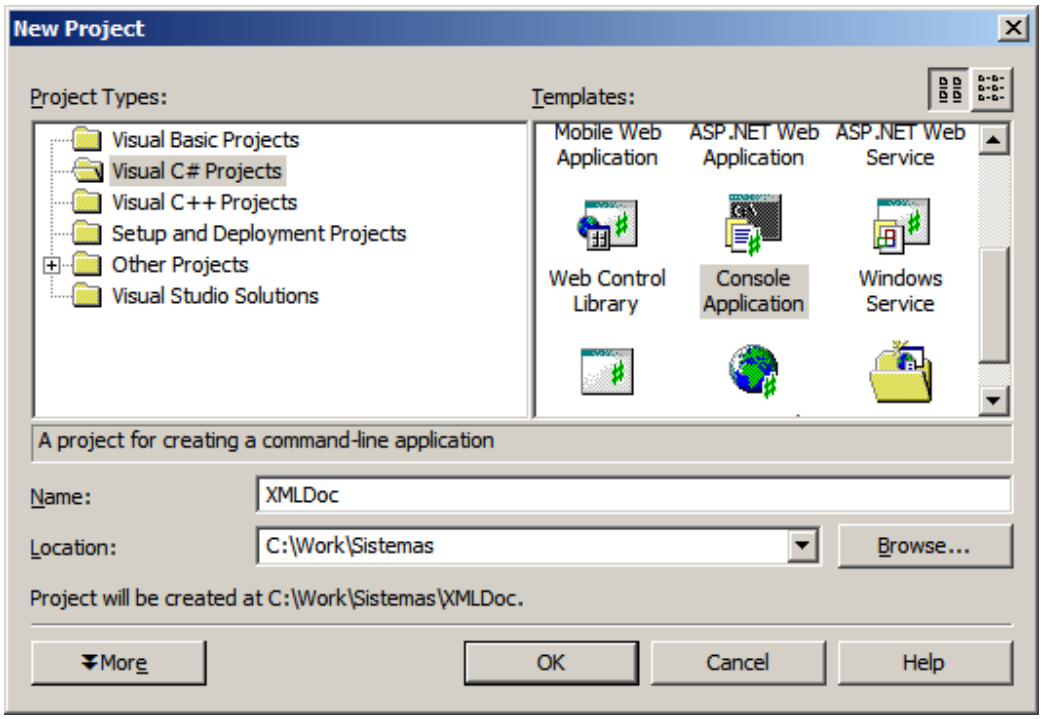
Comentários XML no Visual Studio.NET

Já conhecíamos o fato de que a Microsoft estava adotando o XML para tudo que se pudesse imaginar, mas eu tenho de reconhecer que não imaginava que a documentação de um código poderia ser em XML.

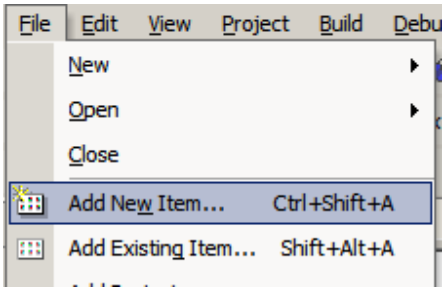
Com a documentação de código feita em XML, que é uma forma muito mais padronizada para visualização de informações, o Visual Studio.NET consegue fazer duas coisas muito interessantes. São elas: Integração com o Intelisense e documentação de seu aplicativo em HTML.

