ASP.NET

From Wikipedia, the free encyclopedia

Not to be confused with UNESCO ASPNet.

ASP.NET is a Web application framework developed and marketed by Microsoft to allow programmers to build dynamic Web sites, Web applications and Web services. It was first released in January 2002 with version 1.0 of the .NET Framework, and is the successor to Microsoft's Active Server Pages (ASP) technology. ASP.NET is built on the Common Language Runtime (CLR), allowing programmers to write ASP.NET code using any supported .NET language. The ASP.NET SOAP extension framework allows ASP.NET components to process SOAP messages.

ASP.NET

Initial release	January 2002
Stable release	4.0.30319.1 (4.0) / 12 April 2010; 2 years ago
Written in	.NET languages
Operating system	Microsoft Windows
Type	Web application framework
License	Proprietary
Website	www.asp.net

ASP.NET

Filename extension	`.aspx`
Internet media type	`text/html`
Developed by	Microsoft

Contents [hide]

Español
فارسی
Suomi
Français
Galego
한국어
हिन्दी
Bahasa Indonesia
Íslenska
Italiano
עברית
Basa Jawa
Lietuvių
مازِرونی
Bahasa Melayu
□□□□□□□□□□□
Nederlands
日本語
Norsk (bokmål)
Polski
Português
Română
Русский
Shqip
Simple English
Slovenčina
Српски / Srpski
Svenska
தமிழ்
ไทย
Türkçe
Українська
Tiếng Việt
中文

History

[edit]

After the release of Internet Information Services 4.0 in 1997, Microsoft began researching possibilities for a new Web application model that would solve common complaints about ASP, especially with regard to separation of presentation and content and being able to write "clean" code.[1] Mark Anders, a manager on the IIS team, and Scott Guthrie, who had joined Microsoft in 1997 after graduating from Duke University, were tasked with determining what that model would look like. The initial design was developed over the course of two months by Anders and Guthrie, and Guthrie coded the initial prototypes during the Fall of 1997.[2]

The initial prototype was called "XSP"; Guthrie explained in a 2007 interview that, "People would always ask what the X stood for. At the time it really didn't stand for anything. XML started with that; XSLT started with that. Everything cool seemed to start with an X, so that's what we originally named it."[1] The initial prototype of XSP was done using Java,[3] but it was soon decided to build the new platform on top of the Common Language Runtime (CLR), as it offered an object-oriented programming environment, garbage collection and other features that were seen as desirable features that Microsoft's Component Object Model platform did not support. Guthrie described this decision as a "huge risk", as the success of their new Web development platform would be tied to the success of the CLR, which, like XSP, was still in the early stages of development, so much so that the XSP team was the first team at Microsoft to target the CLR.

With the move to the Common Language Runtime, XSP was re-implemented in C# (known internally as "Project Cool" but kept secret from the public), and the name changed to ASP+, as by this point the new platform was seen as being the successor to Active Server Pages, and the intention was to provide an easy migration path for ASP developers.[4]

Mark Anders first demonstrated ASP+ at the ASP Connections conference in Phoenix, Arizona on May 2, 2000. Demonstrations to the wide public and initial beta release of ASP+ (and the rest of the .NET Framework) came at the 2000 Professional Developers Conference on July 11, 2000 in Orlando, Florida. During Bill Gates' keynote presentation, Fujitsu demonstrated ASP+ being used in conjunction with COBOL,[5] and support for a variety of other languages was announced, including Microsoft's new Visual Basic .NET and C# languages, as well as Python and Perl support by way of interoperability tools created by ActiveState.[6]

Once the ".NET" branding was decided on in the second half of 2000, it was decided to rename ASP+ to ASP.NET. Mark Anders explained on an appearance on *The MSDN Show* that year that, "The .NET initiative is really about a number of factors, it's about delivering software as a link building service, it's about XML and Web services and really enhancing the Internet in terms of what it can do ... we really wanted to bring its name more in line with the rest of the platform pieces that make up the .NET framework."[4]

