



Introdução e conceitos básicos

ASP.NET - evolução ou enganação ???

O cenário atual : ASP - Active Server Pages - Por fora bela viola por dentro...

Como diria um ditado popular - "Tudo tem um preço. ". A ASP está para programação para Web como o Visual Basic está para a programação para Windows. Você já tentou aprender Perl ? Já tentou usar CGI ? Tente meu caro amigo e depois me diga se o que eu estou escrevendo não é a pura verdade...

A grande virtude da tecnologia ASP foi tornar acessível a muitas pessoas a tarefa de criar sites dinâmicos com acesso a dados de forma simples e descomplicada. GANHOU-SE em produtividade e rapidez, mas em qualidade e segurança...

Mesmo tornando as coisas mais fáceis , criar aplicações usando ASP não é uma tarefa tão simples se comparada com a ferramenta Visual Basic. No Visual Basic , se você for criar um formulário com caixas de texto e botões qual o procedimento ? Ora, você arrasta os controles para o formulário e pronto ... Na ASP as coisas não são tão simples assim ...

Creio que a Microsoft percebeu isto e apresentou , junto com a plataforma .NET , a evolução para ASP : ASP.NET.

O que é então a ASP .NET ?

Podemos dizer que ASP .NET é a próxima geração da ASP e tem o objetivo de poder ser usada para criar sites de grande escala comercial como pequenas aplicações para intranet de uma maneira simples e fácil.

Alguns benefícios da ASP.NET :

- Páginas ASP.NET são compiladas - Quando uma página ASP.NET é requisitada ela é compilada e vai para o cache do servidor ; são assim carregadas mais rápidas
- Páginas ASP.NET são construídas com controles de interface do lado do servidor : Controles de interface básicos (TextBox , Label , etc...) ; Controles de validação , Controles de Dados (DataGrid, etc..) , Controles mais complexos (Calendários , ad rotator , etc..)
- ASP.NET é parte do .NET Framework - O .NET Framework torna disponível mais de 3000 classes que podem ser usadas para as aplicações ASP.NET . Classes para gerar imagens, enviar email, etc... Como a ASP.NET faz parte do .NET Framework todas essas classes podem ser usadas dentro de uma página ASP.NET
- ASP.NET é totalmente orientado a objeto
- Com o Visual Studio .NET o ambiente integrado permite criar uma página apenas arrastando e soltando os controles no formulário Web.

A ASP.NET traz então o desenvolvimento das ferramentas RAD - **Rapid Application Development**- orientado a componentes para a WEB pois fornece : [Web Forms](#) , [Web Controls](#) e [XML Web Services](#).

ASP.NET é igual a ASP ?

Essa você mesmo vai responder. Como ? vamos comparar os códigos de duas páginas que realizam a mesma tarefa : uma feita em ASP e outra em ASP.NET.

- A clássica página para exibir as horas: em ASP e ASP.NET

<pre><HTML> <BODY> <% If Hour(Date) > 12 Then Response.Write("Boa Tarde") Else Response.Write("Bom Dia") End If %>
São <%=Time%> </BODY> </HTML></pre>	<pre><%@ Page Language="VB" %> <script language="VB" runat="server"> Sub Page_Load(Sender AS Object, E as EventArgs) If Hour(Now) >= 12 Then resposta.text= "Boa Tarde" Else resposta.text= "Boa Dia" End If resposta.text = resposta.text & "
 São " & Now() End Sub </script> <HTML> <BODY> <asp:label id="resposta" runat="server"/> </BODY> </HTML></pre>
Código em ASP	Código em ASP.NET

Pode parecer igual , mas não é. É mais estruturado e ... complexo (a primeira vista).

Embora o código ASP seja muito simples (ridículo até) podemos identificar nele as mazelas da ASP:

1. O código é executado linha por linha
2. Temos misturado o código da página (HTML) e o da lógica da página.
3. E difícil reaproveitar o código (embora , devido a simplicidade do mesmo isto não esteja tão evidente)
4. Ele depende da plataforma . (Funciona perfeitamente no IE , e no Netscape ? Opera ? ., etc..)
5. O VBScript , a principal linguagem de script ASP, não diferencia os tipos de variáveis.

O mesmo código em ASP.NET já evidencia o que vem por ai:

1. O código da lógica da página esta separado da interface
2. O código esta estruturado e legível
3. Logo na primeira linha temos novidades: A indicação de qual linguagem estamos usando. No caso VB , mas podemos usar : C# , VB , C++ , etc..
4. Com ASP.NET temos a programação orientada a eventos

Como usar e testar.

A ASP.NET , pelo menos na versão beta 2, NÃO roda no Windows 95 ,98 e ME. Você vai precisar criar coragem e instalar o Windows 2000 , NT ou o XP com o IIS - [Internet Information Service](#) - instalado. (*O ideal é você criar um diretório Virtual onde vai salvar as suas páginas*)

Vai precisar instalar o .NET Framework e alguns pacotes para atualizar o seu sistema. Abaixo o link para fazer o download: (veja também a dica : [.NET Framework - downloads Grátis](#))

<http://msdn.microsoft.com/downloads/default.asp?url=/downloads/sample.asp?url=/msdn-files/027/000/976/msdncompositedoc.xml>

Você pode testar suas páginas em ASP.NET usando o servidor Brinkster - www.brinkster.com - ele hospeda gratuitamente páginas ASP.NET. *(Pelo menos até esta data...)*

Os arquivos criados em ASP.NET possuem a extensão **.aspx** . Você pode usar **qualquer editor de texto** para digitar o código de uma página ASP.NET. Assim se você abrir o bloco de notas e digitar o seguinte código:

- Código ASP.NET para a página que exibirá a mensagem: "**Minha primeira página ASP.NET**"

```
<%@ Page Language="VB" %>

<script language="VB" runat="server">
Sub Page_Load(Sender AS Object, E as EventArgs)
    lblmensagem.Text = "Minha primeira página ASP.NET"
End Sub
</script>

<HTML>
<BODY>
    <asp:label id="lblmensagem" runat="server" />
</BODY>
</HTML>
```

Basta salvar o arquivo com a extensão **.aspx** : Exemplo : **primeiro.aspx** e rodar no servidor IIS.

Obs: *Você pode usar ASP e ASP.NET juntos em uma mesma máquina . Não haverá problemas elas rodarem independente uma da outra.*

ASP.NET uma "linguagem" compilada

O quer dizer "*linguagem*" *compilada* ?

Para entender isto devemos voltar nossos olhos para a plataforma .NET; Na verdade , a grande jogada da Microsoft é a plataforma .NET sendo apresentada como uma nova plataforma sobre a qual podemos desenvolver nossos sistemas voltados para um ambiente distribuído via WEB.

O .NET Framework é a base da plataforma .NET , o ambiente onde podemos criar e executar nossas aplicações quer sejam elas aplicações Web , VB , C# .

Quando criamos uma página ASP.NET , **na primeira vez que ela for executada ou alterada** o código é compilado para um código intermediário , chamado **MSIL** (*Microsoft Intermediate Language*) não importa se você usou VB , C# ou C++ para criar a sua página.

Após o código **MSIL** ter sido criado ele é entregue ao **.NET Framework** que fará a conversão para a linguagem binária e em seguida executar o código.

Esta conversão é feita pelo **CLR - Common Language Runtime** - que gerencia todo o serviço necessário . (memória, tipo de dados , exceções , código , etc..) .
Veja esquema abaixo:



Todo este processo ocorre de forma transparente ao usuário final que nem precisa saber nada sobre o processamento.

Instalando e configurando o Internet Information Services (IIS)

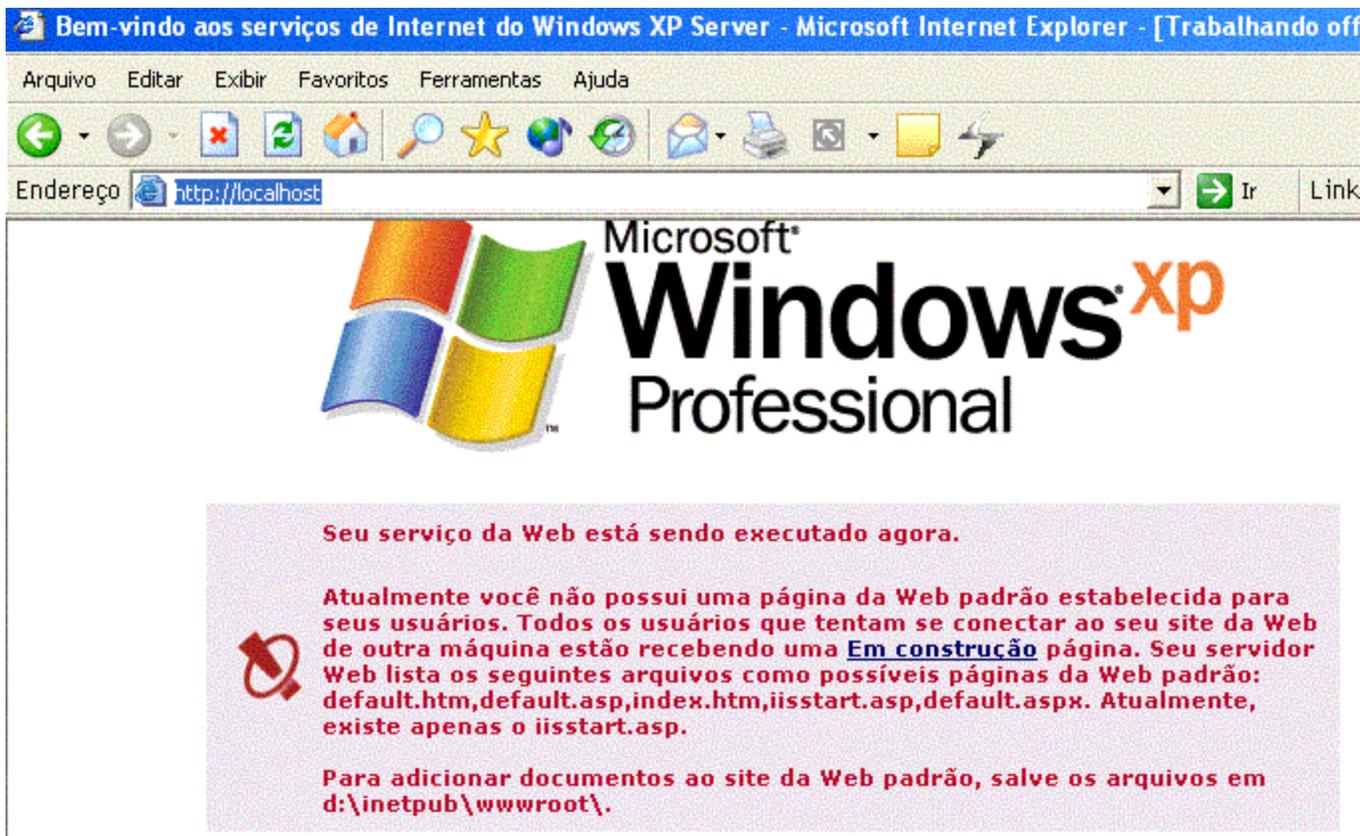
Ao utilizar páginas ASP , quer como programador quer como usuário final , para poder visualizar e testar as páginas ASP você precisava ter um servidor web configurado em sua máquina local. O mais usado para a plataforma Windows era , e é , o [Personal Web Server \(PWS\)](#). Para lembrar como configurar o PWS leia o artigo : [Como configurar o Personal Web Server ?](#)

Se você quer criar projetos Web e testar páginas ASP.NET em sua máquina local vai precisar , da mesma forma , configurar um servidor Web na sua máquina local . Para a plataforma Windows você vai precisar ter o [Internet Information Services - IIS](#) , instalado. Embora o **IIS** faça parte do sistema operacional Windows (*Windows 2000 , Windows XP , Windows NT*) ele não é instalado automaticamente quando da instalação do Windows.

O IIS para Windows XP Professional é distribuído junto com o CD do Windows XP só pode servir a 10 conexões simultâneas e não aproveita todas os recursos do servidor e suporta os seguintes recursos :

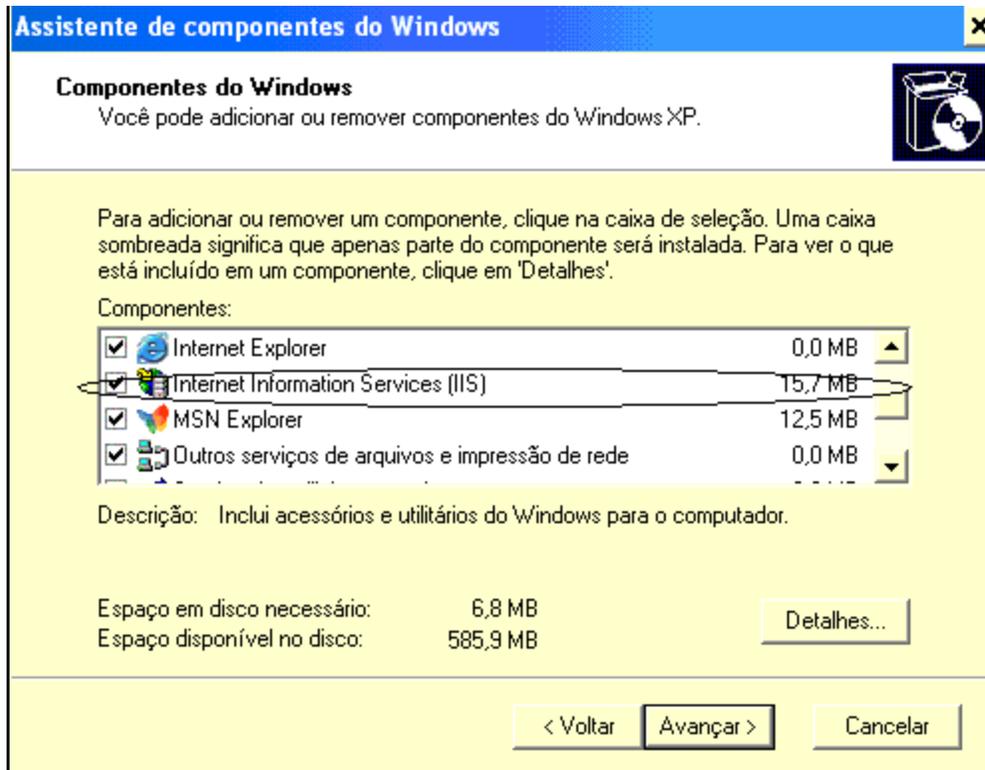
- ASP e PHP
- SSI ([Server sides Includes](#))
- Controles ActiveX
- scripts ISAPI([Internet Server API](#)) e CGI([Common Gateway Interface](#))
- Acesso a banco de dados
- SSL ([Secure Sockets Layer](#))

Para verificar se o **IIS** esta instalado na sua máquina , abra o seu Browser e digite a seguinte solicitação na caixa de endereço : <http://localhost> . Você deverá obter a tela da figura abaixo se o IIS esta instalado e ainda não foi definida nenhuma página como página padrão a ser exibida:



Se ao invés desta imagem você recebeu uma tela com uma mensagem de erro, vai ter que instalar o IIS na sua máquina. Vamos lá ...: *(estou usando o Windows XP Professional)*

- No Windows clique no botão - **Iniciar** - e a seguir em - **Painel de Controle**
- Clique agora na opção - **Adicionar ou Remover Programas**.
- A seguir clique em - **Adicionar/Remover componentes do Windows**
- Procure por - **Internet Information Service** - na lista do Assistente de componentes do Windows.



- No meu caso , como o **IIS** já esta instalado ele esta marcado , no seu caso você deverá marcar a opção e clicar em Avançar

- A seguir basta clicar no botão Avançar e seguir as instruções do Assistente de instalação.

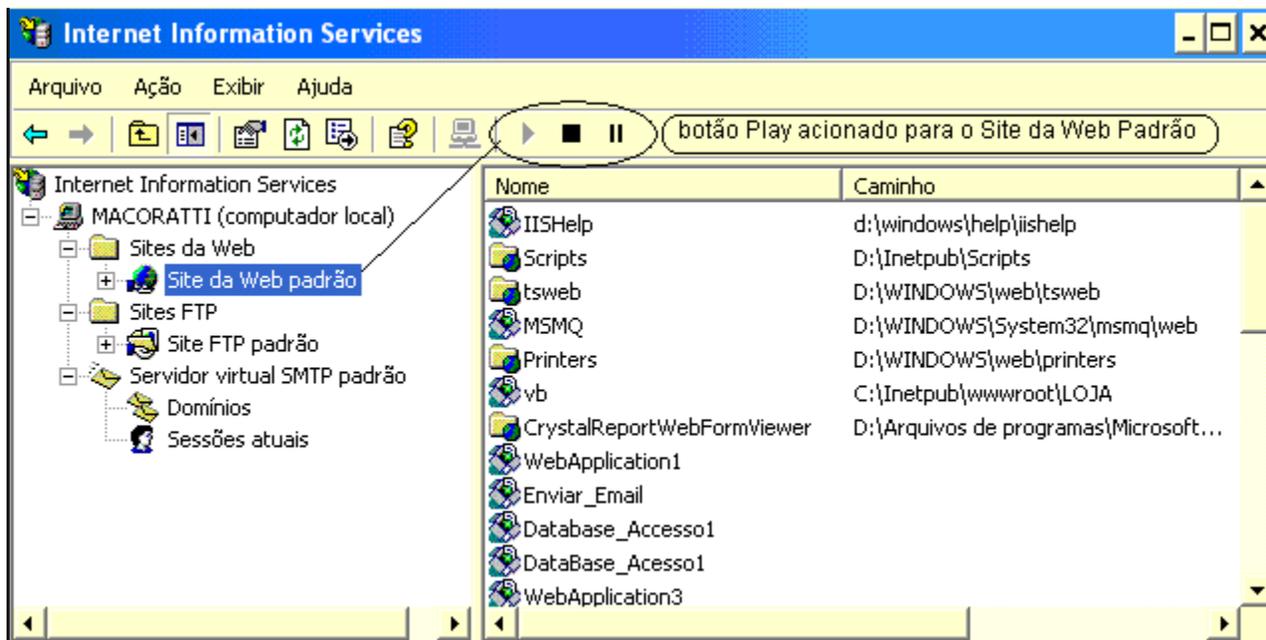
Obs: Antes de instalar o IIS você deve ter o protocolo **TCP/IP** instalado.

Após a instalação o IIS cria uma área de administração para gerenciar as suas Webs. Vamos acessá-la :

1. Clique em - [Iniciar/Painel de Controle](#)
2. A seguir acesse o ícone - [Desempenho e Manutenção](#) - e escolha - [Ferramentas Administrativas](#)
3. Finalmente clique em - [Internet Information Services](#). Abaixo temos a imagem da tela que deve ser exibida:



Por padrão o diretório pessoal de web site é o diretório físico : [c:\inetpub\wwwroot](#) e arquivo **localstart.asp**

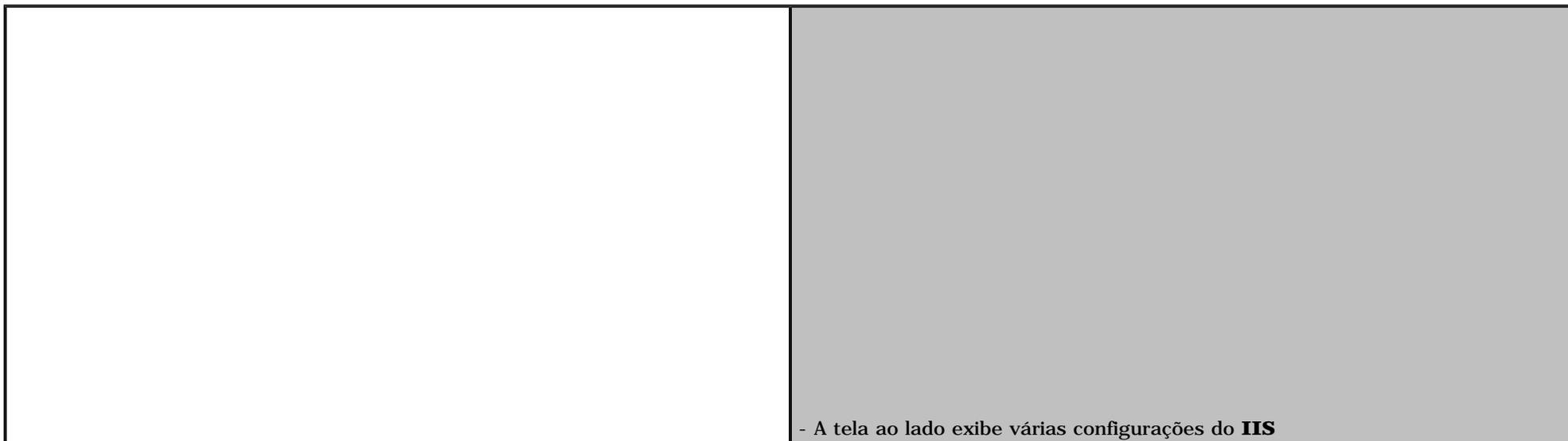


Portanto o IIS cria automaticamente um Web Site Padrão com as extensões do servidor instaladas no diretório `c:\inetpub\wwwroot`

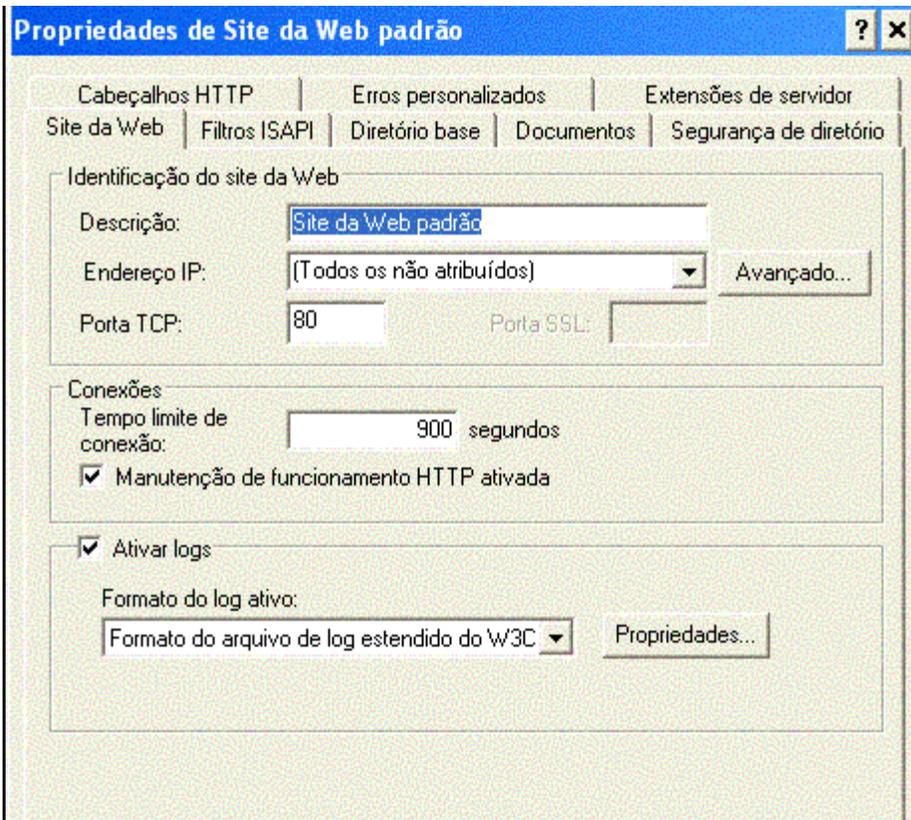
Podemos escolher um dos sites instalados, por exemplo, o **Site da Web padrão**, clicando nele e ver o status do serviço (figura acima) onde temos o botão play acionado indicando que o serviço Web esta ativo e que sua home page pode ser encontrada no endereço <http://localhost> ou http://nome_do_seu_computador ou ainda <http://127.0.0.1> referindo-se a sua própria máquina.

Configurando o IIS

Podemos também configurar algumas propriedades dos sites instalados; vamos clicar com o botão direito do mouse sobre o **Site da Web padrão** e selecionando o menu Propriedades. A tela abaixo mostra a janela obtida:



- A tela ao lado exhibe várias configurações do **IIS**



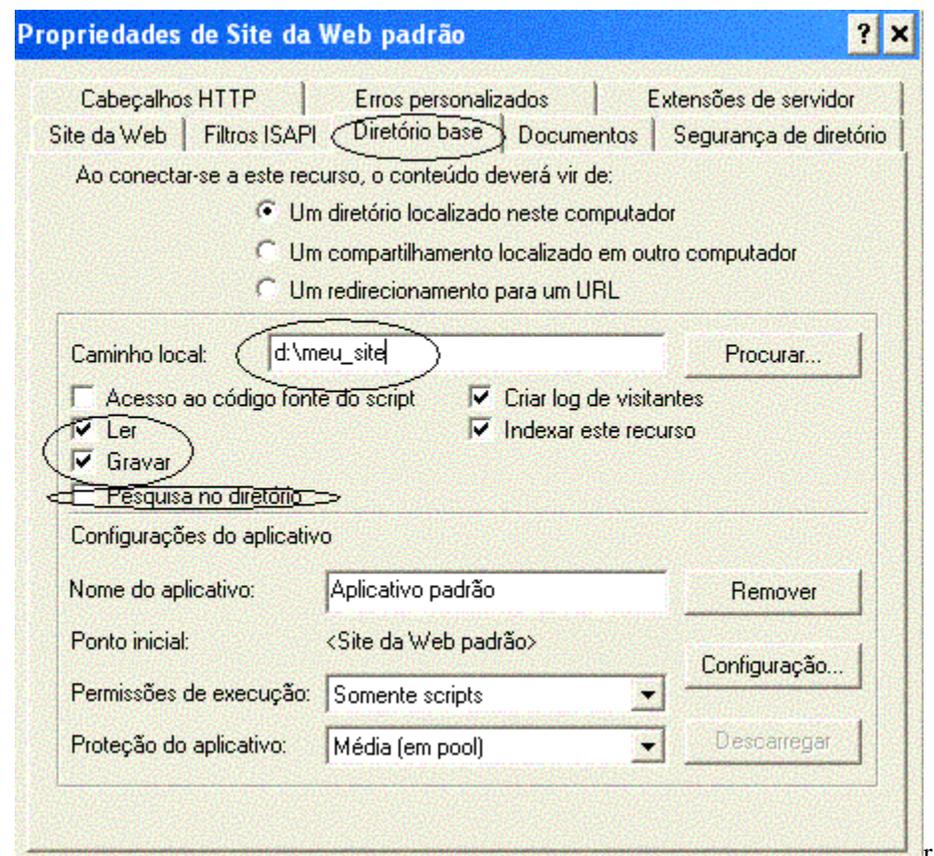
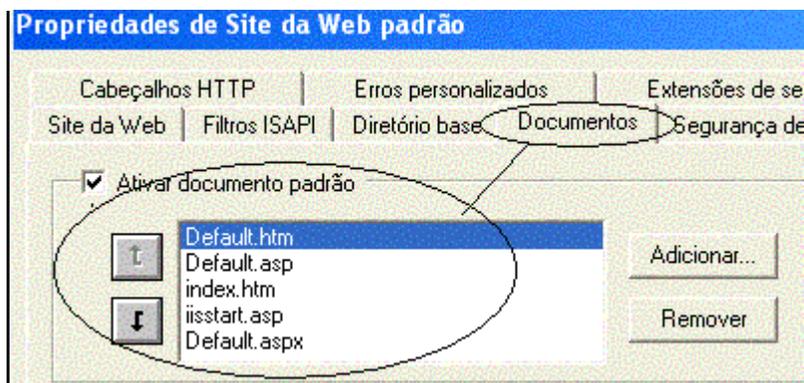
- Vamos focar apenas algumas delas para dar um exemplo.

- Assim se quisermos alterar o diretório padrão : <c:\inetpub\wwwroot> devemos acessar a aba - **Diretório Base** - (conforme figura abaixo)

- Ao lado temos um exemplo onde alteramos o diretório padrão para d:\meu_site e damos acesso para leitura e gravação

- Se ativarmos a opção - **Pesquisa no diretório** - quando algum usuário acessar o este site , se o mesmo não tiver uma página padrão definido , será exibido uma listagem com todos os nomes de documentos contidos na pasta.

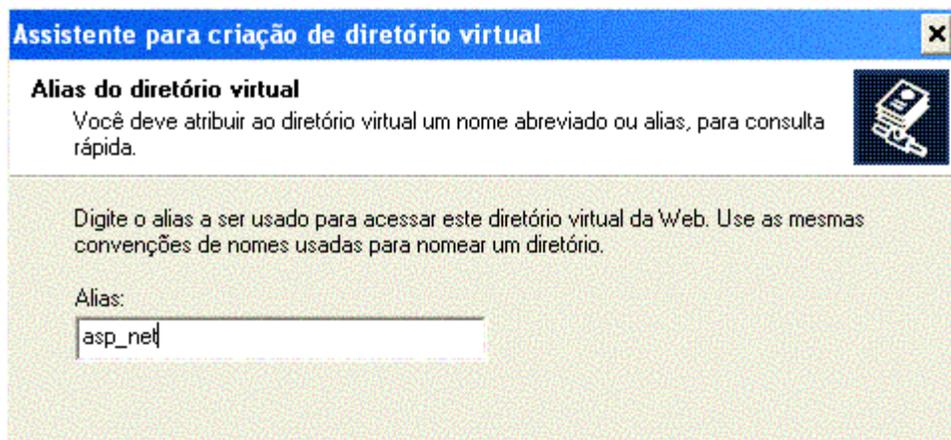
- Na aba - **Documentos** - podemos definir qual o nome padrão iremos dar a primeira página ocultando assim esta extensão. (ver figura abaixo)



Criando diretórios virtuais

Ao criar suas novas páginas Web você pode colocá-las no diretório d:\inetpub\wwwroot ou em um de seus subdiretórios ; assim para você criar um novo site com extensões do servidor siga o seguinte roteiro:

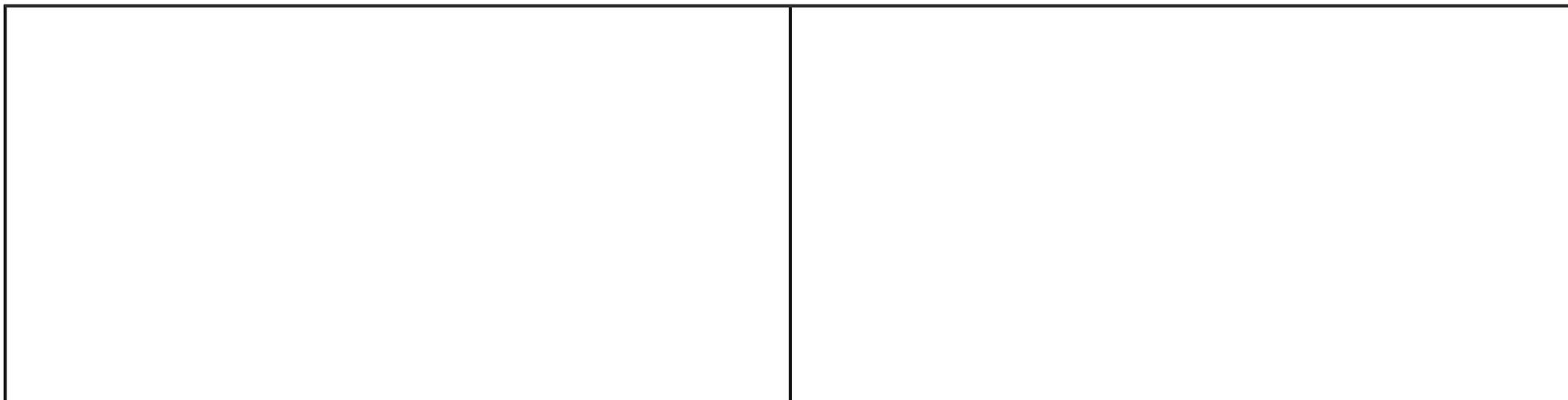
1. clique com o botão direito do mouse sobre a **Pasta web Padrão** e selecione - **Novo** - e a seguir **Diretório Virtual**
2. Clique no botão **Avançar** do Assistente que irá surgir
3. Agora informe o aliás que irá identificar o seu diretório virtual. No nosso caso eu informei o alias : [asp_net](#)

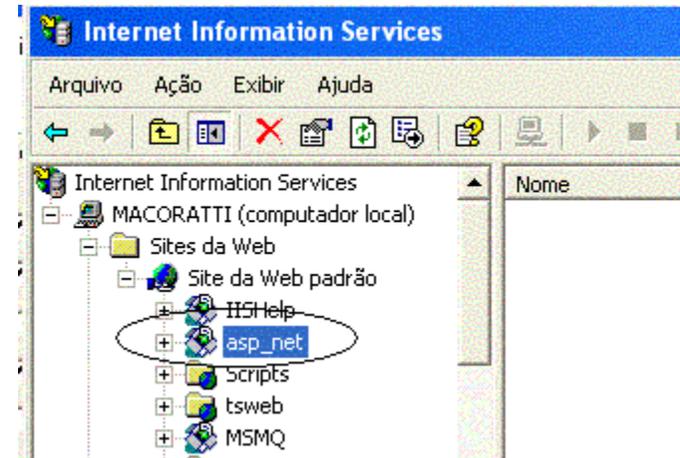
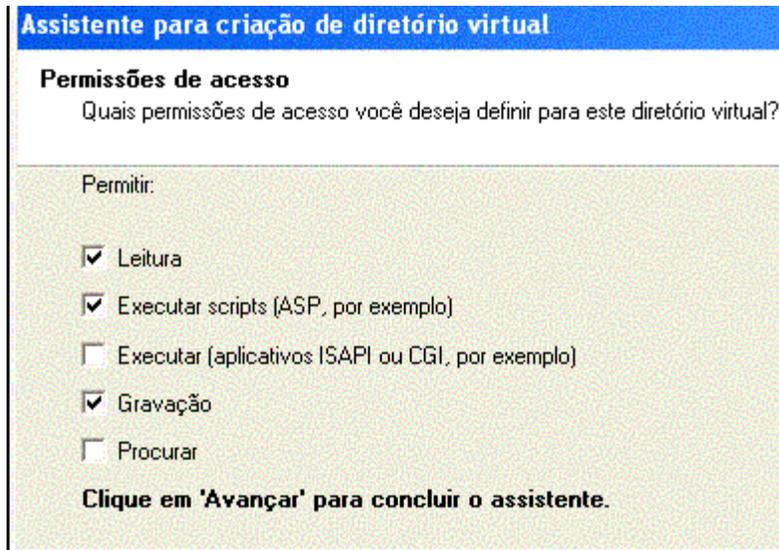


4. Informe a seguir o caminho do diretório onde o seu site vai ser armazenado. No exemplo da figura abaixo eu informei : **d:\inetpub\wwwroot\aspnet**.



5. Defina a seguir as permissões de acesso conforme a esquerda abaixo. Na figura da direita abaixo temos o resultado exibindo o novo site criado. - [asp_net](#).





