Web Forms – Visual Studio 2017

ASP.NET usa o .NET Framework e permite ao desenvolvedor criar páginas Web independentes do browser do usuário final. O servidor detecta o tipo de browser sendo usado e procura adaptar, automaticamente, a resposta enviada ao browser aos recursos que este dispõe.

Além da independência de browser, as seguintes características estão disponíveis para a criação de páginas Web dinâmicas:

- Web Forms podem ser projetados e programados usando ferramentas RAD (Rapid Application Development, tais como Visual Studio e Delphi.
- Web Forms suportam um rico conjunto de controle e são extensíveis, porque elas fornecem suporte para controles criados por desenvolvedores e por terceiros.
- Qualquer uma das linguagens compatíveis com o .NET Framework podem ser usadas para programar as páginas Web Forms do ASP.NET.
- ASP.NET usa a CLR (Common Language Runtime) do .NET Framework e, portanto, se beneficia de seus recursos, tais como segurança de tipos e herança.

Componentes de Web Forms

Uma página Web Form de ASP.NET consiste de uma interface com o usuário e de lógica de programação. A interface com o usuário é útil para exibir informações aos clientes, enquanto a lógica de programação manipula a interação do usuário com a página Web Form.

A interface com o usuário consiste de um arquivo contendo uma linguagem de marcação (ou formatação), tais como HTML ou XML, e controles de servidor. Este arquivo é chamado de página e seu nome de arquivo possui a extensão .aspx.

A funcionalidade para responder às interações do usuário com as páginas Web Forms é implementada pelo uso de linguagens de programação, tais como Visual Basic .NET e C#.

Você pode implementar a lógica de programação no arquivo ASPX ou em um arquivo separado escrito em qualquer linguagem suportada pela CLR, tais como Visual Basic .NET, C# ou Delphi.

Este arquivo separado é chamado de **code behind file** e tem como extensão .aspx.cs, .aspx.vb ou .aspx.pas, dependendo da linguagem usada. Portanto, uma página Web Forms page **consiste de uma página (arquivo ASPX) e um arquivo code behind** (.aspx.cs, .aspx.vb, .aspx.pas).

O modelo de eventos do ASP.Net

No ASP.Net existe algumas etapas que acontecem quando a página é invocada pelo usuário. A seguir temos um resumo destas etapas:

1. A página é executada (ou seja, é invocada pelo browser do usuário cliente) pela primeira vez. A solicitação é recebida pelo servidor web, que verifica que é uma aplicação ASP.Net e, em seguida, invoca o sistema ASP.Net instalado no servidor. Esse sistema cria a página e os objetos de controle, o código de

inicialização é executado e, em seguida, a página é convertida para HTML e retornada ao browser do cliente. Os objetos da página são liberados da memória do servidor web;

2. Em algum momento, o usuário faz algo que dispara um **postback**, tal como clicar um botão (que não é, necessariamente, um botão de submit do HTML). Nesse momento, a página será submetida (enviada) ao servidor com todos os dados do formulário;
3. ASP.Net intercepta a página retornada e recria os objetos da página, cuidando para que eles retornem ao estado em que eles estavam na última vez em que a página foi enviada ao cliente;
4. Em seguida, ASP.Net verifica que operação disparou o postback e, então, executa os métodos associados como tratadores de eventos (tal como Button.Click). Em geral, nesse momento seus métodos farão alguma operação do lado servidor, como atualizar uma tabela de banco de dados ou ler dados de um arquivo e, em seguida, alterará o conteúdo formato e/ou visibilidade dos controles, para exibir novas informações.
5. A página modificada é convertida para HTML, já com as alterações dos conteúdos, formatos e visibilidade de controles citados acima e retornada ao cliente. Os objetos são removidos da memória do servidor. Se outro postback ocorrer, ASP.Net repetirá o processo dos passos 2 a 4.

Em outras palavras, ASP.Net não somente usa os dados do formulário para configurar os objetos de controles da sua página. Ele também os usa para decidir quais eventos disparar. Por exemplo, se ele nota que o texto em um textBox foi modificado desde o último postback, ele dispara um evento para notificar sua página (onChange). É sua decisão, através dos tratadores de eventos que você programou (ou deixou de programar) se você deseja responder a esse evento.

