Macoratti.net Xamarin Android - Apresentando o controle ProgressBar

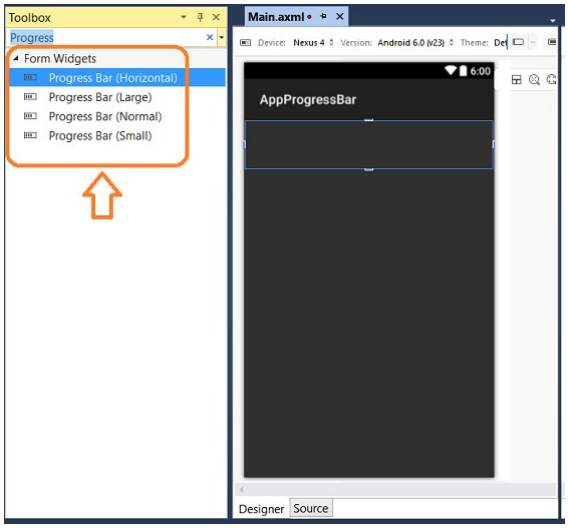
Neste artigo vou apresentar os conceitos básicos sobre o controle **ProgressBar** do Xamarin Android usando o Visual Studio 2015 e a linguagem C#.

O controle **ProgressBar** é usado como um indicador visual do progresso de um operação. Ele exibe uma barra para o usuário representando o quanto da operação foi processada.

Podemos alterar a quantidade de progresso indicada (modificando o tamanho da barra) enquanto ela se move para frente. Existe também um progresso secundário que pode ser exibido em uma barra de progresso que é útil para exibir o progresso intermediário, como o nível de buffer durante a reprodução de um *stream*.

O controle também pode ser usado no modo indeterminado; neste modo a barra de progresso exibe uma animação cíclica sem indicação do progresso. Este modo é usado por aplicações quando o tamanho da tarefa não é conhecida e pode ser um círculo ou uma barra horizontal.

Podemos definir a utilização da **ProgressBar** via código ou no arquivo de layout, e neste caso, a partir da barra de ferramentas (**ToolBox**), temos as seguintes opções que podem ser usadas no arquivo de layout:



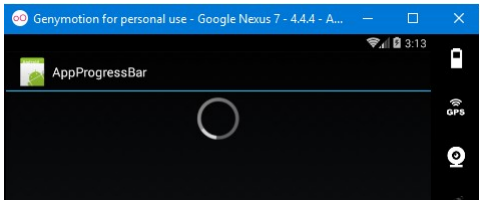
O código gerado no arquivo **.axml** pode ser visto ao lado.

```
<ProgressBar
  style="?android:attr/progressBarStyleLarge"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:id="@+id/progressBar1" />
```

Neste caso temos a propriedade **style** que pode ser preenchida pelos seguintes valores:

```
?android:attr/progressBarStyleHorizontal
?android:attr/textAppearanceLarge
?android:attr/textAppearanceNormal
?android:attr/textAppearanceSmall
```

O primeira corresponde a uma **barra horizontal**, ao passo que os demais definem diferentes tamanhos para a **barra circular**, que pode ser visualizada conforme figura abaixo:



Podemos atribuir o valor da barra de progresso usando o método **Progress**(): `pg.Progress = 25`

Podemos também usar a classe **ProgressDialog** para exibir um indicador de progresso e uma mensagem de texto opcional ou view com uma barra de progresso.

Neste artigo veremos um exemplo básico de utilização do controle com a classe **ProgressDialog**.

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 VS Community 2015

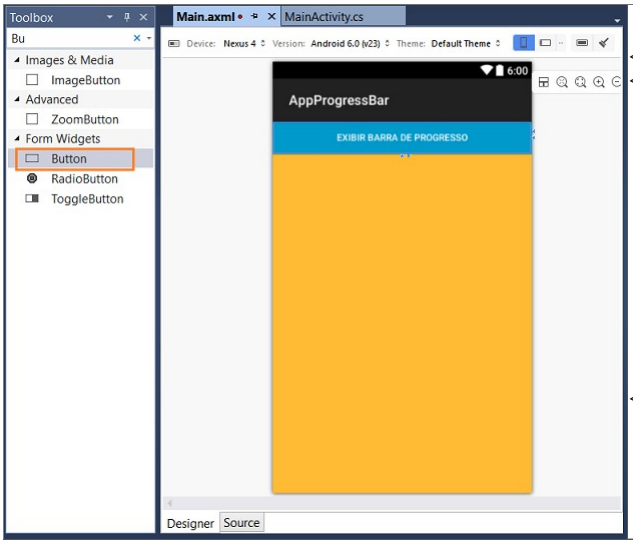
Abra o VS 2015 Community e clique em **New Project;**

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

Informe o nome **AppProgressBar** e clique no botão **OK**;

Abra o arquivo **Main.axml** na pasta **Resources/layout** e no modo **Designer** e a seguir inclua um controle Button a partir da **ToolBox**.