After four years of development, and a series of beta releases in 2000 and 2001, ASP.NET 1.0 was released on January 5, 2002 as part of version 1.0 of the .NET Framework. Even prior to the release, dozens of books had been written about ASP.NET,[7] and Microsoft promoted it heavily as part of its platform for Web services. Guthrie became the product unit manager for ASP.NET, and development continued apace, with version 1.1 being released on April 24, 2003 as a part of Windows Server 2003. This release focused on improving ASP.NET's support for mobile devices.

Characteristics [edit]

Pages [edit]

ASP.NET Web pages, known officially as Web Forms,[8] are the main building block for application development.[9] Web forms are contained in files with an ".aspx" extension; these files typically contain static (X)HTML markup, as well as markup defining server-side Web Controls and User Controls where the developers place all the required static and dynamic content for the Web page. Additionally, dynamic code which runs on the server can be placed in a page within a block `<% -- dynamic code -- %>`, which is similar to other Web development technologies such as PHP, JSP, and ASP. With ASP.NET Framework 2.0, Microsoft introduced a new *code-behind* model which allows static text to remain on the .aspx page, while dynamic code remains in an .aspx.vb or .aspx.cs or .aspx.fs file (depending on the programming language used).[10]

Directives [edit]

A directive is special instructions on how ASP.NET should process the page.[11] The most common directive is <%@ Page %>

which can specify many attributes used by the ASP.NET page parser and compiler.

Examples [edit]

Inline code [edit]

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "---//W3C//DTD XHTML 1.0  //EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
  protected void Page_Load(object sender, EventArgs e)
  {
    // Assign the datetime to label control
    lbl1.Text = DateTime.Now.ToLongTimeString();

  }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Sample page</title>
</head>
<body>
  <form id="form1" runat="server">

    <div>
      The current time is: <asp:Label runat="server" id="lbl1" />
    </div>
  </form>
</body>
</html>
```

The above page renders with the Text "The current time is: " and the <asp:Label> Text is set with the current time, upon render.

Code-behind solutions [edit]

```
<%@ Page Language="C#" CodeFile="SampleCodeBehind.aspx.cs" Inherits="Website.SampleCodeBehind"
AutoEventWireup="true" %>
```

The above tag is placed at the beginning of the ASPX file. The *CodeFile* property of the @ *Page* directive specifies the file (.cs or .vb or .fs) acting as the code-behind while the *Inherits* property specifies the Class from which the Page is derived. In this example, the @ *Page* directive is included in SampleCodeBehind.aspx, then SampleCodeBehind.aspx.cs acts as the code-behind for this page:

```
using System;
namespace Website
{
  public partial class SampleCodeBehind : System.Web.UI.Page
  {
    protected void Page_Load(object sender, EventArgs e)
    {
      Response.Write("Hello, world");
    }
  }
}
```