Observação: lembre-se sempre que as páginas apresentadas no browser não possuem controles ASP. Eles foram convertidos para controles HTML (passos 1 e 5 acima). Já que o protocolo HTTP é completamente "sem estado" e todo o estado disponibilizado pelo ASP.Net é reconstituído, o modelo orientado a eventos é realmente uma emulação. ASP.Net realiza algumas tarefas de fundo (background, escondidas) para suportar esse modelo. O interessante e belo nesse conceito é que o programador iniciante não precisa estar familiarizado com as bases do sistema para obter as vantagens de poder programar eventos no lado servidor

Web Forms server controls

Você pode projetar uma página Web Forms usando controles chamados **Controles de Servidor Web Forms** (Web Forms server controls).

A funcionalidade dos controles de servidor é programada através de tratadores de eventos, como ocorre em Delphi, Visual Basic, C#, dentre outras linguagens.

Os controles de Servidor são diferentes dos controles usuais do Windows porque eles trabalham dentro do ASP.NET Framework. Os diversos tipos de controles de servidor são descrito a seguir:

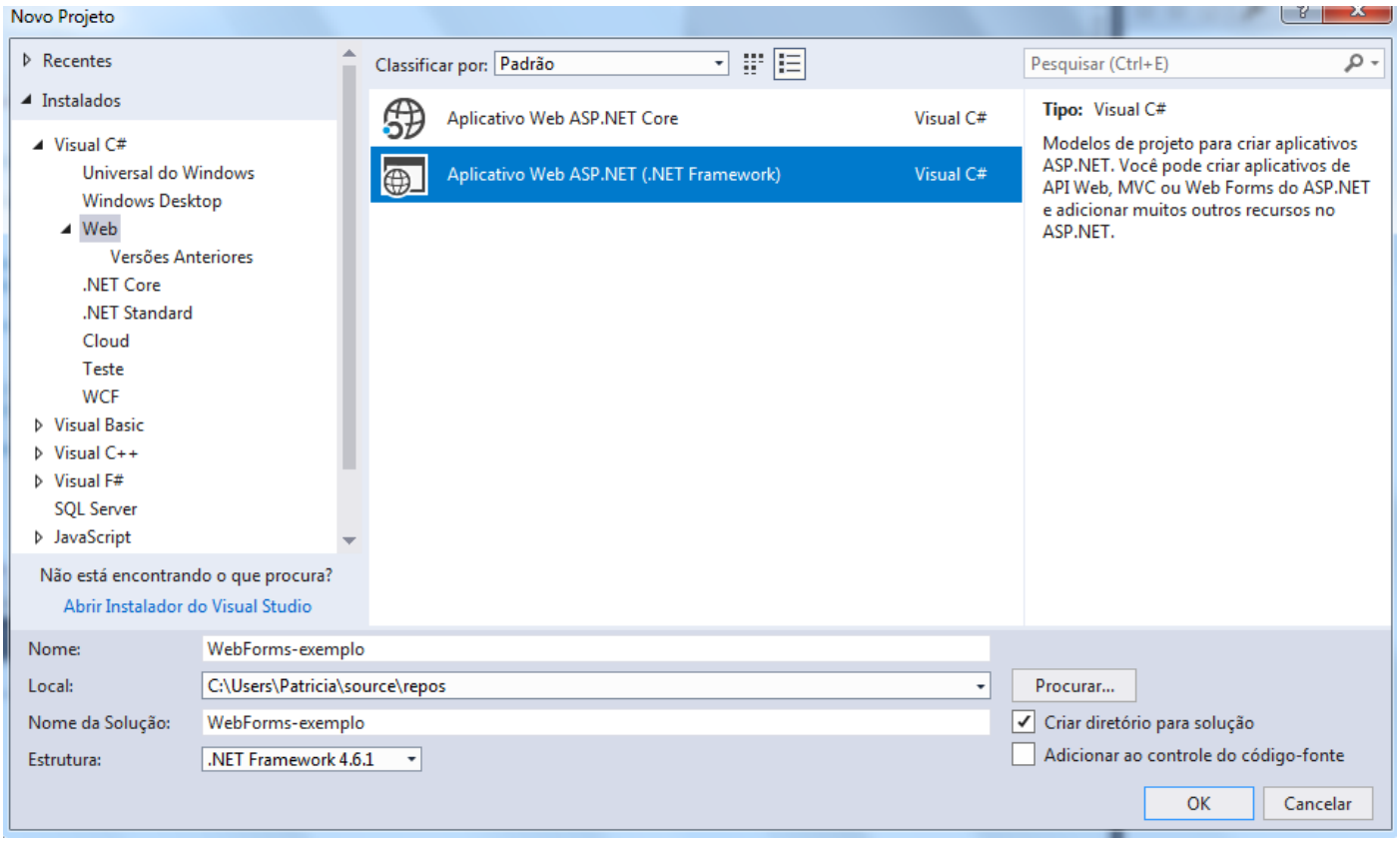
- **Controles de Servidor HTML:** Estes controles referenciam os elementos HTML (como input type=”text”, select, etc) que podem ser usados no código do servidor. Os elementos HTML podem ser convertidos em controles de servidor HTML. Para tanto, você precisa usar atributos, tais como ID e RUNAT, nas tags que usuaria normalmente como controles HTML. Você pode também adicionar esses controles, usando o Visual Studio, através da aba HTML tab da **Caixa de Ferramentas (toolbox)** do Visual Studio. Esses

