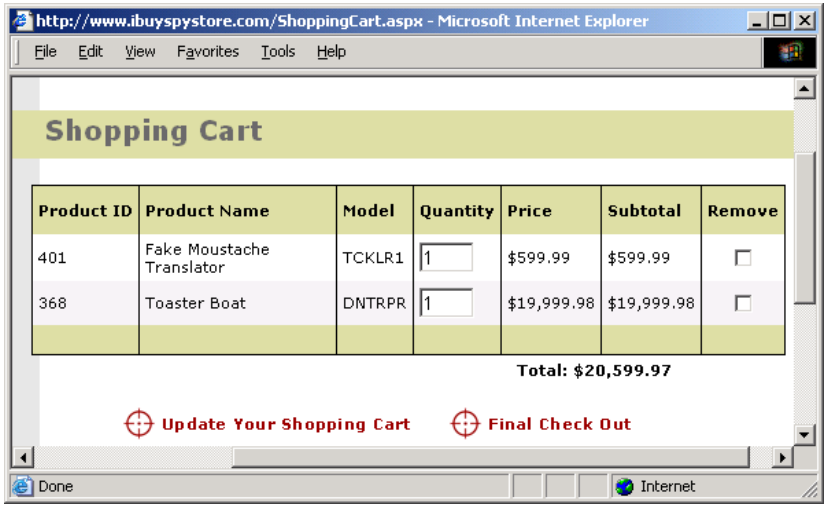
Dynamic Web Pages with ASP.NET

Prof. Dr. Hanspeter Mössenböck
Institute for System Software
Johannes Kepler University Linz

Main Characteristics of ASP.NET

Successor of *Active Server Pages* (ASP),
but with a completely different architecture:



object-oriented
event-based
allows rapid application development (RAD)
rich library of GUI elements (web controls)
users can define their own GUI elements
separation of layout (HTML) and logic (C#)
efficient (compiled server scripts)
automatic state management
authorisation / authentication
...

ASP.NET

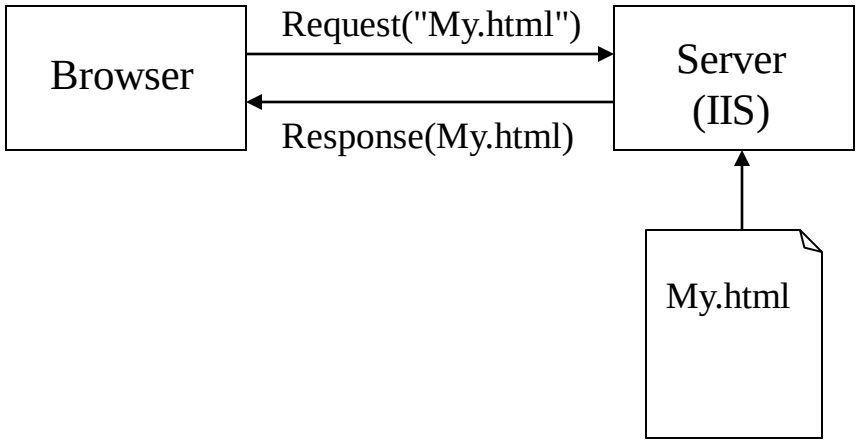
- Simple Dynamic Web Pages
- Web Forms
- Event Handling
- Web Controls
- Validators
- User Controls
- Custom Controls
- State Management
- Configuration of ASP.NET
- Working with Visual Studio .NET

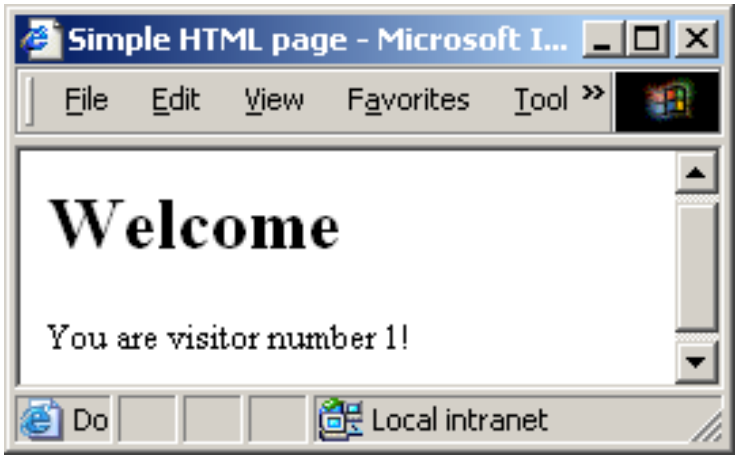
Static Web Pages

Pure HTML

My.html

```
<html>
    <head>
        <title>Simple HTML page</title>
    </head>
    <body>
        <h1>Welcome</h1>
        You are visitor number 1!
    </body>
</html>
```





Dynamic ASPX Pages

Computed values can be inserted into HTML code

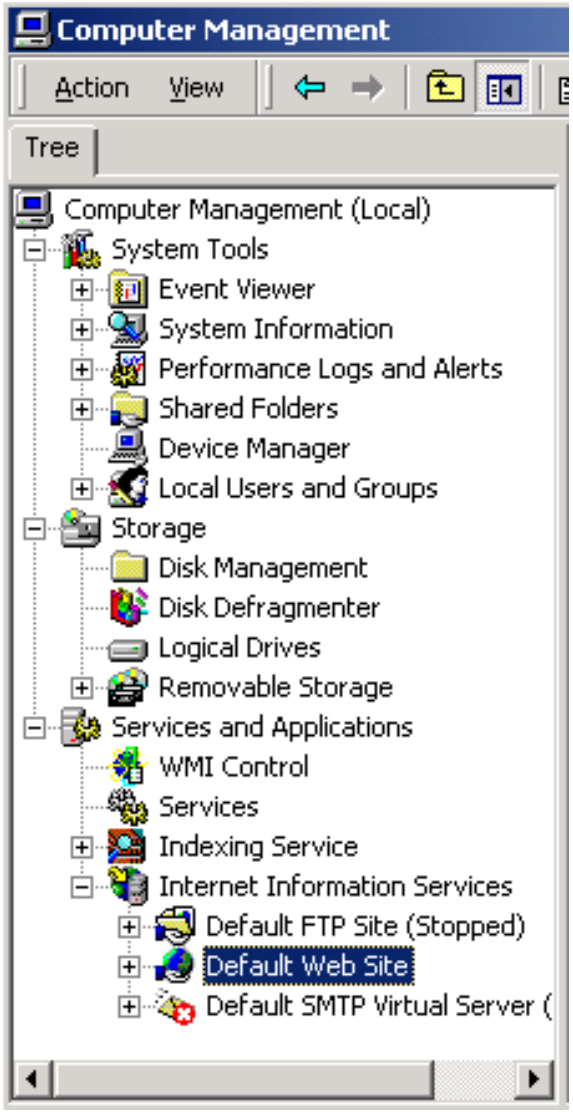
Counter.aspx

```
<%@ 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 must be in a virtual directory.

How to Create a Virtual Directory

(under Windows 2000)



Steps for creating a virtual directory

Control Panel
> Administrative Tools
> Computer Management

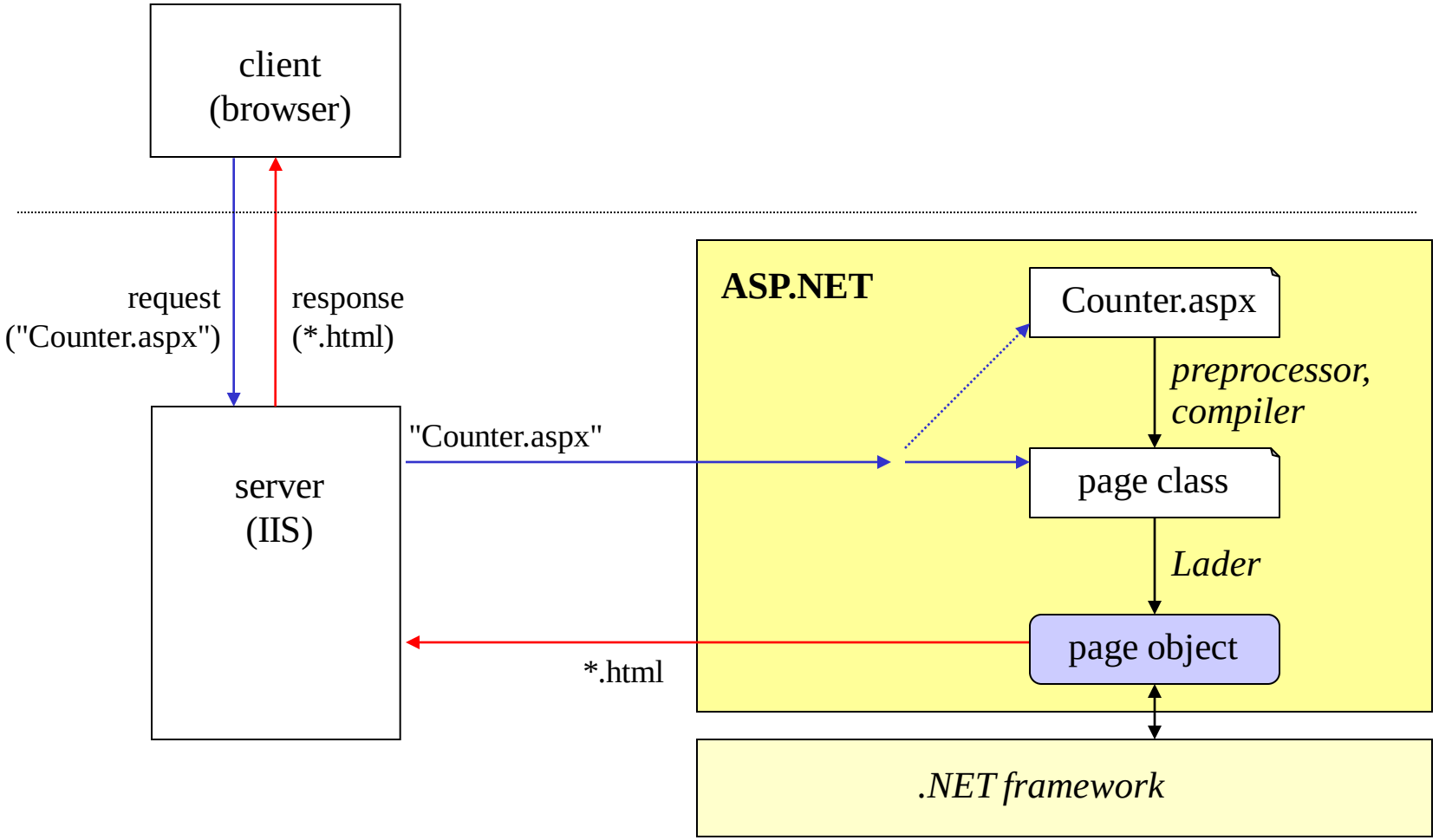
right-click on *Default Web Site*
> New ... Virtual Directory

follow the dialog

All aspx files must be in a virtual directory

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

What Happens Behind the Scene?



HTML Code Returned by the Server

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(...);
      ...
      Response.Write(n);
      %> !
   </body>
</html>
```

Returned HTML code

```
<html>
   <head><title>Page counter</title></head>
   <body>
      <h1>Welcome</h1>
      You are visitor number 6 !
   </body>
</html>
```

- does not contain any script code
- any browser can display this HTML

Code in Script Tags

Counter.aspx

```
<%@ 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>
```

short form for
Response.Write(CounterValue());

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>
```

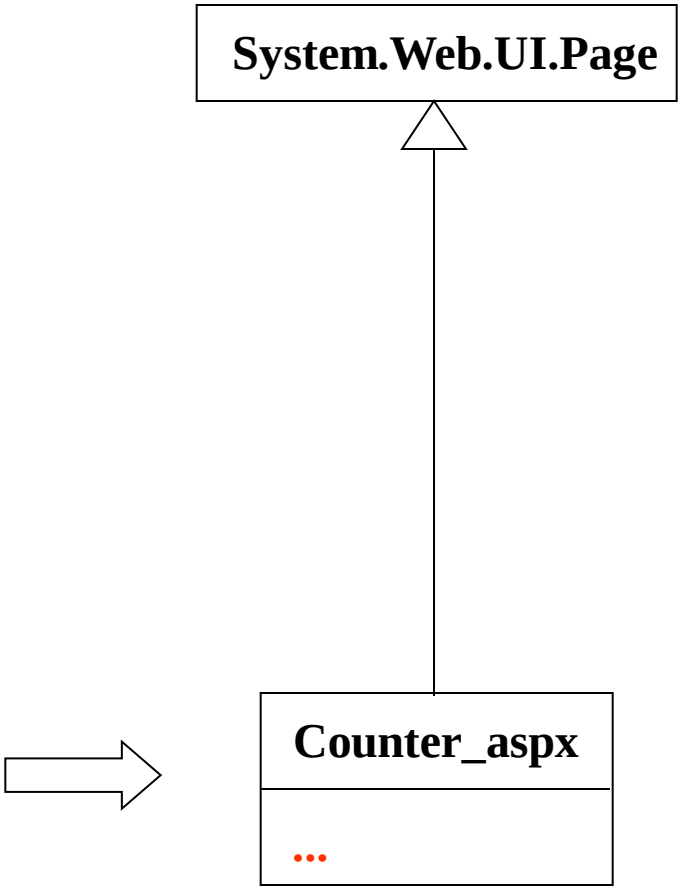
CounterPage.cs

```
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;
    }
}
```

Generated Page Class

aspx page *Counter.aspx*

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



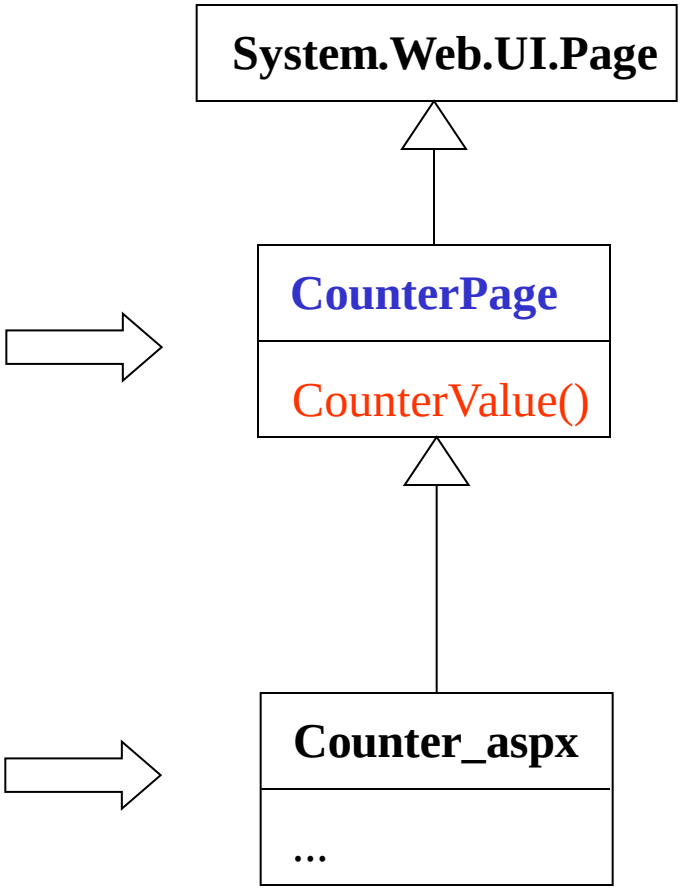
Generated Page Class

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>
```



