ASP.NET

Überblick

- Active Server Pages
- Technik mit der man Webseiten dynamisch erstellen kann
- Hauptanwendungsgebiet:
  Entwicklung verteilter Internetseiten
- ASP ist:
  - objektorientiert
  - Layout und Anwendungslogik werden getrennt
  - stellt reichhaltige Bibliothek von WebControls (GUI-Elemente) zur Gestaltung von Webseiten zur Verfügung
  - WebControls reagieren auf Ereignisse
  - WebControls können mit Properties eingestellt werden
  - Skriptcode (auf Server) in bel. .NET-Sprache wird kompiliert nicht interpretiert
    Anm.: Javascript wird auf Client (Browser) ausgeführt
  - Projektvarianten: AspWebForm, Asp MVC

ASP.NET

Statische HTML Seiten

- HTML: hypertext markup language
- Sprache zur Layoutgestaltung von Webseiten

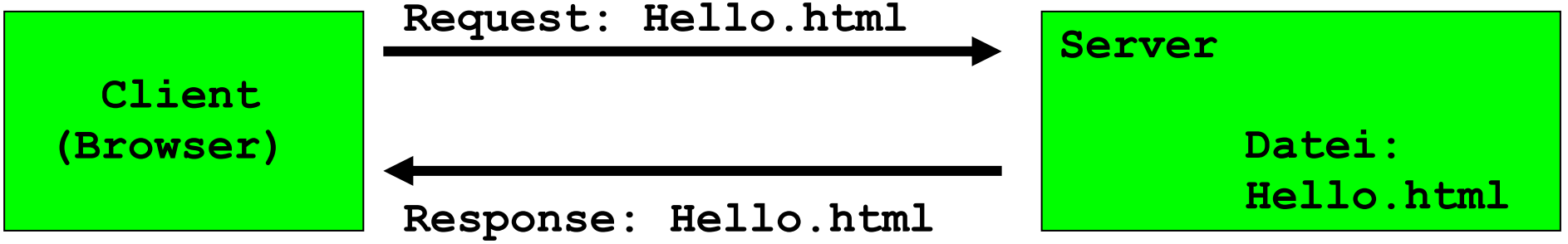
Beispiel:

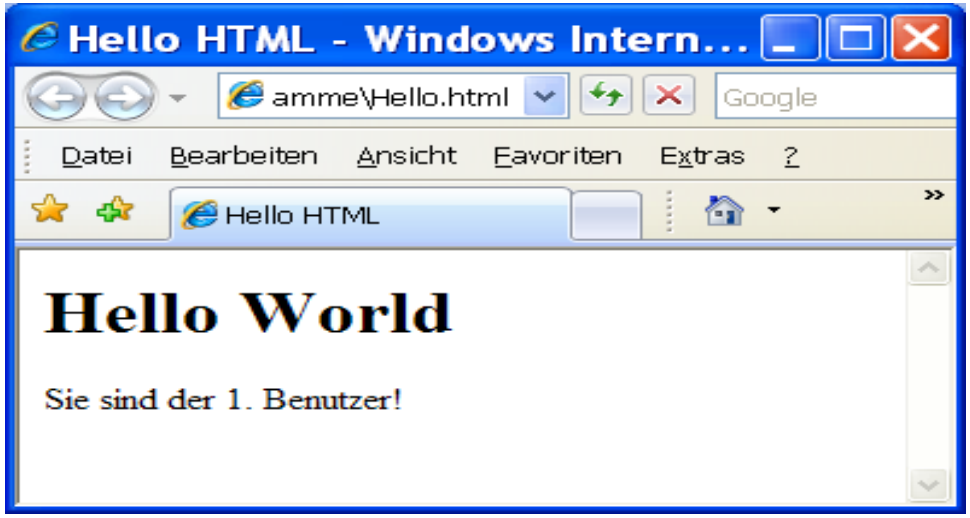
```
<hmtl>
 <head>
   <title> Hello HTML </title>
 </head>
 <body>
  <h1> Hello World </h1>
  Sie sind der 1. Benutzer!
 </body>
</hmtl>
```

