

Apostila de ASP.NET e C# (VISUAL STUDIO 2008)

Prof. Eduardo Rosalém Marcelino (2009/2010)

Criando um cadastro utilizando FormView.....	4
Criando a tabela no BD.....	4
Criando um novo WebSite	5
Configurando o GridView:	6
Configurando o FormView	12
Codificando.....	16
Erro com tipo de dado DateTime.....	19
Configurando o FormView	21
Modo de consulta: (Item Template)	22
Modo de inclusão (InsertItemTemplate):	24
Modo de alteração (EditItemTemplate):.....	25
Configurando o GridView	26
Exibindo um valor personalizado na em uma coluna do Gridview.....	27
Colocando uma mensagem de aviso no linkbutton de excluir do GridView. Substituindo os linkbuttons por botões.....	29
CSS	31
Validadores.....	35
Adicionando a tecnologia AJAX em um site existente	37
Master Page e Formulários já habilitados para AJAX.....	39
Web User Control.....	49
Criando um método no Web User Control	51
Sessão	52
Introdução	52

Modos de estado de sessão	52
Definindo o tempo de vida da sessão.	53
ViewState	54
Request e Response	54
Tela de Login.....	55
Exibindo mensagens com o ScriptManager	56
Cookies	57
Passagem de parâmetro entre telas - Método Get (QueryString) e Método Post	59
Método GET (QueryString).....	59
Método Post	61
Criando uma classe de métodos estáticos	62
Tecnologias de conexão com BD ASP.NET	65
Criando um cadastro sem Formview.....	66
Utilizando o componente MultiView para organizar a tela	84
Criando uma Tela de pedido	91
Mapeando uma tabela utilizando propriedades no lugar de métodos “Get” e “Set”	99
Gravando um cadastro utilizando uma Stored Procedure.....	100
Configurando o Web User Control	106
Alterações na tela de cadastro de funcionários.....	113
Manipulando arquivos Texto.	117
Fazendo upload de arquivos	119
Listas Tipadas com Generics.....	121
Página de Erros Personalizada	126
Erro personalizado através do método OnError	127
Erro personalizado através de página de erro no Web.Config	128

Envio de emails.....	128
Adicionando uma biblioteca Javascript ao seu WebSite.....	130
Publicar o WebSite com IIS.....	133
Web Services	138
O que são web services?	138
Para que servem os WebServices?.....	138
Vantagens.....	138
Segurança:.....	138
O que é DWSL?	139
Principais características	139
Consumindo um Web Service	139
Criando um Método para somar 2 valores	142
Consumindo o Web Service	143
Web Service para realizar a inclusão.....	147
Criando o web service:	147
Consumindo o web service:	149
Listas tipadas em Web Services.....	150
Criando o Web service (continuando o exemplo do web service anterior...):.....	150
Consumindo o Web Service:	151

Criando um cadastro utilizando FormView

Criando a tabela no BD

Crie no SQL Server a seguinte tabela:

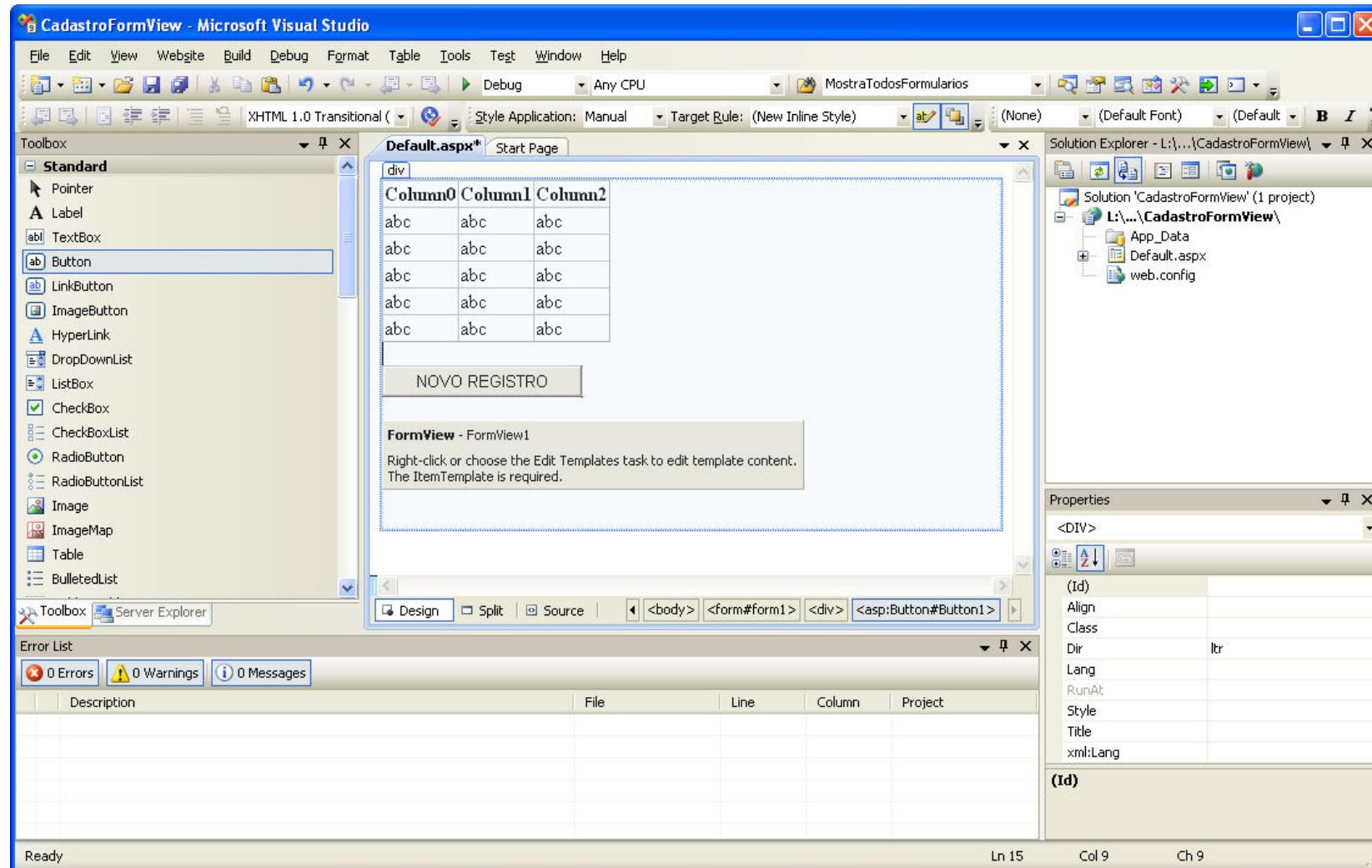
Tabela funcionários:

```
CREATE TABLE [dbo].[FUNCIONARIOS] (
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [NOME] [varchar](50) COLLATE Latin1_General_CI_AI NULL,
    [SEXO] [char](1) COLLATE Latin1_General_CI_AI NULL,
    [NASCIMENTO] [datetime] NULL,
    CONSTRAINT [PK_FUNCIONARIOS] PRIMARY KEY CLUSTERED
    (
        [ID] ASC
    ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Criando um novo WebSite

Crie um novo projeto no menu FILE -> NEW -> WEB SITE.

Monte uma tela com os componentes da figura abaixo:



Configurando o GridView:

The screenshot shows a context menu titled "GridView Tasks" for a GridView control. The "Choose Data Source:" dropdown is set to "(None)". The "New data source..." option is highlighted with a blue selection bar and an arrow pointing to it from the main text area below.

The screenshot shows the "Choose a Data Source Type" step of the Data Source Configuration Wizard. The "Database" option is selected and highlighted with a blue selection bar. A large arrow points from the "New data source..." option in the GridView Tasks menu down to the "Database" button in this wizard. Another arrow points from the "SqlDataSourceListaFuncionarios" text input field in the wizard back up towards the "GridView Tasks" menu.

Data Source Configuration Wizard

Choose a Data Source Type

Where will the application get data from?

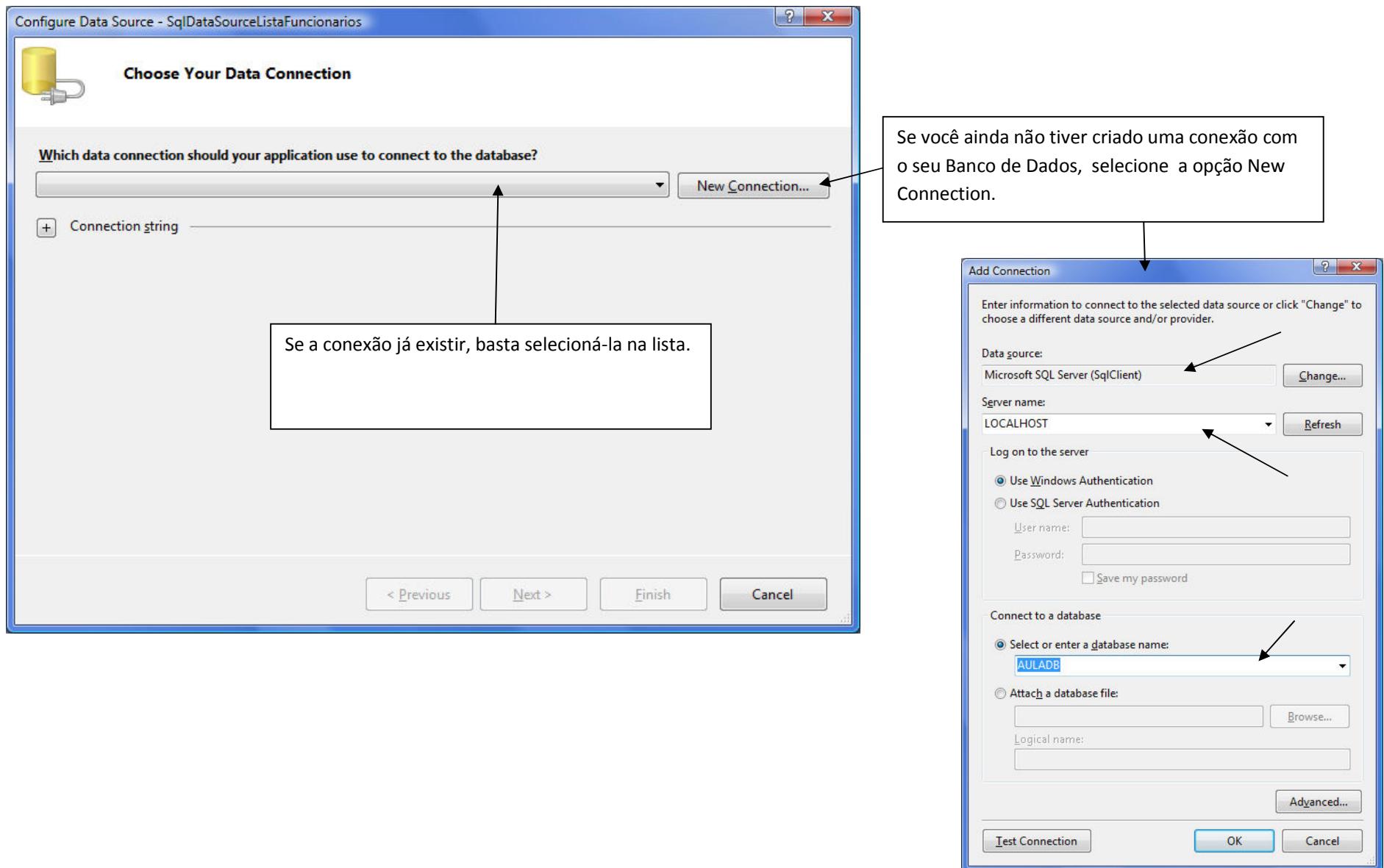
Access Database Database Entity LINQ Object Site Map XML File

Connect to any SQL database supported by ADO.NET, such as Microsoft SQL Server, Oracle, or OLEDB.

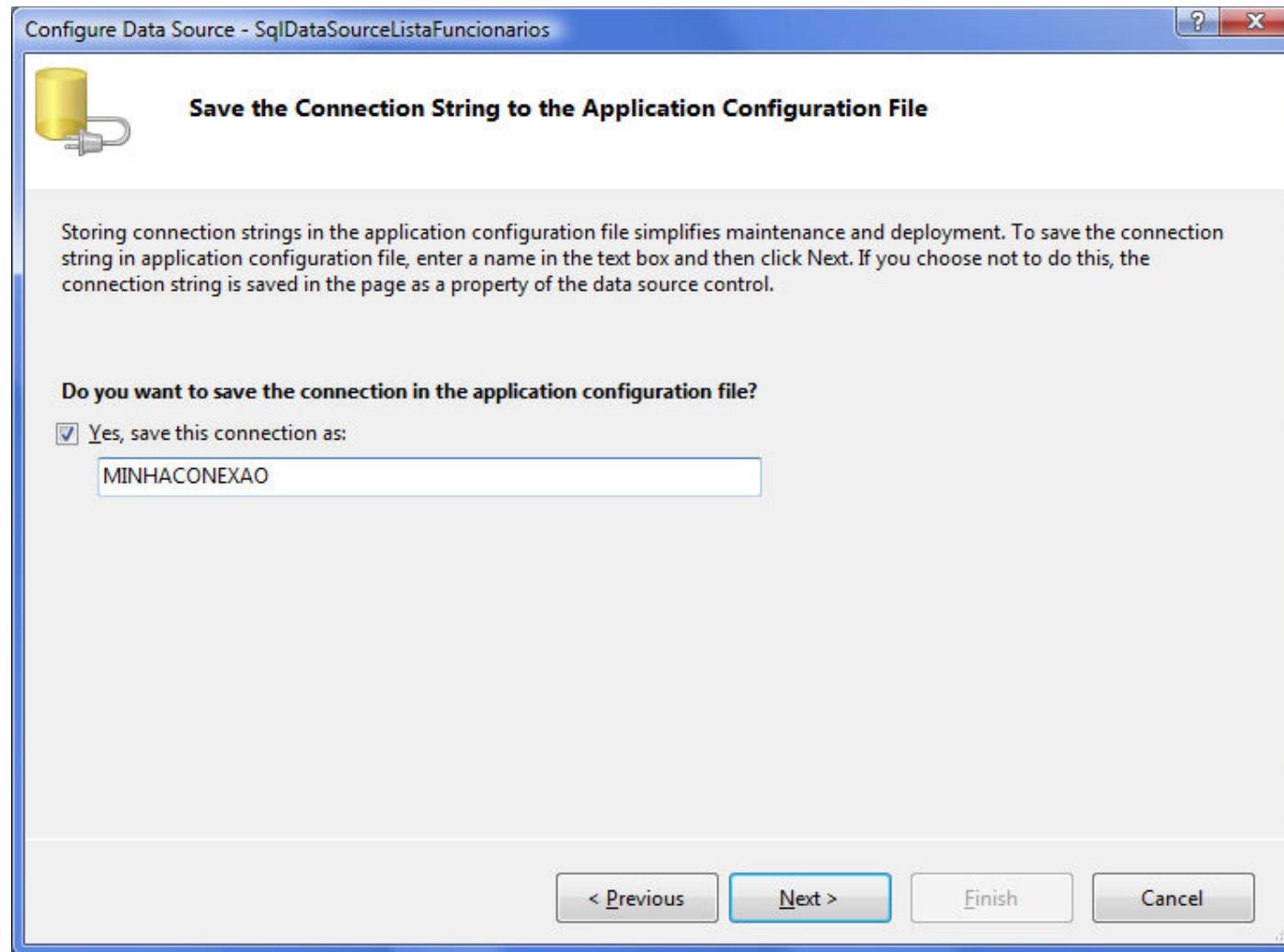
Specify an ID for the data source:

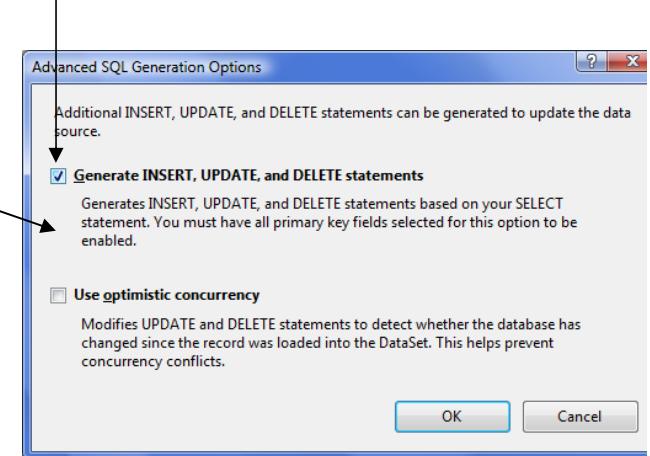
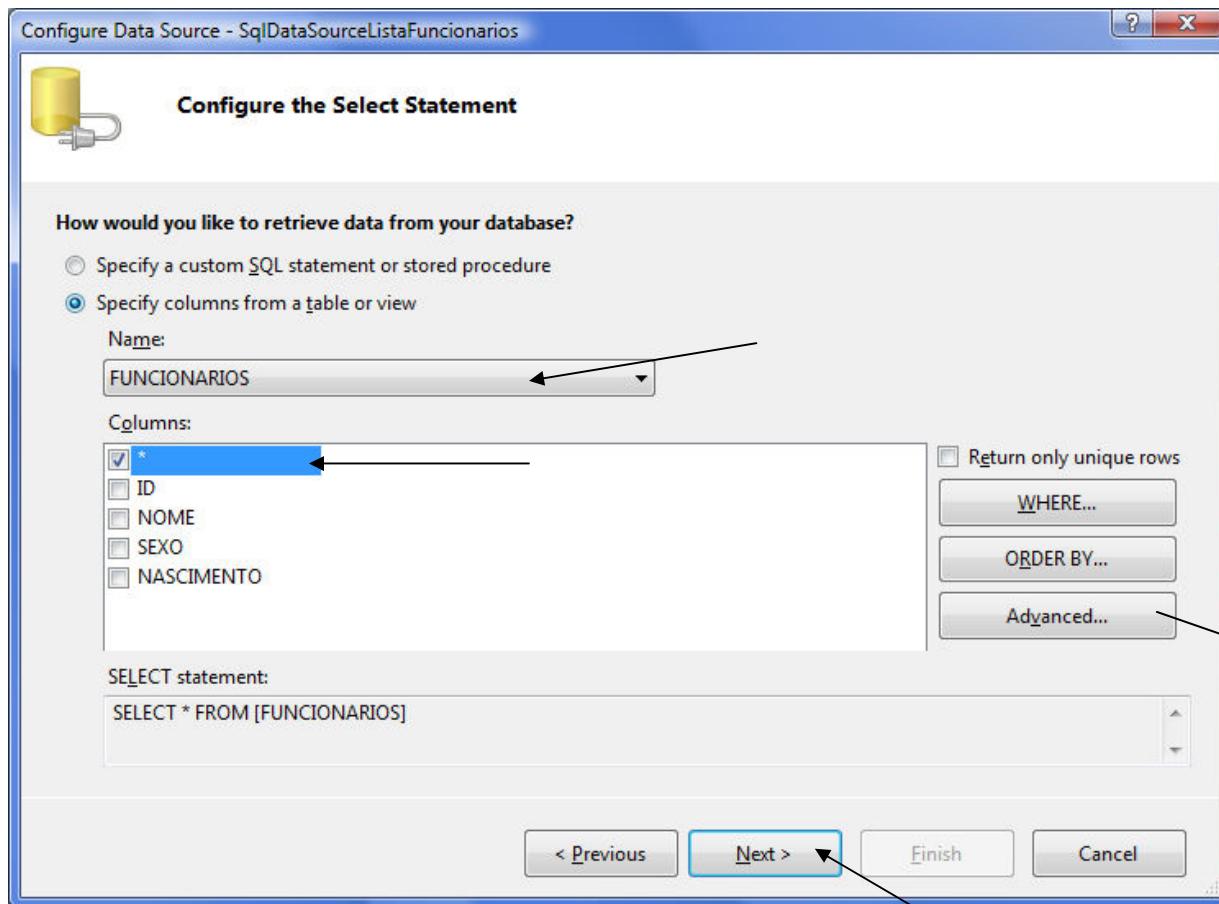
SqlDataSourceListaFuncionarios

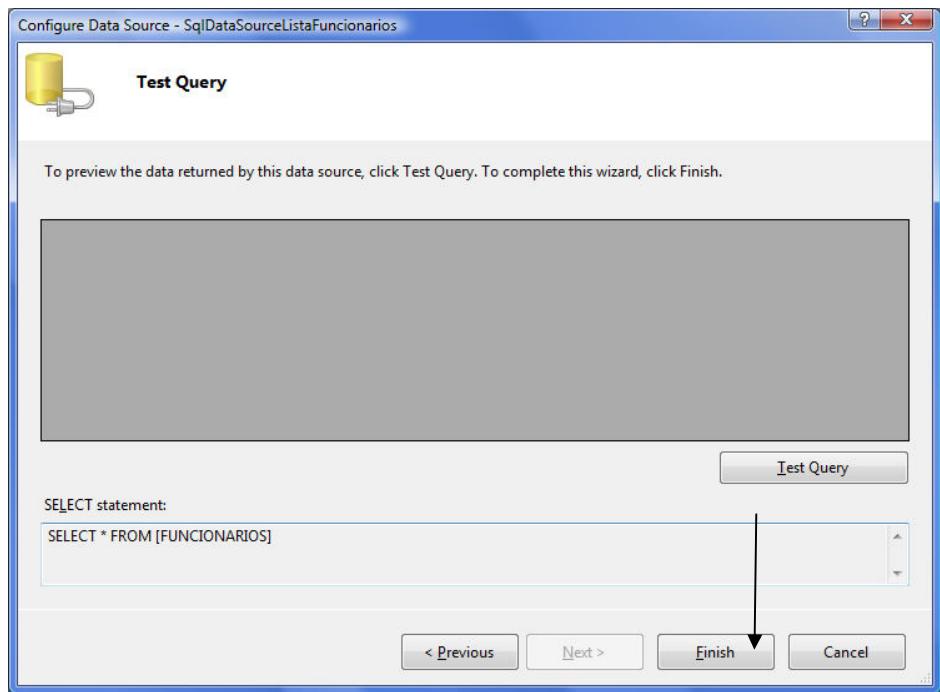
OK Cancel



Dê um nome para a conexão:



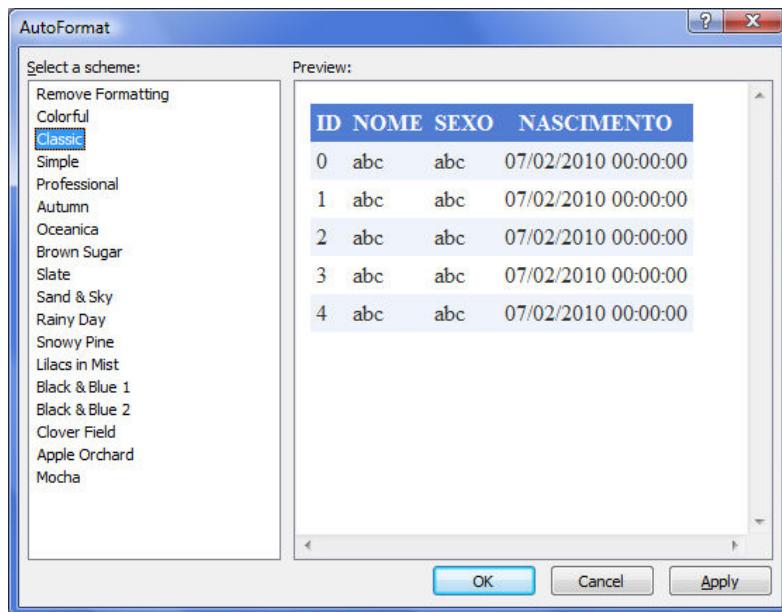




Agora, vamos configurar o Grid

The screenshot shows a 'GridView Tasks' context menu for a GridView control named 'asp:GridView#GridView1'. The menu items are: 'Auto Format...', 'Choose Control Formatting Properties', 'Configure Data Source...', 'Refresh Schema', and 'Edit Columns...'. The 'Choose Control Formatting Properties' item is currently selected. The menu is overlaid on a grid view displaying five rows of data with columns: ID, NOME, SEXO, and NASCIMENTO. The data in the grid is as follows:

ID	NOME	SEXO	NASCIMENTO
0	abc	abc	07/02/2010 00:00:00
1	abc	abc	07/02/2010 00:00:00
2	abc	abc	07/02/2010 00:00:00
3	abc	abc	07/02/2010 00:00:00
4	abc	abc	07/02/2010 00:00:00



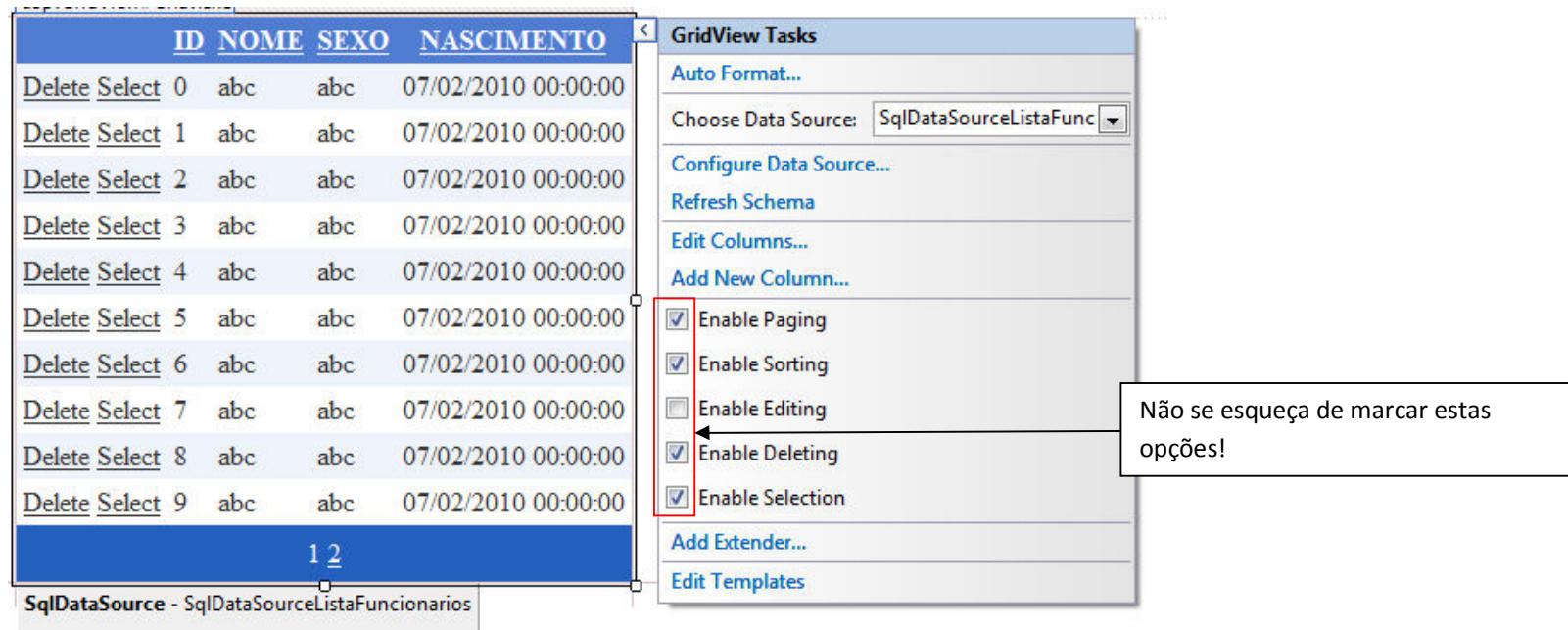
GridView Tasks

- Auto Format...
- Choose Data Source: SqlDataSourceListaFunc
- Configure Data Source...
- Refresh Schema
- Edit Columns...
- Add New Column...
- Enable Paging
- Enable Sorting
- Enable Editing
- Enable Deleting
- Enable Selection
- Add Extender...
- Edit Templates

Não se esqueça de marcar estas opções!

1 2

SqlDataSource - SqlDataSourceListaFuncionarios

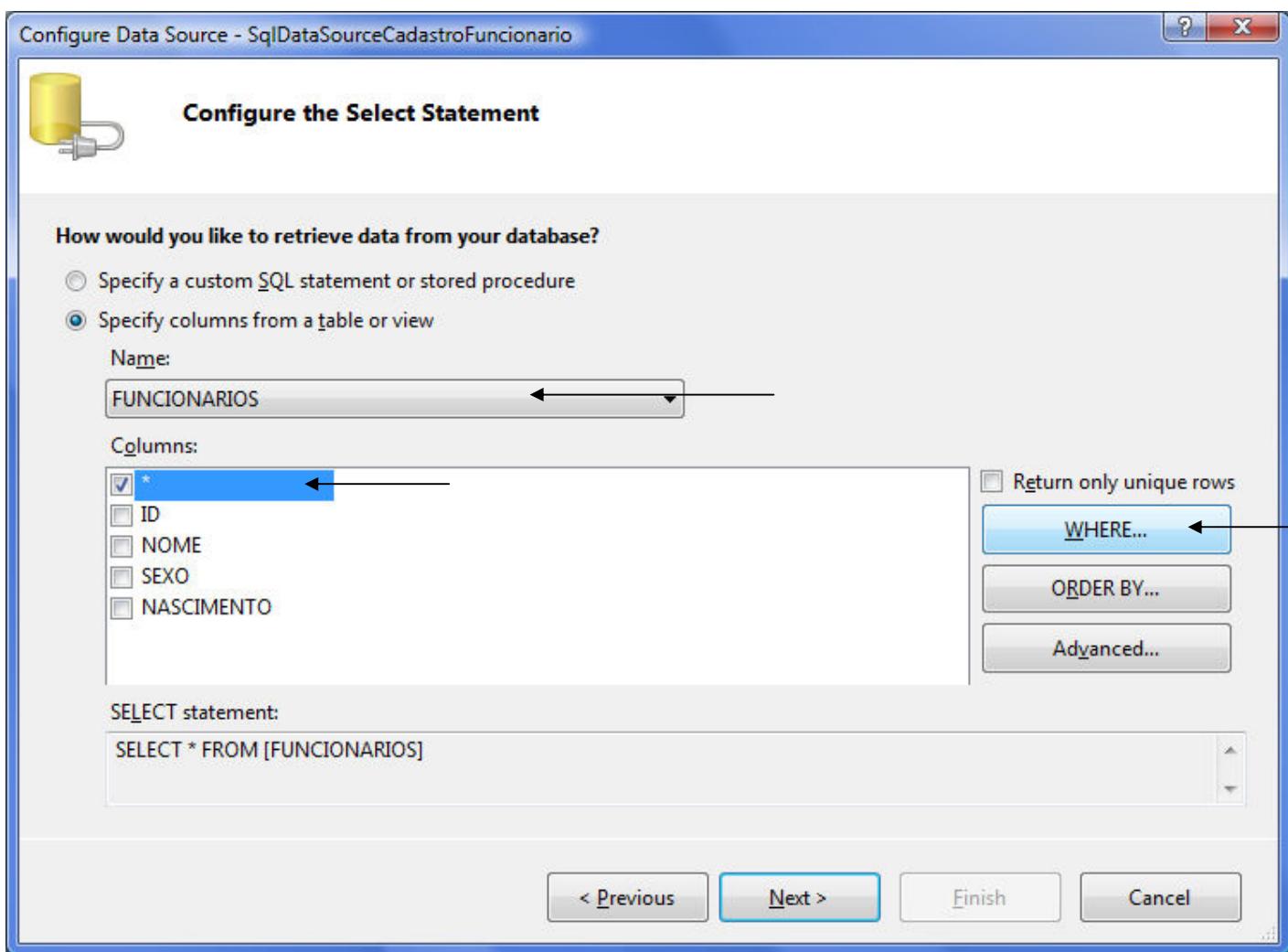


Configurando o FormView

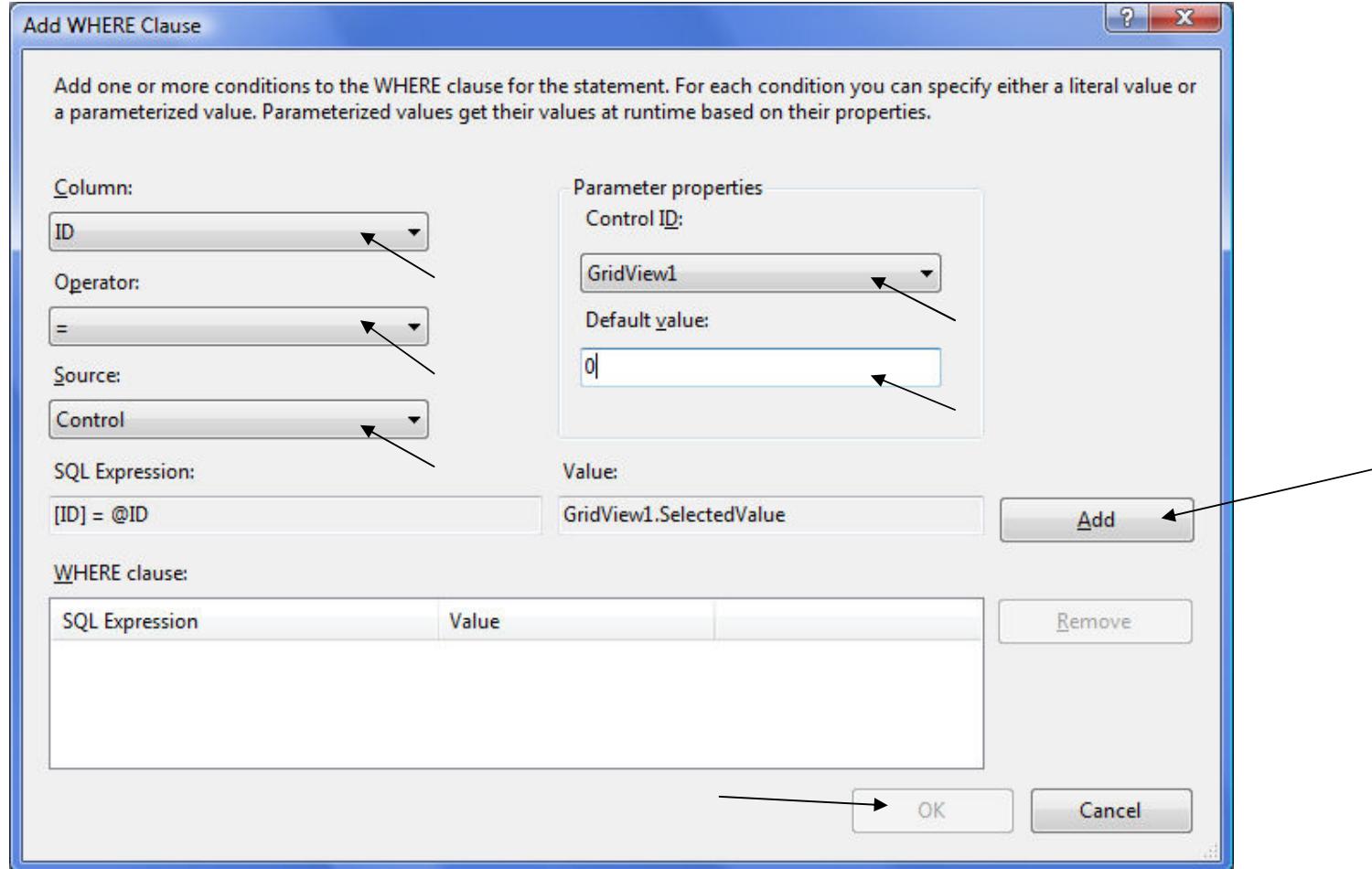
The diagram illustrates the configuration process for a FormView, showing three main windows and associated steps:

- FormView Tasks Dialog:** Shows the "Choose Data Source" dropdown set to "(None)". The option "<New data source...>" is highlighted with a blue arrow pointing from the text above.
- Data Source Configuration Wizard - Choose a Data Source Type:** The "Database" icon is selected and highlighted with a blue arrow. The "Specify an ID for the data source:" field contains "SqlDataSourceCadastroFuncionario".
- Configure Data Source - SqlDataSourceCadastroFuncionario:** The "Connection string" dropdown shows "MINHACONEXAO", which is also highlighted with a blue arrow.

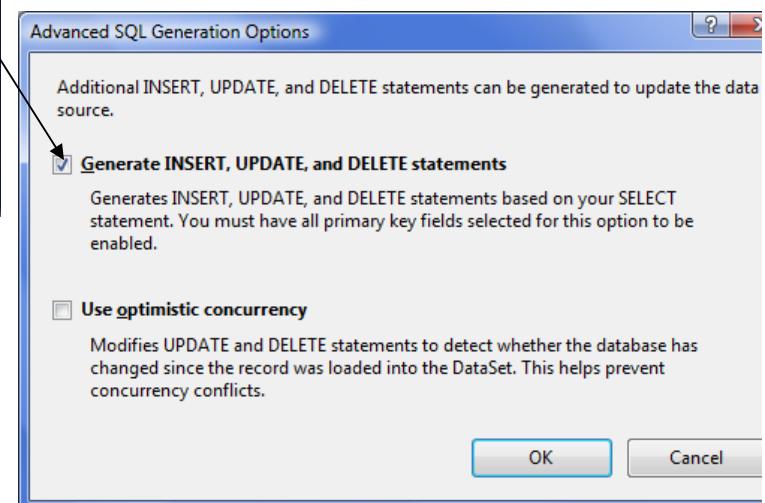
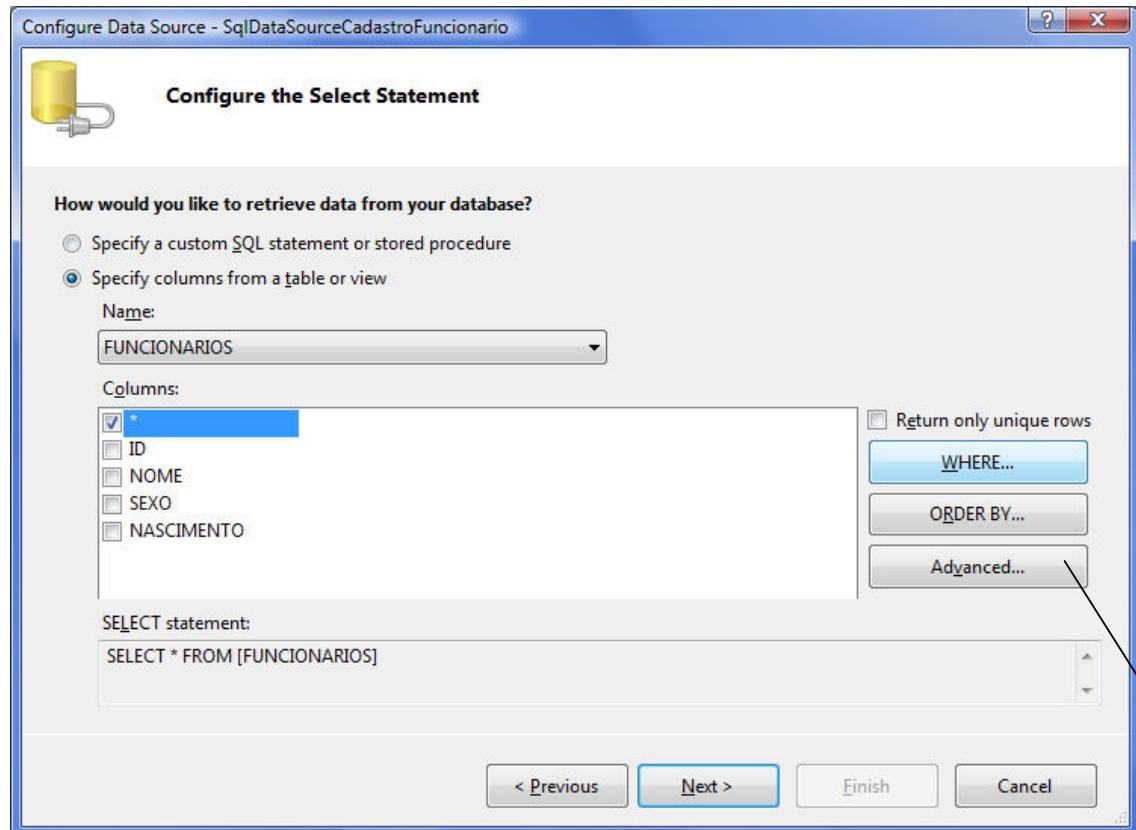
Note: Não podemos utilizar o mesmo datasource da grid, pois este terá uma cláusula WHERE, restringindo os registros a apenas 1 ocorrência.

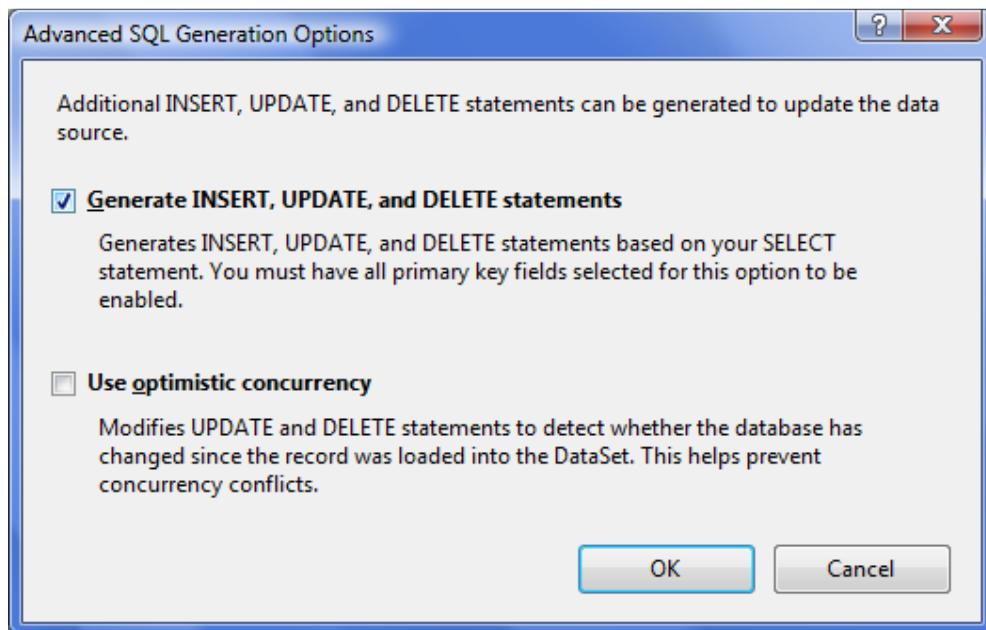


Nesta etapa, iremos configurar o formview para exibir o registro selecionado no gridview.



Retornando Para esta tela, clique agora no botão Advanced:





Codificando

Clique 2x no botão  e coloque este código:

