

Dynamische Webseiten mit ASP.NET Teil 2



- Eigene Steuerelemente
- Zustandsverwaltung und Konfiguration
- Request-Response Klasse
- Navigation
- Authentisierung, Membership & Profile
- Einheitliches Aussehen
- Einheitliche Struktur
- AJAX Erweiterung
- SOAP

Eigene Steuerelemente

- Reichhaltige Bibliothek von Steuerelementen von Microsoft
 - wenn das nicht reicht -> von anderen Herstellern
 - Entwicklung von eigenen neuen Steuerelementen
- Erweiterbarkeit der Steuerelementbibliothek
- Zentrale Eigenschaft moderner Web GUI Frameworks:
- Neue Steuerelemente können:
 - 1. mittels Aggregation (Zusammenfügen) bestehender oder
 - 2. von Grund auf neu erstellt werden
- Als neue, eigenständige Steuerelemente verwendet werden.
- Diese Elemente können verteilt und in anderen Projekten eingesetzt werden
- ASP.NET zwei Mechanismen die Bibliothek von Steuerelementen zu erweitern
 - 1. beliebige Web Controls können zu neuen **zusammengefügt** werden: *User Controls*
 - 2. neue Steuerelemente können von **Grund auf** neu erstellt werden: *Custom Controls*

User Controls

Zusammengesetzte Steuerelemente

User Controls (Beispiel)

- Gruppe von Steuerelementen, die wie ein Element verwendet werden kann



- In **ascx**-Datei beschrieben (z.B. MoneyField.ascx)

```
<%@ Control Inherits="MoneyFieldBase" CodeFile="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="Dollar" Value="0.88" />
    <asp:ListItem Text="Franken" Value="1.47" />
    <asp:ListItem Text="Pfund" Value="0.62" />
</asp:DropDownList>
```

User Controls (Hintergrundcode)

MoneyField.ascx.cs

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

public partial class MoneyFieldBase : UserControl {

    public string Text {
        get { return amount.Text; }
        set { amount.Text = value; }
    }

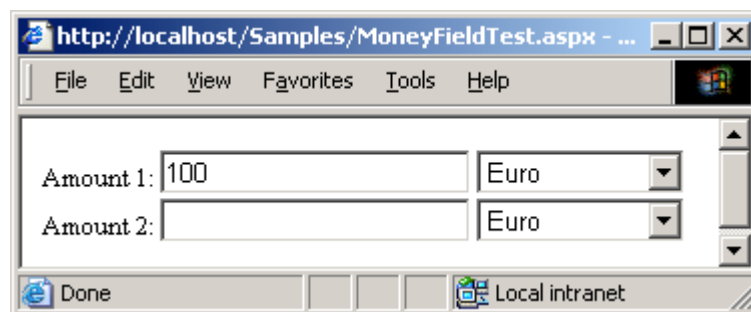
    public double OldFactor {
        get { return ViewState["factor"] == null ? 1 : (double)ViewState["factor"]; }
        set { ViewState["factor"] = value; }
    }

    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 (Verwendung)

- Können mehrmals pro Seite verwendet werden

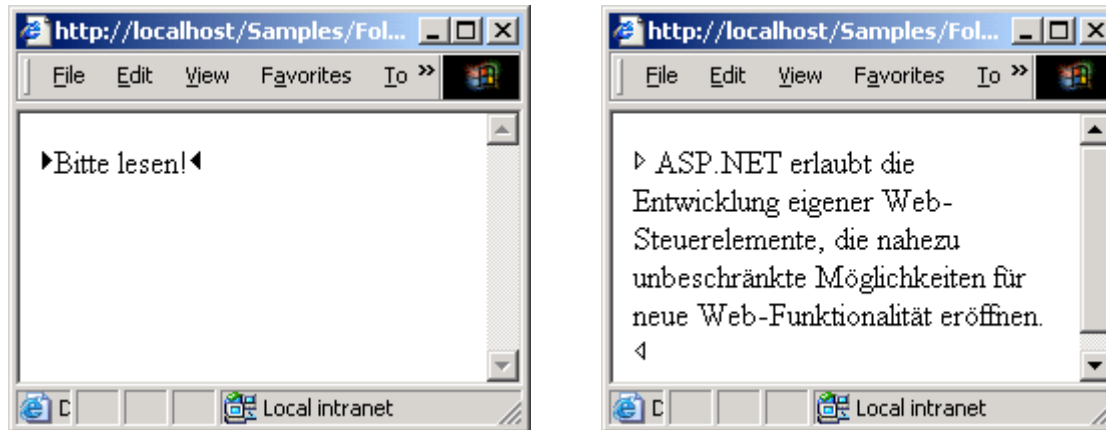
```
<%@ Page Language="C#" %>
<%@ Register TagPrefix="my" TagName="MoneyField" Src="MoneyField.ascx" %>
<html>
<body>
    <form Runat="server">
        Amount 1: <my:MoneyField ID="field1" Text="100" Runat="server" /><br>
        Amount 2: <my:MoneyField ID="field2" Runat="server" />
    </form>
</body>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



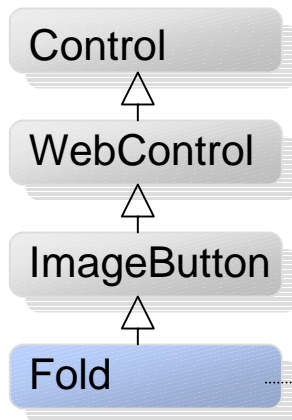
Custom Controls Neue Steuerelemente

Custom Controls (Beispiel)

- Erlauben völlig neue Funktionalität (z.B. Textfaltung)



- Als Unterklasse von Control oder einer ihrer Unterklassen implementiert



Vordergrund-Text: Fold.Text
Hintergrund-Text: ImageButton.AlternateText
Click-Event: von ImageButton geerbt

Muss Render-Methode haben,
die Element nach HTML abbildet

Custom Controls (Beispielklasse Fold)

```
using System; using System.Web.UI; using System.Web.UI.WebControls;
namespace Folds { // custom controls must be declared in a new namespace
    public class Fold : ImageButton {
        public string AlternateText{
            get { return ViewState["AlternateText"]!=null ? "" : (string)ViewState["AlternateText"]; }
            set { ViewState["AlternateText"] = value; }
        }
        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
            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='" + Icon + "Left.gif' border=0 />");
            w.Write(Text);
            w.Write("<img src='" + Icon + "Right.gif'>");
        }
    }
}
```

- ▶ SolidLeft.gif
- ◀ SolidRight.gif
- ▷ HollowLeft.gif
- ◁ HollowRight.gif

Custom Controls (Verwendung)

- Muss in DLL übersetzt werden, die in bin-Verzeichnis steht

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

- Wird wie folgt verwendet

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

Zustandsverwaltung und Konfiguration

Zustandsverwaltung

■ Seitenzustand

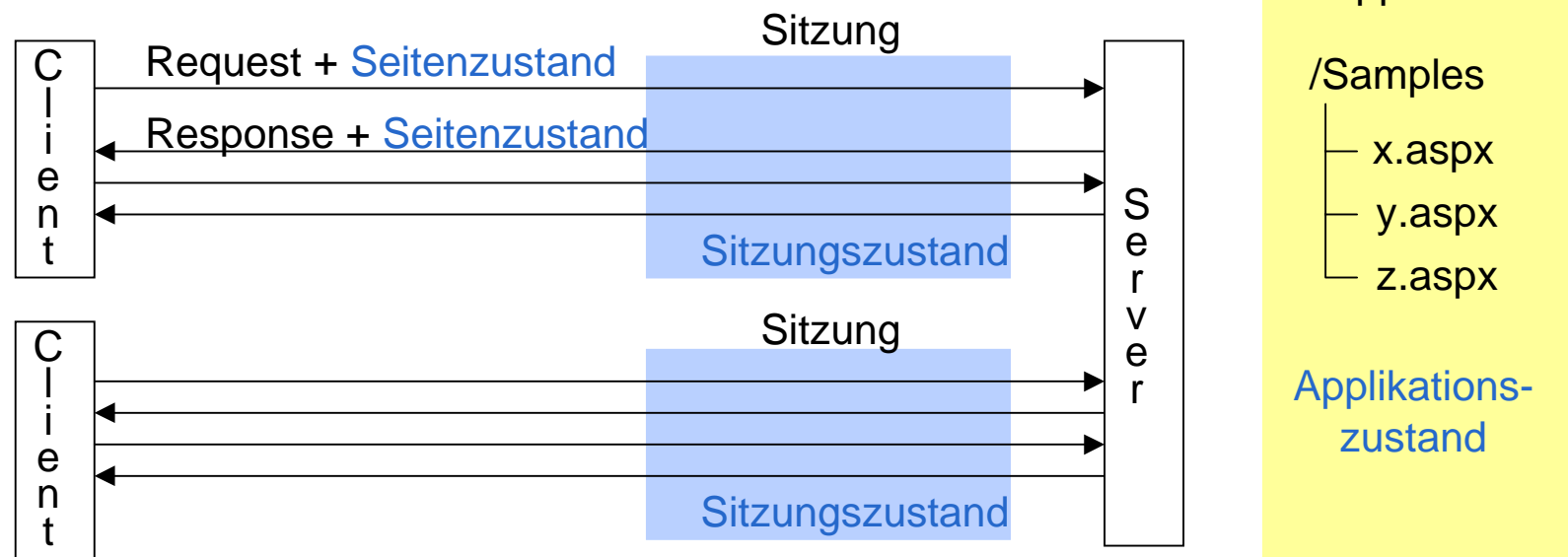
z.B. Inhalt von Textfeldern, Zustand von Checkboxes, ...

■ Sitzungszustand (**Sitzung** = alle Requests vom gleichen Klient innerhalb bestimmter Zeit)

z.B. Warenkorb, Email-Adresse des Kunden, ...

■ Applikationszustand (**Applikation** = alle aspx-Dateien in einem virtuellen Verzeichnis)

z.B. Konfigurationsdaten, Anzahl der Sitzungen, ...



Zugriff auf Zustandsinformationen

■ Seitzenzustand

Schreiben: `ViewState["counter"] = counterVal;`

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

■ Sitzungszustand

Schreiben: `Session["cart"] = shoppingCart;`

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

■ Applikationszustand

Schreiben: `Application["database"] = databaseName;`

Lesen: `string databaseName = (string) Application["databaseName"];`

```
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; }  
    protected virtual StateBag ViewState { get; }  
    public HttpRequest Request { get; }  
    public HttpResponse Response { get; }  
    public string Theme {get;set;}  
    ...  
    ///--- methods  
    public string MapPath(string virtualPath);  
    public virtual void Validate();  
    ...  
}
```

ViewState von
eigenem Control
verwenden

MapPath(virtPath)

bildet virtuelles Verzeichnis auf physisches ab

Validate()

stösst alle Validatoren der Seite (nochmals) an

IsValid

true, wenn keiner der Validatoren
auf der Seiten einen Fehler meldet

IsPostBack

true, wenn Seite auf Grund einer
Rundreise angefordert wurde. Beim
erstmaligen Anfordern der Seite ist
IsPostBack == false

TemplateSourceDirectory

aktuelles virtuelles Verzeichnis,
z.B. "/Samples"

Application und Session

Applikationszustand und
Sitzungszustand

Request und Response

HTTP-Seitenanforderung und
HTTP-Antwort

Initialisierung einer Seite

- Die **Page_Load** Methode wird automatisch aufgerufen
- Seitenspezifische Initialisierungen können dort durchgeführt werden
- Mittels **isPostBack** kann abgefragt werden, ob es sich um den ersten Aufruf dieser Seite in der aktuellen Session handelt

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!this.IsPostBack) {
        this.TextBox1.Text = "Mary";
    }
}
```