ASP.NET

Verarbeitung statischer HTML-Seiten

- Browser fordert Datei über HTTP-Protokoll
- Server liefert Datei oder Fehlercode





ASP.NET

Dynamische aspx-Seiten

```
<%@ Page Language="C#" %>
<html>
  <head> <title>Dynamische ASPX-Seite</title> </head>
  <body>
    <h1> Hello World </h1>
    Sie sind der <%
      int n = 10;      // mit File auf Server hochzaehlen!
      Response.Write(n);
    %>. Besucher dieser Seite!
  </body>
</html>
```

- Page-Directive: Language= (verwendete Sprache)
- Quellcode zwischen: <% und %> (Besser: Code auslagern)
- Weiterleitung an HTML-Ausgabe: Response.Write(n)
  Response-Objekt stellt Antwort des Servers dar

ASP.NET

Alternative Technologien

- JSP Java Server Pages
  - Sind Html-Seiten mit zusätzlichen Html-spezifischen Tags und Javacode
  - Webcontainer generiert aus JSP ein Java-Servlet. Java Servlet bettet Html in Java-Klassen ein

- JSF Java Server Faces
  - Ist Framework auf Basis von Servlets und JSP.
  - Alternative zu Struts
  - Struts: OpenSourceFramework
  - JSF Seite ist spezielle JSP-Seite mit JSF-Tags

- PHP
  - Skriptsprache zur Erstellung dynamischer Webseiten
  - ab PHP 4.0 objektorientierte Programmierung möglich

