



Dynamic Web Pages with ASP.NET

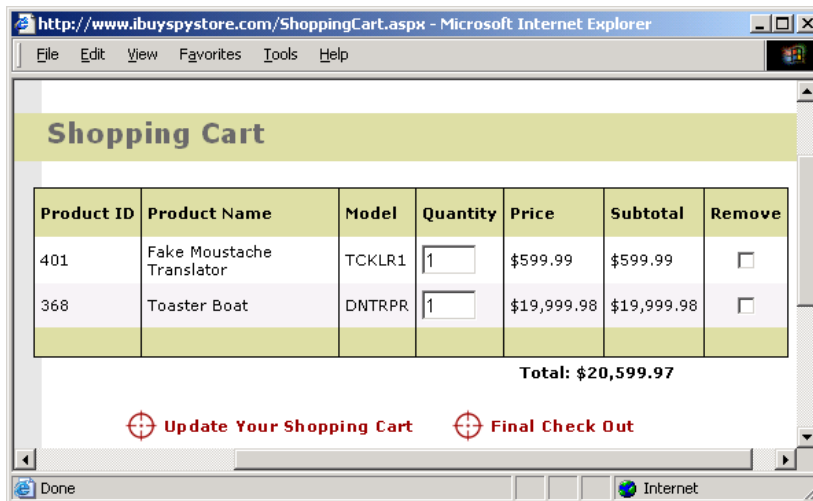
Hanspeter Mössenböck



Main Features of ASP.NET



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



- object-oriented
- event-based
- rich library of Web Controls
- Separation of layout (HTML) and logic (e.g. C#)
- compiled languages instead of interpreted languages
- GUI can be composed interactively with Visual Studio .NET
- better state management
- ...

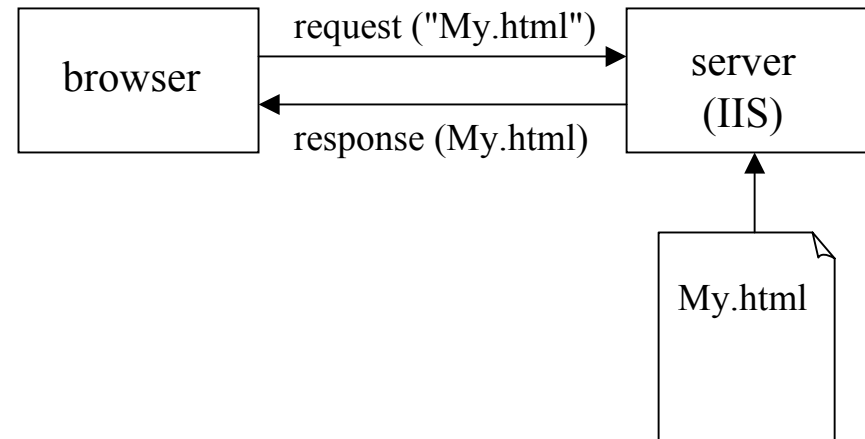
Static Web Pages



Pure HTML

My.html

```
<html>
  <head>
    <title>Static Web Page</title>
  </head>
  <body>
    <h1>Welcome</h1>
    You are visitor number 1!
  </body>
</html>
```

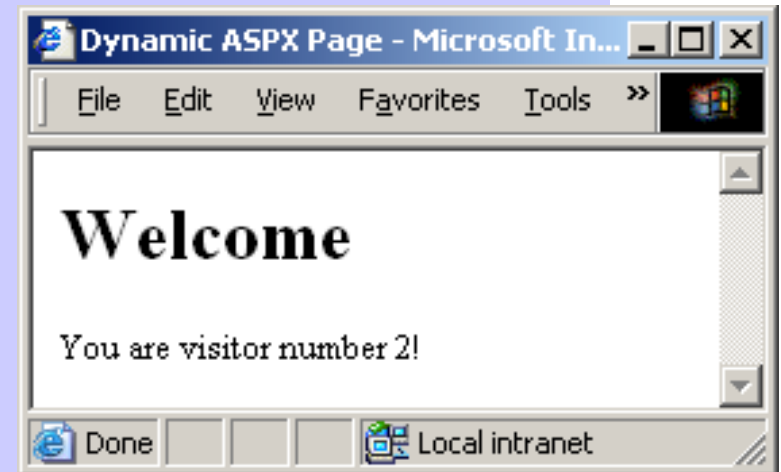


Dynamic ASPX Pages



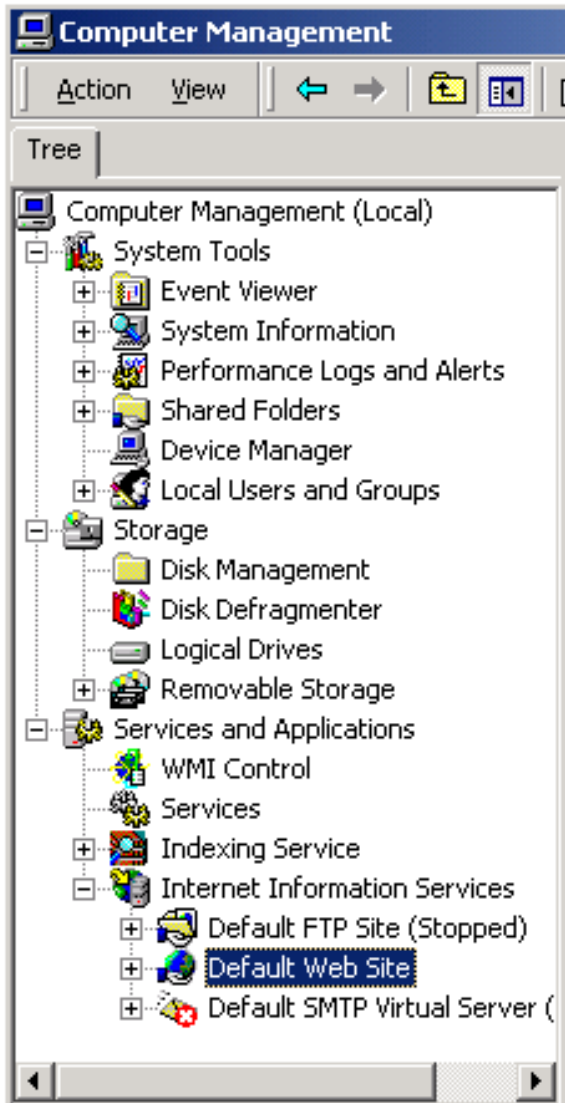
Counter.aspx

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.IO" %>
<html>
  <head> <title>Dynamic ASPX Page</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; } // in case 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



