4. Controles

Uma novidade do ASP.NET 2.0 foi a adição de cerca de 50 novos controles de servidor. Alguns presentes na versão 1.1, como o DataGrid, não estão mais disponíveis na caixa de ferramentas, porém podem ser facilmente adicionados.

Não estudaremos detalhadamente todos os Server Controls.

Você pode criar uma aplicação utilizando basicamente dois grandes grupos de controles: controles HTML, que são os controles padrão conhecidos e Server Controls, ou controles de servidor.

Você pode criar aplicativos usando apenas controles HTML, porém tenha um bom motivo para fazer isto.

Server Controls são controles ricos em funcionalidade, que podem ter seus eventos ou propriedades manipulados no código executado no servidor. Todo controle de servidor é codificado através de uma tag ASP, que não é HTML padrão que conhecemos. Em tempo de execução, o ASP.NET renderiza esta tag em HTML otimizado para o navegador em que a aplicação é executada.

Por exemplo, um dos controles mais simples é o Label. Ao colocar um Label em um WebForm o VS gera o seguinte código:

```
<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
```

Você pode gerenciar a apresentação de seu site manualmente, se preferir.

Ao executarmos a aplicação, a tag ASP:Label é renderizado em uma tag span:

```
<span id="Label1">Label</span>
```

O atributo runat=Server não define que o controle é um controle de servidor, mas sim que ele será visível no código executado no servidor. Isto significa que se adicionarmos este atributo a um controle HTML, este também poderá ser manipulado no servidor, porém de forma mais pobre:

```
<input id="Text1" runat="server" type="text" />
```

Não estudaremos controles HTML.

Acessando eventos e propriedades de controles

Na caixa de propriedades, clicando no botão *Events*  , você tem uma relação de todos os eventos possíveis para um determinado controle. Para que o VS crie automaticamente um manipulador para o evento basta dar um duplo clique no

nome. Alguns manipuladores podem ser criados quando clicamos sobre o próprio controle, como o evento Click de um Button:

VB
```
Protected Sub Button1_Click(ByVal sender As Object,ByVal e _
As System.EventArgs) Handles Button1.Click

End Sub
```

C#
```
protected void Button1_Click(object sender, EventArgs e)
{

}
```

Junto com o evento são passadas duas propriedades, sender, do tipo object, que contem o objeto que disparou o evento, e *e*, do tipo EventArgs[13], que contem algumas propriedades especificas do evento.

Ler ou alterar propriedades de um controle é feito no modelo controle.propriedade, como nos exemplos abaixo:

VB
```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e _
As System.EventArgs) Handles Button1.Click
      Button1.Text = "Gravar"
      Button1.ToolTip = "Clique Aqui para Gravar"
End Sub
```

C#
```
protected void Button2_Click(object sender, EventArgs e)
{
      Button1.Text = "Gravar";
      Button1.ToolTip = "Clique Aqui para Gravar";
}
```

Você pode definir que mais de um controle dispare um mesmo manipulador de evento, internamente basta você verificar qual foi o controle que causou o evento. De uma maneira geral, para ambas as linguagens, basta você selecionar o evento no dropdown na caixa de propriedades. O interessante é como o VS trata isto em tempo de design de forma tão diferenciada. No C# o nome do manipulador do evento é colocada no arquivo ASPX, no atributo OnClick. No VB.NET, é adicionado o nome do controle ao grupo handles, ao final da assinatura do evento. O resultado em tempo de execução obviamente é o mesmo.

Propriedades Comuns

Algumas propriedades são comuns, se não a todos, mas a maioria dos controles. Vamos vê-las agora:

- ID: O nome do controle, deve ser único
- BackColor: Cor de fundo do controle
- BorderColor: Cor de Borda do Controle
- BorderStyle: Estilo da borda do Controle
- BorderWidt: Largura da borda do controle

[13] EventArgs é a classe base de um evento, este argumento pode ser um derivado desta classe.

- CssClass: Indica a classe CSS a ser aplicada no controle
- Enabled: indica se o controle esta habilitado
- EnableTheming: Indica se é o tema do controle estará ativo
- EnableViewState: Indica se o controle vai armazenar seu estado no viewstate da pagina
- Font: Fonte do controle
- SkinID: Indica o ID do skin a ser utilizado. Skins serão estudados mais adiante
- Text: Representa o texto que será exibido ao usuário da aplicação
- ToolTip: é a dica que será exibida ao usuário quando este posicionar o mouse sobre o controle.

Uso de HotKeys

Duas propriedades novas e interessantes, que estão presentes em alguns controles, são AcessKey e AssociatedControlID. Com a primeira definimos uma tecla de atalho para o controle, que no navegador e acionado junto com a tecla ALT. Já com AssociatedControlID podemos definir qual o controle vai receber o foco no caso da combinação de teclas ser acionadas. Este recurso é conhecido como HotKeys.

Vamos estudar agora os três principais controles de servidor, que formam um conjunto básico para a criação de qualquer aplicativo.

Label

Um controle simples, porém poderoso, que nos permite exibir algum texto em uma página Web. Todas suas propriedades podem ser manipuladas no código gerado no servidor.

Alguns exemplos de codificação de controles, mostram primeiro em tempo de design (ASP) e em seguida em tempo de execução (HTML) .