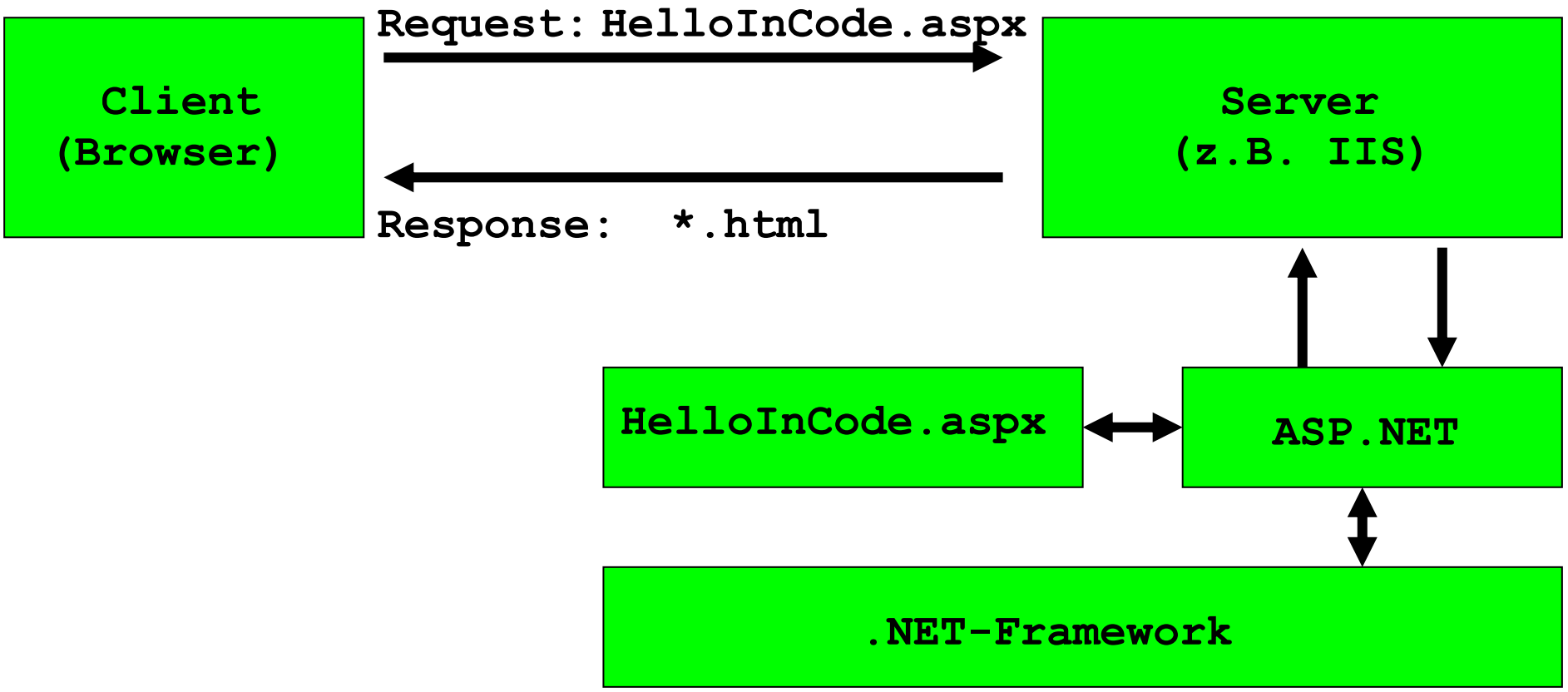
ASP.NET

Virtuelles Verzeichnis

- enthaelt alle zur Webseite gehörenden Dateien
- IIS (Internet Information Server) findet über virtuelles Verzeichnis physikalisches Verzeichnis mit konkreten Dateien
- Seiten werden im Browser mit virtuellem Verzeichnis angesprochen
- Aufruf: http://localhost/Aliasnamen/Dateinamen
  hier: http://localhost/KF_ASP/HelloInCode.aspx
  Anm.: IIS (Internet Information Server) muss laufen
- Anlegen:
  Systemsteuerung|Verwaltung| Internet Informationsdienste
  - Festlegen Aliasnamen
  - Zuordnung zu physikalischem Verzeichnis
  Hinweis: gegebenenfalls ASP.Net beim IIS wie folgt registrieren: aspnet_regiis.exe /r

ASP.NET

Verarbeiten der aspx-Datei



ASP.NET

Erzeugte HTML-Seite

- Server erzeugt dynamisch HTML-Seite und gibt diese an Client zurück (Response)
- Browser zeigt HTML-Seite an

Browser-Quelltext:

```
<html>
  <head>
    <title>Dynamische ASPX-
Seite</title>
  </head>
  <body>
    <h1> Hello World </h1>
    Sie sind der 10. Besucher dieser
Seite!
  </body>
</html>
```

ASP.NET

Serverseitiger Skriptcode in aspx-Datei

```
<%@ Page Language="C#" %>
<html>
  <head>
    <title>Besucherz&auml;hler</title>
    <script Language="C#" Runat="server">
      int CounterValue() {
        int n=10;
        return n;}
    </script>
  </head>
  <body>
    <h1> Hello World </h1>
    Sie sind der <%= CounterValue() %>.Besucher dieser Seite!
  </body>
</html>
```

- Quellcode steht zwischen den <script> Marken

ASP.NET

- Zu welcher Klasse gehört CounterValue()?
  - ASP.NET erzeugt automatisch von Page abgeleitete Klasse und compiliert diese und erstellt Instanz:

```
public class Dateiname_aspx : System.Web.UI.Page,
     System.Web.SessionState.IrequireSessionState{
     ...
     int CounterValue(){...}
     public void Render ...
    }
```

- Klasse Page enthält u. a. Methode Render (von ASP.NET automatisch aufgerufen) zum Erzeugen der HTML-Ausgabe

- Was bedeutet `<%= CounterValue() %>` ?

```
<%= CounterValue() %>
entspricht: <%Response.Write(CounterValue());%>
```

ASP.NET

Auslagern Code in Hintergrundcode

- aspx-Datei:

```
<%@ Page Language="C#" Inherits="CounterPage"
 Src="HelloCBH.cs" %>
<html>
  <head><title>Besucherz&auml;hler</title></head>
  <body>
    <h1> Hello World </h1>
    Sie sind der <%= CounterValue() %>.Besucher dieser Seite!
  </body>
</html>
```

