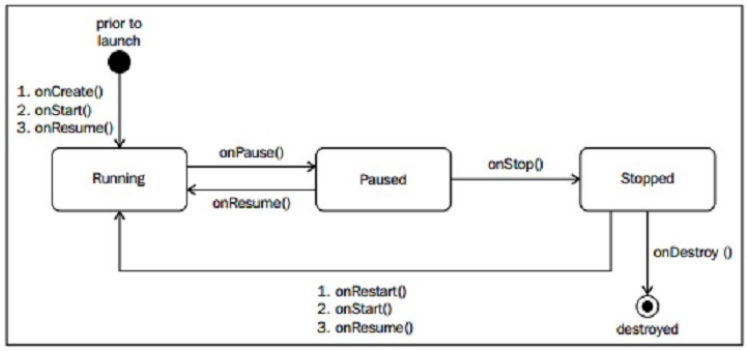
Macoratti.net Xamarin Android - Criando sua primeira Aplicação Android : Activity (Conceitos) - III

Neste artigo vou mostrar como criar uma aplicação **Android** usando os recursos do **Xamarin** no Visual Studio 2015 abordando os principais conceitos relacionados e o conceito de Activity: **Gerenciando a persistência do estado.**

Na **segunda parte do artigo** vimos como passar e recuperar informações entre atividades distintas usando o recurso **Intent** e os métodos **PutExtra() e GetStringExtra().**

Neste artigo veremos como gerenciar a persistência do estado de uma atividade, e novamente eu vou colocar a figura do ciclo de vida de uma **Activity**, pois é importante entender quando os métodos são chamados durante o ciclo de vida:



Enfatizando as ocorrências relacionadas com cada estado podemos fazer um pequeno resumo:

Estados	Ocorrências
Running	As atividades que estão em primeiro plano.(*foreground*) Atividades neste estado tem maior prioridade e só são mortas pelo SO em circunstâncias extremas.
Paused	Atividades pode estar neste estado quando : - O dispositivo esta em repouso - Outra atividade esconde parcialmente a atividade - Uma atividade transparente obstrui a atividade Atividades nestes estado mantêm o seu estado e permanecem anexadas ao gerenciador de janelas. Possuem a segunda prioridade no sistema
Stopped (BackGround)	Uma atividade em *segundo plano ou parada* esta completamente oculta por outra atividade. Elas tentam manter o estado, mas possuem menor prioridade e assim tem mais probabilidade de serem mortas pelo SO para liberar recursos. Até estar morta, uma atividade poderá ser retomada a qualquer momento sem perda de informações de estado. Se a atividade for morta, e, em seguida, o usuário navegar de volta para ela, ela deve ser reiniciada e restaurada ao seu estado anterior *(que esta salvo)*. Isso é responsabilidade do desenvolvedor.

Vamos começar recordando a assinatura do método **OnCreate**() da atividade principal da nossa aplicação :

```
protected override void OnCreate(Bundle bundle)
```

O parâmetro **bundle** passado para o **onCreate**() é uma estrutura de dados que contém o último estado conhecido da atividade, o qual é o mesmo que os valores de variáveis.

Em outras palavras, um **bundle** não nulo significa que o aplicativo esta ***reiniciado*** (*usando o método onRestart()*) e, por conseguinte, deverá restaurar os dados anteriores.

Dessa forma podemos restaurar os dados a partir da estrutura de dados **bundle** quando as atividades forem postas em primeiro plano depois de serem '*mortas*' ou *destruídas* pelo SO (Sistema Operacional).

Porém, neste momento, ainda não fornecemos os mecanismos para serem usados com o objetivo de salvar os dados no interior da estrutura de dados **Bundle**.

Pense no **Bundle** como um container para todas as informações que você desejar persistir. Para fazer isso você usar os métodos **Put*** para inserir dados no Bundle.

Abaixo alguns dos métodos **Put*** disponíveis:

PutString , PutBoolean, PutByte, PutChar, PutFloat, PutLong, PutShort, PutParcelable (usado para objetos).