```
protected void Button1_Click(object sender, EventArgs e)
{
    FormView1.ChangeMode(FormViewMode.Insert);
}
```

Agora, vamos programar 3 eventos do formview:

The screenshot shows the Microsoft Visual Studio interface for a web application named "CadastroFormView".

Toolbox: Standard category selected, showing various web controls like Pointer, Label, TextBox, Button, LinkButton, ImageButton, HyperLink, DropDownList, ListBox, CheckBox, CheckBoxList, RadioButton, RadioButtonList, Image, ImageMap, Table, BulletedList, HiddenField, Literal, Calendar, AdRotator, FileUpload, Wizard, Xml, MultiView.

Code View: Default.aspx.cs tab is active, showing the code for the page.

Design View: Default.aspx tab is active, showing the visual representation of the page. It contains a **SqlDataSource - SqlDataSourceListaFuncionarios** control and a **asp:FormView#FormView1** control.

Properties Window: Shows the properties for **FormView1**. The **ItemDeleted**, **ItemDeleting**, and **ItemUpdated** events have arrows pointing to external code, indicating they are programmed.

Event Handler
DataBinding
DataBound
Disposed
Init
ItemCommand
ItemCreated
ItemDeleted
ItemDeleting
ItemInserted
ItemInserting
ItemUpdated
ItemUpdating
Load
ModeChanged
ModeChanging
PageIndexChanged
PageIndexChanging
PreRender
Unload

Programe os 3 eventos!

O código é o mesmo para os 3 eventos: Vamos atualizar o Gridview:

```
GridView1.DataBind();
```

OBS: para retornar ao modo de desenho do form, pressione SHIFT + F7.

Execute o programa (F5)

The screenshot shows a Microsoft Internet Explorer window with the title bar "http://localhost:49582/CadastroFormView/Default.aspx - Windows Internet Explorer". The address bar displays the same URL. Below the address bar is a toolbar with icons for Back, Forward, Stop, Refresh, and Favorites. The main content area contains a table with the following data:

ID	NOME	SEXO	NASCIMENTO
Excluir Selecionar 2 Adamastor M 25/10/2009 00:00:00			

Below the table, there is a button labeled "NOVO REGISTRO". To the left of the table, the text "ID: 2" is displayed. Underneath the table, the details "NOME: Adamastor", "SEXO: M", and "NASCIMENTO: 25/10/2009 00:00:00" are shown, each preceded by a blue link: "Edit", "Delete", and "New".

Erro com tipo de dado DateTime.

Pode ocorrer um erro com o tipo de Dado **DateTime** quando houver algum campo deste tipo. Para corrigir, faça o seguinte:



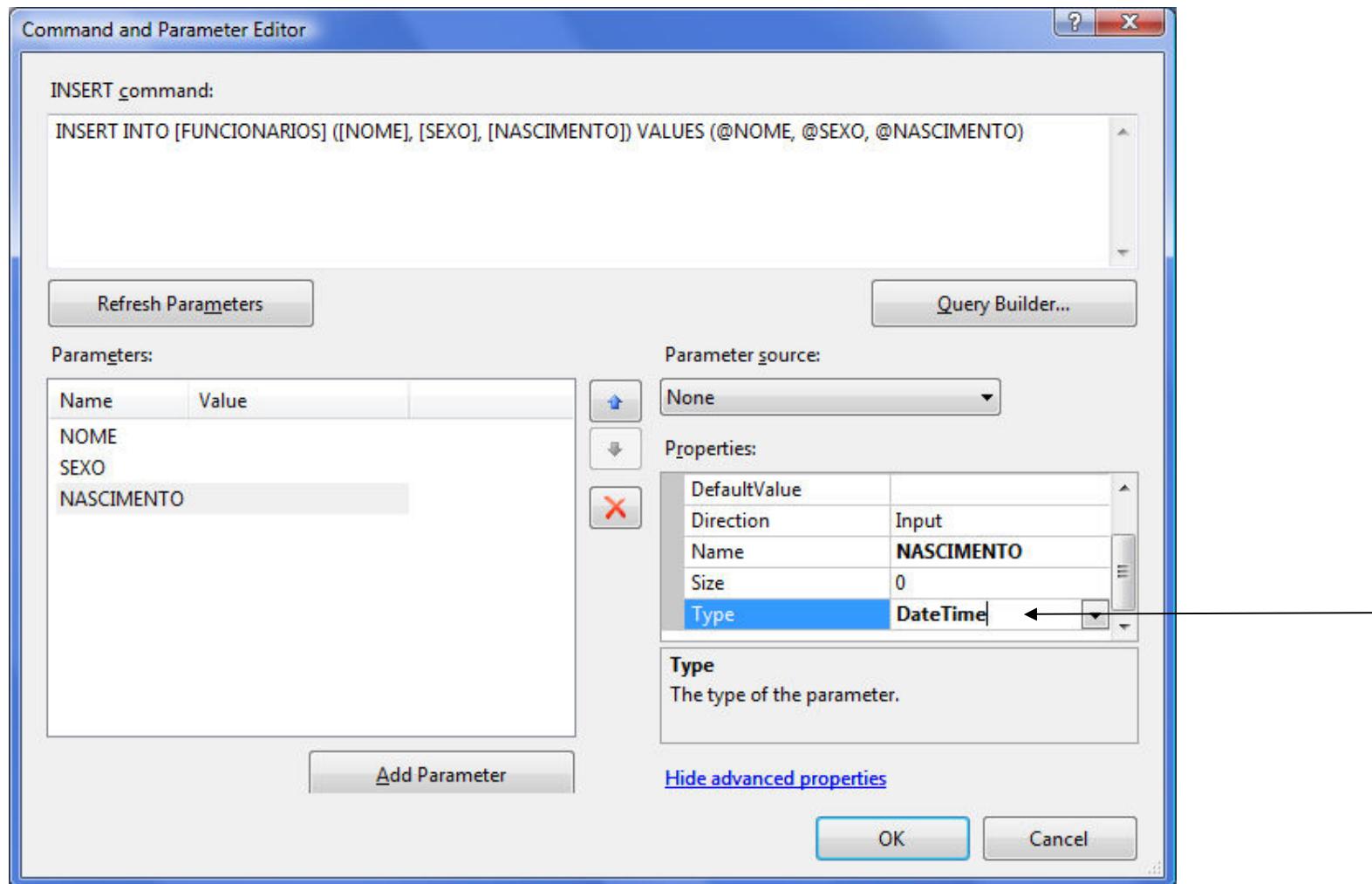
Selecione o **SQLDataSource** referente ao Formview:

Clique na propriedade **InsertQuery**: **InsertQuery** (Query)

The screenshot shows the 'Command and Parameter Editor' dialog box. It contains the following elements:

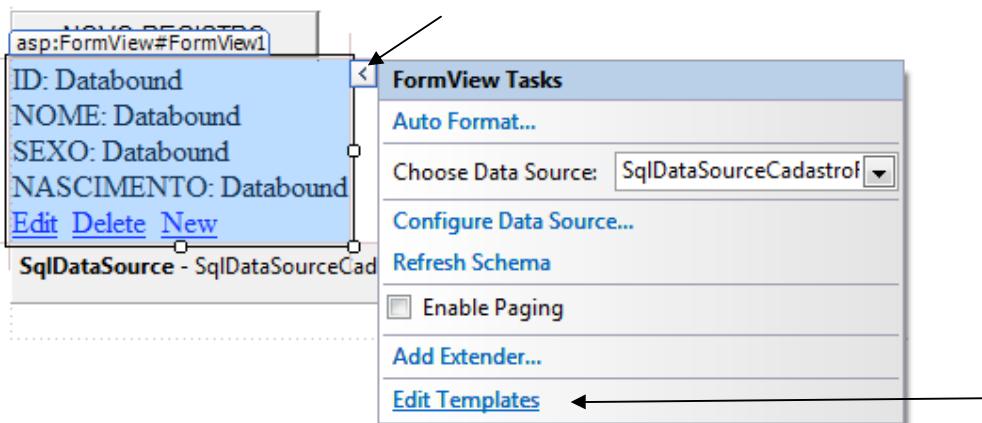
- INSERT command:** Shows the SQL query: `INSERT INTO [FUNCIONARIOS] ([NOME], [SEXO], [NASCIMENTO]) VALUES (@NOME, @SEXO, @NASCIMENTO)`.
- Parameters:** A table with columns 'Name' and 'Value'. It lists three parameters: NOME, SEXO, and NASCIMENTO. The row for 'NASCIMENTO' is highlighted with a blue background.
- Parameter source:** A dropdown menu set to 'None'. Below it are buttons for 'QueryBuilder...', 'None', 'Default Value:', and 'Show advanced properties'.
- Annotations:** Three callout boxes with arrows pointing to specific areas:
 - A box labeled 'Selecionar o(s) campo(s) do tipo DateTime' points to the 'Parameter source' dropdown.
 - A box labeled 'Clique neste link' points to the 'Show advanced properties' link.
 - A box labeled 'Clique neste link' also points to the 'Show advanced properties' link.

Verifique se a propriedade **Type** está com o valor **Datetime**. Se estiver com o valor **Date**, está incorreto. Selecione o tipo **DateTime**, como na figura abaixo:

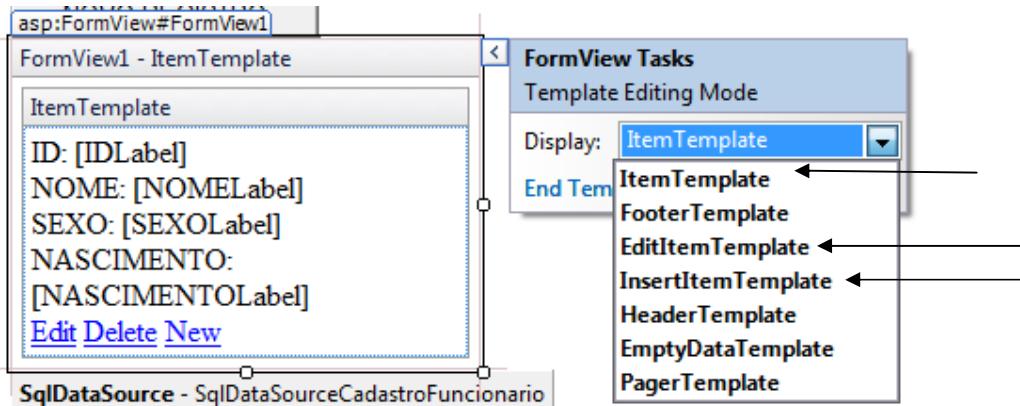


Configurando o FormView

Clique no formview e selecione a opção Edit Templates:



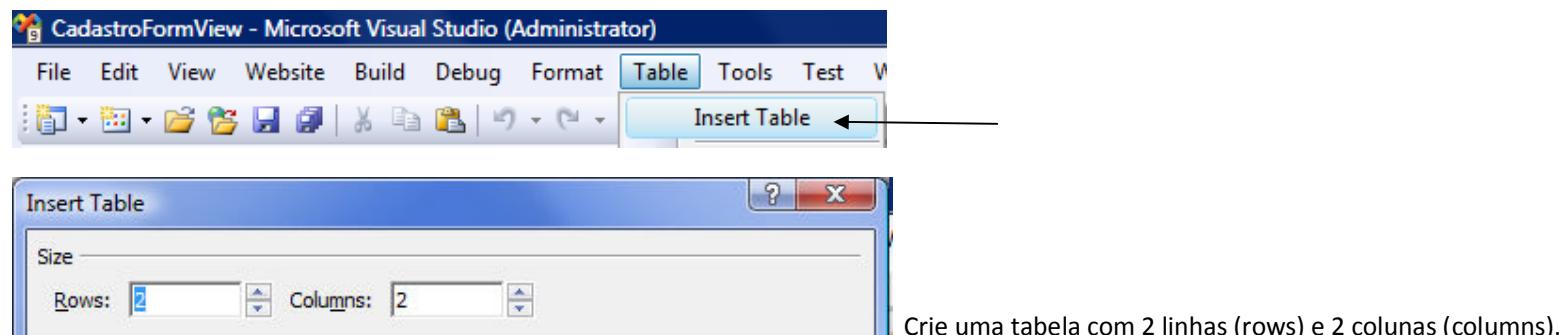
Iremos configurar 3 templates: Modo de consulta (padrão), modo de inclusão e modo de alteração.



Modo de consulta: (Item Template)

Neste modo, os componentes não podem ser editáveis, pois o usuário irá apenas consultar os dados do registro.

Você pode usar uma tabela para organizar os objetos dentro das células. Para criar uma tabela sem ter que codificar, utilize o menu:



Para criar novas linhas na tabela, utilize a tecla TAB. A tela configura deverá ficar assim:

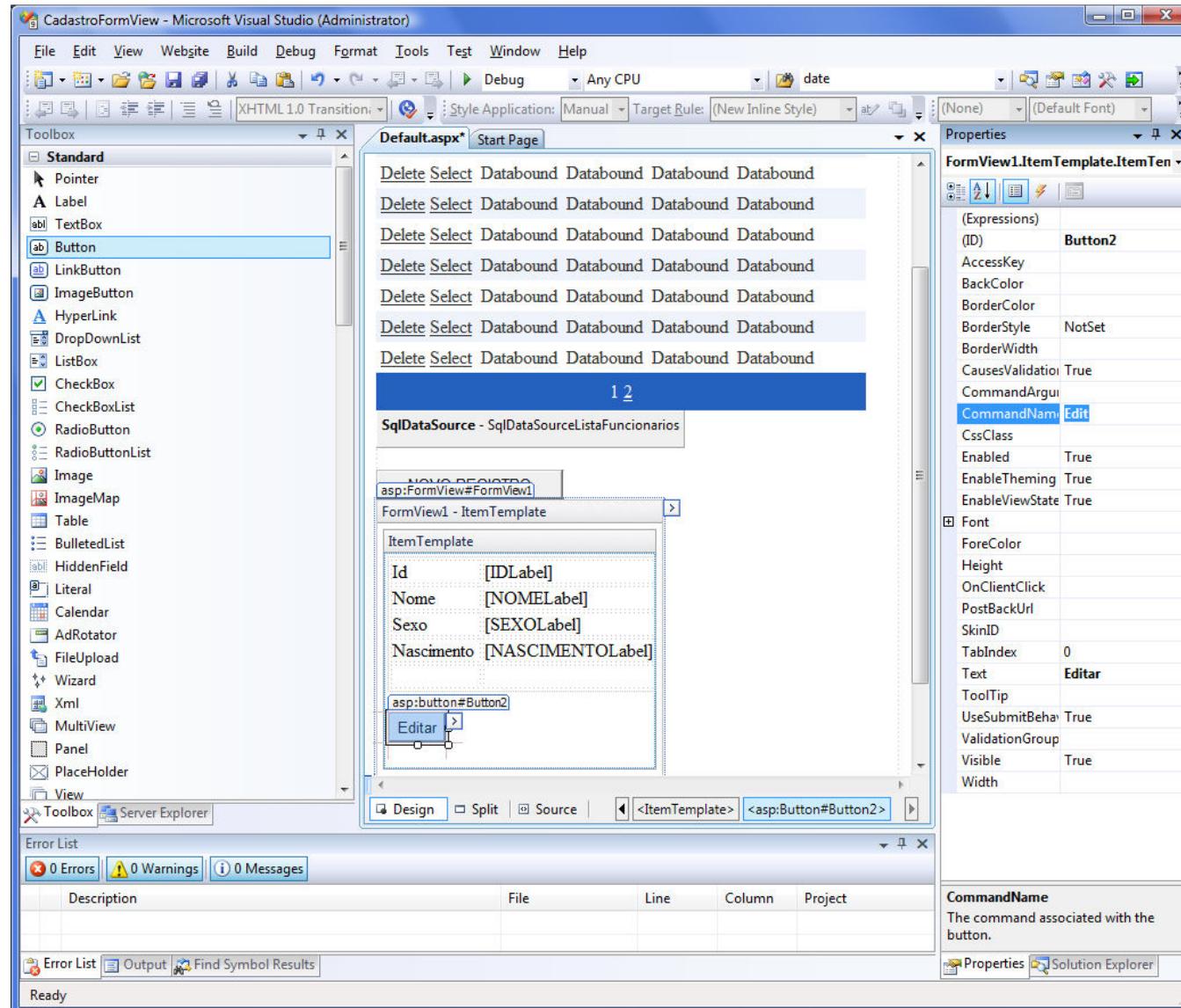
Remova o linkbutton [New](#), afinal, já temos um botão para inserir um novo registro. Também remova o [Delete](#) pois iremos remover pelo GridView.

Se quiser, pode substituir o linkbutton por um button ou um imagebutton. A única propriedade que você deverá desses objetos é a commandName.

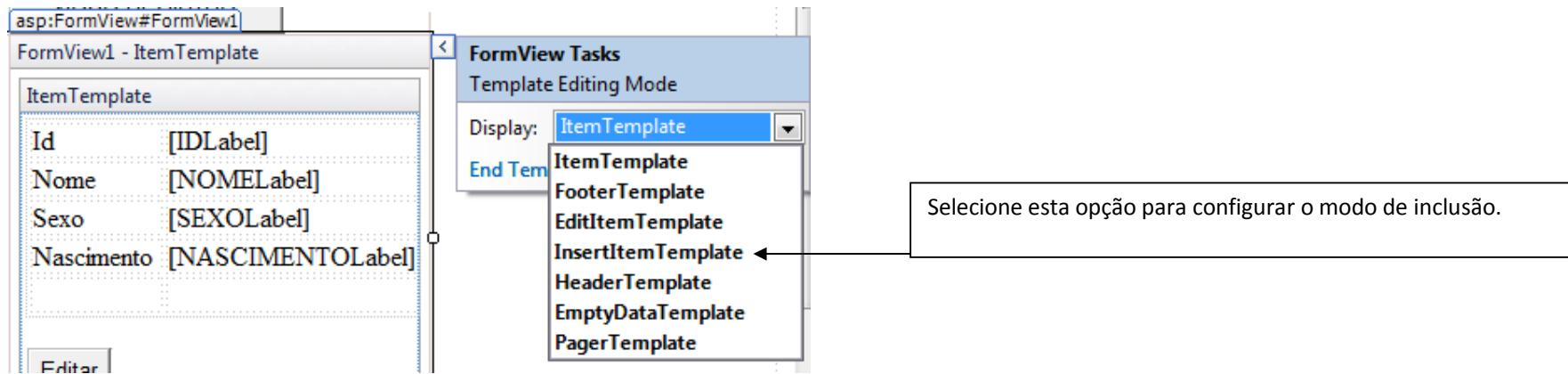
Veja abaixo os valores possíveis para a propriedade :

Ação	CommandName
Novo registro	New
Editar	Edit
Gravar quando for uma alteração	Update
Gravar quando for uma inclusão	Insert
Cancelar	Cancel
Apagar	Delete

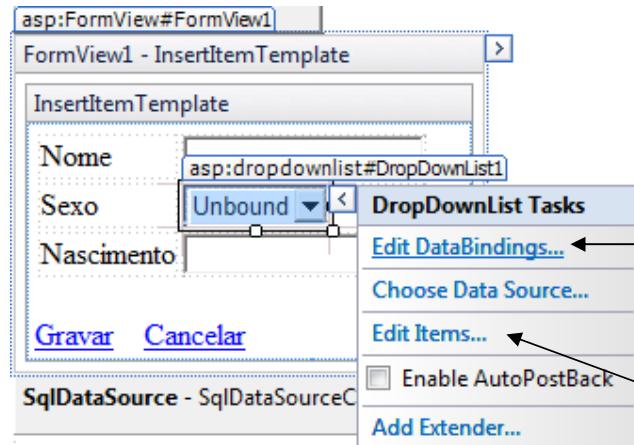
Após as alterações, a tela deverá se assemelhar a figura abaixo:



Modo de inclusão (InsertItemTemplate):



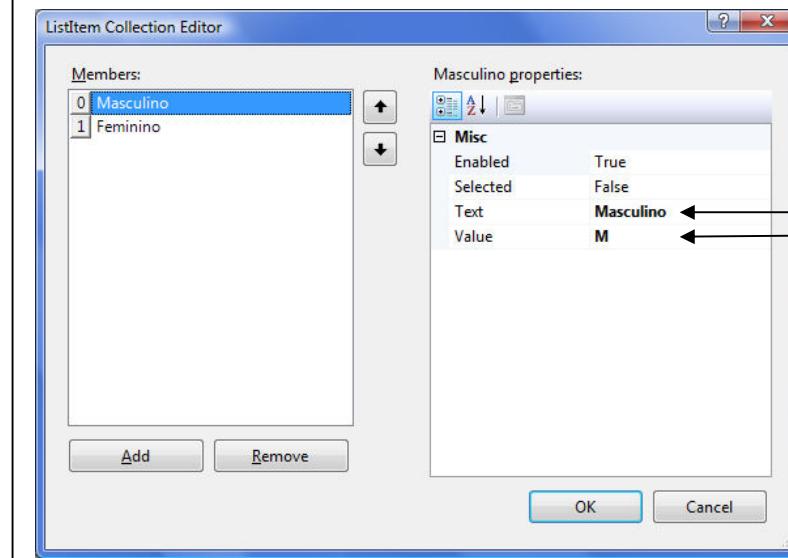
Você também pode utilizar uma tabela para organizar os objetos. Aprovei para traduzir (propriedade Text) ou substituir os linkButton Edit e Cancel por botões.



Para o campo sexo pode ser utilizado o componente DropDownList. Em **Edit DataBindings** deve-se informar o campo que será lido/gravado o valor do campo (neste caso, o campo sexo)



Os valores que irão aparecer no DropDownList devem ser inseridos na opção **Edit items**.

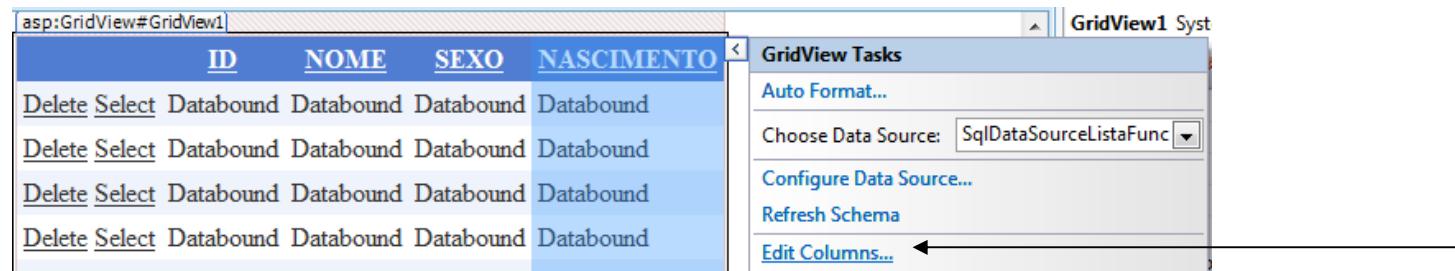


Modo de alteração (EditItemTemplate):

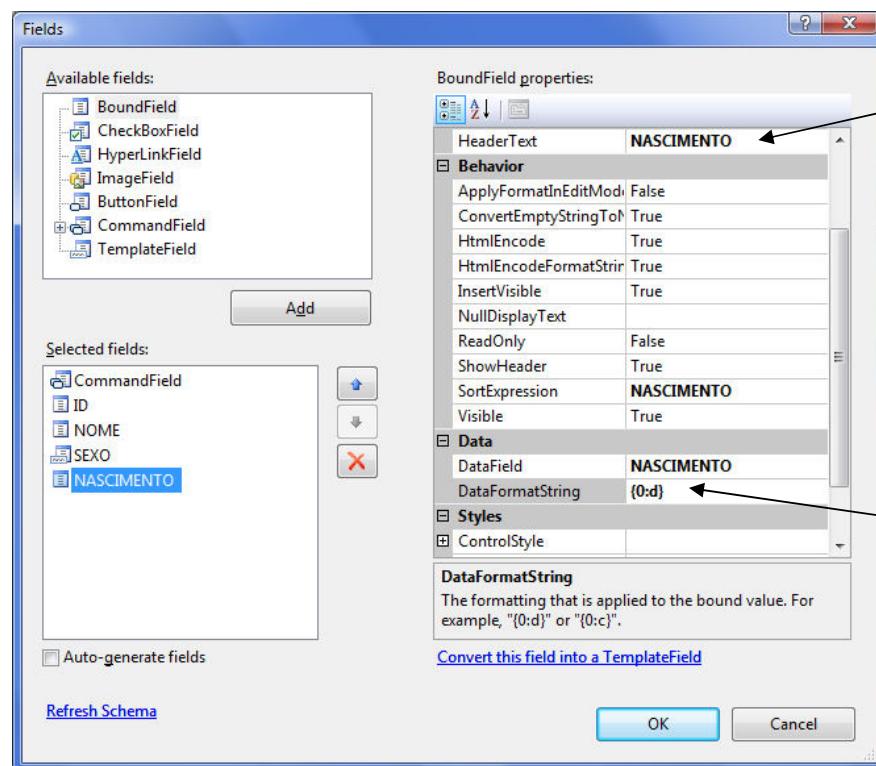
Este modo é idêntico ao modo de inclusão. A única alteração é no linkbutton de Gravar, que em uma alteração tem seu **commandText = Update**.

Configurando o GridView

Para configurar os títulos, exibir ou ocultar colunas, selecione a opção abaixo:



The screenshot shows the 'GridView Tasks' dialog box for 'GridView1'. It includes options like 'Auto Format...', 'Choose Data Source:' (set to 'SqlDataSourceListaFunc'), 'Configure Data Source...', 'Refresh Schema', and 'Edit Columns...'. The 'Edit Columns...' link is underlined and has a red arrow pointing to it from the text below.



The screenshot shows the 'Fields' dialog box. On the left, 'Available fields' include BoundField, CheckBoxField, HyperLinkField, ImageField, ButtonField, CommandField, and TemplateField. 'Selected fields' include CommandField, ID, NOME, SEXO, and NASCIMENTO. The 'NASCIMENTO' field is selected. The 'BoundField properties' section shows:

- HeaderText:** NASCIMENTO
- Behavior:** ApplyFormatInEditMode: False, ConvertEmptyStringToNull: True, HtmlEncode: True, HtmlEncodeFormatString: True, InsertVisible: True, NullDisplayText: True, ReadOnly: False, ShowHeader: True, SortExpression: NASCIMENTO, Visible: True
- Data:** DataField: NASCIMENTO, DataFormatString: {0:d}
- Styles:** ControlStyle (empty)
- DataFormatString:** The formatting that is applied to the bound value. For example, "{0:d}" or "{0:c}".

Buttons at the bottom include 'OK' and 'Cancel'.

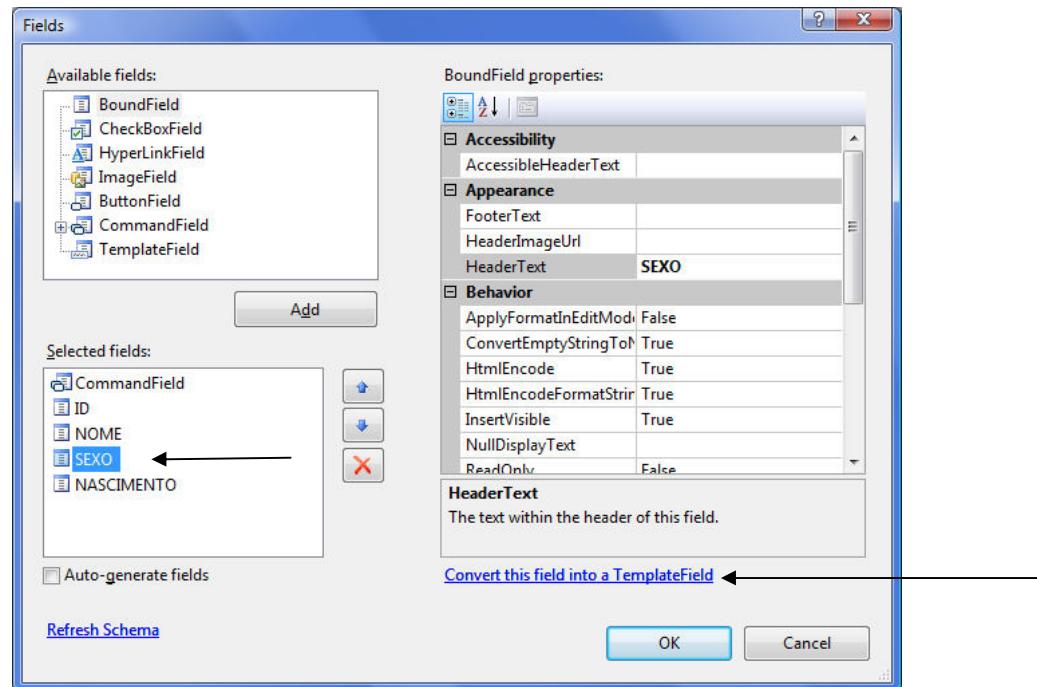
Para alterar o título da coluna, altere a propriedade **HeaderText**:

DataFormatString: Usado para formatar o texto. Neste caso, {0:d} formata a data para exibir apenas a data, ou seja, não exibe a hora.

Exibindo um valor personalizado na em uma coluna do Gridview.

Na tabela, a coluna Sexo armazena os valores F para feminino e M para masculino. Vamos exibir no gridview esses valores.

Para tanto, seleciona a opção Edit columns do gridview, selecione a coluna Sexo e clique em [Convert this field into a TemplateField](#)



No Gridview, selecione agora a opção [Edit Templates](#), como na figura abaixo:

GridView Tasks

- [Auto Format...](#)
- [Choose Data Source:](#) SqlDataSourceListaFunc
- [Configure Data Source...](#)
- [Refresh Schema](#)
- [Edit Columns...](#)
- [Add New Column...](#)
- [Move Column Left](#)
- [Remove Column](#)
- [Enable Paging](#)
- [Enable Sorting](#)
- [Enable Editing](#)
- [Enable Deleting](#)
- [Enable Selection](#)
- [Add Extender...](#)
- [Edit Templates](#) ←

asp:GridView#GridView1

GridView1 - Column[3] - SEXO

ItemTemplate

[Label1]

GridView Tasks

Template Editing Mode

Display: **ItemTemplate**

[End Template Editing](#)

Observe que o template referente a coluna sexo veio selecionado.

Altere o nome do Label1 para LabelSexo (selecione o label1, press. F4 e altere a propriedade ID)

Ao terminar, clique em [End Template Editing](#)

Agora, selecione o evento **RowDataBound** do Gridview e coloque o seguinte código:

```
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    // se for uma linha de dados
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        // encontra o label dentro da linha que está sendo renderizada
        Label LabelSexo = e.Row.FindControl("LabelSexo") as Label;

