

SCRIPT

Não é a linguagem de programação que define o programador,
mas sim sua lógica.

David Ribeiro Guilherme

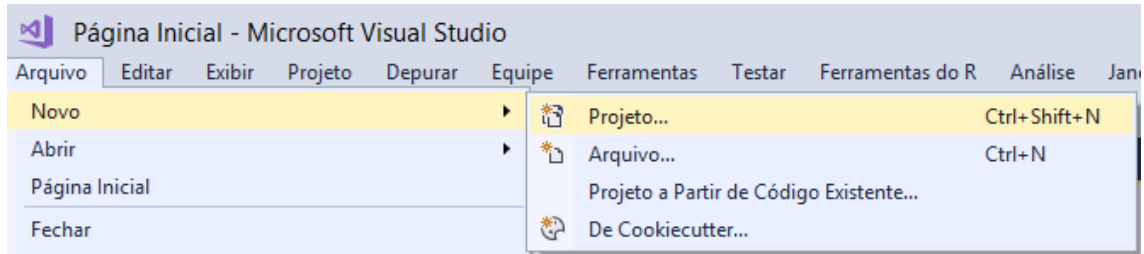
(https://www.pensador.com/frases_de_programador/)

C#

Criando um projeto (ambiente de trabalho)

Criando um projeto no Visual Studio 2017

Clique em Arquivo / Novo / Projeto

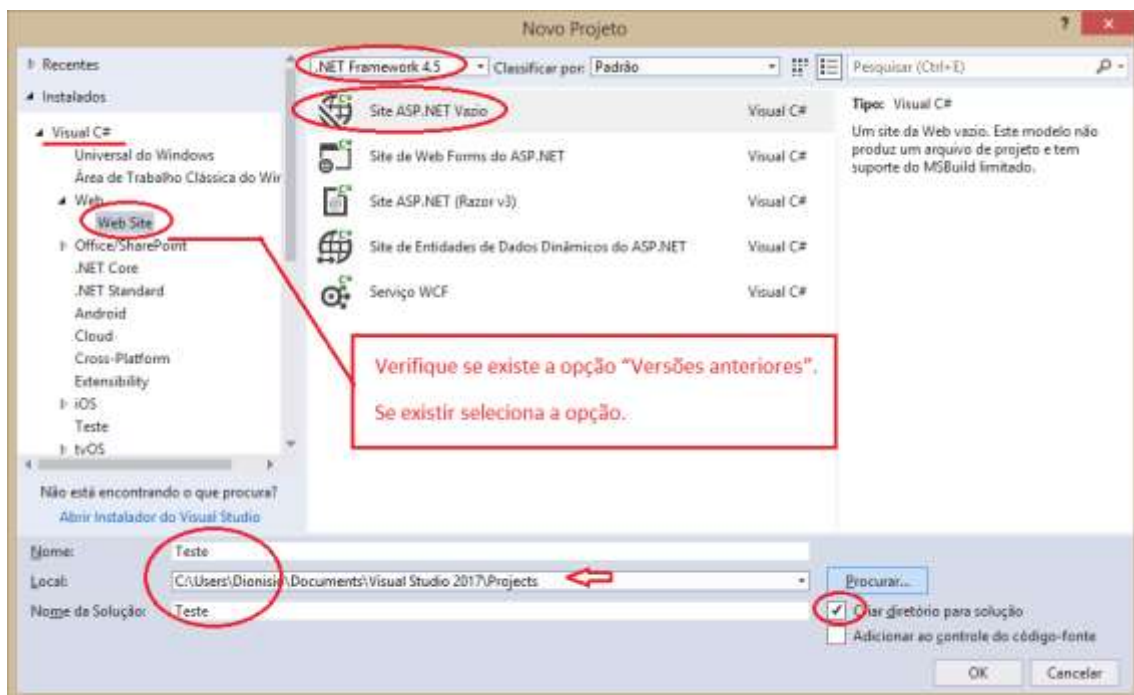


Selecione:

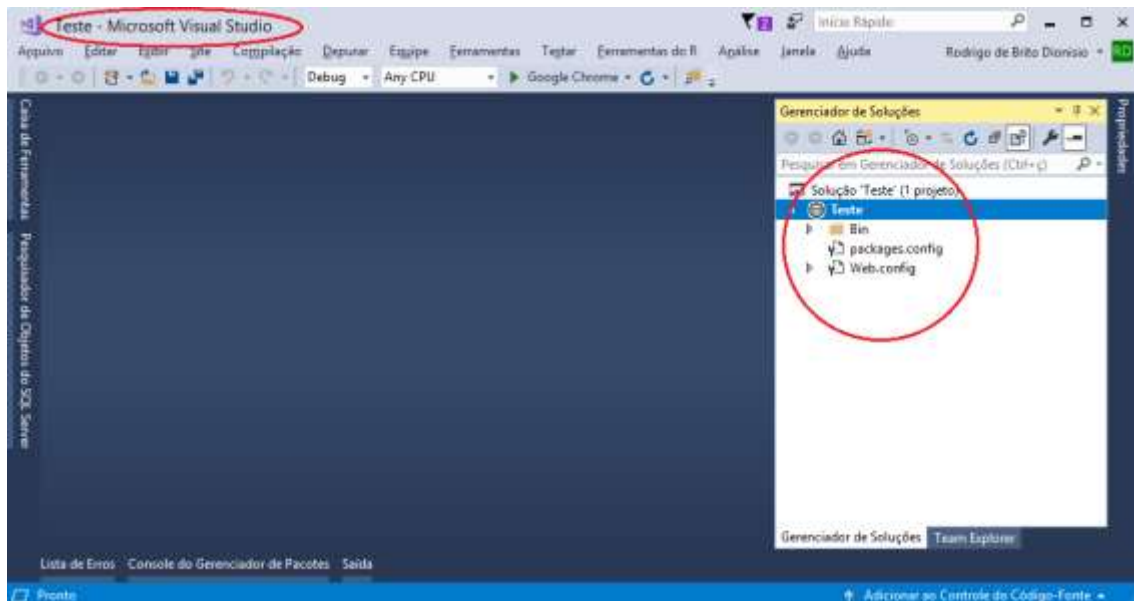
- Visual C#
- Web Site
 - Verifique se não aparece a opção Versões Anteriores
- Site ASP.NET Vazio

Observe:

- Versão: .NET Framework 4.5
- Nome do projeto
- Local onde será salvo
- Deixe habilitado "Criar diretório de solução"

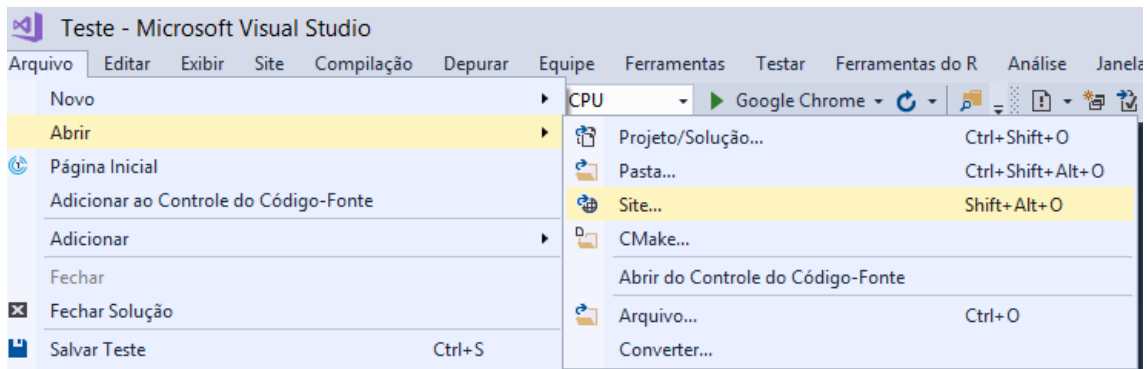


Observe que o projeto foi criado.



Abrindo o projeto no Visual Studio 2017

Clique em: Arquivo / Abrir / Site

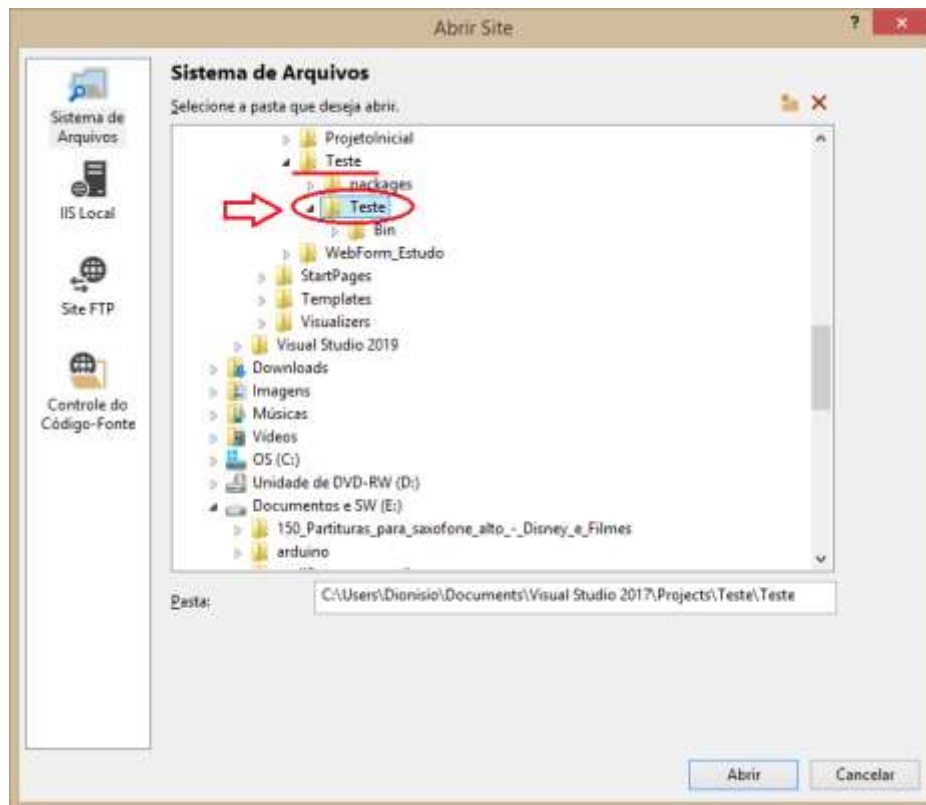


Na caixa de diálogo que se abre procure o local onde você salvou o projeto.

