Macoratti.net Xamarin Android - Customizando um ListView

Neste artigo vou mostrar como exibir informações em um controle **ListView** customizado através da implementação da classe base **BaseAdapter**.

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.

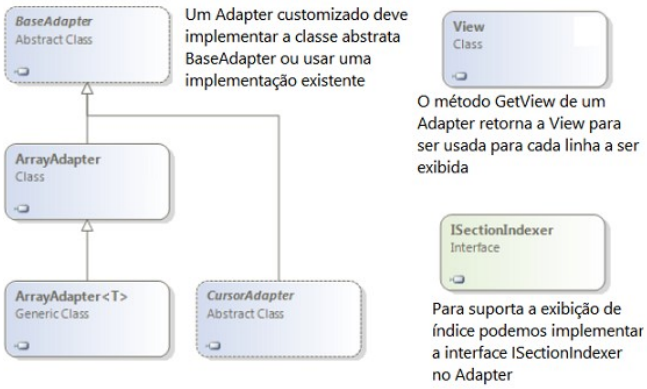
O **ListView** é um componente UI importante de aplicações Android, muito usado para exibir listas curtas de opções de menu a até longas listas de opções. Ele fornece uma maneira simples de apresentar uma lista de rolagem de linhas que podem ser formatadas com um estilo embutido ou personalizados.

O ListView requer um adaptador para alimentá-lo com dados e, 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.

Usar um **ArrayAdapter<string>** é a maneira mais fácil de usar o ListView devido à sua simplicidade, mas ele serve basicamente para exibir apenas uma linha por vez e seus recursos são limitados.

Se você desejar exibir uma coleção de objetos/entidades como uma lista de *Clientes, Produtos, Filmes, etc*. vai querer controlar os dados que deverão ser exibidos ou definir uma apresentação personalizada dos dados, e, o **ArrayAdapter** não fornece recursos para essa tarefa.

Qual a solução ?



Para personalizar o seu **ListView** você terá que implementar a classe abstrata **BaseAdapter** sobrescrevendo os seguintes métodos:

- **Count** - Informa ao controle quantas linhas estão nos dados.
- **GetView** - Retorna uma **View** para cada linha, preenchida com dados.
- **GetItemId** - Retorna um identificador de linha (normalmente o número da linha)
- **this[int]** indexador - Para retornar os dados associados a um número de linha particular

Neste artigo eu vou implementar a customização de um ListView usando um exemplo que exibe uma lista de objetos filmes contendo os dados:

- titulo do filme;
- nome do diretor;
- data de lançamento;

A figura abaixo mostra a aplicação em execução:

Para alcançar esse resultado vamos realizar as seguintes tarefas:

- Definir uma classe de domínio chamada **Filme**;
- Definir um repositório de dados **FilmeRepositorio**;
- Definir um layout que será usado para cada linha a ser exibida no ListView chamado: **Filmes.axml**;
- Definir um Adapter customizado chamado **FilmeAdapter** que implementa a classe abstrata **BaseAdapter** e sobrescreve seus métodos;
- Usar o adapter **FilmeAdapter** na Actitivy principal **Main** , obter os dados e usar a propriedade Adapter da ListView para exibir os itens;

Então vamos ao trabalho...

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

Definindo a classe de domínio e o Repositório

No menu **Project** clique em **Add Class** e informe o nome **Filme.cs**.

A seguir inclua o código abaixo neste arquivo:

```
public class Filme
{
    public int Id { get; set; }
    public string Titulo { get; set; }
    public string Diretor { get; set; }
    public DateTime DataLancamento { get; set; }

    public override string ToString()
    {
        return Titulo + " por " + Diretor;
    }
}
```

Esta classe representa os dados que vamos exibir no ListView.

Crie a classe **FilmesRepositorio.cs** e defina o seu código conforme abaixo:

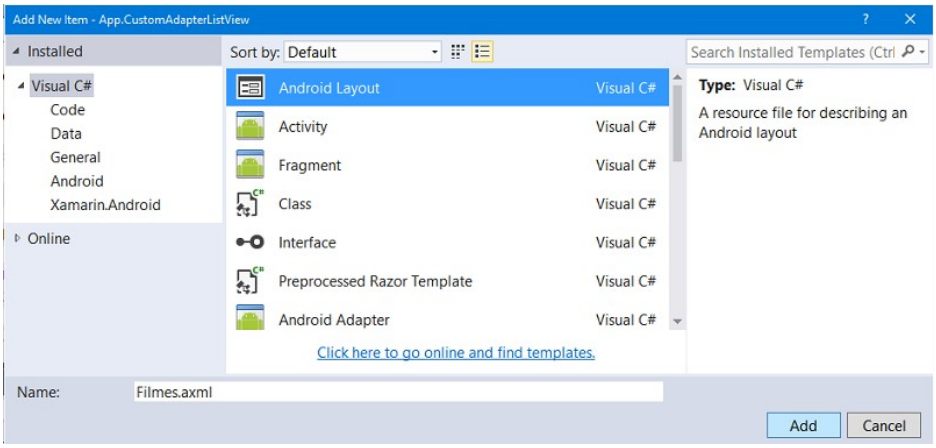
```
public static class FilmesRepositorio
  {
     public static List<Filme> Filmes { get; private set; }

     static FilmesRepositorio()
     {
        Filmes = new List<Filme>();
        for (int i = 0; i < 10; i++)
        {
           AddFilmes();
        }

     }

     private static void AddFilmes()
     {
        Filmes.Add(new Filme
        {
           Id = 1,
           Titulo = "A New Hope",
           Diretor = "George Lucas",
           DataLancamento = new DateTime(1977, 05, 25)
        });

        Filmes.Add(new Filme
        {
           Id = 2,
           Titulo = "The Empire Strikes Back",
           Diretor = "George Lucas",
           DataLancamento = new DateTime(1980, 05, 17)
        });

        Filmes.Add(new Filme
        {
           Id = 3,
           Titulo = "O Reterono de Jedi",
           Diretor = "George Lucas",
           DataLancamento = new DateTime(1983, 05, 25)
        });

        Filmes.Add(new Filme
        {
           Id = 4,
           Titulo = "A ameaça fantasma",
           Diretor = "George Lucas",
           DataLancamento = new DateTime(1999, 05, 19)
        });

        Filmes.Add(new Filme
        {
           Id = 5,
           Titulo = "A vingança dos Sith",
           Diretor = "George Lucas",
           DataLancamento = new DateTime(2005, 05, 19)
        });

        Filmes.Add(new Filme
        {
           Titulo = "Marte",
           Diretor = "J.J. Abrams",
           DataLancamento = new DateTime(2015, 12, 11)
        });
     }
  }
```

Esta classe é usada para fornecer um repositório de dados para nossa aplicação visto que não estamos usando um banco de dados.

Criando o Layout Filmes.axml

Clique com o botão direito sobre a pasta **Resources/layout** e a seguir clique em **Add-> New Item;**

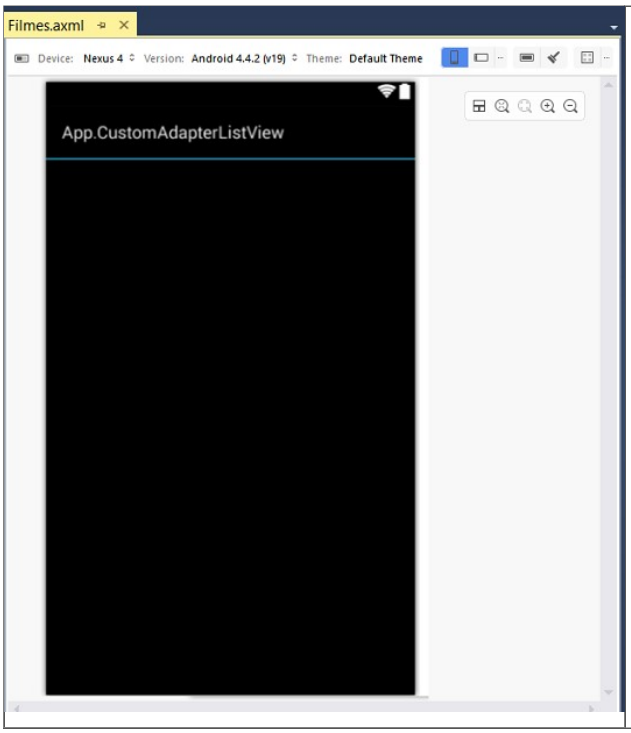
Selecione o template **Layout** e informe o nome **Filmes.axml :**



Inclua, a partir da ToolBox os seguintes controles:

- **3 controle TextView : ids - tituloTextView, diretorTextView e dataLancamentoTextView**

Abaixo vemos o leiaute e o código XML gerado para o layout **Filmes.axml** :




```
<?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">
<TextView
  android:id="@+id/tituloTextView"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:textColor="#000000"
  android:textSize="20dp"
  android:textStyle="bold"
  android:paddingLeft="5dp" />
 <TextView
  android:id="@+id/diretorTextView"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:textColor="#00A14B"
  android:paddingLeft="5dp" />
<TextView
  android:id="@+id/dataLancamentoTextView"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:textColor="#7F3F97"
  android:paddingLeft="5dp" />
</LinearLayout>
```

Este layout será usado para personalizar a exibição dos dados no ListView na implementação da classe **BaseAdapter**.