È definido pela a tag ASP:Label e quando executado renderizado em uma tag span:

```
<asp:Label ID="Label1" runat="server" Text="ASP.NET 2.0
Rules!!"></asp:Label>
```

```
span id="Label1">ASP.NET 2.0 Rules!!</span>
```

TextBox

O textbox é um controle de edição simples. Através da propriedade TextMode podemos utilizá-lo ainda para receber senhas ou mesmo transforma-lo em um controle com múltiplas linhas. Uma novidade é a propriedade AutoCompleteType, que pode receber apenas valores pré-definidos. Ao preencher um segundo controle com o mesmo valor para esta propriedade, o navegador deverá sugerir o preenchimento do valor informado no primeiro campo.
È definido pela a tag ASP:TextBox e quando executado renderizado em uma tag input do tipo text ou password ou então textarea, de acordo com a propriedade textmode.

```
<asp:TextBox ID="TextBox1" runat="server">Normal</asp:TextBox>
<asp:TextBox ID="TextBox2" runat="server"
TextMode="Password">Senha</asp:TextBox>
<asp:TextBox ID="TextBox3" runat="server"
TextMode="MultiLine">MultiLine</asp:TextBox>
```

```
<input name="TextBox1" type="text" value="Normal" id="TextBox1" />
<input name="TextBox2" type="password" id="TextBox2" />
<textarea name="TextBox3" rows="2" cols="20"
id="TextBox3">MultiLine</textarea>
```

Button

O button é um botão de comando básico. Sua principal função é causar um post back para a execução de algum código no servidor. O principal evento é Click, que, obviamente, é disparando quando clicamos no botão. É renderizado em uma tag input do tipo submit.

```
<asp:Button ID="Button1" runat="server" Text="Button" />
```

```
<input type="submit" name="Button1" value="Button" id="Button1" />
```

Uma importante propriedade é a PostBackUrl, que indica para qual página será executado o post back[14].

Para exemplificar, o código abaixo lê a um TextBox de um formulário que originou o PostBack:

VB

```
Dim Nome As String = CType(PreviousPage.FindControl("Nome"), _
TextBox).Text
```

Praticando

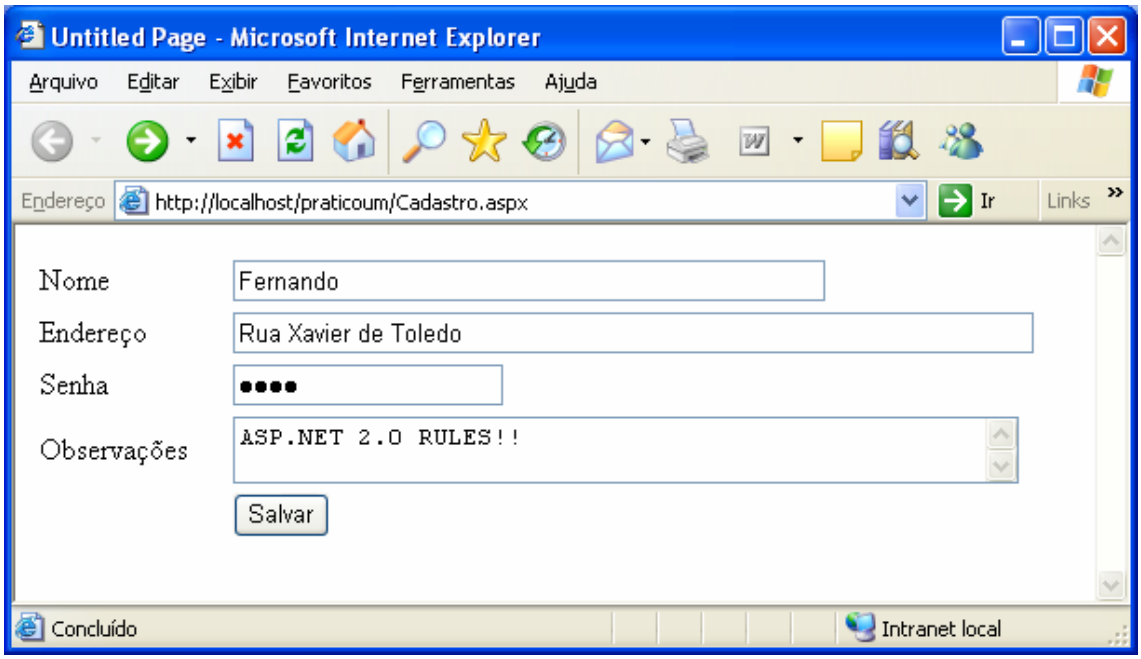
1. Crie uma aplicação ASP.NET 2.O com as seguintes características.

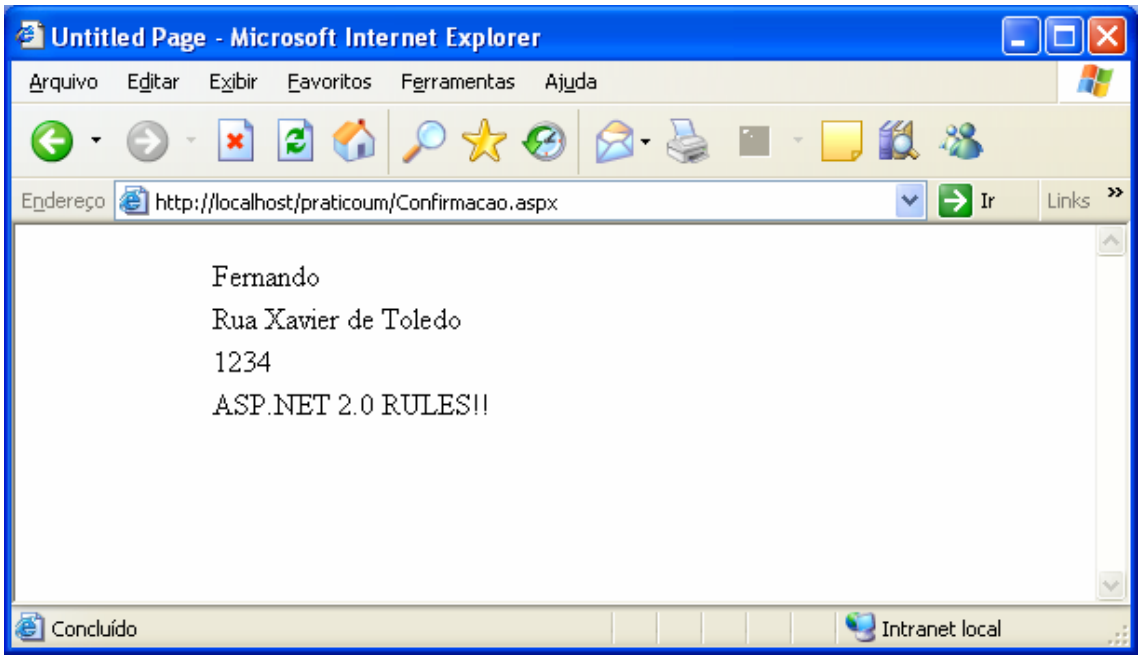
Um WebForm de Cadastro, denominado Cadastro.aspx, onde devem ser preenchidos:

- Nome do Usuário
- Endereço
- Senha
- Campo livre para observações, permitindo a entrada de varias linhas
- Um botão de comando “Confirmar”

Ao clicar em confirmar, o usuário deverá ser redirecionado para outra página, denominada Confirmacao.aspx, onde as informações preenchidas pelo usuário deverão ser exibidas para confirmação.

[14] O recurso de cross-page posting, ou post back entre diferentes páginas, é uma boa novidade do ASP.NET, já explanada em capitulo anterior.





Dicas:

- Configure Cadastro.aspx como página inicial
- Para executar o postback entre uma página e outra utilize a propriedade PostBackUrl do botão de comando.
- Utilize os recursos de HotKeys
- Abaixo o código utilizado no exemplo para recuperar as informações digitadas:

VB

```
Nome.Text = CType(PreviousPage.FindControl("Nome"), TextBox).Text
Endereco.Text = CType(PreviousPage.FindControl("Endereco"),_
    TextBox).Text
Senha.Text = CType(PreviousPage.FindControl("Senha"), TextBox).Text
Observacoes.Text = CType(PreviousPage.FindControl("Observacoes"),_
      TextBox).Text
```

LinkButton

O controle linkbutton é uma mistura de botão de comando com HyperLink. Sua aparência é de um Link, seu comportamento e suas propriedades o assemelham a um Button. Não existe propriedade NavigateUrl, o redirecionamento deve ser codificado no post back ou através da propriedade PostBackUrl.

```
<asp:LinkButton ID="LinkButton1"
runat="server">LinkButton</asp:LinkButton>
```

Nos bastidores, o LinkButton renderiza uma tag *a*, cujo atributo Href, ao invés de conter uma URL, dispara uma função Java Script gerada pelo ASP.NET.

```
<a id="LinkButton1"
href="javascript:__doPostBack('LinkButton1','')">LinkButton</a>
```

ImageButton

Um ImageButton é um botão de comando onde podemos adicionar uma imagem para exibição. Possui todas as funcionalidades do Button, mais a propriedade ImageURL onde definimos a imagem. Em tempo de execução é renderizado como Input do tipo Image.

```
<asp:ImageButton  ID="ImageButton1"  runat="server"  ImageUrl="web.gif"
/>
```

```
<input type="image" name="ImageButton1" id="ImageButton1" src="web.gif" />
```

HyperLInk

O hyperlink permite criar links de navegação, sua propriedade mais importante é NavagateURL, que é valor para a atribuo HREF. É um controle simples que é renderizado com uma tag *a*. Seu poder maior esta na possibilidade de, por exemplo, definir a URL dinamicamente no código do servidor. Através de uso de HotKey pode-se ainda facilitar a navegação na pagina. A propriedade text define o texto de exibição, já ImageUrl pode definir uma imagem a ser exibida ao invés do texto.

```
<asp:HyperLink ID="HyperLink1" runat="server"
NavigateUrl="http:\\www.fernandoamaral.com.br">Clique
Aqui</asp:HyperLink>
```

```
<a id="HyperLink1" href="http:\\www.fernandoamaral.com.br">Clique Aqui</a>
```

CheckBox

O checkbox gera um controle HTML input do tipo checkbox

```
<asp:CheckBox ID="CheckBox1" runat="server" Text="Lembrar" />
```

```
<input id="CheckBox1" type="checkbox" name="CheckBox1" />
```

RadioButton

O RadioButton gera um controle HTML input do tipo radio

```
<asp:RadioButton ID="RadioButton1" runat="server" Text="Lembrar" />
```

```
<input id="RadioButton1" type="radio" name="RadioButton1"
value="RadioButton1" />
```

Image

O controle Image deve ser utilizado para exibição de Imagens. A imagem a ser exibida deve ser definida através da propriedade ImageUrl.

```
<asp:Image ID="Image1" runat="server" ImageUrl="web.gif" />
```

```
<img id="Image1" src="web.gif" />
```

DropDownList e ListBox

O DropDownList e ListBox são controles altamente poderoso. Suas propriedades e eventos trazem recursos que só poderiam ser obtidos com muita mão de obra em uma aplicação WEB. A propriedade AutoPostBack, causa um post back automático ao servidor quando um valor é selecionado. Os controles podem ter seus itens adicionados em tempo de design ou de execução, através da propriedade Itens. Também é possível vinculá-lo a uma fonte de dados qualquer, que pode ser um array, um dataset ou mesmo um SqlDataSource, só para citar alguns.

Para cada item é possível ler e escrever um texto e um valor, armazenados nas propriedades text e value, respectivamente. Isto é muito útil, por exemplo, para recuperar o código de um item selecionado pelo usuário.

```
<asp:DropDownList ID="DropDownList1" runat="server">
```

```
<select name="DropDownList1" id="DropDownList1">
```

O exemplo abaixo adiciona um item a um DropDownList:

VB
```
DropDownList1.Items.Add("Selecione uma opção")
```

C#
```
DropDownList1.Items.Add("Selecione uma opção");
```

Para recuperar o valor da propriedade text bem como value basta ler as propriedades específicas:

```vb
Dim s As String = DropDownList1.SelectedValue
Dim t As String = DropDownList1.SelectedItem.Text
```

```cs
string s = DropDownList1.SelectedValue;
string t = DropDownList1.SelectedItem.Text;
```

O evento SelectedIndexChange é executado quando um item do controle é selecionado. Se a propriedade AutoPostBack estiver marcada como verdadeira, um postback é executado imediatamente, caso contrário o evento só será executado no próximo postback.