Observe que você verá duas pastas com o mesmo nome.

Selecione a segunda pasta com o nome de seu projeto.

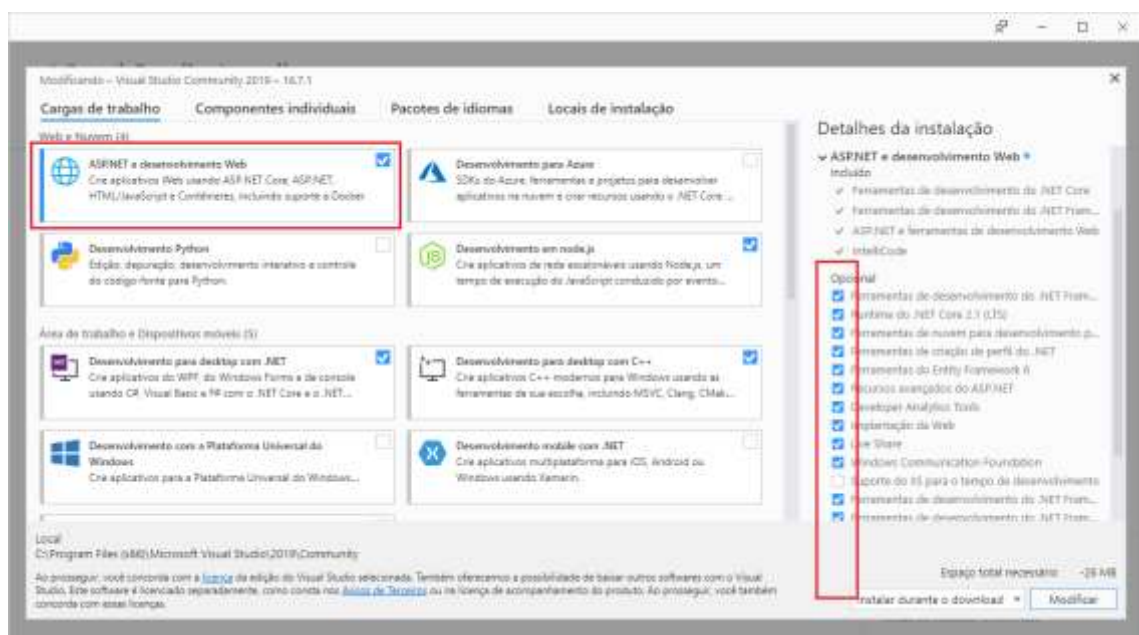
E clique em abrir.



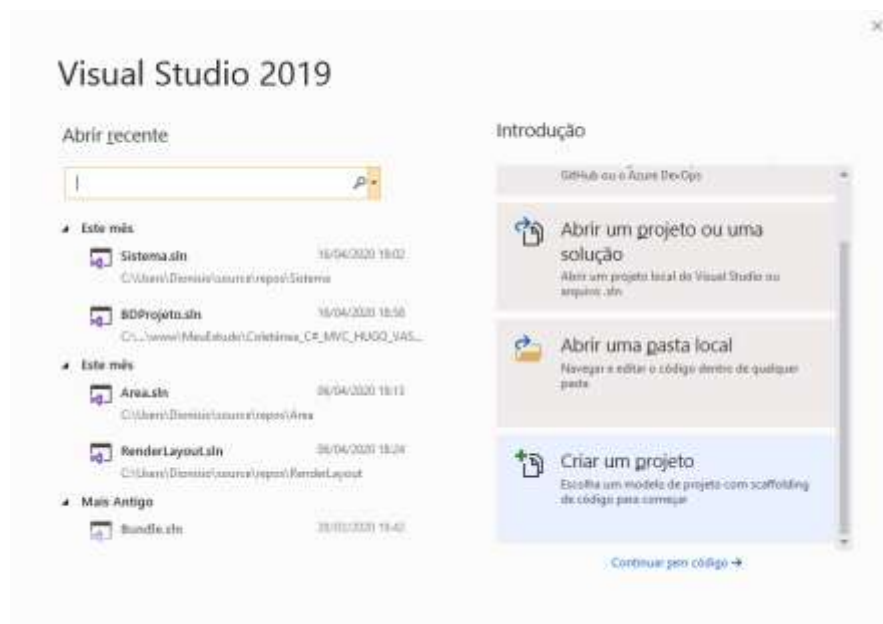
Criando um projeto no Visual Studio 2019

Atenção:

- Verifique se você instalou todas as extensões necessárias.



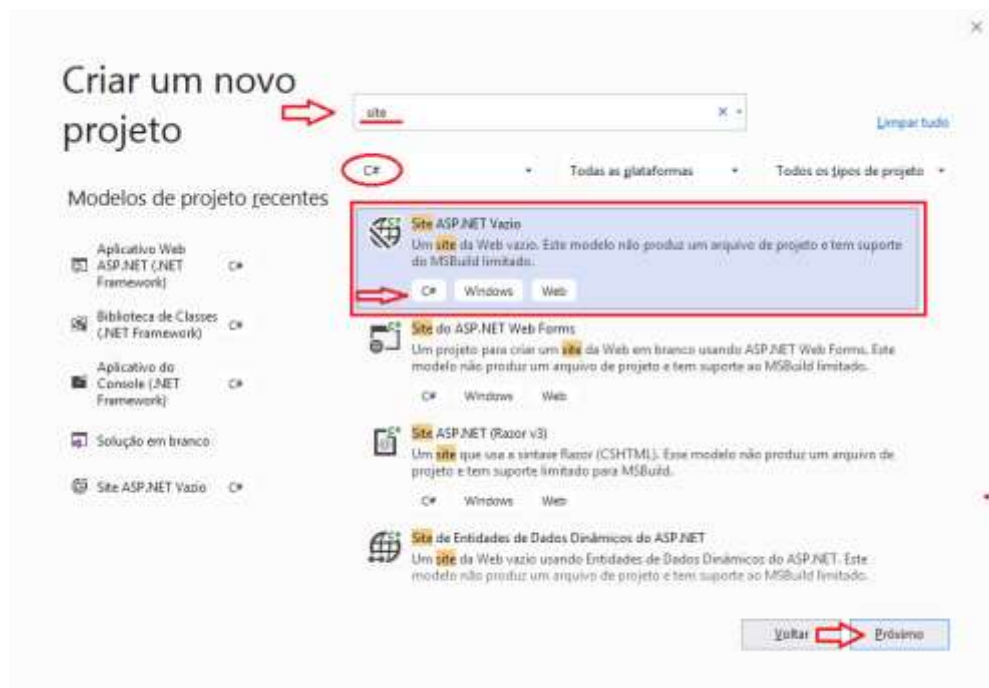
Clique em “Criar um projeto”



Na caixa de diálogo que se abre procure por “Site”.

Não esqueça de selecionar “C#”

Selecione “Site ASP.NET Vazio”, observe a opção “C#”.



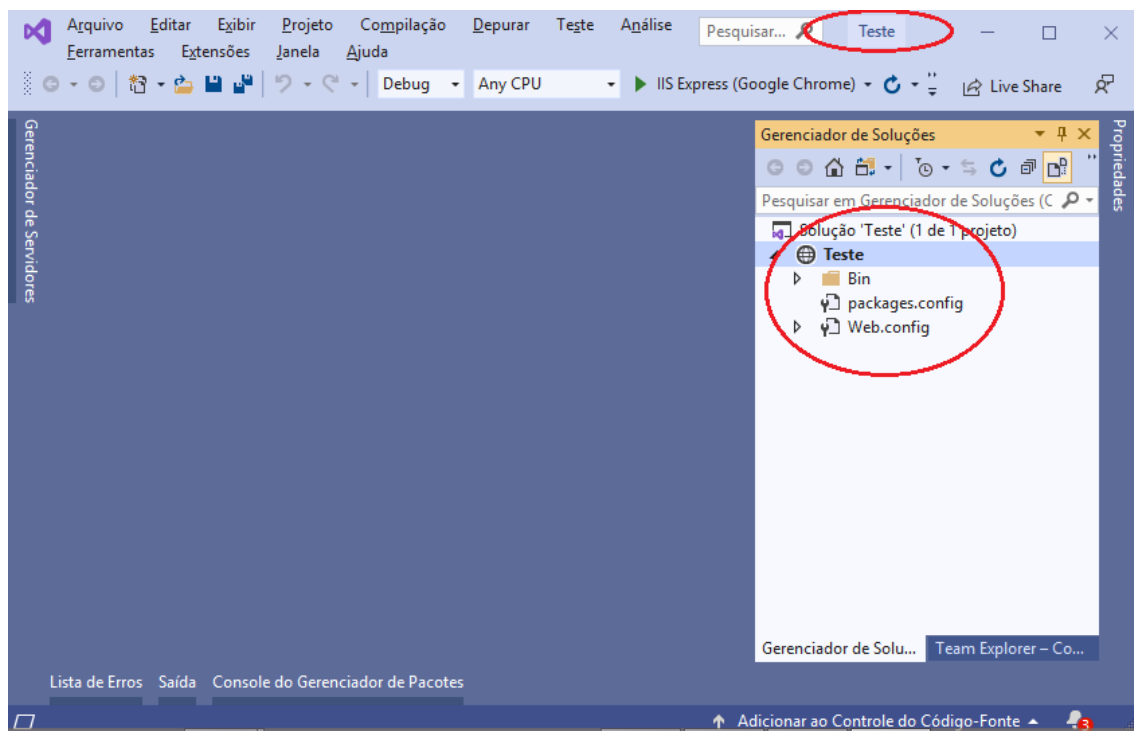
Na próxima janela de diálogo:

- Coloque o nome do projeto
- Escolha o local para salvar o projeto
- Ative a opção “Colocar a solução e o projeto no mesmo diretório”
- Atenção para a versão do Framework “.NET Framework 4.5.2”

