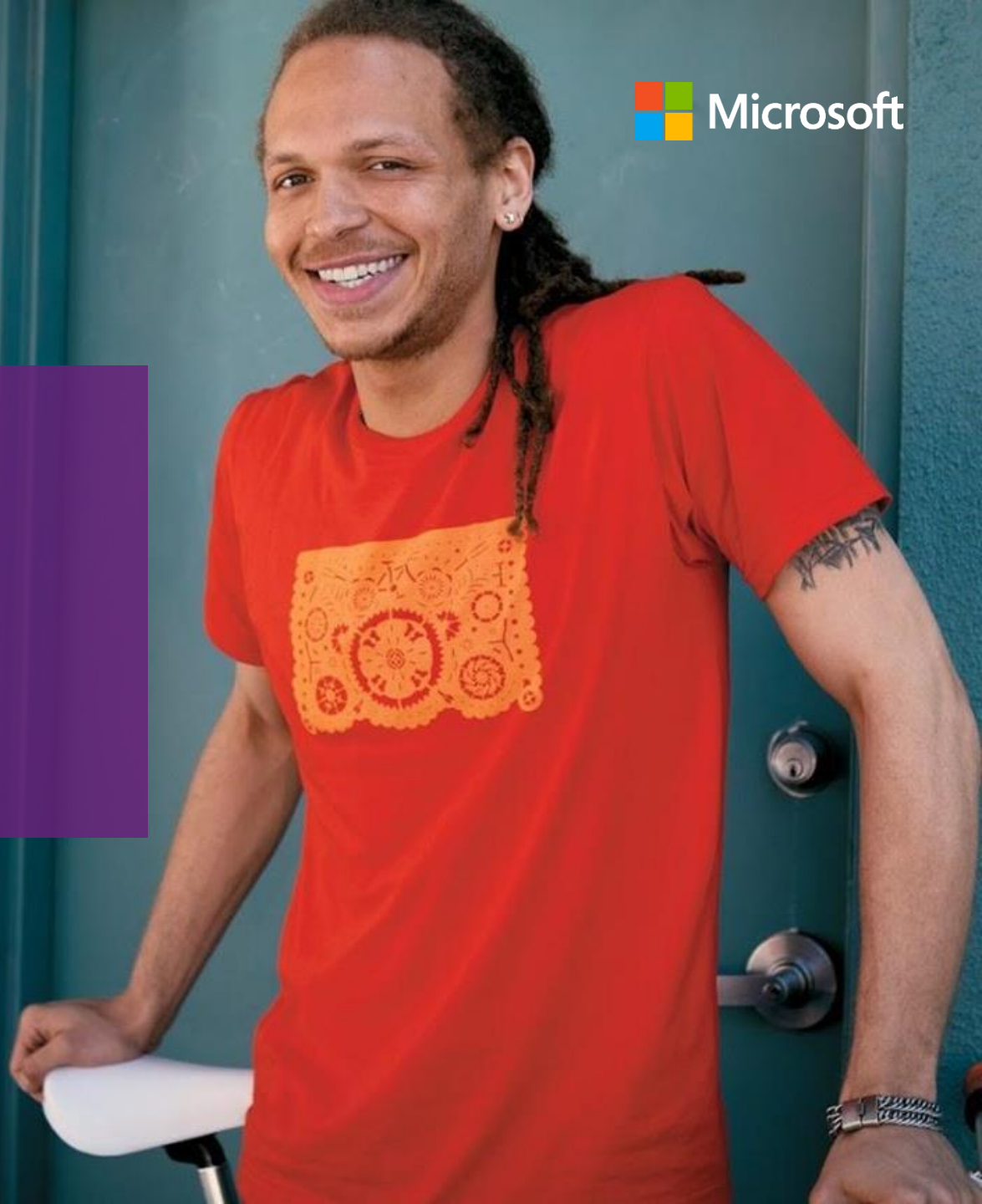
Microsoft Innovation Center PUCRS

# Microsoft Students to Business

Desenvolvimento de Sistemas – 2ª Fase

S2B Students to Business



Microsoft

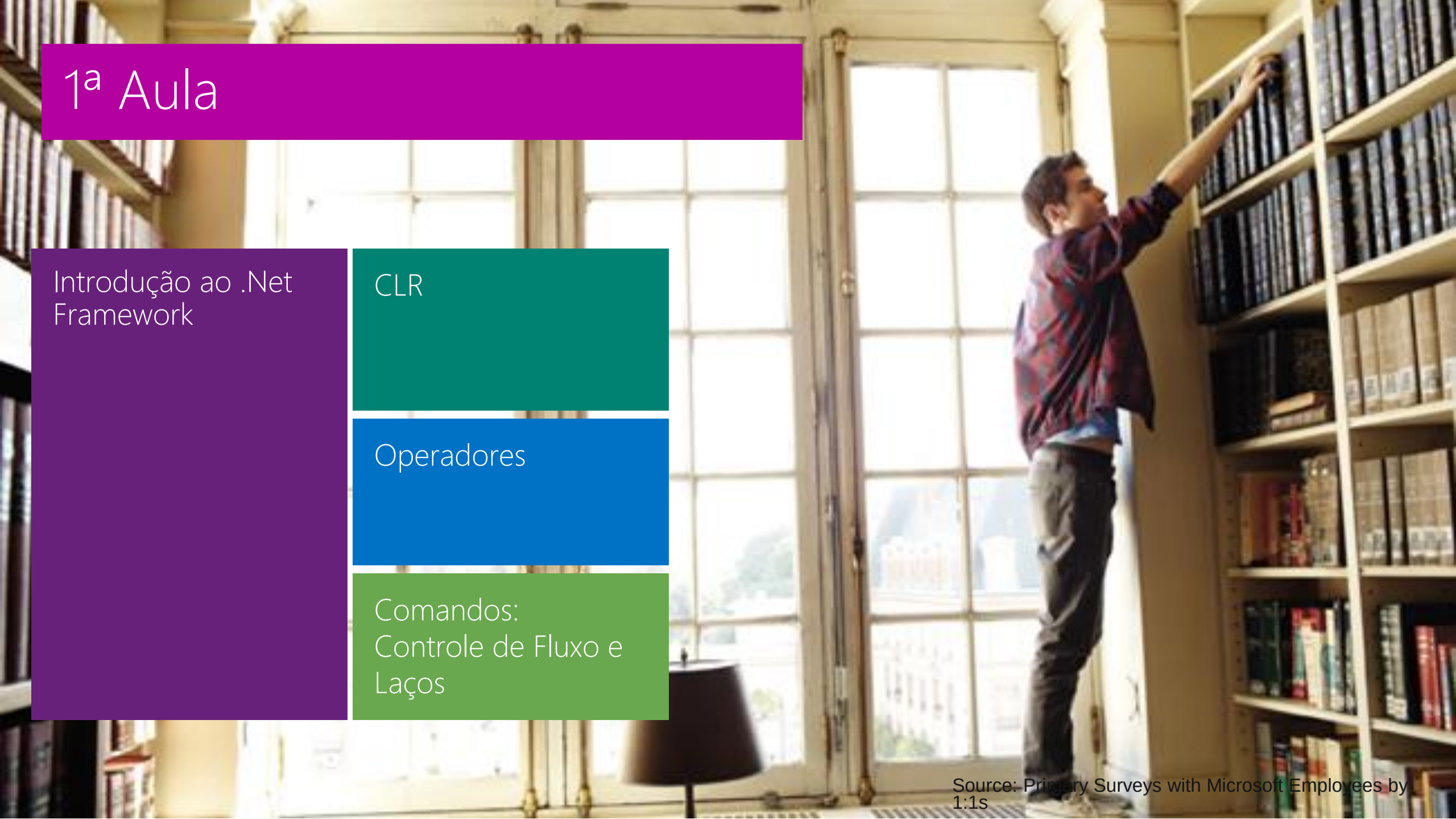
# 1ª Aula

Introdução ao .Net Framework

CLR

Operadores

Comandos: Controle de Fluxo e Laços



# Desenvolvendo para internet parte I

# Introdução

- HTML foi originalmente desenvolvido por Tim Berners-Lee no CERN e popularizado pelo navegador NCSA Mosaic na década de 1990
  HTML 2.0 especificado em 1994
  HTML 3.0 especificado em 1995
  HTML 3.2 especificado em 1997
  HTML 4.0 especificado em 1998
  HTML 4.01 especificado em 1999
  HTML 5 – especificação final em out/2014

# html

- Nasceu com a finalidade de estabelecer uma forma simples para publicar sites na internet.
  Significa de forma literal, linguagem de marcação de hipertexto.
  Hypertext Markup Language

# html

- Documentos são compostos de elementos

  **Um elemento consiste:**

  Marcação (tag) de abertura

  Conteúdo

  Marcação de fechamento

# html

- Uma marcação de abertura consiste:
  - Sinal <
  - Nome da marcação
  - Atributos opcionais
  - Sinal >
- Uma marcação de fechamento consiste:
  - Sinal </
  - Nome da marcação
  - Sinal >

<html>
<td rowspan="3">

</html>

# html

- Alguns elementos são vazios
  - Não possuem conteúdo
  - Um elemento vazio consiste:
    - Sinal <
    - Nome da marcação
    - Atributos opcionais
    - Sinal />
  - Um atributo consiste:
    - Nome do atributo
    - Sinal =
    - Valor do atributo entre aspas

<br />

# html

- Um documento HTML é composto de 3 partes:
  Uma linha contendo o tipo do documento
  Uma seção declarativa de cabeçalho
  Elemento HEAD
  Uma seção de corpo que define o conteúdo do documento
  Elementos BODY ou FRAMESET
  As seções de cabeçalho e corpo deve estar aninhadas dentro do elemento HTML

# html

- A estrutura básica de um documento HTML apresenta as seguintes marcações:

```
<!DOCTYPE html>
<html>
<head>
   Marcações que definem informações sobre o documento
   <title>Título</title>
</head>
<body>
   Marcações que definem o conteúdo do documento
</body>
</html>
```

# html

- A estrutura básica de um documento HTML apresenta as seguintes marcações:

```
<!DOCTYPE html>
<html>
<head>
   Marcações que definem informações sobre o documento
   <title>Título</title>
</head>
<body>
   Marcações que definem o conteúdo do documento
</body>
</html>
```

# html

- A estrutura básica de um documento HTML apresenta as seguintes marcações:

```
<!DOCTYPE html>
<html>
<head>
   Marcações que definem informações sobre o documento
   <title>Título</title>
</head>
<body>
   Marcações que definem o conteúdo do documento
</body>
</html>
```

# html

- Um comentário não é processado pelo navegador
  Um comentários consiste `<!-- Comentário -->`
  Símbolo <!--
  Conteúdo
  Pode ser de múltiplas linhas
  Não pode conter --
  Símbolo -->

# Elementos Básicos - Texto

- Quebra de linha forçada:
  Elemento vazio BR

- Parágrafo:
  Elemento P
  Representa um parágrafo de texto com uma linha em branco após seu fechamento
  Não pode conter elementos de marcação de blocos (como P) aninhados

# Elementos Básicos - Listas

- Elementos permitem a definição de
  - Listas ordenadas
  - Listas sem ordem
  - Listas de definição
  - Listas podem ser aninhadas

# Elementos Básicos - Listas

- Listas ordenadas:
  Elemento OL especifica a lista
  Elemento LI especifica um item da lista
  Navegadores usualmente numeram os itens da lista pela ordem de definição

```
<ol>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
</ol>
```

# Elementos Básicos - Listas

- Listas sem ordem:
  Elemento UL especifica a lista
  Elemento LI especifica um item da lista

```
<ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
</ul>
```

```
<ul>
        <li>Nível 1</li>
        <li>Nível 1
        <ul>
                <li>Nível 2</li>
                <li>Nível 2</li>
        </ul>
        </li>
        <li>Nível 1</li>
</ul>
```

# Elementos Básicos - Links

- Um hiperlink permite a vinculação de um recurso Web fonte com um recurso Web destino
  Um hiperlink possui
  Duas extremidades (fonte e destino), chamadas de âncoras
  Uma direção
  Comportamento padrão de um hiperlink é a recuperação do recurso Web destino
  Hiperlinks não podem ser aninhados

# Elementos Básicos - Links

- Links para recursos:
  Para definir uma âncora fonte
  Elemento A
  Conteúdo define a posição da âncora
  Atributo href especifica o endereço da âncora destino via uma URI
  URIs que designam uma âncora possuem o caractere # seguido do nome/identificador da âncora

```
<p>
Isso é um <a href="links2.html">link</a> para um outro documento.
</p>
```

# Elementos Básicos - Links

- Links para elementos do documento:
  Uma âncora de destino pode ser fragmentos do próprio documento onde está a âncora origem
  Para definir uma âncora de destino
  Elemento A com atributo name e/ou id
  Qualquer elemento com atributo id
  Elemento A define uma âncora fonte
  Atributo href especifica o endereço da âncora destino via uma referência para o identificador do fragmento

```
<a name="destino1">Outro parágrafo de texto.</a>
```

```
<p>
Isso é um <a href="links2.html#destino1">link</a> para
um pedaço de outro documento.
</p>
```

# Elementos Básicos - Tabelas

- Tabelas permitem organizar conteúdo em células por linhas e colunas
  Recomendação W3C:
  Não utilizar tabelas para realizar puramente o layout de documentos, para isso existem folhas de estilo

