Xamarin Android - Apresentando e usando Fragments - II

Neste artigo vou apresentar o conceito de **Fragments** e sua utilização em aplicações **Xamarin Android** no Visual Studio 2015 Community com a linguagem C#.

Na **primeira parte do artigo** eu apresentei os principais conceitos sobre **Fragments** e agora vou mostrar como usar isso na prática em uma aplicação Xamarin Android usando o Visual Studio 2015 Community integrado ao **Xamarin** e a linguagem C#.

Recursos usados:

- Visual Studio Community 2015 ou Xamarin Studio
- Xamarin
- Emulador Android virtual ou físico (veja como emular usando o Vysor)

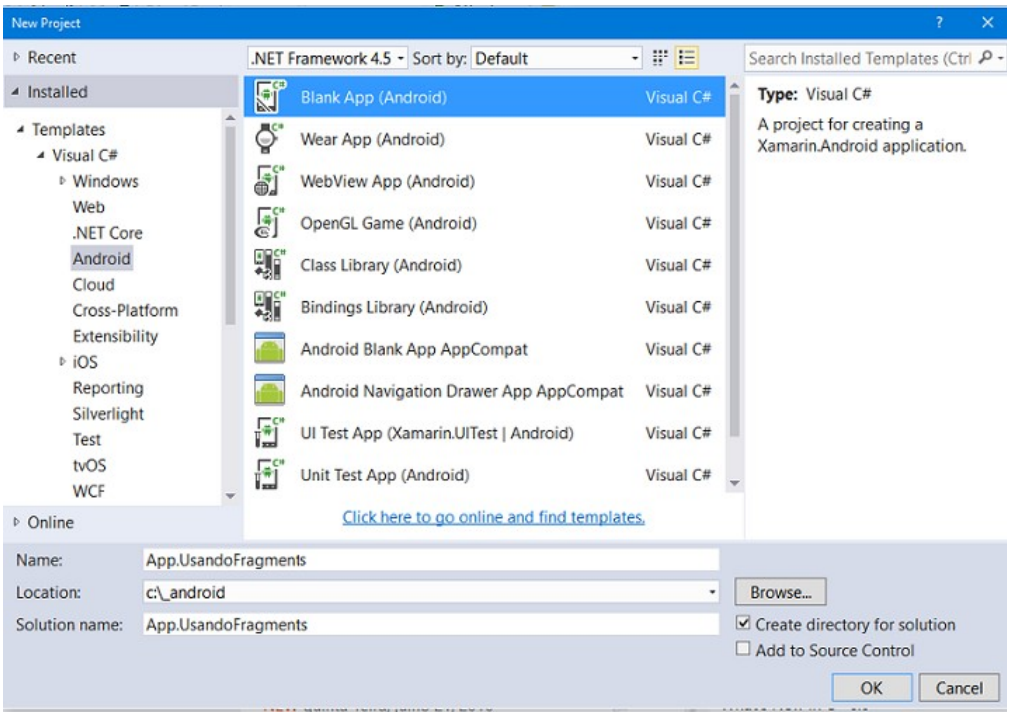
Nota: Baixe e use a versão Community 2015 do VS ela é grátis e é equivalente a versão Professional.

Criando o projeto no Visual Studio 2015 Community

Abra o VS 2015 Community e clique em **New Project;**

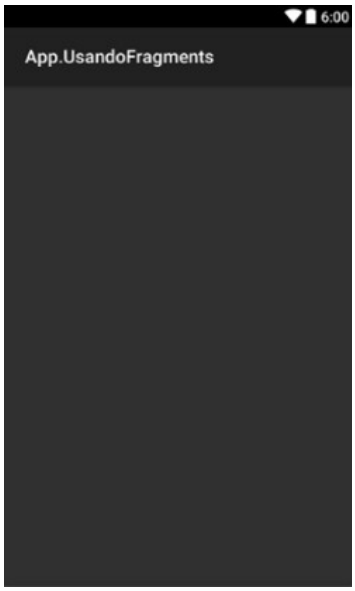
Selecione a linguagem Visual C# e o template **Android -> Blank App(Android)**

Informe o nome **App.UsandoFragments** e clique no botão **OK**;



Será criado o projeto com a View **Main.axml** como view principal e um **button**. Vamos excluir o **button** dessa view e o código que trata o evento **Click** deste button.