Um diretório virtual representa um diretório físico na web e não precisa usar o mesmo nome que o diretório físico e também não precisa ser necessariamente um subdiretório de d:\inetpub\wwwroot. Assim você pode criar o diretório c:\sites_web dando a ele o nome de diretório virtual - Sites. Para acessar o seu site bastaria informar **http://localhost/Sites**

Quando você cria um diretório virtual com o IIS ele também é marcado como um **aplicativo Web**, com isto os arquivos ASP.NET nesse diretório *serão executados por si próprios*, usarão o seu próprio conjunto de dados de sessão local e terão os seus próprios ajustes de configuração.

Pronto !!! esta tudo preparado para você desenvolver projetos Web e testá-los em sua máquina local usando o **IIS**.

Minha primeira página ASP.NET.

Lembra da **ASP - Active Server Pages** ?? Que tal recordar ?

O que é ASP ?

ASP é uma tecnologia de scripts que roda no servidor e permite que os scripts embutidos em uma página HTML sejam executados por um servidor WEB.

- ASP é uma tecnologia da Microsoft
- ASP significa - **Active Server Pages**
- ASP roda sobre o contexto do - **IIS** - **Internet Information Server**
- IIS é um componente que vem com o Windows 2000, Windows XP e é parte do Windows NT 4.0 Option Pack
- ASP também pode ser executado sob o servidor - **PWS** - **Person Web Server** (uma versão reduzida do **IIS**)
- **PWS** pode ser encontrado nos CD's do Windows **95/98**, no site da Microsoft ou no [Super CD ASP Total](#)

O que é um arquivo ASP ?

- Um arquivo ASP é apenas um arquivo do tipo HTML.
- Um arquivo ASP pode conter texto, HTML, XML, e scripts

- Os scripts de um arquivo ASP são executados no servidor
- Um arquivo ASP tem a extensão ".asp"

Como ASP Funciona ?

- Quando um Navegador (Internet Explorer , Netscape, Opera...) requisita um arquivo **HTML** o servidor apenas retorna o arquivo HTML.
- Quando um Navegador (Internet Explorer , Netscape, Opera...) requisita um arquivo **ASP** o servidor (IIS, PWS,...) passa a requisição para **ASP.DLL** (que esta no servidor)
- O **ASP.DLL** lê o arquivo linha por linha e executa o(s) script(s) presente(s) no arquivo ASP.
- Ao final o servidor (IIS,PWS,...) retorna o arquivo ASP para o Navegador no formato HTML.

O que é ASP.NET ?

- ASP.NET ou ASP+ ou ASP 3.0 é a última versão da ASP.
- ASP.NET é a próxima geração ASP
- ASP.NET não é uma [atualização](#) da última versão ASP
- ASP.NET é novo paradigma para utilização de scripts no lado do servidor.
- ASP.NET é parte da plataforma **.NET Framework**

O que é o .NET Framework ?

- O **.NET Framework** é a infraestrutura para a nova plataforma .NET
- O .NET Framework é um ambiente comum para construir, desenvolver e executar aplicações WEB e WEB Services
- O .NET Framework contém uma linguagem comum de runtime - **CLR - Common Language Runtime** - , e livrarias de classes comuns - **NET, ASP.NET e Windows Forms** - que fornece serviços avançados que podem ser integrados em uma variedade de sistemas operacionais
- O .NET Framework fornece também um poderoso ambiente para desenvolvimento de aplicações , simplificando o desenvolvimento e de fácil integração com um diferentes linguagens.
- O .NET Framework atualmente suporta : **C++, C#, Visual Basic, and JScript** (A versão da Microsoft para o JavaScript).
- O Microsoft Visual Studio.NET é o ambiente de desenvolvimento comum para o **.NET Framework**

ASP.NET é melhor ?

- ASP.NET possui uma melhor linguagem de suporte e um grande número de novos controles e componentes baseados em XML além de possui um melhor uso da autenticação.
- ASP.NET aumenta o desempenho pois roda o código compilado.
- ASP.NET usa a nova tecnologia ADO.NET
- ASP.NET suporta a linguagem Visual Basic completa. (não suporta VBScript)
- ASP.NET suporta C# (C Sharp) e C++.
- ASP.NET suporta JScript
- ASP.NET apresenta maior escalabilidade.
- ASP.NET é fácil de configurar e de usar.
- ASP.NET contém um grande conjunto de controles HTML
- ASP.NET contém um novo conjunto de objetos orientados para controles de entrada de dados , como controles para validações e controles de listas.
- ASP.NET apresenta o novo controle data grid que suporta ordenação , paginação e muitos mais recursos.

Como criar páginas ASP.NET ?

Uma página ASP.NET é similar a uma página HTML. Veja abaixo o código para a página - **Ola.htm** - e o resultado da exibição da página no Navegador internet Explorer:

```
<html>
<body bgcolor="blue">
<center>
<h4>Olá Pessoal - isto é uma página HTML.</h4>
</center>
</body>
</html>
```



Para converter a página HTML em uma página ASP.NET basta copiar o arquivo com a extensão **.aspx**. Então renomeando o arquivo - Ola.htm - para Ola.aspx acabamos de criar uma página ASP.NET. O código continha igual e o resultado do processamento também. Verifique e compare : [Ola.htm](#) x [Ola.aspx](#).

Como funciona ?

Fundamentalmente uma página **ASP.NET** é idêntica a uma página HTML.

Uma página HTML possui a extensão **.htm** ; se um Navegador requisita uma página HTML do servidor o servidor envia a página para o Navegador sem nenhuma alteração.

Uma página ASP.NET possui a extensão **.aspx** ; se um Navegador requisita uma página ASP.NET , o servidor processa qualquer código script contido na página e devolve o resultado ao Navegador.

A página ASP.NET - **Ola.aspx** - não contém nenhum script a ser executado , então nada é executado.

A página - **Ola.htm** - é uma página HTML estática e a página ASP.NET - **Ola.aspx** - também será uma página estática. A seguir vamos mostrar como criar páginas ASP.NET dinâmicas.

ASP.NET - criando páginas dinâmicas

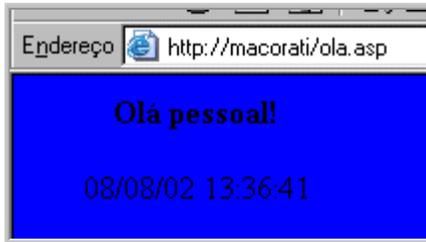
Vamos começar mostrando o código de uma página dinâmica em ASP. Abaixo temos o código da página - **Ola.asp** - (as páginas ASP possuem a extensão **.asp**). Destacamos em azul o código que representa script que será executado pelo servidor . O código script vem entre as tags - **<% %>** e é o código que será executado pelo servidor.

```
<html>
<body bgcolor="blue">
<center>
<h4>Olá pessoal!</h4>
<p><%Response.Write(now())%></p>
</center>
</body>
</html>
```

- **response.write** é um código ASP que devolve uma saída em HTML

- **Now()** é uma função que retorna a data e hora do servidor.

O resultado do processamento do arquivo **Ola.asp** é o seguinte:

	<p>O arquivo ola.asp foi executando no PWS.</p> <p>- A direita temos o código da página ola.asp após o processamento. (<i>Perceba que o código é puro HTML.</i>)</p>	<pre><html> <body bgcolor="blue"> <center> <h4>Olá pessoal!</h4> <p>08/08/02 13:36:41</p> </center> </body> </html></pre>
---	--	---

Para converter a página dinâmica gerada por **ola.asp** em uma página ASP.NET dinâmica basta renomear o arquivo para **Ola.aspx**. O código continua idêntico e não sofre nenhuma alteração. Verifique você mesmo executando as duas páginas e comparando o resultado: [Ola.asp](#) x [Ola.aspx](#)

Ora, então não há nenhuma diferença entre uma página ASP e uma página ASP.NET ?

Bem , vamos responder a esta pergunta.

Como já repetimos acima o arquivo **Ola.asp** possui **tags** especiais que contém código script que o servidor irá executar. No caso do arquivo Ola.asp existe uma limitação que talvez você não tenha percebido : as tags (marcadores) precisam ser colocadas onde você quer que o resultado apareça !

Para fazer isto as tags e os scripts ASP devem ficar misturados com o código HTML, ou seja , é **impossível separar o código script que será executado do código HTML**. Isto leva a um código de difícil leitura e manutenção ; o que convencionou-se chamar de *código espeguetti*.

A ASP.NET resolveu este problema com os **Server controls**. **Server controls** são **tags** que podem ser interpretadas pelo servidor. Existem três tipos de **Server Controls**

- **HTML Server Controls** - Tags HTML tradicionais
- **Web Server Controls** - Novas tagas ASP.NET .
- **Validation Server Controls** - Premitem a validação da entrada de dados.

1- ASP.NET - HTML Server Controls

Os controles de servidor - **server Controls** - HTML são tags HTML padrão , com exceção de possuírem o atributo : **runat="server"** . Vejamos um código que ilustra isto:

```
<%
TimeStamp.InnerText=now( )
%>
<html>
<body bgcolor="aqua">
<center>
<h4>Olá, Pessoal isto é uma página ASP.NET !</h2>
<p id="TimeStamp" runat="server"></p>
</center>
</body>
</html>
```

HTML Server Controls - clique no link para visualizar a página : [aspnet1.aspx](#)

O atributo **runat="server"** no exemplo acima , tornou a **tag <p>** em um controle do servidor - **server control**. O **atributo id** dá nome ao controle e torna possível fazer uma referência a ele no código executável. Agora perceba que o código foi movido para fora do código HTML.

2- ASP.NET - Web Server Controls

Web Server controls são iguais ao HTML server controls , mas são mais complexos. Eles não agem como parte de uma tag HTML existente mas tem existência independente. São frequentemente usados em aplicações interativas como formulários para entrada de dados. **Web server controls** são **tags** que começam com **<asp: .** Vejamos um exemplo a seguir:

```
<%
TimeStamp.Text=now( )
%>
<html>
<body bgcolor="aqua">
<center>
<h4>Olá Pessoal , isto é uma página ASP.NET !</h4>
<p><asp:label id="TimeStamp" runat="server" /></p>
</center>
</body>
</html>
```

Web server controls - clique no link para visualizar a página : [aspnet2.aspx](#)

No código acima o Web server control usou o código **asp:label** . Este é um dos muitos controles do servidor(**server controls**) predefinidos que podem ser interpretados pela ASP.NET.

3- ASP.NET - Validation Server Controls

Os controles de servidor para validação - **Validation server controls** - permitem a você validar uma entrada em controle de entrada de dados do servidor(*TextBox*) e exibir uma mensagem quando a validação falha.

Cada controle de validação realiza uma tarefa específica de validação. Por padrão uma página de validação é executada quando um controle *Button*, *ImageButton*, or *LinkButton* é clicado. Você pode prevenir a validação configurando a propriedade **CausesValidation** como igual a **False**.

ASP.NET - Eventos.

A ASP.NET possui manipuladores de eventos que certificam que o código seja executado no tempo apropriado. Vamos a um exemplo. Observe o código da página **aspnet3.aspx** abaixo e responda a seguinte pergunta :

Como você pode saber quando o código : `<%TimeStamp.Text=now() %>` será executado ?

```
<%  
TimeStamp.Text=now()  
%>  
<html>  
<body bgcolor="aqua">  
<center>  
<h2>Olá ! Esta é uma página ASP.NET.</h2>  
<asp:label id="TimeStamp" runat="server" />  
</center>  
</body>  
</html>
```

A resposta é : **Não dá para saber !!!** Sabe porque ?

Para ter certeza de que o código seja executado no tempo apropriado você deve [incluir um manipulador de evento](#). Vamos fazer isto no código acima para você entender melhor. Veja código do arquivo **aspnet3.aspx** abaixo já com o evento incluído:

```
<script runat="server">  
  Sub Page_Load(Sender As Object,E As EventArgs)  
    TimeStamp.Text=now()  
  End Sub  
</script>  
<html>  
<body bgcolor="aqua">  
<center>  
<h2>Esta é uma página ASP.NET com Evento.</h2>  
<asp:label id="TimeStamp" runat="server" />  
</center>  
</body>  
</html>
```

Gerenciando eventos -clique no link a seguir para ver o resultado - [aspnet3.aspx](#)

Um manipulador de eventos é uma subrotina que executa um código quando um determinado evento ocorrer. No código acima foi incluído o evento **Page_Load** que ocorre quando a página é carregada.

O evento **Page_Load** é um dos muitos eventos que a ASP.NET pode interpretar. Então , quando a página for carregada , ASP.NET chama a rotina **Page_Load** e executa o código associado ao evento. Simples não é mesmo ???

Criando seu primeiro Web Forms

Neste artigo vou falar sobre **Web Forms** ; um nome um tanto 'pomposo' que talvez possa intimidar aos iniciantes ; ledo engano a idéia por trás dos Web Forms e

criar um ambiente de desenvolvimento para a Web intuitivo e fácil de usar . Afinal o grande sucesso do Visual Basic deve-se ao fato de ao seu ambiente de desenvolvimento , onde criamos aplicativos arrastando e soltando controles em um formulário. Pois bem , a proposta dos Web Forms é exatamente esta , e , se você esta familiarizado com o ambiente de desenvolvimento do VB já saiu na frente pois o ambiente de desenvolvimento dos Web Forms 'lembra' muito o IDE do VB. 

Se você esta começando agora com ASP.NET deve ler os artigos anteriores(*veja os links abaixo*) onde eu abordei diversos conceitos básicos sobre ASP.NET:

- [Minha primeira página ASP.NET](#)
- [ASP.NET - evolução ou enganação ???](#)
- [Instalando e configurando o Internet Information Services \(IIS\)](#)

Os Web Forms são usados para desenhar a interface visual com o usuário de forma semelhante como você faz usando a interface IDE do VB. Sendo que uma página Web Forms pode usar qualquer recurso suportado pelo protocolo HTTP , XML , WML , etc..

Se você já trabalhou com ASP percebeu o quanto fica difícil trabalhar com a interface visual , o código HTML e os scripts com a lógica de programação ; no final você acaba sempre obtendo um código todo misturado , o famoso código espaguetei.

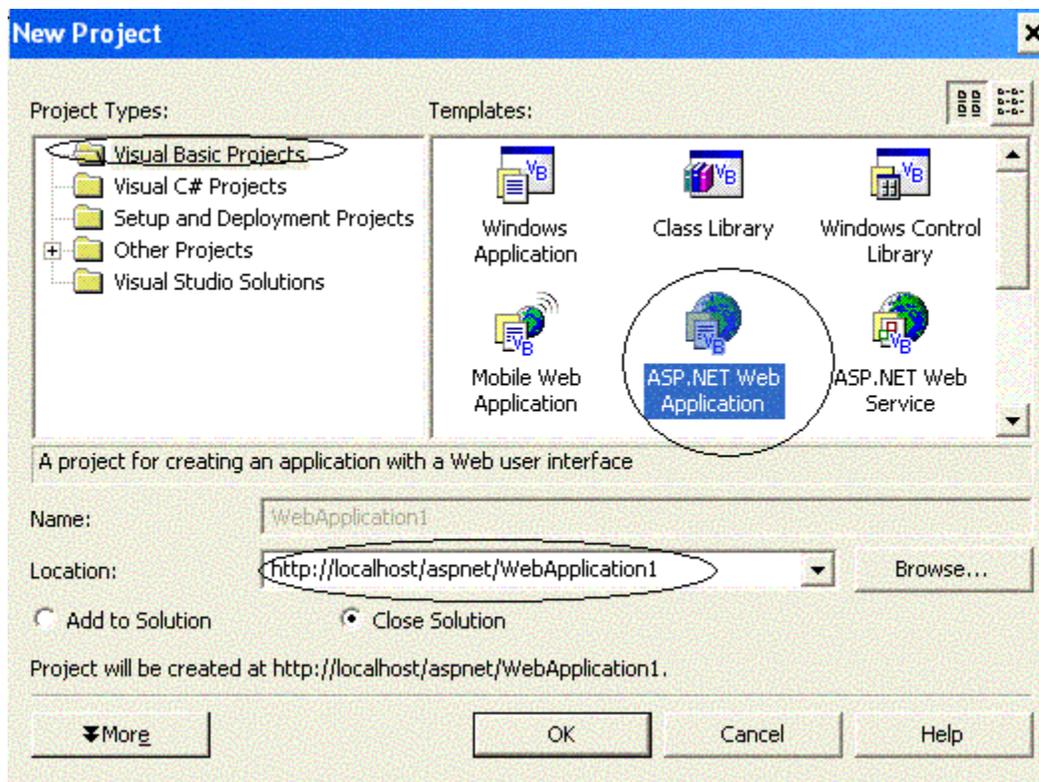
ASP.NET acaba com isto ; ASP.NET mantém a parte visual separada do código com a lógica da aplicação (regras de negócios) , assim você terá dois arquivos : um arquivo **aspx** com o código da parte visual da página e outro **aspx.vb** com lógica da aplicação. (*código-por-trás - Code-behind*).

Para criar Web Forms vamos usar controles disponíveis na toolBox do Visual Studio.NET são eles :

- controles HTML(HTML Controls) - pertencem ao namespace [System.Web.UI.HtmlControls](#) e derivam da classe base [HtmlControls](#)
- controles Web (ASP Controls) - pertencem ao namespace [System.Web.UI.WebControls](#) e derivam da classe base [WebControl](#)

Criando um novo projeto Web Form

1. Inicie um novo projeto no Visual Studio.NET : Selecione **New | Project** no menu **File**
2. Em **Project Types** selecione - [Visual Basic Projects](#)
3. Em **Templates** selecione - [ASP.NET application](#)
4. Na caixa **Name** informe o nome do projeto
5. Em **Location** informe o endereço do servidor Web onde o projeto será criado. (Conforme figura abaixo).



Nas figura abaixo exibimos os controles disponíveis na Toolbox: [Html Controls](#) e [WebForms Controls](#) , o aspecto inicial do [WebForms](#) e os arquivos adicionados ao projeto ASP.NET



The image shows a screenshot of the Visual Studio IDE. On the left is the 'Web Forms' toolbox with various controls like Pointer, Label, TextBox, Button, etc. The main area shows a 'WebForm1.aspx' page in design mode with a grid background and text explaining grid layout mode. Below this is a 'Solution Explorer' window showing the project structure for 'WebApplication1', including files like AssemblyInfo.vb, Global.asax, Styles.css, Web.config, WebApplication1.vsdisco, and WebForm1.aspx.

WebForm1.aspx

The page you are working on is in grid layout mode, and objects will be arranged using absolute (x and y) coordinates.

To use flow layout (top to bottom, as in a word processing document), change the pageLayout property of the DOCUMENT to FlowLayout.

O aspecto do formulário inicial do projeto ASP.NET

Solution Explorer - WebApplication1

Solution 'WebApplication1 (2)' (1 project)

WebApplication1

- References
- AssemblyInfo.vb
- Global.asax
- Styles.css
- Web.config
- WebApplication1.vsdisco
- WebForm1.aspx

Os arquivos adicionados ao projeto ASP.NET

Vejamos o significado de cada arquivo do projeto ASP.NET :

- **WebForm1.aspx e WebForm1.aspx.vb** - forma a única página WEB.
- **AssemblyInfo.vb** - arquivo com informações sobre o projeto com metadados sobre as assemblies
- **Web.Config** - arquivo XML com os dados das configurações sobre cada recurso usado no projeto.
- **Global.asax e Global.vb** - O arquivo **Global.asax** é opcional e server para tratar eventos a nível da aplicação . O arquivo **Global.vb** é um arquivo oculto e dependente do arquivo anterior ; contém código para evento a nível de aplicação.
- **Styles.css** - arquivo de suporte para estilos CSS (**Cascading Style Sheets**)
- **WebApplication1.vsdisco** - Arquivo XML com links para recursos que oferecem informações para Web Services.

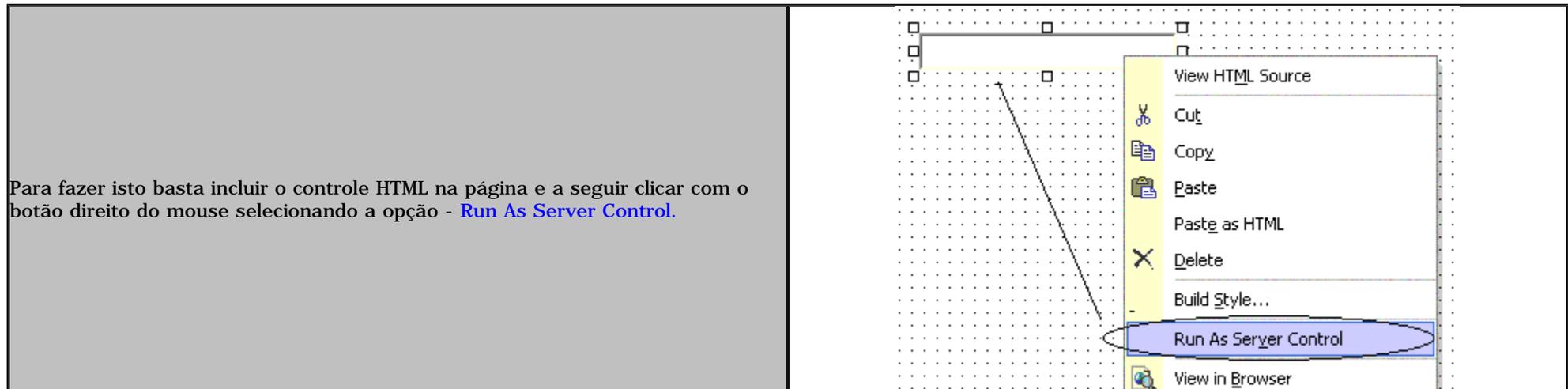
Se você mudar o modo de visualização da página inicial para HTML irá ver o seguinte código :

```
<%@ Page Language="vb" AutoEventWireup="false" Codebehind="WebForm1.aspx.vb" Inherits="Webforms1.WebForm1"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>WebForm1</title>
    <meta name="GENERATOR" content="Microsoft Visual Studio.NET 7.0">
    <meta name="CODE_LANGUAGE" content="Visual Basic 7.0">
    <meta name="vs_defaultClientScript" content="JavaScript">
    <meta name="vs_targetSchema" content="http://schemas.microsoft.com/intellisense/ie5">
  </head>
  <body MS_POSITIONING="GridLayout">
    <form id="Form1" method="post" runat="server">
      </form>
    </body>
  </html>
```

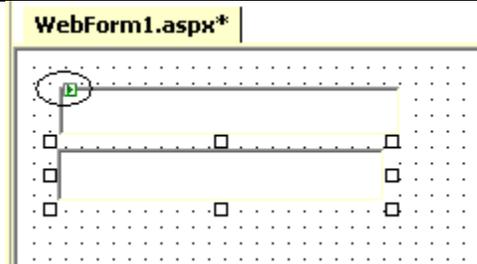
Você pode incluir qualquer tag HTML no código acima pois quando você alterar para o modo de Design , suas alterações estarão visíveis.

Como já dissemos acima para trabalhar com Web Forms podemos usar Controles HTML e Controles WebForms. Qual a diferença ???

Os controles HTML quando incluídos no formulário Web Form não estão disponíveis para o servidor e são tratados como texto comum que é passado diretamente para o Browser ; mas podemos converter os controles HTML em controles HTML de Servidor , assim eles ficarão disponíveis para o servidor.



Abaixo temos uma figura onde exibimos dois controles HTML inseridos em uma página . O primeiro foi transformado em um controle HTML de servidor e tem um ícone no canto superior esquerdo  indicando que o controle é um controle HTML de servidor.



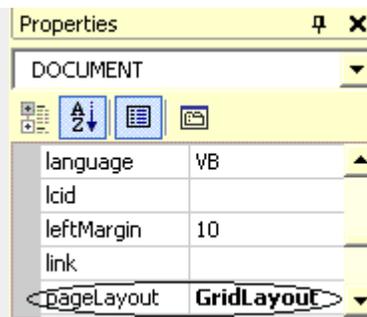
abaixo o código HTML dos dois controles . O código em vermelho representa o controle HTML convertido para o servidor. Perceba a diferença no código:

```
<INPUT style="Z-INDEX: 101; LEFT: 21px; WIDTH: 172px; POSITION: absolute; TOP: 18px; HEIGHT: 25px" type="text" size="23" id="Text1" name="Text1" runat="server">
<INPUT style="Z-INDEX: 102; LEFT: 20px; WIDTH: 165px; POSITION: absolute; TOP: 49px; HEIGHT: 27px" type="text" size="22"></form>
```

Criando o seu primeiro Web Form

Agora vamos criar realmente nosso primeiro **Web Forms** , ou seja , nossa primeira aplicação ASP.NET com Web Forms. Antes de começar você tem que saber que o Web Form funciona em dois modos : **GridLayout** (layout de grade) e **FlowLayout** (layout de fluxo). No modo **GridLayout** os controles podem ser inseridos em posições exatas da página , no modo **FlowLayout** os elementos estão posicionados relativamente uma ao outro.

Para alterar o modo de funcionamento defina a propriedade **PageLayout** do Web Form para **GridLayout**



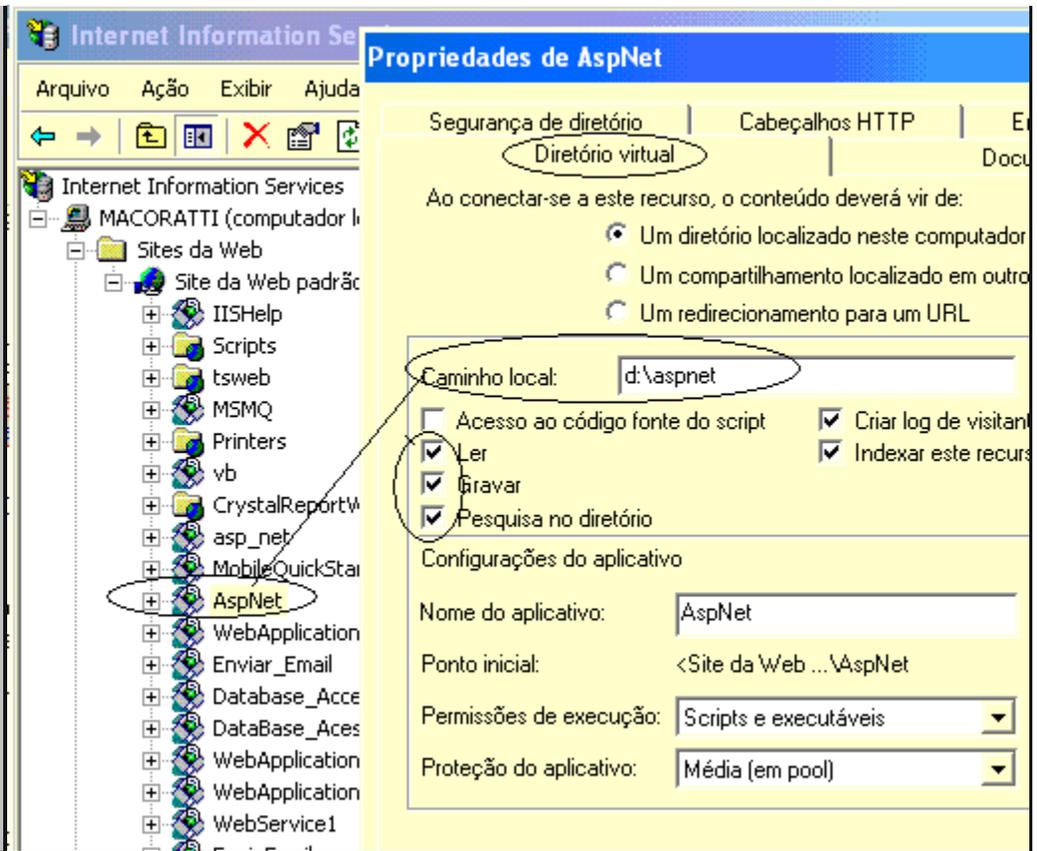
Vamos criar um Web Form com uma [página de que faz a conversão de metros para polegadas](#) onde o usuário deverá informar o seu um valor em metros que será convertido em polegadas . É um exemplo simples , mas servirá aos nossos propósitos :

Nota : Eu criei um diretório virtual com o nome de [aspnet](#) sob a pasta d:\ (d:\aspnet) vinculada a pasta web padrão.

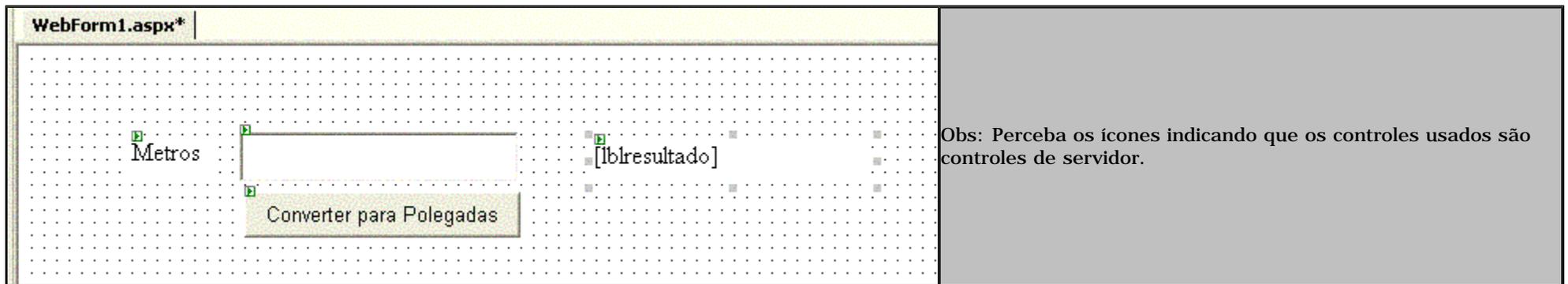
A aplicação será executada acionando a URL :
<http://localhost/aspnet/Webform1/WebForm1.aspx>

Assim se você tiver a plataforma instalada basta abrir o seu Navegador e digitar na caixa endereço :

<http://localhost/aspnet/Webform1/WebForm1.aspx>



- A primeira coisa que iremos fazer é criar o formulário . Fazemos isto usando os controles Web Forms e arrastando-os para o formulário. Usaremos um dois controles **Label** , um **TextBox** e um **Button** , conforme abaixo:



Se você der uma espiada no código irá ver o seguinte :

```

Page Language="vb" AutoEventWireup="false" CodeBehind="WebForm1.aspx.vb" Inherits="WebForm1.WebForm1">
OCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
ML>
<HEAD>
<title>WebForm1</title>
<meta name="GENERATOR" content="Microsoft Visual Studio.NET 7.0">
<meta name="CODE_LANGUAGE" content="Visual Basic 7.0">
<meta name="vs_defaultClientScript" content="JavaScript">
<meta name="vs_targetSchema" content="http://schemas.microsoft.com/intellisense/ie5">
</HEAD>
<body MS_POSITIONING="GridLayout">
<form id="Form1" method="post" runat="server">
<asp:Label id="Label1" style="Z-INDEX: 101; LEFT: 72px; POSITION: absolute; TOP: 58px" runat="server" Wi
<asp:TextBox id="TextBox1" style="Z-INDEX: 102; LEFT: 141px; POSITION: absolute; TOP: 54px" runat="serve
<asp:Button id="Button1" style="Z-INDEX: 103; LEFT: 144px; POSITION: absolute; TOP: 93px" runat="server"
<asp:Label id="lblresultado" style="Z-INDEX: 105; LEFT: 367px; POSITION: absolute; TOP: 60px" runat="ser
</form>
</body>
TML>

```

Nada muito funcional , não é mesmo ??? Vamos dar vida este código , para isto vamos acrescentar um gerenciador de evento associado ao botão **Button1**. Fazemos isto clicando duas vezes sobre o controle **Button** (exatamente como no VB). Depois é só inserir o código que desejamos que seja executado. No nosso caso iremos ter :

```

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim polegadas As Single
    polegadas = Val(txtmetros.Text) / 0.0256
    lblresultado.Text = polegadas & " polegadas "
End Sub

```

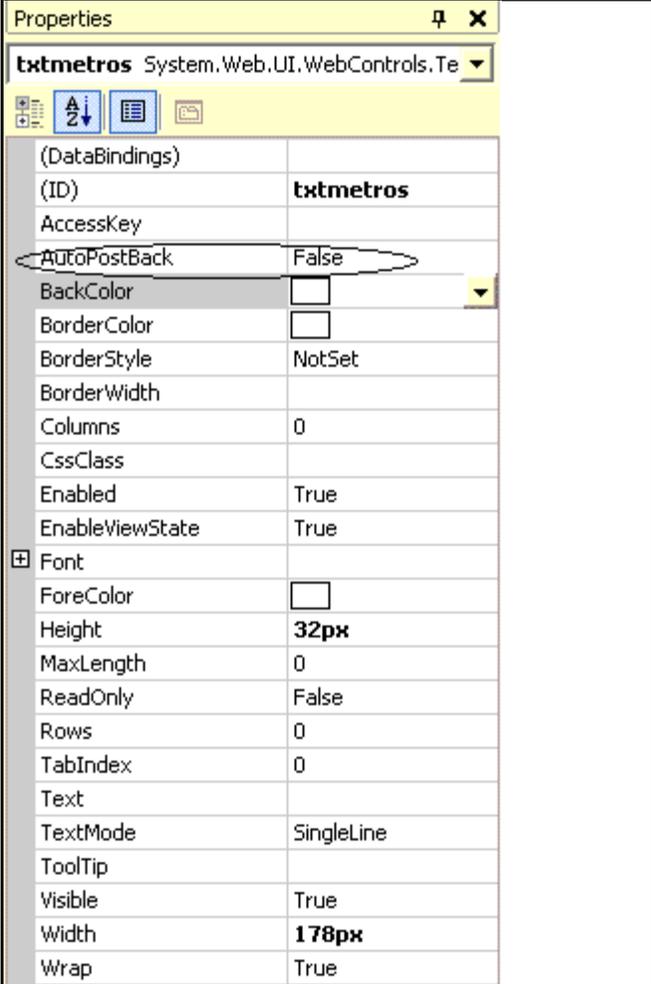
E o resultado ?? Basta executar o projeto e informar um valor e clicar no botão de comando - Converter para Polegadas - obtendo o resultado em polegadas, veja abaixo :

The screenshot shows a web browser window with the address bar containing the URL: `http://localhost/aspnet/Webform1/WebForm1.aspx`. The main content area of the browser displays a simple web form. On the left, the word "Metros" is followed by a text input field containing the number "2". To the right of this field, the text "78,125 polegadas" is displayed. Below the text input field is a button with the text "Converter para Polegadas".

Como funciona ???

Ao iniciar o seu aplicativo ASP.NET o Internet Explorer será iniciado e o formulário **WebForm** será exibido. O servidor cria então uma página HTML com um botão de comando que submeterá o código ao formulário. Ao clicar no botão a página é transmitida de volta ao servidor, alterada e depois restaurada de volta ao cliente. Toda vez que ocorrer uma interação do usuário com um controle que disparar um evento este caminho será seguido, do cliente para o servidor e de volta ao cliente. Deu-se a isto o nome de : **postback**.

