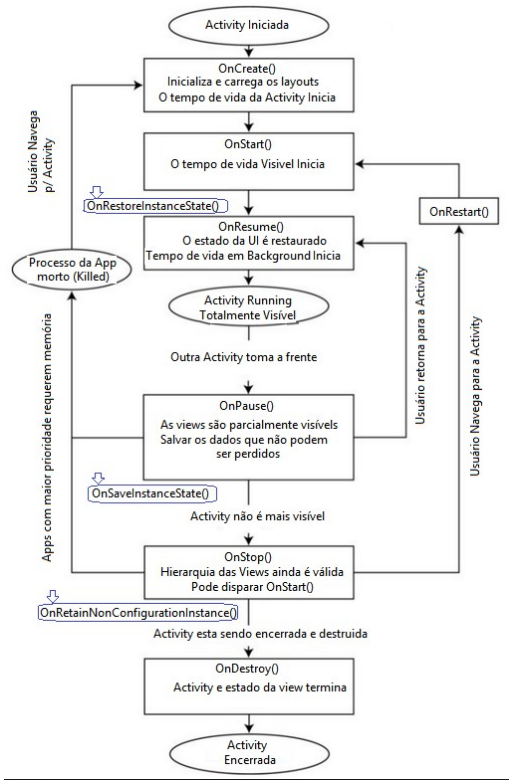
Macoratti.net Xamarin Android - Analisando o ciclo de vida de uma Activity

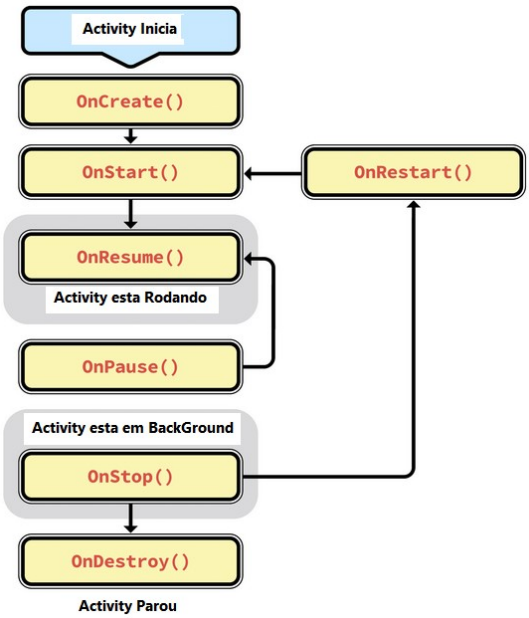
Neste artigo vou analisar o ciclo de vida de uma **Activity (atividade)** em uma aplicação Xamarin Android.

Uma **Activity** ou atividade Android é um componente autônomo de um aplicativo Android que pode ser iniciado, parado, pausado, reiniciado ou recuperado dependendo de vários eventos, incluindo aqueles iniciados pelo usuário e iniciadas pelo sistema. Cada **Activity** representa e gerencia uma **View** que é exibida no dispositivo Android.

As figuras a seguir exibem um resumo do ciclo de vida de uma **Activity** Android.

Temos duas imagens que mostram o ciclo de vida de uma Activity, onde a primeira imagem detalha um pouco mais o que ocorre em cada ciclo de vida e quando são chamados os métodos **OnSaveInstanceState() e OnRestoreInstanceState()**.





Além disso, uma atividade assume os seguintes estados durante o seu ciclo de vida:

Estados	Ocorrências
Running (Rodando)	As atividades que estão em primeiro plano.(*foreground*) Atividades neste estado tem maior prioridade e só são mortas pelo SO em circunstâncias extremas.
Paused (Pausa)	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) **(em segundo plano)**	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.

Com esses conceitos em mente vamos para a parte prática : criar uma aplicação e exibir no **Log** quando cada evento é disparado durante o clico de vida de uma **app**.

