fragments in Xamarin.Android. Over the course of this blog post we will keep using Fragment more which will serve as examples for more complicated scenarios.

In our Main.axml (or any AXML file) add <Fragment> element and set its class attribute to the Fragment we are about to create –

|  |
| --- |
| <?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">  <fragment  class="FragmentSamples.AboutFragment"  android:id="@+id/titles\_fragment"  android:layout\_weight="1"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent" />  </LinearLayout> |

Gist file link: <https://gist.github.com/vkoppaka/76a453988bb79595e94b>

And our About Fragment that the Main Activity links to is really simple, with just overriding the OnCreateView method –

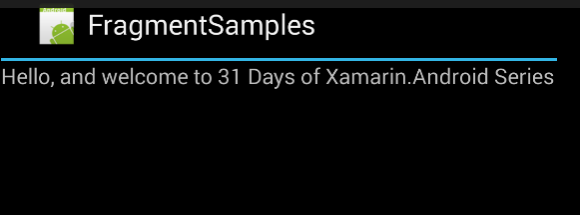
|  |
| --- |
| using Android.App;  using Android.OS;  using Android.Views;  namespace FragmentSamples  {  public class AboutFragment : Fragment  {  public override View OnCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)  {  return inflater.Inflate(Resource.Layout.About, container, false);  }  }  } |

Gist file link: <https://gist.github.com/vkoppaka/b0d7d5eb25405d206132>

If you notice carefully, the only method we override in the Fragment is the OnCreateView method which gets called to draw the user interface of the Fragment. We are just inflating another Layout, About, and returning from this method –

|  |
| --- |
| <?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">  <TextView  android:text="Hello, and welcome to 31 Days of Xamarin.Android Series"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent" />  </LinearLayout> |

And voila, that’s it. If you run your application, you should now see the Fragment in your activity –



That’s it for today, see you all tomorrow.