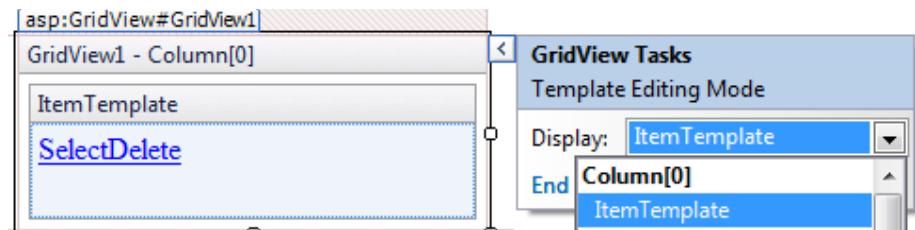
        // altera o texto do label
        if (LabelSexo.Text == "M")
            LabelSexo.Text = "Masculino";
        else if (LabelSexo.Text == "F")
            LabelSexo.Text = "Feminino";
        else
            LabelSexo.Text = "";
    }
}
```

Colocando uma mensagem de aviso no linkbutton de excluir do GridView. Substituindo os linkbuttons por botões.

Primeiro, converta a coluna que contém as opções de **Select Delete** em um **TemplateField**:

The screenshot shows the 'Selected fields:' list on the left containing 'CommandField', 'ID', 'NOME', 'SEXO', and 'NASCIMENTO'. Below it is a checkbox for 'Auto-generate fields'. On the right, there's a 'TemplateField' configuration panel with columns for 'EditImageUrl', 'EditText', 'FooterText', 'HeaderImageUrl', 'HeaderText', and 'InsertImageUrl'. The 'HeaderText' column contains the placeholder 'The text within the header of this field.' At the bottom, a button says 'Convert this field into a TemplateField'.

Agora, escolha a opção [End Template Editing](#) (como no tópico anterior). Verifique se você está editando a coluna 0.



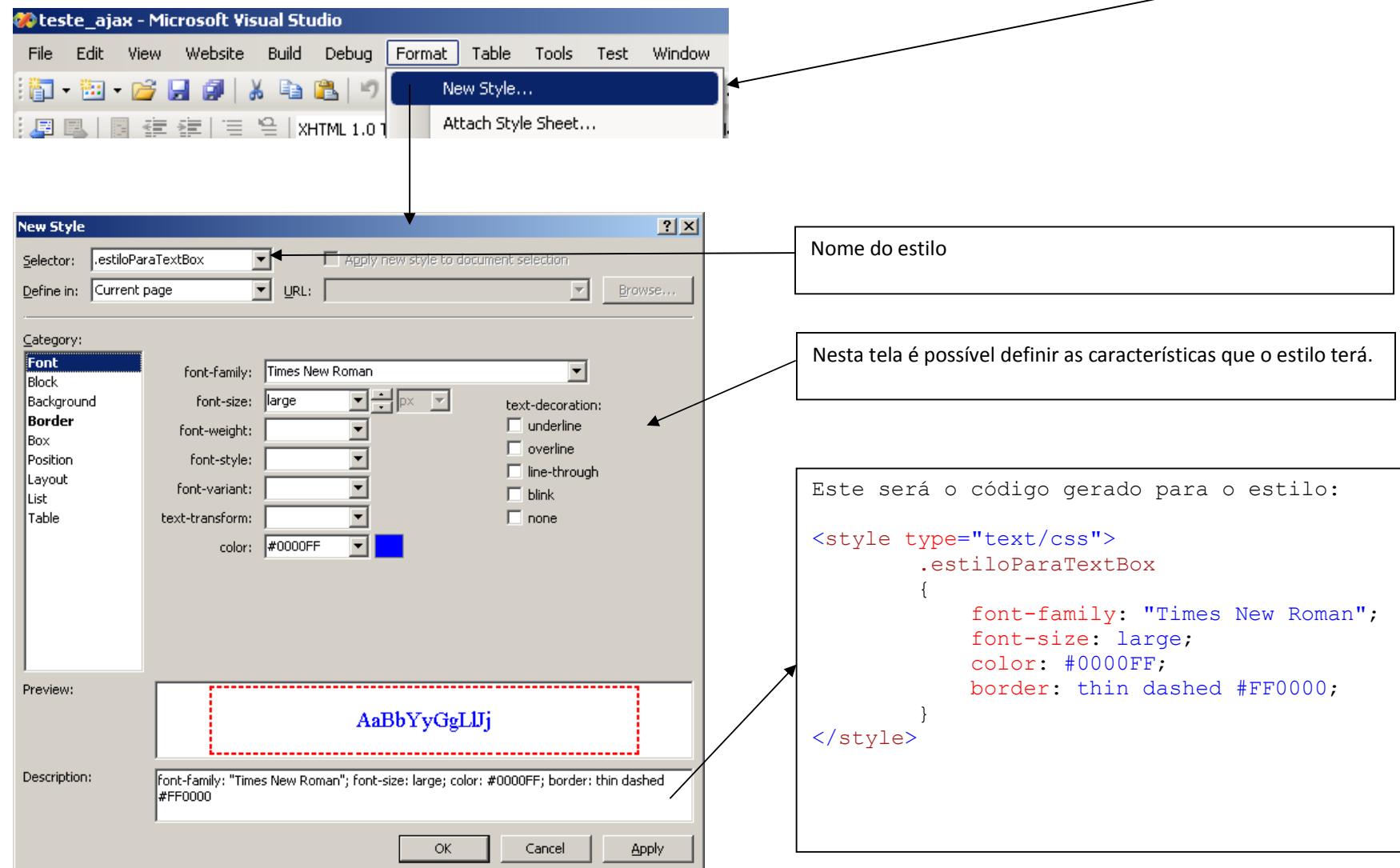
Neste momento, é possível substituir os linkbuttons por outros objetos, como buttons ou imagebuttons, basta preencher a propriedade **CommandName** dos novos objetos com os mesmos valores que haviam nos objetos excluídos. Você também pode traduzir os textos, alterando a propriedade **Text**.

No caso do linkbutton de apagar (Delete), para solicitar a confirmação da ação de excluir:

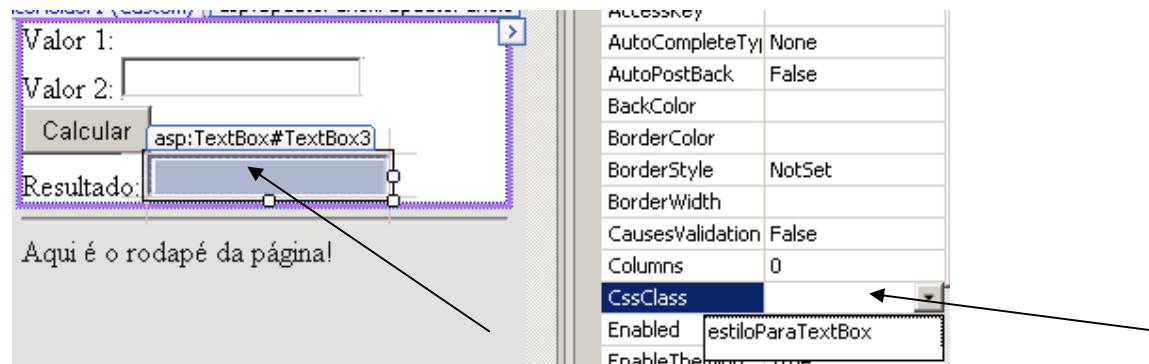
Coloque na propriedade **onclientClick**: **if (confirm('Deseja apagar?')) return true; else return false;**

CSS

Qualquer componente pode se formatado com CSS. Para tanto, é necessário criar um ou mais estilos, para depois aplicá-los aos objetos. Estes estilos podem ser salvos na própria página ou em um arquivo separado. Para adicionar um novo estilo, com o formulário no modo de **Design**, selecione a opção **New Style...**:



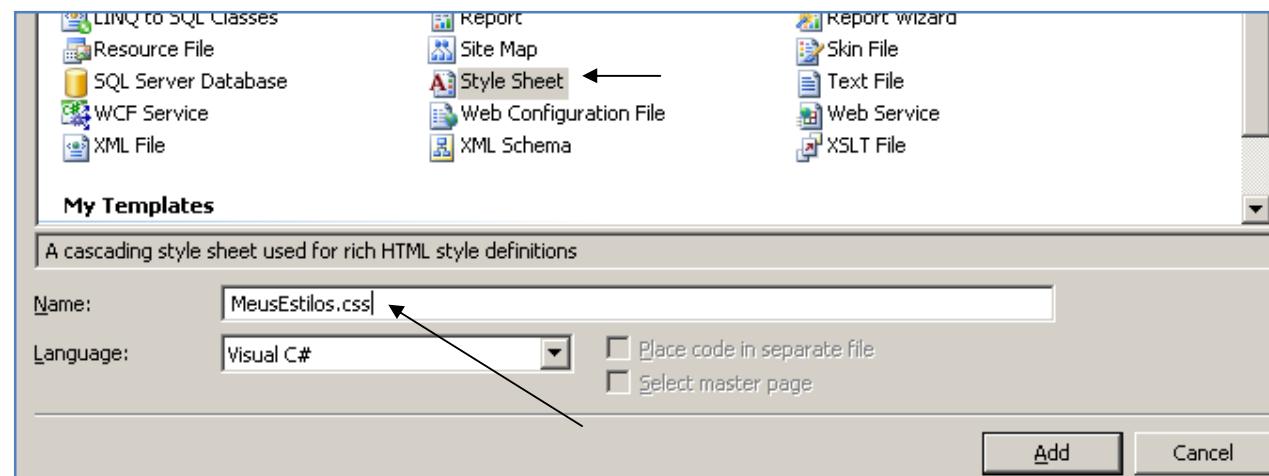
Para aplicar o estilo, basta acessar a propriedade `CssClass` do objeto e selecionar o estilo desejado:



Veja o objeto após aplicação do estilo:



Também é centralizar deixar todos os estilos em um único arquivo. Para fazer isso, acesso o menu File->New->File e escolha a opção  **Style Sheet** :



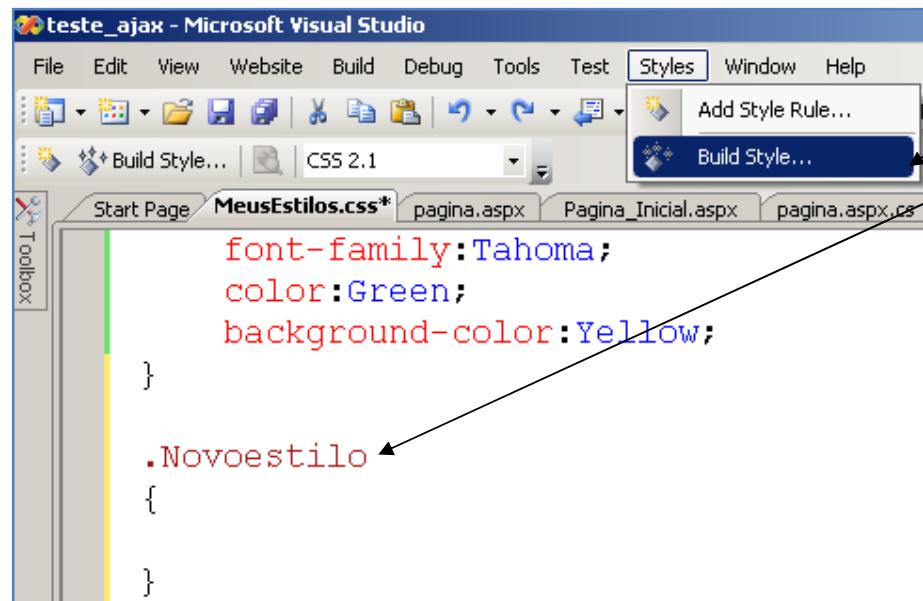
Código do arquivo MeuEstilo.CSS:

```
body
{
    background-color:Fuchsia;
    border-style:solid;
}

.EstiloBotao
{
    font-family:Tahoma;
    color:Green;
    background-color:Yellow;
}
```

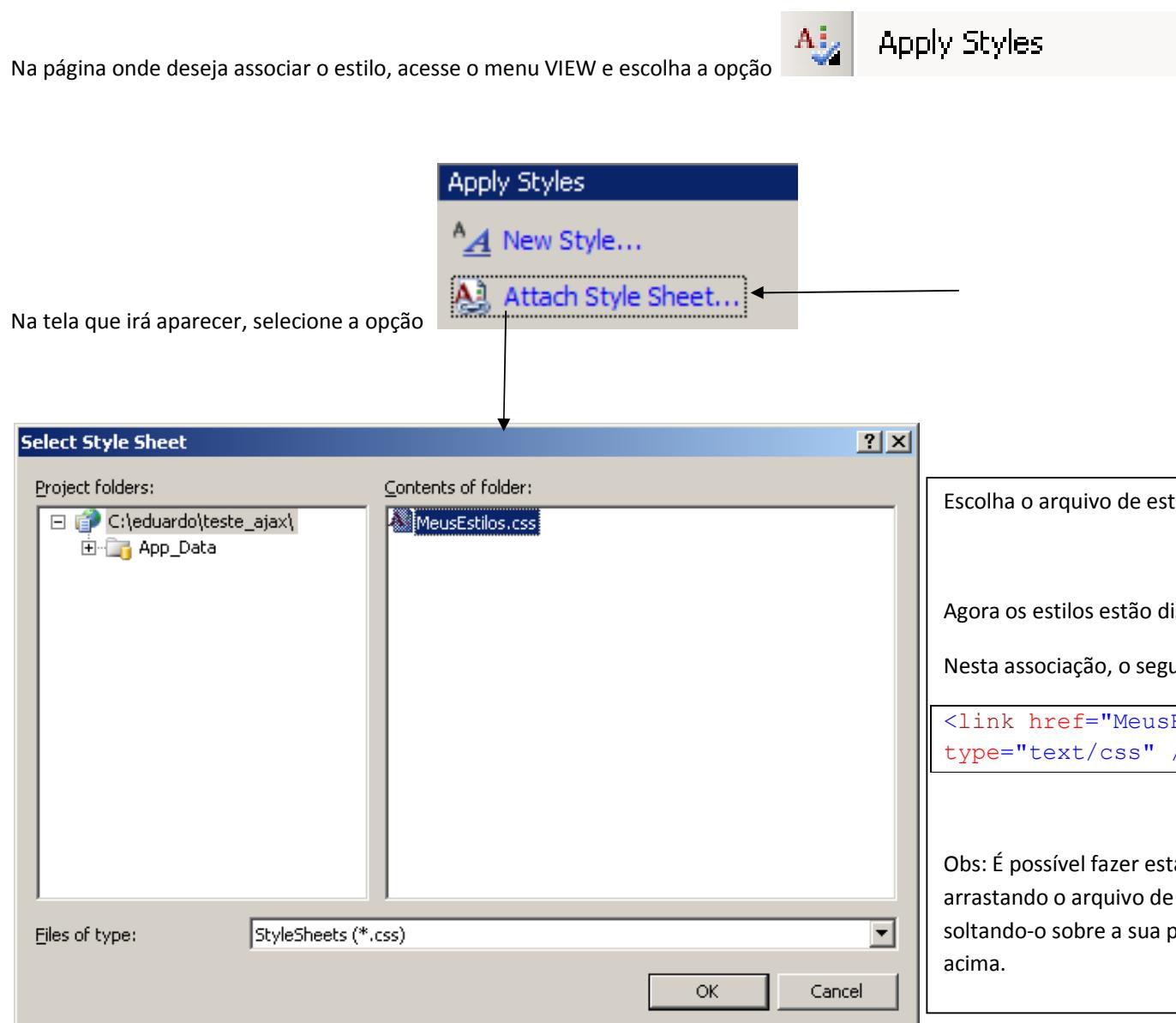
Este estilo será aplicado ao corpo da página em que ele for usado

Inicie os estilos com um “.”



Você pode dar um nome ao estilo e utilizar a opção Build Style para criar seu estilo!

Para associar seu arquivo de estilos a uma determinada página, faça o seguinte:



Validadores

The screenshot shows the 'Validation' section of the ASP.NET toolbox. It lists several validation controls: Pointer, RequiredFieldValidator, RangeValidator, RegularExpressionValidator, CompareValidator, CustomValidator, and ValidationSummary. Each control has a corresponding description box to its right.

- RequiredFieldValidator:** Utilizado quando o campo é de preenchimento obrigatório. Preencha a propriedade CONTROLTOVALIDATE.
- RangeValidator:** Utilizado quando o campo possui uma faixa de valor, ex: Salário entre 0 e 5000. Não esqueça de alterar a propriedade TYPE, informando o tipo de dado que será validado (String, integer, double, etc...). Preencha as propriedades MAXIMUMVALUE E MINIMUMVALUE. Preencha a propriedade CONTROLTOVALIDATE.
- CustomValidator:** Utilizado quando o campo possui uma validação muito específica, que não pode ser feita com os outros validadores. Será possível inserir um código C# para efetuar a validação. Ela será executada no servidor.
- ValidationSummary:** Opcional, é utilizado para exibir o resumo da validação. Não preencha a propriedade CONTROLTOVALIDATE.

The screenshot shows the 'InsertItemTemplate' of a FormView control. It contains three text input fields labeled 'Nome', 'Sexo', and 'Nascimento', each with a red asterisk (*) indicating it is required. Below the fields are two buttons: 'GRAVAR' and 'CANCELAR'. Arrows point from the validation annotations in the FormView template to the corresponding validation descriptions on the right.

Nome: Nome obrigatório: utilize o **RequiredFieldValidator**
Preencha as propriedades: ControlToValidade - objeto que você quer Validar
ErrorMessage: "Preencha o Nome" e Text: "*"

Sexo: Nome obrigatório: utilize o **RequiredFieldValidator**
Preencha as propriedades: ControlToValidade - objeto que você quer Validar
ErrorMessage: "Informe o sexo" e Text: "*"

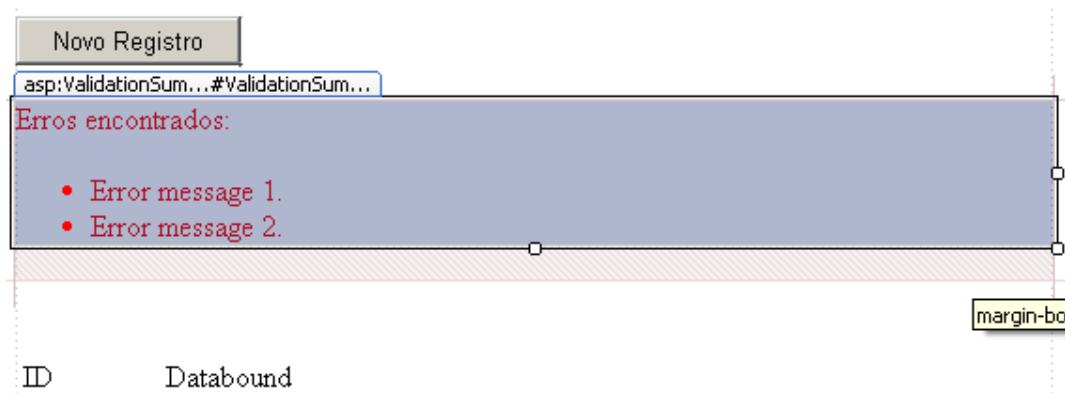
Nascimento: Nome obrigatório: utilize o **CustomValidator**
Não preencha a propriedade ControlToValidate!
ErrorMessage: "Data de nascimento inválida!" e Text: "*"
No evento SERVERVALIDATE coloque o seguinte código:

Preencha a propriedade CAUSESVALIDATION do botão Cancelar com FALSE, caso contrário, ele tentará validar os campos quando o usuário tentar cancelar o cadastro.

```
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args)
{
    args.IsValid = true;

    //encontra o objeto dentro do formview para ser validado.
    string data = (FormView1.FindControl("NASCIMENTOTextBox") as TextBox).Text;
    try
    {
        if (Convert.ToDateTime(data) > DateTime.Now)
        {
            args.IsValid = false;
        }
    }
    catch
    {
        args.IsValid = false;
    }
}
```

Para visualizar as mensagens de erro, coloque na tela (fora do formview) o componente  ValidationSummary . Caso queira digitar um título para as mensagens de erro, preencha a propriedade HEADERTEXT, como no exemplo abaixo, onde digitamos “Erros encontrados”.



Adicionando a tecnologia AJAX em um site existente

Para habilitar a utilização de AJAX, basta colocar na sua página o componente ScriptManager:

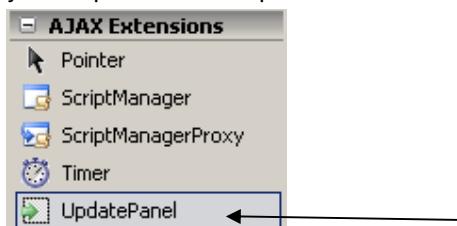


Ele deve ser colocado dentro da tag form:

```
<body>
    <form id="form1" runat="server">
        <div>
            <asp:ScriptManager ID="ScriptManager1" runat="server">
            </asp:ScriptManager>

        </div>
    </form>
</body>
</html>
```

O scriptmanager adiciona em sua página os scripts javascript necessários para o funcionamento da tecnologia AJAX, porém, para utilizar o ajax em uma página, é



necessário adicionar o componente UpdatePanel:

A página abaixo calcula a soma entre 2 valores. Após processar o resultado será exibido em uma caixa de texto. Como foi utilizado a tecnologia AJAX, não haverá “refresh” na página durante o processo de cálculo.

The screenshot shows a simple ASP.NET form. It contains two text boxes labeled "Valor 1" and "Valor 2" with the values "10" and "7" respectively. Below them is a button labeled "Calcular". Underneath the button is a text box labeled "Resultado" which displays the value "17". The entire form is enclosed in a blue border.

Código HTML:

```
<body>
<form id="form1" runat="server">
<div>
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate><!-- Valor 1: -->
            <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <br />
        <!-- Valor 2: -->
            <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
            <br />
            <asp:Button ID="Button1" runat="server" Text="Calcular" OnClick="Button1_Click" />
            <br />
        <!-- Resultado: -->
            <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
        </ContentTemplate>
    </asp:UpdatePanel>
</div>
</form>
</body>
```

Non忘了在 `<ContentTemplate>` 之后添加 `<UpdatePanel>` !!!

Código do botão de calcular:

```
protected void Button1_Click(object sender, EventArgs e)
{
    TextBox3.Text = (Convert.ToInt32(TextBox1.Text) + Convert.ToInt32(TextBox2.Text)).ToString();
}
```

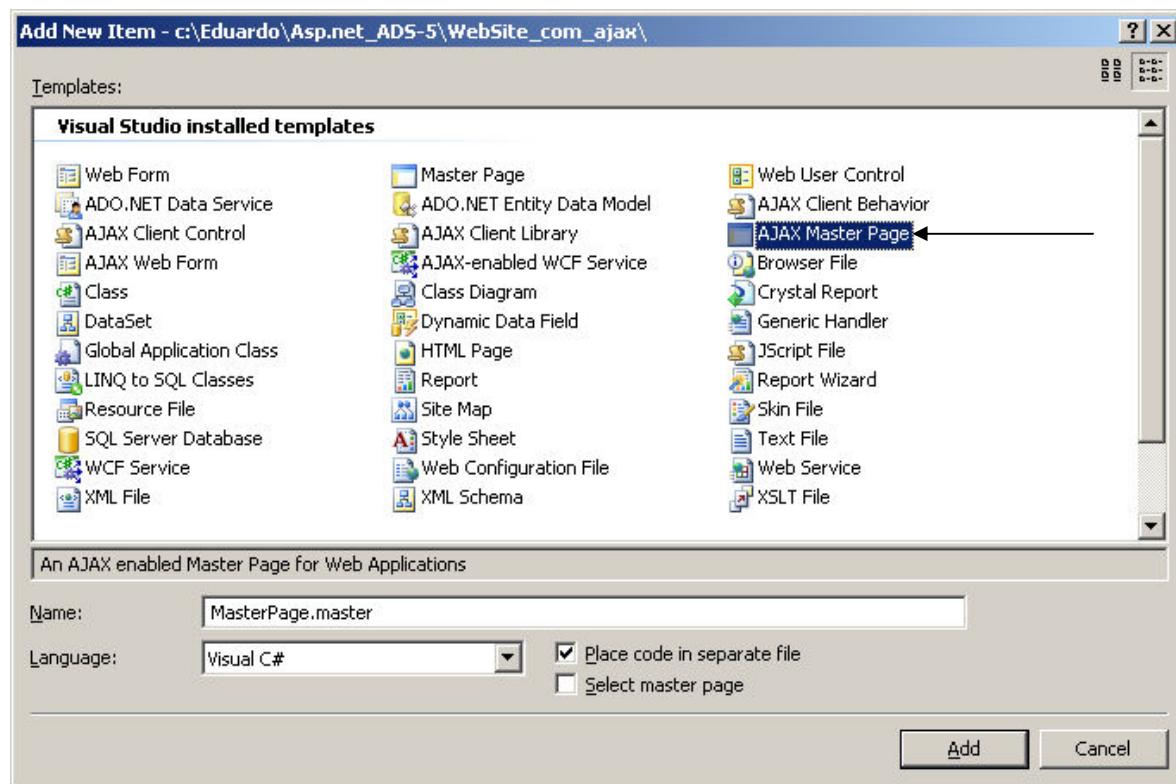
Master Page e Formulários já habilitados para AJAX

Definição: O recurso de Master Pages, sem nenhuma dúvida é uma facilidade que todo desenvolvedor sonhava em ter nos projetos. Com uma Master Page você consegue desenvolver uma página padrão que será utilizada em todo o site, ou seja, é como se fosse uma página default contendo menus, cabeçalhos e rodapés. Qualquer outra página criada, pode herdar a Master Page, o qual você poderá apenas utilizar a área que não seja a da Master Page. Como isto é feito em tempo de execução, você adotar qualquer tipo de manutenção na página, e em tempo de execução é que o .NET monta as duas páginas em apenas uma.

A grande vantagem nisto é que você não terá que dar manutenção em diversas páginas ou User Controls, uma vez criada ou alterada a Master Page, todas as mudanças são refletidas imediatamente nas demais páginas que a utilizam.

Fonte: <http://msdn.microsoft.com/pt-br/library/cc580600.aspx>

Para criar uma Master Page, selecione a opção AJAX MASTER PAGE (fig. Abaixo). Como o próprio nome já diz, esta Master Page já vem com o objeto scriptmanager adicionado, permitindo assim que se utilize a tecnologia AJAX.



Código HTML exemplo de uma Master Page:

```
<body>
  <form id="form1" runat="server">
    <div>
      <asp:ScriptManager ID="ScriptManager1" runat="server" />

      <%--coloque aqui tudo o que você quiser que apareça no topo das páginas,
      como p.ex. um menu do sistema, o logo da empresa, etc..--%>

      <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
        <%--não coloque nada aqui, pois é onde o sistema irá exibir as suas páginas!--%>
      </asp:ContentPlaceHolder>

      <%--coloque aqui tudo aquilo que você quiser que apareça no final das suas páginas,
      como p.ex. o telefone de contato, informações de copyright, etc..--%>

    </div>
  </form>
</body>
```

Exemplo de uma master Page com menu :



Para adicionar o menu na página, utilize o objeto:

Altere a propriedade do menu: Orientation **Horizontal**

Para configurar o menu, clique em Edit Menu Items:

Auto-formatação do menu

Editar os elementos do menu

Adicionar novos itens ao menu

Caminho da página que será exibida. Digite ou selecione na caixa combo. No nosso exemplo, teremos que digitar pois ainda não criamos o arquivo chamado “pagina.aspx”!!!
~/ significa: Raiz do site

Texto que irá aparecer no menu.

A máster Page terá a seguinte aparência:

MasterPage.master* Default.aspx

ScriptManager - ScriptManager1

Calculadora

Aqui é o rodapé da página!

Código fonte da master Page:

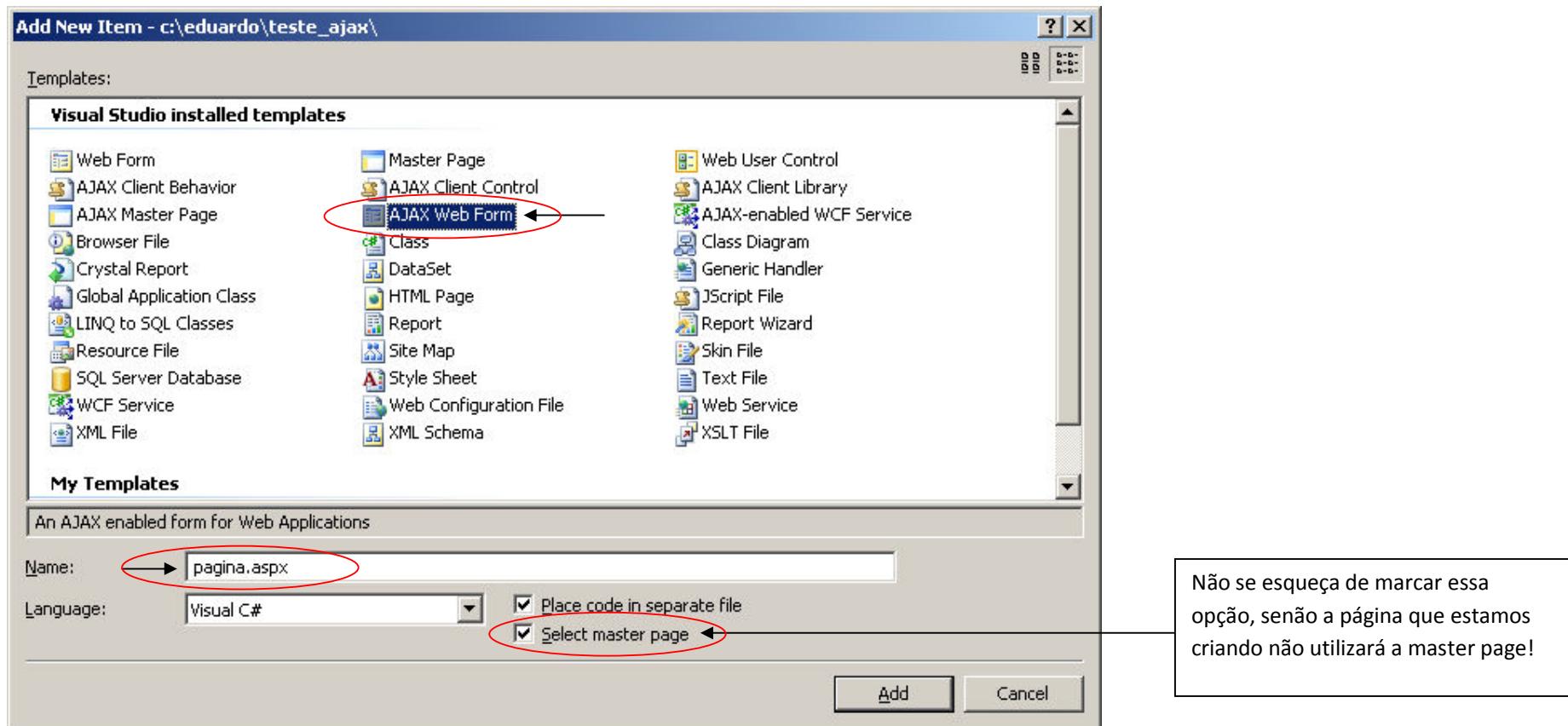
```
<body>
    <form id="form1" runat="server">
        <div>
            <asp:ScriptManager ID="ScriptManager1" runat="server" />

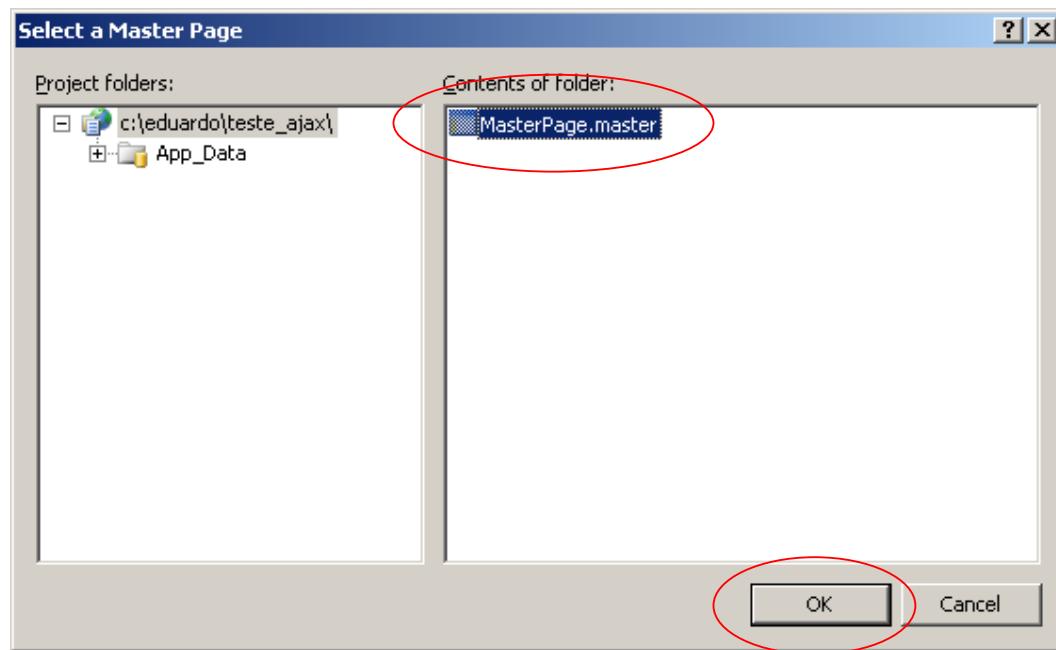
            <asp:Menu ID="Menu1" runat="server" BackColor="#B5C7DE" DynamicHorizontalOffset="2"
                Font-Names="Verdana" Font-Size="0.8em" ForeColor="#284E98" Orientation="Horizontal"
                StaticSubMenuIndent="10px">
                <StaticSelectedStyle BackColor="#507CD1" />
                <StaticMenuItemStyle HorizontalPadding="5px" VerticalPadding="2px" />
                <DynamicHoverStyle BackColor="#284E98" ForeColor="White" />
                <DynamicMenuStyle BackColor="#B5C7DE" />
                <DynamicSelectedStyle BackColor="#507CD1" />
                <DynamicMenuItemStyle HorizontalPadding="5px" VerticalPadding="2px" />
                <StaticHoverStyle BackColor="#284E98" ForeColor="White" />
                <Items>
                    <asp:MenuItem NavigateUrl("~/pagina.aspx" Text="Calculadora" Value="Calculadora">
                    </asp:MenuItem>
                </Items>
            </asp:Menu>
            <hr />

            <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
            </asp:ContentPlaceHolder>

            <hr />
            Aqui é o rodapé da página!
        </div>
    </form>
</body>
```

Agora vamos criar uma página que utilize a nossa master Page:





Agora, só para testar, vamos criar uma página com uma calculadora, por exemplo:

Valor 1:	<input type="text" value="10"/>
Valor 2:	<input type="text" value="7"/>
<input type="button" value="Calcular"/>	
Resultado:	<input type="text" value="17"/>

Código da página:

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
  CodeFile="pagina.aspx.cs" Inherits="pagina" Title="Untitled Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceholder1" runat="Server">
  <asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
      Valor 1:
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
      <br />
      Valor 2:
      <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
      <br />
      <asp:Button ID="Button1" runat="server" Text="Calcular" OnClick="Button1_Click" />
      <br />
      Resultado:
      <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
    </ContentTemplate>
  </asp:UpdatePanel>
</asp:Content>
```

Essa parte pode ser removida.

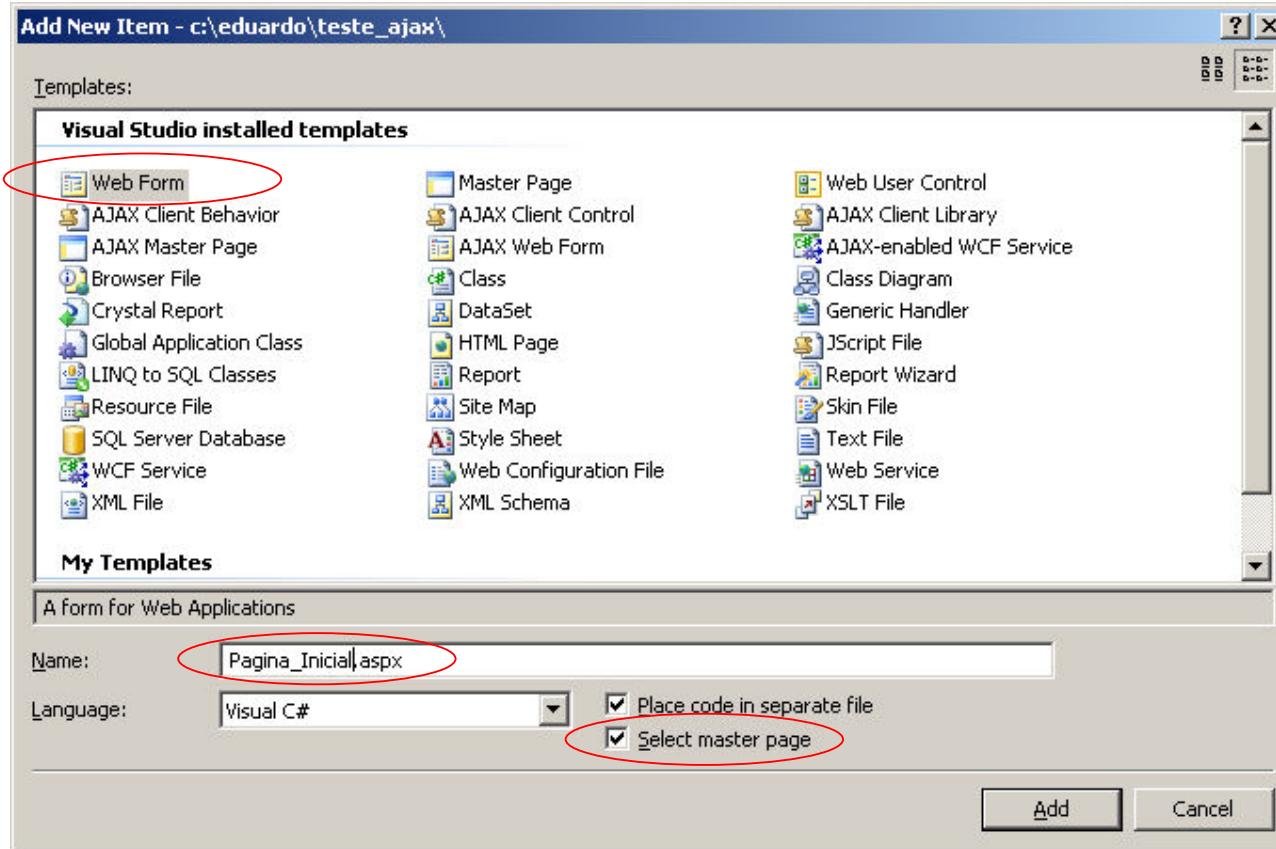
O seu código deve ser colocado entre as TAGS:

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceholder1"
  runat="Server">