Recursos usados:

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

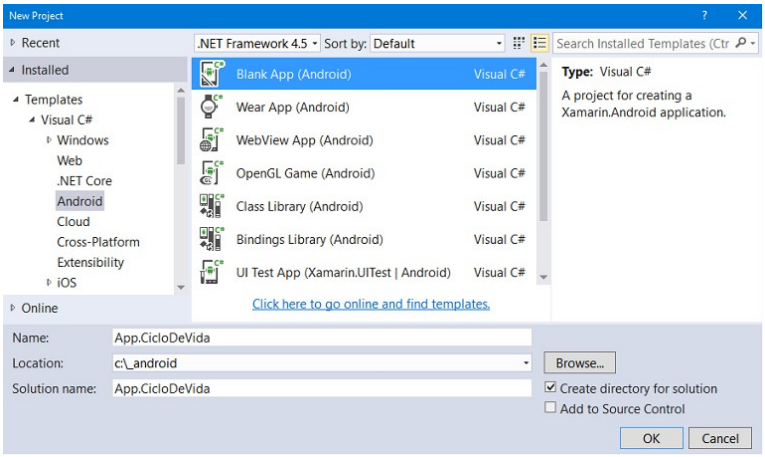
Nota: Baixe e use a versão Community 2015 do VS ela é grátis e é equivalente a versão Professional.

Criando o projeto no VS 2015 Community com Xamarin

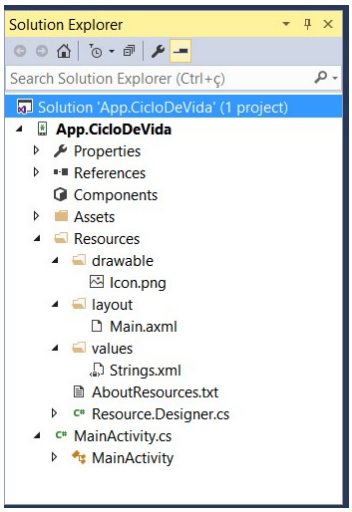
Abra o VS Community 2015 e clique em **New Project**;

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

Informe o nome **App.CicloVida** 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.

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

Vamos alterar o código do arquivo **MainActivity.cs** incluindo a implementação de cada um dos métodos que compõe o ciclo de vida da aplicação.

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

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

        protected override void OnCreate(Bundle bundle)
        {
            Log.Debug(GetType().FullName, "<< OnCreate >>");
            base.OnCreate(bundle);
            SetContentView(Resource.Layout.Main);

            if (bundle != null)
            {
                Log.Debug(GetType().FullName, "<< OnCreate >> - bundle diferente de null");
            }
```

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

        protected override void OnSaveInstanceState(Bundle outState)
        {
            Log.Debug(GetType().FullName, "<< OnSaveInstanceState() >>");
            base.OnSaveInstanceState(outState);
        }

        protected override void OnRestoreInstanceState(Bundle savedInstanceState)
        {
            Log.Debug(GetType().FullName, "<< OnRestoreInstanceState() >>");
            base.OnRestoreInstanceState(savedInstanceState);
        }

        protected override void OnDestroy()
        {
            Log.Debug(GetType().FullName, "<< On Destroy >>");
            base.OnDestroy();
        }

        protected override void OnPause()
        {
            Log.Debug(GetType().FullName, "<< On Pause >>");
            base.OnPause();
        }

        protected override void OnRestart()
        {
            Log.Debug(GetType().FullName, "<< On Restart >>");
            base.OnRestart();
        }

        protected override void OnResume()
        {
            Log.Debug(GetType().FullName, "<< On Resume >>");
            base.OnResume();
        }

        protected override void OnStart()
        {
            Log.Debug(GetType().FullName, "<< On Start >>");
            base.OnStart();
        }

        protected override void OnStop()
        {
            Log.Debug(GetType().FullName, "<< On Stop >>");
            base.OnStop();
        }
    }
}
```

Vamos usar 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.

