



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

Curso C# Vídeo Aulas
Do básico ao intermediário

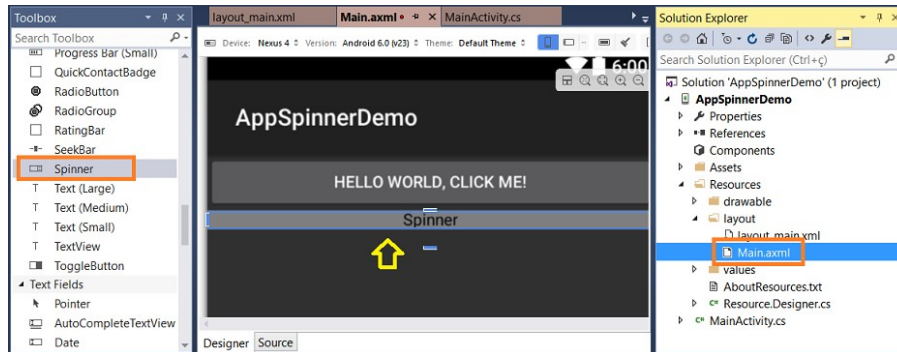
Por um preço justo

Um **Spinner** é um componente visual do Android (*um widget*) que permite selecionar um valor de um conjunto de dados.

No estado padrão ele exibe o valor atualmente selecionado e, quando você interage com o controle ele exibe um menu suspenso com os demais valores e o usuário pode selecionar um novo item. (*Por padrão a classe Spinner espera que o recurso fornecido seja referenciado como um simples TextView*).

Nota: Um **Spinner** se parece com o controle **Combobox** do Windows Forms, **DropDownlist** da ASP .NET ou com a tag **Select** do HTML.

Para incluir um **Spinner** 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 elementos que serão exibidos no **Spinner** 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 Spinner com os dados desejados precisamos usar um **ArrayAdapter** que é uma classe concreta que implementa **BaseAdapter** e se baseia em um array de objetos que funciona como um adaptador.

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 declarada em um arquivo XML** usando o **Spinner** 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.SpinnerSimples2** e clique no botão **OK**;

Será criada uma solução com a seguinte estrutura:



The Solution Explorer shows the project structure for 'App.SpinnerSimples'. The tree view includes: Properties, References, Components, Assets, Resources (drawable, layout, Main.xml), values (Strings.xml, AboutResources.txt), Resource.Designer.cs, and MainActivity.cs.

- **Properties** - Contém o arquivo [AndroidManifest.xml](#) que descreve as funcionalidades e requisitos da sua aplicação Android, e o arquivo [AssemblyInfo.cs](#) contém informação sobre o projeto como número de versão e build.
- **References** - Contém as bibliotecas [Mono.Android](#), [System.Core](#) e todas as bibliotecas usadas no seu projeto;
- **Components** - Contém componentes de terceiros ou desenvolvidos por você usados no seu projeto.

A maioria dos componentes está disponíveis diretamente do **Xamarin Component Store** e são **free** (*não todos*) e prontos para serem usados; (Para incluir um componente clique com o botão direito sobre **Components** e a seguir em [Get More Components](#));

- **Assets e Resources** - Contém arquivos que não são código, como imagens, sons, arquivos XML e qualquer outro recurso que sua aplicação for usar. Os arquivos externos colocados na pasta **Assets** são facilmente acessíveis em tempo de execução através do [Asset Manager](#).

Já os arquivos colocados na pasta **Resources** precisam ser declarados e mantidos em uma lista com os **IDs** dos recursos que você deseja usar em tempo de execução.

De forma geral, todas as imagens, ícones, sons e outros arquivos externos são colocados na pasta **Resources** enquanto que dicionários e arquivos XML são postos na pasta **Assets**;

Na subpasta **layout** temos os arquivos **.xml** que definem as **views** usadas no projeto;

Na subpasta **values** temos o arquivo [Strings.xml](#) onde definimos as strings usadas no projeto;

O arquivo **MainActivity.cs** é um arquivo C# que define a atividade principal da aplicação Android.