Existem ainda diversos métodos interessantes, como itens.clear, que limpa os itens, e propriedades, como itens.count, que retorna o número de itens no controle.

CheckBoxList e RadioButtonList

Os controles CheckBoxList e RadioButtonList em tempo de design são apresentados como um conjunto de controles CheckBox e RadioButton, que podem ter suas propriedades gerenciadas mais facilmente e conectados a uma fonte de dados qualquer. Em tempo de execução, o ASP.NET renderiza cada controle como diversos controles input do tipo checkbox ou radiobutton.

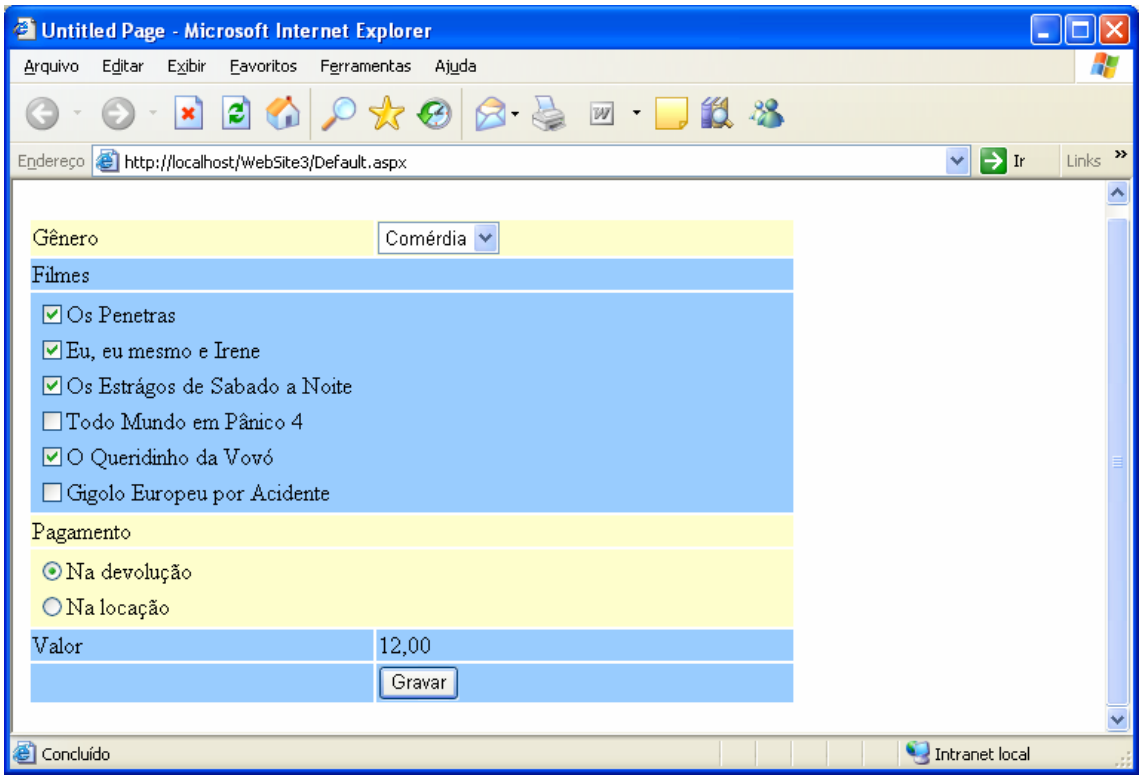
No exemplo abaixo o código verifica se um determinado item de um

```vb
if (CheckBoxList1.Items[0].Selected==true)
{
//
}
```

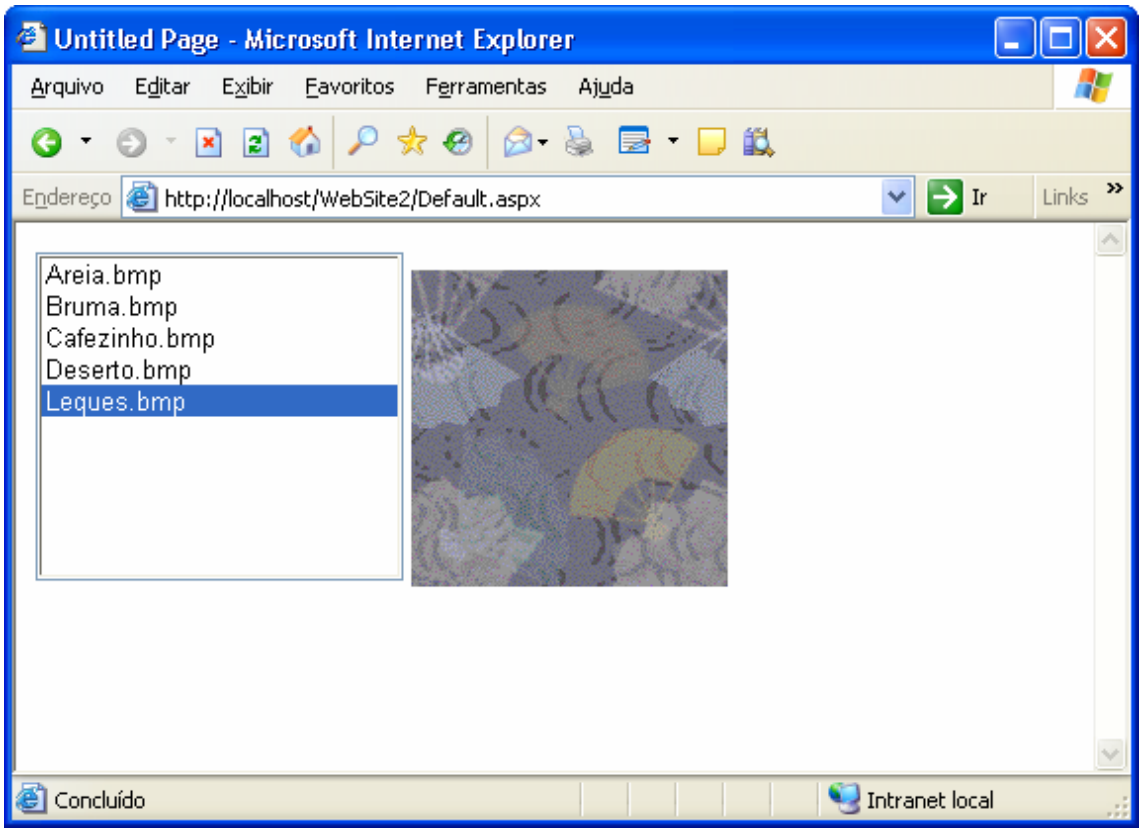
```cs
If CheckBoxList1.Items(0).Selected = True Then
''
End If
```