</ContentTemplate>
```

Clique com o botão direito do mouse sobre a a pagina.aspx e defina-a como página inicial

Agora, vamos criar uma página inicial para o nosso site, só para não se a mesma que a calculadora! Pode ser um Web form mesmo.



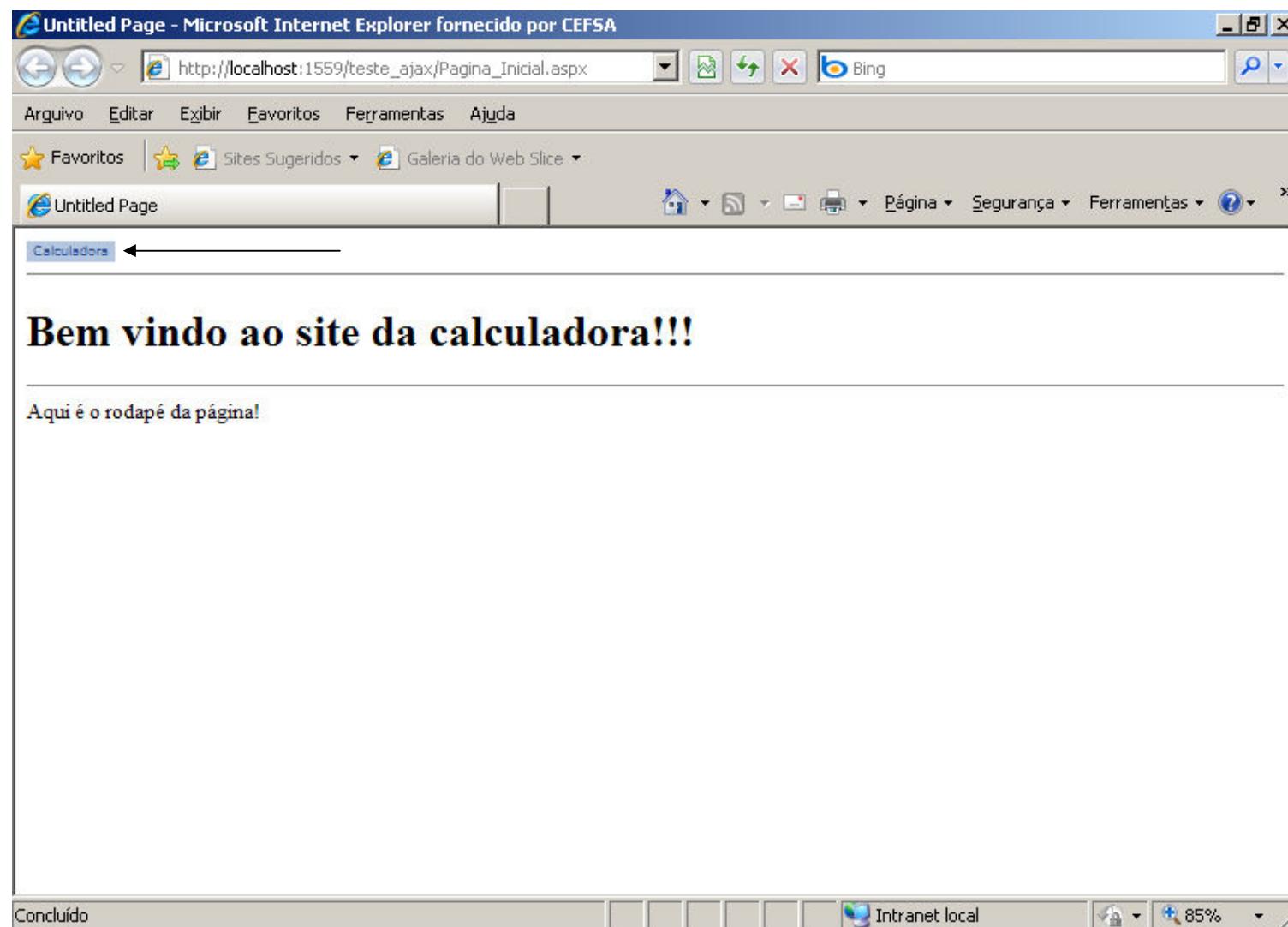
Código:

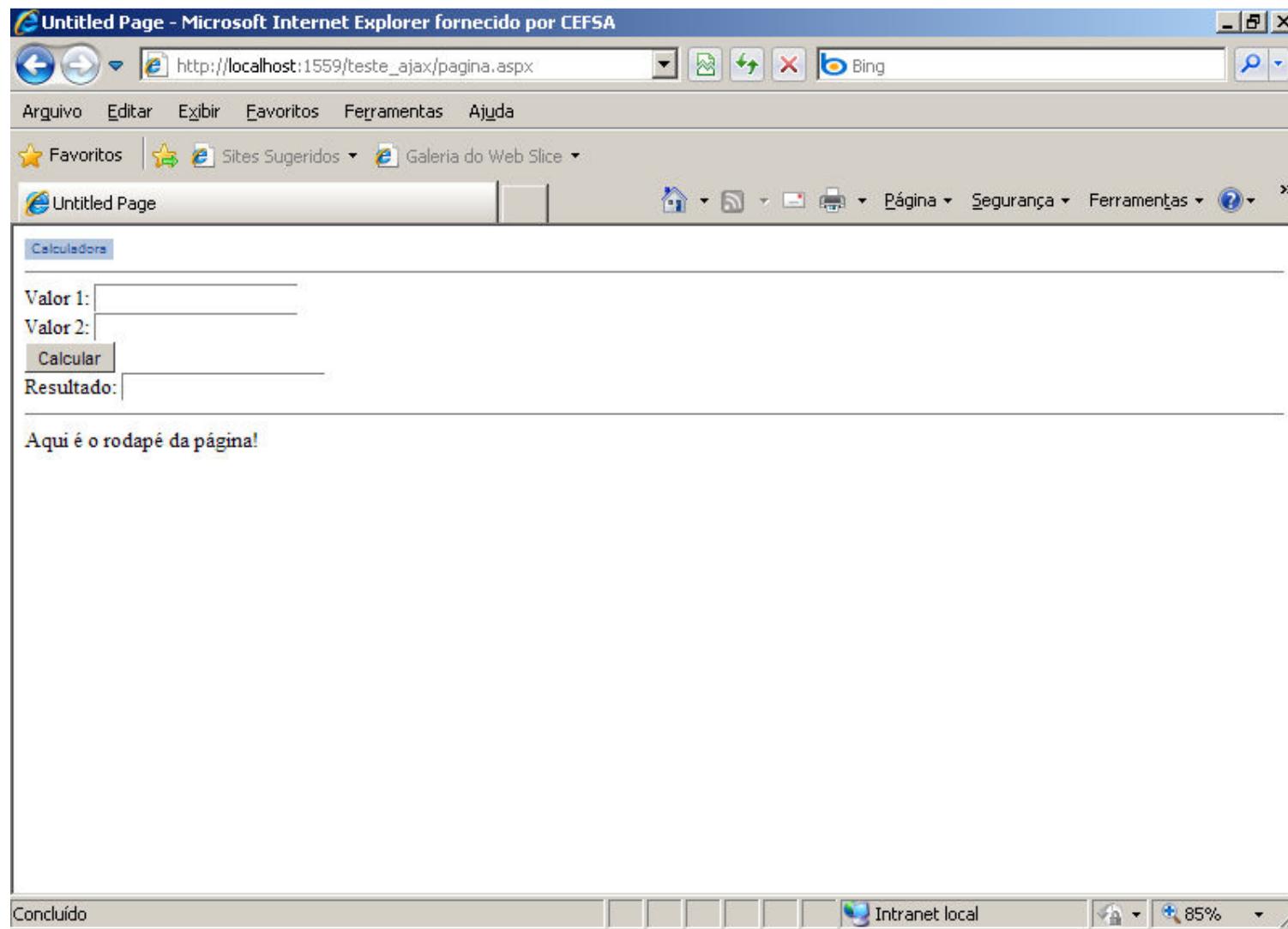
```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
CodeFile="Pagina_Inicial.aspx.cs" Inherits="Pagina_Inicial" Title="Untitled Page" %>

<asp:Content ID="Content1" ContentPlaceholderID="head" Runat="Server">
</asp:Content>

<asp:Content ID="Content2" ContentPlaceholderID="ContentPlaceholder1" Runat="Server">
<h1> Bem vindo ao site da calculadora!!! </h1>
</asp:Content>
```

Defina esta como a página Inicial e execute o programa.



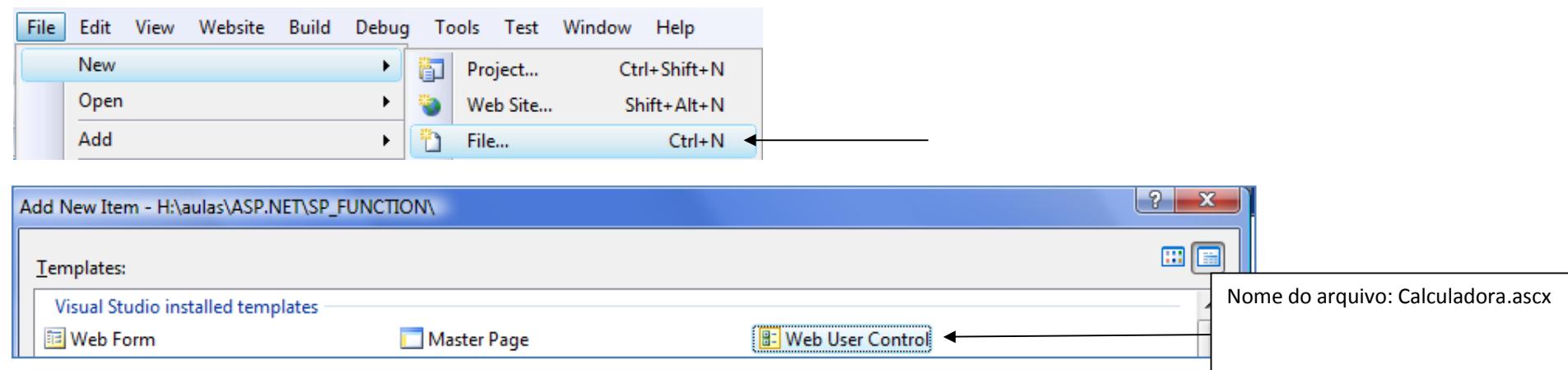


Web User Control

A idéia de um Web User Control é evitar a redundância no código e em componentes visuais, já que centraliza (como em um método) em um componente criado pelo usuário componentes visuais e código que se repetem com muita freqüência em uma aplicação. É possível criar propriedades, métodos e eventos em um Web User Control. As alterações em um Web User Control serão refletidas em todos os lugares onde ele foi utilizado.

Para exemplificar, vamos criar um Web User Control para simular uma calculadora simples. Primeiro, crie um novo WEBSITE.

Para criar um Web User Control no WebSite:



Valor 1:

Valor 2:

Calcular

Resultado:

Código Fonte do formulário:

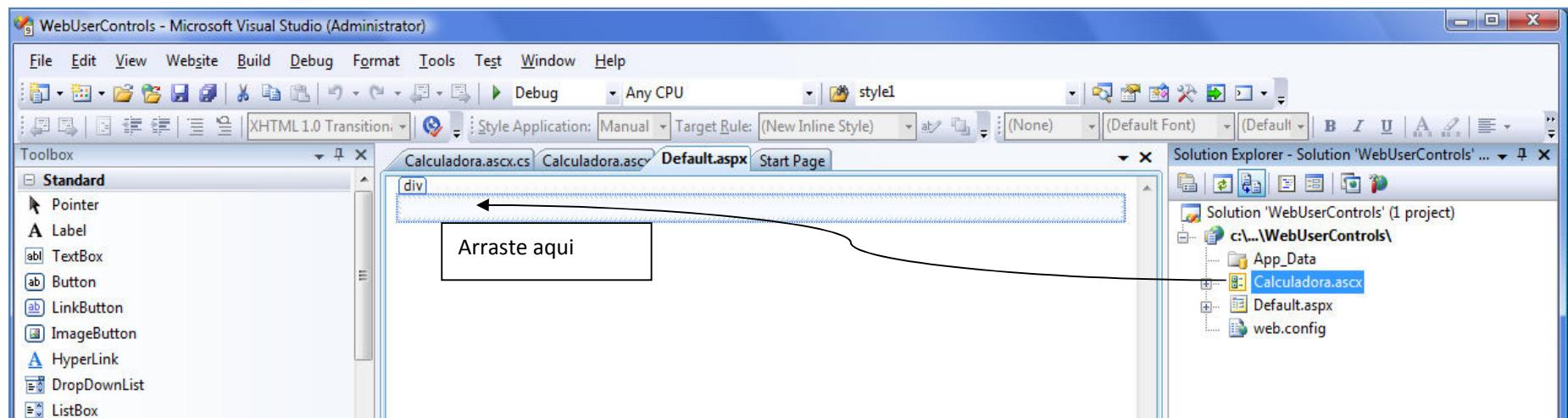
```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="Calculadora.ascx.cs"
Inherits="Calculadora" %>

Valor 1: <asp:TextBox ID="TextBoxValor1" runat="server"></asp:TextBox>
<br />
Valor 2: <asp:TextBox ID="TextBoxValor2" runat="server"></asp:TextBox>
<br />
<asp:Button ID="Button1" runat="server" Text="Calcular" />
<br />
Resultado: <asp:TextBox ID="TextBoxResultado" runat="server"></asp:TextBox>
```

O único objeto programado será o botão de calcular. No evento click coloque o código:

```
protected void Button1_Click(object sender, EventArgs e)
{
    try
    {
        double valor = Convert.ToDouble(textBoxValor1.Text) + Convert.ToDouble(textBoxValor2.Text);
        textBoxResultado.Text = valor.ToString("###,##0.00");
    }
    catch
    {
        textBoxResultado.Text = "Não foi possível calcular";
    }
}
```

Para utilizar o User Control, deixe o seu formulário (Default.aspx) no modo de design (como na imagem abaixo) e arraste o  **Calculadora.ascx** para dentro do seu form:



Após adicionar, o código do formulário Default.aspx ficará assim:

```
<body>
<form id="form1" runat="server">
<div>
    <uc1:Calculadora ID="Calculadora1" runat="server" />
</div>
</form>
</body>
```

Já é possível testar. Pressione F5 e veja como ficou.

Criando um método no Web User Control

Caso queira criar algum método no Web User Control, como por exemplo um método para limpar os campos que seja acessível de fora (publico), faça assim:

Coloque o método abaixo no User control:

```
public void LimparCampos()
{
    TextBoxValor1.Text = "";
    TextBoxValor2.Text = "";
    TextBoxResultado.Text = "";
}
```

Você pode executar o método dentro do próprio User Control ou da tela que está usando-o. Para executar este método na página Default.aspx:

[Default.aspx](#):

The diagram illustrates the interaction between a Web User Control and a page. On the left, a screenshot of a web page shows two text input fields labeled 'Valor 1:' and 'Valor 2:', a 'Calcular' button, and a result text box labeled 'Resultado:'. Below these controls is a button labeled 'Limpar campos do web user control'. An arrow points from the 'Limpar campos do web user control' button to a code block on the right. This code block contains a protected event handler for the 'Button1_Click' event of the 'Button1' control, which calls the 'LimparCampos' method of the 'Calculadora1' user control.

```
protected void Button1_Click(object sender, EventArgs e)
{
    Calculadora1.LimparCampos();
}
```

Sessão

Fontes:

<http://msdn.microsoft.com/pt-br/library/ms178581.aspx>

<http://msdn.microsoft.com/pt-br/library/87069683%28VS.80%29.aspx>

Introdução

O estado de sessão do ASP.NET permite que você armazene e recupere valores para um usuário quando o usuário navega as diferentes páginas ASP.NET que compõem um aplicativo da Web. HTTP é um protocolo sem estado, o que significa que seu servidor Web trata cada solicitação HTTP para uma página como uma solicitação independente; por padrão, o servidor mantém nenhum conhecimento dos valores de variáveis usados durante solicitações anteriores. Como resultado, criar Aplicativos da Web que precisam manter algumas informações do estado cross-request (aplicativos que implementam carrinhos de compras, rolagem de dados e assim por diante) pode ser um desafio. O estado de sessão do ASP.NET identifica solicitações recebidas pelo mesmo navegador durante um intervalo de tempo limitado como uma sessão, e oferece uma maneira de persistir os valores de variáveis pela duração dessa sessão.

O estado de sessão do ASP.NET é por padrão ativado para todos os aplicativos ASP.NET. Variáveis de estado da sessão ASP.NET são facilmente definidos e recuperados usando a propriedade [Session](#) que armazena valores de variáveis de sessão como uma coleção indexada por nome.

Por exemplo, o seguinte exemplo de código cria as variáveis de sessão FirstName e LastName para representar o primeiro nome e o sobrenome de um usuário e define-os para valores recuperados de controles [TextBox](#).

```
Session["FirstName"] = FirstNameTextBox.Text;
```

```
Session["LastName"] = LastNameTextBox.Text;
```

Na sessão podem ser armazenadas variáveis e objetos.

Modos de estado de sessão

Estado de sessão do ASP.NET suporta várias opções diferentes de armazenamento para dados da sessão. Cada opção é identificada por um valor na enumeração [SessionStateMode](#). A lista a seguir descreve os modos de estado de sessão disponíveis:

- Modo [InProc](#), que armazena estado de sessão na memória do servidor Web. Este é o padrão.
- Modo [StateServer](#), que armazena o estado de sessão em um processo separado chamado de serviço de estado ASP.NET. Isso garante que o estado da sessão é preservado se o aplicativo da Web for reiniciado e também disponibiliza o estado da sessão para vários servidores Web.

- Modo [SQLServer](#) armazena o estado de sessão em um banco de dados SQL Server. Isso garante que o estado da sessão é preservado se o aplicativo da Web for reiniciado e também disponibiliza o estado da sessão para vários servidores Web.
- Modo [Custom](#), que permite que você especifique um provedor personalizado de armazenamento.
- Modo [Off](#), que desativa estado de sessão.

Você pode especificar qual modo você deseja que seu estado de sessão ASP.NET use, atribuindo valores de enumeração [SessionStateMode](#) ao atributo mode do elemento [sessionState](#) no arquivo web.config do aplicativo. Modos diferentes de [InProc](#) e [Off](#) requerem parâmetros adicionais.

Definindo o tempo de vida da sessão.

O tempo é definido em minutos. No exemplo abaixo, o tempo é de 60 min. A linha pode ser inserida em qualquer parte do <system.web> dentro do web.config.

```
<system.web>
    <sessionState timeout="60"/>
</system.web>
```

ViewState

Fonte: <http://www.techtips.com.br/programacao/delphi/aspnet/entendendo-o-viewstate-em-aspnet/>

Específico da tecnologia ASP.NET, o ViewState é uma string armazenada em um campo oculto, dentro do HTML gerado para a sua página, que permite guardar o estado de variáveis e objetos entre requisições.

Cada componente colocado em uma página ASP.Net vai salvar seu estado no ViewState, para que cada vez que acontecer um postback esse componente possa voltar a ser exibido exatamente como estava antes, sem precisar fazer nova inicialização ou trazer os dados novamente do banco de dados, e assim por diante. Isso é prático, mas pode deixar o ViewState incrivelmente grande e aumentar bastante o tamanho da sua página – e o tempo de carregamento por consequência.

Você também pode armazenar variáveis no viewstate, assim como faz com sessões, com o intuído de não perder o valor delas entre postbacks. Ex:

```
ViewState["MinhaVariavel"] = minhavariavel;
```

Exemplo do código gerado pelo viewstate na página enviada ao navegador:

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="/wEPDwUKLTg0OTIzODQwNWRk+YMQe4o+Sur+bsDmu/qsJLMvnOM=" />
```

Request e Response

Os objetos Response e Request representam a informação chegando no servidor Web a partir do navegador (Request) e a informação saindo do servidor para o navegador (Response). O objeto Request representa o objeto input e o objeto Response representa o objeto output.

Tela de Login

Para criar uma tela de login, adicione os seguintes componentes:

The screenshot shows a web form titled "Tela de Login". It contains two text input fields labeled "Usuário..." and "Senha...", a "Fazer Login" button, and a label at the bottom labeled "[LabelMensagem]".

```
<body>
    <form id="form1" runat="server">
        <div>
            <h1> Tela de Login</h1>
            Usuário...:<asp:TextBox ID="TextBoxUsuario" runat="server"></asp:TextBox>
            <br />
            Senha.....:<asp:TextBox ID="TextBoxSenha" runat="server" TextMode="Password"></asp:TextBox>
            <br />
            <asp:Button ID="ButtonOK" runat="server" Text="Fazer Login" />
            <hr />
            <asp:Label ID="LabelMensagem" runat="server" Text=""></asp:Label>
        </div>
    </form>
</body>
```

O código do botão será o seguinte:

```
protected void ButtonOK_Click(object sender, EventArgs e)
{
    // após validar se o usuário existe...
    if (TextBoxUsuario.Text.Equals("TESTE") && TextBoxSenha.Text.Equals("123"))
    {
        Session["logado"] = true; // essa será a variável que irá guardar se o usuário está logado ou não.
        Session["erro_de_login"] = null; // não houve erro de login, vamos usar essa variável no page_load...
        Response.Redirect("~/Paginal.aspx"); // redireciona para uma página qualquer....
    }
    else
    {
        LabelMensagem.Text = "Usuário/Senha inválido!";
    }
}
```

Em todas as outras telas do sistema que se deseja controlar o acesso, deve-se checar se o usuário efetuou o login. Para tanto, no evento Page_Load da página, adicione o seguinte código:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["logado"] == null) // se a variável de sessão não existe ou não é true, significa que o usuário não está logado.
    {
        Session["erro_de_login"] = true; // vamos criar uma variável de sessão para indicar que houve erro de login
        Response.Redirect("~/login.aspx");
    }
}
```

Observando-se o código acima, veremos que o sistema redireciona para a tela de login caso o usuário não esteja logado. Para exibir uma mensagem na tela de login, indicando que o usuário tentou acessar uma tela sem ter feito a autenticação, adicione o seguinte código no evento Page_Load na tela de login:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["erro_de_login"] != null)
    {
        LabelMensagem.Text = "Para acessar qualquer área do site é preciso estar autenticado.";
        LabelMensagem.ForeColor = System.Drawing.Color.Red;
        Session["erro_de_login"] = null; // vamos limpar a variável pois já exibimos o erro...
    }
}
```

Para fazer o teste, tente acessar a pagina1.aspx sem ter feito login.

Exibindo mensagens com o ScriptManager

```
ScriptManager.RegisterClientScriptBlock(Page, Page.GetType(), "Erro", "alert('Arquivo não existe!');", true);
```

Cookies

FONTE: <http://msdn.microsoft.com/pt-br/library/ms178194%28VS.80%29.aspx>

Os cookies fornecem um meio em aplicativos da Web para armazenar informações específicas do usuário. Por exemplo, quando um usuário visita seu site, você pode usar cookies para armazenar as preferências do usuário ou outras informações. Quando o usuário visita seu site Web outra vez, o aplicativo pode recuperar as informações que foram armazenadas antes.

Um cookie é um pequeno pedaço de texto que acompanha solicitações e páginas quando elas trafegam entre o servidor Web e o navegador. O cookie contém informações que o aplicativo Web pode ler sempre que o usuário visitar o site. Por exemplo, se um usuário solicita uma página do seu site e seu aplicativo envia não apenas uma página, mas também um cookie contendo a data e a hora, quando o navegador do usuário receber a página, ele também recebe o cookie, que ele armazena em uma pasta no disco rígido do usuário.

Depois, se o usuário solicitar uma página do seu site novamente, quando o usuário inserir a URL, o navegador procura no disco rígido local por um cookie associado com a URL. Se o cookie existe, o navegador envia o cookie ao seu site juntamente com a solicitação da página. Seu aplicativo pode então determinar a data e hora que o usuário visitou pela última vez o site. Você pode usar as informações para exibir uma mensagem para o usuário ou verificar uma data de validade.

Cookies estão associados com um site Web, não com uma página específica, portanto, o navegador e o servidor vão trocar informações de cookie independentemente de qual for a página que o usuário solicite de seu site. Como o usuário visita diferentes sites, cada site pode enviar um cookie para o navegador do usuário igualmente; o navegador armazena todos os cookies separadamente.

Cookies são usados para muitos propósitos, todos relativos a ajudar o site Web a lembrar os usuários. Por exemplo, um site realizando uma pesquisa pode usar um cookie simplesmente como um valor Booleano para indicar se um navegador do usuário já participou na votação, a fim de que o usuário não possa votar duas vezes.

O navegador é responsável por gerenciar cookies em um sistema de usuário. Cookies são enviados ao navegador através do objeto [HttpResponse](#) que expõe um coleção chamada [Cookies](#). Você pode acessar o objeto [HttpResponse](#) como a propriedade [Response](#) de sua classe [Page](#). Quaisquer cookies que você desejar enviar ao navegador deve ser adicionado a esta coleção. Quando criar um cookie, você especifica um [Name](#) e [Value](#). Cada cookie deve ter um nome exclusivo, de forma que ele possa ser identificado posteriormente quando lido pelo navegador. Devido aos cookies serem armazenados pelo nome, nomear dois cookies de forma igual irá fazer com que um seja sobreescrito.

Você pode também definir uma data e hora de validade do cookie. Cookies expirados são excluídos pelo navegador quando um usuário visita o site que gravou os cookies. A validade de um cookie deve ser definida durante o tempo que seu aplicativo considera que o valor do cookie seja válido. Para um cookie nunca expirar efetivamente, você pode definir a data de validade para ser 50 anos a partir de agora.

Os usuários podem limpar os cookies de seu computador a qualquer momento. Mesmo se você armazenar os cookies com longo prazo de validade, um usuário poderá decidir excluir todos os cookies, apagando as configurações que você possa ter armazenado em cookies.

Se você não definir a validade do cookie, o cookie é criado mas ele não é armazenado no disco rígido do usuário. Em vez disso, o cookie é mantido como parte das informações de sessão do usuário. Quando o usuário fecha o navegador, o cookie é descartado. Um cookie não-persistente como este é útil para informação que precisa ser armazenada por um curto período de tempo apenas ou que, por razões de segurança, não deve ser gravado no disco do computador cliente. Por exemplo, cookies não-persistentes são úteis se o usuário estiver trabalhando em um computador público, onde você não deseja gravar o cookie no disco.

Os cookies são normalmente limitados a 4096 bytes e você não pode armazenar mais que 20 cookies por site. Uma limitação de cookie que você pode encontrar é que os usuários podem configurar seu navegador para recusar cookies.

Exemplo:

Exemplo de criação e recuperação de cookies

Nome:

Idade:

↓

```
protected void Button2_Click(object sender, EventArgs e)
{
    HttpCookie aCookie = Request.Cookies["meuCookie"];

    if (aCookie != null) // testa se o cookie existe
    {
        TextBoxNome.Text = aCookie.Values["Nome"].ToString();
        TextBoxIdade.Text = aCookie.Values["Idade"].ToString();
    }
}
```

```
protected void Button1_Click(object sender, EventArgs e)
{
    HttpCookie aCookie = new HttpCookie("MeuCookie"); // cria o objeto para armazenar o cookie
    aCookie.Values["Nome"] = TextBoxNome.Text;
    aCookie.Values["Idade"] = TextBoxIdade.Text;
    aCookie.Values["DataUltimaVisita"] = DateTime.Now.ToString();
    aCookie.Expires = DateTime.Now.AddDays(1); // O cookie abaixo irá expirar em 1 dia.
    Response.Cookies.Add(aCookie); // salva o cookie
}
```

Você também pode usar a classe TimeSpan para criar uma data de expiração mais detalhada, especificando dias, horas, minutos e segundos. Ex:

```
aCookie.Expires = DateTime.Now + new TimeSpan(3, 6, 5, 7); // expira em 3 dias, seis horas, 5 minutos e 7 segundos.
```

Passagem de parâmetro entre telas - Método Get (QueryString) e Método Post

Parte do texto foi retirada de <http://www.htmlstaff.org/ver.php?id=1664>

O HyperText Transfer Protocol (Protocolo de Transferência de Hipertexto - HTTP) é o protocolo de comunicação utilizado para a troca de dados entre um navegador e um servidor web. É o protocolo de comunicação que você aciona quando digita um endereço no seu navegador: http://www...

É para isto que existem os métodos HTTP. Dois desses métodos, associados à transferência de dados de formulários, são muito importantes: o método GET e o método POST.

Método GET (QueryString)

O método GET permite passar parâmetros entre páginas através da URL. Este método expõe os valores passados na URL, o que pode ser perigoso em algumas situações. Há também uma limitação referente ao tamanho da url, que não pode ultrapassar 255 caracteres.

Após o nome da página, o primeiro parâmetro deve iniciar com uma (?) e os outros devem ser separados por um (&).

Exemplos:

<HTTP://localhost/minhapagina.aspx?codigo=36>

<HTTP://localhost/minhapagina.aspx?codigo=36&situacao=Ativo>

Para testar, vamos criar primeiro uma página (default.aspx) onde o usuário poderá informar até 2 valores que depois serão enviados a uma página chamada Pagina2.aspx.

Default.aspx:

The screenshot shows a Windows-style dialog box for 'Default.aspx'. It contains two text input fields: one labeled 'Valor 1:' and another labeled 'valor 2:'. Below the fields is a button labeled 'Enviar valores'.

```
<form id="form1" runat="server">
    Valor 1: <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <br />
    valor 2: <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
    <br />
    <br />
    <asp:Button ID="Button1" runat="server" Text="Enviar valores" OnClick="Button1_Click" />
</form>
```

Código do botão

Enviar valores

```
Response.Redirect("~/Pagina2.aspx?Valor1=" + TextBox1.Text + "&Valor2=" + TextBox2.Text);
```

Agora, vamos criar a página Pagina2.aspx que irá receber estes valores.

Valores recebidos por QueryString (método get)

valor 1:
valor 2:

```
<form id="form1" runat="server">
    Valores recebidos por QueryString (método get)<br />
    <br />
    Valor 1: <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <br />
    Valor 2: <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
</form>
```

O código abaixo deve ser colocado no evento PAGE_LOAD:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Request.Params["Valor1"] != null) // devemos testar se o parâmetro existe antes de usá-lo.
        TextBox1.Text = Request.Params["Valor1"].ToString();

    if (Request.Params["Valor2"] != null) // devemos testar se o parâmetro existe antes de usá-lo.
        TextBox2.Text = Request.Params["Valor2"].ToString();
}
```

Para testar, não se esqueça de colocar como página inicial a default.aspx.

Método Post

POST também é um método do HTTP e é acionado por meio de um formulário HTML através da diretiva method=post incluída na tag <form>.

No método Post, os dados de um formulário são transferidos para processamento sem aparecer na URL da página. Portanto, ele é mais seguro que o método GET pois não expõe os valores para o usuário, além de não ter a limitação de 255 caracteres.

Para acessar os dados de um formulário passado pelo método post, a forma é exatamente a mesma que a realizada no método GET, com a diferença que os objetos são acessados pelo nome.

Se na página2.aspx queremos acessar o TextBox1 que estava na página default.aspx, devemos colocar o seguinte código no evento Page_Load da página 2.aspx:

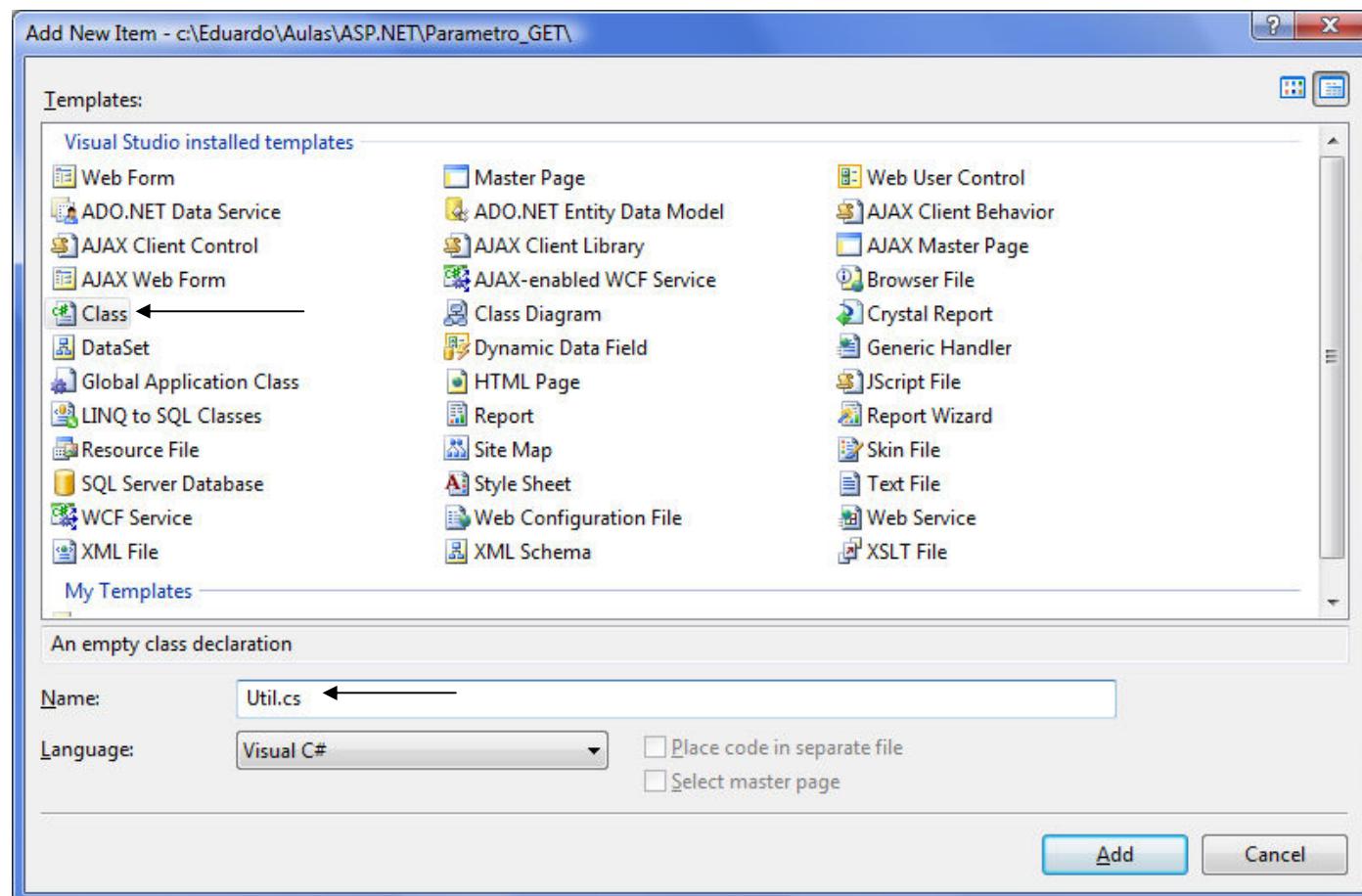
```
if (Request.Params["TextBox1"] != null) // devemos testar se o parâmetro existe antes de usá-lo.  
    TextBox1.Text = Request.Params["TextBox1"].ToString(); // o textBox1 da página2.aspx recebe o valor do textBox1 que estava em  
    default.aspx!
```

Onde o TextBox1 que está em Request representa o TextBox contido na página Default.aspx.

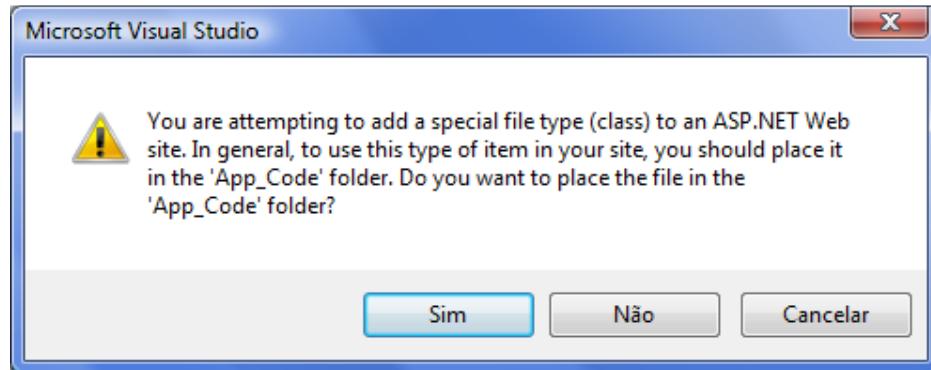
Criando uma classe de métodos estáticos

Em praticamente qualquer programa temos que utilizar diversos métodos que são comuns a várias partes da aplicação, como validação de datas, de números, exibição de mensagens, validação de CPF's, etc. Para estes casos, uma forma simples é centralizar estes métodos em uma única classe. Estes métodos também podem ser estáticos, sendo assim, a classe não precisa ser instanciada para que possamos utilizar seus métodos!

Para criar uma classe, Acesso o menu File->New->File:



Responda “SIM” a pergunta: (ele irá criar uma pasta chamada App_Code e irá colocar sua classe lá).



Ao clicarmos no botão Add, aparecerá uma janela perguntando se gostaríamos de adicioná-lo na pasta App_Code. A pasta App_Code é uma novidade do ASP.NET 2.0, sendo o local onde ficam os códigos-fontes comuns a toda aplicação. O conteúdo desta pasta é compilado automaticamente quando a aplicação é executada pela primeira vez.

Código da classe:

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

public class Util
{
    /// <summary>
    /// Recebe um valor string e testa se o valor é um número válido
    /// </summary>
    /// <param name="numero">Valor a ser testado</param>
    /// <returns>Verdadeiro se o valor é um número, ou false se não for.</returns>
    public static Boolean NumeroValido(string numero)
    {
        try
        {
            Convert.ToInt32(numero);
        }
    }
}
```

```
        return true;
    }
    catch
    {
        return false;
    }
}

/// <summary>
/// Exibe uma mensagem utilizando javascript
/// </summary>
/// <param name="msg">Texto da mensagem</param>
public static void Mensagem(string msg)
{
    Page pagina = HttpContext.Current.Handler as Page; // pega a página atual
    ScriptManager.RegisterClientScriptBlock(pagina, pagina.GetType(),
        "mensagem", "alert('" + msg + "');", true);
}
}
```

Para utilizar a classe:

```
Util.Mensagem("Dados gravados com sucesso!!!");
```

Tecnologias de conexão com BD ASP.NET

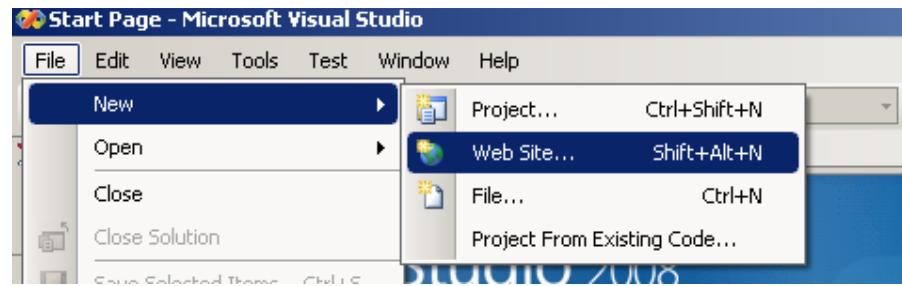
O Asp.net possui diversas tecnologias para acesso a bancos de dados. Sugiro ler estes 2 artigos que são uma excelente fonte de informação sobre este assunto.