No método **OnCreate()** o **Bundle** fica disponível novamente ao programa.

Usando a mesma lógica para obter os valores podemos usar os métodos **Get***.

Os dados são armazenados no formato: **nome:valor**. Você inclui a **chave** e o **valor,** e, então, quando você quer recuperar os dados você informa a **chave** e o método retorna o valor.

Vamos ver como gerenciar os métodos **para salvar e restaurar** e as propriedades do **bundle**. *(Lembrando que a classe Bundle é um mapeamento de valores strings para diversos tipos **Parcelable**.)*

O primeiro método que temos de implementar é o método **OnSaveInstanceState**(), que será acionado imediatamente antes da '*morte*' de uma atividade.

O método **OnSaveInstanceState** é chamado para recuperar o estado por instância de uma atividade antes de ser morto, de forma que o estado pode ser restaurado em Activity.OnCreate (Bundle) ou Activity.OnRestoreInstanceState (Bundle) *(a Bundle povoada por este método será passado para ambos)*.

Geralmente restauramos o estado no método **OnCreate**(). Também é possível fazer isso usando o método **OnRestoreInstanceState**(), mas isso não é muito comum, visto que este método é chamado depois de **OnStart**(), ao passo que **OnCreate**() é chamado antes de **OnStart**().

Além disso o método **OnRestoreInstanceState**() é chamado somente quando a atividade é recriada depois que foi morta pelo SO. Tal situação ocorre quando:

- A orientação do dispositivo é alterada (a atividade é destruída e recriada)
- Existe outra atividade na frente da sua atividade, e, em algum ponto, o SO mata a sua atividade afim de liberar memória. Quando você iniciar a sua atividade o método será chamado.

Com esses conceitos em mente vamos para a parte prática.

Recursos usados:

- **Visual Studio Community 2015**
- **Xamarin**
- **Emulador Android virtual ou físico (veja como emular usando o Vysor)**

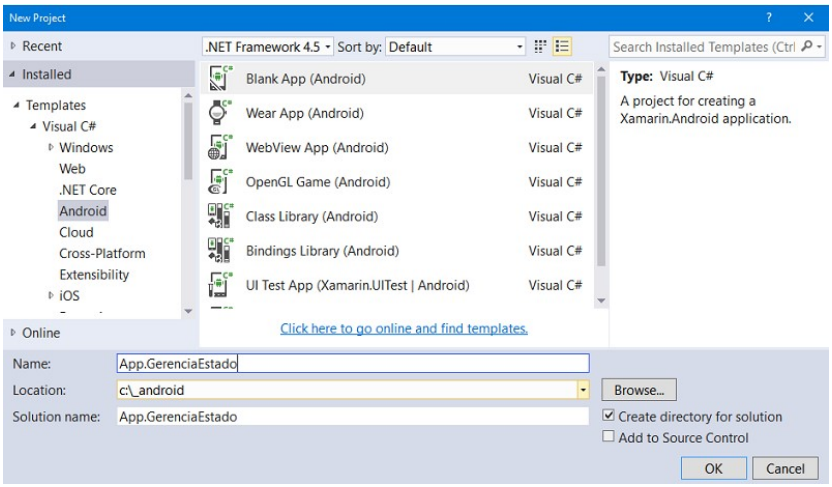
Nota: Baixe e use a versão Community 2015 do VS ela é grátis e é equivalente a versão Professional.

Criando o projeto no VS 2015 Community com Xamarin

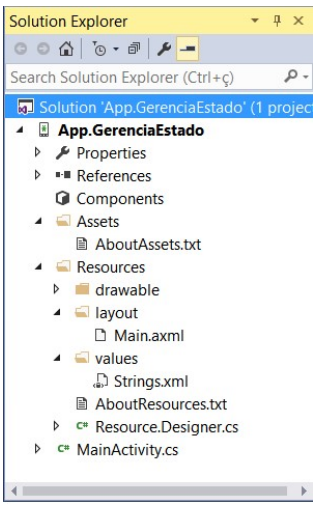
Abra o VS Community 2015 e clique em **New Project**;

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

Informe o nome **App.GerenciaEstado** e clique no botão OK:



Será criada uma solução com a seguinte estrutura:



- **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 desejar usar em tempo de execução.