Abaixo vemos o resultado desta operação e respectivo código XML:




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:minWidth="25px"
    android:minHeight="25px">
</LinearLayout>
```

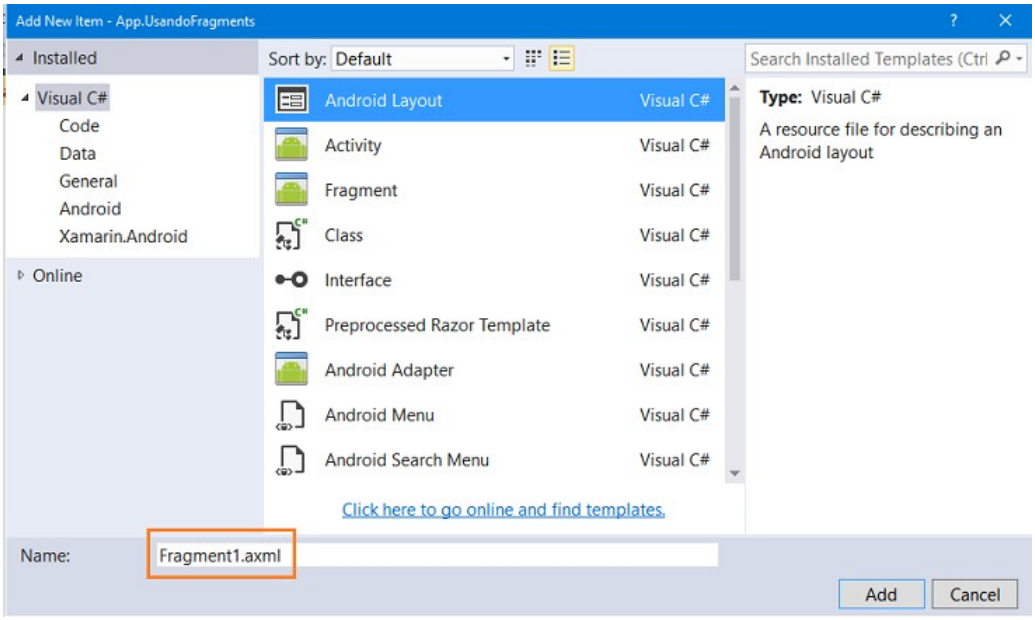
O código da classe **MainActivity.cs** deve ficar assim :

```
using Android.App;
using Android.OS;

namespace App.UsandoFragments
{
    [Activity(Label = "App.UsandoFragments", 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);

        }
    }
}
```

Vamos incluir na pasta **layout** do projeto duas **views**, cada uma representando um **fragment** que iremos exibir na **Activity** principal.

Criando os dois arquivos de Layout que representam os dois Fragments

No menu **Project** clique em **Add New Item** e a seguir selecione o template **Android Layout** e informe o nome **Fragment1.axml** :



Vamos incluir um componente **TextView** nesta **View** ajustando o leiaute conforme a figura abaixo:


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:gravity="center"
   android:background="#323a85"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:minWidth="25px"
   android:minHeight="25px">
   <TextView
      android:gravity="center"
      android:text="Sou o Fragment 1"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/textView1" />
</LinearLayout>
```

Repita o procedimento acima e inclua o layout **Fragment2.axml** ajustando-o conforme mostrado a seguir:


