

Macoratti.net Xamarin Android - ListView - Usando uma ListActivity (e nada mais...)



Neste artigo vou mostrar como incluir informações em um controle **ListView** usando a classe **ListActivity** em uma aplicação Xamarin Android.

Curso C# Vídeo Aulas
Do básico ao intermediário

Por um preço justo

Em meu artigo [Apresentando e usando o controle ListView](#) eu apresentei o controle ListView e mostrei uma forma de utilizar o controle em aplicações Xamarin Android.

De maneira geral para adicionar linhas em um ListView precisamos incluí-lo em nosso Layout e implementar um **IListAdapter** com os métodos que o ListView chama para se auto preencher.

Existe uma forma bem simples de exibir informações em um **ListView** em aplicações Xamarin Android sem precisar de um arquivo de Layout **.axml**.

O Xamarin Android disponibiliza as classes **ListActivity** e **ListAdapter** que você pode usar sem ter que definir qualquer arquivo de Layout XML ou código.

A classe **ListActivity** cria automaticamente um **ListView** e expõe a propriedade **ListAdapter** que podemos usar para preencher as linhas da view com as informações a serem exibidas. Os adaptadores embutidos utilizam o Resource ID como um parâmetro é usado para cada linha, e assim você não precisa criar o seu layout para usar o ListView nesta abordagem.



Tudo o que você precisa fazer é :

- Criar a classe de código da sua Activity herdando de ListActivity;
- Definir a lista de itens a serem exibidos no ListView;
- Atribuir à propriedade ListAdapter os itens a serem exibidos;

Vejamos a seguir um exemplo prático dessa abordagem.

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.UsandoListActivity** e clique no botão **OK**;

Abra o arquivo **MainActivity.cs** e inclua o código abaixo substituindo o código existente:

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

namespace App.UsandoListActivity
{
    [Activity(Label = "App.UsandoListActivity", MainLauncher = true, Icon = "@drawable/icon")]
    public class MainActivity : ListActivity
    {
        //declarando um array de strings
        string[] items;
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

            //definindo os itens a serem exibidos
            items = new string[] { "Visual Basic", "VB .NET", "Visual C#", "Xamarin", "Android", "JavaScript" };
        }
    }
}
```

```

//atribuindo os valores ao ListAdapter
ListAdapter = new ArrayAdapter<String>(this, Android.Resource.Layout.SimpleListItem1, items);
}

//tratando o evento Click do ListView
protected override void OnListItemClick(ListView l, View v, int position, long id)
{
    var t = items[position];
    Android.Widget.Toast.MakeText(this, t, Android.Widget.ToastLength.Short).Show();
}
}
}

```

Este singelo código fala por si. Nele definimos um array de strings que foram preenchidos com alguns valores.

A seguir atribuímos esses valores à propriedade **ListAdapter** usando um **ArrayAdapter** onde usamos um leiaute de linha existente chamado **SimpleListItem1**, que representa uma única linha de texto, para definir a aparência do ListView. Este layout de item contém um único **TextView** permitindo exibir uma única linha de texto.

Nota: Existem diversos layouts de itens de lista incorporados ao Xamarin.Android como : *SimpleListItem2* , *TwoLineListItem* , *ActivityListItem* , *SimpleListItem2* , *TestListItem* , etc.

```
ArrayAdapter<string> adapter = new ArrayAdapter<string>(this, Android.Resource.Layout.SimpleListItem1, items);
```

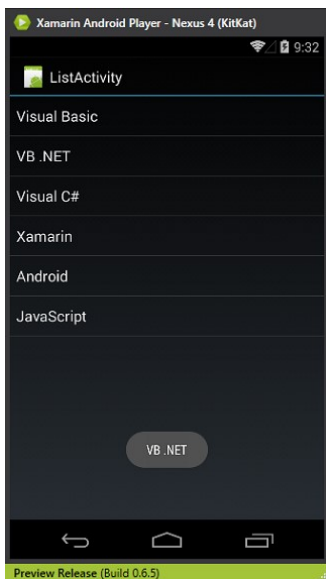
Os argumentos usados são:

- O primeiro argumento é **this** : é o contexto da aplicação;
- O segundo argumento é o leiaute definido no arquivo XML que possui o **TextView** para cada item do array. Estamos usando : **SimpleListItem1**
- O terceiro argumento é o **array** de strings que será usado para preencher o texto da view;

Finalmente realizamos o tratamento do evento **Click** do ListView de forma que ao clicar em um item do controle será exibido um aviso com o nome do item selecionado.

Observe que **não precisamos** usar o método **SetContentView** para definir um arquivo de layout XML vinculada à nossa Activity visto que a classe ListActivity nos fornece uma ListView.

Executando o projeto iremos obter o seguinte resultado:



Aqui vemos o ListView exibindo as informações e o aviso indicando que selecionamos o item **VB .NET**.

Mas e se você quiser fornecer um ListView customizado para exibir os itens ?

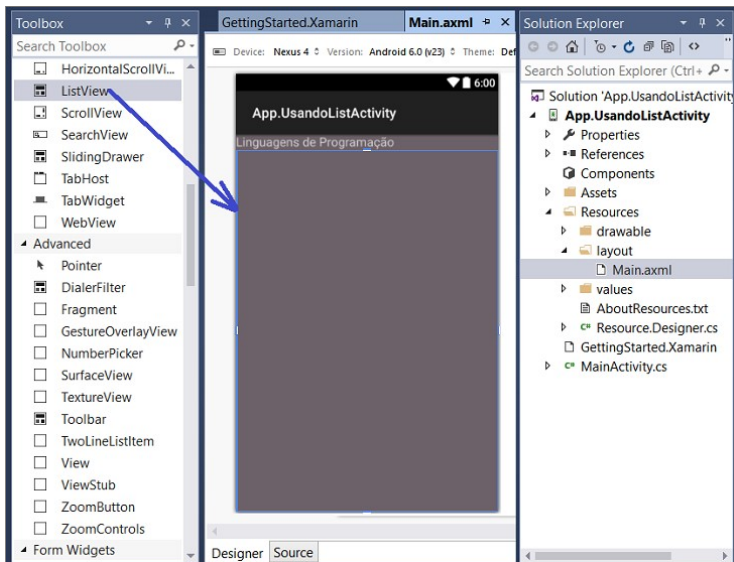
Podemos também fornecer um ListView customizado para exibir os itens.

Para isso basta definir no arquivo de layout **.axml** o controle ListView com a sua customização e, o mais importante, fornecer o nome para o controle como : **@android:id/list**.

Para o exemplo acima, abra o arquivo **Main.axml** na pasta **Resources/layout** e no modo **Designer** inclua um controle **TextView(Medium)** com o texto "*Linguagens de Programação*"; Inclua também um controle **ListView** a partir da **ToolBox** e defina a sua propriedade id como segue:

- **id = @android:id/list**

Abaixo vemos o leiaute no emulador do Xamarin exibindo a tela e ao lado o respectivo código XML gerado :