Alguns controles WebForms sempre irão causar um **postback** quando acionados ; outros terão que ser configurados para que tenham o mesmo comportamento . Vamos comparar os controles Web Forms :

 <p>Properties window for txtmetros (System.Web.UI.WebControls.TextBox). The AutoPostBack property is highlighted with a red oval and set to False.</p> <table border="1"> <tr><td>(DataBindings)</td><td></td></tr> <tr><td>(ID)</td><td>txtmetros</td></tr> <tr><td>AccessKey</td><td></td></tr> <tr><td>AutoPostBack</td><td>False</td></tr> <tr><td>BackColor</td><td></td></tr> <tr><td>BorderColor</td><td></td></tr> <tr><td>BorderStyle</td><td>NotSet</td></tr> <tr><td>BorderWidth</td><td></td></tr> <tr><td>Columns</td><td>0</td></tr> <tr><td>CssClass</td><td></td></tr> <tr><td>Enabled</td><td>True</td></tr> <tr><td>EnableViewState</td><td>True</td></tr> <tr><td>Font</td><td></td></tr> <tr><td> ForeColor</td><td></td></tr> <tr><td> Height</td><td>32px</td></tr> <tr><td> MaxLength</td><td>0</td></tr> <tr><td> ReadOnly</td><td>False</td></tr> <tr><td> Rows</td><td>0</td></tr> <tr><td> TabIndex</td><td>0</td></tr> <tr><td> Text</td><td></td></tr> <tr><td> TextMode</td><td>SingleLine</td></tr> <tr><td> ToolTip</td><td></td></tr> <tr><td> Visible</td><td>True</td></tr> <tr><td> Width</td><td>178px</td></tr> <tr><td> Wrap</td><td>True</td></tr> </table>	(DataBindings)		(ID)	txtmetros	AccessKey		AutoPostBack	False	BackColor		BorderColor		BorderStyle	NotSet	BorderWidth		Columns	0	CssClass		Enabled	True	EnableViewState	True	Font		ForeColor		Height	32px	MaxLength	0	ReadOnly	False	Rows	0	TabIndex	0	Text		TextMode	SingleLine	ToolTip		Visible	True	Width	178px	Wrap	True	<p>-No lado esquerdo temos as propriedades do controle TextBox :</p> <p>- No lado direito temos as propriedades do controle Button</p> <p>O controle TextBox possui uma propriedade que o controle Button não tem ; a propriedade AutoPostBack</p> <p>O controle Button sempre causará um postback quando acionado</p> <p>Já o controle TextBox , para causar um postback quando acionado deverá ter a propriedade AutoPostBack definida como True.</p>	 <p>Properties window for Button1 (System.Web.UI.WebControls.Button). The AutoPostBack property is not visible, but the CausesValidation property is set to True.</p> <table border="1"> <tr><td>(DataBindings)</td><td></td></tr> <tr><td>(ID)</td><td>Button1</td></tr> <tr><td>AccessKey</td><td></td></tr> <tr><td>BackColor</td><td>#FFFF80</td></tr> <tr><td>BorderColor</td><td></td></tr> <tr><td>BorderStyle</td><td>NotSet</td></tr> <tr><td>BorderWidth</td><td></td></tr> <tr><td>CausesValidation</td><td>True</td></tr> <tr><td>CommandArgument</td><td></td></tr> <tr><td>CommandName</td><td></td></tr> <tr><td>CssClass</td><td></td></tr> <tr><td>Enabled</td><td>True</td></tr> <tr><td>EnableViewState</td><td>True</td></tr> <tr><td>Font</td><td></td></tr> <tr><td> ForeColor</td><td></td></tr> <tr><td> Height</td><td>29px</td></tr> <tr><td> TabIndex</td><td>0</td></tr> <tr><td> Text</td><td>Converter para Po</td></tr> <tr><td> ToolTip</td><td></td></tr> <tr><td> Visible</td><td>True</td></tr> <tr><td> Width</td><td>176px</td></tr> </table>	(DataBindings)		(ID)	Button1	AccessKey		BackColor	#FFFF80	BorderColor		BorderStyle	NotSet	BorderWidth		CausesValidation	True	CommandArgument		CommandName		CssClass		Enabled	True	EnableViewState	True	Font		ForeColor		Height	29px	TabIndex	0	Text	Converter para Po	ToolTip		Visible	True	Width	176px
(DataBindings)																																																																																														
(ID)	txtmetros																																																																																													
AccessKey																																																																																														
AutoPostBack	False																																																																																													
BackColor																																																																																														
BorderColor																																																																																														
BorderStyle	NotSet																																																																																													
BorderWidth																																																																																														
Columns	0																																																																																													
CssClass																																																																																														
Enabled	True																																																																																													
EnableViewState	True																																																																																													
Font																																																																																														
ForeColor																																																																																														
Height	32px																																																																																													
MaxLength	0																																																																																													
ReadOnly	False																																																																																													
Rows	0																																																																																													
TabIndex	0																																																																																													
Text																																																																																														
TextMode	SingleLine																																																																																													
ToolTip																																																																																														
Visible	True																																																																																													
Width	178px																																																																																													
Wrap	True																																																																																													
(DataBindings)																																																																																														
(ID)	Button1																																																																																													
AccessKey																																																																																														
BackColor	#FFFF80																																																																																													
BorderColor																																																																																														
BorderStyle	NotSet																																																																																													
BorderWidth																																																																																														
CausesValidation	True																																																																																													
CommandArgument																																																																																														
CommandName																																																																																														
CssClass																																																																																														
Enabled	True																																																																																													
EnableViewState	True																																																																																													
Font																																																																																														
ForeColor																																																																																														
Height	29px																																																																																													
TabIndex	0																																																																																													
Text	Converter para Po																																																																																													
ToolTip																																																																																														
Visible	True																																																																																													
Width	176px																																																																																													

Um **postback** sempre envolve a obtenção de uma nova página do servidor e isto pode fazer com que as coisas fiquem um pouco lentas na sua aplicação. Por isto a propriedade **AutoPostBack** dos controles esta ajustada para **False** de maneira a 'retardar' os eventos de controle até que um controle próprio (**Button**) inicie um **postback**. (Você só deve alterar esta propriedade para situações em que realmente precisar.)

Para evitar esta sobrecarga os eventos dos controles se comportam diferentemente de como agem nos formulários Windows. Assim o evento **Change** do controle **TextBox** ocorre quando um usuário move para um controle diferente na página depois de modificar a caixa de texto e não como nos formulários Windows , quando este evento ocorre sempre que qualquer tecla seja pressionada. Pela mesma razão os eventos de Mouse (**MouseMove** e **KeyPress**) não estão implementados nos controles WebForms.

Eu reconheço que este artigo não apresenta nenhum exemplo realmente interessante , mas nele abordo os conceitos básicos para que em futuros artigos você não fique 'boiando'. Para encerrar vou falar sobre um a questão :

Qual a diferença entre Visual Basic.NET e ASP.NET ?

Criando seu primeiro Web Forms II

Qual a diferença entre Visual Basic.NET e ASP.NET ? Esta foi a pergunta que deixamos no ar no artigo : [Criando seu primeiro Web Forms](#).

ASP.NET é uma linguagem que fica por trás de um Web Form. De forma geral parece-se com HTML com algumas tags especiais. E você não escreve código ASP.NET diretamente mas permite ao *Web Form Designer* criar o código para você. Funciona mais ou menos assim :

1. Você cria um Web Form usando o **Web Form Designer**
2. O Visual Basic cria uma página com código ASP.NET : um arquivo com extensão **.aspx**
3. Quando o arquivo .aspx for solicitado , o servidor WEB interpreta o código ASP.NET e substitui as tags ASP.NET com os elementos HTML necessários.
4. A versão HTML é enviada ao usuário.

Mantendo o estado da página

Um dos grandes problemas enfrentados pelas páginas ASP era manter o estado dos controles e variáveis , ou seja , cada vez que sua página ASP vai ao servidor e volta , todas as informações fornecidas nos controles de interface de usuário são perdidas , pois a página é recriada toda vez que o usuário a solicita. Este problema causou muitas dores de cabeça aos desenvolvedores ASP. 🤔

Felizmente com ASP.NET temos boas notícias : Com ASP.NET o servidor pode fazer este trabalho registrando as informações sobre a posição atual de controles de página em um campo oculto na página WEB . Este campo oculto então é transmitido de volta ao servidor em cada **postback**. Isto é feito de modo transparente e as informações são retidas para cada controle. Você pode ver as informações deste campo se examinar a página HTML em um editor de texto. Calma !!! as informações são criptografadas para evitar olhos curiosos !!! . Abaixo um exemplo :

```
<input type="hidden" name="_VIEWSTATE" value="""%$%$*&(*%+_{(*&""%} ^{HGFJUCJhgfdf=" />
```

Para que isto funcione você deve definir como **True** a propriedade **EnableViewState** do controle , e , isto vale para qualquer controle que você possa modificar com código.

Quando usamos ASP.NET estamos pressupondo que o código é executado no lado do servidor (**server-side**) , mas os eventos de interface são realizados no lado do cliente (**client-side**). Assim qualquer evento do lado do cliente dispara um evento no lado do servidor , com o evento gerado , a página é enviada , processada e devolvida. Podemos resumir as etapas principais envolvidas como :

- **Page_Load** - Primeiro método a ser executado na página servindo para realizar qualquer inicialização necessária.
- **Eventos de controle** - A seguir são executados os eventos gerados pelos controles em resposta a uma ação do usuário.
- **Page_Unload** - Último evento executado ; serve para realizar ações como : fechar arquivos e conexões , destruir objetos criados , etc.

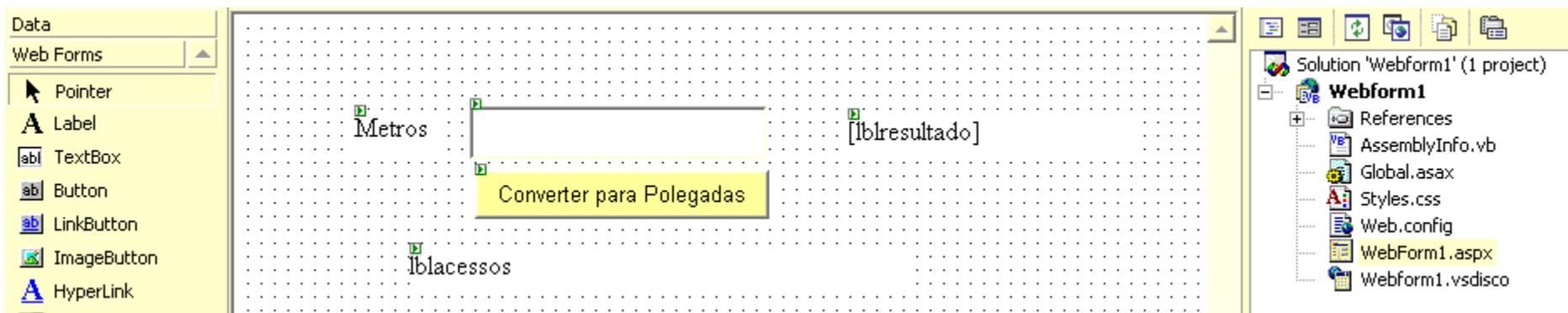
Vamos aproveitar o exemplo usado no artigo - [Criando seu primeiro Web Forms](#) - e fazer uma inclusão no código usado. Vamos supor que desejamos saber quantas vezes a rotina de conversão da página é executada. Para fazer isto eu vou incluir uma **variável contador** que irá ser acrescida no evento **Click** do botão de comando que faz a conversão , de forma que tenhamos uma maneira de contar os acessos à rotina de conversão da página. O valor será exibido em um controle **Label** que também será incluído na página. Será que vai funcionar ??? Vejamos como ficou o código :

```

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim polegadas As Single
    Dim contador As Integer
    polegadas = Val(txtmetros.Text) / 0.0256
    lblresultado.Text = polegadas & " polegadas "
    'código inserido para contar os acessos a página
    contador = contador + 1
    lblacessos.text = contador.ToString() & " acessos a rotina de conversão "
End Sub

```

O layout da nossa página Web Form ficou assim : *(lembre-se que o nome do controle é definido pela propriedade (ID))*



Se você executar a página e clicar uma , duas , três ,... vezes no botão - **Converter para Polegadas** - vai perceber que o contador não sofrerá alterações , sempre irá exibir o valor igual a 1 . Porque ? Por que a página é recriada a cada solicitação e a variável também é recriada e tudo começa do 'zero' a cada solicitação. Então como resolver o problema ???

Se você pensou em armazenar a variável na memória do servidor , considere que em um ambiente com milhares de variáveis (e isto não é difícil de ocorrer na internet) a memória do servidor iria se esgotar e o desempenho vai ser afetado. E agora ?

Para resolver este problema podemos usar o recurso , explicado acima , ele é chamado : **State Bag** ; e permite a uma página ASP.NET restaurar automaticamente o conteúdo dos controles usados na página.

Vejamos como fica o código usando o **State Bag**:

```

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim polegadas As Single
    Dim contador As Integer
    polegadas = Val(txtmetros.Text) / 0.0256
    lblresultado.Text = polegadas & " polegadas "

    contador = CType(ViewState("contador"), Integer)

    contador = contador + 1
    lblacessos.text = contador.ToString() & " acessos a rotina de conversão "

    ViewState("contador") = contador
End Sub

```

Como funciona ?

1 - contador = CType(ViewState("contador"), Integer)

Esta linha de código recupera o valor da variável **contador** armazenada no **State Bag** (ViewState("contador")).

2- ViewState("contador") = contador

Esta linha de código cria o item **contador** na **State Bag**, pois cada item na State Bag é indexado por um nome, e, se ele não existe, será criado da primeira vez.

Pronto !! ao executar a página e clicar no botão para conversão veremos exibido o número de vezes que a rotina foi acessada. Beleza ???

Mas nem tudo é um mar de rosas, embora você possa usar este recurso para o formulário **Web Form atual**, se o usuário navegar para outro formulário Web Form teremos que usar outro método para passar as informações entre as páginas (*veremos como fazer isto em outro artigo*), ou seja, o **State Bag** só funciona para o formulário atual. 😊

Além disto se você abusar deste recurso a saída HTML onde o campo oculto da State Bag é armazenado pode tornar a transmissão da página lenta.

Para terminar este artigo teórico mas fundamental para que você compreenda os conceitos básicos relacionados com ASP.NET vou falar da propriedade **Page.IsPostBack**

Quando a página viaja do cliente para o servidor e retorna do servidor para o cliente dizemos, ou melhor os gringos dizem, que ocorreu um **round-trip** (chic não é mesmo..). Então quando ocorre um **round-trip**. (*pronuncia-se : ráund-trip*) ??

Um **round-trip** ocorre toda vez que a página é solicitada ou quando ocorre um **POST** (envio) de formulário. Pois bem, para saber qual a primeira vez em que a página foi solicitada ou se ocorreu um **POST** usamos a propriedade **.IsPostBack**.

Page.IsPostBack retorna um valor booleano (**True**) se a origem for um **POST** e retorna um (**False**) se for a primeira execução da página.

Com isto em mente podemos usar o evento **Load_Page** no caso do exemplo anterior para obter o mesmo resultado. Veja como ficou o código:

```

Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
If Not Page.IsPostBack Then
    ViewState("contador") = 1
Else
    ViewState("contador") = CInt(ViewState("contador")) + 1
End If
lblacessos.Text = ViewState("contador") & " acessos à rotina de conversão"
End Sub

```

Como funciona ?

Verificamos se a pagina foi solicitada diretamente **sem** um clique de botão em : **If Not Page.IsPostBack** , e então atribuímos o valor inicial igual a 1 ao **State Bag**. Depois é só incrementar o contador , pois a cada clique o valor para **Page.IsPostBack** será sempre igual a **True**.

Executando a página e clicando algumas vezes no botão de conversão teremos:

Metros 195,3125 polegadas

Converter para Polegadas

4 acessos à rotina de conversão

Veja a execução da página em : <http://www.visualbasic.mat.br/webform/conversor.aspx>

Trabalhando com Controles e Web Forms - II - Validação

No artigo anterior - [Trabalhando com Controles Web Forms I](#) - criamos um formulário de **Login** e mostramos como usar alguns controles Web Forms. Neste artigo vamos mostrar como fazer a validação de dados usando os controles de validação de dados presentes na plataforma .NET.

Para poder testar os exemplos deste artigo você deve verificar os seguintes itens no seu computador antes de continuar:

1. Seu sistema operacional deve ser Windows 2000 ou XP .
2. .NET Framework. Não tem !!! 😞 Então pegue em : www.asp.net 😊
3. O IIS deverá estar instalado e configurado - [ASP.NET - Instalando e Configurando o Internet Information Services - IIS](#)

Nota: Uma maneira de contornar a necessidade do IIS é instalar o WebMatrix - www.asp.net - que já vem com um servidor para testes das páginas ASP.NET.

Só para lembrar os artigos sobre ASP.NET publicados até agora são :

1. [ASP.NET - evolução ou enganação ?](#)
2. [ASP.NET - Instalando e Configurando o Internet Information Services - IIS.](#)

3. [ASP.NET - Criando seu primeiro Web Forms.](#)
4. [ASP.NET - Criando seu primeiro Web Forms I](#)
5. [ASP.NET - Trabalhando com Controles e Web Forms](#)

I - Usando controles de validação de dados

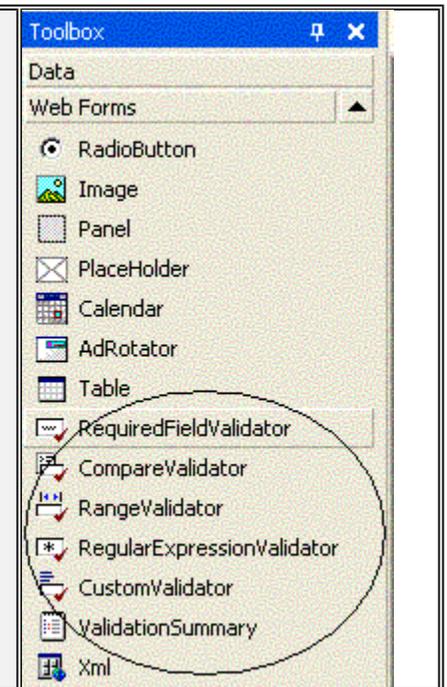
A validação de dados é um tópico essencial em muitas aplicações Web tais como formulários de cadastro , de entrada de valores e informações pessoais , só para citar alguns exemplos muito usados. A tarefa de validar dados pode ser muito trabalhosa pois envolve validar dados tanto no lado do cliente como no lado do servidor. (Se você já usou alguma linguagem de scripts para fazer isto sabe do que eu estou falando...)

O ASP.NET tem um modelo de validação de dados composto de controles que são declarados na página e fazem todo este trabalho de forma quase automática e transparente para você. (Eles chegam até a verificar se o Browser do Cliente possui [JavaScript](#) para fazer a validação do lado do cliente...).

Eu vou aproveitar o formulário de **Login** criado no artigo anterior (citado acima) para implementar a validação de dados usando os controles de validação ASP.NET. Vou acrescentar alguns campos para poder usar mais controles de validação.

Antes vou apresenta a você os controles de validação. Na aba **Web Forms** da **ToolBox** temos os seguintes controles usados para validar dados :

1. [RequiredFieldValidator](#)
2. [CompareValidator](#)
3. [RangeValidator](#)
4. [RegularExpressionValidator](#)
5. [CustomValidator](#)
6. [ValidationSummary](#)



Vamos explicar cada um deles :

- 1- **RequiredFieldValidator** - Torna o controle associado de preenchimento obrigatório e verifica se o usuário digitou ou selecionou algo.Ex: Campos de preenchimento obrigatório.
- 2- **CompareValidator** - Realiza a comparação do valor informando com um valor informado em outro controle ou com uma constante.Ex: Validação de senhas.
- 3- **RangeValidator** - Faz a validação do valor informado verificando se ele se encontra dentro de um intervalo de valores aceitos pela aplicação. Permite-se a validação de um valor máximo , mínimo ou ambos.
- 4- **RegularExpressionValidator** - Verifica se os dados entrados coincide com uma expressão regular. Ex: validações de CEP, RG , CPF , etc..
- 5- **CustomValidator** - Permite que você crie o seu próprio código de validação de dados.
- 6- **ValidationSummary** - Permite a exibição de um resumo de todas as validações feitas na página.

Você pode usar um ou mais controles de validação para validar um campo da sua página ASP.NET , assim você pode definir para um mesmo campo que o seu preenchimento é obrigatório e que o valor a ser informado deve ficar dentro de um intervalo.

A seguir mostramos quais os passos envolvidos no modelo de validação :

1. Ocorre um **Submit** da página
2. Os controles de validação entram em ação para validar os dados referentes aos controles a que estão associados.
3. Cada controle possui a propriedade **IsValid** que indica se a validação foi um sucesso ou não.
4. Após a execução de todos os controles a página , também através da propriedade **IsValid** , indica se os controles estão válidos ou não
5. Se a validação foi efetuado com sucesso a página segue o fluxo normal , caso contrário mensagens de erros são apresentadas ao usuário.

Todo este processo pode ocorrer no lado do servidor ou de forma mista , no lado do servidor e do cliente.

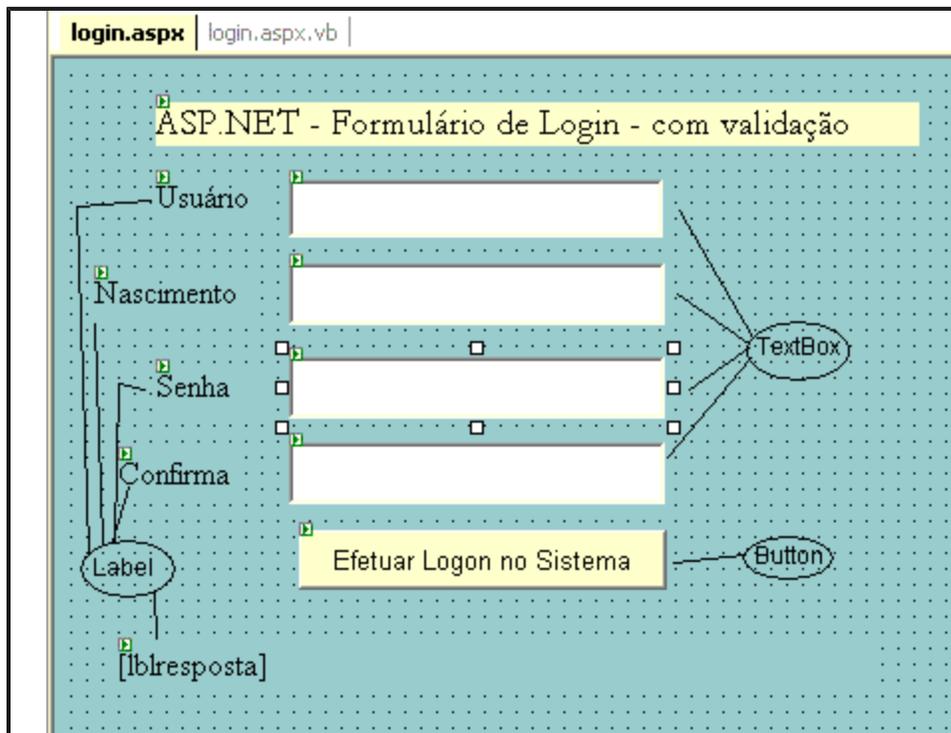
Vamos agora aplicar esta teoria toda ao nosso formulário de login criado no artigo anterior . Vou usar este formulário como base e vou acrescentar alguns campos : um campo de data de nascimento e outro campo de confirmação de senha.

Primeiro eu vou definir quais as validações que eu quero fazer :

- **Vou verificar se o nome do usuário esta sendo informado.**
- **Vou verificar se as senhas conferem.**
- **Vou verificar se a data de nascimento é válida.**

Abra o projeto [login.aspx](#) ou inicie um novo projeto no Visual Studio.NET com as seguintes características (sinta-se a vontade para alterar a seu gosto.)

1. Project Types : Visual Basic Projects
 2. Templates : **ASP.NET Web Applications**
 3. Name : Login
 4. Location : <http://localhost/aspnet/login.aspx>
- Altere o nome do formulário para [Login.aspx](#)
 - Usando a caixa de ferramentas inclua no formulário os seguintes controles : **Label1 , Label2 , Label3 , Label4 ,Label5, TextBox1 , TextBox2 , TextBox3 , TextBox4 e Button1**
 - Configure os controles e o formulário para ter o seguinte layout



- Você deverá alterar as seguintes propriedades dos controles :

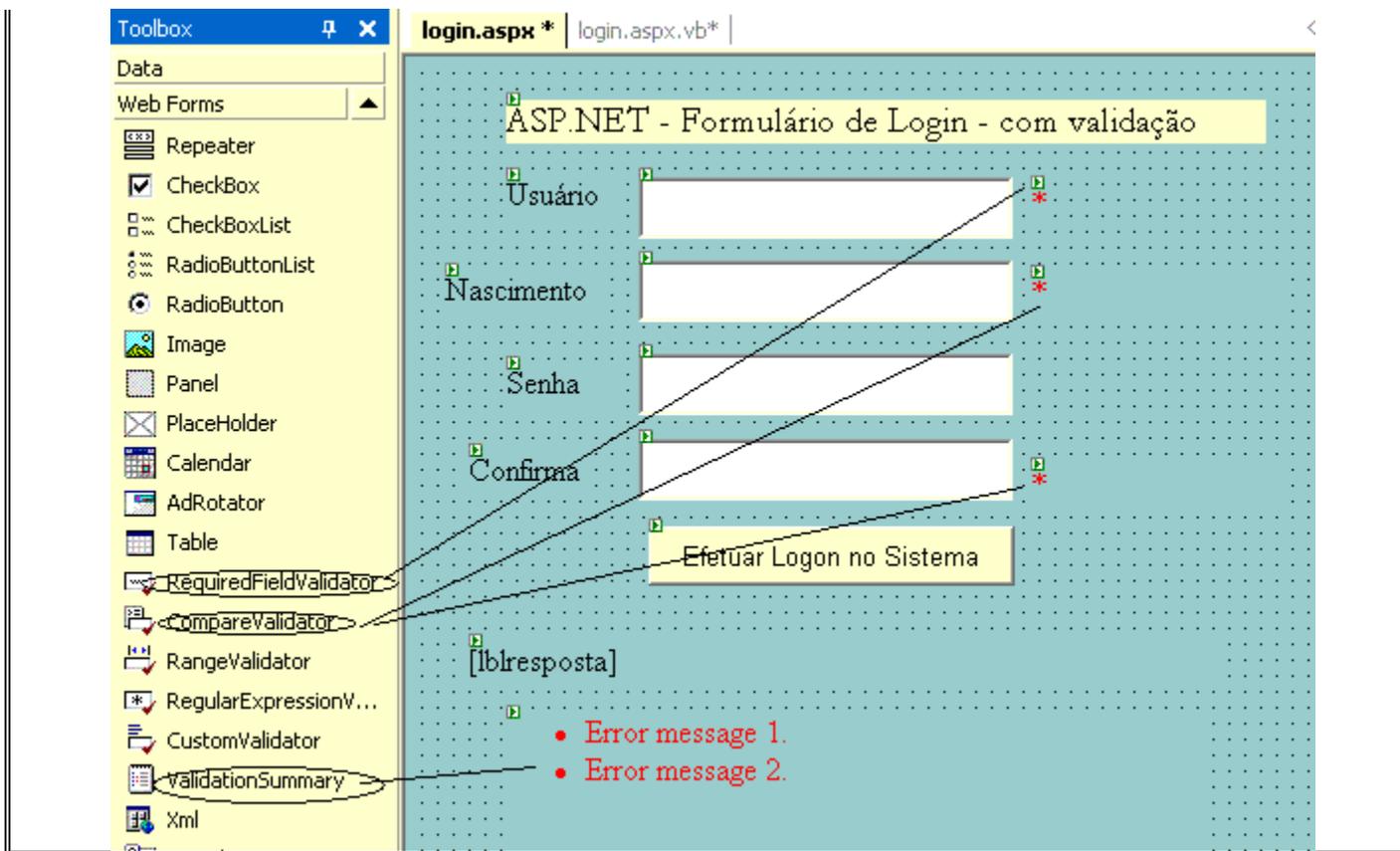
- Label1 - ID = Label1 - Text = ASP.NET - Formulário de Login - com validação Backcolor= #FFFFC0
- Label2 - ID = Label2 - Text = Usuário
- Label3 - ID = Label3 - Text = Nascimento
- Label4 - ID = Label4 - Text = Senha - TextMode = Password
- Label5 - ID = Label5 - Text = Confirma - TextMode = Password
- TextBox1 - ID= TxtUsuario
- TextBox2 - ID= TxtDataNascimento
- TextBox3 - ID = TxtSenha1
- TextBox4 - ID = TxtSenha2
- Button - ID=cmdLogon
- Login.aspx - bgcolor= #add8e6

Para ajustar os controles você pode usar as opções : **Align , Make Same Size , Horizontal spacing , Vertical spacing** do menu **Format**.

Agora vou mostrar como usar os controles de validação em um formulário :

1- Como eu decidi que os campos devem ser de preenchimento obrigatório eu vou ter que usar o controle de validação - [RequiredFieldValidator](#). Para isto arraste o controle da aba [WebForms](#) para próximo de cada campo que eu desejo validar. Veja abaixo:

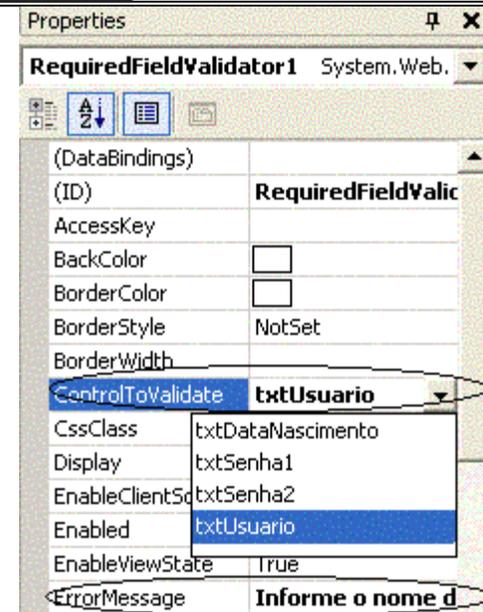




Clique com botão direito do mouse sobre o primeiro controle de validação e na janela **Properties** defina as propriedades como abaixo : (veja figura ao lado)

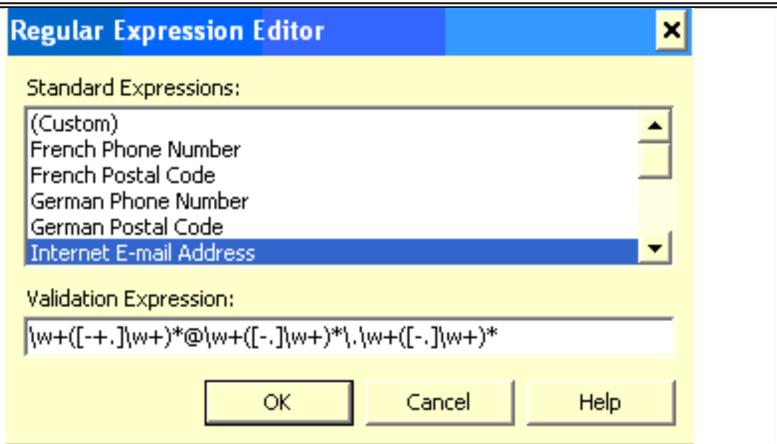
- **ControlToValidate** - `txtUsuario` - o nome do controle a ser validado
- **Text** - * (asterisco) - será exibido ao lado do campo
- **ErrorMessage** - `Informe o nome do usuário` - mensagem exibida no sumário ao pé da página

Repita o processo para cada controle alterando cada uma das propriedades acima conforme o campo a ser validado.

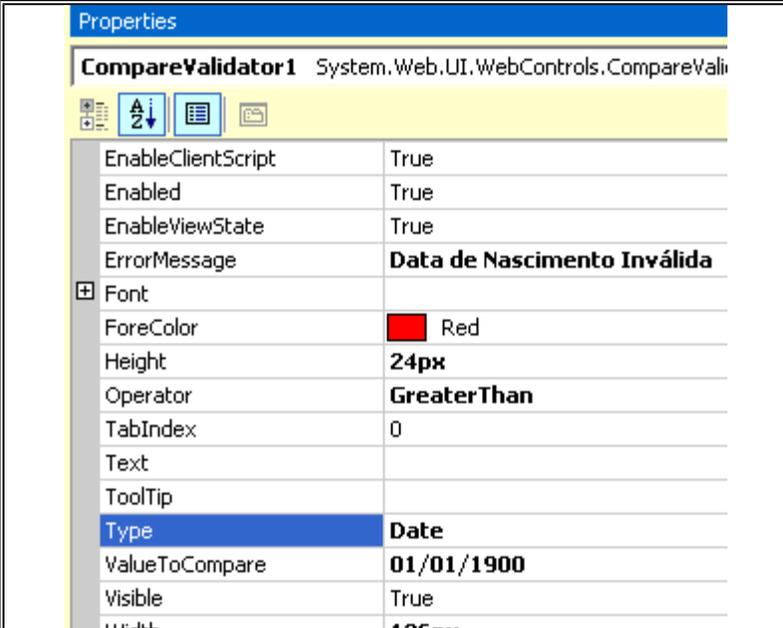


Como eu quero que no nome do usuário seja informado um e-mail válido vou ter que usar outro controle de validação para o campo nome. Vou usar o controle de validação - **RegularExpressionValidator** ; vejamos como ele funciona:

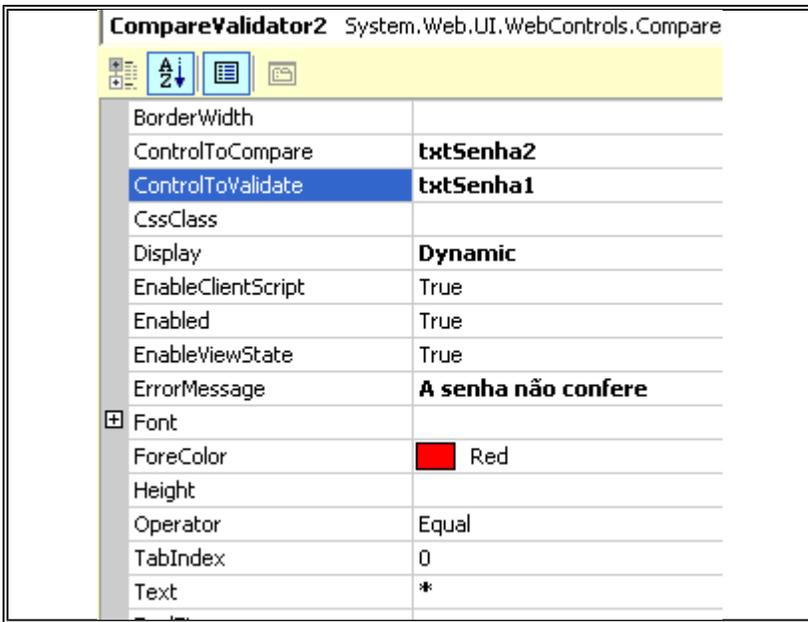
- Arraste o controle de validação da aba **WebForms** para próximo de cada campo **txtUsuario** e na janela **Properties** defina as propriedades como abaixo:

<ol style="list-style-type: none"> 1. ControlToValidate - TxtUsuario 2. Text - * 3. ErrorMessage - Informe um e-mail válido 4. ValidationExpression - Clique no botão com ... e selecione na caixa - Regular Expression Editor - a opção : Internet E-mail Address 5. Display - Dynamic - 	
---	---

- Para validar a **data de nascimento** arraste o controle de validação da aba **WebForms** para próximo de cada campo **txtDataNascimento** e na janela **Properties** defina as propriedades como abaixo:

	<ol style="list-style-type: none"> 1. ControlToValidate - txtDataNascimento 2. Text = * 3. ErrorMessage = Data de Nascimento Inválida 4. ValueToCompare = 01/01/1900 5. Operator = GreaterThan 6. Type = Date 7. Display = Dynamic
---	--

- Para validar a **senha** arraste o controle de validação da aba **WebForms** para próximo de cada campo **txtSenha1** e na janela **Properties** defina as propriedades como abaixo:



Property	Value
BorderWidth	
ControlToCompare	txtSenha2
ControlToValidate	txtSenha1
CssClass	
Display	Dynamic
EnableClientScript	True
Enabled	True
EnableViewState	True
ErrorMessage	A senha não confere
Font	
ForeColor	Red
Height	
Operator	Equal
TabIndex	0
Text	*

1. ControlToValidate = txtSenha1
2. ControlToCompare = txtSenha2
3. Text = *
4. ErrorMessage = **Senha não confere**
5. Operator = **Equal**
6. Display = Dynamic

- Agora só falta incluir o controle para exibir um sumário de erros ; O controle [ValidationSummary](#) deve ser inserido na parte inferior do formulário. Podemos exibir os erros de uma forma diferente (*List* , *BulletList* , *SingleParagraph*) alterando a propriedade **DisplayMode** deste controle , além de alterar a propriedade **Font e Forecolor**.

