DZone Refcardz

CONTENTS INCLUDE:

Core ASP.NET

By Holger Schwichtenberg

ABOUT ASP.NET

ASP.NET stands for "Active Server Pages .NET", however the full name is rarely used. ASP.NET is a framework for the development of dynamic websites and web services. It is based on the Microsoft .NET Framework and has been part of .NET since Version 1.0 was released in January 2002. The current version named 3.5 Service Pack 1 was released in August 2008. The next version, 4.0, is expected to be released at the end of the year 2009.

This Refcard summarizes the most commonly used core functions of ASP.NET. You will find this Refcard useful for some of the most common tasks with ASP.NET, regardless of the version you are using

INSTALLATION

The best development environment for ASP.NET is Microsoft's Visual Studio. You can either use the free Visual Web Developer Express Edition (http://www.microsoft.com/express/vwd/) or any of the commercial editions of Visual Studio (e.g. Visual Studio Professional). The latest version that supports ASP.NET 2.0 and ASP.NET 3.5 is "2008" (internal version: 9.0). The .NET Framework and ASP.NET are part of the setup of Visual Web Developer Express Edition and Visual Studio. However, make sure you install Service Pack 1 for Visual Studio 2008, as this will not only fix some bugs but also add a lot of new features.

ASP.NET needs a server with the HTTP protocol (web server) to run. Visual Web Developer Express 2005/2008 and Visual Studio 2005/2008 contain a webserver for local use on your development machine. The "ASP.NET Development Server" (ADS) will be used when specifying a "File System" location when creating your project. Thus, "HTTP" would mean you address a local or remote instance of Internet Information Server (IIS) or any other ASP.NET enabled web server. ADS is a lightweight server that cannot be reached from other systems. However, there are differences between ADS and IIS, especially in the security model that makes it sometimes hard for beginners to deploy a website to the IIS that was developed with ADS. On the production system you will use IIS and only install the .NET Framework, because Visual Studio is not required here.

Hot Tip If you choose to use Internet Information Server (IIS), install the IIS on your machine before installing the .NET Framework or Visual Studio. If you did not follow this installation order, you may use `aspnet_regiis.exe` to properly register ASP.NET within the IIS.

ASP.NET WEB APPLICATIONS

An ASP.NET application consists of several .aspx files. An .aspx file can contain HTML markup and special ASP.NET markup (called Web Controls) as well as the code (Single Page Model). However, the Code Behind Model which comes with a separate code file called, the "Code Behind File" (.aspx.cs or .aspx.vb), provides a cleaner architecture and better collaboration between Web designers and Web developers. ASP.NET applications may contain several other elements such as configuration files (maximum one per folder), a global application file (only one per web application), web services, data files, media files and additional code files.

There are two types of Web projects: "Website Projects" (File/New/Web Site) and "Web Application Projects" (File/New/Project/Web Application). "Website" is the newer model, while Web Application Projects mainly exist for compatibility with Visual Studio .NET 2002 and 2003. This Refcard will only cover Web Site Projects. Most of this content is also valid for Web Applications.

Hot Tip A well designed ASP.NET application distinguishes itself by having as little code in the Code Behind files and other code files as possible. The large majority of your code should be in referenced Assemblies (DLLs) as they are reusable in other Web applications. If you don't want to put your code into a separate assembly, you at least should use separate classes in the "App_Code" folder within your web project.

