# Contents

# 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) 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

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 State

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 variables. Application state variables are identified by name.[12]

### Session state

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 a single instance, it is still a single 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

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

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

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

## Template engine

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

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

| Extension | Required version | Description |
|---|---|---|
| asax | 1.0 | Global.asax, used for application-level logic [17] |
| ashx | 1.0 | custom HTTP handlers. |
| 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. |
| axd | 1.0 | when enabled in web.config requesting trace.axd outputs application-level tracing. Also used for the special webresource.axd handler which allows control/component developers to package a component/control complete with images, script, css etc. for deployment in a single file (an 'assembly') |
| browser | 2.0 | browser capabilities files stored in XML format; introduced in version 2.0. ASP.NET 2 includes many of these by default, to support common web browsers. These specify which browsers have which capabilities, so that ASP.NET 2 can automatically customize and optimize its output accordingly. Special .browser files are available for free download to handle, for instance, the W3C Validator, so that it properly shows standards-compliant pages as being standards-compliant. Replaces the harder-to-use BrowserCaps section that was in machine.config and could be overridden in web.config in ASP.NET 1.x. |
| config | 1.0 | web.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), however 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). Code behind files (see above) predominantly have the extension ".aspx.cs" or ".aspx.vb" for the two most common languages. Other code files (often containing common "library" classes) can also exist in the web folders with the cs/vb extension. In ASP.NET 2 these should be placed inside the App_Code folder where they are dynamically compiled and available to the whole application. |
| dbml | 3.5 | LINQ to SQL data classes file |
| 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" which means specific for a single aspx or ascx file. |
| sitemap | 2.0 | sitemap configuration files. Default file name is web.sitemap |
| skin | 2.0 | theme skin files. |
| svc | 3.0 | Windows Communication Foundation service file |
| edmx | 3.5 | ADO.NET Entity Framework model |

### Directory structure

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 [18] :

App_Browsers

holds site-specific browser definition files.

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

default directory for databases, such as Access mdb files and SQL Server mdf files. This directory is usually the only one with write access for the application.

App_LocalResources

Contains localized resource files for individual pages of the site. 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

holds alternative themes of the site.

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

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.[19] 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 won't 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.

## Extension

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

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 counterpart: 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 capabilities 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 both page 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. However, in version 2.0, 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

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.[20]

## Development tools

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

| Product | Developer | Licensing |
|---|---|---|
| ASP.NET Intellisense Generator [21] | BlueVision LLC | Free |
| Microsoft Visual Studio | Microsoft | Free and Commercial |
| 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 [22] | Eiffel Software | Free Open Source and Commercial |
| Adobe Dreamweaver | Adobe Systems | Commercial |

## Frameworks

It is not essential to use the standard webforms 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.
- Skaffold.NET, A simple framework for .NET applications, used in enterprise applications.

## Versions

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<br>released together with Visual Studio .NET | • Object oriented web application development supporting Inheritance, Polymorphism and other standard OOP features<br>  • Developers are no longer forced to use Server.CreateObject(...), so early-binding and type safety are possible.<br>• 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<br>released together with Visual Studio .NET 2003 | • Mobile controls<br>• Automatic input validation |
| November 7, 2005 | 2.0 | codename Whidbey<br>released together with Visual Studio 2005 and Visual Web Developer Express<br>and SQL Server 2005 | • New data controls (GridView, FormView, DetailsView)<br>• New technique for declarative data access (SqlDataSource, ObjectDataSource, XmlDataSource controls)<br>• Navigation controls<br>• Master pages<br>• Login controls<br>• Themes<br>• Skins<br>• Web parts<br>• Personalization services<br>• Full pre-compilation<br>• New localization technique<br>• Support for 64-bit processors<br>• Provider class model |
| November 21, 2006 | 3.0 | | • Windows Presentation Foundation (WPF)<br>• Windows Workflow Foundation (WF)<br>• Windows Communication Foundation which can use ASP.NET to host services.<br>• 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)<br>• ASP.NET AJAX included as part of the framework<br>• Support for HTTP pipelining and syndication feeds.<br>• WCF Support for RSS, JSON, POX and Partial Trust<br>• 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<br>• Support for controlling browser history in an ASP.NET AJAX application<br>• Capability to combine multiple Javascript files into a single file for more efficient downloading<br>• New namespaces System.Web.Abstractions and System.Web.Routing |
| April 12, 2010 | 4.0 | Release with Visual Studio 2010 | Parallel extensions and other .NET Framework 4 features |

## Other implementations

The Mono Project supports "[e]verything 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

[1] http://www.asp.net

[2] "Architecture Journal Profile: Scott Guthrie" (http://msdn2.microsoft.com/en-us/library/bb266332.aspx). *The Architecture Journal*. Microsoft. January 2007. . Retrieved 2008-04-20.

[3] Michiel van Otegem (July 24, 2007). "Interview with Scott Guthrie, creator of ASP.NET" (http://www.vanotegem.nl/PermaLink,guid,d9826145-408c-4fb9-8939-79d7e6a19218.aspx). . Retrieved 2008-04-20.

[4] Tim Anderson (October 30, 2007). "How ASP.NET began in Java" (http://www.regdeveloper.co.uk/2007/10/30/asp_net_java_project_cool/). The Register. . Retrieved 2008-04-20.

[5] "Show #9 - ASP.NET" (http://web.archive.org/web/20010413165314/msdn.microsoft.com/theshow/Episode009/). *The MSDN Show*. Microsoft. December 20, 2000. Archived from the original (http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/theshow/Episode009/default.asp) on 2001-04-13. . Retrieved 2008-04-20.

[6] "Bill Gates speech transcript - Professional Developers Conference 2000" (http://www.microsoft.com/presspass/exec/billg/speeches/2000/07-12pdc.aspx). Microsoft. July 11, 2000. . Retrieved 2008-04-20.

[7] "ActiveState Supports Microsoft .NET Framework; Perl .NET & Python .NET Cross-Language Interoperability" (http://findarticles.com/p/articles/mi_m0EIN/is_2000_July_11/ai_63287204). Business Wire. July 11, 2000. . Retrieved 2008-04-20.

[8] "Show #19 - LIVE! from the PDC" (http://msdn.microsoft.com/library/shared/deeptree/asp/rightframe.asp?dtcfg=/archive/deeptreeconfig.xml&url=/archive/en-us/theshow/Episode019/default.asp). *The MSDN Show*. Microsoft. November 15, 2001. . Retrieved 2008-04-20.

[9] (MacDonald & Szpuszta 2005, p. 63)

[10] "Code Behind vs. Code Inline" (http://quickstarts.asp.net/QuickStartv20/aspnet/doc/pages/codebehind.aspx). *Microsoft .NET Framework*. Microsoft. . Retrieved 2010-11-22.

[11] "ASP.NET Web Page Syntax Overview" (http://msdn.microsoft.com/en-us/library/k33801s3.aspx). *Microsoft .NET Framework*. Microsoft. . Retrieved 2010-11-22.

[12] "INFO: ASP.NET State Management Overview" (http://support.microsoft.com/kb/307598). . Retrieved 2007-10-23.

[13] "ViewState in ASP.NET" (http://www.extremeexperts.com/Net/Articles/ViewState.aspx). . Retrieved 2007-10-23.

[14] "ASP.NET ViewState Overview" (http://www.dotnetrobert.com/dotnet/Home/tabid/37/Default.aspx). .

[15] "Encrypting Viewstate in ASP.NET" (http://msdn.microsoft.com/en-us/library/aa479501.aspx). . Retrieved 2009-07-19.

[16] ASP.NET Master Pages Overview (Microsoft Developer Network) (http://msdn2.microsoft.com/en-us/library/wtxbf3hh.aspx)

[17] Global.asax Syntax (http://msdn2.microsoft.com/en-us/library/2027ewzw.aspx)

[18] ASP.NET Web Site Layout from MSDN (http://msdn2.microsoft.com/en-us/library/ex526337.aspx)

[19] (MacDonald & Szpuszta 2005, pp. 7–8)

[20] How to Take Advantage of the IIS 7.0 Integrated Pipeline (http://learn.iis.net/page.aspx/244/how-to-take-advantage-of-the-iis7-integrated-pipeline)

[21] http://www.bluevisionsoftware.com/WebSite/ProductsAndServicesInfo.aspx?ID=9

[22] http://www.eiffel.com/downloads/asp.net.html

[23] "Compatibility - Mono" (http://www.mono-project.com/Compatibility). . Retrieved 2010-10-18.

## References

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

## Further reading

- 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

## External links

- ASP.NET (http://www.dmoz.org/Computers/Programming/Internet/ASP/ASP.NET//) at the Open Directory Project
- Microsoft's Official ASP.NET 3.5 website (http://www.asp.net/)
- ASP.NET on MSDN (http://msdn.microsoft.com/asp.net/)
- Some of new features in ASP.NET 4 and vs 2010 IDE (http://www.codeproject.com/KB/aspnet/Whatis_New_ASP_Net_4.aspx)

# Web application framework

A **web application framework** is a software framework that is designed to support the development of dynamic websites, web applications and web services. The framework aims to alleviate the overhead associated with common activities performed in Web development. For example, many frameworks provide libraries for database access, templating frameworks and session management, and they often promote code reuse.[1]

## History

As the design of the World Wide Web was not inherently dynamic, early hypertext consisted of hand-coded HTML that was published on web servers. Any modifications to published pages needed to be performed by the pages' author. To provide a dynamic web page that reflected user inputs, the Common Gateway Interface (CGI) standard was introduced for interfacing external applications with web servers.[2] CGI could adversely affect server load, though, since each request had to start a separate process.

Programmers wanted tighter integration with the web server to enable high traffic web applications. The Apache HTTP Server, for example, supports modules that can extend the web server with arbitrary code executions (such as mod perl) or forward specific requests to a web server that can handle dynamic content (such as mod jk). Some web servers (such as Apache Tomcat) were specifically designed to handle dynamic content by executing code written in some languages, such as Java.

Around the same time, new languages were being developed specifically for use in the web, such as ColdFusion, PHP and Active Server Pages.

While the vast majority of languages available to programmers to use in creating dynamic web pages have libraries to help with common tasks, web applications often require specific libraries that are useful in web applications, such as creating HTML (for example, JavaServer Faces).

Eventually, mature, "full stack" frameworks appeared, that often gathered multiple libraries useful for web development into a single cohesive software stack for web developers to use. Examples of this include JavaEE (Servlets), WebObjects, OpenACS, Catalyst, Ruby on Rails, Django, and Zend Framework.

## Architectures

Most web application frameworks are based on the MVC pattern. From an architecture perspective, there are generally five major types: request-based, component-based, hybrid, meta, and RIA-based.[3]

### Model view controller (MVC)

Many frameworks follow the Model View Controller (MVC) architectural pattern to separate the data model with business rules from user interface. This is generally considered a good practice as it modularizes code, promotes code reuse, and allows multiple interfaces to be applied.

#### Push-based vs. Pull-based

Most MVC frameworks follow a push-based architecture. These frameworks use actions that do the required processing, and then "push" the data to the view layer to render the results.[4] Struts, Django, Ruby on Rails and Spring MVC are good examples of this architecture. An alternative to this is pull-based architecture, sometimes also called "component-based". These frameworks start with the view layer, which can then "pull" results from multiple controllers as needed. In this architecture, multiple controllers can be involved with a single view. Struts2, Lift, Tapestry, JBoss Seam, Wicket and Stripes are examples of pull-based architectures.

### Content Management Systems

Some projects that have historically been termed content management systems have begun to take on the roles of higher-layer web application frameworks. For instance, Drupal's structure provides a minimal *core* whose function is extended through *modules* that provide functions generally associated with web application frameworks. However, it is debatable whether "management of content" is the primary value of such systems, especially when some, like SilverStripe, provide an object-oriented MVC framework. Add-on *modules* now enable these systems to function as full fledged applications beyond the scope of content management. They may provide functional APIs, functional frameworks, coding standards, and many of the functions traditionally associated with *Web application frameworks*.

## Features

### Web template system

Dynamic web pages usually consist of a static part (HTML) and a dynamic part, which is code that generates HTML. The code that generates the HTML can do this based on variables in a template, or on code. The text to be generated can come from a database, thereby making it possible to dramatically reduce the number of pages in a site.

Consider the example of a real estate agent with 500 houses for sale. In a static web site, the agent would have to create 500 pages in order to make the information available. In a dynamic website, the agent would simply connect the dynamic page to a database table of 500 records.

In a template, variables from the programming language can be inserted without using code, thereby losing the requirement of programming knowledge to make updates to the pages in a web site. A syntax is made available to distinguish between HTML and variables. E.g. in JSP the <c:out> tag is used to output variables, and in Smarty, {$variable} is used.

Many template engines do support limited logic tags, like IF and FOREACH. These are to be used only for decisions that need to be made for the presentation layer, in order to keep a clean separation from the business logic layer, or the M(odel) in the MVC pattern.

## Caching

Web caching is the caching of web documents in order to reduce bandwidth usage, server load, and perceived "lag". A web cache stores copies of documents passing through it; subsequent requests may be satisfied from the cache if certain conditions are met. Some application frameworks provide mechanisms for caching documents and bypassing various stages of the page's preparation, such as database access or template interpretation.

## Security

Some web application frameworks come with authentication and authorization frameworks, that enable the web server to identify the users of the application, and restrict access to functions based on some defined criteria. Drupal is one example that provides role-based access to pages, and provides a web-based interface for creating users and assigning them roles.

## Database access and mapping

Many web application frameworks create a unified API to a database backend, enabling web applications to work with a variety of databases with no code changes, and allowing programmers to work with higher-level concepts. For higher performance, database connections should be pooled as e.g. AOLserver does. Additionally, some object-oriented frameworks contain mapping tools to provide Object-Relational Mapping, which will map objects to tuples.

Other features web application frameworks may provide include transactional support and database migration tools.