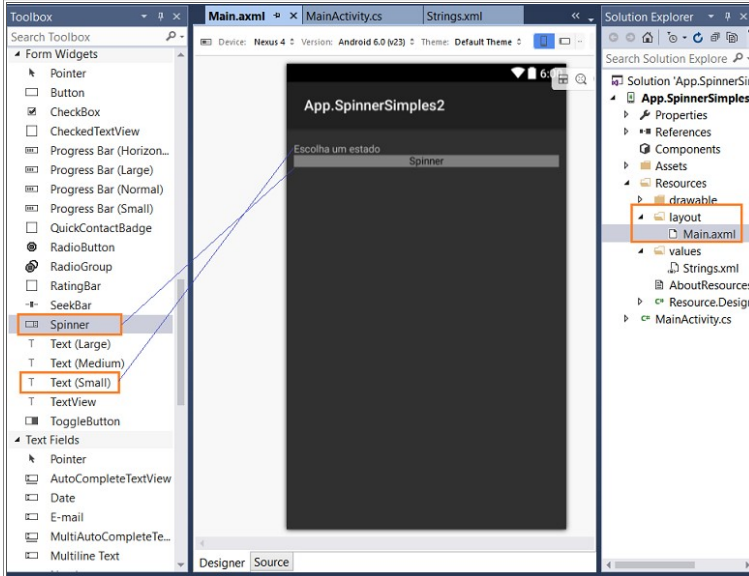
1- Vamos Abrir o arquivo **Main.xml** na pasta **Resources/layout** e no modo **Designer** incluir um novo controle **Spinner** a partir da **ToolBox** e defina a propriedade **id** e **prompt**:

- **@+id/spinDados**
- **android:prompt="@string/estado_prompt"**

Inclua também um componente **Text(Small)** e defina a propriedade **prompt**:

- **android:text="@string/estado_prompt" />**

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



The Designer shows the layout of the app. The Spinner is added to the layout and its properties are set. The Text view is also added and its text is set to the same string resource as the Spinner's prompt.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:padding="10dip"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dip"
        android:text="@string/estado_prompt" />
    <Spinner
        android:id="@+id/spinDados"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:prompt="@string/estado_prompt" />
</LinearLayout>
```

Observe que o atributo **android:prompt** do Spinner e o atributo **android:text** do TextView estão referenciando o mesmo recurso string : **@string/estado_prompt**

Este texto se comporta como um título para a view que será exibida acima do Spinner. Precisamos definir o nome da string **estado_prompt** no arquivo **Strings.xml**.

A seguir abra o arquivo **Strings.xml** que esta na pasta **Resources/values** e altere o seu código conforme abaixo:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="estado_prompt">Escolha um estado</string>
    <string-array name="estados_array">
        <item>São Paulo</item>
        <item>Rio de Janeiro</item>
        <item>Minas Gerais</item>
        <item>Espírito Santo</item>
        <item>Paraná</item>
        <item>Santa Catarina</item>
        <item>Rio Grande do Sul</item>
    </string-array>
</resources>
```

O elemento **<string>** define o título da string referenciado pelo **Spinner** e pelo **TextView** no arquivo de layout.

O elemento **<string_array>** define a lista de strings que serão exibidos na lista da view **Spinner**.

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

Vamos começar definindo os namespaces usados no projeto:

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

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

```
[Activity(Label = "App.SpinnerSimples2", MainLauncher = true, Icon = "@drawable/icon")]
```

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

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

    Spinner spinner = FindViewById<Spinner>(Resource.Id.spinDados);

    try
    {
        var adapter = ArrayAdapter.CreateFromResource(
            this, Resource.Array.estados_array, Android.Resource.Layout.SimpleSpinnerItem);

        adapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDropDownItem);
        spinner.Adapter = adapter;

        spinner.ItemSelected += new EventHandler<AdapterView.ItemSelectedEventArgs>(spinner_ItemSelected);
    }
    catch (Exception ex)
    {
        Log.Debug(GetType().FullName, ex.Message);
    }
}
```

Depois de definir o arquivo **Main.axml** como o conteúdo da nossa view, estamos capturando o widget **Spinner** usando o método **FindViewById(int)**.

O método **CreateFromResource** cria um novo **ArrayAdapter**, e, vincula cada item do array de strings, definidos em **estados_array**, no arquivo **Strings.xml**, ao Spinner.

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 : **SimpleSpinnerItem**
- O terceiro argumento é o **array** de strings que será usado para preencher o texto da view;

O ID **Resource.Array.estados_array** referencia a **string-array** definida acima e o ID **Android.Layout.SimpleSpinnerDropDownItem** referencia o layout padrão para o Spinner.

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

Estamos usando **SetDropDownViewResource()** para definir a aparência de cada item quando o widget for aberto.

Ao final configuramos o **ArrayAdapter (adapter)** para associar todos os itens com o Spinner na sua propriedade Adapter: **spinner.Adapter = adapter;**

Concluindo definindo o evento **ItemSelected** do Spinner para exibir o item selecionado:

```
spinner.ItemSelected += Spinner_ItemSelected;
```

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

```
private void Spinner_ItemSelected(object sender, AdapterView.ItemSelectedEventArgs e)
{
    Spinner spinner = (Spinner)sender;

    string toast = string.Format("Estado selecionado: {0}", spinner.GetItemAtPosition(e.Position));
    Toast.MakeText(this, toast, ToastLength.Long).Show();
}
```

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

Estamos usando a classe **Log** do namespace **Android.Util** para monitorar o ciclo de vida da aplicação durante o debug.

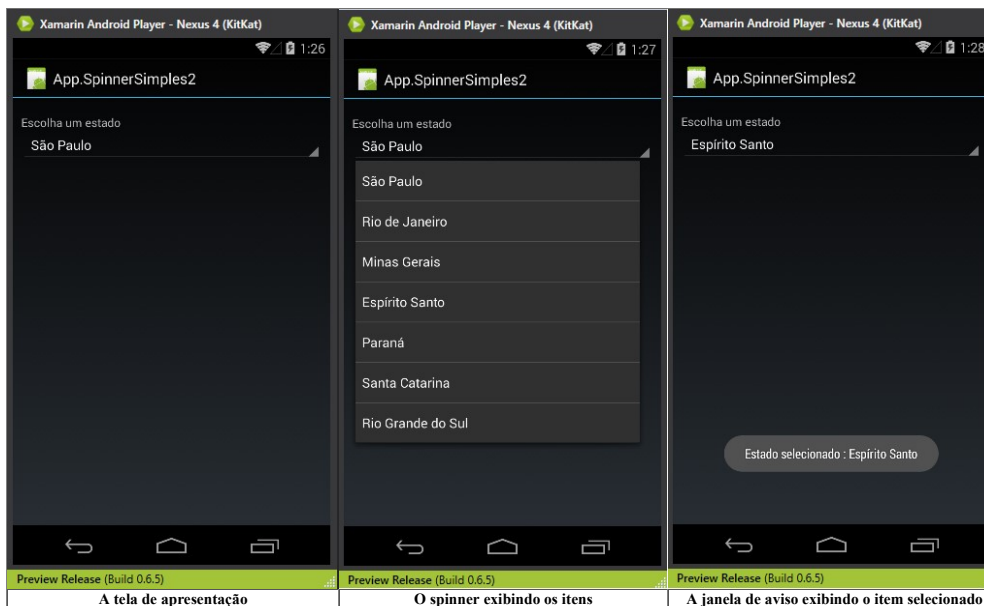
Para criar os log, podemos usar os métodos **Log.v()**, **Log.d()**, **Log.i()**, **Log.w()**, e **Log.e()**.

A ordem em termos de verbosidade do menor para o maior é a seguinte :

- **ERROR** – logs impressos pelo método **Log.Error()**
- **WARN** – logs impressos pelo método **Log.Warn()**
- **INFO** – logs impressos pelo método **Log.Info()**
- **DEBUG** – logs impressos pelo método **Log.Debug()**
- **VERBOSE** – logs impressos pelo método **Log.Verbose()**

Lembre-se de usar este recurso somente em tempo de desenvolvimento. Uma dica é usar uma **tag** constante na classe Log.

Executando o projeto iremos obter o seguinte resultado:



Aguarde em breve mais artigos sobre o componente Spinner em aplicações Xamarin Android.

Pegue o projeto completo aqui : [App.SpinnerSimples2.zip](#) (sem as referências)

(Disse Jesus aos fariseus) Hipócritas, bem profetizou Isaías a vosso respeito, dizendo:

Este povo se aproxima de mim com a sua boca e me honra com os seus lábios, mas o seu coração está longe de mim. Mas, em vão me adoram, ensinando doutrinas que são preceitos dos homens.

Mateus 15:7-9

[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.Widget.ListView/>

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

[José Carlos Macoratti](#)