Contents

Articles

References

Article Licenses

ASP.NET

Developer(s)	Microsoft
Initial release	January 2002
Stable release	4.0.30319.1 (4.0) / 12 April 2010
Written in	.NET Languages
Operating system	Microsoft Windows
Type	Web application framework
License	Proprietary
Website	www.asp.net [1]

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.

History

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.[2] 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 Christmas holidays in 1997.[3]

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."[2] The initial prototype of XSP was done using Java,[4] 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 didn't 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.


Scott Guthrie (Microsoft Developer Division VP) in 2007

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.[5]

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,[6] 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.[7]

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 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."[5]

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,[8] and Microsoft promoted it heavily as part of their 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

Pages

ASP.NET web pages, known officially as "web forms", 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 the .asp.cs page. [10]

Code-behind model

Microsoft recommends dealing with dynamic program code by using the code-behind model, which places this code in a separate file or in a specially designated script tag. Code-behind files typically have names like *MyPage.aspx.cs* or *MyPage.aspx.vb* while the page file is *MyPage.aspx* (same filename as the page file (ASPX), but with the final extension denoting the page language). This practice is automatic in Microsoft Visual Studio and other IDEs. When using this style of programming, the developer writes code to respond to different events, like the page being loaded, or a control being clicked, rather than a procedural walk through the document.

ASP.NET's code-behind model marks a departure from Classic ASP in that it encourages developers to build applications with separation of presentation and content in mind. In theory, this would allow a web designer, for example, to focus on the design markup with less potential for disturbing the programming code that drives it. This is similar to the separation of the controller from the view in model-view-controller frameworks. it is imprtant roll in the web form.

Directives

A directive is special instructions on how ASP.NET should process the page.[11] The most common directive is <%@ Page %> which can specify many things, such as which programming language is used for the server-side code.

Examples

Note that this sample uses code "inline", as opposed to code-behind.

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
  protected void Page_Load(object sender, EventArgs e)
  {
    Label1.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="Label1" />
    </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

```
<%@ 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) acting as the code-behind while the *Inherits* property specifies the Class the Page derives from. 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");
    }
  }
}
```

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

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

Programmers can also build *custom controls* for ASP.NET applications. Unlike user controls, these controls don't 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 (which is not allowed with user controls). By using a Register directive, the control is loaded from the DLL.

Rendering technique

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)