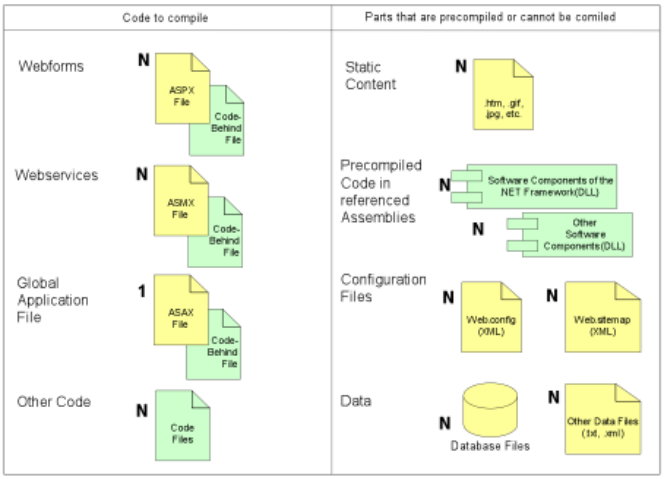


Figure 1: The Content of an ASP.NET Web Application

THE ASP.NET WEBFORM MODEL

ASP.NET uses an object- and event-oriented model for web pages. The ASP.NET Page Framework analyzes all incoming requests as well as the .aspx page that the request is aimed at. The Page Framework creates an object model (alias control tree) based on this information and also fires a series of events. Event handlers in your code can access data, call external code in referenced .NET assemblies and manipulate the object model (e.g. fill a listbox or change the color of a textbox). After all event handlers have executed, the Page Framework renders the current state of the object model into HTML tags with optional CSS formatting, JavaScript code and state information (e.g. hidden fields or cookies). After interacting with the page, the user can issue a new request by clicking a button or a link that will restart the whole process.

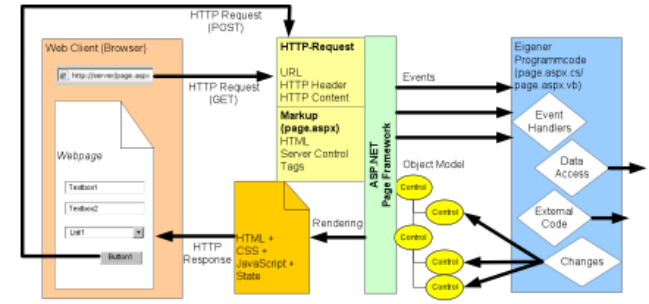


Figure 2: The ASP.NET request/response life cycle

WEB CONTROLS

An ASP.NET page can contain common HTML markup. However, only ASP.NET web controls provide full object- and event-based functionality. Web controls have two representations: In the .aspx files they are tags with the prefix "asp:", e.g. <asp:TextBox>. In the code they are .NET classes, e.g. System.Web.UI.WebControls.TextBox.

Table 1 lists the core members of all web controls that are implemented in the base class "System.Web.UI.WebControls.WebControl".

Name of Member	Description
Id	Unique identifier for a control within a page
ClientID	Gets the unique identifier that ASP.NET generates if more than one control on the page has the same (String) ID.
Page	Pointer to the page where the control lives
Parent	Pointer to the parent control, may be the same as "Page"
HasControls()	True, if the control has sub-controls
Controls	Collection of sub-controls
FindControl("NAME")	Finds a sub-control within the Controls collection by its ID
BackColor, BorderColor, Borderstyle, BorderWidth, Font, ForeColor, Height, Width, ToolTip, TabIndex	Self-explaining properties for the formatting of the control.
CssClas	The name of CSS class that is used for formatting the control
Style	A collection of single CSS styles, if you don't want to use a CSS class or override behavior in a CSS class
EnableViewState	Disables page-scoped state management for this control
Visible	Disables rendering of the control
Enabled	Set to false if you want the control to be disabled in the browser
Focus()	Set the focus to this control
DataBind()	Gets the data (if the control is bound to a data source)
Init()	Fires during initializtion of the page. Last chance to change basic setting e.g. the culture of the current thread that determines the behavior used for rendering the page.
Load()	Fires during the loading of the page. Last to change to do any preparations.
PreRender()	Fires after all user defined event handlers have completed and right before rendering of the page starts. Your last chance to make any changes to the controls on the page!
UnLoad()	Event fires during the unloading of a page.

Table 1: Core Members in the base class system. Web.UI.WebControls.WebControl.

Tables 2, 3 and 4 list the most commonly used controls for ASP.NET web pages. However, there are more controls included in the .NET platform and many more from third parties not mentioned here.

Control	Purpose	Important specific members in addition to the members inherited from WebControl
`<asp:Label>`	Static Text	Text
`<asp:TextBox>`	Edit Text (single line, multiline or password)	TextMode, Text, TextChanged()
`<asp:FileUpload>`	Choose a file for upload	FileName, FileContent, FileBytes, SaveAs()
`<asp:Button>`	Display a clasic button	Click(), CommdName, Command()
`<asp:ImageButton>`	Display a clickable image	Click(), CommdName, Command()
`<asp:LinkButton>`	Display a hyperlink that works like a button	ImageUrl, ImageAlign, Click(), CommdName, Command()
`<asp:CheckBox>`	Choose an option	Text, Checked, CheckedChanged()
`<asp:RadioButton>`	Choose an option	Text, Checked, CheckedChanged()
`<asp:HyperLink>`	Display a hyperlink	NavigateURL, Target, Text
`<asp:Image>`	Display an image	ImageURL, ImageAlign
`<asp:ImageMap>`	Display a clickable image with different regions	ImageURL, ImageAlign, HotSpots, HotSpotsMode, Click()

Table 2: Core controls for ASP.NET web pages.