- Clique em **Criar**

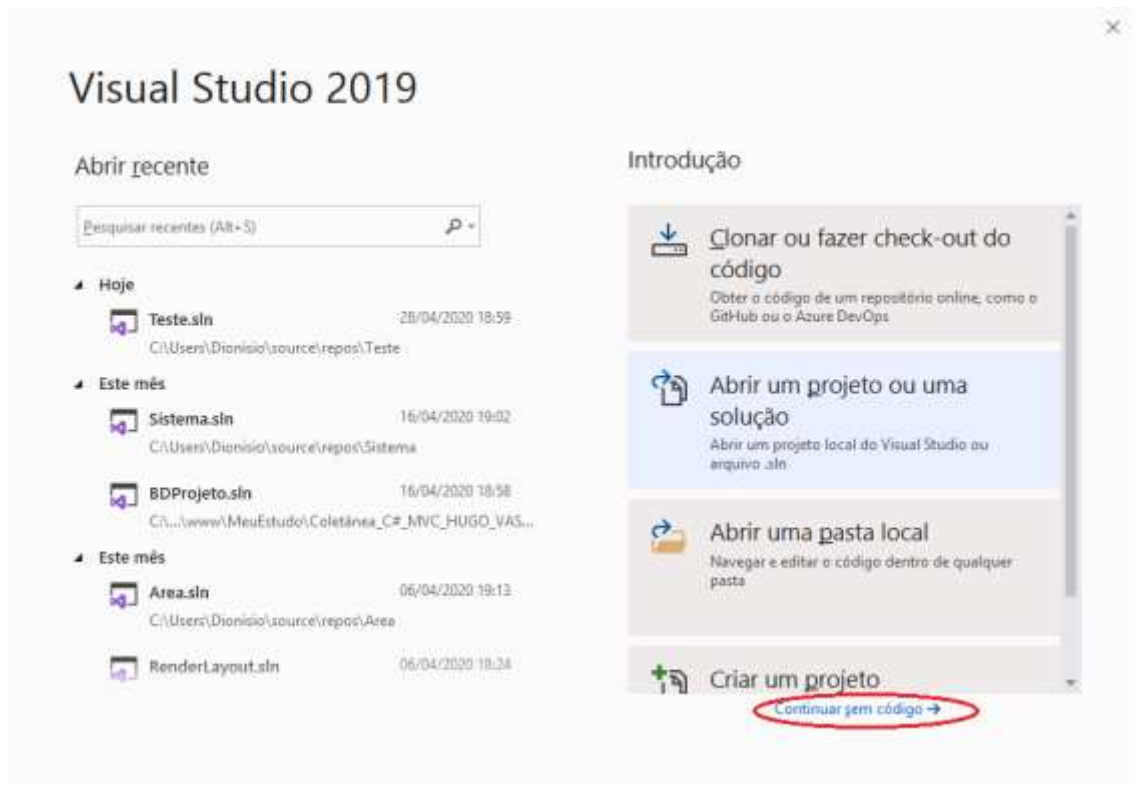


Observe que o projeto foi criado.

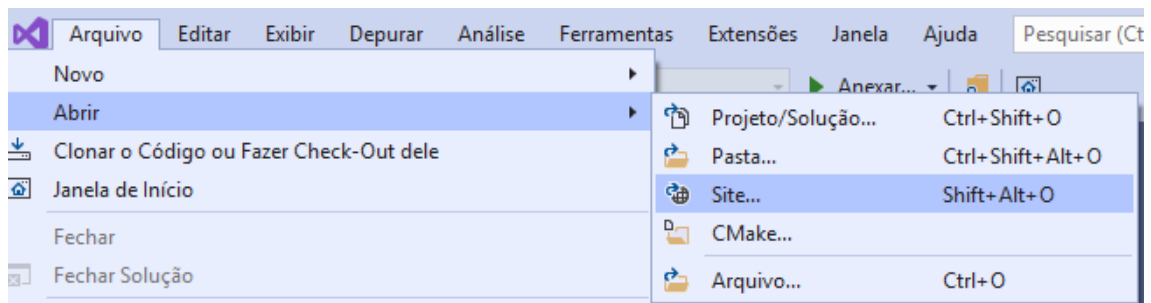


Abrindo o projeto no Visual Studio 2019

Clique em “Continuar sem código”



Clique em: **Arquivo / Abrir / Site...**

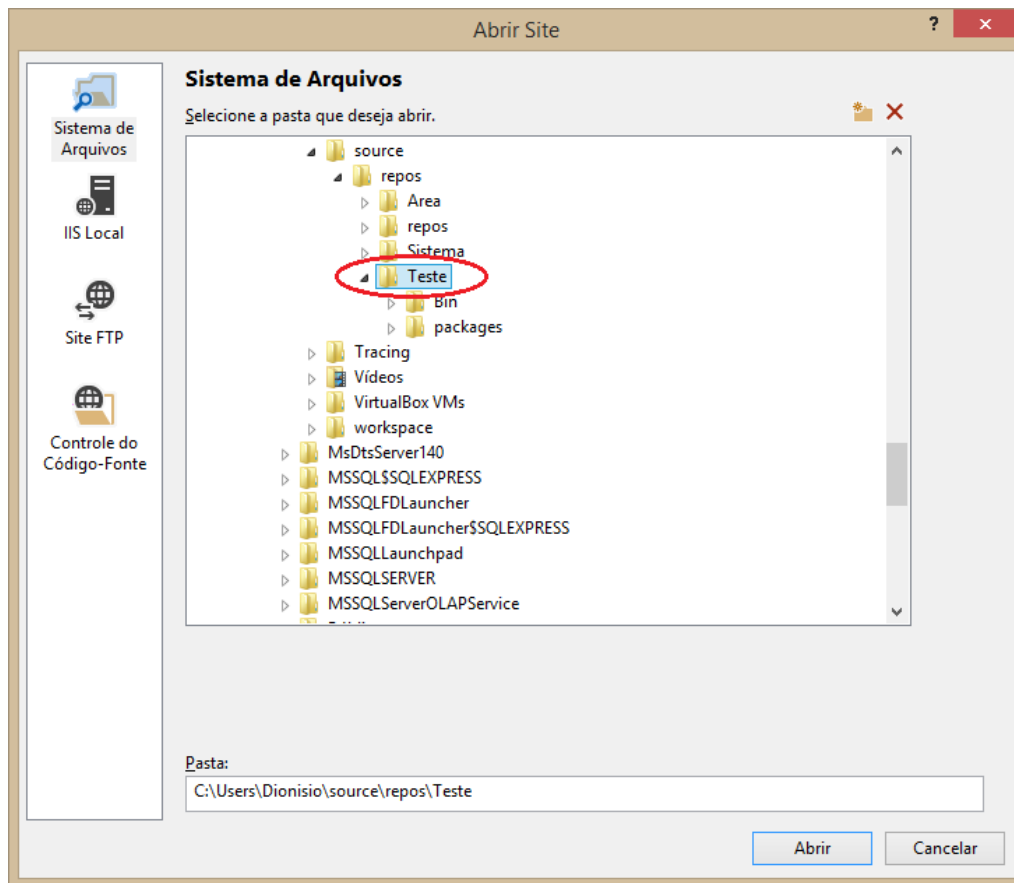


Na caixa de diálogo que se abre procure o local onde você salvou o projeto.

Observe que diferente da versão 2017 você encontrará apenas uma pasta com o nome de seu projeto.

Selecione a pasta.

E clique em abrir.

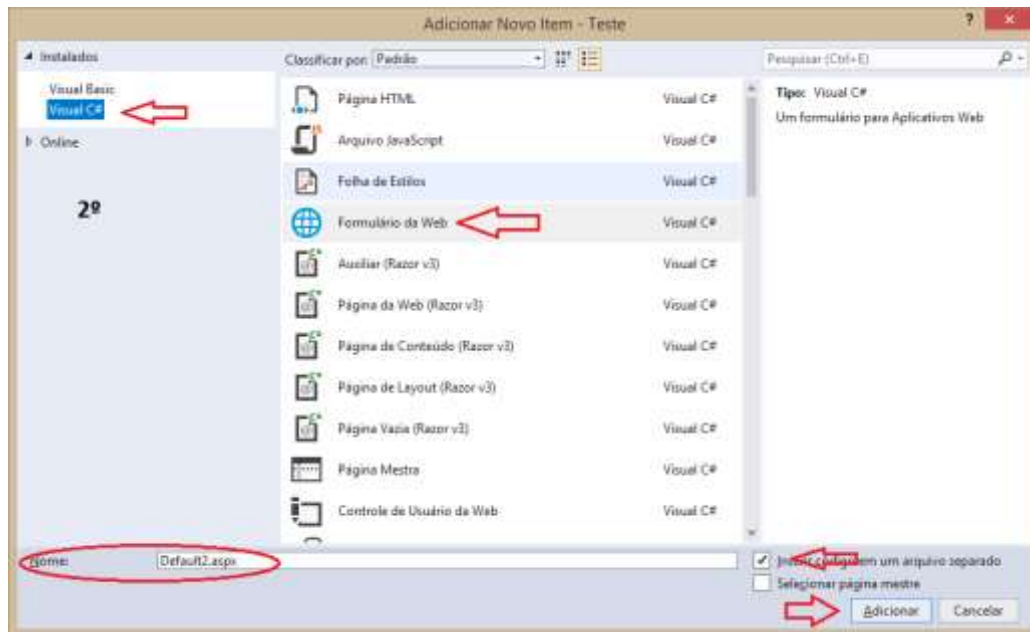


Criando uma página ASPX

Clique com o botão direito do mouse sobre o projeto.



Selecione



Testando a aplicação

Digite o texto a baixo e execute a aplicação.