Control Panel

- > Administrative Tools
- > Computer Management

Right-click on Default Web Site

- > New ... Virtual Directory

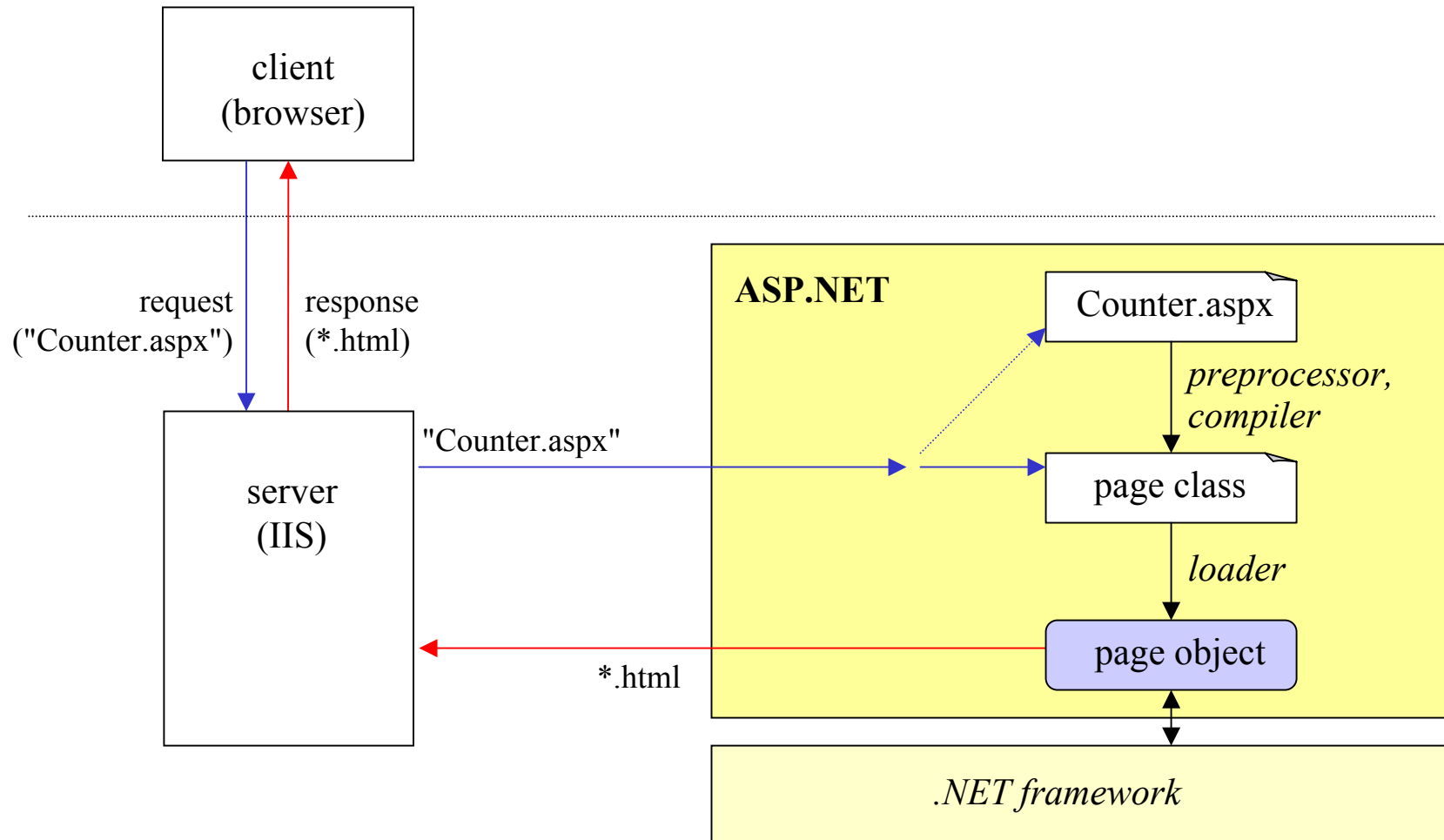
follow the dialog

All aspx files must be in a virtual directory

accessible as

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

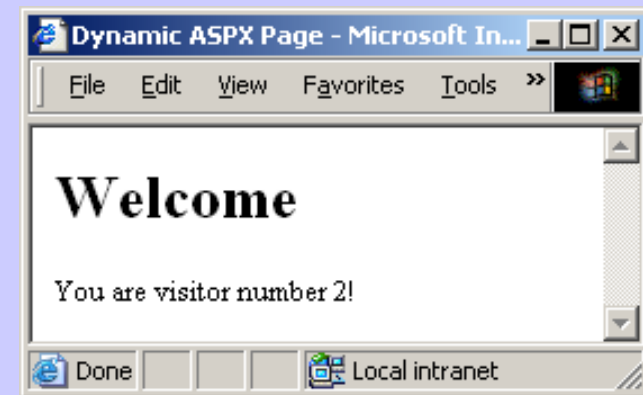
What Happens Behind the Scene?



Code in Script Tags

Counter.aspx

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.IO" %>
<html>
  <head>
    <title>Dynamic ASPX Page</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>
```



Code Behind



Counter.aspx

```
<%@ Page Language="C#" Inherits="CounterPage" Src="CounterPage.cs" %>
<html>
  <head> <title>Dynamic ASPX Page</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