# Elementos Básicos - Tabelas

- Tabela:
  Elemento TABLE
  Contêm todos os demais elementos da tabela
  Atributo summary especifica um resumo do propósito da tabela (acessibilidade!)
  Atributo witdth especifica a largura da tabela
  Percentagem
  Pixel

# Elementos Básicos - Tabelas

- Título:
  Elemento CAPTION
  Especifica o título da tabela como seu conteúdo
  Deve aparecer como primeiro elemento aninhado ao elemento da tabela

```
<table summary="um exemplo de tabela simples com linhas e colunas">
        <caption>Tabela básica</caption>
        <tr><th>Ano</th><th>Vendas</th></tr>
        <tr><td>2008</td><td>1,1m</td></tr>
        <tr><td>2009</td><td>1,9m</td></tr>
</table>
```

# Elementos Básicos - Tabelas

- Linhas:

  Elemento TR

  Atua como um contêiner para uma linha de células de uma tabela

```
<table summary="um exemplo de tabela simples com linhas e colunas">
        <caption>Tabela básica</caption>
        <tr><th>Ano</th><th>Vendas</th></tr>
        <tr><td>2008</td><td>1,1m</td></tr>
        <tr><td>2009</td><td>1,9m</td></tr>
</table>
```

# Elementos Básicos - Tabelas

- Células:

  Podem conter dois tipos de informação: cabeçalho e dados

  Podem ser vazias

  Elemento TH

  Define uma célula que possui informação de cabeçalho

  Elemento TD

  Define uma célula que possui informação de dados

  O conjunto de células da linha define o número de colunas da tabela

```
<table summary="um exemplo de tabela simples com linhas e colunas">
        <caption>Tabela básica</caption>
        <tr><th>Ano</th><th>Vendas</th></tr>
        <tr><td>2008</td><td>1,1m</td></tr>
        <tr><td>2009</td><td>1,9m</td></tr>
</table>
```

# Elementos Básicos - Tabelas

- Células expandidas:
  Células podem se expandir por múltiplas linhas ou colunas
  Atributo rowspan especifica o número de linhas ocupada por uma célula
  Atributo colspan especifica o número de colunas ocupada por uma célula
  Cuidado para não definir células que se sobreponham!

```
<table summary="um exemplo de tabela simples com linhas e colunas expandidas">
        <caption>Tabela expandida</caption>
        <tr><th colspan="2">Um cabeçalho expandido</th></tr>
        <tr><td>2008</td><td>1,1m</td></tr>
        <tr><td>2009</td><td>1,9m</td></tr>
</table>
```

# Elementos Básicos - Imagens

- Mecanismo para inclusão de imagens em documentos
  PNG, JPEG, GIF, etc
  Elemento IMG
  Atributo src especifica o endereço URI da imagem
  Atributo alt especifica uma descrição textual alternativa para a imagem (acessibilidade!)

```
<img alt="Uma imagem" src="image1.png" height="500" width="870">
```

# Formulários

- Formulários representam fragmentos de documentos que contêm elementos de interação com o usuário chamados de controles
- Representam pontos de entrada de dados a serem enviados para processamento em um servidor

# Formulários

- Formulário:
  Elemento FORM
  Atua como um contêiner para os controles
  Especifica
  A entidade que irá receber os dados do formulário através do atributo action
  O método (get ou post) pelo qual os dados serão enviados ao servidor através do atributo method
  O formato de codificação dos dados enviados ao servidor através do atributo enctype
  application/x-www-form-urlencoded é o valor padrão

```
<form method="get">
...
</form>
```

# Formulários

- Controles:
  HTML define vários controles: botões de ação, botões de seleção, botões de rádio, caixas de seleção, caixas de texto, seleção de arquivos, controles escondidos, objetos
  Controles possuem um valor inicial (que nunca muda) e um valor atual (que muda de acordo com a interação do usuário e scripts)

# Formulários

- Caixas de texto simples:
  Elemento INPUT com atributo type text
  Elemento INPUT com atributo type password
  Texto é renderizado com os caracteres obfuscados
  Atributo size especifica o número de caracteres do tamanho do controle
  Atributo value especifica o valor inicial do controle
  Atributo maxlenght especifica o número máximo de caracteres que pode ser fornecido para o controle

```
<input name="texto" id="texto" type="text">
```

# Formulários

- Caixas de texto múltiplo:
  Elemento TEXTAREA
  Conteúdo do elemento define o valor inicial
  Atributo cols especifica a quantidade de caracteres na horizontal
  Atributo rows especifica o número de linhas

```
<textarea rows="5"
cols="30">
Valor inicial
</textarea>
```

# Formulários

- Caixas de seleção:
  Elemento SELECT
  Fornecem um meio se selecionar valores dentro de um conjunto de opções
  Atributo size especifica o número de linhas de opções que é mostrado pelo navegador
  Navegador usualmente escolhe o tipo de elemento visual que será mostrado em função deste número
  Ex.: lista de seleção ou menu drop-down
  Atributo multiple especifica se é permitida a seleção de múltiplos valores

```
<select name="selecao" id="selecao" size="3"
multiple="multiple">
<option>1</option>
<option>2</option>
<option>3</option>
</select>
```

# Formulários

- Caixas de seleção:
  Elemento OPTION especifica as opções que podem ser selecionadas
  Conteúdo do elemento especifica o texto que é apresentado como opção de seleção
  Atributo label especifica um valor a ser utiliza como texto de apresentação ao invés do conteúdo do elemento
  Atributo value especifica o valor inicial do elemento, se não utiliza o valor do conteúdo
  Atributo selected especifica que a opção está pré-selecionada
  Deve existir pelo menos uma opção pré-selecionada para evitar erros

```
<select name="selecao" id="selecao" size="3"
multiple="multiple">
<option>1</option>
<option>2</option>
<option>3</option>
</select>
```

# Formulários

- Botões de seleção:
  Elemento INPUT com atributo type checkbox
  Representa controles de seleção binária (ligado ou desligado)
  Atributo value especifica o valor inicial do controle (obrigatório)
  Atributo checked especifica se o controle está ligado ou desligado
  Botões de seleção são agrupados pelo valor do atributo id
  Permite que múltiplos botões estejam ligados

```
<input name="cidade" type="checkbox" value="1">Porto Alegre
<input name="cidade" type="checkbox" value="2">Florianópolis
<input name="cidade" type="checkbox" value="3">Curitiba
```

# Formulários

- Botões de rádio:
  Elemento INPUT com atributo type radio
  Representa controles de seleção binária (ligado ou desligado)
  Atributo value especifica o valor inicial do controle (obrigatório)
  Atributo checked especifica se o controle está ligado ou desligado
  Deve existir um dos botões ligado para evitar erros
  Botões de seleção são agrupados pelo valor do atributo id
  Somente um botão do grupo pode estar ligado, ou seja, são mutuamente exclusivos

```
<input type="radio" name="sexo" value="m"
checked="checked">Masculino
<input type="radio" name="sexo" value="f">Feminino
```

# Formulários

- Dados escondidos:
  Elemento INPUT com atributo type hidden
  Não representa um controle que é visual
  Utilizado para armazenar dados que são submetidos junto ao formulário como uma forma de implementação de mecanismo de seção
  Atributo value especifica o valor inicial do controle

```
<input type="hidden" value="Este texto é escondido!">
```

# Formulários

- Botões de ação:
  Três tipos de botões
  Botão de submissão (submit) – enviar dados do formulários para o servidor
  Botão de reset (reset) – restaurar os valores inicias dos controles do formulário
  Botão de pressão (push) – sem ação padrão, com scripts associados a seus eventos
  Dois elementos diferentes
  Elemento INPUT
  Elemento BUTTON
  Provê possibilidades mais ricas de renderização

```
<input type="submit" value="OK">
<button type="reset">Limpar</button>
<button type="button">Clique Aqui!</button>
```

# Formulários

- Botões de submissão:
  Elemento INPUT com atributo type submit
  Atributo value especifica o rótulo do botão
  Elemento BUTTON com atributo type submit
  Permite que o rótulo do botão seja definido pelo conteúdo do elemento
  Por exemplo, pode-se utilizar uma imagem como conteúdo

# Formulários

- Botões de reset:
  Elemento INPUT com atributo type reset
  Atributo value especifica o rótulo do botão
  Elemento BUTTON com atributo type reset
  Permite que o rótulo do botão seja definido pelo conteúdo do elemento
  Por exemplo, pode-se utilizar uma imagem como conteúdo

# Formulários

- Botões de pressão:
  Elemento INPUT com atributo type button
  Atributo value especifica o rótulo do botão
  Elemento BUTTON com atributo type button
  Permite que o rótulo do botão seja definido pelo conteúdo do elemento
  Por exemplo, pode-se utilizar uma imagem como conteúdo

# Formulários

- Maiores informações sobre HTML e suas tags:
  http://www.w3.org/
  http://www.w3schools.com/html/
  http://www.w3schools.com/tags/

# javascript

- JavaScript é
  - Uma linguagem de script interpretada
  - Orientada a objetos (baseada em protótipos)
  - Dinâmica
  - Fracamente tipada
  - Navegadores suportam scripts que rodam código no lado-cliente
  - JavaScript é o nome “comum” de versões da linguagem, que foi padronizada como ECMAScript
  - Baseadas na versão padronizada, mas com funcionalidades adicionais

# javascript

- JavaScript possui múltiplas versões, suportadas ou não pelos diversos navegadores
  Versão padrão:
  ECMAScript 262 5th Edition

# javascript

JavaScript no documento:

- Código inline

```
<script type="text-javascript">
Código
</script>
```

- Código em arquivo externo

```
<script type="text-javascript" src="arquivo.js">
</script>
```

# javascript

- Elemento SCRIPT:
  Pode aparecer múltiplas vezes dentro dos elementos HEAD e BODY
  No HEAD usualmente colocam-se funções
  No BODY usualmente colocam-se código e chamada a funções que geram conteúdo dinamicamente
  O script pode ser definido dentro do conteúdo do elemento ou através de referência via atributo src
  A linguagem de script definida via atributo type
- Elemento NOSCRIPT:
  Deve ser avaliado no caso de scripts não suportados ou desabilitado no navegador
  Conteúdo do elemento é utilizado ao invés do elemento SCRIPT

# javascript

- Exemplo:

```
<!DOCTYPE html >
<html>
<head>
   <title>Título</title>
</head>
<body>
<script type="text/javascript">
   document.write("<p>Alô Mundo!</p>");
</script>
<noscript>
   <p>Por favor, habilite o JavaScript em seu navegador.</p>
</noscript>
</body>
</html>
```

# javascript

- Para escrever código que se comunica com os elementos dos navegadores, JavaScript faz uso de diversas APIs
  Algumas padronizadas pelo W3C
  DOM – Document Object Model
  Permite manipular elementos, conteúdos e estilos de documentos
  XMLHttpRequest
  Permite adicionar conteúdo adicional sem a necessidade de carregar um novo documento
  Elemento básico para o AJAX

# javascript

- Exemplo:

```
<html>
<head>
     <script language="javascript">

         function Carregar()
         {
             document.getElementById("texto").innerHTML =
"Pronto...";
         }

    </script>
</head>
<body>
    <a href="#" onclick="Carregar()">Próxima página</a>
    <div id="texto"></div>
</body>
</html>
```

# Links Úteis

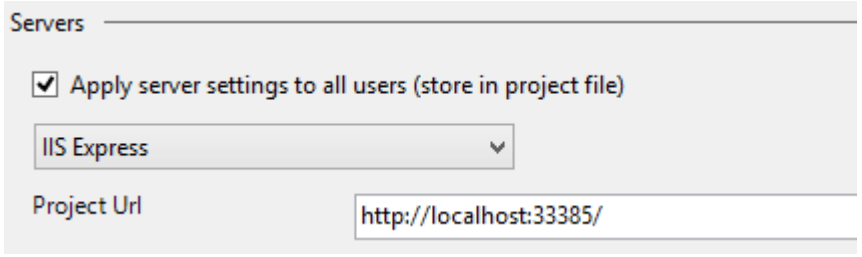
- Maiores informações sobre JavaScript:
  http://www.ecma-international.org/publications/standards/Ecma-262.htm
  http://www.w3schools.com/js/

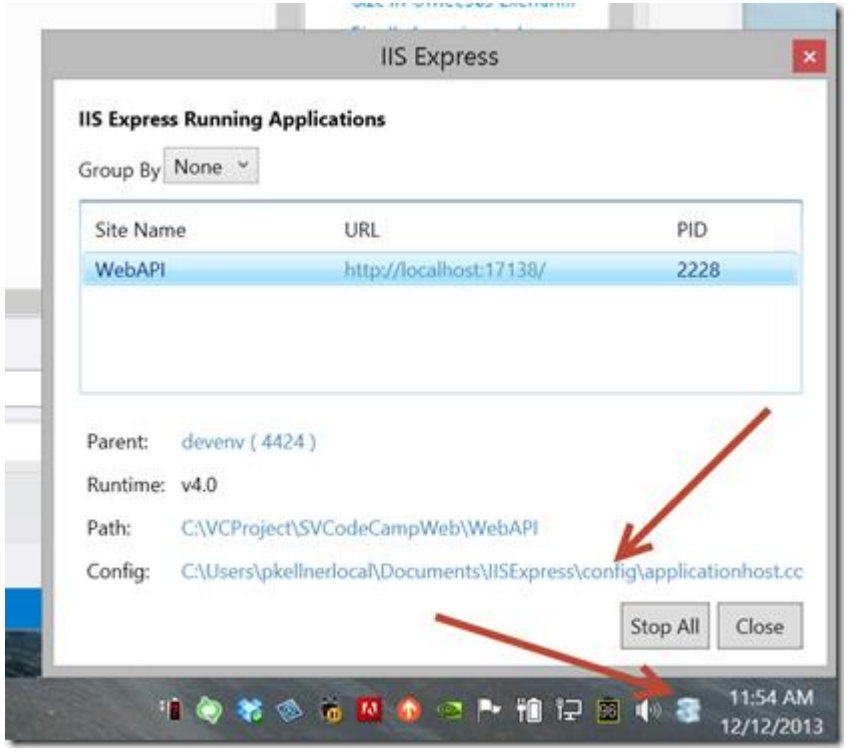
# Internet Information Services (IIS)