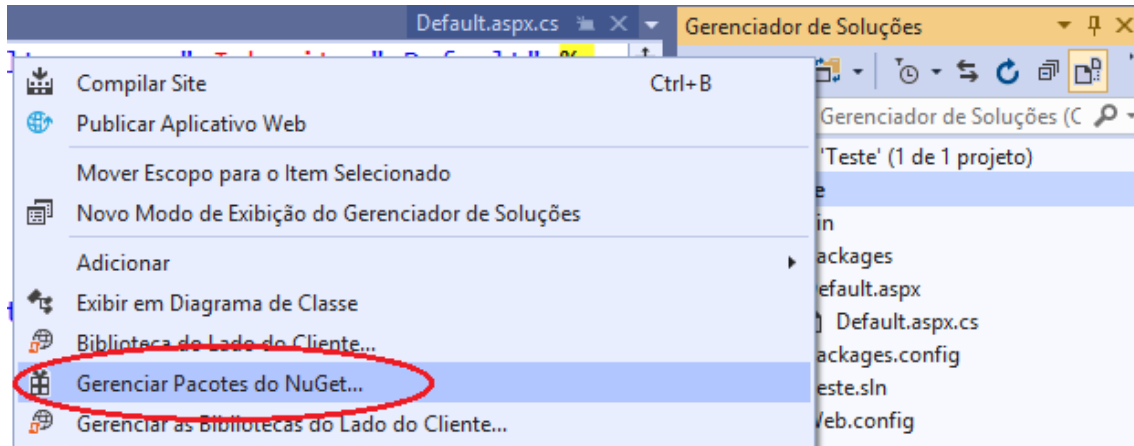


Primeira página ASPX

Incorporando o Bootstrap na aplicação

Clique com o botão direito do mouse sobre o projeto.

Observação: A aplicação não deve estar em execução.



Na aba **Procurar** digite "*bootstrap*", selecione a versão e clique em "Instalar".



Fazendo o link do arquivo ASPX com o Bootstrap

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
```

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head runat="server">
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
```

```
<title>Primeira Página</title>
```

```
<link href="Content/bootstrap.min.css" rel="stylesheet" />
```

```
</head>
```

```
<body>
```

```
<form id="form1" runat="server">
```

```
<div>
```

```
<h1>Primeira página ASPX</h1>
```

```
</div>
```

```
</form>
```

```
<script src="Scripts/jquery-3.0.0.min.js"></script>
```

```
<script src="Scripts/bootstrap.min.js"></script>
```

```
</body>
```

```
</html>
```

Execute a aplicação

Primeira página ASPX

Componentes ASPX

TextBox - Text

```
<asp:TextBox ID="TextBox1" runat="server" CssClass="form-control"></asp:TextBox>
```

TextBox - Text

TextBox - Number

```
<asp:TextBox ID="TextBox2" type="number" runat="server" CssClass="form-control"></asp:TextBox>
```

TextBox - Number

TextBox - Date

```
<asp:TextBox ID="TextBox3" type="date" runat="server" CssClass="form-control"></asp:TextBox>
```

TextBox - Date

abril de 2020

dom	seg	ter	qua	qui	sex	sáb
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2

TextBox - E-mail

```
<asp:TextBox ID="TextBox4" type="email" runat="server" CssClass="form-control"></asp:TextBox>
```

TextBox - E-mail

TextBox - URL

```
<asp:TextBox ID="TextBox5" type="url" runat="server" CssClass="form-control">
</asp:TextBox>
```

TextBox - URL

TextBox - Placeholder

```
<asp:TextBox ID="TextBox6" placeholder="Digite seu nome" runat="server"
CssClass="form-control"></asp:TextBox>
```

TextBox - Placeholder

TextBox - Required

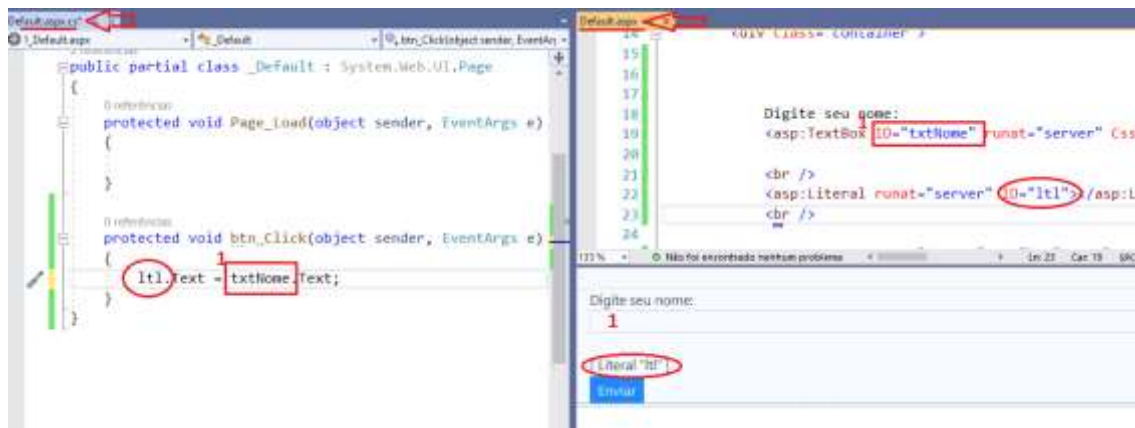
```
<asp:TextBox ID="TextBox7" required runat="server" CssClass="form-control">
</asp:TextBox>
```

TextBox - Required

Capturando dados de um TextBox

Observe que o “ID” do componente é muito importante, ele deve ser único no arquivo (não deve se repetir) e será utilizado para manipulá-lo no arquivo ASPX.CS.

Digite o código a seguir, não esqueça de criar o evento do botão automaticamente (com a aplicação parada, em modo design de um clique duplo sobre o botão).



Execute a aplicação

Digite seu nome:

Fulano

Enviar

DropDownList

```
<asp:DropDownList runat="server" ID="UF" CssClass="form-control">
  <asp:ListItem Selected="True" Value="-1">Selecione um estado</asp:ListItem>
  <asp:ListItem Value="SP">São Paulo</asp:ListItem>
  <asp:ListItem Value="RJ">Rio de Janeiro</asp:ListItem>
  <asp:ListItem Value="MG">Minas Gerais</asp:ListItem>
</asp:DropDownList>
```

Pegando os valores do DropDownList

using System;

```
public partial class _Default : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) { }

    protected void btn_Click(object sender, EventArgs e) {
```

```

        Itl.Text = UF.SelectedValue; // Pegando o valor
        Itl.Text += " - ";
        Itl.Text += UF.SelectedItem; // Pegando o texto
    }
}

```

The screenshot shows a web form with a dropdown menu containing 'Minas Gerais'. Below the dropdown is a label 'MG - Minas Gerais' and a blue button labeled 'Enviar'.

RadioButtonList - Vertical

Selecione um estado:

```

<asp:RadioButtonList runat="server" ID="UF">
    <asp:ListItem Value="SP" Selected="True" >São Paulo</asp:ListItem>
    <asp:ListItem Value="RJ">Rio de Janeiro</asp:ListItem>
    <asp:ListItem Value="MG">Minas Gerais</asp:ListItem>
</asp:RadioButtonList>

```

The screenshot shows a web form with the text 'Selecione um estado:' followed by a vertical list of radio buttons. The first option, 'São Paulo', is selected. The other options are 'Rio de Janeiro' and 'Minas Gerais'.

RadioButtonList - Horizontal

Selecione um estado:

```

<asp:RadioButtonList runat="server" ID="UF" RepeatDirection="Horizontal">
    <asp:ListItem Value="SP" Selected="True" >São Paulo</asp:ListItem>
    <asp:ListItem Value="RJ">Rio de Janeiro</asp:ListItem>
    <asp:ListItem Value="MG">Minas Gerais</asp:ListItem>
</asp:RadioButtonList>

```

The screenshot shows a web form with the text 'Selecione um estado:' followed by a horizontal list of radio buttons. The first option, 'São Paulo', is selected. The other options are 'Rio de Janeiro' and 'Minas Gerais'.

Pegando os valores do RadioButtonList

```
using System;
```

```
public partial class _Default : System.Web.UI.Page{  
    protected void Page_Load(object sender, EventArgs e) {  
  
        protected void btn_Click(object sender, EventArgs e) {  
            Itl.Text = UF.Selected.Value; // Pegando o valor  
            Itl.Text += " - ";  
            Itl.Text += UF.SelectedItem; // Pegando o texto  
        }  
  
    }  
}
```

Selecione um estado:

☐ São Paulo

☒ Rio de Janeiro

☐ Minas Gerais

RJ - Rio de Janeiro

Enviar

Colocando um evento no DropDownList

Para este exemplo vamos criar dois **Panels** e dentro deste panels vamos inserir os itens que achamos interessante.

Será um panel para itens masculino e um outro panel para itens femininos.

Posteriormente iremos criar um **DropDownList** e colocaremos um evento nele, quando selecionarmos uma opção o Panel referente a opção selecionada será exibido, ficando o outro oculto.

Para colocar um evento no DropDownList, vá em modo design (a aplicação não pode estar ativa) e dê um clique duplo sobre o componente. Depois ativo a opção de **AutoPostBack**.

Pronto, um método (parecido como o do botão) será criado.

Código ASPX


```

<asp:DropDownList runat="server" ID="ddlOpcao" CssClass="form-control"
AutoPostBack="true" OnSelectedIndexChanged="ddlOpcao_SelectedIndexChanged">
    <asp:ListItem Selected="True" Value="-1">Selecione...</asp:ListItem>
    <asp:ListItem Value="M">Artigos masculinos</asp:ListItem>
    <asp:ListItem Value="F">Artigos Femininos</asp:ListItem>
</asp:DropDownList>

```

```

<asp:Panel ID="panelNeuto" runat="server" Visible="false">
    <h2>Nenhum artigo selecionado</h2>
</asp:Panel>

```

```

<asp:Panel ID="panelMasculino" runat="server" Visible="false">
    <h2>Artigos Masculinos</h2>
</asp:Panel>

```

```

<asp:Panel ID="panelFeminino" runat="server" Visible="false">
    <h2>Artigos Femininos</h2>
</asp:Panel>

```

Código ASPX.CS

```
using System;
```

```

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // vamos exibir ou ocultar os panels
        panelNeuto.Visible = true;
        panelFeminino.Visible = false;
        panelMasculino.Visible = false;
    }

    protected void ddlOpcao_SelectedIndexChanged(object sender, EventArgs e)
    {
        switch (ddlOpcao.SelectedValue.ToString())
        {
            case "-1":
                panelNeuto.Visible = true;
                panelFeminino.Visible = false;
                panelMasculino.Visible = false;
                break;
            case "M":
                panelNeuto.Visible = false;
                panelFeminino.Visible = false;

```

```

        panelMasculino.Visible = true;
        break;
    case "F":
        panelNeuto.Visible = false;
        panelFeminino.Visible = true;
        panelMasculino.Visible = false;
        break;
    }
}
}

```

Visão do Browser

Selecione...
Nenhum artigo selecionado

Artigos masculinos
Artigos Masculinos

Artigos Femininos
Artigos Femininos

Colocando um evento no RadioButtonList

Para colocarmos um evento no RadioButtonList devemos seguir exatamente os mesmos passos descritos no tópico anterior (colocar um evento no DropDownList).