Generated Class 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();
        }
    ...
    }
}
```

ASP.NET

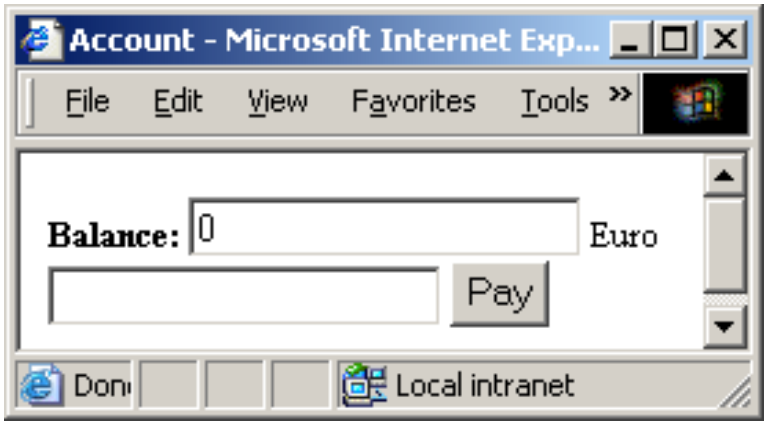
- Simple Dynamic Web Pages
- Web Forms
- Event Handling
- Web Controls
- Validators
- User Controls
- Custom Controls
- State Management
- Configuration of ASP.NET
- Working with Visual Studio .NET

HTML Forms (With CGI Scripts)

```
...
<body>
    <form action="http://www.fake.com/cgi-bin/myprog" 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>
```

CGI script *myprog*

- reads *total* and *amount*
- sends back a new HTML text in which *total* and *amount* have new values



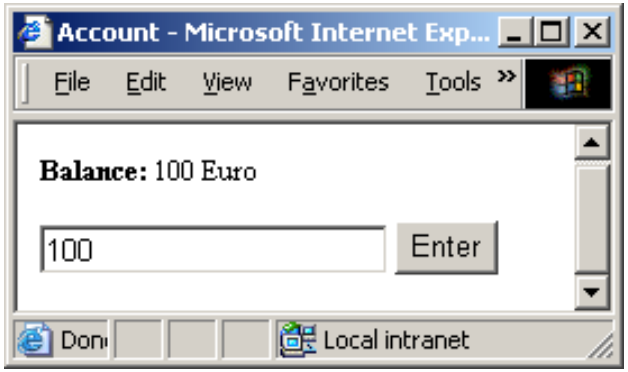
Problems

- CGI programming is tedious
- restricted to HTML elements
- must manage the state of text fields manually when the page is sent back

Web Forms in ASP.NET

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" Runat="server" />
        </form>
    </body>
</html>
```



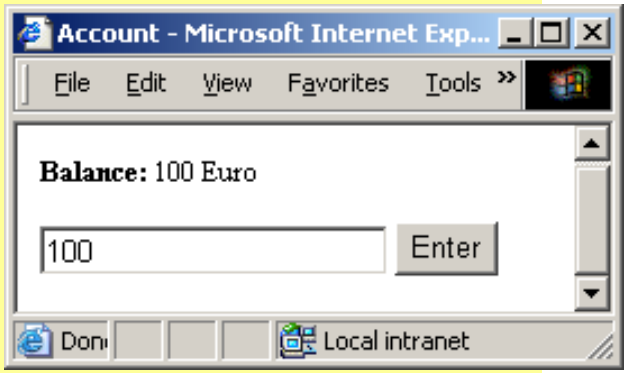
Web Forms in ASP.NET

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>
```

Adder.aspx.cs

```
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 Code Sent Back by the Server

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 sent back

```
<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>
```

General Notation for Web Controls


```
<asp:ClassName PropertyName="value" ... Runat="server" />
```

Example

```
<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 {...} }
   ...
}
```

All web control classes are in the namespace System.Web.UI

Alternative Notation

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

Advantages of Web Forms

- The page is an object
  one can access all its properties and methods:
  page.IsPostBack, *page.User*, *page.FindControl*(), ...

- All GUI elements are objects
  one can access all their properties and methods:
  amount.Text, *amount.Font*, *amount.Width*, ...

- One can implement custom GUI elements

- Web pages can access the whole .NET library
  databases, XML, RMI, ...

- The state of all GUI elements is retained
  amount.Text does not need to be set before the page is sent back

Web Controls (selection)

Label **abc**

TextBox

Button Click

RadioButton Radio

CheckBox Check

DropDownList Linz

ListBox
apples
pears
bananas

Calendar

DataGrid

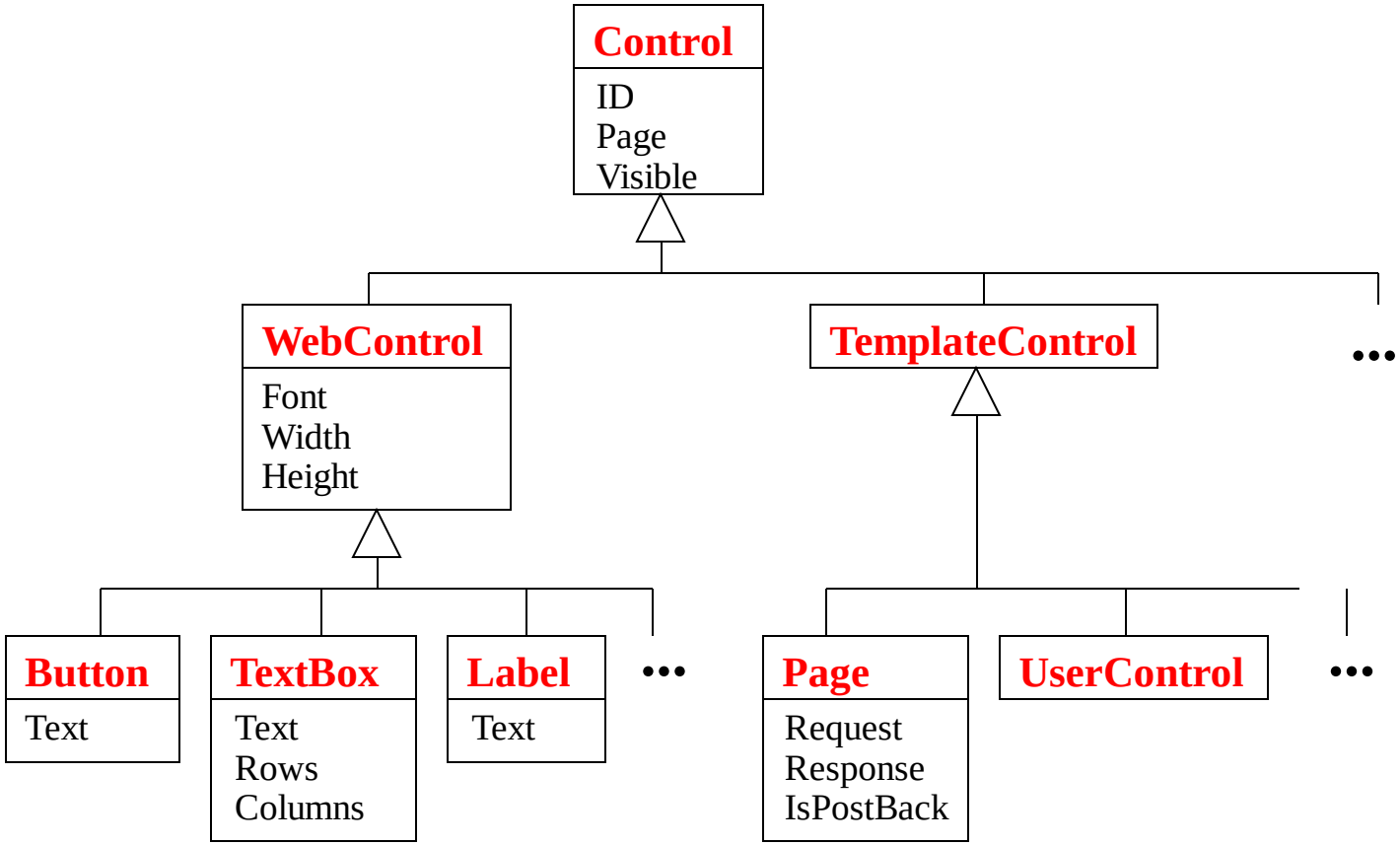
EmployeeID	FirstName	LastName
1	Nancy	Davolio
2	Andrew	Fuller
3	Janet	Leverling
4	Margaret	Peacock
5	Steven	Buchanan
6	Michael	Suyama
7	Robert	King
8	Laura	Callahan
9	Anne	Dodsworth

...

User Controls

Custom Controls

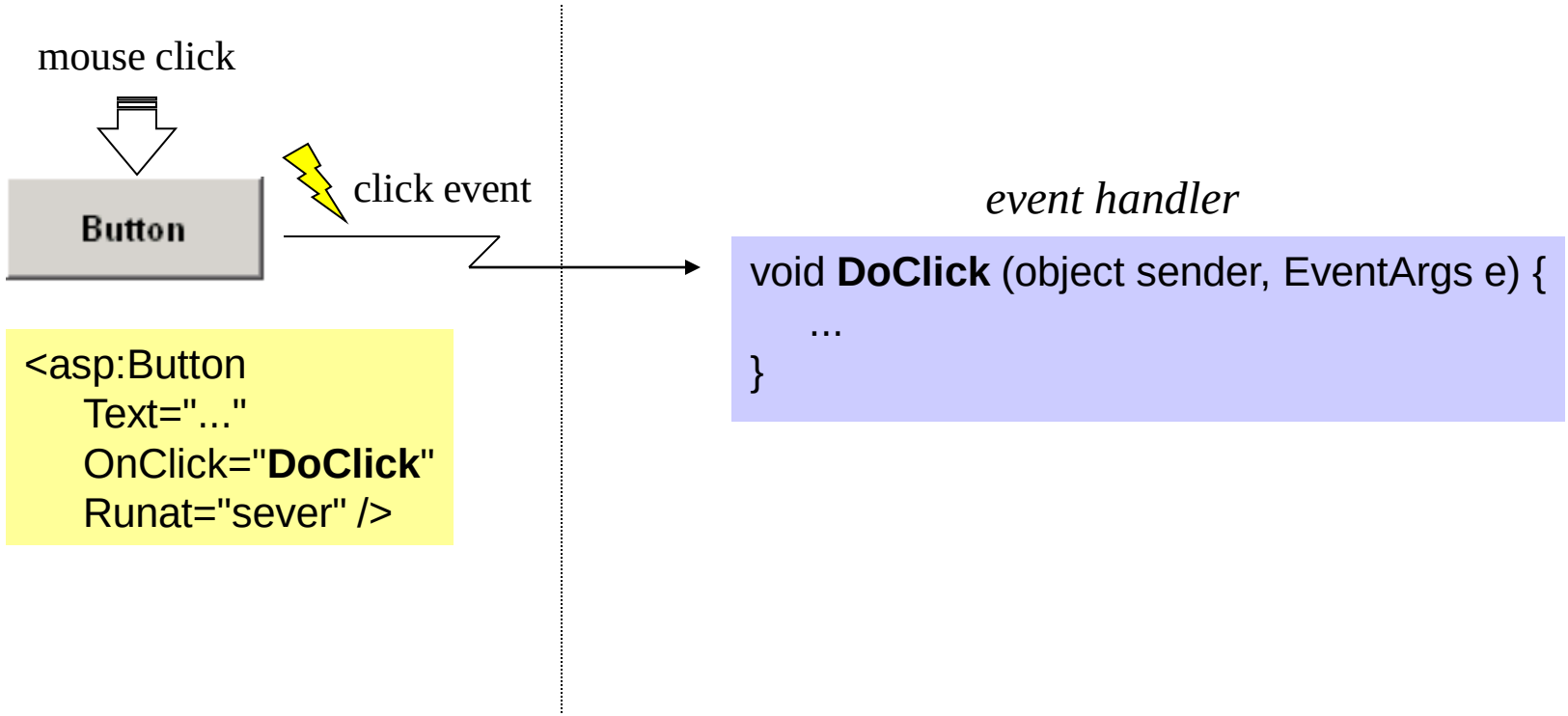
Web Control Hierarchy



ASP.NET

- Simple Dynamic Web Pages
- Web Forms
- Event Handling
- Web Controls
- Validators
- User Controls
- Custom Controls
- State Management
- Configuration of ASP.NET
- Working with Visual Studio .NET

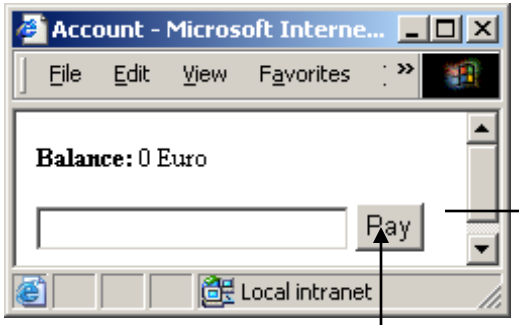
Event-based Processing



Kinds of Events

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

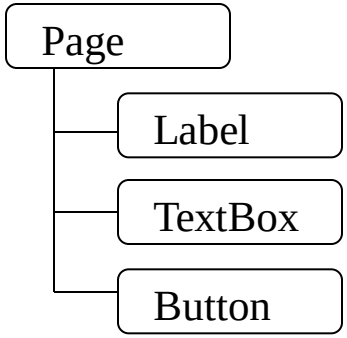
Round Trip of a Web Page



Click

round trip event

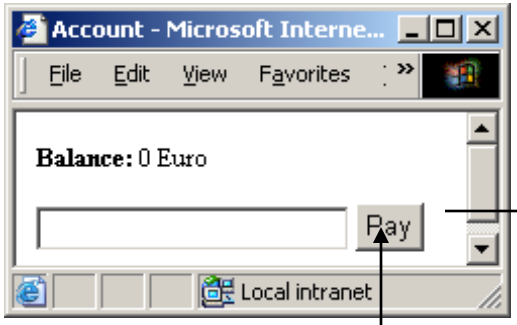
+ page state



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

Client *Server*

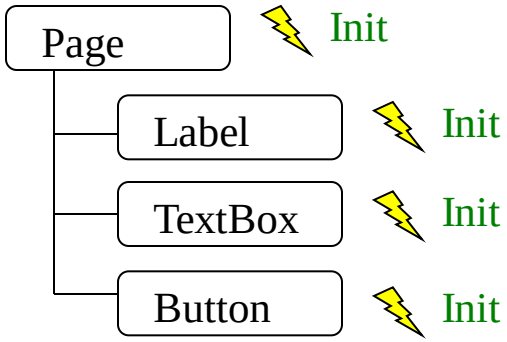
Round Trip of a Web Page



round trip event

+ page state

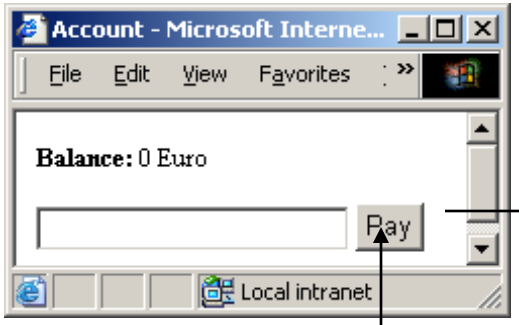
Click



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

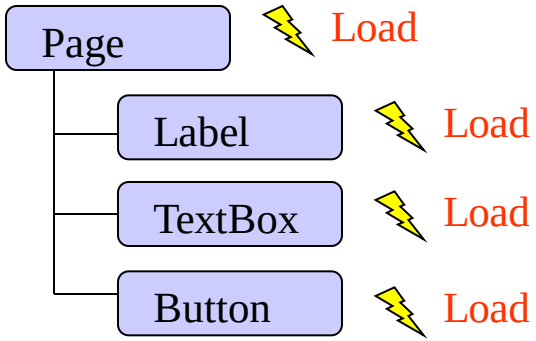
Client *Server*

Round Trip of a Web Page



round trip event

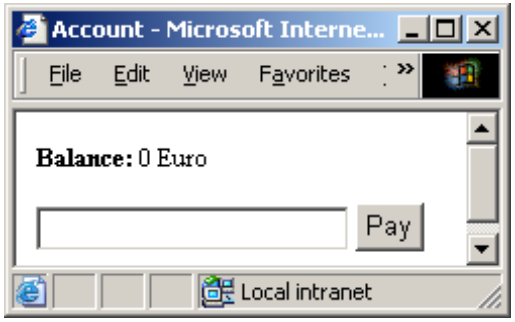
+ page state

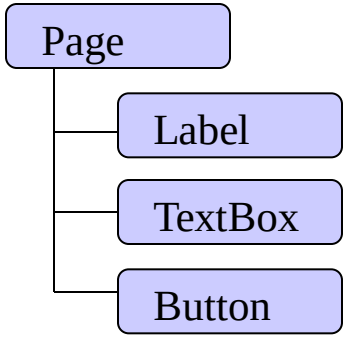


3. **Loading**
   - load controls with the values that the user has entered (page state)
   - raise *Load* events

Client *Server*

Round Trip of a Web Page



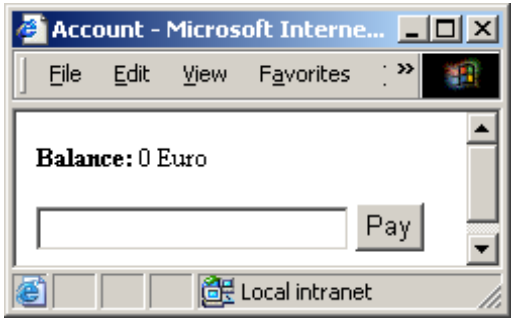


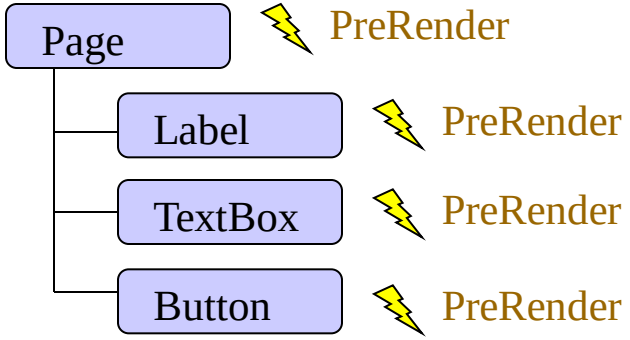
4. **Action**
   handle event(s)
   (Click, TextChanged, ...)