```
Imports System
Namespace Website
  Public Partial Class SampleCodeBehind
          Inherits System.Web.UI.Page
          Protected Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
             Response.Write("Hello, world")
          End Sub
  End Class
End Namespace
```

In this case, the Page_Load() method is called every time the ASPX page is requested. The programmer can implement event handlers at several stages of the page execution process to perform processing.

User controls [edit]

User controls are encapsulations of sections of pages which are registered and used as controls in ASP.NET. User controls are created as ASCX markup files. These files usually contain static (X)HTML markup, as well as markup defining server-side Web controls. These are the locations where the developer can place the required static and dynamic content. A user control is compiled when its containing page is requested and is stored in memory for subsequent requests. User controls have their own events which are handled during the life of ASP.NET requests. An event bubbling mechanism provides the ability to pass an event fired by a user control up to its containing page. Unlike an ASP.NET page, a user control cannot be requested independently; one of its containing pages is requested instead.

Custom controls [edit]

Programmers can also build *custom controls* for ASP.NET applications. Unlike user controls, these controls do not have an ASCX markup file, having all their code compiled into a dynamic link library (DLL) file. Such custom controls can be used across multiple Web applications and Visual Studio projects.

Rendering technique [edit]

ASP.NET uses a *visited composites* rendering technique. During compilation, the template (.aspx) file is compiled into initialization code which builds a control tree (the composite) representing the original template. Literal text goes into instances of the Literal control class, and server controls are represented by instances of a specific control class. The initialization code is combined with user-written code (usually by the assembly of multiple partial classes) and results in a class specific for the page. The page doubles as the root of the control tree.

Actual requests for the page are processed through a number of steps. First, during the initialization steps, an instance of the page class is created and the initialization code is executed. This produces the initial control tree which is now typically manipulated by the methods of the page in the following steps. As each node in the tree is a control represented as an instance of a class, the code may change the tree structure as well as manipulate the properties/methods of the individual nodes. Finally, during the rendering step a visitor is used to visit every node in the tree, asking each node to render itself using the methods of the visitor. The resulting HTML output is sent to the client.

After the request has been processed, the instance of the page class is discarded and with it the entire control tree. This is a source of confusion among novice ASP.NET programmers who rely on class instance members that are lost with every page request/response cycle.

State management [edit]

ASP.NET applications are hosted by a Web server and are accessed using the stateless HTTP protocol. As such, if an application uses stateful interaction, it has to implement state management on its own. ASP.NET provides various functions for state management. Conceptually, Microsoft treats "state" as GUI state. Problems may arise if an application needs to keep track of "data state"; for example, a finite-state machine which may be in a transient state between requests (lazy evaluation) or which takes a long time to initialize. State management in ASP.NET pages with authentication can make Web scraping difficult or impossible.

Application [edit]

Application state is held by a collection of shared user-defined variables. These are set and initialized when the `Application_OnStart` event fires on the loading of the first instance of the application and are available until the last instance exits. Application state variables are accessed using the `Applications` collection, which provides a wrapper for the application state. Application state variables are identified by name.[12]

Session state [edit]

Server-side session state is held by a collection of user-defined session variables that are persistent during a user session. These variables, accessed using the `Session` collection, are unique to each session instance. The variables can be set to be automatically destroyed after a defined time of inactivity even if the session does not end. Client-side user session is maintained by either a cookie or by encoding the session ID in the URL itself.[12]

