

Administração Central
Cetec Capacitações

Apostila de ASP.NET

Administração Central Cetec Capacitações

1. Programando classes com C#

Devemos lembrar que uma classe é composta por atributos e métodos, ou seja, vamos definir suas características e suas ações.

Primeiro passo é definir como será a classe e qual a sua funcionalidade, com isso, poderemos definir suas características e suas ações.

Pensando nisso, vamos criar uma classe **OPERACOES**, como mostra a figura 1.

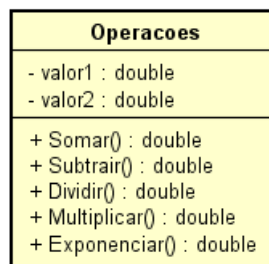


Figura 1 – Diagrama UML da classe Operacoes

1.1. Criando o projeto de Classe

Para criarmos um projeto de biblioteca (classes) no Visual Studio devemos selecionar através da template **Windows**, a opção **Class Library**, como mostra a figura 2.

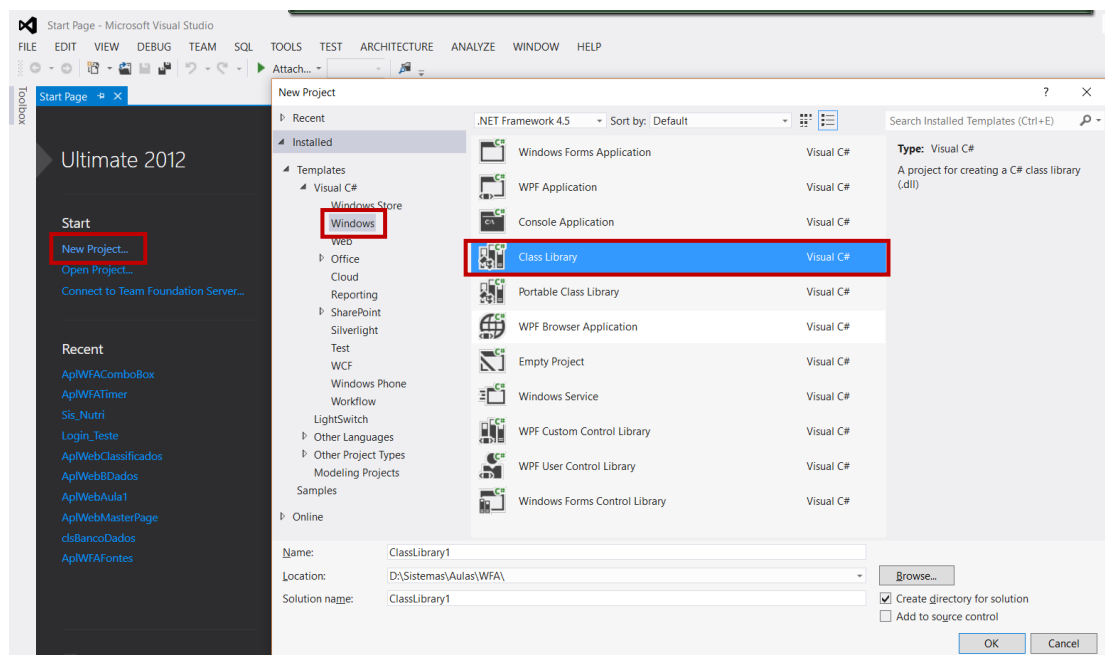


Figura 2 - Criando uma classe

Administração Central Cetec Capacitações

Após a seleção, na parte inferior da tela, devemos inserir o nome do projeto da classe, como mostra a figura 3.

