ASP.NET

Ing. Cesare Monti

out line

- cos'è
  - chi come dove e quando
- come funziona la baracca
- oggetti - handling HTML
- eventi
- validatori
- ambiente di esecuzione

cos'è

- la risposta Microsoft alle esigenze del server side
- l'evoluzione di ASP
- una tecnologia per la scrittura di pagine dinamiche utilizzando i linguaggi di casa Microsoft

come funziona

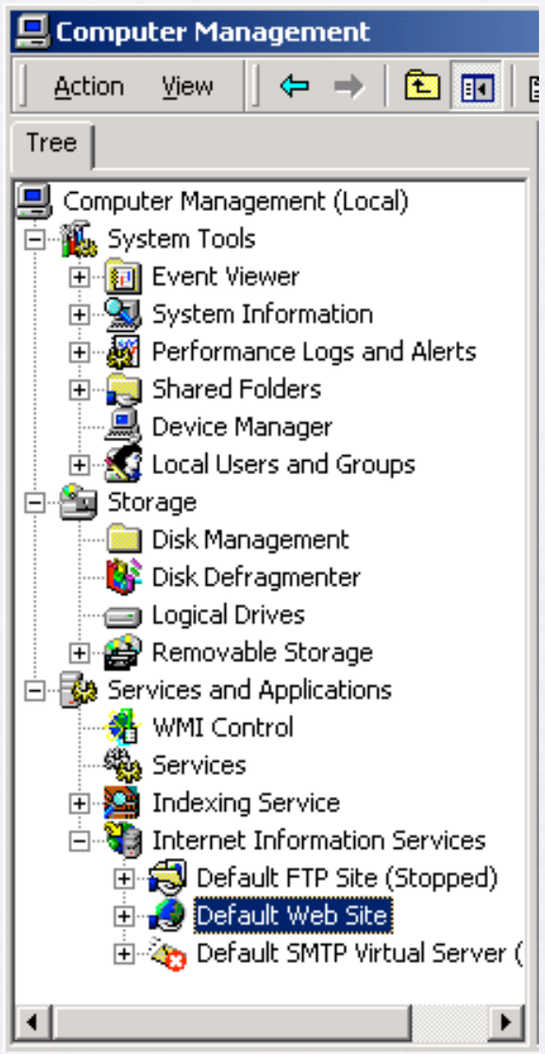
- ...siamo su un server ...
  - ci serve un container (as Java do)

come funziona

- Il container oltre ai parametri di esecuzione (che vedremo alla fine) necessita di “contenere” tutti i file da “eseguire” direttamente all’interno di una virtual directory impostabile sul server web

- il server web si chiama Internet Information Service (IIS)

come funziona



Steps for creating a virtual directory

Control Panel
> Administrative Tools
> Computer Management

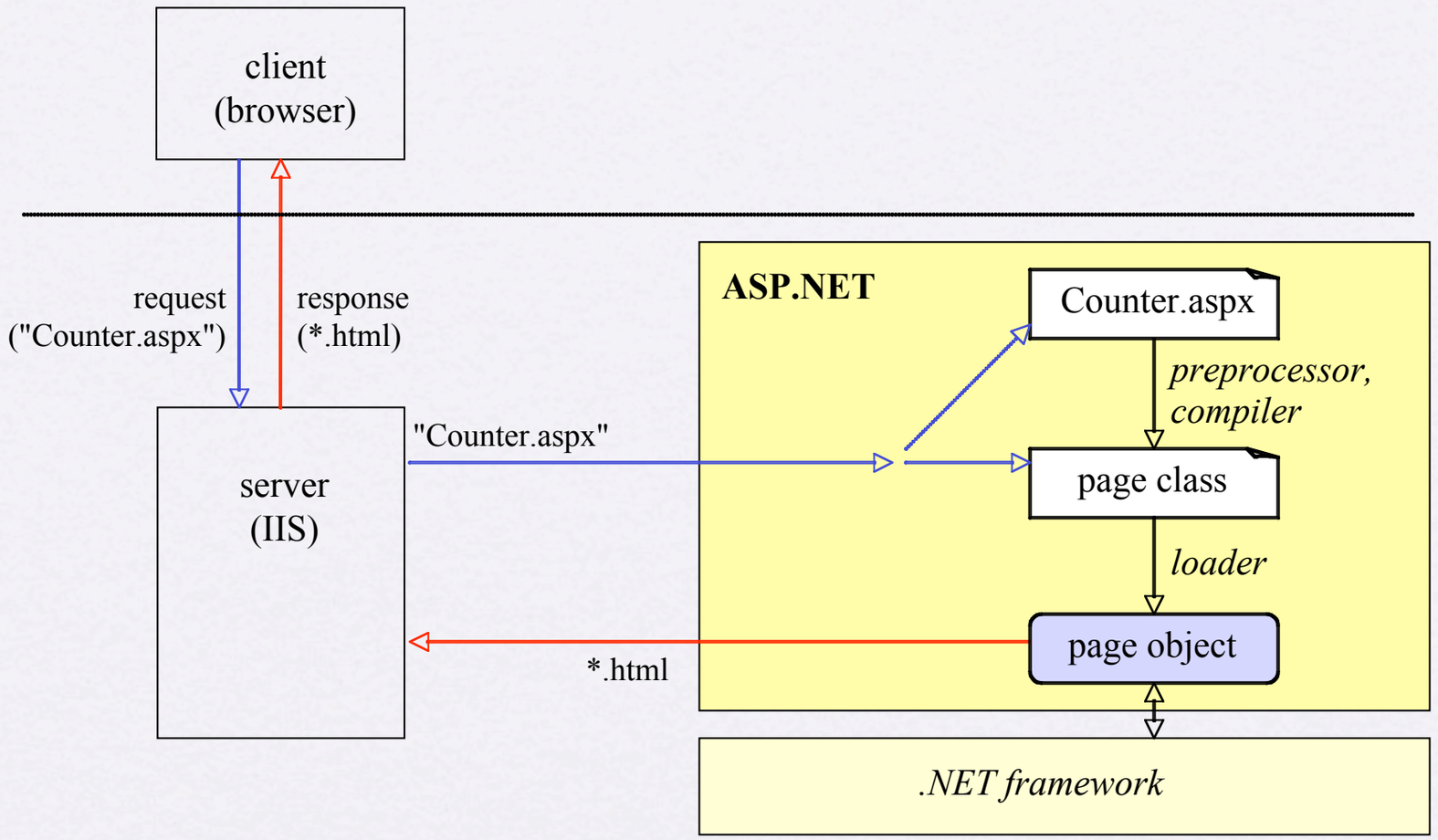
right-click on *Default Web Site*
> New ... Virtual Directory

follow the dialog

All aspx files must be in a virtual directory

accessible as
http://<site-url>/<virtualDirName>/myfile.aspx

come funziona



counter.aspx

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.IO" %>
<html>
        <head> <title>Page counter</title> </head>
        <body>
                <h1>Welcome</h1>
                You are visitor number <%
                FileStream s = new FileStream("c:\\Data\\Counter.dat", FileMode.OpenOrCreate);
                int n;
                try {
                        BinaryReader r = new BinaryReader(s);
                        n = r.ReadInt32();
                } catch { n = 0; }  // if the file is empty
                n++;
                s.Seek(0, SeekOrigin.Begin);
                BinaryWriter w = new BinaryWriter(s);
                w.Write(n); s.Close();
                Response.Write(n);
                %> !
        </body>
</html>
```

counter.aspx in script tags

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.IO" %>
<html>
	<head>
		<title>Page counter</title>
		<script Language="C#" Runat="Server">
			int CounterValue() {
				FileStream s = new FileStream("c:\\Data\\Counter.dat", FileMode.OpenOrCreate);
				...
				n = r.ReadInt32();
				n++;
				...
				return n;
			}
		</script>
	</head>
	<body>
		<h1>Welcome</h1>
		You are visitor number <%=CounterValue()%> !
	</body>
</html>
```

counter.aspx in "Code behind"

Counter.aspx

```
<%@ Page Language="C#" Inherits="CounterPage" Src="CounterPage.cs" %>
<html>
        <head> <title>Page counter</title> </head>
        <body>
            <h1>Welcome</h1>
            You are visitor number <%=CounterValue()%> !
        </body>
</html>
```

CounterPage.cs

CounterPage.cs

```
using System.IO;

public class CounterPage : System.Web.UI.Page {
        public int CounterValue() {
            FileStream s = new FileStream("c:\\Data\\Counter.dat", FileMode.OpenOrCreate);
            ...
            n = r.ReadInt32();
            n++;
            ...
            return n;
        }
}
```

ma … in un modo o nell'altro …

aspx page *Counter.aspx*

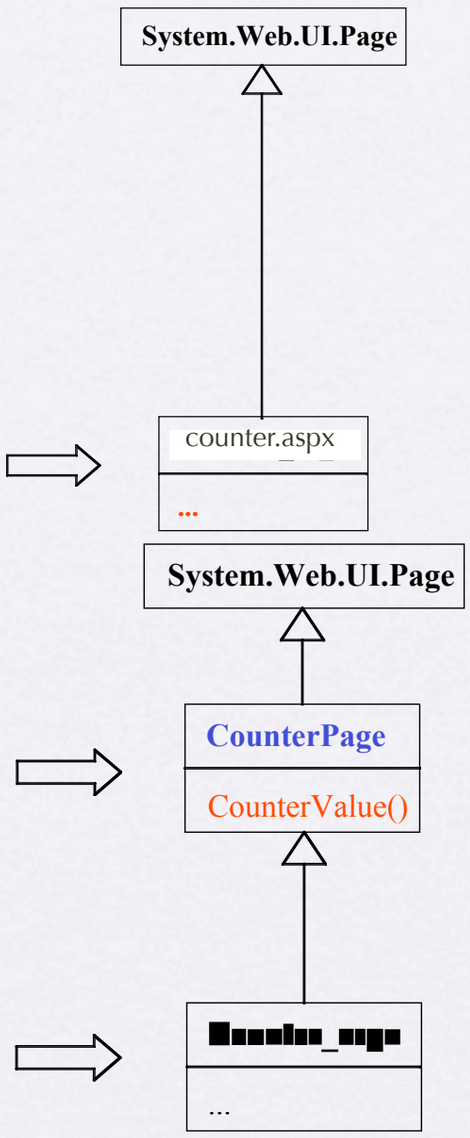
```
<%@ Page Language="C#"%>
<html>
   <body>
      ... <%= ... %>...
   </body>
</html>
```

