Macoratti.net C# - Programação Assíncrona como : Async e Task

Neste artigo vou revisar os conceitos da programação assíncrona na linguagem C#.

A programação assíncrona é um poderoso recurso da linguagem C# que permite que você continue com a execução do seu programa na thread principal enquanto uma tarefa de longa duração é executada na sua própria thread separadamente da thread principal.

Quando a tarefa de longa duração estiver completa, ela permitirá que a thread principal saiba se tarefa foi concluída com sucesso ou se a operação falhou.

Os tipos de retorno dos métodos/funções assíncronos

Na programação assíncrona, os métodos assíncronos podem ter três tipos possíveis de retorno :

- void -O retorno do tipo **void** é usado em manipuladores de eventos, lembrando que **void** não retorna nada. Logo, se você chamar um método assíncrono com tipo de retorno **void**, seu código de chamada deverá estar apto a continuar a execução do código sem ter que esperar pela conclusão do método assíncrono;

- Task - Com métodos assíncronos que possuem um tipo de retorno **Task** você pode utilizar o operador **await** para pausar a execução da thread atual até que o método assíncrono chamado tenha sido concluído. Note que um método assíncrono que retorna um tipo **Task** não retorna um operando, assim, se ele fosse escrito como um método síncrono normal, seria um método com tipo de retorno **void**.

- Task<TResult> - Métodos assíncronos que têm uma declaração de retorno têm um tipo de retorno **TResult**. Em outras palavras, se o método assíncrono retornar um booleano, você deve criar um método assíncrono com um tipo de retorno **Task<bool>**.

Vamos então criar um exemplo onde iremos usar esses tipos de retorno.

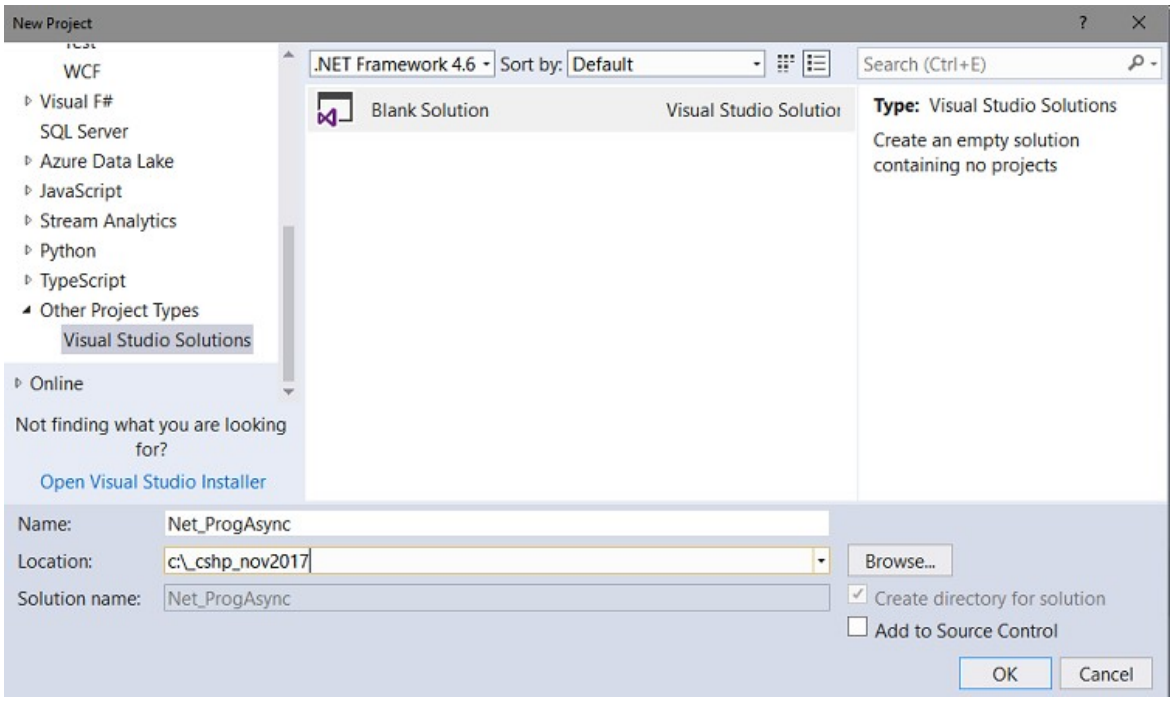
Recursos Usados:

- Visual Studio 2017 Community

Programação Assíncrona com retorno void

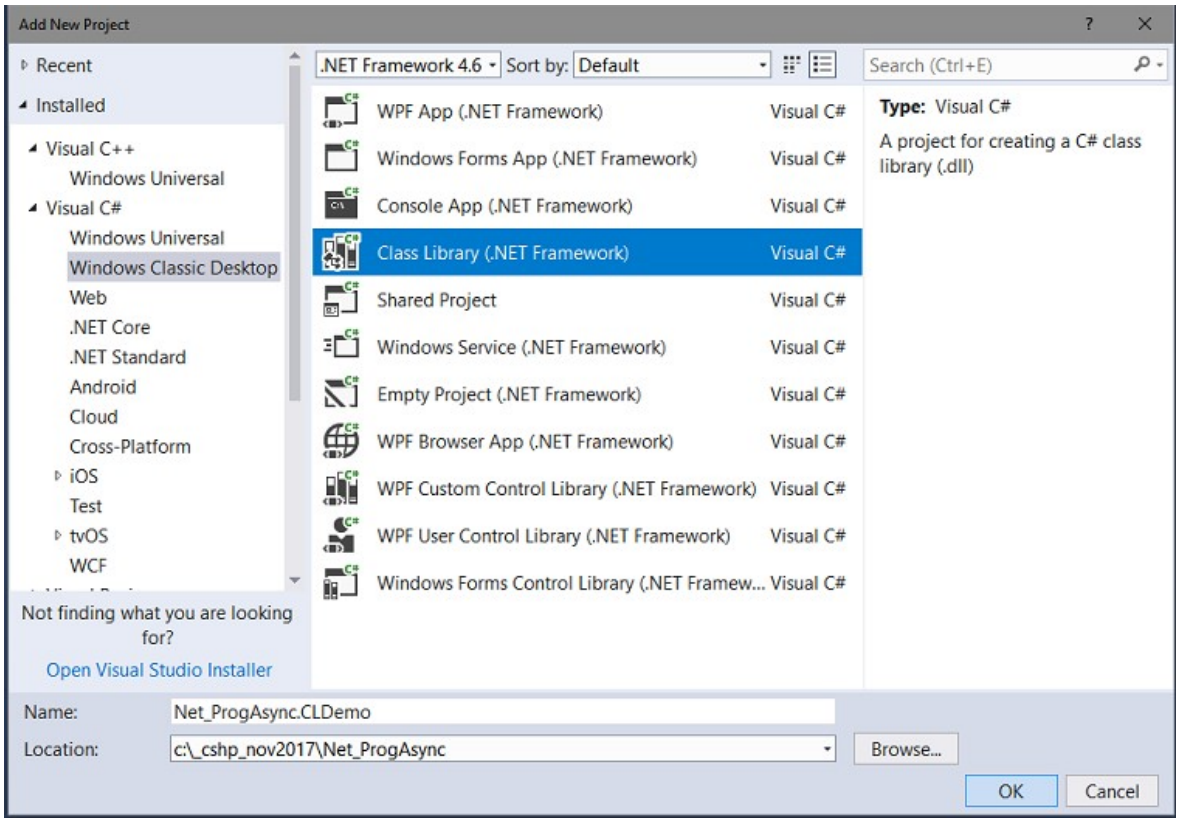
Abra o VS 2017 Community e crie uma solução em branco usando o template: **Other Project Types -> Visual Studio Solutions;**

Escolha **Blank Solution** e informe o nome **Net_ProgAsync** e clique em OK;



A seguir no menu **File** clique em **Add -> New Project e escolha Visual C# -> Windows Classic Desktop -> Class Library (.NET Framework);**

Informe o nome **Net_ProgAsync.CLDemo** e clique em OK;

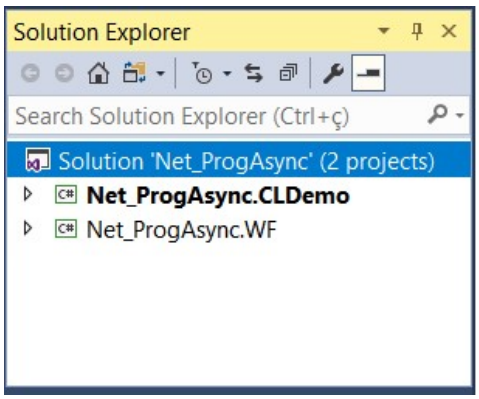


O projeto **Class Library** vai criar uma classe **Class1.cs** no projeto altere o nome dessa classe para **Exemplo.cs.**

Após isso inclua outro projeto na solução via opção **File-> Add -> New Project** e escolha **Windows Forms App (.NET Framework);**

Informe o nome **Net_ProgAsync.WF** e clique em OK.