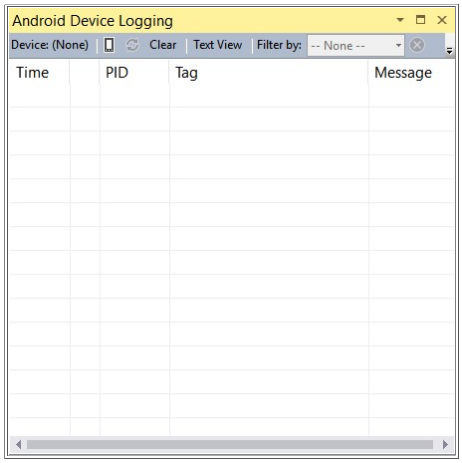
No exemplo usamos o código : **Log.Debug(GetType().FullName, " texto ");** que envia uma mensagem para a janela de Debug.

- GetType().FullName - Retorna o nome do processo que é o nome do nosso projeto **: App.CicloDeVida.MainActivity.**

- texto - O texto que desejamos exibir

Para monitorar o log basta ativar o **Android Device Logging** no menu do Visual Studio: **Tools -> Android -> Android Device Logging**

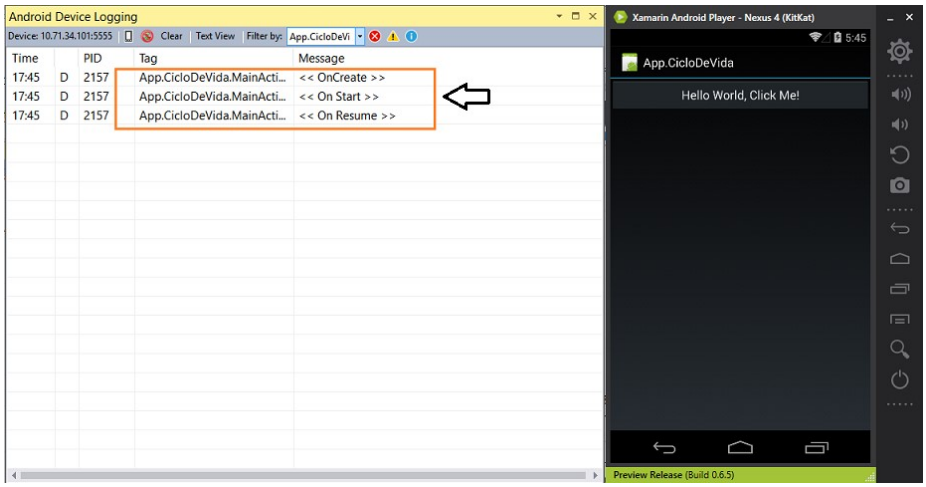
E em **Filter by** selecionar **App.CicloDeVida.MainActivity**:



Agora é só alegria.

Executando a aplicação com o **Android Device Logging** ativo e com filtro definido teremos:

1- Acitivity Running - Iniciou a aplicação

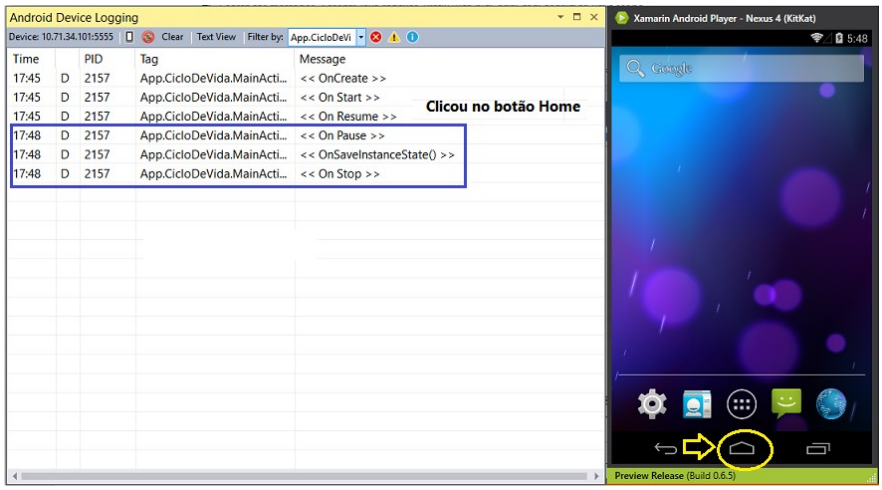


OnCreate()- Primeira método a ser executado quando a Activity é iniciada. Responsável por carregar os layouts XML e outras operações de inicialização. Executado apenas uma vez durante o ciclo de vida.

OnStart()- Chamada logo após o *OnCreate*() terminar – e também quando uma Activity que estava em background volta a possuir foco.

OnResume() - Chamado na inicialização da Activity *(após OnStart())* e também quando uma Activity volta a possuir foco e esta pronta para interagir com o usuário;

2- Clicou no botão Home

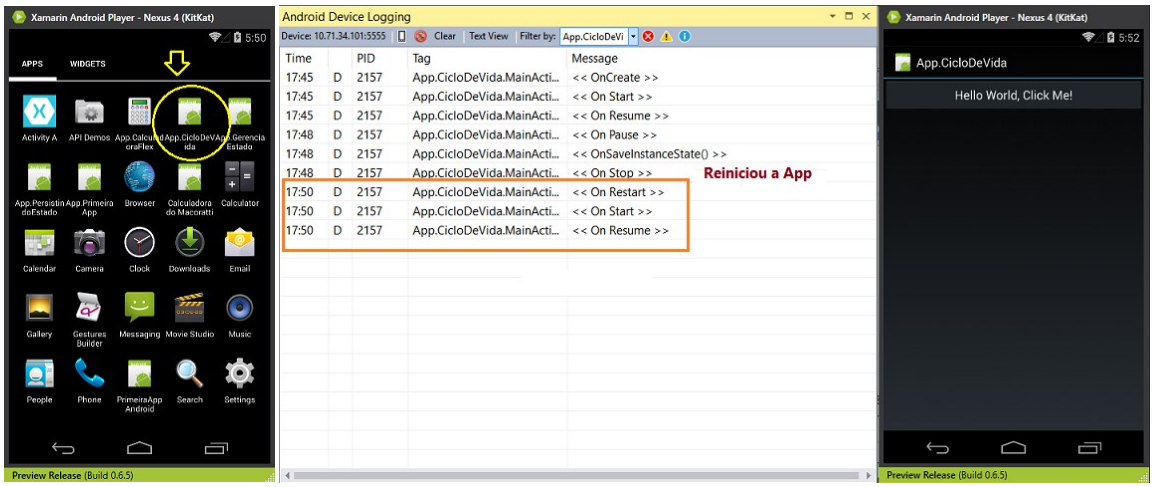


OnPause() - Primeira método invocado quando a Activity perde o foco (quando outra Activity vem à frente) e vai para segundo plano;

OnSaveInstanceState() - Permite persistir o estado da Activity. É chamada antes da Activity ser destruída.

OnStop() – Chamado quando a Activity fica totalmente encoberta por outra Activity e não é mais visível pelo usuário

