Macoratti.net C# - Programação Assíncrona com async e await (revisitado) - I

Neste artigo vou revisar os conceitos da programação assíncrona com **async e await** da linguagem C#.

Já sabemos que a programação assíncrona usando **async e await** melhora a capacidade de reposta da sua aplicação.

Assim, você deve usar este recurso quando enfrentar atividades que podem bloquear sua aplicação.

A seguir temos uma lista de tarefas nas quais você pode usar a programação assíncrona para incrementar a responsividade da sua aplicação:

Acesso a Web • HttpClient • SyndicationClient Acesso a Arquivos • StorageFile • StreamWriter • StreamReader • XmlReader	Acesso a Imagens • MediaCapture • BitmapEncoder • BitmapDecoder WCF • Asynchronous Service Operations

Você já sabe também que as palavras-chave **async e await** são essenciais na programação assíncrona. Usando essas palavras-chave, podemos criar métodos assíncronos.

Mas o que mais caracteriza os métodos assíncronos?

Devemos considerar o seguinte :

1. A assinatura do método deve incluir o modificador **async;**
2. O método deve ter um tipo de retorno da **Task<TResult>, Task ou void;**
3. As declarações de método devem incluir pelo menos uma única expressão **await** - isso diz ao compilador que o método precisa ser suspenso enquanto a operação aguardada estiver ocupada.
4. Por último, o nome do método deve terminar com o sufixo **"async"** *(mesmo que isso seja mais convencional do que o necessário).*

Por que essa convenção seria significativa, você pergunta ?

Bem, essa é a convenção usada no .NET Framework 4.5 e versões mais recentes para facilitar a busca e o intellisense. É também uma forma de sinalizar para quem ler o seu código que aquele método é assíncrono.

Definindo a aplicação exemplo do artigo

Vamos criar uma aplicação **Windows Forms** usando os controles **Timer, 2 Buttons e TextBox.**

Neste formulário teremos um método que simula a leitura de texto a partir de uma fonte de dados que leva 8

segundos para ser completada. Desejamos que nossa aplicação Window Forms se mantenha responsiva durante toda a operação.

Para fazer isso vamos incluir um **Timer** no formulário da aplicação Windows Forms e um botão onde o usuário vai clicar para esperar por 8 segundos antes que o texto seja retornado e exibido na **Label** no formulário.

Durante todo o processo o **timer** vai continuar a exibir a hora atual *(incluindo os segundos)* na barra de título do formulário.

Vamos agora criar o projeto.

Recursos Usados neste artigo :

- VS 2017 Community (update 15.5)

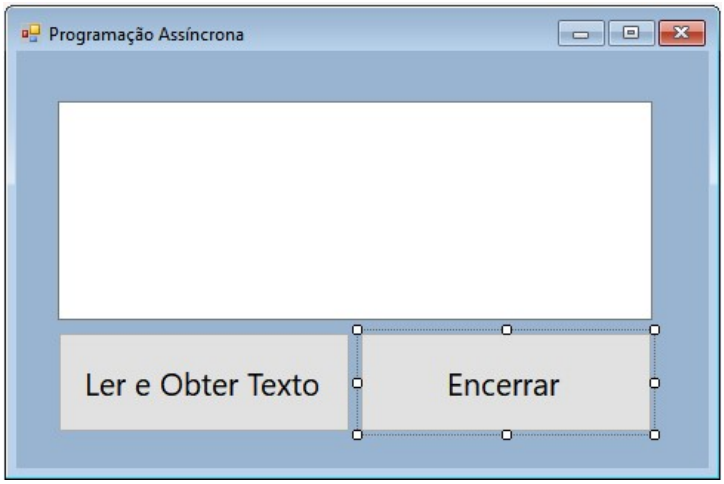
Criando o projeto Windows Forms

Abra o VS 2017 Community e crie um projeto do tipo **Windows Forms App** chamado **WF_ProgAssinc**.

No formulário **Form1.cs** incluaos seguintes controles a partir da ToolBox:

- 1 TextBox - txtTexto
- 1 Button - btnGetTexto
- 1 Button - btnSair
- 1 Timer - timer1

Abaixo vemos o leiaute do formulário:



Começe definindo o código do evento **Load** do formulário para iniciar o Timer:

```
private void Form1_Load(object sender, EventArgs e)
 {
    timer1.Start();
 }
```

A seguir no evento **Tick** do **Timer** inclua o código para exibir na barra de títulos do formulário a hora atualizada:

```
private void Form1_Load(object sender, EventArgs e)
```

```
{
      this.Text = DateTime.Now.ToLongTimeString();
}
```

Criando o método Assíncrono

Agora inclua o código abaixo que criar o método assíncrono **LerTextoAsync**()

```
private async Task<string> LerTextoAsync()
{
   await Task.Delay(TimeSpan.FromMilliseconds(8000));
   return "Macoratti .net - artigo programação assíncrono na linguagem c#";
}
```

Observe que usamos a palavra **async** na assinatura do método e que **Task<string>** nos diz que que este método retorna uma string. A string será retornada após 8 segundos.