## URL mapping

A framework's URL mapping facility is the mechanism by which the framework interprets URLs. Some frameworks, such as Drupal and Django, match the provided URL against pre-determined patterns using regular expressions, while some others use URL Rewriting to translate the provided URL into one that the underlying engine will recognize. Another technique is that of graph traversal such as used by Zope, where a URL is decomposed in steps that traverse an object graph (of models and views).

A URL mapping system that uses pattern matching or URL rewriting allows more "friendly" URLs to be used, increasing the simplicity of the site and allowing for better indexing by search engines. For example, a URL that ends with "/page.cgi?cat=science&topic=physics" could be changed to simply "/page/science/physics". This makes the URL easier to read and provides search engines with better information about the structural layout of the site. A graph traversal approach also tends to result in the creation of friendly URLs. A shorter URL such as "/page/science" tends to exist by default as that is simply a shorter form of the longer traversal to "/page/science/physics".

## Ajax

Ajax, shorthand for "*Asynchronous JavaScript and XML*", is a web development technique for creating interactive web applications. The intent is to make web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire web page does not have to be reloaded each time the user requests a change. This is intended to increase the web page's interactivity, speed, and usability.

Due to the complexity of Ajax programming in Javascript, there are numerous Ajax frameworks that exclusively deal with Ajax support. Some Ajax frameworks are even embedded as a part of larger frameworks. For example, the Prototype JavaScript Framework is included in Ruby on Rails.

With the increased interest in developing "Web 2.0" Rich Media Applications, the complexity of programming directly in Ajax and Javascript has become so apparent that compiler technology has stepped in, to allow developers to code in high-level languages such as Java, Python and Ruby. The first of these compilers was Morfik followed by Google Web Toolkit, with ports to Python and Ruby in the form of Pyjamas and RubyJS following some time after. These compilers and their associated widget set libraries make the development of Rich Media Ajax Applications much more akin to that of developing Desktop applications.

### Automatic configuration

Some frameworks minimize web application configuration through the use of introspection and/or following known conventions. For example, many Java frameworks use Hibernate as a persistence layer, which can generate a database schema at runtime capable of persisting the necessary information. This allows the application designer to design business objects without needing to explicitly define a database schema. Frameworks such as Ruby on Rails can also work in reverse, that is, define properties of model objects at runtime based on a database schema.

### Web services

Some frameworks provide tools for creating and providing web services. These utilities may offer similar tools as the rest of the web application.

## References

[1] Multiple (wiki). "Web application framework" (http://docforge.com/wiki/Web_application_framework). *Docforge*. . Retrieved 2010-01-19.

[2] "CGI: Common Gateway Interface" (http://hoohoo.ncsa.uiuc.edu/cgi/intro.html). . Retrieved 2007-07-29.

[3] Shan, Tony (2006-10-24). "Taxonomy of Java Web Application Frameworks" (http://portal.acm.org/citation.cfm?id=1190953). Proceedings of 2006 IEEE International Conference on e-Business Engineering (ICEBE 2006). . Retrieved 2010-10-10.

[4] Thomson, Kris (2003-10-29). "Clarification on MVC Pull and MVC Push" (http://www.theserverside.com/patterns/thread.tss?thread_id=22143). . Retrieved 2007-07-29.

## External links

- ALPHA Framework / ALPHA CMS (http://www.localhost-ltd.com/)
- web2py (http://www.web2py.com/)
- Drupal (http://www.drupal.org/)
- CMS matrix (http://www.cmsmatrix.org/)
- NanoMVC (http://www.nanomvc.com/)
- jspx
- JDHTML

# .NET Framework

| | |
|---|---|
| **Developer(s)** | Microsoft |
| **Initial release** | 13 February 2002 |
| **Stable release** | 4.0 (4.0.30319.1) / 12 April 2010 |
| **Operating system** | Windows 98 or later, Windows NT 4.0 or later |
| **Type** | Software framework |
| **License** | MS-EULA, BCL under Microsoft Reference Source License[1] |
| **Website** | msdn.microsoft.com/netframework [2] |

The **Microsoft .NET Framework** is a software framework for Microsoft Windows operating systems. It includes a large library, and it supports several programming languages which allows language interoperability (each language can utilize code written in other languages.) The .NET library is available to all the programming languages that .NET supports.

The framework's Base Class Library provides user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. The class library is used by programmers, who combine it with their own code to produce applications.

Programs written for the .NET Framework execute in a software (as contrasted to hardware) environment, known as the Common Language Runtime (CLR). The CLR is an application virtual machine so that programmers need not consider the capabilities of the specific CPU that will execute the program. The CLR also provides other important services such as security, memory management, and exception handling. The class library and the CLR together constitute the .NET Framework.

The .NET Framework is intended to be used by most new applications created for the Windows platform. In order to be able to develop and not just run applications, it is required to have Microsoft's SDK for Windows 7 or .NET Framework 4 (or newer) or Visual Studio 2010 installed on your computer.

## Principal design features

Interoperability

Because computer systems commonly require interaction between new and older applications, the .NET Framework provides means to access functionality that is implemented in programs that execute outside the .NET environment. Access to COM components is provided in the System.Runtime.InteropServices and System.EnterpriseServices namespaces of the framework; access to other functionality is provided using the P/Invoke feature.

Common Runtime Engine

The Common Language Runtime (CLR) is the execution engine of the .NET Framework. All .NET programs execute under the supervision of the CLR, guaranteeing certain properties and behaviors in the areas of memory management, security, and exception handling.

Language Independence

The .NET Framework introduces a Common Type System, or CTS. The CTS specification defines all possible datatypes and programming constructs supported by the CLR and how they may or may not interact with each other conforming to the Common Language Infrastructure (CLI) specification. Because of this feature, the .NET Framework supports the exchange of types and object instances between libraries and applications written using any conforming .NET language.

Base Class Library

The Base Class Library (BCL), part of the Framework Class Library (FCL), is a library of functionality available to all languages using the .NET Framework. The BCL provides classes which encapsulate a number of common functions, including file reading and writing, graphic rendering, database interaction, XML document manipulation and so on.

Simplified Deployment

The .NET Framework includes design features and tools that help manage the installation of computer software to ensure that it does not interfere with previously installed software, and that it conforms to security requirements.

Security

The design is meant to address some of the vulnerabilities, such as buffer overflows, that have been exploited by malicious software. Additionally, .NET provides a common security model for all applications.

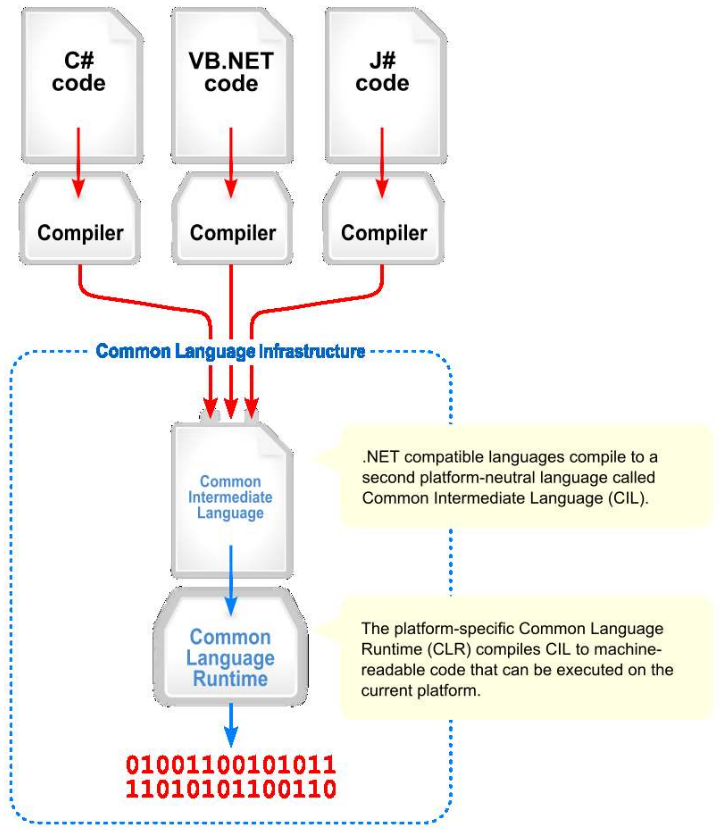
Portability

The design of the .NET Framework allows it to theoretically be platform agnostic, and thus cross-platform compatible. That is, a program written to use the framework should run without change on any type of system for which the framework is implemented. While Microsoft has never implemented the full framework on any system except Microsoft Windows, the framework is engineered to be platform agnostic,[3] and cross-platform implementations are available for other operating systems (see Silverlight and the Alternative implementations section below). Microsoft submitted the specifications for the Common Language Infrastructure (which includes the core class libraries, Common Type System, and the Common Intermediate Language),[4] [5] [6] the C# language,[7] and the C++/CLI language[8] to both ECMA and the ISO, making them available as open standards. This makes it possible for third parties to create compatible implementations of the framework and its languages on other platforms.

## Architecture

### Common Language Infrastructure (CLI)

The purpose of the Common Language Infrastructure (CLI), is to provide a language-neutral platform for application development and execution, including functions for exception handling, garbage collection, security, and interoperability. By implementing the core aspects of the .NET Framework within the scope of the CLI, this functionality will not be tied to a single language but will be available across the many languages supported by the framework. Microsoft's implementation of the CLI is called the Common Language Runtime, or CLR.


C# code
VB.NET code
J# code
Compiler
Compiler
Compiler
Common Language Infrastructure
Common Intermediate Language
.NET compatible languages compile to a second platform-neutral language called Common Intermediate Language (CIL).
Common Language Runtime
The platform-specific Common Language Runtime (CLR) compiles CIL to machine-readable code that can be executed on the current platform.
01001100101011
11010101100110
Visual overview of the Common Language Infrastructure (CLI)

### Assemblies

The CIL code is housed in .NET assemblies. As mandated by specification, assemblies are stored in the Portable Executable (PE) format, common on the Windows platform for all DLL and EXE files. The assembly consists of one or more files, one of which must contain the manifest, which has the metadata for the assembly. The complete name of an assembly (not to be confused with the filename on disk) contains its simple text name, version number, culture, and public key token. The public key token is a unique hash generated when the assembly is compiled, thus two assemblies with the same public key token are guaranteed to be identical from the point of view of the framework. A private key can also be specified known only to the creator of the assembly and can be used for strong naming and to guarantee that the assembly is from the same author when a new version of the assembly is compiled (required to add an assembly to the Global Assembly Cache).

### Metadata

All CIL is self-describing through .NET metadata. The CLR checks the metadata to ensure that the correct method is called. Metadata is usually generated by language compilers but developers can create their own metadata through custom attributes. Metadata contains information about the assembly, and is also used to implement the reflective programming capabilities of .NET Framework.

### Security

.NET has its own security mechanism with two general features: Code Access Security (CAS), and validation and verification. Code Access Security is based on evidence that is associated with a specific assembly. Typically the evidence is the source of the assembly (whether it is installed on the local machine or has been downloaded from the intranet or Internet). Code Access Security uses evidence to determine the permissions granted to the code. Other code can demand that calling code is granted a specified permission. The demand causes the CLR to perform a call stack walk: every assembly of each method in the call stack is checked for the required permission; if any assembly is not granted the permission a security exception is thrown.

When an assembly is loaded the CLR performs various tests. Two such tests are validation and verification. During validation the CLR checks that the assembly contains valid metadata and CIL, and whether the internal tables are correct. Verification is not so exact. The verification mechanism checks to see if the code does anything that is 'unsafe'. The algorithm used is quite conservative; hence occasionally code that is 'safe' does not pass. Unsafe code will only be executed if the assembly has the 'skip verification' permission, which generally means code that is installed on the local machine.

.NET Framework uses Application Domains as a mechanism for isolating code running in a process. Application Domains can be created and code can be loaded into or unloaded from them independent of other Application Domains. This helps increase the fault tolerance of the application, as faults or crashes in one Application Domain do not affect the rest of the application. Application Domains can also be configured independently with different security privileges. This can help increase the security of the application by isolating potentially unsafe code. The developer, however, has to split the application into subdomains; it is not done by the CLR.

## Class library

| **Namespaces in the BCL**[9] |
|---|
| System |
| System. CodeDom |
| System. Collections |
| System. Diagnostics |
| System. Globalization |
| System. IO |
| System. Resources |
| System. Text |
| System. Text.RegularExpressions |

The .NET Framework includes a set of standard class libraries. The class library is organized in a hierarchy of namespaces. Most of the built in APIs are part of either System.* or Microsoft.* namespaces. These class libraries implement a large number of common functions, such as file reading and writing, graphic rendering, database interaction, and XML document manipulation, among others. The .NET class libraries are available to all CLI compliant languages. The .NET Framework class library is divided into two parts: the Base Class Library and the Framework Class Library.

The Base Class Library (BCL) includes a small subset of the entire class library and is the core set of classes that serve as the basic API of the Common Language Runtime.[9] The classes in mscorlib.dll and some of the classes in System.dll and System.core.dll are considered to be a part of the BCL. The BCL classes are available in both .NET Framework as well as its alternative implementations including .NET Compact Framework, Microsoft Silverlight and Mono.

The Framework Class Library (FCL) is a superset of the BCL classes and refers to the entire class library that ships with .NET Framework. It includes an expanded set of libraries, including Windows Forms, ADO.NET, ASP.NET, Language Integrated Query, Windows Presentation Foundation, Windows Communication Foundation among others. The FCL is much larger in scope than standard libraries for languages like C++, and comparable in scope to the standard libraries of Java.

### Memory management

The .NET Framework CLR frees the developer from the burden of managing memory (allocating and freeing up when done); instead it does the memory management itself. To this end, the memory allocated to instantiations of .NET types (objects) is done contiguously[10] from the managed heap, a pool of memory managed by the CLR. As long as there exists a reference to an object, which might be either a direct reference to an object or via a graph of objects, the object is considered to be in use by the CLR. When there is no reference to an object, and it cannot be reached or used, it becomes garbage. However, it still holds on to the memory allocated to it. .NET Framework includes a garbage collector which runs periodically, on a separate thread from the application's thread, that enumerates all the unusable objects and reclaims the memory allocated to them.