Initialisierung einer Applikation: Global.asax

- Eine ASP.NET-Applikation besteht aus mehreren ASPX Seiten
- Zusätzlich Global.asax um für alle Seiten der Applikation gemeinsame Ereignisse zu behandeln und Zustände zu verwalten

Global.asax

```
<%@ Application Inherits="Global" CodeFile="Global.asax.cs" %>
```

Application
und Session Property
mit Page gemeinsam

Global.asax.cs

```
public partial class Global : System.Web.HttpApplication {  
    public Global() {InitializeComponent();}  
  
    protected void Application_Start(Object sender, EventArgs e) {...}  
    protected void Application_End(Object sender, EventArgs e) {...}  
  
    protected void Session_Start(Object sender, EventArgs e){...}  
    protected void Session_End(Object sender, EventArgs e){...}  
  
    protected void Application_BeginRequest(Object sender, EventArgs e) {...}  
    protected void Application_EndRequest(Object sender, EventArgs e){...}  
}
```

Anwendung

Sitzung

Aufruf

- Definiere Skript Mapping, das in einigen Controls verwendet wird.

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

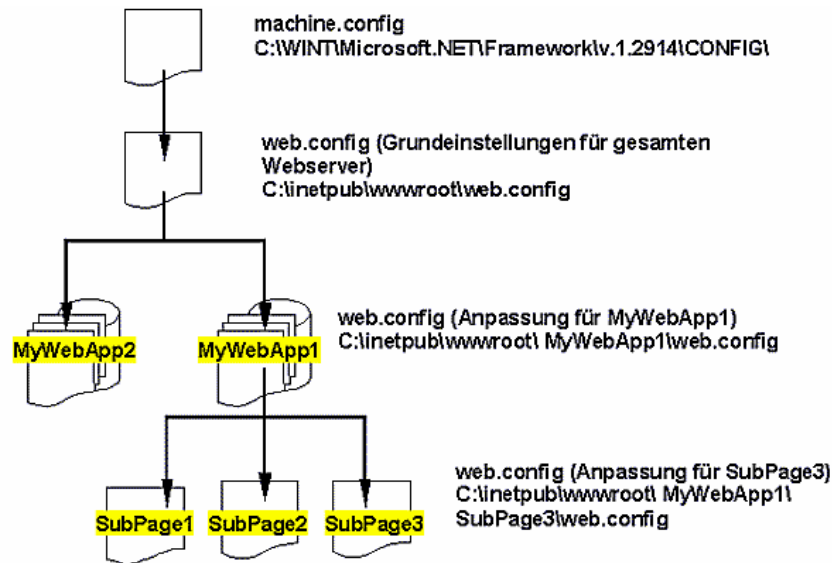
<script runat="server">

static string _pagePath;

void Application_Start(object sender, EventArgs e)
{
    ScriptManager.ScriptResourceMapping
        .AddDefinition("jquery",new ScriptResourceDefinition
    {
        Path = "~/scripts/jquery-1.7.2.min.js",
        DebugPath = "~/scripts/jquery-1.7.2.min.js",
        CdnPath = "http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.4.1.min.js",
        CdnDebugPath = "http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.4.1.js"
    });
}

</script>
```

machine.config und web.config



<http://www.aspheute.com/artikel/20010802.htm>

machine.config

- Systemweite Konfigurationsdatei
- Steht im .NET-Frameworkverzeichnis

web.config

- Spezifische Konfigurationsdatei
- Kann in jedem virtuellen Verzeichnis oder in Unterverzeichnissen stehen
- Überschreibt Konfigurationen aus machine.config oder aus übergeordneten Verzeichnissen

■ Konfigurationsdateien sind "natürlich" alle in XML Form

Beispiel: Applikationsparameter

web.config

ConfigSections
Handlers
ConfigSections

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <configSections>
    <section name="appSettings" type="System.Config.NameValueFileSectionHandler" ../>
    <configSections>
  </system.web>
  <authorization> <deny users="?" /> </authorization>
  </system.web>
  <appSettings>
    <add key="author" value="hm" />
    <add key="organisation" value="JKU" />
  </appSettings>
  ... ---> weitere Sections siehe Anhang
</configuration>
```

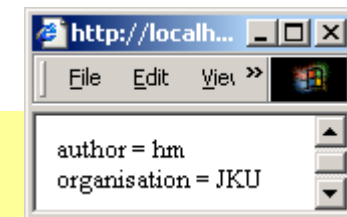
gibt an, welche Klasse für spezifische
Sektion zuständig ist; meist in
machine.config definiert

Einstellungen für Microsoft Klassen

Anwendungsspezifische Werte

■ Kann in Webseiten angesprochen werden

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



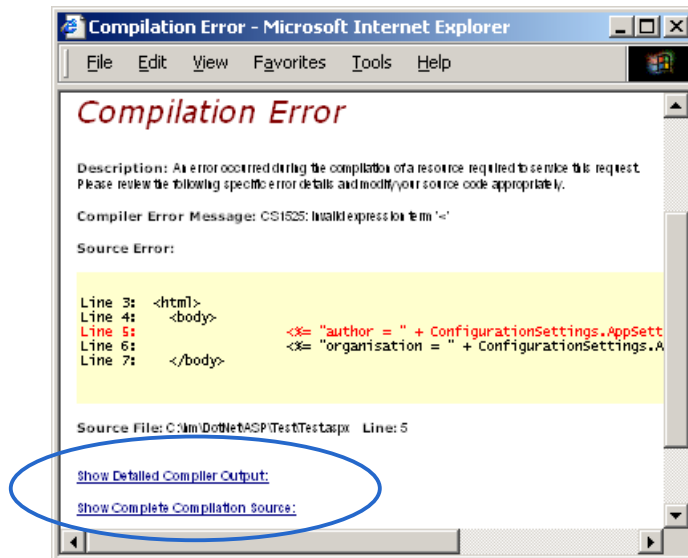
Beispiel: Debug

```
<?xml version="1.0" encoding="UTF-8"?>  
<configuration>  
  <system.web>  
    <compilation debug="true" >/  
    ...  
  </system.web>  
  ...  
</configuration>
```

```
<%@ Page Language="C#" Debug="true" %>
```

auch pro Datei möglich

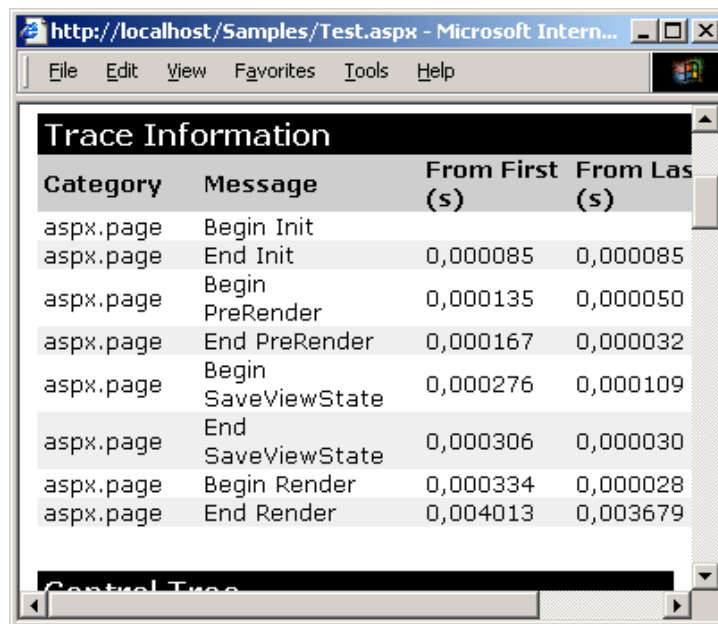
■ Zeigt bei Fehlern genaue Ursache



Beispiel: Tracing

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

■ Zeigt bei korrekten Seiten Trace-Ausgaben (am Schluss)



The screenshot shows a web browser window with the address bar displaying 'http://localhost/Samples/Test.aspx - Microsoft Intern...'. The browser's menu bar includes 'File', 'Edit', 'View', 'Favorites', 'Tools', and 'Help'. The main content area displays 'Trace Information' as a table with four columns: 'Category', 'Message', 'From First (s)', and 'From Last (s)'. The table lists various events for 'aspx.page' including 'Begin Init', 'End Init', 'Begin PreRender', 'End PreRender', 'Begin SaveViewState', 'End SaveViewState', 'Begin Render', and 'End Render', each with corresponding timestamps. Below the table, a 'Control Tree' is partially visible.

Category	Message	From First (s)	From Last (s)
aspx.page	Begin Init		
aspx.page	End Init	0,000085	0,000085
aspx.page	Begin PreRender	0,000135	0,000050
aspx.page	End PreRender	0,000167	0,000032
aspx.page	Begin SaveViewState	0,000276	0,000109
aspx.page	End SaveViewState	0,000306	0,000030
aspx.page	Begin Render	0,000334	0,000028
aspx.page	End Render	0,004013	0,003679

Request-Response

Klasse 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; }  
        public String MapPath (String virtualPath)  
    ...  
}
```

Request Property
in Page

```
<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>
```

■ UserHostName

Domain-Name des Client

■ UserHostAddress

IP-Nummer des Client

■ MapPath

konvertiert virtuellen (Datei-) Pfad in physischen

■ QueryString

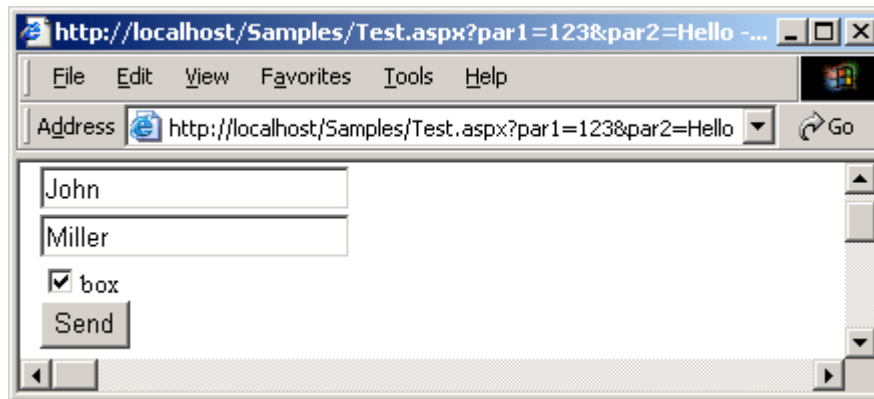
hole mit URL mitgegebene Werte
z.B. ?name="Hugo"&

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


HttpRequest (Request- und Formularparameter)