Praticando

1. Crie um aplicativo para uma locadora, onde o usuário seleciona um gênero em um DropDownList, em seguida o sistema preenche um CheckBoxList com a relação dos filmes do gênero selecionado. O usuário então seleciona os filmes que deseja locar. Um RadioButtonList, o mesmo informa a forma de pagamento, o sistema calcula o valor total da locação. Ao clicar em gravar, o software verifica se todos os dados foram preenchidos corretamente, e emite um recibo de locação, que deve ser desenvolvido em outro WebForm.



2. Crie uma nova aplicação ASP.NET 2.0. No diretório raiz crie uma pasta Imagens. Coloque algumas imagens de no máximo 300 pixels (largura e altura) neste diretório. Faça o ASP.NET ler a relação de imagens no diretório e preencher um ListBox. Ao selecionar a imagem no ListBox, o sistema deverá exibi-la.



Abaixo segue um exemplo de como você pode ler os arquivos de um diretório e lista-los em um Listbox:

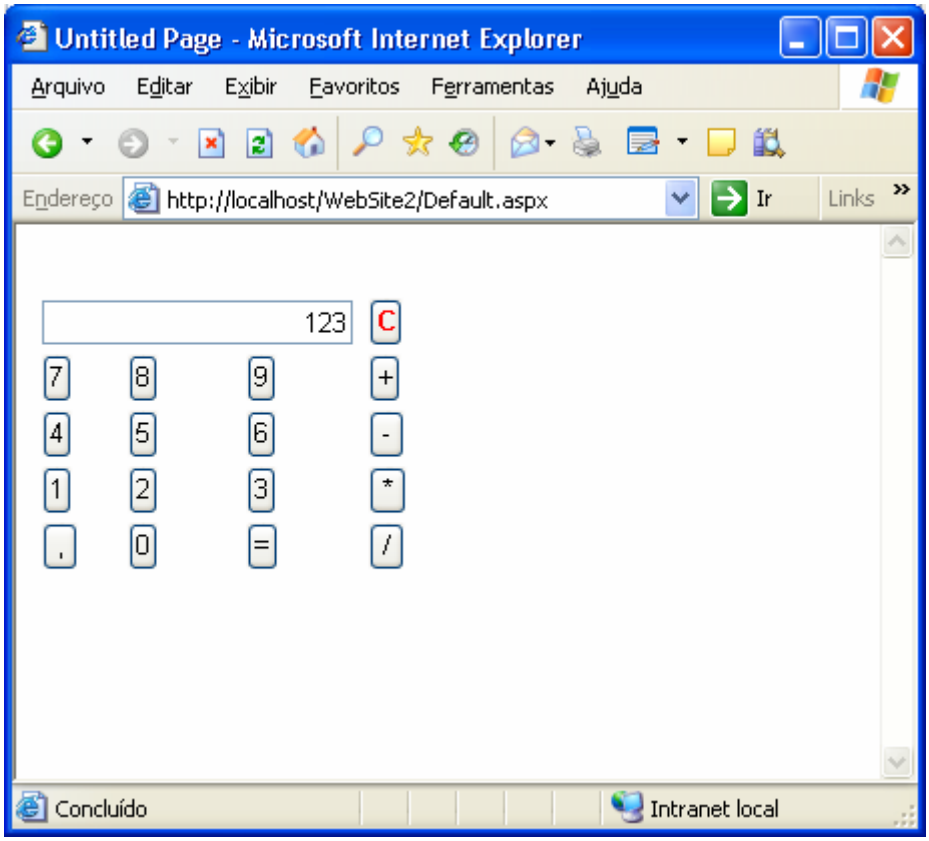
VB
```
If Not Page.IsPostBack Then
      Dim diretorio As New DirectoryInfo(Server.MapPath("Imagens"))
      With ListBoxImagens
            .DataSource = diretorio.GetFiles("*.bmp")
            .DataBind()
      End With
End If
```

C#
```
if ( ! Page.IsPostBack)
{
      DirectoryInfo diretorio = new
      DirectoryInfo(Server.MapPath("imagens"));
      ListBoxImagens.DataSource = diretorio.GetFiles("*.bmp");
      ListBoxImagens.DataBind();
}
```

É preciso incluir o namespace system.io para utilizar a classe DirectoryInfo.

3. Faça uma calculadora com operações básicas, para rodar em ambiente web.

O ideal seria implementar a calculadora utilizando JavaScript, pois não há justificativas para “idas e voltas” ao servidor. Porém como nosso objetivo é aprender ASP.NET 2.0, faça a mesma em ASP.NET 2.0!.



Calendar

O Calendar é um calendário que é renderizado utilizando uma tag *table*, e na sua forma mais simples tem pelo menos 7 KB de texto. É excelente para exibição e entrada de datas. Possuiu um conjunto poderosíssimo de propriedades, métodos e eventos que permitem praticamente exibi-lo de qualquer forma imaginada.