Abaixo vemos o leiaute no emulador do Xamarin 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:minWidth="25px"
  android:minHeight="25px"
  android:background="@android:color/holo_orange_light">
  <Button
    android:text="Exibir Barra de Progresso"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/btnExibirBarra"
    android:background="@android:color/holo_blue_dark" />
</LinearLayout>
```

Definimos um controle Button com o id igual a **btnExibirBarra** e com cor de fundo blue_dark em um layout **LinearLayout** com cor de fundo orange_light.

Vamos definir a barra de progresso no evento Click e definir algumas de suas propriedades e métodos em tempo de execução.

Agora vamos definir o código no arquivo **MainActivity.cs** vinculado a nossa view **Main.axml**.

Abra o arquivo **MainActivity.cs** e altere o código desse arquivo conforme abaixo:

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

namespace AppProgressBar
{
  [Activity(Label = "AppProgressBar", MainLauncher = true, Icon = "@drawable/icon")]
  public class MainActivity : Activity
  {
    Button btnMostraBarra;
    int statusBarra;

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

      // Set our view from the "main" layout resource
      SetContentView(Resource.Layout.Main);
      btnMostraBarra = FindViewById<Button>(Resource.Id.btnExibirBarra);
      btnMostraBarra.Click += BtnMostraBarra_Click;
    }

    private void BtnMostraBarra_Click(object sender, System.EventArgs e)
    {
      ProgressDialog pbar = new ProgressDialog(this);

      pbar.SetCancelable(true);
      pbar.SetMessage("Carregando módulos do sistema...");
      pbar.SetProgressStyle(ProgressDialogStyle.Horizontal);
      pbar.Progress = 0;
      pbar.Max = 100;
      pbar.Show();

      statusBarra = 0;

      new Thread(new ThreadStart(delegate
      {
        while (statusBarra < 100)
        {
          statusBarra += 5;
          pbar.Progress += statusBarra;
          Thread.Sleep(400);
        }
        RunOnUiThread(() => { pbar.SetMessage("Módulos carregados..."); });
        RunOnUiThread(() => { Toast.MakeText(this, "Módulos carregados com sucesso.", ToastLength.Long).Show(); });
      })).Start();

    }
  }
}
```

Vamos entender o código :

1- Definimos uma variável do tipo Button que vai receber a instância do **Button** definido no Layout e uma variável do tipo **int** para controlar o progresso da barra:

```
Button btnMostraBarra;
int statusBarra;
```

2- Fizemoa a vinculação da nossa Activity com o arquivo de layout **Main** e criamos a instância do botão usado no Layout e definimos o evento **Click** do botão :

```
SetContentView(Resource.Layout.Main);
btnMostraBarra = FindViewById<Button>(Resource.Id.btnExibirBarra);
btnMostraBarra.Click += BtnMostraBarra_Click;
```

3- No evento **Click** do botão :

a - Criamos uma instância da classe **ProgressDialog**: **ProgressDialog pbar = new ProgressDialog(this);**

b - Definimos os valores para a barra de progresso : A mensagem , o estilo da barra, o valor inicial e o valor máximo, exibimos a barra e definimos o valor da variável **statusBarra** igual a zero:

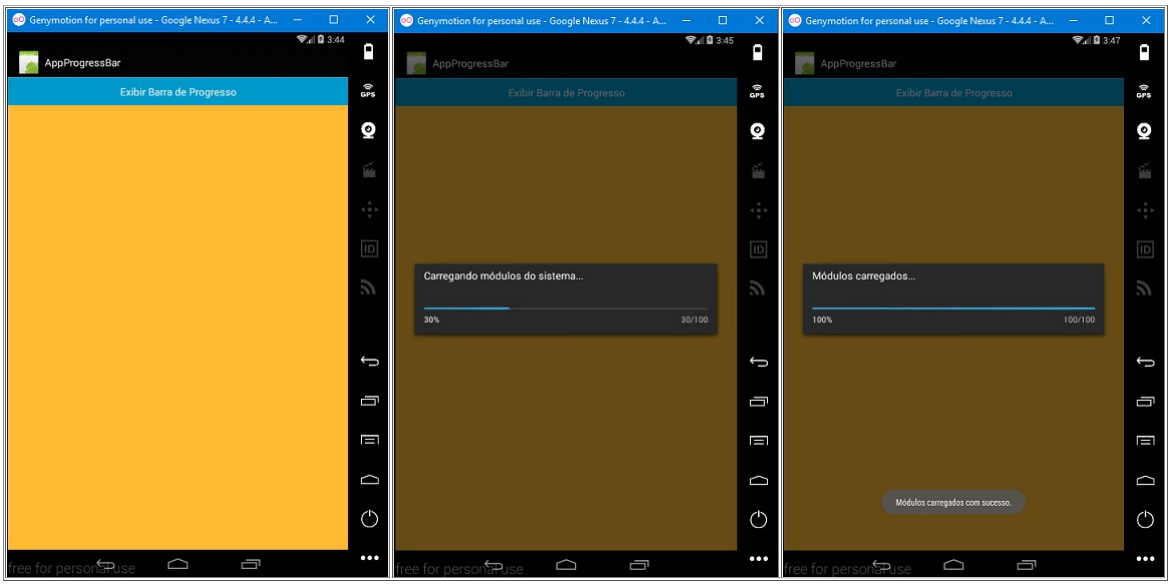
```
pbar.SetCancelable(true);
pbar.SetMessage("Carregando módulos do sistema...");
pbar.SetProgressStyle(ProgressDialogStyle.Horizontal);
pbar.Progress = 0;
pbar.Max = 100;
pbar.Show();

statusBarra = 0;
```