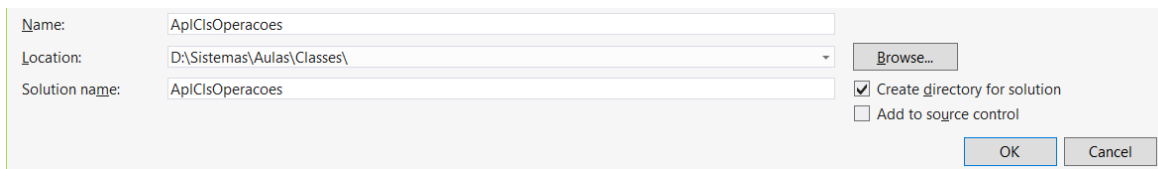
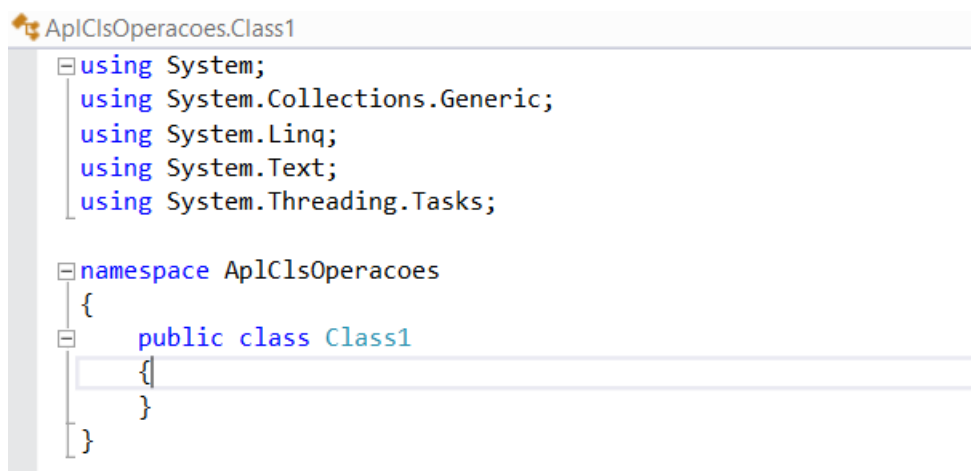


Figura 3 - Inserindo o nome da classe

Em seguida teremos o ambiente de desenvolvimento da classe, dentro do Visual Studio, conforme mostra a figura 4. Com isso, podemos criar os atributos e seus métodos.



```

AplClsOperacoes.Class1
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

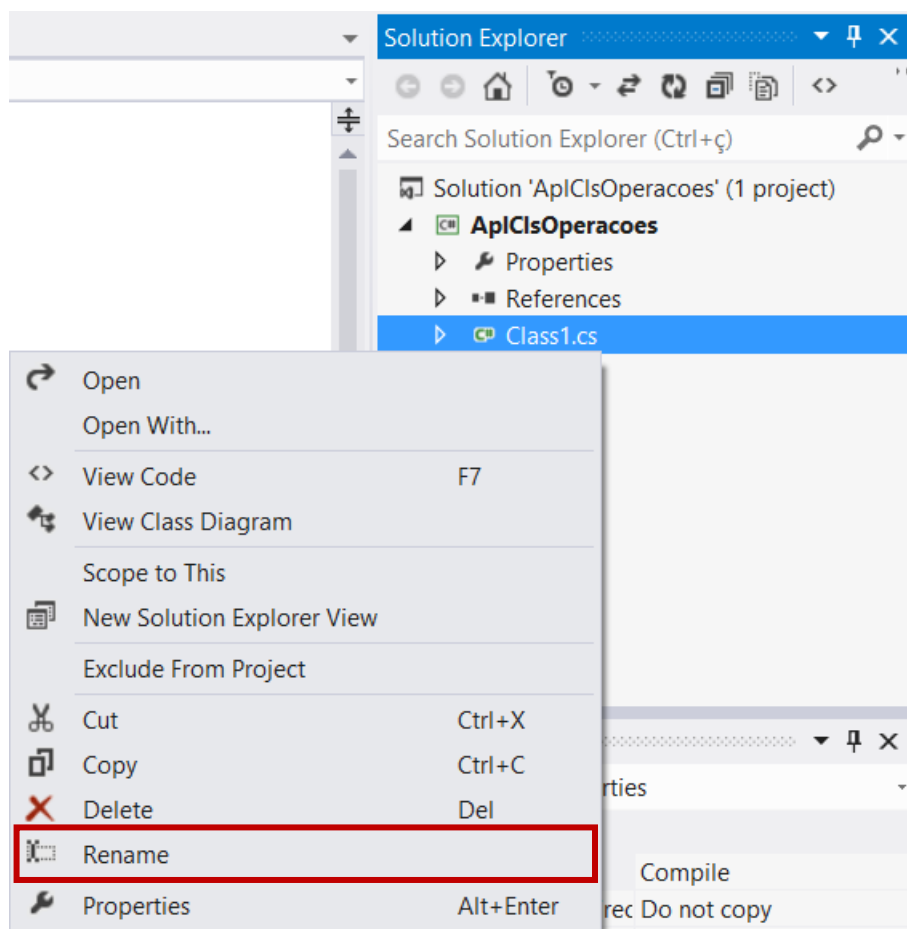
namespace AplClsOperacoes
{
    public class Class1
    {
    }
}
    
```

Figura 4 – Ambiente de programação da classe

Por default a classe tem o nome **Class1**, portanto iremos modificá-la para **Operacoes**, conforme a definição da figura 1.