code behind

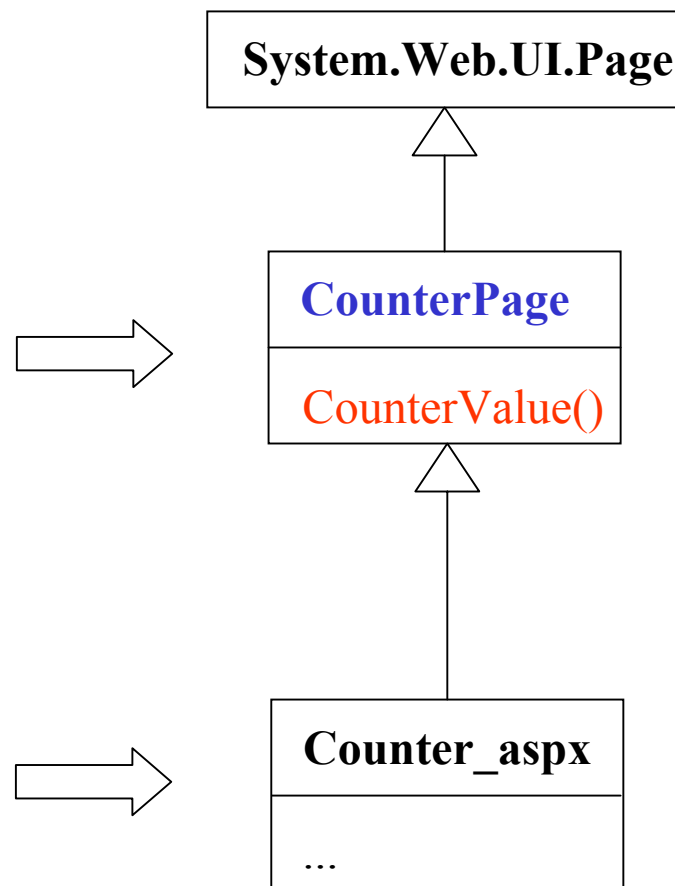
CounterPage.cs

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

aspx page

Counter.aspx

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



HTML Forms



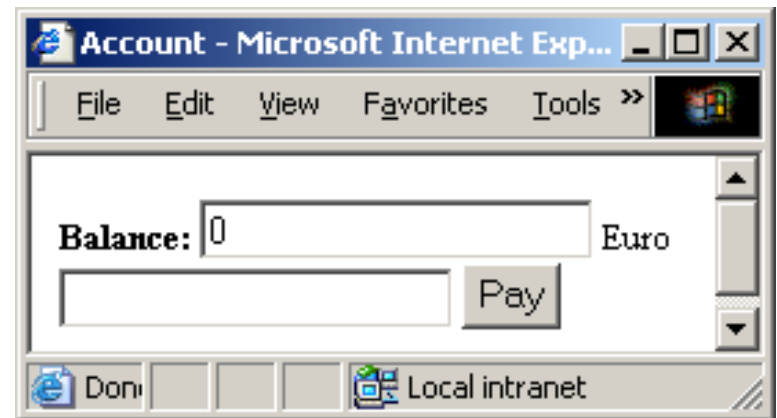
```
...  
<body>  
  <form action="http://www.fake.com/cgi-bin/myprog" method="post">  
    <b>Balance:</b>  
    <input type="text" name="total" readonly value="0"> Euro<br>  
    <input type="text" name="amount">  
    <input type="submit" name="ok" value="Pay">  
  </form>  
</body>
```

CGI program *myprog*

- reads *total* and *amount*
- returns new HTML text in which *total* and *amount* have new values

Problems

- CGI programming is tricky
- restricted to HTML elements
- must set the state of the text field before the page is returned



Web Forms under ASP.NET



```
<%@ Page Language="C#" Inherits="AdderPage" Src="Adder.aspx.cs"%>
```

Adder.aspx

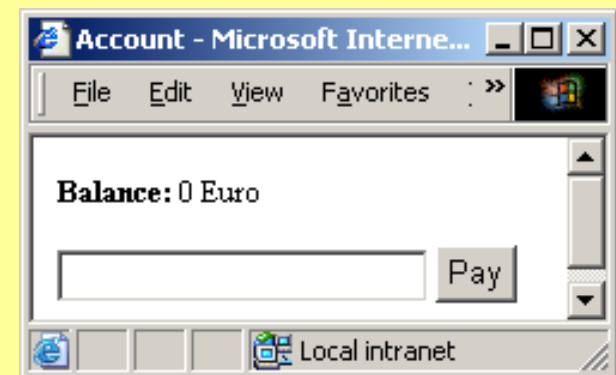
```
<html>
  <head><title>Account</title></head>
  <body>
    <form method="post" Runat="server">
      <b>Balance:</b>
      <asp:Label ID="total" Text="0" Runat="server"/> Euro<br><br>
      <asp:TextBox ID="amount" Runat="server"/>
      <asp:Button ID="ok" Text="Pay" OnClick="ButtonClick" Runat="server" />
    </form>
  </body>
</html>
```

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

Adder.aspx.cs

```
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();
  }
}
```



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 returned

Web Controls



Label

abc

TextBox

Button

RadioButton

☐ Radio

CheckBox

☐ Check

DropDownList

▼

ListBox

- apples
- pears
- bananas

Calendar

< Februar 2003 >						
Mo	Di	Mi	Do	Fr	Sa	So
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	1	2
3	4	5	6	7	8	9