É exatamente igual...

Pegando os valores do CheckBoxList

ASPX

CheckBoxList

```
<asp:CheckBoxList runat="server" ID="chkSemana">
```

```

<asp:ListItem Value="Dom">Domingo</asp:ListItem>
<asp:ListItem Selected="True" Value="Seg">Segunda-feira</asp:ListItem>
<asp:ListItem Value="Ter">Terça-feira</asp:ListItem>
<asp:ListItem Value="Qua">Quarta-feira</asp:ListItem>
<asp:ListItem Value="Qui">Quinta-feira</asp:ListItem>
<asp:ListItem Value="Sex">Sexta-feira</asp:ListItem>
<asp:ListItem Value="Sab">Sábado</asp:ListItem>
</asp:CheckBoxList>

<hr />
<asp:Literal runat="server" ID="Itl"></asp:Literal>
<hr />

<asp:Button runat="server" ID="btn" CssClass="btn btn-primary" Text="Enviar"
OnClick="btn_Click" />

```

Visão do Browser

CheckBoxList

☐ Domingo

☒ Segunda-feira

☐ Terça-feira

☐ Quarta-feira

☐ Quinta-feira

☐ Sexta-feira

☐ Sábado

Enviar

ASPX.CS

```
using System;
```

```

public partial class _Default : System.Web.UI.Page{
    protected void Page_Load(object sender, EventArgs e) { }

    protected void btn_Click(object sender, EventArgs e) {

        Itl.Text = "";
        for (int i = 0; i < chkSemana.Items.Count; i++) {

```

```

        if (chkSemana.Items[i].Selected) {
            Itl.Text += chkSemana.Items[i].Text + " ";
        }
    }
}
}

```

Testando a aplicação

CheckBoxList

☐ Domingo

☒ Segunda-feira


☐ Terça-feira

☒ Quarta-feira

☐ Quinta-feira

☒ Sexta-feira

☐ Sábado

Segunda-feira Quarta-feira Sexta-feira 

Validação simples - DropDownList

.aspx

```

<asp:DropDownList ID="ddlTeste" runat="server" CssClass="form-control">
    <asp:ListItem Value="0">Produto Masc.</asp:ListItem>
    <asp:ListItem Selected="True" Value="">Selecione um item</asp:ListItem>
    <asp:ListItem Value="1">Produto Fem.</asp:ListItem>
</asp:DropDownList>

<asp:Literal ID="ItlMSG" runat="server"></asp:Literal>

```

.aspx.cs

```

if (ddlTeste.SelectedValue=="") {
    ItlMSG.Text = "<span class='text-danger'>Selecione um item </span>";
    return;
} else {

```

```

        ItlMSG.Text = "";
    }

```

Browser



Validação simples - RadioButtonList

.aspx

```

<asp:RadioButtonList ID="rblTeste" runat="server">
    <asp:ListItem Value="0">Produto Masc.</asp:ListItem>
    <asp:ListItem Value="1">Produto Fem.</asp:ListItem>
</asp:RadioButtonList>
<asp:Literal ID="ltrlbl" runat="server"></asp:Literal>

```

.aspx.cs

```

if (rblTeste.SelectedValue == "") {
    ltrlbl.Text = "<span class='text-danger'>Selecione um item </span>";
    return;
} else {
    ltrlbl.Text = "";
}

```

Browser



Estruturas Básicas

Comentários

Por linha:

```
// Os comentários por linha se iniciam com duas barras
```

Por bloco:

```
/*  
Comentários por bloco.  
Esse tipo de comentário pode conter várias linhas.  
Iniciar com "barra-asterisco" e terminar com "asterisco-barra".  
*/
```

Evite fazer comentários desnecessários no código, pois pode dificultar sua legibilidade e manutenção.

Declaração de Variáveis

Variáveis devem possuir um tipo e um identificador (nome).

```
//Tipo    Identificador  
int      anoNasc;  
double   peso;  
char     sexo;  
bool     ativo;
```

Declaração de múltiplas variáveis.

```
DateTime dataNascimento, dataFormatura, dataCasamento;
```

Regras na Declaração de Variáveis

Os identificadores de variáveis devem seguir algumas regras:

- O primeiro caractere deve ser uma letra ou ‘_’
- A partir do segundo caractere, podem ser usados: Letras, Números ou ‘_’.
- Se o identificador for igual a uma palavra-chave do C#, ele deve iniciar com ‘@’

```
string abc; // ok  
string _abc; // ok  
string a3b55; // ok  
string 1Lab; // Erro
```

```
string _lab_05; // ok
string x-y; // Erro
string int; // Erro
```

Inicialização de Variáveis

Atribuição (operador '=')

```
anoNasc = 1980;
peso = 65.7;
sexo = 'M';
```

Declaração e inicialização simultânea

```
double altura = 1.8;
```

Declaração e inicialização múltipla

```
int n1 = 10, n2 = 20, n3 = 30;
```

Todas as variáveis devem ser inicializadas antes de serem utilizadas ou teremos um **Erro de compilação**.

Alteração do Valor de Variáveis

Outros exemplos de uso de variáveis.

```
int numero = 33;
int novoNumero = numero + 1; // novoNumero = 34

int a = 3;
a = a + 1; // a = 4
```

Constantes

Uma constante é uma variável que não pode ter seu valor alterado após a inicialização. As constantes auxiliam o entendimento do código e evitam erros na programação. O modificador **const** é utilizado.

```
const int CHAVE = 36521;
CHAVE = 123; // ERRO DE COMPILAÇÃO
```

Tipos de Dados

Números inteiros

C#	CTS	Descrição
sbyte	System.SByte	8 bits, com sinal
byte	System.Byte	8 bits, sem sinal
short	System.Int16	16 bits, com sinal
ushort	System.UInt16	16 bits, sem sinal
int	System.Int32	32 bits, com sinal
uint	System.UInt32	32 bits, sem sinal
long	System.Int64	64 bits, com sinal
ulong	System.UInt64	64 bits, sem sinal

Números decimais

C#	CTS	Descrição
float	System.Single	32 bits, com sinal
double	System.Double	64 bits, sem sinal
decimal	System.Decimal	128 bits, com sinal

Observação: O Uso do tipo **decimal** pode ocasionar queda de performance.

Booleano

C#	CTS	Descrição
bool	System.Boolean	Verdadeiro ou false

Caractere

C#	CTS	Descrição
char	System.Char	16 bits, 1 caractere

Conversão de Tipos de Dados

Algumas vezes precisamos atribuir um valor de um tipo a uma variável de outro tipo.

```
int pagar = 1000;  
double valor = pagar;
```


A conversão pode ser feita

De um tipo “menor” para um tipo “maior”

Ex: **int** para **double**

De um tipo “maior” para um tipo “menor”

Ex: **long** para **int**

Este processo é chamado de **casting**

Casting Implícito

O compilador cuida de todo o processo de conversão.

Ocorre quando é preciso converter de um tipo “**menor**” para um tipo “**maior**”.

Não ocorre perda de informação.

```
float n2 = 22.36;  
double n3 = n2;
```

Casting explícito

O programador é responsável pela conversão dos dados.

A conversão ocorre de um tipo “**maior**” para um tipo “**menor**”

Pode ocorrer perda de valor.

```
double valor = 256378.365;  
int novoValor = (int)valor;
```

Exemplos

```
double valor = 256378.365;  
float novoValor = (float)valor;
```

Cuidado com o casting explícito

```
int valor = 899523647;  
short novoValor = valor; // O número 899523647 é muito grande para ser  
                          // armazenado em uma variável do tipo short
```

```
int valor = (int)7.7; // a casa decimal será perdida
```

Operadores Aritméticos

Operador	Descrição
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo

Operadores de Comparação

Operador	Descrição
==	Igual
!=	Diferente
<	Menor que
>	Maior que
<=	Menor ou igual a
>=	Maior ou igual a

Operadores Lógicos

Operador	Descrição
!	Negação
	OU
&&	E

Outros Operadores Importantes

Operador	Exemplo
++	x++ ++x
--	x-- --x
+=	x += 2
-=	x -= 5
*=	x *= 3
/=	x /= 4
%=	x%= 2

Estrutura de Controle 'if-else'

Sintaxe básica

```
if (<condição_booleana>){  
    <código_se_condição_verdadeira>;
```

Opcionalmente, pode existir uma cláusula else

```
if (<condição_booleana>){
    <código_se_condição_verdadeira>;
}else{
    <código_se_condição_falsa>;
}
```

A condição booleana pode ser qualquer expressão lógica
O resultado deve ser true ou false

```
int x = 55;
if (x > 333){
    <código_se_condição_verdadeira>;
}else{
    <código_se_condição_falsa>;
}
```

Para blocos compostos por uma linha, não é necessário usar { e }

```
int x = 20, y;
if (x > 10)
    y = x * 2;
else
    y = x * 3;
```

É preciso ter cuidado com essa sintaxe

```
int x = 20, y, z;
if (x > 10)
    y = x * 2;
    z = 1;
```

É preciso ter cuidado com essa sintaxe. A chamada z = 1 será executada sempre .

Operador Ternário

Outra possibilidade é o **operador ternário** em substituição ao if-else.

```
int idade = 15;
bool ativo = (idade > 18) ? true : false;
```

Estrutura de Controle 'switch-case'

A estrutura **switch-case** funciona de forma semelhante ao if-else.