Agora já temos todos os recursos para validar o nosso formulário. Executando o projeto e fazendo uma simulação para que as mensagens de erro seja exibidas teremos o resultado como na figura abaixo:

The screenshot shows a web browser window with the address bar containing 'http://localhost/aspnet/login/login.aspx'. The page title is 'ASP.NET - Formulário de Login - com validação'. The form has four input fields: 'Usuário' (empty), 'Nascimento' (01/01/1200), 'Senha' (masked with dots), and 'Confirma' (masked with dots). A 'Efetuar Logon no Sistema' button is at the bottom. Below the form, three red error messages are listed: 'Nome do usuário deve ser informado', 'Data de Nascimento Inválida', and 'A senha não confere'.

Agora você já sabe como podemos validar os dados de um formulário usando ASP.NET.

Acessando e exibindo dados com ADO.NET

Vamos neste artigo falar um pouco sobre o acesso a banco de dados usando ASP.NET com ADO.NET. Se você sabe como acessar uma base de dados usando o VB.NET vou lhe dar uma boa notícia : " para criar uma página ASP.NET que acesse um banco de dados você não vai ter que fazer alteração alguma , é praticamente igual ao código usado no VB.NET". Na verdade podemos fazer este serviço usando a linguagem C# também (vou mostrar isto em outro artigo) ; e , se você ainda não percebeu o código VB.NET é integralmente reutilizado em uma página ASP.NET.

Se você quer lembrar como acessar um banco de dados usando o VB.NET pode ler os seguintes artigos: (veja o link [VB.NET](#) para mais artigos)

- [ADO .NET - O Acesso aos dados](#)
- [Meu Primeiro Acesso a Dados com o VB.NET](#)

Vou acessar uma base de dados Access usando ADO.NET com código VB.NET e exibir os dados de uma das tabelas em uma página ASP.NET.

Não vou me ater a teoria relacionada ao ADO.NET pois já fiz isto em artigos anteriores , vou apenas dar uma repassada nos conceitos básicos envolvidos na operação de acesso a base de dados. (Vou acessar a tabela **Authors** do banco de dados **Biblio.mdb** que esta na pasta [c:\teste](#))

A primeira coisa a ter em mente é que como vamos criar uma página ASP.NET , e o arquivo com o código terá a extensão **.aspx**. Dei o nome **acesso.aspx** ao arquivo e usei o editor Bloco de notas para digitar todo o código VB.NET.

1- Temos que usar os namespaces [System.Data](#) e [System.Data.OleDb](#) pois vou acessar uma base de dados Access. (As tags <% %> indicam que o código é código de script) ; informamos a linguagem usada no código abaixo e indicamos que o código será rodado no servidor:

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.OleDb" %>
<script language="VB" runat="server">
....
```

2 - A primeira coisa a fazer para acessar uma base de dados usando ADO.NET é criar uma conexão. Veja abaixo o código que cria uma conexão com a base de dados **Biblio.mdb**. (O código será colocado no evento **Load_Page** que é o primeiro evento que ocorre quando a página for carregada):

```
sub Page_Load(sender as Object, e as EventArgs)

Dim connString as String

connString = "PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA SOURCE=C:\teste\Biblio.mdb;"

Dim objConnection as OleDbConnection
objConnection = New OleDbConnection(connString)
objConnection.Open() 'abre a conexao
.....
```

3- Após abrir a conexão usando o provedor OLEDB.NET precisamos criar uma consulta para selecionar os dados que desejamos da tabela **Authors**. Vou Selecionar todos os registros cujo campo **Au_ID** seja menor ou igual a 13 . (<=)

A consulta SQL ficará então assim : **"SELECT * FROM Authors Where Au_ID <= 13"** e o código da consulta será :

```
'defina a consulta SQL
Dim strQuery as String = "SELECT * FROM Authors Where Au_ID <= 13"

' Cria um objeto Command
Dim objCommand as OleDbCommand
objCommand = New OleDbCommand(strQuery, objConnection)

.....
```

5- Vamos usar o objeto **DataReader** para ler os dados do banco de dados , para isto temos que criar e configura este objeto. Veja abaixo:

```
' define um OleDbDataReader
Dim objDataReader as OleDbDataReader
objDataReader = objCommand.ExecuteReader(CommandBehavior.CloseConnection)

.....
```

6- Basta agora exibir os dados em um controle **DataGrid** . Atribuímos ao DataGrid (DataSource) o objeto DataReader criado e fazemos a vinculação dos controles. (DataBind()):

```
'exibe os dados em um datagrid
dgDB.DataSource = objDataReader
dgDB.DataBind()
....
```

Obs: poderíamos também percorrer o DataReader usando um código como o abaixo:

```
While objDataReader.Read()
    objDataReader("Coluna")
End While
```

Para encerrar o arquivo temos que usar o seguinte código : Nele fechamos a conexão o DataReader e o script.

```
'fecha a conexao e o datareader
objDataReader.Close()

end sub
</script>
<asp:DataGrid id="dgDB" runat="server" />
```

O código completo do arquivo [access.aspx](#) é o seguinte: Você pode copiá-lo usando o Bloco de notas , wordpad ou Visual Studio.NET.

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.OleDb" %>
<script language="VB" runat="server">

sub Page_Load(sender as Object, e as EventArgs)

Dim connString as String

connString = "PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA SOURCE=C:\teste\Biblio.mdb;"

Dim objConnection as OleDbConnection
objConnection = New OleDbConnection(connString)
objConnection.Open()      'abre a conexao

'defina a consulta SQL
Dim strQuery as String = "SELECT * FROM Authors Where Au_ID <= 13"

' Cria um objeto Command
Dim objCommand as OleDbCommand
objCommand = New OleDbCommand(strQuery, objConnection)

' define um OleDbDataReader
Dim objDataReader as OleDbDataReader
objDataReader = objCommand.ExecuteReader(CommandBehavior.CloseConnection)
```

```
'exibe os dados em um datagrid
dgDB.DataSource = objDataReader
dgDB.DataBind()

'fecha a conexao e o datareader
objDataReader.Close()

end sub
</script>
<asp:DataGrid id="dgDB" runat="server" />
```

Nota : Perceba o código por trás da página definindo o DataGrid : `<asp:DataGrid id="dgDB" runat="server" />`

A seguir copie o arquivo **acesso.aspx** para o seu diretório virtual e execute a página acesso.aspx. O resultado será :

Endereço  <http://localhost/acesso.aspx>

Au_ID	Author	Year Born
1	Jacobs, Russell	1957
2	Metzger, Philip W.	
3	Boddie, John	
4	Sydow, Dan Parks	1957
6	Lloyd, John	1957
8	Thiel, James R.	
10	Ingham, Kenneth	1957
12	Wellin, Paul	
13	Kamin, Sam	

Simple !! Não é mesmo ??? . Em breve novos artigos sobre ASP.NET , aguarde... 😊

Operações básicas com Banco de dados

Neste artigo vamos aprender como usar as classes ADO.NET para realizar as tarefas básicas como : acessar , obter dados , incluir dados , excluir dados e atualizar dados em páginas ASP.NET (o código usado é VB.NET). Neste artigo vou me ater aos banco de dados Access e SQL Server por questão de tempo e para não tornar o artigo muito extenso. Vamos lá...

Sempre que formos trabalhar com banco de dados em páginas ASP.NET deveremos importar os seguintes [namespaces](#):

Banco de dados Access

Banco de Dados SQL Server

```
System.Data
System.Data.OleDb
```

```
System.Data
System.Data.SqlClient
```

Nota: Você pode usar o namespace `System.Data.OleDb` com o Microsoft SQL Server e também se for acessar o Oracle. (è claro que não terá acesso as funcionalidades do namespace `System.Data.SqlClient`).

Criando um conexão com o banco de dados

Você já deve estar careca de saber que a primeira coisa que você precisa fazer para acessar um banco de dados é criar uma conexão com o banco de dados.

SQL Server	Access
<pre><%@ Import Namespace="System.Data" %> <%@ Import NameSpace="System.Data.SqlClient" %> <% Dim conexao As SqlConnection conexao = New SqlConnection("server=localhost; database=Pubs; uid=sa") conexao.Open() %></pre>	<pre><%@ Import Namespace="System.Data" %> <%@ Import NameSpace="System.Data.OleDb" %> <% Dim conexao As OleDbConnection conexao = New OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0; DataSource=c:\teste.mdb") conexao.Open() %></pre>

Como funciona ?

1. A classe `conexao` é inicializada passando a string de conexão como parâmetro ao construtor da classe `XXXConnection`
2. A conexão é então aberta via método `Open` da classe `XXXConnection`

Nota: Você não precisa definir um parâmetro **Provider** para a conexão usando as classes `SqlConnection`. Estas classes trabalham diretamente com o protocolo **TDS** (*Tabular Data Stream*). Você também não pode usar um Data Source Name (DSN) quando abrir uma conexão com a classe `SqlConnection`, para isto, use o namespace `System.Data.OleDb`

Na conexão com o Access temos que informar o nome do provedor `OLEDB - Microsoft.Jet.OLEDB.4.0` - e o caminho do banco de dados no servidor. Para conectar com outro banco de dados deverá usar um provedor específico. Assim para conectar com o Oracle existe o provedor `MSDAORA`.

Para usar um **DSN** com a classe `OleDbConnection` podemos usar o seguinte código :

```
Conexao = New OleDbConnection( "DSN=Nome_DSN" )
```

Neste caso você é forçado a usar o `OLEDB` para o provedor `ODBC` e não o provedor `OLEDB` nativo do banco de dados. Isto pode afetar o desempenho de acesso aos dados.

Ao invocar o método `Open`, a conexão tem 15 segundos para ser efetivada antes de expirar por time out. Para alterar este valor padrão você pode definir outro valor, o código abaixo aumenta o valor para 90 segundos:

```
Conexao.ConnectionTimeout = 90
```

Incluindo dados com ADO.NET

Agora que já sabemos como criar uma conexão com um banco de dados , vamos usar esta conexão para incluir dados em uma tabela do banco de dados. Vamos usar o comando SQL - **Insert** - cuja sintaxe é a seguinte :

INSERT NomeDaTabela (coluna1, coluna2...) VALUES (valor1, valor2...)

Exemplo :

INSERT Teste (Nome , Endereco, Telefone) VALUES ('Macoratti', 'Rua Teste 100', '11-1234-7852)

As etapas necessárias para que o comando SQL **INSERT** seja executado em uma página ASP.NET são:

1. Criar e abrir um conexão com um banco de dados
2. Criar um comando que representa a cláusula SQL -**Insert** - que deverá ser executada.
3. Executar o comando.

SQL Server

```
< %@ Import Namespace="System.Data" %>
< %@ Import Namespace="System.Data.SqlClient" %>

< %
Dim conexao As SqlConnection
Dim comando As SqlCommand

conexao = New SqlConnection("server=localhost;uid=sa;pwd=senha;database=TesteDB")
conexao.Open()
comando = New SqlCommand( "Insert Teste ( coluna ) Values ( 'Teste' )", conexao )
comando.ExecuteNonQuery()
conexao.Close()
%>
```

-A classe **SqlCommand** é inicializada com dois parâmetros : o comando que vai executar e a conexão.

-A cláusula **Insert** vai incluir um novo registro na tabela Teste do banco de dados **TesteDB**

- O comando é executado invocando o método **ExecuteNonQuery** da classe SqlCommand. Este método é usado para enviar para executar um comando SQL que não retorna registros.

Access

```

<%@ Import Namespace="System.Data" %>
<%@ Import NameSpace="System.Data.OleDb" %>

<%
Dim conexao As OleDbConnection
Dim myCommand As OleDbCommand

conexao = New OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;DataSource=c:\Teste.mdb" )
conexao.Open()
comando = New OleDbCommand("Insert INTO Teste ( Nome ) Values ( 'Macoratti' )", conexao )
comando.ExecuteNonQuery()
conexao.Close()
%>

```

Nota: Se a string que você estiver incluindo na tabela contiver uma apóstrofe interna (') você deve tomar cuidado. Se você usar a sintaxe padrão :

```
INSERT INTO Teste ( Nome ) Values ( 'Macdonald's' )
```

Vai obter um erro por que o apóstrofe (') interna de Macdonald's é interpretada como final da string. Para contornar este problema devemos usar dois sinais de apóstrofes. Assim :

```
INSERT INTO Teste ( Nome ) Values ( 'Macdonald''s' )
```

Atualizando dados com ADO.NET

Para atualizar os registros existente em uma tabela de um banco de dados usamos o comando SQL **UPDATE**. A sintaxe é a seguinte :

```
UPDATE tabela SET coluna1 = valor1, coluna2 = valor2...
WHERE criterio
```

Exemplo :

```
UPDATE Teste SET Nome = 'Macoratti'
WHERE Nome = 'Jose'
```

Para executar um comando **UPDATE** em uma página ASP.NET temos os seguintes passos:

1. Criar e abrir uma conexão com o banco de dados
2. Criar um comando que representa a cláusula SQL -**Update** - que deverá ser executada.
3. Executar o comando.

SQL Server

```
<%@ Import Namespace="System.Data" %>
<%@ Import NameSpace="System.Data.SqlClient" %>

<%
Dim conexao As SqlConnection
Dim comando As SqlCommand

conexao = New SqlConnection( "server=localhost;uid=sa;pwd=senha;database=TesteDB" )
conexao.Open()
comando = New SqlCommand( "UPDATE Teste SET Nome='Macoratti' WHERE Nome='Jose'", conexao )
comando.ExecuteNonQuery()
conexao.Close()
%>
```

Access

```
<%@ Import Namespace="System.Data" %>
<%@ Import NameSpace="System.Data.OleDb" %>

<%
Dim conexao As OleDbConnection
Dim comando As OleDbCommand

conexao = New OleDbConnection( "PROVIDER=Microsoft.Jet.OLEDB.4.0;DataSource=c:\teste.mdb" )
conexao.Open()
comando = New OleDbCommand( "UPDATE Teste SET Nome='Macoratti' WHERE Nome = 'Jose'", conexao )
comando.ExecuteNonQuery()
conexao.Close
%>
```

- O comando **UPDATE** pode afetar mais de um registro. Quando você executa um comando **Update** ele altera cada registro que satisfaz a cláusula **Where**.

- Podemos determinar o número de registros afetados por um comando **Update** em uma página ASP.NET capturando o valor retornado pelo método `ExecuteNonQuery()`. Basta incluir o código abaixo :

```
registrosAfetados = comando.ExecuteNonQuery()
Response.Write( "O comando UPDATE modificou " & registrosAfetados.ToString() & " registros!" )
```

Excuindo dados com ADO.NET

Para excluir dados de uma tabela de um banco de dados usamos o comando DELETE cuja sintaxe básica é :

DELETE tabela WHERE criterio

Exemplo:

```
DELETE Teste WHERE Nome = 'Macoratti'
```

SQL Server

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>

<%
Dim conexao As SqlConnection
Dim comando As SqlCommand

conexao = New SqlConnection( "server=localhost;uid=sa; pwd=senha;database=TesteDB" )
conexao.Open()
comando = New SqlCommand( "DELETE Teste WHERE Nome='Macoratti'", conexao )
comando.ExecuteNonQuery()
conexao.Close()
%>
```

Access

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.OleDb" %>

<%
Dim conexao As OleDbConnection
Dim comando As OleDbCommand

conexao = New OleDbConnection( "PROVIDER=Microsoft.Jet.OLEDB.4.0;DataSource=c:\testeDB.mdb" )
conexao.Open()
comando = New OleDbCommand("DELETE FROM Teste WHERE Nome='Macoratti'", conexao )
comando.ExecuteNonQuery()
conexao.Close()
%>
```

- No Access usamos **DELETE FROM** ao invés de DELETE usado no SQL Server.
- O comando **DELETE** excluir todos os registros que obedecerem a condição definida na cláusula Where.
- Podemos determinar o número de registros afetados por um comando **Delete** em uma página ASP.NET capturando o valor retornado pelo método `ExecuteNonQuery()`. Basta incluir o código abaixo :

```
registrosAfetados = comando.ExecuteNonQuery()
Response.Write( "O comando DELETE excluiu " & registrosAfetados.ToString() & " registros !" )
```

Usando DropDownList em páginas ASP.NET

O controle `DropDownList` é bastante versátil e de grande utilidade em páginas na Web. Saber como usar este controle é muito importante.

Preencher um controle `dropdownlist` com as informações de uma base de dados Access ou SQL Server é muito simples no ASP.NET. Neste artigo eu vou mostrar como você pode fazer isto com base de dados Access e SQL Server.

No artigo - [Preenchendo um controle DropDowlist](#) - dei uma introdução básica sobre como preencher um controle `DropDownList` ; neste artigo estarei

ampliando os conceitos e mostrando como obter os seguintes resultados:

1. [Preencher um controle DropDownList](#)
2. [Preencher um DropDownList e incluir um valor padrão](#)
3. [Preencher um DropDownList : exibindo um valor texto e retorna um índice na seleção de um item](#)
4. [Sincronizando dois controles DropDownList](#)

Para este artigo eu vou usar a base de dados no Access/SQL Server chamada **Nortwind.mdb** (**fiz uma cópia alterando o seu nome para Nwind2002.mdb**) e neste banco de dados a tabela **Produtos** e a tabela **Detalhes do Pedido** ; cada uma terá a seguinte estrutura :

Produtos : Tabela			
	Nome do campo	Tipo de dados	
	CódigoDoProduto	AutoNumeraçãc	Número atribuído automaticamente a um novo produto.
	NomeDoProduto	Texto	
	CódigoDoFornecedor	Número	Mesma entrada que na tabela Fornecedores.
	CódigoDaCategoria	Número	Mesma entrada que na tabela Categorias.
	QuantidadePorUnidade	Texto	(por exemplo, caixa com 24 unidades, garrafa de 1 litro)
	PreçoUnitário	Moeda	
	UnidadesEmEstoque	Número	
	UnidadesPedidas	Número	
	NívelDeEstoque	Número	Mínimo de unidades a manter em estoque.
	Descontinuado	Sim/Não	Sim significa que o item não está mais disponível.

Detalhes do Pedido : Tabela			
	Nome do campo	Tipo de dados	
	NúmeroDoPedido	Número	○
	CódigoDoProduto	Número	○
	PreçoUnitário	Moeda	
	Quantidade	Número	
	Desconto	Número	

Use o Visual Studio.NET ou o bloco de notas (a escolha é sua ... 😊) e crie um arquivo com a extensão **.aspx** (**dica** : para salvar o arquivo com esta extensão no bloco de notas clique em **Salvar Como** e informe o nome do arquivo completo com extensão entre **aspas**). Ex: **ddl.aspx**.

Para recordar abaixo relacionei novamente as **propriedades** de um controle **DropDownList** :

- **AutoPostBack** - Se for True causa um envio (**post**) do formulário quando o cliente altera o item selecionado.
- **DataSource** - Referencia a fonte de dados que o controle usa para preencher os itens.
- **DataTextField** - É usado para preencher o campo **Text** dos itens.
- **DataValueField** - Usado para preencher o campo **Value** dos itens.
- **Items** - Coleção de objetos **ListItem** onde cada objeto representa um item.
- **SelectedItem** - Uma referência o item selecionado.
- **SelectedIndex** - Informa o índice do item selecionado. O primeiro tem índice igual a zero.

Evento de um **DropDownList** : **OnSelectedIndexChanged** - Iniciado quando o controle tem a propriedade **AutoPostBack** igual a **True** e ocorre mudança no item selecionado.

Agora vamos ao trabalho... 😊

1- Exibindo os dados de uma tabela Access em um controle DropDownList

a- Exibindo os dados da tabela **Produtos** do banco de dados **Nwind2002.mdb** (Access) em um controle **DropDownList**.

Estou usando um **DataSet** para obter os dados da tabela **Produtos** e estou definindo o preenchimento do controle no código. Demos nome **dp1.aspx** ao arquivo abaixo , e, ele foi copiado para a pasta : **x:\inetpub\wwwroot\dp**

```

<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.OleDb" %>

<html>
<head>
<title>Dropdown List</TITLE>

<script language="VB" runat="server">
Sub Page_Load(Source as Object, E as EventArgs)

if not Page.IsPostBack then
  'declara as variáveis usadas no código
  Dim strSQL As string
  Dim strConn As String
  Dim Conn As OleDbConnection
  Dim da As OleDbDataAdapter
  Dim ds As DataSet

  'define a string com o comando SQL e a string de conexão usando um provedor OLEDB
  strSQL="Select * from Produtos"
  strConn = "PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA SOURCE=c:\teste\Nwind2002.mdb"

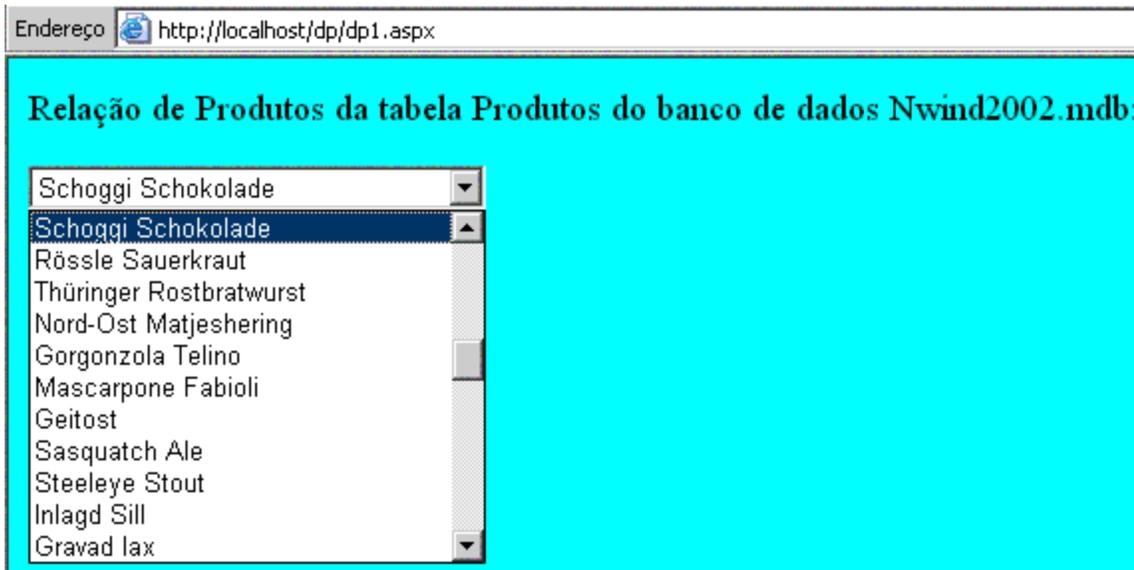
  'Abre a conexão com a fonte de dados
  Conn = New OleDbConnection(strConn)
  Conn.Open()

  try
    'cria o objeto DataAdapter
    da = New OleDbDataAdapter(strSQL, Conn)
    'Cria e preenche o DataSet
    ds = New DataSet()
    da.Fill(ds,"Produtos")
    'Define o preenchimento do controle DropDownList
    ddl1.DataTextField = "NomeDoProduto"
    ddl1.DataValueField = "NomeDoProduto"
    ddl1.DataSource = ds.Tables("Produtos").DefaultView
    ddl1.DataBind()
  Finally
    Conn.Close()
  end try
end if

End Sub
</script>
</head>
<body bgcolor="aqua">
<font verdana size="4">Relação de Produtos da tabela Produtos do banco de dados Nwind2002.mdb:</font>
<form id="Form1" runat="server">
<asp:DropDownlist id="ddl1" AutoPostBack="True" runat="server"/>
</form>
</body>
</html>

```

O Resultado da execução do arquivo [dp1.aspx](#) é o seguinte:



2- Inserindo um valor padrão em um controle DropDownList

Aproveitando o código acima vou mostrar como você pode incluir um valor padrão que será exibido quando o controle for exibido. Vou inserir o texto : "Selecione um produto"

Para obter o resultado basta incluir o código :

```
ddl1.items.Insert(0,"Selecione um produto")  
ddl1.SelectedIndex = 0
```

conforme abaixo após :

<pre>... try 'cria o objeto DataAdapter da = New OleDbDataAdapter(strSQL, Conn) 'Cria e preenche o DataSet ds = New DataSet() da.Fill(ds,"Produtos") 'Define o preenchimento do controle DropDownList ddl1.DataTextField = "NomeDoProduto" ddl1.DataValueField = "NomeDoProduto" ddl1.DataSource = ds.Tables("Produtos").DefaultView ddl1.DataBind() ddl1.items.Insert(0,"Selecione um produto") ddl1.SelectedIndex = 0</pre>	
---	--

```

Finally
Conn.Close()
end try
...

```

3- Preencher um DropDownList exibindo um valor texto e retornando o índice na seleção de um item

Vamos neste item preencher o controle acessando os dados da tabela **Produtos** do banco de **Teste** no **SQL Server**. (Esta tabela esta originalmente no banco de dados Northwind eu fiz os ajustes importando as tabelas para um banco de dados Teste)

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Databases' folder is expanded to show the 'Teste' database. The 'Tables' folder under 'Teste' is also expanded, showing a list of tables including 'Produtos'. The 'Produtos' table is selected, and its properties are shown in the 'General' tab on the right. The table structure is as follows:

Key	ID	Name	Data Type	Size(...)	Nulls	Default
		CódigoDoProduto	int	4	<input type="checkbox"/>	
		NomeDoProduto	nvarchar	40	<input type="checkbox"/>	
		CódigoDoFornec...	int	4	<input checked="" type="checkbox"/>	
		CódigoDaCategoria	int	4	<input checked="" type="checkbox"/>	
		QuantidadePorU...	nvarchar	25	<input checked="" type="checkbox"/>	
		PreçoUnitário	money	8	<input checked="" type="checkbox"/>	
		UnidadesEmEst...	smallint	2	<input checked="" type="checkbox"/>	

Vamos exibir os valores do campo **NomeDoProduto** no controle , e , quando houver uma seleção pelo usuário , o campo **CódigoDoProduto** será retornado. Perceba que a declaração **Import** no código fazem referência ao Namespace - **System.Data.SqlClient** - pois o acesso é em uma base de dados SQL Server ; o acesso é feito usando um objeto **DataReader**

```

<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>

<html>
<head>
<title>Preenchendo um Dropdown</TITLE>

<script language="VB" runat="server">
Sub Page_Load(Source as Object, E as EventArgs)

if not Page.IsPostBack then

'define a string com o comando SQL e a string de conexão usando um provedor SQLClient
Dim strSQL As String = "Select NomeDoProduto , CódigoDoProduto from Produtos"
Dim strConn As String = "server=(local);Trusted_Connection=yes;database=Teste"

Dim Conn As New SqlConnection(strConn)
Dim objDr As SqlDataReader
Dim Cmd as New SqlCommand(strSQL,Conn)

Conn.Open()

objDR = Cmd.ExecuteReader(System.Data.CommandBehavior.CloseConnection)

ddl.DataSource = objDR
ddl.DataBind()
end if
End Sub

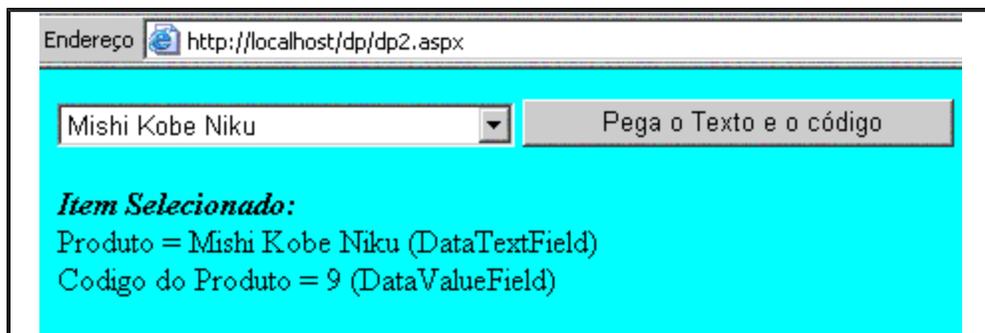
'Rotina que é chamada no evento Click do botão de comando
Sub PreencheControle(Source as Object , E as EventArgs)
label1.text = "<b><i>Item Selecionado: </i></b><br>Produto = " & _
ddl.SelectedItem.text & " (DataTextField) <br> " & _
"Codigo do Produto = " & ddl.SelectedItem.value & " (DataValueField)"
End Sub

</script>

</head>
<body bgcolor="aqua">
'define o controle DropDownList e o botão de comando que no evento OnClick invoca a rotina PreencheControle
<form id="Form1" runat="server">
<asp:DropDownlist id="ddl" DataValueField="CódigoDoProduto" DataTextField="NomeDoProduto" runat="server" />
<asp:Button id="button1" Text=" Pega o Texto e o código" onclick="PreencheControle" runat="server" />
</form>
'define o controle Label
<asp:Label id="label1" runat="server" />
</body>
</html>

```

Ao executar o código ([arquivo dp2.aspx](#)) teremos o resultado:



Preenchendo um controle DropDownList

Preencher uma caixa de combinação (*dropdownlist*) com as informações de uma base de dados Access ou SQL Server é muito simples no ASP.NET. Neste artigo eu vou mostrar como você pode fazer isto.

Para este artigo eu vou criar uma base de dados no Access chamada **Formula1.mdb** e neste banco de dados a tabela **Pilotos** e a tabela **Corridas**; cada uma terá a seguinte estrutura:

Pilotos : Tabela			
	Nome do campo	Tipo de dados	
	CodigoPiloto	AutoNumeração	
	NomePiloto	Texto	
	NumeroPiloto	Texto	
	EquipePiloto	Texto	
	Ativo	Sim/Não	

Corridas : Tabela			
	Nome do campo	Tipo de dados	
	CodigoCorrida	AutoNumeraçãc	
	NomeCorrida	Texto	
	LocalCorrida	Texto	
	DataCorrida	Data/Hora	
	Ativo	Sim/Não	

Tabela Pilotos com os nomes, números e equipes dos pilotos da fórmula 1 atual. Tabela Corridas com as corridas de fórmula 1 da temporada.

Vou criar uma página ASP.NET onde irei carregar uma caixa de combinação com o nome (NomeCorrida) das corridas e oito, eu disse oito, caixa de combinações com os nomes dos Pilotos. A idéia é criar um formulário onde o usuário possa fazer um palpite quanto a classificação do grid de largada. Só vou implementar o preenchimento das combos.