Usamos a palavra **await** para incluir um ponto de suspensão na execução do método até que a tarefa aguardada seja concluída. A tarefa representa um trabalho em andamento.

Lembre-se que **await** só pode ser usado em um método assíncrono modificado pela palavra-chave **async**.

No evento **Click** do botão de comando **Ler e Obter Texto** inclua o código a seguir:

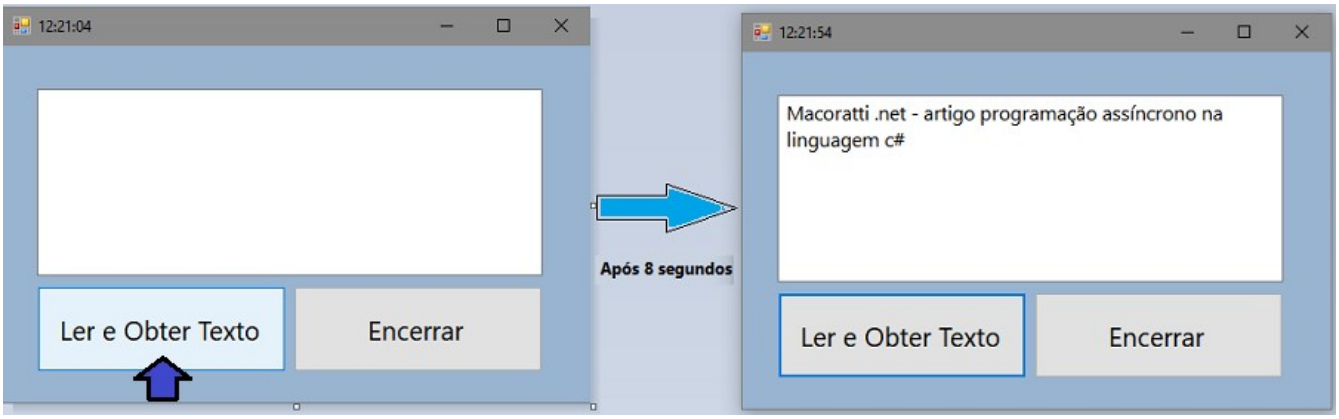
```
private async Task<string> LerTextoAsync()
{
   await Task.Delay(TimeSpan.FromMilliseconds(8000));
   return "Macoratti .net - artigo programação assíncrono na linguagem c#";
}
```

No evento **Click** do botão **Encerrar** inclua o código que sai da aplicação:

```
private void btnSair_Click(object sender, EventArgs e)
{
   Application.Exit();
}
```

Executando a aplicação e analizando o resultado

Execute a aplicação e a seguir clique no botão **'Ler e Obter Texto':**



Ao iniciar a aplicação você vai notar que a hora atual ser exibida na barra de títulos do formulário.

Após clicar no botão de comando e aguardar o texto ser exibido na caixa de texto você vai perceber que o formulário continua responsivo e que você pode mover o formulário e realizar outras operações como clicar no botão **Encerrar**.

A hora também vai continuar ser atualizada na barra de tarefas do formulário mostrando que tudo esta responsivo em nosso projeto.

Assim, esse simples exemplo mostra o funcionamento da programação assíncrona com **async e await.**

Na **próxima parte do artigo** veremos como evitar o **deadlock** ou bloqueio morto ou mortal usando o **async**.

Eu sou a videira verdadeira, e meu Pai é o lavrador.
Toda a vara em mim, que não dá fruto, a tira; e limpa toda aquela que dá fruto, para que dê mais fruto.
João 15:1,2

- **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# - Usando Reflection na prática - I - Macoratti.net
- C# - Programação Assincrona - Macoratti.net
- VB .NET - Programação assíncrona com Async e Wait - Macoratti.net
- WPF - Usando async/await para melhorar o desempenho - Macoratti.net
- VB .NET - Usando Tasks - Macoratti.net
- C# - Executando Tasks - Macoratti.net
- C# - Tasks x Threads. Qual a diferença - Macoratti.net

José Carlos Macoratti