ASP.NET supports three modes of persistence for server-side session variables:[12]

In-Process Mode

The session variables are maintained within the ASP.NET process. This is the fastest way; however, in this mode the variables are destroyed when the ASP.NET process is recycled or shut down.

ASPState Mode

ASP.NET runs a separate Windows service that maintains the state variables. Because state management happens outside the ASP.NET process, and because the ASP.NET engine accesses data using .NET Remoting, ASPState is slower than In-Process. This mode allows an ASP.NET application to be load-balanced and scaled across multiple servers. Because the state management service runs independently of ASP.NET, the session variables can persist across ASP.NET process shutdowns. However, since session state server runs as one instance, it is still one point of failure for session state. The session-state service cannot be load-balanced, and there are restrictions on types that can be stored in a session variable.

SqlServer Mode

State variables are stored in a database, allowing session variables to be persisted across ASP.NET process shutdowns. The main advantage of this mode is that it allows the application to balance load on a server cluster, sharing sessions between servers. This is the slowest method of session state management in ASP.NET.

View state [edit]

View state refers to the page-level state management mechanism, utilized by the HTML pages emitted by ASP.NET applications to maintain the state of the Web form controls and widgets. The state of the controls is encoded and sent to the server at every form submission in a hidden field known as `__VIEWSTATE`. The server sends back the variable so that when the page is re-rendered, the controls render at their last state. At the server side, the application may change the viewstate, if the processing requires a change of state of any control. The states of individual controls are decoded at the server, and are available for use in ASP.NET pages using the `ViewState` collection.[13] [14]

The main use for this is to preserve form information across postbacks. View state is turned on by default and normally serializes the data in every control on the page regardless of whether it is actually used during a postback. This behavior can (and should) be modified, however, as View state can be disabled on a per-control, per-page, or server-wide basis.

Developers need to be wary of storing sensitive or private information in the View state of a page or control, as the base64 string containing the view state data can easily be de-serialized. By default, View state does not encrypt the `__VIEWSTATE` value. Encryption can be enabled on a server-wide (and server-specific) basis, allowing for a certain level of security to be maintained.[15]

Server-side caching [edit]

ASP.NET offers a "Cache" object that is shared across the application and can also be used to store various objects. The "Cache" object holds the data only for a specified amount of time and is automatically cleaned after the session time-limit elapses.

Other [edit]

Other means of state management that are supported by **ASP.NET** are *cookies*, caching, and using the query string.

Template engine [edit]

When first released, ASP.NET lacked a template engine. Because the .NET Framework is object-oriented and allows for inheritance, many developers would define a new base class that inherits from "System.Web.UI.Page", write methods there that render HTML, and then make the pages in their application inherit from this new class. While this allows for common elements to be reused across a site, it adds complexity and mixes source code with markup. Furthermore, this method can only be visually tested by running the application - not while designing it. Other developers have used include files and other tricks to avoid having to implement the same navigation and other elements in every page.

ASP.NET 2.0 introduced the concept of "master pages", which allow for template-based page development. A Web application can have one or more master pages, which, beginning with ASP.NET 2.0, can be nested.[16] Master templates have place-holder controls, called *ContentPlaceHolders* to denote where the dynamic content goes, as well as HTML and JavaScript shared across child pages.

Child pages use those ContentPlaceHolder controls, which must be mapped to the place-holder of the master page that the content page is populating. The rest of the page is defined by the shared parts of the master page, much like a mail merge in a word processor. All markup and server controls in the content page must be placed within the ContentPlaceHolder control.