List controls display several items that the user can choose from. The selectable items are declared static in the .aspx file or created manually using the `Items` collection or created automatically by using data binding. For data binding you can fill `DataSource` with any enumerable collection of .NET objects. `DataTextField` and `DataValueField` specify which properties of the objects in the collection are used for the list control.

Hot Tip If you bind a collection of primitive types such as strings or numbers, just leave `DataTextField` and `DataValueField` empty.

Hot Tip Setting `AppendDataBoundItems` to true will add the databound items to the static items declared in the .aspx file. This will allow the user to select values that don't exist in the data source such as the values "All" or "None"

Control	Purpose	Important specific members in addition to the members inherited from WebControl
`<asp:DropDownList>`	Allows the user to select a single item from a drop-down list	Items.Add(), Items.Remove(), DataSource,, DataTextField, DataValueField, AppendDataBoundItems, SelectedIndez, SelectedItem, SelectedValue, SelectedIndexChanged()
`<asp:ListBox>`	Single or multiple selection box	Items.Add(), Items.Remove(), DataSource,, DataTextField, DataValueField, AppendDataBoundItems, SelectedIndez, SelectedItem, SelectedValue, SelectedIndexChanged(), Rows, SelectionMode
`<asp:CheckBoxList>`	Multi selection check box group	Items.Add(), Items.Remove(), DataSource,, DataTextField, DataValueField, AppendDataBoundItems, SelectedIndez, SelectedItem, SelectedValue, SelectedIndexChanged(), RepeatLayout, RepeatDirection
`<asp:RadioButtonList>`	Single selection radio button group	Items.Add(), Items.Remove(), DataSource,, DataTextField, DataValueField, AppendDataBoundItems, SelectedIndez, SelectedItem, SelectedValue, SelectedIndexChanged(), RepeatLayout, RepeatDirection
`<asp:BulletedList>`	List of items in a bulleted format	Items.Add(), Items.Remove(), DataSource,, DataTextField, DataValueField, AppendDataBoundItems, SelectedIndez, SelectedItem, SelectedValue, SelectedIndexChanged(), BulletImageUrl, BulletStyle

Table 3: List Controls for ASP.NET web pages

Validation Controls check user input. They always refer to one input control `ControlToValidate` and display a text `ErrorMessage` if the validation fails. They perform the checks in the browser using JavaScript and also on the server. The client side validation can be disabled by setting `EnableClientScript` to false. However, the server side validation cannot be disabled for security reasons.

Control	Purpose	Important specific members in addition to the members inherited from WebControl
`<asp:RequiredFieldValidator>`	Checks if a user changed the initial value of an input control	ControlToValidate, ErrorMessage, Display, EnableClientScript, IsValid, InitialValue
`<asp:CompareValidator>`	Compares the value entered by the user in an input control with the value entered in another input control, or with a constant value	ControlToValidate, ErrorMessage, Display, EnablesClientScript, IsValid, ValueToCompare, Type, ControlToCompare
`<asp:RangeValidator>`	Checks whether the value of an input control is within a specified range of values	ControlToValidate, ErrorMessage, Display, EnableClientScript, IsValid, MinimumValue, MaximumValue, Type
`<asp:RegularExpressionValidator>`	Checks if the user input matches a given regular	ControlToValidate, ErrorMessage, Display, EnavleClientScript, IsValid, ValidationExpression
`<asp:CustomValidator>`	Performs custom checks on the server and optional also on the client using JavaScript	ControlTValidate, ErrorMessage, Display, EnableClientScript, IsValid, ValidateEmptyText, Client ValidationFunction, ServerValidate()