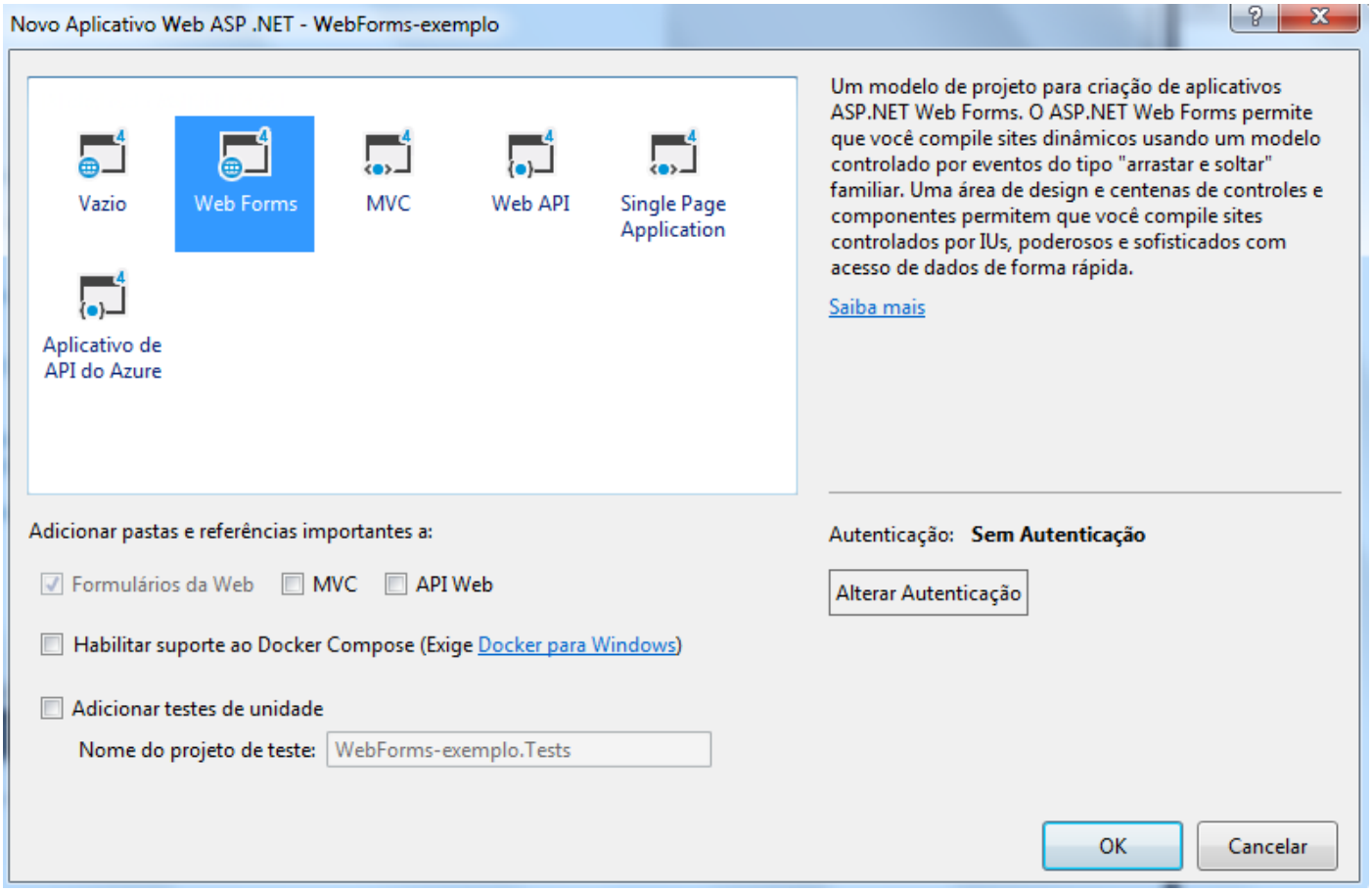
controles são úteis quando você já tem uma página HTML pronta, que utiliza controles padrão HTML e apenas deseja convertê-la para ASP.NET sem usar os controles de Servidor ASP.NET.