Use o Visual Studio.NET ou o bloco de notas (a escolha é sua ... 😊) e cria um arquivo com a extensão .aspx (**dica** : para salvar o arquivo com esta extensão no bloco de notas clique em *Salvar Como* e informe o nome do arquivo completo com extensão entre *aspas*). Ex: **f1Palpite.aspx**. (este será o nome do nosso arquivo). Agora digite o código conforme mostrado abaixo: arquivo **f1palpite.aspx**

```
<%@ Import Namespace="System.Data.OLEDB" %>  
<script language="VB" runat="server">
```

```
Sub Page_Load(Src As Object, E As EventArgs)
```

```
Dim strConn as string = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source= " & server.MapPath & "/dados/Formula1.mdb"
```

```
Dim Conn1 as OLEDBConnection
```

```
Dim Rdr1 as OLEDBDataReader
```

```
Dim Cmd1 as OLEDBCommand
```

```
Dim strSQL as string
```

```
Conn1=New OLEDBConnection(strConn)
```

```
strSQL="select distinct NomeCorrida from Corridas"
```

```
Cmd1=New OLEDBCommand(strSQL,Conn1)
```

```
Conn1.Open()
```

```
Rdr1=Cmd1.ExecuteReader()
```

```
cr.DataSource = Rdr1
```

```
cr.DataBind()
```

```
Rdr1.Close()
```

```
strSQL="select NomePiloto from Pilotos"
```

```
Cmd1.commandtext=strSQL
```

```
Rdr1=Cmd1.ExecuteReader()
```

```
pl1.DataSource = Rdr1
```

```
pl1.DataBind()
```

```
Rdr1.Close()
```

```
strSQL="select NomePiloto from Pilotos"
```

```
Cmd1.commandtext=strSQL
```

```
Rdr1=Cmd1.ExecuteReader()
```

```
pl2.DataSource = Rdr1
```

```
pl2.DataBind()
```

```
Rdr1.Close()
```

```
strSQL="select NomePiloto from Pilotos"
```

```
Cmd1.commandtext=strSQL
```

```
Rdr1=Cmd1.ExecuteReader()
```

```
pl3.DataSource = Rdr1
```

```
pl3.DataBind()
```

```
Rdr1.Close()
```

```
strSQL="select NomePiloto from Pilotos"
```

```
Cmd1.commandtext=strSQL
```

```
Rdr1=Cmd1.ExecuteReader()
```

```
pl4.DataSource = Rdr1
```

```
pl4.DataBind()
```

```
Rdr1.Close()
```

```
strSQL="select NomePiloto from Pilotos"
```

```
Cmd1.commandtext=strSQL
```

```
Rdr1=Cmd1.ExecuteReader()
```

```
pl5.DataSource = Rdr1
```

```
pl5.DataBind()
```

```
Rdr1.Close()

strSQL="select NomePiloto from Pilotos"
Cmd1.commandtext=strSQL
Rdr1=Cmd1.ExecuteReader()
pl6.DataSource = Rdr1
pl6.DataBind()
Rdr1.Close()

strSQL="select NomePiloto from Pilotos"
Cmd1.commandtext=strSQL
Rdr1=Cmd1.ExecuteReader()
pl7.DataSource = Rdr1
pl7.DataBind()
Rdr1.Close()

strSQL="select NomePiloto from Pilotos"
Cmd1.commandtext=strSQL
Rdr1=Cmd1.ExecuteReader()
pl8.DataSource = Rdr1
pl8.DataBind()
Rdr1.Close()
Conn1.close()

End Sub

</script>

<html><head>
<title>Caixa de Combinação</title>
</head>
<body bgcolor="aqua">

<form runat="server">
<asp:Table runat="server" GridLines="both" BorderWidth="0px">

<asp:TableRow>
<asp:TableCell>NomeCorrida</asp:TableCell>
<asp:TableCell><ASP:DropDownList id="cr" datatextfield="NomeCorrida" runat="server"/>
</asp:TableCell>
</asp:TableRow>

<asp:TableRow>
<asp:TableCell>Grid de Classificação - Largada</asp:TableCell>
</asp:TableRow>

<asp:TableRow>
<asp:TableCell>Grid - 1o. Lugar</asp:TableCell>
<asp:TableCell><ASP:DropDownList id="pl1" datatextfield="NomePiloto" runat="server"/>
</asp:TableCell>
</asp:TableRow>

<asp:TableRow>
<asp:TableCell>Grid - 2o. Lugar</asp:TableCell>
```

```

<asp:TableCell><ASP:DropDownList id="pl2" datatextfield="NomePiloto" runat="server"/>
</asp:TableCell>
</asp:TableRow>

<asp:TableRow>
<asp:TableCell>Grid - 3o. Lugar</asp:TableCell>
<asp:TableCell><ASP:DropDownList id="pl3" datatextfield="NomePiloto" runat="server"/>
</asp:TableCell>
</asp:TableRow>

<asp:TableRow>
<asp:TableCell>Grid - 4o. Lugar</asp:TableCell>
<asp:TableCell><ASP:DropDownList id="pl4" datatextfield="NomePiloto" runat="server"/>
</asp:TableCell>
</asp:TableRow>

<asp:TableRow>
<asp:TableCell>Grid - 5o. Lugar</asp:TableCell>
<asp:TableCell><ASP:DropDownList id="pl5" datatextfield="NomePiloto" runat="server"/>
</asp:TableCell>
</asp:TableRow>

<asp:TableRow>
<asp:TableCell>Grid - 6o. Lugar</asp:TableCell>
<asp:TableCell><ASP:DropDownList id="pl6" datatextfield="NomePiloto" runat="server"/>
</asp:TableCell>
</asp:TableRow>

<asp:TableRow>
<asp:TableCell>Grid - 7o. Lugar</asp:TableCell>
<asp:TableCell><ASP:DropDownList id="pl7" datatextfield="NomePiloto" runat="server"/>
</asp:TableCell>
</asp:TableRow>

<asp:TableRow>
<asp:TableCell>Grid - 8o. Lugar</asp:TableCell>
<asp:TableCell><ASP:DropDownList id="pl8" datatextfield="NomePiloto" runat="server"/>
</asp:TableCell>
</asp:TableRow>

</asp:Table>
</form>
</body></html>

```

Explicando o código : Todo código esta no evento Load da página

1- definimos a string de conexão usando um provedor OLEDB , pois o banco de dados Formula1.mdb é um banco de dados Microsoft Access; a seguir declaramos as variáveis objeto que iremos usar no código.

```

Dim strConn as string = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source= " & server.MapPath & "/dados/Formula1.mdb"
Dim Conn1 as OLEDBConnection
Dim Rdr1 as OLEDBDataReader
Dim Cmd1 as OLEDBCommand
Dim strSQL as string

```

2- Abrimos a conexão : `Conn1=New OleDbConnection(strConn)`

3- Criamos um objeto `OleDbCommand` na conexão , e executamos a propriedade `ExecuteReader` do objeto `DataReader`. O objeto `cr` é a caixa de combinação definida mais abaixo no código. Fechamos o `DataReader` pois ele mantém um conexão direta com a base de dados

```
strSQL="select distinct NomeCorrida from Corridas"
Cmd1=New OleDbCommand(strSQL,Conn1)
Conn1.Open()
Rdr1=Cmd1.ExecuteReader()
cr.DataSource = Rdr1
cr.DataBind()
Rdr1.Close()
```

4- O código abaixo é repetido para os demais controles : Definimos o tipo do objeto `OleDbCommand` (`cmd1`) e executamos os mesmos passos para a caixa de combinação `pl1`. (`pl2`, `pl3`, `pl4`, ...)

```
strSQL="select NomePiloto from Pilotos"
Cmd1.commandtext=strSQL
Rdr1=Cmd1.ExecuteReader()
pl1.DataSource = Rdr1
pl1.DataBind()
Rdr1.Close()
```

5- O código abaixo refere-se aos controles usados na página. Definimos o objeto do tipo `Table` e em cada célula o objeto `DropDownList` (que eu estou chamando *caixa de combinação*) com a propriedade `id` identificando cada controle e a propriedade `datatextfield` indicando o campo vinculado ao controle.

```
<asp:Table runat="server" GridLines="both" BorderWidth="0px">
<asp:TableRow>
  <asp:TableCell>NomeCorrida</asp:TableCell>
  <asp:TableCell><ASP:DropDownList id="cr" datatextfield="NomeCorrida" runat="server"/>
</asp:TableCell>
</asp:TableRow>
<asp:TableRow>
  <asp:TableCell>Grid de Classificação - Largada</asp:TableCell>
</asp:TableRow>
<asp:TableRow>
  <asp:TableCell>Grid - 1o. Lugar</asp:TableCell>
  <asp:TableCell><ASP:DropDownList id="pl1" datatextfield="NomePiloto" runat="server"/>
</asp:TableCell>
</asp:TableRow>
.....
```

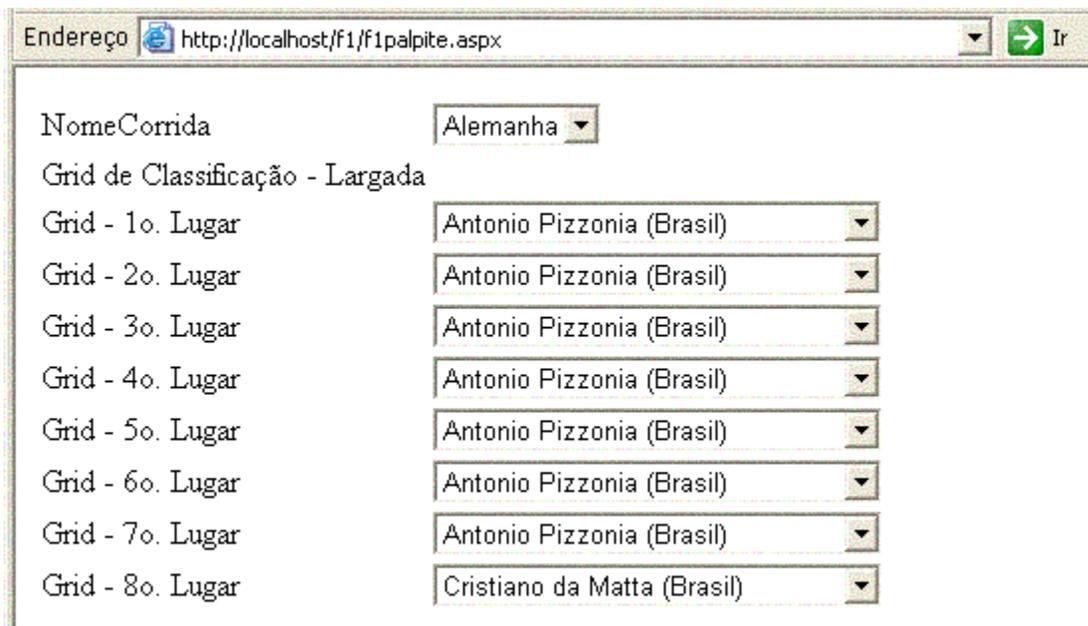
Vejam as **propriedades** de um controle `DropDownList` :

- `AutoPostBack` - Se for True causa um envio (`post`) do formulário quando o cliente altera o item selecionado.
- `DataSource` - Referencia a fonte de dados que o controle usa para preencher os itens.
- `DataTextField` - É usado para preencher o campo `Text` dos itens.
- `DataValueField` - Usado para preencher o campo `Value` dos itens.
- `Items` - Coleção de objetos `ListItem` onde cada objeto representa um item.
- `SelectedItem` - Uma referência o item selecionado.

- [SelectedIndex](#) - Informa o índice do item selecionado. O primeiro tem índice igual a zero.

Evento de um DropDownList : [OnSelectedIndexChanged](#) - Iniciado quando o controle tem a propriedade [AutoPostBack](#) igual a **True** e ocorre mudança no item selecionado.

O arquivo **f1palpite.aspx** foi colocado no diretório **f1** subordinado a `\inetpub\wwwroot`. Ao executarmos a página teremos o resultado:



Veja o resultado on-line em : <http://www.visualbasic.mat.br/f1/f1palpite.aspx>

Exibindo e editando dados em DataGrid

O desenvolvimento para Web recebeu um tratamento especial no Visual Studio .NET ; a ASP.NET veio com o objeto de se tornar uma ferramenta RAD para aplicações WEB ; novas ferramentas , muitas melhorias e no final a vida do desenvolvedor ficou muito mais fácil. Neste artigo eu vou mostrar como exibir e editar dados usando o componente DataGrid . Você poderá verificar como ficou muito mais fácil realizar estas tarefas básicas que antes consumiam muito tempo e código. Na verdade o componente DataGrid será objeto de uma série de artigos abordando todas suas funcionalidades , se , fossemos escrever tudo sobre o

DataGrid em um único artigo ele seria muuuito extenso. Então vamos por partes... 😊

No artigo [Acessando e exibindo dados com ADO.NET](#) eu mostrei como acessar uma base de dados e exibí-los em um DataGrid. Se você ficou decepcionado com a aparência final dos dados exibidos no DataGrid , eu tenho uma boa notícia: no artigo apenas estava focando o acesso aos dados e portanto não configurei o DataGrid para exibir os dados de uma forma mais amigável. Neste artigo vou mostrar como fazer isto.

Para você fazer rodar os exemplos deste artigo você vai precisar ter instalado o .NET Framework e o IIS configurado com um diretório de trabalho. Se você tiver o Visual Studio .NET fica mais fácil você trabalhar , mas você pode usar também o bloco de notas para gerar o código dos exemplos deste artigo.

Nota: Outra opção interessante é usar o [ASP.NET Web Matrix](#) uma ferramenta leve voltada ao desenvolvimento de aplicações ASP.NET.

1- Exibindo dados em um DataGrid :

Na figura abaixo o resultado da exibição dos dados da tabela Produtos do banco de dados Northwind.mdb . Estamos exibindo os produtos com código inferior a 11. (Sql = "SELECT CódigoDoProduto, NomeDoProduto, PreçoUnitário FROM produtos Where CódigoDoProduto < 11")

Endereço  http://localhost/dg1/datagridExibeDados.aspx

Exibindo Dados em um DataGrid

CódigoDoProduto	NomeDoProduto	PreçoUnitário
1	Chai	10
2	Chang	20
3	teste	20
4	Chef Anton's Cajun Seasoning	22
5	Chef Anton's Gumbo Mix	21,35
6	Grandma's Boysenberry Spread	25
7	Uncle Bob's Organic Dried Pears	40
8	Northwoods Cranberry Sauce	40
9	Mishi Kobe Niku	97
10	Ikura	31

Vamos ver como é o código relacionado. Abra o editor de sua preferência e insira o código abaixo , salvando a seguir o arquivo com o nome de : [DataGridExibeDados.aspx](#)

```
<%@Import Namespace="System.Data" %>
<%@Import Namespace="System.Data.OleDb"%>
<%@ Page Language="VB" clienttarget=uplevel%>
```

```
<script language="VB" runat="server">
```

Sub Page_Load(Sender As Object, E As EventArgs)

```
Dim oConexao As OleDbConnection
Dim oDataAdapter As OleDbDataAdapter
Dim oDataSet As DataSet
Dim Sql As String
Dim i As Integer
```

Monta uma instrução SQL para selecionar as colunas da tabela produtos cujo código do produto seja inferior a 11

```
Sql = "SELECT CódigoDoProduto, NomeDoProduto, PreçoUnitário FROM produtos Where CódigoDoProduto < 11"
```

Abre conexão com o banco de dados que deve esta na pasta : d:\inetpub\wwwroot\dados\Northwind.mdb

```
oConexao = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=d:\inetpub\wwwroot\dados\Northwind.mdb")
oConexao.Open
```

Cria um novo objeto OleDbDataAdapter

```
oDataAdapter = New OleDbDataAdapter(Sql, oConexao)
```

Preenche o DataSet com o conteúdo selecionado da tabela

```
oDataSet = New DataSet  
oDadapter.Fill(oDataSet, "produtos")
```

Fecha os objetos DataAdapter e Connection

```
oConexao.Close()  
oDadapter = Nothing  
oConexao = Nothing
```

exibe uma visão customizada da tabela produtos inserida no dataset

```
DatagridExibeDados.DataSource = oDataSet.Tables("produtos").DefaultView  
DatagridExibeDados.DataBind()
```

End Sub

```
</script>  
  
<html>  
<body>  
  
<form runat="server" method="post">  
  
<H3>Exibindo Dados em um DataGrid</H3>  
  
<asp:DataGrid id="DatagridExibeDados" runat="server"  
Width="700"  
BackColor="aqua"  
BorderColor="Blue"  
ShowFooter="false"  
CellPadding=3  
CellSpacing="0"  
Font-Name="Verdana"  
Font-Size="8pt"  
HeaderStyle-BackColor="#aaaadd"  
MaintainState="false"  
>  
</form>  
</body>  
</html>
```

Como eu já tratei do acesso aos dados no artigo anterior eu vou apenas falar sobre o código usado para configuração :

```
<asp:DataGrid id="DatagridExibeDados" runat="server"
  Width="700"
  BackColor="aqua"
  BorderColor="Blue"
  ShowFooter="false"
  CellPadding=3
  CellSpacing="0"
  Font-Name="Verdana"
  Font-Size="8pt"
  HeaderStyle-BackColor="#aaaadd"
  MaintainState="false"
/>
```

Percebeu como é fácil configurar a exibição dos dados ? Eu apenas declarei o DataGrid em `id="DatagridExibeDados"` e usei as propriedades pertinentes a exibição : `BackColor` , `BorderColor` , `Font-Name` , `Font-Size` ,etc... . A seguir vamos ver configurar as propriedades que permitem a edição de dados em um DataGrid.

2- Editando dados em um DataGrid :

A tarefa de edição de dados em grid , consumia tempo e código no velho e bom ASP . Com ASP.NET tudo ficou mais fácil. Vamos ver como editar os dados da tabela Produtos do banco de dados Northwind.mdb . Os dados já foram exibidos no exemplo acima , agora só temos que configurar as propriedades do DataGrid. Vamos lá...

- Fazendo a conexão com a fonte de dados :

```
<%@Import Namespace="System.Data" %>
<%@Import Namespace="System.Data.OleDb"%>
<%@ Page Language="VB" clienttarget=uplevel%>
<html>
<head>
<script language="VB" runat="server">
```

Sub Page_Load(Sender As Object, E As EventArgs)

```
  If Not Page.IsPostBack Then
    BindData()
  End If
```

End Sub

Public Sub BindData()

```
Dim oConexao As OleDbConnection
Dim oDataAdapter As OleDbDataAdapter
Dim oDataSet As DataSet
Dim Sql As String
Dim i As Integer
```

'Monta uma instrução SQL para selecionar as colunas da tabela produtos cujo código do produto seja inferior a 11

```
Sql = "SELECT CódigoDoProduto, NomeDoProduto, PreçoUnitário FROM produtos Where CódigoDoProduto < 11"
```

'Abre conexão com o banco de dados que deve esta na pasta : d:\inetpub\wwwroot\dados\Northwind.mdb

```
oConexao = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=d:\inetpub\wwwroot\dados\Northwind.mdb")
oConexao.Open
```

```

' Cria um novo objeto OleDbDataAdapter
oDadapater = New OleDbDataAdapter(Sql, oConexao)

' Preenche o DataSet com o conteúdo selecionado da tabela
oDataSet = New DataSet
oDadapater.Fill(oDataSet, "produtos")

' Fecha os objetos DataAdapter e Connection
oConexao.Close()
oDadapater = Nothing
oConexao = Nothing

' Exibe uma visão customizada da tabela produtos inserida no dataset
DatagridEditaDados.DataSource = oDataSet.Tables("produtos").DefaultView
DatagridEditaDados.DataBind()
End Sub

```

Aqui não temos nada de novo, eu apenas criei uma rotina chamada BindData() onde eu coloquei o código para fazer a conexão pois eu vou chamar esta rotina mais de uma vez no código para fazer a edição dos dados.

Para permitir a edição no DataGrid vamos ter que incluir a propriedade **<columns></Columns>** na definição do DataGrid e a seguir definir o tipo de coluna que vamos editar. Podemos fazer isto através das propriedades : **EditCommandColumn** e **BoundColumn** para cada coluna que desejamos exibir. Vejamos o código

```

<asp:DataGrid id="DatagridEditaDados" runat="server"
AutoGenerateColumns="False"
Width="700"
BackColor="aqua"
BorderColor="Blue"
ShowFooter="false"
CellPadding=3
CellSpacing="0"
Font-Name="Verdana"
Font-Size="8pt"
HeaderStyle-BackColor="#aaaadd"
MaintainState="false"
OnEditCommand="DataGrid_Edit"
OnCancelCommand="DataGrid_Cancel"
OnUpdateCommand="DataGrid_Atualiza">

<Columns>
  <asp:EditCommandColumn ButtonType="LinkButton" CancelText="Cancelar" EditText="Editar" UpdateText="Atualizar" />
  <asp:BoundColumn DataField="CodigoDoProduto" HeaderText="Código Do Produto" ReadOnly="True" />
  <asp:BoundColumn DataField="NomeDoProduto" HeaderText="Nome Do Produto" />
  <asp:BoundColumn DataField="PreçoUnitário" HeaderText="Preço Unitário" />
</Columns>

</asp:DataGrid>

```

Como funciona ?

EditCommandColumn ; gera um conjunto de botões - **Edit , Update e Cancel**
 Definimos então que o tipo de botão deve ser um link - **ButtonType="LinkButton"**

e o texto que deve ser associado a cada link

- **CancelText= "Cancelar" EditText= "Editar" UpdateText= "Atualizar"**

BoundColumn : Faz uma associação simples com um campo e permite a edição deste campo.

DataField - define o nome do campo associado e

HeaderText - define o texto que vamos exibir na coluna do DataGrid

Obs: como não deseja permitir a edição do campo **CódigoDoProduto** definimos a propriedade **ReadOnly** igual a **True**.

A primeira parte esta pronta : configuramos e definimos as colunas para edição ; se você rodar a página agora irá obter o resultado abaixo. Mas se clicar no link **Editar** nada vai acontecer , pois falta definir os eventos e seus códigos associados.

Endereço  http://localhost/dg1/datagridEditaDados.aspx

Editando registros em um DataGrid

	Código Do Produto	Nome Do Produto	Preço Unitário
Editar	1	Chai	20
Editar	2	Chang	19
Editar	3	Aniseed Syrup	10
Editar	4	Chef Anton's Cajun Seasoning	22
Editar	5	Chef Anton's Gumbo Mix	21,35
Editar	6	Grandma's Boysenberry Spread	25
Editar	7	Uncle Bob's Organic Dried Pears	40
Editar	8	Northwoods Cranberry Sauce	40
Editar	9	Mishi Kobe Niku	97
Editar	10	Ikura	31

>

Para permitir a edição vamos definir primeiro os eventos e as rotinas associadas que desejamos chamar. Neste caso vamos trabalhar com três eventos : **OnEditCommand** , **OnCancelCommand** e **OnUpdateCommand**.

Vamos definir então cada evento com a rotina a associada . Primeiro definimos os eventos na definição do DataGrid:

OnEditCommand= "DataGrid_Edit"

OnCancelCommand= "DataGrid_Cancel"

OnUpdateCommand= "DataGrid_Atualiza"

A seguir insira o código para cada rotina associada ao evento

```
Public Sub DataGrid_Edit(Source As Object, E As DataGridCommandEventArgs)
    DatagridEditaDados.EditItemIndex = E.Item.ItemIndex
    BindData()
End Sub
```

rotina associada ao evento - OnEditCommand

```
Public Sub DataGrid_Cancel(Source As Object, E As DataGridCommandEventArgs)
    DatagridEditaDados.EditItemIndex = -1
    BindData()
End Sub
```

rotina associada ao evento - OnCancelCommand

```
Public Sub DataGrid_Atualiza(Source As Object, E As DataGridCommandEventArgs)
```

```
Dim myConnection As OleDbConnection
Dim myCommand As OleDbCommand
```

```
Dim txtProduto As TextBox = E.Item.Cells(2).Controls(0)
Dim txtPreco As TextBox = E.Item.Cells(3).Controls(0)
Dim strAtualiza As String
```

```
strAtualiza = "UPDATE Produtos SET " & _
    "NomeDoProduto = " & txtProduto.Text & ", " & _
    "PreçoUnitário = " & txtPreco.Text & " " & _
    "WHERE CódigoDoProduto = " & E.Item.Cells(1).Text
```

```
myConnection = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=d:\inetpub\wwwroot\dados\Northwind.mdb")
```

```
myCommand = New OleDbCommand(strAtualiza, myConnection)
```

```
myConnection.Open()
myCommand.ExecuteNonQuery()
```

```
DatagridEditaDados.EditItemIndex = -1
BindData()
```

```
End Sub
```

rotina associada ao evento - OnUpdateCommand

Obs: A atualização é feita através da instrução SQL - [UPDATE / SET](#) .

Executando a página e clicando no link - [Editar](#) - os links - Atualizar e Cancelar irão surgir e os campos estarão abertos para edição. Podemos então editar e atualizar ou cancelar a edição conforme figura abaixo:

Editando registros em um DataGrid

	Código Do Produto	Nome Do Produto	Preço Unitário
Atualizar Cancelar	1	<input type="text" value="Chai"/>	<input type="text" value="10"/>
Editar	2	Chang	20
Editar	3	teste	20
Editar	4	Chef Anton's Cajun Seasoning	22
Editar	5	Chef Anton's Gumbo Mix	21,35
Editar	6	Grandma's Boysenberry Spread	25
Editar	7	Uncle Bob's Organic Dried Pears	40
Editar	8	Northwoods Cranberry Sauce	40
Editar	9	Mishi Kobe Niku	97
Editar	10	Ikura	31

Veja os links funcionando em :

Exibindo dados:	http://www.visualbasic.mat.br/dg/datagridExibeDados.aspx
Editando dados	http://www.visualbasic.mat.br/dg/datagridEditaDados.aspx

Paginando dados em um DataGrid

Já abordamos em outro artigo como exibir dados usando o componente DataGrid ([ASP.NET - Exibindo e editando dados em DataGrid](#)) . Quando a quantidade de dados é grande ficaria inviável exibir todos os dados em uma única página usando o DataGrid . A navegação ficaria prejudicada.

Neste artigo vou mostrar como exibir dados em várias páginas usando o componente DataGrid usando o recurso da paginação. Fazer isto no ASP.NET é muito mais fácil e simples do que no velho e bom ASP.

Para implementar a paginação dos dados no componente DataGrid basta definir as seguintes propriedades :

- 1-) `AllowPaging="True"` - ativa a paginação no componente DataGrid
- 2-) `PageSize="15"` - define o número de registros a ser exibido em cada página
- 3-) `OnPageIndexChanged="dgProdutos_Paged"` - Define o evento que será usado para ativar a paginação.

Para customizar a página usamos a tag **PagerStyle** que permite a navegação pelas páginas de dados. Nesta tag definimos :

1- A propriedade **Mode** pode receber dois valores que definem como serão os links de navegação :

- **NumericPages** - valores numéricos indicam as páginas
- **NextPrev** - definem links para a página anterior e a próxima página

2- A propriedade **Position** pode ter um dos seguintes valores : **Top , Bottom ou TopAndBottom**

Abaixo temos o código da página - **dgPagar.aspx** - que exibe os dados da tabela **Produtos** do banco de dados **Northwind.mdb** . A pagina utiliza um componente DataGrid identificada pelo nome **dgProdutos** e dois componentes Label - *lblmessage* .

```
<% @Import Namespace="System.Data" %>
<% @Import Namespace="System.Data.OleDb" %>

<script language="vb" runat="server">

Sub Page_Load(sender as Object, e as EventArgs)

    If Not Page.IsPostBack then
        'Temos que fazer a vinculação dos dados apenas na primeira vez que a página for carregada
        'cada subsequente postback terá a vinculação feita no evento dgProdutos_Paged
        BindData()
    End If

End Sub

Sub BindData()
    '1. Cria a conexao
    Dim myConnection As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:/teste/Northwind.mdb"
    Dim cn As OleDbConnection = New OleDbConnection(myConnection)

    '2. Cria o objeto command passando a string SQL
    Const strSQL as String = "SELECT CódigoDoProduto , NomeDoProduto , PreçoUnitário FROM Produtos"
    Dim myCommand as New OleDbCommand(strSQL, cn)

    '3. Cria o objeto DataAdapter
    Dim myDA as New OleDbDataAdapter()
    myDA.SelectCommand = myCommand

    '4. Preenche o DataSet
    Dim myDS as New DataSet()
    myDA.Fill(myDS)

    'Define a fonte de dados do datagrid
    dgProdutos.DataSource = myDS
    dgProdutos.DataBind()

    'mostra a informacao na pagina atual
    lblMessage.Text = "Pagina " & dgProdutos.CurrentPageIndex+ 1 & " de " & dgProdutos.PageCount

End Sub

Sub dgProdutos_Paged(sender as Object , e as DataGridPageChangedEventArgs)
    dgProdutos.CurrentPageIndex = e.NewPageIndex
    BindData()
End Sub
</script>
```

```

<html>
<body>
<h3> Dados da tabela Produtos :<code>SELECT CódigoDoProduto,NomeDoProduto,PreçoUnitário FROM Produtos</code></h3>

<form runat="server">
<br>
<asp:datagrid id="dgProdutos" runat="server" BorderWidth="0"
CellPadding="2" Width="100%"
Font-Name="Verdana"
Font-Size="Smaller"
AutoGenerateColumns="False"

HeaderStyle-HorizontalAlign="Left"
HeaderStyle-Font-Bold="True"
HeaderStyle-BackColor="Navy"
HeaderStyle-ForeColor="White"

AlternatingItemStyle-BackColor="#d4d4d4"

AllowPaging="True"
PageSize="15"
OnPageIndexChanged="dgProdutos_Paged">

<PagerStyle Mode="NextPrev" HorizontalAlign="Right" ForeColor="White" BackColor="Navy" NextPageText="Próxima >> " PrevPageText=" << Anterior">
</PagerStyle>

<Columns>

<asp:BoundColumn HeaderText="Cód. Produto" DataField="CódigoDoProduto" ItemStyle-HorizontalAlign="Left" />
<asp:BoundColumn HeaderText="Nome Produto" DataField="NomeDoProduto" ItemStyle-HorizontalAlign="Left" />
<asp:BoundColumn HeaderText="Preço Unitário" DataField="PreçoUnitário" ItemStyle-HorizontalAlign="Left" />

</Columns>

</asp:datagrid>
<asp:label HorizontalAlign="Center" runat="server" id="lblMessage" Font-Name="Verdana" Font-Italic="True" Font-Size="Smaller" />
</form>
</body>
</html>

```

O código que formata as colunas do **DataGrid** é definida no código :

```

<asp:BoundColumn HeaderText="Cód. Produto" DataField="CódigoDoProduto" ItemStyle-HorizontalAlign="Left" />
<asp:BoundColumn HeaderText="Nome Produto" DataField="NomeDoProduto" ItemStyle-HorizontalAlign="Left" />
<asp:BoundColumn HeaderText="Preço Unitário" DataField="PreçoUnitário" ItemStyle-HorizontalAlign="Left" />

```

O método `OnPageIndexChanged` tem uma propriedade `e` do tipo **DataGridPageChangedEventArgs** que possui a propriedade `NewPageIndex` que irá armazenar a página atual. Para fazer com que o DataGrid mostre os dados da página temos que definir a propriedade `CurrentPageIndex` :

```
dgProdutos.CurrentPageIndex = e.NewPageIndex
```

Para executar a página basta copiar o arquivo **dgPaginar.aspx** para o seu diretório de trabalho. O resultado do processamento é exibido na figura abaixo:

Endereço <http://localhost/dgpg/dgPaginar.aspx> Ir Links

Mais de 60.000 ofertas Shopping Assine Veja TÊNIS Novo Ford EcoSport Classificados: Empregos FAREJADOR IG

Dados da tabela Produtos :SELECT CódigoDoProduto, NomeDoProduto, PreçoUnitário FROM Produtos

Cód. Produto	Nome Produto	Preço Unitário
31	Gorgonzola Telino	12,5
32	Mascarpone Fabioli	32
33	Geitost	2,5
34	Sasquatch Ale	14
35	Steeleye Stout	18
36	Inlagd Sill	19
37	Gravad lax	26
38	Côte de Blaye	263,5
39	Chartreuse verte	18
40	Boston Crab Meat	18,4
41	Jack's New England Clam Chowder	9,65
42	Singaporean Hokkien Fried Mee	14
43	Ipoh Coffee	46
44	Gula Malacca	19,45
45	Røgede sild	9,5

<< Anterior Próxima >>

Página 3 de 6

Acesse a página no link : <http://www.visualbasic.mat.br/dgpg/dgPaginar.aspx>

Exibindo imagens em um controle DataGrid

ASP.NET é uma ferramenta sensacional para o desenvolvimento Web. Além de oferecer uma interface muito amigável traz muitos controles que facilitam a vida de qualquer desenvolvedor. Um destes controles é o DataGrid . Posso dizer sem medo de errar que é um dos componentes mais fáceis de usar e configurar para o desenvolvimento Web.

Neste artigo vou mostrar como podemos exibir imagens diretamente em um controle DataGrid.

Suponha que você tenha um banco de dados de produtos com os dados de seus produtos ,e , que um destes dados seja o nome da imagem do produto. Vou então criar um banco de dados chamado - **Produtos.mdb** - padrão Access com uma tabela chamada **produtos** que possui a seguinte estrutura e dados:

produtos : Tabela		produtos : Tabela			
Nome do campo	Tipo de dados	id	Nome	Imagem	Comentario
AutoNumeração	AutoNumeração	1	Super CD Visual Basic	cds.gif	Tudo para Visual Basic
Nome	Texto	2	Super CD ASP Total	cds.gif	Tudo para ASP por um
Imagem	Texto	3	Super CD .NET	cds.gif	VB.NET , ASP.NET e
Comentario	Memorando	4	ASP, ADO , Banco de dados na Web	livro_asp.gif	ASP , ADO e ao aces
		5	Escola Infantil	boy4.gif	Sistema para escola ir
		6	Video Locadora	cameras25.gif	Sistema completo par:
		7	Clinica Dentária	medic2c.gif	Sistema para gerencia

Você já percebeu que não estamos armazenando a imagem no banco de dados , apenas uma referência a ela - o nome da imagem.

Pois bem vamos então ter que acessar o banco de dados e exibir os dados e a imagem em um controle DataGrid.

O código completo eu já estou dando abaixo :

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.OleDb" %>
```

```
<script language="VB" runat="server">
```

Sub Page_Load(Source as Object, E as EventArgs)

```
'Dim strConn as string = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & server.mappath("\data\produtos.mdb")
```

```
Dim strConn as string = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\teste\produtos.mdb"
```

```
Dim SQL as string = "Select * from Produtos"
```

```
Dim Conn as New OleDbConnection(strConn)
```

```
Dim objDR as OleDbDataReader
```

```
Dim Cmd as New OleDbCommand(SQL, Conn)
```

```
MyConn.Open()
```

```
objDR= Cmd.ExecuteReader(system.data.CommandBehavior.CloseConnection)
```

```
dg.DataSource = objDR
```

```
dg.DataBind()
```

```
End Sub
```

```
</script>
```

```
<html>
```

```
<head>
```

```
<meta name="GENERATOR" Content="ASP Express 3.0b1">
```

```
<title>Exibindo imagens em um DataGrid</title>
```

```
</head>
```

```
<body>
```

```

<form id="form1" runat="server">
<div align="center">
<asp:DataGrid runat="server"
  Id="dg"
  GridLines="Both"
  cellpadding="0"
  cellspacing="3"
  Headerstyle-BackColor="#8200C5"
  Headerstyle-Forecolor="white"
  Headerstyle-Font-Name="Verdana"
  Headerstyle-Font-Bold="True"
  Headerstyle-Font-Size="14"
  BackColor="#C6C000"
  Font-Name="Verdana"
  Font-Size="10"
  BorderColor="blue"
  AutoGenerateColumns="False">

<columns>

  <asp:BoundColumn HeaderStyle-HorizontalAlign="Center" DataField="Nome" HeaderText="Nome"></asp:BoundColumn>
  <asp:TemplateColumn HeaderStyle-HorizontalAlign="Center">
    <HeaderTemplate>Imagens</HeaderTemplate>
    <ItemTemplate>
      <div align="center"><IMG SRC='<%# Container.DataItem("Imagem") %>' Border="0"><br> </div>
    </ItemTemplate>
  </asp:TemplateColumn>
  <asp:BoundColumn DataField="Comentario" HeaderText="Comentário"></asp:BoundColumn>

</columns>
</asp:DataGrid></div>
</form>
</body>

```

código fonte do arquivo **mostraImagens.aspx**

Podemos dividir o código em duas partes :

1 - O script ASP.NET onde eu estou definindo uma string de conexão e um provedor que faz o acesso a banco de dados usando um objeto **DataReader** e um objeto **Command** onde eu estou selecionando todos os registros da tabela produtos via instrução SQL e a seguir vinculando os dados retornados ao controle **DataGrid** via propriedade **DataSource**.