```
<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

Klasse HttpResponse

Response Property
in Page

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

```
string strFile = request.MapPath("test.gif");  
Stream stmRead = File.OpenRead(strFile);  
int nBufferSize = 255, nReadBytes = 0, nTotalBytes = 0;  
byte[] arrByte = new byte[nBufferSize];  
Response.ClearContent();  
Response.ContentType = "image/gif";  
while (0 != (nReadBytes = stmRead.Read(arrByte, 0 , nBufferSize-1)))  
{  
    nTotalBytes += nReadBytes;  
    if (nReadBytes == nBufferSize)  
        Response.BinaryWrite(arrByte);  
    else {  
        byte[] arrCopy = new byte[nReadBytes];  
        Array.Copy(arrByte, 0, arrCopy, 0, nReadBytes);  
        Response.BinaryWrite(arrCopy);  
    }  
}  
Response.End();
```

ContentType

MIME-Typ (z.B. text/html)

Output

HTML-Rückgabestrom; wird
mit Write beschrieben

StatusCode

z.B. 200 für "ok" oder
404 für "page not found"

direkte Rückgabe eines
Bildes

in Page_Load
Methode

Navigation

Aufruf anderer ASPX Seiten

- Navigation zu anderer ASPX Seite über **Response.Redirect**

- Sendet Browser ein "Redirect"

Vorteil: eine beliebige URL kann angegeben werden

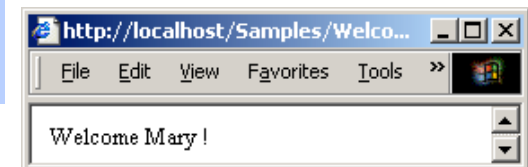
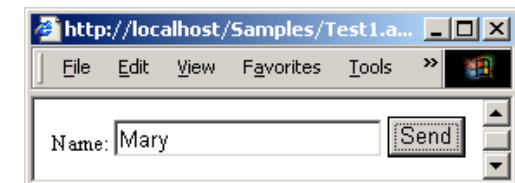
Nachteil: Seitenzustand geht verloren!

- in Session retten
- in URL mitgeben

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"] %> !
```

... Aufruf anderer ASPX Seiten

- Navigation zu anderer ASPX Seite über **Server.Transfer**
- Wird auf dem Server verarbeitet

Nachteil: funktioniert nur für Seiten auf dem gleichen Server

Vorteil: der Seitenzustand der Aufrufer Seite-kann abgefragt werden

Test1.aspx.cs

```
void DoClick (object sender, EventArgs e) {  
    Server.Transfer("Welcome.aspx",true);  
}
```

preserveState= true
um Seitenzustand zu
übertragen

Welcome.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)  
{  
    string s = Request.Form["name"];  
    this.Label1.Text = "Welcome "+s;  
}
```

hole Inhalt der "name"
TextBox

Authentisierung, Membership & Profile

Beispiel: Authorisierung

- Wer darf die Seiten eines bestimmten Verzeichnisses besuchen?
- Das Verzeichnis muss eine *web.config*-Datei der folgenden Art enthalten

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

users="user1,user2,..."

* alle Benutzer

? anonyme Benutzer

name Benutzer, die sich mit diesem Namen
 authentifiziert haben

machine.config enthält

`<allow users="*" />`

Das ist somit Standard, wenn kein `<allow ...>`
angegeben wird

Forms-Authentifizierung (Konfiguration)

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="Clear">
          <user name="kurt" password="kurt123" />
        </credentials>
        <credentials passwordFormat="SHA1">
          <user name="peter" password="328854132BF61A37C6B4A64BE7B23D03B74F8F83" />
        </credentials>
      </forms>
    </authentication>
    ...
  </system.web>
  ...
</configuration>
```

keine anonyme Benutzer erlaubt

oder MD5

```
string encryptedPwd =
FormsAuthentication.HashPasswordForStoringInConfigFile("myPwd", "SHA1");
```

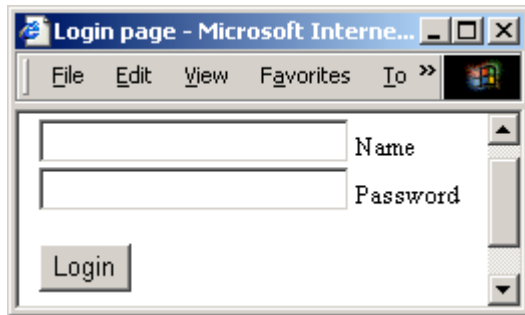
- Die Benutzer "peter" und "kurt" sind mit ihren Passwörtern am Server gespeichert

Authentifizierung Modes

- None** Keine Authentifizierung.
Alle Benutzer sind anonym.
- Windows** Es wird der Login-Name und das Passwort der Windows-Anmeldung benutzt.
Der IIS übernimmt die Anmeldung; Benutzer muss Windows Account verfügen (wird nicht von Cassini unterstützt)
- Forms** Benutzer wird durch ein selbstgeschriebenes Login-Formular authentifiziert.
Benutzer werden über ASP.NET verwaltet
+ gute Integration in .NET; eigene Benutzerverwaltung möglich
- kein Schutz von nicht ASPX Seiten (html, jpeg, ...)

Forms-Authentifizierung (Ablauf)

1. Anonymer Benutzer versucht auf *A.aspx* zuzugreifen
2. Auf Grund von `<deny users="?" />` und `<forms loginUrl="Login.aspx">` wird er auf *Login.aspx* umgeleitet



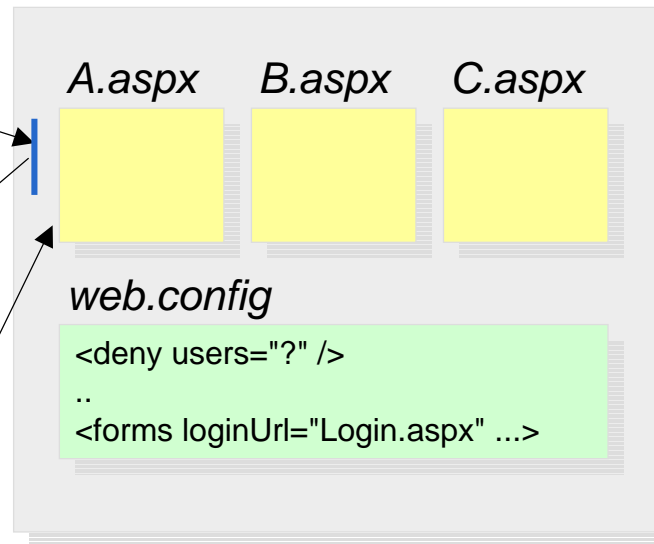
3. Benutzer gibt Name und Passwort ein und wird authentifiziert

nein

4. Authentifizierung erfolgreich?

ja

geschütztes Verzeichnis



5. Benutzer wird auf *A.aspx* weitergeleitet und darf nun auch alle anderen Seiten in diesem Verzeichnis besuchen (weil authentifiziert)

Geschützte Unterseiten

■ Stehen in einem Unterverzeichnis der Applikation (z.B. Members/)

- Application
 - ...
 - Members
 - xxx.aspx
 - yyy.aspx

■ Unterverzeichnis muss eine *web.config*-Datei mit <authorization> Elem. haben

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

Unbekannte Benutzer müssen sich per Login-Seite authentifizieren

Benutzer Peter und Mike werden ohne weiteres Login zu den Member-Seiten durchgelassen

■ Wenn unbekannter Benutzer zu einer Member-Seite kommt, wird er zur Login-Seite umgeleitet

■ Wenn angemeldete Benutzer (oder Peter oder Mike) zu einer Member-Seite kommen, werden sie durchgelassen

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))
          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>
```

überprüft Credentials
in Config Datei

zur ursprünglichen
Seite zurück

Abmelden: Logout.aspx

```
<%@Page Language="C#" %>
<%@ Import Namespace="System.Web.Security" %>

<script language="C#" runat=server>
    void Page_Load(object sender, EventArgs e){
        FormsIdentity Identity = (FormsIdentity) User.Identity;
        lb_identity.Text = Identity.Name;
    }

    void Logout_click(object sender, EventArgs e){
        FormsAuthentication.SignOut();
        Response.Redirect("login.aspx");
    }
</script>

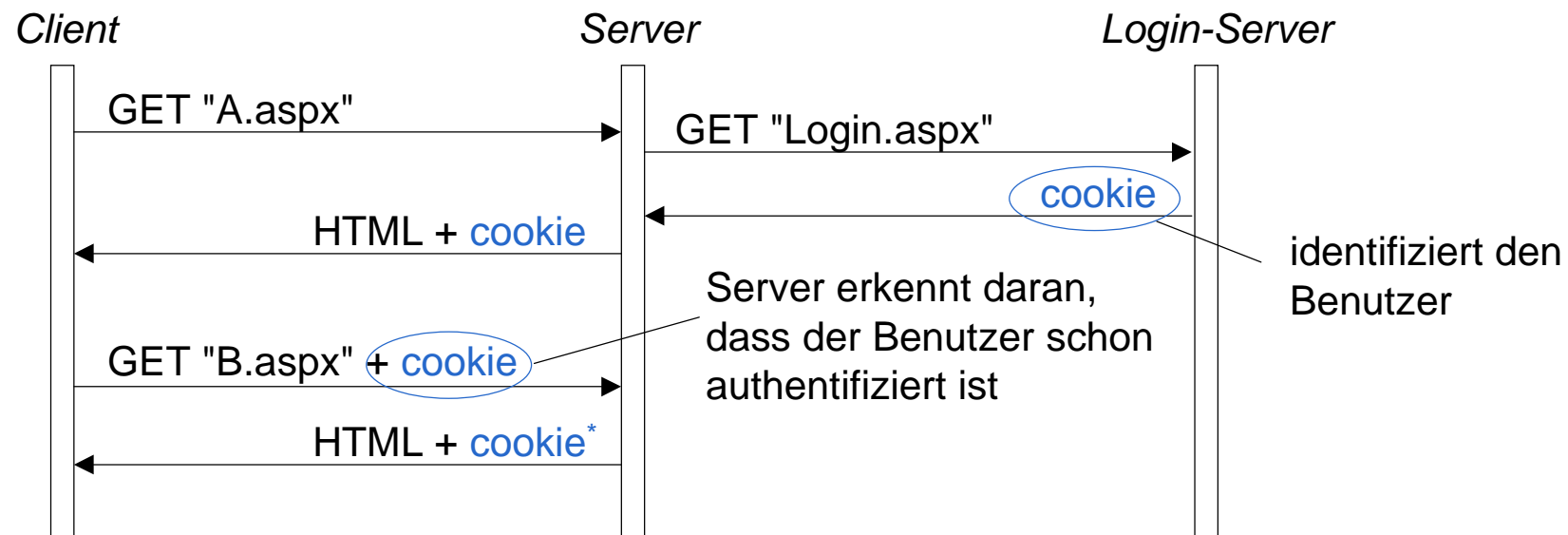
<form runat="server" OnLoad="Page_Load">
    <p> Sie sind angemeldet als:
        <asp:Label id="lb_identity" runat="server" />
    </p>
    <asp:Button id="Logout" runat="server" Text="Abmelden"
        OnClick="Logout_click" />
</form>
```

Bestimme Identität
des Benutzer

zur Login Seite

Benutzeridentifizierung über Cookies

Wie merkt sich ASP.NET, ob ein Benutzer authentifiziert ist?