Code behind *CounterPage.cs*

```
public class CounterPage : System.Web.UI.Page {
   public int CounterValue() {
      ...
   }
}
```

aspx page *Counter.aspx*

```
<%@ Page ... Inherits="CounterPage"%>
<html>
   <body>
      ... <%=CounterValue()%>...
   </body>
</html>
```



counter.aspx

```
namespace ASP {
        using System.IO;
        ...
        public class Counter_aspx : CounterPage {
                private static bool __initialized = false;
                private static ArrayList __fileDependencies;
                public Counter_aspx() {
                        ArrayList dependencies;
                        if ((__initialized == false)) { ... }
                }
                public override string TemplateSourceDirectory {
                        get { return "/Samples"; }
                }
                private void __BuildControlTree(Control __ctrl) {
                        __ctrl.SetRenderMethodDelegate(new RenderMethod
(this.__Render__control1));
                }
                private void __Render__control1(HtmlTextWriter __output,
Control parameterContainer) {
                        __output.Write("\r\n<html>\r\n\t<head> <title>Page
counter</title> </head>\r\n\t<body>\r\n\t\t" +
                                "<h1>Welcome</h1>\r\n\t\tYou are visitor number
");
                        __output.Write(CounterValue());
                        __output.Write(" !\r\n\t</body>\r\n</html>\r\n");
                }
                protected override void FrameworkInitialize() {
                        __BuildControlTree(this);
                        this.FileDependencies = __fileDependencies;
                        this.EnableViewStateMac = true;
this.Request.ValidateInput();
                }
        ...
        }
}
```

oggetti

System.Web

- Compilation
- Handlers
- Hosting
- UI
  - HtmlControls
  - WebControls
- Caching
- Security
- Configuration
- SessionState

System.Windows.Forms

- Design
- ComponentModel

System.Drawing

- Drawing2D
- Printing
- Imaging
- Text

System.Data

- OleDb
- Common
- SqlClient
- SQLTypes

System.Xml

- Xsl
- Schema
- XPath
- Serialization

System

- Collections
- IO
- Security
- Runtime
  - InteropServices
  - Remoting
  - Serialization
- Configuration
- Reflection
- Text
- Diagnostics
- Resources
- Threading
- Globalization

Oggetto pagina

Class Page

```
public class Page: TemplateControl {
   //--- properties
   public  ValidatorCollection Validators { get; }
   public bool IsValid { get; }
   public bool IsPostBack { get; }
   public virtual string TemplateSourceDirectory { get; }
   public HttpApplicationState Application { get; }
   public virtual HttpSessionState Session { get; }
   public HttpRequest Request { get; }
   public HttpResponse Response { get; }
   ...
   //--- methods
   public string MapPath(string virtualPath);
   public virtual void Validate();
   ...
}
```

MapPath(virtPath)
maps the virtual directory to the physical one

Validate()
starts all validators on the page

IsValid
true, if none of the validators on the page reported an error

IsPostBack
true, if the page was sent to the server in a round trip. If the page was requested for the first time IsPostBack == false

TemplateSourceDirectory
current virtual directory, e.g. "/Samples"

Application and **Session**
application state and session state

Request und **Response**
HTTP request and HTTP response

Oggetto request

Class HttpRequest

```
public class HttpRequest {
   public string UserHostName { get; }
   public string UserHostAddress { get; }
   public string HttpMethod { get; }
   public HttpBrowserCapabilities Browser { get; }
   public NameValueCollection Form { get; }
   public NameValueCollection QueryString { get; }
   public NameValueCollection Cookies { get; }
   public NameValueCollection ServerVariables { get; }
   ...
}
```

UserHostName
domain name of the client

UserHostAddress
IP number of the client

```
<body>
   <%= "address = " + Request.UserHostAddress %><br>
   <%= "method = " + Request.HttpMethod %><br>
   <%= "browser = " + Request.Browser.Browser %><br>
   <%= "version = " + Request.Browser.Version %><br>
   <%= "supports JS = " + Request.Browser.JavaScript %><br>
   <%= "server = " + [redacted] %>
</body>
```

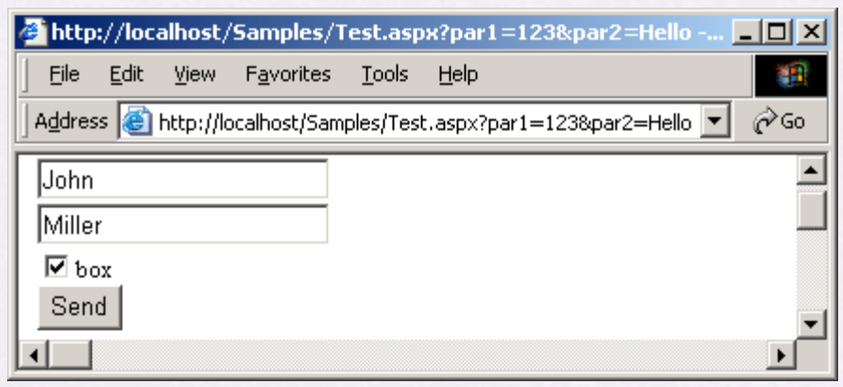
```
address = 127.0.0.1
method = GET
browser = IE
version = 6.0
supports JS = True
server = Microsoft-IIS/5.0
```

request and form

HttpRequest (Request and Form Parameters)

```
<form Runat="server">
   <asp:TextBox ID="text1" Runat="server" /><br>
   <asp:TextBox ID="text2" Runat="server" /><br>
   <asp:CheckBox ID="checkbox" Text="box" Runat="server" /><br>
   <asp:Button ID="button" Text="Send" OnClick="DoClick" Runat="server" />
   <asp:Label ID="lab" Runat="server" />
</form>
```

```
void DoClick (object sender, EventArgs e) {
   lab.Text = "Query string<br>";
   foreach (string par in Request.QueryString.Keys)
      lab.Text += par + " = " + Request.QueryString[par] + "<br>";
   lab.Text += "<br>Form parameters<br>";
   foreach (string par in Request.Form.Keys)
      lab.Text += par + " = " + Request.Form[par] + "<br>";
}
```



Query string
   par1 = 123
   par2 = Hello
Form parameters
   __[redacted] ...
   text1 = John
   text2 = Miller
   checkbox = on
   button = Send

response

Class HttpResponse

```
public class HttpResponse {
   //--- properties
   public string ContentType { get; set; }
   public TextWriter Output { get; }
   public int StatusCode { get; set; }
   public HttpCookieCollection Cookies { get; set; }
   ...
   //--- methods
   public void Write(string s); // various overloaded versions
   public void Redirect(string newURL);
   ...
}
```

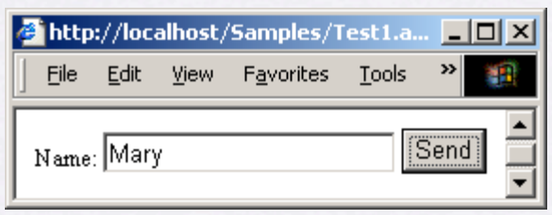
ContentType
MIME type (e.g. text/html)

Output
HTML response stream; can be written to with Write

StatusCode
e.g. 200 for "ok" or
404 for "page not found"

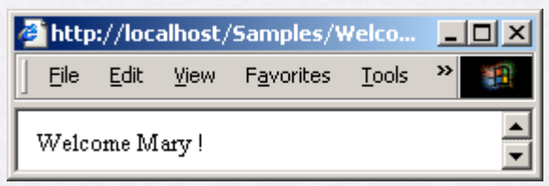
Test1.aspx

```
<form Runat="server">
   Name: <asp:TextBox ID="name" Runat="server" />
   <asp:Button Text="Send" OnClick="DoClick" Runat="server" />
</form>

void DoClick (object sender, EventArgs e) {
   Response.Redirect("Welcome.aspx?name=" + name.Text);
}
```



Welcome.aspx

```
Welcome <%= Request.QueryString["name"] %> !
```



html

```
...
<body>
        <form action="http://www.unsito.com/cgi.exe" method="post">
            <b>Balance:</b>
            <input type="text" name="total" readonly value="0"> Euro<br>
            <input type="text" name="amount">
            <input type="submit" name="ok" value="Pay">
        </form>
</body>
```

... si leggono i valori di total e amount e li si sottomettono all'URL nella action ...

adder.aspx

```
<%@ Page Language="C#" Inherits="AdderPage" Src="Adder.aspx.cs"%>
<html>
        <head><title>Account</title></head>
        <body>
                <form method="post" Runat="server">
                        <b>Balance:</b>
                        <asp:Label ID="total" Text="0" Runat="server"/> Euro<br><br>
                        <asp:TextBox ID="amount" Runat="server"/>
                        <asp:Button ID="ok" Text="Enter" OnClick="ButtonClick" Runat="server" />
                </form>
        </body>
</html>
```

```
using System; using System.Web.UI; using System.Web.UI.WebControls;
public class AdderPage : Page {
        protected Label total;
        protected TextBox amount;
        protected Button ok;
        public void ButtonClick (object sender, EventArgs e) {
                int totalVal = Convert.ToInt32(total.Text);
                int amountVal = Convert.ToInt32(amount.Text);
                total.Text = (totalVal + amountVal).ToString();
                                                        }
}
```

html handling

Counter.aspx

```
<%@ Page Language="C#"
   Inherits="AdderPage"
   Src="Adder.aspx.cs"%>
<html>
   <head><title>Account</title></head>
   <body>
      <form method="post"
Runat="server">
         <b>Balance:</b>
         <asp:Label ID="total" Text="0"
            Runat="server"/> Euro<br><br>
         <asp:TextBox ID="amount"
            Runat="server"/>
         <asp:Button ID="ok"
            Text="Enter"
            OnClick="ButtonClick"
            Runat="server" />
      </form>
   </body>
</html>
```

```
<html>
        <head> <title>Account</title> </head>
        <body>
              <form name="_ctl0" method="post"
                    action="Adder.aspx" id="_ctl0">
                    <input type="hidden" name="__VIEWSTATE"
value="dDwxNTg0NTEzNzMyO3Q8O2w8aTwxP" +
                          "js+O2w8dDw7bDxpPDE
+Oz47bDx0PHA8cDxs"+
                          "PFRleHQ7PjtsPDEwMDs+Pjs+Ozs+Oz4+Oz4
+" +"Oz7uOgbDI3uKWY/X5D1Fw8zmjTZkwg==" />
                    <b>Balance:</b>
                    <span id="total">100</span>
                    Euro<br><br>
                    <input type="text" name="amount"
                          value="100" id="amount" />
                    <input type="submit" name="ok"
                          value="Enter" id="ok" />
              </form>
        </body>
</html>
```

html handling

- general notation:

- <asp:ClassName PropertyName="value" ... Runat="server" />
  - <asp:Label ID="total" Text="Hello" ForeColor="Red" Runat="server" />

```
public class Label: WebControl {
        public virtual string ID { get {...} set {...} }
        public virtual string Text { get {...} set {...} }
        public virtual Color ForeColor { get {...} set {...} }
        ...
}
```