Para realizar esta alteração devemos clicar sobre o objeto que está na caixa de solução do projeto (**Solution Explorer**), e em seguida clicamos com o botão direito do mouse, como mostra a figura 5.

Administração Central Cetec Capacitações



Neste menu, iremos selecionar a opção **Rename**, e assim alterar o nome da classe para **Operacoes**. Caso apareça a caixa de mensagem conforme a figura 5, clicar no botão **Sim** para que o Visual Studio possa alterar todas as referências da classe antiga para a classe nova.

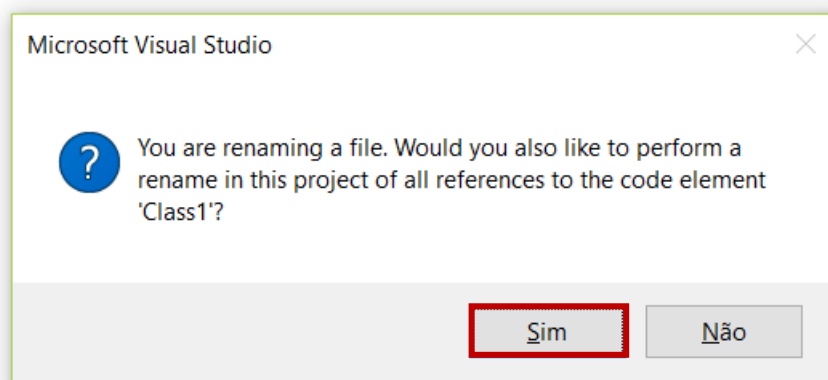


Figura 5 - Tela de confirmação de alteração de nome da classe

Administração Central
Cetec Capacitações

Após os procedimentos acima mencionados, sua classe deve estar conforme a figura 6.

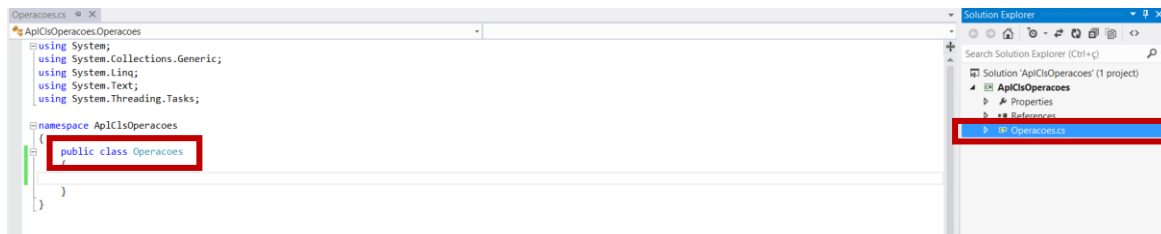


Figura 6 - Nome da classe alterada para Operacoes

1.2. Inserindo atributos/métodos na classe

Antes inserirmos os atributos devemos conhecer as instruções **set {}** e **get {}**.

1.2.1. Atributo set {}

Utiliza-se esta instrução quando queremos que o atributo e/ou método receba valores de fora do objeto, ou seja, o sistema irá enviar dados que o usuário digitou, por exemplo:

```
Operacoes objOpera = new Operacoes();  
//O atributo valor1 recebe o valor do usuário  
objOpera.valor1 = 10;  
//O atributo valor2 recebe o valor do usuário  
objOpera.valor2 = 20;
```

1.2.2. Atributo get {}

Utiliza-se esta instrução quando queremos que o atributo e/ou método devolva valores do objeto para o sistema, ou seja, o objeto irá enviar dados que o método calculou e enviará de retorno ao sistema, por exemplo:

```
Operacoes objOpera = new Operacoes();  
objOpera.valor1 = 10;  
objOpera.valor2 = 20;
```

Administração Central
Cetec Capacitações

//A variável dbISoma recebe o resultado da classe,
através do método Somar();
dbISoma = objOpera.Somar();

Podemos criar nossos atributos de duas formas:

1ª Forma – Declarações Independentes dos atributos

Os atributos da classe **Operacoes** serão privados e o método/propriedade **Somar** terá suas ações independentes, como mostra a figura 7.

```
public class Operacoes
{
    private double valor1, valor2;

    public double Somar
    {
        set {
            this.valor1 = value;
        }
    }

    public double Somar
    {
        get
        {
            return this.valor1;
        }
    }
}
```

Figura 7 – 1ª Forma - Declarações independentes dos atributos

2ª Forma – Declarações Únicas dos atributos

Os atributos da classe **Operacoes** deverão ser públicos e poderão receber/enviar dados, como mostra a figura 8.

Administração Central Cetec Capacitações

```
public class Operacoes
{
    public double valor1 { get; set; }
    public double valor2 { get; set; }

    public double Somar()
    {
        return this.valor1 + this.valor2;
    }
}
```

Figura 8 - Declarações Únicas dos atributos

Quando as classes estiverem finalizadas, devemos gerar a biblioteca, ou seja, o arquivo **.dll**. Para isso, devemos clicar com o botão direito sobre a aplicação (**ApIClsOperacoes**) e selecionar a opção **Build**, como mostra a figura 9.

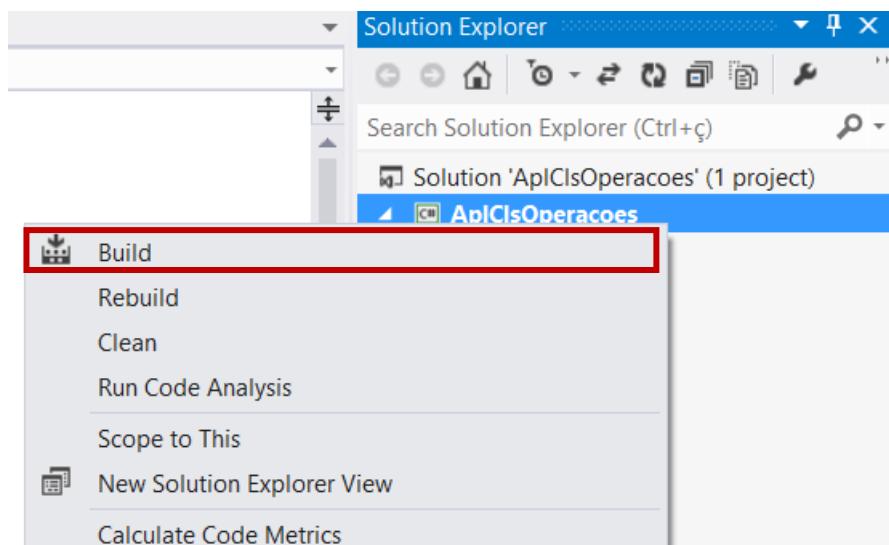


Figura 9 - Build da classe Operacoes

Após a seleção da opção, no canto inferior do ambiente de programação, aparecerá o início e fim da construção da sua biblioteca, como mostra a figura 10.

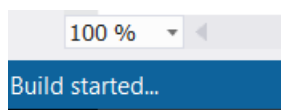


Figura 10 – Build da classe inicializado

Administração Central
Cetec Capacitações

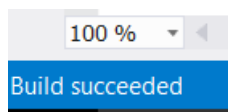


Figura 11 - Build da classe finalizado

A biblioteca foi gerada na pasta Debug do projeto, como mostra a figura 12.



Dados (D:) > Sistemas > Aulas > Classes > AplClsOperacoes > AplClsOperacoes > bin > Debug				
Nome	Data de modificaç...	Tipo	Tamanho	
 AplClsOperacoes.dll	01/11/2016 16:07	Extensão de aplica...	5 KB	
 AplClsOperacoes.pdb	01/11/2016 16:07	Program Debug D...	12 KB	

Figura 12 - Local da biblioteca gerada (.dll)

1.3. Instanciando a classe

Primeiramente devemos criar um projeto novo do tipo **ASP.NET Empty Web Application**.