Angaben über Cookies in Konfigurationsdatei

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

Name des zu
erzeugenden
Cookies

Cookies sollen
verschlüsselt
werden

Cookies sollen
nach 20 Minuten
verfallen

Vordefinierte Login-Steuererelemente

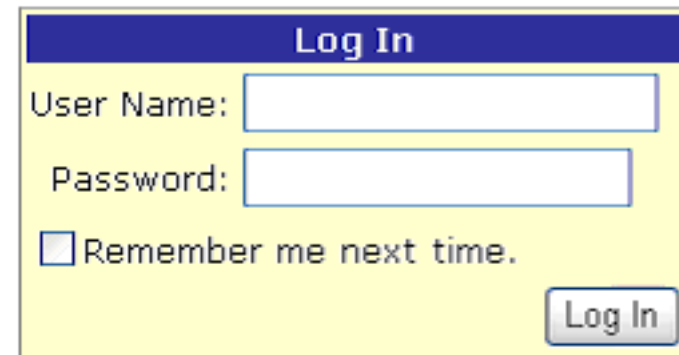
Login

Login.aspx

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

- Wird auf Login-Seite verwendet, die in *web.config* angegeben wurde.
- Wenn Authentifizierung ok, wird der Benutzer zur ursprünglich gewünschten Seite weitergeleitet.

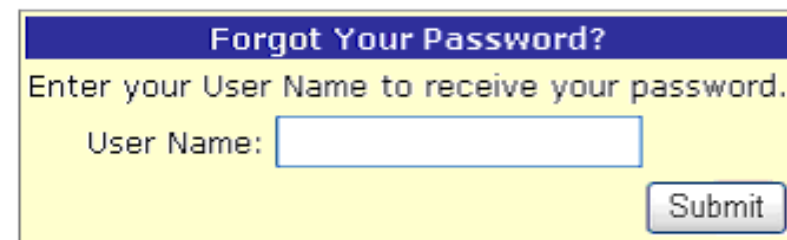
```
<authentication mode="Forms">  
  <forms loginUrl="login.aspx" />  
</authentication>
```



PasswordRecovery

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

- Schickt das Passwort per Email an den Benutzer
- Email und Passwort sind in Benutzerdaten gespeichert (siehe später)



muss in Member-DB
eingetragen sein

...Vordefinierte Login-Steuerelemente

LoginStatus

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

Anzeige

[Login](#) wenn Benutzer noch nicht angemeldet
[Logout](#) wenn Benutzer angemeldet

Login-Link führt zu einer Seite, die man in *web.config* spezifizieren kann

LoginView und LoginName

```
<asp:LoginView Runat="server">
  <AnonymousTemplate>
    Not logged in; please <asp:LoginStatus Runat="Server" />
  </AnonymousTemplate>
  <LoggedInTemplate> Welcome:
    <asp:LoginName Runat="server" /> please
    <asp:LoginStatus Runat="Server" /> when finished
  </LoggedInTemplate>
</asp:LoginView>
```

Text, der angezeigt wird, wenn der Benutzer nicht angemeldet ist

Text, der angezeigt wird, wenn der Benutzer angemeldet ist

LoginName: Name, mit dem sich der Benutzer angemeldet hat

Membership-Klassen



Membership (in *System.Web.Security*)

- Verwaltet Benutzermenge

```
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) {...}
```

- Speichert Benutzer in einer Datenbank (Access, SQL Server, benutzerdefiniert)
Standard-DB MS Access: *ApplicationDir/data/AspNetDB.mdb*
- Benutzer wird über Cookie oder URL-rewriting identifiziert
- *ValidateUser* wird von Login-Steuerelement verwendet

MembershipUser (in *System.Web.Security*)

- Enthält Daten eines einzelnen Benutzers
 - Name
 - Passwort
 - Email-Adresse
 - letztes Login-Datum
 - Passwort-Frage
 - ...

Es muss eine Registrierungsseite geben, die diese Daten erfasst und mit *CreateUser* oder *UpdateUser* abspeichert

Rollen



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

- Verwaltet Rollen (z.B. *Admin*, *Employee*, *User*, ...)

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

- Zugriff auf Seiten eines Verzeichnisses kann auch durch Rollen geschützt werden. Erfordert *Web.config* in diesem Verzeichnis:
- Speicherung der Information kann roleManager->Provider ähnlich wie bei Membership gesteuert werden; z.B. in einer DB

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

Nur Benutzer mit der Rolle *Admin* dürfen auf die Seiten dieses Verzeichnisses zugreifen

Alle anderen Benutzer dürfen nicht zugreifen

Personalisierung



Beliebige Einstellungen von Benutzerprofilen

- Gespeichert als Name/Wert-Paare
- Werden in *Web.config* definiert

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

- Können über *Profile*-Property von *Page* angesprochen werden

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

- Sind statisch getypt
- Werden in einer Datenbank gespeichert
- Werden nur bei Bedarf geladen
- Benutzeridentifikation über Cookies oder URL-Rewriting

} Unterschied zum
Sitzungszustand!

Auswahl der Personalisierungsdatenbank



Durch Provider-Klasse geregelt

- Kann in *web.config* spezifiziert werden

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

- Provider für Access und SQL Server mitgeliefert
- Standard-Provider ist Access
Schreibt auf *ApplicationDir/DATA/AspNetDB.mdb*
- Benutzer kann eigene Provider schreiben und einhängen

Einheitliches Aussehen

■ CSS Files können über CssClass Property direkt verwendet werden

- Nachteil: CSS Files werden nicht durch ASP.NET verwaltet sondern über z.B. IIS

```
<head runat="server">
    <title>Untitled Page</title>
    <link type="text/css" rel="stylesheet" href="StyleSheet.css" />
</head>
<body>
    <asp:Label ID="Label1" runat="server" Text="Page 1" CssClass="test" ></asp:Label>
```

■ CSS als Link zu einer (beliebigen) anderen ASPX Seite

- Vorteile: werden von ASP.NET verwaltet
- können zur Laufzeit verändert werden und dem Klienten angepasst werden
- Nachteil: Einstellungen sind nicht in GUI Builder sichtbar

```
<head runat="server">
    <title>Untitled Page</title>
    <link type="text/css" rel="stylesheet" href="StyleSheet.aspx"/>
</head>
```