Exemplo:

```
int mes = 5;

switch (mes) {
    case 1:
        ltl.Text = "Janeiro";
        break;
    case 2:
        ltl.Text = "Fevereiro";
        break;
    case 3:
        ltl.Text = "Março";
        break;
    case 4:
        ltl.Text = "Abril";
        break;
    default:
        ltl.Text = "?????";
        break;
}
```

Exemplo:

```
string mes = "mar";

switch (mes) {
    case "jan":
        ltl.Text = "Janeiro";
        break;
    case "fev":
        ltl.Text = "Fevereiro";
        break;
    case "mar":
        ltl.Text = "Março";
        break;
    case "abr":
        ltl.Text = "Abril";
        break;
    default:
        ltl.Text = "?????";
        break;
}
```

Exemplo:

```
char mes = 'J';
```

```

switch (mes) {
    case 'J':
        ltl.Text = "Janeiro";
        break;
    case 'F':
        ltl.Text = "Fevereiro";
        break;
    case 'M':
        ltl.Text = "Março";
        break;
    case 'A':
        ltl.Text = "Abril";
        break;
    default:
        ltl.Text = "?????";
        break;
}

```

Se um bloco **case** for vazio, o **break** não precisa ser especificado.

```
int mes = 4;
```

```

switch (mes) {
    case 1:
    case 2:
    case 3:
        // O bloco será executado se mês for igual a 1,2 ou 3
        ltl.Text = "1º Trimestre";
        break;
    case 4:
    case 5:
    case 6:
        // O bloco será executado se mês for igual a 4, 5 ou 6
        ltl.Text = "2º Trimestre";
        break;
    case 7:
    case 8:
    case 9:
        // O bloco será executado se mês for igual a 7, 8 ou 9
        ltl.Text = "3º Trimestre";
        break;
    case 10:
    case 11:
    case 12:
        //O bloco será executado se mês for igual a 10, 11 ou 12
        ltl.Text = "4º Trimestre";
        break;
}

```

```
default:
    ltl.Text = "?????";
    break;
}
```

Estrutura de Repetição 'while'

Repete determinado código enquanto uma condição booleana for verdadeira. A condição é testada no início do bloco.

```
int valor = 1;

ltl.Text = "";

while (valor <= 10) {
    ltl.Text += valor + " ";
    valor++; // 1 2 3 4 5 6 7 8 9 10
}
```

Se a condição for falsa já no primeiro teste, o bloco não é executado.

```
int valor = 11;

ltl.Text = "";

while (valor <= 10) {
    // O Bloco nunca será executado
}
```

Estrutura de Repetição 'do-while'

Semelhante ao **while**. Repete o código enquanto uma condição booleana for verdadeira.

A condição é testada no final do bloco. Como o teste da condição é feito no final, o bloco sempre será executado.

```
int valor = 1;
ltl.Text = "";

do {

    ltl.Text += valor + " ";
    valor++; ++; // 1 2 3 4 5 6 7 8 9 10

} while (valor <= 10);
```

Observe o exemplo a baixo, mesmo a condição sendo falsa o bloco será executado uma vez.

```
int valor = 11;
ltl.Text = "";

do {

    ltl.Text += valor + " ";
    valor++; // 11

} while (valor <= 10);
```

Estrutura de Repetição 'for'

Repete a execução de determinado código enquanto uma condição for verdadeira. Permite declarar uma ou mais variáveis.

Sintaxe:

```
for (inicializador; condição Booleana; iterador) {
    // Código repetido enquanto a condição for verdadeira
}
```

Exemplo:

```
ltl.Text = "";

for ( int i = 0; i <= 10 ; i++) {
    ltl.Text += i + " "; // 0 1 2 3 4 5 6 7 8 9 10
}
```

Atenção: Se nenhuma das informações forem passadas para o "FOR" ele entrará em um LOOP Infinito.

```
for ( ; ; ) {
    // Loop Infinito
}
```

Estrutura de Repetição 'foreach'

Permite iterar sobre todos os elementos de uma coleção de dados.

```
int[] num = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };
```

```
ltl.Text = "";
foreach (var item in num) { // Cada elemento do array é armazenado em item
    ltl.Text += item + " "; // 1 2 3 4 5 6 7 8 9 0
}
```

Controlando Iterações

As palavras-chave **break** e **continue** podem ser usadas para controlar iterações em estruturas de repetição.

- **break** pode ser usado também no **switch-case**. Força a saída de um loop.
- **continue** força novo teste da condição.

Controlando Iterações com 'break'

Exemplo de uso do **break**

```
ltl.Text = "";
for (int i = 0; i <= 100; i++) {

    if (i == 7) {
        break; // Termina a execução do loop
    }

    ltl.Text += i + " "; // 0 1 2 3 4 5 6

}
```

Controlando Iterações com 'continue'

Exemplo de uso do **continue**

```
ltl.Text = "";
for (int i = 0; i <= 100; i++) {

    if ((i % 2 == 0) || (i % 3 == 0) || (i % 5 == 0) || (i % 7 == 0)) {
        continue; // Termina a execução da iteração atual
    }

    ltl.Text += i + " ";
    // 1 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
}
```


Elementos Estáticos e Não Estáticos

Uma classe pode definir atributos e/ou métodos estáticos e não-estáticos ao mesmo tempo.

Os atributos e métodos **não estáticos** pertencem a cada objeto da classe

Atributos e métodos **estáticos** pertencem à classe.

Regra para acesso

- Elementos não-estáticos podem acessar elementos estáticos
- Elementos estáticos **não** podem acessar elementos não-estáticos.

Exemplo: Utilização de método estático em um array de objetos

Código Pilha (Motor)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

public class MotorPilha {

    private static int cont = 0;
    private static AlunoTeste[] al = new AlunoTeste[10];

    public static void Insert(AlunoTeste aluno) {
        al[cont] = aluno;
        cont++;
    }

    public static string Mostrar() {
        string imprimir = "";
        for (int i = 0; i < cont; i++) {
            imprimir += "<div class='alert alert-success'>" + al[i].Imprimir() + "</div>";
        }
        return imprimir;
    }

}
```

Código ASPX.CS

```
using System;
using System.Collections.Generic;
```

```

using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : System.Web.UI.Page {

    AlunoTeste al;

    protected void Page_Load(object sender, EventArgs e) {

    }

    protected void btn_Click(object sender, EventArgs e) {
        string nome = txtNome.Text;
        int idade = Convert.ToInt32(txtIdade.Text);

        al = new AlunoTeste(nome, idade);
        MotorPilha.Insert(al);
        lblDados.Text = MotorPilha.Mostrar();

        txtIdade.Text = "";
        txtNome.Text = "";
    }
}

```

Exercício 01

Crie a classe abaixo.

Crie um layout.

Trabalhe a entrada e saída dos dados. O método Imprimir deve retornar todos os dados da classe.

Pessoa
- pes_nome : string - pes_sobrenome : string - pes_idade : int
+ gets() + Sets() + Imprimir() : string

Sobre os exercício

- Crie os formulários para cadastro.
- Independente da finalidade do formulário, o importante neste momento é treinar os componentes TextBox, DropDownList, RadioButtonList e CheckBoxList.
- Treine os eventos.
 - Utilizando o botão

- Utilizando o DropDownList
- Utilizando o RadioButtonList

A criatividade é muito importante.

Exercício 002

Crie a classe abaixo.

Crie um layout simples.

Trabalhe a entrada e saída dos dados. O método Imprimir deve retornar todos os dados da classe

Carro
- car_nomeDoProprietario : string - car_modelo : string - car_ano : int - car_cor : string - car_placa : string
+ gets() + Sets() + Imprimir() : string

Exercício 003

Crie a classe abaixo.

Crie um layout simples.

Trabalhe a entrada e saída dos dados. O método Imprimir deve retornar todos os dados da classe

O método Calcular é um método privado e deve retornar o valor do desconto.

Veículo
- vei_modelo : string - vei_ano : int - vei_cor : string - vei_placa : string - vei_cidade: string - vei_estado : string - vei_valor : double - vei_desconto : double
+ gets() + Sets() - CalcularDesconto() : double + Imprimir() : string

Elementos Estáticos e Não Estáticos

Uma classe pode definir atributos e/ou métodos estáticos e não-estáticos ao mesmo tempo.

Os atributos e métodos **não estáticos** pertencem a cada objeto da classe

Atributos e métodos **estáticos** pertencem à classe.

Regra para acesso

- Elementos não-estáticos podem acessar elementos estáticos
- Elementos estáticos **não** podem acessar elementos não-estáticos.

Exemplo: Utilização de método estático em um array de objetos

Código Pilha (Motor)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

public class MotorPilha {

    private static int cont = 0;
    private static AlunoTeste[] al = new AlunoTeste[10];

    public static void Insert(AlunoTeste aluno) {
        al[cont] = aluno;
        cont++;
    }

    public static string Mostrar() {
        string imprimir = "";
        for (int i = 0; i < cont; i++) {
            imprimir += "<div class='alert alert-success'>" + al[i].Imprimir() + "</div>";
        }
        return imprimir;
    }

}
```

Código ASPX.CS

```
using System;
using System.Collections.Generic;
```

```

using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : System.Web.UI.Page {

    AlunoTeste al;

    protected void Page_Load(object sender, EventArgs e) { }

    protected void btn_Click(object sender, EventArgs e) {
        string nome = txtNome.Text;
        int idade = Convert.ToInt32(txtIdade.Text);

        al = new AlunoTeste(nome, idade);
        MotorPilha.Insert(al);
        lblDados.Text = MotorPilha.Mostrar();

        txtIdade.Text = "";
        txtNome.Text = "";
    }
}

```

Exercício 004

Crie a classe abaixo.

Trabalhe a entrada e saída dos dados. O método Imprimir deve retornar todos os dados da classe.

Persista os dados em um array e exiba as informações em uma div.

Animal
- ani_nome : string - ani_raça: string - ani_proprietário : string - ani_telefone : string
+ gets() + Sets() + Imprimir() : string

Exercício 005

Crie a classe abaixo.

Trabalhe a entrada e saída dos dados. O método Imprimir deve retornar todos os dados da classe.

Persista os dados em um array e exiba as informações em uma Tabela.

Cliente
- cli_nome : string - cli_sobrenome: string - cli_rg : string - cli_cpf : string
+ gets() + Sets() + Imprimir() : string

Exercício 006

Crie a classe abaixo.

Se julgar necessário crie outros métodos para auxiliar sua codificação.

Trabalhe a entrada e saída dos dados.

Persista os dados em um array e exiba as informações em uma Tabela.

Utilize um DropDownList para o campo estado.

Endereco
- end_ rua : string - end_numero: string - end_bairro: string - end_cidade: string - end_estado: string - end_cep: string
+ Gets() + sSets()

Exercício 007

Crie a classe abaixo.