De forma geral, todas a 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 .axml 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# onde a aplicação é iniciada e representa a atividade da aplicação Android.

Nota : A pasta **Drawable** contém recursos como imagens png, jpg, etc., usadas no aplicativo. Ela contém múltiplas pastas especificas para cada resolução possível em uma aplicação Android. Numa aplicação típica Android você vai acabar encontrando as pastas: Drawable-LDPI, Drawable-mdpi, Drawable-hdpi, Drawable-xhdpi, Drawable-xxhdpi, etc.

O arquivo **MainActivity**.cs, como o nome já sugere, é onde esta definida a **Activity** da aplicação Android.

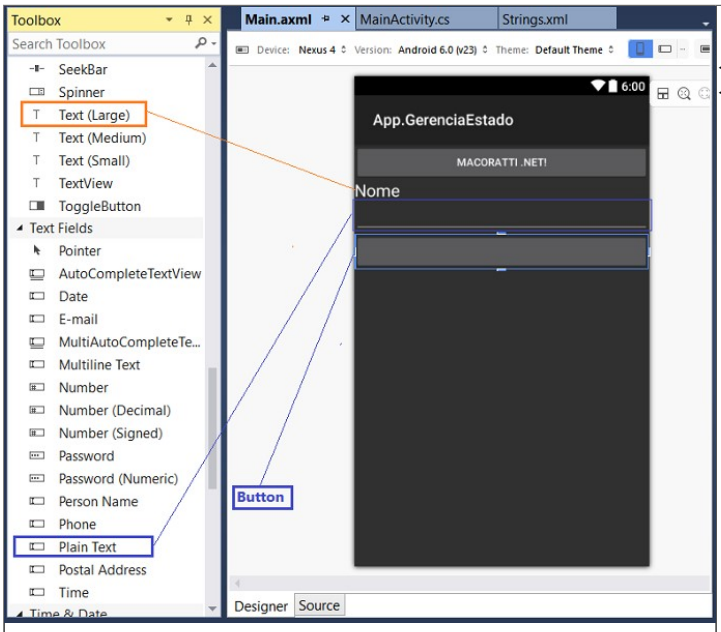
Abrindo o arquivo **MainActivity.cs** veremos o seguinte código:

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

namespace App.PrimeiraApp
{
    [Activity(Label = "App.PrimeiraApp", MainLauncher = true, Icon = "@drawable/icon")]
    public class MainActivity : Activity
    {
        int count = 1;

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

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

            // Get our button from the layout resource,
            // and attach an event to it
            Button button = FindViewById<Button>(Resource.Id.MyButton);

            button.Click += delegate { button.Text = string.Format("{0} clicks!", count++); };
        }
    }
}
```

Vamos alterar o arquivo Main.axml incluindo 3 componentes :

1. TextView - id = **@+id/textView1**
2. EditText - id = **@+id/edtNome**
3. Button - id = **@+id/button1** e Text = ""

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

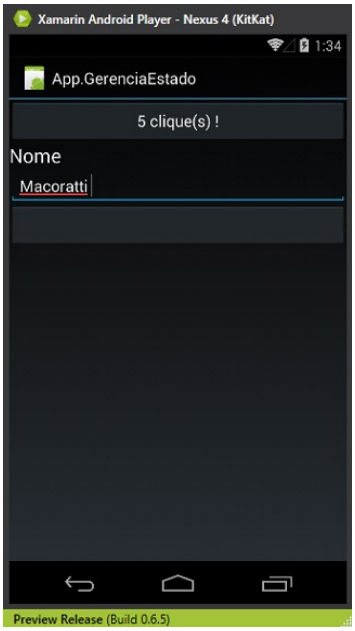


A partir da **ToolBox** arrastamos e soltamos os controles:

- **Text(Large)** - Nome
- **Plain Text** - para receber a entrada do usuário
- **Button** - Com Text vazio