Ao final teremos uma solução contendo dois projetos conforme mostra a figura abaixo:

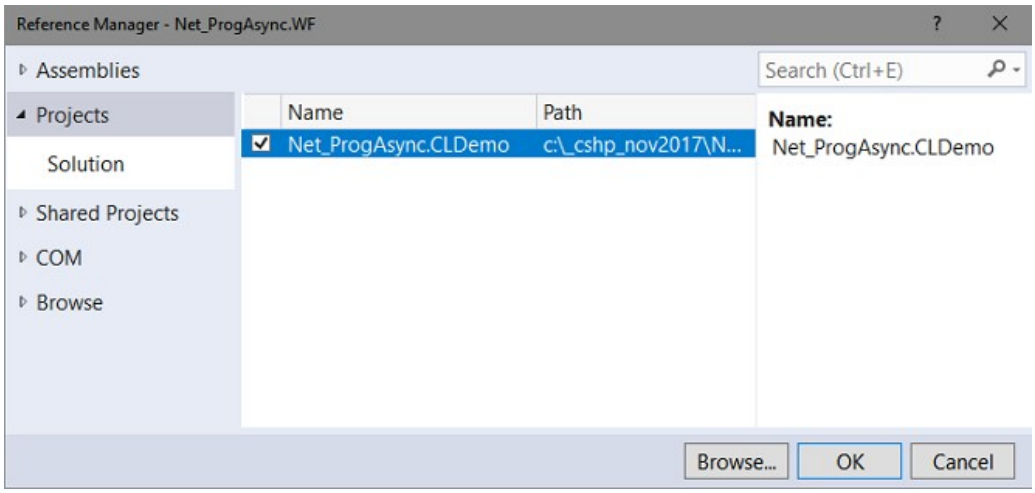


Projeto Windows Forms : Referenciando o projeto Class Library

Vamos incluir uma referência no projeto **Windows Forms** ao projeto **Class Library.**

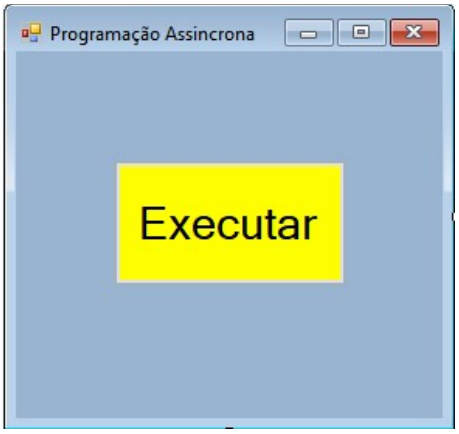
Na janela Solution Explorer clique com o botão direito sobre o projeto **Windows Forms (Net_ProgAsync.WF)** e a seguir clique em **Add -> Reference;**

Clique na guia **Projects** e marque o projeto **Net_ProgAsync.CLDemo** e clique em OK;



Para não esquecer vamos definir o projeto Windows Forms como o projeto a ser executado. Clique com o botão direito sobre o projeto e a seguir clique em : **Set As Startup Project.**

Abra o formulário **Form1.cs** do projeto e inclua um botão de comando neste formulário com o **Name** igual **btnExecutar** e Text igual a **Executar** conforme mostra a figura abaixo:



Antes de podermos usar o evento **Click** do botão vamos criar os métodos Assíncronos no projeto Class Library: **Net_ProgAsync.CLDemo**

Projeto Class Library : Criando métodos Assíncronos

Abra o arquivo **Exemplo.cs** do projeto **Class Library** e vamos criar os seguintes métodos na classe **Exemplo**:

1- Método **Task_TResult_Async**() que possui um retorno do tipo **Task<TResult>**, que no exemplo é um Boolean.

Este método apenas verifica se um determinado ano é bissexto retornando true em caso positivo:

```
async Task<bool> Task_TResult_Async()
{
    return await Task.FromResult<bool>(DateTime.IsLeapYear(DateTime.Now.Year));
}
```

2- Método **Task_Void_Async** que retorna void. O método retorna um **Task** para permitir usar o operador **await**.