Table 4: Validation controls for ASP.NET web pages

Hot Tip For the `CustomValidator` you can optionally write a JavaScript function that performs client side validation. The function has to look like this:

```
<script type="text/javascript">
   function ClientValidate(source, args)
   {
      if (x > 0) // Any condition
      { args.IsValid=true; }
      else
      { args.IsValid=false; }
   }
</script>
```

THE PAGE CLASS

All web pages in ASP.NET are .NET classes that inherit from the base class "System.Web.UI.Page". The class `Page` has associations to several other objects such as `Server`, `Request`, `Response`, `Application`, `Session` and `ViewState` (see figure 3). Therefore, developers have access to a wide array of properties, methods and events within their code. Table 5 lists the most important members of a `Page` and its dependent classes. Please note that the `Page` class has the class `Control` in its inheritance hierarchy and therefore shares a lot of members with the `WebControl` class (e.g. `Init()`, `Load()`, `Controls`, `FindControl`). However, these members are not repeated here.

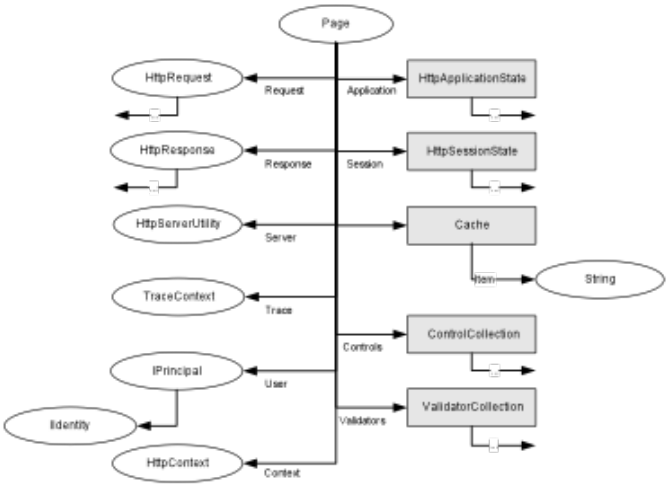


Figure 3: Object Model of "System.Web.UI.Page"

Member	Description
`Page Title`	Title string of the Page
`Page.IsPostBack`	True, if page is being loaded in response to a client postback. False if it is being loaded for the first time.
`Page.IsAsync`	True, if the page is loaded in an asynchronous request (i.e. AJAX request)
`Page.IsValid`	True, if all validation server controls in the current validation group validated successfully
`Page.Master`	Returns the MasterPage object associated with this page
`Page.PreviousPage`	Gets the page that transferred control to the current page (only available if using Server.Transfer, not available with Response.Redirect)
`Page.SetFocus(Control ControlID)`	Sets the browser focus to the specified control (using JavaScript)
`Trace.Write`	Writes trace information to the trace log.
`User.Identity.IsAuthenticated`	True, if the user has been authenticated.

User.Identity.AuthenticationType	Type of Authentication used (Basic, NTLM, Kerberos, etc)
User.Identity.Name	Name of the current user
Server.MachineName	Name of the computer the web server is running on
Server.GetLastError()	Gets the Exception object for the last exception
Server.HtmlEncode(Text)	Applies HTML encoding to a string
Server.UrlEncode(Pah)	Applies URL encoding to a string
Server.MapPath(Path)	Maps the given relative path to an absolute path on the web server
Server.Transfer(Path)	Stops the execution of the current page and starts executing the given page as part of the current HTTP request
Request.AcceptTypes	String array of cliet-supported MIME accept types.
Request.Browser	Provides information about the browser
Request.ClientCertificate	Provides the certificate of the client, if SSL client authentication is used
Request.Cookies	The list of cookies that the browser sent to the web server
Request.Form	The name and value of the input fields the browser sent to the web server
Request.Headers	Data from the HTTP header the browser sent to the web server
Request.IsAuthenticated	True, if the user is authenticated
Request.IsSecureConnection	True, if SSL is used
Request.Path	Virtual path of the HTTP request (without server name)
Request.QueryString	Name/Value pairs the browser sent as part of the URL
Request.ServerVariables	Complete list of name/value pairs with information about the server and the current request
Request.Url	Complete URL of the request
Request.UrlReferrer	Refering URL of the request (Previous page, the browser visited)
Request.UserAgent	Browser identification
Request.UserHostAddress	IP address of the client
Request.UserLanguages	Preferred languages of the user (determined by browser settings)
Response.BinaryWrite(bytes)	Writes information to an HTTP response output stream.
Response.Write(string)	Writes information to an HTTP response output stream.
Response.WriteFile(string)	Writes the specified file directly to an HTTP response output stream.
Response.BufferOutput	True if the output to client is buffered
Response.Cookies	Collection of cookies that shall be sent to the browser
Response.Redirect(Path)	Redirects a client to a new URL using the HTTP status code 302
Response.StatusCode	HTTP status code (integer) of the output returned to the client
Response.StatusDescription	HTTP status string of the output returned to the client
Session.SessionID	Unique identifier for the current session (a session is user specific)
Session.Item	Gets or sets individual session values.
Session.IsCookieless	True, if the ID for the current sessions are embedded in the URL. False, if its stored in an HTTP cookie
ViewState.Item	Gets or sets the value of an item stored in the ViewState, which is a hidden field used for state management witin a page
Application.Item	Gets or sets the value of an item stored in the application state, which is an application-scope state management facility

