Macoratti.net Xamarin Android - Apresentando e usando o componente GridView

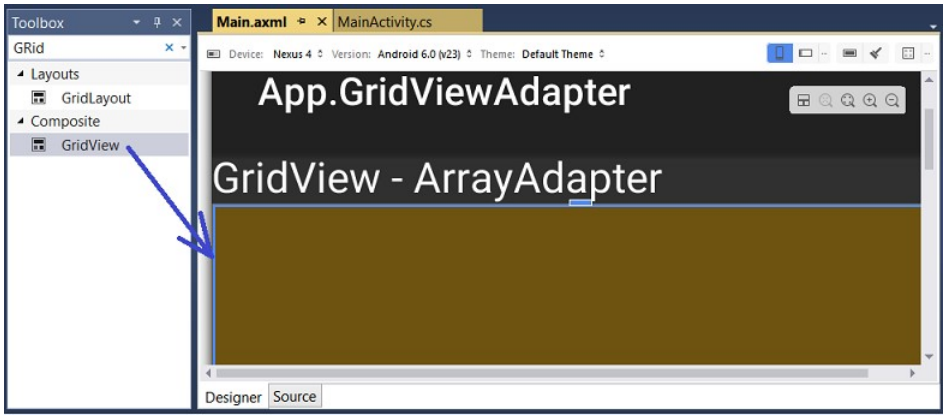
Neste artigo vou apresentar os conceitos básicos sobre o componente **GridView** e sua utilização em aplicações **Android** usando o Visual Studio com Xamarin e a linguagem C#.

O **GridView** é um **ViewGroup** à que exibe os itens em uma grade bidimensional *(linhas e colunas),* rolável. Os itens de grade são inseridos automaticamente para o layout usando um **ListAdapter**.

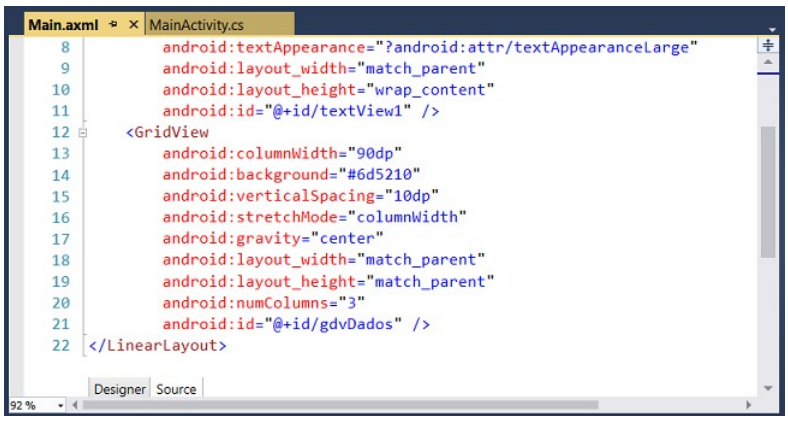
O ListView e GridView são subclasses da classe **AdapterView** e podem ser preenchidos fazendo uma vinculação a **Adapter** (adaptador), que recupera os dados de uma fonte externa e cria uma exibição que representa cada entrada de dados.

Um **Adapter** atual como uma ponte entre componentes de interface do usuário e a fonte de dados que preenche os dados no componente de interface do usuário. O adaptador pode ser usado para fornecer os dados um Spinner, ListView, GridView , etc.

Para incluir um **GridView** em uma view podemos usar a caixa de ferramentas (**ToolBox**) no ambiente do Visual Studio:



Ou podemos também definir o componente no arquivo XML que representa a view :



Os principais atributos do controle GridView são :

Atributo	Descrição
android:id	ID que identifica de forma única o layout
android:columnWidth	Especifica a largura fixa para cada coluna. Os valores podem ser expressos em : px, dp, sp, em, ou mm.
android:gravity	Especifica a gravidade dentro de cada célula. Os valores possíveis são: top, bottom, left, right, center, center_vertical, center_horizontal, etc. (***Nota***: ***android:gravity*** *afeta como os filhos de um componente vão ser alinhados. Não confundir com* ***android:layout_gravity*** *que é como o componente vai se alinhar dentro do* ***parent****)*
android:horizontalSpacing	Define o padrão de espaçamento horizontal entre as colunas. Os valores podem ser expressos em : px, dp, sp, em, ou mm.
android:numColumns	Define quantas colunas vai exibir. Pode ser um valor inteiro, como "100" ou **auto_fit** que significa exibir tantas colunas quanto possível para preencher o espaço disponível.

android:stretchMode	Define como as colunas devem esticar para preencher o espaço vazio disponível, se houver. Os valores possíveis são: • none - O esticamente esta desabilitado; • spacingWidth - O espaçamento entre cada coluna é esticado; • columnWidth - Cada coluna é esticada por igual; • spacingWidthUniform - O espaçamento entre cada coluna é uniformemente esticado;
android:verticalSpacing	Define o padrão de espaçamento vertical entre as linhas. Os valores podem ser expressos em : px, dp, sp, em, ou mm.