DataGrid

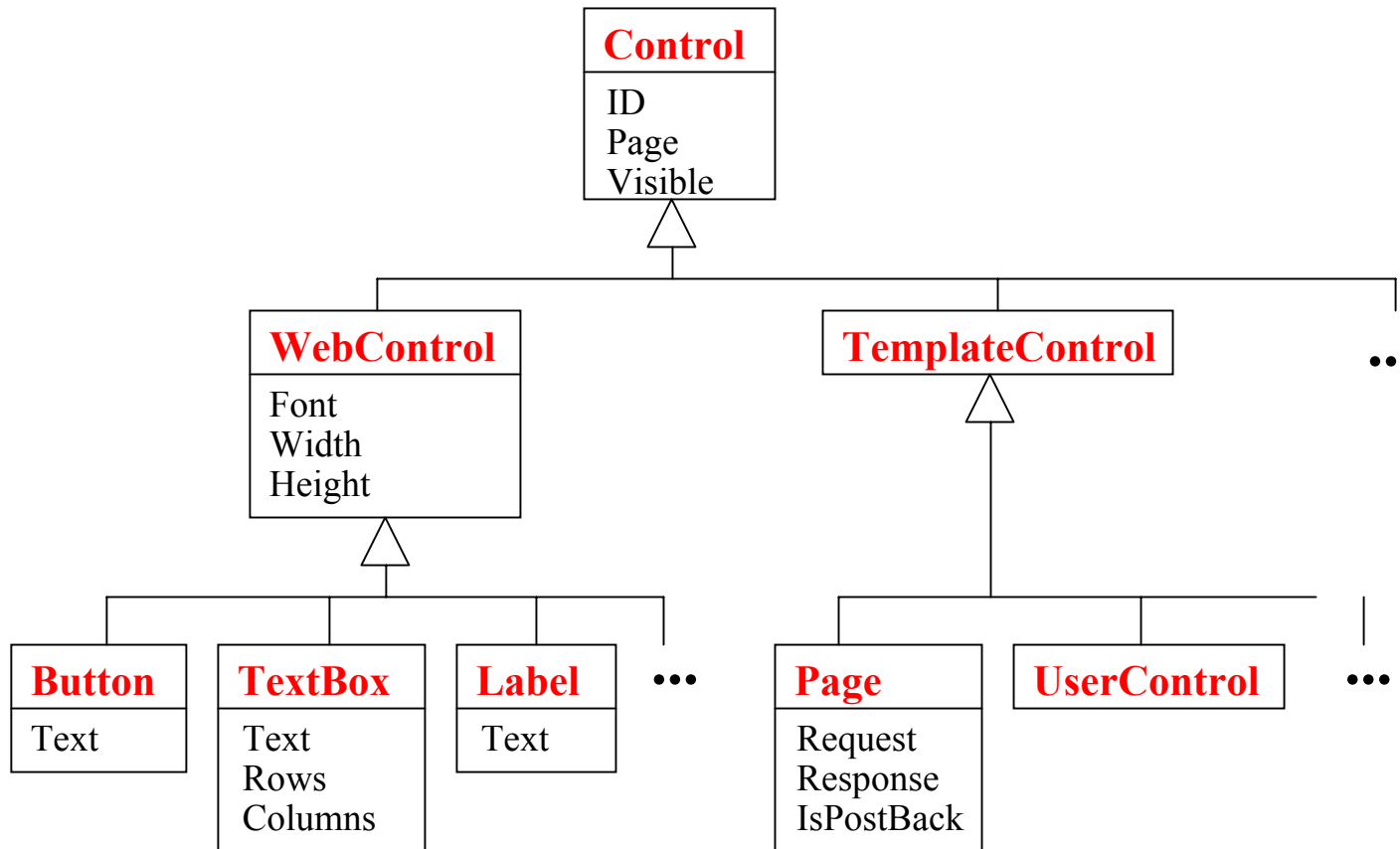
EmployeeID	FirstName	LastName
1	Nancy	Davolio
2	Andrew	Fuller
3	Janet	Leverling
4	Margaret	Peacock
5	Steven	Buchanan
6	Michael	Suyama
7	Robert	King
8	Laura	Callahan
9	Anne	Dodsworth

...