```
Dim strConn as string = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\teste\produtos.mdb"
```

```
Dim SQL as string = "Select * from Produtos"
```

```
Dim Conn as New OleDbConnection(strConn)
```

```
Dim objDR as OleDbDataReader
```

```
Dim Cmd as New OleDbCommand(SQL, Conn)
```

```
MyConn.Open()
```

```
objDR= Cmd.ExecuteReader(system.data.CommandBehavior.CloseConnection)
```

```
dg.DataSource = objDR
```

2- O código que define um controle DataGrid (id= "dg") , configura o cabeçalho e o texto e define quais colunas exibir. Para exibir a imagem eu uso comando :
<IMG SRC='<%# Container.DataItem("Imagem") %>' %>

```
<form id="form1" runat="server">
<div align="center">
<asp:DataGrid runat="server"
  Id="dg"
  GridLines="Both"
  cellpadding="0"
  cellspacing="3"
  HeaderStyle-BackColor="#8200C5"
  HeaderStyle-Forecolor="white"
  HeaderStyle-Font-Name="Verdana"
  HeaderStyle-Font-Bold="True"
  HeaderStyle-Font-Size="14"
  BackColor="#C6C000"
  Font-Name="Verdana"
  Font-Size="10"
  BorderColor="blue"
  AutoGenerateColumns="False">

<columns>

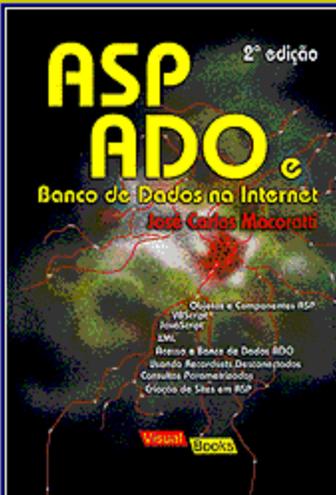
  <asp:BoundColumn HeaderStyle-HorizontalAlign="Center" DataField="Nome"HeaderText="Nome"></asp:BoundColumn>
  <asp:TemplateColumn HeaderStyle-HorizontalAlign="Center">
    <HeaderTemplate>Imagens </HeaderTemplate>
    <ItemTemplate>
      <div align="center"><IMG SRC='<%# Container.DataItem("Imagem") %>' Border="0"><br> </div>
    </ItemTemplate>
  </asp:TemplateColumn>
  <asp:BoundColumn DataField="Comentario" HeaderText="Comentário"></asp:BoundColumn>

</columns>
</asp:DataGrid></div>
</form>
```

Perceba que os campos **Nome** e **Comentário** estão sendo vinculados e definidos via **DataField** .

Executando o projeto no seu Navegador teremos:

Endereço  http://localhost/img/mostraImagens.aspx  Ir  Link

Nome	Imagens	Comentário
Clinica Dentária		Sistema para gerenciar clinica dentária.
Super CD Visual Basic		Tudo para Visual Basic num CD.
Super CD ASP Total		Tudo para ASP por um preço inacreditável.
Super CD .NET		VB.NET , ASP.NET e C# a sua porta de entrada para o mundo .NET.
ASP , Banco de dados na Web		ASP , ADO e ao acesso aos dados em páginas ASP. Conceitos com exemplos práticos
Escola Infantil		Sistema para escola infantil. Completo com os fontes.
Video Locadora		Sistema completo para gerenciar uma video locadora.

É claro que você pode mostrar imagens mais ajustadas ao Grid . Ficamos por aqui...

Usando o componente DataList

A plataforma .NET oferece muitas vantagens no desenvolvimento de aplicações para internet : facilidade de uso , pouco código , recursos de IDE integrado do **RAD** - [Rapid Application Development](#) , componentes otimizados , e por ai vai...

Neste artigo eu vou falar do componente **DataList** , ele é um componente ideal para exibir um conjunto de dados a partir de uma fonte de informações , de um vetor , banco de dados , etc. Seu objetivo é claro: ser leve e usar pouco código.

O **DataList** define [templates](#) para exibição de dados. Nele podemos ter os seguinte [templates](#) :

1. **Header** - primeiro template para cabeçalho
2. **Item** - local onde iremos exibir os itens de informação
3. **AlternateItem** - usado de forma intercalada com o template item
4. **EditItem** - usado para editar as informações
5. **SelectedItem** - usado para exibir os itens selecionados
6. **Separator** - template entre os itens
7. **Footer** - último template para rodapé

Podemos personalizar a exibição do componente através das seguintes propriedades:

1. **RepeatLayout** = " [Flow](#) | [Table](#) " - Podemos usar Flow ou Table , indicando se os itens serão exibidos em sequência ou em uma estrutura de tabela
2. **GridLines** = " [None](#) | [Horizontal](#) | [Vertical](#) | [Both](#) " - Indica se iremos exibir linhas de grade no componente de acordo com as opções permitidas
3. **RepeatColumns**= "NúmeroColunas" - Informa a quantidade de colunas que serão usadas para exibir os dados.
4. **RepeatDirection** = " [Vertical](#) | [Horizontal](#) " - Indica a direção na qual os dados serão preenchidos.
5. **ShowHeader** = " [False](#) | [True](#) " - indica se o template cabeçalho será exibido.
6. **ShowFooter** = " [False](#) | [True](#) " - indica se o template rodapé será exibido.
7. **DataSource** = "<% expressão databinding %>" - Indica a fonte de dados
8. **OnCancelCommand** = "OnCancelCommandMethod" - habilita o comando para cancelar a operação.
9. **OnDeleteCommand** = "OnDeleteCommandMethod" - habilita o comando para deletar dados.
10. **OnEditCommand** = "OnEditCommandMethod" - habilita o comando para editar dados.
11. **OnUpdateCommand** = "OnUpdateCommandMethod" - habilita o comando para atualizar dados.

Para inserir o valor de um campo em um template que atual sobre um registro em uma das seções : [item](#) , [SelectedItem](#) , [AlternatingItem](#) ou [EditItem](#) devemos usar o **DataBinder.Eval**. O registro a ser exibido é descrito como um **Container.DataItem** usando a seguinte sintaxe:

[DataBinder.Eval](#) ([Container.DataItem](#), "Campo")

Para exibir o campo [NomeDoProduto](#) de um registro fazemos assim :

```
<%# DataBinder.Eval(Container.DataItem, "NomeDoProduto") %>
```

Vamos agora a um exemplo de utilização do DataList. Vou acessar e exibir os dados da tabela Produtos do banco de dados Access **Northwind.mdb**. Vou usar um objeto [DataSet](#) para obter os dados da tabela e exibir no componente. O código completo da página - **datalist1.aspx** - e dado abaixo :

```

<%@Import Namespace="System.Data" %>
<%@Import Namespace="System.Data.OleDb"%>
<%@ Page Language="VB" clienttarget=uplevel%>

<script language="VB" runat="server">

Sub Page_Load(Sender As Object, E As EventArgs)
Dim oDataAdapter As OleDbDataAdapter
Dim sSql As String = "SELECT * FROM Produtos Where CódigoDoProduto < 20 ORDER BY NomeDoProduto"
Dim oConnection As New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\teste\Northwind.mdb")

oDataAdapter = New OleDbDataAdapter(sSql, oConnection)

Dim oDataSet As New DataSet()

oConnection.Open
oDataAdapter.Fill(oDataSet, "produtos")

Produto.DataSource = oDataSet.Tables("produtos").DefaultView
Produto.DataBind()

End Sub

</script>

<html>
<body topmargin="0" leftmargin="0" marginwidth="0" marginheight="0" bgcolor="#00FFFF">
<form runat=server>

<table width="805" border=0 cellpadding=5 cellspacing=0>
<tr valign=top>
<td width="863" bgcolor="#33CC99">

<asp:DataList id="Produto" runat="server"
gridlines="Both"
RepeatColumns="4"
RepeatDirection="Horizontal"
CellPadding=3
CellSpacing="0"
Font-Name="Arial"
Font-Size="8pt">

<ItemTemplate>
<b><%# DataBinder.Eval(Container.DataItem, "NomeDoProduto") %></b><br>
<font><%# DataBinder.Eval(Container.DataItem, "QuantidadePorUnidade") %></font><br>
Código: <%# DataBinder.Eval(Container.DataItem, "CódigoDoProduto") %><br>
Quantidade em Estoque: <%# DataBinder.Eval(Container.DataItem, "UnidadesEmEstoque") %><br>
<%# FormatCurrency(DataBinder.Eval(Container.DataItem, "PreçoUnitário")) %><br>
</ItemTemplate>

<HeaderTemplate>
<tr height=1>
<td colspan=4><strong>Catálogo de Produtos:</strong></td>
<tr height=2 bgcolor="#a0522d"><td colspan=4></td></tr>

```

```

</tr>
</HeaderTemplate>

<FooterTemplate>
  <tr height= 1><td colspan=2></td></tr>
</FooterTemplate>

</asp:DataList>
</td></tr>
</table>
</form>
</body>
</html>

```

Os pontos principais do código são :

<pre> <%@Import Namespace="System.Data" %> <%@Import Namespace="System.Data.OleDb"%> <%@ Page Language="VB" clienttarget=uplevel%> </pre>	<p>Como vou acessar uma base de dados Access tenho que usar o namespace - System.Data.OleDb</p>
<pre> Dim oDataAdapter As OleDbDataAdapter Dim sSql As String = "SELECT * FROM Produtos Where CódigoDoProduto < 20 ORDER BY NomeDoProduto" Dim oConnection As New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\teste\Northwind.mdb") </pre>	<ol style="list-style-type: none"> 1. No evento Load da página estou declarando o objeto DataAdapter 2. Declaro a string SQL (selecione somente os produtos com código menor que 20) 3. Crio uma conexão usando o provedor OLE DB
<pre> oDataAdapter = New OleDbDataAdapter(sSql, oConnection) Dim oDataSet As New DataSet() oConnection.Open oDataAdapter.Fill(oDataSet, "produtos") </pre>	<ul style="list-style-type: none"> • Crio o objeto DataAdapter • Crio o objeto DataSet • Abro a conexão • Preencho o DataSet
<pre> Produto.DataSource = oDataSet.Tables("produtos").DefaultView Produto.DataBind() </pre>	<ul style="list-style-type: none"> • Defino o modo de exibição • faço a vinculação com componente
<pre> <asp:DataList id="Produto" runat="server" gridlines="Both" RepeatColumns="4" RepeatDirection="Horizontal" CellPadding=3 CellSpacing="0" Font-Name="Arial" Font-Size="8pt"> </pre>	<ul style="list-style-type: none"> • Defino o componente DataList • Faço algumas configurações de exibição : mostrar linhas de grade , com 4 colunas , na direção horizontal

```

<ItemTemplate>
  <b><%# DataBinder.Eval(Container.DataItem, "NomeDoProduto")
%></b><br>
  <%# DataBinder.Eval(Container.DataItem, "QuantidadePorUnidade")
%></font><br>
  Código: <%# DataBinder.Eval(Container.DataItem, "CódigoDoProduto")
%><br>
  Quantidade em Estoque: <%# DataBinder.Eval(Container.DataItem,
"UnidadesEmEstoque") %><br>
  <%# FormatCurrency(DataBinder.Eval(Container.DataItem,
"PreçoUnitário")) %><br>
</ItemTemplate>

```

- Defino através do DataBinder cada campo que desejo exibir no template : **ItemTemplate**

```

<HeaderTemplate>
  <tr height= 1 >
    <td colspan=4><strong>Catálogo de Produtos:</strong></td>
  <tr height=2 bgcolor="#a0522d"><td colspan=4></td></tr>
</HeaderTemplate>

<FooterTemplate>
  <tr height= 1><td colspan=2></td></tr>
</FooterTemplate>

</asp:DataList>
</td></tr>
</table>
</form>
</body>
</html>

```

- Defino o cabeçalho - template => **HeaderTemplate**
- Defino o rodapé : template - **FooterTemplate**

Ao chamar o arquivo **datalist1.aspx** no servidor IIS teremos:

Endereço <http://localhost/datalist/datalist1.aspx>

IG mail morango Grandes Negócios IG Shopping último segundo

Catálogo de Produtos:

Alice Mutton 20 latas de 1 kg Código: 17 Quantidade em Estoque: 0 R\$ 1,00	Aniseed Syrup 12 garrafas de 550 ml Código: 3 Quantidade em Estoque: 13 R\$ 10,00	Carnarvon Tigers Pacote de 16 kg Código: 18 Quantidade em Estoque: 42 R\$ 62,50	Chai 10 caixas x 20 pacotes Código: 1 Quantidade em Estoque: 39 R\$ 20,00
Chang 24 garrafas de 12 oz Código: 2 Quantidade em Estoque: 17 R\$ 19,00	Chef Anton's Cajun Seasoning 48 vidros de 6 oz Código: 4 Quantidade em Estoque: 53 R\$ 22,00	Chef Anton's Gumbo Mix 36 caixas Código: 5 Quantidade em Estoque: 0 R\$ 21,35	Genen Shouyu 24 garrafas de 250 ml Código: 15 Quantidade em Estoque: 39 R\$ 15,50
Grandma's Boysenberry Spread 12 vidros de 8 oz Código: 6 Quantidade em Estoque: 120 R\$ 25,00	Ikura 12 vidros de 200 ml Código: 10 Quantidade em Estoque: 31 R\$ 31,00	Konbu Caixa de 2 kg Código: 13 Quantidade em Estoque: 24 R\$ 6,00	Mishi Kobe Iiku 18 pacotes de 500 g Código: 9 Quantidade em Estoque: 29 R\$ 97,00
Northwoods Cranberry Sauce 12 vidros de 12 oz Código: 8 Quantidade em Estoque: 6 R\$ 40,00	Pavlova 32 caixas de 500 g Código: 16 Quantidade em Estoque: 29 R\$ 17,45	Queso Cabrales Pacote de 1 kg Código: 11 Quantidade em Estoque: 22 R\$ 21,00	Queso Manchego La Pastora 10 pacotes de 500 g Código: 12 Quantidade em Estoque: 86 R\$ 38,00
Teatime Chocolate Biscuits 10 caixas x 12 unidades Código: 19 Quantidade em Estoque: 25 R\$ 9,20	Tofu 40 pacotes de 100 g Código: 14 Quantidade em Estoque: 35 R\$ 23,25	Uncle Bob's Organic Dried Pears 12 pacotes de 1 lb Código: 7 Quantidade em Estoque: 15 R\$ 40,00	

Para ver o exemplo funcionando acesse o link : <http://www.visualbasic.mat.br/dtl/datalist1.aspx>

Simple e prático, o componente DataList é ótimo para exibir dados em páginas WEB. A seguir veremos como usar os recursos de edição de dados no componente **DataList** . Aguarde

Validação de formulário : CEP , Email e CPF

A ASP.NET veio para facilitar ainda mais o desenvolvimento para WEB , trazendo consigo todo um conjunto inovações que visam facilitar a vida do desenvolvedor WEB. A começar com o ambiente de desenvolvimento onde podemos ter uma interface parecida com a do Visual Basic , onde basta arrastar um componente visual para criar o código a ele associado.

Uma das muitas tarefas na qual o ASP.NET facilita a vida do desenvolvedor e a validação de dados de formulário. Se você já uso ASP ou outra linguagem de script para Web sabe o que validar um campo de *Email* , *Cep* ou *CPF*. Realmente dá trabalho. Com ASP .NET a tarefa ficou muito simples , pois ela disponibiliza controles específicos para validação de dados que associados aos controles de formulários realizam a validação de forma simples.

Uma grande vantagem no modelo de validação do ASP.NET e que não precisamos saber onde ela será executada , se no servidor ou no Browser pois ela se adapta

ao tipo de Browser que o usuário estiver usando. Se o Browser for incompatível a validação será feita apenas no servidor.

No artigo - [Trabalhando com Controles e Web Forms - II - Validação](#) - eu mostrei quais os controles e tipos de validação . Hoje vou mostrar apenas como podemos validar o *CEP* , *CPF* e *Email* usando o controle [RegularExpressionValidator](#).

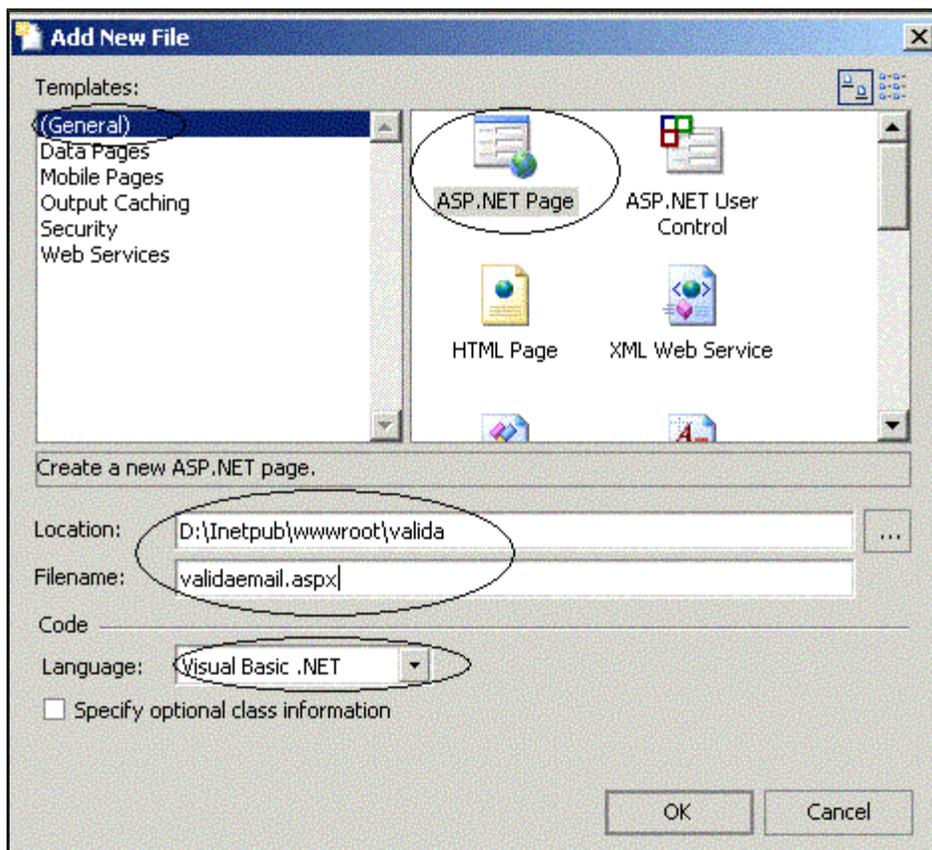
Para testar os exemplos deste artigo você vai precisar do seguinte:

1. Seu sistema operacional deve ser Windows 2000 ou XP .
2. .NET Framework. Não tem !!! 😞 Então pegue em : www.asp.net 😊
3. O IIS deverá estar instalado e configurado - [ASP.NET - Instalando e Configurando o Internet Information Services - IIS](#)

Se você não tem o IIS ou não quer usá-lo pode usar o WebMatrix , uma ferramenta da Microsoft que disponibiliza um Web Server para testes que é fácil de usar. Para baixar o WebMatrix clique no link : www.asp.net

Vou mostrar como fazer a validação de **CEP** , **Email** e **CPF** e vou usar o [WebMatrix](#) para escrever e executar o código. Então se você já baixou e instalou o WebMatrix vamos iniciar executando o programa:

1- O iniciar o programa após a tela de apresentação iremos ter a tela inicial conforme abaixo:



- Nela nos informamos o tipo de projeto que iremos criar. No nosso caso uma página ASP.NET

- A localização do arquivo . Estou usando o diretório padrão de trabalho [d:\inetpub\wwwroot\valida](#)

- O nome do arquivo . Vou começar como a validação de email. Meu arquivo se chamará [validaemail.aspx](#)

- A linguagem usada será a VB.NET (poderíamos usar C#)

- A área de trabalho podemos ter 4 tipos de visões:

1. **Design** - Mostra a interface visual dos controles no formulário
2. **HTML** - exibe o código HTML
3. **Code** - exibe o código das funções e rotinas
4. **All** - mostra todo o código : HTML e scripts

Na janela ao lado já digitei o código do arquivo - **validaemail.aspx** usado para efetuar a validação de Email usando o controle validador - [regularExpressionValidator](#).

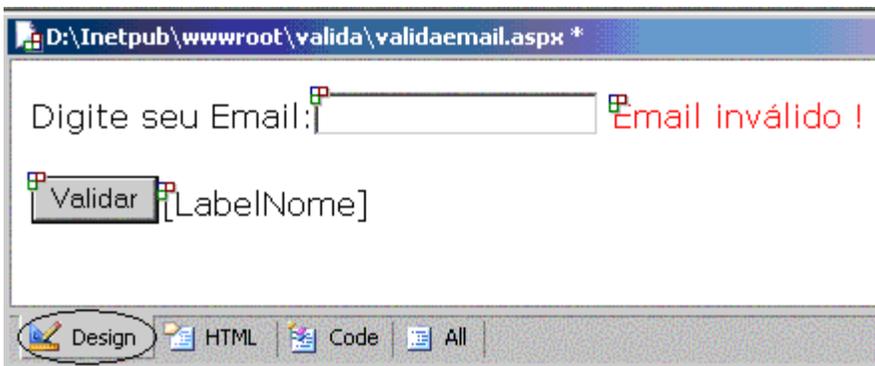
Note que eu tenho que informar:

- qual o controle que eu estou validando em - [ControlToValidate](#).
- O texto que será exibido caso a validação seja inválida
- A expressão de validação que eu desejo usar. ("[\S+@\S+\.\S{2,3}](#)")

```

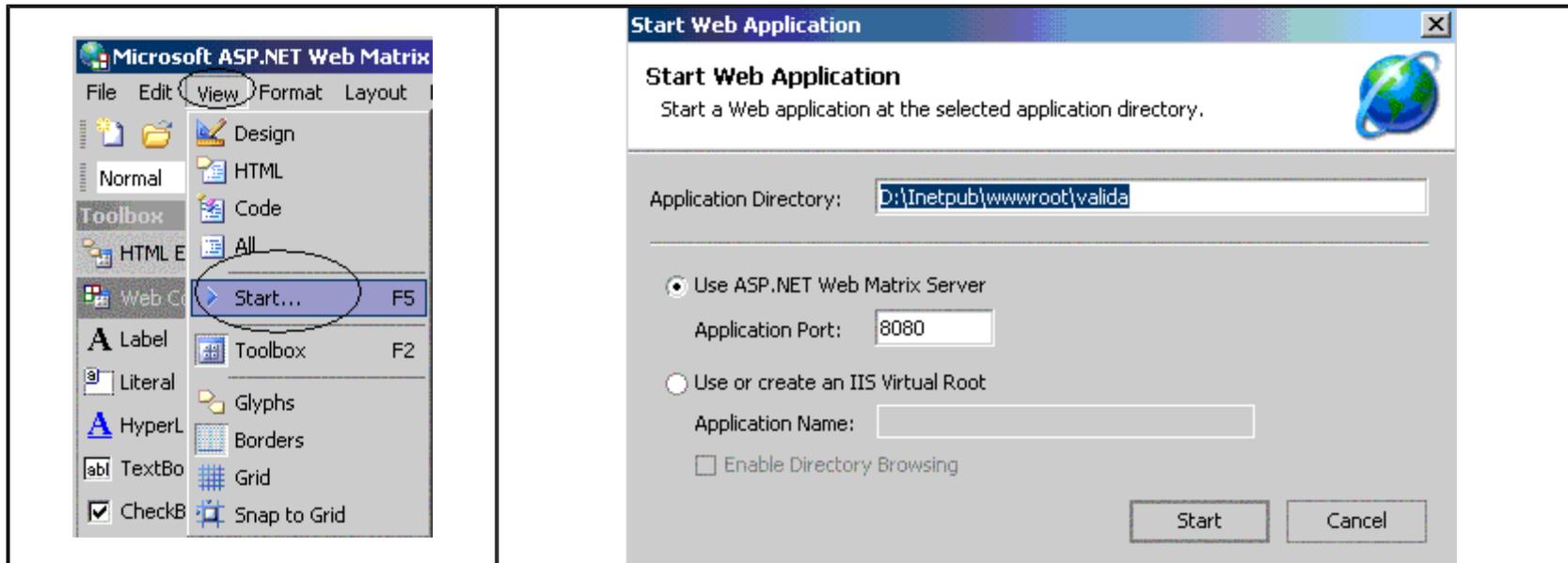
D:\Inetpub\wwwroot\valida\validaemail.aspx *
<%@ Page Language="VB" clienttarget=uplevel %>
<html>
<script language="VB" runat="server">
Sub valida(sender As Object, e As EventArgs)
    If (Page.IsValid) Then
        LabelNome.Text = "Email válido !"
    End If
End Sub
</script>
<body>
<form runat="Server">
    <font face="Verdana">
        Digite seu Email:<asp:TextBox Id="email" RunAt="Server" />
        <asp:regularExpressionValidator
            ControlToValidate="email"
            text="Email inválido !"
            validationExpression="\S+@\S+\.\S{2,3}"
            runat="Server"/>
    <p>
        <asp:Button Id="btValidar" Text="Validar" OnClick="valida" RunAt="Server"/>
        <asp:Label Id="LabelNome" RunAt="Server"/>
    </form>
</body>
</html>
Design HTML Code All
  
```

Para ver o layout do formulário e os controles usados em sua forma visual clique na aba **Design** e você terá:

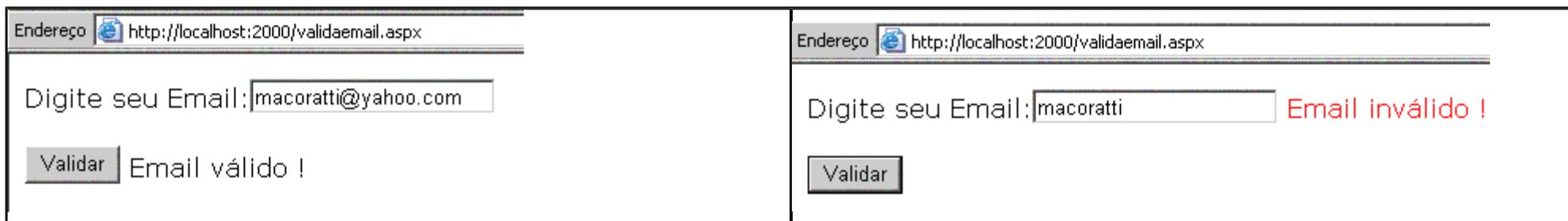


Vamos agora testar o código escrito. Para isto vamos usar o servidor do [WebMatrix](#). Para isto faça o seguinte :

- no menu principal selecione **View** e a seguir **Start**
- Na janela **Start Web Application** indique o diretório onde salvou o arquivo e escolha a opção como abaixo:



Ao executar e realizar os testes teremos como resultado , para email válido e inválido as seguintes telas:



Para as validações de CEP e CPF o esquema se repete o que muda e a expressão de validação. (Em outro artigo estarei entrando em detalhes sobre as [regular expressions](#)). O código é dado abaixo:

1 - Validar CEP - arquivo [validacep.aspx](#) (`validationExpression= "\\d{ 5} \\-\\d{ 3} "`)

```

<%@ Page Language="VB" clienttarget=uplevel %>
<html>

<script language="VB" runat="server">

Sub Valida(sender As Object, e As EventArgs)
If (Page.IsValid) Then
    LabelNome.Text = "CEP válido !"
End If
End Sub

</script>

<body>

<form runat="Server">
<font face="Verdana">

Digite seu CEP:< asp:TextBox Id="cep" RunAt="Server" />
<asp:regularExpressionValidator
ControlToValidate="cep"
text="CEP inválido !"
validationExpression="\d{ 5}\-\d{ 3} "

runat="Server"/>

<p>

<asp:Button Id="btValidar" Text="Validar" OnClick="Valida" RunAt="Server"/>
<asp:Label Id="LabelNome" RunAt="Server"/>
</form>

</body>
</html>

```

2- Validar CPF - arquivo [validacpf.aspx](#) (validationExpression="^\d{2}\.\d{3}\.\d{3}\-\d{2}\$")

```

<%@ Page Language="VB" clienttarget=uplevel %>
<html>

<script language="VB" runat="server">

Sub Valida(sender As Object, e As EventArgs)
If (Page.IsValid) Then
    LabelNome.Text = "CPF válido !"
End If
End Sub

</script>

<body>

<form runat="Server">

```

```
<font face="Verdana">  
Digite seu CPF: <asp:TextBox Id="cpf" RunAt="Server" />  
  
<asp:regularExpressionValidator  
ControlToValidate="cpf"  
text="CPF inválido !"  
validationExpression="^\d{2}\.\d{3}\.\d{3}\-\d{2}$"  
runat="Server"/>  
  
<p>  
<asp:Button Id="btValidar" Text="Validar" OnClick="Valida" RunAt="Server"/>  
<asp:Label Id="LabelNome" RunAt="Server"/>  
</form>  
  
</body>  
</html>
```

Como você pode ver o WebMatrix pode ser uma ferramenta valiosa para você que deseja desenvolver páginas ASP.NET. Aproveite ela não custa nada...

Usando recursos básicos em uma página ASP.NET com acesso a dados

A primeira coisa que você deve fazer neste artigo é acessar o link : <http://www.visualbasic.mat.br/f1/> . Você terá acesso ao site do **Super Bolão Fórmula 1** , um site feito em ASP.NET com algumas funcionalidades básicas que eu vou explicar neste artigo.

- A primeira página que será exibida será a página mostrada ao lado
- Esta página é a gerada pelo arquivo default.aspx (perceba que não foi preciso eu digitar o nome da página no link)
- É uma página de login , e o seu código já foi explicado no artigo : [ASP.NET](#) -

Endereço  http://www.visualbasic.mat.br/f1/



Chave :

Senha:

[Criando um formulário de Login.](#)

- Se você acessar novamente a página vai perceber uma coisa : a imagem vai mudar . Isto vai ocorrer de forma aleatória para 5 imagens que eu cataloguei no código da página.

- Este efeito é possível por que eu estou usando o **Ad Rotator** e é dele que eu vou começar falando neste artigo...

O componente AdRotator - Gerenciando Banners

O componente AdRotator permite o gerenciamento de banners (aquelas figuras , geralmente animadas , com propaganda nos sites ; vou chamá-los a partir de agora de letreiros). A idéia na utilização dos letreiros é personalizar e direcionar a resposta do site ao usuário ; assim sua mensagem será mais eficiente e o usuário ficará mais satisfeito. Desta forma o componente AdRotator permite:

1. cadastrar vários letreiros que serão exibidos de forma aleatória ou com relação a sua importância .
2. exibir letreiros de acordo com o contexto.

Nosso site - Super Bolão F1 - é composto de alguns formulário que acessam uma base de dados ; estes formulário permitem :

- Incluir uma aposta
- Alterar uma aposta
- Ver sua aposta para uma determinada corrida
- Ver a aposta de todo um grupo para uma determinada corrida

O primeiro formulário é o formulário - **default.aspx** - que é o formulário de login do sistema ; onde o usuário deverá informar sua chave e senha para ter acesso ao site. Este formulário possui uma figura que muda de forma aleatória ; aqui estamos usando um componente AdRotator , embora o efeito seja melhor para letreiros dinâmicos. Veja como declaramos o componente [AdRotator](#) :

```
<asp:AdRotator
  id=myAdRotator runat=server
  AdvertisementFile="RandomAd.xml"
  BorderWidth=2
/>
```

Ao lado temos a declaração do nosso componente AdRotator no arquivo [default.aspx](#). Nele destacamos:

id - representa a identificação do componente

AdvertisementFile - indica o arquivo XML de configuração

Todas as configurações deste componente ficam armazenadas em um arquivo XML que pode ficar no diretório raiz da aplicação ou em um diretório central. O conteúdo do nosso arquivo XML - **RandomAd.xml** - é o seguinte :