- Tutti i controlli web sono nel namespace *System.Web.UI*

- *è anche accettata la notazione:*

```
<asp:Label ID="total" ForeColor="Red" Runat="server" >
        Hello
</asp:Label>
```

html handling

- vantaggi:
  - La pagina è un oggetto
  - tutti gli elementi visuali sono oggetti
    - ... e ognuno può implementarne altri a piacimento
  - ogni pagina può accedere all'intero spazio di namespace di .NET
  - lo stato di ogni elemento nella pagina è persistente

html handling

Label **abc**

TextBox

Button Click

RadioButton Radio

CheckBox Check

DropDownList Linz

ListBox

Calendar

DataGrid

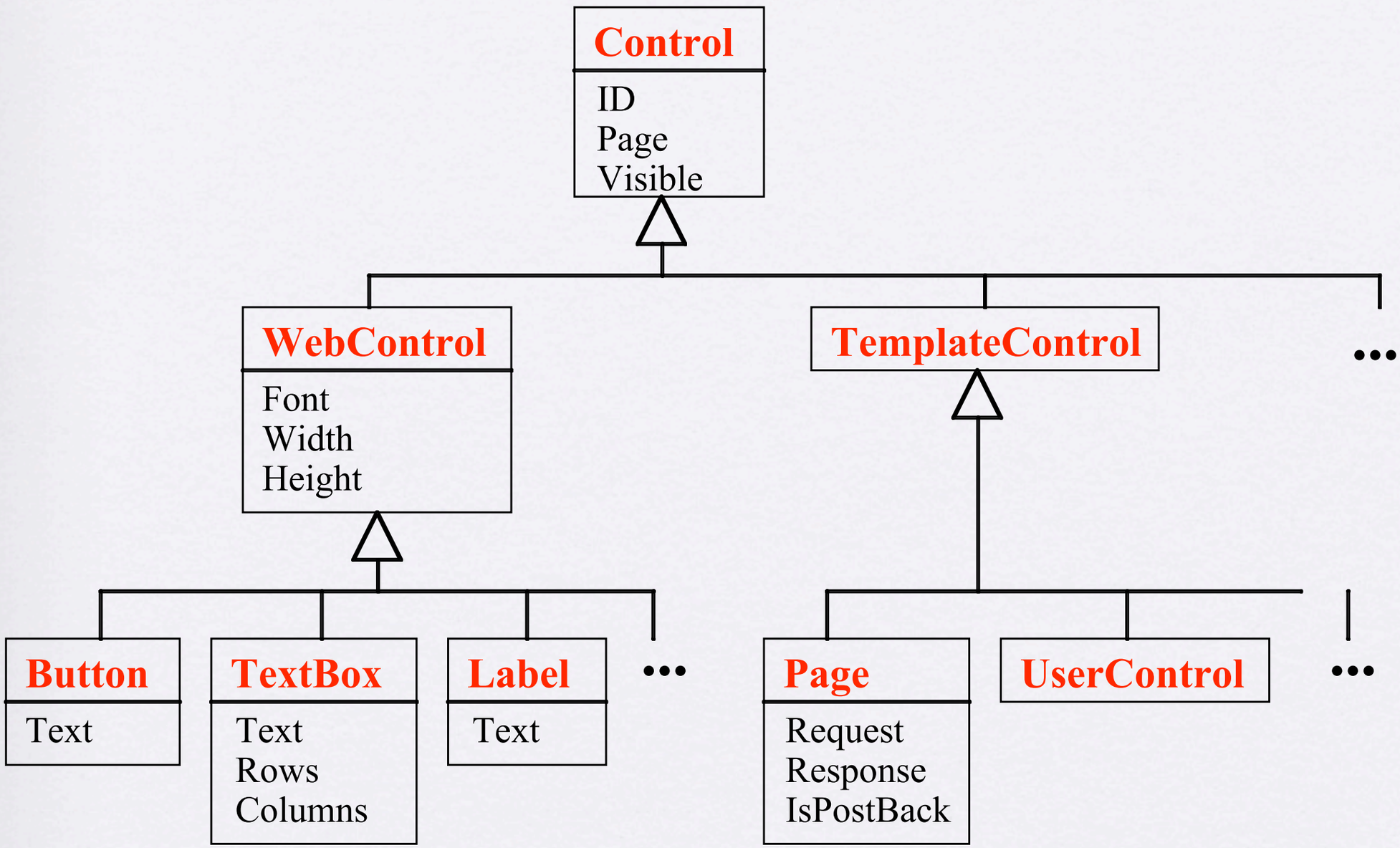
EmployeeID	FirstName	LastName
1	Nancy	Davolio
2	Andrew	Fuller
3	Janet	Leverling
4	Margaret	Peacock
5	Steven	Buchanan
6	Michael	Suyama
7	Robert	King
8	Laura	Callahan
9	Anne	Dodsworth

...

User Controls

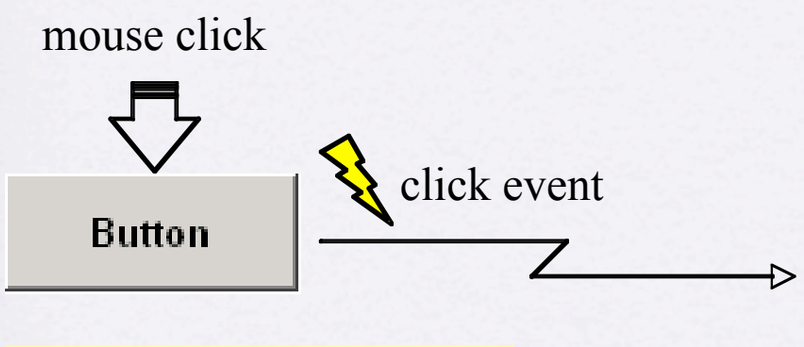
Custom Controls

html handling



Eventi

- il modello a eventi ... non è banale



event handler

```
void DoClick (object sender, EventArgs e) {
    ...
}
```

```
<asp:Button
    Text="..."
    OnClick="DoClick"
    Runat="sever" />
```

Client *Server*

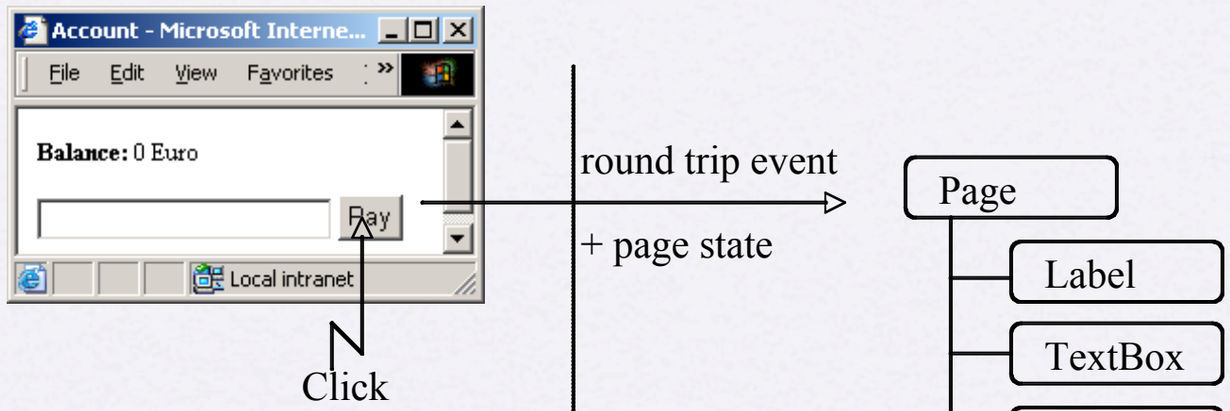
Eventi

- Tipi di eventi

Control	Event	When does the event occur?↲
all	**Init** **Load** **PreRender** **Unload**	• when the control is created • after the data that were sent by the browser have been loaded into the control • before HTML code for this control is generated • before the control is removed from memory
Button	**Click**	when the button was clicked
TextBox	**TextChanged**	when the contents of the TextBox changed
CheckBox	**CheckedChanged**	when the state of the CheckBox changed
ListBox	**SelectedIndexChanged**	when a new item from the list has been selected