The .NET Garbage Collector (GC) is a non-deterministic, compacting, mark-and-sweep garbage collector. The GC runs only when a certain amount of memory has been used or there is enough pressure for memory on the system. Since it is not guaranteed when the conditions to reclaim memory are reached, the GC runs are non-deterministic. Each .NET application has a set of roots, which are pointers to objects on the managed heap (*managed objects*). These include references to static objects and objects defined as local variables or method parameters currently in scope, as well as objects referred to by CPU registers.[10] When the GC runs, it pauses the application, and for each object referred to in the root, it recursively enumerates all the objects reachable from the root objects and marks them as reachable. It uses .NET metadata and reflection to discover the objects encapsulated by an object, and then recursively walk them. It then enumerates all the objects on the heap (which were initially allocated contiguously) using reflection. All objects not marked as reachable are garbage.[10] This is the *mark* phase.[11] Since the memory held by garbage is not of any consequence, it is considered free space. However, this leaves chunks of free space between objects which were initially contiguous. The objects are then *compacted* together to make used memory contiguous again.[10] [11] Any reference to an object invalidated by moving the object is updated to reflect the new location by the GC.[11] The application is resumed after the garbage collection is over.

The GC used by .NET Framework is actually *generational*.[12] Objects are assigned a *generation*; newly created objects belong to *Generation 0*. The objects that survive a garbage collection are tagged as *Generation 1*, and the Generation 1 objects that survive another collection are *Generation 2* objects. The .NET Framework uses up to Generation 2 objects.[12] Higher generation objects are garbage collected less frequently than lower generation objects. This helps increase the efficiency of garbage collection, as older objects tend to have a larger lifetime than newer objects.[12] Thus, by removing older (and thus more likely to survive a collection) objects from the scope of a collection run, fewer objects need to be checked and compacted.[12]

## Standardization and licensing

In August 2000, Microsoft, Hewlett-Packard, and Intel worked to standardize CLI and the C# programming language. By December 2001, both were ratified ECMA standards (ECMA 335 [13] and ECMA 334 [14]). ISO followed in April 2003 - the current version of the ISO standards are ISO/IEC 23271:2006 and ISO/IEC 23270:2006.[15] [16]

While Microsoft and their partners hold patents for the CLI and C#, ECMA and ISO require that all patents essential to implementation be made available under "reasonable and non-discriminatory terms". In addition to meeting these terms, the companies have agreed to make the patents available royalty-free.

However, this does not apply for the part of the .NET Framework which is not covered by the ECMA/ISO standard, which includes Windows Forms, ADO.NET, and ASP.NET. Patents that Microsoft holds in these areas may deter non-Microsoft implementations of the full framework.[17]

On 3 October 2007, Microsoft announced that much of the source code for the .NET Framework Base Class Library (including ASP.NET, ADO.NET, and Windows Presentation Foundation) was to have been made available with the final release of Visual Studio 2008 towards the end of 2007 under the shared source Microsoft Reference License.[1]

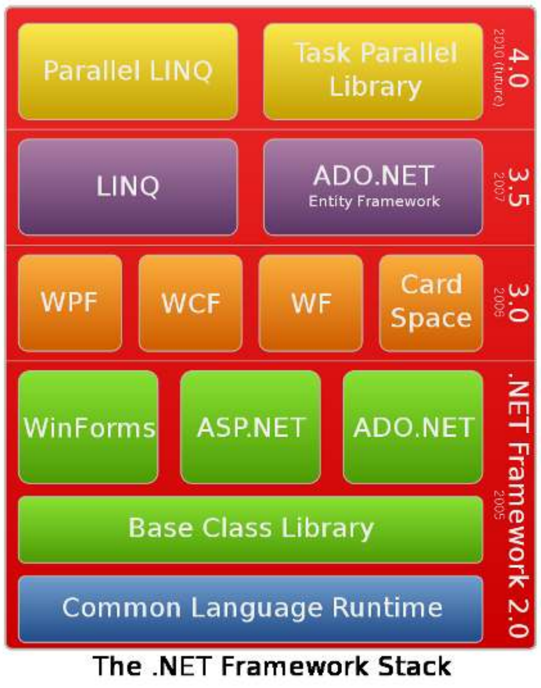
The source code for other libraries including Windows Communication Foundation (WCF), Windows Workflow Foundation (WF), and Language Integrated Query (LINQ) were to be added in future releases. Being released under the non Open-source Microsoft Reference License means this source code is made available for debugging purpose only, primarily to support integrated debugging of the BCL in Visual Studio.

## Versions

Microsoft started development on the .NET Framework in the late 1990s originally under the name of Next Generation Windows Services (NGWS). By late 2000 the first beta versions of .NET 1.0 were released.[18]

Version 3.0 of the .NET Framework is included with Windows Server 2008 and Windows Vista. Version 3.5 is included with Windows 7, and can also be installed on Windows XP and the Windows Server 2003 family of operating systems.[19] On April 12, 2010, .NET Framework 4 was released alongside Visual Studio 2010.

The .NET Framework family also includes two versions for mobile or embedded device use. A reduced version of the framework, the .NET Compact Framework, is available on Windows CE platforms, including Windows Mobile devices such as smartphones. Additionally, the .NET Micro Framework is targeted at severely resource-constrained devices.



The .NET Framework stack.

| Version | Version Number | Release Date | Visual Studio | Default in Windows |
|---|---|---|---|---|
| 1.0 | 1.0.3705.0 | 2002-02-13 | Visual Studio .NET | |
| 1.1 | 1.1.4322.573 | 2003-04-24 | Visual Studio .NET 2003 | Windows Server 2003 |
| 2.0 | 2.0.50727.42 | 2005-11-07 | Visual Studio 2005 | Windows Server 2003 R2 |
| 3.0 | 3.0.4506.30 | 2006-11-06 | | Windows Vista, Windows Server 2008 |
| 3.5 | 3.5.21022.8 | 2007-11-19 | Visual Studio 2008 | Windows 7, Windows Server 2008 R2 |
| 4.0 | 4.0.30319.1 | 2010-04-12 | Visual Studio 2010 | |

A more complete listing of the releases of the .NET Framework may be found on the List of .NET Framework versions.

## Criticism

Some concerns and criticism relating to .NET include:

- Applications running in a managed environment tend to require more system resources than similar applications that access machine resources more directly.
- Unobfuscated managed CIL bytecode can often be easier to reverse-engineer than native code.[20] [21] One concern is over possible loss of trade secrets and the bypassing of license control mechanisms. Since Visual Studio .NET (2002), Microsoft has included a tool to obfuscate code (Dotfuscator Community Edition).[22]
- Newer versions of the framework (3.5 and up) are not pre-installed in versions of Windows below Windows 7 (They are available via Windows Updates though). For this reason, applications must lead users without the framework through a procedure to install it. Some developers have expressed concerns about the large size of the .NET Framework runtime installers for end-users. The size is around 54 MB for .NET 3.0, 197 MB for .NET 3.5, and 250 MB for .NET 3.5 SP1 (while using web installer the typical download for Windows XP is around 50 MB, for Windows Vista - 20 MB). The size issue is partially solved with .NET 4 installer (x86 + x64) being 54 MB and not embedding full runtime installation packages for previous versions. The .NET 3.5 SP1 full installation package includes the full runtime installation packages for .NET 2.0 SP2 as well as .NET 3.0 SP2 for multiple operating systems (Windows XP/Server 2003 and Windows Vista/Server 2008) and for multiple CPU architectures (x86, x86-64, and IA-64).
  - The first service pack for version 3.5 mitigates this concern by offering a lighter-weight client-only subset of the full .NET Framework. Two significant limitations should be noted, though.[23] Firstly, the client-only subset is only an option on an existing Windows XP SP2 system that currently has no other version of the .NET Framework installed. In all other scenarios, the client-only installer will install the full version of the .NET Framework 3.5 SP1. Secondly, the client-only framework does not have a 64-bit option. However, the 4 release of the .NET Framework Client Profile will be available on all operating systems and all architectures (excluding Itanium) supported by the full .NET Framework.[24]
- The .NET Framework currently does not provide support for calling Streaming SIMD Extensions (SSE) via managed code. However, Mono has provided support for SIMD Extensions as of version 2.2 within the Mono.Simd namespace; Mono's lead developer Miguel de Icaza has expressed hope that this SIMD support will be adopted by the CLR ECMA standard.[25] Streaming SIMD Extensions have been available in x86 CPUs since the introduction of the Pentium III. Some other architectures such as ARM and MIPS also have SIMD extensions. In case the CPU lacks support for those extensions, the instructions are simulated in software.
- While the standards that make up .NET are inherently cross platform, Microsoft's full implementation of .NET is only supported on Windows. Microsoft does provide limited .NET subsets for other platforms such as XNA for Windows, XBOX 360 and Windows Phone 7; and Silverlight for Windows, Mac OS X, and Windows Phone 7. Alternative implementations of the CLR, base class libraries, and compilers also exist (sometimes from other vendors). While all of these implementations are based on the same standards, they are still different implementations with varying levels of completeness in comparison to the full .NET version Microsoft ships for Windows and are on occasion incompatible.

## Alternative implementations

The Microsoft .NET Framework is the predominant implementation of .NET technologies. Other implementations for parts of the framework exist. Although the runtime engine is described by an ECMA/ISO specification, other implementations of it may be encumbered by patent issues; ISO standards may include the disclaimer, "Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights."[26] It is more difficult to develop alternatives to the base class library (BCL), which is not described by an open standard and may be subject to copyright restrictions. Additionally, parts of the BCL have Windows-specific functionality and behavior, so implementation on non-Windows platforms can be problematic.

Some alternative implementations of parts of the framework are listed here.

- Microsoft's Shared Source Common Language Infrastructure is a *shared source implementation* of the CLR component of the .NET Framework. However, the last version only runs on Microsoft Windows XP SP2, and does not contain all features of version 2.0 of the .NET Framework.
- Microsoft's .NET Micro Framework is a .NET platform for extremely resource-constrained devices. It includes a small version of the .NET CLR and supports development in C# and debugging (in an emulator or on hardware), both using Microsoft Visual Studio. It also features a subset of the .NET base class libraries (about 70 classes with about 420 methods), a GUI framework loosely based on Windows Presentation Foundation, and additional libraries specific to embedded applications.
- Mono is an up-to-date implementation of the CLI and the .NET Base Class Library (BCL), and provides additional functionality. It is dual-licensed under free software and proprietary software licenses. Mono is actively being developed by Novell, Inc. It includes support for ASP.NET, ADO.NET, and Windows Forms libraries for a wide range of architectures (including iPhone, Wii and others) and operating systems. It also includes C# and VB.NET compilers.
- CrossNet [27] is an implementation of the CLI and portions of the .NET Base Class Library (BCL). It is free software using the open source MIT License. It parses .NET assemblies and generates standard C++ code, compilable with any ANSI C++ compiler on any platform.
- Portable.NET (part of DotGNU) provides an implementation of the Common Language Infrastructure (CLI), portions of the .NET Base Class Library (BCL), and a C# compiler. It supports a variety of CPUs and operating systems.

## References

[1] Scott Guthrie (3 October 2007). "Releasing the Source Code for the NET Framework" (http://weblogs.asp.net/scottgu/archive/2007/10/03/releasing-the-source-code-for-the-net-framework-libraries.aspx). . Retrieved 15 September 2010.

[2] http://msdn.microsoft.com/netframework

[3] "Scott Guthrie: Silverlight and the Cross-Platform CLR" (http://channel9.msdn.com/shows/Going+Deep/Scott-Guthrie-Silverlight-and-the-Cross-Platform-CLR). Channel 9. 30 April 2007. .

[4] "ECMA 335 - Standard ECMA-335 Common Language Infrastructure (CLI)" (http://www.ecma-international.org/publications/standards/Ecma-335.htm). ECMA. 1 June 2006. . Retrieved 1 June 2008.

[5] ISO/IEC 23271:2006 (http://standards.iso.org/ittf/PubliclyAvailableStandards/c042927_ISO_IEC_23271_2006(E)_Software.zip)

[6] "Technical Report TR/84 Common Language Infrastructure (CLI) - Information Derived from Partition IV XML File" (http://www.ecma-international.org/publications/techreports/E-TR-084.htm). ECMA. 1 June 2006. .

[7] "ECMA-334 C# Language Specification" (http://www.ecma-international.org/publications/standards/Ecma-334.htm). ECMA. 1 June 2006. .

[8] "Standard ECMA-372 C++/CLI Language Specification" (http://www.ecma-international.org/publications/standards/Ecma-372.htm). ECMA. 1 December 2005. .

[9] "Base Class Library" (http://msdn.microsoft.com/netframework/aa569603.aspx). . Retrieved 1 June 2008.

[10] "Garbage Collection: Automatic Memory Management in the Microsoft .NET Framework" (http://web.archive.org/web/20070703083608/http://msdn.microsoft.com/msdnmag/issues/1100/GCI/). Archived from the original (http://msdn.microsoft.com/msdnmag/issues/1100/GCI/) on 3 July 2007. . Retrieved 1 June 2008.

[11] "Garbage collection in .NET" (http://www.csharphelp.com/archives2/archive297.html). . Retrieved 1 June 2008.

[12] "Garbage Collection—Part 2: Automatic Memory Management in the Microsoft .NET Framework" (http://web.archive.org/web/20070626080134/http://msdn.microsoft.com/msdnmag/issues/1200/GCI2/default.aspx). Archived from the original (http://msdn.microsoft.com/msdnmag/issues/1200/GCI2/default.aspx) on 26 June 2007. . Retrieved 1 June 2008.