O método não retorna nenhum resultado, por isso é um método que retorna **void**. Porém para poder usar o operador await retornamos um tipo **Task**:

```
        async Task Task_Void_Async()
{
    await Task.Delay(5000);
    Console.WriteLine("5 segundos de atraso");
}
```

3- Método **Task_LongaDuracao** que vai chamar os métodos anteriores e exibirá o resultado da verficação do ano bissexto. Note que estamos usando a palavra **await** na chamada dos dois métodos:

O operador **await** é aplicado a uma tarefa em um método assíncrono para inserir um ponto de suspensão na execução do método até que a tarefa aguardada seja concluída. A tarefa representa um trabalho em andamento.

```
public async Task Task_LongaDuracao()
{
```

```
        bool isAnoBissexto = await Task_TResult_Async();
        Console.WriteLine($"{DateTime.Now.Year} {(isAnoBissexto ? " é " : " não é ")} um Ano Bissexto");
        await Task_Void_Async();
    }
```

Agora podemos retornar ao projeto Windows Forms e incluir o código abaixo no evento **Click** do botão de comando **Executar**:

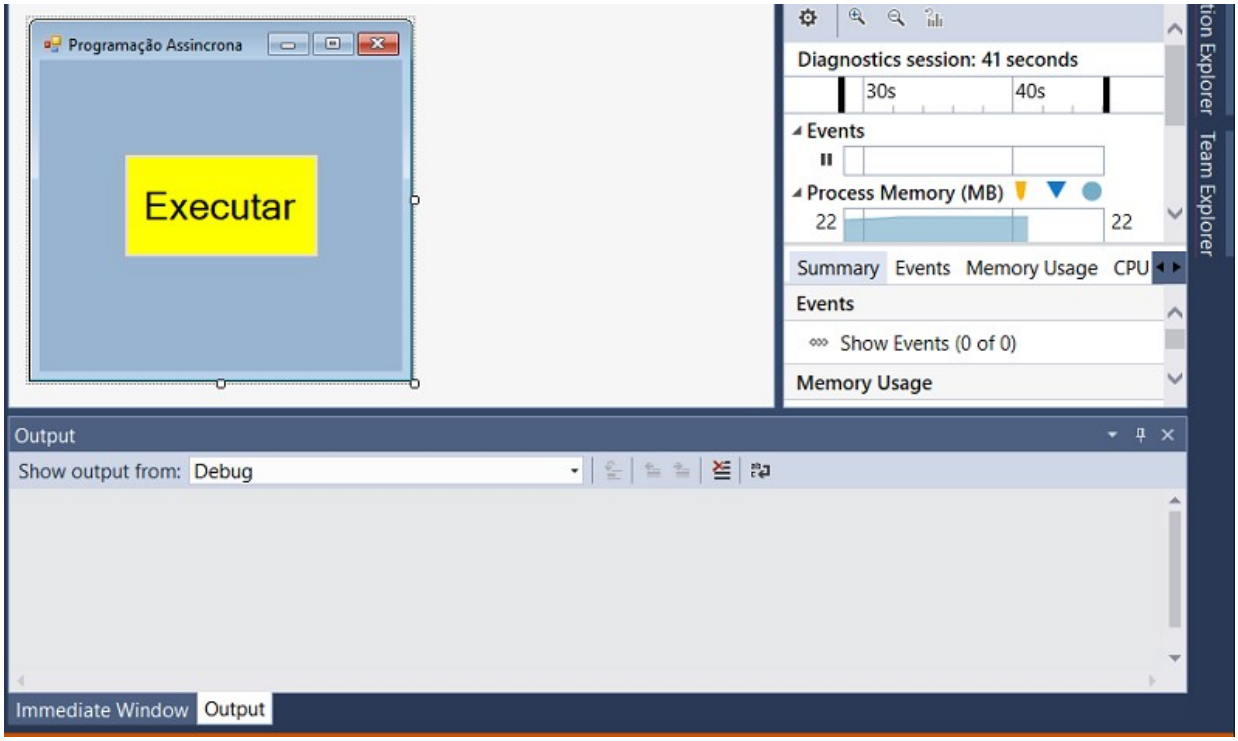
```
using Net_ProgAsync.CLDemo;
using System;
using System.Windows.Forms;

namespace Net_ProgAsync.WF
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private async void btnExecutar_Click(object sender, EventArgs e)
       {
          Console.WriteLine("Evento do botão foi iniciado : cliquei aqui");

          Exemplo oProgAsync = new Exemplo();
          await oProgAsync.Task_LongaDuracao();

          Console.WriteLine("Evento do botão foi concluído");
        }
    }
}
```

Neste código a primeira coisa que fizemos foi incluir a palavra **async** no evento pois estamos usando a palavra **await** na chamada do método **Task_LongaDuracao**() da classe **Exemplo**(), cuja instância criamos no código do botão.