```
<?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">
    <Button
        android:id="@+id/MyButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/Hello" />
    <TextView
        android:text="Nome"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView1" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/edtNome" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/button1" />
</LinearLayout>
```

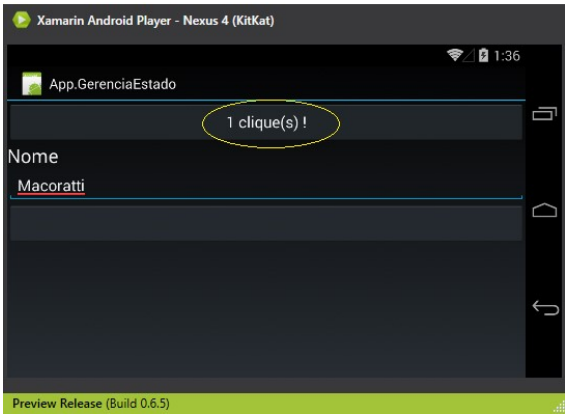
Executando a aplicação e digitando o texto: **Macoratti** e clicando 5 vezes no botão de comando, teremos o resultado mostrado abaixo:



Ao clicar no botão de comando veremos o texto do botão indicar o incremento no contador.

O problema é que o contador esta sendo gerenciado localmente no método e a variável **contador** não esta sendo persistida.

Para provar isso basta mudar a orientação do seu dispositivo android de retrato para paisagem girando o aparelho.



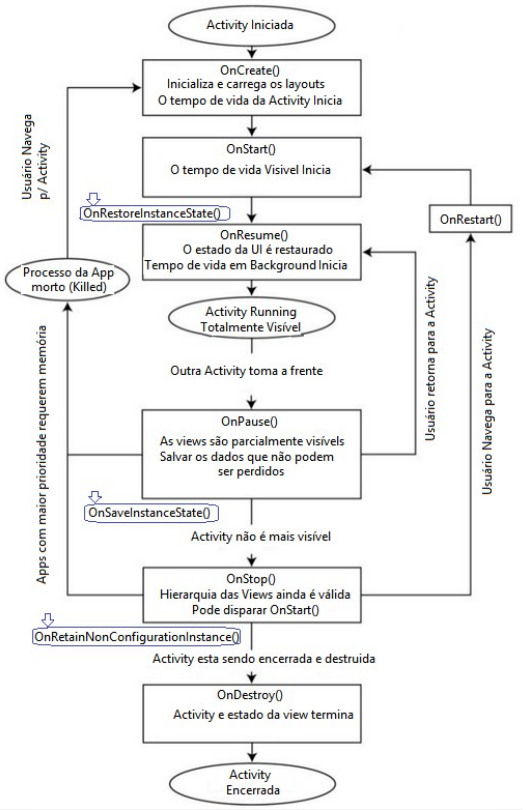
Após mudar a orientação ao clicar no botão de comando notamos que o contador é iniciado novamente o que mostra que o seu valor anterior que era igual a 5 se perdeu.

Já o nome digitado no **EditText** foi persistido e permaneceu exibido no componente.

Como podemos restaurar os valores de variáveis e de componentes quando a Activity for destruída e recriada novamente ?

Usando os métodos **OnSaveInstanceState() e OnRestoreInstanceState().**

Como isso funciona ?



No método **OnSaveInstanceState**() salvamos os dados que desejamos persistir.

Os métodos **OnRestoreInstanceState**() e **OnCreate**() recebem o objeto **Bundle** que contém os dados persistidos.

Em algumas situações o sistema pode encerrar sua **Activity**, seja por falta de recursos ou por mudança de orientação da tela (que foi o nosso caso).

Nesses casos, o método **OnSaveInstanceState** é chamado para permitir ao programador salvar algum dado da **Activity**.

O método é chamado com uma referencia a um Bundle **(bundle)**, e, nessa instância salvamos os dados que queremos persistir.

Quando a **Activity** for recriada esse **bundle** é passado para o método **OnCreate** e **OnRestoreInstanceState**.

Vamos então mostrar que isso é verdade implementando esses métodos em nossa aplicação.

Abaixo temos a implementação dos métodos citados:

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

namespace App.GerenciaEstado
{
    [Activity(Label = "App.GerenciaEstado", MainLauncher = true, Icon = "@drawable/icon")]
    public class MainActivity : Activity
    {
        int contador = 1;
        EditText edtxtNome;
        Button btnNome;

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

            SetContentView(Resource.Layout.Main);

            edtxtNome = FindViewById<EditText>(Resource.Id.edtNome);
            btnNome = FindViewById<Button>(Resource.Id.button1);

            Button button = FindViewById<Button>(Resource.Id.MyButton);
            button.Click += delegate { button.Text = string.Format("{0} clique(s) !", contador++); };
        }

        protected override void OnSaveInstanceState(Bundle outState)
        {
            base.OnSaveInstanceState(outState);

            outState.PutString("NOME_USUARIO", edtxtNome.Text.ToString());
            outState.PutInt("CONTADOR", contador);
        }

        protected override void OnRestoreInstanceState(Bundle savedInstanceState)
        {
            base.OnRestoreInstanceState(savedInstanceState);
```

```
            btnNome.Text = savedInstanceState.GetString("NOME_USUARIO");
            contador = savedInstanceState.GetInt("CONTADOR");
        }
    }
}
```