<http://www.linhadecodigo.com.br/Artigo.aspx?id=1512>

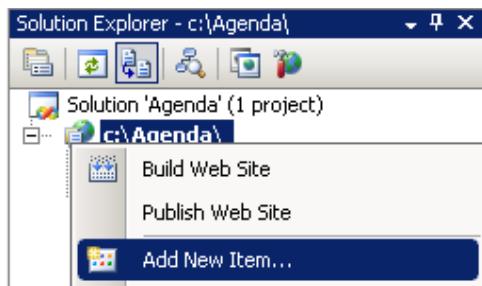
<http://www.linhadecodigo.com.br/artigo/1580/acesso-a-dados-em-aplicacoes-aspnet---conceitos-basicos---parte-3.aspx>

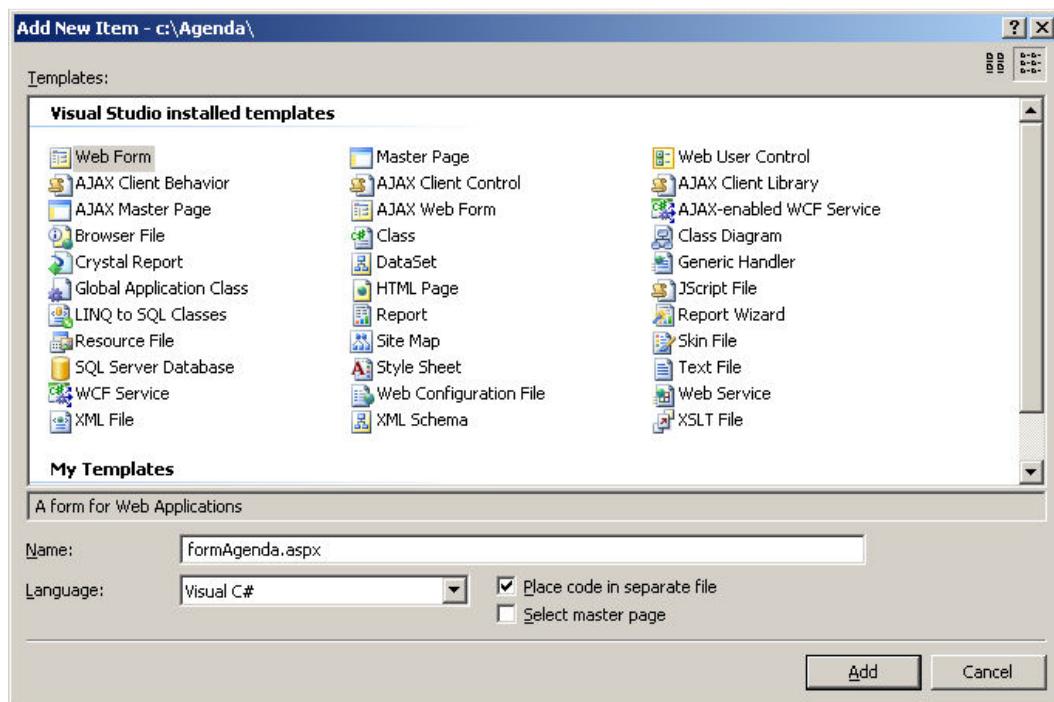
Criando um cadastro sem Formview

1º passo – Crie um projeto de website no asp.net :



2º passo – Apague a página default.aspx e crie uma nova página do tipo Web form:



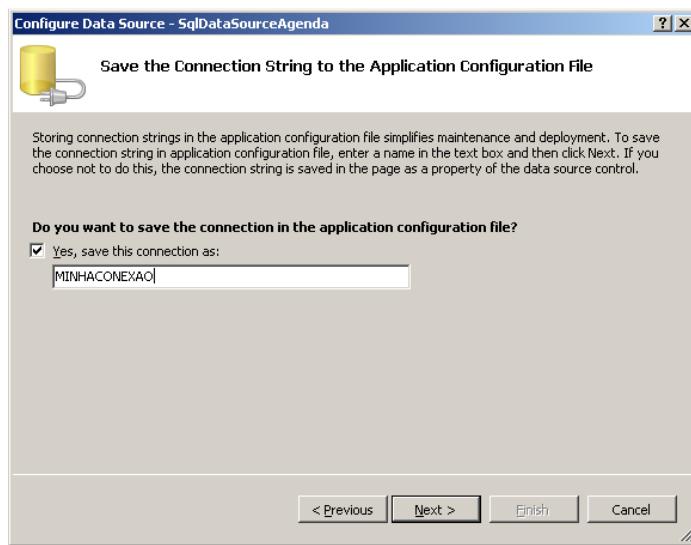


3º Passo – Colocar um gridview no formulário e configurar a conexão dele. Não esqueça de dar um nome para a conexão que será salva no WEB.config:

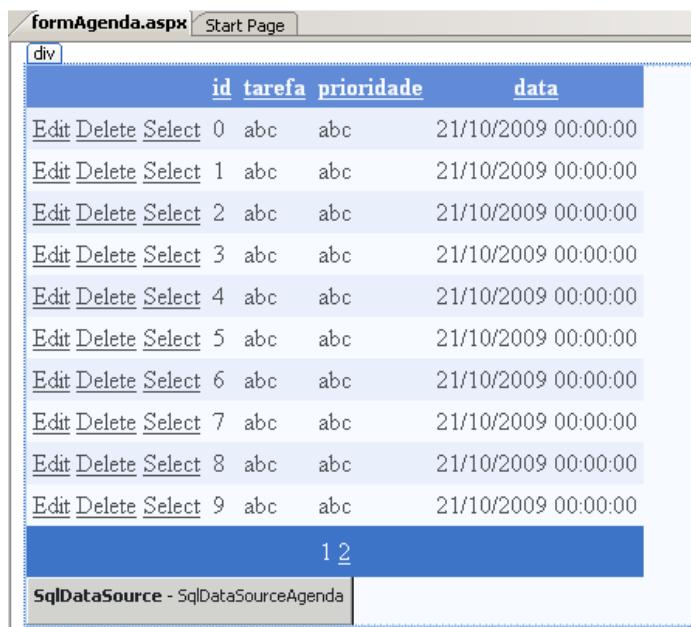
Não esqueça de criar a tabela no sql server!!!! O campo id é auto-incremento!!!!!!

Table - dbo.agenda		
Column Name	Data Type	Allow Nulls
id	int	<input type="checkbox"/>
tarefa	varchar(50)	<input checked="" type="checkbox"/>
prioridade	char(1)	<input checked="" type="checkbox"/>
data	datetime	<input checked="" type="checkbox"/>

Salve a conexão. A string de conexão será salva no web.config com o nome MINHACONEXAO:



Sua tela deverá ficar parecida com esta:

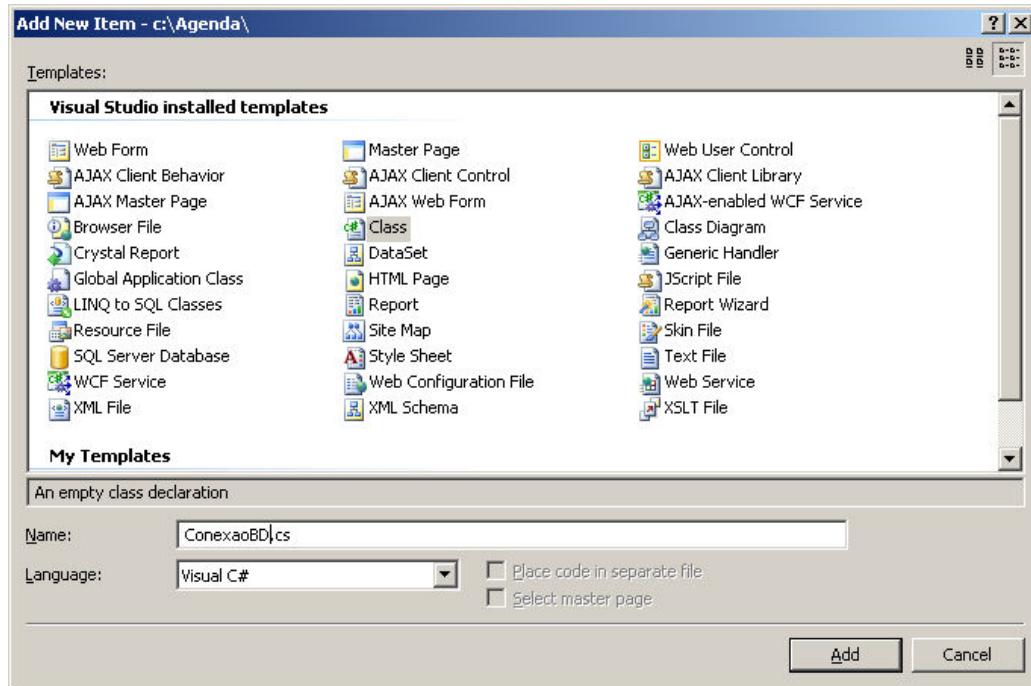


A screenshot of the 'formAgenda.aspx' page. The page title is 'formAgenda.aspx' and there is a 'Start Page' link. The main content is a table with the following data:

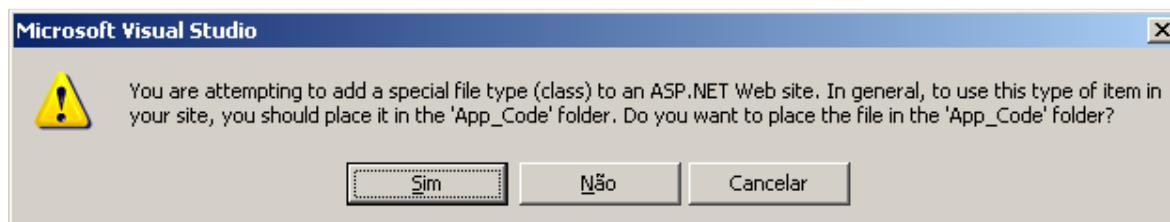
	<u>id</u>	<u>tarefa</u>	<u>prioridade</u>	<u>data</u>
Edit	Delete	Select	0 abc	21/10/2009 00:00:00
Edit	Delete	Select	1 abc	21/10/2009 00:00:00
Edit	Delete	Select	2 abc	21/10/2009 00:00:00
Edit	Delete	Select	3 abc	21/10/2009 00:00:00
Edit	Delete	Select	4 abc	21/10/2009 00:00:00
Edit	Delete	Select	5 abc	21/10/2009 00:00:00
Edit	Delete	Select	6 abc	21/10/2009 00:00:00
Edit	Delete	Select	7 abc	21/10/2009 00:00:00
Edit	Delete	Select	8 abc	21/10/2009 00:00:00
Edit	Delete	Select	9 abc	21/10/2009 00:00:00

At the bottom of the table, there are two buttons: '1' and '2'. The footer of the page shows 'SqlDataSource - SqlDataSourceAgenda'.

4º Passo – Criar uma classe para conexão com BD



Responda SIM para a pergunta:



Adicione os seguintes namespace's:

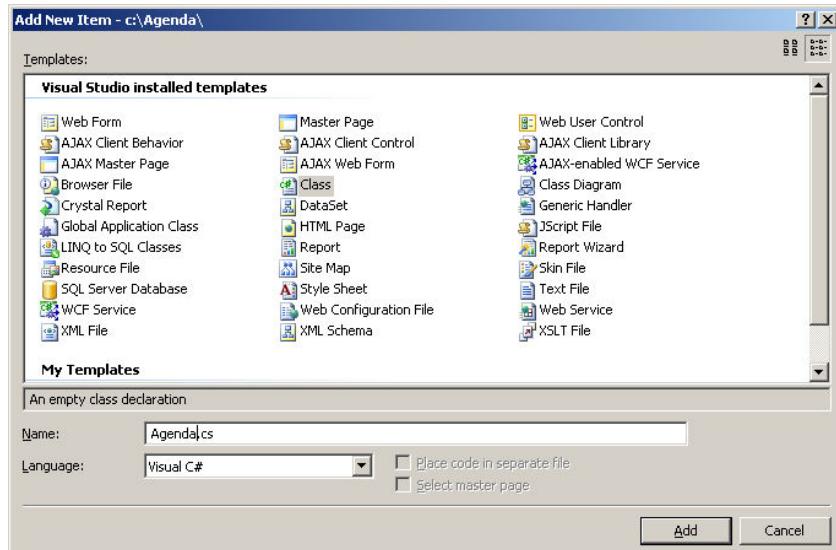
```
using System.Data.SqlClient;
using System.Web.Configuration;
```

O código da classe deverá ser:

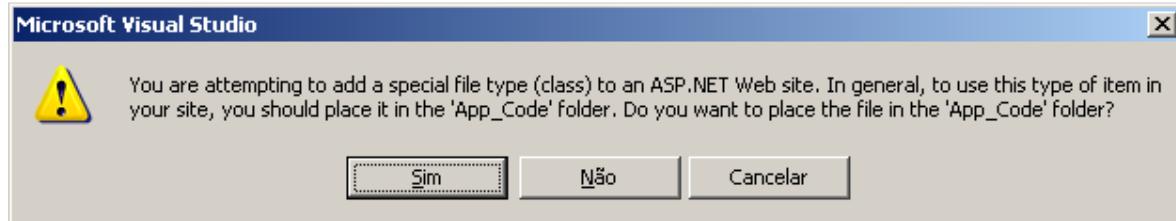
```
public class ConexaoBD
{
    public static SqlConnection getConexao()
    {
        string strconexao = WebConfigurationManager.ConnectionStrings["MINHACONEXAO"].ToString();

        SqlConnection conexao = new SqlConnection(strconexao);
        conexao.Open();
        return conexao;
    }
}
```

5º Passo: Agora vamos mapear a tabela Agenda em uma classe do mesmo nome:



Responda novamente "sim" :



Para cada campo da tabela, crie uma variável privada, e um método GET e SET (publico) para acesso. O código da classe ficará assim:

```
public class Agenda
{
    private int id;
    private string tarefa;
    private char prioridade;
    private DateTime data;

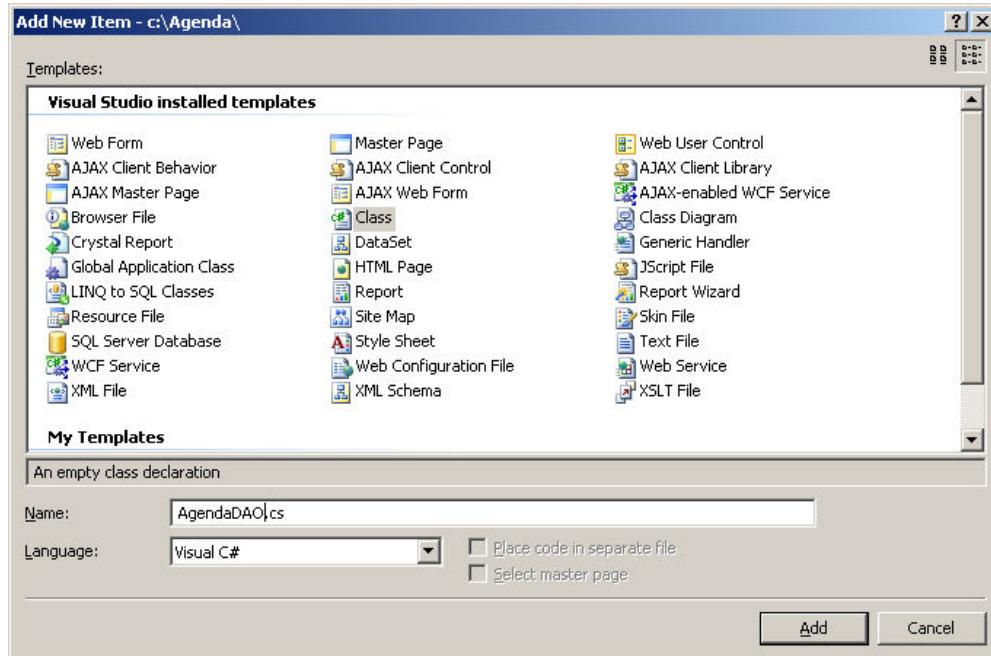
    public void setId(int id) { this.id = id; }
    public int getId() { return id; }

    public void setTarefa(string tarefa) { this.tarefa = tarefa; }
    public string getTarefa() { return tarefa; }

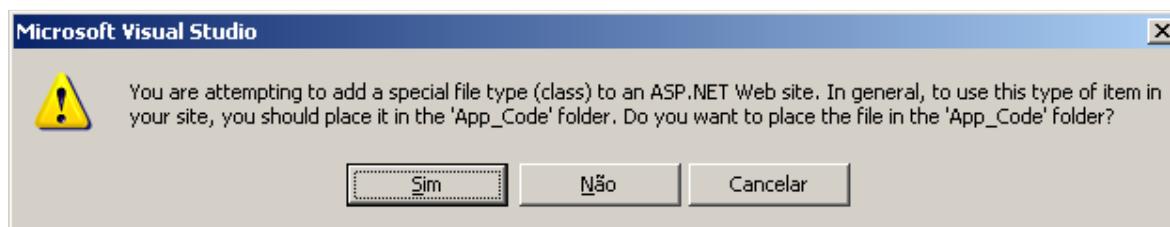
    public void setPrioridade(char prioridade) { this.prioridade = prioridade; }
    public char getPrioridade() { return prioridade; }

    public void setData(DateTime data) { this.data = data; }
    public DateTime getData() { return data; }
}
```

6º Passo: Criar a classe DAO (Data Access Object) para efetuar as tarefas de inclusão, alteração e exclusão:



Novamente, responda “sim” :



Colocar o namespace:

```
using System.Data.SqlClient;
```

O código da classe será o seguinte:

```
public static void Inserir(Agenda a)
{
    // set dateformat dmy; este comando serve para alterar a forma como o SQL Server entende o formato de
    data

    string sql = String.Format("set dateformat dmy; " +
        "insert into agenda(tarefa, prioridade, data)" +
        "values ('{0}', '{1}', '{2}')", a.getTarefa(),
        a.getPrioridade(), a.getData());
    SqlConnection conexao = ConexaoBD.getConexao();

    SqlCommand comando = new SqlCommand(sql, conexao);
    comando.ExecuteNonQuery();
    conexao.Close();
}

public static void Altera(Agenda a)
{
    string sql = string.Format("set dateformat dmy; update agenda set " +
        "tarefa = '{0}', prioridade = '{1}', data = '{2}' " +
        "where id = {3}", a.getTarefa(),
        a.getPrioridade(), a.getData(), a.getId());

    SqlConnection conexao = ConexaoBD.getConexao();

    SqlCommand comando = new SqlCommand(sql, conexao);
    comando.ExecuteNonQuery();
    conexao.Close();
}

public static void Apaga(int id)
{
```

```
string sql = "delete from agenda where id = " + id.ToString();  
SqlConnection conexao = ConexaoBD.getConexao();  
SqlCommand comando = new SqlCommand(sql, conexao);  
comando.ExecuteNonQuery();  
conexao.Close();  
}
```

7º Passo: Alterar o formulário para que possamos fazer inclusão, alteração e exclusão:

Crie um form parecido com este na mesma tela onde você colocou o gridview:

The screenshot shows a browser window with the title bar "formAgenda.asp". Below the title bar is a navigation menu with links: "Edit", "Delete", "Select", "abc", "abc", and "21/10/2009 00:00:00". The main content area displays two rows of data from a GridView. The first row has columns: "Edit", "Delete", "Select", "8", "abc", "abc", and "21/10/2009 00:00:00". The second row has columns: "Edit", "Delete", "Select", "9", "abc", "abc", and "21/10/2009 00:00:00". Below the GridView is a blue horizontal bar with the numbers "1 2". At the bottom of the page is a "SqlDataSource - SqlDataSourceAgenda" control.

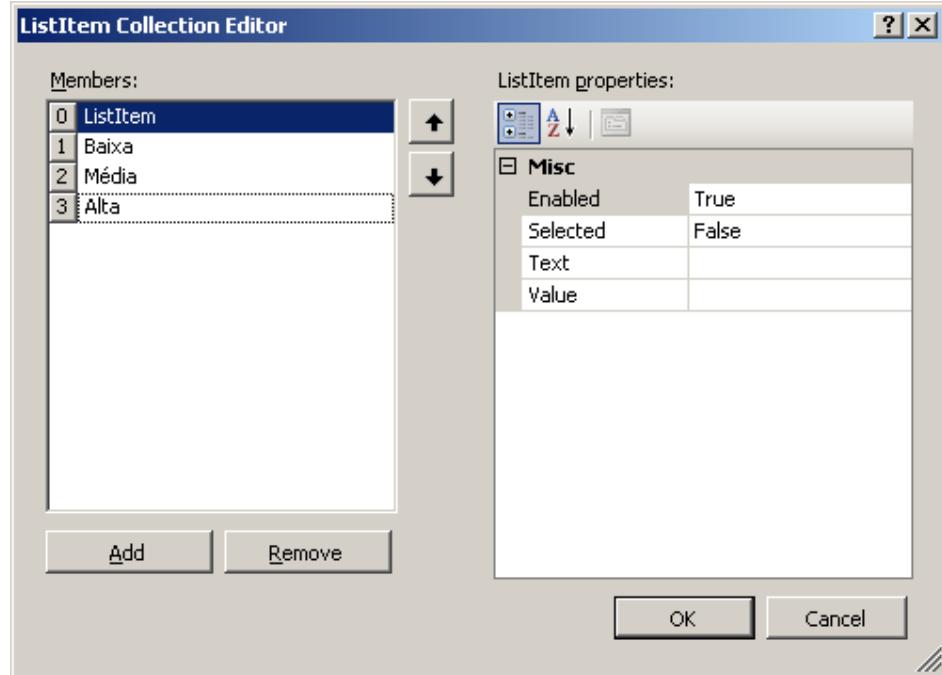
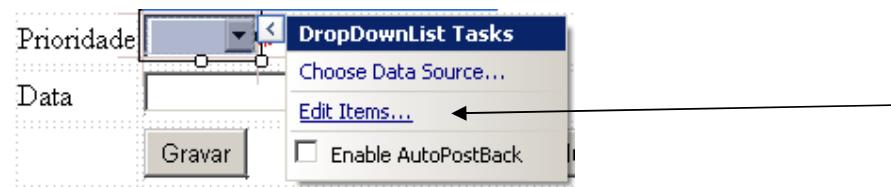
- Error message 1.
- Error message 2.

The screenshot shows an ASP.NET form with the following fields:

- Id**: A text input field.
- Tarefa**: A text input field with a red asterisk (*) indicating it is required.
- Prioridade**: A dropdown list with a red asterisk (*) indicating it is required.
- Data**: A date input field with a red asterisk (*) indicating it is required.

Below the form are three buttons: "Gravar", "Alterar", and "Excluir". The entire form is enclosed in a dashed border.

Coloque nos itens do campo prioridade:



O Text e Value do primeiro item estão vazios.
(um espaço em branco)

Text = Baixa Value = B

Text = Média Value = M

Text = Alta Value = A

Código do botão gravar:

```
protected void btnGravar_Click(object sender, EventArgs e)
{
    Agenda agenda = new Agenda();

    agenda.setTarefa(TextBoxTarefa.Text);
    agenda.setPrioridade(DropDownListPrioridade.SelectedValue[0]);
    agenda.setData(Convert.ToDateTime(TextBoxData.Text));
    AgendaDAO.Inserir(agenda);

    GridView1.DataBind();
}
```

Código do botão Alterar:

```
protected void btnAltera_Click(object sender, EventArgs e)
{
    Agenda agenda = new Agenda();

    agenda.setId(Convert.ToInt16(TextBoxId.Text));
    agenda.setTarefa(TextBoxTarefa.Text);
    agenda.setPrioridade(DropDownListPrioridade.SelectedValue[0]);
    agenda.setData(Convert.ToDateTime(TextBoxData.Text));
    AgendaDAO.Alterar(agenda);

    GridView1.DataBind();
}
```

Código do botão Excluir:

```
protected void BtnExclui_Click(object sender, EventArgs e)
{
    AgendaDAO.Apaga(Convert.ToInt16(TextBoxId.Text));
    GridView1.DataBind();
}
```

Ao executar (F5), selecione a opção abaixo e clique OK.



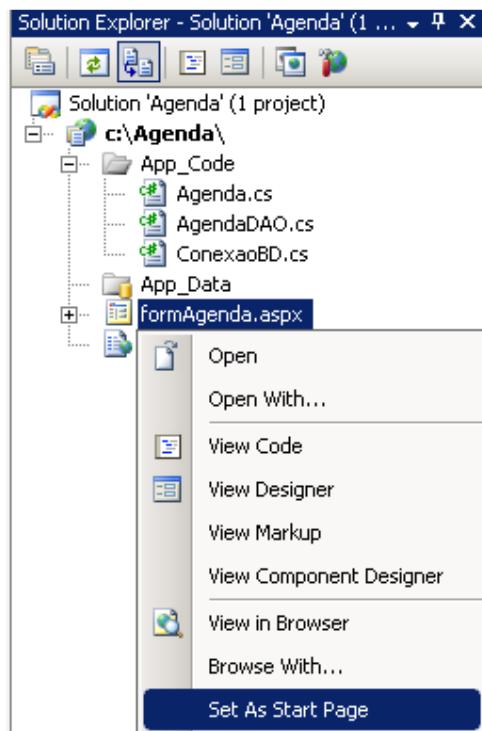
Para incluir, preencha todos os campos, menos o ID. Para alterar, digite o ID e os outros dados e clique em alterar. Para apagar, digite apenas o ID.

Se ao executar (F5) aparecer a tela abaixo, é porque você não definiu a página inicial (veja como abaixo)

Directory Listing -- /Agenda/

Wednesday, October 21, 2009 08:25 PM	<dir>	App_Code
Wednesday, October 21, 2009 07:46 PM	<dir>	App_Data
Wednesday, October 21, 2009 08:25 PM	6,174	formAgenda.aspx
Wednesday, October 21, 2009 08:20 PM	1,426	formAgenda.aspx.cs
Wednesday, October 21, 2009 08:22 PM	8,315	web.config

Version Information: ASP.NET Development Server 9.0.0.0



8º - Criando a consulta – Quando o usuário clicar no linkbutton “select” iremos preencher os campos da tela com os valores do registro selecionado.

Crie o seguinte método na classe AgendaDAO:

Este método irá receber um id e irá retornar um objeto agenda com os campos preenchidos com o que havia na tabela. Se o registro não existir na tabela, será devolvido null.

```
public static Agenda getAgenda(int id)
{
    string sql = "select * from agenda where " +
        "id = " + id.ToString();
    SqlConnection conexao = ConexaoBD.getConexao();
    SqlCommand comando = new SqlCommand(sql, conexao);
    SqlDataReader reader = comando.ExecuteReader();

    if (reader.Read() == true) // verifica se há registro, lê e armazena em buffer
    {
        Agenda agenda = new Agenda();
        agenda.setId(Convert.ToInt16(reader["id"].ToString()));
        agenda.setTarefa(reader["tarefa"].ToString());
        agenda.setPrioridade(Convert.ToChar(reader["prioridade"].ToString()));
        agenda.setData(Convert.ToDateTime(reader["data"].ToString()));

        conexao.Close();
        return agenda;
    }
    else
    {
        conexao.Close();
        return null;
    }
}
```

Transforme a coluna do gridview que tem os comandos “edit select delete” em um Templatefield:

The screenshot shows a browser window with several tabs open. The active tab is 'formAgenda.aspx'. A context menu, titled 'GridView Tasks', is displayed over a 'GridView' control. The 'Edit Columns...' option is highlighted with a red arrow. The grid view itself contains four rows of data with columns labeled 'id', 'tarefa', 'prioridade', and 'data'. Each row has three links: 'Edit', 'Select', and 'Delete'.

The screenshot shows the 'Fields' dialog box. In the 'Available fields' list, 'CommandField' is selected and highlighted with a red arrow. In the 'Selected fields' list, 'CommandField' is also present and highlighted with a red arrow. The 'CommandField properties' section shows various properties like 'DeleteText' (Delete), 'EditText' (Edit), and 'HeaderText'. At the bottom of the dialog, there is a link 'Convert this field into a TemplateField' with a red arrow pointing to it. The 'OK' and 'Cancel' buttons are visible at the bottom right.

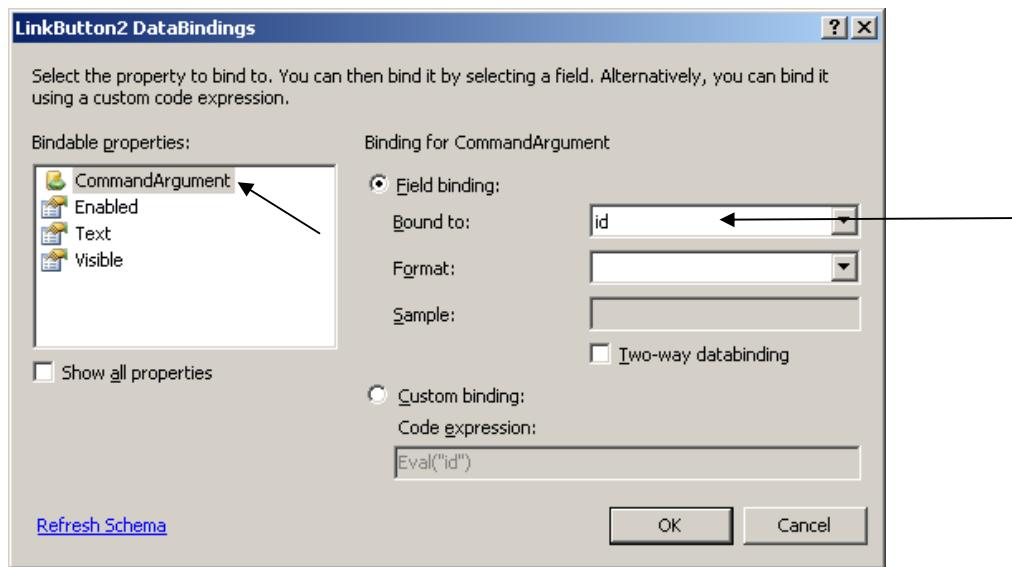
Agora edit o templatefield:

The screenshot shows a Windows context menu titled "GridView Tasks" open over a Microsoft ASP.NET GridView control. The menu includes options like "Auto Format...", "Choose Data Source:" (set to "SqlDataSourceAgenda"), "Configure Data Source...", "Refresh Schema", "Edit Columns...", "Add New Column...", "Move Column Left", "Remove Column", and several checkboxes for enabling features: "Enable Paging" (checked), "Enable Sorting" (checked), "Enable Editing" (unchecked), "Enable Deleting" (unchecked), and "Enable Selection" (unchecked). At the bottom of the menu is a "Edit Templates..." button.

id	tarefa	prioridade	data
0	abc	abc	21/10/2009 00:00:00
1	abc	abc	21/10/2009 00:00:00
2	abc	abc	21/10/2009 00:00:00
3	abc	abc	21/10/2009 00:00:00
4	abc	abc	21/10/2009 00:00:00
5	abc	abc	21/10/2009 00:00:00
6	abc	abc	21/10/2009 00:00:00
7	abc	abc	21/10/2009 00:00:00
8	abc	abc	21/10/2009 00:00:00
9	abc	abc	21/10/2009 00:00:00

Deixe apenas o select e na opção Edit DataBindings:

The screenshot shows the ASP.NET Properties window for a LinkButton control named "LinkButton2". The "Select" button is highlighted. Below it, the "Edit DataBindings..." button is visible, indicated by a red arrow.



Nas propriedades do linkbutton **Select**, programe o evento **Command**:



Código do evento Command:

```
// pega o id selecionado
int id = Convert.ToInt16(e.CommandArgument.ToString());

// lê a agenda selecionada
Agenda agenda = AgendaDAO.getAgenda(id);

// preencher os campos da tela
TextBoxId.Text = agenda.getId().ToString();
TextBoxData.Text = agenda.getData().ToString();
TextBoxTarefa.Text = agenda.getTarefa();
DropDownListPrioridade.SelectedValue = agenda.getPrioridade().ToString();
```

Utilizando o componente MultiView para organizar a tela

1º - Coloque um botão para inserir novos registros logo abaixo do grid:

Nome do botão: buttonNovo - Texto: "Novo registro"

O código C# do botão iremos colocar depois.

2º - Vamos fazer o seguinte: Vamos exibir apenas uma parte da tela por vez. Ou vamos exibir o grid, ou vamos exibir o formulário. Quando o usuário entrar para criar um novo registro ou alterá-lo, vamos exibir apenas o form. Caso contrário, iremos exibir apenas o grid. Note que isso não é uma regra. É apenas uma forma de se criar um cadastro, uma opção. Vamos aproveitar para ver como funciona um novo componente: o MultiView.



Clique no modo “Source” para ver o código do formulário e, insira o trecho de código em amarelo no local indicado:
(observe que o código que não está em amarelo já existe no seu programa!!!!)

Você deverá inserir o código em 3 lugares:

Primeiro: Logo após a <div> do seu formulário:

```
<form id="form1" runat="server">
<div>
    <asp:MultiView ID="MultiView1" runat="server">
        <asp:View ID="viewGrid" runat="server">
            <asp:GridView ID="GridView1" runat="server" AllowPaging="True" AllowSorting="True"
```

Segundo: Logo após o seu botão “novo registro” - antes do objeto validation summary (se houver):

```
</asp:SqlDataSource>
<br />
<asp:Button ID="ButtonNovo" runat="server" Text="Novo Registro" />
</asp:View>
<asp:View ID="viewForm" runat="server">
    <asp:ValidationSummary ID="ValidationSummary1" runat="server" />
    <br />
    <table class="style1">
```

Terceiro: No fim da tabela que vc usou para montar o formulário, antes de fechar a div e o form:

```
</table>
</asp:View>
</asp:MultiView>
</div>
</form>
```

3º - O componente multiview, por padrão, não exibe nenhuma das views. Coloque no pageload para exibir, inicialmente, a view que está o gridview:

Observe que o evento Page_Load já existe na sua página!

```
protected void Page_Load(object sender, EventArgs e)
{
    if (IsPostBack == false) // só executa se for a primeira vez que a página é carregada
    {
        MultiView1.SetActiveView(viewGrid); // exibe a viewGrid (view que tem a grid)
    }
}
```

Se você executar o seu programa, deverá aparecer algo próximo disto:

	<u>id</u>	<u>tarefa</u>	<u>prioridade</u>	<u>data</u>
Selecionar	2	Assistir a peça da bela e a fera	M	25/10/2009 00:00:00
Selecionar	3	ver o jogo do timao	A	1/12/2009 00:00:00

[Novo Registro](#)

4º - Coloque um botão para cancelar, do lado do botão gravar.

Id = buttonCancelar

Text = Cancelar

CausesValidation = False

No evento onclick, coloque o seguinte código:

```
protected void ButtonCancelar_Click(object sender, EventArgs e)
{
    MultiView1.SetActiveView(viewGrid);
}
```

5º - Altere o seguinte no botão de apagar:

Na propriedade onclientClick: **if (confirm('Deseja apagar?')) return true; else return false;**

No evento click do botão, acrescente a linha amarela:

```
AgendaDAO.Apaga(Convert.ToInt16(textBoxId.Text));
GridView1.DataBind();
MultiView1.SetActiveView(viewGrid);
```

6º - Para evitar redundância no código, vamos centralizar em um método toda a parte de habilitar/desabilitar os controles:

Crie este método:

```
private void Modo(string operacao)
{
    bool valor;
    if (operacao.Equals("alteracao") ||
        operacao.Equals("inclusao"))
        valor = true;
    else
        valor = false;

    ButtonCancelar.Enabled = true; // este botão vai ficar sempre habilitado!!
    btnGravar.Enabled = valor;
    BtnExclui.Enabled = !valor;
    btnAltera.Enabled = !valor;

    TextBoxId.Enabled = false; // esse campo nunca poderá ser alterado!!!
    TextBoxData.Enabled = valor;
    TextBoxTarefa.Enabled = valor;
    DropDownListPrioridade.Enabled = valor;

    // se o modo for inclusao, também devemos limpar todos os campos!
    if (operacao.Equals("inclusao"))
    {
        TextBoxId.Text = "";
        TextBoxTarefa.Text = "";
        TextBoxData.Text = "";
        DropDownListPrioridade.SelectedIndex = -1;
    }
}
```

7º - Altere o evento click do botão gravar, colocando o seguinte código :

```
Agenda agenda = new Agenda();
agenda.setTarefa(TextBoxTarefa.Text);

agenda.setPrioridade(DropDownListPrioridade.SelectedValue[0]);
agenda.setData(Convert.ToDateTime(TextBoxData.Text));

if (TextBoxId.Text.Trim().Length != 0)
{
    // se foi informado o Id, é porque é uma alteração.
    agenda.setId(Convert.ToInt16(TextBoxId.Text));
    AgendaDAO.Alterar(agenda);
}
else
{
    // se não foi informado o Id, trata-se de uma inclusão
    AgendaDAO.Inserir(agenda);
}
GridView1.DataBind();
MultiView1.SetActiveView(viewGrid);
```

8º - Remova todo o código que havia no evento click do botão Alterar e coloque apenas este:

```
Modo("alteracao");
```

9º - No evento click botão Novo registro, coloque o seguinte código:

```
MultiView1.SetActiveView(viewForm);
Modo("inclusao");
```

10º - Altere o evento Command do linkbutton “selecionar” do grid: Adicione as linhas em amarelo:

(observe que o código que não está em amarelo já existe no seu programa!!!!)

```
int id = Convert.ToInt16(e.CommandArgument.ToString());  
  
Agenda agenda = AgendaDAO.getAgenda(id);  
  
TextBoxId.Text = agenda.getId().ToString();  
TextBoxTarefa.Text = agenda.getTarefa();  
TextBoxData.Text = agenda.getData().ToString();  
DropDownListPrioridade.SelectedValue = agenda.getPrioridade().ToString();  
MultiView1.SetActiveView(viewForm);  
Modo("consulta");
```

Opcional:

Usando um enumerador para evitar que o programador envie uma operação que não exista para o método MODO, além de auxiliá-lo durante a codificação.

Primeiro, crie um enumerador com as opções que são aceitas no método MODO: Crie antes do evento page_load da página:

```
private enum Operacao { inclusao, alteracao, consulta };
```

Agora, vamos alterar o método MODO, observe que o código em amarelo é o que foi alterado:

```
private void Modo(Operacao operacao)
{
    bool valor;
    if (operacao == Operacao.alteracao ||
        operacao == Operacao.inclusao)
        valor = true;
    else
        valor = false;

    ButtonCancelar.Enabled = true; // este botão vai ficar sempre habilitado!!
    btnGravar.Enabled = valor;
    BtnExclui.Enabled = !valor;
    btnAltera.Enabled = !valor;

    TextBoxId.Enabled = false; // esse campo nunca poderá ser alterado!!!
    TextBoxData.Enabled = valor;
    TextBoxTarefa.Enabled = valor;
    DropDownListPrioridade.Enabled = valor;

    // se o modo for inclusao, também devemos limpar todos os campos!
    if (operacao == Operacao.inclusao)
    {
        TextBoxId.Text = "";
        TextBoxTarefa.Text = "";
        TextBoxData.Text = "";
        DropDownListPrioridade.SelectedIndex = -1;
    }
}
```

Agora, no resto do código, altere onde havia:

Modo("consulta"); **altere para** Modo(Operacao.consulta);

Modo("inclusao"); **altere para** Modo(Operacao.inclusao);

Modo("alteracao"); **altere para** Modo(Operacao.alteracao);

Criando uma Tela de pedido

A tabela usada neste exemplo é a seguinte:

```
CREATE TABLE [dbo].[Produtos] (
    [ProdutoId] [int] NOT NULL,
    [ProdutoDescricao] [varchar](50) NULL,
    [Valor_unitario] [decimal](18, 2) NULL,
    CONSTRAINT [PK_Produtos] PRIMARY KEY CLUSTERED
(
    [ProdutoId] ASC
) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
) ON [PRIMARY]
```

Primeiro, coloque uma grid na tela que irá exibir todos os produtos da tabela acima. Altere todas as colunas, transformando-as em templatefield. Em cada template field, dê um nome ao label que irá exibir o valor da coluna. Ex: na coluna ProdutoId, coloque o nome do label: "labelProdutoId"

Pedido de Produtos



A aparência do grid deverá ficar parecida com:

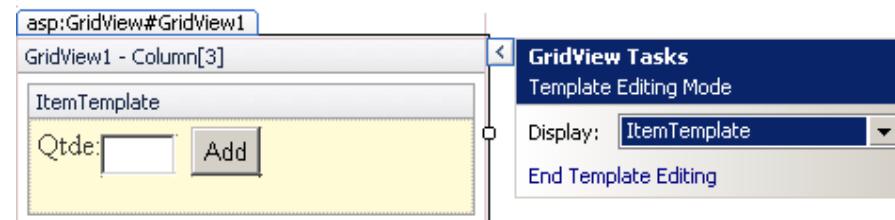
Pedido de Produtos

ProdutoId	ProdutoDescricao	Valor unitario	Qtde	Add
Databound	Databound	Databound	<input type="text"/>	<input type="button" value="Add"/>
Databound	Databound	Databound	<input type="text"/>	<input type="button" value="Add"/>

Para colocar a quantidade e o botão Add, crie uma coluna do tipo templateField.