Neste projeto novo vamos utilizar a classe que acabamos de criar. Antes de utilizar os recursos da classe, é preciso importá-la ao projeto. Para adicionar uma classe externa, ou seja, adicionar uma referência externa, devemos seguir os seguintes passos:

1. Na **Solution Explorer**, ao lado direito da tela, devemos clicar com o botão direito do mouse na opção **References**, e aparecerá a tela conforme mostra a figura 13:

Administração Central Cetec Capacitações

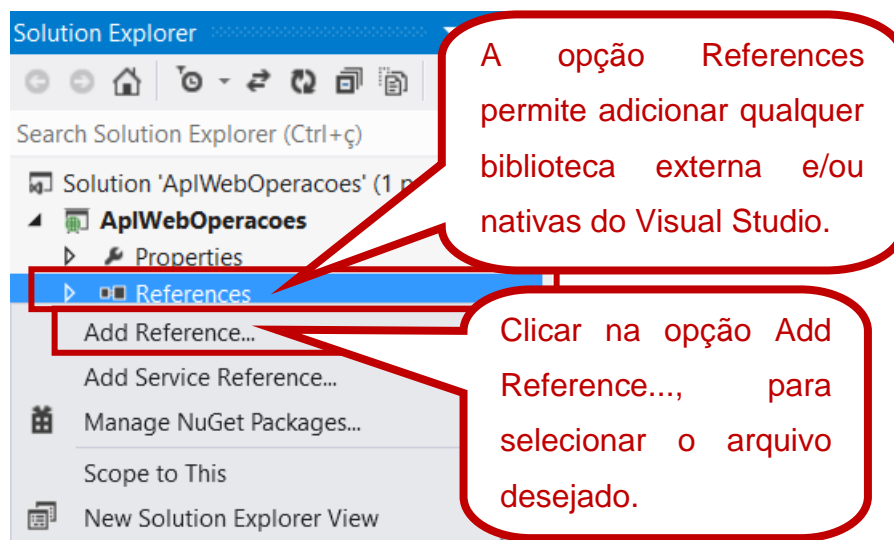


Figura 13 - Adicionando uma referência (Add Reference...)

2. Após clicar no botão **Add References...**, aparecerá a tela, conforme mostra a figura 14:

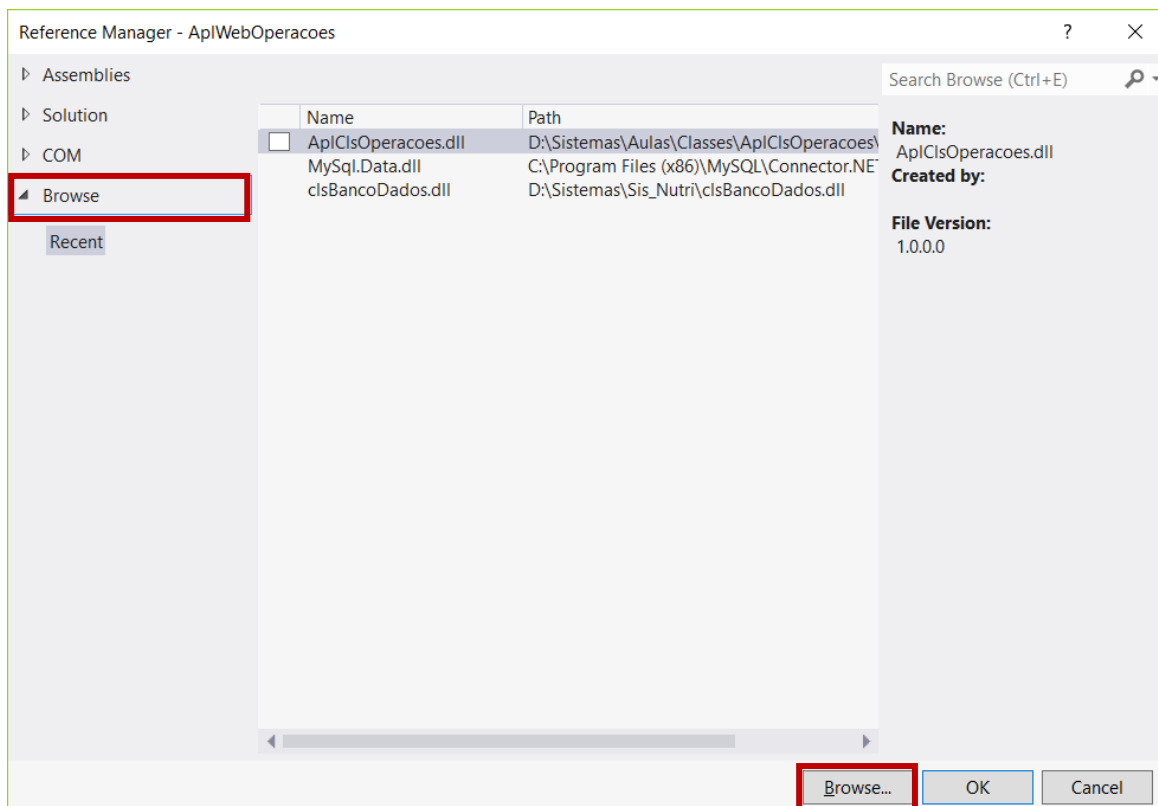


Figura 14 -- Busca de Referências externas (Add Rederences...)