- Conjunto integrado de serviços para um servidor Web
  Permite publicar conteúdo e disponibilizar arquivos e aplicações em um ambiente Internet/Intranet
  Dotado de uma interface administrativa gráfica
  Baseado no conceito de Diretório Virtual
  Meio indicado de instalação:
  Web Platform Installer http://www.microsoft.com/web/

# Servidor Web de Desenvolvimento

- IIS Express
  Utilizado durante o desenvolvimento da aplicação
  Não necessita de configurações adicionais



# Web.config

- Arquivo no formato XML
  Informações de configuração da sua aplicação, tais como string de conexão a fontes de dados, páginas de erro, modo de compilação, etc.
  Armazenar valores e parâmetros que sejam comuns a toda aplicação.

```
<configuration>
    <appSettings>
        <add key="appConexao" value="Provider=SQLOLEDB.1;Persist Security Info=False;
        User ID=sa;Initial Catalog=Artigos;Data Source=dbICARO" />
    </appSettings>
</configuration>
```

```
//Usando a variável na aplicação
string strConexao =
    System.Configuration.ConfigurationManager.AppSettings("appConexao");
```

# Web.config

- Arquivo no formato XML
  Informações de configuração da sua aplicação, tais como string de conexão a fontes de dados, páginas de erro, modo de compilação, etc.
  Armazenar valores e parâmetros que sejam comuns a toda aplicação.

```
<configuration>
    <appSettings>
        <add key="appConexao" value="Provider=SQLOLEDB.1;Persist Security Info=False;
        User ID=sa;Initial Catalog=Artigos;Data Source=dbICARO" />
    </appSettings>
</configuration>
```

```
//Usando a variável na aplicação
string strConexao =
    System.Configuration.ConfigurationManager.AppSettings("appConexao");
```

# Desenvolvimento para Internet parte II

# Ciclo de Vida

- Uma página Web Forms passa por um ciclo de vida completo no servidor Web depois do pedido inicial do cliente (roundtrips)
  Ciclo é disparado no modelo request/response do protocolo HTTP

# Ciclo de Vida

- O ciclo de vida inclui diversos passos de processamento
  - Relacionados à página
  - Relacionados à aplicação Web
  - Estrutura de eventos bastante longa e complexa

# Ciclo de Vida

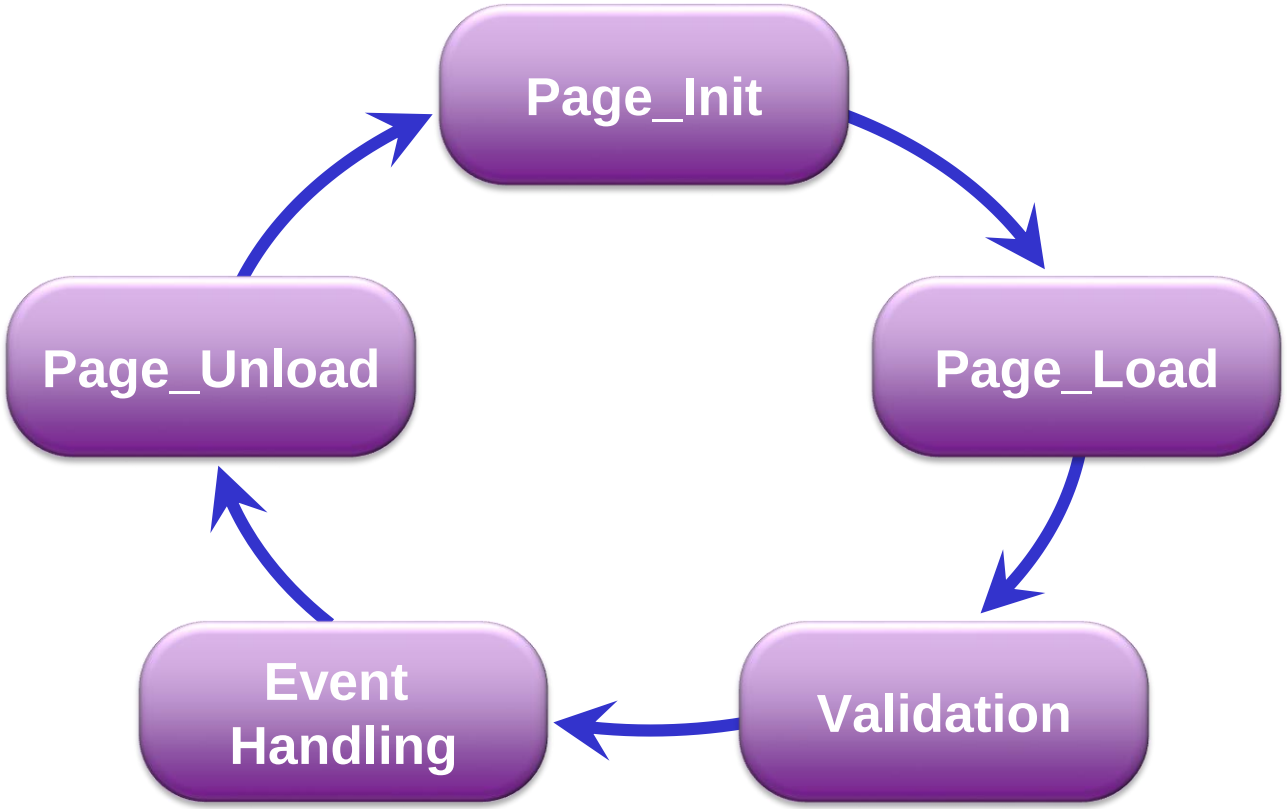
- Gerenciamento de memória
- Tratamento de exceções
- Compilação
- Segurança
- Outros

# Ciclo de Vida

- O ciclo de vida inclui diversos passos de processamento
  - Relacionados à página
  - Relacionados à aplicação Web
  - Estrutura de eventos bastante longa e complexa

# Ciclo de Vida

- O ciclo de vida de uma página ASP.NET apresenta cinco estágios básicos:



# Ciclo de Vida

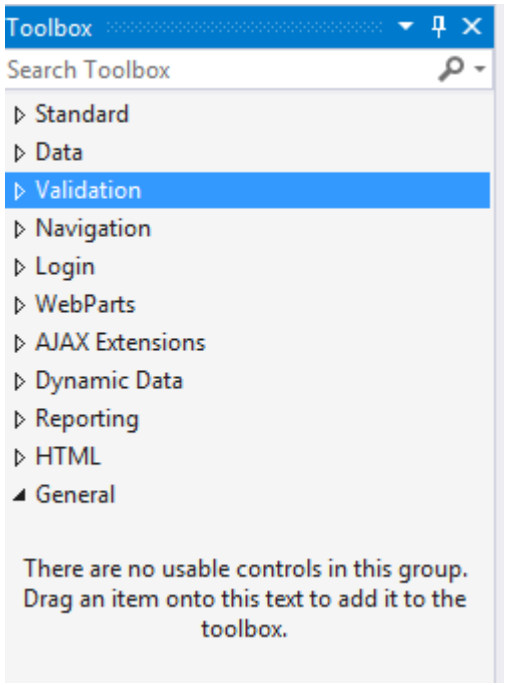
- Fases gerais no ciclo de vida de uma página:
  Requisição da página (request)
  Início (start) – propriedades básicas da página são criadas
  Inicialização (initialization) – criação dos controles da página
  Carregamento (load) – dados dos controles são atualizados no caso de um postback
  Validação (validation) – método de validação é executado sobre os controles de validação
  Tratamento de eventos de postback (event handling) – execução de métodos de eventos associados no caso de um postback
  Renderização (rendering) – HTML de resposta é gerado
  Descarregamento (unload) – realizada a limpeza dos objetos utilizados

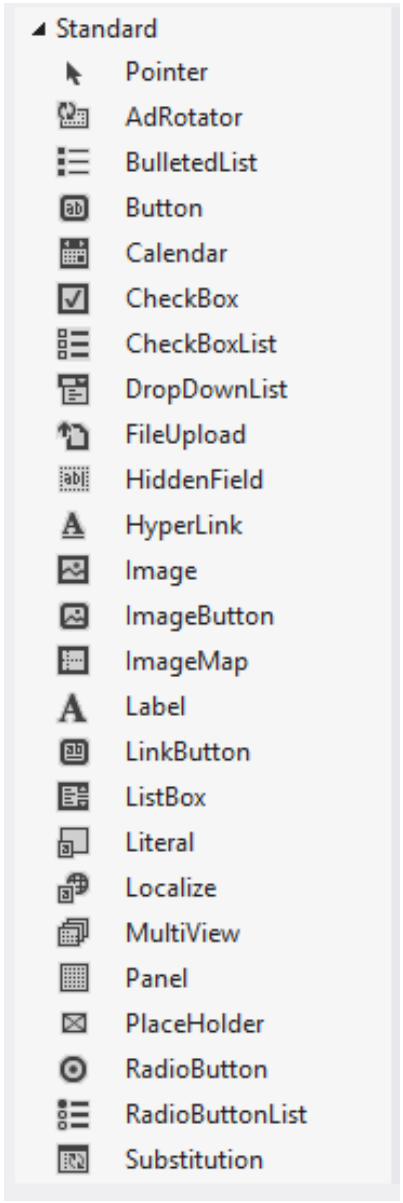
# Ciclo de Vida

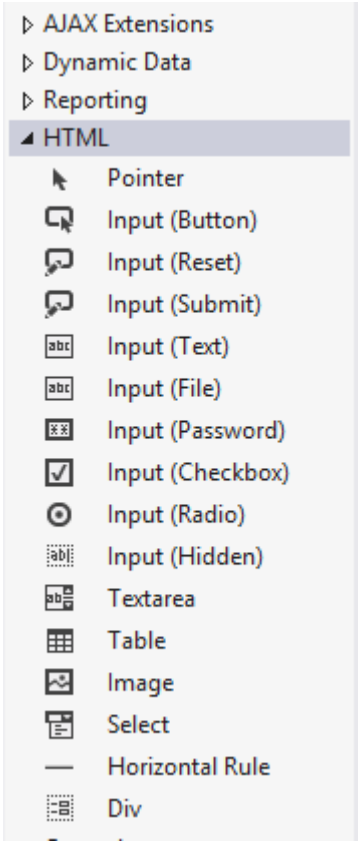
- ASP.Net fornece componentes para a construção de interfaces com o usuário em Web Forms

| **Característica** | **Server Controls** | **HTML Controls** |
|---|---|---|
| Eventos no servidor | Possibilidade de eventos específicos no servidor | Apenas postback |
| Gerência de Estado | Mantido através dos roundtrips | Não mantém estado |
| Adaptação | Detecta o browser e adapta-se | Sem adaptação |
| Propriedades | Características do .NET Framework | Apenas atributos HTML |

# Controles


Toolbox
Search Toolbox
Standard
Data
Validation
Navigation
Login
WebParts
AJAX Extensions
Dynamic Data
Reporting
HTML
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.