[13] http://www.ecma-international.org/publications/standards/Ecma-335.htm

[14] http://www.ecma-international.org/publications/standards/Ecma-334.htm

[15] ISO/IEC 23271:2006 - Information technology - Common Language Infrastructure (CLI) Partitions I to VI (http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=42927)

[16] ISO/IEC 23270:2006 - Information technology - Programming languages - C# (http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=42926)

[17] "Microsoft's Empty Promise" (http://www.webcitation.org/5io4tT8mO). *Free Software Foundation*. 16 July 2009. Archived from the original (http://www.fsf.org/news/2009-07-mscp-mono) on 5 August 2009. . Retrieved 3 August 2009. "*However, there are several libraries that are included with Mono, and commonly used by applications like Tomboy, that are not required by the standard. And just to be clear, we're not talking about Windows-specific libraries like ASP.NET and Windows Forms. Instead, we're talking about libraries under the System namespace that provide common functionality programmers expect in modern programming languages*"
[18] "Framework Versions" (http://ben.skyiv.com/clrversion.html). .
[19] Microsoft. "Microsoft .NET Framework 3.5 Administrator Deployment Guide" (http://msdn.microsoft.com/library/cc160717.aspx). . Retrieved 26 June 2008.
[20] "Reverse Engineering Risk Assessment" (http://www.preemptive.com/images/documentation/Reverse_Engineering_Risk_Assessment.pdf). .
[21] Gartner, Inc. as reported in "Hype Cycle for Cyberthreats, 2006", September 2006, Neil MacDonald; Amrit Williams, et al.
[22] Dotfuscator Community Edition 4.0
[23] *.NET Framework Client Profile Deployment Scenarios* (http://download.microsoft.com/download/5/a/a/5aa86d6c-969b-42d8-bc6b-30e02bfeccf0/NETFXClientProfile_DeploymentGuide.htm#_Toc205028507)
[24] "'.NET Framework 4 Client Profile - Introduction'" (http://www.webcitation.org/5kHCQjtFS). Archived from the original (http://blogs.msdn.com/jgoldb/archive/2009/05/27/net-framework-4-client-profile-introduction.aspx) on 2009-10-04. . Retrieved 2009-10-02.
[25] *Mono's SIMD Support: Making Mono safe for Gaming* (http://tirania.org/blog/archive/2008/Nov-03.html)
[26] ISO 9001:2008, Foreword
[27] http://www.codeplex.com/crossnet

## External links

- .NET Framework Developer Center (http://msdn.microsoft.com/netframework/)

# Active Server Pages

| | |
|---|---|
| **Developer(s)** | Microsoft |
| **Stable release** | 3.0 (no further versions planned) |
| **Type** | Web application framework |
| **License** | Proprietary |

**Active Server Pages** (**ASP**), also known as *Classic ASP* or *ASP Classic*, was Microsoft's first server-side script-engine for dynamically-generated web pages. Initially released as an add-on to Internet Information Services (IIS) via the Windows NT 4.0 Option Pack (ca 1998), it was subsequently included as a free component of Windows Server (since the initial release of Windows 2000 Server). ASP.NET has superseded ASP.

ASP 2.0 provided six built-in objects: Application, ASPError, Request, Response, Server, and Session. Session, for example, represents a cookie-based session that maintains the state of variables from page to page. The Active Scripting engine's support of the Component Object Model (COM) enables ASP websites to access functionality in compiled libraries such as DLLs.

Web pages with the *.asp* file extension use ASP, although some web sites disguise their choice of scripting language for security purposes (e.g. still using the more common *.htm* or *.html* extension). Pages with the *.aspx* extension use compiled ASP.NET (based on Microsoft's .NET Framework), which makes them faster and more robust than server-side scripting in ASP, which is interpreted at run-time; however, ASP.NET pages may still include some ASP scripting. The introduction of ASP.NET led to use of the term *Classic ASP* for the original technology.

Programmers write most ASP pages using VBScript, but any other Active Scripting engine can be selected instead with the `@Language` directive or the <script language="language" runat="server"> syntax. JScript (Microsoft's implementation of ECMAScript) is the other language that is usually available. PerlScript (a derivative of Perl) and others are available as third-party installable Active Scripting engines.

## History

Based on the *dbWeb* and *iBasic* tools created by Aspect Software Engineering, ASP was one of the first web application development environments that integrated web application execution directly into the web server, 9 months after the 1996 release of NeXT's (now Apple) WebObjects. This was done in order to achieve high performance compared to calling external executable programs or CGI scripts which were the most popular method for writing web applications at that time.

Prior to Microsoft's release of ASP for IIS 3, web programmers working in IIS relied on IDC and HTX files combined with ODBC drivers to display and manipulate dynamic data and pages. The basics of these file-formats and structures continued in use, at least in part, in the implementation of the early versions of ASP.

The third-party products *Halcyon InstantASP* (iASP) and *Chili!Soft ASP* run ASP on platforms other than the Microsoft Windows operating systems. Neither alternative to real ASP fully emulates every feature, and may require additional components to achieve functionality with which traditional ASP has no issues, such as database connectivity. Support for Microsoft Access databases can become a particular issue on non-Windows systems.

iASP can use the VBScript and JScript languages - unlike Chili!Soft ASP which uses JScript. Microsoft's ASP can use both and has the potential to have other languages make use of the scripting engine. iASP was written in Java, and as such will run on almost any operating system. iASP appears to be no longer available or at least hard to find.

Examples of other languages available are Perl and TCL, although they are not as widely known or used for ASP scripting. An Apache Webserver mod runs an ASP-like Perl script language.[1]

Chili!Soft, initially released in 1997, was acquired by Cobalt Networks on May 24, 2000. Sun Microsystems subsequently purchased Cobalt Networks (December 7, 2000). Chili!Soft was renamed "Sun ONE Active Server Pages", then later renamed to "Sun Java System Active Server Pages". Chilisoft ASP was written in C/C++ and is tied rather tightly to specific web server versions. According to Sun, "Sun Java System Active Server Pages" has entered its End Of Life".[2]

## Versions

ASP has gone through three major releases:

- ASP version 1.0 (distributed with IIS 3.0) in December 1996
- ASP version 2.0 (distributed with IIS 4.0) in September 1997
- ASP version 3.0 (distributed with IIS 5.0) in November 2000

As of 2010 ASP 3.0 is available in IIS 6.0 on Windows Server 2003 and IIS 7.0 on Windows Server 2008.

Many people regard ASP.NET as the newest release of ASP, but the two products use very different technologies. ASP.NET relies on the .NET Framework and is a compiled language, whereas ASP is strictly an interpreted scripting language.

ASP 3.0 made relatively few changes to the ASP 2.0 codebase. One of the most important additions was the Server.Execute [3] methods, as well as the ASPError object [4].[5] Microsoft's What's New in IIS 5.0 [6] lists some additional changes.

Some systems run "Classic ASP" sites as standalone applications: for example ASPexplore, a software package that runs Microsoft Active Server Pages offline.

## Sample usage

In ASP, programmers can use any scripting language compatible with Microsoft's Active Scripting standard. The default scripting language (in classic ASP) is VBScript:

```
<html>
<body>
```

```
<% Response.Write "Hello World!" %>
```

```
</body>
</html>
```

Or in a simpler format both work similarly

```
<html>
<body>
```

```
<%= "Hello World!" %>
```

```
</body>
</html>
```

The examples above print "Hello World!" into the body of an HTML document.

To connect to an Access Database:

```
<%
      Set oConn = Server.CreateObject("ADODB.Connection")
      oConn.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" & Server.MapPath("DB.mdb")
      Set rsUsers = Server.CreateObject("ADODB.Recordset")
      rsUsers.Open "SELECT UserID FROM Users", oConn,1,3
%>
```

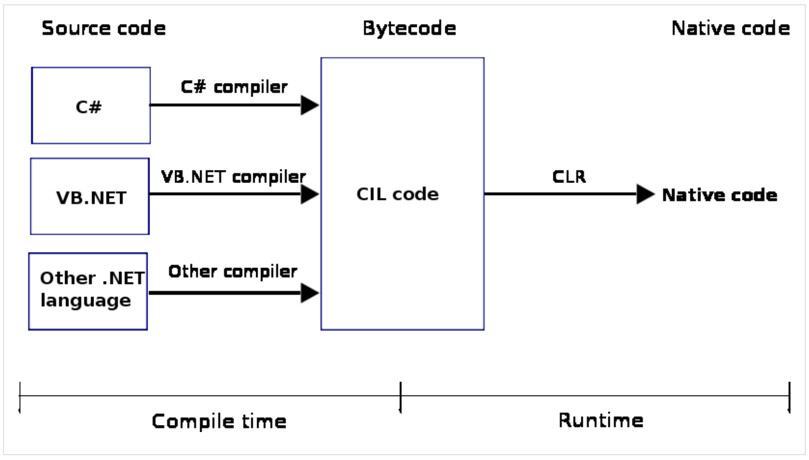
## References

[1] "Apache::ASP" (http://www.apache-asp.org/). Chamas Enterprises Inc. . Retrieved 2009-01-08.
[2] "Sun Java System Active Server Pages" (http://www.sun.com/software/chilisoft/). Sun Microsystems. . Retrieved 2008-12-31.
[3] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/iissdk/html/db562da1-d49d-4fe5-9747-64ef530de23f.asp
[4] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/iissdk/html/541697df-5fb9-40a4-8fa1-380b4717cbf1.asp
[5] 4 Guys From Rolla's A Look at ASP 3.0 (http://www.4guysfromrolla.com/webtech/010700-1.shtml)
[6] http://www.microsoft.com/technet/prodtechnol/windows2000serv/reskit/iisbook/c01_programmability.mspx?mfr=true

## External links

- ASP on MSDN (http://msdn.microsoft.com/en-us/library/aa286483.aspx)

# Common Language Runtime



The **Common Language Runtime** (**CLR**) is a core component of Microsoft's .NET initiative. It is Microsoft's implementation of the Common Language Infrastructure (CLI) standard, which defines an execution environment for program code. In the CLR, code is expressed in a form of bytecode called the Common Intermediate Language (CIL, previously known as MSIL—Microsoft Intermediate Language).

Developers using the CLR , write code in a language such as C# or VB.NET. At compile time, a .NET compiler converts such code into CIL code. At runtime, the CLR's just-in-time compiler converts the CIL code into code native to the operating system. Alternatively, the CIL code can be compiled to native code in a separate step prior to runtime by using the Native Image Generator (NGEN). This speeds up all later runs of the software as the CIL-to-native compilation is no longer necessary.

Although some other implementations of the Common Language Infrastructure run on non-Windows operating systems, Microsoft's .NET Framework implementation runs only on Microsoft Windows operating systems.

## Services

The CLR allows programmers to ignore many details of the specific CPU that will execute the program. It also provides other important services, including the following:

- Memory management
- Thread management
- Exception handling
- Garbage collection
- Security

## References

- Common Language Runtime Overview (http://msdn2.microsoft.com/en-us/library/ddk909ch.aspx) (Microsoft MSDN)
- "Standard ECMA-335, Common Language Infrastructure (CLI)" (http://www.ecma-international.org/publications/standards/Ecma-335.htm). *Ecma Standards*. ECMA International. Retrieved January 1, 2010.

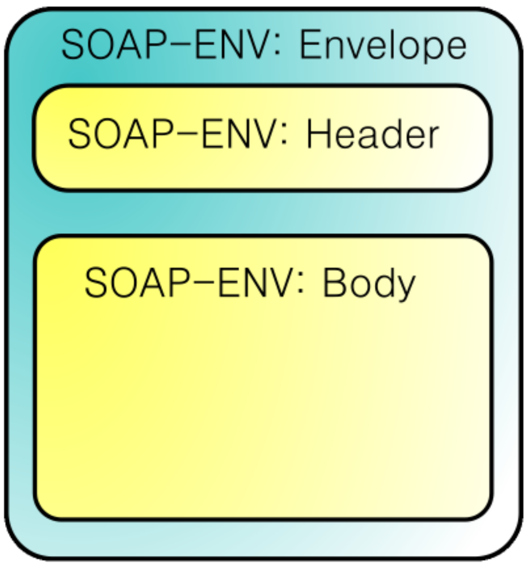
# SOAP

**SOAP**, originally defined as **Simple Object Access Protocol**, is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks. It relies on Extensible Markup Language (XML) for its message format, and usually relies on other Application Layer protocols, most notably Remote Procedure Call (RPC) and Hypertext Transfer Protocol (HTTP), for message negotiation and transmission. SOAP can form the foundation layer of a web services protocol stack, providing a basic messaging framework upon which web services can be built. This XML based protocol consists of three parts: an envelope, which defines what is in the message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing procedure calls and responses.

As an example of how SOAP procedures can be used, a SOAP message could be sent to a web-service-enabled web site, for example, a real-estate price database, with the parameters needed for a search. The site would then return an XML-formatted document with the resulting data, e.g., prices, location, features. Because the data is returned in a standardized machine-parseable format, it could then be integrated directly into a third-party web site or application.

The SOAP architecture consists of several layers of specifications: for message format, Message Exchange Patterns (MEP), underlying transport protocol bindings, message processing models, and protocol extensibility. SOAP is the successor of XML-RPC, though it borrows its transport and interaction neutrality and the envelope/header/body from elsewhere (probably from WDDX).

## History



SOAP structure

**SOAP** once stood for 'Simple Object Access Protocol' but this acronym was dropped with Version 1.2 of the standard.[1] Version 1.2 became a W3C recommendation on June 24, 2003. The acronym is sometimes confused with **SOA**, which stands for Service-oriented architecture; however SOAP is different from SOA.

SOAP was originally designed by Dave Winer, Don Box, Bob Atkinson, and Mohsen Al-Ghosein in 1998 in a project for Microsoft (where Atkinson and Al-Ghosein were already working at the time)[2] , as an object-access protocol. The SOAP specification [3] is currently maintained by the XML Protocol Working Group [4] of the World Wide Web Consortium.