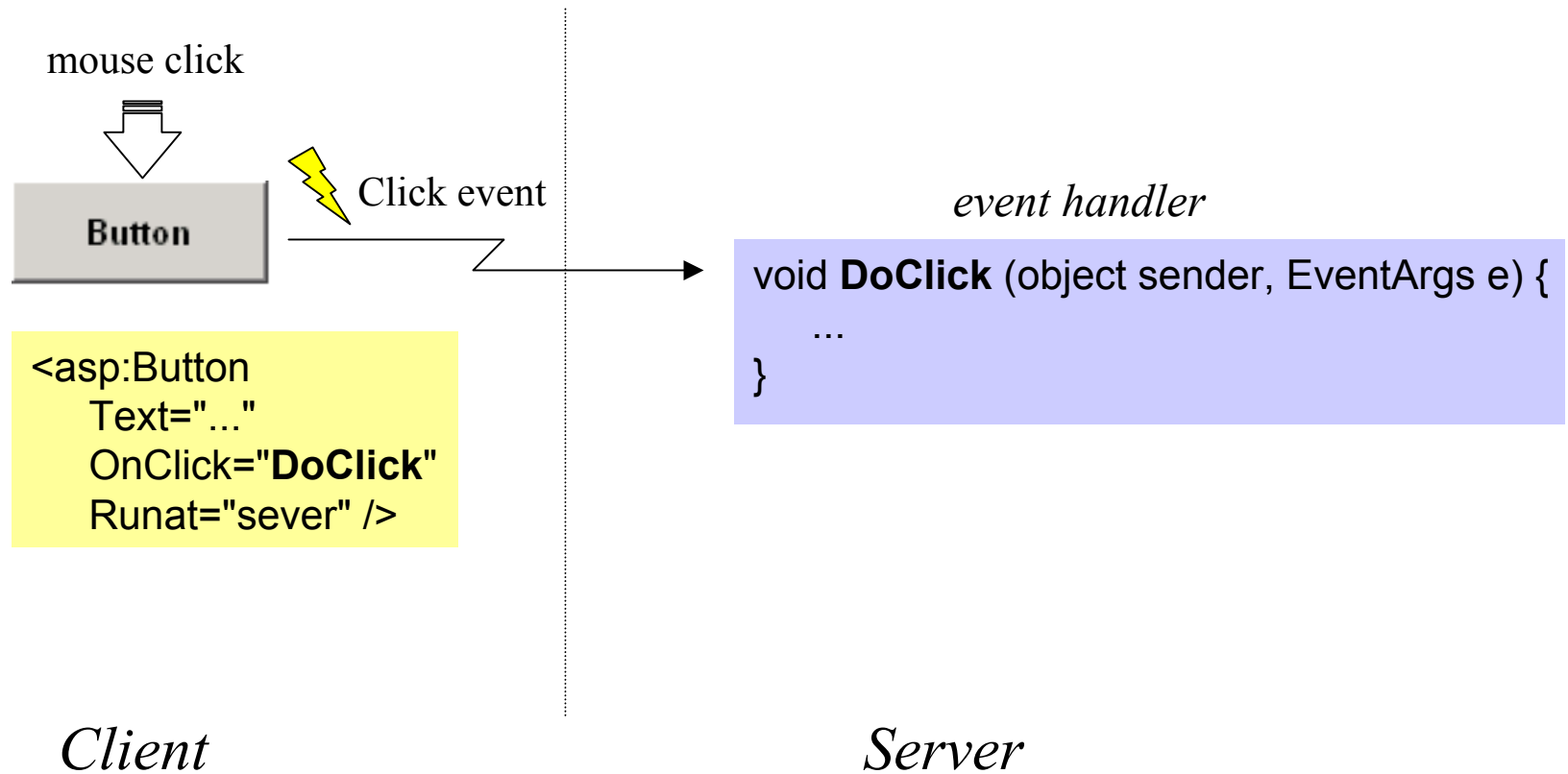
User Controls

Custom Controls

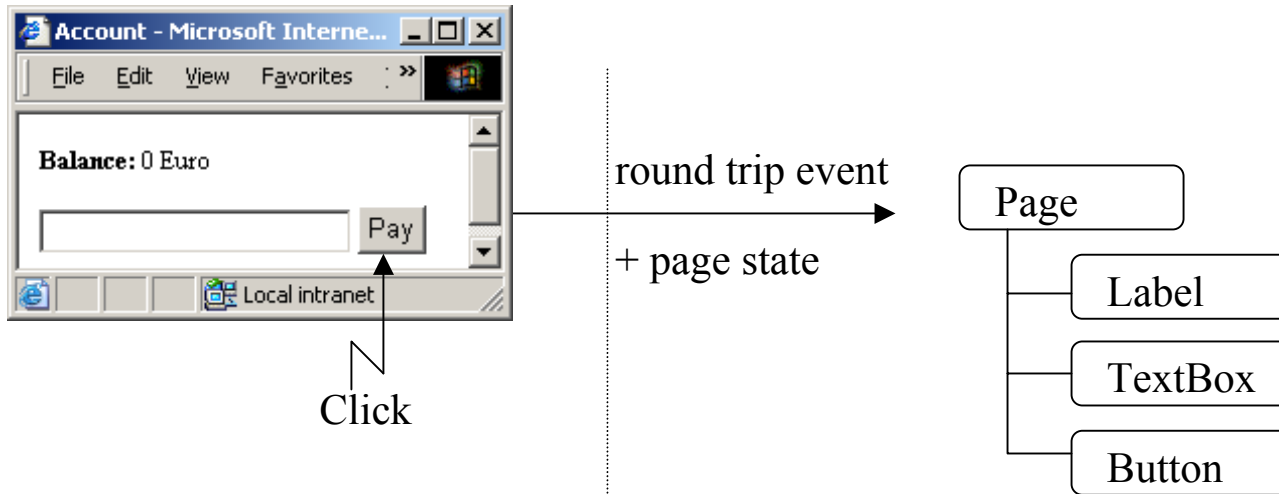
Web Control Hierarchy



Event-based Processing



Round Trip of a Web Page



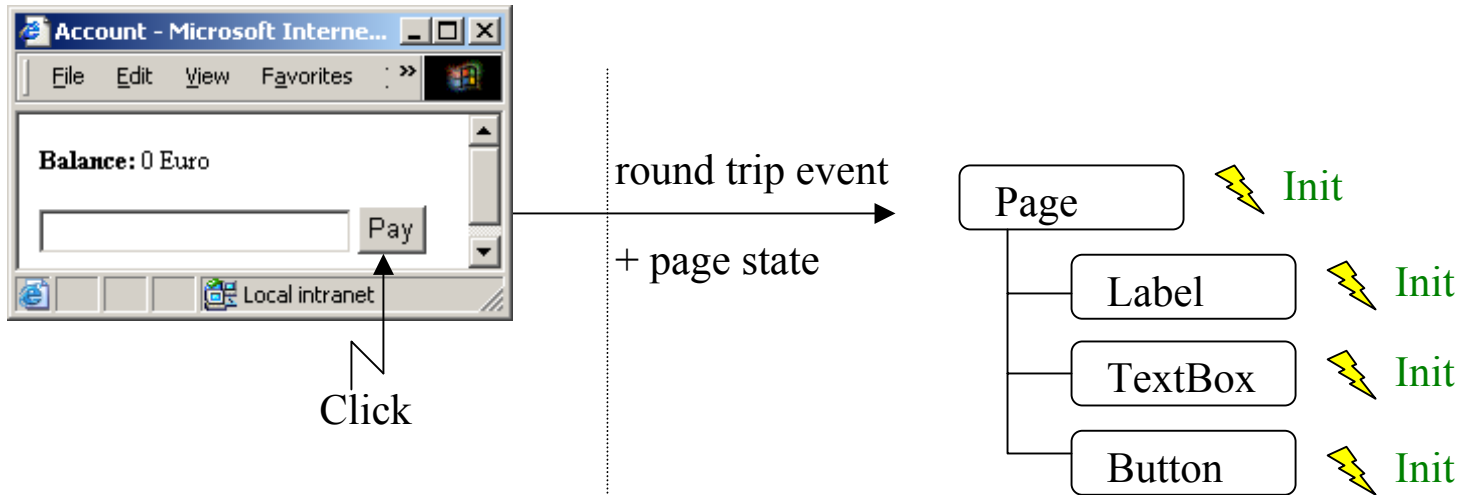
1. Creation

create page object and GUI objects

Client

Server

Round Trip of a Web Page



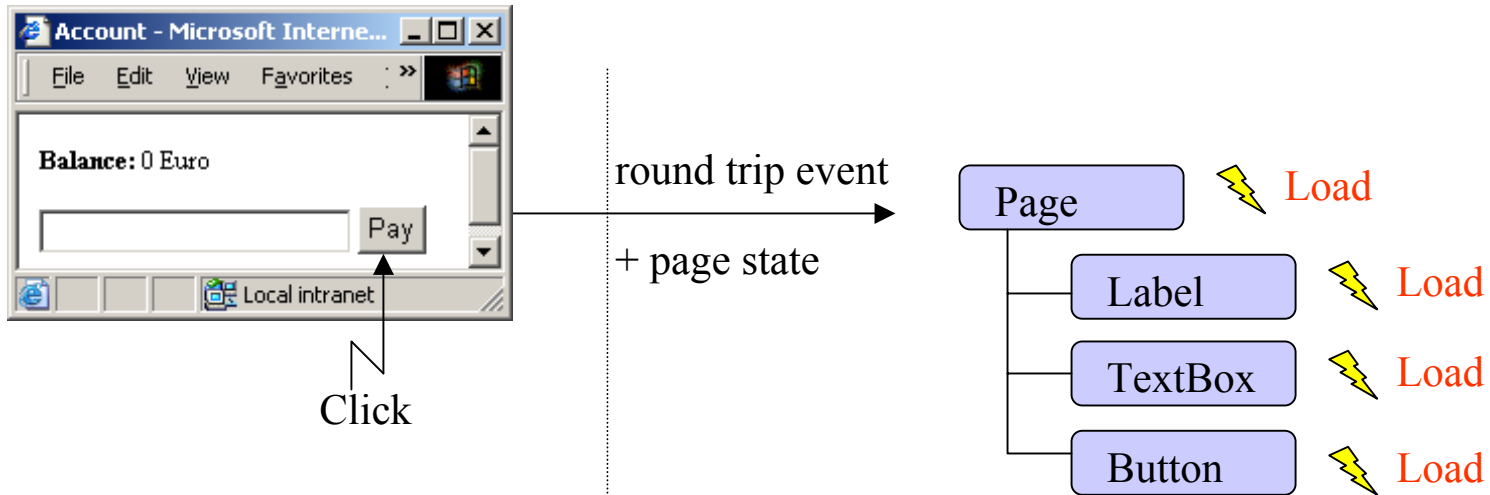
2. Initialisation

- raise *Init* events

Client

Server

Round Trip of a Web Page



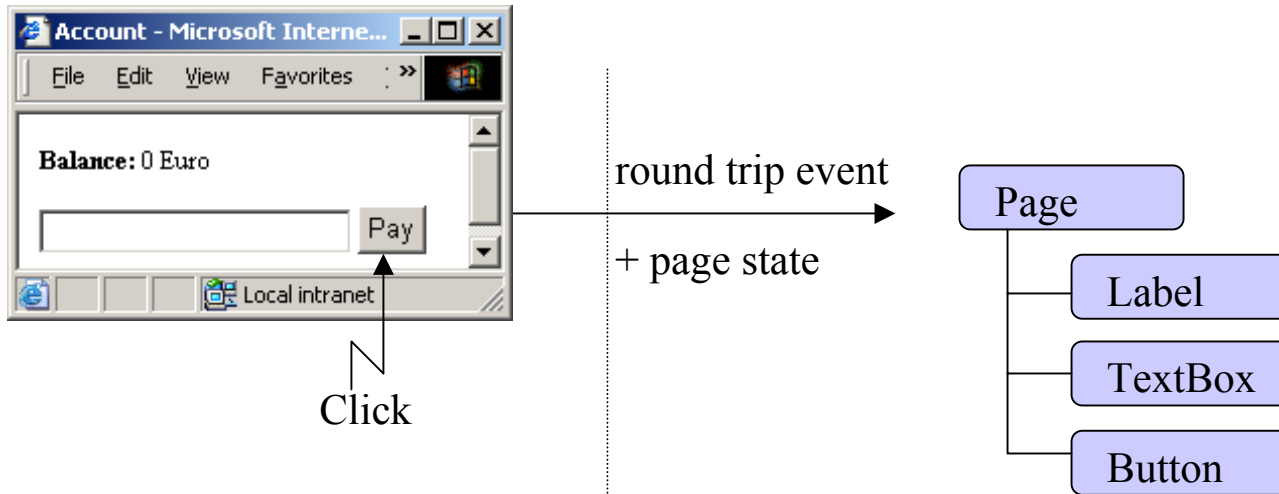
3. Loading

- load GUI objects with the values that the user has entered (page state)
