Day 23: TabLayout

Today, we will explore one of the most common navigational patterns in Mobile Apps: Tabbed Layout. We will look at how we can create a Tabbed Layout in Android using Xamarin.Android using the “TabLayout” widget.

To use TabLayout in Xamarin.Android we first must install the Xamarin Support Design Library using Nuget. The command to install the package is –

|  |
| --- |
| PM> Install-Package Xamarin.Android.Support.Design |

Once the package is installed, we can now move on adding the TabLayout widget. TabLayout widget goes hand in hand with the ViewPager view which is used to flip left and right through Fragments (or even pages).

Let’s see how we can add a TabLayout and ViewPager by looking at the AXML file –

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:local="http://schemas.android.com/apk/res-auto"  android:layout\_width="fill\_parent"  android:layout\_height="fill\_parent">  <include  android:id="@+id/toolbar"  layout="@layout/toolbar" />  <LinearLayout  android:orientation="vertical"  android:layout\_width="fill\_parent"  android:layout\_height="fill\_parent"  android:id="@+id/main\_content"  android:layout\_below="@id/toolbar">  <android.support.design.widget.TabLayout  android:id="@+id/sliding\_tabs"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  local:tabMode="scrollable" />  <android.support.v4.view.ViewPager  android:id="@+id/viewpager"  android:layout\_width="match\_parent"  android:layout\_height="0px"  android:layout\_weight="1"  android:background="@android:color/white" />  </LinearLayout>  </RelativeLayout> |

Gist file link: <https://gist.github.com/vkoppaka/2969a1ca15d67d84400f>

If you notice the snippet carefully, we are calling the “TabLayout” widget from the “android.support.design.widget” namespace and the ViewPager from the “android.support.v4.view” namespace. The TabLayout being in the design library gets its custom attributes from its own namespace hence the tabMode attribute above points to an xml namespace –

|  |
| --- |
| xmlns:local="http://schemas.android.com/apk/res-auto" |

Once the AXML is defined, our next step is to configure the Activity and the Fragments that the ViewPager is going to show. Let’s take a look at our Activity first –

|  |
| --- |
| using Android.App;  using Android.OS;  using Android.Runtime;  using Android.Support.Design.Widget;  using Android.Support.V4.App;  using Android.Support.V4.View;  using Fragment = Android.Support.V4.App.Fragment;  namespace TabSample  {  [Activity(Label = "TabSample", MainLauncher = true, Icon = "@drawable/icon")]  public class MainActivity : FragmentActivity  {  protected override void OnCreate(Bundle bundle)  {  base.OnCreate(bundle);  SetContentView(Resource.Layout.main);  var fragments = new Fragment[]  {  new PeopleFragment(),  new FilmsFragment(),  new StarshipsFragment(),  new VehiclesFragment(),  new SpeciesFragment(),  new PlanetsFragment()  };  var titles = CharSequence.ArrayFromStringArray(new[]  {  "People",  "Films",  "Starships",  "Vehicles",  "Species",  "Planets"  });  var viewPager = FindViewById<ViewPager>(Resource.Id.viewpager);  viewPager.Adapter = new TabsFragmentPagerAdapter(SupportFragmentManager, fragments, titles);  // Give the TabLayout the ViewPager  var tabLayout = FindViewById<TabLayout>(Resource.Id.sliding\_tabs);  tabLayout.SetupWithViewPager(viewPager);  }  }  } |

Gist file link: <https://gist.github.com/vkoppaka/b58bcd9ce089de49d5d9>

The activity, unlike the activity files we have looked so far, inherits from “FragmentActivity”, indicating the activity would deal with ViewPagers and Fragments. The next step in the Activity is to defined list of Fragments that would be shown by the TabLayout. Hence, we create an array of Fragments that the TabLayout shows. Next up is defining the titles of each of the Tab Fragments which is done via the CharSequence.ArrayFromStringArray method.

We now have the Fragment and Titles definitions, so the next step is to tell the ViewPager about them. We do this using the .Adapter property on the ViewPager (prior to this, we need to find the ViewPager using the familiar FindViewById<T> syntax). The Adapter takes 3 arguments: the first one being the SupportFragmentManager from the FragmentActivity and the second and third being our fragments and titles.

Finally, we wrap up the Activity by asking the TabLayout to SetupWithViewPager using the ViewPager we just defined above.

Let us now take a look at one of the Fragments that we defined above. The Fragment used for this sample is very simple, but you can construct as complex UI as you like using all the controls and layouts we learned in this series. Here is the AXML file for Films Layout that binds to FilmsFragment –

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">  <TextView  android:text="Fragment that shows all Starwars Films information"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_margin="30dp" />  </LinearLayout> |

|  |
| --- |
| using Android.OS;  using Android.Support.V4.App;  using Android.Views;  namespace TabSample  {  public class FilmsFragment : Fragment  {  public override View OnCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)  {  return inflater.Inflate(Resource.Layout.Films, container, false);  }  }  } |

Creating these Fragments should feel very familiar to you from our Day 17 blog post.

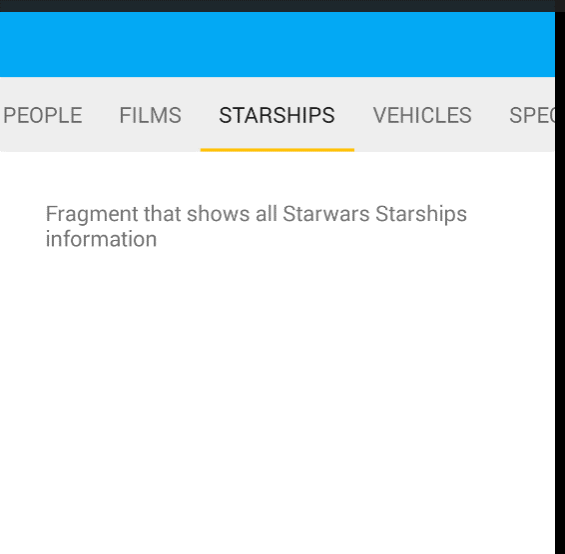
Finally, let us take a look at the code for the Adapter that binds everything together –

|  |
| --- |
| using Android.Support.V4.App;  using Java.Lang;  namespace TabSample  {  public class TabsFragmentPagerAdapter : FragmentPagerAdapter  {  private readonly Fragment[] fragments;  private readonly ICharSequence[] titles;  public TabsFragmentPagerAdapter(FragmentManager fm, Fragment[] fragments, ICharSequence[] titles) : base(fm)  {  this.fragments = fragments;  this.titles = titles;  }  public override int Count  {  get  {  return fragments.Length;  }  }  public override Fragment GetItem(int position)  {  return fragments[position];  }  public override ICharSequence GetPageTitleFormatted(int position)  {  return titles[position];  }  }  } |

Gist file link: <https://gist.github.com/vkoppaka/7d623fc8c5547b22e632>

The FragmentAdapter inherits from FragmentPagerAdapter Base class and we need to override the GetItem method which returns a Fragment at a given position and GetPageTitleFormatted method which returns Tab title at a given position. We should also be overriding the Count property to tell Android on how many Tabs it needs to paint.

If we run the application after all the code is in place, we should see swipeable Tabs using the TabLayout widget.



## Note

If you happen have the application crash after adding tabs, please clean and rebuild the solution and everything should be good to go.

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