When a request is made for a content page, ASP.NET merges the output of the content page with the output of the master page, and sends the output to the user.

The master page remains fully accessible to the content page. This means that the content page may still manipulate headers, change title, configure caching etc. If the master page exposes public properties or methods (e.g. for setting copyright notices) the content page can use these as well.

Other files [edit]

Other file extensions associated with different versions of ASP.NET include:

Extension	Introduced in version	Description
asax	1.0	This is global application file.You can use this file to define global variable(Variable that can be accessed from any Web page in the Web application.) It is mostly used to define the overall application event related to application & session object.Global.asax, used for application-level logic [17]
ascx	1.0	User Control, used for User Control files logic [18]
ashx	1.0	custom HTTP handlers Do not have a user interface.
asmx	1.0	Web service pages. From version 2.0 a Code behind page of an asmx file is placed into the app_code folder.
aspx	1.0	An ASP.NET Web Forms page that can contain Web controls and presentation and business logic. http://msdn.microsoft.com/en-us/library/2w aw kw 1c.aspx
axd	1.0	w hen enabled in w eb.config requesting trace.axd outputs application-level tracing. Also used for the special w ebresource.axd handler w hich allow s control/component developers to package a component/control complete w ith images, script, css etc. for deployment in one file (an 'assembly')
brow ser	2.0	brow ser capabilities files stored in XML format; introduced in version 2.0. ASP.NET 2 includes many of these by default, to support common Web brow sers. These specify w hich brow sers have w hich abilities, so that ASP.NET 2 can automatically customize and optimize its output accordingly. Special .brow ser files are available for free dow nload to handle, for instance, the W3C Validator, so that it properly show s standards-compliant pages as being standards-compliant. Replaces the harder-to-use Brow serCaps section that w as in machine.config and could be overridden in w eb.config in ASP.NET 1.x.
config	1.0	w eb.config is the only file in a specific Web application to use this extension by default (machine.config similarly affects the entire Web server and all applications on it), how ever ASP.NET provides facilities to create and consume other config files. These are stored in XML format.
cs/vb	1.0	Code files (cs indicates C#, vb indicates Visual Basic, fs indicates F#). Code behind files (see above) predominantly have the extension ".aspx.cs" or ".aspx.vb" for the tw o most common languages. Other code files (often containing common "library" classes) can also exist in the Web folders w ith the cs/vb extension. In ASP.NET 2 these should be placed inside the App_Code folder w here they are dynamically compiled and available to the w hole application.
cshtml	4.1	View s (mixed C# and HTML using Razor syntax)
dbml	3.5	LINQ to SQL data classes file
edmx	3.5	ADO.NET Entity Framew ork model
master	2.0	master page file. Default file name is Master1.master

resx	1.0	resource files for internationalization and localization. Resource files can be global (e.g. messages) or "local" w hich means specific for one aspx or ascx file.
sitemap	2.0	sitemap configuration files. Default file name is w eb.sitemap
skin	2.0	theme skin files.
svc	3.0	Window s Communication Foundation service file
vbhtml	4.1	View s (mixed VB and HTML using Razor syntax)

Directory structure

[edit]

In general, the ASP.NET directory structure can be determined by the developer's preferences. Apart from a few reserved directory names, the site can span any number of directories. The structure is typically reflected directly in the URLs. Although ASP.NET provides means for intercepting the request at any point during processing, the developer is not forced to funnel requests through a central application or front controller.

The special directory names (from ASP.NET 2.0 on) are:[19]

App_Code

This is the "raw code" directory. The ASP.NET server automatically compiles files (and subdirectories) in this folder into an assembly which is accessible in the code of every page of the site. App_Code will typically be used for data access abstraction code, model code and business code. Also any site-specific http handlers and modules and Web service implementation go in this directory. As an alternative to using App_Code the developer may opt to provide a separate assembly with precompiled code.

App_Data