c - A seguir iniciamos uma nova Thread onde temos um laço **While** que vai incrementar a barra de progresso (*estamos usando um atraso de 400 milisegundos)*. Ao final exibimos a mensagem na barra e uma mensagem **Toast** ao usuário:

```
new Thread(new ThreadStart(delegate
{
   while (statusBarra < 100)
   {
      statusBarra += 5;
      pbar.Progress += statusBarra;
      Thread.Sleep(400);
   }
   RunOnUiThread(() => { pbar.SetMessage("Módulos carregados..."); });
   RunOnUiThread(() => { Toast.MakeText(this, "Módulos carregados com sucesso.", ToastLength.Long).Show(); });
})).Start();
```

Observe que usamos o método **RunOnUIThread**() usando expressões lambdas para exibir as mensagens. Esse método executa a ação especificada na thread da UI. Se a thread atual for a thread da UI, então a ação é executada imediatamente. Se a thread atual não for a thread da interface do usuário, a ação é enviada para a fila de eventos da thread.

Executando o projeto usando o emulador do **Xamarin Android Player** e emulando o **Genymotion** iremos obter o seguinte resultado:



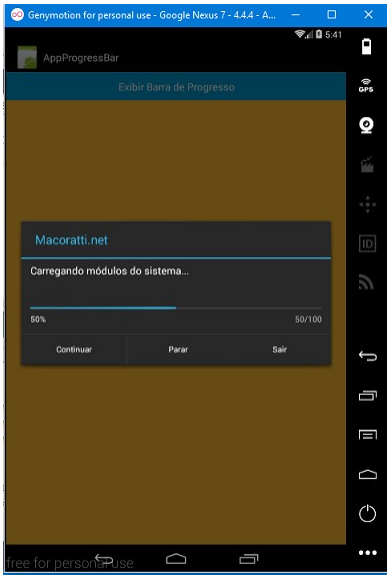
Podemos ainda incluir botões de opções no diálogo da **progressbar** para que o usuário possa interagir durante o processamento da tarefa.

Vamos alterar o código do evento Click do botão de comando incluindo as linhas destacadas em azul conforme mostrado a seguir:

```
private void BtnMostraBarra_Click(object sender, System.EventArgs e)
{
        ProgressDialog pbar = new ProgressDialog(this);

        pbar.SetCancelable(true);
        pbar.SetCanceledOnTouchOutside(false);
        pbar.SetTitle("Macoratti.net");
        pbar.SetMessage("Carregando módulos do sistema...");
        pbar.SetProgressStyle(ProgressDialogStyle.Horizontal);
        pbar.Progress = 0;
        pbar.SetButton("Sair", (s, ev) => { Toast.MakeText(this, "Saindo...", ToastLength.Short).Show(); });
        pbar.SetButton2("Continuar", (s, ev) => { Toast.MakeText(this, "Continuando...", ToastLength.Short).Show(); });
```

```
        pbar.SetButton3("Parar", (s, ev) => { Toast.MakeText(this, "Parando...", ToastLength.Short).Show(); });
        pbar.Max = 100;
        pbar.Show();

        statusBarra = 0;

        new Thread(new ThreadStart(delegate
        {
            while (statusBarra < 100)
            {
                statusBarra += 5;
                pbar.Progress += statusBarra;
                Thread.Sleep(400);
            }
            RunOnUiThread(() => { pbar.SetMessage("Módulos carregados..."); });
            RunOnUiThread(() => { Toast.MakeText(this, "Módulos carregados com sucesso.", ToastLength.Long).Show(); });
        })).Start();

    }
```

Executando o projeto novamente agora temos um título, e 3 opções : **Continuar, Parar, Sair** , que deverão ser tratadas via código. No Exemplo eu estou apenas exibindo mensagens Toast.



Este é um exemplo bem básico que mostra o uso da classe **ProgressDialog** para indicar o processamento de uma tarefa.

Pegue o projeto aqui : **AppProgressBar.zip** (sem as referências)

Porque há um só Deus, e um só Mediador entre Deus e os homens, Jesus Cristo homem.
1 Timóteo 2:5

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**
- VB.NET 2005 - Controles - Macoratti.net
- Seção de Jogos do site Macoratti .net
- 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
- Seção Mobile/Xamarin do site Macoratti .net
- https://developer.android.com/reference/android/app/Activity.html
- https://developer.xamarin.com/api/type/Android.Widget.ProgressBar/

José Carlos Macoratti