Configure o template para deixá-lo como na figura abaixo. O nome do edit é TextBoxQtde.

Pedido de Produtos



Agora, vamos criar uma tabela temporária para guardar os produtos selecionados pelo cliente:

No code behind, coloque:

```
using System.Data;
```

Crie esta propriedade que irá representar nossa tabela temporária (DataTable), que ficará armazenada em uma session para manter seu valor entre os postback's!

```
private DataTable TabelaTemporaria
{
    get
    {
        if (Session["TabelaTemporaria"] == null)
        {
            Session["TabelaTemporaria"] = CriaDataTableItemPedido();
        }
        return Session["TabelaTemporaria"] as DataTable;
    }
    set
    {
        Session["TabelaTemporaria"] = value;
    }
}
```

O método abaixo cria uma tabela temporária para guardar os dados do produto que o cliente escolher.

```
public static DataTable CriaDataTableItemPedido()
{
    DataTable mDataTable = new DataTable();

    DataColumn mDataColumn;
    mDataColumn = new DataColumn();
    mDataColumn.DataType = typeof(int);
    mDataColumn.ColumnName = "ProdutoId";
    mDataTable.Columns.Add(mDataColumn);

    mDataColumn = new DataColumn();
    mDataColumn.DataType = typeof(string);
    mDataColumn.ColumnName = "ProdutoDescricao";
    mDataTable.Columns.Add(mDataColumn);

    mDataColumn = new DataColumn();
    mDataColumn.DataType = typeof(int);
    mDataColumn.ColumnName = "Quantidade";
    mDataTable.Columns.Add(mDataColumn);

    mDataColumn = new DataColumn();
    mDataColumn.DataType = typeof(double);
    mDataColumn.ColumnName = "Valor_Unitario";
    mDataTable.Columns.Add(mDataColumn);

    mDataColumn = new DataColumn();
    mDataColumn.DataType = typeof(double);
    mDataColumn.ColumnName = "Valor_total";
    mDataTable.Columns.Add(mDataColumn);

    return mDataTable;
}
```

Agora, vamos programar o botão ADD do grid, para inserir na tabela temporária os valores do produto selecionado.

Programe o evento click do botão ADD como no código abaixo:

Obs: por questões de simplicidade, não foram feitas as validações. Verifique se você nomeou os labels como está no código abaixo!

```

// descobre a linha do grid que efetuado o pressionamento do botão.
GridViewRow row = ((sender as Button).NamingContainer as GridViewRow);

// pega o id do produto
Label LabelProdutoId = row.FindControl("LabelProdutoId") as Label;

// descobre o textbox que guarda a quantidade
TextBox TextBoxQtde = row.FindControl("TextBoxQtde") as TextBox;

// descobre o label que guarda a descrição do produto
Label LabelProdutoDescricao = row.FindControl("LabelProdutoDescricao") as Label;

// descobre o label que guarda o valor unit. do produto
Label LabelValorUnit = row.FindControl("LabelValorUnit") as Label;

// adiciona os dados no datatable temporário
DataRow registro;
registro = TabelaTemporaria.NewRow();
registro["ProdutoId"] = LabelProdutoId.Text;
registro["ProdutoDescricao"] = LabelProdutoDescricao.Text;
registro["quantidade"] = TextBoxQtde.Text;
registro["Valor_Unitario"] = LabelValorUnit.Text;
registro["Valor_Total"] = Convert.ToInt16(TextBoxQtde.Text) * Convert.ToDouble(LabelValorUnit.Text);

TabelaTemporaria.Rows.Add(registro);

// atualiza o grid de itens comprados com os dados do dataTable
GridViewItens.DataSource = TabelaTemporaria;
GridViewItens.DataBind();

TextBoxQtde.Text = "";

```

Para exibir os dados do dataTable, adicione um novo grid, logo abaixo do grid dos produtos:

Column0	Column1	Column2
abc	abc	abc

Para “varrer” a tabela temporária, utilize um foreach. Este processo será importante quando você for passar os dados da tabela temporária para a tabela física.

```
foreach (DataRow registro in TabelaTemporaria.Rows)
{
    // para acessar os campos do dataTable, user registro["nome_do_campo"].ToString();
    // Exemplo: registro["produtoId"].ToString()
}
```

Agora, vamos criar uma forma de se excluir itens que o usuário escolheu, ou seja, vamos remover itens de sua sexta de compra!

Primeiro, vamos configurar o grid que representa os itens escolhidos pelo usuário: vamos criar uma coluna para cada campo que queremos exibir, no caso, vamos criar uma coluna para cada campo da tabela temporária. E iremos fazer o Bind em cada coluna com o seu respectivo campo do dataTable.

Quando for configurar as colunas do grid, não esqueça de desmarcar a seguinte opção: Auto-generate field (vide figura abaixo)

Cookies_datatable - Microsoft Visual Studio

File Edit View Website Build Debug Format Table Tools Test Window Help

App_Code/Util.cs web.config Default.aspx.cs **Default.aspx*** Start Page

Databound Databound Databound Qtde: Add

1

SqlDataSource - SqlDataSourceListaProdu

asp:GridView#GridViewItens

Cód.	Produto	Descrição	Quantidade
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound

Listar dados da tabela tempo

Design Split Source

Fields

Available fields:

- BoundField
- CheckBoxField
- HyperLinkField
- ImageField
- ButtonField
- CommandField
- TemplateField

Add

Selected fields:

- Cód.
- Produto
- Descrição
- Quantidade
- Valor unitário
- Valor Total
- TemplateField

Auto-generate fields

BoundField properties:

InsertVisible	True
NullDisplayText	
ReadOnly	False
ShowHeader	True
SortExpression	
Visible	True

Data

DataField: **ProdutoDescricao**

Styles

ControlStyle

DataField
The field to which this field is bound.

Convert this field into a TemplateField

OK Cancel

Error List				
X 0 Errors ! 0 Warnings i 0 Messages				
	Description	File	Line	Column

Error List Output

Ready

Ln 104

Col 24

Ch 24



Criando um cadastro se...

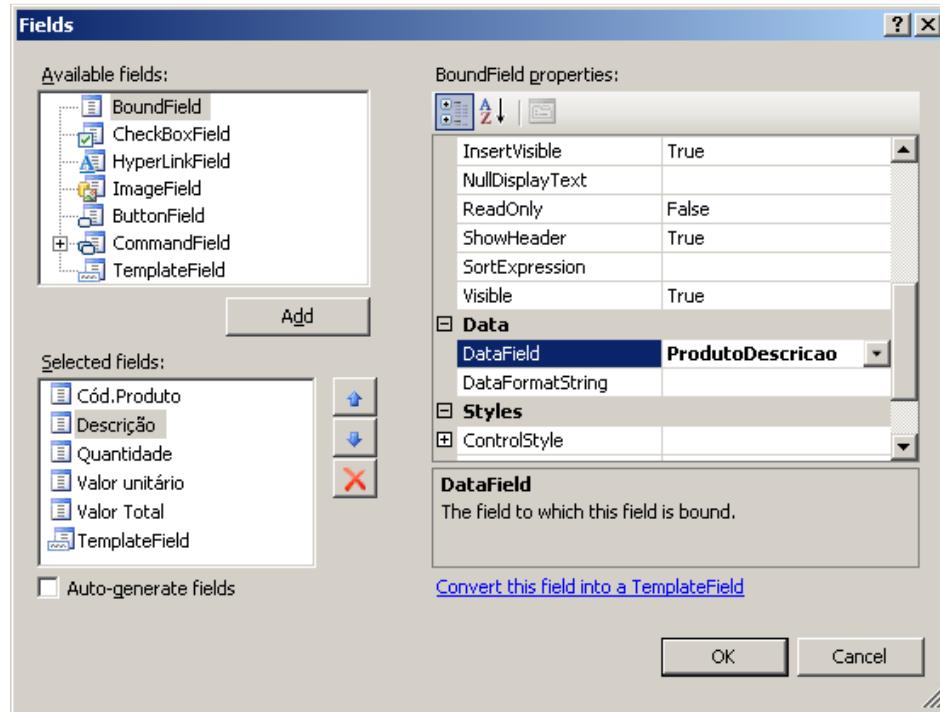
Cookies_datatable - ...

Cookies_datatable

imagem - Paint

PT

« 22:31

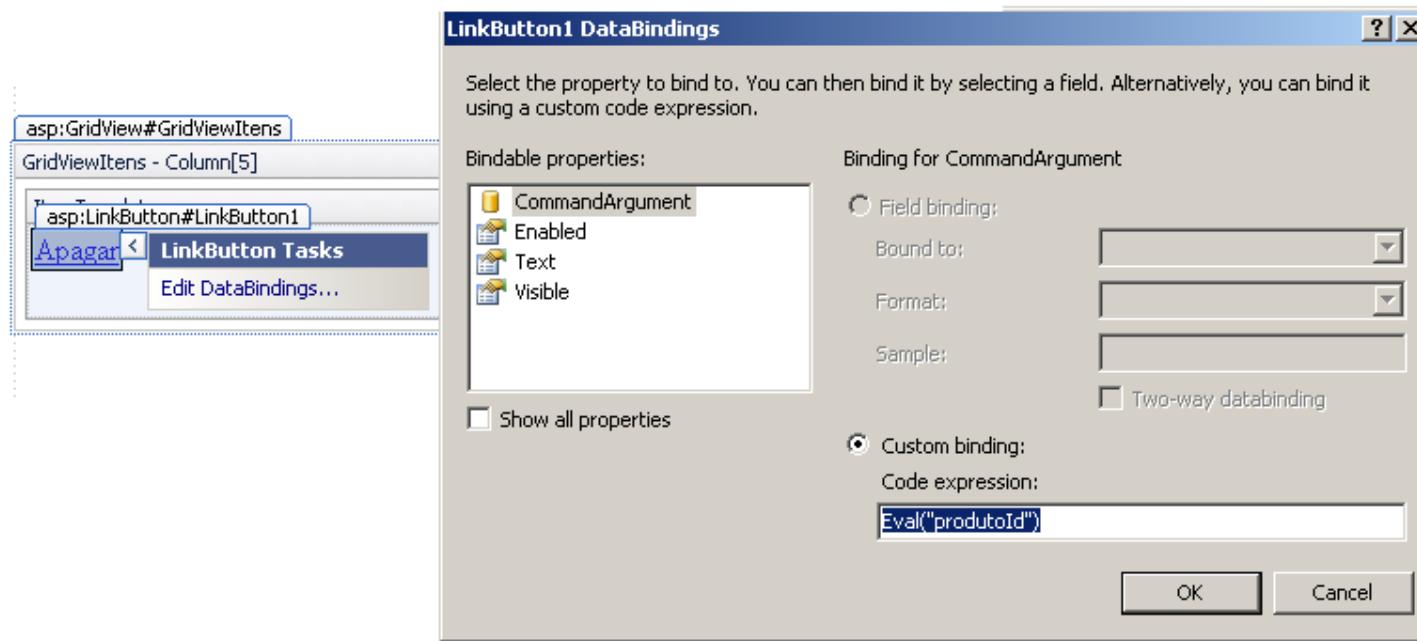


Vamos aproveitar a figura acima para explicar como configurar o Bind dos campos. Veja o exemplo na coluna "Descrição". Você deve preencher a propriedade "DataField" com o nome do respectivo campo no DataTable.

A grid deverá ficar parecida com a imagem a seguir:

Cód. Produto	Descrição	Quantidade	Valor unitário	Valor Total	
Databound	Databound	Databound	Databound	Databound	Apagar
Databound	Databound	Databound	Databound	Databound	Apagar
Databound	Databound	Databound	Databound	Databound	Apagar
Databound	Databound	Databound	Databound	Databound	Apagar
Databound	Databound	Databound	Databound	Databound	Apagar

Configure o linkbutton Apagar da figura acima, preenchendo o “custom binding” do commandArgunt com Eval("produtoId") . Fazendo isso, vamos preencher com o código do produto a propriedade commandArgument. Veja a figura abaixo:



No evento click do linkbutton, coloque o código abaixo para apagar do datatable o valor selecionado

```
string id = (sender as LinkButton).CommandArgument;
DataRow[] linhas = TabelaTemporaria.Select("produtoId = " + id);
TabelaTemporaria.Rows.Remove(linhas[0]);
GridViewItens.DataSource = TabelaTemporaria;
GridViewItens.DataBind();
```

Mapeando uma tabela utilizando propriedades no lugar de métodos “Get” e “Set”

Crie as variáveis privadas com letra minúscula, para diferenciar das propriedades, que iniciam com letra maiúscula.

As propriedades tem a facilidade de poderem ser utilizadas como variáveis (atributos da classe), e não como métodos.

```
public class Funcionario
{
    // variáveis privadas
    private int id;
    private string nome;
    private char sexo;
    private DateTime nascimento;

    //propriedades públicas

    public int Id
    {
        get { return id; }
        set { id = value; }
    }

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

    public char Sexo
    {
        get { return sexo; }
        set { sexo = value; }
    }

    public DateTime Nascimento
    {
        get { return nascimento; }
        set { nascimento = value; }
    }
}
```

Gravando um cadastro utilizando uma Stored Procedure

Uma das principais vantagens na utilização de Stored Procedures para gravação dos dados está na centralização do processo de validação e gravação dos dados em um único local, deixando o processo mais rápido e eficiente.

Primeiro, vamos criar uma SP para gravar os dados de um funcionário da tabela funcionários.

Estrutura da tabela funcionários:

```
CREATE TABLE [dbo].[FUNCIONARIOS] (
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [NOME] [varchar](50) COLLATE Latin1_General_CI_AI NULL,
    [SEXO] [char](1) COLLATE Latin1_General_CI_AI NULL,
    [NASCIMENTO] [datetime] NULL,
    CONSTRAINT [PK_FUNCIONARIOS] PRIMARY KEY CLUSTERED
    (
        [ID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Stored Procedure para Inserir um novo funcionário:

Obs: se você já tiver essa SP criada, altere o comando CREATE para ALTER no início da SP.

```
ALTER procedure CadastraFunc
(
    @nome      varchar(50) ,
    @sexo       char(1) ,
    @nascimento datetime,
    @ID_INSERIDO int OUTPUT
)
as
begin
    if len(rtrim(ltrim(@nome))) = 0
    begin
        RAISERROR('Preencha o nome corretamente!', 16, 1)
        return
    end

    if Upper(@sexo) <> 'M' and Upper(@Sexo) <> 'F'
    begin
        RAISERROR('O sexo deve conter "F" ou "M"', 16, 1)
        return
    end
```

```

if @nascimento > getdate()
begin
    RAISERROR('O sujeito nem nasceu ainda!', 16, 1)
    return
end

-- se chegou até aqui é porque os dados estão corretos

insert into funcionarios
(nome, sexo, nascimento)
values
(@nome, @sexo, @nascimento)

-- retorna no parâmetro de saída o ID do funcionário que acabou de ser inserido
set @ID_INSERIDO = scope_identity()
end

```

Agora vamos alterar a classe DAO para gravar utilizando uma SP:

```

public static void Inserir(Funcionario func)
{
    SqlConnection conexao = ConexaoBD.getConexao();

    //Observe que informamos aqui o nome da Stored Procedure!
    SqlCommand comando = new SqlCommand("CadastraFunc", conexao);

    // agora vamos alterar o tipo do comando para Stored Procedure.
    comando.CommandType = CommandType.StoredProcedure;

    //Criando e Preenchendo os parâmetros de entrada...
    comando.Parameters.AddWithValue("@Nome", func.Nome.ToString());
    comando.Parameters.AddWithValue("@Sexo", func.Sexo.ToString());
    comando.Parameters.AddWithValue("@Nascimento", func.Nascimento );

    comando.Parameters.AddWithValue("@Id_Inserido","");
    comando.Parameters["@Id_Inserido"].Direction = ParameterDirection.Output; // parâmetro de saída!!!!!

    //Executa a SP
    comando.ExecuteNonQuery();

    // retorna o id inserido
    func.Id = Convert.ToInt32(comando.Parameters["@ID_INSERIDO"].Value);
    conexao.Close();
}

```

```
}
```

No formulário, o código do botão GRAVAR ficaria assim:

```
protected void bt_gravar_Click(object sender, EventArgs e)
{
    try
    {
        Funcionario func = new Funcionario();

        func.Nome = txtNome.Text;
        func.Sexo = slcSexo.SelectedValue[0];
        func.Nascimento = Convert.ToDateTime(txtNascimento.Text);
        FuncionarioDAO.Inserir(func);
        GridView1.DataBind();

        ScriptManager.RegisterClientScriptBlock(Page, Page.GetType(), "Gravado",
            "alert('Registro inserido! ID: " + func.Id.ToString() + "');", true);
    }
    catch (Exception erro)
    {
        ScriptManager.RegisterClientScriptBlock(Page, Page.GetType(), "Erro",
            "alert('" + erro.Message + "');", true);
    }
}
```

Criando um cadastro mestre-detalhe utilizando Web User Control para o módulo do detalhe.

Uma das vantagens de se separar algumas partes da interface em Web User Controls é a reutilização do código, já que o Web User Control pode ser utilizado em outras telas, além da diminuição da complexidade, já que o código da tela em questão fica mais menor.

Para tanto, vamos utilizar o cadastro de funcionários do tópico anterior. Vamos adicionar à ele a opção de se cadastrar dependentes para um determinado funcionário.

Primeiro, crie a tabela de dependentes no banco de dados:

```
CREATE TABLE Dependentes (
    [Id] [int] IDENTITY(1,1) primary key NOT NULL,
    [Func_id] [int] NULL,
    [Dep_Nome] [varchar](50) NULL,
    [Dep_Nascimento] [datetime] NULL
```

Agora, vamos criar a classe de mapeamento da tabela dependentes (VO – Value Object):

```
public class Dependente
{
    private int id;
    private int func_id;
    private string dep_nome;
    private DateTime dep_Nascimento;

    public int Id
    {
        get { return id; }
        set { id = value; }
    }

    public int Func_id
    {
        get { return func_id; }
        set { func_id = value; }
    }
}
```

```
public string Dep_nome
{
    get { return dep_nome; }
    set { dep_nome = value; }
}

public DateTime Dep_Nascimento
{
    get { return dep_Nascimento; }
    set { dep_Nascimento = value; }
}
```

Agora, vamos criar a classe DAO (Data Access Object) para inclusão, alteração, exclusão e consulta de dependentes:

Colocar o namespace:

```
using System.Data.SqlClient;
```

O código da classe será o seguinte:

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

/// <summary>
/// Summary description for DependenteDAO
/// </summary>
public class DependenteDAO
{

    public static void Inserir(Dependente d)
    {
        // set dateformat dmy; este comando serve para alterar a forma como o SQL Server entende o formato de data
```

```

        string sql = String.Format("set dateformat dmy; " +
            "insert into Dependentes(Func_id, Dep_Nome, Dep_Nascimento) " +
            "values ('{0}', '{1}', '{2}')", d.Func_id, d.Dep_nome, d.Dep_Nascimento);
        SqlConnection con = conexao.getConexao();

        SqlCommand comando = new SqlCommand(sql, con);
        comando.ExecuteNonQuery();
        con.Close();
    }

    public static void Alterar(Dependente d)
    {
        string sql = string.Format("set dateformat dmy; update dependentes set " +
            "Func_id = {0}, Dep_Nome = '{1}', Dep_Nascimento = '{2}' " +
            "where id = {3}", d.Func_id, d.Dep_nome, d.Dep_Nascimento, d.Id);

        SqlConnection con = conexao.getConexao();

        SqlCommand comando = new SqlCommand(sql, con);
        comando.ExecuteNonQuery();
        con.Close();
    }

    public static void Apagar(int id)
    {
        string sql = "delete from dependentes where id = " + id.ToString();

        SqlConnection con = conexao.getConexao();

        SqlCommand comando = new SqlCommand(sql, con);
        comando.ExecuteNonQuery();
        con.Close();
    }

    public static Dependente getDependente(int id)
    {
        string sql = "select * from dependentes where id = " + id.ToString();
        SqlConnection con = conexao.getConexao();
        SqlCommand comando = new SqlCommand(sql, con);
        SqlDataReader reader = comando.ExecuteReader();

        if (reader.Read()) // se retornou algum registro

```

```

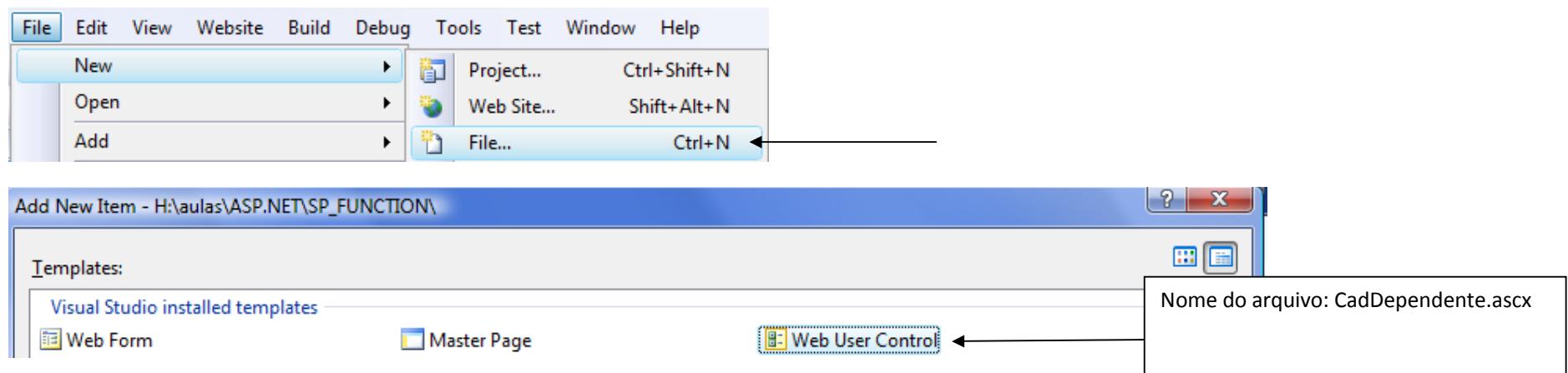
{
    Dependente dep = new Dependente();
    dep.Id = Convert.ToInt16(reader["id"].ToString());
    dep.Func_id = Convert.ToInt16(reader["func_id"].ToString());
    dep.Dep_nome = reader["dep_nome"].ToString();
    dep.Dep_Nascimento = Convert.ToDateTime(reader["dep_nascimento"].ToString());

    con.Close();
    return dep;
}
else
{
    con.Close();
    return null;
}
}
}

```

Configurando o Web User Control

Agora, vamos criar o Web User Control para manipulação dos dependentes:

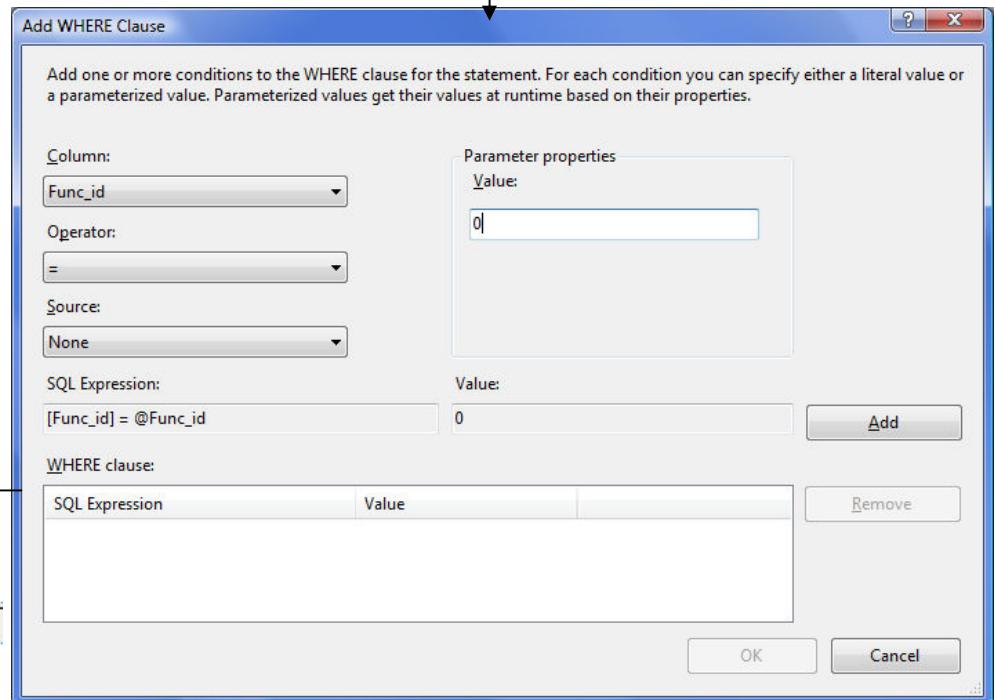


O cadastro pode ser feito qualquer outro que já fizemos anteriormente. Aparência da tela:

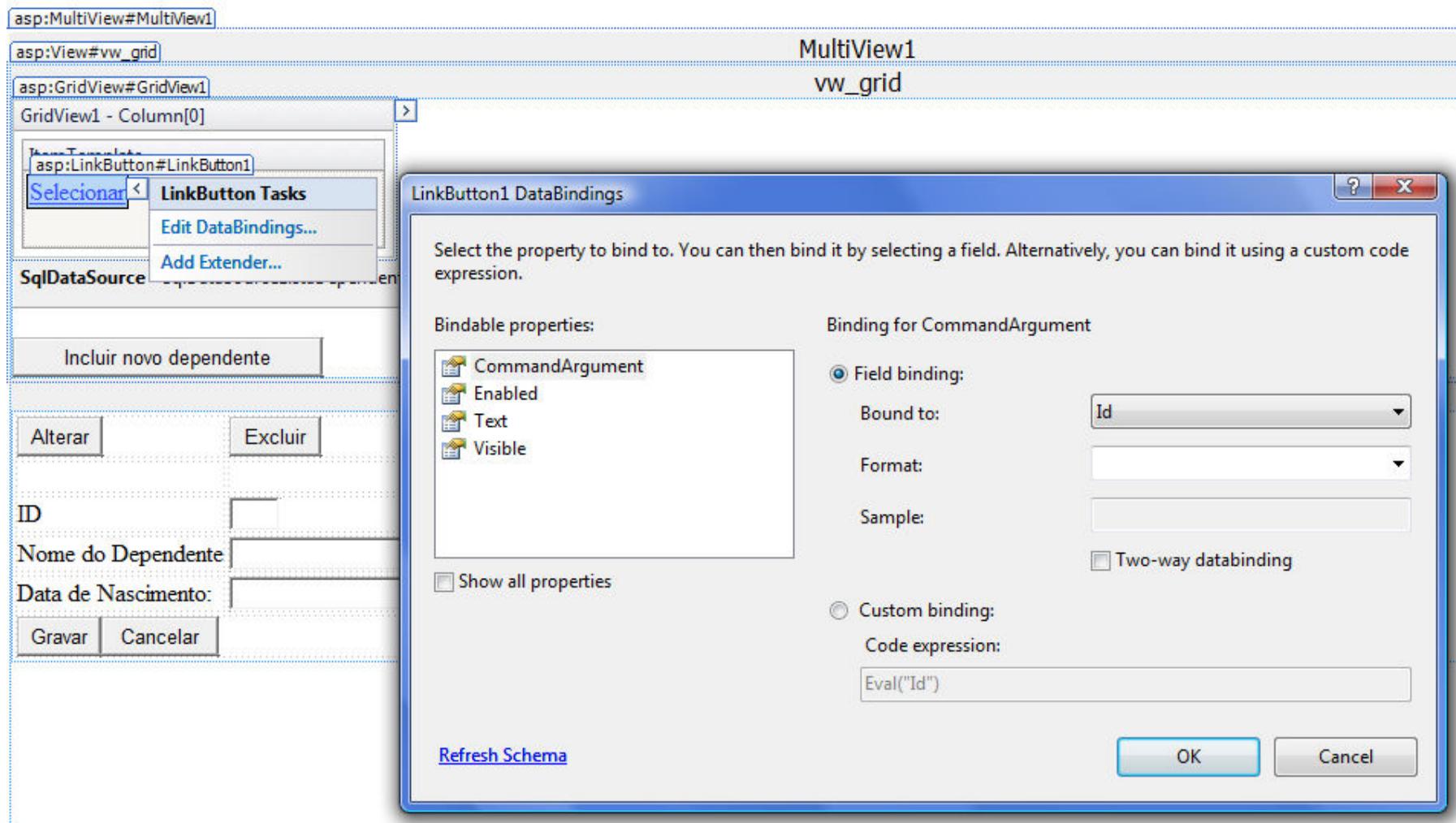
The screenshot shows a web page with the following components:

- Header:** `asp:MultiView#MultiView1` and `asp:View#vw_grid`.
- GridView:** A table with columns **Id**, **Func_id**, **Dep Nome**, **Dep**, and **Nascimento**. Each row contains a link labeled "Selecionar" and several "Databound" placeholder cells.
- Pagination:** A blue bar at the bottom of the grid with the numbers **1** and **2**.
- SqlDataSource:** A control labeled **SqlDataSource - SqlDataSourceListaDependentes**.
- Buttons:** "Incluir novo dependente", "Alterar", and "Excluir".
- Details View:** A dashed box containing fields for "ID" (input), "Nome do Dependente" (text box), "Data de Nascimento" (input), and buttons "Gravar" and "Cancelar".

Quando for configurar o GridView, não esqueça de criar um parâmetro para sejam exibidos apenas os dependentes do funcionário selecionado.



No gridview, transforme a coluna que tem o link de selecionar em um templatefield. Selecione a opção “Edit Databindings” e preencha a propriedade CommandArgument com o Id do dependente, como na figura abaixo:



Programe o evento **Command** do linkbutton Selecionar, colocando o seguinte código:

```
protected void LinkButton1_Command(object sender, CommandEventArgs e)
{
    int id = Convert.ToInt16(e.CommandArgument.ToString());

    Dependente dep = DependenteDAO.getDependente(id);

    TextBoxId.Text = dep.Id.ToString();
    TextBoxNome.Text = dep.Dep_nome.ToString();
    TextBoxNascimento.Text = dep.Dep_nome.ToString();
    MultiView1.SetActiveView(vw_form);
    Modo(Operacao.consulta);
}
```

O cadastro, de forma geral, é o mesmo que o explicado anteriormente na seção **Utilizando o componente MultiView para organizar a tela**

Segue abaixo o código completo do User Control de Dependente:

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

public partial class CadDependente : System.Web.UI.UserControl
{
    private enum Operacao { inclusao, alteracao, consulta };

    private int CodigoFuncionario
    {
        get
        {
            if (Session["codigoFuncionario"] == null)
                return 0;
            else
                return (int)Session["codigoFuncionario"];
        }
    }
```

```

    set
    {
        Session["codigoFuncionario"] = value;
    }
}

public void InicializaTela(int func_id)
{
    CódigoFuncionario = func_id;

    Funcionario func = FuncionarioDAO.GetFuncionario(CódigoFuncionario);
    LblNomeFunc.Text = func.Nome.ToString();

    SqlDataSourceListaDependentes.SelectParameters["Func_id"].DefaultValue = CódigoFuncionario.ToString();
    GridView1.DataBind();
    MultiView1.SetActiveView(vw_grid);
}

private void Modo(Operacao operacao)
{
    bool valor;
    if (operacao == Operacao.alteracao ||
        operacao == Operacao.inclusao)
        valor = true;
    else
        valor = false;

    ButtonCancelar.Enabled = true; // este botão vai ficar sempre habilitado!!
    ButtonGravar.Enabled = valor;
    ButtonExcluir.Enabled = !valor;
    ButtonAlterar.Enabled = !valor;

    TextBoxId.Enabled = false; // esse campo nunca poderá ser alterado!!!
    TextBoxNascimento.Enabled = valor;
    TextBoxNome.Enabled = valor;

    // se o modo for inclusao, também devemos limpar todos os campos!
    if (operacao == Operacao.inclusao)
    {
        TextBoxId.Text = "";
        TextBoxNome.Text = "";
        TextBoxNascimento.Text = "";
    }
}

```

```
}

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

protected void LinkButton1_Command(object sender, CommandEventArgs e)
{
    int id = Convert.ToInt16(e.CommandArgument.ToString());

    Dependente dep = DependenteDAO.getDependente(id);

    TextBoxId.Text = dep.Id.ToString();
    TextBoxNome.Text = dep.Dep_nome.ToString();
    TextBoxNascimento.Text = dep.Dep_Nascimento.ToString("dd/MM/yyyy");
    MultiView1.SetActiveView(vw_form);
    Modo(Operacao.consulta);
}

protected void ButtonNovo_Click(object sender, EventArgs e)
{
    MultiView1.SetActiveView(vw_form);
    Modo(Operacao.inclusao);
}

protected void ButtonAlterar_Click(object sender, EventArgs e)
{
    Modo(Operacao.alteracao);
}

protected void ButtonExcluir_Click(object sender, EventArgs e)
{
    DependenteDAO.Apagar(Convert.ToInt16(TextBoxId.Text));
    GridView1.DataBind();
    MultiView1.SetActiveView(vw_grid);
}

protected void ButtonCancelar_Click(object sender, EventArgs e)
{
    MultiView1.SetActiveView(vw_grid);
}
```

```
}

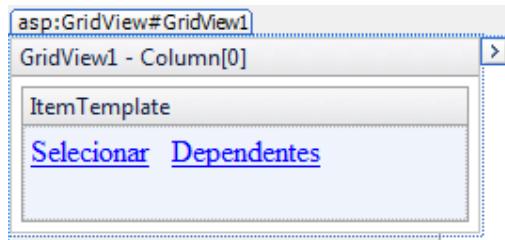
protected void ButtonGravar_Click(object sender, EventArgs e)
{
    Dependente dep = new Dependente();
    dep.Dep_nome = TextBoxNome.Text;
    dep.Func_id = CódigoFuncionario;
    dep.Dep_Nascimento = Convert.ToDateTime(TextBoxNascimento.Text);

    if (TextBoxId.Text.Trim().Length != 0)
    {
        // se foi informado o Id, é porque é uma alteração.
        dep.Id = Convert.ToInt16(TextBoxId.Text);
        DependenteDAO.Alterar(dep);
    }
    else
    {
        // se não foi informado o Id, trata-se de uma inclusão
        DependenteDAO.Inserir(dep);
    }
    GridView1.DataBind();
    MultiView1.SetActiveView(vw_grid);
}
}
```

Alterações na tela de cadastro de funcionários

A tela de cadastro de funcionários deverá ser dividida utilizando-se o componente Multiview, na última view coloque o Web User Control de dependente, como na tela abaixo:

No form de funcionários, junto com o linkbutton de selecionar, inclua também um linkbutton para exibir os dependentes:



Em ambos os campos, selecione a opção **"Edit Databindings"** e preencha a propriedade **CommandArgument** com o Id do Funcionário.

Veja abaixo como ficará a tela de cadastro de funcionários:

asp:MultiView#MultiView1

asp:View#vw_grid

asp:GridView#GridView1

	ID	NOME	SEXO	NASCIMENTO	
Selecionar	Dependentes	Databound	Databound	Databound	
Selecionar	Dependentes	Databound	Databound	Databound	
Selecionar	Dependentes	Databound	Databound	Databound	
Selecionar	Dependentes	Databound	Databound	Databound	
Selecionar	Dependentes	Databound	Databound	Databound	
		1 2			

SqlDataSource - SqlDataSourceListaFunc

[Novo Funcionário](#)

CADASTRO DE FUNCIONÁRIOS

ID

Nome

Sexo

Nascimento

[GRAVAR](#) [CANCELAR](#)

Dependentes do funcionário: [Label](#)

[MultiView "MultiView1"]

[Retornar para a Lista de Funcionários](#)

MultiView1

vw_grid

Vw_form

vw_dep

Insira aqui o web user control de dependentes.

Para inserir, é só arrastar o arquivo

 CadDependente.ascx diretamente do

Solution Explorer.

Este linkbutton NÃO ESTÁ dentro do Web User Control! Ele faz parte do form de funcionários.

O cadastro de funcionários não foi totalmente finalizado. Ele está pronto apenas para a operação de inclusão. Porém, o objetivo aqui é ver como utilizar o cadastro de dependentes juntamente com o cadastro de funcionários. O aluno pode, posteriormente, finalizar o cadastro de funcionários com base no cadastro de dependentes.

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

public partial class CadFuncionarios : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            MultiView1.SetActiveView(vw_grid);
        }
    }

    protected void ButtonGravar_Click(object sender, EventArgs e)
    {
        Funcionario func = new Funcionario();

        try
        {
            func.Nome = TextBoxNome.Text;
            func.Sexo = TextBoxSexo.Text[0];
            func.Nascimento = Convert.ToDateTime(TextBoxNascimento.Text);
            FuncionarioDAO.Inserir(func);
            GridView1.DataBind();
            ScriptManager.RegisterClientScriptBlock(this, this.GetType(),
                "Msg", "alert('Gravado com sucesso!!!');", true);
        }
        catch (Exception erro)
        {
            ScriptManager.RegisterClientScriptBlock(this, this.GetType(),
                "Erro", "alert('Erro ao gravar! Erro:' + erro.Message.Replace("'", " ") +
                "');", true);
        }
        MultiView1.SetActiveView(vw_grid);
    }
}
```

```
protected void LinkButton1_Command(object sender, CommandEventArgs e)
{
    int func_id = Convert.ToInt16(e.CommandArgument);

    Funcionario func = FuncionarioDAO.GetFuncionario(func_id);
    TextBoxID.Text = func.Id.ToString();
    TextBoxNascimento.Text = func.Nascimento.ToString("dd/MM/yyyy");
    TextBoxSexo.Text = func.Sexo.ToString();

    MultiView1.SetActiveView(Vw_form);
}

protected void LinkButtonDep_Command(object sender, CommandEventArgs e)
{
    int func_id = Convert.ToInt16(e.CommandArgument);

    CadDependente1.InicializaTela(func_id);
    MultiView1.SetActiveView(vw_dep);

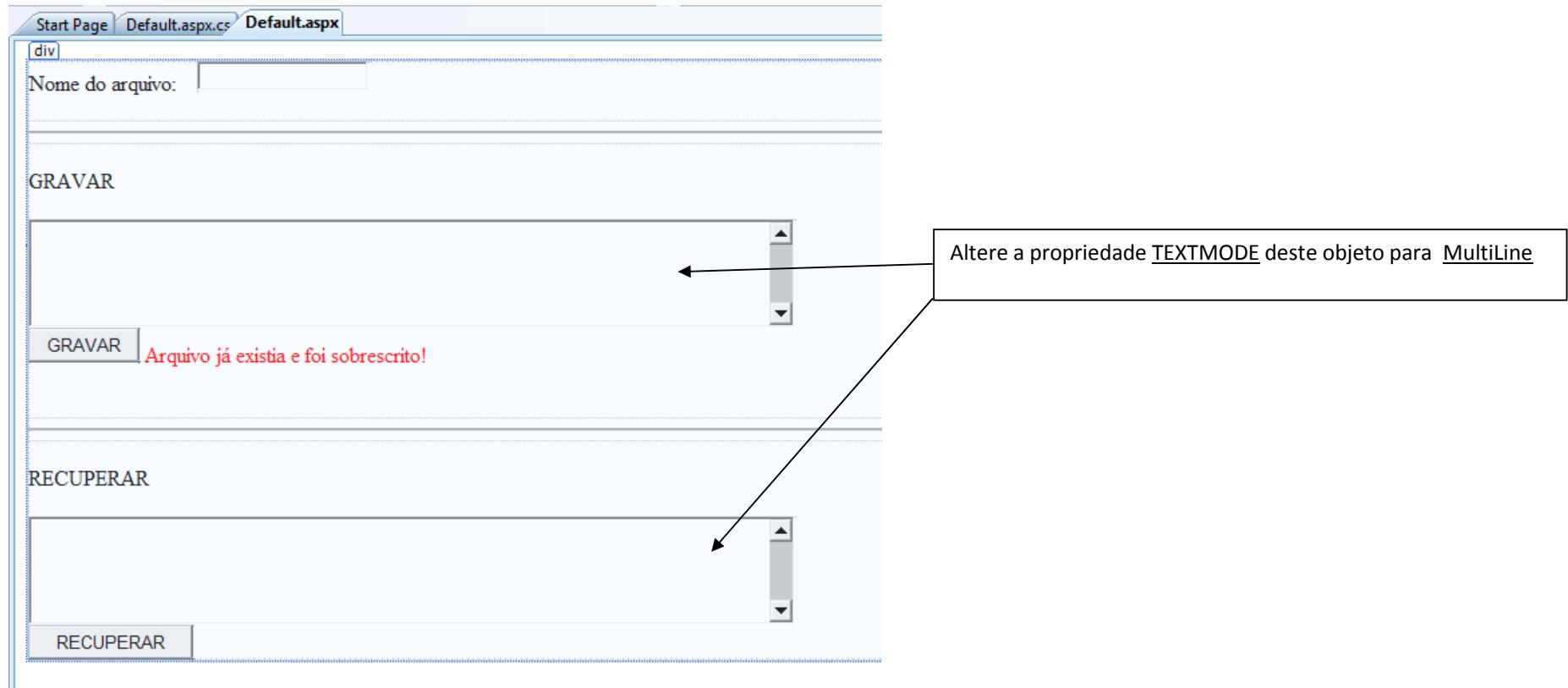
}

protected void ButtonCancelar_Click(object sender, EventArgs e)
{
    MultiView1.SetActiveView(vw_grid);
}

protected void ButtonNovo_Click(object sender, EventArgs e)
{
    TextBoxID.Text = "";
    TextBoxNome.Text = "";
    TextBoxNascimento.Text = "";
    TextBoxSexo.Text = "";
    MultiView1.SetActiveView(Vw_form);
}

protected void LinkButtonRetornar_Click(object sender, EventArgs e)
{
    MultiView1.SetActiveView(vw_grid);
}
```

Manipulando arquivos Texto.