Entre seus eventos mais significativos, temos o SelectionChange, que é disparado quando ocorre uma alteração na seleção do calendário, e VisibleMonthChange, que ocorre quando o mês visível no calendário é alterado. A propriedade SelecteDate nos permite ler a data selecionada.

No exemplo abaixo um label exibe a data selecionada pelo usuário:

VB
```
Protected Sub Calendar1_SelectionChanged(ByVal sender As _
Object, ByVal e As System.EventArgs) Handles _
Calendar1.SelectionChanged
      Lbldata.Text = String.Format("A data selecionada é {0:d}",_
      Calendar1.SelectedDate)
End Sub
```

C#
```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
      Lbldata.Text = String.Format("A data selecionada é {0:d}",
      Calendar1.SelectedDate);
}
```

Através das propriedades SelectionMode podemos definir se o usuário poderá selecionar um único dia, uma semana, um mês inteiro ou nada. Caso ele possa selecionar, por exemplo, uma semana, podemos recuperar os dias selecionados através de um laço *for each*, que percorre a coleção de objetos DateTime da propriedade SelectedDates:

VB
```
Dim s As System.DateTime
For Each s In Calendar1.SelectedDates
      ListBox1.Items.Add(String.Format("Data Selecionada {0:d}", s))
Next
```

C#
```
foreach (System.DateTime s in Calendar1.SelectedDates)
{
      ListBox1.Items.Add(String.Format("Data Selecionada {0:d}", s));
}
```

AdRotator

O AdRotator nos permite incluir um banner em nosso site, onde o próprio ASP.NET através de um arquivo de configuração, vai cuidar da rotatividade das exibições de imagens. O segredo todo está em um arquivo de configuração, que pode ser um XML ou mesmo uma tabela de banco de dados.

Para testar este controle, crie um arquivo XML dentro do diretório App_Data de sua aplicação (se não existir o diretório, crie-o também), com o seguinte texto:

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
    <Ad>
      <ImageUrl>~/imagens/cafezinho.bmp</ImageUrl>
      <NavigateUrl>http://www.cafezinho.com</NavigateUrl>
      <AlternateText>Café</AlternateText>
      <Keyword>Café</Keyword>
      <Impressions>40</Impressions>
    </Ad>
  <Ad>
    <ImageUrl>~/imagens/deserto.bmp</ImageUrl>
    <NavigateUrl>http://www.deserto.com</NavigateUrl>
    <AlternateText>Deserto</AlternateText>
    <Keyword>Site Deserto</Keyword>
    <Impressions>20</Impressions>
  </Ad>
  </Advertisements>
```

Certifique-se que você tem uma pasta imagens com as imagens informadas. Você pode configurar os parâmetros de acordo com sua preferência: NavigateUrl é a URL para qual o usuário será direcionado quando clicar no banner, AlternateText é o texto a ser exibido no local da imagem. Impressions representa o número de impressões do banner. No exemplo acima, note que o primeiro banner deverá aparecer em dobro.

Para vincular o seu AdRotator ao arquivo XML basta informá-lo na propriedade AdvertisementFile.

Um arquivo XML é sensível a caracteres maiúsculos e minúsculos.

View e MultiView

Mais uma novidade do ASP.NET 2.0. Provavelmente você já necessitou que parte de sua aplicação ficasse invisível em determinados momentos. Claro que isso não era uma tarefa impossível, mas com os controles MultiView e View isto ficou mais simples.

A idéia é: Dentro de um Multiview você agrega um ou mais Views. Em cada um dos Views você coloca controles conforme a necessidade. A partir daí é só determinar qual View estará visível através da propriedade ActiveViewIndex.

VB
```
MultiView1.ActiveViewIndex = 0
```

C#
```
MultiView1.ActiveViewIndex = 0;
```

Wizard

Continuando com as novidades, o Wizard é um assistente para páginas Web, onde você pode determinar passos (Steps). Em cada Step você tem uma área onde pode colocar seus controles conforme a necessidade.

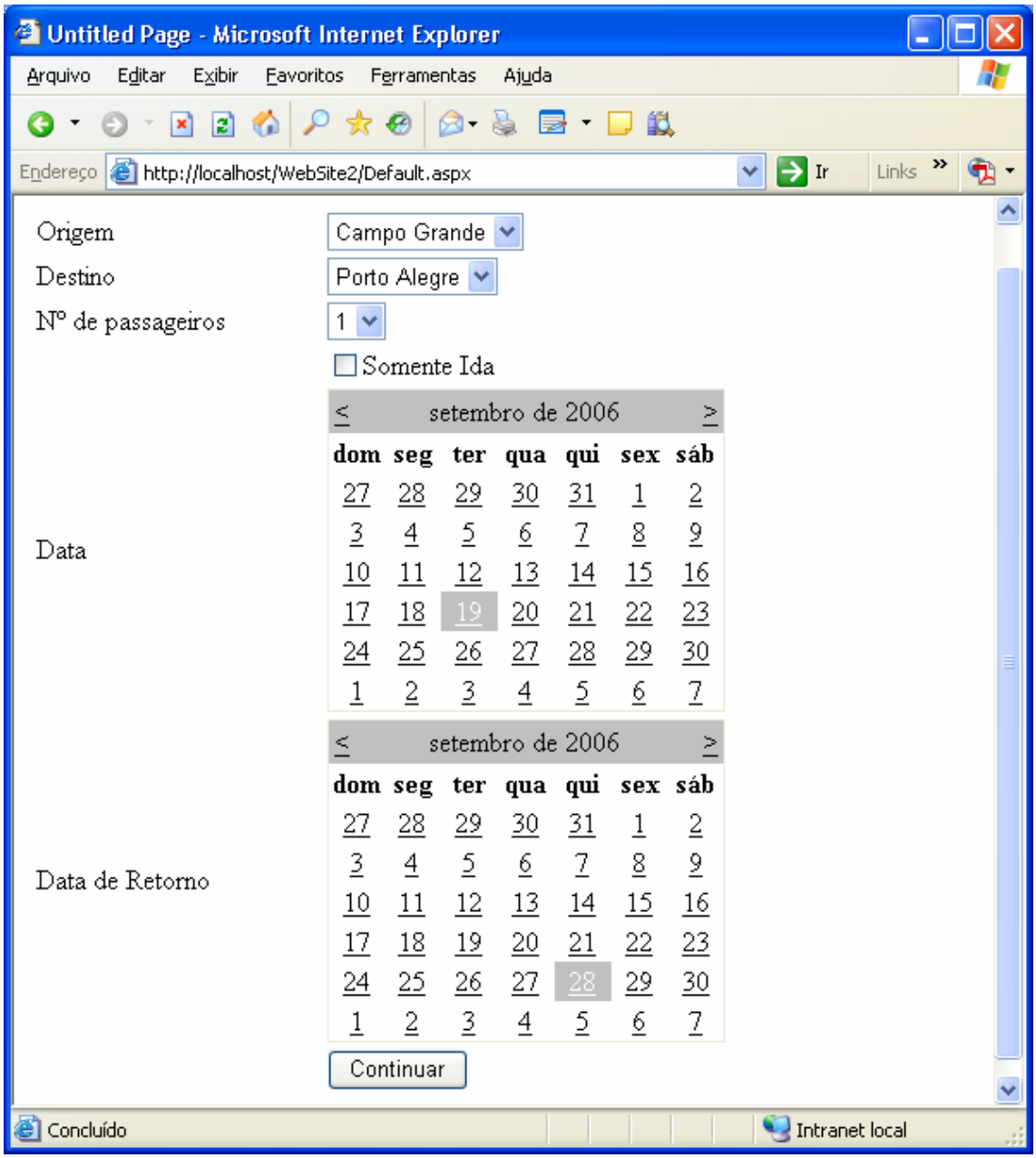
Entre os eventos tempos NextButtonClick, que é disparado sempre que o usuários clicar no botão Next, FinishButtonClick, disparado no encerramento, e CancelButtonClick, disparado ao final.

O mais interessante de tudo é que Wizard mantém estado de tudo o que é colocado em seus passos. Isto significa que você pode fazer o processamento dos dados ao final do assistente.

O controle é totalmente configurável, desde aparências até texto dos botões.

Praticando

1. Crie uma aplicação de pesquisa de hábitos alimentares. A pesquisa deve ser diferenciada para homens e mulheres. A primeira tarefa do usuário é informar o sexo, só então é exibida a pesquisa. Utilize controles Multiview e View para criar a aplicação.
2. Crie uma aplicação para uma companhia aérea vender passagens. No primeiro passo o cliente informa a origem, o destino, o número de passageiros, se a passagem é ida e volta ou apenas ida, a data da viagem e a data da volta. No segundo passo a aplicação mostra os horários com respectivos preços tanto para a ida quanto para a volta. No terceiro passo o usuário informa seus dados, bem como a forma de pagamento. No quarto e último passo a venda é confirmada. Use um controle Wizard para desenvolver a aplicação e controles Calendar para seleção das datas.



Controles de validação

O ASP.NET desde sua primeira versão tem integrador alguns controles que permitem validar entradas do usuário, tanto no cliente como no servidor, de forma simples e rápida.