Standard
Pointer
AdRotator
BulletedList
Button
Calendar
CheckBox
CheckBoxList
DropDownList
FileUpload
HiddenField
HyperLink
Image
ImageButton
ImageMap
Label
LinkButton
ListBox
Literal
Localize
MultiView
Panel
PlaceHolder
RadioButton
RadioButtonList
Substitution

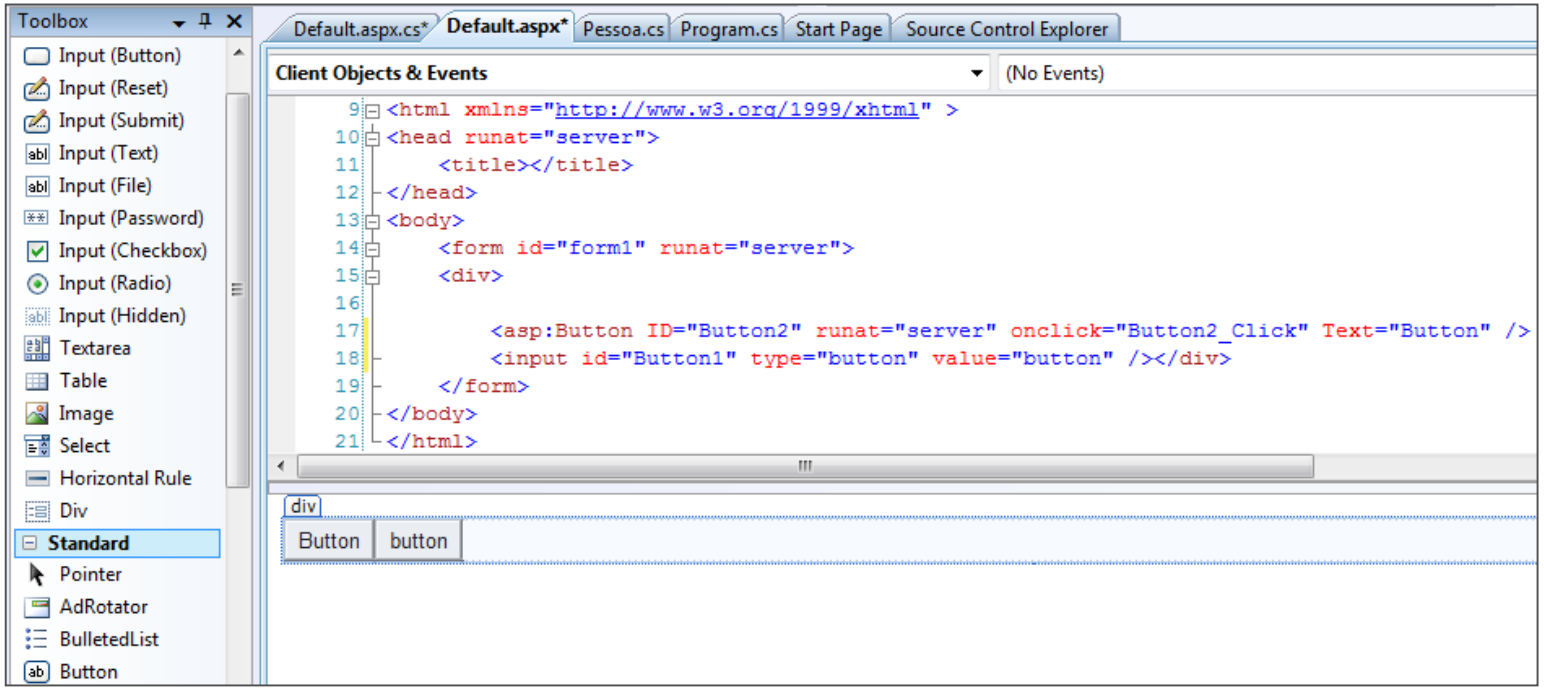

AJAX Extensions
Dynamic Data
Reporting
HTML
Pointer
Input (Button)
Input (Reset)
Input (Submit)
Input (Text)
Input (File)
Input (Password)
Input (Checkbox)
Input (Radio)
Input (Hidden)
Textarea
Table
Image
Select
Horizontal Rule
Div

# Controles

- Para adicionar um controle Server Control
  Arraste o controle desejado da aba Standard da Toolbox
  Para adicionar um controle HTML Control
  Arraste o controle desejado da aba HTML da Toolbox

# Controles

- Para adicionar um controle Server Control
  Arraste o controle desejado da aba Standard da Toolbox
  Para adicionar um controle HTML Control
  Arraste o controle desejado da aba HTML da Toolbox



# Controles

- Alguns controles básicos:
  Button: Botão clicável
  TextBox: Caixa para digitação de texto
  CheckBox: Caixa para selecionar ou não um item
  Label: Texto que não pode ser editado diretamente
  ListBox: Lista para escolha de uma ou mais opções
  RadioButton: Caixa para selecionar ou não um item.

# Controles - Básicos

- Label
  Representa um componente de texto que pode ser alterado programaticamente
  Para texto estático, utilizar HTML diretamente
  Para alterar o texto apresentado:
  Propriedade Text
  Button
  Representa um controle de botão que ao ser clicado executa uma submissão (um postback) de um formulário para o servidor
  Outros estilos de “botões” incluem os componentes LinkButton e ImageButton

# Controles - ListBox

- Permite a seleção de um ou vários elementos de uma lista
  - Dados armazenados na coleção Items
  - Qualquer tipo de objetos
  - Usualmente strings
  - Opções para configurar os dados:
  - Propriedade DataSource com a fonte de dados
  - Adição direta na coleção de itens via método Add()
  - Remoção direta da coleção de itens via método Remove() e RemoveAt()

# Controles - ListBox

- Para configurar os dados visíveis/retornados em objetos com DataSource:
  Propriedade DataValueField  especifica o nome do valor do elemento da fonte de dados
  Propriedade DataTextField especifica o nome do dado “visual” do elemento da fonte de dados

# Controles - DropDownList

- Semelhante ao ListBox porém os elementos ficam “escondidos” até a seleção e somente um deles pode ser selecionado

# Controles - CheckBox

- Permite indicar um elemento com a informação de aceitação/rejeição
  Para obter a seleção do usuário:
  Propriedade Checked retorna true ou false dependendo se o item está marcado ou não
  Controle CheckBoxList gerencia uma coleção de itens mostrados em diversas caixas de seleção

# Controles - RadioButton

- Permite a seleção de um único elemento dentre várias opções
  O grupo de botões deve estar configurado com o mesmo nome na propriedade GroupName para que a seleção seja exclusiva
  Para obter a seleção do usuário:
  Propriedade Checked retorna true ou false dependendo se o item está marcado ou não
  Controle RadioButtonList gerencia uma coleção de itens mostrados em diversos botões de seleção

# Controles - Outros

- Gridview – tabela para exibição de dados de fácil preenchimento e integração com banco de dados, com controle de paginação e ordenação automáticos e suporte a templates

| CódigoDoCliente | NomeDoContato | CargoDoContato | Cidade | País | Telefone |
|---|---|---|---|---|---|
| ALFKI | Maria Anders | Representante de Vendas | Berlin | Alemanha | 030-0074321 |
| ANATR | Ana Trujillo | Proprietário | México D.F. | México | (5) 555-4729 |
| ANTON | Antonio Moreno | Proprietário | México D.F. | México | (5) 555-3932 |
| AROUT | Thomas Hardy | Representante de Vendas | London | Reino Unido | (171) 555-7788 |
| BERGS | Christina Berglund | Administrador de Pedidos | Luleå | Suécia | 0921-12 34 65 |
| 1 2 3 4 5 6 7 8 9 10 ... | | | | | |

# Controles - Outros

- DataList: Mecanismo parecido com o GridView, porém, com menos recursos
  Repeater: Mecanismo parecido com o GridView, porém mais flexível e leve

# Controles - Outros

- SiteMap – permite criar um menu de navegação baseado na página que o usuário está acessando.

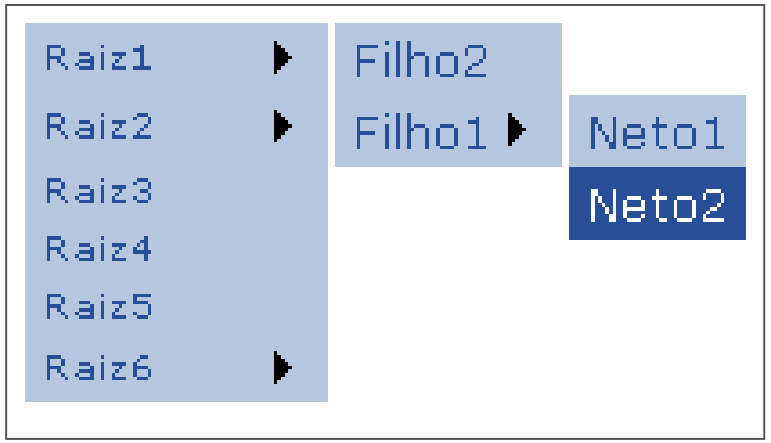
Página Principal : Área Administrativa : Cadastros : Clientes

Página Principal : Área Administrativa : Cadastros : Fornecedores

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="principal.aspx" title="Página Principal"  description="">
    <siteMapNode url="adm.aspx" title="Área Administrativa"  description="">
      <siteMapNode url="cadastro.aspx" title="Cadastros"  description="">
        <siteMapNode url="fornecedores.aspx" title="Fornecedores"  description="" />
        <siteMapNode url="clientes.aspx" title="Clientes"  description="" />
      </siteMapNode>
    </siteMapNode>
  </siteMapNode>
</siteMap>
```

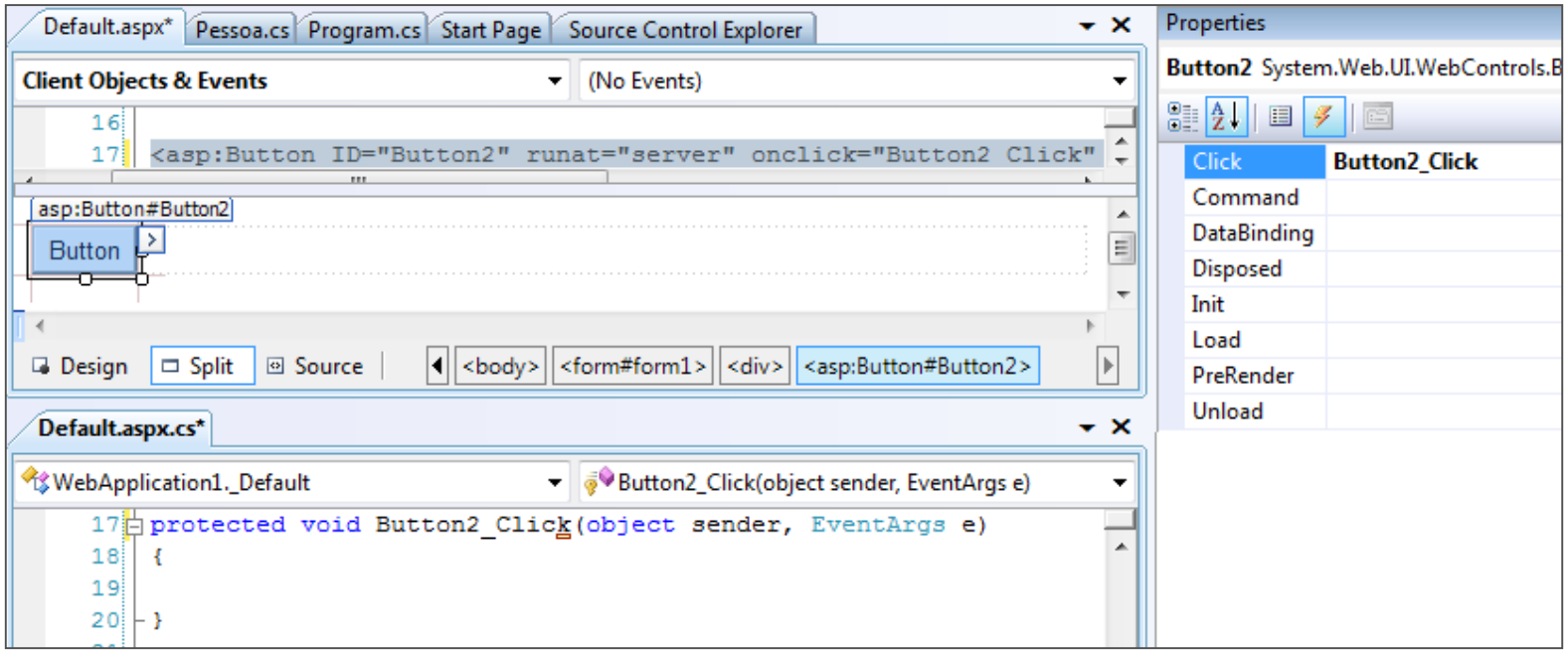
# Controles - Outros

- Menu – permite criar um menu com links de maneira estática ou dinâmica. Pode utilizar o mesmo arquivo XML do SiteMap.



# Controles - Outros

- Muitos eventos são disparados através de ações de usuários captadas pelo navegador
  O código para manipular o evento disparado é executado no servidor
  Quando o código completa sua execução, a página web pronta é enviada de volta ao navegador (contendo código html e script)



# Partial Types

- Permite dividir a implementação de um determinado tipo em diversos arquivos.
  Disponível para classes, estruturas e interfaces.
  Definidos pela palavra-chave partial.

# Partial Types

- Quando podem ser utilizados:
  Quando trabalhamos com código gerado automaticamente, código pode ser adicionado à classe sem ter que recriar o arquivo fonte.
  Partial Types permitem que dois ou mais desenvolvedores trabalhem no mesmo tipo, enquanto ambos têm seus arquivos checados para edição, sem interferir um no outro.

# Partial Types

- Quando podem ser utilizados:
  Quando trabalhamos com código gerado automaticamente, código pode ser adicionado à classe sem ter que recriar o arquivo fonte.
  Partial Types permitem que dois ou mais desenvolvedores trabalhem no mesmo tipo, enquanto ambos têm seus arquivos checados para edição, sem interferir um no outro.

# Laboratório 12

# Customização de Layout

- ASP.NET fornece o conceito de master pages e content pages para a definição de layouts de páginas em uma aplicação web

  Permite

  a criação de sites cujo layout é consistente entre as diversas páginas

  a reutilização de conteúdo e funcionalidades

# Customização de Layout

- Uma master page define a aparência e comportamento que são compartilhados por um grupo de páginas

  Um conjunto de content pages possuem o conteúdo das páginas que referenciam a master page para produzir o resultado final da combinação dos elementos

# Master Page

- Vantagens:
  Criar uma Herança Visual para o Web Site
  Manutenção centralizada, não é necessário mudar o código em várias páginas, apenas em uma
  Facilidade na criação do layout
  Reaproveitamento de código

# Master Page

- São arquivos ASP.NET com a extensão ".master"
  Contêm HTML, controles, código, etc
  Não representam uma página completa, mas elementos que são incorporados em outros web forms em tempo de execução
  Possuem a diretiva @Master ao invés da diretiva @Page

# Master Page

- Diretiva @Master