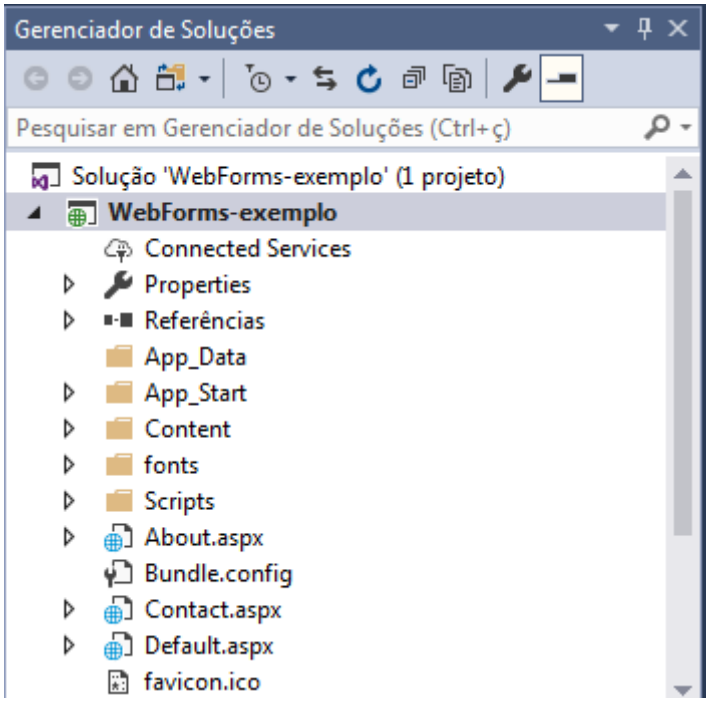
- **Controles de Servidor ASP.NET:** Estes controles não são mapeados um-a-um para os controles de servidor HTML (por exemplo, ASP.NET possui um controle **Calendar** que HTML não possui). Controles de Servidor ASP.NET incluem os controles tradicionais dos formulários HTML, tais como caixas de texto e botões, e controles complexos, como objetos de tabelas e de grid de dados.
- **Controles de Validação:** Estes controles são usados para validar a digitação de dados de entrada feita pelo usuário em campos ou controles de formulários. **Controles de Validação** podem ser anexados aos demais controles de entrada de dados para verificar se os dados estão dentro de regras pré-estabelecidas, como limites, formato, etc.
- **Controles de Usuário:** Estes controles são criados a partir de páginas Web Forms já existentes e podem ser usados em outras páginas Web Forms, de forma a reaproveitar código.