Na opção Browse, mostrará todas as bibliotecas que já foram incluídas anteriormente no Visual Studio instalado

Administração Central Cetec Capacitações

do computador.

3. Após clicar no botão **Browse**, aparecerá a tela como mostra a figura 15:

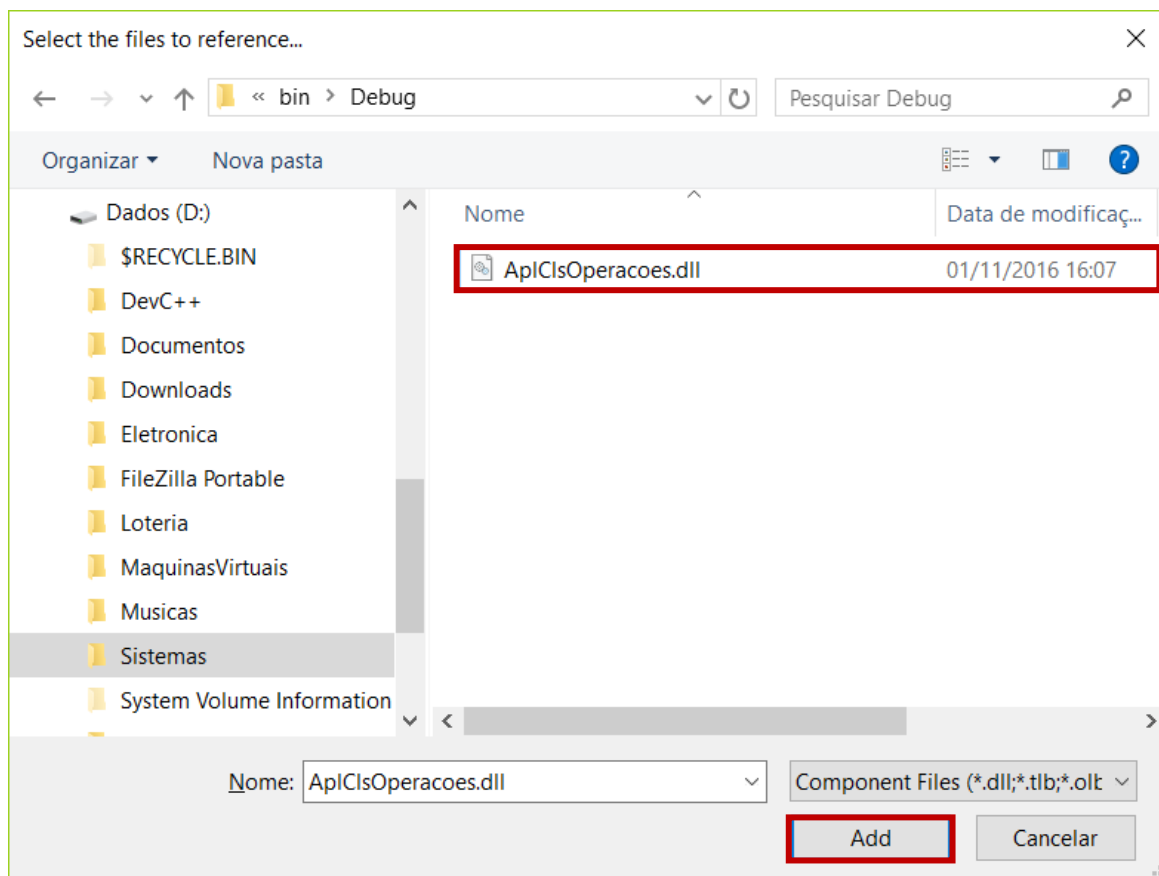


Figura 15 - Selecionando uma referência externa (Add References...)