Table 5: Most important members of the Page class and its associated classes

A TYPICAL PAGE

Figure 4 shows the typical content of an .aspx page and Figure 5 the content of a typical code behind class. The sample used is a registration form with three fields: Name, Job Title and Email Address.

```
<%@ Page Language="C#" AutoEventWireup="true"
  CodeFile="PageName.aspx.cs" Inherits="PageName" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//en"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
 <title>Registration Page</title>
 <link href="MyStyles.css" rel="stylesheet" type="text/css"/>
 <style type="text/css">
  .Headline
  {
   font-size: large; font-weight: bold;
  }
 </style>
</head>
<body>
 <form id="c_Form" ruanat="server">
 <div>
  <asp:Label runat="server" ID="C_Headline" Text="Please register:"
    class="Headline"></asp:label>
  <p>Name:
   <asp:TextBox ID="C_Name" runat="server"></asp:TextBox>
   <asp:RequiredFieldValidator ID="C_NameVal" ControlToValidate="C_
    Name" ruanat="server" ErrorMessage="Name required"></
    asp:RequiredFieldValidator>
  </p>
  <p>Job Title:
   <asp:DropDownList ID="C_JobTitle" runat="server">
    <asp:ListItem Text="Software Developer" Value="SD"></
     asp:ListItem>
    <asp:ListItem Text="Software Architect" Value="SA"></
     asp:ListItem>
   </asp:DropDownList>
  </p>
  <p>EMail:
   <asp:TextBox ID="C_EMail" runat="server"></asp:TextBox>
   <asp:RequiredFieldValidator Id="C_EMailVal1" ControlToValidate="C_
    EMail" runat="server" ErrorMessage="EMail required"></
    asp:RequiredFieldValidator>
   <asp:RegularExpressionValidator ID="C_EMailVal2"
    ControlToValidate="C_EMail" runat="server" ErrorMessage="Email
    not valid" ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\
    w+)*\.\w+([-.]\w+)*">
   </asp:RegularExpressionValidator>
  </p>
  <p>
   <asp:Button ID="C_register" runat="server" Text="Register"
    onclick="C_Register_Click"/>
  </p>
 </div>
 </form>
</body>
</html>
```

Page Directive
HTML Tags
Reference to Style Sheet Files
Inline Styles
Form Tag(exactly one per page)
Dynamic Label
Static Labels
Web Control with Subcontrols
Simple Web Control
Validation Controls

Figure 4: Typical content of an ASPX file

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class PageName : System.Web.UI.Page
 {
  protected void Page_Load(object sender, EventArgs e)
   {
    // If an authenticated users starts using this page,
    // use his login name in the name textbox
    if (!Page.IsPostBack && Page.User.Identity.IsAuthenticated)
    {
     this.C_Name.Text = Page.User.Identity.Name;
     this.C_Name.Enabled = false;
    }
   }
```

All page classes must derive from "page"
The "Load" event is fired every time the Page is requested
All page classes must derive from "page"