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:gravity="center"
   android:background="#529FD5"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:minWidth="25px"
   android:minHeight="25px">
   <TextView
      android:text="Sou o Fragment 2"
      android:gravity="center"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/textView1" />
</LinearLayout>
```

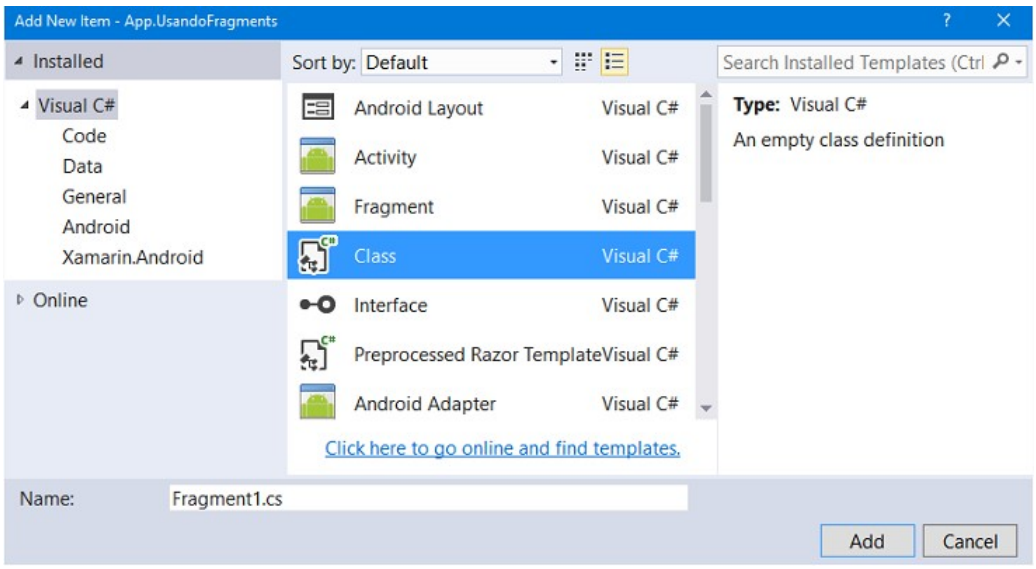
Temos assim dois arquivos de Layout representando cada um a View de um fragment que iremos exibir na Activity principal.

Criando as classes que herdam da classe base Fragment e renderizando as views

Precisamos criar agora duas classes **Fragment1.cs e Fragment2.cs** onde iremos definir o código para renderizar cada view no evento **OnCreateView**().

Assim, para fornecer um layout para um fragmento, você deve implementar o método de retorno de chamada **onCreateView()**, que o sistema Android chama no momento em que o fragmento deve desenhar o layout. A implementação deste método deve retornar uma **View**, que é a raiz do layout do fragmento.

No menu **Project** clique em **Add Class** e a seguir selecione o nome **Fragment1.cs** :



Inclua o código abaixo nesta classe:

```
using Android.App;
using Android.OS;
using Android.Views;