Código completo:

OBS: Não esqueça de adicionar: `using System.IO;`

```
protected void Page_Load(object sender, EventArgs e)
{
    if ( !IsPostBack )
    {
        lblAviso.Visible = false;
    }
}
```

```
protected void ButtonGravar_Click(object sender, EventArgs e)
{
    // Com este código, o arquivo será salvo no mesmo diretório que está a aplicação

    FileInfo FI;
    StreamWriter SW;
    string Nome;

    Nome = Request.PhysicalApplicationPath + TextBoxArquivo.Text;
    FI = new FileInfo(Nome);

    if (FI.Exists)
        lblAviso.Visible = true;
    else
        lblAviso.Visible = false;

    SW = FI.CreateText();
    SW.WriteLine(TextBoxGravar.Text);
    SW.Close();
}
```

```
protected void ButtonRecuperar_Click(object sender, EventArgs e)
{
    StreamReader SR;
    string Nome;
    FileInfo FI;

    Nome = Request.PhysicalApplicationPath + TextBoxArquivo.Text;
    FI = new FileInfo(Nome);
    if (FI.Exists)
    {
        SR = new StreamReader(Nome, true);
        TextBoxRecuperar.Text = SR.ReadToEnd();
        SR.Close();
    }
    else
        ScriptManager.RegisterClientScriptBlock(Page, Page.GetType(), "Erro", "alert('Arquivo não existe!');", true);
}
```

Fazendo upload de arquivos

Monte uma tela como a da figura abaixo. Temos os componentes: FileUpload, Button e Label



Código da página:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:FileUpload ID="FileUpload1" runat="server" />
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" Text="Enviar Arquivo" OnClick="Button1_Click" />
            &nbsp;&nbsp;&nbsp;
            <asp:Label ID="LabelSituacaoUpload" runat="server" Text=""></asp:Label>
        </div>
    </form>
</body>
</html>
```

No evento click do botão “Enviar Arquivo”, digite:

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (FileUpload1.HasFile)
    {
        try
        {
            FileUpload1.SaveAs(Request.PhysicalApplicationPath + FileUpload1.FileName); // basta essa linha p/ salvar
            LabelSituacaoUpload.Text = string.Format("File: {0} {1} kb - Content Type {2} ",
                FileUpload1.PostedFile.FileName, FileUpload1.PostedFile.ContentLength, FileUpload1.PostedFile.ContentType);
        }
        catch (Exception ex)
        {
            LabelSituacaoUpload.Text = "ERRO: " + ex.Message.ToString();
        }
    }
    else
    {
        LabelSituacaoUpload.Text = "Você deve escolher um arquivo para o upload.";
    }
}
```

No código acima, o arquivo será salvo na mesma pasta que está o projeto.

É possível restringir os arquivos pelo tamanho (utilize FileUpload1.PostedFile.ContentLength), por alguma característica do nome (utilize FileUpload1.PostedFile.FileName) e pelo tipo do arquivo (FileUpload1.PostedFile.ContentType).

Se quiser salvar os arquivos em uma pasta, basta especificar o nome da pasta antes do nome do arquivo. Ex:

```
FileUpload1.SaveAs(Request.PhysicalApplicationPath + "FOTOS\\\" +FileUpload1.FileName);
```

Obs: São 2 barras mesmo!

Listas Tipadas com Generics

Generics introduzem no .NET Framework o conceito de parâmetros de tipos, que tornam possíveis a estruturação de classes e métodos que adiam a especificação de um ou mais tipos até que a classe ou método seja declarada e instanciada pelo código do cliente. Com generics, é possível criar listas baseadas em um determinado tipo de dado.

Classe Cidade

```
public class Cidade
{
    private int id;

    public int Id
    {
        get { return id; }
        set { id = value; }
    }
    private string nome;

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

Método na classe AlunoDAO

O método abaixo retorna uma lista do tipo Cidade (onde cada elemento da lista pode armazenar um objeto do tipo Cidade). Observe que List<cidade> é o tipo da lista!

```
public static List<Cidade> ListaCidades()
{
    SqlConnection conexao = ConexaoBD.Getconexao();
    string cmd = "select * from cidades";
    SqlCommand executor = new SqlCommand(cmd, conexao);
    SqlDataReader leitor = executor.ExecuteReader();

    Cidade cidade;
    List<Cidade> lista = new List<Cidade>(); // instancia uma lista vazia

    while (leitor.Read() == true) // enquanto houver dados...
    {
        cidade = new Cidade();
        cidade.Id = Convert.ToInt32(leitor["id"]);
        cidade.Nome = leitor["nome"].ToString();

        lista.Add(cidade);
    }
    return lista;
}
```

Coloque um DropDownList na página e no evento Page_Load coloque a seguinte instrução:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        DropDownListCidades.DataSource = CidadeDAO.ListaCidades();
        DropDownListCidades.DataTextField = "Nome"; // valor que aparece para o usuário
        DropDownListCidades.DataValueField = "Id"; // valor que pode ser pego na prop. SelectedValue
        DropDownListCidades.DataBind(); // atualiza o combo com os itens da lista
    }
}
```

Lista de cidades

Unbound

Ao executar a página:

Lista de cidades

S.B. do Campo

S.B. do Campo

São Paulo

São Caetano

Santo andre

Se quiser preencher um GridView com os dados da lista de cidades:

```
GridView3.DataSource = CidadeDAO.ListaCidades();  
GridView3.DataBind();
```

Id	Nome
1	S.B. do Campo
2	São Paulo
3	São Caetano
4	Santo andre

Classe Aluno. Observe a propriedade Cidade do aluno!

```
public class Aluno
{
    int id;

    public int Id
    {
        get { return id; }
        set { id = value; }
    }
    string nome;

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

    public double Mensalidade
    {
        get { return mensalidade; }
        set { mensalidade = value; }
    }

    Cidade cidade;

    public Cidade Cidade
    {
        get { return cidade; }
        set { cidade = value; }
    }
}
```

Classe AlunoDAO – Criando uma lista que retorna um DataTable. Usando o datatable para preencher uma lista tipada.

```
public class AlunoDAO
{
    public static DataTable getAlunosDataTable()
    {
        SqlConnection conexao = ConexaoBD.Getconexao();
        string cmd = "select * from alunos";
        SqlDataAdapter adapter = new SqlDataAdapter(cmd, conexao);
        DataTable datatable = new DataTable();
        adapter.Fill(datatable);
        return datatable;
    }

    public static List<Aluno> getAlunosList()
    {
        DataTable datatable = getAlunosDataTable();
        Aluno aluno;
        List<Aluno> lista = new List<Aluno>();

        foreach (DataRow row in datatable.Rows)
        {
            aluno = new Aluno();
            aluno.Id = Convert.ToInt32(row["id"]);
            aluno.Nome = row["nome"].ToString();
            aluno.Mensalidade = Convert.ToDouble(row["mensalidade"]);
            aluno.Cidade = CidadeDAO.getCidade(Convert.ToInt32(row["Cidadeid"]));

            lista.Add(aluno);
        }

        return lista;
    }
}
```

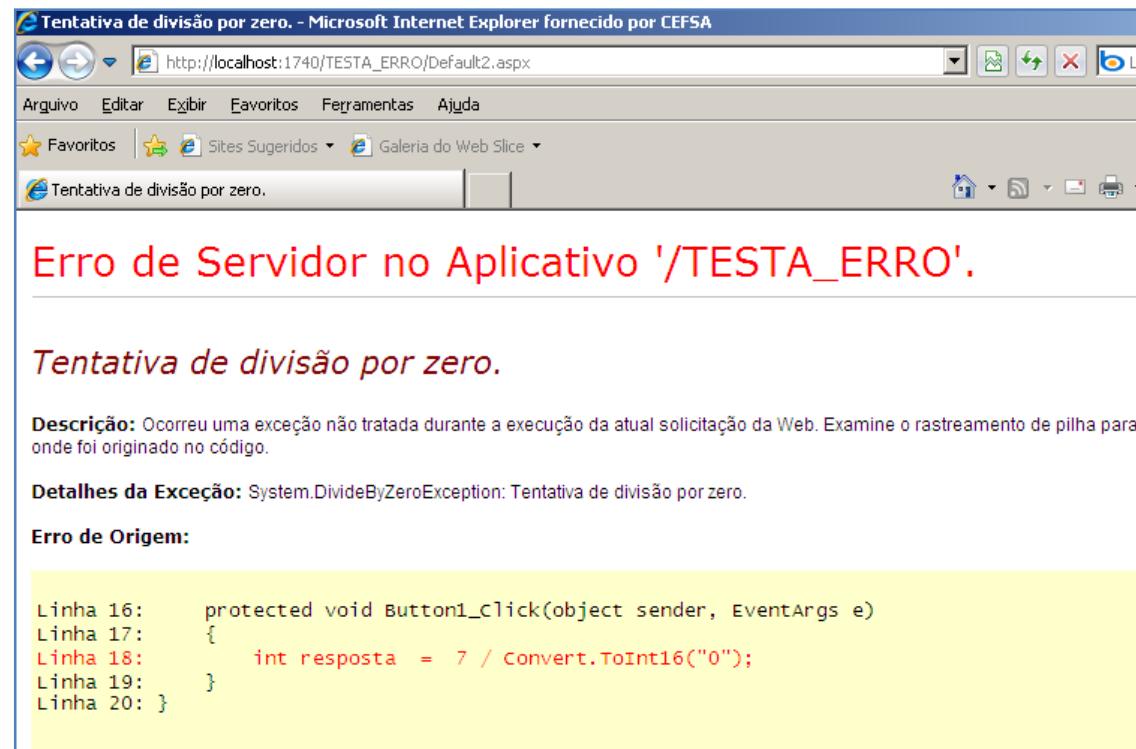
Página de Erros Personalizada

Parte deste tópico foi retirado de: <http://www.linhadecodigo.com.br/ArtigoImpressao.aspx?id=2389>

Erros acontecem e sempre vão acontecer mesmo nas mais grandiosas aplicações. Se não podemos nos livrar totalmente deles, pelo menos temos que tentar encarar uma situação desagradável com esta com a maior classe possível. O ASP.NET nos prove alguns recursos para tratamento de erro, no sentido de exibirmos mensagens padronizadas e mais amigáveis para o usuário.

Para exemplificar, crie uma página que tenha 1 botão. E no código do botão, digite:

```
protected void Button1_Click(object sender, EventArgs e)
{
    int resposta = 7 / Convert.ToInt16("0");
}
```



The screenshot shows a Microsoft Internet Explorer window with the title "Tentativa de divisão por zero. - Microsoft Internet Explorer fornecido por CEFSA". The address bar shows "http://localhost:1740/TESTA_ERRO/Default2.aspx". The page content is as follows:

Erro de Servidor no Aplicativo '/TESTA_ERRO'.

Tentativa de divisão por zero.

Descrição: Ocorreu uma exceção não tratada durante a execução da atual solicitação da Web. Examine o rastreamento de pilha para onde foi originado no código.

Detalhes da Exceção: System.DivideByZeroException: Tentativa de divisão por zero.

Erro de Origem:

```
Linha 16:     protected void Button1_Click(object sender, EventArgs e)
Linha 17:     {
Linha 18:         int resposta = 7 / Convert.ToInt16("0");
Linha 19:     }
Linha 20: }
```

Este código vai causar um erro, já que não é possível dividir por zero!

Erro personalizado através do método OnError

A página tem um método chamado **OnError** que é disparado sempre que ocorre um erro. Neste método, podemos fazer um tratamento de erro. Basta sobrescrever o método (override):

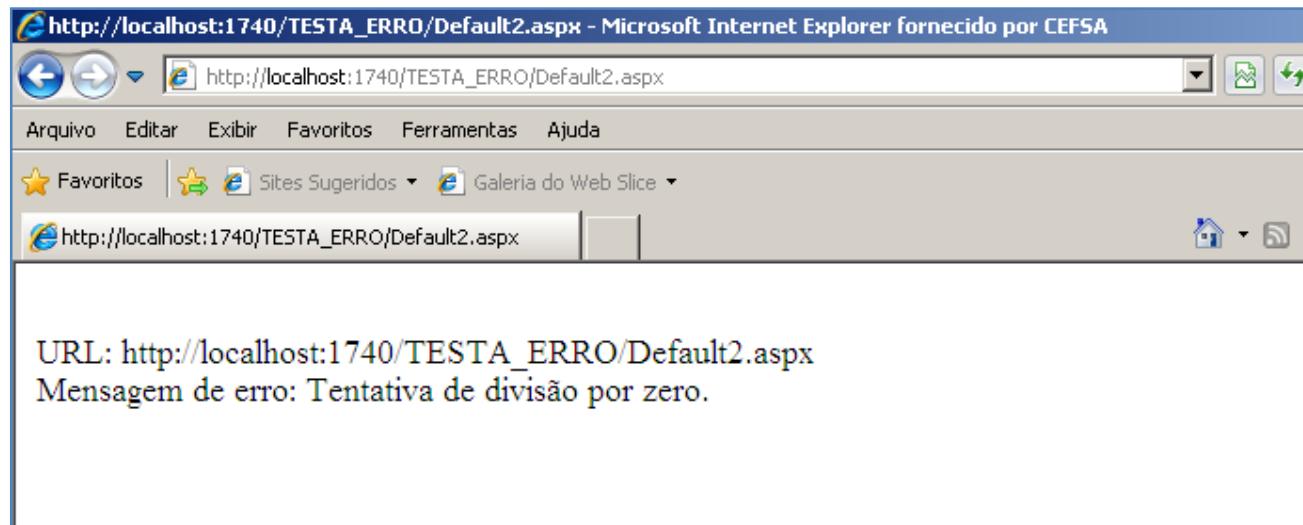
```
protected override void OnError(EventArgs e)
{
    // At this point we have information about the error
    HttpContext ctx = HttpContext.Current;

    Exception exception = ctx.Server.GetLastError();

    string errorInfo =
        "<br>URL: " + ctx.Request.Url.ToString() +
        "<br>Mensagem de erro: " + exception.Message;

    ctx.Response.Write(errorInfo);
    ctx.Server.ClearError();
    base.OnError(e); // executa o código do pai
}
```

Execute a página e clique no botão. Observe que o erro irá aparecer de forma diferente:



Erro personalizado através de página de erro no Web.Config

Envio de emails

Crie um formulário com os seguintes campos:

The screenshot shows a Microsoft Visual Studio interface with the 'Default.aspx.cs' and 'Default.aspx' tabs at the top. The main area displays a web form titled 'Enviando Emails'. The form contains several input fields and a text area:

- De: [Text Box]
- Assunto: [Text Box]
- Para: [Text Box]
- Email de origem: [Text Box]
- Senha: [Text Box]
- Servidor SMTP: [Text Box]
- Texto do email: [Text Area with scroll bar]
- Enviar email: [Button]

No botão de Enviar, digite:

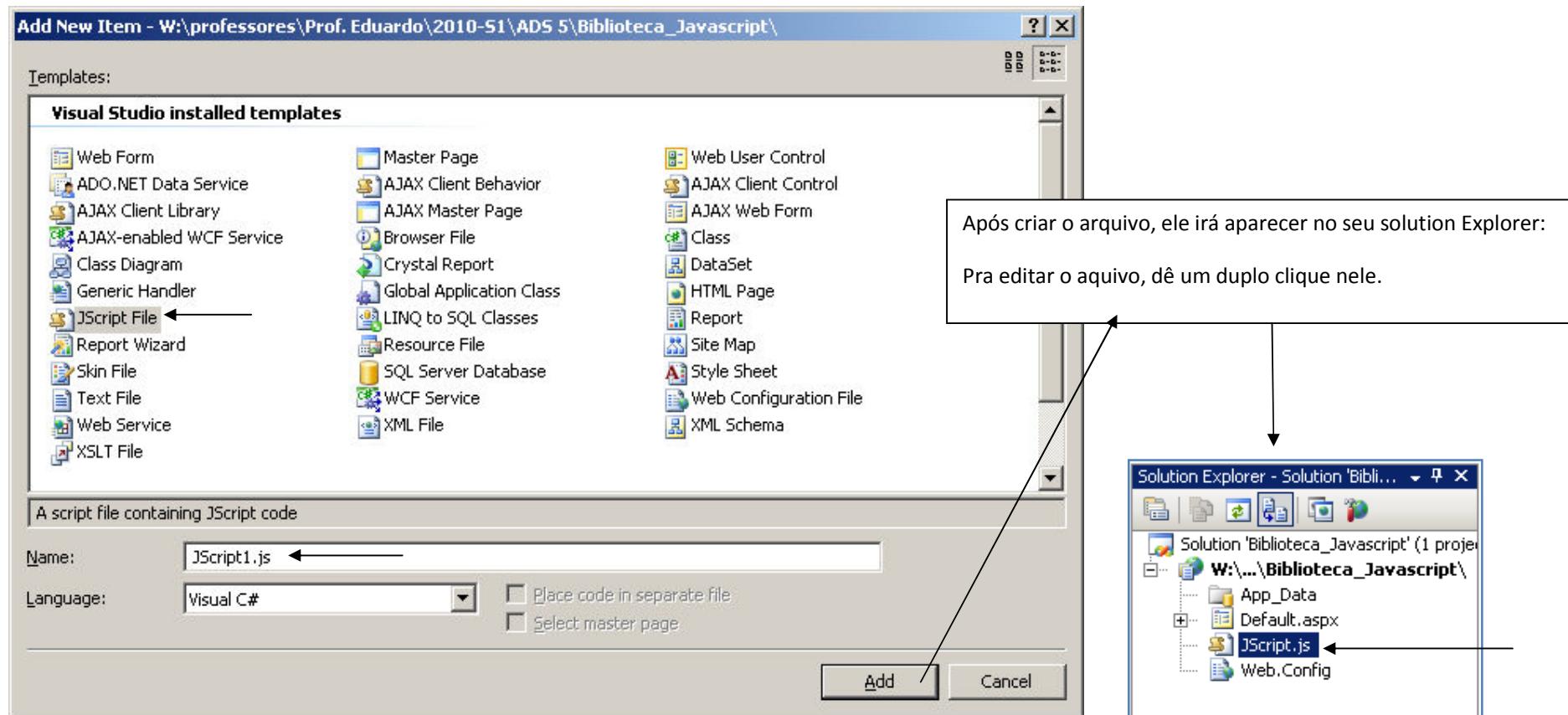
```
protected void Button1_Click(object sender, EventArgs e)
{
    MailMessage mailMessage = new MailMessage();
    mailMessage.From = txtDe.Text;
    mailMessage.Subject = txtAssunto.Text;
    mailMessage.To = txtPara.Text;
    mailMessage.Body = txtMensagem.Text;
    mailMessage.Fields["http://schemas.microsoft.com/cdo/configuration/smtpauthenticate"] = 1;
    mailMessage.Fields["http://schemas.microsoft.com/cdo/configuration/sendusername"] = TxtEmailOrigem.Text;
    mailMessage.Fields["http://schemas.microsoft.com/cdo/configuration/sendpassword"] = txtSenha.Text;

    SmtpMail.SmtpServer = txtSmtp.Text;
    SmtpMail.Send(mailMessage);
}
```

Adicionando uma biblioteca Javascript ao seu WebSite

Uma das vantagens de se adicionar uma biblioteca Javascript é a redução na redundância de código nesta linguagem, já que todas as funções compartilhadas entre diversas páginas estarão concentradas em um único local.

Para criar uma biblioteca de código Javascript, dentro do seu projeto, acesse o menu File-New->File e escolha a opção JScript File:



Abaixo, temos um exemplo de funções em javascript que foram colocadas dentro do arquivo Jscript1.js:

```
// JScript File

// As 3 funções abaixo servem para formatar valores
monetários durante a digitação.
// Veja um exemplo de utilização em
tabelaPrecoProduto.ascx.cs
// Eduardo.
function mascara(o, f) {
    v_obj = o
    v_fun = f
    setTimeout("execmascara()", 1)
}

function execmascara() {
    v_obj.value = v_fun(v_obj.value)
}
function mreais(v) {
    v = v.replace(/\D/g, "") //Remove tudo o que não é dígito
    v = v.replace((\d{2})$/, ",\$1") //Coloca a vírgula
    v = v.replace((\d{3},\d{2})$/g, "\$1.\$2") //Coloca o
primeiro ponto
    return v
}
function minteiro(v) {
    v = v.replace(/\D/g, "") //Remove tudo o que não é dígito
    return v
}

//Esta função tem por objetivo formatar um campo data,
adicionando
//as "/" automaticamente conforme o usuário vai digitando a
data.
function FormataData(objeto)
{
    if (objeto.value.length == 2 || objeto.value.length == 5 )
    {
        objeto.value = objeto.value+ "/";
    }
}
```

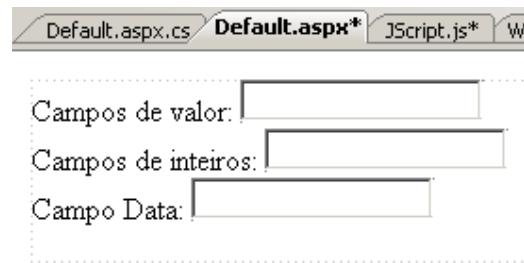
Para permitir que uma página utilize essa biblioteca Javascript, adicione a seguinte linha:

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
    <script src="JScript.js" type="text/javascript"></script> ←
</head>
<body>
```

Esta é a linha que você deve adicionar dentro da TAG Head da sua página:

Para utilizar as funções javascript, vamos fazer o seguinte exemplo:



No Evento Page_Load, vamos adicionar nos textbox's as funções javascript.

O evento onKeyPress ocorre quando o usuário pressiona uma tecla. O evento onKeyUp ocorre quando o usuário libera a tecla.

O evento onfocus ocorre quando o textbox recebe o foco. Este evento foi programado só para mostrar como poderíamos incluir um código javascript sem o uso da biblioteca que criamos.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        TextBox1.Attributes["onfocus"] = "javascript:this.select(); ";
        TextBox1.Attributes["onkeypress"] = "mascara(" + TextBox1.ID + ",mreais)";

        TextBox2.Attributes["onfocus"] = "javascript:this.select(); ";
        TextBox2.Attributes["onkeypress"] = "mascara(this,minteiro)";

        TextBox3.Attributes["onfocus"] = "javascript:this.select(); ";
        TextBox3.Attributes["onkeyup"] = "FormataData(this)";
    }
}
```

Publicar o WebSite com IIS

1. Primeiro: Você tem que ter instalado no computador o IIS. Se não tiver, vai precisar do CD do Windows para instalá-lo, ou poderá baixar o pacote de instalação à parte no site da Microsoft.

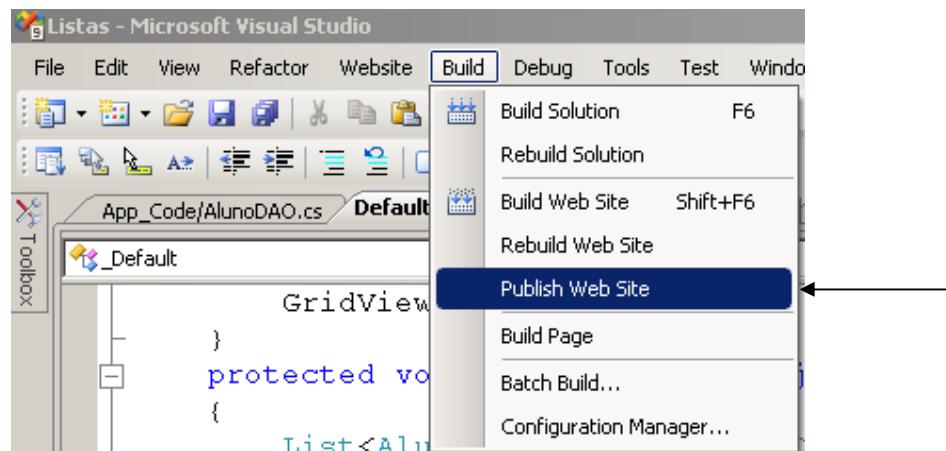
IIS 6.0

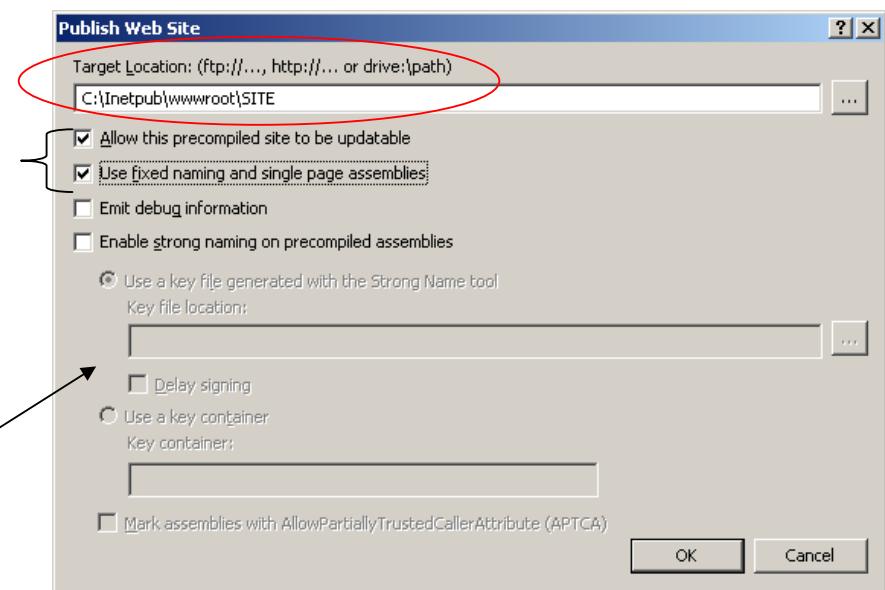
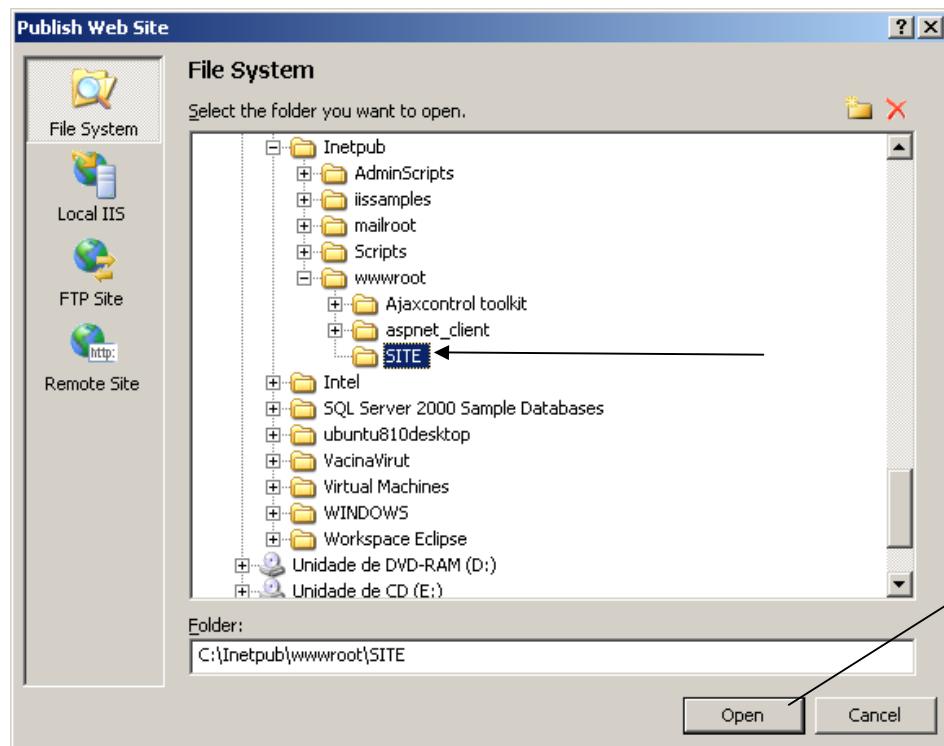
<http://www.microsoft.com/downloads/details.aspx?familyid=f9c1fb79-c903-4842-9f6c-9db93643fdb7&displaylang=en>

2. Após ter instalado o IIS 6.0...

Crie uma pasta chamada **SITE** dentro de **C:\Inetpub\wwwroot**

Abra algum projeto asp.net seu no visual Studio e publique-o na pasta que vc criou:







3. Abra o Internet Information Services (Painel de controle -> Ferramentas Administrativas)

Vamos configurar um diretório virtual que execute o aplicativo que publicamos na pasta SITE:

Internet Information Services

Arquivo Ação Exibir Ajuda

PC_VP (computador local)

Sites da Web

Site da Web padrão

IISHelp Scripts Explorar Abrir Procurar

TOOLKIT

Iniciar Parar Pausar

Novo Diretório virtual...

Crie um novo diretório virtual nos sites da WEB Padrão

Assistente para criação de diretório virtual

Alias do diretório virtual

Você deve atribuir ao diretório virtual um nome abreviado ou alias, para consulta rápida.

Digite o alias a ser usado para acessar este diretório virtual da Web. Use as mesmas convenções de nomes usadas para nomear um diretório.

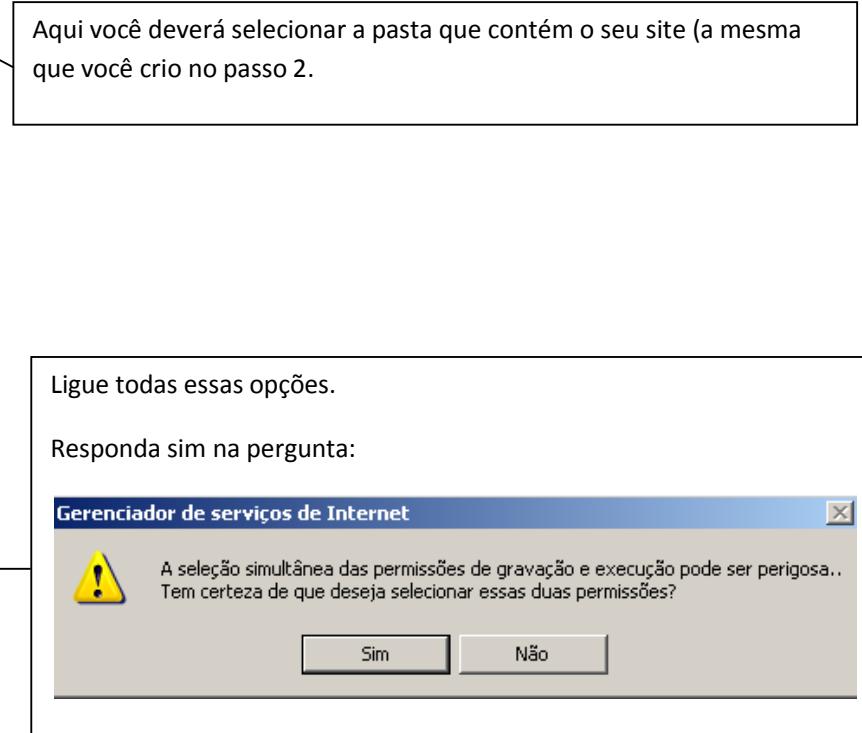
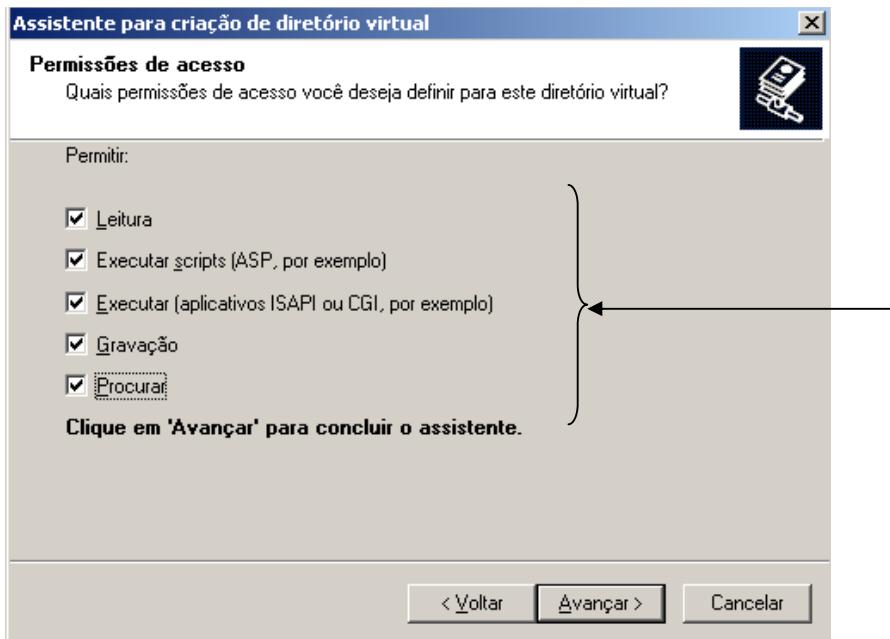
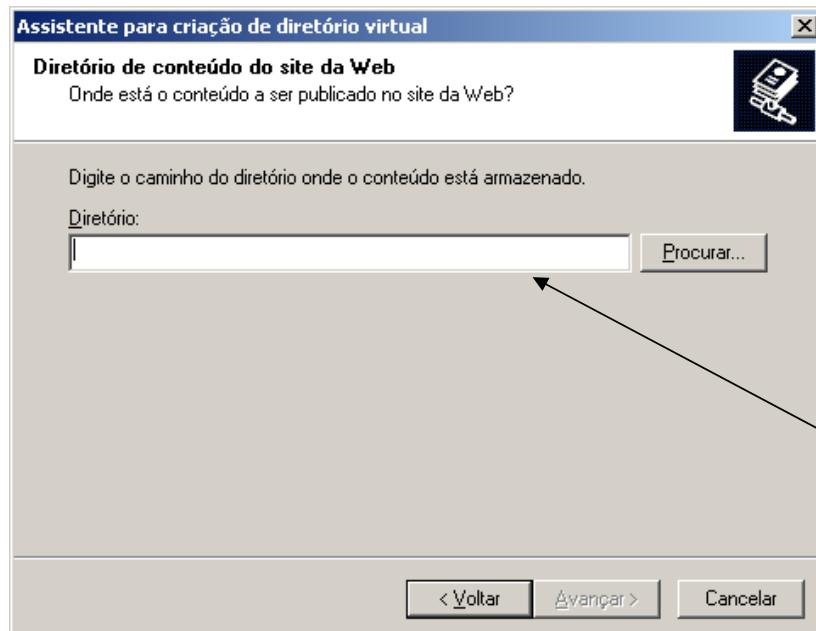
Alias:

Dê um apelido para o seu diretório virtual. Não utilize acentos ou qualquer outro caractere especial.