Eventi

- Round trip

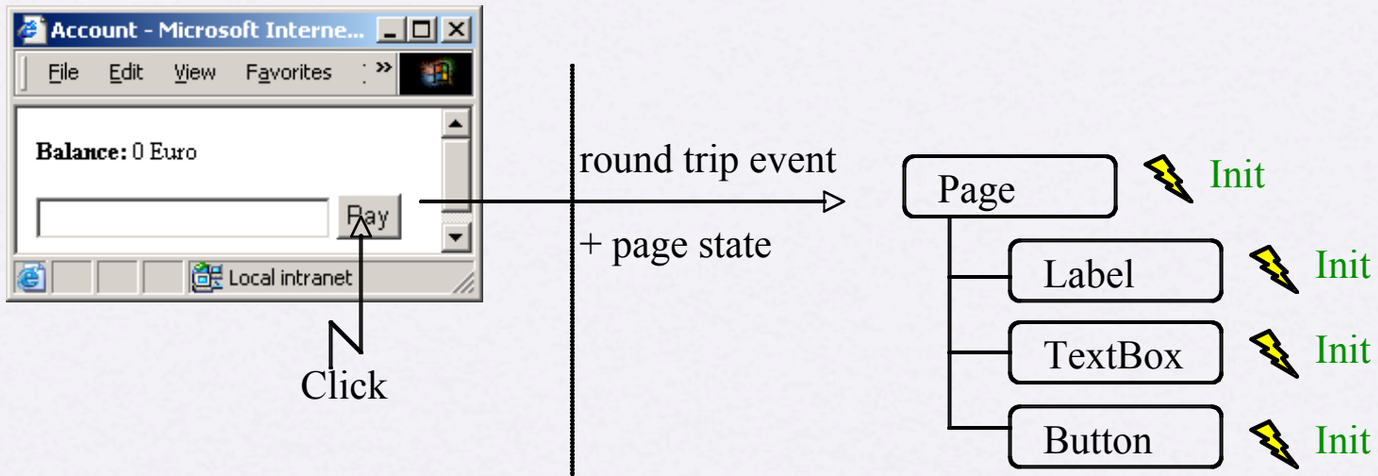


1. **Creation**
   create page object and its controls

Client | *Server*

Eventi

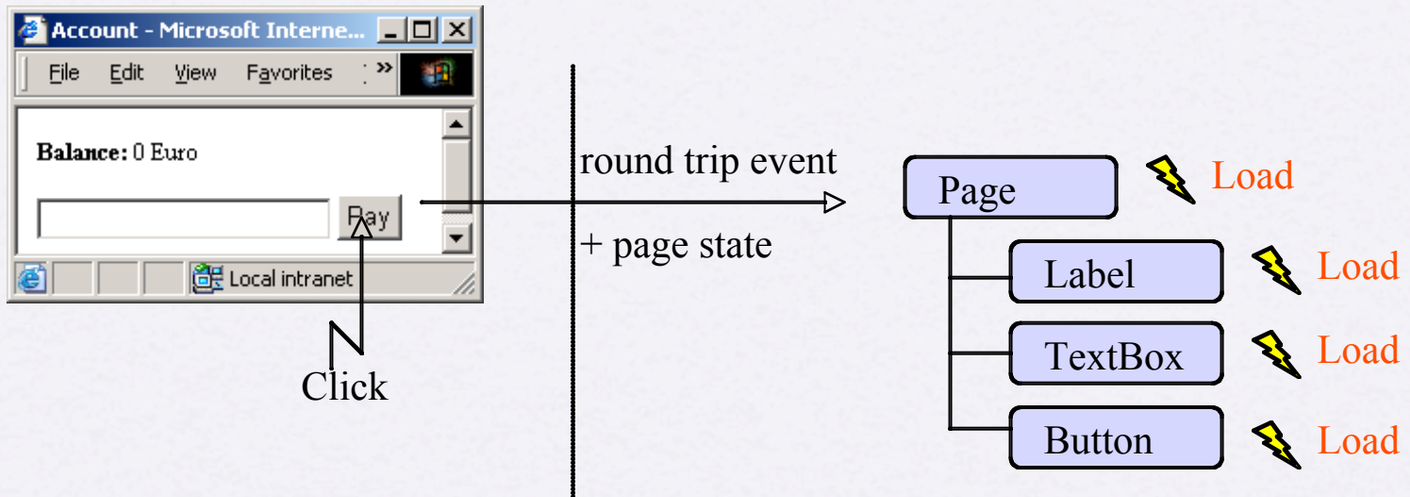
- round trip



2. **Initialisation**
   - raise *Init* events

Client | *Server*

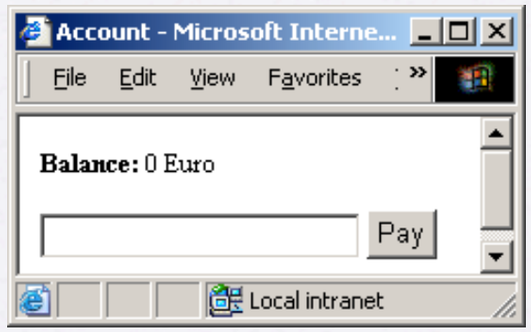
Eventi

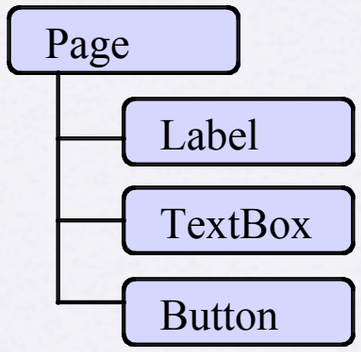


3. **Loading**
   - load controls with the values that the user has entered (page state)
   - raise *Load* events

Client | *Server*

Eventi



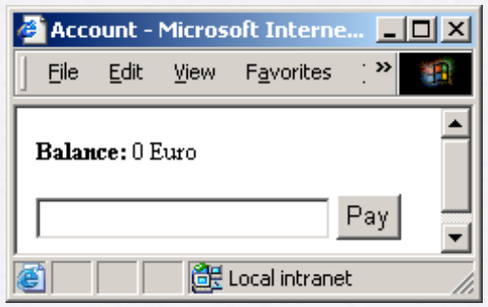


4. **Action**
   handle event(s)
   (Click, TextChanged, ...)

Client

Server

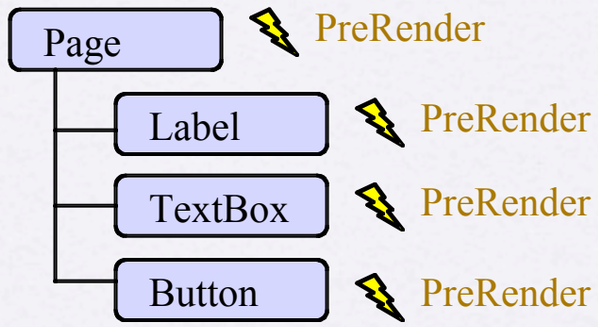
Eventi




```
<html>
   ...
   <input type="text" ...>
   <input type="button" ...>
   ...
</html>
```

HTML

+ page state

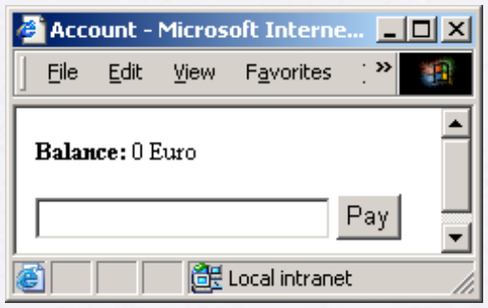


5. **Rendering**
   - raise *PreRender* events
   - call *Render* methods of all controls, which render the controls to HTML

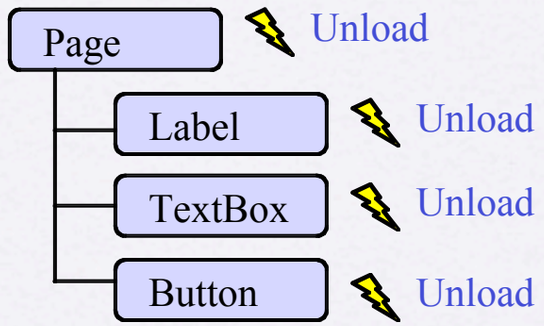
Client

Server

Eventi




```
<html>
   ...
   <input type="text" ...>
   <input type="button" ...>
   ...
</html>
```



6. **Unloading**
   - raise *Unload* events for cleanup actions

Client

Server

Eventi

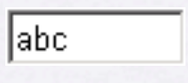
- Quali producono un Round Trip?

Round trip events (cause an immediate round trip)

Click

```
<asp:Button Text="click me" Runat="server"
    OnClick="DoClick" />
```

Delayed events (are handled at the next round trip)



TextChanged

```
<asp:TextBox Runat="server"
    OnTextChanged="DoTextChanged" />
```

SelectedIndexChanged

```
<asp:ListBox Rows="3" Runat="server"
    OnSelectedIndexChanged="DoSIChanged" />
```

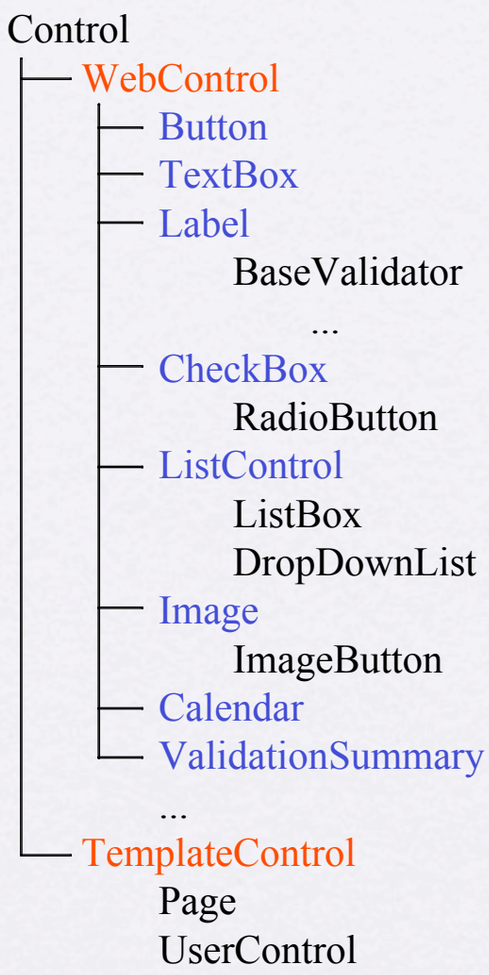
AutoPostBack (causes a delayed event to lead to an immediate round trip)

TextChanged

```
<asp:TextBox Runat="server"
    AutoPostBack="true"
    OnTextChanged="DoTextChanged" />
```

Controlli web

Web Control Hierarchy



Controlli web

Class Control

```
public class Control: ... {
    public virtual string ID { get; set; }
    public virtual ControlCollection Controls { get; }
    public virtual Control Parent { get; }
    public virtual Page Page { get; set; }
    public virtual bool Visible { get; set; }
    protected virtual StateBag ViewState { get; }
    public virtual bool EnableViewState { get; set; }
    ...
    public virtual bool HasControls();
    public virtual Control FindControl (string id);
    public virtual void DataBind();
    protected virtual void LoadViewState (object state);
    protected virtual object SaveViewState();
    protected virtual Render (HtmlTextWriter w);
    ...
    public event EventHandler Init;
    public event EventHandler Load;
    public event EventHandler DataBinding;
    public event EventHandler PreRender;
    public event EventHandler Unload;
    ...
}
```

Properties

- name of the control
- nested controls
- enclosing control
- page to which the control belongs
- should the control be visible?
- state of this control (see later)
- should the state be persistent?

Methods

- does the control have nested controls?
- searches for a nested control with the name id
- loads data from a data source
- loads the state from the request stream
- saves the state to the response stream
- renders the control to HTML

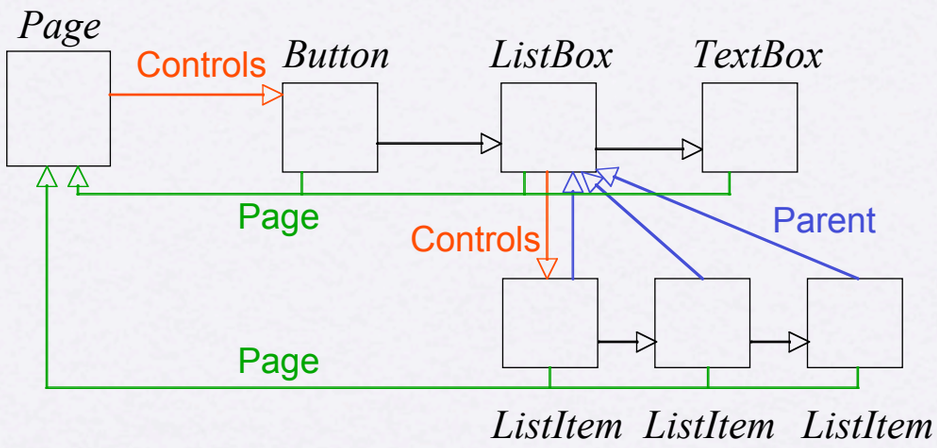
Events

- after the control was created
- after the state was loaded from the request
- after DataBind was called
- before the control is rendered to HTML
- before the control is released

Controlli web

Properties of Class Control

Containment relationship



ViewState

```
public void ButtonClick (object Button, EventArgs e) {
   int clicks = ViewState["nClicks"] == null ? 0 : (int) ViewState["nClicks"];
   ViewState["nClicks"] = ++clicks;
}
```