- raise *Load* events

Client

Server

Round Trip of a Web Page



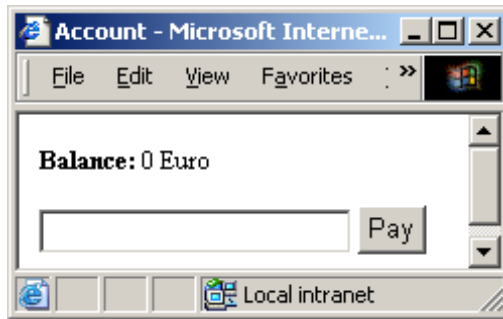
4. Action

handle round trip event(s)
(*Click*, *TextChanged*, ...)

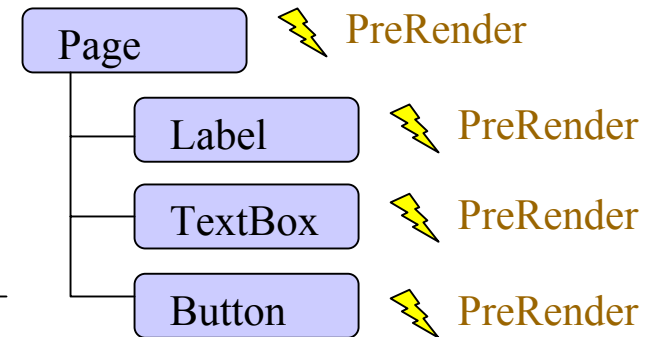
Client

Server

Round Trip of a Web Page



round trip event
+ page state



HTML

+ page state

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

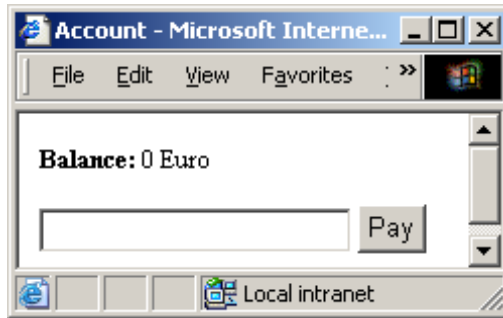
5. Rendering

- raise *PreRender* events
- call *Render* methods of all GUI objects, which render them to HTML code