- Ausgelagerter Code. CodeBehind-Datei: HelloCBH.cs
  Klasse Dateiname_aspx wird von CounterPage abgeleitet

```
public class CounterPage : System.Web.UI.Page {
  public int CounterValue() { int n=0;  return n; }
}
```

ASP.NET

HTML-Formular

- kann beliebig viele Formulare (<form>) enthalten
- Formulare enthalten Formularelemente
  - Textfelder <input type="text"...>
  - Buttons <input type="button"...>
  - Ckeckboxes <input type="ckeckbox"...>
  - ...
- Übergabe der Daten an Empfänger als Name-Wert-Paare
  - Methode: post
    Daten in Eingabestrom des Empfängers
  - Methode: get
    Daten werden an Adresse der Seite angehängt
- Arten möglicher Empfänger
  - E-Mail Empfänger z.B.: action="mailto:xy@gmy.de"
  - Programm z.B.: action="http://fh.de/myprog"
  - Webseite z.B.: action="http://fh.de/test.asp"

ASP.NET

Web-Formulare

- aspx-Datei: Kasse.aspx

```
<%@ Page Language="C#" Inherits="AdderPage"
Src="Kasse.aspx.cs" %>
<html><head><title>Kassenstand</title></head>
  <body>
    <form method="post" Runat="server">
      <b>Kassenstand:</b>
      <asp:Label ID="total" Runat="server"> 0 </asp:Label>
      Euro<br><br>
      <asp:TextBox ID="amount" Runat="server"/>
      <asp:Button ID="ok" Text="Einzahlen" Runat="server"
       onclick="HandleClick"/>
    </form>
  </body>
</html>
```

- Elemente über Property ID im Code ansprechbar

ASP.NET

- Codebehind-Datei: Kasse.aspx.cs

```
... // erforderliche using-Anweisungen
public class AdderPage : Page {
  protected Label total;          // siehe ID
  protected TextBox amount;       // siehe ID
  protected Button ok;            // siehe ID

  public void HandleClick(object sender, EventArgs e) {
    int totalVal = Convert.ToInt32(total.Text);
    int amountVal = Convert.ToInt32(amount.Text);
    total.Text = (totalVal + amountVal).ToString();}
}
```

- alle Steuerelemente der aspx-Datei sind durch Felder in der Codebehind-Datei repräsentiert
- ASP konvertiert eigene Steuerelemnte in Standard-HTML
- <input type="hidden"...> enthält Formularzustand (siehe HTML-Quellcode im Browser)

ASP.NET

ASP 2.0

- verwendet CodeBeside statt CodeBehind (weiter möglich)
- Änderung in aspx-Datei:

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

- Änderung in Quellcodedatei:

```
... // erforderliche using-Anweisungen
public partial class AdderPage : Page {
//  protected Label total;          // siehe ID
//  protected TextBox amount;       // siehe ID
//  protected Button ok;            // siehe ID
  ...}
```

- Felder der partiellen Klasse automatisch erstellt

ASP.NET

Ereignisbehandlung von ASP.NET

- Webformulare folgen ereignisgesteuertem Modell
- Benutzerinteraktion löst Ereignis aus
- Ereignisse werden auch automatisch (vom System) ausgelöst (Bsp.: Laden einer Seite)
- Varianten der Ereignisbehandlung
  - Ereignis wird am Browser behandelt z.B. durch Javascript-Code der am Client ausgeführt wird
  - Ereignis soll am Server behandelt werden
    => 1. Ereignis + Seiteninhalt an Server senden
       2. Methode am Server ausführen und Seite erstellen
       3. Seite an Browser zurücksenden
    Zyklusname: Rundreise (round trip)

  Frage: Führen alle ausgelösten Ereignisse zu einer Rundreise?