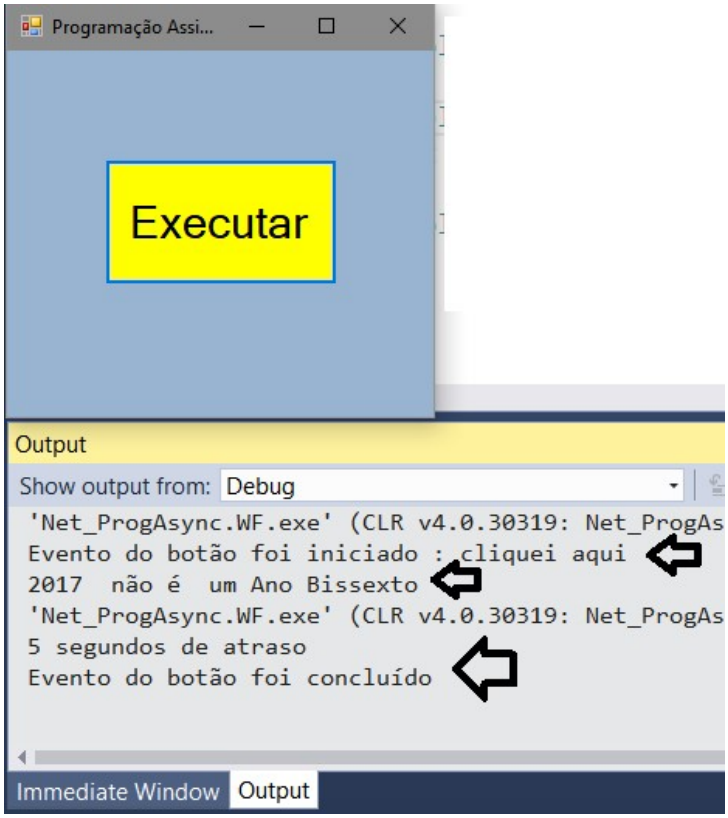
A palavra **await** só pode ser usada em um método assíncrono modificado pela palavra-chave **async**. Esse tipo de método, definido pelo uso do modificador **async** e, geralmente, contendo uma ou mais expressões **await**, é conhecido como um método assíncrono.

Executando o projeto Windows Forms e verificando o resultado

Vamos executar a solução que irá apresentar o formulário **Form1.cs.** Antes de clicar no botão de comando torne a janela **Output** (Ctrl+ Alt + O) visível conforme abaixo:



Ao clicar no botão de comando você verá a exibição do resultado na janela **Output** e vai notar que o formulário executa a operação e continua responsivo.



Para deixar bem claro que estamos executando operações assíncronas vamos alterar o código do método **Task_LongaDuracao** conforme abaixo:

```
public async Task Task_LongaDuracao()
{
   bool isAnoBissexto = await Task_TResult_Async();
```

```
        for (int i = 0; i <= 10000; i++)
        {
            i++;
        }
        isAnoBissexto = await Task_TResult_Async();

        Console.WriteLine($"{DateTime.Now.Year} {(isAnoBissexto ? " é " : " não é ")} um Ano Bissexto");
        await Task_Void_Async();

        Task taskTResultAsync = Task_TResult_Async();
        for (int i = 0; i <= 10000; i++)
        {
            i++;
        }
        await taskTResultAsync;
    }
```

Agora estamos incluindo uma demora maior na execução das tarefas do método para você notar que o formulário Windows Forms fica liberado executando em uma thread separada enquanto a execução do método **Task_LongaDuracao** realiza as tarefas designadas.

Pegue o projeto aqui : **Net_ProgAsync.zip**

"Mas nós não recebemos o espírito do mundo, mas o Espírito que provém de Deus, para que pudéssemos conhecer o que nos é dado gratuitamente por Deus. "

1 Coríntios 2:12

- **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 (@macorati) | Twitter**
- **C# - Compreendendo as palavras chaves Constantes ... - Macoratti.net**

José Carlos Macoratti