4. Após a seleção do arquivo **(.dll)**, irá aparecer no gerenciamento de referências, como mostra a figura 16:

Administração Central Cetec Capacitações

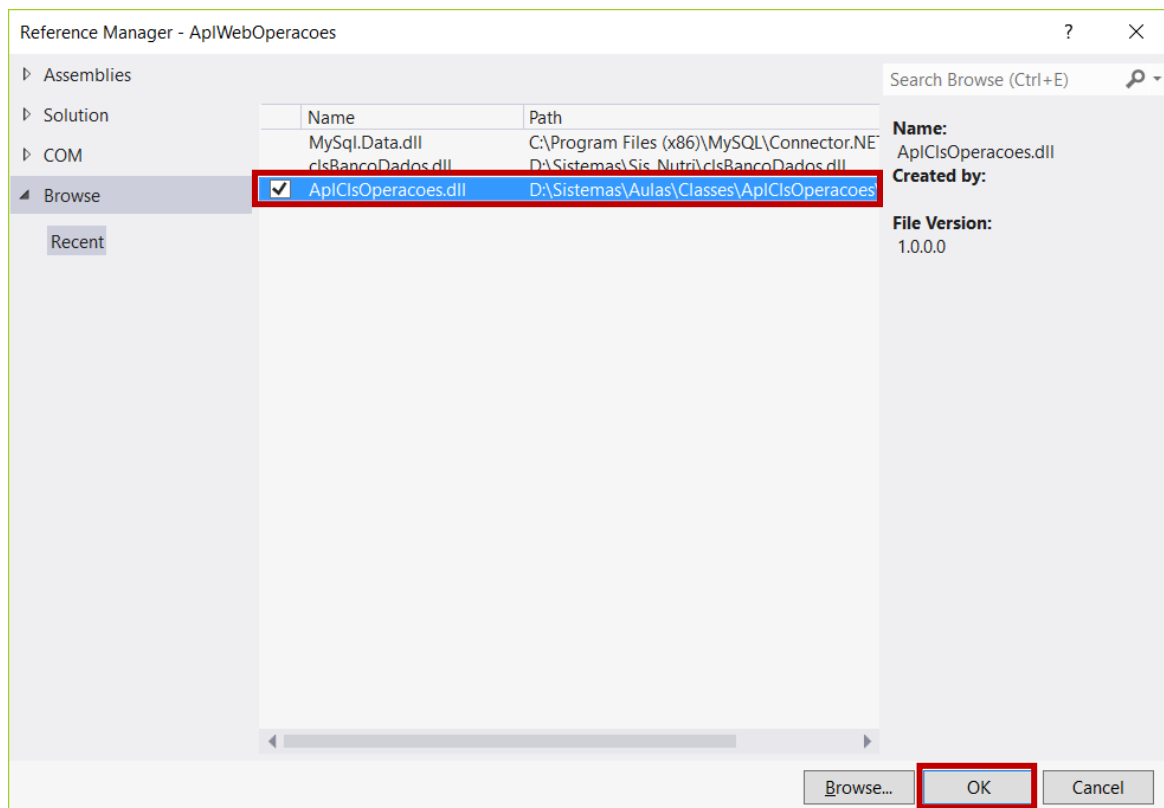


Figura 16 - Confirmando a seleção uma referência externa (Add References...)

a. Em seguida clicar no botão **OK**.

Após a seleção do arquivo (.dll), o mesmo irá aparecer na relação de referências, como mostra a figura 17:

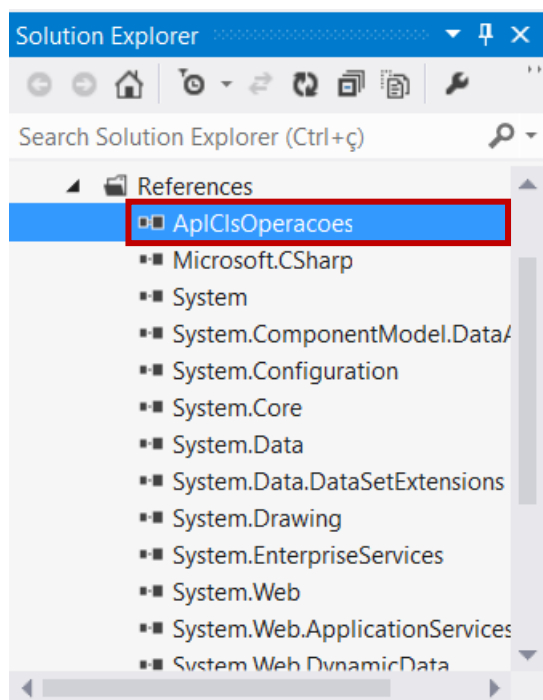
Administração Central
Cetec Capacitações

Figura 17 - Relação de Referências (References)

A partir de agora, a biblioteca está pronta para ser usada no projeto.