Client

Server

Round Trip of a Web Page





HTML

+ page state

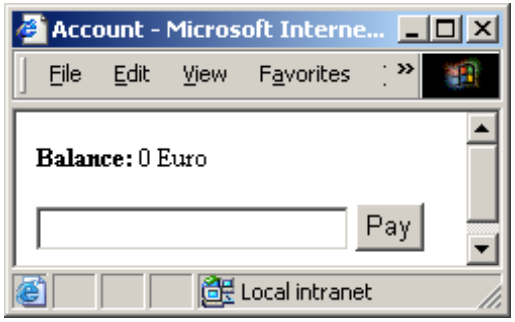
```
<html>
   ...
   <input type="text" ...>
   <input type="button" ...>
   ...
</html>
```

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

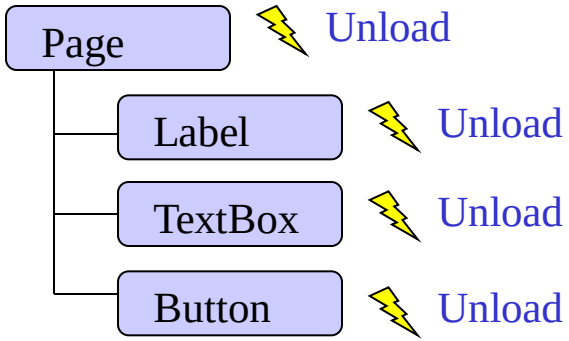
Client

Server

Round Trip of a Web Page




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



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

Client

Server

Which Events Cause a 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" />
```

ASP.NET

- Simple Dynamic Web Pages
- Web Forms
- Event Handling
- Web Controls
- Validators
- User Controls
- Custom Controls
- State Management
- Configuration of ASP.NET
- Working with Visual Studio .NET

Web Control Hierarchy

- Control
  - WebControl
    - Button
    - TextBox
    - Label
      - BaseValidator
        - ...
    - CheckBox
      - RadioButton
    - ListControl
      - ListBox
      - DropDownList
    - Image
      - ImageButton
    - Calendar
    - ValidationSummary
    - ...
  - TemplateControl
    - Page
    - UserControl

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

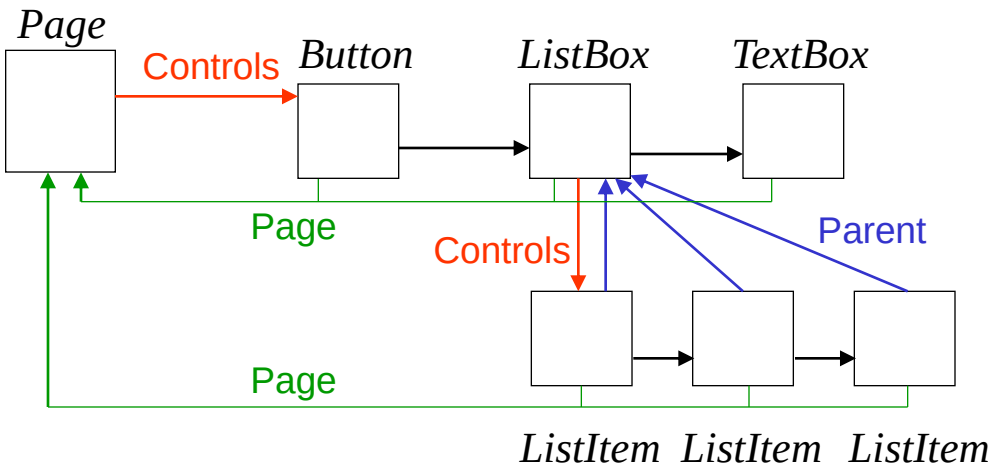
- does the control have nested controls?
- searches for a nested control with the name id
- loads data from a data source
- loads the state from the request stream
- saves the state to the response stream
- renders the control to HTML

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

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)

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 B, int G);
}
```

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;
```

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" ... />
```

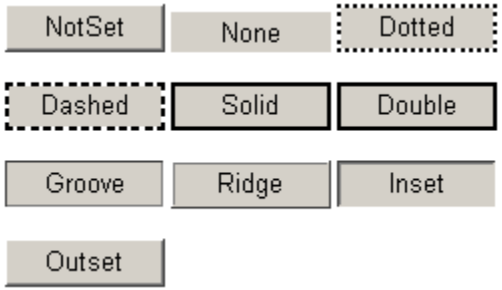
setting the font in the script code:

```
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;
```

WebControl (Other Properties)

BorderStyle

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



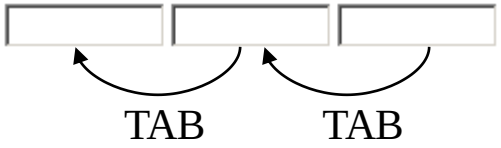
Enabled

```
<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

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

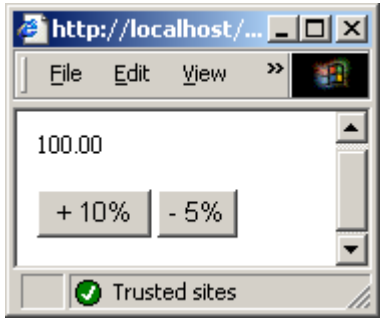
- either in the code behind
- or in <script> tags of the page

Button *(Command Event)*

Command Event

useful is 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");
}
```

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" />
```


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

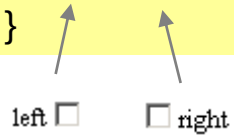
```
<asp:TextBox TextMode="MultiLine"
    Rows="2" Columns="15" Wrap="true" Runat="server" />
line 1
line 2
line 3
</asp:TextBox>
```

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
}
```



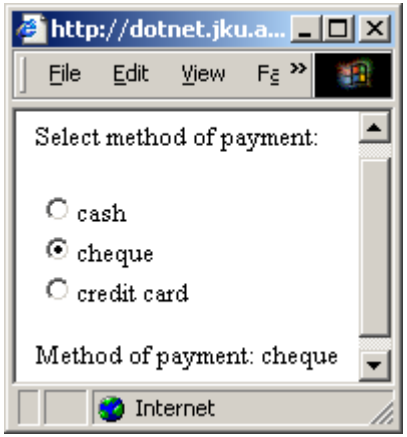

```
<form Runat="server">
    <asp:CheckBox ID="apples" Text="Apples" Runat="server" /><br>
    <asp:CheckBox ID="pears" Text="Pears" Runat="server" /><br>
    <asp:CheckBox ID="bananas" Text="Bananas" Runat="server" /><br>
    <asp:Button Text="Buy" OnClick="DoClick" Runat="server" /> <br><br>
    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

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;
}
```



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

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;
}
```

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;
}
```

ListBox (Even Simpler)

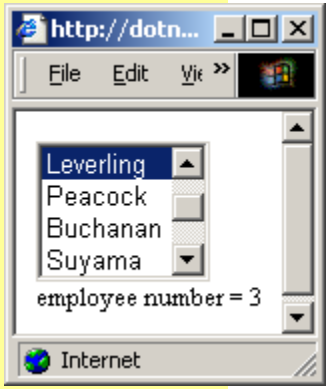
If *Text* and *Value* are equal, one can use the followin simple solution

```
<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;
}
```

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;
   }
}
```



Class DropDownList

```
public class DropDownList: ListControl {
   // same interface as ListControl
}
```

statically specified DropDownList

```
<form Runat="server">
   <asp:DropDownList ID="list" OnSelectedIndexChanged="HandleSelect"
      AutoPostBack="true" Runat="server" >
      <asp:ListItem Text="United States" Value="USA" />
      <asp:ListItem Text="Great Britain" Value="GB" />
      <asp:ListItem Text="Germany" Value="D" />
      <asp:ListItem Text="France" Value="F" />
      <asp:ListItem Text="Italy" Value="I" />
   </asp:DropDownList><br>
   <asp:Label ID="lab" Runat="server" />
</form>

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

DropDownList can also be filled dynamically (like *ListBox*)

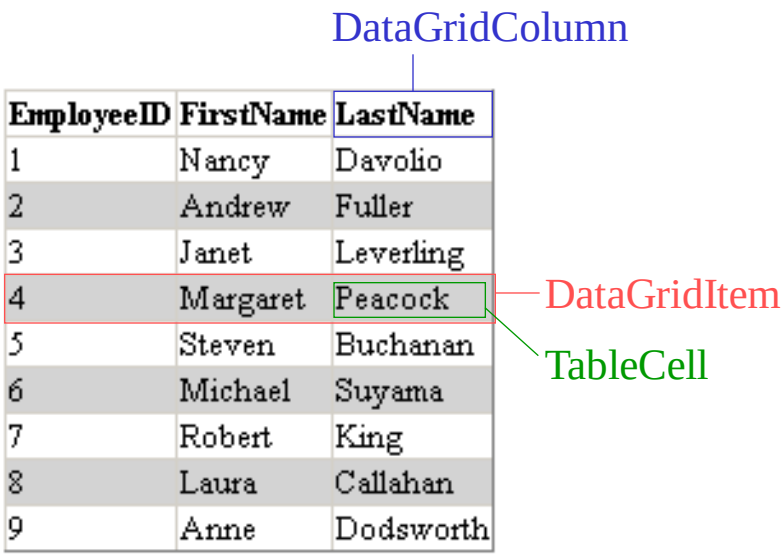
Class DataGrid

```
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; }
    ...
}
```

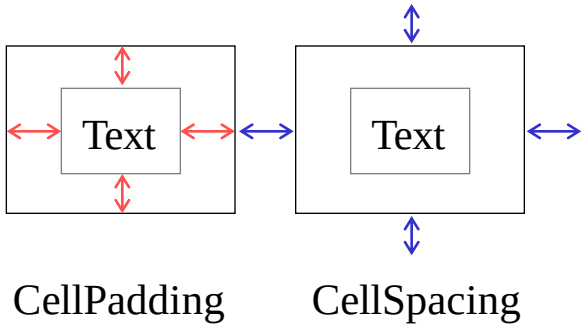
```
public class TableCell: WebControl {
    public virtual string Text { get; set; }
    public virtual bool Wrap { get; set; }
    ...
}
```



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
}
```



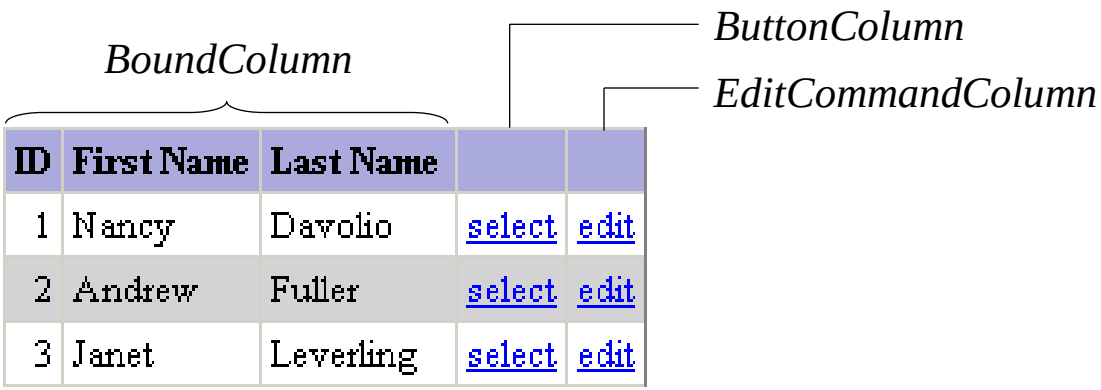

```
public class TableItemStyle: Style {
    public FontInfo Font { get; }
    public Color ForeColor { get; set; }
    public Color BackColor { get; set; }
    public Unit Width { get; set; }
    public Unit Height { get; set; }
    ...
}
```

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

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



DataGrid (Column Kind)

BoundColumn Is automatically bound to a columns of the data source

DataTextField = "*dbColumnName*"

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*

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*"

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*

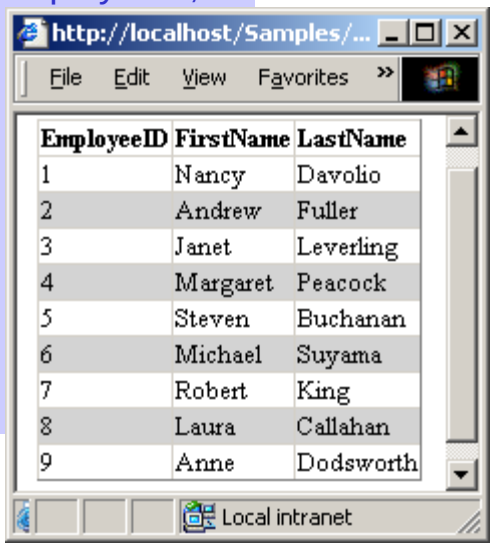
```
void HandleItemCommand (object sender, DataGridCommandEventArgs e) {...}
```

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

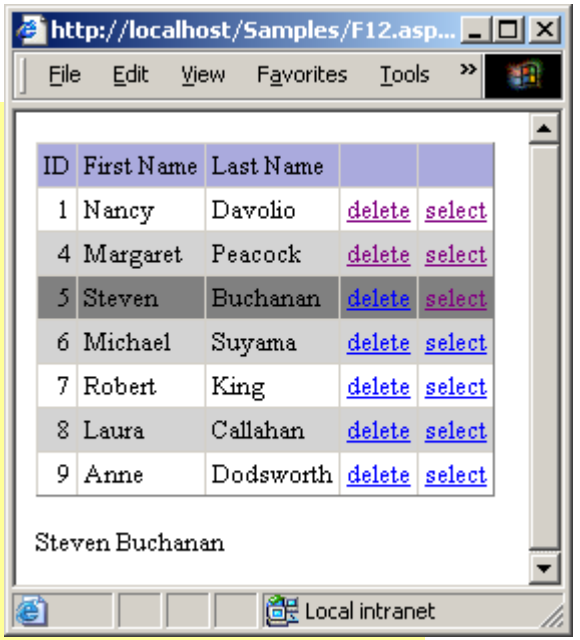
DataGrid (Simple Example)