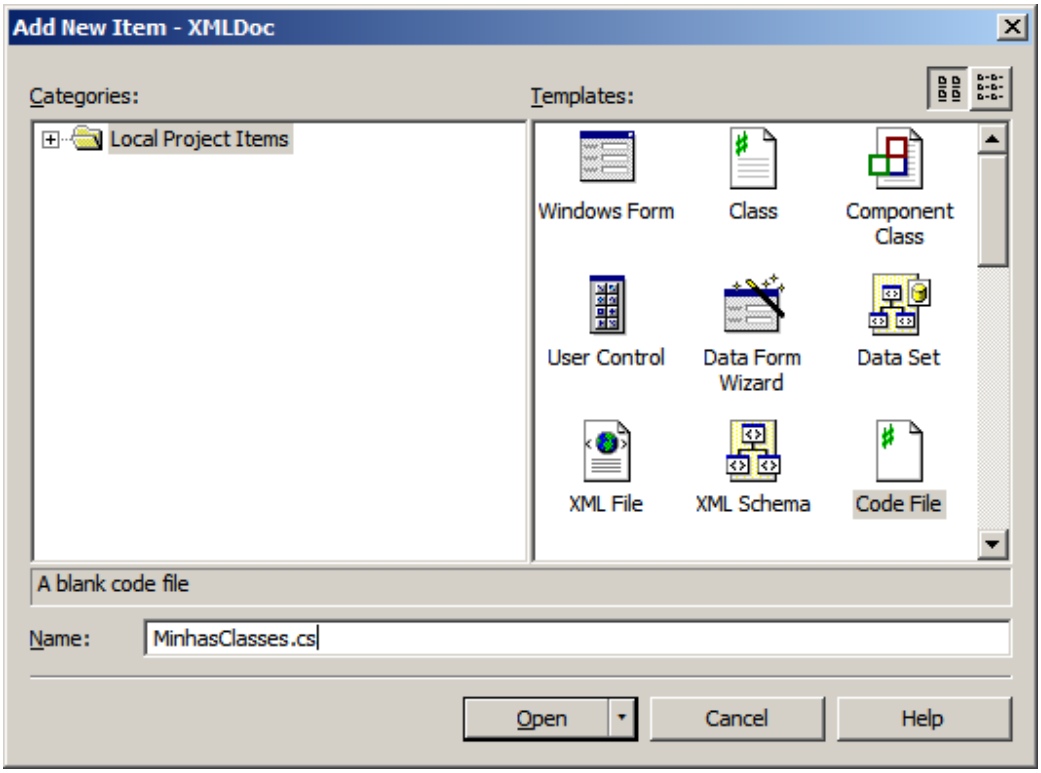
Para ver como funciona a integração de nossa documentação XML com o intelisense do Visual Studio .NET, iremos criar uma pequena aplicação conforme o passo-a-passo a seguir. Vamos lá:

Crie uma nova Console Application com o nome XMLDoc:



Crie um novo Item do tipo Code File com o nome Classe.cs:





Agora, insira o seguinte código nesta folha de códigos em branco.

```
using System;

namespace XMLDoc
{
	public class TesteDoc
	{
		public void EscreverNome()
		{ Console.WriteLine("TesteDoc"); }

		public void EscreveData()
		{ Console.WriteLine(DateTime.Now.ToShortDateString()); }

		public void EscreveHora()
		{ Console.WriteLine(DateTime.Now.ToShortTimeString()); }
	}
}
```

Vamos conhecer a magia do Visual Studio.NET.

Posicione o cursor do teclado sobre a sintaxe *public class TesteDoc* e pressione a tecla / três vezes. Note que na última, o Visual Studio.NET escreve para você um bloco XML com a tag *summary*.

Preencha o a tag com o seguinte conteúdo:

```
/// <summary>
/// Esta classe possui algumas dicas para documentação XML.
/// </summary>
```

Bom, agora vamos fazer o mesmo com nossos métodos públicos. Para não alongar muito, inserimos todos os comentários de uma vez, conforme a listagem a seguir. O resultado final deve ser:

```
using System;

namespace XMLDoc
{
	/// <summary>
	/// Esta classe possui algumas dicas para documentação XML.
	/// </summary>
	public class TesteDoc
	{
		/// <summary>
		/// Este método escreve o nome da classe no Console.
		/// </summary>
		/// <param name="descricaoAplicacao">Descrição da
aplicação</param>
		/// <param name="numeroQualquer">Numero qualquer a ser
escrito.</param>
		public void EscreverNome(string descricaoAplicacao, int
numeroQualquer)
		{
			string mensagem = this.ToString() + "::" +
descricaoAplicacao +                          "::" + numeroQualquer.ToString();
			Console.WriteLine(mensagem);
		}

		/// <summary>
		/// Este método escreve a data atual no Console.
		/// </summary>
		public void EscreveData()
		{ Console.WriteLine(DateTime.Now.ToShortDateString()); }

		/// <summary>
		/// Este método escreve a hora atual no Console.
		/// </summary>
		public void EscreveHora()
		{ Console.WriteLine(DateTime.Now.ToShortTimeString()); }
	}

}
```

Agora que já fizemos a documentação da classe e de seus membros, vamos ver no que isto pode ser interessante, ou melhor, muito interessante.

Volte para o arquivo de seu projeto chamado Class1.cs.