```
protected void C_Register_Click(object sender, EventArgs e)
 {
  if (Page.IsValid) // if all validation controls succedded
  { // call business logic and
   if (BL.Register(this.C_Name.Text, this.C_EMail.Text, this.C_
    JobTitle.SelectedValue))
   { // redirect to confirmation page
    Response.Redirect("RegistrationConfirmation.aspx");
   }
   else
   { // change the headline
    this.C_Headline. = "You are already registered!";
   }
  }
 }
}
```

Reaction to a User's Action

Call Business Logic

Redirect to annother Page

Changing a Property of a Webcontrol

Figure 5: Typical content of a Code Behind file

STATE MANAGEMENT

State management is a big issue in web applications as the HTTP protocol itself is stateless. There are three standard options for state management: hidden files, URL parameters and cookies. However, ASP.NET has some integrated abstractions from these base mechanisms know as `View State`, `Session State`, and `Application State`. Also, the direct use of cookies is supported in ASP.NET.

Hot Tip Disabling the View State (EnableViewState=false in a control) will significantly reduce the size of the page sent to the browser. However, you will have to take care of the state management of the controls with disabled View State on your own. Some complex controls will suffer the loss of functionality without View State.

The following code snippet shows how to set values for a counter stored in each of these mechanisms:

```
ViewState["Counter"] = CurrentCounter_Page + 1;
Session["Counter"] = CurrentCounter_Session + 1;
Application["Counter"] = CurrentCounter_Application + 1;
Response.Cookies["Counter"].Value = (CurrentCounter_User +
   1).ToString();
Response.Cookies["Counter"].Expires = DateTime.MaxValue; // no
   expiration
```

Figure 6: Setting Values

Hot Tip When reading value from these objects, you have to check first if they already exist. Otherwise you will recieve the exception "NullReferenceExeception: Object reference not set to an instance of an object."

The following code snippet shows how to read the current counter value from each of these mechanisms: Next Column ---->

```
long CurrentCounter_Application, CurrentCounter_ApplicationLimited,
CurrentCounter_Session, CurrentCounter_Page, CurrentCounter_User;
if (Application["Counter"] == null) { CurrentCounter_Application =
 0; }
 else  { CurrentCounter_Application = Convert.ToInt64(Application["C
  ounter"]); }
if (Session["Counter"] == null)  { CurrentCounter_Session = 0; }
 else { CurrentCounter_Session = Convert.
  ToInt64(Session["Counter"]);}
if (ViewState["Counter"] == null) { CurrentCounter_Page = 0; }
 else {   CurrentCounter_Page = Convert.
  ToInt64(ViewState["Counter"]);  }
if (Request.Cookies["Counter"] == null)  { CurrentCounter_User = 0; }
 else { CurrentCounter_User = Convert.ToInt64(Request.
  Cookies["Counter"].Value); }
```

Figure 7: Reading Values

CONFIGURATION

All configurations for ASP.NET applications are stored in XML-based configuration files with the fixed name "web.config". In addition to the configuration files in the application root folder, subfolders may also contain a web.config that overrides parent settings. Also, there are the global configuration files machine.config and web.config in the folder \Windows\Microsoft.NET\Framework\v2.0.50727\CONFIG that provide some default settings for all web applications. (Note: v2.0.50727 is still correct for ASP.NET 3.5!).

Visual Studio and Visual Web Developer create a default root configuration file in your web project that contains a lot of internal setting for ASP.NET 3.5 to work properly. Figure 6 shows a fragment from a web.config file with settings that are often used.

```
<!-- Connection strings -->
<connectionStrings>
 <add name="RegistrationDatabase" connectionString="Data
Source=EO2;Initial Catalog= RegistrationDatabase;Integrated
Security=True" providerName="System.Data.SqlClient" />
</connectionStrings>
<!-- User defined settings -->
<AppSettings>
 <and key="WebmasterEMail" value="hs@IT-Visions.de" />
</appSettings>