Se julgar necessário crie outros métodos para auxiliar sua codificação.

Trabalhe a entrada e saída dos dados.

Persista os dados em um array e exiba as informações em uma Tabela.

Utilize um DropDownList para o campo estado e RadioButtonList para o campo sexo.

Individuo
- ind_nome : string - ind_cpf: string - ind_sexo: char - ind_estado
+ Gets() + sSets()

Exercício 008

Crie a classe abaixo.

Se julgar necessário crie outros métodos para auxiliar sua codificação.

Trabalhe a entrada e saída dos dados.

Persista os dados em um array e exiba as informações em uma Tabela.

Utilize um DropDownList para o campo estado e RadioButtonList para o campo sexo.

Funcionario
- fun_nome : string - fun_cpf: string - fun_rg: string - fun_rua:string - fun_numero:string - fun_cidade:string - fun_estado:string - fun_cep:string - fun_sexo: char
+ Gets() + sSets()

Master Page

Master Page é um layout padrão que será utilizado por todas as outras páginas, o que representa uma herança visual.

Com isso conseguimos uma maior facilidade de manutenção do site, pois ao modificarmos a Master Page, todas as demais serão modificadas.

A Master Page não é executada, o que é executado são as páginas que herdam as características visuais da Master Page.

Para inserir uma Master Page:

- Botão Direito sobre o projeto:
 - Add / New Item
 - Master page
 - Linguagem Visual C#
 - Coloque um nome Sugestivo
 - “MasterPage.Master”
 - Add

Observação:

A **Master Page** possui a extensão .Master

Da mesma forma que as páginas .aspx ela possui um **Code Hidden.cs**.

É necessário uma página que herde as características da Master Page para que ela seja executada.

ContentPlaceholder

```
<asp:ContentPlaceholder id="ContentPlaceholder1" runat="server">
```

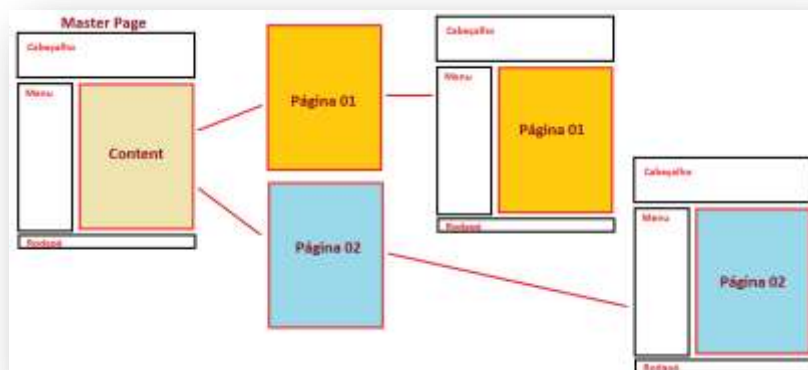
Local disponível para as páginas filhas

```
</asp:ContentPlaceholder>
```

São locais onde serão substituídos os conteúdos de outras páginas.

Aquilo que não é padrão.

As páginas filhas terão seu conteúdo exibido neste local.



Estrutura Básica HTML / MasterPage

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <asp:ContentPlaceHolder id="head" runat="server">
        Local reservado para os Scripts das Páginas filhas
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
                Local reservado para o conteúdo principal das páginas filhas
            </asp:ContentPlaceHolder>
        </div>
    </form>
</body>
</html>
```

Construtor e sobrecarga

Inicializando todos os atributos no momento de instanciação do objeto.

Exemplo 01 : Classe Aluno

```
public class Aluno {

    private int idade;
    private string nome;

    public Aluno() { } // Construtor padrão

    // Construtor - Inicializando todos os atributos no momento de instanciação do objeto.
    public Aluno(int idade, string nome) {
        this.idade = idade;
        this.nome = nome;
    }

    public int Idade {
        get {
            return idade;
        }

        set {
            idade = value;
        }
    }

    public string Nome {
        get {
            return nome;
        }

        set {
            nome = value;
        }
    }

    // sobrecarga
    public override string ToString() {

        return "<div class='alert alert-danger'>" +
            " <strong> Nome: </strong> " + nome +
            " <strong> Idade: </strong> " + idade +
            "</div>";

    }
}
```

.ASPX

```
<asp:TextBox ID="txtnome" runat="server"></asp:TextBox>
<asp:TextBox ID="txtIdade" type="number" runat="server"></asp:TextBox>
<asp:Button ID="btn" runat="server" Text="Enviar" OnClick="btn_Click" />

<asp:Literal ID="ltl" runat="server"></asp:Literal>
```

.ASPX.CS

```
protected void btn_Click(object sender, EventArgs e) {

    Aluno a = new Aluno(Convert.ToInt32(txtIdade.Text), txtnome.Text);
    ltl.Text = a.ToString();

}
```

Exemplo 02 : Classe Aluno2

```
public class Aluno2 {

    public int Idade { get; set; }
    public string Nome { get; set; }

    public Aluno2() { } // Construtor Padrão

    // Construtor - Inicializando todos os atributos no momento de instanciação do objeto.
    public Aluno2(int idade, string nome) {
        Idade = idade;
        Nome = nome;
    }

    // Sobrecarga
    public override string ToString() {
        return "<div>" +
            "<strong> Nome: </strong>" + Nome +
            "<strong> Idade: </strong>" + Idade +
            "</div>";
    }

}
```

.ASPX

```
<asp:TextBox ID="txtnome" runat="server"></asp:TextBox>
<asp:TextBox ID="txtIdade" type="number" runat="server"></asp:TextBox>
<asp:Button ID="btn" runat="server" Text="Enviar" OnClick="btn_Click" />
<asp:Literal ID="ltl" runat="server"></asp:Literal>
```

.ASPX.CS

```
protected void btn_Click(object sender, EventArgs e) {  
    Aluno2 a = new Aluno2(Convert.ToInt32(txtIdade.Text), txtnome.Text);  
    Itl.Text = a.ToString();  
}
```

Array Dinâmico - List

Observação:

Neste exemplo iremos trabalhar apenas com os métodos Add() e clear() do List.

Classe Aluno

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
  
public class Aluno {  
  
    private int idade;  
    private string nome;  
  
    public Aluno() { } // Construtor padrão  
  
    // Construtor - Inicializando todos os atributos no momento de instanciação do objeto.  
    public Aluno(int idade, string nome) {  
        this.idade = idade;  
        this.nome = nome;  
    }  
  
    public int Idade {  
        get {  
            return idade;  
        }  
  
        set {  
            idade = value;  
        }  
    }  
  
    public string Nome {  
        get {  
            return nome;  
        }  
  
        set {  
            nome = value;  
        }  
    }  
}
```

```

    }

    //sobrecarga
    public override string ToString() {
        return "<div class='alert alert-danger'>" +
            " <strong> Nome: </strong> " + nome +
            " <strong> Idade: </strong> " + idade +
            "</div>";
    }
}

```

Classe AlunoDB

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

// Importar
using System.Collections;

public class AlunoDB {

    public static List<Aluno> aluno { get; set; } = new List<Aluno>();

    public static void Insert(Aluno a) {
        aluno.Add(a);
    }

    public static string Select() {
        string str = "";
        foreach (var item in aluno) {
            str += item.ToString();
        }
        return str;
    }

    public static void DeleteAll() {
        aluno.Clear();
    }

}

```

.ASPX

```

<asp:TextBox ID="txtnome" runat="server"></asp:TextBox>
<asp:TextBox ID="txtIdade" type="number" runat="server"></asp:TextBox>
<asp:Button ID="btnDelte" runat="server" Text="Limpar Array" OnClick="btnDelte_Click" />
<asp:Button ID="btn" runat="server" Text="Enviar" OnClick="btn_Click" />

```

<asp:Literal ID="Itl" runat="server"></asp:Literal>

.ASPX.CS

```
public partial class Teste_TestAluno3 : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) {

    }

    protected void btn_Click(object sender, EventArgs e) {

        Aluno a = new Aluno(Convert.ToInt32(txtIdade.Text), txtnome.Text);
        AlunoDB.Insert(a);
        Itl.Text = AlunoDB.Select();

    }

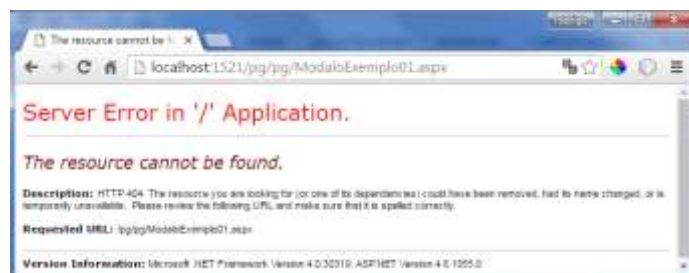
    protected void btnDelte_Click(object sender, EventArgs e) {
        AlunoDB.DeleteAll();
        Itl.Text = AlunoDB.Select();
    }
}
```

Links que perdem a referência

No .NET é comum um link (como o exemplo abaixo) perder a referência.

Código

```
<a href="pg/ModaloExemplo01.aspx" class="btn-danger btn">ModaloExemplo01.aspx</a>
<a href="pg/ModaloExemplo02.aspx" class="btn-success btn">ModaloExemplo02.aspx</a>
<a href="pg/ModaloExemplo03.aspx" class="btn-inverse btn">ModaloExemplo03.aspx</a>
```



Para evitar este problema utilizamos o seguinte Script.

```
<a href='<%=ResolveUrl("pg/ModaloExemplo01.aspx") %>' class="btn-danger
btn">ModaloExemplo01.aspx</a>

<a href='<%=ResolveUrl("pg/ModaloExemplo02.aspx") %>' class="btn-success btn">ModaloExemplo02.aspx</a>

<a href='<%=ResolveUrl("pg/ModaloExemplo03.aspx") %>' class="btn-inverse btn">ModaloExemplo03.aspx</a>
```

O mesmo pode acontecer com os links com **Script**, para evita este problema podemos utilizar o **ResolveURL** conforme modelo abaixo.