```
<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; " +
            "initial catalog=Northwind; user id=sa; password=; Trusted_Connection=true");
      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;
   }
}
```



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>
```

DataGrid (Code Behind for the Previous Example)

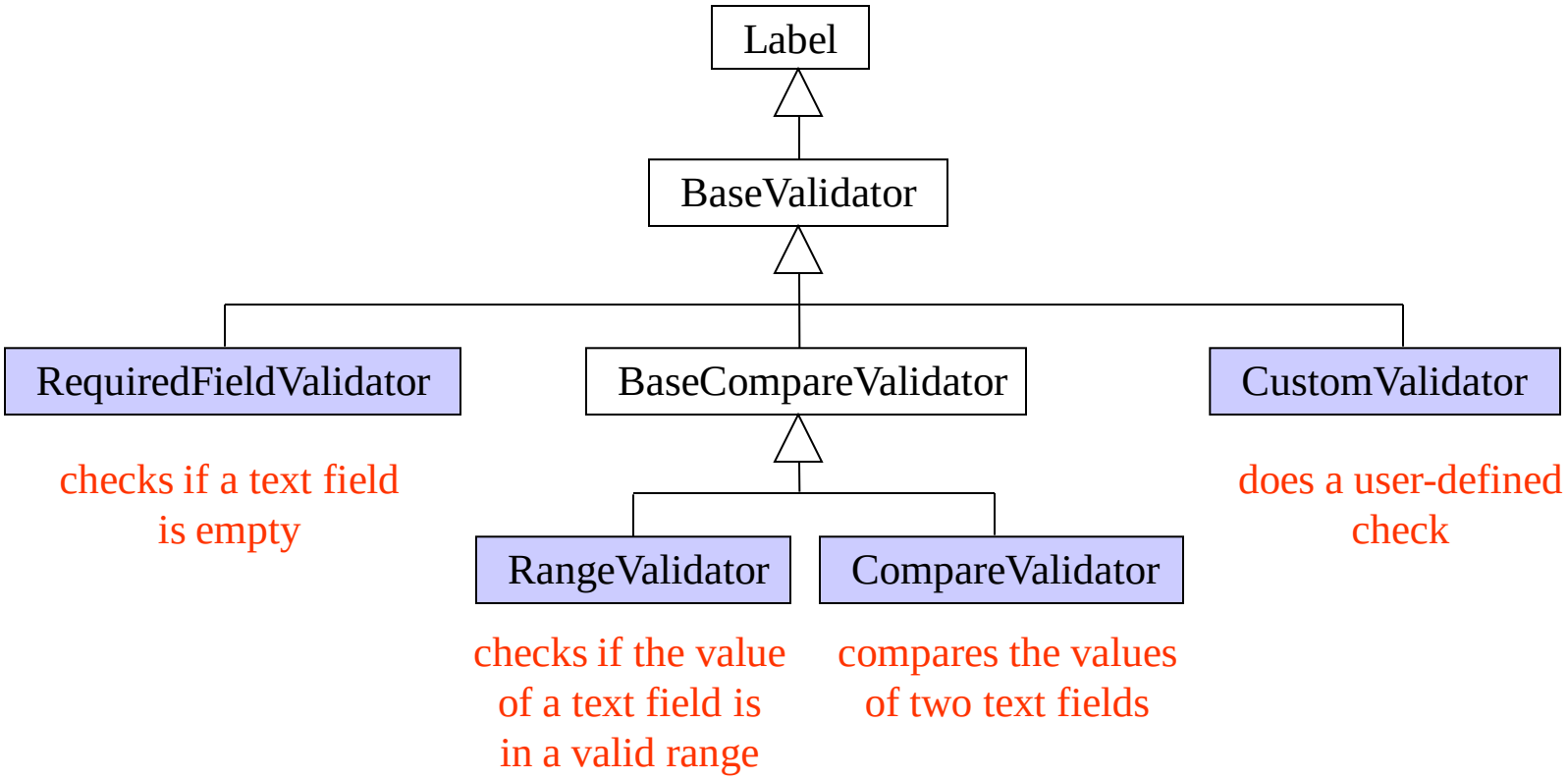

```
public class BasePage: Page {
    protected DataGrid grid;
    protected Label label;
    DataView dataView;

    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;
    }
}
```

ASP.NET

- Simple Dynamic Web Pages
- Web Forms
- Event Handling
- Web Controls
- Validators
- User Controls
- Custom Controls
- State Management
- Configuration of ASP.NET
- Working with Visual Studio .NET

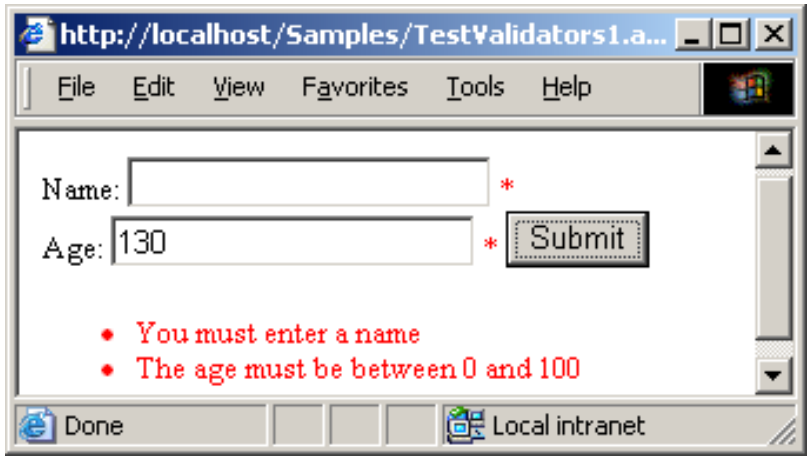
Validators

Objects for plausibility checks



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" />
```



ASP.NET

- Simple Dynamic Web Pages
- Web Forms
- Event Handling
- Web Controls
- Validators
- User Controls
- Custom Controls
- State Management
- Configuration of ASP.NET
- Working with Visual Studio .NET

User Controls (Example)

Group of controls that can be used like a *single* control

Described in an ascx file (e.g. MoneyField.ascx)

```
<%@ Control Inherits="MoneyFieldBase" Src="MoneyField.ascx.cs" %>
<asp:TextBox ID="amount" Runat="server" />
<asp:DropDownList ID="currency" AutoPostBack="true"
                  OnSelectedIndexChanged="Select" Runat="server">
                  <asp:ListItem Text="Euro" Value="1.0" Selected="true" />
                  <asp:ListItem Text="Dollars" Value="0.88" />
                  <asp:ListItem Text="Fracs" Value="1.47" />
                  <asp:ListItem Text="Pounds" Value="0.62" />
</asp:DropDownList>
```

User Controls (Code Behind)

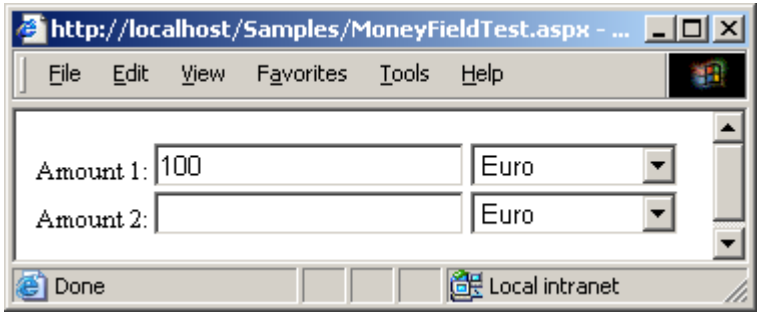
MoneyField.ascx.cs

```
using System;  using System.Web.UI;  using System.Web.UI.WebControls;

public class MoneyFieldBase : UserControl {
    protected TextBox amount;
    protected DropDownList currency;

    public string Text {
        get { return amount.Text; }
        set { amount.Text = value; }
    }
    public double OldFactor {
        get { return ViewState["factor"] == null ? 1 : (double)ViewState["factor"]; }
        set { ViewState["factor"] = value; }
    }
    public void Select (object sender, EventArgs arg) {
        try {
            double val = Convert.ToDouble(amount.Text);
            double newFactor = Convert.ToDouble(currency.SelectedItem.Value);
            double newVal = val / OldFactor * newFactor;
            amount.Text = newVal.ToString("f2");
            OldFactor = newFactor;
        } catch (Exception) {
            amount.Text = "0";
        }
    }
}
```

User Controls (Usage)

Multiple instances of them can be used on the same page

```
<%@ Page Language="C#" %>
<%@ Register TagPrefix="my" TagName="MoneyField" Src="MoneyField.ascx" %>
<html>
<body>
    <form Runat="server">
        Amount 1: <my:MoneyField ID="field1" Text="100" Runat="server" /><br>
        Amount 2: <my:MoneyField ID="field2" Runat="server" />
    </form>
</body>
</html>>
```

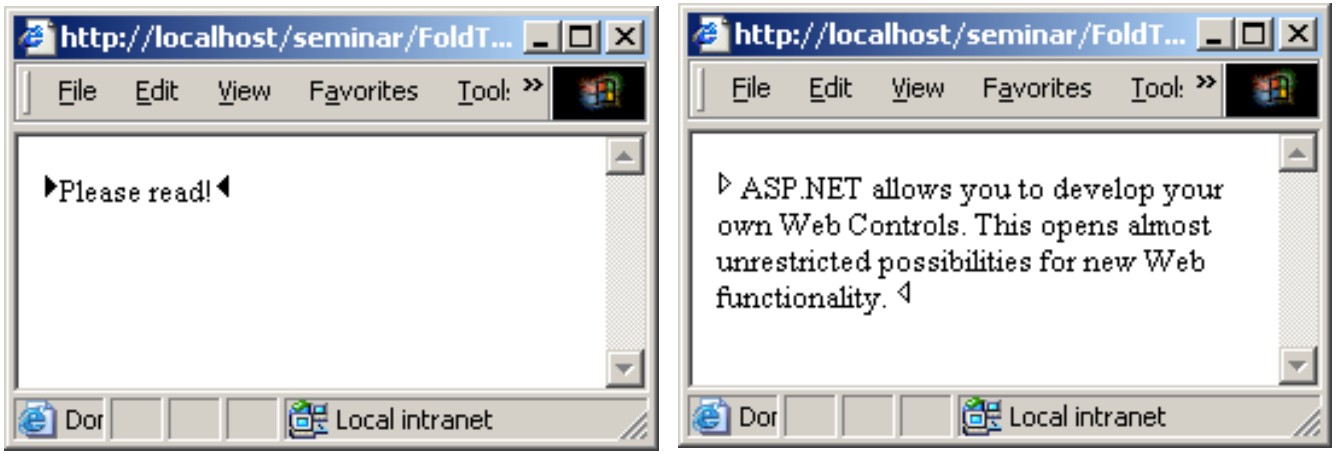


ASP.NET

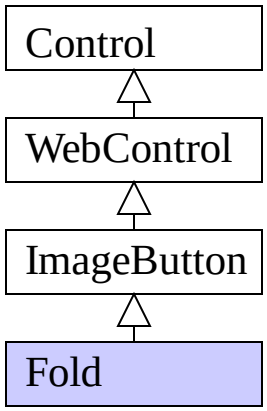
- Simple Dynamic Web Pages
- Web Forms
- Event Handling
- Web Controls
- Validators
- User Controls
- Custom Controls
- State Management
- Configuration of ASP.NET
- Working with Visual Studio .NET

Custom Controls (Example)

Allow you to implement completely new functionality (e.g. text folding)



Must be implemented as a (direct or indirect) subclass of *Control*



foreground text:	property Fold.Text
background text:	property ImageButton.AlternateText
Click event:	inherited from ImageButton

must override the *Render* method,
which translates this control to HTML

Custom Controls (Example: Class Fold)

```
using System; using System.Web.UI; using System.Web.UI.WebControls;
namespace Folds {  // custom controls must be declared in a namespace
   public class Fold : ImageButton {
      public string Text {
         get { return ViewState["Text"]==null ? "" : (string)ViewState["Text"]; }
         set { ViewState["Text"] = value; }
      }
      public string Icon {
         get { return ViewState["Icon"]==null ? "Solid" : (string)ViewState["Icon"]; }
         set { ViewState["Icon"] = value; }
      }
      public Fold() : base() { Click += new ImageClickEventHandler(FoldClick);}
      void FoldClick (object sender, ImageClickEventArgs e) {
         string s = Text; Text = AlternateText; AlternateText = s;  // AlternateText from ImageButton
         if (Icon == "Solid") Icon = "Hollow"; else Icon = "Solid";
      }
      protected override void Render (HtmlTextWriter w) {
         w.Write("<input type=image name=" + this.UniqueID);
         w.Write(" src='" + TemplateSourceDirectory + "/" + Icon + "Left.gif' border=0 />");
         w.Write(Text);
         w.Write("<img src='" + TemplateSourceDirectory + "/" + Icon + "Right.gif'>");
      }
   }
}
```

- SolidLeft.gif
- SolidRight.gif
- HollowLeft.gif
- HollowRight.gif

Custom Controls (Usage)

Must be compiled into a DLL, which has to be stored in the *\bin* directory