After SOAP was first introduced, it became the underlying layer of a more complex set of Web Services, based on Web Services Description Language (WSDL) and Universal Description Discovery and Integration (UDDI). These services, especially UDDI, have proved to be of far less interest, but an appreciation of them gives a more complete understanding of the expected role of SOAP compared to how web services have actually developed.

## The SOAP specification

The SOAP specification defines the messaging framework which consists of:

- The **SOAP processing model** defining the rules for processing a SOAP message
- The **SOAP extensibility model** defining the concepts of SOAP features and SOAP modules
- The **SOAP underlying protocol binding** framework describing the rules for defining a binding to an underlying protocol that can be used for exchanging SOAP messages between SOAP nodes
- The **SOAP message construct** defining the structure of a SOAP message

### SOAP processing model

The SOAP processing model describes a distributed processing model, its participants, the **SOAP nodes** and how a SOAP receiver processes a SOAP message. The following SOAP nodes are defined:

- **SOAP sender**

A SOAP node that transmits a SOAP message.

- **SOAP receiver**

A SOAP node that accepts a SOAP message.

- **SOAP message path**

The set of SOAP nodes through which a single SOAP message passes.

- **Initial SOAP sender (Originator)**

The SOAP sender that originates a SOAP message at the starting point of a SOAP message path.

- **SOAP intermediary**

A SOAP intermediary is both a SOAP receiver and a SOAP sender and is targetable from within a SOAP message. It processes the SOAP header blocks targeted at it and acts to forward a SOAP message towards an ultimate SOAP receiver.

- **Ultimate SOAP receiver**

The SOAP receiver that is a final destination of a SOAP message. It is responsible for processing the contents of the SOAP body and any SOAP header blocks targeted at it. In some circumstances, a SOAP message might not reach an ultimate SOAP receiver, for example because of a problem at a SOAP intermediary. An ultimate SOAP receiver cannot also be a SOAP intermediary for the same SOAP message.

## Transport methods

SOAP makes use of an internet application layer protocol as a transport protocol. Critics have argued that this is an abuse of such protocols, as it is not their intended function and therefore not a role they fulfill well. Proponents of SOAP have drawn analogies to successful uses of protocols at various levels for tunneling other protocols.

Both SMTP and HTTP are valid application layer protocols used as Transport for SOAP, but HTTP has gained wider acceptance as it works well with today's Internet infrastructure; specifically, HTTP works well with network firewalls. SOAP may also be used over HTTPS (which is the same protocol as HTTP at the application level, but uses an encrypted transport protocol underneath) with either simple or mutual authentication; this is the advocated WS-I method to provide web service security as stated in the WS-I Basic Profile 1.1.

This is a major advantage over other distributed protocols like GIOP/IIOP or DCOM which are normally filtered by firewalls. *SOAP over AMQP* is yet another possibility that some implementations support.

## Message format

XML was chosen as the standard message format because of its widespread use by major corporations and open source development efforts. Additionally, a wide variety of freely available tools significantly eases the transition to a SOAP-based implementation. The somewhat lengthy syntax of XML can be both a benefit and a drawback. While it promotes readability for humans, facilitates error detection, and avoids interoperability problems such as byte-order (Endianness), it can retard processing speed and can be cumbersome. For example, CORBA, GIOP, ICE, and DCOM use much shorter, binary message formats. On the other hand, hardware appliances are available to accelerate processing of XML messages.[5] [6] Binary XML is also being explored as a means for streamlining the throughput requirements of XML.

## Sample SOAP message

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:GetStockPrice xmlns:m="http://www.example.org/stock">
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

## Technical critique

### Advantages

- SOAP is versatile enough to allow for the use of different transport protocols. The standard stacks use HTTP as a transport protocol, but other protocols are also usable (e.g., JMS[7] , SMTP[8] ).
- Since the SOAP model tunnels fine in the HTTP get/response model, it can tunnel easily over existing firewalls and proxies, without modifications to the SOAP protocol, and can use the existing infrastructure.

### Disadvantages

- Because of the verbose XML format, SOAP can be considerably slower than competing middleware technologies such as CORBA. This may not be an issue when only small messages are sent.[9] To improve performance for the special case of XML with embedded binary objects, the Message Transmission Optimization Mechanism was introduced.
- When relying on HTTP as a transport protocol and not using WS-Addressing or an ESB, the roles of the interacting parties are fixed. Only one party (the client) can use the services of the other. Developers must use polling instead of notification in these common cases.

## References

[1] SOAP Version 1.2 specification (http://www.w3.org/TR/soap12-part1/#intro)
[2] Exclusive .NET Developer's Journal "Indigo" Interview with Microsoft's Don Box (http://dotnet.sys-con.com/node/45908)
[3] http://www.w3.org/TR/soap/
[4] http://www.w3.org/2000/xp/Group/
[5] IBM Datapower (http://www-306.ibm.com/software/integration/datapower/xa35/)
[6] IBM Zurich XML Accelerator Engine (http://www.research.ibm.com/XML/IBM_Zurich_XML_Accelerator_Engine_paper_2004May04.pdf)
[7] http://www.w3.org/TR/2009/CR-soapjms-20090604/
[8] http://www.w3.org/TR/2001/WD-soap12-part0-20011217/#SMTP
[9] IBM Developer works (http://www-128.ibm.com/developerworks/library/ws-pyth9/)

## External links

- W3C SOAP page (http://www.w3.org/TR/soap/)
- SOAP Version 1.2 specification (http://www.w3.org/TR/soap12/)
- SOAP Tutorial (http://www.w3schools.com/soap/default.asp)

# Server-side scripting

**Server-side scripting** is a web server technology in which a user's request is fulfilled by running a script directly on the web server to generate dynamic web pages. It is usually used to provide interactive web sites that interface to databases or other data stores. This is different from client-side scripting where scripts are run by the viewing web browser, usually in JavaScript. The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores.

From security point of view, server-side scripts are never visible to the browser as these scripts are executes on the server and emit HTML corresponding to user's input to the page.

When the server serves data in a commonly used manner, for example according to the HTTP or FTP protocols, users may have their choice of a number of client programs (most modern web browsers can request and receive data using both of those protocols). In the case of more specialized applications, programmers may write their own server, client, and communications protocol, that can only be used with one another.

Programs that run on a user's local computer without ever sending or receiving data over a network are not considered clients, and so the operations of such programs would not be considered client-side operations.

## Explanation

In the earlier days of the web, server-side scripting was almost exclusively performed by using a combination of C programs, Perl scripts and shell scripts using the Common Gateway Interface (CGI). Those scripts were executed by the operating system, mnemonic coding and the results simply served back by the web server. Nowadays, these and other on-line scripting languages such as ASP and PHP can often be executed directly by the web server itself or by extension modules (e.g. mod_perl or mod php) to the web server. Either form of scripting (i.e., CGI or direct execution) can be used to build up complex multi-page sites, but direct execution usually results in lower overhead due to the lack of calls to external interpreters

Dynamic websites are also sometimes powered by custom web application servers, for example the Python "Base HTTP Server" library, although some may not consider this to be server-side scripting.

## Examples

Some server-side scripting languages:

- ASP
- ANSI C scripts
- ColdFusion Markup Language (*.cfm)
- Java via JavaServer Pages (*.jsp)
- JavaScript using Server-side JavaScript (*.ssjs)
- PHP (*.php)
- SMX (*.smx)
- Lasso (*.lasso)
- WebDNA (*.dna,*.tpl)
- Progress® WebSpeed® (*.r,*.w)

## External links

- Server-side scripting [1] at the Open Directory Project

## References

[1] http://www.dmoz.org/Computers/Programming/Internet/Server_Side_Scripting/

# Active Scripting

**Active Scripting** (formerly known as **ActiveX Scripting**) is the technology used in Windows to implement component-based scripting support. It is based on COM (more precisely, OLE Automation) and allows installation of additional scripting engines in the form of COM modules.

## Uses and history

The Active Scripting technologies were first released in 1996, with the release of the Microsoft Internet Explorer 3.0 (August 1996) and Internet Information Services 3.0 products (December 1996).

Usual applications of Active Scripting include Active Server Pages (ASP) server scripts, Internet Explorer, and Windows Script Host (WSH) scripts automating routine tasks, including use for login scripts, Registry manipulation, and the like. Other administrative uses include Windows Management Instrumentation and Active Directory Service Interfaces. Active Scripting can also be used for general-purpose scripting, such as database programming, text-processing, rapid prototyping, and application macro/scripting programming; some applications use Active Scripting as the main automation method, others do not have a macro facility but the components are available for use via the API; or one may opt to add a language and/or tool not available by default, like programming Microsoft Excel in Perl or REXX rather than Visual Basic for Applications (VBA) or transferring data from a terminal emulator to word processor by way of a spreadsheet when they have dissimilar macro tools or none at all.

For many of the above uses, Active Scripting is an addition to Windows which could be said to be similar to the functionality of Unix shell scripts, as well as an incrementation upon batch files (command.com), Windows NT style shell scripts (cmd.exe) and can even be said to be, by way of VBScript, the replacement for QBasic – which was last available on the supplementary disc for Windows 95. The majority of the languages used for Active Scripting mentioned below are therefore glue languages and Perl is the most commonly-used third-party script engine.

The interfaces to Active Scripting engines are public, so any developer can create his own applications that are programmable in Active Scripting languages as well as engines for additional languages.

VBScript and JScript engines are included with the default installation of Windows versions after Windows 95, and are an optional install with CE. According to Microsoft and third-party documentation, Visual Basic for Applications (VBA) is a third default scripting engine and is part of the Windows installation and therefore present even if there is not an installation of Microsoft Office, WordPerfect Office, or other software packages which are VBA-programmable. Active Scripting engines for other languages are also available; many are free, some are proprietary (commercial), and at least one shareware engine (a Tcl engine in the beta stage of development) is extant. For example, one can add support for Perl and Python scripting to Windows by installing the ActiveState Active Scripting engines which are included in the ActivePerl and ActivePython distributions. The standard PHP installation for Windows includes an engine known as ActivePHP and PHPScript in various versions. Scripting engines implementing other variants of Basic, Haskell, PHP, REXX (multiple versions), Delphi, XSLT, Tcl, Ruby and other languages are also available.

In Windows, CScript.exe at the command line and WScript.exe running in the GUI are the main means of implementation of installed Active Script languages. Clicking on an icon or running from the command line, a script, the Run dialogue, &c. will by default run a plain text file containing the code. Windows Script Files (.wsf) are an XML file which can contain more than one script in more than one language in addition to other elements, and are executed by the Windows Script Host.

The third-party shell Take Command can, as of version 10, be configured for direct interoperability with the script host and its installed engines.

The script host, related components, and engines are able to be integrated into and called from Windows applications just like any other component.

### Deprecation

Active Scripting is now deprecated in favor of .NET[1] , with some continuing use of run-time interpretation of JScript with JScript .NET and VBScript within ASP.NET. No new versions of the active scripting engines will be developed and they are now being supported by Microsoft's *Sustaining Engineering Team*, who are responsible for bug fixes and security enhancements. However version 5.6 of the scripting engines will continue to be shipped with future releases of Microsoft Windows and IIS[2] . Microsoft has also introduced Windows PowerShell which can expose applications via PowerShell cmdlets and/or PowerShell providers.

Originally, the .NET Framework had a scripting technology of its own and a separate scripting IDE called *Visual Studio for Applications* (VSA)[3] [4] , and the interfaces to the technology were also available via Active Scripting, allowing even .NET-unaware applications to be scripted using .NET languages. VSA was also meant to replace Visual Basic for Applications[5] .

However, that entire technology was deprecated in version 2.0 of the .NET Framework,[5] leaving no clear upgrade path for applications desiring Active Scripting support (although "scripts" can be created in C#, VBScript, Visual Basic .NET, and other .NET languages, which can be compiled and executed at run-time via libraries installed as part of the standard .NET runtime).

## See also

- ActiveX
- Dynamic Language Runtime

## References

[1] *Introducing JScript.NET* (http://msdn.microsoft.com/en-us/library/ms974588.aspx#scripting0714_topic1), by Andrew Clinick of Microsoft Corporation, in Scripting Clinic on MSDN (July 14, 2000)

[2] *Rumours of VBScript's Death Have Been Greatly Exaggerated* (http://blogs.msdn.com/ericlippert/archive/2004/04/09/110508.aspx), on Eric Lippert's Blog *Fabulous Adventures In Coding* on MSDN (April 09, 2004)

[3] *Script Happens .NET* (http://msdn.microsoft.com/en-us/library/ms974577.aspx), article by Andrew Clinick of Microsoft Corporation, in Scripting Clinic on MSDN (July 25, 2001)

[4] *Microsoft Takes Wraps Off VSA Development Technology* (http://redmondmag.com/news/article.asp?EditorialsID=126), by Scott Bekker on Redmondmag.com (January 16, 2001)

[5] *VSA scripting in .NET* (http://www.codeproject.com/csharp/vsascripting.asp), by Mark Belles on The Code Project

## External links

- Future of VBScript Language (http://blogs.msdn.com/ericlippert/archive/2004/04/09/110508.aspx) - Information about the future of Active Scripting technologies.

# Dynamic web page

A **dynamic web page** is a kind of web page that has been prepared with fresh information (content and/or layout), for each individual viewing. It is not static because it changes with the time (ex. a news content), the user (ex. preferences in a login session), the user interaction (ex. web page game), the context (parametric customization), or any combination of the foregoing.

## Properties associated with dynamic web pages

Classical hypertext navigation occurs among "static" documents, and, for *web users*, this experience is reproduced using static web pages, meaning that a page retrieved by different users at different times is always the same, in the same form.

However, a web page can also provide a *live* user experience. Content (text, images, form fields, etc.) on a web page can change in response to different contexts or conditions. In dynamic sites, page content and page layout are created separately. The content is retrieved from a database and is placed on a web page only when needed or asked. This allows for quicker page loading, and it allows just about anyone with limited web design experience to update their own website via an administrative tool. This set-up is ideal for those who wish to make frequent changes to their websites including text and image updates, e.g. e-commerce.