Altere o código deste arquivo para que ele fique da seguinte maneira:

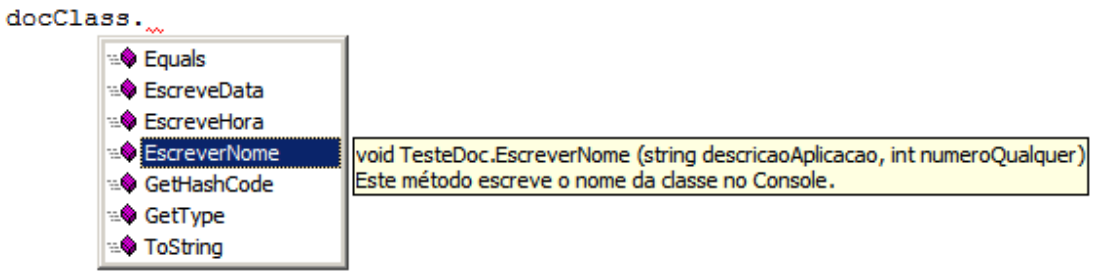
```
using System;

namespace XMLDoc
{
	/// <summary>
	/// Summary description for Class1.
	/// </summary>
	class Class1
	{
		/// <summary>
		/// The main entry point for the application.
		/// </summary>
		[STAThread]
		static void Main(string[] args)
		{
			TesteDoc docClass = new TesteDoc();
		}
	}
}
```

Insira logo depois da sintaxe TesteDoc docClass = new TesteDoc(); o seguinte código:

```
docClass
```

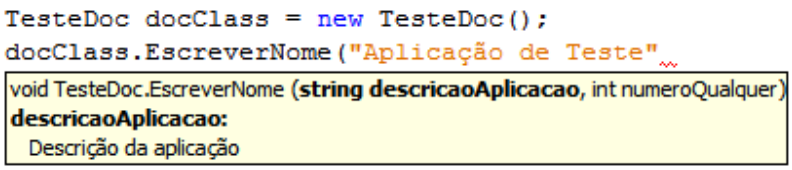
e pressione a tecla "." (ponto). Note que o Intelisense entrou em ação.

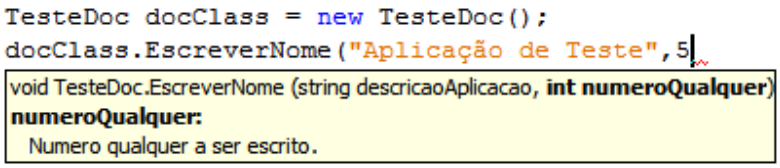


Mova o cursor por sobre os métodos que você criou e note que a descrição nas tags XML que você colocou está vindo como uma descrição sobre o funcionamento do método.

Isto é extremamente interessante, pois em muitos casos dispensa que uma documentação impressa seja consultada a todo momento e facilita bastante a programação.

Continue escrevendo conforme as imagens seguintes.





Veja a descrição explicativa dos parâmetros, é simplesmente incrível!

Deixe o código do arquivo Class1.cs da seguinte maneira (para ficar funcional):

```
using System;


namespace XMLDoc
{
	/// <summary>
	/// Summary description for Class1.
	/// </summary>
	class Class1
	{
		/// <summary>
		/// The main entry point for the application.
		/// </summary>
		[STAThread]
		static void Main(string[] args)
		{
			TesteDoc docClass = new TesteDoc();
			docClass.EscreverNome("Aplicação de Teste",5);
			docClass.EscreveData();
			docClass.EscreveHora();
		}
	}
}
```

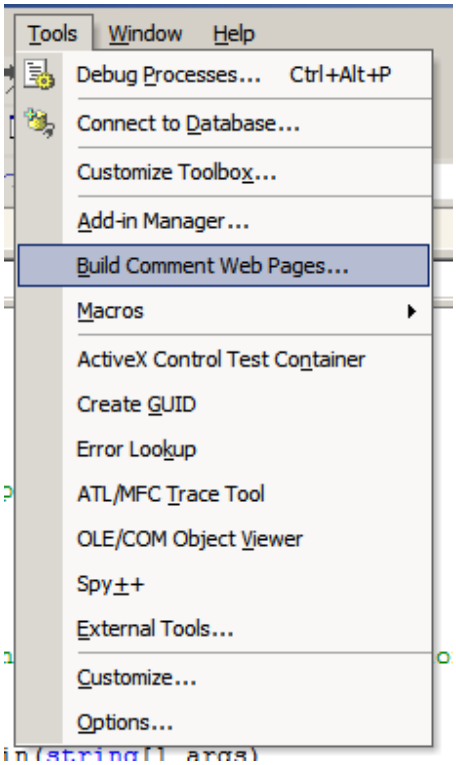
Apesar de isto ser uma novidade extremamente fantástica, ainda não é a melhor parte do que isto pode nos proporcionar.