```
csc /target:library /out:bin/Fold.dll Fold.cs
```

Used as follows

```
<%@ Page Language="C#" %>
<%@ Register TagPrefix="my" Namespace="Folds" Assembly="Fold" %>
<html> <body>
    <form Runat="server">
        <my:Fold Text="..." AlternateText="..." Runat="server" />
    </form>
</body> </html>
```

ASP.NET

- Simple Dynamic Web Pages
- Web Forms
- Event Handling
- Web Controls
- Validators
- User Controls
- Custom Controls
- State Management
- Configuration of ASP.NET
- Working with Visual Studio .NET

3 Kinds of States

Page state

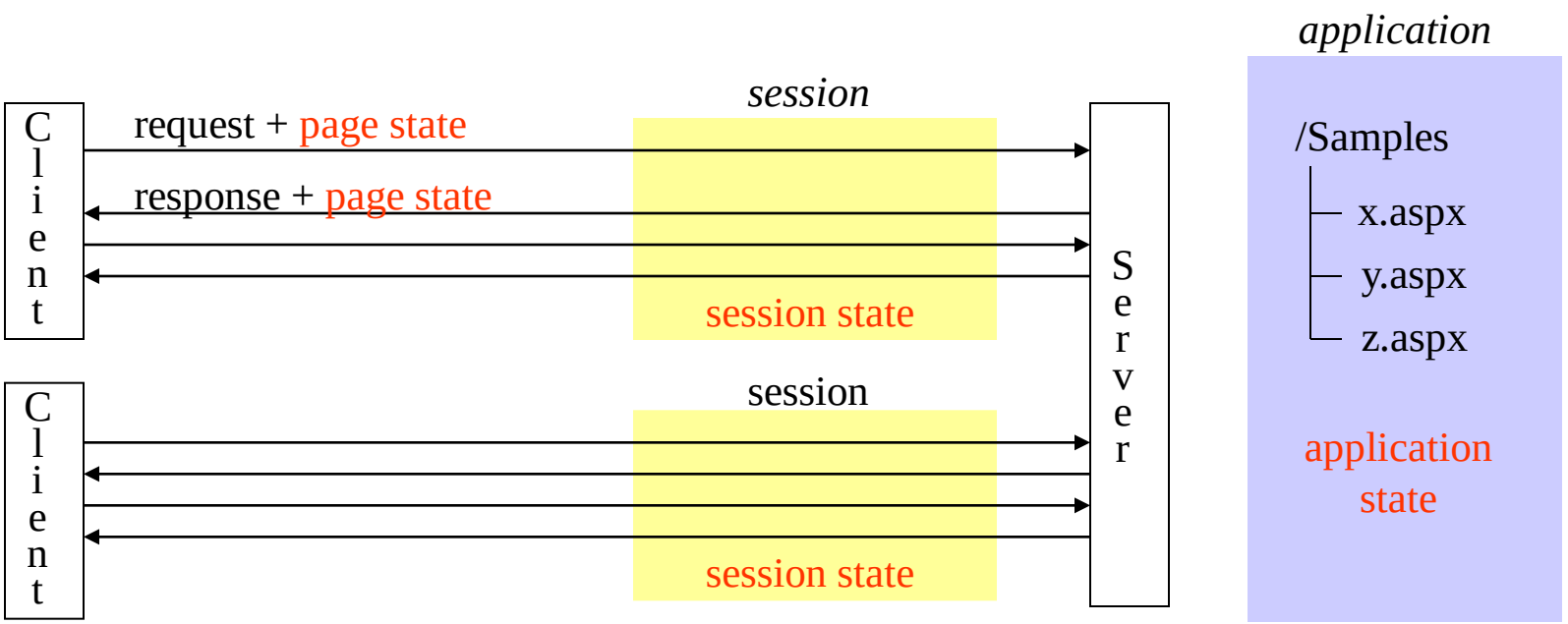
e.g. contents of TextBoxes, state of CheckBoxes, ...

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, ...



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

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, z.B. "/Samples"

Application and **Session**

application state and session state

Request und **Response**

HTTP request and HTTP response

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

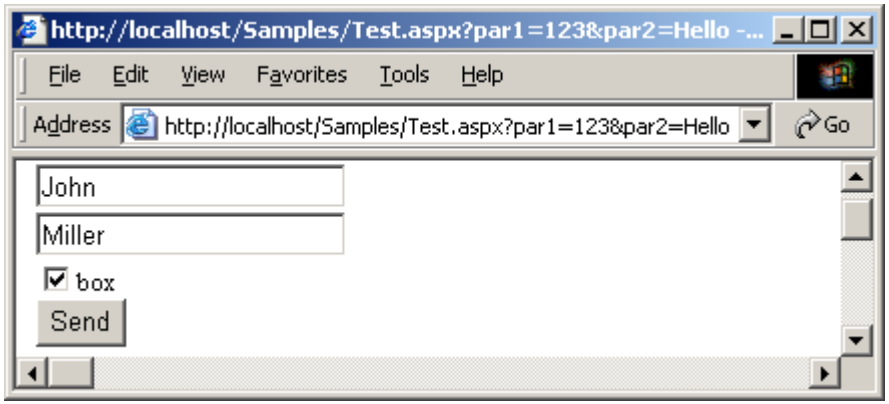
```
<body>
    <%= "address = " + Request.UserHostAddress %><br>
    <%= "method = " + Request.HttpMethod %><br>
    <%= "browser = " + Request.Browser.Browser %><br>
    <%= "version = " + Request.Browser.Version %><br>
    <%= "supports JS = " + Request.Browser.JavaScript %><br>
    <%= "server = "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

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

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
- __VIEWSTATE = dDwxMTYxMTk1 ...
- text1 = John
- text2 = Miller
- checkbox = on
- button = Send

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

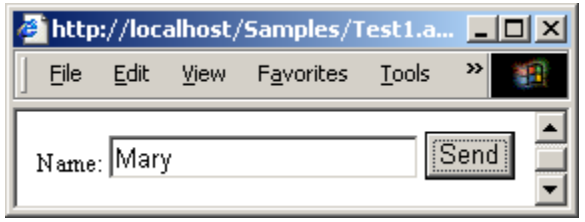
HTML response stream; can be written to with Write

StatusCode

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

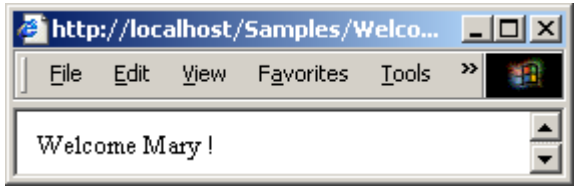
Test1.aspx

```
<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

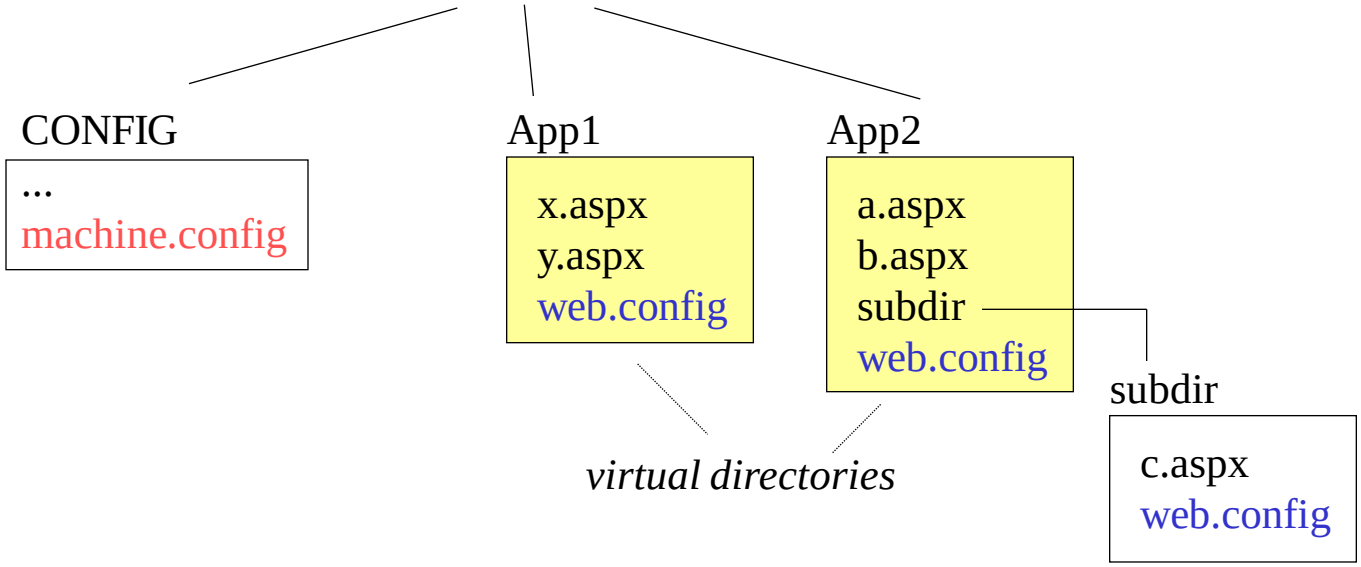
```
Welcome <%= Request.QueryString["name"] %> !
```



ASP.NET

- Simple Dynamic Web Pages
- Web Forms
- Event Handling
- Web Controls
- Validators
- User Controls
- Custom Controls
- State Management
- Configuration of ASP.NET
- Working with Visual Studio .NET

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

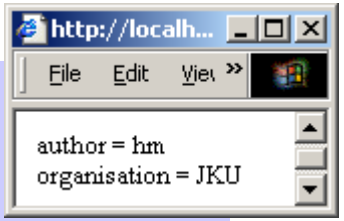
Configuration files are written in XML

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

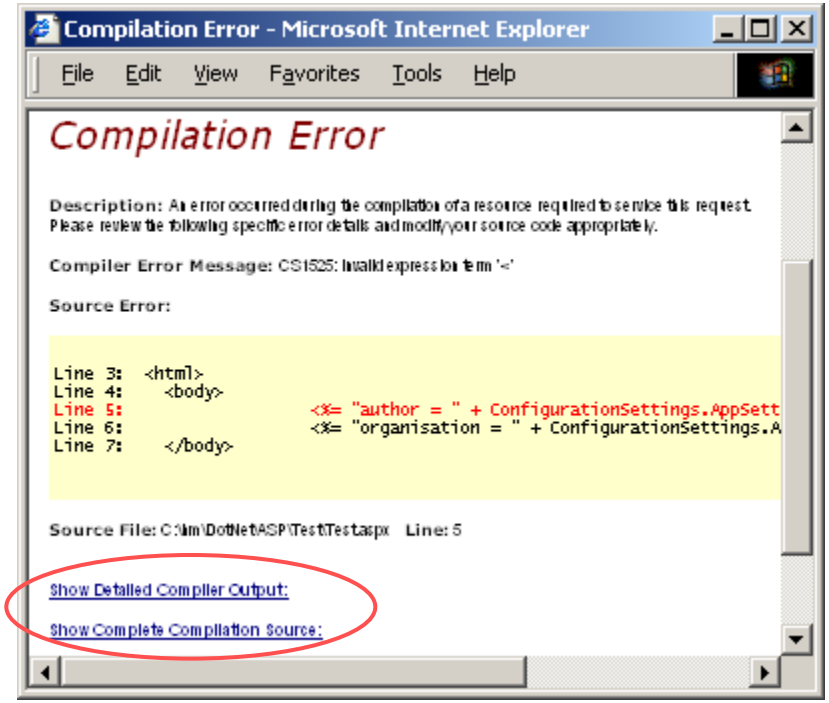



```
<%@Page Language="C#" %>
<%@ Import Namespace="System.Configuration" %>
<html>
   <body>
      <%= "author = " + ConfigurationSettings.AppSettings["author"] %><br>
      <%= "organisation = " + ConfigurationSettings.AppSettings["organisation"] %><br>
    </body>
</html>
```

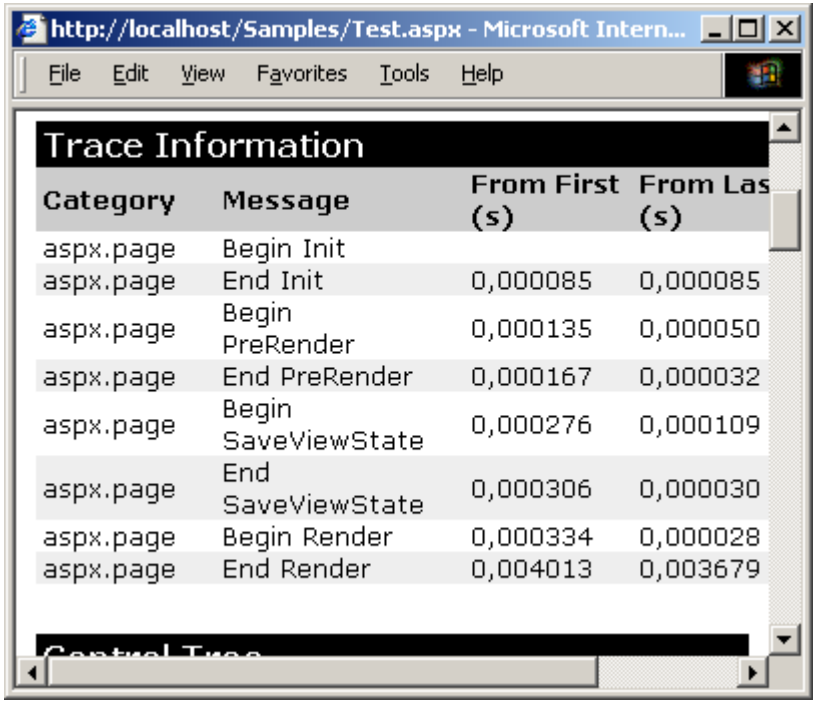
Example: Tracing

```
<?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



Authorisation

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

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.

Forms Authentication (Configuration)

web.config

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <system.web>
      <authorization>
         <deny users="?" />
      </authorization>
      <authentication mode="Forms">
         <forms loginUrl="Login.aspx" name="mycookie" protection="All" timeout="20">
            <credentials passwordFormat="MD5">
               <user name="peter"    password="85C69322756E01FD4A7A22DE55E19743"/>
               <user name="wolfgang" password="85C69322756E01FD4A7A22DE55E19743"/>
            </credentials>
         </forms>
      </authentication>
      ...
   </system.web>
   ...
</configuration>
```

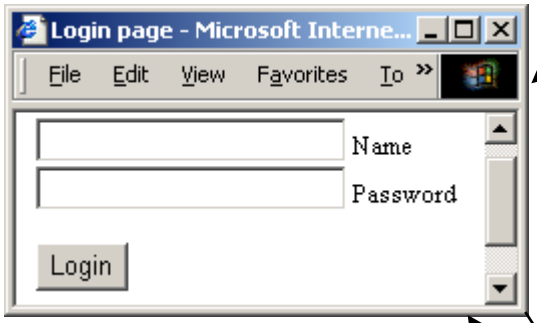
```
string encryptedPwd =
FormsAuthentication.HashPasswordForStoringInConfigFile("myPwd", "MD5");
```

The users "peter" and "wolfgang" as well as their passwords are stored on the server

Authentication (How it Works)

1. Anonymous user tries to access *A.aspx*

2. Because of <deny users="?"/> and <forms loginUrl="Login.aspx"> the user is redirected to *Login.aspx*



3. User enters her name and password and is authenticated

no

yes

4. Authentication successful?

directory requiring authentication

A.aspx *B.aspx* *C.aspx*

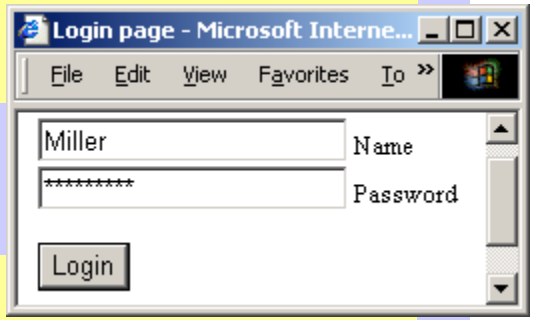
web.config

```
<deny users="?" />
..
<forms loginUrl="Login.aspx" ...>
```

5. User is redirected to *A.aspx* and may also access all other pages of this directory now (because she is authenticated)

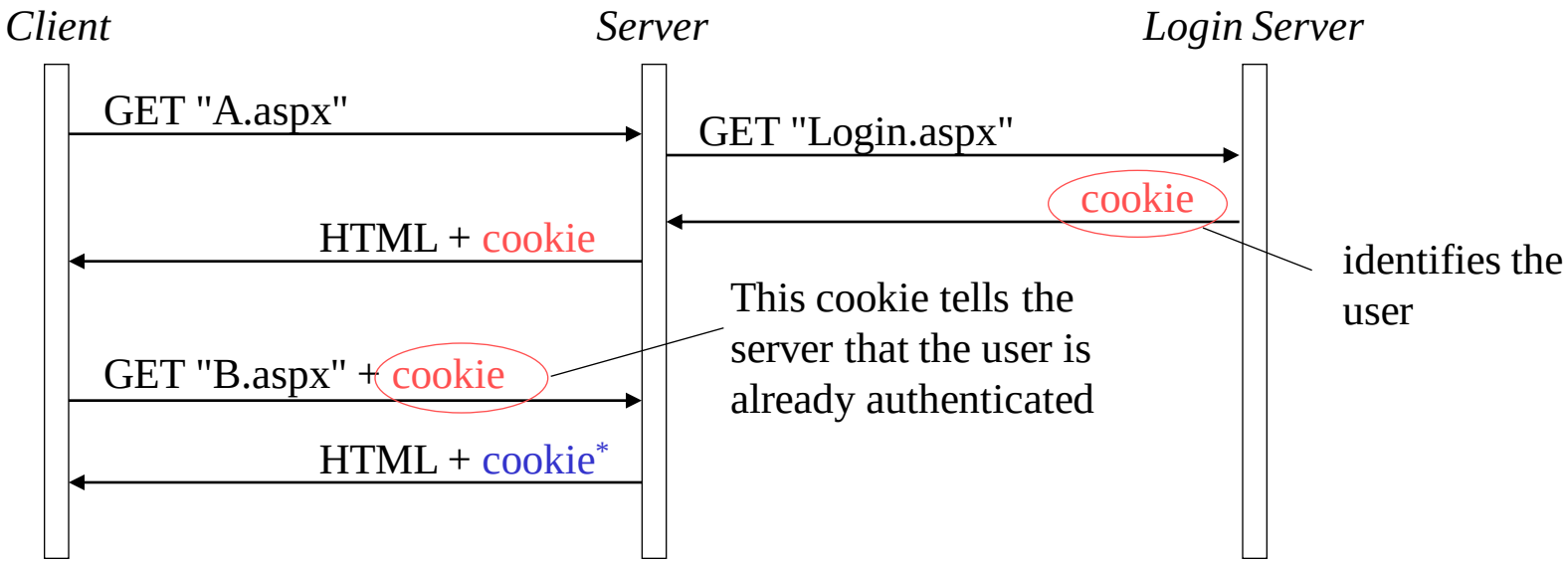
Login.aspx

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.Web.Security" %>
<html>
   <head>
      <title>Login page</title>
      <script Language="C#" Runat="server">
         void Authenticate (object sender, EventArgs e) {
            if (FormsAuthentication.Authenticate(user.Text, pwd.Text) || user.Text == "Karin")
               FormsAuthentication.RedirectFromLoginPage(user.Text, false);
            else
               msg.Text = "-- authentication failed";
         }
      </script>
   </head>
   <body>
         <form Runat="server">
            <asp:TextBox ID="user" Runat="server"/> Name<br>
            <asp:TextBox ID="pwd" TextMode="Password" Runat="server"/> Password<br><br>
            <asp:Button ID="button" Text="Login" OnClick="Authenticate" Runat="server" />
            <br><br>
            <asp:Label ID="msg" Runat="server" />
         </form>
   </body>
</html>
```



User Identification With Cookies

How does ASP.NET remember if a user is already authenticated?