3- Clicou na ícone da App para reiniciar a aplicação

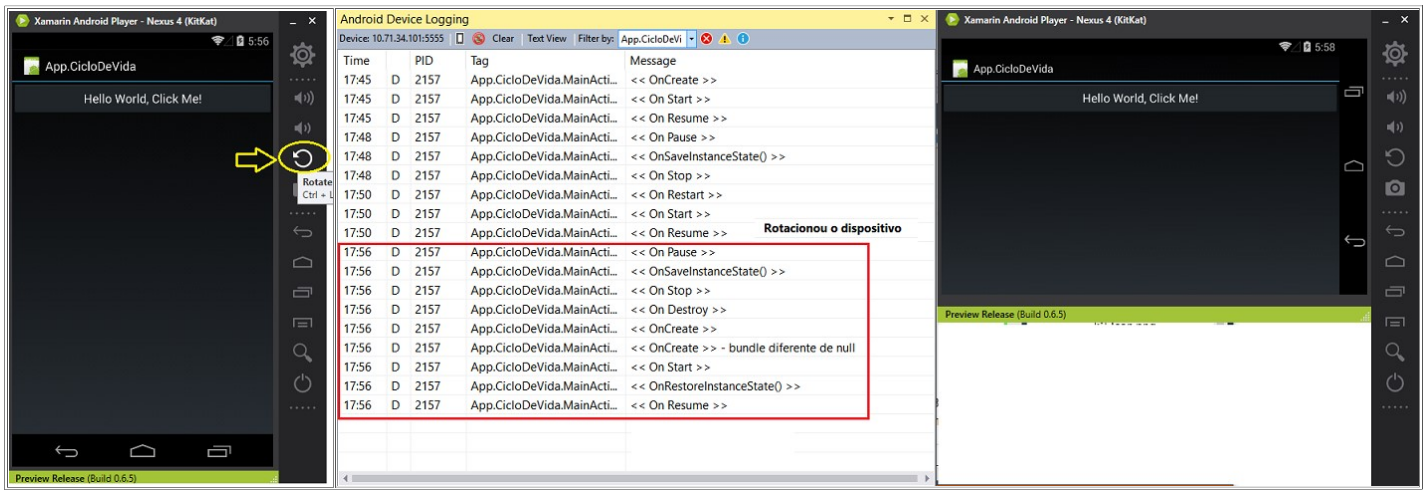


OnRestart() - Chamado antes da OnStart(), quando uma Activity volta a ter o foco depois de estar em background.

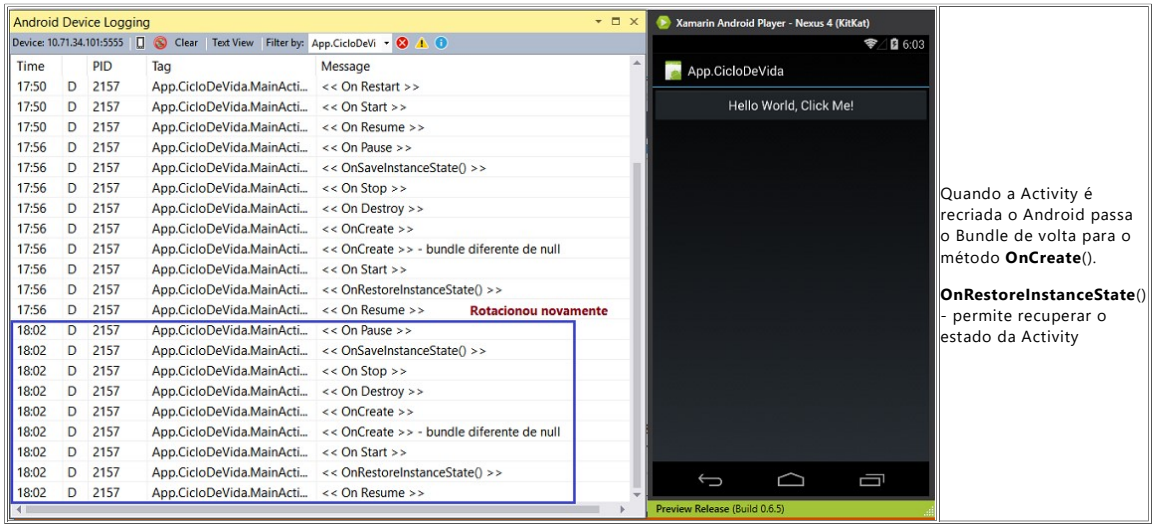
OnStart()- Chamada logo após o *OnCreate*() terminar – e também quando uma Activity que estava em background volta a possuir foco.