The App_Data ASP.NET Directory is the default directory for any database used by the ASP.NET Website. These databases might include Access (mdb) files or SQL Server (mdf) files. The App_Data is the only directory with Write Access enabled for the ASP.NET web application.:[20]

App_LocalResources

E.g. a file called CheckOut.aspx.fr-FR.resx holds localized resources for the French version of the CheckOut.aspx page. When the UI culture is set to French, ASP.NET will automatically find and use this file for localization.

App_GlobalResources

Holds resx files with localized resources available to every page of the site. This is where the ASP.NET developer will typically store localized messages etc. which are used on more than one page.

App_Themes

Adds a folder that holds files related to themes which is a new ASP.NET feature that helps ensure a consistent appearance throughout a Web site and makes it easier to change the Web site’s appearance when necessary.

App_WebReferences

holds discovery files and WSDL files for references to Web services to be consumed in the site.

Bin

Contains compiled code (.dll files) for controls, components, or other code that you want to reference in your application. Any classes represented by code in the Bin folder are automatically referenced in your application.

Performance [edit]

ASP.NET aims for performance benefits over other script-based technologies (including Classic ASP) by compiling the server-side code to one or more DLL files on the Web server.[21] This compilation happens automatically the first time a page is requested (which means the developer need not perform a separate compilation step for pages). This feature provides the ease of development offered by scripting languages with the performance benefits of a compiled binary. However, the compilation might cause a noticeable but short delay to the Web user when the newly-edited page is first requested from the Web server, but will not again unless the page requested is updated further.

The ASPX and other resource files are placed in a virtual host on an Internet Information Services server (or other compatible ASP.NET servers; see Other implementations, below). The first time a client requests a page, the .NET Framework parses and compiles the file(s) into a .NET assembly and sends the response; subsequent requests are served from the DLL files. By default ASP.NET will compile the entire site in batches of 1000 files upon first request. If the compilation delay is causing problems, the batch size or the compilation strategy may be tweaked.

Developers can also choose to pre-compile their "codebehind" files before deployment, using MS Visual Studio, eliminating the need for just-in-time compilation in a production environment. This also eliminates the need of having the source code on the Web server. It also supports pre-compile text.

Extension [edit]

Microsoft has released some extension frameworks that plug into ASP.NET and extend its functionality. Some of them are:

ASP.NET AJAX

An extension with both client-side as well as server-side components for writing ASP.NET pages that incorporate AJAX functionality.

ASP.NET MVC Framework

An extension to author ASP.NET pages using the MVC architecture.

ASP.NET compared with ASP classic [edit]

ASP.NET simplifies developers' transition from Windows application development to Web development by offering the ability to build pages composed of *controls* similar to a Windows user interface. A Web control, such as a *button* or *label*, functions in very much the same way as its Windows counterparts: code can assign its properties and respond to its events. Controls know how to render themselves: whereas Windows controls draw themselves to the screen, Web controls produce segments of HTML and JavaScript which form parts of the resulting page sent to the end-user's browser.

ASP.NET encourages the programmer to develop applications using an event-driven GUI model, rather than in conventional Web-scripting environments like ASP and PHP. The framework combines existing technologies such as JavaScript with internal components like "ViewState" to bring persistent (inter-request) state to the inherently stateless Web environment.

Other differences compared to ASP classic are:

- Compiled code means applications run faster with more design-time errors trapped at the development stage.
- Significantly improved run-time error handling, making use of exception handling using try-catch blocks.
- Similar metaphors to Microsoft Windows applications such as controls and events.
- An extensive set of controls and class libraries allows the rapid building of applications, plus user-defined controls allow commonly-used Web template, such as menus. Layout of these controls on a page is easier because most of it can be done visually in most editors.