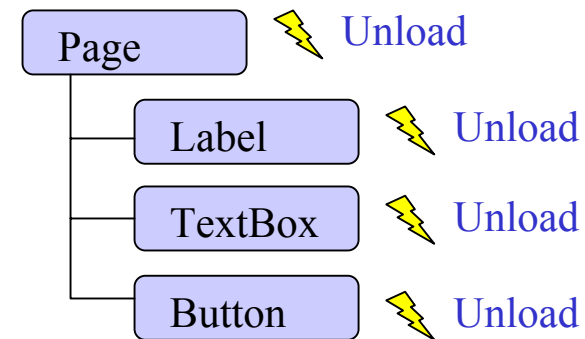
Client

Server

Round Trip of a Web Page



round trip event
+ page state



```

<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

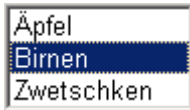
```
<asp:Button Text="click me" Runat="server"
  OnClick="DoClick" />
```

Delayed Events (are only 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)



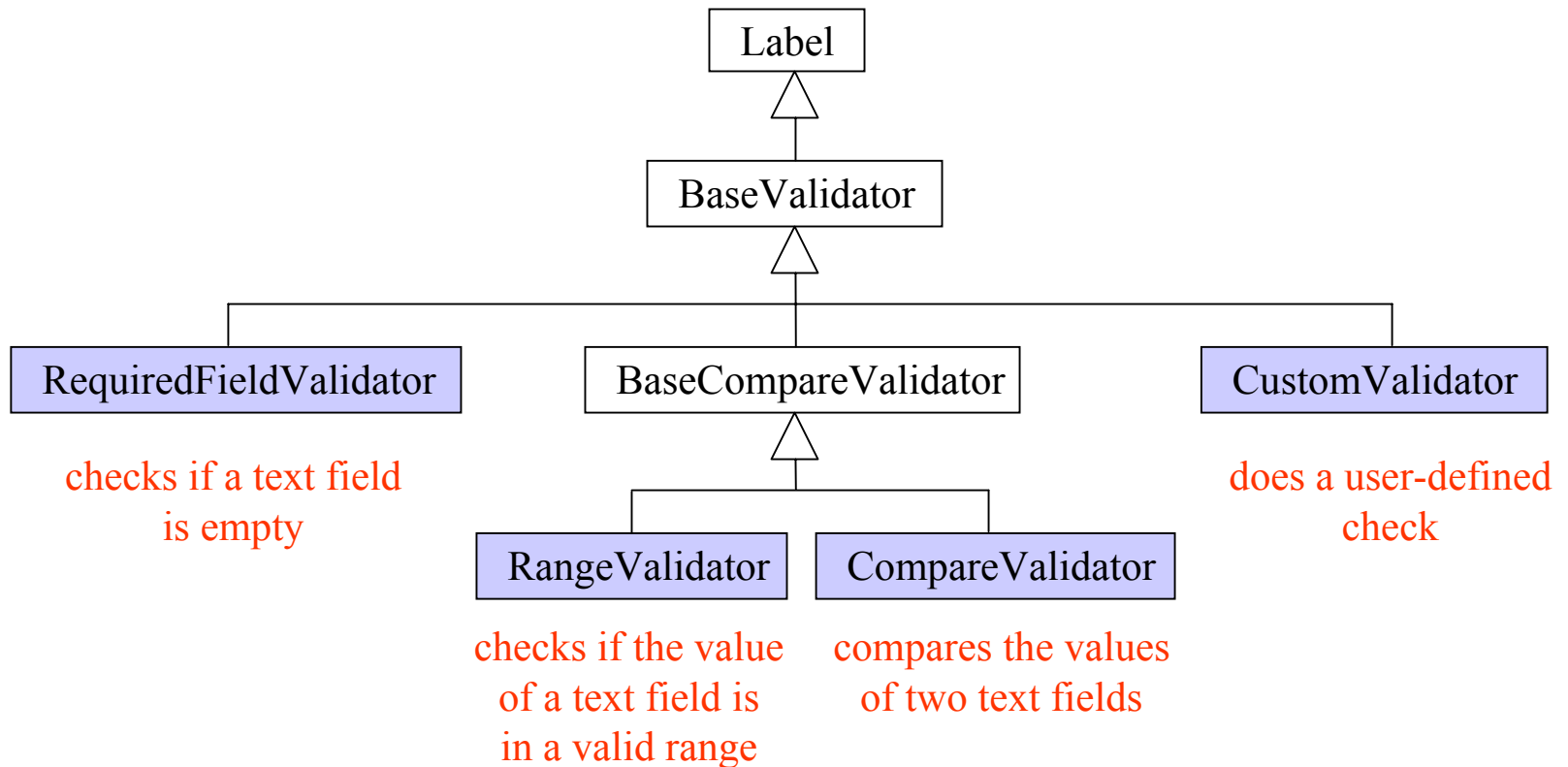
TextChanged

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

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" />
```

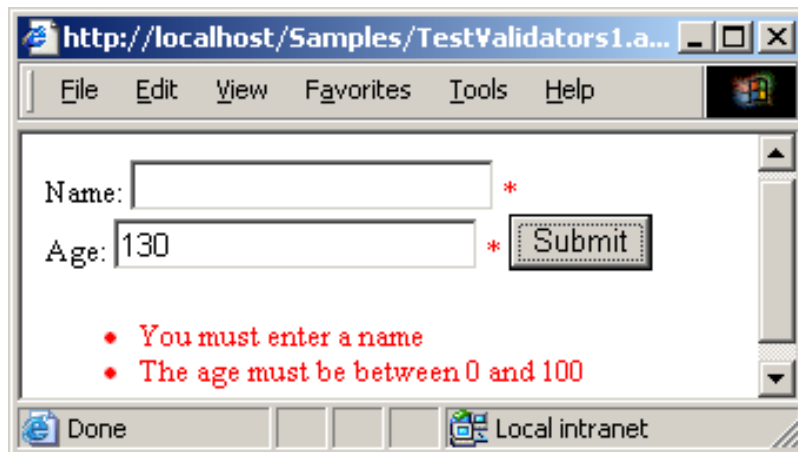

Age:

```
<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" />
```



User Controls

Group of Web Controls, that can be used like a single element



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;