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.ClearContent();
    Response.ContentType = "text/css";
    Response.Write(".test {color: red;background-color: #FFC0C0;}");
    Response.End();
}
```

StyleSheet.aspx.cs

The Microsoft Way: Themen und Skins

- Standardeinstellungen für Steuerelemente durch "Templates"

Skin: Einstellungen für ein bestimmtes Steuerelement

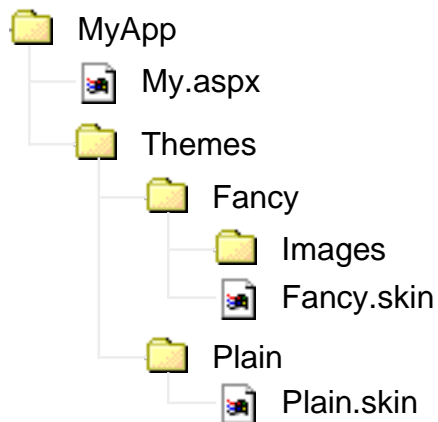
```
<asp:Label ForeColor="#585880" Font-Size="0.9em" Font-Name="Verdana" />
```

Kann durch einen Namen bezeichnet werden

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

Alle nicht gesetzten Attribute bleiben unverändert

Thema: Sammlung von Skins in einer Datei mit der Endung .skin



Fancy.skin

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

Plain.skin

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

Setzen von Themen

■ Global für die ganze Applikation

web.config

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

■ Für eine einzelne Seite

■ in der *Page*-Direktive

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

■ im Hintergrundcode

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

■ Setzen im *PreInit*-Ereignis

■ *Page* hat ein Property namens *Theme*

Explizite Auswahl von Skins

■ Auswahl eines einzelnen Skin-Elements

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

■ Auswahl einer ganzen Skin-Datei

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

Beispiel: personalisiertes Thema

■ Profil-Property definieren

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

■ Benutzer setzt Thema bei einem seiner Besuche

Set Theme

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

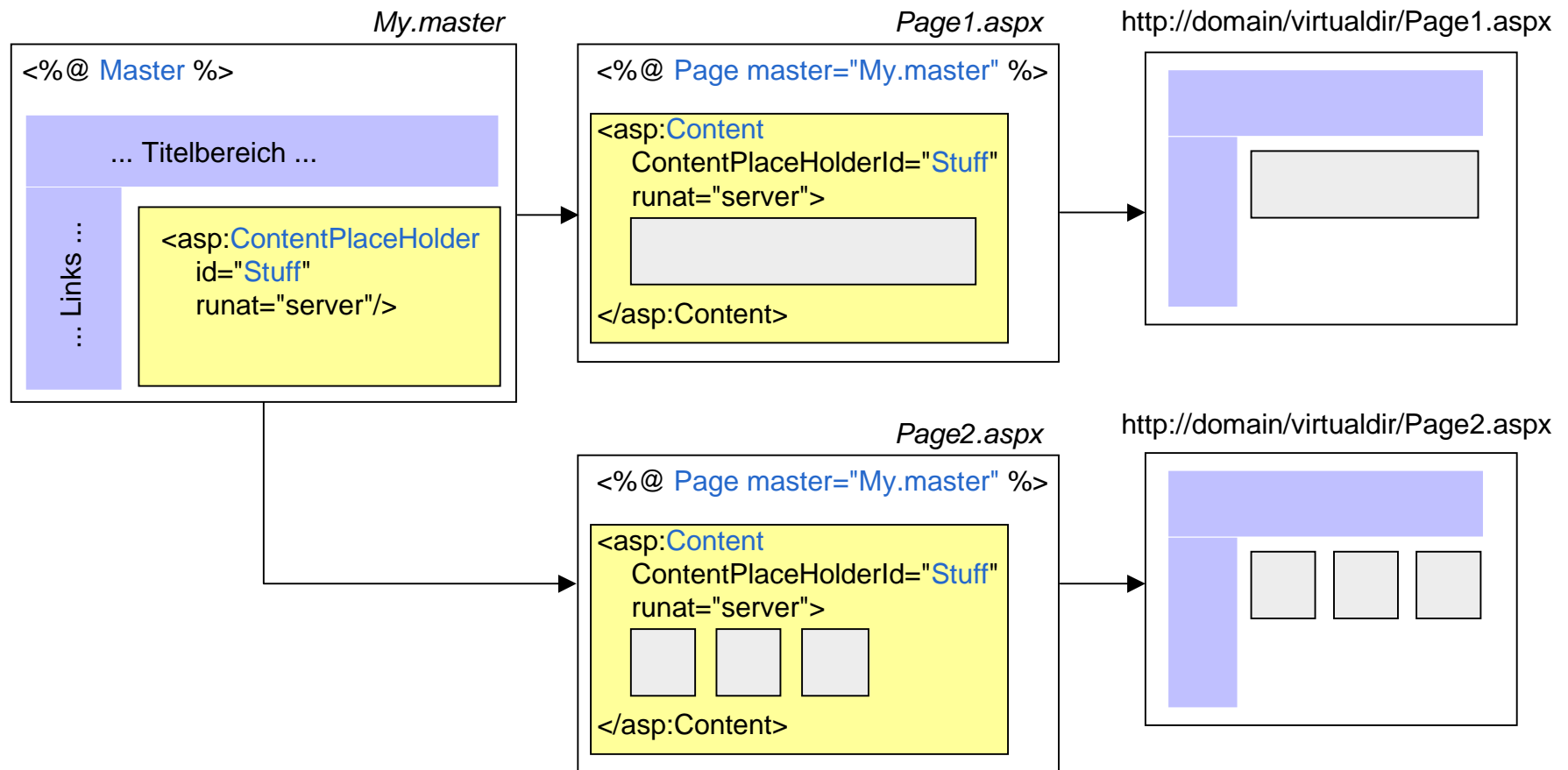
■ Thema wird bei jedem weiteren Besuch verwendet

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

Einheitliche Seitenstruktur

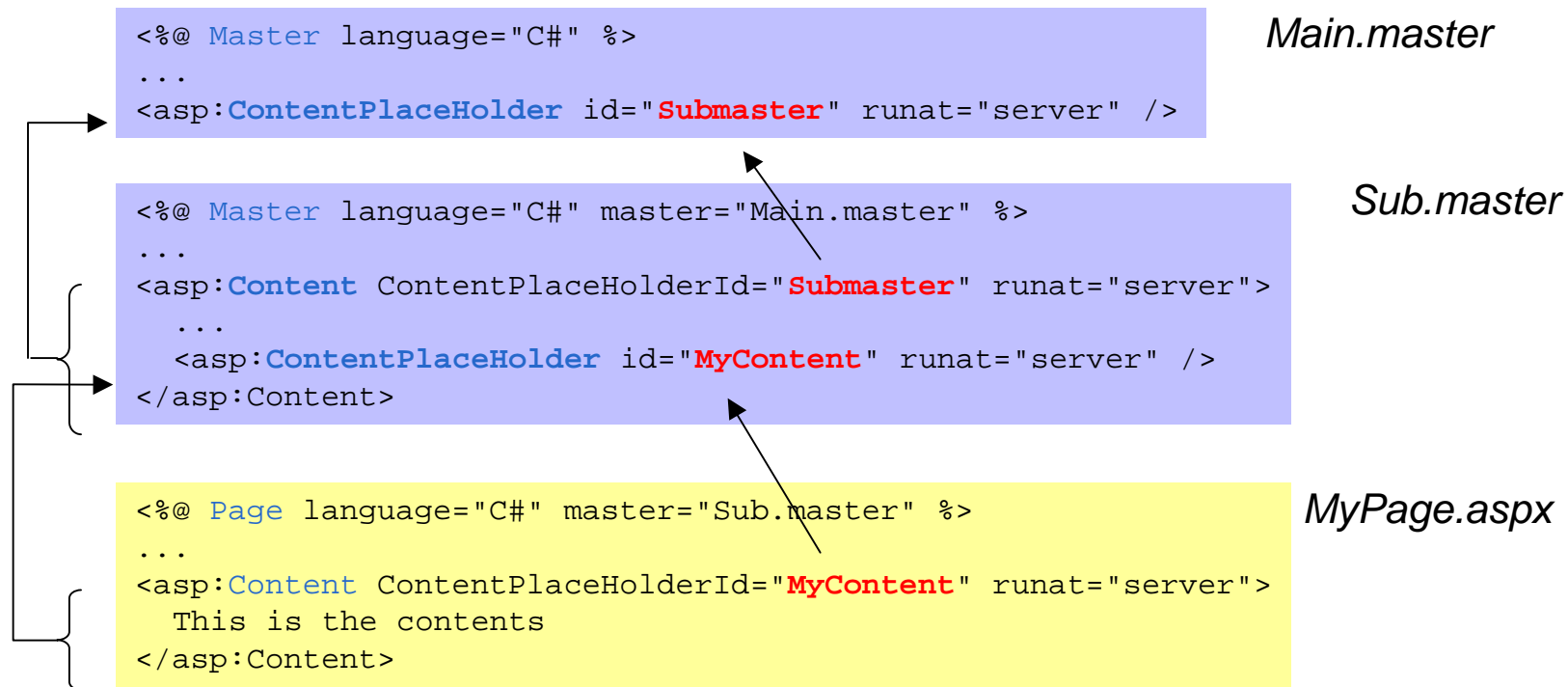
Masterseiten Idee

■ Einheitliches Layout für alle Seiten



Eigenschaften von Masterseiten

- Können beliebiges HTML und beliebige ASP.NET-Steuerelemente enthalten
- Können Hintergrundcode haben
- Können geschachtelt werden



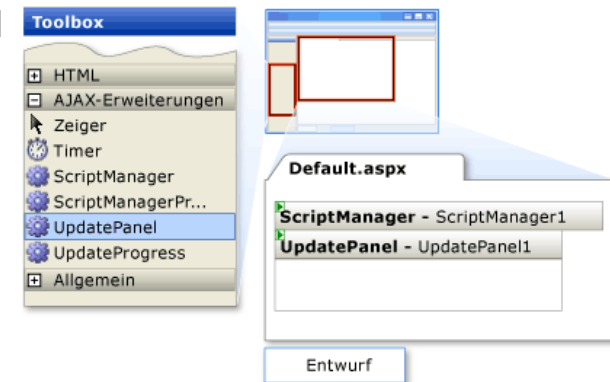
- Master kann auch über *web.config* allen Seiten zugewiesen werden
- Man kann unterschiedliche Master für verschiedene Browsertypen deklarieren

AJAX Erweiterung

Das UpdatePanel

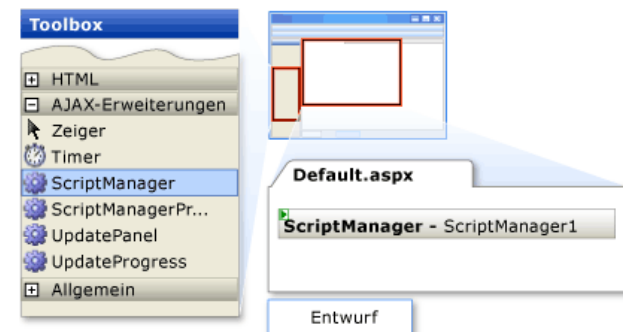
■ Teile der Seite können damit unabhängig aktualisiert werden

- Reduzieren von ganzseitigen Aktualisierungen und vermeiden von Flackern beim Seitenaufbau
- Durchführen einer Client/Server-Kommunikation im AJAX-Stil
 - *ohne Klientenskripte schreiben zu müssen.*



■ Vor dem UpdatePanel muss ein ScriptManager Control eingefügt sein

- Verwaltet ASP.NET-AJAX-Skriptbibliotheken
- Verwaltet Teilrendering von Seiten
- Verwaltet die Generierung von Klientenproxyklassen



■ Header Teil der ASPX Seit

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<%@ Page Language="C#" %>

<html dir="ltr" xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Untitled 1</title>
  <meta http-equiv="Page-Enter" content="RevealTrans(Duration=0,Transition=5)" />
  <script runat="server" type="text/c#">
    protected void Timer1_Tick(object sender, EventArgs e) {
      Label1.Text = "Panel refreshed at: " +
        DateTime.Now.ToLongTimeString() + "." + DateTime.Now.Millisecond;
    }
  </script>
</head>
```

Trick um Flickern zu vermindern:
enables Double Buffering

Serverseitiger Teil der asynchron
ausgeführt werden soll

... Beispiel

■ der Inhalt der Seite

```
<body>
  <form id="form1" runat="server">
    <asp:ScriptManager runat="server" ID="ScriptManager1">
    </asp:ScriptManager>
    <asp:UpdatePanel runat="server" ID="UpdatePanel1">
      <ContentTemplate>
        <asp:Label runat="server" Text="Page not refreshed yet." ID="Label1">
        </asp:Label>
        <asp:Button ID="Button1" runat="server" Text="Refresh"
          OnClick="Timer1_Tick" />
        <br />
        <asp:Image ID="Image1" runat="server" ImageUrl="smiley-face.jpg" />
      </ContentTemplate>
      <Triggers>
        <asp:AsyncPostBackTrigger ControlID="Timer1" EventName="Tick" />
      </Triggers>
    </asp:UpdatePanel>
    <asp:Timer runat="server" ID="Timer1" Interval="1000" Enabled="true"
      OnTick="Timer1_Tick"></asp:Timer>
  </form>
</body>
</html>
```



Knopf für direkten Update

UpdatePanel triggert auf
Timer Event (wird über Panel
Properties gesetzt)

Ajax Timer der jede Sekunde
aufgerufen wird

SOAP

Web Services - SOAP

■ SOAP: Simple Object Access Protocol

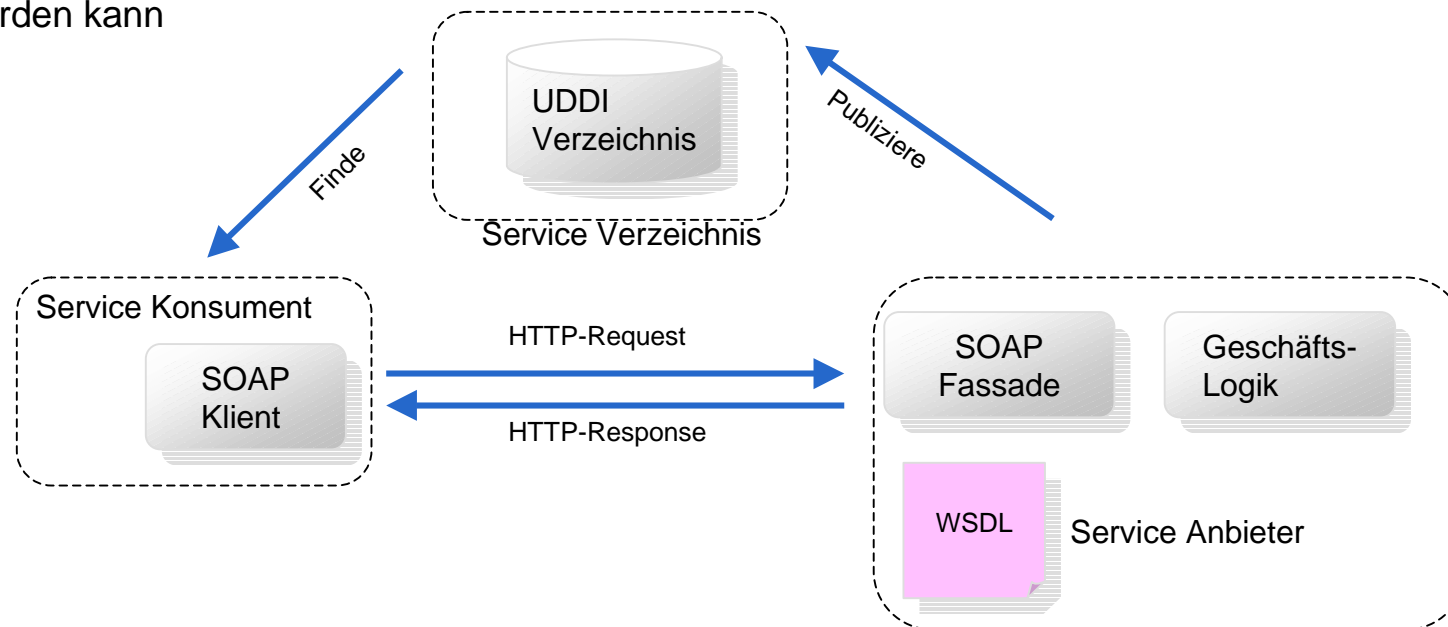
- Definiert ein **Standard-Format** für den Transport von XML Daten, via HTTP, SMTP und FTP für den Zugriff auf Web Services

■ WSDL: Web Service Description Language

- **Beschreibt die Schnittstelle** eines Web Service (Ein-/Ausgabe Parameter, Protokoll Bindung, etc.)