- ASP.NET uses the multi-language abilities of the .NET Common Language Runtime, allowing Web pages to be coded in VB.NET, C#, J#, Delphi.NET, Chrome, etc.
- Ability to cache the whole page or just parts of it to improve performance.
- Ability to use the code-behind development model to separate business logic from presentation.
- Ability to use true object-oriented design for programming pages and controls
- If an ASP.NET application leaks memory, the ASP.NET runtime unloads the AppDomain hosting the erring application and reloads the application in a new AppDomain.
- Session state in ASP.NET can be saved in a Microsoft SQL Server database or in a separate process running on the same machine as the Web server or on a different machine. That way session values are not lost when the Web server is reset or the ASP.NET worker process is recycled.
- Versions of ASP.NET prior to 2.0 were criticized for their lack of standards compliance. The generated HTML and JavaScript sent to the client browser would not always validate against W3C/ECMA standards. In addition, the framework's browser detection feature sometimes incorrectly identified Web browsers other than Microsoft's own Internet Explorer as "downlevel" and returned HTML/JavaScript to these clients with some of the features removed, or sometimes crippled or broken. In version 2.0 however, all controls generate valid HTML 4.0, XHTML 1.0 (the default) or XHTML 1.1 output, depending on the site configuration. Detection of standards-compliant Web browsers is more robust and support for Cascading Style Sheets is more extensive.
- Web Server Controls: these are controls introduced by ASP.NET for providing the UI for the Web form. These controls are

state managed controls and are WYSIWYG controls.

Criticism

[edit]

On IIS 6.0 and lower, pages written using different versions of the ASP framework cannot share Session State without the use of third-party libraries. This criticism does not apply to ASP.NET and ASP applications running side by side on IIS 7. With IIS 7, modules may be run in an integrated pipeline that allows modules written in any language to be executed for any request.[22]

Development tools

[edit]

Several available software packages exist for developing ASP.NET applications:

Software	Developer	Licensing
ASP.NET Intellisense Generator [2]	BlueVision LLC	Free
Microsoft Visual Studio	Microsoft	Free and commercial
Microsoft Visual Web Developer Express	Microsoft	Registerware
CodeGear Delphi	Embarcadero Technologies	Commercial
Macromedia HomeSite	Adobe Systems	Commercial
Microsoft Expression Web	Microsoft	Commercial
Microsoft SharePoint Designer	Microsoft	Free
MonoDevelop	Novell and the Mono community	Free open source
SharpDevelop	ICSharpCode Team	Free open source
Eiffel for ASP.NET [3]	Eiffel Software	Free open source and commercial
Adobe Dreamweaver	Adobe Systems	Commercial

Frameworks

[edit]

It is not essential to use the standard Web forms development model when developing with ASP.NET. Noteworthy frameworks designed for the platform include:

- Base One Foundation Component Library (BFC) is a RAD framework for building .NET database and distributed computing applications.
- DotNetNuke is an open-source solution which comprises both a Web application framework and a content management system which allows for advanced extensibility through modules, skins, and providers.

- Castle Monorail, an open-source MVC framework with an execution model similar to Ruby on Rails. The framework is commonly used with Castle ActiveRecord, an ORM layer built on NHibernate.
- Spring.NET, a port of the Spring framework for Java.
- Survey Project is an open-source Web based survey and form engine framework written in ASP.NET and C#.

Versions

[edit]

The ASP.NET releases history tightly correlates with the .NET Framework releases:

Date	Version	Remarks	New ASP.NET related features
January 16, 2002	1.0	First version released together with Visual Studio .NET	• Object-oriented Web application development supporting inheritance, polymorphism and other standard OOP features ◦ Developers are no longer forced to use Server.CreateObject(...), so early-binding and type safety are possible. • Based on Windows programming; the developer can make use of DLL class libraries and other features of the Web server to build more robust applications that do more than simply rendering HTML (e.g. exception handling)
April 24, 2003	1.1	released together with Windows Server 2003 released together with Visual Studio .NET 2003	• Mobile controls • Automatic input validation