Criando uma página de login

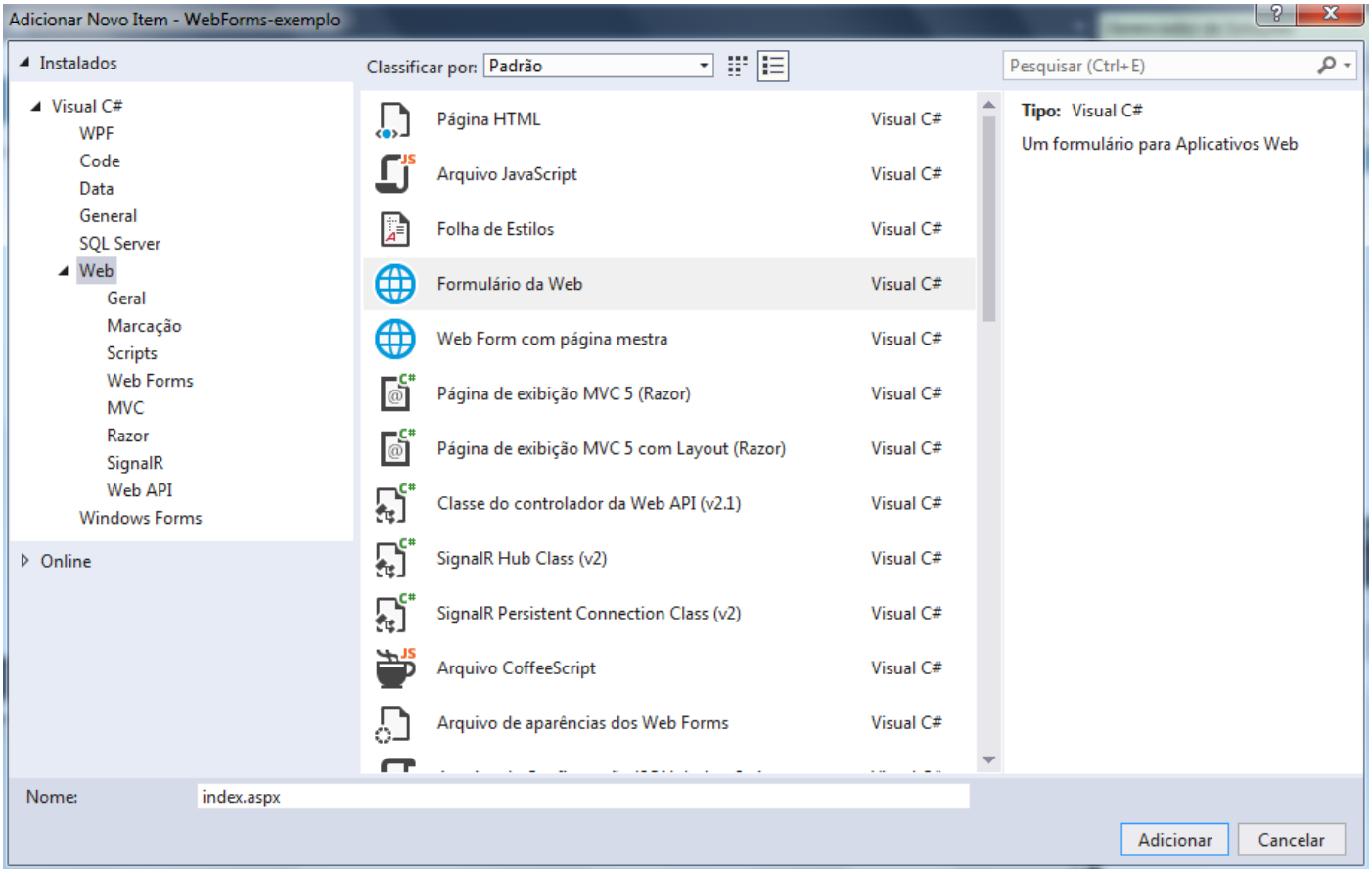
Criar um novo projeto (**Arquivo – Novo – Projeto**) e selecionar **Web – Aplicativo Web ASP.NET (.NET Framework):**







Inserir novo item. Clicar com o botão da direita do mouse sobre o nome do projeto e selecionar **Adicionar – Novo item – Web – Formulário da Web**) e atribuir o nome **index.aspx**:



Na página criada, ir em Design acrescentar os seguintes elementos:

- **Label** (id= **lblMensagem**)
- **TextBox** (id= **txtNome**)
- **Button** (**id=btnProcessar**, **Text=Processar**)