Specifications about Cookies in the configuration file

```
<forms loginUrl="Login.aspx" name="mycookie" protection="All" timeout="20" >
```

name of the cookie to be generated

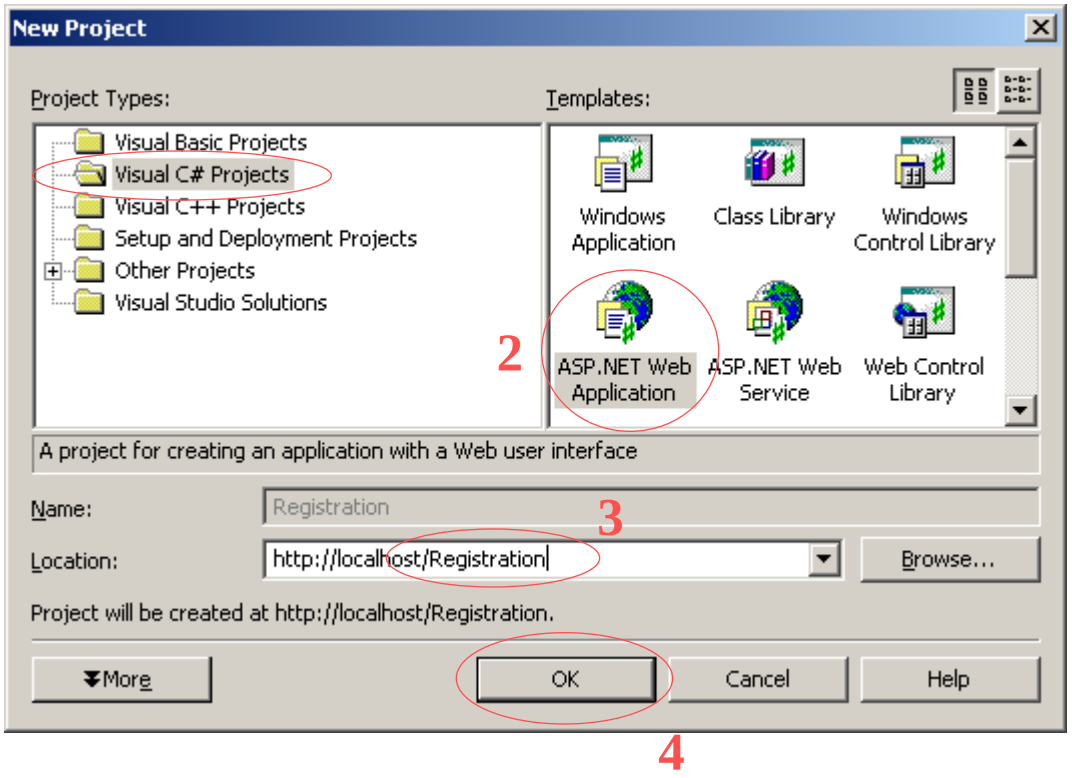
cookies should be encrypted

cookies should become invalid after 20 minutes

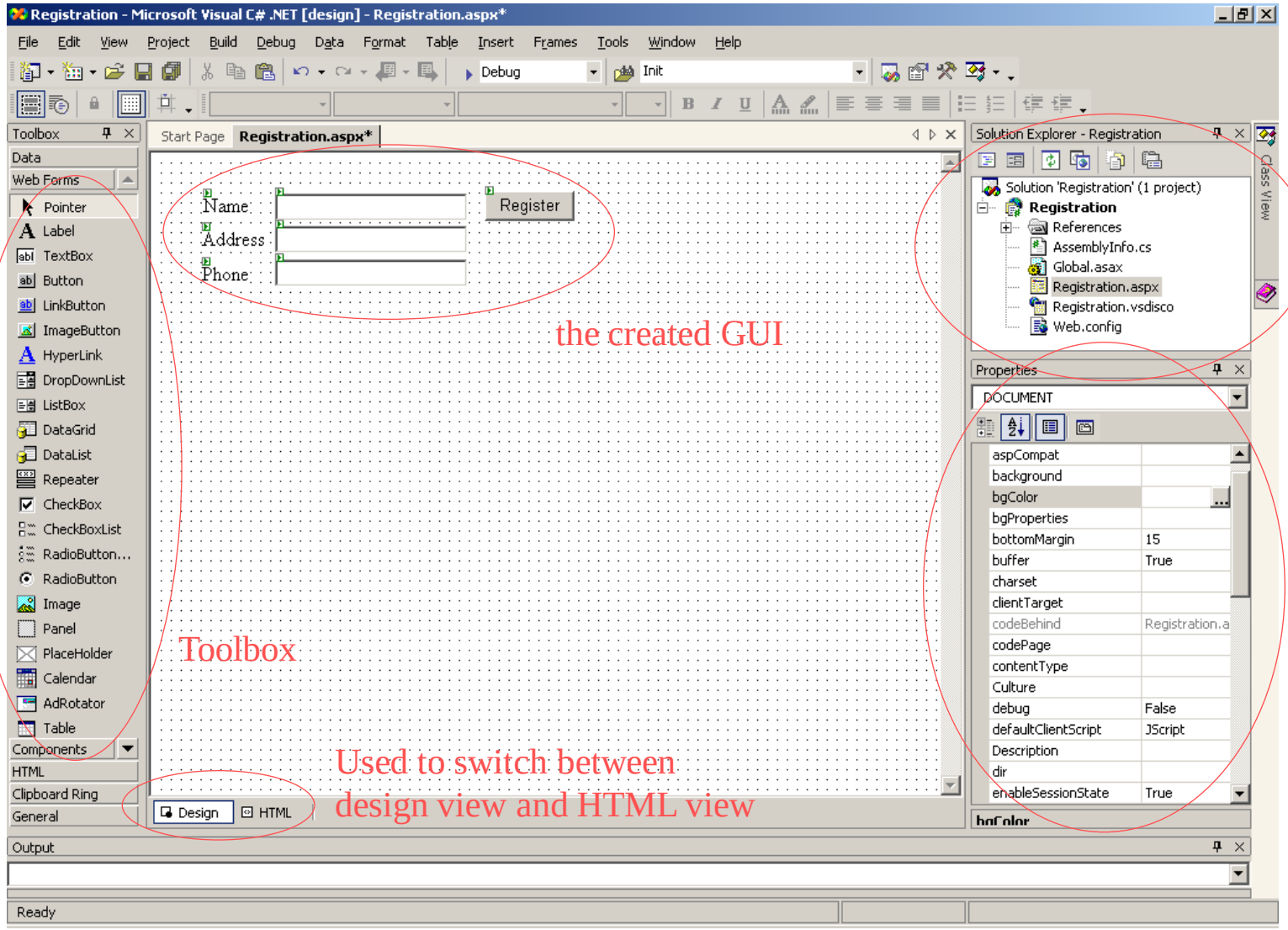
ASP.NET

- Simple Dynamic Web Pages
- Web Forms
- Event Handling
- Web Controls
- Validators
- User Controls
- Custom Controls
- State Management
- Configuration of ASP.NET
- Working with Visual Studio .NET

Creating a Project



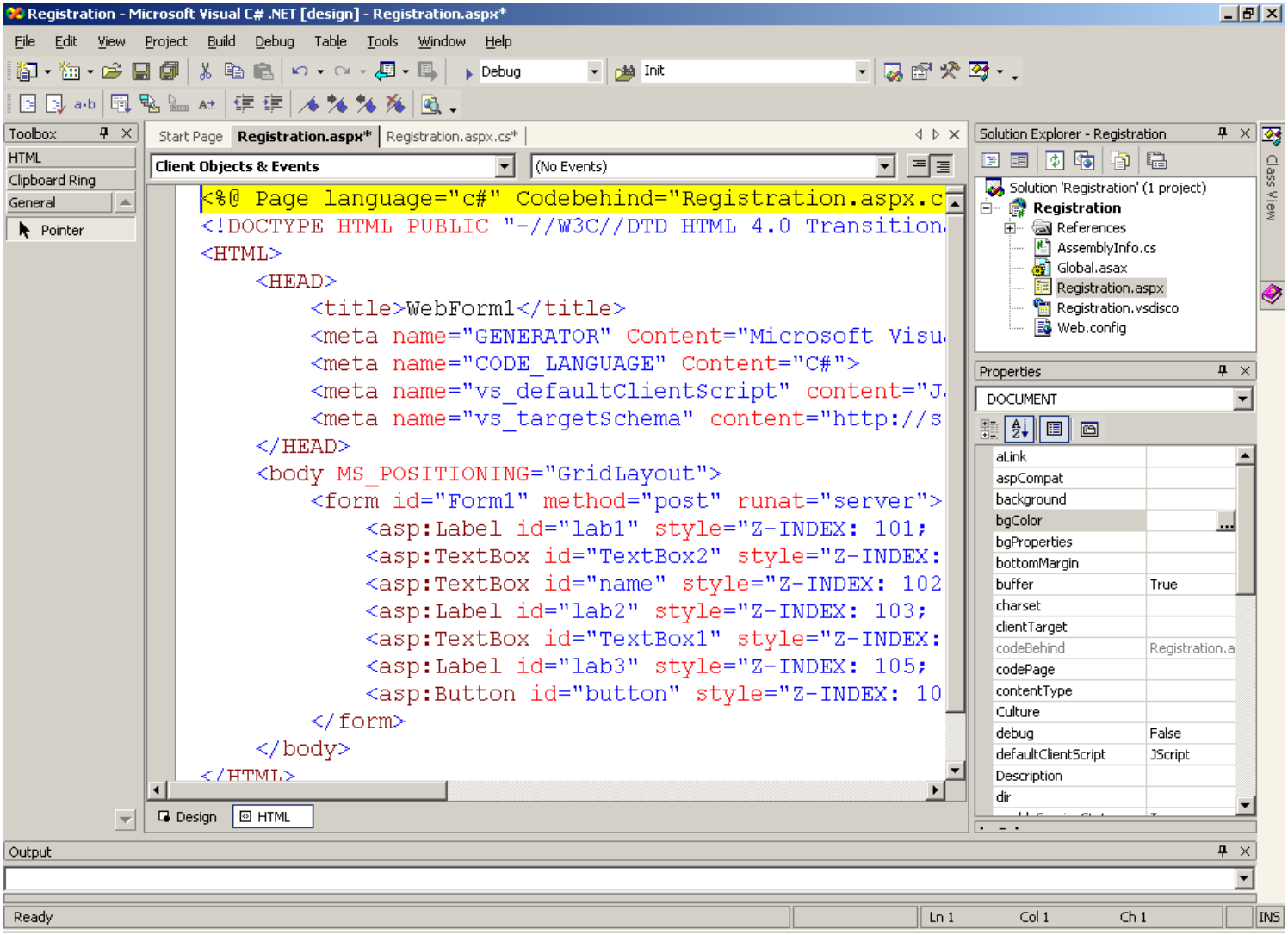
Composing a GUI With Drag & Drop



Solution Explorer

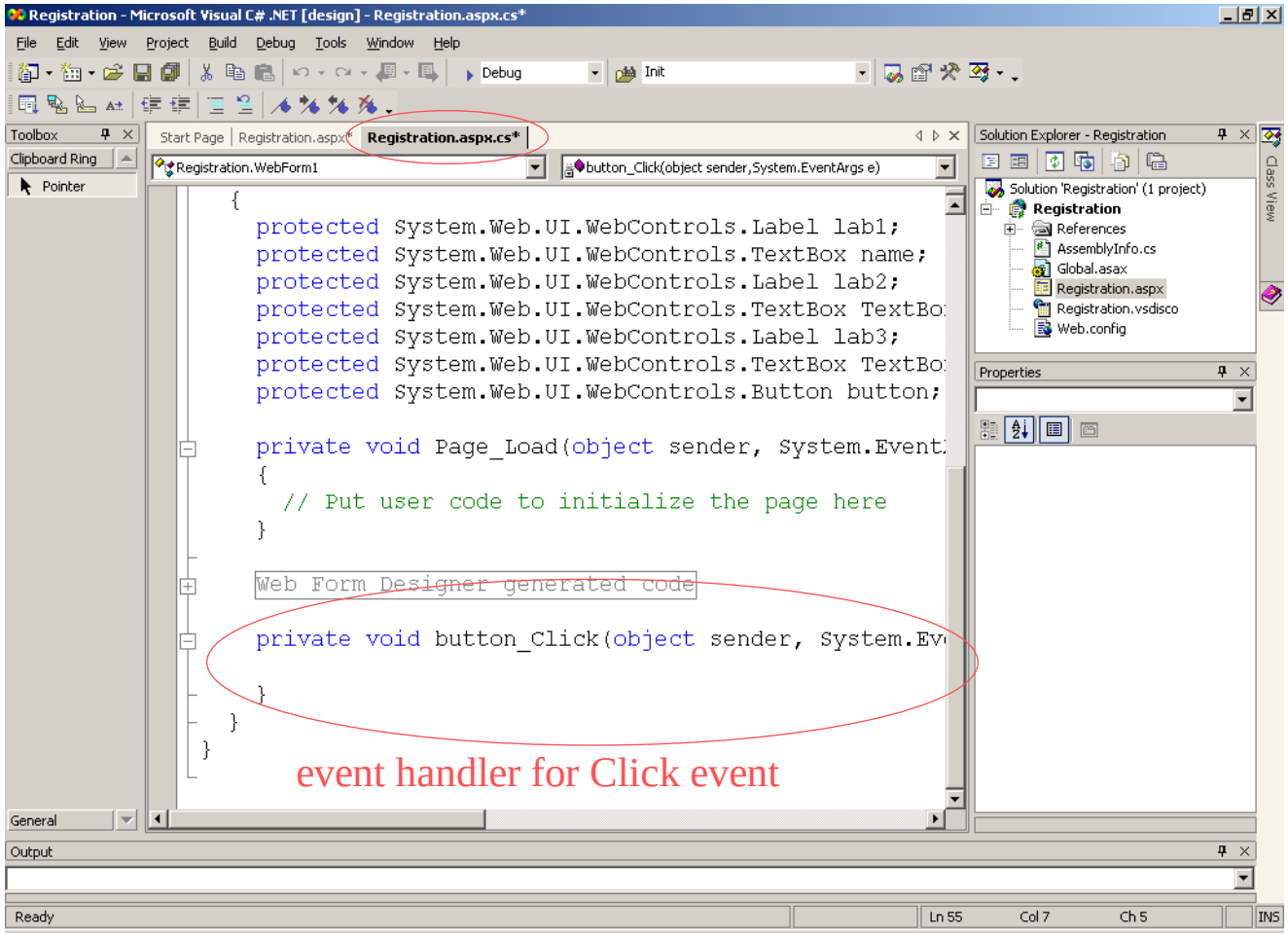
Property Window

HTML View


Registration - Microsoft Visual C# .NET [design] - Registration.aspx*
File Edit View Project Build Debug Table Tools Window Help
Debug
Init
Toolbox
HTML
Clipboard Ring
General
Pointer
Start Page
Registration.aspx*
Registration.aspx.cs*
Client Objects & Events
(No Events)
<%@ Page language="c#" Codebehind="Registration.aspx.c
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transition
<HTML>
<HEAD>
<title>WebForm1</title>
<meta name="GENERATOR" Content="Microsoft Visu
<meta name="CODE_LANGUAGE" Content="C#">
<meta name="vs_defaultClientScript" content="J
<meta name="vs_targetSchema" content="http://s
</HEAD>
<body MS_POSITIONING="GridLayout">
<form id="Form1" method="post" runat="server">
<asp:Label id="lab1" style="Z-INDEX: 101;
<asp:TextBox id="TextBox2" style="Z-INDEX:
<asp:TextBox id="name" style="Z-INDEX: 102
<asp:Label id="lab2" style="Z-INDEX: 103;
<asp:TextBox id="TextBox1" style="Z-INDEX:
<asp:Label id="lab3" style="Z-INDEX: 105;
<asp:Button id="button" style="Z-INDEX: 10
</form>
</body>
</HTML>
Design
HTML
Solution Explorer - Registration
Solution 'Registration' (1 project)
Registration
References
AssemblyInfo.cs
Global.asax
Registration.aspx
Registration.vsdisco
Web.config
Class View
Properties
DOCUMENT
aLink
aspCompat
background
bgColor
bgProperties
bottomMargin
buffer True
charset
clientTarget
codeBehind Registration.a
codePage
contentType
Culture
debug False
defaultClientScript JScript
Description
dir
Output
Ready
Ln 1 Col 1 Ch 1
INS