```
<?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:background="#6d6169"
    android:minWidth="25px"
    android:minHeight="25px">
    <TextView
        android:text="Linguagens de Programação"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView1" />
    <ListView
        android:minWidth="25px"
        android:minHeight="25px"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@android:id/list" />
</LinearLayout>
```

É importante frisar que a propriedade id do controle deve ser definida como : **@android:id/list**.

Após isso altere o código do arquivo **MainActivity.cs** conforme abaixo, incluindo a linha em azul, onde usamos o método **SetContentView()** para definir o arquivo de layout:

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

namespace App.UsandoListActivity
{
    [Activity(Label = "App.UsandoListActivity", MainLauncher = true, Icon = "@drawable/icon")]
    public class MainActivity : ListActivity
    {
        //declarando um array de strings
        string[] items;
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

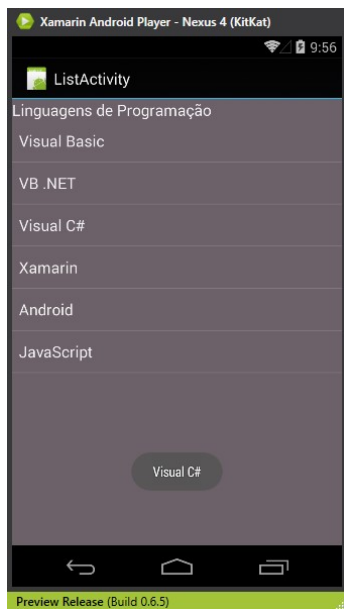
            SetContentView(Resource.Layout.Main)

            //definindo os itens a serem exibidos
            items = new string[] { "Visual Basic", "VB .NET", "Visual C#", "Xamarin", "Android", "JavaScript" };

            //atribuindo os valores ao ListAdapter
            ListAdapter = new ArrayAdapter<String>(this, Android.Resource.Layout.SimpleListItem1, items);
        }

        //tratando o evento Click do ListView
        protected override void OnListItemClick(ListView l, View v, int position, long id)
        {
            var t = items[position];
            Android.Widget.Toast.MakeText(this, t, Android.Widget.ToastLength.Short).Show();
        }
    }
}
```

Executando agora o projeto iremos obter:



Simples assim...

Pegue o projeto completo aqui : [App.UsandoListActivity.zip](#) (sem as referências)

Porque Deus enviou o seu Filho ao mundo, não para que condenasse o mundo, mas para que o mundo fosse salvo por ele. Quem crê nele não é condenado; mas quem não crê já está condenado, porquanto não crê no nome do unigênito Filho de Deus. João 3:17,18

[Veja os Destaques e novidades do SUPER DVD Visual Basic \(sempre atualizado\) : clique e confira !](#)

Quer migrar para o VB .NET ?

- Veja mais sistemas completos para a plataforma .NET no [Super DVD .NET](#) , confira...
- [Curso Básico VB .NET - Vídeo Aulas](#)

Quer aprender C# ??

- Chegou o [Super DVD C#](#) com exclusivo material de suporte e vídeo aulas com curso básico sobre C#.
- [Curso C# Básico - Vídeo Aulas](#)

Quer aprender os conceitos da Programação Orientada a objetos ?

- [Curso Fundamentos da Programação Orientada a Objetos com VB .NET](#) NEW

Quer aprender o gerar relatórios com o ReportViewer no VS 2013 ?

- [Curso - Gerando Relatórios com o ReportViewer no VS 2013 - Vídeo Aulas](#) NEW

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 \(@macoratti\) | Twitter](#)
- [Xamarin - 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/>
- [Xamarin Android - Tratando eventos de forma declarativa](#)

[José Carlos Macoratti](#)