Nota: Abaixo uma tabela que indica o significado das unidades de medidas usadas no Android:

Unidade	Significado / Uso
px	Correspondente ao número de pixels da tela / Use em casos específicos
sp	**Scale-independent Pixels** - Relativa à resolução da tela mas considera o tamanho da fonte usada / Usada para definir tamanho de fontes.
dip ou dp	**Density-independent Pixels** - Relativa à resolução da tela. Ex : Se a resolução da tela é de 160 dpi, significa que um dp representa 1 pixel em um total de 160. / Prefira usar dp a px
in	Polegadas - baseada no tamanho físico da tela.
mm	Milímetros - baseada no tamanho físico da tela.
pt	Pontos - 1/72 de uma polegada, baseado no tamanho físico da tela.

Os elementos que serão exibidos no **GridView** podem ser obtidos de diversas formas :

- ArrayList ;
- Array de strings e/ou imagens definido no arquivo de recursos (Resources);
- De uma consulta a um banco de dados ;

De forma geral para preencher um **GridView** com os dados desejados precisamos usar um **ArrayAdapter** que é uma classe concreta que implementa **BaseAdatper** e se baseia em um array de objetos e que funcionada como um adapter.

Um **Adapter (adaptador)** serve como ponte entre os componentes de interface do usuário (UI) e a fonte de dados que preenche os dados nestes componentes. Ele trata e envia os dados para a respectiva view que pode obter esses dados exibindo-os a seguir em diferentes tipos de views como **spinner, list view, grid view, etc.**

O Xamarin Android fornece várias subclasses de adaptador que são úteis para recuperar diferentes tipos de dados e **views** para uma **AdapterView**.

Os adaptadores mais comuns são ArrayAdapter, BaseAdapter, CursorAdapter, SimpleCursorAdapter, SpinnerAdapter e WrapperListAdapter.

Vamos ver na prática como exibir uma lista de itens de uma fonte de dados usando o **GridView** em uma aplicação Android criada no Visual Studio com Xamarin.

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;**

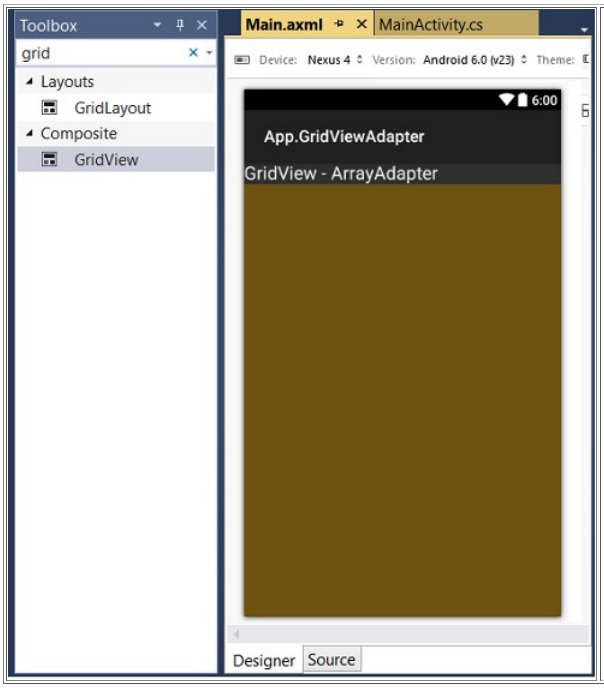
Selecione a linguagem Visual C# e o template **Android -> Blank App(Android)**

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

Abra o arquivo **Main.axml** na pasta **Resources/layout** e no modo **Designer** incluir um novo controle **GridView** a partir da **ToolBox** e definir a seguintes propriedades:

- **id = @+id/gdvDados**

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">
  <GridView
    android:columnWidth="90dp"
    android:background="#6d5210"
    android:verticalSpacing="10dp"
    android:stretchMode="columnWidth"
    android:gravity="center"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:numColumns="3"
    android:id="@+id/gdvDados" />
</LinearLayout>
```

Agora podemos iniciar a implementação do código para exibir itens no **GridView** no arquivo **MainActivity.cs**.

Vamos começar definindo os namespaces usados no projeto:

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

A seguir a declaração da Activity como sendo a principal e a definição do ícone da aplicação:

```
[Activity(Label = "App.GridViewAdaper", MainLauncher = true, Icon = "@drawable/icon")]
```

Antes de implementar o método **OnCreate**() vamos definir as variáveis que iremos usar :

```
  GridView gv;
  ArrayAdapter adpt;
  ArrayList lista;
```

No método **OnCreate**() vamos incluir o código abaixo:

```
    protected override void OnCreate(Bundle bundle)
    {
        base.OnCreate(bundle);

        // Set our view from the "main" layout resource
        SetContentView(Resource.Layout.Main);

        gv = FindViewById<GridView>(Resource.Id.gdvDados);
        Dados();
        adpt = new ArrayAdapter(this, Android.Resource.Layout.SimpleListItem1, lista);
        gv.Adapter = adpt;

        gv.ItemClick += Gv_ItemClick;

    }
```

Por padrão, o **ArrayAdapter** cria uma view (*TextView*) para cada item do array chamando **toString()** em cada item e coloca o conteúdo em um **TextView**.

No exemplo estamos usando a lista de strings definida em **lista,** e estamos usando 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.*

```
adapter = new ArrayAdapter(this, Android.Resource.Layout.SimpleListItem1, lista);
```

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;

Concluindo usamos a propriedade **Adapter** que retorna o adaptador atualmente em uso nesta **ListView** e exibe os dados na view:

```
gv.Adapter = adpt;
```

O método **Dados**() vai apenas preencher o arraylist com as informações que desejamos exibir:

```
private void Dados()
{
   lista = new ArrayList();
   lista.Add("Maria");
   lista.Add("Pedro");
   lista.Add("João");
   lista.Add("Sandra");
   lista.Add("Carlos");
   lista.Add("Eduardo");
   lista.Add("Andre");
   lista.Add("Marcia");
   lista.Add("Debora");
   lista.Add("Cristina");
   lista.Add("Luiz");
   lista.Add("Antonio");
   lista.Add("Paulo");
   lista.Add("Beatriz");
}
```

Concluindo definindo o evento **ItemClick** do GridView para exibir o item selecionado:

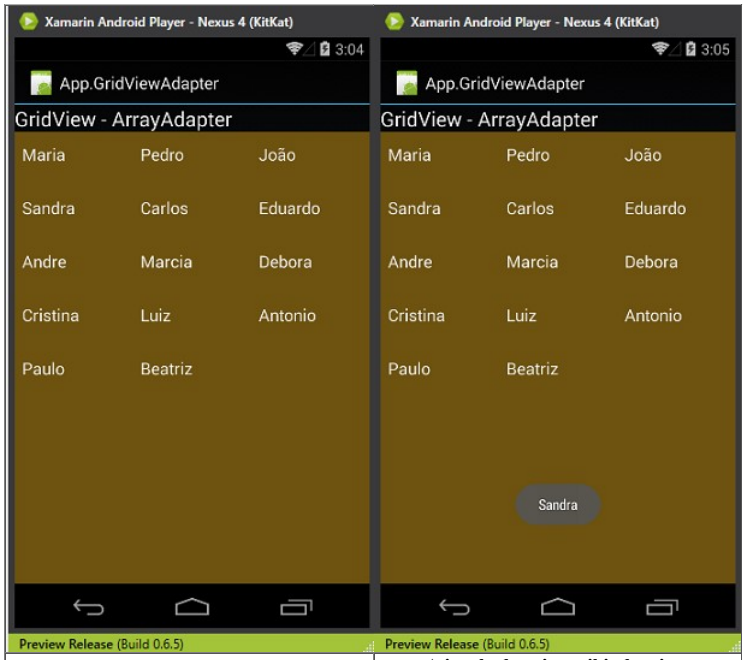
```
gv.ItemClick += Gv_ItemClick;
```

No evento **ItemClick** do **GridView** definimos uma janela de aviso usando a classe **Toast** para exibir qual o item foi selecionado :

```
private void Gv_ItemClick(object sender, AdapterView.ItemClickEventArgs e)
{
   Toast.MakeText(this, lista[e.Position].ToString(), ToastLength.Short).Show();
}
```

Para saber mais sobre diálogos de alerta e avisos veja o artigo : Xamarin Android - Exibindo uma janela de Alerta com AlertDialog.Builder

Executando o projeto iremos obter o seguinte resultado:



O gridview exibindo os itens	A janela de aviso exibindo o item selecionado

No **próximo artigo** vou mostrar como exibir imagens em um controle **GridView** em aplicações Xamarin Android.

Pegue o projeto completo aqui : **App.GridViewAdapter.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

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.Widget.ListView/
- https://developer.xamarin.com/api/property/Android.Widget.ListView.Adapter/

José Carlos Macoratti