Criando a classe FilmeAdapter que implementa BaseAdatper

No menu **Project** clique em **Add Class** e informe o nome **FilmeAdapter**.cs e a inclua o código abaixo nesta classe:

```
public class FilmeAdapter : BaseAdapter<Filme>
  {
     private readonly Activity context;
     private readonly List<Filme> filmes;

     public FilmeAdapter(Activity context, List<Filme> filmes)
     {
        this.context = context;
        this.filmes = filmes;
     }

     public override Filme this[int position]
     {
        get
        {
           return filmes[position];
```

```
        }
    }

    public override int Count
    {
        get
        {
            return filmes.Count;
        }
    }

    public override long GetItemId(int position)
    {
        return filmes[position].Id;
    }

    public override View GetView(int position, View convertView, ViewGroup parent)
    {
        var view = convertView ?? context.LayoutInflater.Inflate(Resource.Layout.Filmes, parent, false);

        var txtTitulo = view.FindViewById<TextView>(Resource.Id.tituloTextView);
        var txtDiretor = view.FindViewById<TextView>(Resource.Id.diretorTextView);
        var txtLancamento = view.FindViewById<TextView>(Resource.Id.dataLancamentoTextView);

        txtTitulo.Text = filmes[position].Titulo;
        txtDiretor.Text = "Dirigido por: " + filmes[position].Diretor;
        txtLancamento.Text = "Lançado em : " + filmes[position].DataLancamento.ToShortDateString();

        return view;
    }
}
```

A classe **BaseAdapter** é uma classe abstrata usada para implementação de um **Adapter** que pode ser usado em um ListView, Spinner e Gridview.

A classe **FilmeAdapter** herda de **BaseAdapter** e implementa os métodos :

- **this[int position]**
- **GetItem()**
- **GetItemId()**
- **Count()**
- **GetView(int position, View convertView, ViewGroup parent)**
  - **position** - é o índice do item da view;
  - **convertView** - a view a ser usada;
  - **parent** - o pai da view;

Desses métodos o mais importante é o método **GetView**() que retorna uma view correspondendo aos dados a serem exibidos;

É dentro do método **GetView**() que vamos transformar o arquivo de layout **Filmes.axml** em uma view contendo o leiaute do item da lista usando o método **inflate** da classe **LayoutInflater**.

O código é muito usado :

```
var view = convertView ?? context.LayoutInflater.Inflate(Resource.Layout.Filmes, parent, false);
```

Este método cria uma nova view para cada filme adicionado ao **FilmeAdapter**.

Quando ele for chamado, a **View** é passada, o que normalmente é um objeto reciclado, então temos uma verificação para ver se o objeto é nulo. Se o objeto for nulo, uma **view** é instanciada e configurada com as propriedades desejadas para a apresentação dos itens.

Após isso estamos prontos para usar a view na Activity principal.

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

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

namespace App.CustomAdapterListView
{
    [Activity(Label = "App.CustomAdapterListView", MainLauncher = true, Icon = "@drawable/icon")]
    public class MainActivity : Activity
    {
```

```
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);
            SetContentView(Resource.Layout.Main);

            var filmesListView = FindViewById<ListView>(Resource.Id.filmeslistView);

            filmesListView.FastScrollEnabled = true;

            filmesListView.ItemClick += FilmesListView_ItemClick;

            var filmesAdapter = new FilmeAdapter(this, FilmesRepositorio.Filmes);

            filmesListView.Adapter = filmesAdapter;
        }

        private void FilmesListView_ItemClick(object sender, AdapterView.ItemClickEventArgs e)
        {
            Toast.MakeText(this, FilmesRepositorio.Filmes[e.Position].ToString(), ToastLength.Long).Show();
        }
    }
}
```

No código do arquivo **MainActivity** estamos usando um **adapter** customizado (**FilmeAdapter**) para o ListView que vai atuar como uma fonte de dados para o controle exibindo os itens que foram adicionados no repositório de dados (**FilmesRepositorio**).

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

Executando o projeto iremos obter o seguinte resultado:

Simples assim...

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

Porque todo aquele que faz o mal odeia a luz, e não vem para a luz, para que as suas obras não sejam reprovadas.
Mas quem pratica a verdade vem para a luz, a fim de que as suas obras sejam manifestas, porque são feitas em Deus.
João 3:20,21

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/
- Xamarin Android - Tratando eventos de forma declarativa

José Carlos Macoratti