namespace App.UsandoFragments
{
```

```
    public class Fragment1 : Fragment
    {
        public override View OnCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
        {
            View view = inflater.Inflate(Resource.Layout.Fragment1, container, false);
            return view;
        }
    }
}
```

Para retornar um layout de onCreateView(), é possível inflá-lo a partir de um **recurso de layout** definido no XML. Para ajudar a fazer isto, o onCreateView() fornece um objeto **LayoutInflater**.

O parâmetro container passado para onCreateView() é o pai de ViewGroup *(do layout da atividade*) em que o layout do fragmento será inserido.

O parâmetro **savedInstanceState** é um Bundle que fornece dados sobre a instância anterior do fragmento, se o fragmento estiver sendo retomado.

O método **inflate()** usa três argumentos:

- O **ID** do recurso do layout que você quer inflar;
- O ViewGroup que será pai do layout inflado. Passar o container é importante para que o sistema aplique os parâmetros de layout à view raiz do layout inflado, especificado pela view pai em que está ocorrendo;
- Um **booleano** que indica se o layout inflado deve ser anexado à ViewGroup *(o segundo parâmetro)* durante a inflação *(neste caso, isto é falso, pois o sistema já está inserindo o layout inflado no container, retornar como verdadeiro criaria um grupo de vistas redundante no layout final)*.

Este é o modo de criar um fragmento que fornece um layout.

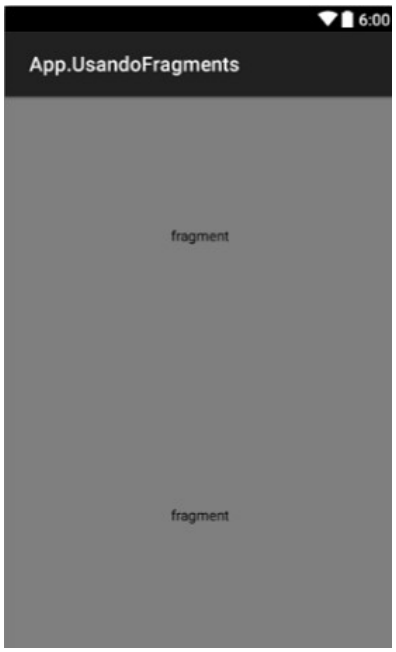
Repita o procedimento acima e crie a classe **Fragment2.cs** com o código abaixo:

```
using Android.App;
using Android.OS;
using Android.Views;

namespace App.UsandoFragments
{
    public class Fragment2 : Fragment
    {
        public override View OnCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
        {
            View view = inflater.Inflate(Resource.Layout.Fragment2, container, false);
            return view;
        }
    }
}
```

A seguir, é preciso adicionar cada fragmento à **Activity** (atividade) e vamos fazer isso no arquivo **Main.axml.**

Abra o arquivo **Main.axml** e no modo **Source** inclua o código XML abaixo *(destacado em azul):*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:minWidth="25px"
   android:minHeight="25px">
<fragment
      class="App.UsandoFragments.Fragment1"
      android:layout_height="match_parent"
      android:layout_width="match_parent"
      android:layout_weight="1"
      android:id="@+id/fragment1" />
   <fragment
      class="App.UsandoFragments.Fragment2"
      android:layout_height="match_parent"
      android:layout_width="match_parent"
      android:layout_weight="1"
      android:id="@+id/fragment2" />
</LinearLayout>
```



Agora é só alegria...

Executando o projeto iremos visualizar na **Activity principal** os dois **Fragments** que criamos no projeto:

Aguarde mais artigos sobre a criação de aplicações com Xamarin.Android.

Pegue o projeto completo aqui : **App.UsandoFragments.zip** (*sem as referências*)

Por isso sinto prazer nas fraquezas, nas injúrias, nas necessidades, nas perseguições, nas angústias por amor de Cristo. Porque quando estou fraco então sou forte.
2 Coríntios 12:10-10

Referências:

- Seção VB .NET do Site Macoratti.net
- Super DVD .NET - A sua porta de entrada na plataforma .NET
- Super DVD Vídeo Aulas - Vídeo Aula sobre VB .NET, ASP .NET e C#
- Super DVD C# - Recursos de aprendizagens e vídeo aulas para C#
- Seção C# do site Macoratti.net
- Seção ASP .NET do site Macoratti .net
- Curso Básico VB .NET - Vídeo Aulas
- Curso C# Básico - Vídeo Aulas
- Curso Fundamentos da Programação Orientada a Objetos com VB .NET NEW
- **Macoratti .net | Facebook**
- **macoratti - YouTube**
- **Jose C Macoratti (@macorati) | Twitter**
- Xamarim - Desenvolvimento Multiplataforma com C# ... - Macoratti.net
- Xamarin - Apresentando Xamarin.Forms - Macoratti.net
- Xamarin.Forms - Olá Mundo - Criando sua primeira ... - Macoratti.net
- Xamarin.Forms - Olá Mundo - Anatomia da aplicação - Macoratti.net
- https://developer.xamarin.com/api/type/Android.App.AlertDialog/

José Carlos Macoratti