- programmers can store arbitrary data in *ViewState*
- *ViewState* is stored in a hidden field of the HTML page
- this here is the *ViewState* of *Page* (*ViewState* of *Button* is protected)

Controlli web

Class WebControl

```
public class WebControl: Control {
    public virtual Unit Width { get; set; }
    public virtual Unit Height { get; set; }
    public virtual FontInfo Font { get; set; }
    public virtual Color ForeColor { get; set; }
    public virtual Color BackColor { get; set; }
    public virtual Unit BorderWidth { get; set; }
    public virtual Color BorderColor { get; set; }
    public virtual BorderStyle BorderStyle { get; set; }
    public virtual bool Enabled { get; set; }
    public virtual short TabIndex { get; set; }
    public virtual string ToolTip { get; set; }
    ...
}
```

Colors

namespace: System.Drawing

```
public struct Color {
    public static Color Blue { get; }
    public static Color Red { get; }
    public static Color Yellow { get; }
    ...
    public static Color FromArgb (int R, int G, int B);
}
```

Units of Measurement

```
public struct Unit {
    public Unit (double value, UnitType type);
    public double Value { get; }
    public UnitType Type { get; }
    ...
}
public enum UnitType {Cm, Em, Ex, Inch,
    Mm, Percentage, Pica, Pixel, Point
}
```

setting properties in a web page: default: Pixel

```
<asp:TextBox ID="tb" Width="100" ... />
<asp:TextBox ID="tb" Width="10cm" ... />
<asp:TextBox ForeColor="Red" ... />
```

setting properties in the script code:

```
tb.Width = 100;  // default: Pixel
tb.Width = new Unit(10, UnitType.Cm);
tb.ForeColor = Color.Red;
```

Controlli web

WebControl (Fonts)

Fonts

```
public sealed class FontInfo {
    public string Name { get; set; }
    public FontUnit Size { get; set; }
    public bool Bold { get; set; }
    public bool Italic { get; set; }
    public bool Underline { get; set; }
    ...
}
public struct FontUnit {
    public FontUnit (Unit size);
    public FontUnit (FontSize size);
    public Unit Unit { get; }
    public FontSize Type { get; }
    ...
}
public enum FontSize { AsUnit, XSmall,
    Small, Medium, Large, XLarge, ... }
```

setting the font in a web page:

```
<asp:Button ID="b1" Font-Name="Arial"
   Font-Size="Large" Font-Bold="true" .../>
<asp:Button ID="b2" Font-Name="Times"
   Font-Size="12px" Font-Italic="true" ... />
```

setting the font in the script code:

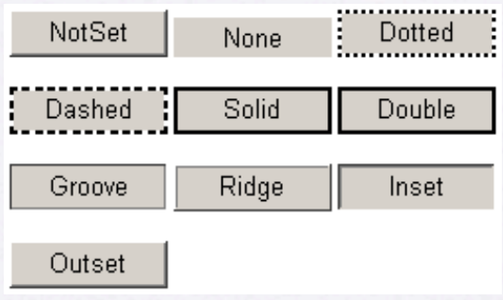
```
b1.Font.Name = "Arial";
b1.Font.Size = new FontUnit(FontSize.Large);
b1.Font.Bold = true;
b2.Font.Name = "Times";
b2.Font.Size = new FontUnit(12);
b2.Font.Italic = true;
```

Controlli web

WebControl (Other Properties)

BorderStyle

```
public enum BorderStyle {
    NotSet, None, Dotted, Dashed,
    Solid, Double, Groove, Ridge,
    Inset, OutSet
}
```



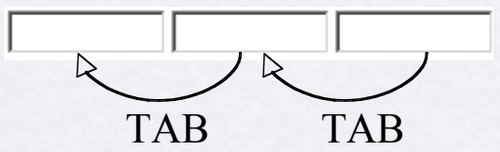
Enabled

```
<asp:Button Enabled="false" ... />
```

displays the control, but deactivates it

TabIndex

```
<asp:TextBox TabIndex="3" ... />
<asp:TextBox TabIndex="2" ... />
<asp:TextBox TabIndex="1" ... />
```



sequence in which the controls are visited when the TAB key is pressed

Controlli web

Class Button

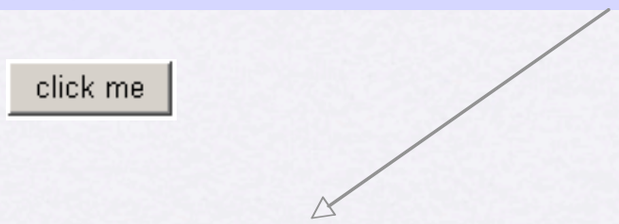
```
public class Button: WebControl {
    //--- properties
    public string Text { get; set; }
    public string CommandName { get; set; }
    public string CommandArgument { get; set; }
    public bool CausesValidation { get; set; }
    //--- events
    public event EventHandler Click;
    public event CommandEventHandler Command;
}
```

caption of the button

for handling *Command* events.

should the validators run when the page is sent to the server?
default = true

```
<asp:Button Text="click me" OnClick="DoClick" Runat="server" />
```




```
public void DoClick (object sender, EventArgs e) {
    ...
}
```

delegate EventHandler

- either in the code behind
- or in <script> tags of the page

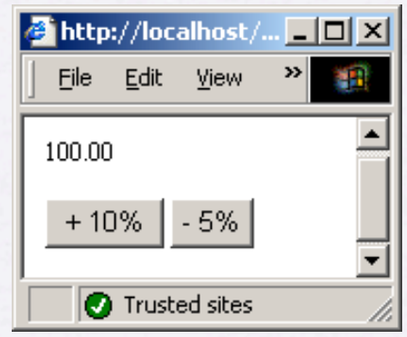
Controlli web

Button *(Command Event)*

Command **Event**

useful if multiple buttons on a page should be handled by the same event handler

```
<form Runat="server">
   <asp:Label ID="label" Text="100.00" Runat="server" />
   <br><br>
   <asp:Button Text="+ 10%"
      CommandName="add" CommandArgument="0.1"
      OnCommand="DoCommand" Runat="server" />
   <asp:Button Text="- 5%"
      CommandName="sub" CommandArgument="0.05"
      OnCommand="DoCommand" Runat="server" />
</form>
```




```
public void DoCommand (object sender, CommandEventArgs e) {
   double total = Convert.ToDouble(label.Text);
   if (e.CommandName == "add")
      total += total * Convert.ToDouble(e.CommandArgument);
   else if (e.CommandName == "sub")
      total -= total * Convert.ToDouble(e.CommandArgument);
   label.Text = total.ToString("f2");
}
```

Controlli web

Class TextBox

```
public class TextBox: WebControl {
   //--- properties
   public virtual string Text { get; set; }
   public virtual TextBoxMode TextMode { get; set; }
   public virtual int MaxLength { get; set; }
   public virtual int Columns {get; set; }
   public virtual int Rows { get; set; }
   public virtual bool Wrap { get; set; }
   public virtual bool ReadOnly { get; set; }
   public virtual bool AutoPostBack { get; set; }
   //--- events
   public event EventHandler TextChanged;
}
```

```
public enum TextBoxMode {
   MultiLine, Password, SingleLine
}
```

true: *TextChanged* causes an immediate round trip

raised when the RETURN key is pressed or when the cursor leaves the TextBox

```
<asp:TextBox Text="sample" Runat="server" />
```

sample

```
<asp:TextBox TextMode="Password" MaxLength="10" Runat="server" />
```

****

```
<asp:TextBox TextMode="MultiLine"
   Rows="2" Columns="15" Wrap="true" Runat="server" />
line 1
line 2
line 3
</asp:TextBox>
```

line 1
line 2

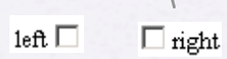
Controlli web

Class CheckBox

```
public class CheckBox: WebControl {
   //--- properties
   public virtual bool Checked { get; set; }
   public virtual string Text { get; set; }
   public virtual TextAlign TextAlign { get; set; }
   public virtual bool AutoPostBack { get; set; }
   //--- events
   public event EventHandler CheckedChanged;
}
```

raised when *Checked* changes

```
public enum TextAlign {
   Left, Right
}
```




```
<form Runat="server">
   <asp:CheckBox ID="apples" Text="Apples" Runat="server" /><br>
   <asp:CheckBox ID="pears" Text="Pears" Runat="server" /><br>
   <asp:CheckBox ID="bananas" Text="Bananas" Runat="server" /><br>
   <asp:Button Text="Buy" OnClick="DoClick" Runat="server" /> <br><br>
   <asp:Label ID="label" Runat="server" />
</form>

void DoClick (object sender, EventArgs e) {
   label.Text = "You bought: ";
   if (apples.Checked) label.Text += "Apples ";
   if (pears.Checked) label.Text += "Pears ";
   if (bananas.Checked) label.Text += "Bananas ";
}
```

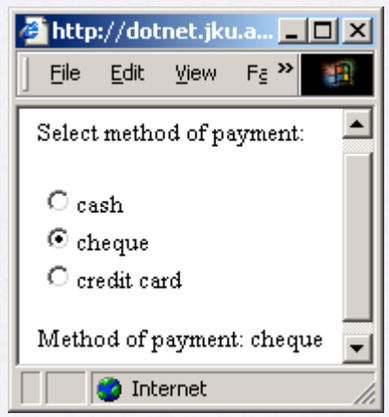
Controlli web

Class RadioButton

```
public class RadioButton: CheckBox {
   public virtual string GroupName { get; set; }
}
```

all radio buttons of the same group must have the same group name

```
<form Runat="server">
   <p>Select method of payment:</p>
   <asp:RadioButton ID="cash" Text="cash" GroupName="payment"
      OnCheckedChanged="RadioChanged" AutoPostBack="true"
      Runat="server" /><br>
   <asp:RadioButton ID="cheque" Text="cheque" GroupName="payment"
      OnCheckedChanged="RadioChanged" AutoPostBack="true"
      Runat="server" /><br>
   <asp:RadioButton ID="card" Text="credit card" GroupName="payment"
      OnCheckedChanged="RadioChanged" AutoPostBack="true"
      Runat="server" /><br><br>
   <asp:Label ID="label" Runat="server" />
</form>

void RadioChanged (object sender, EventArgs e) {
   label.Text = "Method of payment: ";
   if (cash.Checked) label.Text += cash.Text;
   if (cheque.Checked) label.Text += cheque.Text;
   if (card.Checked) label.Text += card.Text;
}
```



Controlli web

Class ListControl

Base class of ListBox, DropDownList, ...

```
public class ListControl: WebControl {
   //--- properties
   public virtual ListItemCollection Items { get; set; }
   public virtual ListItem SelectedItem { get; }
   public virtual int SelectedIndex { get; set; }
   public virtual string DataTextFormatString { get; set; }
   public virtual object DataSource { get; set; }
   public virtual string DataTextField { get; set; }
   public virtual string DataValueField { get; set; }
   public virtual bool AutoPostBack { get; set; }
   //--- events
   public event EventHandler SelectedIndexChanged;
}
```

```
public sealed class ListItem {
   public string Text { get; set; }
   public string Value { get; set; }
   public bool Selected { get; set; }
}
```