```
<%@ Master Language="C#"%>
```

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

Controle ContentPlaceHolder

Provê a localização onde os conteúdos das content pages serão incluídos

A master page pode conter diversos desses controles

Demais componentes não são incluídos dentro do ContentPlaceHolder

```
<asp:ContentPlaceHolder ID="MainContent" runat="server"/>
```

# Content Page

- São páginas web que referenciam uma master page
  Possuem conteúdos próprios que serão mesclados com a master page

# Content Page

- São arquivos ASP.NET com a extensão ".master"
  Contêm HTML, controles, código, etc
  Não representam uma página completa, mas elementos que são incorporados em outros web forms em tempo de execução
  Possuem a diretiva @Master ao invés da diretiva @Page

# Master Page

- Diretiva @Page
  - Inclui o atributo MasterPageFile para referenciar a master page

```
<%@ Page Language="C#" MasterPageFile="~/PaginaMestre.master"%>
```

  - Controle Content
  - Contém o conteúdo específico da página a ser mesclado com a master page
  - São mapeados para os componentes ContentPlaceHolder da master page
  - Atributo ContentPlaceHolderID deve indicar o ID do ContentPlaceHolder

```
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
   ...
</asp:Content>
```

# Master Page

- Diretiva @Page
  - Inclui o atributo MasterPageFile para referenciar a master page

```
<%@ Page Language="C#" MasterPageFile="~/PaginaMestre.master"%>
```

  - Controle Content
  - Contém o conteúdo específico da página a ser mesclado com a master page
  - São mapeados para os componentes ContentPlaceHolder da master page
  - Atributo ContentPlaceHolderID deve indicar o ID do ContentPlaceHolder

```
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
   ...
</asp:Content>
```

# Bootstrap

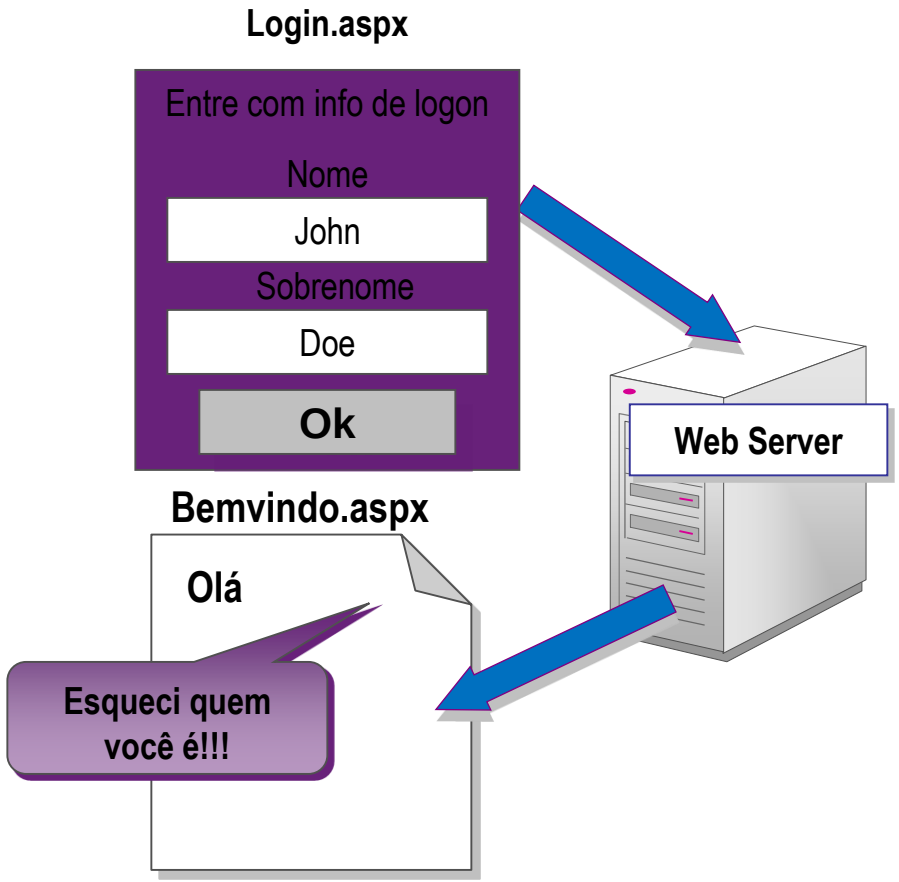
- Templates do Visual Studio 2013 passaram a utilizar o “Bootstrap”
  Framework de temas e layout criado pelo Twitter.
  Utiliza CSS3 para criação de páginas responsivas, com isto a página se adapta dinamicamente para diferentes navegadores e tamanhos de tela
  **Recursos**
  http://Bootswatch.com
  http://getbootstrap.com
  http://www.w3schools.com/bootstrap/

# Laboratório 13

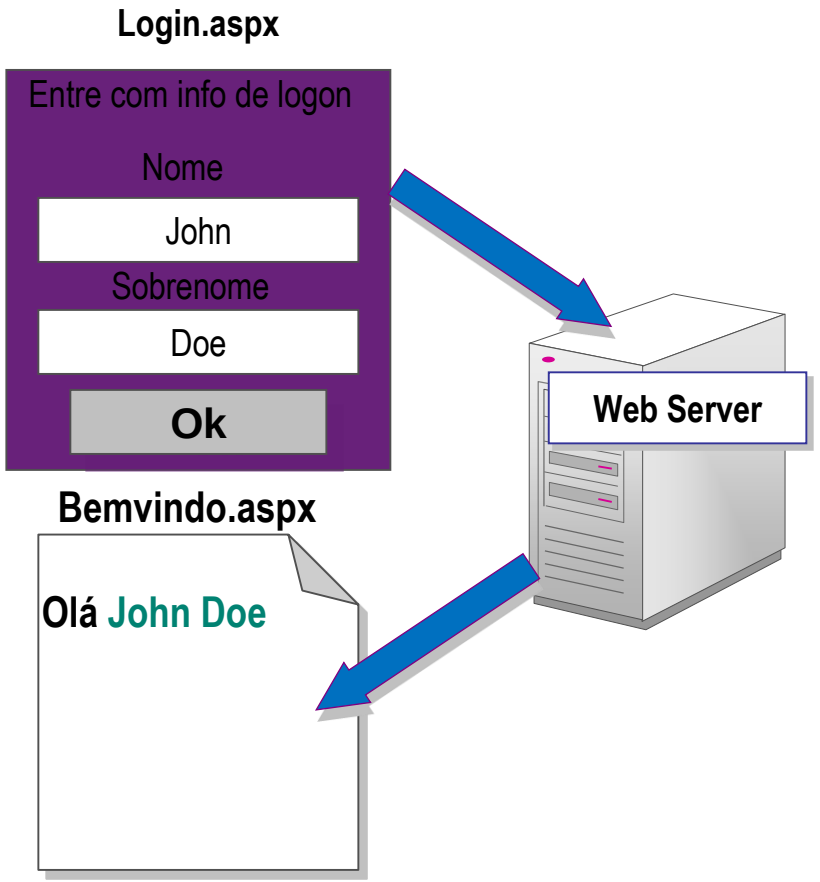
# Desenvolvendo para internet parte III

# O que é gerenciamento de estado?

## Sem gerenciamento de estado




## Com gerenciamento de estado



# Tipos de gerenciamento de estado

| Gerenciamento de estado do lado servidor | Gerenciamento de estado do lado cliente |
| --- | --- |
| Application<br>• Informação disponível para todos os usuários da aplicação web | Cookies<br>• Arquivo texto armazena informação para manter estado |
| Session<br>• Informação disponível apenas para o usuário da sessão específica | ViewState<br>• Mantém valores (principalmente dos controles) entre as requisições das páginas |
| Database<br>• Usa o suporte de um banco de dados para manter estado do Web site | Query strings<br>• Informação acrescentada no fim da URL (comando GET do HTTP) |

# Session

- Uma das formas mais simples de manutenção de estado é através de variáveis de sessão
- Por padrão, estas informações estão armazenadas no próprio processo do ASP.NET
- É possível armazenar informações de sessão em um processo separado (um servidor de estado) ou até mesmo em um Sistema Gerenciador de Banco de Dados

# Session

- Uma variável de sessão está associada exclusivamente a uma única sessão.
- Isto significa que um dado armazenado em uma variável de sessão com nome X para o usuário João não será visível na variável de sessão de mesmo nome do usuário Pedro, e vice-versa.

```
Session[“X"] = “S2B”;
```

```
string nome = (string)Session[“X"];
```

# Session - Eventos associados

- Presentes no Global.asax

```
protected void Session_Start(Object sender, EventArgs e)
{
        //Evento disparado quando a uma sessão é iniciada.
}

protected void Session_End(Object sender, EventArgs e)
{
        //Evento disparado quando a sessão é finalizada.
}
```

# Application

- Variável de estado da aplicação
  Visível em toda aplicação para TODOS usuários
  Exemplos de uso:
  Chat
  Contador de Acessos
  Exemplo:

```
Application[“ContadorAcessos”] = 0;
```

# Application - Eventos associados

- Presentes no Global.asax

```
protected void Application_Start(Object sender, EventArgs e)
{
        //Evento disparado quando a aplicação é iniciada.
}

protected void Application_End(Object sender, EventArgs e)
{
        //Evento disparado quando uma aplicação é finalizada.
}
```

# ViewState

- Mantêm automaticamente os valores de controles de servidor entre um postback e outro
- Internamente funciona como um campo oculto (hidden) um pouco mais sofisticado

# ViewState

- Uma página ASP.NET possui um campo oculto para o armazenamento do ViewState:

```
<input        type="hidden"
      name="__VIEWSTATE"
      id="__VIEWSTATE"
value="/wEPDwUJNzgzNDMwNTMzZGS8mO25pQR00V4slvgSxG3dEvK+hA==“  />
```

- Note que os dados não são exibidos em texto plano, por questões de segurança

# ViewState

- Pode-se ainda adicionar manualmente valores a um ViewState, lembrando que você vai conseguir recuperá-los apenas na mesma página

```
ViewState.Add("Nome“, “Bill“);
```

```
String nome = (string) ViewState[“Nome”];
```

# Cookie

- Trata-se de um pequeno arquivo de texto que é armazenado na máquina do usuário
  Usado, por exemplo, em sites de comércio eletrônico, para exibir as preferências e características do usuário
  Pode identificar o usuário mesmo dias depois de seu acesso a página
  O grande problema dos cookies é que o usuário simplesmente pode desabilitar este recurso em seu navegador

# Cookie

## Escrevendo um Cookie

```
//Cria um novo cookie, passando o nome no construtor
HttpCookie cookie = new HttpCookie(“Nome”);

//Determina o valor o cookie
cookie.Value = “Márcio”;
//Configura o cookie para expirar em 1 minuto
DateTime dtNow = DateTime.Now;
TimeSpan tsMinute = new TimeSpan(0, 0, 1, 0);
cookie.Expires = dtNow + tsMinute;
//Adiciona o cookie
Response.Cookies.Add(cookie);
```

# Cookie

## Lendo um Cookie

```
//Pega o nome do que cookie que o usuário informou
String strCookieName = NameField.Text;

//Captura o cookie
HttpCookie cookie = Request.Cookies[strCookieName];

String strCookieValue = “Vazio”;

//Certifica-se que o cookie existe
if (cookie != null)
   strCookieValue = cookie.Value.ToString();
```

# Cookie

## Lendo um Cookie

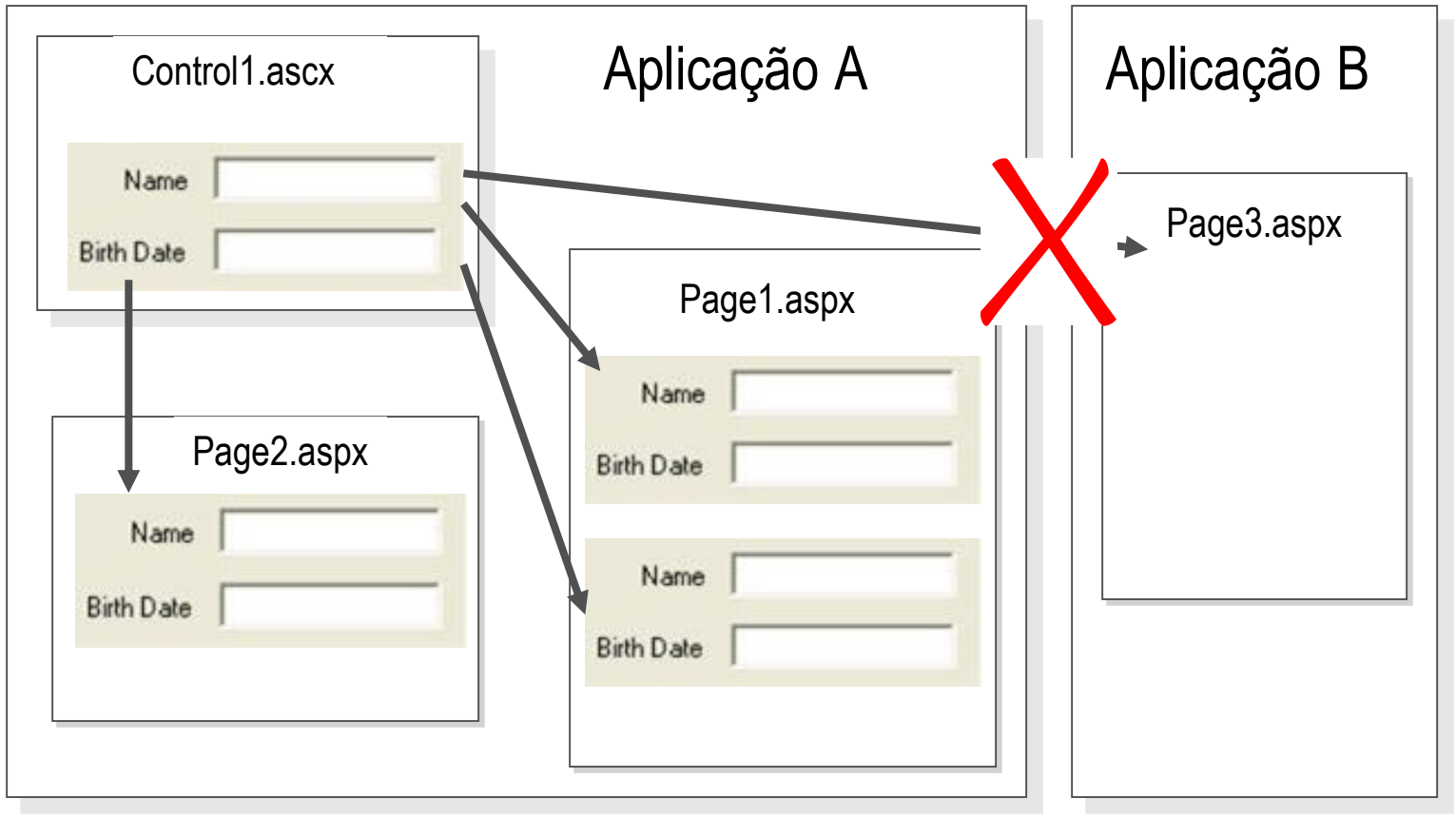
```
//Pega o nome do que cookie que o usuário informou
String strCookieName = NameField.Text;