```
<Advertisements>
<Ad>
  <ImageUrl>senn7.gif</ImageUrl>
  <NavigateUrl>http://www.macoratti.net</NavigateUrl>
  <AlternateText>Banco de dados e Visual Basic</AlternateText>
  <Keyword>bolao1</Keyword>
  <Impressions>20</Impressions>
</Ad>
<Ad>
  <ImageUrl>Senna3.gif</ImageUrl>
  <NavigateUrl>http://www.macoratti.net</NavigateUrl>
  <AlternateText>Super Bolao F1</AlternateText>
  <Keyword>bolao2</Keyword>
  <Impressions>20</Impressions>
</Ad>
<Ad>
  <ImageUrl>Senna17.jpg</ImageUrl>
  <NavigateUrl>http://www.macoratti.net</NavigateUrl>
  <AlternateText>Super Bolao F1</AlternateText>
  <Keyword>bolao3</Keyword>
  <Impressions>20</Impressions>
</Ad>
<Ad>
  <ImageUrl>sunsetbsb.jpg</ImageUrl>
  <NavigateUrl>http://www.macoratti.net</NavigateUrl>
  <AlternateText>Super Bolao F1</AlternateText>
  <Keyword>bolao4</Keyword>
  <Impressions>20</Impressions>
</Ad>
</Advertisements>
```

A primeira coisa que o arquivo xml usado como arquivo de configuração para cadastramento dos letreiros deve obedecer é a nomenclatura e sintaxe da linguagem XML.

A tag raiz do arquivo é a tag - **<Advertisements>**

A tag - **<Ad>** representa cada letreiro cadastrado. Cada bloco **<Ad>** pode conter as seguintes configurações :

- **<ImageUrl>** : A URL do arquivo imagem

- **<NavigateUrl>** : A URL para qual o usuário será direcionado se clicar no letreiro

- **<AlternateText>** : O texto que será exibido no atributo ALT da tag HTML IMG

- **<Keyword>** : define a categoria para a tag **<Ad>**

- **<Impressions>** : A quantidade relativa de exibições que o letreiro terá

Nosso arquivo xml especifica 4 letreiros (na verdade 4 imagens estáticas , pois eu não tive tempo de criar ou procurar por imagens dinâmicas.): *senn7.gif* , *Senna3.gif* , *Senna17.gif* e *sunsetbsb.jpg* (as 3 imagens são de um piloto conhecido ??? e a última é uma imagem de um pôr de sol no DF).

- Se você clicar em qualquer banner irá ser direcionado para a página : <http://www.macoratti.net>
- O texto exibido quando você passa como o mouse sobre as figuras é : *Super Bolão F1*.
- *bolao1* , *bolao2* , *bolao3* e *bolao4* são as keywords definidas para cada letreiro.
- Cada letreiro tem uma incidência de 20 , ou seja , eles terão a mesma probabilidade de aparecer.

Inserindo dados em um banco de dados Access

Esta precisando fazer um banco de dados na web ? Fazer um cadastro de clientes ? Neste artigo eu vou mostrar como podemos inserir dados em um banco de dados Access usando a ADO.NET via páginas ASP.NET com VB.NET. 😊

Vou usar um banco de dados que criei para esta finalidade chamado **Teste.mdb** . Este arquivo possui a tabela **Clientes** na qual vamos incluir dados. sua estrutura é :

Clientes : Tabela	
Nome do campo	Tipo de dados
id	AutoNumeração
nome	Texto
endereco	Texto
cep	Texto
uf	Texto
email	Texto

- É uma tabela simples onde usamos uma chave primária no campo id com a propriedade autonumeração.

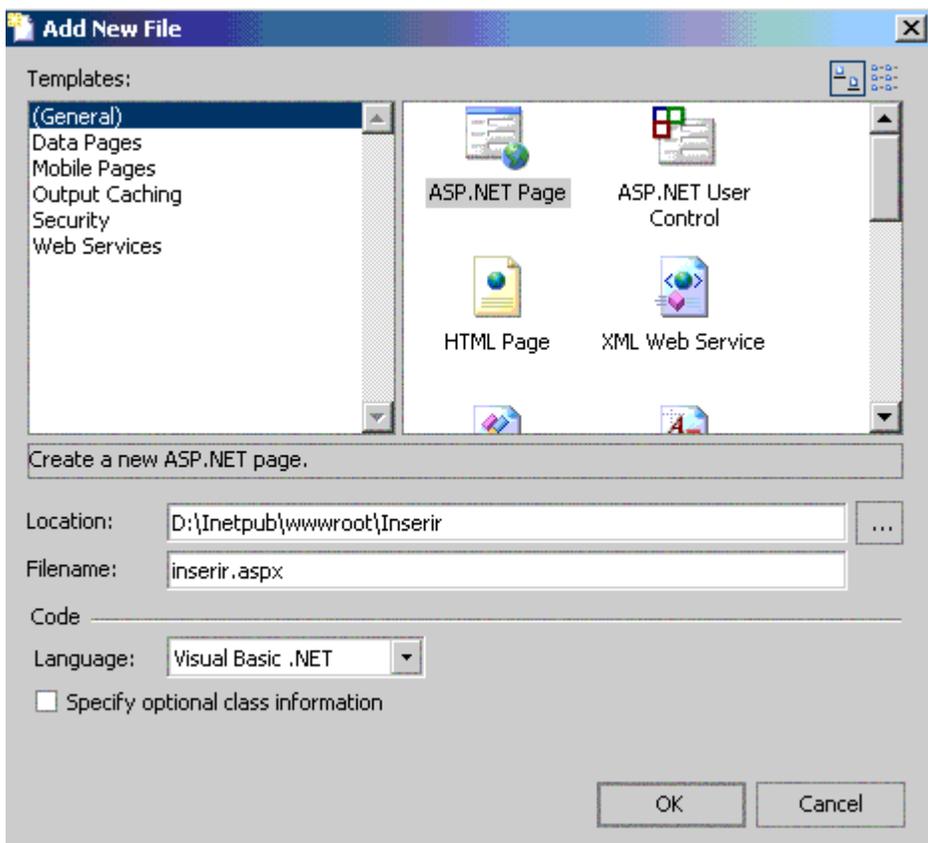
Quando você vai desenvolver uma aplicação para Web deve estar atento as mudanças que ocorrem em relação ao desenvolvimento para o desktop. Quem esta acostumado a usar o Visual Studio com seu IDE integrado para desenvolver aplicações Windows certamente não ia se sentir muito confortável em desenvolver para a Web usando um editor com poucos recursos.

Se você não possui o Visual Studio .NET não fique triste ! 😊. Felizmente agora temos uma ferramenta que facilita muito o desenvolvimento para Web e que possui uma interface bem parecida com o IDE do VS. Estou falando do [WebMatrix](#) uma ferramenta **grátis** (*isto mesmo é grátis*) que você pode usar para desenvolver aplicações ASP.NET. Neste artigo eu vou estar usando o WebMatrix para criar a aplicação que inclui dados na tabela Clientes.

Após fazer o download (www.asp.net) e instalar o [WebMatrix](#) , ao iniciá-lo irá surgir a janela abaixo. Você deverá selecionar um **template** e o tipo de aplicação que deseja criar.

- Como vamos criar uma página [ASP.NET](#) vou selecionar - **General** - e **ASP NET Page**.

- Em **location** informe o local onde deseja salvar sua página



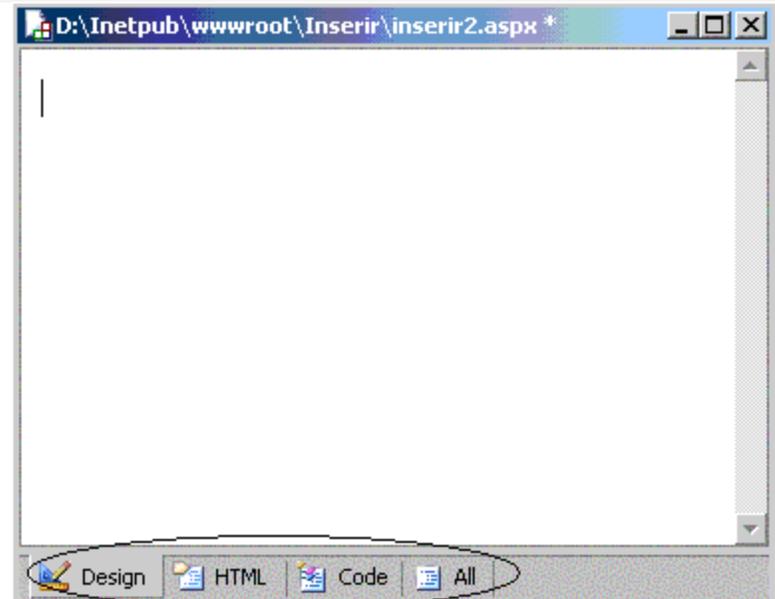
- Em **FileName** - o nome que deseja dar ao seu arquivo.
- Não esqueça de selecionar a linguagem , no nosso caso : **VB.NET**
- Finalmente clique em **OK**.

Pronto será criado um arquivo na pasta : [D:\Inetpub\wwwroot\Inserir](#) com o nome : **inserir.aspx**

O ambiente do WebMatrix é parecido com o IDE do VS. A esquerda temos a caixa de ferramentas com os controles que você pode arrastar e soltar na área de trabalho.

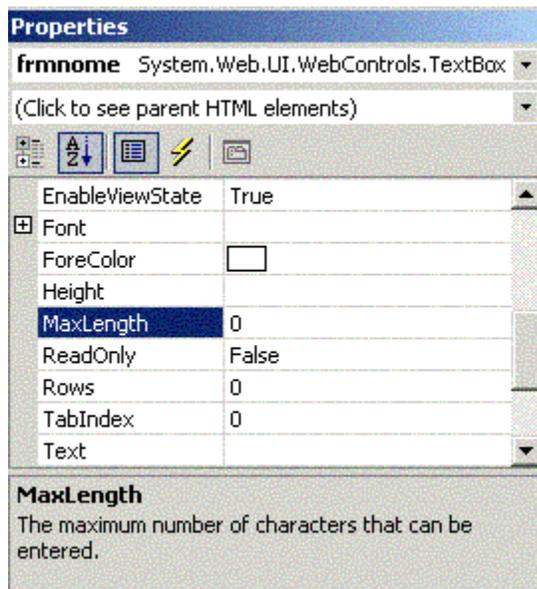
A área de trabalho oferece 4 modos de visualização :

1. Design - visualiza a interface com os controles Web
2. HTML - exibe o código HTML do projeto
3. Code - exibe o código ASP.NET
4. All - exibe todo o código do arquivo



A janela de propriedades dos controles que usamos no projeto exibe as propriedades do controle e permite uma configuração ajustada de cada controle. (Veja

abaixo)



- A propriedade **ID** - identifica o controle

- Podemos então configurar o controle atribuindo valores a cada uma de suas propriedades e ver a visualização na área de trabalho no modo **Design**.

No modo **design** eu vou criar um formulário com os seguintes componentes , conforme figura abaixo:

- [5 TextBox](#) - para cada campo do banco de dados onde vamos incluir as informações
- [5 RequiredFieldValidator](#) - para efetuar a validação dos campos
- [3 Buttons](#) - para incluir , ver os dados e limpar os controles
- [1 Label](#) - para exibir mensagens
- [1 DataGrid](#) - para mostrar os dados cadastrados

Para maiores detalhes sobre a utilização dos controles web e controle para a validação leia em :

1. [Criando seu primeiro Web Forms](#)
2. [Trabalhando com Controles e Web Forms - II - Validação](#)

The image shows a web development interface with a 'Cadastro de Clientes' form. The form has five input fields: 'Nome', 'Endereço', 'Cep', 'Estado', and 'Email'. Each field has a red error message indicating it is required. Below the form are three buttons: 'Inserir', 'Ver Clientes', and 'Limpar'. A data grid below the form has three columns: 'Column0', 'Column1', and 'Column2', with five rows of 'abc' values. A 'Toolbox' on the left lists various web controls like Label, TextBox, CheckBox, etc.

O código HTML , exibido na guia HTML , esta exibido a seguir. Nele eu estou destacando em negrito o código para os controles Web , em azul o código para os controles para validação. O código para o datagrid esta destacado em vermelho.

```
<html>
<head>
<title>ASP.NET - Inserir registros em um banco de dados Access</title>
<meta content="ASP Express 3.0" name="GENERATOR" />
</head>
<body bgcolor="#66cc99">
<form id="form1" runat="server">
<h1 align="center">Cadastro de Clientes
</h1>
<div align="center">
<table align="center" border="0">
<tbody>
<tr>
<td align="right">
<font face="Verdana" size="2">Nome :</font></td>
<asp:textbox id="frmnome" runat="server" Width="210px" MaxLength="50"></asp:textbox>
<asp:RequiredFieldValidator id="vldnome" runat="server" ControlToValidate="frmnome" ErrorMessage="O nome é obrigatorio" display="dynamic">
O Nome é obrigatorio!

```

```

</asp:RequiredFieldValidator>
<font face="Verdana"><font size="2"></font></font></td>
</tr>
<tr>
<td align="right">
<font face="Verdana" size="2">Endereço :</font></td>
<td>
<asp:textbox id="frmendereco" runat="server" Width="210px" MaxLength="50"></asp:textbox>
<asp:RequiredFieldValidator id="vldendereco" runat="server" ControlToValidate="frmendereco" ErrorMessage="O endereço é obrigatório." display="dynamic">
O endereço é obrigatório !
</asp:RequiredFieldValidator>
<font face="Verdana"><font size="2"></font></font></td>
</tr>
<tr>
<td align="right">
<font face="Verdana" size="2">Cep :</font></td>
<td>
<asp:textbox id="frmcep" runat="server" Width="121px"></asp:textbox>
<asp:RequiredFieldValidator id="vldcep" runat="server" ControlToValidate="frmcep" ErrorMessage="O cep é obrigatório." display="dynamic">
O cep é obrigatório !
</asp:RequiredFieldValidator>
<font face="Verdana" size="2"></font></td>
</tr>
<tr>
<td align="right">
<font face="Verdana" size="2">Estado :</font></td>
<td>
<asp:textbox id="frmestado" runat="server" Width="40px" MaxLength="2"></asp:textbox>
<asp:RequiredFieldValidator id="vldestado" runat="server" ControlToValidate="frmestado" ErrorMessage="O estado é obrigatório." display="dynamic">
O estado é obrigatório !
</asp:RequiredFieldValidator>
<font face="Verdana" size="2"></font></td>
</tr>
<tr>
<td align="right">
<font face="Verdana" size="2">Email :</font></td>
<td>
<asp:textbox id="frmemail" runat="server" Width="210px" MaxLength="80"></asp:textbox>
<asp:RequiredFieldValidator id="vldemail" runat="server" ControlToValidate="frmemail" ErrorMessage="O email é obrigatório." display="dynamic">
O email é obrigatório !
</asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td align="middle" colspan="2">
<p align="center">
<asp:button id="button1" onclick="doInserir" runat="server" Text="Inserir"></asp:button>
<asp:Button id="button2" onclick="Exibir" runat="server" Text="Ver Clientes" causesvalidation="false"></asp:Button>
<asp:Button id="button3" onclick="limpartextboxes" runat="server" Text="Limpar" causesvalidation="false"></asp:Button>
</p>
</td>
</tr>
</tbody>

```

```

</table>
</div>
<b><i>
<div align="center"><asp:Label id="label1" runat="server"></asp:Label>
</div>
</i></b>
<hr />
<div align="center">
<asp:Datagrid id="MyDataGrid" runat="server" Width="50%" GridLines="Both" cellpadding="0" cellspacing="0" Headerstyle-BackColor="#00CCFF" Headerstyle-Font-Name="Arial" Headerstyle-Font-Size="14" Headerstyle-Font-Bold="true" BackColor="#33FFCC" Font-Name="Arial" Font-Size="11pt" BorderColor="Black" AllowPaging="True" PageSize="5" PagerStyle-Mode="NumericPages" PagerStyle-PageButtonCount="5" PagerStyle-HorizontalAlign="Center" OnPageIndexChanged="Page_Change" visible="false" Font-Names="Comic Sans MS">
<HeaderStyle font-size="14pt" font-names="Arial" font-bold="True" bgcolor="#00CCFF"></HeaderStyle>
<PagerStyle horizontalalign="Center" pagebuttoncount="5" mode="NumericPages"></PagerStyle>
</asp:Datagrid>
</div>
</form>
</body>
</html>

```

Nota: Os nomes dos controles usados na página são :

- DataGrid = [MyDataGrid](#)
- Label = [label1](#)
- Buttons = [button1](#), [button2](#), [button3](#)
- TextBox = [frmnome](#), [frmendereco](#), [frmcep](#), [frmestado](#), [frmemail](#)
- RequireFieldValidator - [vldnome](#), [vldendereco](#), [vldcep](#), [vldestado](#), [vldemail](#)

Na guia **Code** temos a exibição do código de script ASP.NET. No código temos as seguintes rotinas :

- [doInserir \(\)](#) - Realiza a conexão com a base de dados e faz a inclusão das informação na tabela Clientes
- [Exibir\(\)](#) - Torna visível os controles label e datagrid e chama a rotina BindData()
- [Page_Change\(\)](#) - Se houver alguma alteração na página atualiza o datagrid
- [BindData\(\)](#) - Faz a vinculação dos dados e realiza a exibição no datagrid
- [limpartextboxes\(\)](#) - Limpa as caixas de texto do formulário.

```

<%@ Page Language="vb" %>
<%@ import Namespace="System.Data" %>
<%@ import Namespace="System.Data.OleDb" %>
<script runat="server">

```

[Dim nome, endereco, cep, uf, email as string](#)

Sub doInserir(Source as Object, E as EventArgs)

[Dim MySQL as string = "Insert into Clientes \(nome, endereco , cep , uf , email \) values \(@nome, @endereco ,@cep , @uf , @email\)"](#)

[Dim myConn As OleDbConnection = New OleDbConnection\("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=d:/teste/dados/Teste.mdb"\)](#)

[Dim Cmd as New OleDbCommand\(MySQL, MyConn\)](#)

[cmd.Parameters.Add\(New OleDbParameter\("@nome", frmnome.text\)\)](#)

[cmd.Parameters.Add\(New OleDbParameter\("@endereco", frmendereco.text\)\)](#)

```
cmd.Parameters.Add(New OleDbParameter("@cep", frmcep.text))  
cmd.Parameters.Add(New OleDbParameter("@uf", frmestado.text))  
cmd.Parameters.Add(New OleDbParameter("@email", frmemail.text))
```

```
MyConn.Open()
```

```
cmd.ExecuteNonQuery
```

```
MyConn.Close()
```

```
label1.visible="true"
```

```
BindData()
```

```
label1.text = "Os dados foram salvos na base de dados clientes com sucesso !"
```

```
End Sub
```

```
Sub Exibir(Source as Object, E as EventArgs)
```

```
mydatagrid.visible="true"
```

```
label1.visible="false"
```

```
BindData
```

```
End Sub
```

```
Sub Page_Change(sender As Object, e As DataGridPageChangedEventArgs)
```

```
mydatagrid.visible="true"
```

```
MyDataGrid.CurrentPageIndex = e.NewPageIndex
```

```
BindData
```

```
End Sub
```

```
Sub BindData()
```

```
Dim MySQL as string = "Select * from Clientes order by id"
```

```
Dim myConn As OleDbConnection = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;" & _  
"Data Source=d:/inetpub/wwwroot/dados/Teste.mdb")
```

```
Dim ds as DataSet=New DataSet()
```

```
Dim Cmd as New OleDbDataAdapter(MySQL,MyConn)
```

```
Cmd.Fill(ds,"Clientes")
```

```
MyDataGrid.Datasource=ds.Tables("Clientes").DefaultView
```

```
MyDataGrid.DataBind()
```

```
MyConn.Close()
```

```
End Sub
```

```
Sub limpertextboxes(Source as Object, E as EventArgs)
```

```
Dim myForm As Control = Page.FindControl("form1")
```

```
Dim ctl As Control
```

```
For Each ctl In myForm.Controls
```

```
If ctl.GetType().ToString().Equals("System.Web.UI.WebControls.TextBox") Then
```

```
CType(ctl, TextBox).Text = ""
```

```
End If
```

```
Next ctl
```

End Sub

1-) Eu estou usando um instrução SQL **Insert Into**

- Dim MySQL as string = "Insert into Clientes (nome, endereco , cep , uf , email) values (@nome, @endereco ,@cep , @uf , @email)"

para incluir os dados na tabela clientes. Note que a sintaxe no exige o simbolo @ no inicio dos valores.

2-) A conexão com a fonte de dados é feita usando um provedor OleDb :

Dim myConn As OleDbConnection = New **OleDbConnection**("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=d:/teste/dados/Teste.mdb")

3-) Defini um objeto Command :

Dim Cmd as New OleDbCommand(MySQL, MyConn)

4-) Passei os parâmetros para o objeto commad

```
cmd.Parameters.Add(New OleDbParameter("@nome", frmnome.text))  
cmd.Parameters.Add(New OleDbParameter("@endereco", frmendereco.text))  
cmd.Parameters.Add(New OleDbParameter("@cep", frmcep.text))  
cmd.Parameters.Add(New OleDbParameter("@uf", frmestado.text))  
cmd.Parameters.Add(New OleDbParameter("@email", frmemail.text))
```

5-) Abri a conexão e executei o comando :

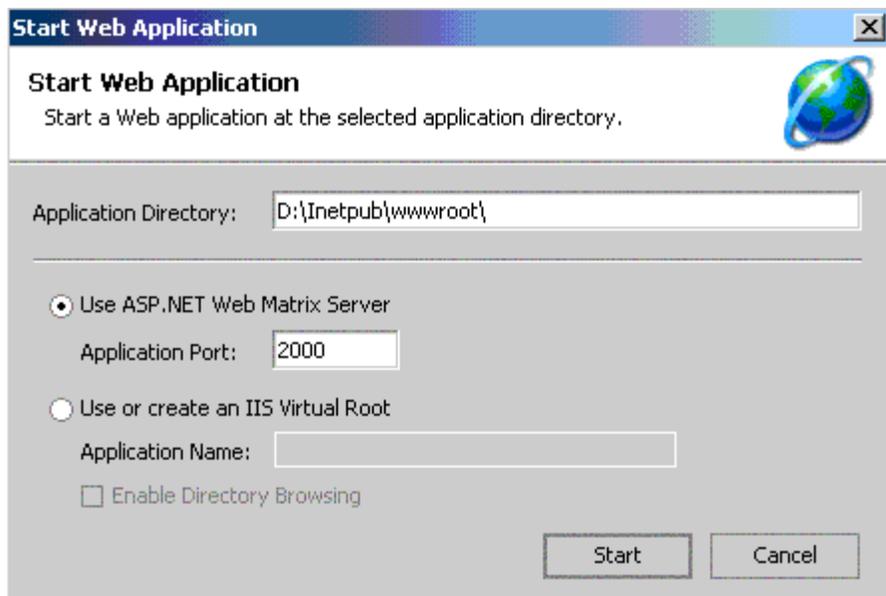
```
MyConn.Open()  
cmd.ExecuteNonQuery
```

Você pode agora testar o script executando o arquivo **inserir.aspx** no seu navegador . Se você tiver o **IIS** instalado e configurado corretamente basta chamar o arquivo a partir do seu servidor web padrão : <http://localhost/inserir.aspx>



Se você não tem o IIS , fique tranquilo , o **WebMatrix** fornece um servidor embutido para testar o seu código. Basta clicar no botão

e você vai obter a tela abaixo:



- Nela você pode escolher entre usar o [Servidor IIS](#) ou o servidor [WebMatrix](#)
- Apenas informe o diretório da aplicação e a porta
- Clique no botão - **Start**
- Pronto sua página deve ser executada.

Se você fez tudo corretamente vai obter a página da aplicação conforme abaixo. Agora fique a vontade para cadastrar os dados de seus clientes. Você pode expandir a aplicação e adaptar ao seu caso particular : *um cadastro de alunos , de produtos , etc..*

http://localhost:2000/inserir.aspx

Cadastro de Clientes

Nome :

Endereço :

Cep :

Estado :

Email :

id	nome	endereço	cep	uf	email
1	Macoratti	Rua Projetada , 100	70589-100	DF	macoratti@yahoo.com
2	Janice	Rua Mirassol , 200	15054-100	SP	janice@bol.com.br
3	Jefferson	Pça da Luz , 500	03666-050	SP	jefferson@yahoo.com
4	Jessica	Av. Girassol , 400	14506-850	SP	jessica@uol.com.br
1					

Veja a página funcionando no link : <http://www.visualbasic.mat.br/inserir.aspx>

Criando um carrinho de compras

Que tal um carrinho de compras ? Carrinho de compras ??? Sim , um carrinho de compras feito em ASP.NET para você adaptar ao seu negócio e incluir no seu site. Além de aprender conceitos importantes de relacionados a tecnologia ASP.NET vai economizar uma grana.

Neste artigo eu vou mostrar como criar e implementar um carrinho de compras . Vamos usar os seguintes ingredientes :

- [Session](#)
- [DataSet](#)
- [DataGrid](#)
- [DataTable](#)

O objetivo deste artigo é mostrar a você conceitos básicos , tais como :

- Criar um interface com o usuário usando os controles de Servidor
- Construir uma tabela dinâmica usando a classe [DataTable](#)
- Exibir os dados selecionados em um [DataGrid](#)
- Remover itens de um [DataGrid](#)
- Totalizar o valor dos itens selecionados

Não vou me preocupar com a aparência da interface , deixo esta parte para você incrementar, OK ?

Criando a interface com o usuário

A interface , como já disse , vai ser simples , vamos usar os seguintes componentes :

1. Um Web Control [DropDownList](#) para exibir os produtos que vamos oferecer. O valor de cada produto estará associado a cada item.
2. Um controle Web [TextBox](#) onde o usuário poderá informar a quantidade do produto que deseja.
3. Um controle Web [Button](#) para incluir o pedido no carrinho de compras
4. Um componente Web [DataGrid](#) que irá exibir o conteúdo do carrinho de compras
5. Um controle Web [Label](#) que irá exibir o valor total do pedido

Você pode usar o Visual Studio .NET para criar a interface ; eu vou usar o editor do *Visual Studio* (você pode usar até Bloco de notas para digitar o código ; se for masoquista... 😊). Veja o código abaixo :

```

<body>
<form runat="server">

Produto: <br>
<asp:DropDownList id=Produtos runat="server">
<asp:ListItem Value="25,00">Super CD Visual Basic - 2003</asp:ListItem>
<asp:ListItem Value="15,00">Super CD Asp Total - 2003</asp:ListItem>
<asp:ListItem Value="100,00">Sistema - Clinicas Dentárias</asp:ListItem>
<asp:ListItem Value="60,00">Sistema - Escola Infantil</asp:ListItem>
<asp:ListItem Value="70,00">Sistema - Videolocadora</asp:ListItem>
</asp:DropDownList> <br>

Quantidade: <br>
<asp:textbox id="txtQuantidade" runat="server" /> <br> <br>
<asp:Button id=btnIncluir runat="server" Text="Incluir no Carrinho" onClick="IncluirNoCarrinho" /> <br> <br>
<asp:DataGrid id=dg runat="server" ondeletecommand="Exclui_Item">

<columns>
<asp:buttoncolumn buttontype="LinkButton" commandname="Delete" text="Remover Item" />
</columns>
</asp:DataGrid>

<br> <br>
Total:
<asp:Label id=lblTotal runat="server" />

</form>
</body>
</html>

```

O código acima preenche o componente **DropDownList** identificado como produtos (**id=produtos**) com 5 produtos ; cada item da lista está associado a seu valor.

A caixa de texto - TextBox - **txtquantidade** - permite que o usuário altere a quantidade dos produtos.

O botão de comando - Button - **btnIncluir** - possui o texto - Incluir no Carrinho . Perceba o evento - **OnClick** - associado ao botão que irá chamar a subrotina - **IncluirNoCarrinho**.

O DataGrid (**id=dg**) irá exibir os valores que serão criados dinamicamente usando um objeto DataTable . O objeto DataTable será criado via código a cada pedido feito pelo usuário.

O controle label (**id=lblTotal**) - irá exibir o valor total dos pedidos

O processamento do código acima irá exibir a estrutura básica da interface com o usuário. O resultado é exibida na tela abaixo :

Criando a estrutura da tabela

Nesta etapa vamos usar o objeto DataTable para criar dinamicamente uma representação de uma tabela do banco de dados na memória. Vamos criar uma tabela via código ; as colunas são representadas pela propriedade **columns** e as linhas pela propriedade **rows**. Após criar a tabela representada pelo objeto DataTable basta vinculá-la ao objeto DataGrid para exibir os dados nela contido.

A estrutura da nossa tabela será bem simples , mas servirá aos nossos propósitos , ela será a seguinte :

Coluna	Tipo de Dados	Autoincrementar	Unico
Codigo	Inteiro	Sim	Sim
Quantidade	Inteiro	Não	Não
Produto	String	Não	Não
Custo	Decimal	Não	Não

O campo - **Codigo** - foi definido com a propriedade - *Autoincrementar* - igual a SIM - desta forma ele será automaticamente incrementado a cada inclusão.

A tabela deverá ser criada quando a página for carregada pela primeira vez. Somente não iremos definir as linhas da tabela , elas serão criadas quando o usuário , selecionar um produto na página. O código para criar a estrutura básica da tabela é o seguinte :

O início do código do arquivo - **cesta.aspx** - deverá ser o seguinte :

```
<%@ Page debug=true Language="VB" ContentType="text/html" ResponseEncoding="iso-8859-1" %>
```

```
<%@ Page debug=true Language="VB" ContentType="text/html" ResponseEncoding="iso-8859-1" %>
```

```
<html>  
<head>  
<title>Carrinho de Compras</title>  
<script runat="server">
```

'cria um nova instância dos objetos Datatable e DataRow

Dim objDT As System.Data.DataTable

Dim objDR As System.Data.DataRow

Nele estamos ativando o modo de depuração , definindo a linguagem usada (VB .NET) e declarando as variáveis objeto do tipo **DataTable** e **DataRow**.

A seguir o código associado ao evento **Load** da página onde iremos invocar a função - [CriaCarrinhoDeCompras\(\)](#) - que irá criar a tabela.

```
Private Sub Page_Load(s As Object, e As EventArgs)  
    If Not IsPostBack Then  
        CriaCarrinhoDeCompras\(\)  
    End If  
End Sub
```

O código da função - [CriaCarrinhoDeCompras\(\)](#) - :

```
Function CriaCarrinhoDeCompras()  
    objDT = New System.Data.DataTable("Carrinho")  
    objDT.Columns.Add("Codigo", GetType(Integer))  
    objDT.Columns("Codigo").AutoIncrement = True  
    objDT.Columns("Codigo").AutoIncrementSeed = 1  
  
    objDT.Columns.Add("Quantidade", GetType(Integer))  
    objDT.Columns.Add("Produto", GetType(String))  
    objDT.Columns.Add("Custo", GetType(Decimal))  
  
    Session("Carrinho") = objDT  
End Function
```

No código definimos as colunas da tabela e armazenamos o objeto na sessão identificada com o nome de - [Carrinho](#).

Incluindo os itens selecionados no carrinho de compras e totalizando

Para incluir itens no carrinho de compras temos que criar novas linhas e então inclui-las na posição apropriada da tabela. O código da função **IncluirNoCarrinho** é dado a seguir:

```

Sub IncluirNoCarrinho(s As Object, e As EventArgs)
objDT = Session("Carrinho")
Dim Produto = Produtos.SelectedItem.Text

Dim blnMatch As Boolean = False

For Each objDR In objDT.Rows
  If objDR("Produto") = Produto Then
    objDR("Quantidade") += txtQuantidade.Text
    blnMatch = True
  Exit For
End If
Next

If Not blnMatch Then
  objDR = objDT.NewRow
  objDR("Quantidade") = txtQuantidade.Text
  objDR("Produto") = Produtos.SelectedItem.Text
  objDR("Custo") = Decimal.Parse(Produtos.SelectedItem.Value)
  objDT.Rows.Add(objDR)
End If

Session("Carrinho") = objDT

dg.DataSource = objDT
dg.DataBind()

lblTotal.Text = FormatCurrency(GetItemTotal(),2)

End Sub

```

O código acima é executado quando o usuário clica no botão - incluir no carrinho . O evento [onClick](#) chama o procedimento - [IncluirNoCarrinho](#).

```

objDT = Session("Carrinho")
Dim Produto = Produtos.SelectedItem.Text

```

As linhas acima do código obtém o carrinho da sessão , se ele existir , e retorna o produto selecionado da caixa de listagem - dropdownlist. Como já foi criada uma nova instância da classe DataRow no início do código, podemos criar novas linhas para incluir no objeto DataTable.

```

For Each objDR In objDT.Rows
  If objDR("Produto") = Produto Then
    objDR("Quantidade") += txtQuantidade.Text
    blnMatch = True
  Exit For
End If
Next

```

O código acima verifica se o produto selecionado já existe na tabela e incrementa a sua quantidade.

```

If Not blnMatch Then
  objDR = objDT.NewRow
  objDR("Quantidade") = txtQuantidade.Text
  objDR("Produto") = Produtos.SelectedItem.Text

```

```
objDR("Custo") = Decimal.Parse(Produtos.SelectedItem.Value)
objDT.Rows.Add(objDR)
End If
```

Neste código estamos incluindo a linha na tabela , a condição existe , pois somente podemos incluir um produto que ainda não exista na tabela.

A última linha de código : `lblTotal.Text = FormatCurrency(GetItemTotal(),2)` faz a totalização dos valores dos produtos chamando o procedimento - **GetItemTotal()** , cujo código é o seguinte :

```
Function GetItemTotal() As Decimal
Dim intCounter As Integer
Dim decTotal As Decimal

For intCounter = 0 To objDT.Rows.Count - 1
objDR = objDT.Rows(intCounter)
decTotal += (objDR("Custo") * objDR("Quantidade"))
Next

Return decTotal
End Function
```

Excluindo itens do carrinho de compras

Para encerrar vamos mostrar como excluir itens do carrinho de compras , assim o usuário poderá considerar e voltar atrás na compra de um produto. A rotina é a seguinte :

```
Sub Exclui_Item(s As Object, e As DataGridCommandEventArgs)
objDT = Session("Carrinho")
objDT.Rows(e.Item.ItemIndex).Delete()
Session("Carrinho") = objDT

dg.DataSource = objDT
dg.DataBind()

lblTotal.Text = FormatCurrency(GetItemTotal(),2)
End Sub
```

Estou usando o método **Delete** do objeto **Rows** e a seguir atribuindo o objeto a sessão definida. Em seguida vinculamos o objeto ao DataGrid e exibimos o valor total na label do formulário.

A linha de código definida no DataGrid que ativa a exclusão é a seguinte :

```
<asp:buttoncolumn buttontype="LinkButton" commandname="Delete" text="Remover Item" />
```

O código completo é dado a seguir :

```
<%@ Page debug=true Language="VB" ContentType="text/html" ResponseEncoding="iso-8859-1" %>
```

```
<html>
```

```
<head>
```

```
<title> Carrinho de Compras</title>
```

```
<script runat="server">
```

```
Dim objDT As System.Data.DataTable
```

```
Dim objDR As System.Data.DataRow
```

```
Private Sub Page_Load(s As Object, e As EventArgs)
```

```
    If Not IsPostBack Then
```

```
        CriaCarrinhoDeCompras()
```

```
    End If
```

```
End Sub
```

```
Function CriaCarrinhoDeCompras()
```

```
    objDT = New System.Data.DataTable("Carrinho")
```

```
    objDT.Columns.Add("Codigo", GetType(Integer))
```

```
    objDT.Columns("Codigo").AutoIncrement = True
```

```
    objDT.Columns("Codigo").AutoIncrementSeed = 1
```

```
    objDT.Columns.Add("Quantidade", GetType(Integer))
```

```
    objDT.Columns.Add("Produto", GetType(String))
```

```
    objDT.Columns.Add("Custo", GetType(Decimal))
```

```
    Session("Carrinho") = objDT
```

```
End Function
```

```
Sub IncluirNoCarrinho(s As Object, e As EventArgs)
```

```
    objDT = Session("Carrinho")
```

```
    Dim Produto = Produtos.SelectedItem.Text
```

```
    Dim blnMatch As Boolean = False
```

```
    For Each objDR In objDT.Rows
```

```
        If objDR("Produto") = Produto Then
```

```
            objDR("Quantidade") += txtQuantidade.Text
```

```
            blnMatch = True
```

```
            Exit For
```

```
        End If
```

```
    Next
```

```
    If Not blnMatch Then
```

```
        objDR = objDT.NewRow
```

```
        objDR("Quantidade") = txtQuantidade.Text
```

```
        objDR("Produto") = Produtos.SelectedItem.Text
```

```
        objDR("Custo") = Decimal.Parse(Produtos.SelectedItem.Value)
```

```
        objDT.Rows.Add(objDR)
```

```
    End If
```

```
    Session("Carrinho") = objDT
```

```
    dg.DataSource = objDT
```

```
    dg.DataBind()
```

```
    lblTotal.Text = FormatCurrency(GetItemTotal(), 2)
```

End Sub**Function GetItemTotal() As Decimal**

Dim intCounter As Integer

Dim decCalculaTotal As Decimal

For intCounter = 0 To objDT.Rows.Count - 1

objDR = objDT.Rows(intCounter)

decCalculaTotal += (objDR("Custo") * objDR("Quantidade"))

Next

Return decCalculaTotal

End Function**Sub Exclui_Item(s As Object, e As DataGridCommandEventArgs)**

objDT = Session("Carrinho")

objDT.Rows(e.Item.ItemIndex).Delete()

Session("Carrinho") = objDT

dg.DataSource = objDT

dg.DataBind()

lblTotal.Text = FormatCurrency(GetItemTotal(),2)

End Sub

</script>

</head>

<body>

<form runat="server">

Produto:

<asp:DropDownList id=Produtos runat="server">

<asp:ListItem Value="25,00">Super CD Visual Basic - 2003</asp:ListItem>

<asp:ListItem Value="15,00">Super CD Asp Total - 2003</asp:ListItem>

<asp:ListItem Value="100,00">Sistema - Clinicas Dentárias</asp:ListItem>

<asp:ListItem Value="60,00">Sistema - Escola Infantil</asp:ListItem>

<asp:ListItem Value="70,00">Sistema - Videolocadora</asp:ListItem>

</asp:DropDownList>

Quantidade:

<asp:textbox id="txtQuantidade" runat="server" />

<asp:Button id=btnIncluir runat="server" Text="Incluir no Carrinho" onClick="IncluirNoCarrinho" />

<asp:DataGrid id=dg runat="server" ondeletecommand="Exclui_Item">

<columns>

<asp:buttoncolumn buttontype="LinkButton" commandname="Delete" text="Remove Item" />

</columns>

</asp:DataGrid>

Total:

<asp:Label id=lblTotal runat="server" />

</form>

</body>

</html>

Agora é só você acessar o carrinho de compras no link a seguir e testar...: [CarrinhoCompras.](#) 😊

Produto:

Quantidade:

	Codigo	Quantidade	Produto	Custo
Remover Item	1	10	Super CD Asp Total - 2003	15
Remover Item	2	15	Super CD Visual Basic - 2003	25

Total: R\$ 525,00

Enviando Emails

O envio de e-mail é feito por intermédio do - **SMTP** - (*Simple Mail Transfer Protocol*) , porta 25 . Você pode configurar o serviço SMTP no seu **IIS** - *Internet Information Server* - , ou se desejar , pode usar um componente de terceiros ; dê uma olhada no link - freesmtp.net. Pode usar também o componente **CDONTS** (*Collaboration Data Objects for Windows NT Server*) ou um servidor SMTP válido.

Para enviar emails no ASP.NET você tem que importar a classe que contém os métodos e propriedades que fazem este serviço. Estou falando da classe : [System.Web.Mail](#)

Depois é só você configurar os métodos e propriedades do objeto que você vai instanciar a partir da classe. As principais são :

Métodos/Propriedades	Descrição
Body	Corpo do e-mail
BodyFormat	formato do e-mail
Cc	Enviar uma cópia para
Attachments	Anexar arquivos
From	origem do e-mail
Subject	Assunto
To	destino do e-mail
Priority	nível de prioridade
SMTPServer	o domínio do servidor SMTP usado

Vou mostrar primeiro um exemplo básico e padrão : um formulário que depois de preenchido envia um email para o endereço informado. O aspecto do formulário deverá ser o seguinte:

Endereço http://localhost/Email/email3.aspx

IG mail morango Grandes Negócios IG Shopping

Preencha o formulário

Nome :

Endereco:

Email :

Enviar email

O código do formulário acima é o seguinte :