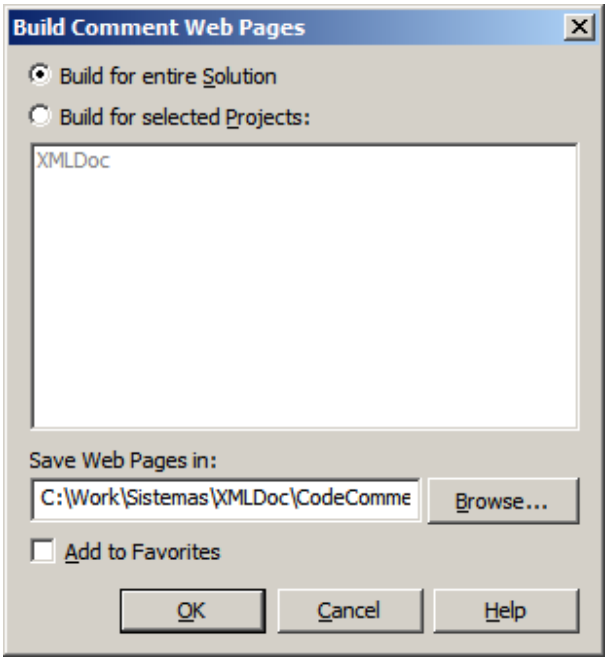
Imagine depois de ter terminado de desenhar sua classe, você pode querer disponibilizar uma documentação On line, muito apresentável, e que fornecesse informações suficientes para que qualquer desenvolvedor consiga compreender sua aplicação... Imaginou? Bom, voltemos para o mundo real.

O Visual Studio.NET faz tudo isto para você (parece propaganda, não acha?). E o melhor, sem você ser obrigado a digitar uma linha de HTML ou javascript ou alguma outra destas linguagens de Internet.

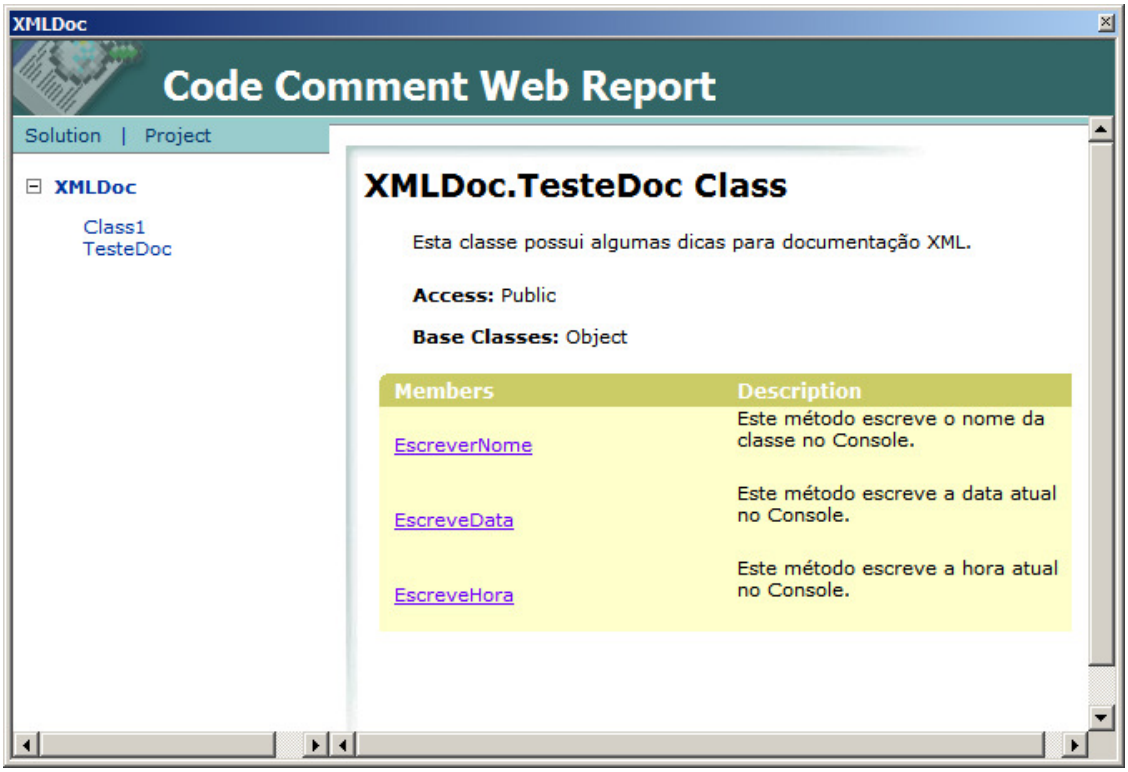
Agora que você escreveu suas tags XML dentro do código e que a Microsoft realmente adotou o XML, vamos gerar nossa documentação "ideal" com alguns cliques.