November 7, 2005	2.0	codename Whidbey released together with Visual Studio 2005 and Visual Web Developer Express and SQL Server 2005	• New data controls (GridView, FormView, DetailsView) • New technique for declarative data access (SqlDataSource, ObjectDataSource, XmlDataSource controls) • Navigation controls • Master pages • Login controls • Themes • Skins • Web parts • Personalization services

			• Full pre-compilation • New localization technique • Support for 64-bit processors • Provider class model
November 21, 2006	3.0		• Windows Presentation Foundation (WPF) • Windows Workflow Foundation (WF) • Windows Communication Foundation which can use ASP.NET to host services. • Windows CardSpace which uses ASP.NET for login roles.
November 19, 2007	3.5	Released with Visual Studio 2008 and Windows Server 2008	• New data controls (ListView, DataPager) • ASP.NET AJAX included as part of the framework • Support for HTTP pipelining and syndication feeds. • WCF support for RSS, JSON, POX and Partial Trust • All the .NET Framework 3.5 changes, like LINQ etc.
August 11, 2008	3.5 Service Pack 1	Released with Visual Studio 2008 Service Pack 1	• Incorporation of ASP.NET Dynamic Data • Support for controlling browser history in an ASP.NET AJAX application • Ability to combine multiple JavaScript files into one file for more efficient downloading • New namespaces System.Web.Abstractions and System.Web.Routing
April 12, 2010	4.0	Parallel extensions and other .NET Framework 4 features	

Other implementations [edit]

The Mono Project supports "everything in .NET 4.0 except WPF, EntityFramework and WF, limited WCF."[23] ASP.NET can be run with Mono using one of three options: Apache hosting using the mod_mono module, FastCGI hosting, and XSP.

Notes [edit]

1. ^ a b "Architecture Journal Profile: Scott Guthrie". *The Architecture Journal*. Microsoft. January 2007. Archived from the original on 10 April 2008. Retrieved 2008-04-20.

2. ^ Michiel van Otegem (July 24, 2007). "Interview with Scott Guthrie, creator of ASP.NET". Retrieved 2008-04-20.
3. ^ Tim Anderson (October 30, 2007). "How ASP.NET began in Java". The Register. Archived from the original on 19 April 2008. Retrieved 2008-04-20.
4. ^ a b "Show #9 - ASP.NET". *The MSDN Show*. Microsoft. December 20, 2000. Archived from the original on 2001-04-13. Retrieved 2008-04-20.
5. ^ "Bill Gates speech transcript - Professional Developers Conference 2000". Microsoft. July 11, 2000. Archived from the original on 10 April 2008. Retrieved 2008-04-20.
6. ^ "ActiveState Supports Microsoft .NET Framework; Perl .NET & Python .NET Cross-Language Interoperability". Business Wire. July 11, 2000. Retrieved 2008-04-20.
7. ^ "Show #19 - LIVE! from the PDC". *The MSDN Show*. Microsoft. November 15, 2001. Retrieved 2008-04-20.
8. ^ Staff (2001-11). "Overview of ASP.NET and Web Forms". Microsoft. Retrieved 2011-06-05.
9. ^ (MacDonald & Szpuszta 2005, p. 63)
10. ^ "Code Behind vs. Code Inline". *Microsoft .NET Framework*. Microsoft. Archived from the original on 11 November 2010. Retrieved 2010-11-22.
11. ^ "ASP.NET Web Page Syntax Overview". *Microsoft .NET Framework*. Microsoft. Retrieved 2010-11-22.
12. ^ a b c "INFO: ASP.NET State Management Overview". Retrieved 2007-10-23.
13. ^ "ViewState in ASP.NET". Archived from the original on 14 October 2007. Retrieved 2007-10-23.
14. ^ "ASP.NET ViewState Overview".
15. ^ "Encrypting Viewstate in ASP.NET". Retrieved 2009-07-19.
16. ^ ASP.NET Master Pages Overview (Microsoft Developer Network)
17. ^ Global.asax Syntax
18. ^ [1]
19. ^ ASP.NET Web Site Layout from MSDN
20. ^ ASP.NET Directory Structure
21. ^ (MacDonald & Szpuszta 2005, pp. 7–8)
22. ^ How to Take Advantage of the IIS 7.0 Integrated Pipeline
23. ^ "Compatibility - Mono". Archived from the original on 1 November 2010. Retrieved 2010-10-18.

References

[edit]

- MacDonald, Matthew; Szpuszta, Mario (2005). *Pro ASP.NET 2.0 in C# 2005* (1st edition ed.). Apress. ISBN 1-59059-496-7.

Further reading

[edit]

- Anne Boehm: *Murachs ASP.NET 3.5 Web Programming with VB 2008*, July 21, 2008, Mike Murach and Associates, ISBN 978-1-890774-47-9
- Stephen Walther: *ASP.NET 3.5 Unleashed*, December 28, 2007, Sams Publishing, ISBN 0-672-33011-3 ISBN 0-672-33011-3

- Stephen Walther: *Data Access in the ASP.NET 2.0 Framework (Video Training)*, September 26, 2007, Sams Publishing, ISBN 0-672-32952-2
- Israel B. Ocbina: *Mastering VB.NET and C#. 7th Edition. October 22, 2004. by Cyberocbina© Cafê. A .NET Developers Edition*.

External links

[edit]

- www.asp.net — Microsoft ASP.NET 4.0 official site
- ASP.NET at the Open Directory Project
- ASP.NET on MSDN
- Some of new features in ASP.NET 4 and vs 2010 IDE

Wikibooks has more on the topic of
ASP.NET