```
<%@ Page Language="vb" AutoEventWireup="false"%>
<%@Import Namespace="System.Web.Mail" %>

<script runat="server">
Sub EnviaEmail(Source as Object, E as EventArgs)
Dim sMsg as String

sMsg+="Aqui esta a informação que foi informada no formulário." & vbcrLf
sMsg+="Nome : " & txtNome.Text & vbcrLf
sMsg+="Endereco : " & txtEndereco.Text & vbcrLf

Dim objEmail as New MailMessage

objEmail.To=txtEmail.text
objEmail.BCC="macoratti@yahoo.com"
objEmail.FROM="macoratti@pop.com.br"
objEmail.SUBJECT="Assunto :"
objEmail.BODY=sMsg
objEmail.BodyFormat = MailFormat.Text
SmtMail.SmtpServer = "smtp2.seuservidor.com.br"

SmtMail.Send(objEmail)
End Sub
</script>

<html>
<body bgcolor="aqua">
<h1>Preencha o formulário</h1>
```

```

<Form id= "form1" runat= "server">
<table>
<tr>
<td align= "right">Nome : </td>
<td><asp:TextBox id= "txtNome" maxlength= "40" Text= "" runat= "server" />
<asp:RequiredFieldValidator id= "nomeValido" ControlToValidate= "txtNome" Display= "Static" InitialValue= ""
runat= "server"> Informe um nome ! </asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td align= "right">Endereco: </td>
<td><asp:TextBox id= "txtEndereco" maxlength= "40" runat= "server" />
<asp:RequiredFieldValidator id= "EnderecoValido" ControlToValidate= "txtEndereco" Display= "Static" InitialValue= ""
runat= "server"> Informe um endereco ! </asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td align= "right">Email : </td>
<td><asp:TextBox id= "txtEmail" runat= "server" />
<asp:RequiredFieldValidator id= "EmailValido" ControlToValidate= "txtEmail" Display= "Static" InitialValue= ""
runat= "server"> Informe um Email ! </asp:RequiredFieldValidator>
</td>
</tr>
<td align= "right"><asp:Button id= "btnEmail" Text= "Enviar email" onclick= "EnviaEmail" runat= "server" /></td>
</table>
</Form>
</body>
</html>

```

Neste exemplo eu estou enviando apenas um email sem anexos, sendo que a mensagem já esta criada. Poderia ter incluído uma caixa de texto que permitisse ao usuário digitar a mensagem.

Eu estou usando o controle de validação - **RequiredFieldValidator** - que obriga o preenchimento de todos os campos do formulário. Para maiores detalhes sobre o assunto leia o artigo : [Trabalhando com Controles e Web Forms - II](#).

Eu estou usando o serviço do meu servidor SMTP , na linha de código : `SmtpMail.SmtpServer = "smtp2.seuservidor.com.br"` , você deve alterar para o nome do seu servidor.

Nota: A classe [System.Web.Mail](#) não suporta autenticação segura do servidor SMTP. Portanto se o seu servidor SMTP requerer autenticação o exemplo acima não vai funcionar. Você vai ter que usar um componente de terceiros. (Veja o site www.asppemail.com)

Enviando Emails : HTML e anexos

O envio de e-mail é feito por intermédio do - **SMTP** - (*Simple Mail Transfer Protocol*) , porta 25 . Você pode configurar o serviço SMTP no seu **IIS** - *Internet Information Server* - , ou se desejar , pode usar um componente de terceiros ; dê uma olhada no link - freesmtp.net. Pode usar também o componente **CDONTS** (*Collaboration Data Objects for Windows NT Server*) ou um servidor SMTP válido.

Para enviar emails no ASP.NET você tem que importar a classe que contém os métodos e propriedades que fazem este serviço. Estou falando da classe : [System.Web.Mail](#)

Nota: A classe [System.Web.Mail](#) não suporta autenticação segura do servidor SMTP. Portanto se o seu servidor SMTP requerer autenticação o exemplo acima não vai funcionar. Você vai ter que usar um componente de terceiros. (Veja o site www.aspemail.com) . ☹️

Para lembrar abaixo temos os principais métodos e propriedades da classe que iremos usar:

Métodos/Propriedades	Descrição
Body	Corpo do e-mail
BodyFormat	formato do e-mail
Cc	Enviar uma cópia para
Attachments	Anexar arquivos
From	origem do e-mail
Subject	Assunto
To	destino do e-mail
Priority	nível de prioridade
SMTPServer	o domínio do servidor SMTP usado

Enviando um Email simples

No artigo - [Enviando Emails](#) - eu mostrei a forma mais simples de enviar um email , neste artigo iremos ver como enviar um email no formato HTML e com anexos. Abaixo temos um novo exemplo de envio de email simples usando o servidor SMTP do IIS.

<pre><% @Page Language="VB" %> <% @Import Namespace="System.Web.Mail" %> <% Dim strPara As String Dim strDe As String Dim strTexto As String strPara = "macoratti@yahoo.com" strDe = "webmaster@visualbasic.mat.br" strTexto = "Ola , Macoratti" SmtpMail.Send(strDe, strPara, strTexto,"Esta eh uma mensagem de teste de envio de email") Response.Write("Email foi enviado !!!") %></pre>	<p>- Este simples script é tudo o que você precisa para enviar um email usando o Serviço SMTP do IIS.</p> <p>- Estou usando o namespace : "System.Web.Mail" onde encontramos a classe <code>SmtpMail</code> , cujo método estático <code>Send</code> pode aceitar até quatro parâmetros:</p> <p><code>SmtpMail.Send(From, To, Subject, BodyText);</code></p>
---	--

Enviando Email no formato HTML

Vamos então avançar um pouco. A classe **MailMessage** permite a definição do formato na método - **BodyFormat** - . A variável string `strBody` contém as tags HTML com a mensagem a ser enviada.

```

<%@ Page Language="VB" clienttarget=uplevel %>
<% @Import Namespace="System.Web.Mail" %>
<%

Dim strBody As String

Dim msgMail As new MailMessage()

msgMail.To = "macoratti@yahoo.com"
msgMail.Cc = "webmaster@visuabasic.mat.br"
msgMail.From = "webmaster@visuabasic.mat.br"
msgMail.Subject = "Olá, Macoratti , ai vai outro email..."

msgMail.BodyFormat = MailFormat.Html;

strBody = "<html><body><b>Olá, Pessoal</b> , <font color=blue> uma abraço do Macoratti</font></body></html>"

msgMail.Body = strBody

SmtpMail.Send(msgMail)

Response.Write("Email enviado no formato HTML !")
%>

```

Nota: Estamos usando o método **Send** sobrecarregado que não esta retornando valores após o envio da mensagem de email. Como estamos usando o serviço [SMTP do IIS os emails](#) que falharem irão ser enviados para a pasta [BadMail](#).

Enviando Email com anexos

```

<%@ Page Language="VB" clienttarget=uplevel %>
<% @Import Namespace="System.Web.Mail" %>
<%

Dim strBody As String

Dim msgMail As new MailMessage()

msgMail.To = "macoratti@yahoo.com"
msgMail.Cc = "webmaster@visuabasic.mat.br"
msgMail.From = "webmaster@visuabasic.mat.br"
msgMail.Subject = "Olá, Macoratti , ai vai outro email..."

msgMail.BodyFormat = MailFormat.Text;

strBody = "Este é mais um teste de envio de email com anexo usando o STMP do IIS"

msgMail.Body = strBody

msgMail.Attachments.Add(New MailAttachment("d:\teste\teste.txt"))

SmtpMail.Send(msgMail)

Response.Write("Email enviado com anexo !")
%>

```

A unica diferença aqui é linha :

```
msgMail.Attachments.Add(New MailAttachment("d:\teste\teste.txt"))
```

que podemos também usar da seguinte forma:

```
MailAttachment mAnexo = new MailAttachment("c:\teste\teste.pdf");
IList msgAttachments = msgMail.Attachments;
msgAttachments.Add(mAnexo);
```

Aqui os anexos são tratados através da interface **IList** (encontrada no namespace *System.Collections*) que fornece um método **Add** com o qual incluímos o anexo.

Exibindo dados de uma planilha Excel em uma página ASP.NET

Já tratei em diversos artigos da interação VB com o Excel. Veja a relação de links abaixo. Agora vamos ver como fazer isto usando código VB.NET.

1. [Excel & Visual Basic - Importando e Exportando dados](#)
2. [OLE - Conceitos.](#)
3. [Excel - Abrindo uma Planilha e Exportando p/ uma Tabela](#)
4. [Gerando gráficos no Excel via automação OLE no Visual Basic](#)
5. [SQL - Gravando os dados de uma planilha Excel em uma tabela de um banco de dados Access](#)

Neste artigo eu vou mostrar como exibir dados de uma planilha Excel em uma página ASP.NET usando código VB.NET. Você vai precisar ter o Excel e o VB.NET instalados.

Vamos criar uma planilha Excel para usarmos neste artigo :

1. Abra o Excel e crie uma planilha com os dados ao lado.
2. A seguir selecione as linhas e colunas com os dados e no Menu - **Inserir** , selecione a opção **Nome** e a seguir **Definir**. (fig 1.0)
3. Na janela - **Definir Nome** - informe o nome : **TesteXLS** , na caixa de texto - Nomes da pasta de Trabalho. (fig 2.0)
4. A seguir no Menu - **Arquivo** - opção - **Salvar Como** informe o nome da planilha - **VBExcel.xls** *no seu diretório de trabalho definido no IIS . (O caminho padrão é /inetpub/wwwroot)*

	A	B
1	Nome	Endereco
2	Macoratti	Av. Sol , 100
3	Jessica	R. Girasssol , 123
4	Jefferson	Trav. Particular , 09
5	Janice	Av. Lua , 200
6		

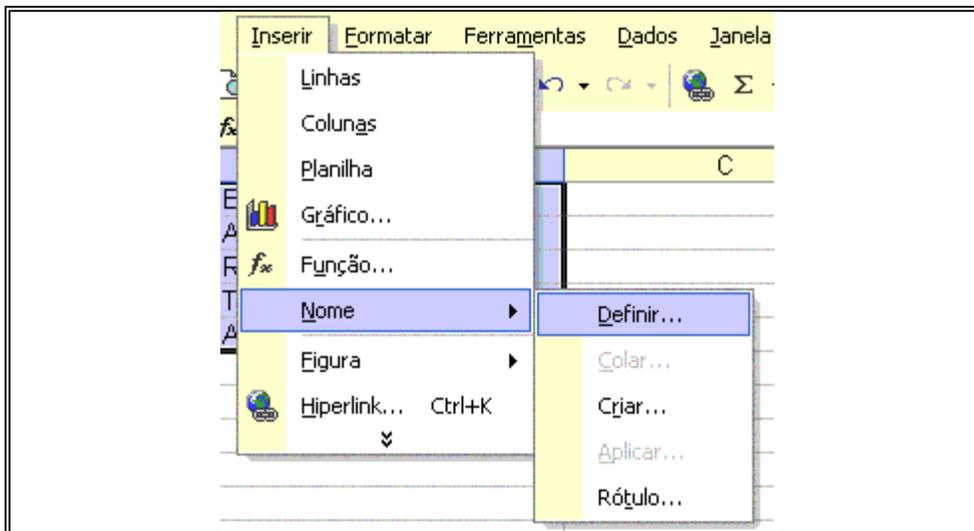


Fig 1.0

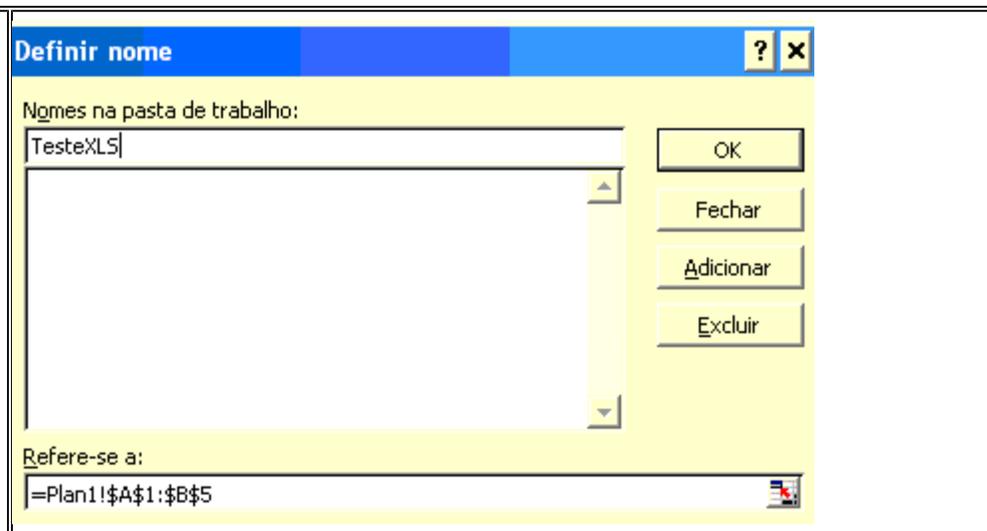


Fig 2.0

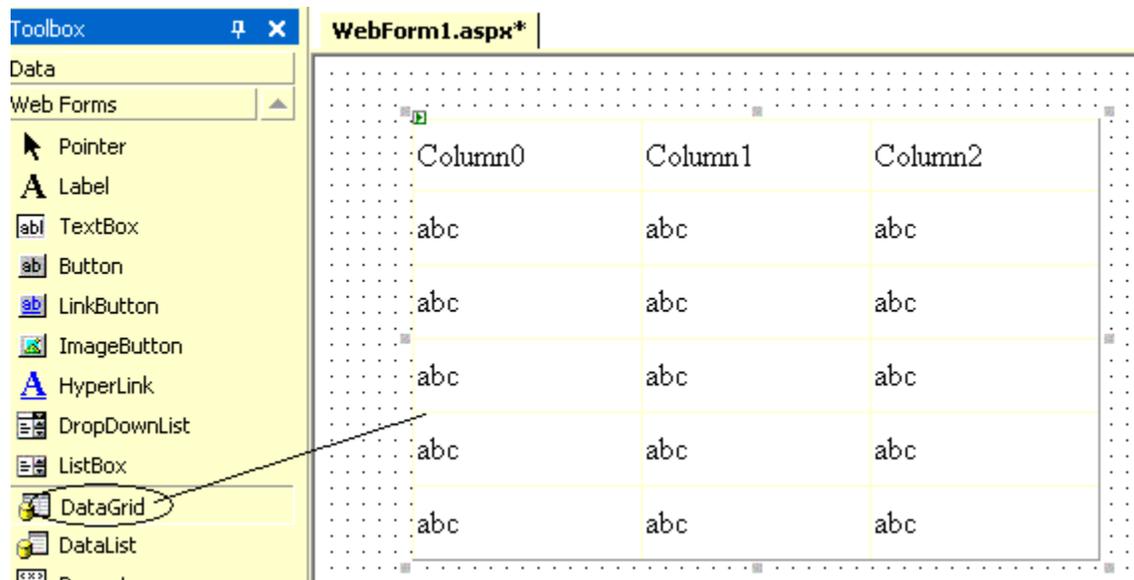
Vamos agora criar o projeto no Visual Studio .NET :

1 - Inicie um novo projeto no Visual Studio.NET com as seguintes características (*sinta-se a vontade para alterar a seu gosto.*)

1. Project Types : **Visual Basic Projects**
2. Templates : **ASP.NET Web Application**
3. Name : **ExcelVBnet**
4. Location : **<http://localhost/ExcelVBnet>**

2- Vamos importar o namespace : **Imports** System.Data.OleDb

3- Agora vamos incluir um componente DataGrid no formulário - **Webform1.aspx** (*Se você não esteve vendo o formulário clique com o botão direito do mouse no arquivo - **Webform1.aspx** - no **Solution Explorer** e selecione - **View Designer***)



4- Vamos agora inserir o código no evento [Page_Load](#) do formulário , para isto clique duas vezes no formulário ou no **Solution Explorer** clique com o botão direito do mouse sobre **WebForm1.aspx** e selecione - [View Code](#).

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    ' Cria variáveis que iremos usar no projeto
    Dim i, j As Integer

    ' Cria uma string de conexão com o planilha Excel
    Dim sConnectionString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & Server.MapPath("../VBExcel.xls") _
    & ";" & "Extended Properties=Excel 8.0;"

    ' Cria o objeto connection usando a string de conexão
    Dim objConn As New OleDbConnection(sConnectionString)
    ' abre a conexão com a fonte de dados
    objConn.Open()

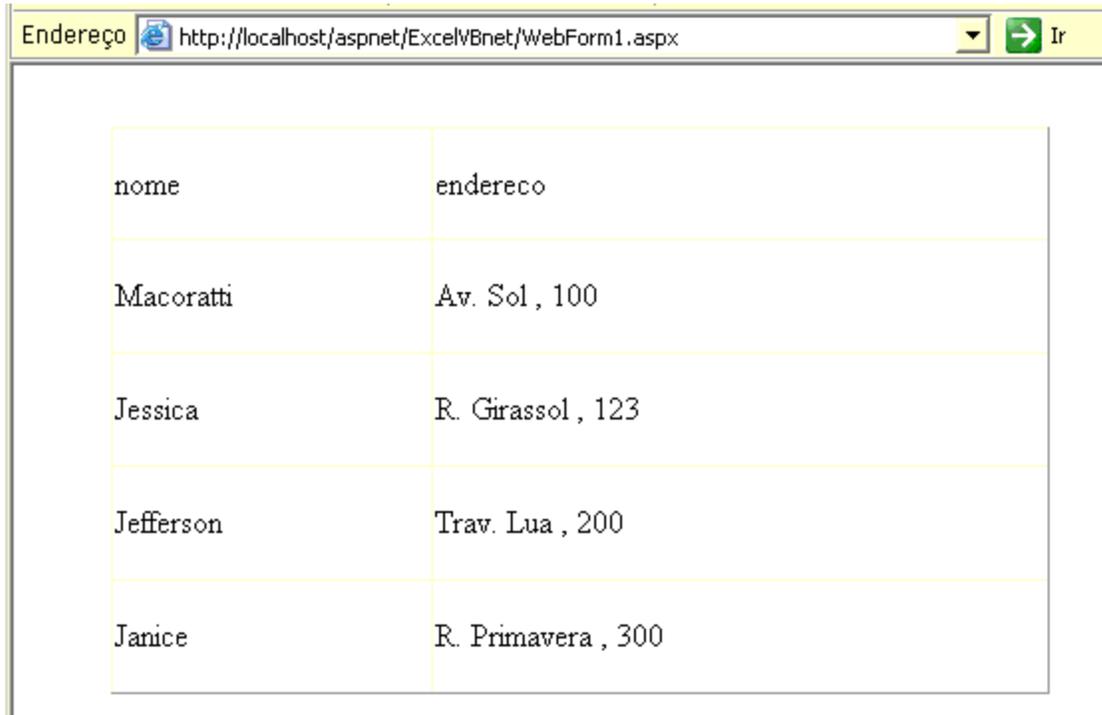
    ' Cria um novo OleDbCommand que retorna os dados da planilha
    Dim objCmdSelect As New OleDbCommand("SELECT * FROM TesteXLS", objConn)
    ' Cria um novo OleDbDataAdapter que é usado para construir o DataSet baseado na instrução SQL
    Dim objAdapter1 As New OleDbDataAdapter()
    ' Passa o comando Select para o adapter.
    objAdapter1.SelectCommand = objCmdSelect
    ' Cria um novo DataSet para manipular a informação da planilha
    Dim objDataset1 As New DataSet()
    ' Preenche o DataSet com as informações da planilha

    objAdapter1.Fill(objDataset1, "XLData")
    ' Constroi uma tabela a partir dos dados originais
    DataGrid1.DataSource = objDataset1.Tables(0).DefaultView
    DataGrid1.DataBind()
End Sub
```

```
' Fecha e libera os objetos
objConn.Close()
```

End Sub

5- Vamos salvar o projeto na opção : **File | Save All** e a seguir vamos construir o projeto na opção : **Build | Build ExcelVBnet**. Já estamos prontos para visualizar a página com os dados da planilha. No **Solution Explorer** clique com o botão direito do mouse sobre **WebForm1.aspx** e selecione **View in Browser**.



nome	endereço
Macoratti	Av. Sol , 100
Jessica	R. Girassol , 123
Jefferson	Trav. Lua , 200
Janice	R. Primavera , 300

Acima vemos os dados da planilha exibidos na página ASP.NET.

Web Services , tendência ou moda ?

Não olhe os web services , serviços via web , apenas como uma tecnologia do momento. Se você analisar todo o histórico do desenvolvimento de sistemas no percurso da redução de custos , aumento de produtividade e melhoria de desempenho e qualidade , vai chegar á conclusão que os web services vieram para virar a página desta história. Portanto aqui vai uma sugestão : [procure compreender e dominar os web services...](#)

Os web services não são uma tecnologia proprietária da Microsoft nem de outra empresa , são uma necessidade que vem amadurecendo já há algum tempo e que parece que esta na hora de ser colhida. Os web services querem dizer - automação e integração de aplicativos , e a interligação dos softwares de apoio aos negócios implica em : *redução de erros , redução de estoques , aumento de receita e diminuição de quadro de pessoal* , afinal deverá haver uma redução no quadro de desenvolvedores e gerentes... 😊

Os web services estão apoiados em [XML e no protocolo SOAP](#) ; o XML descreve e define os web services , e , como os web services são usados para disponibilizar serviços interativos na Internet , devem ser acessados por qualquer aplicação usando um protocolo universal ; aqui é que entra o **SOAP (Simple Object Access Protocol)** ; **SOAP** é baseado em HTTP e XML.

Existem três maneiras diferentes de se comunicar com um Web Service : **GET , POST e SOAP**. É o método SOAP que a plataforma .NET usa de forma transparente para acessar web services ; SOAP é um padrão baseado em XML e já existia antes da plataforma .NET ser criada.

Algumas das características do protocolo *SOAP* :

1. Definido pelo consórcio W3C
2. Baseado em XML para intercâmbio de informações
3. Padrão utilizado para acessar Web Services
4. Utiliza HTTP como protocolo de transporte
5. É constituído pelos seguintes elementos :
 - a. **Envelope** - Elemento raiz do documento XML
 - b. **Header** - Cabeçalho opcional que define um namespace para o elemento.
 - c. **Body** - Elemento obrigatório com a informação a ser transportada para o destino.

Como o **SOAP** trafega sobre **HTTP** ele utiliza **POST e GET** usando a porta **80** , a base de funcionamento da web ; sua base é XML , o que quer dizer texto padronizado (dado com metadado) ; estas características permitem a comunicação entre as mais diferentes plataformas pois é comum a quase todas elas .Nestas condições uma requisição SOAP feita por um cliente Windows é perfeitamente compreendida e aceita por um Servidor Solaris.

A linguagem XML(*uma ferramenta para descrever dados*) é ideal para a Web ; mas para descrever os Web Services usamos a linguagem **WSDL (web Service Description Language)** . Todo web service tem o seu WSDL . O WSDL é um documento em XML que descreve os protocolos que podem ser utilizados para acessar o web service. No WSDL estão definidos : *a URL de acesso , o nome do web service , a descrição de cada método e como fazer a solicitação via SOAP , GET ou POST* . Podemos fazer a requisição WSDL via Web .Ex: <http://localhost/webserice/service.asmx?WSDL>

Nota: Um documento WSDL define um schema XML para descrever um web service.

Definir web services também não é tão simples assim , por trás dos web Services existe uma infra-estrutura organizada para fazer tudo isto funcionar de forma adequada. A estrutura básica é a seguinte :

- **Web Services Directories** - um diretório central onde estarão localizados os web services oferecidos pelas empresas. Ex: www.uddi.org
- **Web Services Discovery** - Um mecanismo para localizar web services. Utiliza a linguagem WSDL . O Web Service possui uma especificação chamada DISCO (*DISCO*very).
- **Web Services Description** - Provê uma descrição de serviços que define as interações que o web service suporta.
- **Web Service Wire Formats** - Define o protocolo para comunicação com os web services.

Resumindo , um web service é uma programa transformado em um serviço que deverá ser cadastrado e disponibilizado num formato eletrônico padronizado. Se o cadastro for feito no padrão **UDDI (Conjunto de registros e diretório de busca de web services)** se as informações foram escritas no formato XML se a descrição do serviço for feito no padrão WSDL e se as trocas de mensagens forem feitas usando o protocolo SOAP temos ai um Web Service.

A Microsoft ao lançar o .NET esta entrando na briga para oferecer uma plataforma para criar e chamar Web Services , e, eu acho que eles fizeram um bom trabalho. Mas podemos chamar os Web Services de outras plataformas e sistemas operacionais , por exemplo , podemos usar o Windows para chamar web services , basta apenas instalar o SOAP Toolkit em <http://msdn.microsoft.com/soap> (Win 98/ME/2000/XP e NT).

Vamos dar uma olhada em alguns web services que já estão disponíveis . Acesse o link : <http://www.xmethods.com> , na seção : **Xmethods Demo Services** temos uma relação de web services ativos :

Endereço <http://www.xmethods.com/>

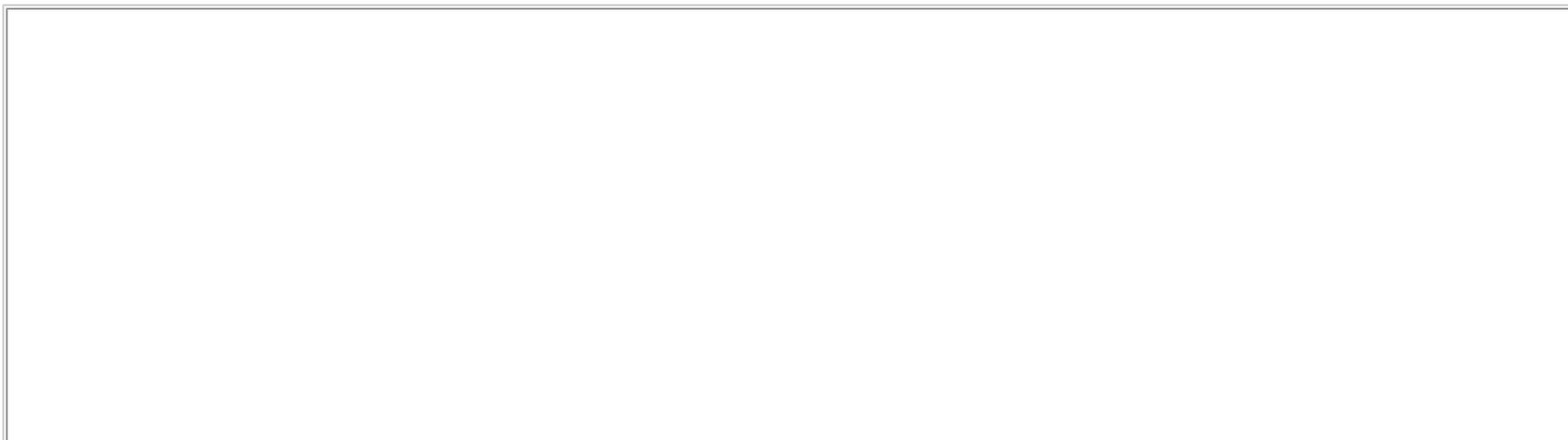
walterjones	DOC	AustralianPostCode	Australian Postcode ,Location Web service	MS .NET
rpamplona	RPC	WebChex ACH SOAP (T\$\$ - R Pamplona)	Web Services for Electronic Check Payment and Tracking through the ACH	Delphi

Click [here](#) to see the full list.

XMethods Demo Services

Style	Service Name	Description	Implementation
RPC	XMethods Query Service	Provides a SOAP RPC interface to XMethods for query operations	GLUE
RPC	FedEx Tracker	Access to FedEx Tracking information	SOAPLite
RPC	BabelFish	Interface for AltaVista's Babelfish service.	SOAPLite
RPC	XMethods Filesystem	Virtual Filesystem service with 1MB quota.	Apache SOAP
RPC	Weather - Temperature	Current temperature in a given U.S. zipcode region.	Apache SOAP
RPC	Barnes and Noble Price Quote	Returns price of a book at BN.com given an ISBN number.	Apache SOAP
RPC	Domain Name Checker	Checks whether a domain name is available or not.	GLUE
RPC	Currency Exchange Rate	Exchange rate between any two currencies.	GLUE
RPC	California Traffic Conditions	California highway conditions.	Apache SOAP
RPC	Delayed Stock Quote	20 minute delayed stock quote	GLUE
RPC	eBay Price Watcher	Checks current bid price of an eBay auction	Apache SOAP

Escolha um link e veja a descrição do web service :



Endereço [http://www.xmethods.com/ve2/ViewListing.poj?sessionId=Hn02-Js0kybAG4f9F-zN4pzf\(QhxieSRM\)?key=uuid:D784C184-99B2-DA25-ED45-3665D11A12E5](http://www.xmethods.com/ve2/ViewListing.poj?sessionId=Hn02-Js0kybAG4f9F-zN4pzf(QhxieSRM)?key=uuid:D784C184-99B2-DA25-ED45-3665D11A12E5)[Home](#) · [Interfaces](#) · [Tools](#) · [Implementations](#) · [Manage](#) · [Register](#) · [Tutorials](#) · [Mailing List](#) · [About](#)

Currency Exchange Rate

[Help](#) [Message Board](#)

Click on the "View RPC Profile" link below to quickly see the interface's methods, parameters, associated SOAPAction, method Namespace URI, and endpoint URL. For more information on the RPC Profiler, click [here](#). *This function is only applicable for RPC-style interfaces.*

WSDL	http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl Analyze WSDL View RPC Profile
-------------	---

SOAP Binding	CurrencyExchangeBinding
---------------------	-------------------------

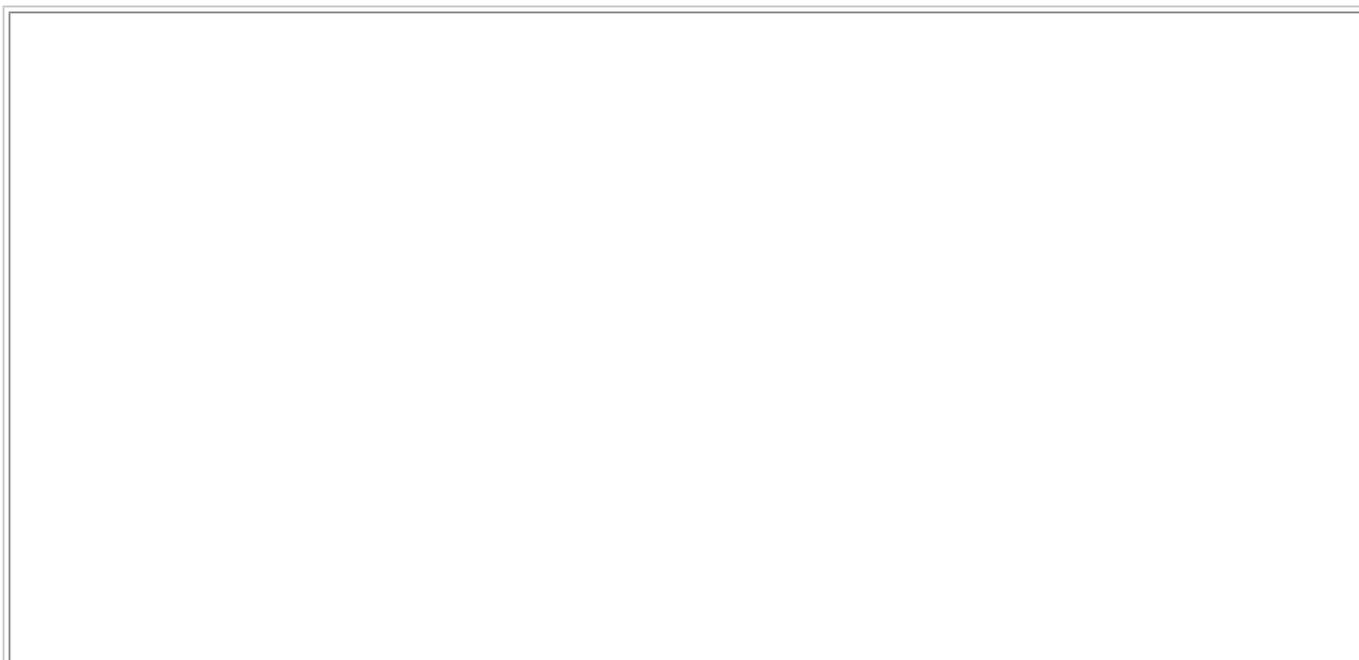
Key	uuid:D784C184-99B2-DA25-ED45-3665D11A12E5
------------	---

Owner:	xmethods.net
---------------	------------------------------

For more Info:

Description:	Exchange rate between any two currencies.
---------------------	---

Observe que na descrição do serviço temos o endereço do arquivo WSDL que descreve o Web Service . Veja na figura abaixo:



Endereço  <http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl>

```
<?xml version="1.0" ?>
- <definitions name="CurrencyExchangeService" targetNamespace="http://www.xmethods.net
  xmlns:tns="http://www.xmethods.net/sd/CurrencyExchangeService.wsdl" xmlns:xsd="ht
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns="http://schemas.xmlsoap
- <message name="getRateRequest">
  <part name="country1" type="xsd:string" />
  <part name="country2" type="xsd:string" />
</message>
- <message name="getRateResponse">
  <part name="Result" type="xsd:float" />
</message>
- <portType name="CurrencyExchangePortType">
  - <operation name="getRate">
    <input message="tns:getRateRequest" />
    <output message="tns:getRateResponse" />
  </operation>
</portType>
- <binding name="CurrencyExchangeBinding" type="tns:CurrencyExchangePortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  - <operation name="getRate">
    <soap:operation soapAction="" />
    - <input>
      <soap:body use="encoded" namespace="urn:xmethods-CurrencyExchange"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    - <output>
      <soap:body use="encoded" namespace="urn:xmethods-CurrencyExchange"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
```

José Carlos Macoratti - www.macoratti.net