Em seguida incluiremos um objeto da classe **Panel**, que agrupa outros objetos e permite mostrá-los apenas quando necessário. Vamos configurar a propriedade **Visible=False** para Panel. Dentro do objeto Panel incluiremos mais um elemento **Label**:

```
<form id="form1" runat="server">
   <div>
       <asp:Label ID="lblMsg" runat="server" Text="Digite o seu nome: "> </asp:Label>
       <asp:TextBox ID="txtNome" runat="server"></asp:TextBox>
       <asp:Button ID="btnProcesar" runat="server" Text="Processar" />
   </div>
   <asp:Panel ID="pnlNome" runat="server" Visible="False">
       <asp:Label ID="lblNome" runat="server"></asp:Label>
   </asp:Panel>
</form>
```

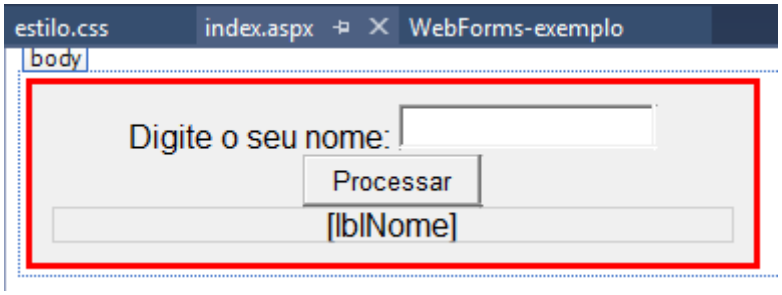
Na pasta **Content** (procurar no **Gerenciador de Soluções**), adicionar um arquivo CSS (nomear com **estilo.css**) e acrescentar a seguinte declaração:

```
#form1 {
    font-family: Arial;
    border: 3px solid red;
    width: 40%;
    height: 35%;
    padding: 10px;
    background-color: #F0F0F0;
    text-align: center;
}
```

No arquivo index.aspx inserir o CSS:

```
<link rel="stylesheet" type="text/css" href="Content/estilo.css" />
```

Vamos ter a seguinte página:

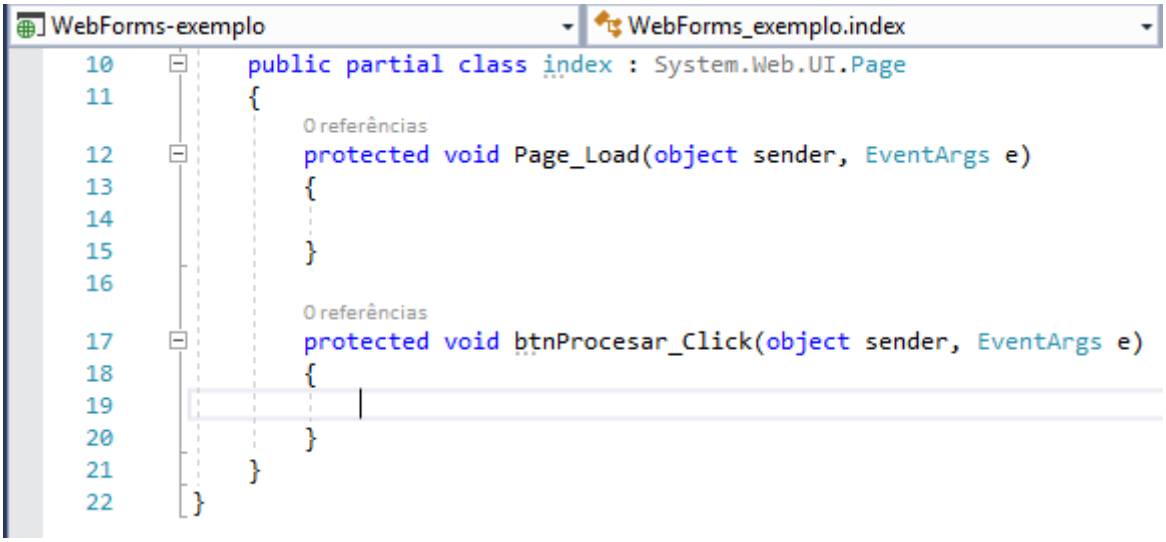


Eventos

Quando acessamos o **código fonte da página** no browser verificamos que o código abaixo é criado: os dois labels, o campo TextBox (input type=”text”) e um botão.


```
<div>
       <span id="lblMensagem">Digite o seu nome: </span>
       <input name="txtNome" type="text" id="txtNome" />
       <input type="submit" name="btnProcesar" value="Processar" id="btnProcesar" />
 </div>
```