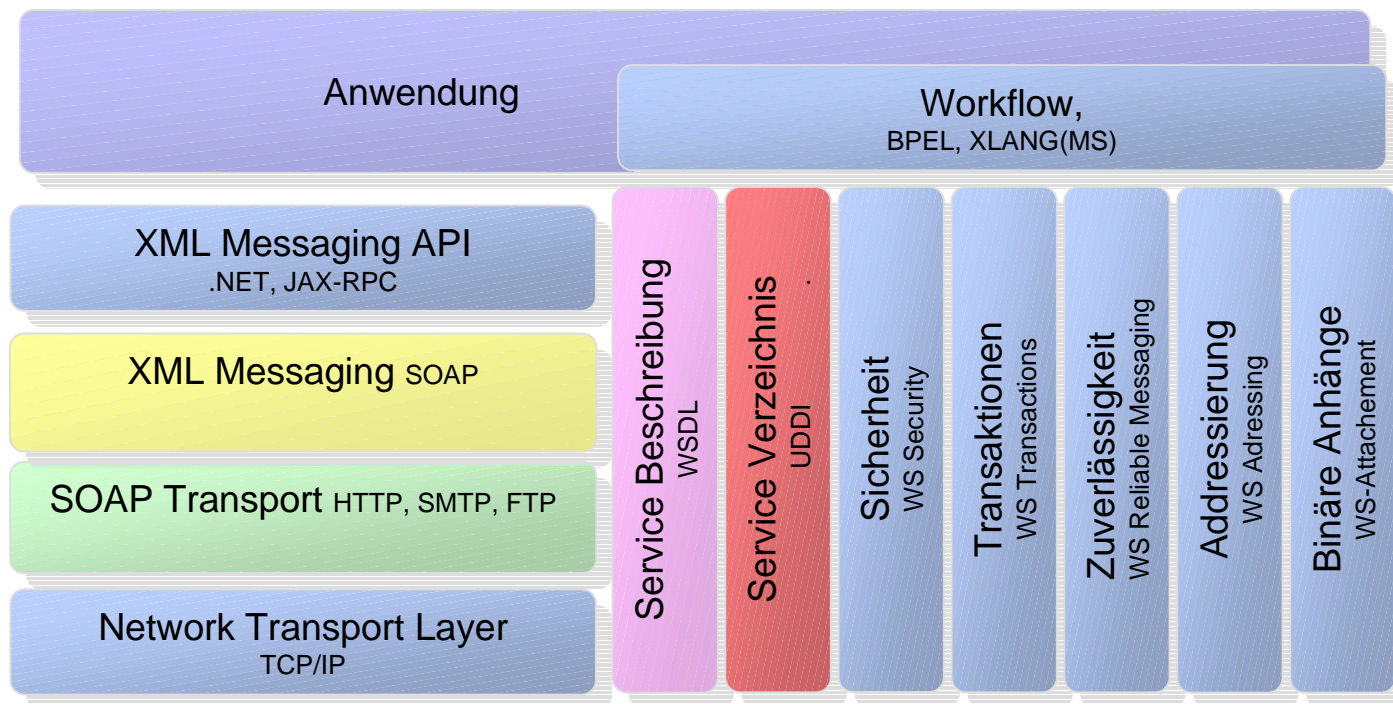
■ UDDI: Universal Description, Discovery, and Integration

- (Weltweites) **Verzeichnis** von Web Services in dem nach verschiedenen Kriterien gesucht werden kann

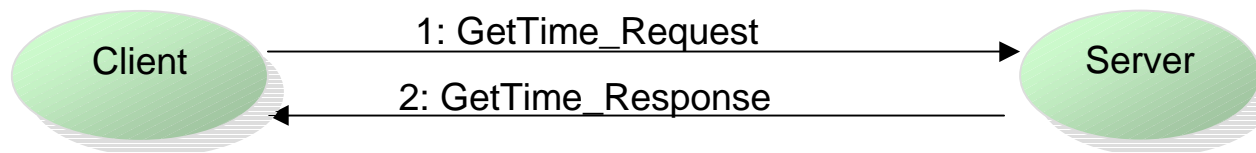


Erweiterter Web Service Stack

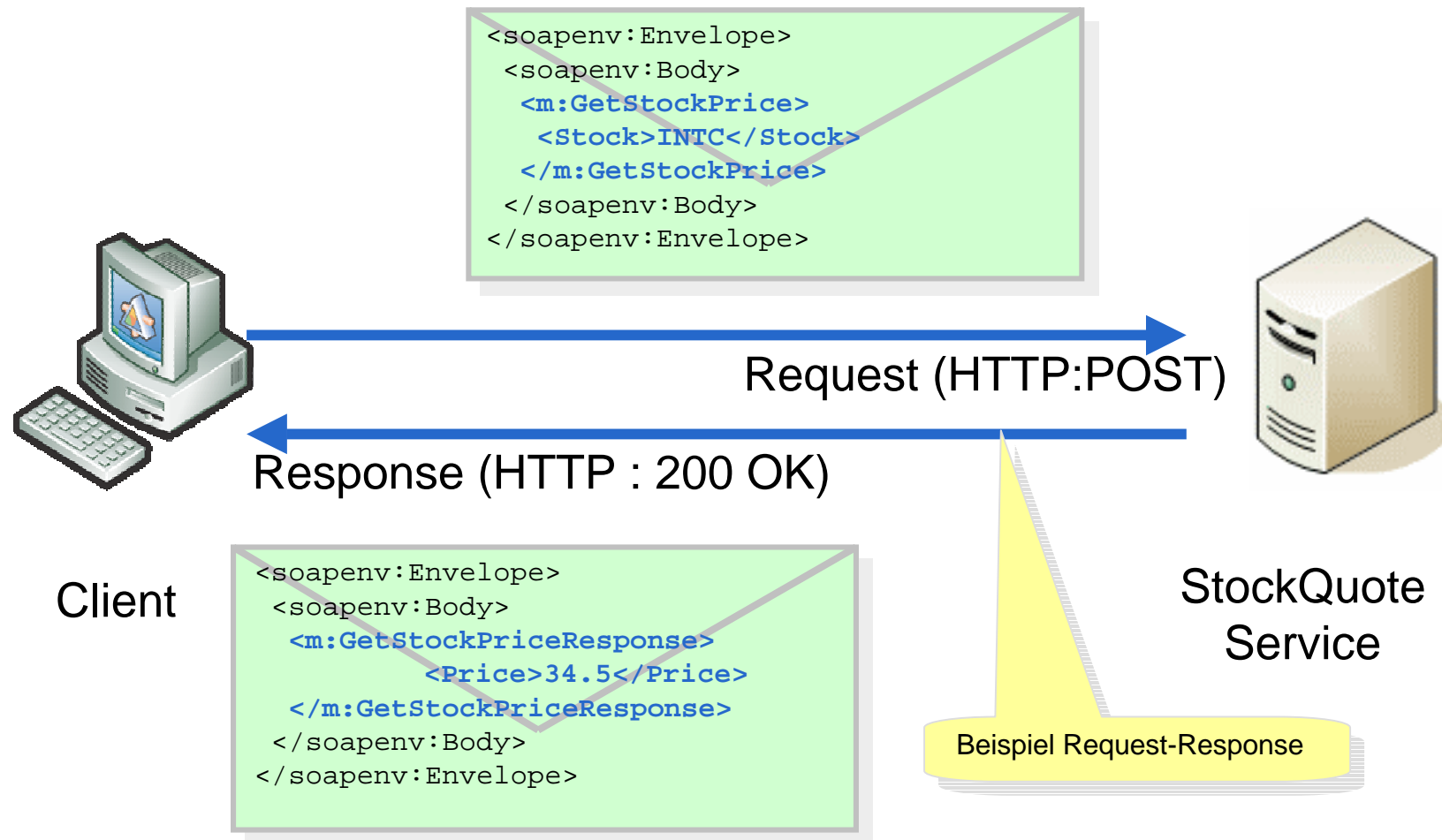
- In konkreten Anwendungsfall von Web Services sind Erweiterungen notwendig für Sicherheit, Transaktionssteuerung, etc.
- Werden angeboten
 - vom Anbieter des WS Infrastruktur (Microsoft, IBM, Open Source)
 - Standards z.T. noch nicht etabliert -> proprietäre, z.T. nicht interoperable Lösungen



- einfaches Nachrichtenformat in XML
 - für Verpackung beliebiger Anwendungsdaten
 - einzelne Nachrichten („one-way“)
- Unabhängig vom Transportprotokoll
- SOAP ist erweiterbar
 - Sicherheit
 - Authentifikation
 - etc.
- Verschiedene Meldungs austauschmuster durch kombination einzelner Nachrichten (*Message Exchange Patterns*)
 - one-way, request-response



Funktionsweise von SOAP über HTTP



SOAP und .NET

- IIS und ASP.NET-Infrastruktur unterstützen Web-Services

- .NET Framework bietet eine Reihe von

- Basisklassen
- Attributen
- Kommandozeilen Werkzeuge: WSDL.EXE

für die Realisierung von Web-Services

- Visual Studio.NET bietet Werkzeuge zur Erstellung von Web-Services

- Entwicklung
- Testumgebung
- Erstellung von Proxys:
- Verwaltung des IIS

.NET-Namensräume



- **System.Web.Services**
 - um Web-Services zu erstellen (z.B.: WebService, WebMethod)
- **System.Web.Services.Configuration**
 - um SOAP zu erweitern
- **System.Web.Services.Description**
 - um WSDL zu erstellen und zu bearbeiten
- **System.Web.Services.Protocols**
 - zur Implementierung von Protokollen (z.B. SOAP-HTTP)
- **System.Xml.Serialization**
 - zur XML-Serialisierung bei Datentransfer