Typisierung der Ereignisse

- Rücksendeereignisse (postback events)
  - lösen Rundreise aus
  - Bsp.: Button-Click

- verzögerte Ereignisse (cached events)
  - lösen keine Rundreise aus
  - Ereignis wird zwischengespeichert und erst bei der nächsten Rundreise behandelt
  - Bsp.: ändern Textfeld

- Rundreise auslösen bei verzögertem Ereignis
  - setze Attribut: AutoPostBack="true"
  - Bsp.:
```
<asp:TextBox Id="Name" Runat="server"
             AutoPostBack="true"/>
```
    - TextChanged-Event führt hier zur Rundreise

ASP.NET

Lebenszyklus einer Seite

- bei jeder Rundreise wird neues Seitenobjekt erzeugt
- Seitenobjekt durchläuft feste Verarbeitungsschritte (siehe unten)
- Seitenobjekt generiert HTML-Code
- HTML-Code wird an Klienten gesendet
- HTML-Code von Browser angezeigt

- Lebenszyklus:
  1. Erzeugung Seitenobjekt mit allen Steuerelementen und Verkettung im Baum
  2. Intialisierung der Steuerelemente mit init-Ereignis
  3. Laden alter Zustand und einfügen der Veränderungen
  4. Behandeln aller Ereignisse
  5. Abbilden nach HTML (Render-Methode)
  6. Entladen (Dispose-Methode und Unload-Ereignis)

Verarbeitungsklassen der ASP-Pipeline

- HttpApplication
  Speicherort für Daten während der Lebensdauer einer Applikation
- HttpContext
  Daten der aktuellen Anfrage
  z. B. HttpRequest und HttpResponse
- HttpModules
  Erlauben komplexe anwendungsweite Operationen
  z. B. Authorisierung und Authentifizierung
- HttpHandler (Schnittstelle: IhttpHandler)
  Werden am Ende der Verarbeitungs-Pipline Ausgeführt

ASP.NET

Steuerelemente für Webformulare

- vordefinierte Steuerelemente
  - TextBox
  - Button
  - Label
  - DataGrid
  - GridView
  - . . .
- Basisklasse aller Steuerelemente: Control
- WebControl
  - von Control abgeleitet
  - ist Basisklasse aller grafischen Steuerelemente

- Eigene Steuerlemente sind definierbar als
  - zusammengesetzte Steuerelemente (User Controls)
  - selbstgeschriebene Steuerelemente (Custom Controls)

ASP.NET

Zusammengesetzte Steuerelemente

- sind Zusammenfassung von HTML-Code und vorhandenen Steuerelementen
- werden in .ascx Datei gespeichert
- von zusammengesetzten Steuerelementen ausgelöste Ereignisse müssen in dieser ascx- Datei (bzw. im Hintergrundcode) behandelt werden
- Verwendung:
  - mit Register-Direktive

    Bsp.: <% Register TagPrefix="mein" TagName=
    "CompSteuerEl" Src="CompSteuerEl.ascx" %>
    ...
    <mein:CompSteuerEl ID = .../>

  - .ascx-Datei muss angegeben werden
- Bsp. MoneyField.aspx

ASP.NET

Selbstgeschriebene Steuerlemente

- frei definierbar solange auf HTML abbildbar
- müssen von Control oder deren Unterklassen abgeleitet sein
- müssen immer Methode Render überschreiben

- Verwendung
  - müssen vor Verwendung in Assembly compiliert werden
  - müssen im Unterverzeichnis bin des virtuellen Verzeichnisses abgelegt werden
  - Register-Direktive in verwendender aspx-Datei:

```
<% Register TagPrefix="mein" Namespace=
        "Test" Assembly="TestCustCtrl" %>
```

- Eigene Ereignisse können ausgelöst werden
- Bsp.: Fold.aspx

ASP.NET

Validatoren

- dienen der Überprüfung von Benutzereingaben sowie der Minimierung der Rundreise (Performance)
- sind eigene Steuerelemente
- sind von Klasse BaseValidator abgeleitet
  BaseValidator ist von Label abgeleitet
- jedem Validator ist Steuerelement zugeordnet
- im Fehlerfall wird auch Property IsValid der Seite auf „false" gesetzt
- Plausibilitätsprüfungen können am Client (Skriptsprache) und/oder am Server stattfinden
- ASP.NET implementiert Validatoren, die Javascript unterstützen in clientseitig ausführbarem Javascriptcode (=> keine unnötige Rundreise)
  Falls Browser nicht Javascript-fähig ist erfolgt die Überprüfung nur am Server