-1 or 0, 1, 2, 3, ...

e.g. "width = {0,f2} cm"

raised when a new *ListItem* is selected

DataSource	arbitrary object that implements *ICollection* (*DataView*, *Array*, *ArrayList*, *SortedList*, ...)
DataTextField	for *DataView*: name of the column that contains the text to be displayed
DataValueField	for *DataView*: name of the column that contains the value which corrsponds to the displayed text

Controlli web

Class ListBox

```
public class ListBox: ListControl {
   public virtual int Rows { get; set; }
   public virtual ListSelectionMode SelectionMode { get; set; }
}
```

```
public enum ListSelectionMode {
   Single, Multiple
}
```

statically specified list

```
<form Runat="server">
   <asp:ListBox ID="list" Rows="3" Runat="server" >
      <asp:ListItem Text="United States" Value="USA" Runat="server" />
      <asp:ListItem Text="Great Britain" Value="GB" Runat="server" />
      <asp:ListItem Text="Germany" Value="D" Runat="server" />
      <asp:ListItem Text="France" Value="F" Runat="server" />
      <asp:ListItem Text="Italy" Value="I" Runat="server" />
   </asp:ListBox><br><br>
   <asp:Button OnClick="ButtonClick" Text="Show" Runat="server" /><br>
   <asp:Label ID="lab" Runat="server" />
</form>

void ButtonClick (object sender, EventArgs e) {
   lab.Text = "The selected country has the international car code ";
   if (list.SelectedItem != null) lab.Text += list.SelectedItem.Value;
}
```

Controlli web

ListBox (Dynamically Specified List)

```
<form Runat="server">
   <asp:ListBox ID="list" Rows="3" AutoPostBack="true"
      OnSelectedIndexChanged="Show" Runat="server" /> <br><br>
   <asp:Button Text="Fill" OnClick="Fill" Runat="server" /> <br><br>
   <asp:Label ID="lab" Runat="server" />
</form>

void Fill (object sender, EventArgs e) {
   SortedList data = new SortedList();
   data["United States"] = "USA";
   data["Great Britain"] = "GB";
   data["France"] = "F";
   data["Italy"] = "I";
   list.DataSource = data;
   list.DataTextField = "Key";       // take the text from the Key property of the items
   list.DataValueField = "Value";  // take the value from the Value property of the items
   list.DataBind();
}
void Show (object sender, EventArgs e) {
   lab.Text = "The selected country has the international car code ";
   if (list.SelectedItem != null) lab.Text += list.SelectedItem.Value;
}
```

Controlli web

ListBox (Even Simpler)

If *Text* and *Value* are equal, one can use the followin simple solution

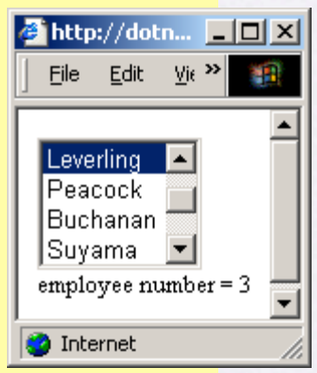
```
<form Runat="server">
   <asp:ListBox ID="list" Rows="3" AutoPostBack="true"
      OnSelectedIndexChanged="Show" Runat="server" /> <br><br>
   <asp:Button Text="Fill" OnClick="Fill" Runat="server" /> <br><br>
   <asp:Label ID="lab" Runat="server" />
</form>

void Fill (object sender, EventArgs e) {
   list.DataSource = new string[] {"D", "F", "GB", "I", "USA"};
   list.DataBind();
}
void Show (object sender, EventArgs e) {
   lab.Text = "The selected country has the international car code ";
   if (list.SelectedItem != null) lab.Text += list.SelectedItem.Value;
}
```

Controlli web

ListBox (List Generated From a Database)

```
<form OnInit="PageInit" Runat="server">
   <asp:ListBox ID="list" DataTextField="LastName" DataValueField="EmployeeID"
      OnSelectedIndexChanged="HandleSelect" AutoPostBack="true" Runat="server" /><br>
   <asp:Label ID="label" Runat="server" />
</form>
```

```
public class BasePage : Page {
   protected ListBox list;
   protected Label label;

   public void PageInit (object sender, EventArgs e) {
      DataSet ds = new DataSet();
      SqlConnection con = new SqlConnection("data source=127.0.0.1\\NETSDK; " +
      "initial catalog=Northwind; user id=sa; password=; Trusted_Connection=true");
      string cmdString = "SELECT * FROM Employees";
      SqlDataAdapter adapter = new SqlDataAdapter(cmdString, con);
      adapter.Fill(ds, "Employees");
      if (ds.HasErrors) ds.RejectChanges(); else ds.AcceptChanges();
      list.DataSource = ds.Tables["Employees"].DefaultView;
      list.DataBind();
   }
   public void HandleSelect (object sender, EventArgs e) {
      label.Text = "employee number = ";
      if (list.SelectedItem != null) label.Text += list.SelectedItem.Value;
   }
}
```



Controlli web

Class DataGrid

```
public class DataGrid: BaseDataList {
    //--- properties
    public virtual object DataSource { get; set; }
    public virtual bool AutoGenerateColumns { get; set; }
    public virtual DataGridColumnCollection Columns {get;}
    public virtual DataGridItemsCollection Items { get; set; }
    public virtual DataGridItem SelectedItem { get; set; }
    public virtual int SelectedIndex { get; set; }
    ...
}
```

```
public class DataGridColumn: ... {
    public virtual string HeaderText { get; set; }
    public virtual string FooterText { get; set; }
    public virtual TableItemStyle HeaderStyle {get;}
    public virtual TableItemStyle FooterStyle {get;}
    ...
}
```

```
public class DataGridItem: ... {
    public virtual TableCellCollection Cells { get; }
    ...
}
```

```
public class TableCell: WebControl {
    public virtual string Text { get; set; }
    public virtual bool Wrap { get; set; }
    ...
}
```

DataGridColumn

EmployeeID	FirstName	LastName
1	Nancy	Davolio
2	Andrew	Fuller
3	Janet	Leverling
4	Margaret	Peacock
5	Steven	Buchanan
6	Michael	Suyama
7	Robert	King
8	Laura	Callahan
9	Anne	Dodsworth

DataGridItem

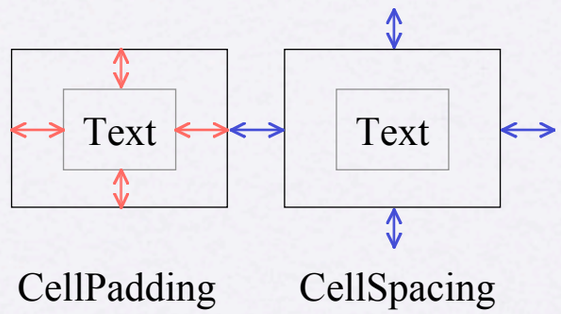
TableCell

Controlli web

DataGrid (Formatting)

```
public class DataGrid: BaseDataList {
   //--- properties
   ...
   public virtual GridLines GridLines { get; set; }
   public virtual int CellPadding { get; set; }
   public virtual int CellSpacing { get; set; }
   public virtual bool ShowHeader { get; set; }
   public virtual bool ShowFooter { get; set; }
   public virtual TableItemStyle AlternatingItemStyle { get; }
   public virtual TableItemStyle HeaderStyle { get; }
   public virtual TableItemStyle FooterStyle { get; }
   public virtual TableItemStyle ItemStyle { get; }
   public virtual TableItemStyle SelectedItemStyle { get; }
   ...
}
```

```
public enum GridLines {
   Both, Horizontal, None, Vertical
}
```




```
public class TableItemStyle: Style {
   public FontInfo Font { get; }
   public Color ForeColor { get; set; }
   public Color BackColor { get; set; }
   public Unit Width { get; set; }
   public Unit Height { get; set; }
   ...
}
```

```
<asp:DataGrid HeaderStyle-Font-Bold="true" Runat="server">
   <ItemStyle ForeColor="Red" Font-Name="Times" />
   <AlternatingItemStyle BackColor="LightGray" />
</asp:DataGrid>
```

Controlli web

DataGrid (Methods and Events)

```
public class DataGrid: BaseDataList {
    ...
    //--- methods
    public override void DataBind();
    ...
    //--- events
    public event DataGridCommandEventHandler ItemCommand;
    public event DataGridCommandEventHandler EditCommand;
    public event DataGridCommandEventHandler CancelCommand;
    public event DataGridCommandEventHandler UpdateCommand;
    public event DataGridCommandEventHandler DeleteCommand;
    public event EventHandler SelectedIndexChanged;
}
```

Events are raised depending on the column kind

BoundColumn

ButtonColumn

EditCommandColumn

ID	First Name	Last Name		
1	Nancy	Davolio	select	edit
2	Andrew	Fuller	select	edit
3	Janet	Leverling	select	edit

Controlli web

DataGrid (Column Kind)

<asp:**BoundColumn ...**

Is automatically bound to a column of the data source

DataField = "*dbColumnName*"

<asp:**ButtonColumn ...**

Every line contains a button which raises an *ItemCommand*

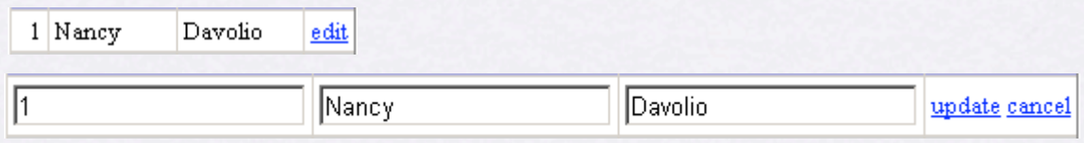
ButtonType = "LinkButton" | "PushButton"
Text = "*buttonLabel*"
CommandName = "Select" | "Delete" | "*anyText*"

CommandName is passed to the *ItemCommand*

<asp:**EditCommandColumn ...**

Every line contains an *edit* button. If it is clicked it is replaced with an *update* and a *cancel* button.