```
<script src='<%= ResolveUrl("~/bootstrap/js/jquery-1.12.3.min.js")%'></script>
<script src='<%= ResolveUrl("~/bootstrap/js/bootstrap.min.js")%'></script>
```

MODAL

Visão do Browser



Código ASPX

```
<div class="container">
```

```
    <!-- HTML Gatilho modal -->
```

```
    <button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal">
        Gatilho HTML - Modal
    </button>
```

```
    <asp:Button ID="btn" runat="server" Text="Gatilho ASPX - Modal" CssClass="btn btn-success"
        OnClick="btn_Click" />
```

```
</div>
```

```
    <!-- Modal -->
```

```
    <div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel"
        aria-hidden="true">
```

```
        <div class="modal-dialog">
```

```
            <div class="modal-content">
```

```
                <div class="modal-header">
```

```
                    <h5 class="modal-title" id="exampleModalLabel">Modal title</h5>
```

```
                    <button type="button" class="close" data-dismiss="modal" aria-label="Close">
```

```
                        <span aria-hidden="true">&times;</span>
```

```
                    </button>
```

```
                </div>
```

```
                <div class="modal-body">
```

```
                    ...
```

```
                </div>
```



```

        <div class="modal-footer">
            <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
            <button type="button" class="btn btn-primary">Save changes</button>
        </div>
    </div>
</div>
</div>

<script src='<%= ResolveUrl("../Scripts/jquery-3.5.1.min.js") %>'></script>
<script src='<%= ResolveUrl("../Scripts/bootstrap.bundle.min.js") %>'></script>

```

Código aspx.cs

```

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

public partial class pg_Modal1 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void btnModal_Click(object sender, EventArgs e)
    {
        Page . ClientScript . RegisterStartupScript ( this . GetType (), "script",
        "<script> $('#IdDaModal') . modal('show') ; </script>" , false);
    }
}

```

Sessão

A sessão permite armazenar qualquer tipo de dado na memória do servidor.

A informação é protegida porque nunca é exibida para o usuário, além de pertencer exclusivamente a uma sessão específica.

A sessão é ideal para o armazenamento de informações, como os itens de um carrinho de compras, enquanto o usuário navega entre as páginas.

Criar Sessão:

```
Session["nome"] = "Dionisio";
```

Persistir a Sessão:

```
lblSessao.Text = Convert.ToString(Session["nome"]);
```

Remover Sessão:

```
Session.Remove("nome");
```

Exemplo:

Neste exemplo Criaremos quatro páginas simples.

Na primeira página a **SESSÃO** será criada automaticamente.

Nas demais páginas a **SESSÃO** será persistida.

Na última página teremos um botão com a finalidade de **ENCERRAR A SESSÃO**.

Visão do Browser

[Home pg1 pg2 pg3](#)

[Home pg1 pg2 pg3](#)

[Home pg1 pg2 pg3](#)

[Home pg1 pg2 pg3](#)

HOME

PÁGINA 01

PÁGINA 02

PÁGINA 03

Fulano

Fulano

Fulano

Fulano

Remover Sessão

HOME

```
<a href="Default.aspx">Home</a>
<a href="pag1.aspx">pg1</a>
<a href="pag2.aspx">pg2</a>
<a href="pag3.aspx">pg3</a>
<br />
<br />
<h1>HOME</h1>
```

```
<asp:Label ID="lblSessao" runat="server" Text="---"
"></asp:Label>
```

```
protected void Page_Load(object sender, EventArgs e) {
    Session["nome"] = "Dionisio";
    // lblSessao.Text = "Bem Vindo: " + Session["nome"];
    lblSessao.Text = Convert.ToString(Session["nome"]);
}
```

PÁGINA 01

```

<a href="Default.aspx">Home</a>
<a href="pag1.aspx">pg1</a>
<a href="pag2.aspx">pg2</a>
<a href="pag3.aspx">pg3</a>
<br />
<br />
<h1>PÁGINA 01</h1>

<asp:Label ID="lblSessao" runat="server" Text="---"
"></asp:Label>

```

PÁGINA 02

```

<a href="Default.aspx">Home</a>
<a href="pag1.aspx">pg1</a>
<a href="pag2.aspx">pg2</a>
<a href="pag3.aspx">pg3</a>
<br />
<br />
<h1>PÁGINA 02</h1>

<asp:Label ID="lblSessao" runat="server" Text="---"
"></asp:Label>

```

PÁGINA 03

```

<a href="Default.aspx">Home</a>
<a href="pag1.aspx">pg1</a>
<a href="pag2.aspx">pg2</a>
<a href="pag3.aspx">pg3</a>
<br />
<br />
<h1>PÁGINA 03</h1>

<asp:Label ID="lblSessao" runat="server" Text="---"
"></asp:Label><br /><br />

<asp:Button ID="btn" runat="server" Text="Remover
Sessão" OnClick="btn_Click" />

```

Redirecionamento

```
Response.Redirect("TipoEmpresa.aspx");
```

```

protected void Page_Load(object sender, EventArgs e) {
    if (Session["nome"] == null ) {

        lblSessao.Text = "Sessão não encontrada –
        Direcionar para uma página de erro";

    } else {
        lblSessao.Text =
        Convert.ToString(Session["nome"]);
    }
}

```

```

protected void Page_Load(object sender, EventArgs e) {
    if (Session["nome"] == null ) {

        lblSessao.Text = "Sessão não encontrada –
        Direcionar para uma página de erro";

    }else{

        lblSessao.Text =
        Convert.ToString(Session["nome"]);
    }
}

```

```

protected void Page_Load(object sender, EventArgs e) {
    if (Session["nome"] == null ) {

        lblSessao.Text = "Sessão não encontrada –
        Direcionar para uma página de erro";

    } else {

        lblSessao.Text =
        Convert.ToString(Session["nome"]);
    }
}

protected void btn_Click(object sender, EventArgs e) {
    Session.Remove("nome");
    lblSessao.Text = "Sessão não encontrada –
    Direcionar para uma página de erro";
}

```

Exemplo simples de login

Para este exemplo iremos criar uma classe “LOG”, uma interface para login e três páginas, Adm (para administrador), Func (para funcionário) e uma página de erro.

Ao digitarmos a senha correta seremos direcionados para a página de destino correta.

Caso contrário uma mensagem de erro será exibida na tela de login.

Deveremos validar os dados com a sessões.

Caso o usuário tente acessar a página direto pela url sem fazer login deverá ser direcionado para a página de erro.

Log.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

public class log {
    private string email;
    private string senha;
    private string nome;

    public log(string email, string senha, string nome) {
        this.nome = nome;
        this.senha = senha;
        this.email = email;
    }

    public string Email {
        get { return email; }
        set { email = value; }
    }

    public string Senha {
        get { return senha; }
        set { senha = value; }
    }

    public string Nome {
        get { return nome; }
        set { nome = value; }
    }
}
```

Font

LOGIN

E-mail:

Senha:

LOGAR

ASPX

```
<div class="row">

    <div class="col-md-12 text-center">
        <h1>LOGIN</h1>
        <hr />
    </div>

    <div class="col-md-4"></div>

    <div class="col-md-4">
        <label>
            E-mail:
        </label>
        <asp:TextBox runat="server" required="required" ID="txtEmail" type="email"
            CssClass="form-control"></asp:TextBox>
        <br />

        <label>
            Senha:
        </label>
        <asp:TextBox runat="server" required="required" ID="txtSenha" type="password"
            CssClass="form-control"></asp:TextBox>
        <br />
        <br />

        <asp:Button runat="server" ID="btn" CssClass="btn btn-primary btn-block" Text="LOGAR"
            OnClick="btn_Click" />
        <br />
        <br />

        <asp:Literal runat="server" ID="ltl"></asp:Literal>

    </div>
</div>
```

ASPX.cs

```
using System;
```

```

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

public partial class Scripts_log1 : System.Web.UI.Page {

    protected void Page_Load(object sender, EventArgs e) { }

    protected void btn_Click(object sender, EventArgs e) {

        log logAdm = new log("adm@adm.com", "123", "Administrador");
        log logFunc = new log("func@adm.com", "123", "Funcionario");

        if (logAdm.Email == txtEmail.Text && logAdm.Senha == txtSenha.Text) {
            Session["ADM"] = logAdm;
            Response.Redirect("../ADM.aspx");

        }else if (logFunc.Email == txtEmail.Text && logFunc.Senha == txtSenha.Text) {
            Session["FUNC"] = logAdm;
            Response.Redirect("../Funcionario.aspx");

        } else {
            ltl.Text = "<div class='alert alert-danger'> ERRO </div>";
        }
    }
}

```

Front – Página ADM.aspx

```

<h1>ADM</h1>
<asp:Literal runat="server" ID="ltl"></asp:Literal>
<asp:Button runat="server" ID="btn" OnClick="btn_Click" Text="Sair" />

```

Front – Página ADM.aspx.cs

```

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

public partial class ADM : System.Web.UI.Page {

    protected void Page_Load(object sender, EventArgs e) {

        log l;
        if(Session["ADM"] != null) {

```

```

        l = (log)Session["ADM"];
        Itl.Text = l.Nome;
    } else {
        Response.Redirect("ERRO.aspx");
    }
}

protected void btn_Click(object sender, EventArgs e) {
    Session.Remove("ADM");
    Response.Redirect("Scripts/log1.aspx");
}
}

```



Criptografia Básica

Visão do Browser



Código ASPX

```

Senha:
<asp:TextBox ID="txtPassword" runat="server"></asp:TextBox>
<br />
<br />
<asp:Label ID="lblSenha" runat="server" Text="---"></asp:Label>
<br />
<br />
<asp:Button ID="btnSenha" runat="server" Text="Gerar Senha" OnClick="btnSenha_Click"/>

```

Código ASPX.CS

```

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

//-----
using System.Text;
using System.IO;
using System.Security.Cryptography;

```

```
public partial class pg_Criptografia : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) {}

    protected void btnSenha_Click(object sender, EventArgs e) {

        UnicodeEncoding UE = new UnicodeEncoding();
        byte [] HashValue, MessageBytes = UE.GetBytes(txtPassword.Text);
        SHA512Managed SHhash = new SHA512Managed();
        string strHex = "";

        HashValue = SHhash.ComputeHash(MessageBytes);
        foreach (byte b in HashValue) {
            strHex += String.Format("{0:x2}", b);
        }
        lblSenha.Text = strHex;
    }
}
```