ASP.NET

- vordefinierte Validatoren
  - RequiredFieldValidator
  - RangeValidator
  - CompareValidator (benötigt 2 Steuerelemente)
  - RegularExpressionValidator
  - CustomValidator
  - Validation Summary

Beispiel: Verwendung eines RangeValidators

```
<asp:RangeValidator ID="ageVal"
 ControlToValidate="age" Text="*"
 MinimumValue="0" MaximumValue="100" Type="Integer"
 ErrorMessage="Das Alter muss zwischen 0 und 100 liegen"
Runat="server" />
```

ASP.NET

Zustandsverwaltung

- Klassifikation: Seiten-, Sitzungs-, Applikationszustand
- Seitenzustand
  - wird in verstecktem Feld _VIEWSTATE verpackt und auf Rundreise mitgeschickt
  - Seite hat Indexer ViewState (Dictionary) in der man Werte ablegen kann Bsp.: ViewState["Zaehler"] = 1;
- Sitzungszustand
  - Sitzung beginnt mit erstem Zugriff auf Webseite eines virtuellen Verzeichnisses und läuft nach einer gewissen Zeit automatisch aus
  - Zeit kann über Property Timeout verlängert werden
  - jede Sitzung wird durch eindeutigen Bezeichner SessionID identifiziert (abfragbar)
  - Sitzungszustand ist in Property Session der Webseite Gespeichert.Bsp.:Session["cart"] = Warenkorb;//Indexer

ASP.NET

- Applikationszustand
  - alle Zugriffe auf Seiten eines virtuellen Verzeichnisses bilden Applikation
  - Zustand wird in Property Application gespeichert
    Aufgrund potentiell konkurrierender Zugriffe sollte Zugriff mit Lock versehen werden:

```
Bsp.:   Application.Lock();
        Application["Datenbank"] = Datenbankname;
        Application.UnLock();
```

  - Zustandswerte (Name-Wert Paare) können über Index angesprochen werden (Indexer)
  - Applikation wird beim ersten Zugriff auf eine Webseite gestartet und kann nicht explizit beendet werden
  - Applikation läuft solange bis WebServer neu gestartet wird

ASP.NET

Die zentrale Klasse Page

```
public class Page : TemplateControl,IHttpHandler{
 . . .
 // Properties
 public virtual ControlCollection Controls{get;}
 public ValidatorCollection Validators{get;}
 public bool IsValid{get;}
 public bool IsPostBack{get;}
 public HttpApplicationState Application{get;}
 public virtual HttpSessionState Session{get;}
 public HttpRequest Request{get;}
 public HttpResponse Response{get;}
 . . .
}
```

ASP.NET

Wichtige Properties der Klasse Page

- Controls: Sammlung aller Steuerelemente der Webseite
- Validators: Sammlung aller Validatoren der Webseite
- Seitenproperty IsValid wird auf false gesetzt, falls Property IsValid von mindestens einem Validator false ist
- IsPostBack ist true wenn die Seite bei einer Rundreise zurückgeschickt wird. Beim ersten Zugriff ist sie false
- Application liefert Applikationszustand
- Session liefert Sitzungszustand
- Request verpackt den Clientauftrag (Clientanforderung der Webseite an den Server)
- Response enthält Properties und Methoden zum Aufbau der Html-Antwort des Servers an den Client

ASP.NET

Authentifizierung Login Steuerelement

- spezielle Steuerelemente (ab ASP 2.0)unterstützen Login- und Authentifizierungsvorgang
- Steuerelemente:
  - LoginStatus: prüft ob Benutzer schon angemeldet ist
  - LoginView: zeigt Texte in Abhängigkeit vom Status an
  - LoginName: zeigt Anmeldenamen an
  - Login: vollständiger Logindialog

    <asp: Login Runat="server"/>
  - PasswordRecovery: Unterstützung für vergessene Passworte