OnResume() - Chamado na inicialização da Activity *(após OnStart())* e também quando uma Activity volta a possuir foco e esta pronta para interagir com o usuário;

4- Mudou a orientação do dispositivo para Paisagem:



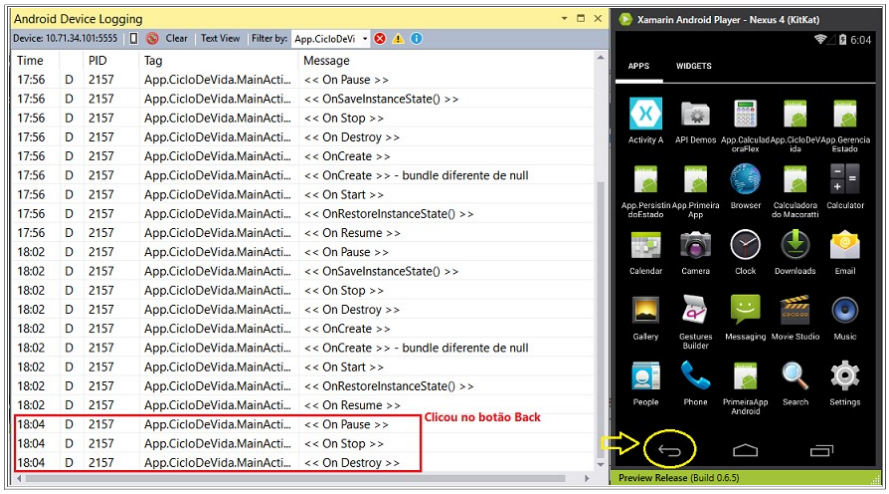
5- Mudou a orientação do dispositivo para Retrato



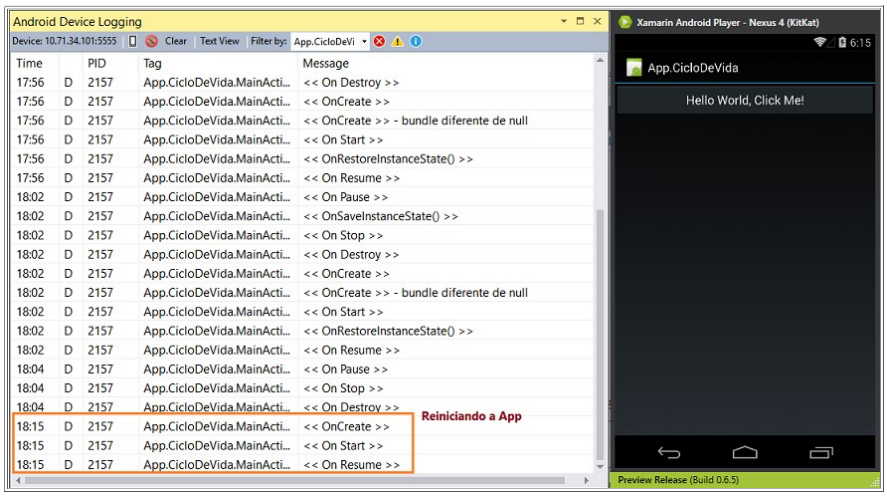
Quando a Activity é recriada o Android passa o Bundle de volta para o método **OnCreate**().

OnRestoreInstanceState() - permite recuperar o estado da Activity

6- Clicou no botão Back



7- Reiniciou a aplicação



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

Segui a paz com todos, e a santificação, sem a qual ninguém verá o Senhor; Hebreus 12:14

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/
- https://developer.xamarin.com/api/type/Android.Util.Log/

José Carlos Macoratti