## Two types of dynamic web sites

### Client-side scripting and content creation

Using client-side scripting to change interface behaviors *within* a specific web page, in response to mouse or keyboard actions or at specified timing events. In this case the dynamic behavior occurs within the presentation.

Such web pages use presentation technology called rich interfaced pages. Client-side scripting languages like JavaScript or ActionScript, used for Dynamic HTML (DHTML) and Flash technologies respectively, are frequently used to orchestrate media types (sound, animations, changing text, etc.) of the presentation. The scripting also allows use of remote scripting, a technique by which the DHTML page requests additional information from a server, using a hidden Frame, XMLHttpRequests, or a Web service.

The Client-side content is generated on the user's computer. The web browser retrieves a page from the server, then processes the code embedded in the page (often written in JavaScript) and displays the retrieved page's content to the user.

The innerHTML property (or write command) can illustrate the client-side dynamic page generation: two distinct pages, A and B, can be regenerated as document.innerHTML = A and document.innerHTML = B; or "on load dynamic" by document.write(A) and document.write(B).

There are also some utilities and frameworks for converting HTML files into JavaScript files. For example webJS[1] uses innerHTML property for rendering pages from converted HTML on client-side.

The first "widespread used" version of JavaScript was 1996 (with Netscape 3 an ECMAscript standard).

### Server-side scripting and content creation

Using server-side scripting to change the supplied page source *between* pages, adjusting the sequence or reload of the web pages or web content supplied to the browser. Server responses may be determined by such conditions as data in a posted HTML form, parameters in the URL, the type of browser being used, the passage of time, or a database or server state.

Such web pages are often created with the help of server-side languages such as PHP, Perl, ASP, ASP.NET, JSP, ColdFusion and other languages. These server-side languages typically use the Common Gateway Interface (CGI) to produce *dynamic web pages*. These kinds of pages can also use, on the client-side, the first kind (DHTML, etc.).

Server-side dynamic content is more complicated: (1) The client sends the server the request. (2) The server receives the request and processes the server-side script such as [PHP] based on the query string, HTTP POST data, cookies, etc.

The dynamic page generation was made possible by the Common Gateway Interface, stable in 1993. Then Server Side Includes pointed a more direct way to deal with server-side scripts, at the web servers.

### Combining client and server side

Ajax is a web development technique for dynamically interchanging content with the server-side, without reloading the web page. Google Maps is an example of a web application that uses Ajax techniques and database.

## Disadvantages

- Search engines work by creating indexes of published HTML web pages that were, initially, "static". With the advent of dynamic web pages, often created from a private database, the content is less visible[2] . Unless this content is duplicated in some way (for example, as a series of extra static pages on the same site), a search may not find the information it is looking for. It is unreasonable to expect generalized web search engines to be able to access complex database structures, some of which in any case may be secure.

## History

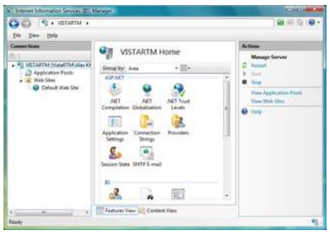
It is difficult to be precise about "dynamic web page beginnings" or chronology, because the precise concept makes sense only after the "widespread development of web pages": HTTP protocol has been in use since 1990, HTML, as standard, since 1996. The web browsers explosion started with 1993's Mosaic. It is obvious, however, that the concept of dynamically driven websites predates the internet, and in fact HTML. For example, in 1990, before the general public use of the internet, a dynamically driven remotely accessed menu system was implimented by Susan Biddlecomb, at the University of Southern California BBS on a 16 line TBBS system with TDBS add-on.

## References

- "The Information Revolution", J. R. Okin. ISBN 0976385740. Ed. Ironbound Press, 2005. 350 pp.
- "Learning VBScript", P. Lomax. ISBN 1565922476. Ed. O'Reilly, 1997. sec. C13.

[1] http://code.google.com/p/webjs/
[2] http://www.hypergurl.com/articles/dynamic.html

# Internet Information Services



Screenshot of IIS 7's IIS Manager console

| | |
|---|---|
| **Developer(s)** | Microsoft |
| **Stable release** | 7.5 / 10/22/2009 |
| **Operating system** | Microsoft Windows |
| **Type** | Server |
| **License** | Proprietary |
| **Website** | www.microsoft.com/iis [1] |

**Internet Information Services** (**IIS**) – formerly called **Internet Information Server** – is a web server application and set of feature extension modules created by Microsoft for use with Microsoft Windows. It is the second most used web server behind Apache HTTP Server. As of March 2010, it served 22.7% of all websites on the Internet according to Netcraft.[2] The protocols supported in IIS 7.5 include: FTP, FTPS, SMTP, NNTP, and HTTP/HTTPS.

## Versions

- IIS 1.0, Windows NT 3.51 available as a free add-on
- IIS 2.0, Windows NT 4.0
- IIS 3.0, Windows NT 4.0 Service Pack 3
- IIS 4.0, Windows NT 4.0 Option Pack
- IIS 5.0, Windows 2000
- IIS 5.1, Windows XP Professional, Windows XP Media Center Edition (requires retail CD)
- IIS 6.0, Windows Server 2003 and Windows XP Professional x64 Edition
- IIS 7.0, Windows Server 2008 and Windows Vista (Home Premium, Business, Enterprise, Ultimate Editions)
- IIS 7.5, Windows Server 2008 R2 and Windows 7

IIS is not turned on by default when Windows is installed, but it can be selected from the list of optional features. It is available in all editions of Windows Vista and Windows 7, including Home Basic, but some features are not supported on client versions of Windows.

## History

The first Microsoft web server was a research project at European Microsoft Windows NT Academic Centre (EMWAC), part of the University of Edinburgh in Scotland, and was distributed as freeware.[3] However since the EMWAC server was unable to scale sufficiently to handle the volume of traffic going to microsoft.com, Microsoft was forced to develop its own webserver, IIS.[4]

IIS was initially released as a set of web-based services for Windows NT 3.51.

IIS 2.0 added support for the Windows NT 4.0 operating system.

IIS 3.0 introduced the Active Server Pages dynamic scripting environment.[5]

IIS 4.0 dropped support for the Gopher protocol and was released as part of an "Option Pack" for Windows NT 4.0.

IIS 5.0 shipped with Windows 2000 and introduced additional authentication methods, management enhancements including a new MMC based administration application, support for the WebDAV protocol, and enhancements to ASP [6]

IIS 5.1 shipped with Windows XP Professional, and was nearly identical to IIS 5.0 on Windows 2000 except for several limitations Microsoft introduced. IIS 5.1 supported only 10 simultaneous connections and supported only a single web site.[7] [8]

IIS 6.0 added support for IPv6 and included a new worker process model that increased security as well as reliability.[9]

IIS 7.0 was a complete redesign and rewrite of IIS, which shipped with Windows Vista and Windows 2008. IIS 7.0 included a new modular design that allowed for a lessened attack surface and increased performance. IIS 7.0 also introduced a hierarchical configuration system allowing for simpler site deploys, a new Windows Forms based management application, new command line management options, and increased support for the .NET Framework.[10] IS 7.0 on Vista does not limit the number of allowed connections as IIS on XP did, but limits concurrent requests to 10 (Windows Vista Ultimate, Business, and Enterprise Editions) or 3 (Vista Home Premium). Additional requests are queued, which hampers performance, but they are not rejected as with XP.

The current shipping version of IIS is 7.5 for Windows 7 and Windows Server 2008 R2. IIS 7.5 introduced improvements to WebDAV / FTP modules, improvements to command line administration in PowerShell, and improvements to security with a newly included Best Practices Analyzer tool and process isolation for application pools.[11]

## Security

Earlier versions of IIS were hit with a number of vulnerabilities, chief among them CA-2001-19 [12] which led to the infamous Code Red worm; however, both versions 6.0 and 7.0 currently have no reported issues with this specific vulnerability.[13] [14] In IIS 6.0 Microsoft opted to change the behaviour of pre-installed ISAPI handlers,[15] many of which were culprits in the vulnerabilities of 4.0 and 5.0, thus reducing the attack surface of IIS. In addition, IIS 6.0 added a feature called "Web Service Extensions" that prevents IIS from launching any program without explicit permission by an administrator.

In the current release, IIS 7, the components are provided as modules so that only the required components have to be installed, thus further reducing the attack surface. In addition, security features are added such as Request Filtering, which rejects suspicious URLs based on a user-defined rule set.

By default IIS 5.1 and lower run websites in-process under the SYSTEM account,[16] a default Windows account with 'superuser' rights. Under 6.0 all request handling processes have been brought under a Network Services account with significantly fewer privileges so that should there be a vulnerability in a feature or in custom code it won't necessarily compromise the entire system given the sandboxed environment these worker processes run in. IIS 6.0 also contained a new kernel HTTP stack (http.sys) with a stricter HTTP request parser and response cache for both static and dynamic content.

There are various built-in security features from Microsoft. Many companies offer third-party security tools and features, also known as "Web App Firewalls, or Web Application Firewalls." The advantage of such tools is that they offer much more comprehensive elements (such as easy-to-use GUI, etc.) that aid in protecting an IIS installation with an additional layer of protection at a higher level. Though no security system is ever complete, most admins choose to run an application-layer firewall and an Intrusion Prevention System.

## Features

IIS 7 is built on a modular architecture. Modules, also called extensions, can be added or removed individually so that only modules required for specific functionality have to be installed. IIS 7 includes native modules as part of the full installation. These modules are individual features that the server uses to process requests and include the following:

- HTTP modules – Used to perform tasks specific to HTTP in the request-processing pipeline, such as responding to information and inquiries sent in client headers, returning HTTP errors, and redirecting requests.
- Security modules – Used to perform tasks related to security in the request-processing pipeline, such as specifying authentication schemes, performing URL authorization, and filtering requests.
- Content modules – Used to perform tasks related to content in the request-processing pipeline, such as processing requests for static files, returning a default page when a client does not specify a resource in a request, and listing the contents of a directory.
- Compression modules – Used to perform tasks related to compression in the request-processing pipeline, such as compressing responses, applying Gzip compression transfer coding to responses, and performing pre-compression of static content.
- Caching modules – Used to perform tasks related to caching in the request-processing pipeline, such as storing processed information in memory on the server and using cached content in subsequent requests for the same resource.
- Logging and Diagnostics modules – Used to perform tasks related to logging and diagnostics in the request-processing pipeline, such as passing information and processing status to HTTP.sys for logging, reporting events, and tracking requests currently executing in worker processes.

IIS 5.0 and higher support the following authentication mechanisms:

- Basic access authentication
- Digest access authentication
- Integrated Windows Authentication
- .NET Passport Authentication (not supported in Windows Server 2008 and above)

IIS 7.5 includes the following additional security features:

- Client Certificate Mapping
- IP Security
- Request Filtering
- URL Authorization

Authentication changed slightly between IIS 6.0 and IIS 7, most notably in that the anonymous user which was named "IUSR_{machinename}" is a built-in account in Vista and future operating systems and named "IUSR". Notably, in IIS 7, each authentication mechanism is isolated into its own module and can be installed or uninstalled.

### Extensions

IIS releases new feature modules between major version releases to add new functionality. The following extensions are available for IIS 7:

- **FTP Publishing Service** – Lets Web content creators publish content securely to IIS 7 Web servers with SSL-based authentication and data transfer.
- **Administration Pack** – Adds administration UI support for management features in IIS 7, including ASP.NET authorization, custom errors, FastCGI configuration, and request filtering.
- **Application Request Routing** – Provides a proxy-based routing module that forwards HTTP requests to content servers based on HTTP headers, server variables, and load balance algorithms.
- **Database Manager** – Allows easy management of local and remote databases from within IIS Manager.
- **Media Services** – Integrates a media delivery platform with IIS to manage and administer delivery of rich media and other Web content.
- **URL Rewrite Module** – Provides a rule-based rewriting mechanism for changing request URLs before they are processed by the Web server.
- **WebDAV** – Lets Web authors publish content securely to IIS 7 Web servers, and lets Web administrators and hosters manage WebDAV settings using IIS 7 management and configuration tools.
- **Web Deployment Tool** – Synchronizes IIS 6.0 and IIS 7 servers, migrates an IIS 6.0 server to IIS 7, and deploys Web applications to an IIS 7 server.

## See also

- Windows Activation Services
- Apache HTTP Server
- Lighttpd
- PWS
- List of FTP server software
- List of mail servers
- Comparison of web servers
- Metabase
- ASP.NET
- Windows Communication Foundation
- LogParser – SQL-like querying of various log

## References

[1] http://www.microsoft.com/iis
[2] "Netcraft Web Server Survey, November 2010" (http://news.netcraft.com/archives/2010/11/05/november-2010-web-server-survey.html). . Retrieved 2010-11-28.
[3] "Windows NT Internet Servers" (http://support.microsoft.com/kb/120734). Microsoft. July 10, 2002. . Retrieved 2008-05-26.
[4] Dave Kramer (December 24, 1999). "A Brief History of Microsoft on the Web" (http://www.microsoft.com/misc/features/features_flshbk.htm). Microsoft. .
[5] "Microsoft ASP.NET 2.0 Next Stop on Microsoft Web Development Roadmap" (http://www.directionsonmicrosoft.com/sample/DOMIS/update/2004/08aug/0804a2nsow.htm). .
[6] "Chapter 1 - Overview of Internet Information Services 5.0" (http://technet.microsoft.com/en-us/library/bb742405.aspx). . Retrieved 2010-10-25.
[7] "Internet Information Services 5.1" (http://www.microsoft.com/windowsxp/evaluation/features/iis.mspx). . Retrieved 2007-07-20.
[8] "IIS Answers - IIS5.1 on XP Pro" (http://www.iisanswers.com/IIS51.htm). . Retrieved 2010-10-25.
[9] "What's New In IIS 6.0?" (http://www.devx.com/webdev/Article/17085/). . Retrieved 2010-11-25.