Ao digitar o evento **OnClick** no elemento **btnProcessar**, aparecerá um menu de contexto perguntando se deve criar um novo evento ou será atribuído um evento já declarado. Nesse caso, o único evento declarado é o Page_Load (conferir no arquivo **index.aspx.cs**):



No código fonte temos a seguinte alteração:

```
<asp:Button ID="btnProcesar" runat="server" Text="Processar" OnClick="btnProcesar_Click" />
```

Ao clicar nesse botão, dispara-se o evento **btnProcessar_Click**. Perceba que em todas as tags existe o atributo **runat="server"** que vai fazer com que essas linhas sejam processadas no servidor antes da página ser enviada para o browser no usuário quando este fizer a requisição como vimos acima.

Todo evento em Asp.Net tem dois parâmetros, **Object Sender** e **EventArgs**, que representam o objeto que disparou o evento e argumentos (parâmetros adicionais) dessa chamada.

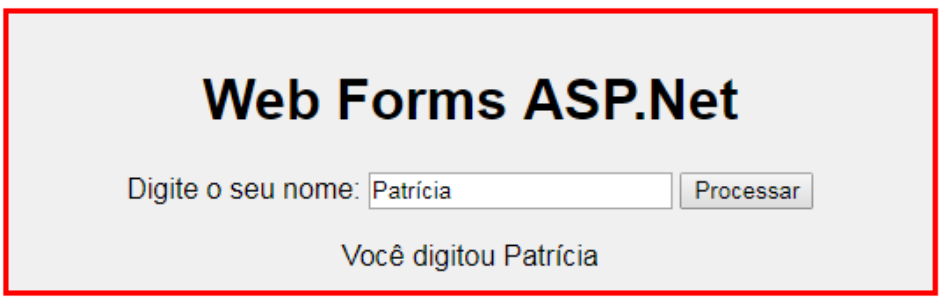
Existe um evento **Page_Load**, que é disparado sempre que se carrega a página. Verifique no arquivo **index.aspx.cs**.

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

        }

        protected void btnProcesar_Click(object sender, EventArgs e)
        {
            lblNome.Text = "Você digitou " + txtNome.Text;
            pnlNome.Visible = true;
        }
    }
}
```

Arquivo **index.aspx.cs**



```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="index.aspx.cs"
Inherits="WebForms_exemplo.index" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <link rel="stylesheet" type="text/css" href="Content/estilo.css" />
    <title></title>
</head>
<body>