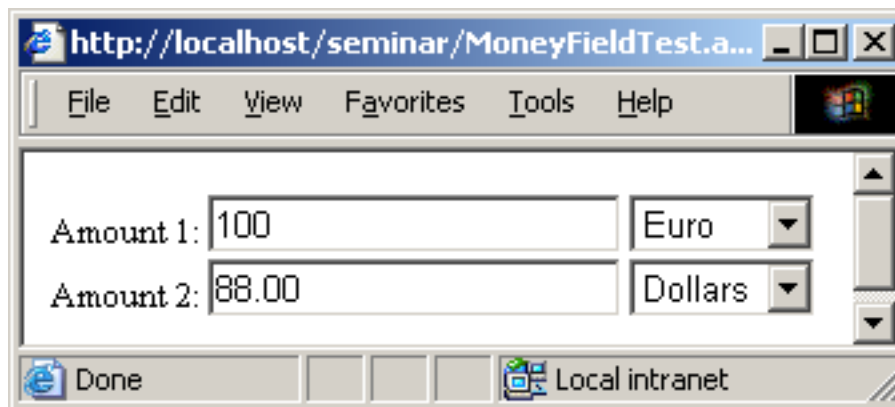
    public void Select(object sender, EventArgs arg) {
        ...
    }
}
```

User Controls

Can be used multiple times per page

MoneyFieldTest.aspx

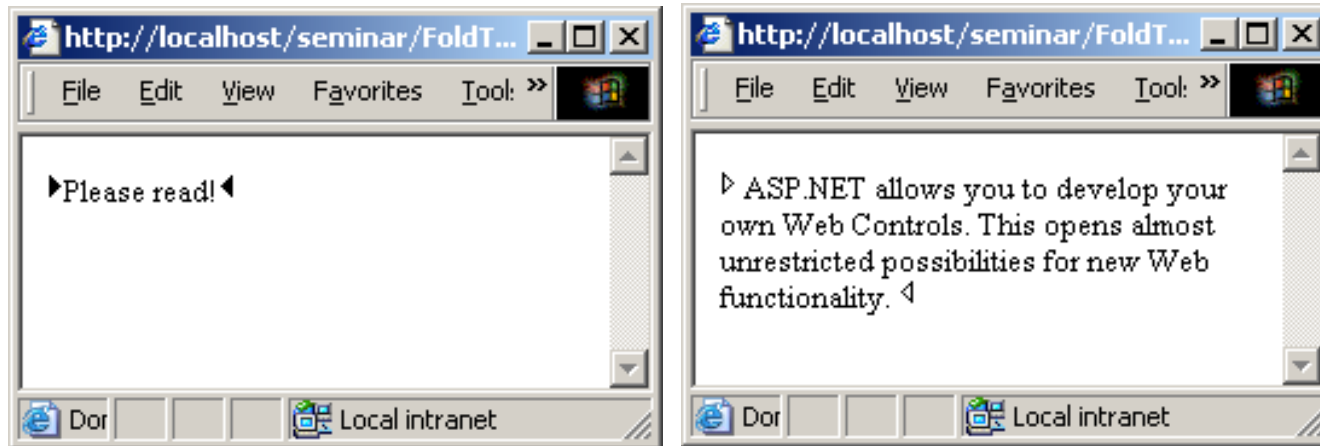
```
<%@ Page Language="C#" %>
<%@ Register TagPrefix="my" TagName="MoneyField" Src="MoneyField.ascx" %>
<html>
<body>
  <form Runat="server">
    Amount 1: <my:MoneyField ID="field1" Runat="server" /><br>
    Amoun 2: <my:MoneyField ID="field2" Runat="server" />
  </form>
</body>
</html>>
```



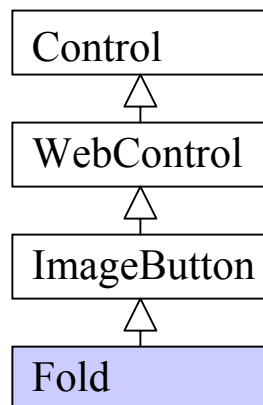
Custom Controls



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



Implemented as an (indirect) subclass of *Control*



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

Custom Controls

Class Fold

```
using System.Web.UI;
using System.Web.UI.WebControls;
namespace Folds {
    public class Fold : ImageButton {
        ...
        protected override void Render(HtmlTextWriter w) { ... }
    }
}
```

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

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

Used as follows

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

State Management



Page state

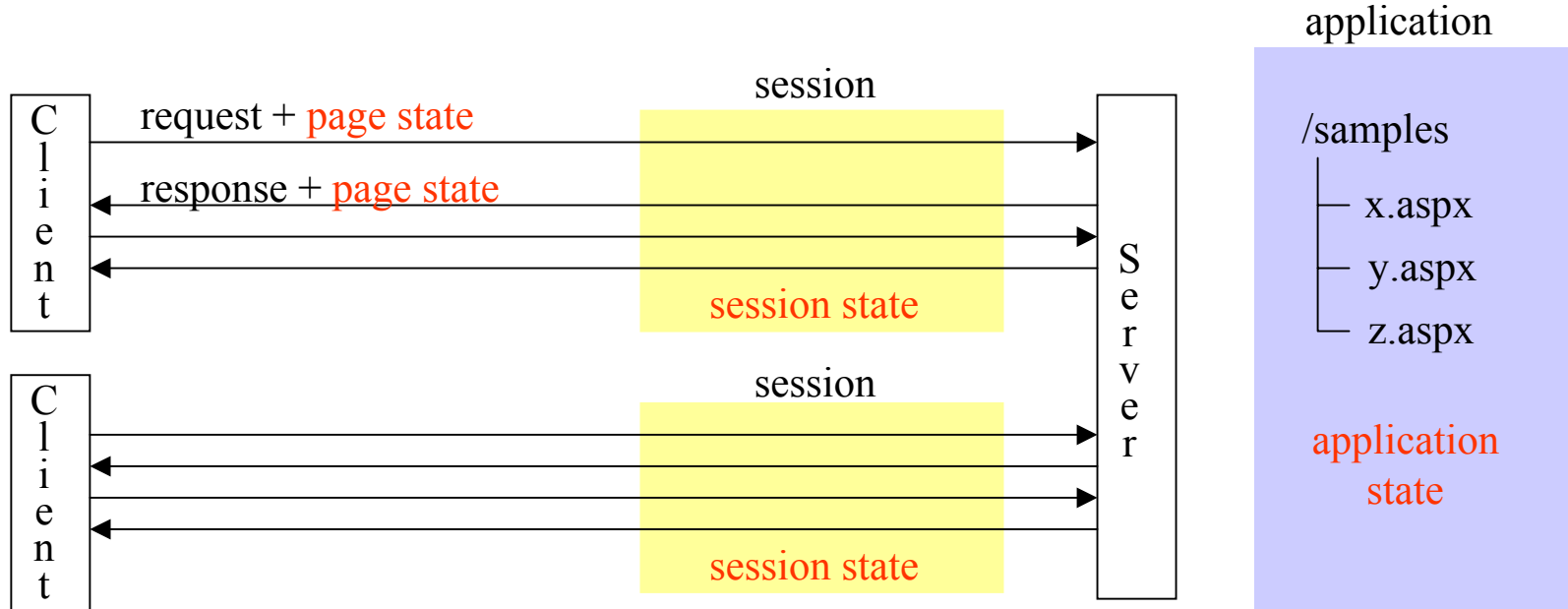
e.g. contents of text fields, state of check boxes, ...

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, ...



Accessing State Information

Page state

setting: `ViewState["counter"] = counterVal;`
getting: `int counterVal = (int) ViewState["counter"];`

Session state

setting: `Session["cart"] = shoppingCart;`
getting: `DataTable shoppingCart = (DataTable) Session["cart"];`

Application state

setting: `Application["database"] = databaseName;`
getting: `string databaseName = (string) Application["databaseName"];`

Summary



Advantages of ASP.NET over ASP

- object-oriented
- event-based
- prefabricated controls and custom controls
- separation between layout (HTML) and logic (e.g. C#)
- compiled code instead of interpreted code
- simple and powerful state management