[10] "IIS 7.0: Explore The Web Server For Windows Vista and Beyond" (http://msdn.microsoft.com/en-us/magazine/cc163453.aspx). . Retrieved 2010-11-25.

[11] "What's New in Web Server (IIS) Role in Windows 2008 R2" (http://technet.microsoft.com/en-us/library/dd560629(WS.10).aspx). . Retrieved 2010-11-25.

[12] http://www.cert.org/advisories/CA-2001-13.html

[13] "Vulnerability Report: Microsoft Internet Information Services (IIS) 6" (http://secunia.com/advisories/product/1438/?task=statistics). . Retrieved 2008-10-14.

[14] "Vulnerability Report: Microsoft Internet Information Services (IIS) 7" (http://secunia.com/advisories/product/17543/?task=statistics). . Retrieved 2008-10-14.

[15] "IIS Installs in a Locked-Down Mode (IIS 6.0)" (http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/54257c42-d723-4b12-badf-f4902c195821.mspx?mfr=true). MSDN. . Retrieved 2007-07-20.

[16] "HOW TO: Run Applications Not in the Context of the System Account in IIS#Default Installation" (http://support.microsoft.com/kb/319067/). . Retrieved 2007-07-20.

## External links

- IIS.NET (http://www.iis.net) – The Official Microsoft IIS Site
- IIS 7.5 Product Page (http://www.microsoft.com/iis) – Windows Server 2008 R2
- Security Guidance for IIS (http://www.microsoft.com/technet/security/prodtech/iis.mspx) – Microsoft TechNet
- Microsoft Web Platform Installer (http://www.microsoft.com/web/downloads/platform.aspx) – A free tool to install IIS and other components of the Microsoft Web Platform
- Microsoft WebsiteSpark Program (http://www.microsoft.com/web/websitespark) – A program that offers IIS and other Microsoft Web Platform products at no upfront cost for 3 years

# Common Language Infrastructure

The **Common Language Infrastructure (CLI)** is an open specification developed by Microsoft that describes the executable code and runtime environment that form the core of the Microsoft .NET Framework and the free and open source implementations Mono and Portable.NET. The specification defines an environment that allows multiple high-level languages to be used on different computer platforms without being rewritten for specific architectures.

## Overview

Among other things, the CLI specification describes the following four aspects:-

The Common Type System (CTS)

A set of data types and operations that are shared by all CTS-compliant programming languages.

Metadata

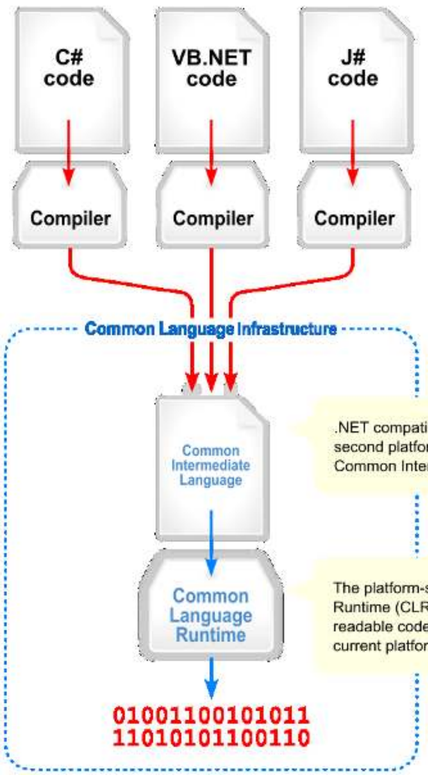
Information about program structure is language-agnostic, so that it can be referenced between languages and tools, making it easy to work with code written in a language you are not using.

Common Language Specification (CLS)

A set of base rules to which any language targeting the CLI should conform in order to interoperate with other CLS-compliant languages. The CLS rules define a subset of the Common Type System.

Virtual Execution System (VES)

The VES loads and executes CLI-compatible programs, using the metadata to combine separately generated pieces of code at runtime.


C# code
VB.NET code
J# code
Compiler
Compiler
Compiler
Common Language Infrastructure
Common Intermediate Language
.NET compatible languages compile to a second platform-neutral language called Common Intermediate Language (CIL).
Common Language Runtime
The platform-specific Common Language Runtime (CLR) compiles CIL to machine-readable code that can be executed on the current platform.
01001100101011
11010101100110
Visual overview of the Common Language Infrastructure (CLI)

All compatible languages compile to Common Intermediate Language (CIL), which is an intermediate language that is abstracted from the platform hardware. When the code is executed, the platform-specific VES will compile the CIL to the machine language according to the specific hardware.

## Standardization and licensing

In August 2000, Microsoft, Hewlett-Packard, Intel, and others worked to standardize CLI. By December 2001, it was ratified by the ECMA, with ISO standardization following in April 2003.

Microsoft and its partners hold patents for CLI. ECMA and ISO require that all patents essential to implementation be made available under "reasonable and non-discriminatory (RAND) terms", but interpretation of this has led to much controversy, particularly in the case of Mono.

As of July 2009[1] . Microsoft applied C# and CLI under the Community Promise [2], which, in certain situations, protects software developers from Microsoft software patents.

## Support for dynamic languages

The Common Language Infrastructure currently has no built-in support for Dynamically typed languages because the existing Common Intermediate Language is statically typed[3] .

The Dynamic Language Runtime is an ongoing effort to bring this support to the CLR.

## Implementations

- The .NET Framework is built on the Common Language Runtime, Microsoft's commercial implementation of the CLI for desktop and server systems, and also encompasses a large collection of programming frameworks and libraries.
- Shared Source Common Language Infrastructure is a reference implementation of the CLI available from Microsoft, under the Shared source licensing program.
- .NET Compact Framework is Microsoft's commercial implementation of the CLI for portable devices and Xbox 360.
- Microsoft Silverlight, an implementation for use in web browsers - for Microsoft Windows and Mac OS X.
- Mono development platform is an open source implementation of CLI and accompanying technologies, sponsored by Novell.
- Portable.NET, part of the dotGNU project, is a Free Software implementation of ECMA-335 by Free Software Foundation.
- VMKit part of Low Level Virtual Machine toolkit as of version 2.3. Implements very incomplete and alpha stage support of a Virtual Execution System. It is based on DotGNU, Portable.NET corelib and class libraries.

## Notes

[1] "The ECMA C# and CLI Standards - Port 25: The Open Source Community at Microsoft" (http://port25.technet.com/archive/2009/07/06/the-ecma-c-and-cli-standards.aspx). 2009-07-06. .

[2] http://www.microsoft.com/interop/cp/default.mspx

[3] Pobar, Joel (2007-01-03). "CLR Dynamic languages under the hood" (http://blogs.msdn.com/joelpob/archive/2005/07/01/434728.aspx). . Retrieved 2008-01-26.

## References

- "Standard ECMA-335, Common Language Infrastructure (CLI)" (http://www.ecma-international.org/publications/standards/Ecma-335.htm). *ECMA International*. Retrieved August 31, 2005.
- "ISO/IEC 23271, Common Language Infrastructure" (http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=42927). *ISO*. Retrieved September 27, 2006.
- "ECMA C# and Common Language Infrastructure Standards" (http://msdn.microsoft.com/en-us/netframework/aa569283.aspx). *Microsoft Corporation*. Retrieved October 13, 2009.

# Common Intermediate Language

**Common Intermediate Language** (**CIL**, pronounced either "sil" or "kil") (formerly called **Microsoft Intermediate Language** or **MSIL**) is the lowest-level human-readable programming language defined by the Common Language Infrastructure specification and used by the .NET Framework and Mono. Languages which target a CLI-compatible runtime environment compile to CIL, which is assembled into bytecode. CIL is an object-oriented assembly language, and is entirely stack-based. It is executed by a virtual machine.

CIL was originally known as Microsoft Intermediate Language (MSIL) during the beta releases of the .NET languages. Due to standardization of C# and the Common Language Infrastructure, the bytecode is now officially known as CIL.

## General information

During compilation of .NET programming languages, the source code is translated into CIL code rather than platform or processor-specific object code. CIL is a CPU- and platform-independent instruction set that can be executed in any environment supporting the Common Language Infrastructure, such as the .NET runtime on Windows, or the cross-platform Mono runtime. In theory, this eliminates the need to distribute separate binaries for different platforms and CPU types. CIL code is verified for safety during runtime, providing better security and reliability than natively compiled binaries.

The execution process looks like this:

1. Source code is converted to Common Intermediate Language, CIL's equivalent to Assembly language for a CPU.
2. CIL is then assembled into bytecode and a .NET assembly is created.
3. Upon execution of a .NET assembly, its bytecode is passed through the runtime's JIT compiler to generate native code. (Ahead-of-time compilation eliminates this step at run time.)
4. The native code is executed by the computer's processor.

## Instructions

*See also: List of CIL instructions*

CIL bytecode has instructions for the following groups of tasks:

- Load and store
- Arithmetic
- Type conversion
- Object creation and manipulation
- Operand stack management (push / pop)
- Control transfer (branching)
- Method invocation and return
- Throwing exceptions
- Monitor-based concurrency

## Computational model

*See: Common Intermediate Language syntax*

The Common Intermediate Language is object-oriented and stack-based. That means that data is pushed on a stack instead of pulled from registers like in most CPU architectures.

In x86 it might look like this:

add eax, edx mov ecx, eax

The corresponding code in IL can be rendered as this:

```
ldloc.0
ldloc.1
add
stloc.0    // a = a + b or a += b;
```

Here are two locals that are pushed on the stack. When the add-instruction is called the operands get popped and the result is pushed. The remaining value is then popped and stored in the first local.

### Object-oriented concepts

This extends to object-oriented concepts as well. You may create objects, call methods and use other types of members such as fields.

CIL is designed to be object-oriented and every method needs (with some exceptions) to reside in a class. So does this static method:

```
.class public Foo
{
    .method public static int32 Add(int32, int32) cil managed
    {
         .maxstack 2
         ldarg.0 // load the first argument;
         ldarg.1 // load the second argument;
         add     // add them;
         ret     // return the result;
    }
}
```

This method does not require any instance of Foo to be declared because it is static. That means it belongs to the class and it may then be used like this in C#:

```
int r = Foo.Add(2, 3);    //5
```

In CIL:

```
ldc.i4.2
ldc.i4.3
call int32 Foo::Add(int32, int32)
stloc.0
```

**Instance classes**

An instance class contains at least one constructor and some instance members. This class has a set of methods representing actions of a Car-object.

```
.class public Car
{
    .method public specialname rtspecialname
        instance void .ctor(int32, int32) cil managed
    {
        /* Constructor */
    }

    .method public void Move(int32) cil managed
    {
        /* Omitting implementation */
    }

    .method public void TurnRight() cil managed
    {
        /* Omitting implementation */
    }

    .method public void TurnLeft() cil managed
    {
        /* Omitting implementation */
    }

    .method public void Brake() cil managed
    {
        /* Omitting implementation */
    }
}
```

**Creating objects**

In C# class instances are created like this:

```
Car myCar = new Car(1, 4);
Car yourCar = new Car(1, 3);
```

And these statements are roughly the same as these instructions:

```
ldc.i4.1
ldc.i4.4
newobj instance void Car::.ctor(int, int)
stloc.0    // myCar = new Car(1, 4);
ldc.i4.1
ldc.i4.3
newobj instance void Car::.ctor(int, int)
stloc.1    // yourCar = new Car(1, 3);
```

**Invoking instance methods**

Instance methods are invoked like the one that follows:

```
myCar.Move(3);
```

In CIL:

```
ldloc.0    // Load the object "myCar" on the stack
ldc.i4.3
call instance void Car::Move(int32)
```

### Metadata

.NET records information about compiled classes as Metadata. Like the type library in the Component Object Model, this enables applications to support and discover the interfaces, classes, types, methods, and fields in the assembly. The process of reading such metadata is called *reflection*.

Metadata can be data in the form of *attributes*. Attributes can be custom made by extending from the Attribute class. This is a very powerful feature.

## Example

Below is a basic Hello, World program written in CIL. It will display the string "Hello, world!".

```
.assembly Hello {}
.assembly extern mscorlib {}
.method static void Main()
{
     .entrypoint
     .maxstack 1
     ldstr "Hello, world!"
     call void [mscorlib]System.Console::WriteLine(string)
     call string[mscorlib]System.Console::ReadLine()
     pop
     ret
}
```

The following code is more complex in number of opcodes.

*This code can also be compared with the corresponding code in the article about Java Bytecode.*

```
static void Main(string[] args)
{
outer:
    for (int i = 2; i < 1000; i++)
    {
         for (int j = 2; j < i; j++)
         {
              if (i % j == 0)
                   goto outer;
         }
         Console.WriteLine(i);
     }
}
```

In CIL syntax it looks like this:

```
.method private hidebysig static void Main(string[] args) cil managed
{
  .entrypoint
  .maxstack  2
  .locals init (int32 V_0,
                int32 V_1)

  IL_0000:  ldc.i4.2
            stloc.0
            br.s       IL_001f
  IL_0004:  ldc.i4.2
            stloc.1
            br.s       IL_0011
  IL_0008:  ldloc.0
            ldloc.1
            rem
            brfalse.s  IL_0000
            ldloc.1
            ldc.i4.1
            add
            stloc.1
  IL_0011:  ldloc.1
            ldloc.0
            blt.s      IL_0008
            ldloc.0
            call       void [mscorlib]System.Console::WriteLine(int32)
            ldloc.0
            ldc.i4.1
            add
            stloc.0
  IL_001f:  ldloc.0
            ldc.i4     0x3e8
            blt.s      IL_0004
            ret
}
```

This is just a representation of how CIL looks like near VM-level. When compiled the methods are stored in tables and the instructions are stored as bytes inside the assembly, which is a Portable Executable-file (PE).