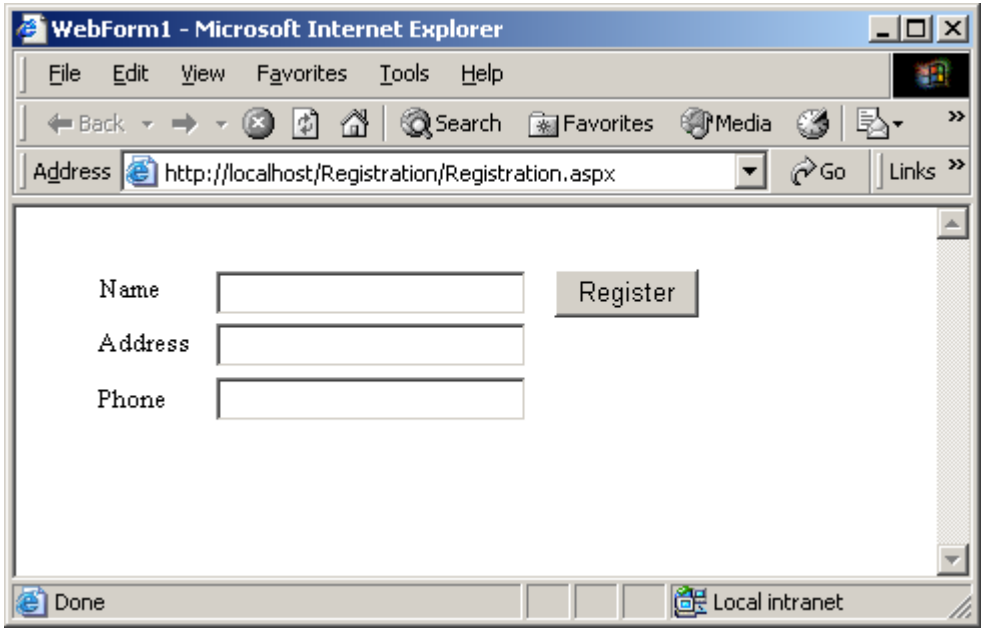
Event Handling

Double click on a Button creates an event handler in the code behind



Executing the Page

Menu: Debug | Start

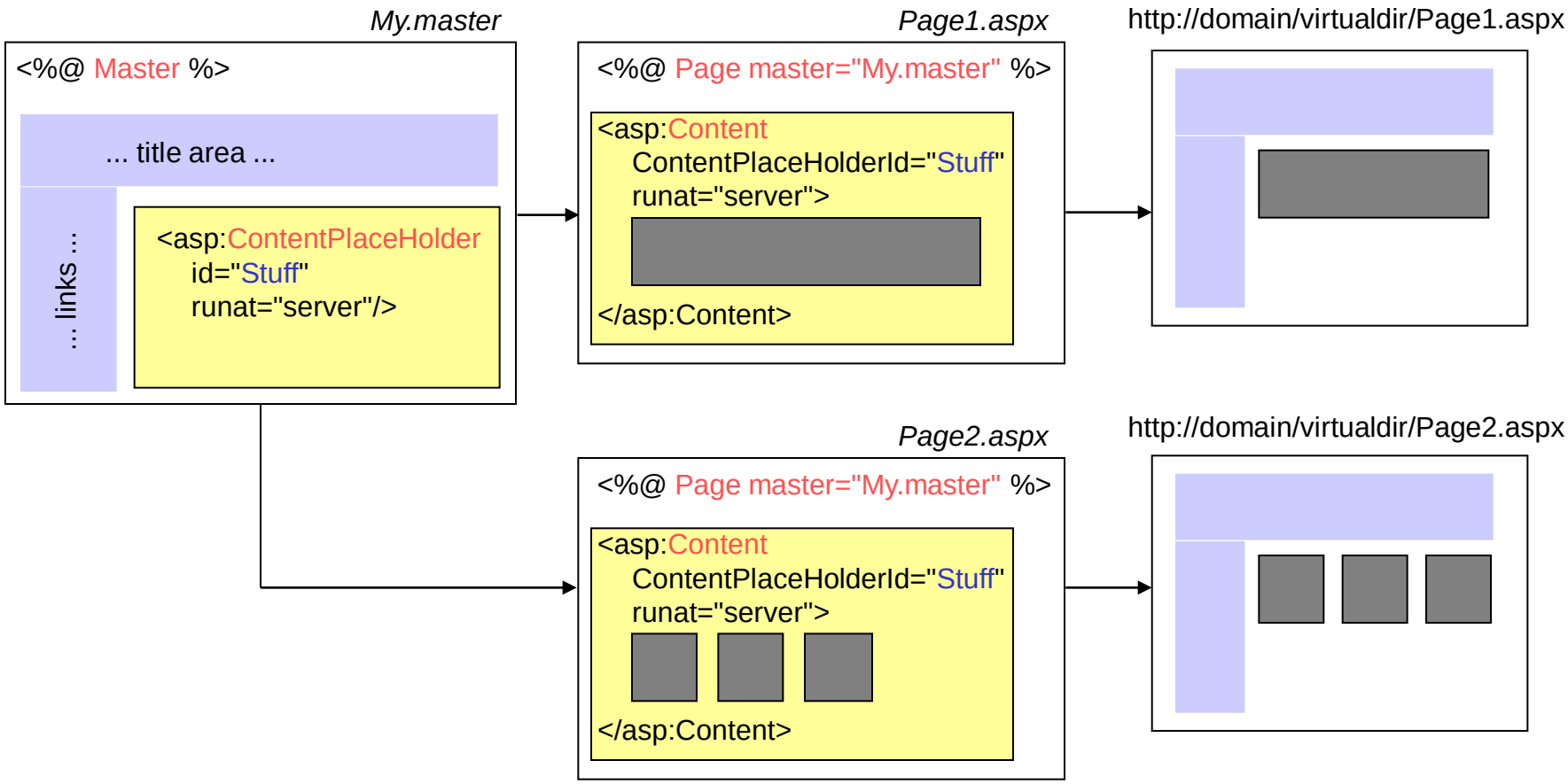


New Features in ASP.NET 2.0

- Master pages
- Navigation
- Themes and skins
- Personalisation
- Authentication and membership classes
- New Controls for database access
- ...

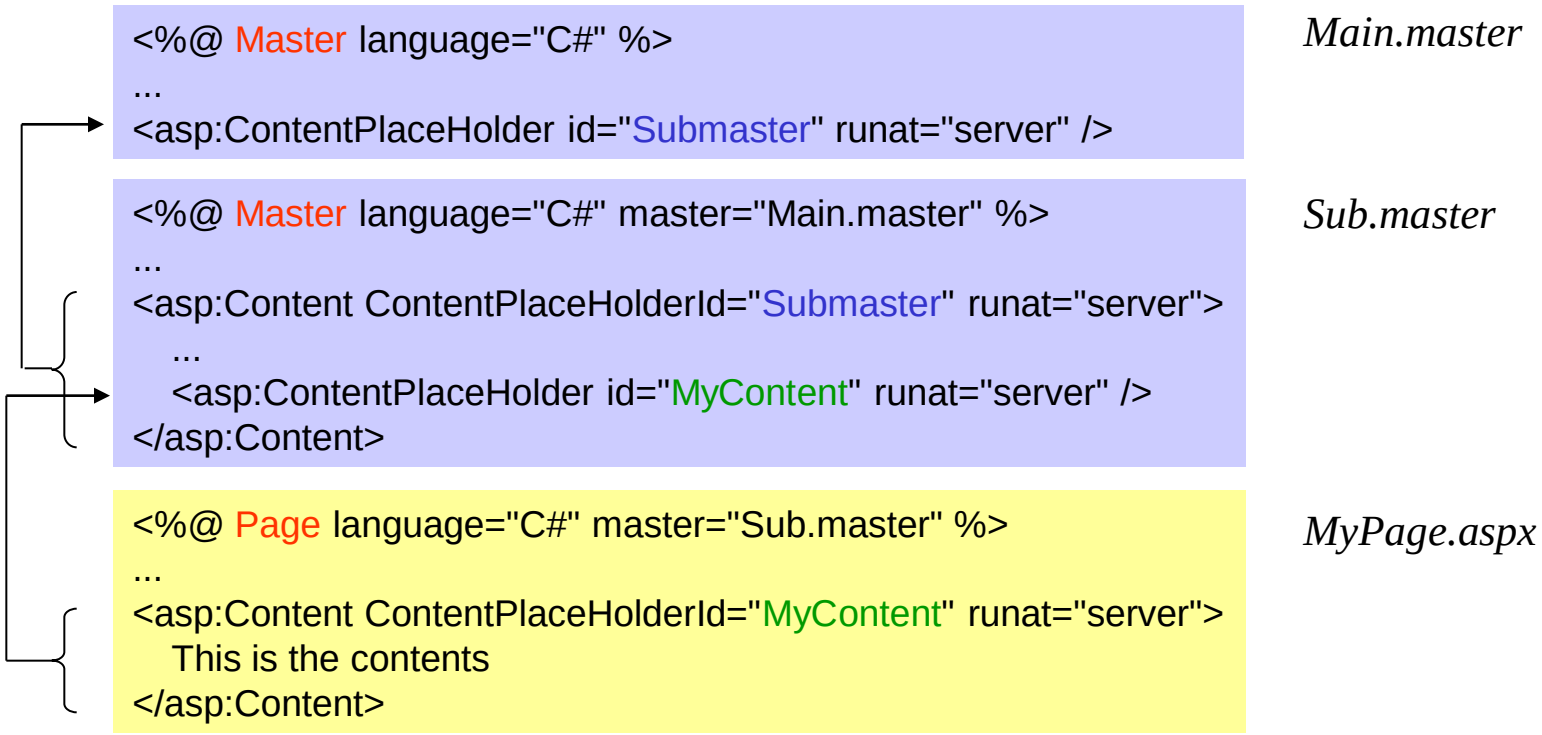
Idea

Unform layout for all pages



Characteristics of Master Pages

- May contain arbitrary HTML code and arbitrary ASP.NET web controls
- May have a code behind
- Canbe nested



Main.master

```
<%@ Master language="C#" %>
...
<asp:ContentPlaceHolder id="Submaster" runat="server" />
```

Sub.master

```
<%@ Master language="C#" master="Main.master" %>
...
<asp:Content ContentPlaceHolderId="Submaster" runat="server">
    ...
    <asp:ContentPlaceHolder id="MyContent" runat="server" />
</asp:Content>
```

MyPage.aspx

```
<%@ Page language="C#" master="Sub.master" %>
...
<asp:Content ContentPlaceHolderId="MyContent" runat="server">
    This is the contents
</asp:Content>
```

- Master can be assigned to all pages of a site by specifying it in *web.config*
- Users can define different master pages fr different browser types

Code Behind

As a base class of the page class (as it was so far)

```
<% Master language="C#" inherits="MyBase" src="My.master.cs" %>
```

My.master

```
public class MyBase: System.Web.UI.MasterPage {...}
```

My.master.cs

As a partial class

```
<% Master language="C#" compilewith="My.master.cs" classname="MyLogic" %>
```

My.master

```
public partial class MyLogic {...}
```

My.master.cs

Web controls of a page need not be declared as fields any more in the code behind file!

New Features in ASP.NET 2.0

- Master pages
- Navigation
- Themes and skins
- Personalisation
- Authentication and membership classes
- New Controls for database access
- ...

Site Maps

Page hierarchy of an application is described in XML

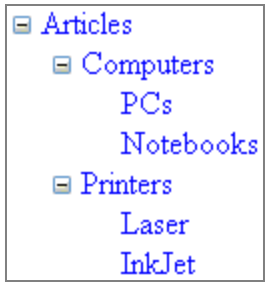
app.sitemap

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap>
   <siteMapNode title="Articles" description="Home" url="Articles.aspx" >
      <siteMapNode title="Computers" url="Computers.aspx">
         <siteMapNode title="PCs" url="PCs.aspx" />
         <siteMapNode title="Notebooks" url="Notebooks.aspx" />
      </siteMapNode>
      <siteMapNode title="Printers" url="Printers.aspx">
         <siteMapNode title="Laser" url="Laser.aspx" />
         <siteMapNode title="InkJet" url="InkJet.aspx" />
      </siteMapNode>
   </siteMapNode>
</siteMap>
```

Each of these pages uses the same Master. This Master contains a *TreeView* control. Thus the *TreeView* is re-displayed on every page.

TreeView control can be bound to this hierarchy

```
<asp:SiteMapDataSource ID="data" runat="server"/>
<asp:TreeView DataSourceID="data" runat="server"/>
```



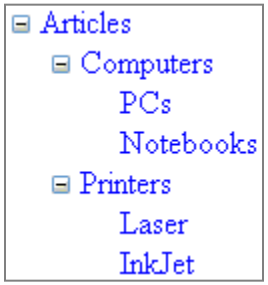
connects to *app.sitemap* by default

Navigation With SiteMapPath

```
<asp:SiteMapPath runat="server"/>
```

Shows the path to the current page if this page belongs to a Sitemap

Articles > Computers > PCs



New Features in ASP.NET 2.0

- Master pages
- Navigation
- Themes and skins
- Personalisation
- Authentication and membership classes
- New Controls for database access
- ...

Themes and Skins

Allow users to define standard settings for web controls

Skin: settings for a specific control

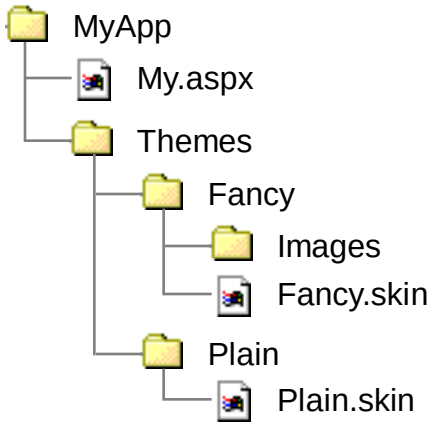
```
<asp:Label ForeColor="#585880" Font-Size="0.9em" Font-Name="Verdana" />
```

can be given a name