//Captura o cookie
HttpCookie cookie = Request.Cookies[strCookieName];

String strCookieValue = “Vazio”;

//Certifica-se que o cookie existe
if (cookie != null)
   strCookieValue = cookie.Value.ToString();
```

# Laboratório 14

# User Controls

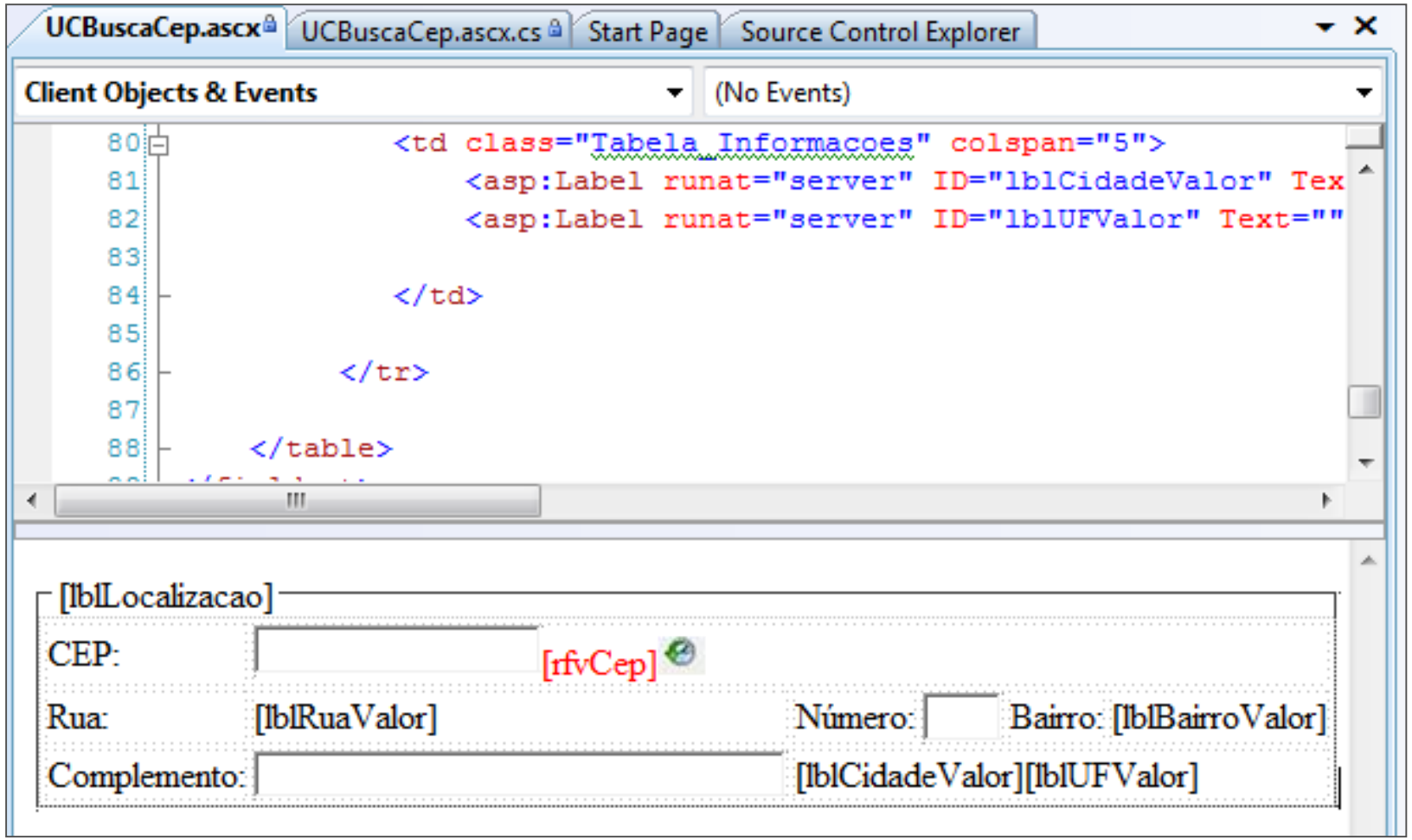
- Controle web criado por um usuário.
  Possui extensão .ascx
  Herda de System.Web.UI.UserControl
  Simplificam a reusabilidade de código e componentes de interface com o usuário dentro de uma aplicação Web
  Contém HTML, mas não as tags <HTML>,<BODY> ou <FORM>
  Contém código para gerenciar seus próprios eventos
  Podem agregar vários controles em um componente reusável

# User Controls



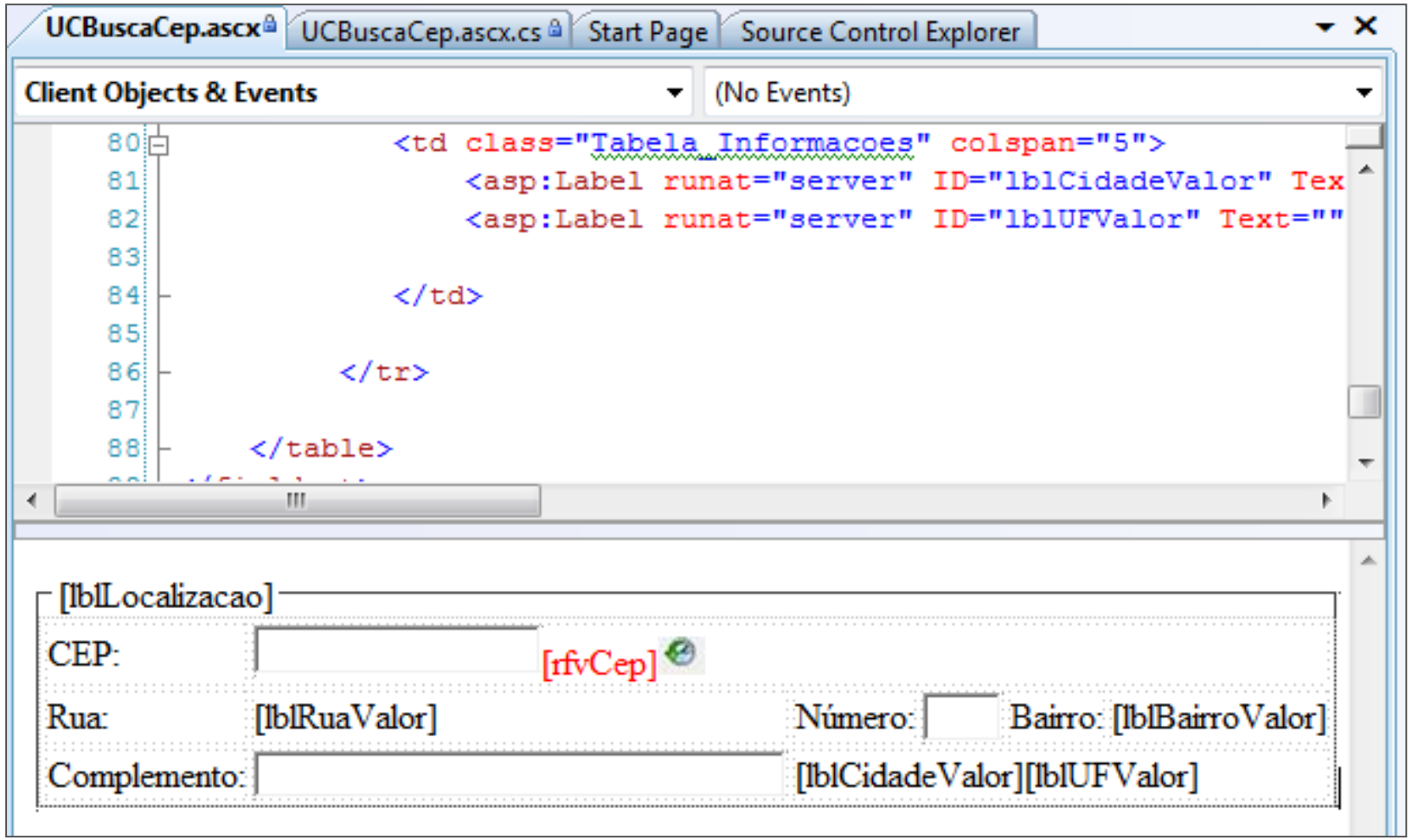
# User Controls

Exemplo de User Control



# Eventos e Delegates

Exemplo de User Control



# Eventos e Delegates

- Conceitos:
  Evento: ação que pode ser gerenciada/manipulada através de código
  Delegate: membro da classe responsável por "delegar" as ações correspondentes a ocorrência de um evento ao(s) manipulador(es) de eventos correspondentes
  Manipulador de Evento: método responsável pela execução de ações em reação a ocorrência de um evento

# Eventos e Delegates

Cinco passos para se trabalhar com eventos

- Passo 1: declarar o delegate contendo a assinatura do manipulador de evento correspondente ao evento

```
public delegate void FazAlgoDelegate(int x);
```

- Passo 2: declarar o evento (deve ser do mesmo tipo do delegate correspondente)

```
public class UmaClasse
{
    public event FazAlgoDelegate UmEvento;
}
```

# Eventos e Delegates

Cinco passos para se trabalhar com eventos

- Passo 3: disparar o evento na chamada de algum método da classe

```
public class UmaClasse
{
        ...
        public void MetodoEvento(int x) {
                UmEvento(x); }
}
```

- Passo 4: assinar o evento indicando o manipulador de eventos do mesmo através de uma instância

```
UmaClasse obj = new UmaClasse();
obj.UmEvento += new FazAlgoDelegate(ManipuladorEvento);
```

# Eventos e Delegates

- Passo 5: implementar o manipulador de evento (deve respeitar a mesma assinatura definida pelo delegate do evento)

```
public void ManipuladorEvento(int x)
{
      label1.Text = x.ToString();
}
```

# Eventos e Delegates

- Passo 5: implementar o manipulador de evento (deve respeitar a mesma assinatura definida pelo delegate do evento)