    <form id="form1" runat="server">
        <div>
            <h1>Web Forms ASP.Net</h1>
            <asp:Label ID="lblMensagem" runat="server" Text="Digite o seu nome: "></asp:Label>
            <asp:TextBox ID="txtNome" runat="server"></asp:TextBox>
            <asp:Button ID="btnProcesar" runat="server" Text="Processar"
OnClick="btnProcesar_Click" />
        </div>
        <br />
        <asp:Panel ID="pnlNome" runat="server" Visible="False">
            <asp:Label ID="lblNome" runat="server"></asp:Label>
        </asp:Panel>
    </form>
</body>
</html>
```

Arquivo **index.aspx**

Todo controle de servidor é postado no servidor e processado. Todos possuem a propriedade id, que é seu nome. Devem ter a propriedade runat=”server” configurada, para que funcionem.

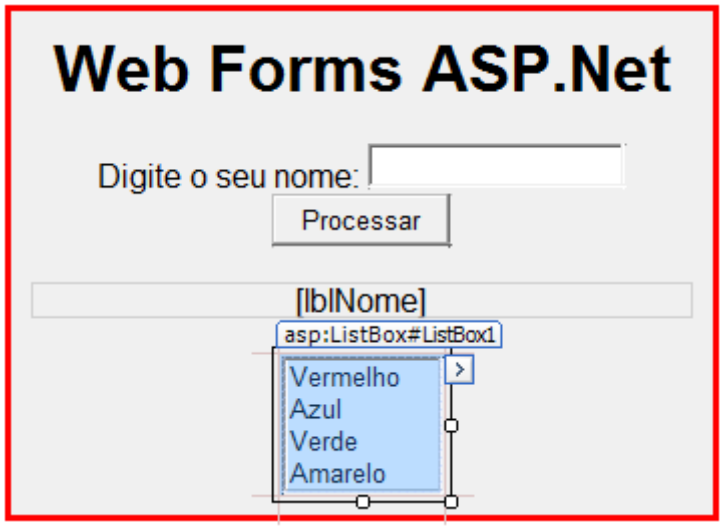
Id	nome do controle, para acesso em eventos
Text	texto exibido pelo controle, pode ser alterado em eventos
Visible	indica se o controle está ou não visível
Enabled	indica se o controle, mesmo visível, está disponível para uso ou não
BackColor	cor de fundo do controle
BorderColor	cor da borda do controle
BorderWidth	espessura da borda, em pixels
ForeColor	cor de frente
Font	fonte usado para o texto do controle
ToolTip	texto exibido quando o mouse passa sobre o controle

ListBox

Exibe uma lista de opções, como a tag Select do HTML. É composto por vários itens asp:ListItem, cada um exibindo uma das opções.

O evento **OnSelectedIndexChanged** é disparado quando se muda o item selecionado e o formulário é postado para o servidor (geralmente quando se pressiona um botão ocorre o post).

Acrescentaremos o seguinte ListBox em nosso exemplo:



No arquivo **index.aspx** teremos as seguintes modificações:

```
        <asp:Panel ID="pnlNome" runat="server" Visible="False">
            <asp:Label ID="lblNome" runat="server"></asp:Label>
        </asp:Panel>
        <br />
        <asp:ListBox ID="ListBox1" runat="server"
OnSelectedIndexChanged="ListBox1_SelectedIndexChanged" AutoPostBack="true">
            <asp:ListItem  Value="Vermelho">Vermelho</asp:ListItem>
            <asp:ListItem  Value="Azul">Azul</asp:ListItem>
            <asp:ListItem  Value="Verde">Verde</asp:ListItem>
            <asp:ListItem  Value="Amarelo">Amarelo</asp:ListItem>
        </asp:ListBox>
    </form>
</body>
</html>
```

Index.aspx

No código acima foram acrescentados duas propriedades ao ListBox: **OnSelectedIndexChanged** e **AutoPostBack**. A propriedade **OnSelectedIndexChanged** se refere ao evento que invocará o script **ListBox1_SelectedIndexChanged** listado abaixo. A propriedade **AutoPostBack** faz com que o valor selecionado seja enviado ao servidor e o evento onSelectedIndexChanged seja executado. Sem o uso dessa propriedade, temos que usar um botão que envie os dados para o servidor, num evento OnClick.

Também acrescentaremos o seguinte código apenas para exemplo no arquivo **index.aspx.cs**:

```
        protected void ListBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            switch (ListBox1.SelectedValue)
            {
                case "Vermelho":
                    {
                        lblNome.Text = "Você digitou " + ListBox1.SelectedValue;
                        pnlNome.Visible = true;
                        break;
                    }
```

```
                case "Azul":
                    {
                        lblNome.Text = "Você digitou " + ListBox1.SelectedValue;
                        pnlNome.Visible = true;
                        break;
                    }
                 case "Verde":
                    {
                        lblNome.Text = "Você digitou " + ListBox1.SelectedValue;
                        pnlNome.Visible = true;
                        break;
                    }
                  case "Amarelo":
                    {
                        lblNome.Text = "Você digitou " + ListBox1.SelectedValue;
                        pnlNome.Visible = true;
                        break;
                    }
            }


        }
```

Index.aspx.cs

Fonte:

- Desenvolvimento de Aplicações para Internet I - ASP.Net com C# (Apostila Prof. Francisco da Fonseca Rodrigues – Agosto/2017)
- Getting Started with ASP.NET 4.5 Web Forms and Visual Studio 2013
- ASP.NET Page Life Cycle Overview