Beispiel: TimeService

TimeService.asmx

gespeichert in Web
Server Verzeichnis

WebService-Direktive

```
<%@ WebService Language="C#" Class="TimeService" %>
```

```
using System;
```

```
using System.Web.Services;
```

Meist Code Behind

WebService erweitern

```
[WebService(Namespace = "http://tempuri.org/")]
```

```
public class TimeService : WebService {
```

```
    [WebMethod(Description="Returns the current time")]
```

```
    public string GetTime(bool shortForm) {
```

```
        if (shortForm) return DateTime.Now.ToShortTimeString();
```

```
        else return DateTime.Now.ToLongTimeString();
```

```
    }
```

```
}
```

Attribut [WebMethod] deklariert
Web-Service-Methode

- in asmx-Datei mit @WebService-Direktive

```
<%@ WebService Language="C#" Class="MyWebService" %>
```

- ableiten von Basisklasse System.Web.Services.WebService

```
public class MyWebService : WebService {
```

- Kennzeichnung und Einstellungen über .NET-Attribute

Kennzeichnung von Web-Service-Methoden

Festlegen von Nachrichtenformat und Kodierung

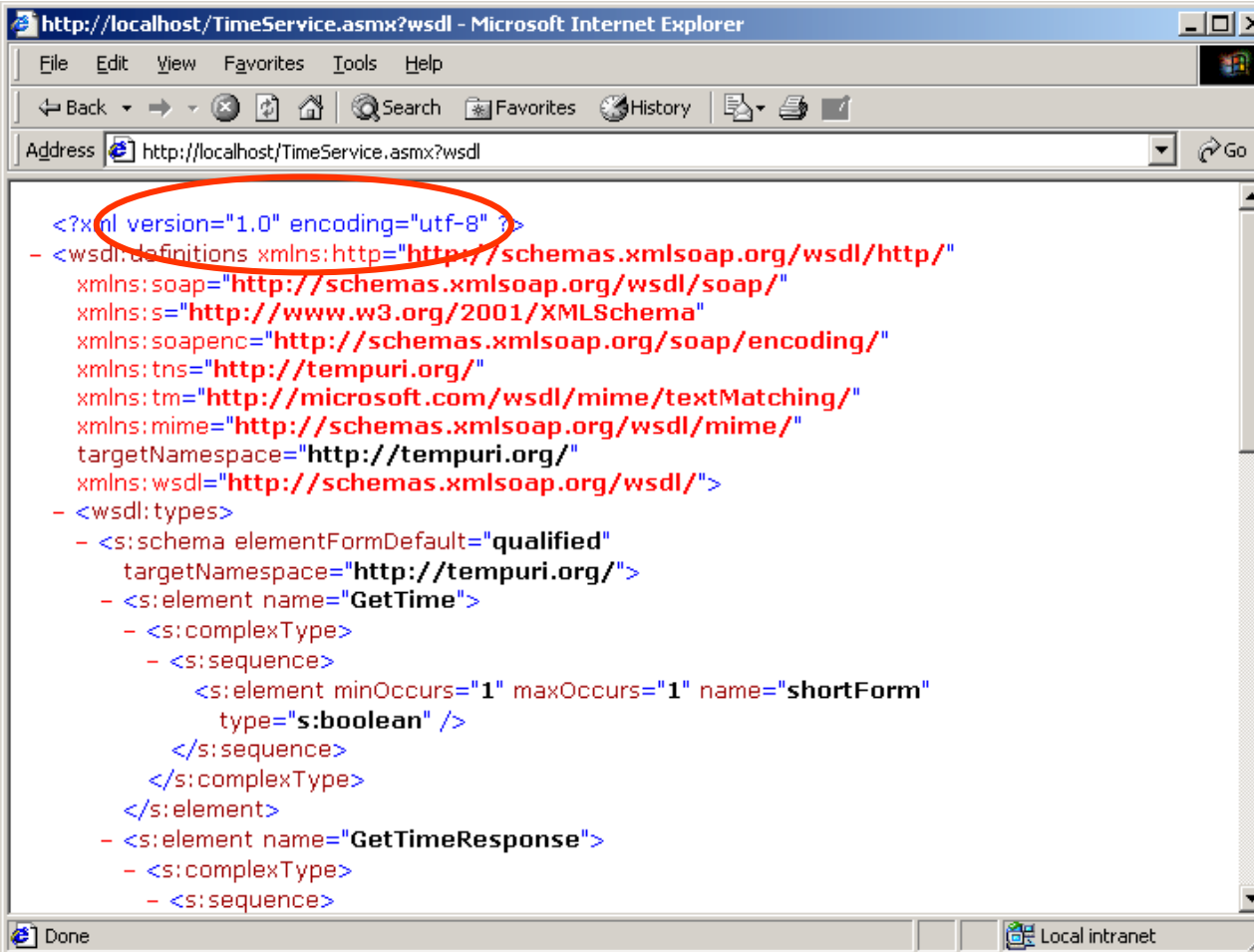
zu verwendende Namensräume und XML-Elemente

etc.

```
[WebMethod(Description= "comment ")]  
[...]  
public Returntype MyWebMethod( ...) {  
    ...  
}
```

Anzeige der Schnittstelle

- <web-service-url>?wsdl liefert die WSDL



```
<?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:s="http://www.w3.org/2001/XMLSchema"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:tns="http://tempuri.org/"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  targetNamespace="http://tempuri.org/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
- <wsdl:types>
- <s:schema elementFormDefault="qualified"
  targetNamespace="http://tempuri.org/">
- <s:element name="GetTime">
- <s:complexType>
- <s:sequence>
- <s:element minOccurs="1" maxOccurs="1" name="shortForm"
  type="s:boolean" />
- </s:sequence>
- </s:complexType>
- </s:element>
- <s:element name="GetTimeResponse">
- <s:complexType>
- <s:sequence>
```

Web Service Test-Aufruf im Browser

The following operations are supported. For a formal definition, please review the [Service Description](#).

- [GetTime](#)
Returns the current time

This web service

Recommendation

Each XML Web service on the Web publishes XML Web service metadata. Your XML Web service's Internet like URLs, they need to be replaced with actual values.

For XML Web service attribute's NameSpace. XML Web service i "http://microsoft.c

Click [here](#) for a complete list of operations.

GetTime
Returns the current time

Test
To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
shortForm	true

Invoke

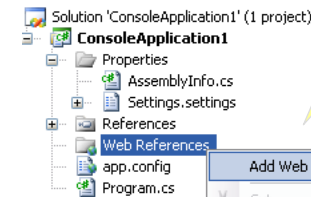
SOAP
The following is a sample SOAP request and response. The **placeholders** shown need to be replaced with actual values.

```
<?xml version="1.0" encoding="utf-8" ?>
<string
  xmlns="http://tempuri.org/">16:45</string>
```

Beispiel: Einfacher .NET-Client

wSDL.exe erzeugt Proxy für Client

```
> wsdl.exe /namespace:TimeClient /out:TimeService.cs  
http://localhost:8080/TimeService.asmx?wsdl
```



Oder in VS Add
WebReference

■ Es wird folgende Proxy Klasse generiert

```
namespace TimeClient {  
using ...  
  
public class TimeService : System.Web.Services.Protocols.SoapHttpClientProtocol {  
    /// <remarks/>  
    public TimeService() {  
        this.Url = "http://localhost/TimeService.asmx";  
    }  
  
    [System.Web.Services.Protocols.SoapDocumentMethodAttribute("http://tempuri.org/GetTime",  
        RequestNamespace="http://tempuri.org/",  
        ResponseNamespace="http://tempuri.org/",  
        Use=System.Web.Services.Description.SoapBindingUse.Literal,  
        ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped)]  
  
    public string GetTime(bool shortForm) {  
        object[] results = this.Invoke("GetTime", new object[] {  
            shortForm});  
        return ((string)(results[0]));  
    }  
    ...  
}
```

- Client-Programm erzeugt TimeService-Objekt und ruft GetTime auf

```
using System;
using TimeClient; //Namespace des erzeugten Proxies

public class NetClient {
    public static void Main(string[] args) {
        TimeService service = new TimeService();
        Console.WriteLine("Die Zeit am Server ist: ");
        string time = service.GetTime(true);
        Console.WriteLine(time);
    }
}
```

Erweiterte SOAP Einstellungen

.NET Klassen



.NET beinhaltet Klassen zur Unterstützung für

- Basisklasse für den SOAP Aufruf
 - *Client Proxy erbt von diesem*
- Attribute zur Bestimmen von Nachrichtenformat und Kodierung
- Kodierung von .NET-Datentypen
- Entwickeln von Header-Einträgen
- Handhabung des Lebenszyklus von Web-Services

Klasse SoapHttpClientProtocol

```
public class SoapHttpClientProtocol {  
    //--- properties  
    public bool AllowAutoRedirect {get; set;}  
    public CookieContainer CookieContainer {get; set;}  
    public ICredential Credentials {get; set;}  
    public bool PreAuthenticate {get; set;}  
    public IWebProxy Proxy {get; set;}  
    public int Timeout {get; set;}  
    public string Url {get; set;}  
  
    ...  
    //--- methods  
    protected object[] Invoke(string methodName,  
                               object[] parameters);  
}
```

Timeout

Timeout des Aufrufs

Url

URL des Web Services

AllowRedirect

Umleitungen werden gefolgt

CookieContainer

Gespeicherte Cookies

Credentials

Benutzerauthentisierung Daten
z.B. NetworkCredentials

PreAuthenticate

Beim ersten Aufruf werden
Credential Daten mitgeliefert

Proxy

Proxy für den Aufruf

Invoke

Eigentlicher Aufruf

Bestimmen von Nachrichtenformat und Kodierung

Einbezug von abgeleiteten Typen

- **SoapInclude** Attribute erlaubt Aufnahme von Typen

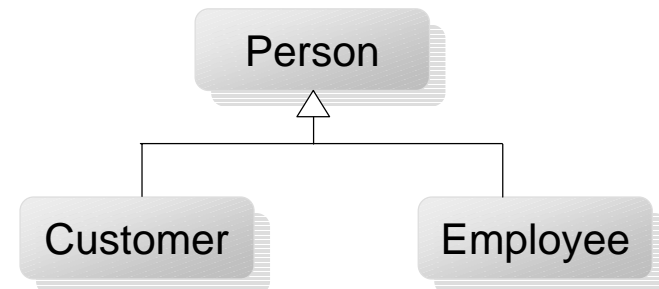
➔ Wichtig bei Spezialisierungen

Beispiel: PersonService

Web-Method mit Rückgabewert
vom Typ Person

```
public class PersonService : WebService {  
  
    [WebMethod]...  
    public Person[ ] GetAll() {...}  
}
```

Person hat 2 Spezialisierungen
Customer und Employee



➔ Customer und Employee müssen explizit in die Web-Beschreibung aufgenommen werden !