```
public void ManipuladorEvento(int x)
{
      label1.Text = x.ToString();
}
```

# Laboratório 15

# Desenvolvendo para internet parte IV

# AJAX

# Convergência Web e Desktop

Aplicação Desktop
Interativa
Rápida
Difícil Implantação
Desatualizada
Roda no Cliente

Aplicação Web
Estática
Lenta
Fácil Implementação
Sempre Atualizada
Roda no Servidor

## RIA - Rich Internet Application

# Experiência do Usuário

- A web hoje é dinâmica?
  - O mesmo conteúdo é apresentado a todos os usuários
  - Aplicações Web ainda perdem de aplicações Desktop
  - Alternativas
    - Java Applets
    - Silverlight
    - Macromedia Flash

# Problemas em uma Aplicação Web

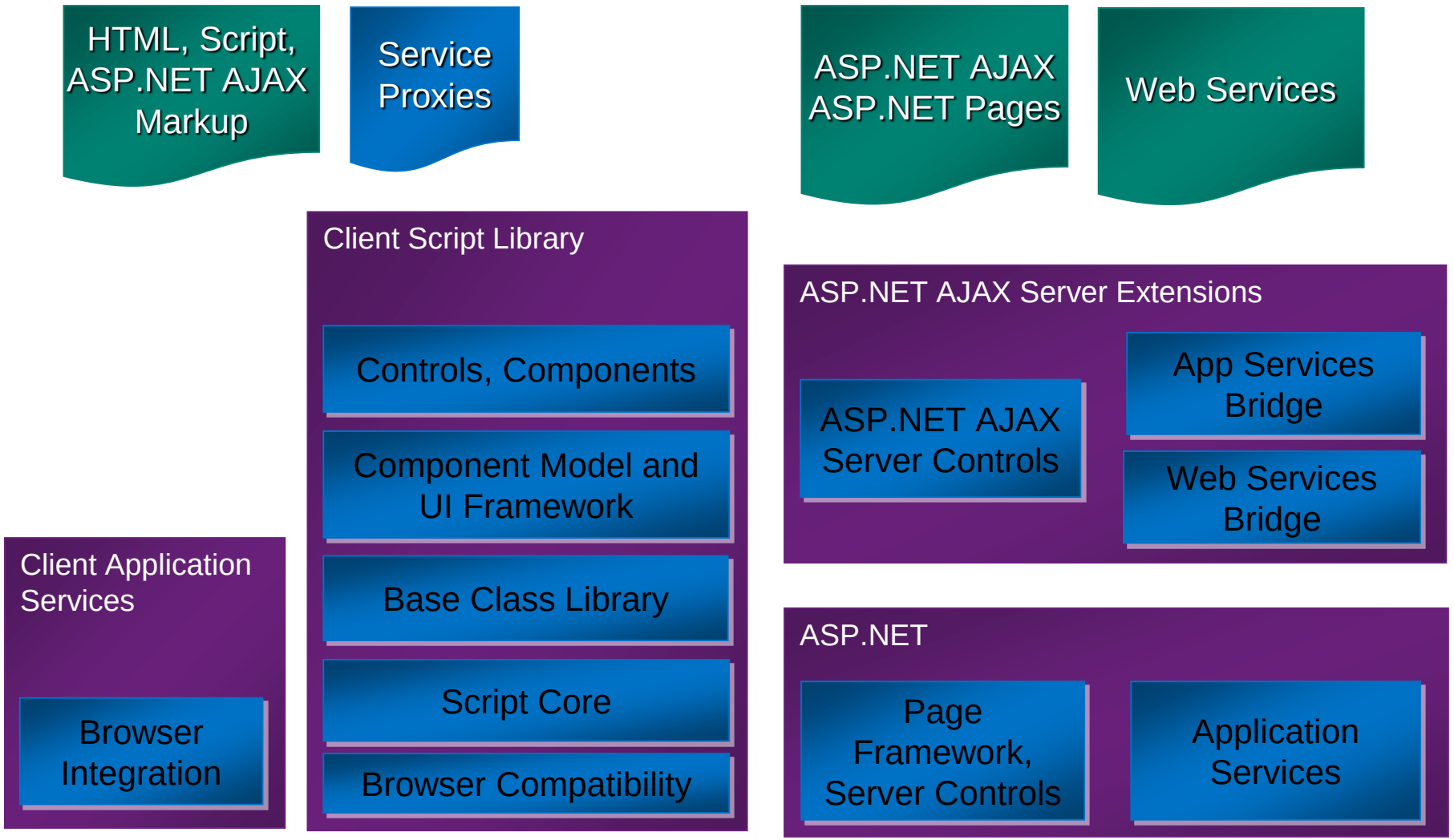
- Post-backs forçam que a página seja recarregada a cada clique.
  Não mantém o estado da página naturalmente (stateless).
  Interfaces ricas são de difícil concepção.
  “Lenta” em relação a aplicações de clientes ricos (desktop).

# Solução para Aplicações Web

- RIA – Rich Internet Application
  Web 2.0

AJAX

# ASP.NET AJAX


HTML, Script, ASP.NET AJAX Markup
Service Proxies
ASP.NET AJAX ASP.NET Pages
Web Services
Client Script Library
Controls, Components
Component Model and UI Framework
Base Class Library
Script Core
Browser Compatibility
Client Application Services
Browser Integration
ASP.NET AJAX Server Extensions
ASP.NET AJAX Server Controls
App Services Bridge
Web Services Bridge
ASP.NET
Page Framework, Server Controls
Application Services

# Renderização Parcial de Páginas

- A renderização de partes de páginas é suportada por um conjunto de controles do servidor e scripts no cliente
  Permite atualizar de forma assíncrona pedaços de uma página sem a necessidade do postback completo da página
  Principais componentes envolvidos:
  XMLHttpRequest
  ScriptManager
  UpdatePanel
  UpdateProgress

# XmlHttpRequest

- Objeto que a linguagem JavaScript  implementa  e está presente nos navegadores
  Tem a capacidade de executar a leitura remota de dados de forma assíncrona, permitindo assim a execução de outras tarefas imediatamente após a chamada
  Retorna dados em formato XML e texto
  PADRÃO RECONHECIDO PELO W3C! http://www.w3.org/TR/XMLHttpRequest/

# ScriptManager

- Disponível em System.Web.Ui
  Gerencia elementos AJAX em uma página ASP.NET
  Componentes e scripts
  Renderização parcial de páginas
  Requisições do cliente
  Respostas do servidor
  Uso obrigatório se forem utilizados os componentes UpdatePanel, UpdateProgress e Timer

# ScriptManager

- Proriedades:
  EnablePartialRendering – deve possuir valor true (valor-padrão) para habilitar renderização parcial de páginas; alterável somente antes ou durante o evento Init  da página
  SupportsPartialRendering – deve possuir valor true para habilitar renderização parcial de páginas; se não atribuído, o valor é obtido através de consulta ao navegador

# UpdatePanel

- Disponível em System.Web.UI
  Controle ASP.NET AJAX que cria um painel atualizável em uma página ASP.NET AJAX, permitindo postbacks baseados em XmlHttpRequest
  É possível colocar múltiplos componentes UpdatePanel em uma mesma página

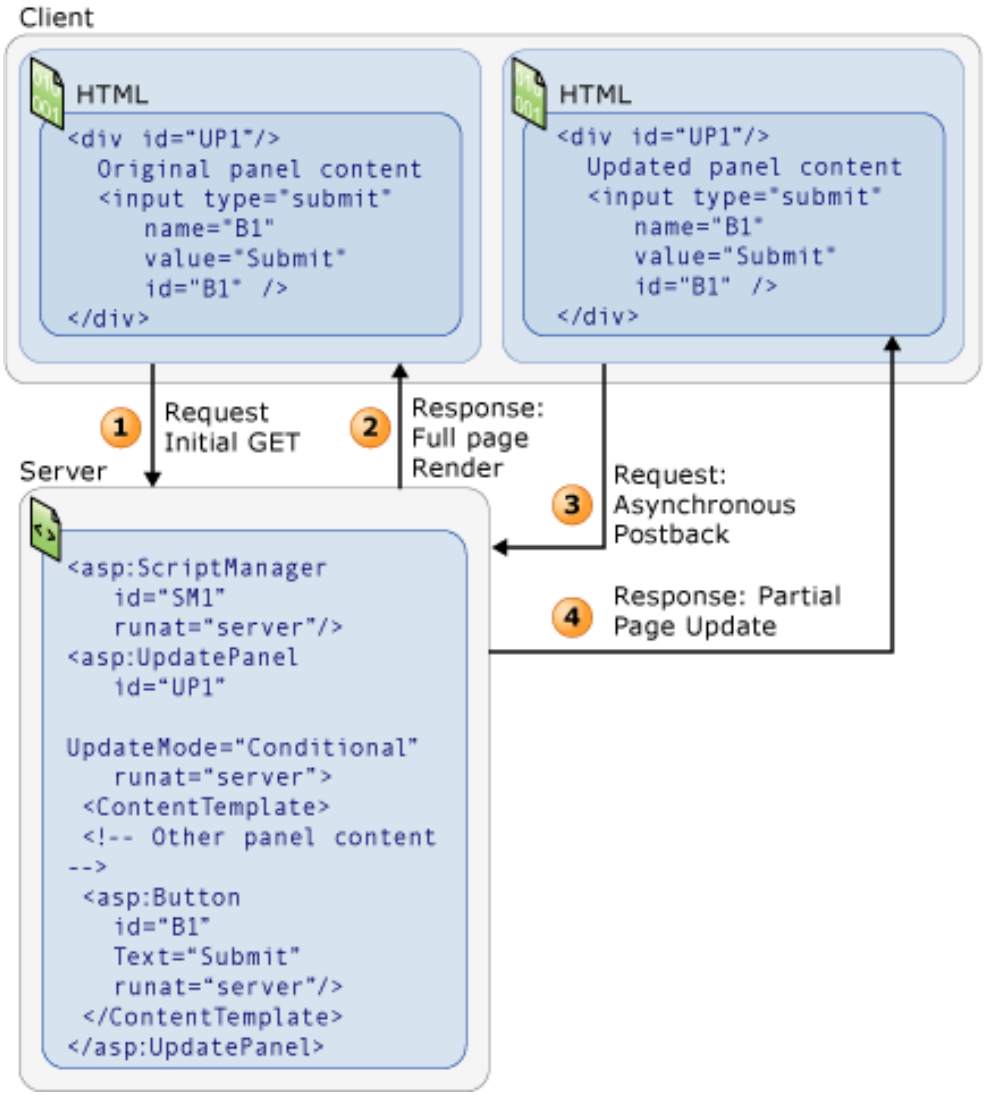
# UpdatePanel

```
<asp:UpdatePanel ID="UpdateMaster“ runat="server">
    <ContentTemplate>
        <div>
         <asp:GridView ID="GridView1" runat="server" />
                    ...
         </asp:GridView>
        </div>
    </ContentTemplate>
</asp:UpdatePanel>
```

# UpdatePanel

- Ciclo de vida de um postback utilizando o ASP.NET AJAX:
  ASP.NET AJAX intercepta as ações de postback da página
  Usa XMLHttpRequest para disparar o postback ao servidor que ocorre normalmente
  Apenas os conteúdos dos UpdatePanel são retornados
  As regiões alteradas no UpdatePanel são atualizadas no cliente
  Todos os postbacks gerados por controles dentro do UpdatePanel serão tratados como postback Ajax com atualizações incrementais da página
  Postbacks para controles fora do UpdatePanel transcorrerão da forma convencional por padrão

# UpdatePanel



# UpdatePanel

- Propriedades:
  UpdateMode – define quando é realizado a atualização do painel
  Always – sempre realiza atualização a qualquer postback
  Conditional – realiza atualização quando um postback assíncrono específico ocorre

# UpdatePanel

- Por padrão, todos os controles dentro de um UpdatePanel podem disparar eventos para o postback assíncrono daquele UpdatePanel
  Controles fora de um UpdatePanel podem também disparar um postback assíncrono em um UpdatePanel
  Adiciona-se Triggers em um UpdatePanel para permitir que outros controles disparem postback assíncrono

```
<asp:UpdatePanel ID="UpdateMaster“ runat="server">
    ...
    <Triggers>
        <asp:AsyncPostBackTrigger ControlID=“Button1" EventName="Click" />
    </Triggers>
</asp:UpdatePanel>
```

# UpdatePanel

- Provê feedback no processo de atualização durante um postback assíncrono

```
<asp:UpdateProgress ID="UpdateProgress1" runat="server">
    <ProgressTemplate>
                ...
    </ProgressTemplate>
</asp:UpdateProgress>
```

# UpdatePanel

- Propriedades:
  AssociatedUpdatePanelID – referência para o UpdatePanel cujo postback assíncrono indica que o UpdateProgress deve ser mostrado; se o valor não for informado, será associado, por padrão, a qualquer postback assíncrono da página
  DynamicLayout – indica se o componente possui uma área reservada (false)ou não (true) no design da página
  por padrão

# Laboratório 16

# ASP.NET AJAX Control Toolkit

- Um rico conjunto de controles e extenders que transformam a tarefa de construir uma interface rica utilizando ASP.NET AJAX uma tarefa simples e rápida
  Exemplos de fácil compreensão
  SDK que simplifica a criação e reutilização de seus próprios controles
  Código fonte e documentação completa
  Mais de 30 componentes e extenders
  Disponível em:
  https://www.nuget.org/packages/AjaxControlToolkit/

# Toolkit Controls

Accordion

AlwaysVisibleControl

Animation

CascadingDropDown

CollapsiblePanel

ConfirmButton

DragPanel

DropDown

DropShadow

DynamicPopulate

FilteredTextBox

HoverMenu

ModalPopup

MutuallyExlcusiveCheckBox

NoBot

NumericUpDown

PagingBulletedList

PasswordStrength

PopupControl

Rating

ReorderList

ResizableControl

RoundedCorners

Slider

TextBoxWatermark

ToggleButton

UpdatePanelAnimation

ValidatorCallout

# Aprimorando Controles Existentes

- Control Extenders

  Estender controles ASP.NET com funcionalidades de controles ASP.NET AJAX

  Encapsular comportamentos tanto no lado do cliente quanto do lado do servidor

  Mesmo modelo de programação de controles ASP.NET