Sempre que abrir um projeto no qual a referência AplClsOperacoes.Dll foi inserida, não será mais necessário adicioná-la.

1.3.1.Importando a biblioteca ao ambiente de desenvolvimento

Para importarmos a biblioteca para o ambiente de desenvolvimento, devemos utilizar a instrução **using**, como mostra a figura 18:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.UI;  
using System.Web.UI.WebControls;  
using AplClsOperacoes;
```

Figura 18 - Importando classe AplClsOperacoes

Administração Central
Cetec Capacitações

Não esqueçam, sem adicionar a referência no projeto, NÃO será possível importar a classe `ApIClsOperacoes`

Após os procedimentos de importação, agora iremos instanciar a classe em nosso projeto, como mostra a figura 19.

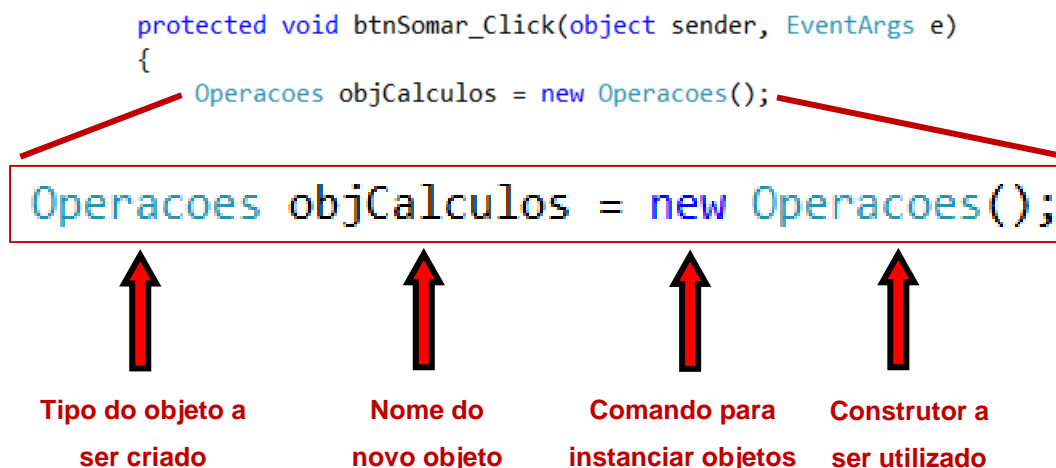


Figura 19 - Instanciando uma classe

Após o objeto instanciado, agora devemos enviar os dados do usuário para o objeto (classe), como mostra a figura 20.

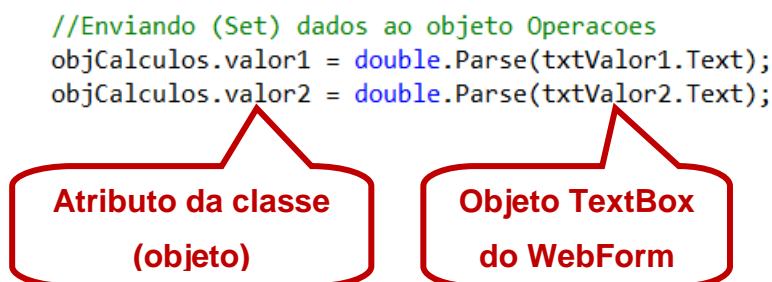


Figura 20 - Enviando dados para o objeto instanciado

Após o envio de todos os dados necessário ao objeto, podemos solicitar os dados de retorno através dos métodos de retorno (get), como mostra a figura 21.

Administração Central
Cetec Capacitações

```
//Recebendo (Get) dados do objeto Operacoes  
lblResultado.Text = objCalculos.Somar().ToString();
```

**Objeto Label do
WebForm**

**Método (get) da
classe (objeto)**

O **ToString()**, é um método comum a todos os objetos do C# dentro do Visual Studio, e tem como função converter qualquer conteúdo para string.