```
<asp:Label SkinID="red" ForeColor="#FF0000" Font-Name="Arial" Font-Bold="true" />
```

All other attributes of the control remain unchanged

Theme: collection of skins; stored in a file with the ending *.skin* in a *Themes* subdirectory



Fancy.skin

```
<asp:Label .../>
<asp:Button .../>
<asp:TextBox ...>/
```

Plain.skin

```
<asp:Label .../>
<asp:Button .../>
<asp:TextBox ...>/
```

Setting a Theme

Globally for the whole application

```
<configuration>
   <system.web>
      <pages theme="Fancy" />
      ...
   </system.web>
</configuration>
```

web.config

For a single page

- either in the *Page* directive

```
<%@ Page Language="C#" Theme="Fancy" %>
...
```

- or in the code behind

```
public class PageBase: Page {
   public void Page_PreInit(object sender, EventArgs e) {
      Theme = "Fancy";
   }
}
```

- ASP.NET 2.0 has a new *PreInit* event
- *Page* has a property with the name *Theme*

Explicitly Selecting a Skin

Selecting a skin element

Fancy.skin

```
<asp:Label ForeColor="#585880" Runat="server" />
<asp:Label SkinID="Red" ForeColor="#FF0000" Runat="server" />
...
```

```
<%@ Page Language="C#" Theme="Fancy" %>
...
<asp:Label Runat="server">color #585880"</asp:Label>
<asp:Label SkinID="Red" Runat="server">color #FF0000"</asp:Label>
...
```

Themes
- Fancy
  - Fancy.skin

Selecting a skin file

Fancy.skin

```
<asp:Label ForeColor="#585880" Runat="server" />
...
```

Red.skin

```
<asp:Label ForeColor="#FF0000" Runat="server" />
...
```

```
<%@ Page Language="C#" Theme="Fancy" %>
...
<asp:Label Runat="server">color #585880"</asp:Label>
<asp:Label SkinID="Red" Runat="server">color #FF0000"</asp:Label>
```

Themes
- Fancy
  - Fancy.skin
  - Red.skin

New Features in ASP.NET 2.0

- Master pages
- Navigation
- Themes and skins
- Personalisation
- Authentication and membership classes
- New Controls for database access
- ...

Personalisation

Allows the creation of user profiles

- Stored as name/value pairs
- Defined in *web.config*

```
<system.web>
   <personalization>
      <profile>
         <property name="User" type="System.String" />
         <property name="LastVisit" type="System.DateTime" />
      </profile>
   </personalization>
</system.web>
```

- Can be accessed through the *Profile* property of the *Page* class

```
label.Text = "Welcome " + Profile.User;
Profile.LastVisit = DateTime.Now;
```

- Are statically typed!
- Are stored in a database (i.e. persistent)
- Are only loaded on demand
- Users are identified via cookies or URL rewriting

} differences to the session state!

Selecting a Personalisation Database

Done via Provider classes

- Can be specified in *web.config*

```
<personalization defaultProvider="AspNetAccessProvider">
```

- Providers for MS Access and MS SQL Server are available by default
- Standard provider is MS Access
  Writes to the file *ApplicationDir*/DATA/AspNetDB.mdb
- Users can write and install their own providers

Example: Personalized Theme

Defining a profile property for themes

```
<personalization>
   <profile>
      <property name="Theme" type="System.String"/>
   </profile>
</personalization>
```

Users can set their preferred theme during one of their visits


```
void SetTheme(object sender, EventArgs e) {
   Profile.Theme = textBox.Text;
}
```

This theme is then used during later visits

```
void Page_PreInit(object sender, EventArgs e) {
   Theme = Profile.Theme;
}
```

New Features in ASP.NET 2.0

- Master pages
- Navigation
- Themes and skins
- Personalisation
- Authentication and membership classes
- New Controls for database access
- ...

Login Controls 1/2

LoginStatus

```
<asp:LoginStatus Runat="server" />
```

displays

Login — if the user is not yet logged in
Logout — if the user is logged in

Login link leads to a page that can be specified in *web.config*

```
<authentication mode="Forms">
   <forms loginUrl="login.aspx" />
</authentication>
```

LoginView and LoginName

```
<asp:LoginView Runat="server">
  <AnonymousTemplate>
    You are not logged in. Click the Login link to sign in.
  </AnonymousTemplate>
  <LoggedInTemplate>
    You are logged in. Welcome,
    <asp:LoginName Runat="server" />!
  </LoggedInTemplate>
</asp:LoginView>
```

<AnonymousTemplate> -------- text that is displayed if the user is not yet logged in

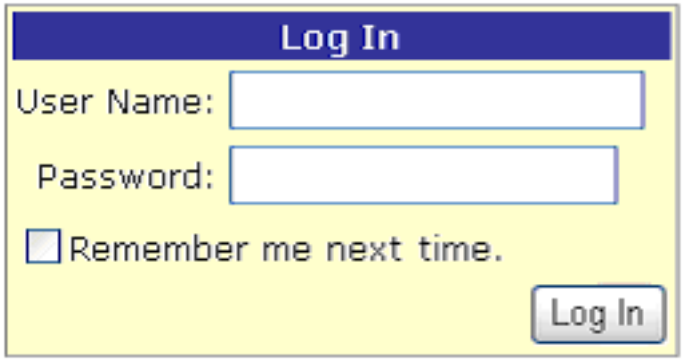
<LoggedInTemplate> -------- text that is displayed if the user is already logged in

LoginName: name that the user used for his login

Login Controls 2/2

Login

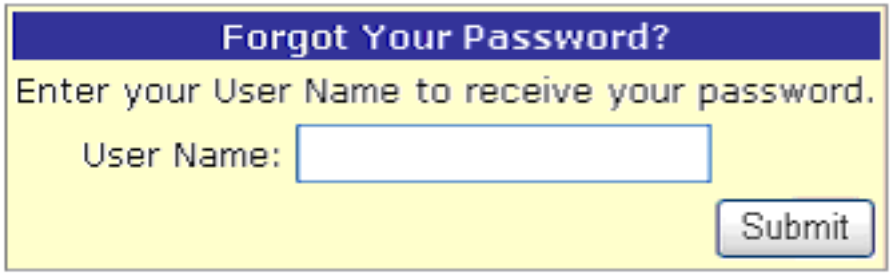
```
<asp:Login Runat="server" />
```



- is used on the login page that was specified in *web.config*.
- if the authentication succeeds the user is redirected to the page that was originally requested by him.

PasswordRecovery

```
<asp:PasswordRecovery Runat="server">
   <MailDefinition
      from="mailto:admin@dotnet.jku.at"
      cc="your password" />
</asp:PasswordRecovery>
```



- sends an email with the password to the user
- email address and password are stored in the user data (see later)

Membership Classes

Membership (in *System.Web.Security*)

- Maintains a set of users

```
static MembershipUser CreateUser(string name, string password) {...}
static MembershipUser GetUser(string name) {...}
static void UpdateUser(MembershipUser user) {...}
static bool DeleteUser(string name) {...}
static bool ValidateUser(string name, string password) {...}
```

- Users are stored in a database (Access, SQL Server, user-defined)
  Default is MS Access: *ApplicationDir*/data/AspNetDB.mdb
- Users are identified via cookies or URL rewriting
- *ValidateUser* is called from the *Login* web control

MembershipUser (in *System.Web.Security*)

- Contains the data of a single user
  - name
  - password
  - email address
  - last login date
  - password recovery question
  - ...

There must be a web page on which the user can enter this data. The data is then stored in the database using *CreateUser* or *UpdateUser*

Protected Member Pages

- Stored in a subdirectory of the application directory (e.g. Members/)
  - Application
    ...
    - Members
      - xxx.aspx
      - yyy.aspx
- This subdirectory must have a *web.config* file with an <authorization> element

```
<configuration>
   <system.web>
      <authorization>
         <deny users="?" />
         <allow users="Peter, Mike" />
      </authorization>
   </system.web>
</configuration>
```

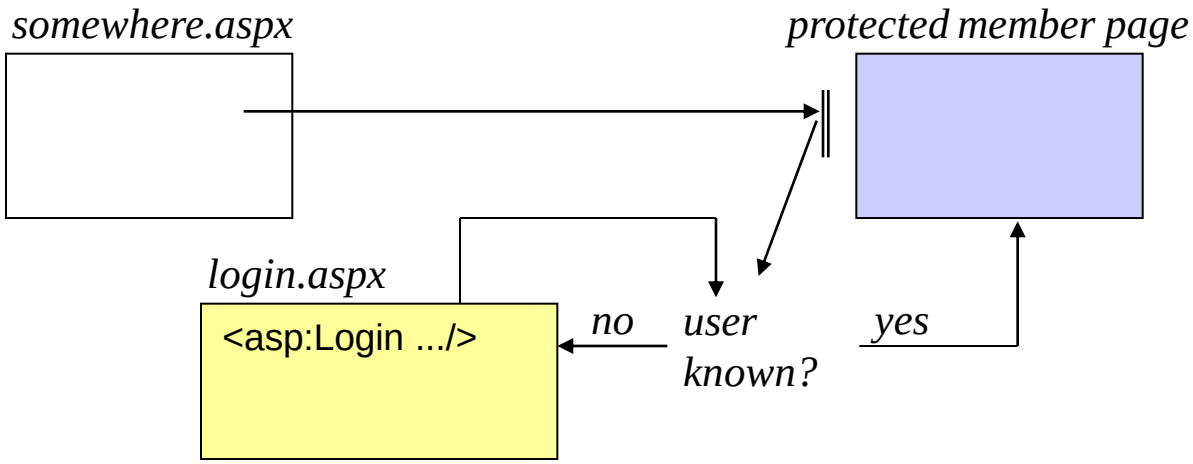
unknown users must register at a login page

users Peter and Mike can access the member pages without login

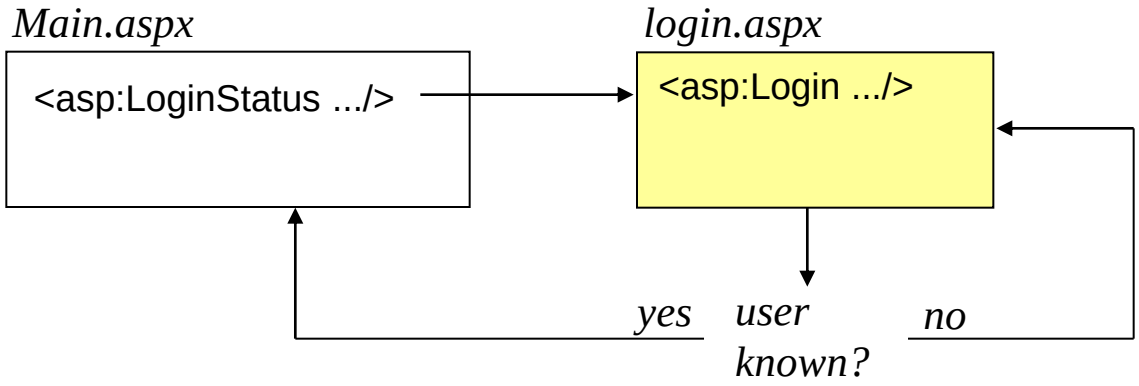
- If unknown users try to access a member page they are redirected to the login page.
- If authenticated users (or Peter or Mike) try to access a member page they are pased through

Authentication

Access to protected member pages



Login via LoginStatus control



Roles

Class *Roles* (in *System.Web.Security*)

- manaes user roles (e.g. *Admin*, *Employee*, *User*, ...)

```
static void CreateRole(string roleName) {...}
static void AddUserToRole(string userName, string roleName) {...}
...
```

- Access to the pages of a directory can also be managed via roles.
  Requires *web.config* in this directory:

```
<system.web>
   <roleManager enabled="true" />
   <authorization>
      <allow roles="Admin" />
      <deny users="*" />
   </authorization>
<system.web>
```

only users with the role *Admin* are allowed to access the pages in this directory

all other users are denied access

New Features in ASP.NET 2.0

- Master pages
- Navigation
- Themes and skins
- Personalisation
- Authentication and membership classes
- New Controls for database access
- ...

Visualization of Database Tables

AccessDataSource, SqlDataSource, XmlDataSource

```
<asp:AccessDataSource ID="data" Runat="server"
    DataFile="db.mdb"
    SelectCommand="SELECT id, name, subject FROM Students" />
```

- The web control specifies the database name and the SQL command
- Creates a *DataSet* that can be displayed in a *ListBox*, *DropDownList*, *GridView*, etc.

GridView

```
<asp:GridView DataSourceID="data" Runat="server" AllowSorting="true" />
```

id	name	subject
9955004	Miller	Computer Science
9934128	Johnson	Economics
9955332	Howard	Computer Science
9882076	Keen	Physics
9834524	Feldman	Economics

- Shows the *AccessDataSource* with the name *data*
- Columns can be sorted by clicking on the column name
- Allows various ways of formatting

DropDownList With AccessDataSource

```
<asp:AccessDataSource ID="data" Runat="server"
   DataFile="db.mdb"
   SelectCommand="SELECT DISTINCT subject FROM Students" />

<asp:DropDownList DataSourceID="data" DataTextField="subject" Runat="server" />
```

Parameters of Select Commands

```
<asp:TextBox id="subj" AutoPostBack="true" Runat="server" />

<asp:AccessDataSource ID="data" Runat="server"
   DataFile="db.mdb"
   SelectCommand="SELECT id, name, subject FROM Students WHERE subject=@par"
   <SelectParameters>
      <asp:ControlParameter name="par" ControlID="subj" PropertyName="Text" />
   </SelectParameters>
</asp:AccessDataSource>
```

Computer Science

id	name	subject
9955004	Miller	Computer Science
9955332	Howard	Computer Science

Editing a GridView

```
<asp:AccessDataSource ID="data" Runat="server"
    DataFile="db.mdb"
    SelectCommand="SELECT id, name, subject FROM Students"
    UpdateCommand="UPDATE Students SET name=@name, subject=@subject  WHERE id=@id"
    DeleteCommand="DELETE FROM Students WHERE id=@id" />

<asp:GridView ID="grid" DataSource="data" Runat="server"
    DataKeyNames="id"
    AutoGenerateEditButton="true"
    AutoGenerateDeleteButton="true" />
```

column values

	id	name	subject
Edit Delete	9955004	Miller	Computer Science
Edit Delete	9934128	Johnson	Economics
Edit Delete	9955332	Howard	Computer Science
Edit Delete	9882076	Keen	Physics
Edit Delete	9834524	Feldman	Economics

	id	name	subject
Edit Delete	9955004	Miller	Computer Science
Update Cancel	9934128	Johnson	Economics
Edit Delete	9955332	Howard	Computer Science
Edit Delete	9882076	Keen	Physics
Edit Delete	9834524	Feldman	Economics

Additionally, events are raised that can be caught and handled

Detailed View of a Database Record

DetailsView

```
<asp:AccessDataSource ID="data" Runat="server"
   DataFile="db.mdb"
   SelectCommand="SELECT id, name, subject FROM Students" />

<asp:DetailsView DataSourceID="data" Runat="server"
   DataKeyNames="id"
   AllowPaging="true"
   PagerSettings-Mode="NextPrevFirstLast" />
```

id	9955332
name	Howard
subject	Computer Science

<< < > >>

- Used if the record is too big for a single line
- This view can also be edited (like a *GridView*)
- Allows various ways of formatting

Summary

Summary

Advantages of ASP.NET over ASP

- object-oriented
- event-based
- predefined and user-defined web controls
- separation between layout (HTML) and business logic (e.g. C#)
- compiled code instead of interpreted server scripts
- convenient state management

Examples available at

http://dotnet.jku.at/book/samples/	Examples from the book
http://dotnet.jku.at/book/exercises/	Exercises from the book
http://dotnet.jku.at/buch/solutions/	Solutions to the exercises