```
<asp:TextBox runat="server" ID="TextBox1" />
<asp:AutoCompleteExtender runat="server" ID="AC1”
        TargetControlID="TextBox1"
        ServicePath="AutoComplete.asmx“
        ServiceMethod="GetWords" Enabled="true"
        MinimumPrefixLength="1" />
```

# Laboratório 17

# Web Services

# O que é Web Service?

- É um serviço disponível na Internet, através de um Servidor Web
  Possui funções contendo suas regras de negócios, que podem ser acessadas através de aplicativos
  Possibilita a comunicação entre Sistemas
  Tecnologia que torna possível realizar transações, troca de dados entre empresas,  que antes eram difíceis ou impossíveis

# Web Services - Características

- São baseados em Padrões da Web
  Os dados trafegam em formato XML, através do protocolo SOAP
  Independente de plataforma, ou seja, sistemas heterogêneos podem se comunicar facilmente
  Pode retornar vários tipos de dados, como por exemplo uma tabela do banco de dados

# Por que Web Services?

- Necessidade de Integração entre Negócios (B2B)
  Tendência do Software como Serviço

# Pré-Web Services

- Disquete, Email, FTP ou Compartilhamento de Redes
  Comunicação Aplicativo X Aplicativo
  Banco de Dados
  Arquivos (Ex: Texto, como CSV; ou em Protocolo Específico)
  Invocação de Objetos Remotos (Ex: CORBA, DCOM, IIOP, ORB)

# Pré-Web Services - Problemas

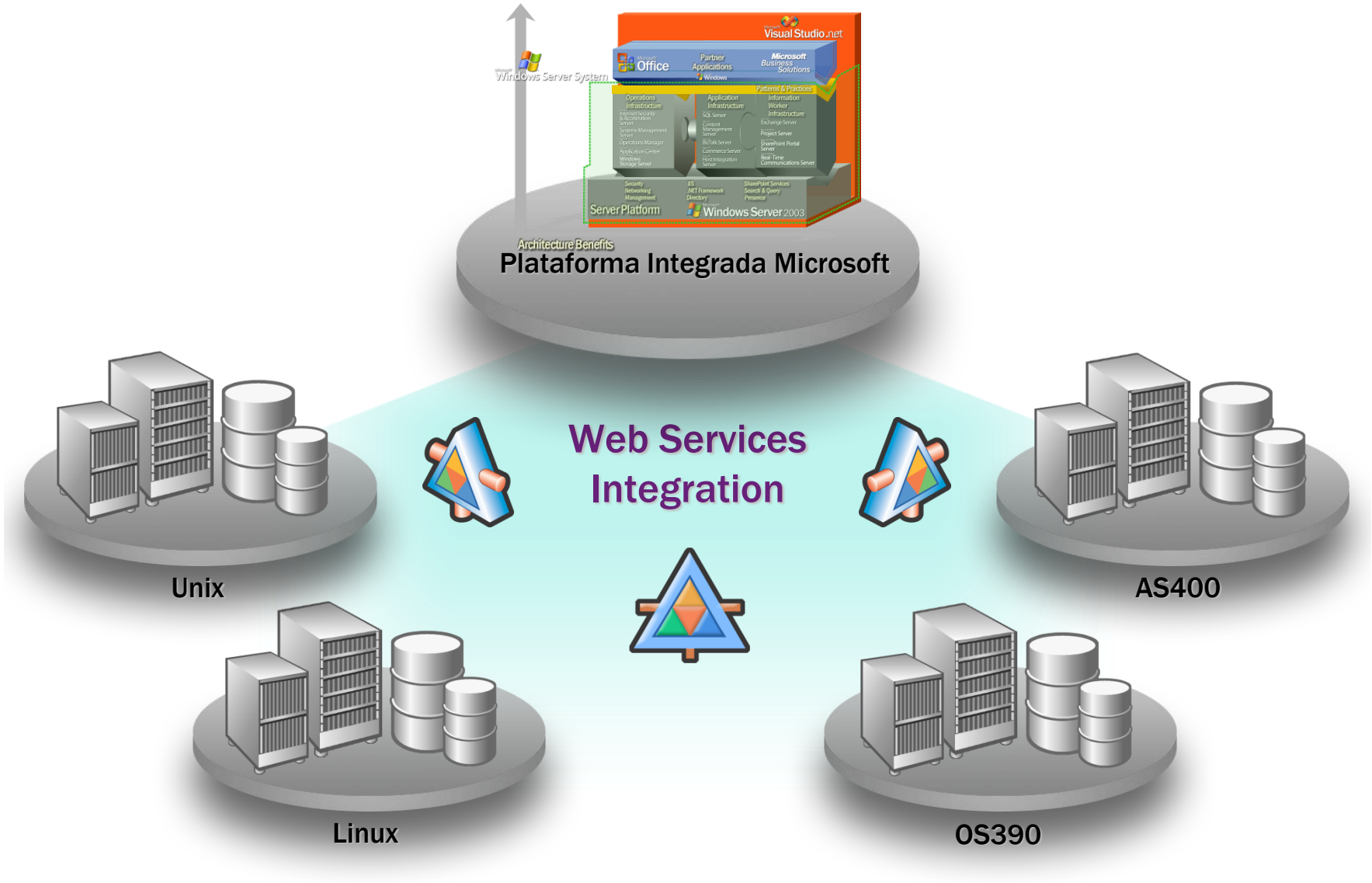
- Tecnologias dependentes de plataforma
  Difícil integração (necessidade de bridge)
  Criava-se um pequeno protocolo de comunicação
  Necessidade de Transformação de Dados
  Pouco segura, complexas, caras, baixa produtividade e pouco robustas

# Uma Solução Integrada...

- Deve ser independente de software e hardware
  Prover Serviços através das barreiras existentes entre diferentes companhias
  Promover automação

  Como conseguir tudo isso?
  Web Services!

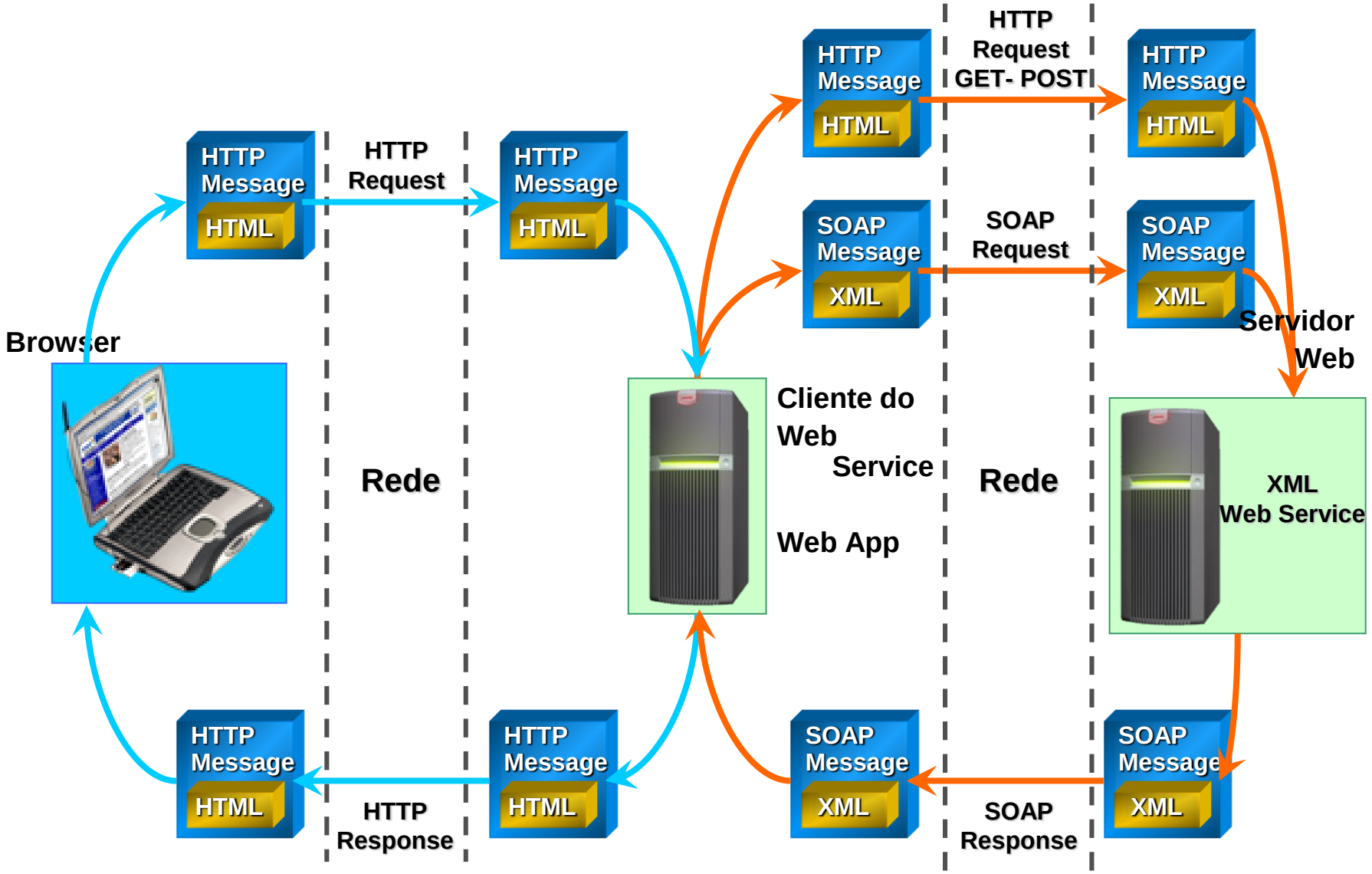
# Integração entre Plataformas



# Infraestrutura

- Web Services provêm meios de objetos interagirem utilizando a Internet como meio de transmissão ("middleware")
  Baseado em diversos protocolos padrões:
  Simple Object Access Protocol (SOAP)
  Universal Description, Discovery and Integration (UDDI)
  Web Services Description Language (WSDL)

# Web App + Web Services



# O que é o WSDL?

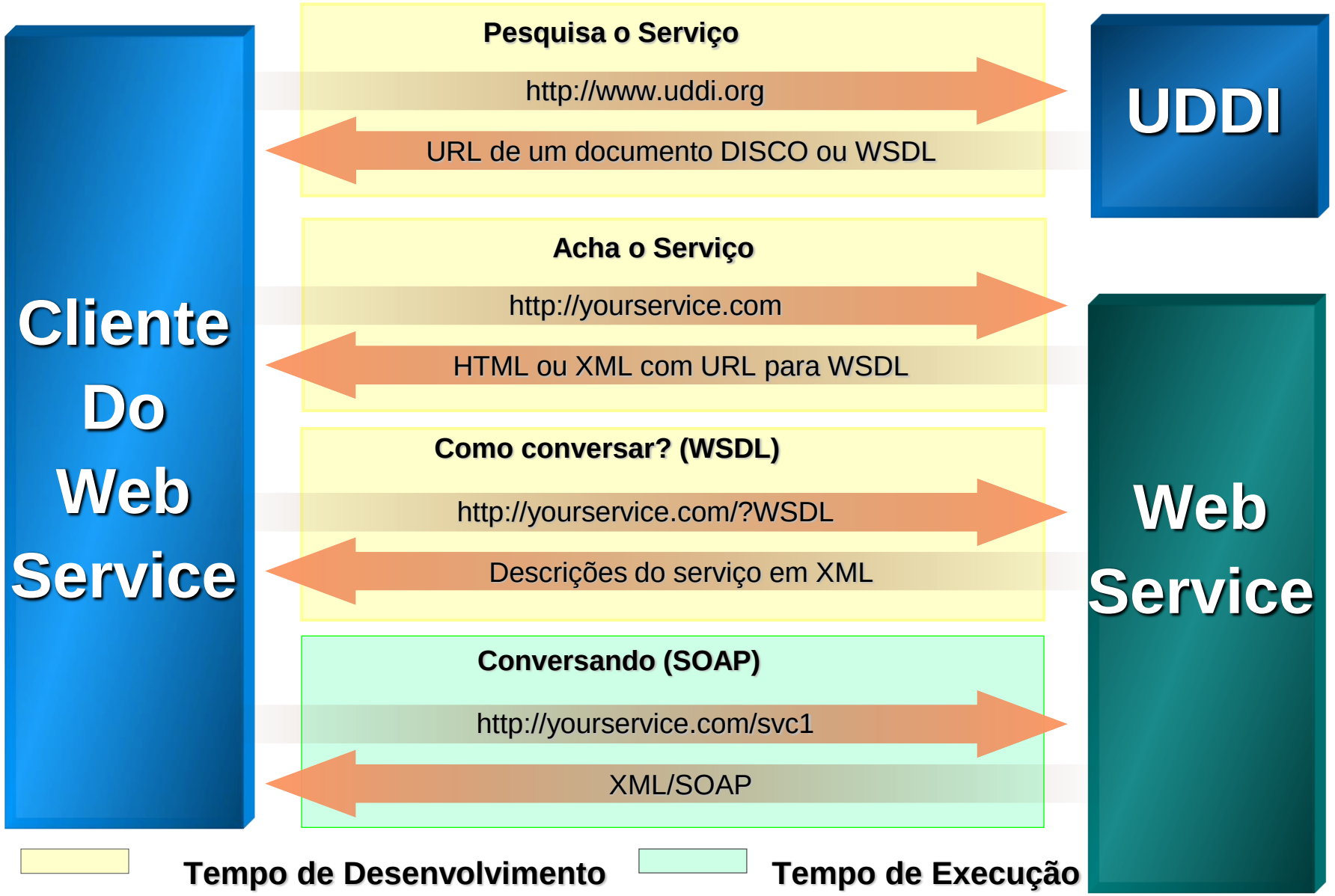
- WSDL – Web Services Description Language
  Documento XML que define as interfaces de seu Web Service
  Mostra os métodos e suas assinaturas

# O que é o DISCO e UDDI?

- DISCO (Discovery of WS)
  & UDDI (Universal Description, Discovery and Integration)
  "Engine de Busca por Web Services"
  "Páginas Amarelas"

  http://uddi.xml.org/

# Funcionamento



# Laboratório 18