Beispiel: PersonService

■ Klassen Person, Customer und Employee

```
public abstract class Person { ...}  
public class Customer : Person { ...}  
public class Employee : Person { ...}
```

PersonService definiert Web-Method GetAll mit Rückgabotyp Person[]
Mit SoapInclude werden Customer und Employee inkludiert

```
<%@ WebService Language="C#" Class="PersonService" %>  
using System; ... using System.Xml.Serialization;  
public class PersonService : WebService {  
    [WebMethod] [SoapRpcMethod]  
    [SoapInclude(typeof(Customer)), // Subtypen von Person  
     SoapInclude(typeof(Employee))] // (für Aufrufe via SOAP-HTTP)  
    [XmlInclude(typeof(Customer)),   // Subtypen von Person  
     XmlInclude(typeof(Employee))]   // (für Aufrufe via HTTP-GET / HTTP-POST)  
    public Person[] GetAll() {  
        Person[] data = new Person[2];  
        data[0] = new Customer("1", "Max Customer", "EMP-33");  
        data[1] = new Employee("EMP-33", "Tom Employee");  
        return data;  
    }  
}
```

Lebenszyklus

Klasse WebService

■ Serverseitige Basisklasse

```
public class WebService: TemplateControl {  
    //--- properties  
    public  HttpSessionState Application { get; }  
    public  HttpSessionState Session { get; }  
    public  HttpContext Context { get; }  
        public IPrincipal User {get;}  
    ...  
    //--- methods  
        ...  
}
```

Context

HttpContext des aktuellen Aufrufs

User

Information über Aufrufer

Application und Session

Applikationszustand und
Sitzungszustand

■ Web-Service-Objekte sind zustandslos

- werden bei jedem Methodenaufruf neu erzeugt

■ Daten können aber in Properties von


- Application-Zustandsobjekt oder

```
public HttpSessionState Application {get;}
```

- Session-Zustandsobjekt

```
public HttpSessionState Session {get;}
```

gespeichert werden



```
public sealed class HttpSessionState : ICollection, IEnumerable {  
    public object this[ string name ] {get; set;}  
    public object this[ int index ] {get; set;}  
    ...  
}
```


Beispiel: StateDemo

Web-Service StateDemo demonstriert Speicherung von Daten

```
<%@ WebService Language="C#" Class="StateDemo" %>
using System.Web.Services;
[WebService(Namespace="http://dotnet.jku.at/StateDemo/")]
public class StateDemo : WebService {
```

Methode IncApplication erhöht Property "Hit" bei Application-Objekt

```
[WebMethod()]
public int IncApplication() {
    int hit = (Application["Hit"] == null) ? 0 : (int) Application["Hit"];
    hit++;
    Application["Hit"] = hit;
    return hit;
}
```

...Beispiel: StateDemo

Parameter EnableSession ermöglicht Sessions
IncSession erhöht Property "Hit" bei Session-Zustandsobjekt

```
[WebMethod(EnableSession=true)]  
public int IncSession() {  
    int hit = (Session["Hit"] == null) ? 0 : (int) Session["Hit"];  
    hit++;  
    Session["Hit"] = hit;  
    return hit;  
}
```

■ Eigene Steuerelemente

- User Controls
- Custom Controls

Zusammengesetzt aus bestehenden Web Controls

Von Grund auf neu geschriebene Web Controls

■ Zustandsverwaltung und Konfiguration

- Seite/Session/Applikation

■ Request-Response

- "Low-Level" Programmierung

■ Authentisierung, Membership & Profile

■ Einheitliches Aussehen

- CSS und Skins

■ Navigation

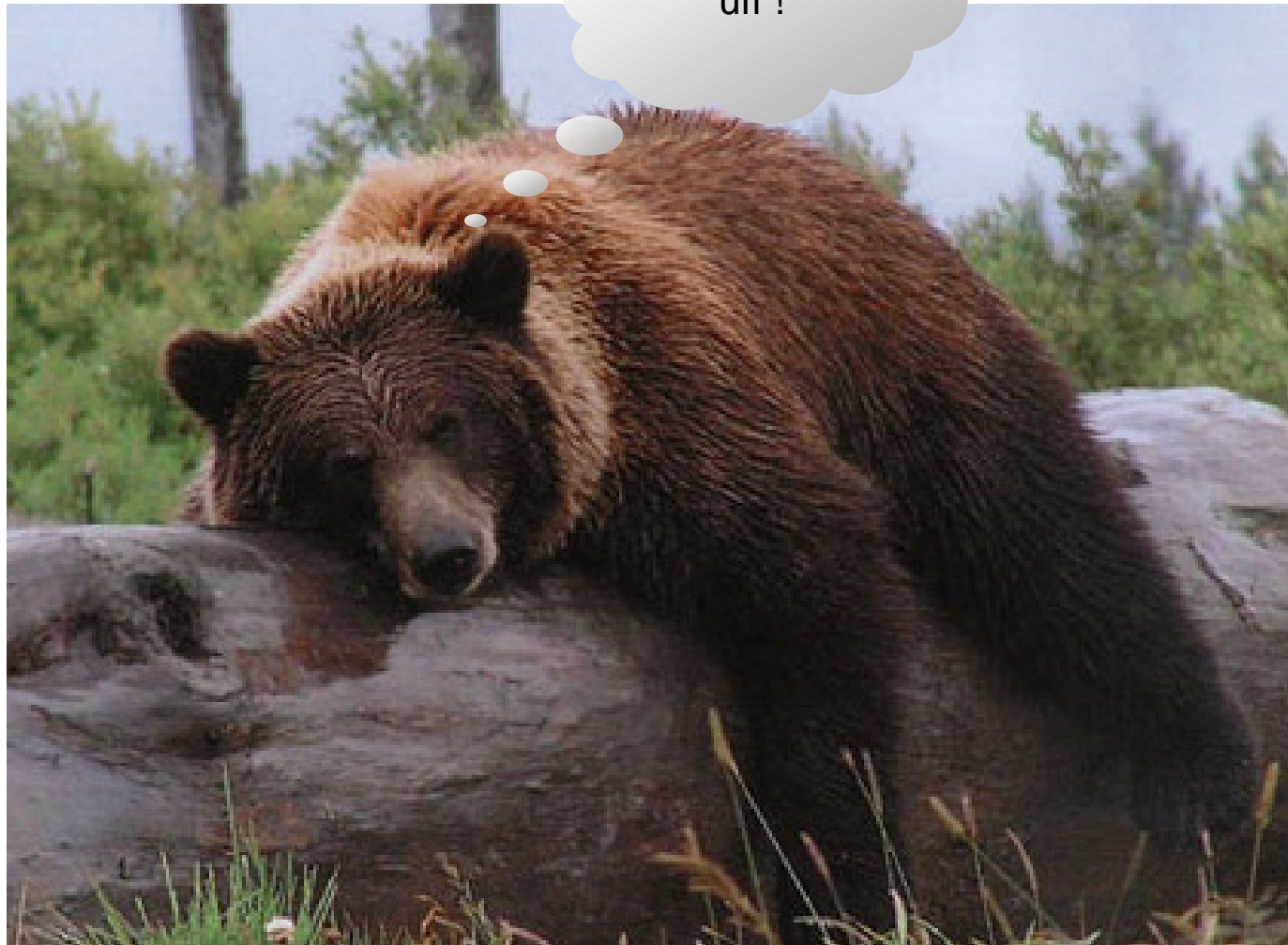
- Baumnavigation und Sitemap

■ Einheitliche Struktur

- Master Seiten

■ Web Services

Fragen?



Konfigurationsabschnitte in ASP.NET

Section Name	Beschreibung
<appSettings>	Hier können Sie benutzerdefinierte Einträge wie DSNs eintragen. Dieser Handler ist sowohl in der machine.config als auch in allen web.configs zulässig
<authentication>	Einstellungen für die ASP.NET Authentifizierungsunterstützung; notwendig um einen User durch einen Login eindeutig zu identifizieren
<authorization>	Einstellungen für die ASP.NET Autorisierungsunterstützung; weist einem User anhand seiner Credentials den ihm zugewiesenen Bereich der Web Site zu
<browserCaps>	Einstellungen für Browsereigenschaften
<compilation>	Enthält Einstellungen für die ASP.NET Kompilierungsoptionen
<customErrors>	Hier können Sie benutzerdefinierte Fehlermeldungen definieren, d.h. bei einem aufgetretenen, spezifischen Fehler wird per Redirect zu der eigens erstellten Seite gelinkt
<globalization>	Enthält länderspezifische Einstellungen, wie z.B. Zeichensätze und Ländercodes (LCID, LoCALE Identifier)
<httpHandlers>	Mappt Seiten-Requests auf das jeweilige Interface (IHandler oder IHttpHandlerFactory)
<httpModules>	Organisiert HTTP Module (eine Klasse oder Assembly (dll)) einer Applikation
<httpRuntime>	Enthält die Einstellungen für die ASP.NET HTTP Runtime. Dieser Handler ist sowohl in der machine.config als auch in allen web.configs zulässig
<identity>	Steuert die Application Identity mittels Client Impersonation
<machineKey>	Einstellungen für die Erstellung von Schlüsseln anhand von Algorithmen (SHA1, MD5 3DES) welche über den RNGCryptoServiceProvider erstellt werden
<pages>	Hier können Webseiten-spezifische Einstellungen gemacht werden
<processModel>	Einstellungen für das ASP.NET Process Model des IIS. Z.b. für Timeouts, Request Limits
<securityPolicy>	Definiert die Mappings für Policy Dateien. Dieser Handler ist sowohl in der machine.config als auch in allen web.configs zulässig
<sessionState>	Konfiguriert den Session State des HTTP Modules
<trace>	Einstellungen für die ASP.NET Trace Services. Diese Einstellungen sind ausschließlich für Testzwecke, also wenn Sie die ASP.NET Applikation entwickeln bzw. testen, da diese sensitive Daten enthält. Hat also im tatsächlichen Einsatz nichts verloren
<trust>	Legt die Code Access Permissions für eine bestimmte Applikation fest. Diesen Einstellungen liegen wieder Policy Dateien zugrunde. Dieser Handler ist sowohl in der machine.config als auch in allen web.configs zulässig
<webServices>	Beinhaltet Einstellungen für Web Services. Z.b. für die Festlegung der Protokolltypen, MIME Typen oder für die SOAP Extensions

Seitenhierarchie einer Applikation wird in XML beschrieben

app.sitemap

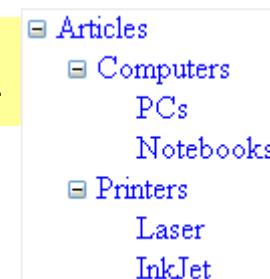
```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap>
  <siteMapNode title="Articles" description="Home" url="Articles.aspx" >
    <siteMapNode title="Computers" url="Computers.aspx">
      <siteMapNode title="PCs" url="PCs.aspx" />
      <siteMapNode title="Notebooks" url="Notebooks.aspx" />
    </siteMapNode>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

Jede dieser Seiten
benutzt denselben
Master, auf dem auch
der *TreeView* steht.
Dadurch wird der
TreeView immer
wieder angezeigt.

TreeView kann daran gebunden werden

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

verbindet sich standardmässig zu *app.sitemap*



Navigationshilfe SiteMapPath

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

Zeigt den Weg zur aktuellen Seite an, falls diese Teil einer Sitemap ist

[Articles](#) > [Computers](#) > PCs