ButtonType = "LinkButton" | "PushButton"
EditText = "*editButtonLabel*"
UpdateText = "*updateButtonLabel*"
CancelText = "*cancelButtonLabel*"



Controlli web

DataGrid (Event Handling)

Kind of the raised event

column kind	*condition*	*raised events*
ButtonColumn	CommandName == "Select" CommandName == "Delete" CommandName == arbitrary	ItemCommand + SelectedIndexChanged ItemCommand + DeleteCommand ItemCommand
EditCommandColumn	click on *edit* button click on *update* button click on *cancel* button	ItemCommand + EditCommand ItemCommand + UpdateCommand ItemCommand + CancelCommand

Event parameter of *ItemCommand*

void HandleItemCommand (object sender, DataGridCommandEventArgs e) {...}

column kind		*e.CommandName*
ButtonColumn		CommandName
EditCommandColumn	Edit-Button UpdateButton CancelButton	"Edit" "Update" "Cancel"

Controlli web

DataGrid (Simple Example)

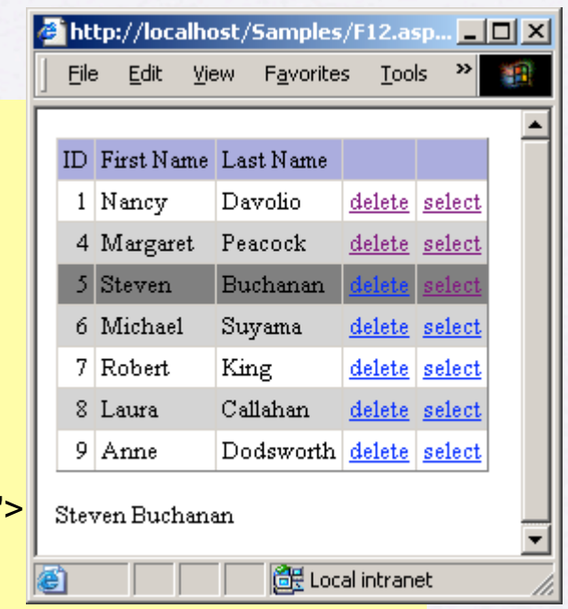
```
<form OnInit="PageInit" Runat="server">
   <asp:DataGrid ID="grid" Runat="server" />
</form>
```

```
public class BasePage : Page {
   protected DataGrid grid;

   public void PageInit (object sender, EventArgs e) {
      DataSet ds = new DataSet();
      SqlConnection con = new SqlConnection("data source=127.0.0.1\\NETSDK; " +
                "[illegible]");
      string sqlString = "SELECT EmployeeID, FirstName, LastName FROM Employees";
      SqlDataAdapter adapter = new SqlDataAdapter(sqlString, con);
      adapter.Fill(ds, "Employees");
      if (ds.HasErrors) ds.RejectChanges(); else ds.AcceptChanges();
      grid.DataSource = ds.Tables["Employees"].DefaultView;
      grid.DataBind();

      grid.HeaderStyle.Font.Bold = true;
      grid.AlternatingItemStyle.BackColor = System.Drawing.Color.LightGray;
   }
}
```

Controlli web

DataGrid (Example With ButtonColumn)

```
<form OnLoad="PageLoad" Runat="server">
   <asp:DataGrid ID="grid" Runat="server"
         AutoGenerateColumns="false"
         CellPadding="3"
         HeaderStyle-BackColor="#aaaadd"
         AlternatingItemStyle-BackColor="LightGray"
         OnDeleteCommand="DeleteRow"
         OnSelectedIndexChanged="SelectRow" >
      <Columns>
         <asp:BoundColumn HeaderText="ID" DataField="EmployeeID">
               <ItemStyle HorizontalAlign="Right" />
         </asp:BoundColumn>
         <asp:BoundColumn HeaderText="First Name" DataField="FirstName" />
         <asp:BoundColumn HeaderText="Last Name" DataField="LastName" />
         <asp:ButtonColumn ButtonType="LinkButton" Text="delete" CommandName="Delete" />
         <asp:ButtonColumn ButtonType="LinkButton" Text="select" CommandName="Select" />
      </Columns>
   </asp:DataGrid><br>
   <asp:Label ID="label" Runat="server" />
</form>
```



Controlli web

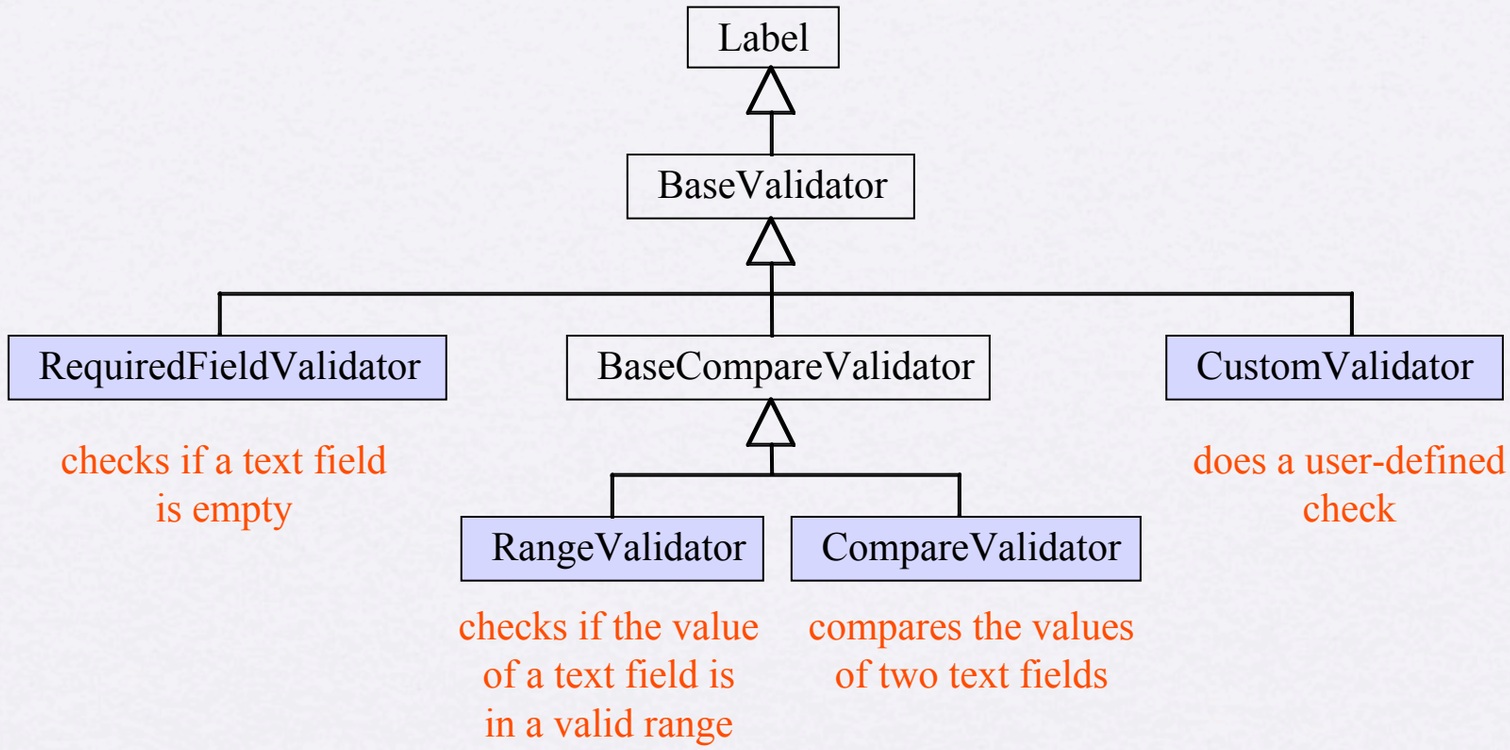
DataGrid (Code Behind for the Previous Example)

```
public class BasePage: Page {
   protected DataGrid grid;
   protected Label label;
   DataView dataView;

   public void PageLoad (object sender, EventArgs e) {
      DataSet ds;
      if (!IsPostBack) {
         ... // load ds from the database
         Session["Data"] = ds;
      } else ds = (DataSet)Session["Data"];
      dataView = ds.Tables["Employees"].DefaultView;
      grid.DataSource = dataView;
      grid.DataBind();
   }

   public void DeleteRow (object sender, DataGridCommandEventArgs e) {
      dataView.Delete(e.Item.DataSetIndex);     // deletes data only in the DataSet
      grid.DataSource = dataView;               // but not in the database
      grid.DataBind();
   }

   public void SelectRow (object sender, EventArgs e) {
      grid.SelectedItemStyle.BackColor = System.Drawing.Color.Gray;
      label.Text = grid.SelectedItem.Cells[1].Text + " " + grid.SelectedItem.Cells[2].Text;
   }
}
```

Validatori

Validators

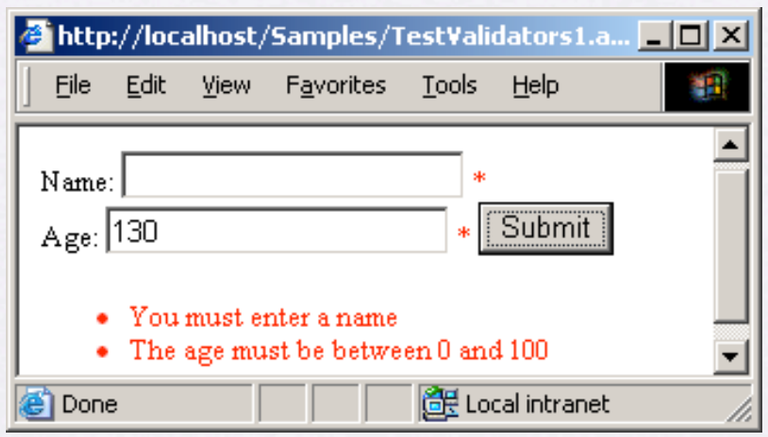
Objects for plausibility checks



Validatori

Validators (Example)

```
Name:
<asp:TextBox ID="name" Runat="server" />
<asp:RequiredFieldValidator ControlToValidate="name" Text="*"
    ErrorMessage="You must enter a name" Runat="server" />
<br>
Age:
<asp:TextBox ID="age" Runat="server" />
<asp:RangeValidator ControlToValidate="age" Text="*"
    MinimumValue="0" MaximumValue="100" Type="Integer"
    ErrorMessage="The age must be between 0 and 100" Runat="server" />
<asp:Button Text="Submit" OnClick="DoClick" Runat="server" />
<asp:ValidationSummary Runat="server" />
```