Siga os passos sugeridos nas imagens:

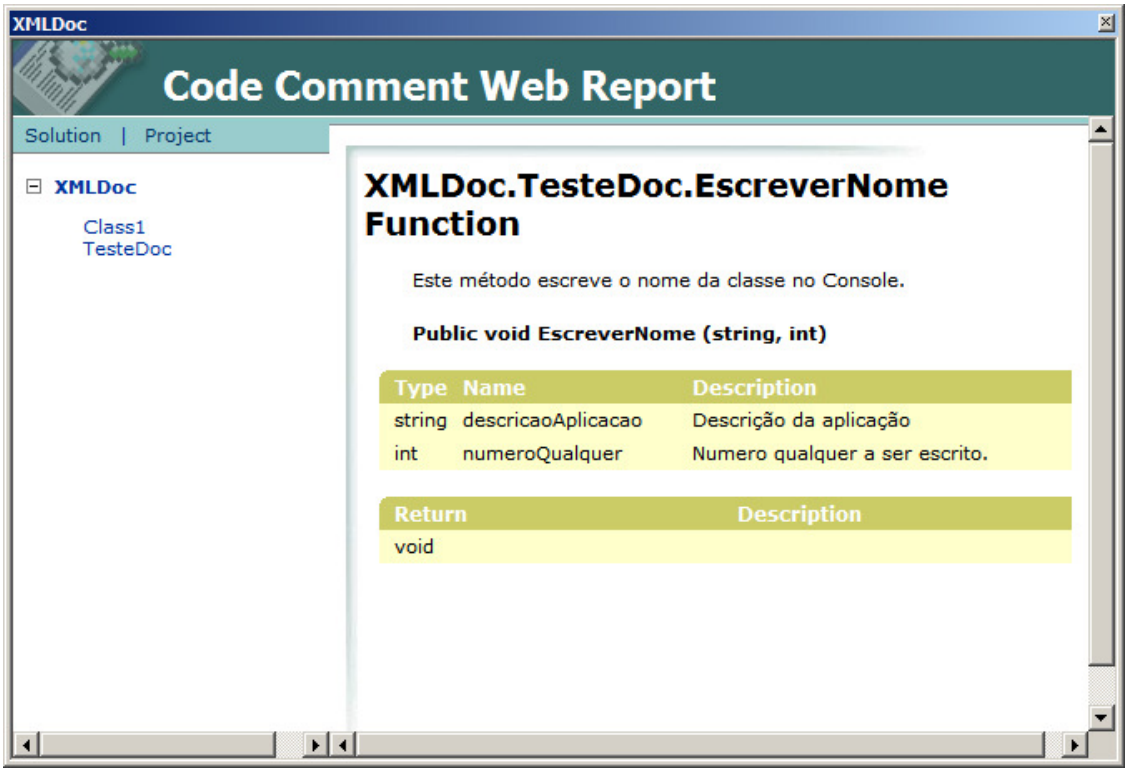




Ao clicar sobre o botão OK desta janela, a documentação será gerada no diretório indicado em *Save Web Pages In*:



Tudo o que descrevemos logo acima está pronto e funcionando.



Esta documentação pode lhe ser de muita valia. Por isso é uma boa prática escrever a documentação do código com tags XML. Vou colocar isso em um livro de Design Patters ☺.

Fabio Camara, MCP – Trabalha na CRK Financial Business como Technical Manager, sendo responsável pelo desenvolvimento de projetos como o SPB, principalmente os módulos de mensageria e criptografia. Escreveu os livros "Projetos com Windows DNA e .NET" e "Orientação a Objeto com .NET" dentre outros, editados pela Visual Books Editora. Pode ser contatado pela home page C# Br (http://www.csharpbr.com.br).