1- No método **OnSaveInstanceState**()

Usamos os métodos **PuString**(chave,valor) e **PutInt**(chave,valor) para armazenar o texto digitado e o valor do contador:

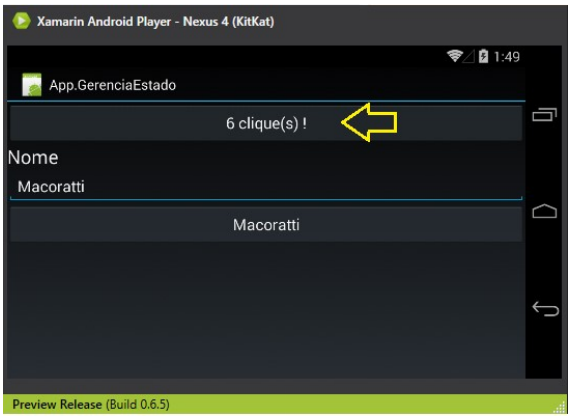
```
outState.PutString("NOME_USUARIO", edtxtNome.Text.ToString());
outState.PutInt("CONTADOR", contador);
```

2 - No método **OnResotreInstanceState**()

Usamos os métodos **GetString**(chave) atribuindo o valor do texto ao texto do controle Button e **GetInt**(chave) para recuperar o valor do contador atribuindo-o à variável contador:

```
btnNome.Text = savedInstanceState.GetString("NOME_USUARIO");
contador = savedInstanceState.GetInt("CONTADOR");
```

Agora se executarmos a aplicação novamente e mudarmos a orientação e clicarmos no botão de comando que veremos que o valor do contador foi preservado:



Por padrão os dados dos componentes da **view (tela)** são salvos e restaurados automaticamente. As variáveis usadas devem ser tratadas e persistidas pelo desenvolvedor.

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

E esta é a mensagem que dele (Jesus) ouvimos, e vos anunciamos: que Deus é luz, e não há nele trevas nenhumas.
Se dissermos que temos comunhão com ele, e andarmos em trevas, mentimos, e não praticamos a verdade.
1 João 1:5,6

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/recipes/android/fundamentals/intent/
- https://developer.xamarin.com/recipes/android/fundamentals/activity/

José Carlos Macoratti