Para este exemplo, utilize “SITE”

Este apelido será utilizado para acessar a sua aplicação asp.net

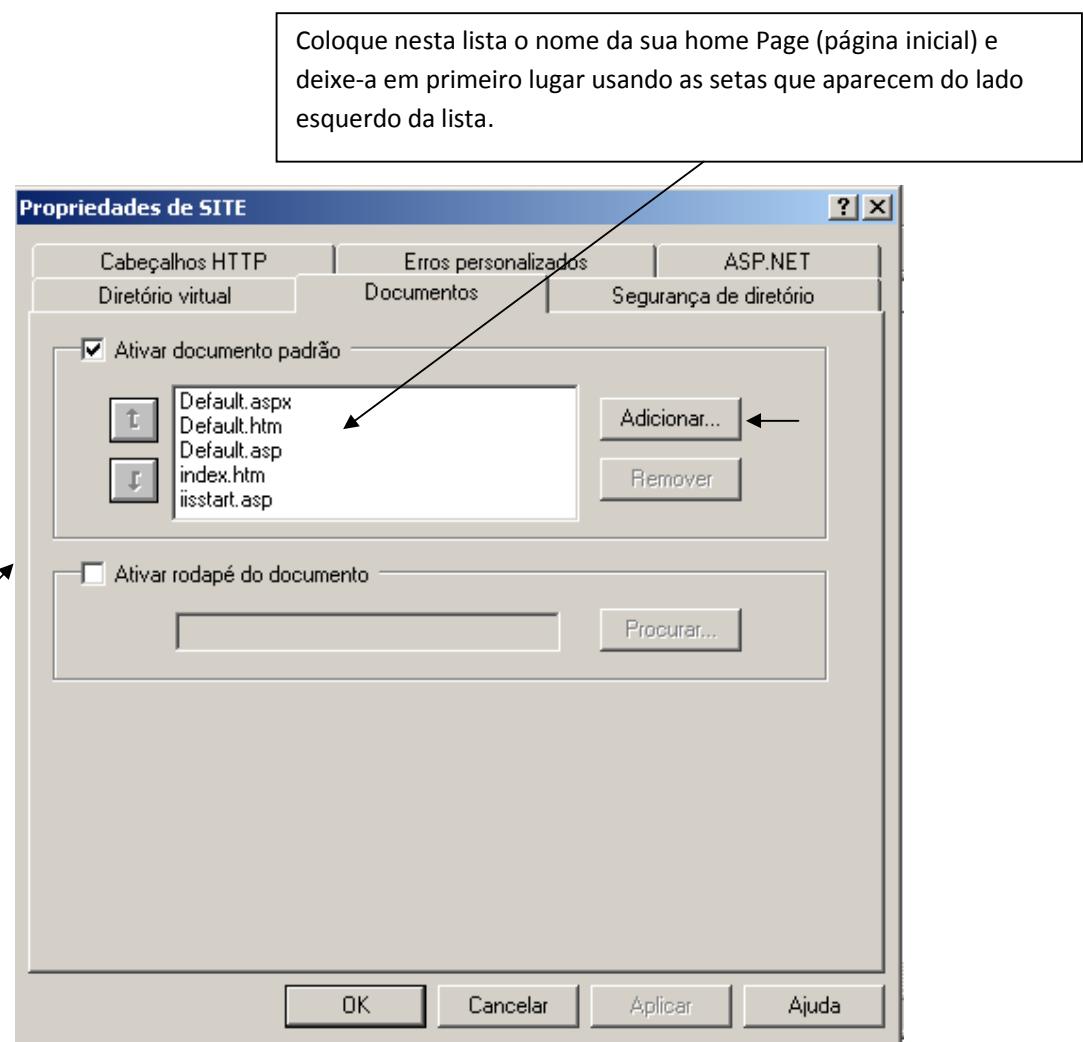
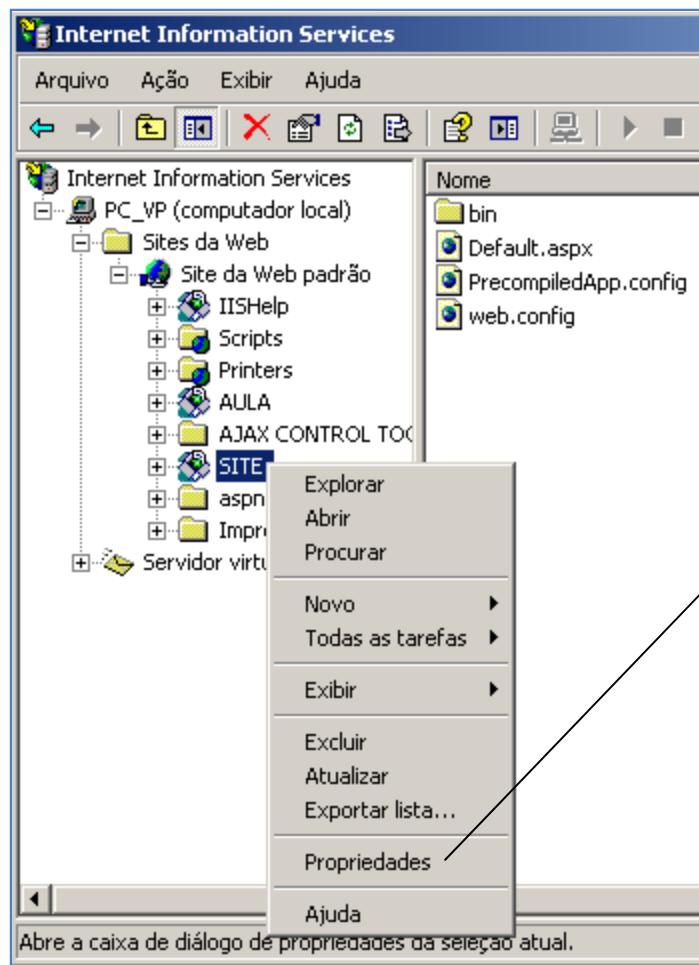
< Voltar Avançar > Cancelar



Para executar o seu site, digite no navegador web: <HTTP://localhost/site/default.aspx> (caso sua página inicial tenha o nome de default.aspx)

Para não ter que digitar o nome da página, ex: <HTTP://localhost/site/>, precisamos definir a página default (padrão), ou seja, a página que será exibida sempre que digitarmos apenas o caminho do site. Como exemplo, imagine o site <WWW.UOL.com.br>. Quando você vai utilizá-lo, não precisa especificar no endereço a página que deseja acessar. Ele irá automaticamente para a home do site.

Para definir uma página default:



Web Services

Fontes:

<http://www.ime.usp.br/~reverbel/SOD-06/trabalhos/fachada-ws/node2.html>

http://imasters.uol.com.br/artigo/1680/webservices/visao_geral_sobre_webservices/

http://www.oficinadanet.com.br/artigo/447/o_que_e_web_service

O que são web services?

Web services é o nome dado à tecnologia que permite a comunicação entre aplicações de uma maneira independente de sistema operacional e de linguagem de programação. Simplificando, um webservice é a maneira prática e eficaz de aplicativos se comunicarem via internet. Surgiu por um consórcio de empresas lideradas pela Microsoft e IBM, e hoje se tornou um padrão do W3C.

Para que servem os WebServices?

Praticamente para quase tudo no que você possa pensar em troca de dados e informações. Como ele é baseado no protocolo SOAP (Simple Object Access Protocol), qualquer plataforma que interprete rotinas HTTP e manipule XML pode utilizar os dados dos webservices sem qualquer problema. Como quase sempre os webservices utilizam o protocolo HTTP, quase sempre não haverá surpresas com Firewall quando sua aplicação sair do servidor de aplicação para a versão final do cliente.

Vantagens

Os dados são trafegados em formato XML. Assim, todos os dados possuem <tags> e isso faz com que estes dados tenham consistência, sem falhas, permitindo a troca de dados mais robustos como “Arrays”, e por trafegar em XML, ele é multi-plataforma. Existem muitas outras, mas estas são as principais.

Segurança:

Muitas empresas temiam, no passado, prover funcionalidades na Internet devido ao medo de expor seus dados. Mas com advento dos Web services elas podem publicar serviços de forma simples e que são totalmente isolados da base de dados.

Integração de sistemas:

Muitos consideram que os Web services corrigem um grande problema da informática: a integração de sistemas. Os Web services permitem que a integração de sistemas seja realizada de maneira comprehensível, reutilizável e padronizada. É uma tentativa de organizar um cenário cercado por uma grande variedade de diferentes aplicativos, fornecedores e plataformas.

O que é DWSL?

É a sigla de (Webservice Description Language), padrão baseado em XML para descrever o serviço, trazendo informações como por exemplo os métodos do webservice. Funciona como uma espécie de “TypeLibrary” do Webservice alem de ser usado para a validação das chamadas dos métodos.

Principais características

- Uso intenso de XML: XML é uma linguagem para representação de dados que é extensível e naturalmente independente de plataforma, além de ser amplamente utilizada pela indústria. Em web services, tanto a descrição de um serviço quanto a comunicação entre serviços é feita usando XML.
- Grande apoio da indústria: web services têm recebido grande apoio da indústria, particularmente por parte da IBM e da Microsoft. É possível, por exemplo, que um web service implantado num servidor de aplicações J2EE acesse outro web service implantado numa plataforma .NET.
- Baseado em padrões abertos: web services são baseados em uma série de padrões abertos e amplamente difundidos, tais como XML, HTTP, SOAP, WSDL e UDDI. Isso assegura que implementações compatíveis com as especificações sejam interoperáveis.
- Amigável a firewalls: as mensagens trocadas entre web services tipicamente usam HTTP como protocolo de transporte, o que em boa parte dos casos evita problemas com firewalls.
- Uso de URIs para identificação: os web services são identificados por uma URI (que tipicamente é uma URL), um formato muito utilizado devido à popularização da web e de fácil assimilação pelos humanos.

Consumindo um Web Service

De fato, a comunicação realizada entre o WebService e a aplicação que está o consumindo pode ser realizada através de 3 protocolos de comunicação:

Http-Get: as informações são enviadas como parte da URL (querystrings) na chamada do WebService, permitindo apenas o envio de pares contendo parâmetro/valor.

Ex: teste.asmx?nome=valor&nome1=valor1&nome2=valor2

Http-Post: é semelhante ao protocolo Http-Get mas ao invés de enviar as informações na própria URL, insere os pares de parâmetro/valor na requisição HTTP. Ex:

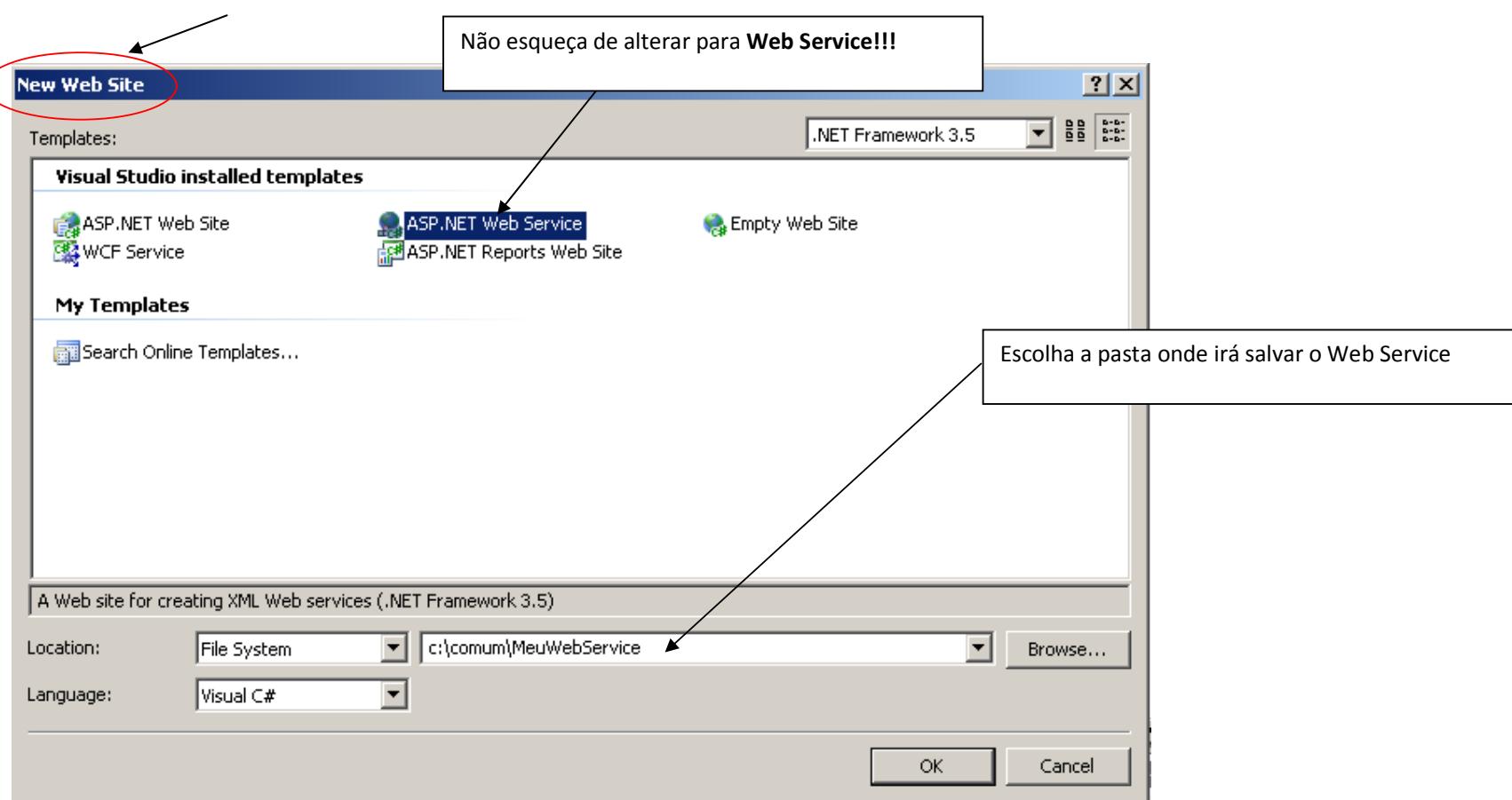
```
<form action="teste.asmx">
  <input type="hidden" name="nome" value="valor">
  <input type="submit">
</form>
```

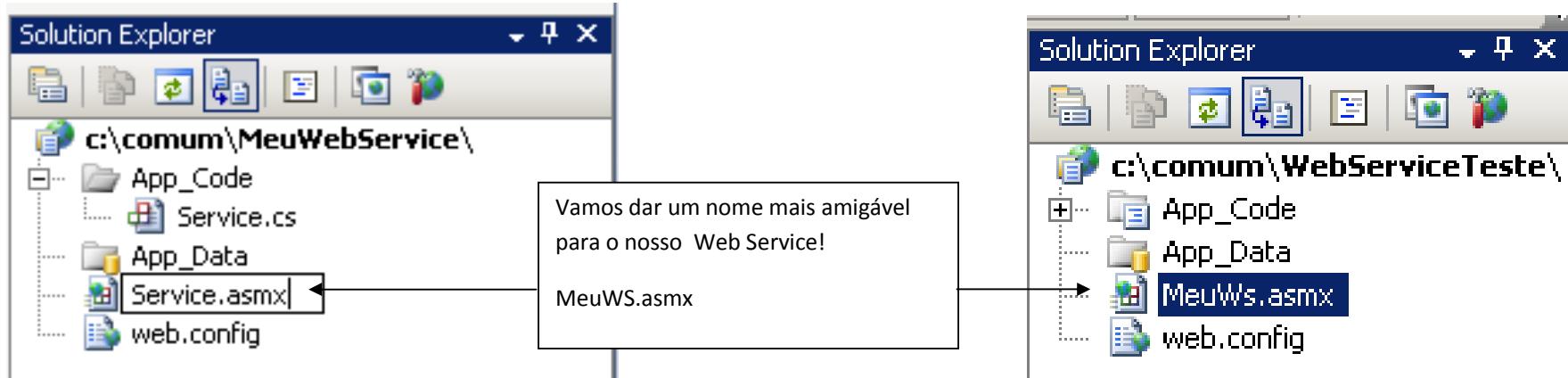
O formulário HTML acima insere um parâmetro oculto (hidden) chamado "nome" com o valor "valor" na requisição HTTP e quando o formulário é enviado (através do

botão submit), o servidor é capaz de extrair os pares de parâmetro/valor enviados.

SOAP: o SOAP (Simple Object Access Protocol) é um padrão bem mais rico, pois ao invés de enviar apenas pares de parâmetro/valor, ele utiliza XML para transmitir informações, permitindo a transmissão de objetos mais complexos como classes, objetos, tipos ricos de dados, etc.

Criando um Web Service:





Vamos colocar os nossos métodos publicados no **Service.cs** que está na pasta APP_Code

Ele já traz o seguinte código nesse arquivo:

```
12 public class Service : System.Web.Services.WebService
13 {
14     public Service () {
15
16         //Uncomment the following line if using designed components
17         //InitializeComponent();
18     }
19
20     [WebMethod]
21     public string HelloWorld() {
22         return "Hello World";
23     }
24 }
25 }
```

Os métodos declarados com [WebMethod] são aqueles que o web service irá disponibilizar na WEB.

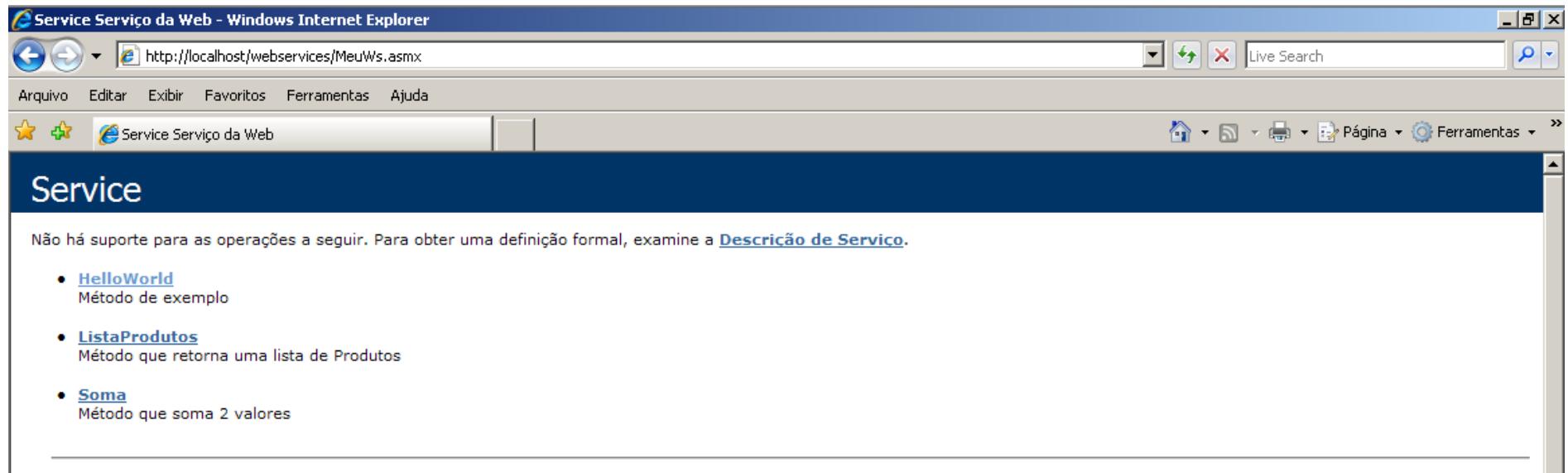
Criando um Método para somar 2 valores

Logo abaixo do método de exemplo HelloWorld, coloque o método a seguir:

```
[WebMethod(Description="Método de exemplo")]
public string HelloWorld() {
    return "Hello World";
}

[WebMethod(Description = "Método que soma 2 valores" )]
public int Soma (int valor1, int valor2)
{
    return valor1 + valor2;
}
```

Vamos agora publicar o WebService no IIS para podermos testá-lo em outra aplicação WEB: Para publicá-lo siga os passos explicados no tópico anterior sobre como publicar Web sites. Após publicar, você pode testar digitando o endereço do seu WebService no navegador web:

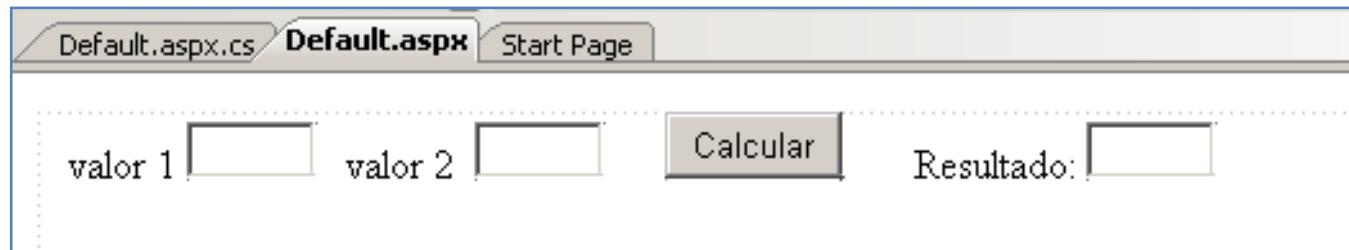


The screenshot shows a Microsoft Internet Explorer window titled "Service Serviço da Web - Windows Internet Explorer". The address bar contains the URL "http://localhost/webservices/MeuWs.asmx". The main content area displays a service page for a Web Service named "Service". The page header says "Service" and the sub-header says "Não há suporte para as operações a seguir. Para obter uma definição formal, examine a [Descrição de Serviço](#)". Below this, there is a list of methods:

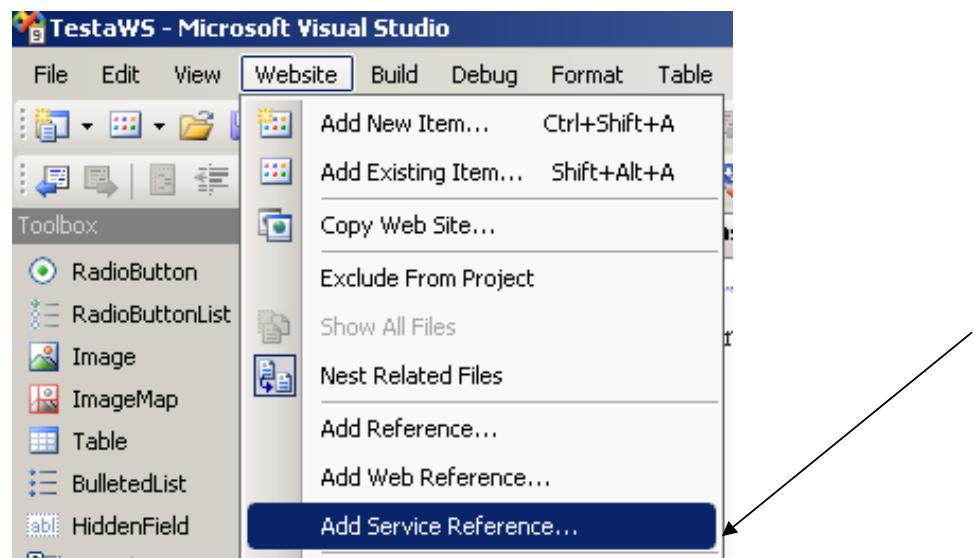
- [HelloWorld](#)
Método de exemplo
- [ListaProdutos](#)
Método que retorna uma lista de Produtos
- [Soma](#)
Método que soma 2 valores

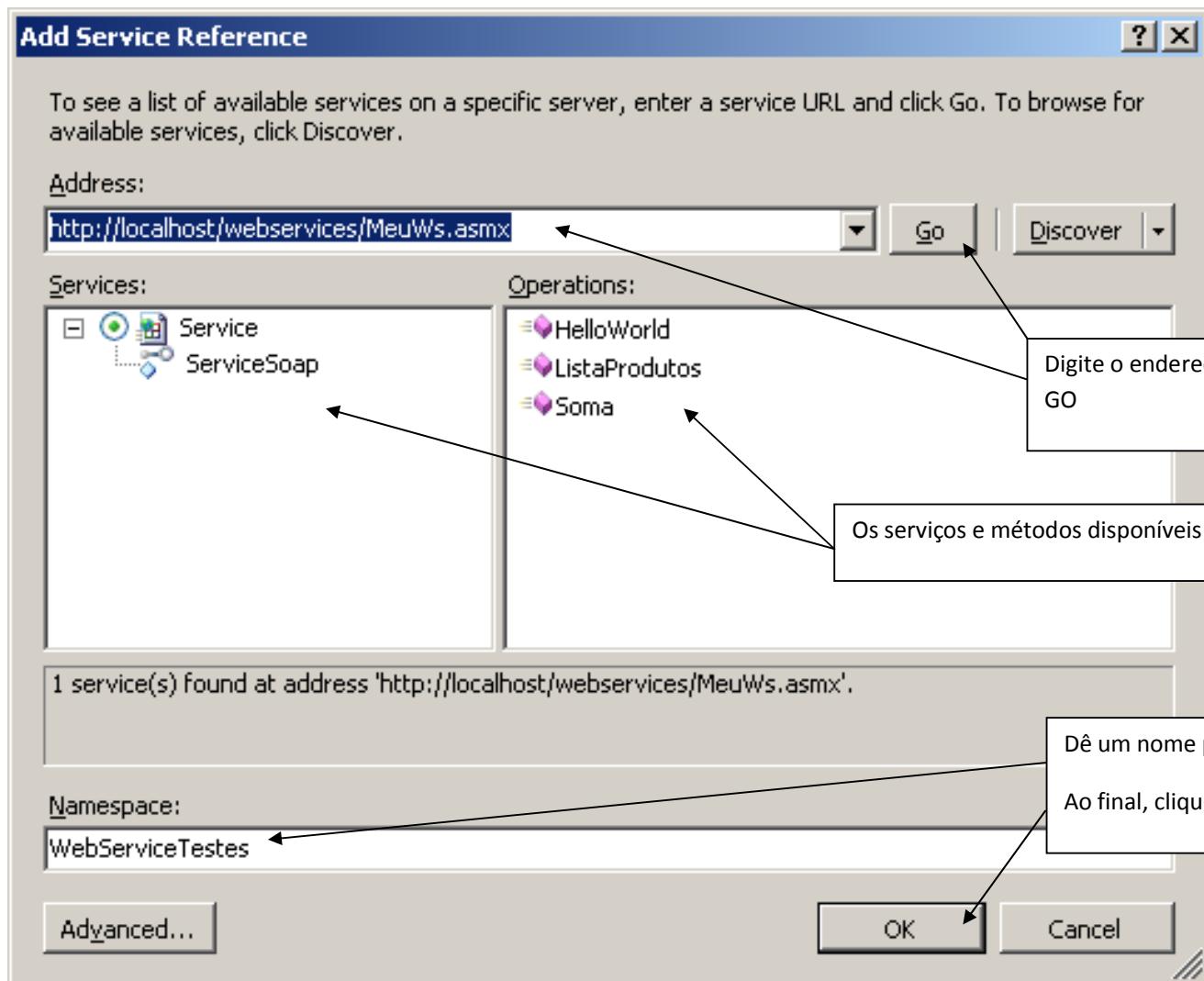
Consumindo o Web Service

Para consumir (utilizar) o web service, vamos criar uma OUTRA aplicação WEB. Na página principal, coloque os seguintes objetos:



Agora, vamos adicionar ao nosso site uma referência ao nosso Web Service:





No código fonte da página, adicione o namespace:

```
using WebServiceTestes;
```

Adicione o seguinte código ao botão calcular:

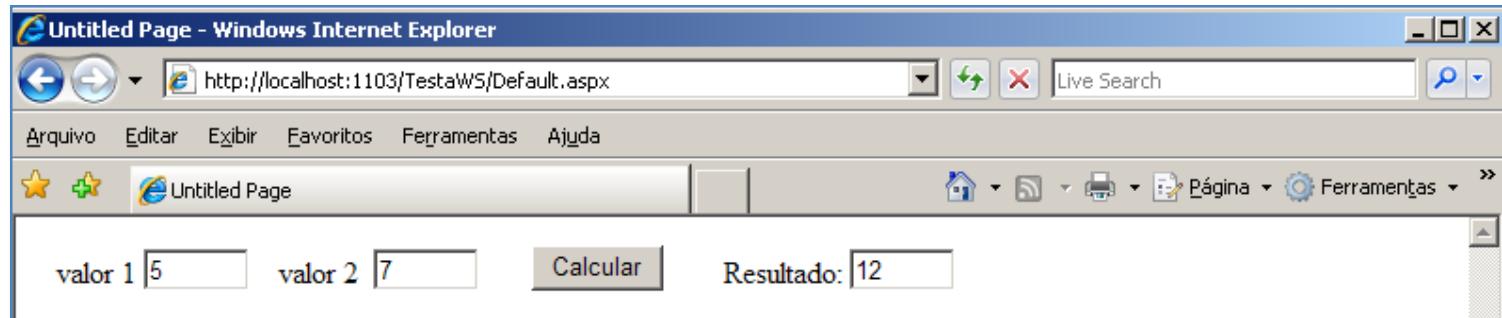
```
protected void Button1_Click(object sender, EventArgs e)
{
    ServiceSoapClient ws = new ServiceSoapClient();

    int v1 = Convert.ToInt32(TextBox1.Text);
    int v2 = Convert.ToInt32(TextBox2.Text);

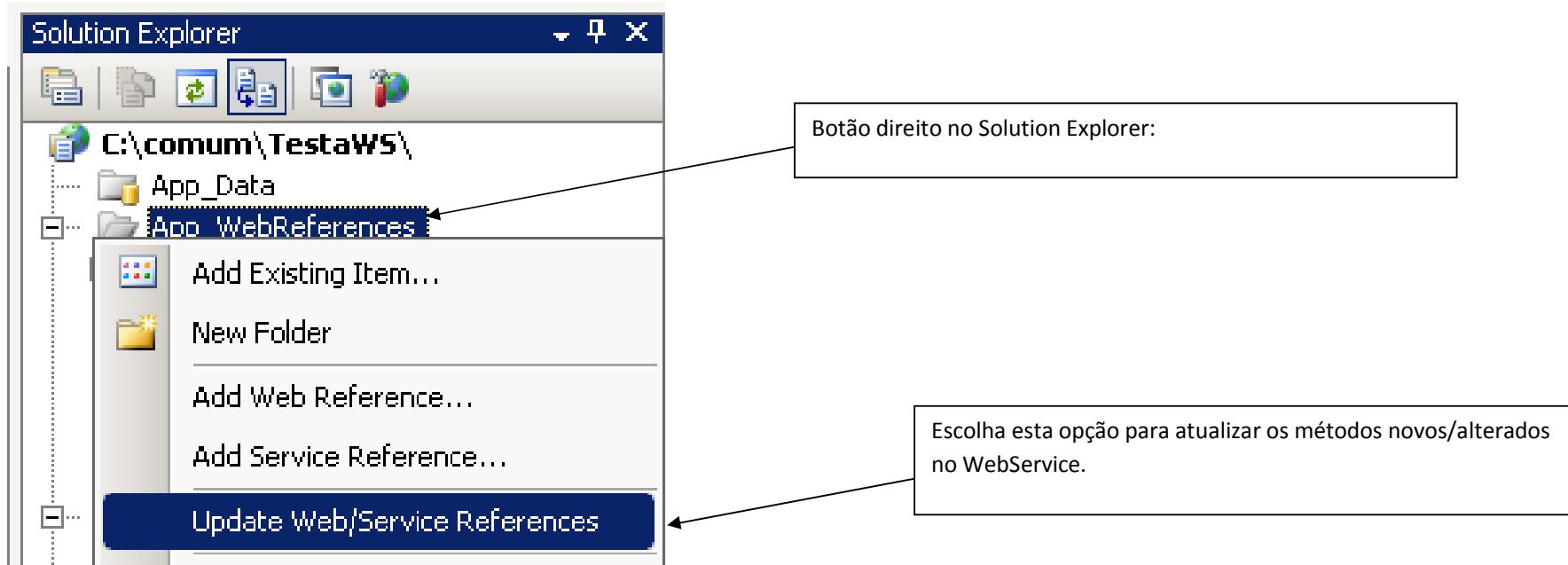
    int resp = ws.Soma(v1, v2);

    TextBox3.Text = resp.ToString();
}
```

Execute a aplicação, preencha os valores e clique no botão!

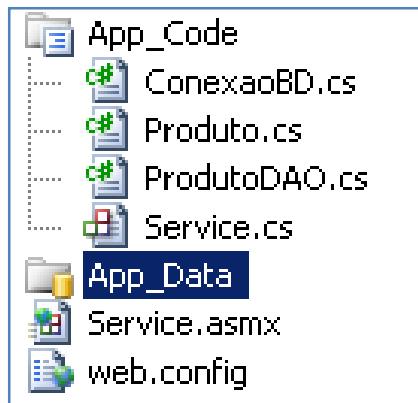


Se criar novos métodos no seu Web Service e quiser utilizá-los aqui, basta atualizar as referências:



Web Service para realizar a inclusão

Criando o web service:



Crie um web service como descrito anteriormente. Crie as classes como mostra a figura ao lado. O código delas será listado a seguir:

```
public class ConexaoBD
{
    public static SqlConnection GetConexao()
    {
        string str = WebConfigurationManager.ConnectionStrings["MINHACONEXAO"].ToString();

        SqlConnection conexao = new SqlConnection(str);
        conexao.Open();
        return conexao;
    }
}
```

```
public class Produto
{
    int produtoId;
    string produtoNome;
    double produtoValor;

    public int ProdutoId
    {
        get { return produtoId; }
        set { produtoId = value; }
    }
}
```

```

    }

    public string ProdutoNome
    {
        get { return produtoNome; }
        set { produtoNome = value; }
    }

    public double ProdutoValor
    {
        get { return produtoValor; }
        set { produtoValor = value; }
    }
}

```

```

public class ProdutoDAO
{
    public static void InserirProduto(Produto p)
    {
        SqlConnection conexao = ConexaoBD.GetConexao();

        string sql = string.Format("insert into Produtos (ProdutoNome, ProdutoValor)"
            + " values ('{0}', {1})", p.ProdutoNome,p.ProdutoValor.ToString().Replace(',', '.'));

        SqlCommand comando = new SqlCommand(sql, conexao);
        comando.ExecuteNonQuery();

        conexao.Close();
    }
}

```

Na classe Service.cs, vamos publicar nosso método de inserir produto:

```

[WebMethod(Description="Cadastrar um produto no Banco de dados")]
public void InserirProduto(Produto p)
{
    ProdutoDAO.InserirProduto(p);
}

```

Consumindo o web service:

Na sua página principal (na página onde você irá testar seu Web Service), crie um form simples para entrada dos dados do produto.

Adicione a referência do web service como já visto anteriormente.

Digite o seguinte código no botão:

The diagram illustrates the connection between an ASPX page and its code-behind file. On the left, the 'Default.aspx' page is shown with two text boxes labeled 'Nome do produto:' and 'Valor do produto:', and a button labeled 'Inserir Produto'. An arrow points from this interface to the code-behind on the right.

```
protected void Button1_Click(object sender, EventArgs e)
{
    Produto p = new Produto();

    p.ProdutoNome = TextBoxNome.Text;
    p.ProdutoValor = Convert.ToDouble(TextBoxValor.Text);

    ServiceSoapClient ss = new ServiceSoapClient();
    ss.InserirProduto(p);
}
```

Listas tipadas em Web Services

Criando o Web service (continuando o exemplo do web service anterior...):

Adicione o namespace `using System.Collections.Generic;` nas classes:

- ProdutoDAO
- Service.cs

Na classe `ProdutoDAO`, vamos criar um método para retornar uma lista de produtos:

```
public static List<Produto> ListaDeProdutos()
{
    SqlConnection conexao = ConexaoBD.GetConexao();
    string sql = "select * from produtos";
    SqlCommand comando = new SqlCommand(sql, conexao);
    SqlDataReader leitor = comando.ExecuteReader();

    List<Produto> lista = new List<Produto>();

    Produto p;
    while (leitor.Read())
    {
        p = new Produto();
        p.ProdutoId = Convert.ToInt32(leitor["ProdutoId"]);
        p.ProdutoNome = leitor["ProdutoNome"].ToString();
        p.ProdutoValor = Convert.ToDouble(leitor["ProdutoValor"]);

        lista.Add(p);
    }
    return lista;
}
```

No arquivo `Service.cs`, vamos publicar o método:

```
[WebMethod(Description = "Lista de produtos")]
public List<Produto> ListaDeProdutos()
{
    return ProdutoDAO.ListaDeProdutos();
}
```

Publique o Web Service.

Consumindo o Web Service:

Na página default.aspx, coloque um grid e um botão:

Antes de digitar o código, atualize as referências do web service!

Exibir lista de produtos		
Column0	Column1	Column2
abc	abc	abc

```
protected void Button2_Click(object sender, EventArgs e)
{
    ServiceSoapClient ss = new ServiceSoapClient();

    // o retorno do web service vem como um vetor.
    // o código abaixo transforma o vetor em uma lista do tipo Produto.
    GridView1.DataSource = ss.ListaDeProdutos().ToList<Produto>();

    GridView1.DataBind();
}
```

Ao clicar no botão, o sistema irá exibir a relação de produtos no grid:

Exibir lista de produtos		
ProdutoId	ProdutoNome	ProdutoValor
1	camisa	150
2	boné	15

É possível fazer a mesma coisa com o DropDownList. Só lembre-se de transformar o retorno do método em uma lista!