<system.web>
 <!-- Specify a login page -->
 <!-- Use the URL for storing the authentication ID if cookies are
not allowed -->
 <!-- Set the authentication timeout to 30 minutes -->
 <authentication mode="Forms">
  <forms loginUrl="Login.aspx" cookieless="AutoDetect" timeout="30">
  </forms>
 </authentication>
 <!-- Deny all unauthorizd access to this application -->
 <authorization>
  <deny users="?" />
 </authorization>
 <!-- Use the URL for storing the session ID if cookies are not
allowed -->
 <!-- Set the session timeout to 30 minutes -->
 <sessionState cookieless="AutoDetect" timeout="30"></sessionState>
```

Mechanism	Scope	Lifetime	Base Mechanism	Data Type	Storing Value	Reading Value
View State	Single user on a single page	Leaving the current page	Hidden FIeld "ViewState"	Object (any serializable.NET data type)	Page.ViewState	Page.ViewState
Session State	Latest interaction of a single user with the web page	Limited number of minutes after the last request from the user	Cookie ("ASPSessionJD...") or URL Parameter "(S(...))" plus server side store (local RAM, RAM on dedicated server or database)	Object. Object must be serializable if the store is not the local RAM	Page.Session	Page.Session
Cookies	A single User	Closing of the browser or dedicated point in time	Cookie	String	Page.Response.Cookies	Page.Response.Cookies
Application State	All users	Shutting down the web application	Local RAM	Object	Page.Application	Page.Application

```
<!-- Display custom error pages for remote users -->
<customErrors mode="RemoteOnly" defaultRedirect="GenericErrorPage.
  htm">
 <error statusCode="403" redirect="NoAccess.htm" />
 <error statusCode="404" redirect="FileNotFound.htm" />
</customErrors>
<!-- Turn on debugging -->
<compilation debug="false">
```

Figure 8: Typical setting in the web.config file.

Hot Tip Please make sure you turn debugging off again before deploying your application as this decreases execution performance.

DEPLOYMENT

ASP.NET applications can be deployed as source code via the so called "XCopy deployment". This means you copy the whole content of the web project folder to the production system and configure the target folder on the production system as an IIS web application (e.g. using the IIS Manager). The production web server will automatically compile the application during the first request and recompile automatically if any of the source files changed.

However, you can precompile the application into .NET assemblies to improve protection of your intellectual property and increase execution speed for the first user. Precompilation can be performed through Visual Studio/Visual Web developer (Menu "Build/Publish Website") or the command line tool aspnet_compiler.exe.

Hot Tip Download the "Visual Studio 2008 Web Deployment Projects" from microsoft.com. This is an Add-In that provides better control over the precompilation process.

ABOUT THE AUTHOR

Holger Schwichtenberg is one of Europe's best-known experts on .NET and Windows PowerShell. He holds both a Master's degree and a Ph.D. in business informatics. Microsoft recognizes him as a Most Valuable Professional (MVP) since 2003. He is a .NET Code Wise Member, an MSDN Online Expert and an INETA speaker. He regularly gives high-level talks at conferences such as TechEd, Microsoft Summit, BASTA and IT Forum. He is the CEO of the German based company www.IT-Visions.de that provides consulting and training for many companies throughout Europe.

Publicaitions

Holger Schwichtenberg has published more than twenty books for Addison Wesley and Microsoft Press in Germany, as well as about 400 journal articles. His recent book "Essential PowerShell" has also been published by Addison Wesley in English.

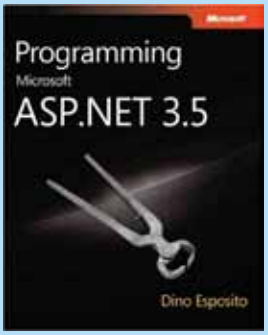
Blog

www.dotnet-doktor.de (German)

Website

www.IT-Visions.de/en

RECOMMENDED BOOK



An in-depth guide to the core features of Web development with ASP.NET, this book goes beyond the fundamentals. It expertly illustrates the intricacies and uses of ASP.NET 3.5 in a single volume. Complete with extensive code samples and code snippets in Microsoft Visual C# 2008, this is the ideal reference for developers who want to learn what s new in ASP.NET 3.5, or for those building professional-level Web development skills.

BUY NOW

books.dzone.com/books/programming-asp-net

DZone

DZone communities deliver over 4 million pages each month to more than 1.7 million software developers, architects and decision makers. DZone offers something for everyone, including news, tutorials, cheatsheets, blogs, feature articles, source code and more. **"DZone is a developer's dream,"** says PC Magazine.

DZone, Inc.
1251 NW Maynard
Cary, NC 27513

888.678.0399
919.678.0300

Refcardz Feedback Welcome
refcardz@dzone.com

Sponsorship Opportunities
sales@dzone.com

ISBN-13: 978-1-934238-49-3
ISBN-10: 1-934238-49-X
50795
9 781934 238493
$7.95

Version 1.0