Na versão 2.0 a principal novidade foi a inclusão da propriedade ValidationGroup, que permite que agrupemos grupos diferentes de validação em um mesmo WebForm. Na versão anterior isto era um problema, ou fazíamos a validação manualmente ou dividíamos a nossa lógica de negocio em dois formulários diferentes.

Os controles de validação são:

RequiredFieldValidator: Permite validar se um controle foi preenchido.

RangeValidator: Permite verificar se o valor informado esta entre determinado intervalo.
RegularExpressionValidator: Permite validar o valor informado com uma expressão regular
CompareValidator: Permite comparar dois valores.
CustomValidator: Permite que customizemos nossa expressão de validação.

Um controle de validação vai exibir uma mensagem caso a validação falhe. Com o controle ValidationSummary podemos agrupar as mensagens de todos os controles em um único local ou numa mesma caixa de dialogo.

Se você quer garantir que um controle tenha um valor preenchido e seja, por exemplo, um inteiro maior que zero, você terá que usar além de um RangeValidator um RequiredFiledValidador, apenas este último obriga o preenchimento do valor.

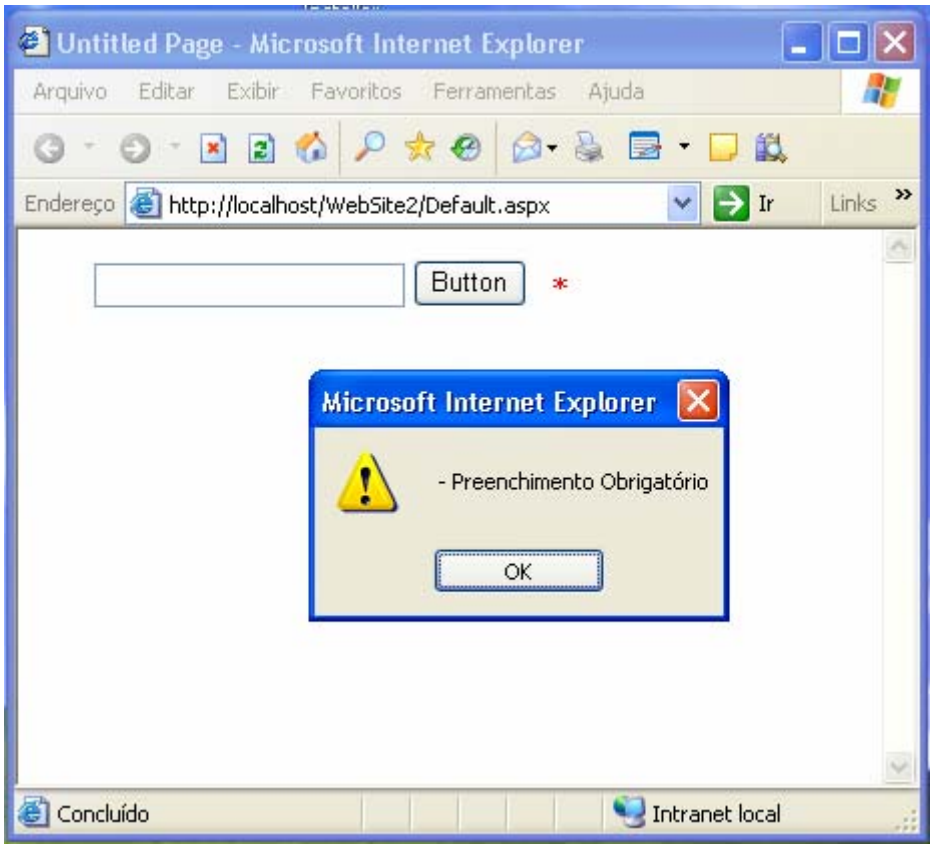
Um controle de validação tem duas propriedades para exibição de mensagens: Text e ErrorMessage. Se você utilizar um ValidationSummary você pode trabalhar com as duas: A mensagem em ErrorMessage será exibida no ValidationSummay, e text no proprio controle de validação (um * por exemplo).

Para criar grupos de validação, basta escolher um nome para o grupo e preencher as propriedades ValidationGroup tanto dos controles a serem validados quanto dos controles de validação.

Se você quiser causar um post back sem disparar os validadores, basta setar a propriedade CausesValidation do controle para False.

Para obrigar o preenchimento de um textbox:

- Coloque num formulário um controle TextBox, um RequiredFieldValidator e um ValidationSummary e um Button
- Configure a propriedade ControlToValidade do RequiredFieldValidator para o TextBox adicionado no furmaulario
- Preencha a propriedade text do RequiredFieldValidator para *, e ErrorMessage para “Preenchimento Obrigatório”
- Configure a propriedade ShowMessageBox do ValidationSummary para true, e ShowSummary para false.
- Rode a aplicação
- Clique no botão sem preencher qualquer informação no textbox

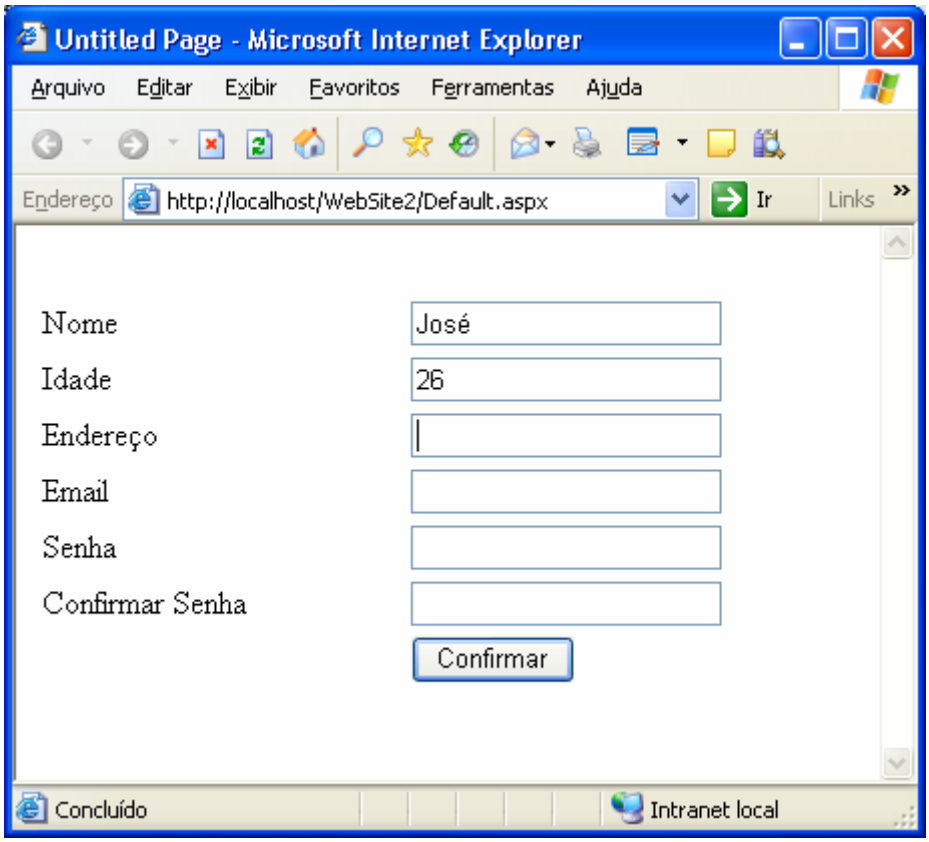


Por fim, para verificar também no servidor se todas as validações foram feitas[15], verifique a propriedade isvalid da página.

Praticando

1.Crie uma aplicação de cadastro de clientes, onde devem ser entrados os campos Nome, Idade, Endereço, Email e Senha e confirmação de senha. Os campos Nome, endereço, Email e senha são de preenchimento obrigatório. O campo idade deve ser preenchido com um valor inteiro entre 18 e 100. O email deve ser válido e a senha deve conicidir com sua confirmação.

[15] Esta é uma prática recomendada



2. Crie uma aplicação que começe com uma página de apresentação e que tenha um link para uma página de cadastro, onde são cadastrados alunos na parte supeiro da página e professores na parte inferios. No topo da página deve ser selecionado se quem esta preechendo é uma aluno ou professor. Coloque controles de validação para validar apenas os dados da opção escolhida. Coloque também um botão cancelar que retorna a página incial sem fazer qualquer validação.