Lo Stato

3 Kinds of States

Page state

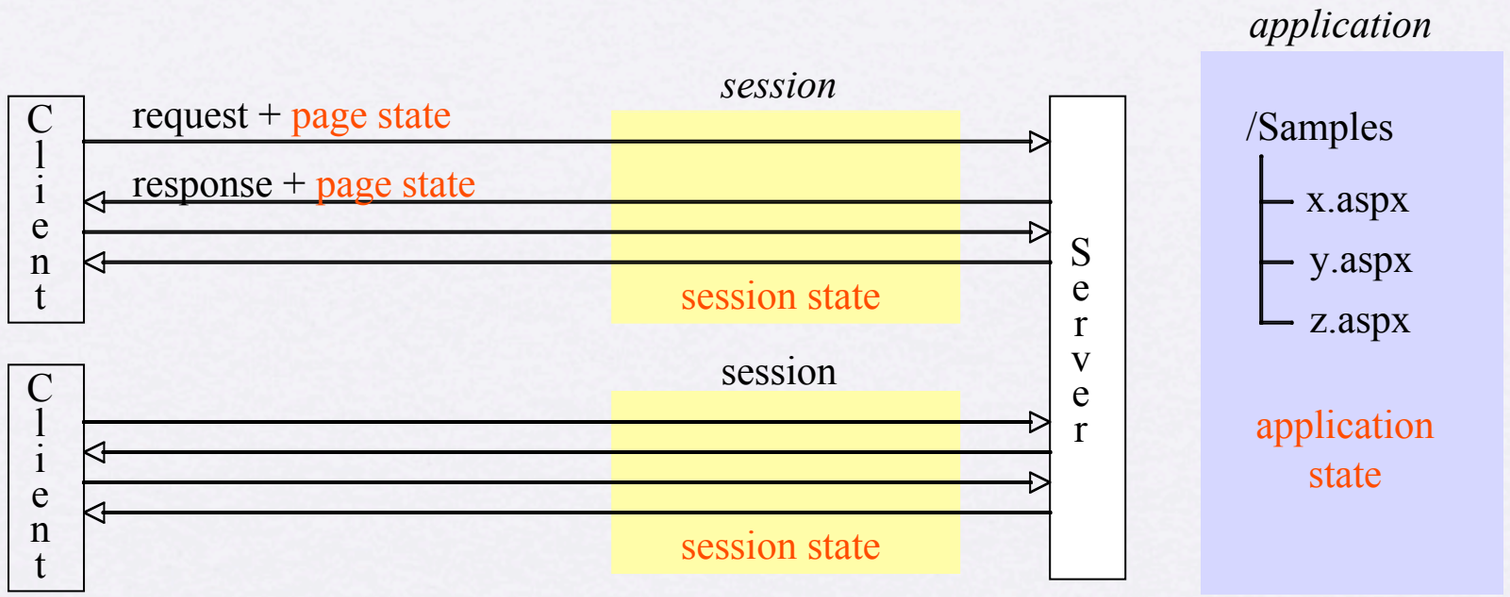
e.g. contents of TextBoxes, state of CheckBoxes, ...

Session state (**session** = all requests from the same client within a certain time)

e.g. shopping cart, email address of a client, ...

Application state (**Application** = all aspx files in the same virtual directory)

e.g. configuration data, number of sessions, ...



Lo stato

How to Access State Information

Page state

writing: `ViewState["counter"] = counterVal;`
reading: `int counterVal = (int) ViewState["counter"];`

Session state

writing: `Session["cart"] = shoppingCart;`
reading: `DataTable shoppingCart = (DataTable) Session["cart"];`

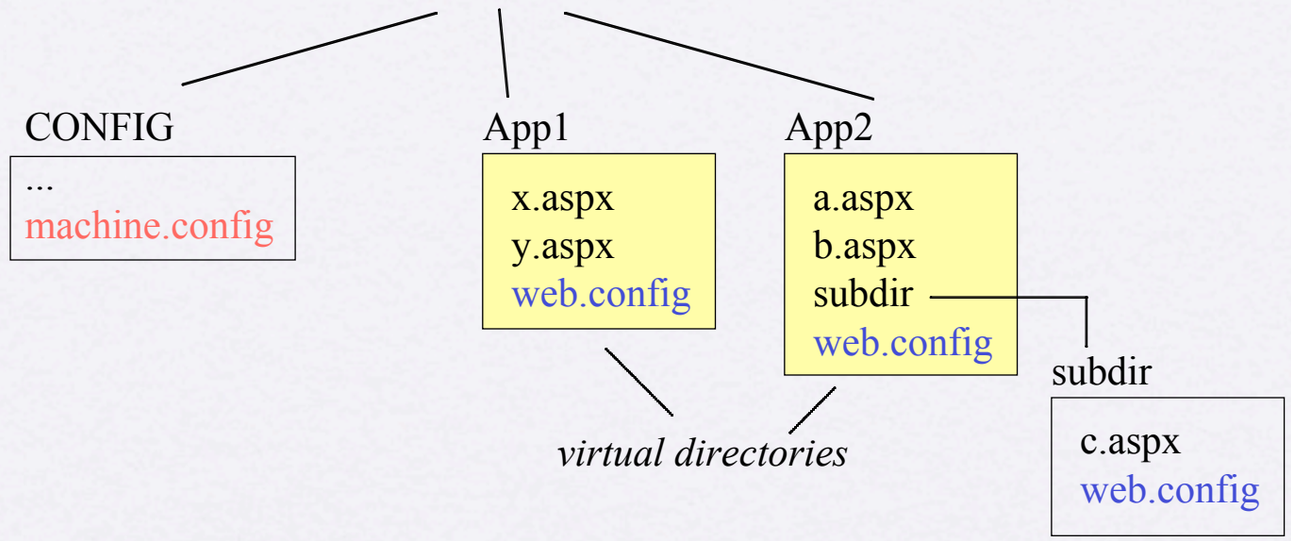
Application state

writing: `Application["database"] = databaseName;`
reading: `string databaseName = (string) Application["databaseName"];`

ViewState, *Session* and *Application* are properties of the *Page* class

l'ambiente

machine.config and web.config



machine.config

- global configuration file
- stored in the .NET Framework directory

web.config

- specific configuration file
- stored in a virtual directory or in its subdirectories
- overwrites configurations from *machine.config* or from other configuration files further up the hierarchy

Configuration files are written in XML

l'ambiente

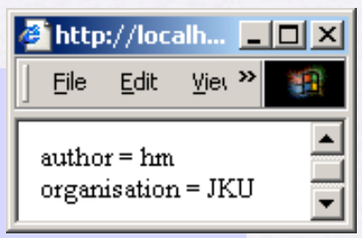
Example: Application Parameters

web.config

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <appSettings>
      <add key="author" value="hm" />
      <add key="organisation" value="JKU" />
   </appSettings>
   ...
</configuration>
```

Can be accessed in ASP.NET pages

```
<%@Page Language="C#" %>
<%@ Import Namespace="System.Configuration" %>
<html>
   <body>
      <%= "author = " + ConfigurationSettings.AppSettings["author"] %><br>
      <%= "organisation = " + ConfigurationSettings.AppSettings["organisation"] %><br>
    </body>
</html>
```

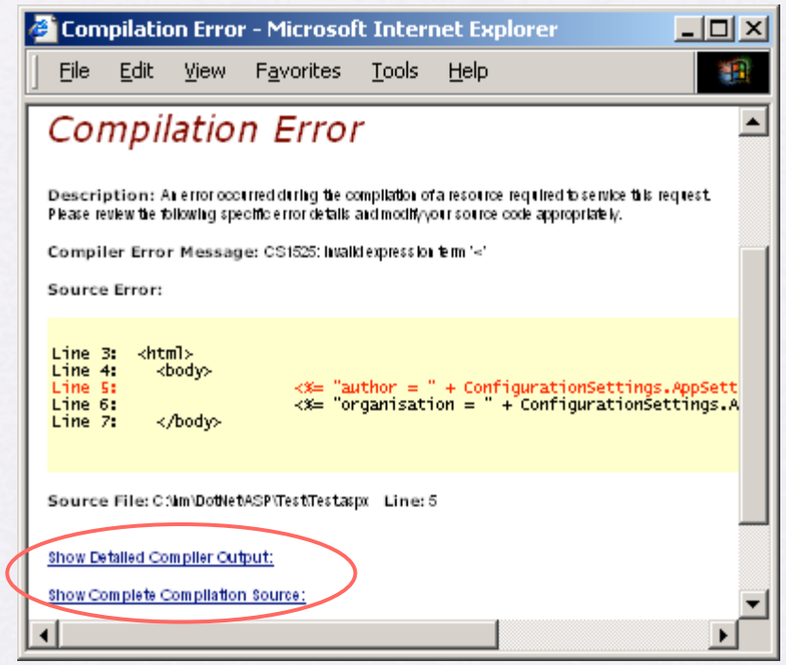


l'ambiente

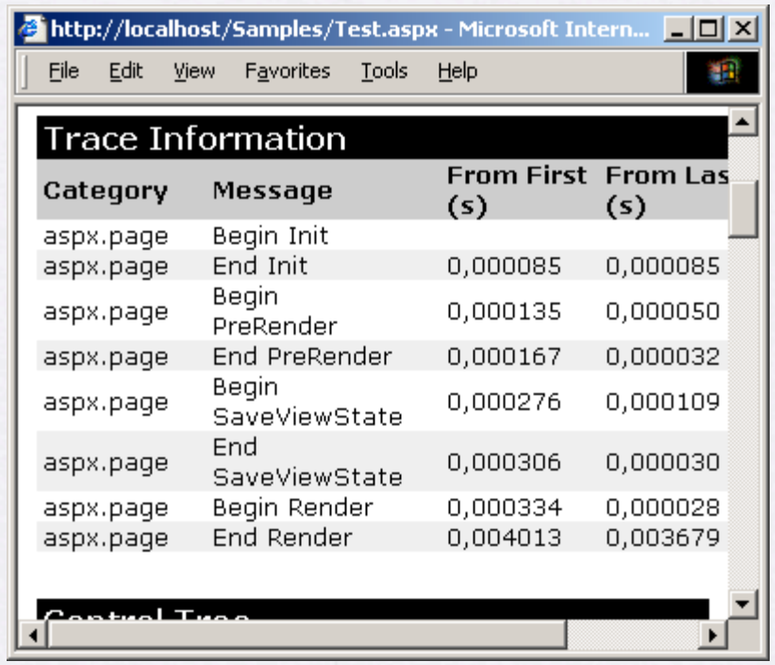
Example: Tracing

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <system.web>
      <trace enabled="true" pageOutput="true" />
      ...
   </system.web>
   ...
</configuration>
```

Gives detailed error diagnostics



Shows a trace if the page is correct



l'ambiente

Authorization

Who may visit the pages of a specific directory?

The directory must have a *web.config* file with the following contents

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <system.web>
      <authorization>
         <allow users="admin" />
         <deny users="?" />
      </authorization>
      ...
   </system.web>
   ...
</configuration>
```

users="*user1*, *user2*, ...**"**

*	all users
?	anonymous users
name	users who have authenticated themselves with this name

machine.config contains
`<allow users="*" />`
This is default if no <allow ...> is specified

l'ambiente

Authentication

4 kinds

None	No authentication. All users are anonymous.
Windows	Uses the login name and the password of the Windows login. Makes only sense for local servers.
Passport	Users are redirected to the login page of the Passport server where they can authenticate themselves (with their user name and password).
Forms	Users are authenticated by a custom login page.

Tool

- uno solo:
  - Microsoft Visual Studio .NET

Link

- http://dotnet.jku.at
- http://www.microsoft.com
- www.asp.net