## Generation

A CIL assembly and instructions are generated by either a compiler or a utility called the *IL Assembler* (ILASM) that is shipped with the execution environment.

Assembled IL can also be disassembled into code again using the *IL Disassembler* (ILDASM). There are other tools such as .NET Reflector that can decompile IL into a high-level language (e.g. C# or VB). This makes IL a very easy target for reverse engineering. This trait is shared with Java Bytecode. However, there are tools that can obfuscate the code, and do it so that the code cannot be disassembled but still be runnable.

## Execution

### Just-in-time compilation

Just-in-time compilation involves turning the byte-code into code immediately executable by the CPU. The conversion is performed gradually during the program's execution. JIT compilation provides environment-specific optimization, runtime type safety, and assembly verification. To accomplish this, the JIT compiler examines the assembly metadata for any illegal accesses and handles violations appropriately.

### Ahead-of-time compilation

CLI-compatible execution environments also come with the option to do a Ahead-of-time compilation (AOT) of an assembly to make it execute faster by removing the JIT process at runtime.

In the .NET Framework there is a special tool called the Native Image Generator (NGEN) that performs the AOT. In Mono there is also an option to do an AOT.

## See also

- List of CIL instructions

## External links

- Common Language Infrastructure (Standard ECMA-335) [13]
- "ECMA C# and Common Language Infrastructure Standards" on MSDN [1]
- Hello world program in CIL
- Kenny Kerr's intro to CIL (called MSIL in the tutorial) [2]
- Speed: NGen Revs Up Your Performance With Powerful New Features -- MSDN Magazine, April 2005 [3]

## References

[1] http://msdn.microsoft.com/en-us/netframework/aa569283.aspx
[2] http://weblogs.asp.net/kennykerr/archive/2004/09/07/introduction-to-msil-part-1-hello-world.aspx
[3] http://msdn.microsoft.com/en-us/magazine/cc163808.aspx

# Software framework

In computer programming, a **software framework** is an abstraction in which common code providing generic functionality can be selectively overridden or specialized by user code, thus providing specific functionality. Frameworks are a special case of software libraries in that they are reusable abstractions of code wrapped in a well-defined Application programming interface (API), yet they contain some key distinguishing features that separate them from normal libraries.

Software frameworks have these distinguishing features that separate them from libraries or normal user applications:

1. **inversion of control** - In a framework, unlike in libraries or normal user applications, the overall program's flow of control is not dictated by the caller, but by the framework.[1]
2. **default behavior** - A framework has a default behavior. This default behavior must actually be some useful behavior and not a series of no-ops.
3. **extensibility** - A framework can be extended by the user usually by selective overriding or specialized by user code providing specific functionality.
4. **non-modifiable framework code** - The framework code, in general, is not allowed to be modified. Users can extend the framework, but not modify its code.

There are different types of software frameworks: conceptual, application, domain, platform, component, service, development, etc.[2]

## Rationale

The designers of software frameworks aim to facilitate software development by allowing designers and programmers to devote their time to meeting software requirements rather than dealing with the more standard low-level details of providing a working system, thereby reducing overall development time.[3] For example, a team using a web application framework to develop a banking web-site can focus on the operations of account withdrawals rather than the mechanics of request handling and state management.

By way of contrast, an in-house or purpose-built framework might be specified for the same project by a programming team as they begin working the overall job — specifying software needs based on first defining data types, structures and processing has long been taught as a successful strategy for top down design. Contrasting software data, its manipulation, and how a software system's various grades and kinds of users will need to either input, treat, or output the data are then used to specify the user interface(s) — some types of access being privileged and locked to other user types — all defining the overall user interfaces which to the users are the visible in-house Framework for the custom coded project. In such a case, each sort of operation, user interface code and so forth need written and separately integrated into the job at hand also more or less adding to necessary testing and validation.

It can be argued that frameworks add to "code bloat", and that due to customer-demand driven applications needs both competing and complementary frameworks sometimes end up in a product. Further, some cognoscenti argue, because of the complexity of their APIs, the intended reduction in overall development time may not be achieved due to the need to spend additional time learning to use the framework — which criticism is certainly valid when a special or new framework is first encountered by a development staff. If such a framework is not utilized in subsequent job taskings, the time invested ascending the framework's learning curve might be more expensive than purpose written code familiar to the project's staff — many programmers keep aside useful boilerplate for common needs.

• However, it could be argued that once the framework is learned, future projects might be quicker and easier to complete — the concept of a framework is to make a one-size-fits-all solution set, and with familiarity, code production should logically be increased. There are no such claims made about the size of the code eventually bundled with the output product, nor its relative efficiency and conciseness. Using any library solution necessarily pulls in extras and unused extraneous assets unless the software is a compiler-object linker making a tight (small, wholly controlled, and specified) executable module.

• The issue continues, but a decade-plus of industry experience has shown that the most effective frameworks turn out to be those that evolve from re-factoring the common code of the enterprise, as opposed to using a generic "one-size-fits-all" framework developed by third-parties for general purposes. An example of that would be how the user interface in such an application package as an Office suite grows to have common look, see, feel and data sharing attributes and methods as the once disparate bundled applications become unified — hopefully a suite which is tighter and smaller as the newer evolved one can be a product sharing integral utility libraries and user interfaces.

This trend in the controversy brings up an important issue about frameworks. Creating a framework that is elegant, as opposed to one that merely solves a problem, is still an art rather than a science. "Software elegance" implies clarity, conciseness, and little waste (extra or extraneous functionality — much of which is user defined). For those frameworks that generate code, for example, "elegance" would imply the creation of code that is clean and comprehensible to a reasonably knowledgeable programmer (and which is therefore readily modifiable), as opposed to one that merely generates correct code. The elegance issue is why relatively few software frameworks have stood the test of time: the best frameworks have been able to evolve gracefully as the underlying technology on which they were built advanced. Even there, having evolved, many such packages will retain legacy capabilities bloating the final software as otherwise replaced methods have been retained in parallel with the newer methods.

## Examples

Software frameworks typically contain considerable housekeeping and utility code in order to help bootstrap user applications, but generally focus on specific problem domains, such as:

- Artistic drawing, music composition, and mechanical CAD[4] [5]
- Compilers for different programming languages and target machines.[6]
- Financial modeling applications[7]
- Earth system modeling applications[8]
- Decision support systems[9]
- Media playback and authoring
- Web applications
- Middleware

## Architecture

According to Pree,[10] software frameworks consist of *frozen spots* and *hot spots*. *Frozen spots* define the overall architecture of a software system, that is to say its basic components and the relationships between them. These remain unchanged (frozen) in any instantiation of the application framework. *Hot spots* represent those parts where the programmers using the framework add their own code to add the functionality specific to their own project.

Software frameworks define the places in the architecture where application programmers may make adaptations for specific functionality—the hot spots.

In an object-oriented environment, a framework consists of abstract and concrete classes. Instantiation of such a framework consists of composing and subclassing the existing classes.[11]

When developing a concrete software system with a software framework, developers utilize the hot spots according to the specific needs and requirements of the system. Software frameworks rely on the Hollywood Principle: "Don't call us, we'll call you."[12] This means that the user-defined classes (for example, new subclasses), receive messages from the predefined framework classes. Developers usually handle this by implementing superclass abstract methods.

## See also

- Application framework
- Class (computer science)
- Design pattern (computer science)
- Don't repeat yourself
- Enterprise Architecture framework
- Implicit invocation
- Programming paradigm
- Web application framework

## References

[1] Riehle, Dirk (2000), *Framework Design: A Role Modeling Approach* (http://www.riehle.org/computer-science/research/dissertation/diss-a4.pdf), Swiss Federal Institute of Technology,

[2] Shan, Tony (2006). "Taxonomy of Java Web Application Frameworks" (http://portal.acm.org/citation.cfm?id=1190953). Proceedings of 2006 IEEE International Conference on e-Business Engineering (ICEBE 2006). . Retrieved 2010-10-10.

[3] "Framework" (http://docforge.com/wiki/Framework). *DocForge*. . Retrieved 15 December 2008.

[4] Vlissides, J M; Linton, M A (1990), "Unidraw: a framework for building domain-specific graphical editors", *ACM Transactions of Information Systems* **8** (3): 237–268, doi:10.1145/98188.98197

[5] Johnson, R E (1992), "Documenting frameworks using patterns", *Proceedings of the Conference on Object Oriented Programming Systems Languages and Applications* (ACM Press): 63–76

[6] Johnson, R E; McConnell, C; Lake, M J (1992), Giegerich, R; Graham, S L, eds., "The RTL system: a framework for code optimization", *Proceedings of the International workshop on code generation* (Springer-Verlag): 255–274

[7] Birrer, A; Eggenschwiler, T (1993), *Frameworks in the financial engineering domain: an experience report*, Springer-Verlag, pp. 21–35

[8] Hill, C; DeLuca, C; Balaji, V; Suarez, M; da Silva, A (2004), "Architecture of the Earth System Modeling Framework (ESMF)", *Computing in Science and Engineering*: 18–28

[9] Gachet, A (2003), "Software Frameworks for Developing Decision Support Systems - A New Component in the Classification of DSS Development Tools", *Journal of Decision Systems* **12** (3): 271–281

[10] Pree, W (1994), "Meta Patterns-A Means For Capturing the Essentials of Reusable Object-Oriented Design", *Proceedings of the 8th European Conference on Object-Oriented Programming* (Springer-Verlag): 150–162

[11] Buschmann, F (1996), *Pattern-Oriented Software Architecture Volume 1: A System of Patterns. Chichester*, Wiley, ISBN 0471958697

[12] Larman, C (2001), *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process* (2nd ed.), Prentice Hall, ISBN 0130925691

# Template (software engineering)

The term **template**, when used in the context of software engineering has various technical specifications, but is generally identified as any processing element that can be combined with a data model and processed by a template engine to produce a **result document**.

## Overview

Template-based text processing by a preprocessor or macro engine has a long history.

# Session (computer science)

In computer science, in particular networking, a **session** is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices, or between a computer and user (see Login session). A session is set up or established at a certain point in time, and torn down at a later point in time. An established communication session may involve more than one message in each direction. A session is typically, but not always, stateful, meaning that at least one of the communicating parts needs to save information about the session history in order to be able to communicate, as opposed to stateless communication, where the communication consists of independent requests with responses.

**Communication sessions** may be implemented as part of protocols and services at the application layer, at the session layer or at the transport layer in the OSI model.

- Application layer examples:
  - HTTP sessions, which allow associating information with individual visitors
  - A telnet remote login session
- Session layer example:
  - A Session Initiation Protocol (SIP) based Internet phone call
- Transport layer example:
  - A TCP session, which is synonymous to a TCP virtual circuit, a TCP connection, or an established TCP socket.

In the case of transport protocols that do not implement a formal session layer (e.g., UDP) or where sessions at the session layer are generally very short-lived (e.g., HTTP), sessions are maintained by a higher level program using a method defined in the data being exchanged. For example, an HTTP exchange between a browser and a remote host may include an HTTP cookie which identifies state, such as a unique session ID, information about the user's preferences or authorization level.

Protocol version HTTP/1.1 makes it possible to reuse the same TCP session for a sequence of service requests and responses (a sequence of file transfers) in view to reduce the session establishment time, while HTTP/1.0 only allows a single request and response during one TCP session. However, this transport layer session mechanism should not be confused with a so-called HTTP session, since it does not last a sufficiently long time, and does not provide application level interactive services such as dynamic web pages.

## Software implementation

TCP sessions are typically implemented in software using child processes and/or multithreading, where a new process or thread is created when the computer establishes or joins a session. HTTP sessions are typically not implemented using one thread per session, but by means of a database with information about the state of each session. The advantage with multiple processes or threads is relaxed complexity of the software, since each thread is an instance with its own history and encapsulated variables. The disadvantage is large overhead in terms of system resources, and that the session may be interrupted if the system is restarted.

When a client may connect to any in a cluster of servers, a special problem is encountered in maintaining consistency when the servers must maintain session state. The client must either be directed to the same server for the duration of the session, or the servers must transmit server-side session information via a shared file system or database. Otherwise, the client may reconnect to a different server than the one it started the session with, which will cause problems when the new server does not have access to the stored state of the old one...

## Server side web sessions

Server-side sessions are handy and efficient, but can become difficult to handle in conjunction with load-balancing/high-availability systems and are not usable at all in embedded systems with no storage. The load-balancing problem can be solved by using shared storage or by applying forced peering between each client and a single server in the cluster, although this can compromise system efficiency and load distribution.

A method of using server-side sessions in systems without mass-storage is to reserve a portion of RAM for storage of session data. This method is applicable for servers with a limited number of clients (e.g. router or access point with infrequent or disallowed access to more than one client at a time).

In the two scenarios above, using client-side sessions could provide advantages over server-side sessions: in the first case by removing the limitations applied to load-balancing algorithms (which usually translates to load distribution optimisation), and in the second case by allowing the use of sessions in web applications when server disk space or RAM is not available or sufficient for this storage.

## Client side web sessions

Client-side sessions use cookies and cryptographic techniques to maintain state without storing as much data on the server. When presenting a dynamic web page, the server sends the current state data to the client (web browser) in the form of a cookie. The client saves the cookie in memory or on disk. With each successive request, the client sends the cookie back to the server, and the server uses the data to "remember" the state of the application for that specific client and generate an appropriate response.

This mechanism may work well in some contexts; however, data stored on the client is vulnerable to tampering by the user or by software that has access to the client computer. To use client-side sessions where confidentiality and integrity are required, the following must be guaranteed:

1. Confidentiality: Nothing apart from the server should be able to interpret session data.
2. Data integrity: Nothing apart from the server should manipulate session data (accidentally or maliciously).
3. Authenticity: Nothing apart from the server should be able to initiate valid sessions.

To accomplish this, the server needs to encrypt the session data before sending it to the client, and modification of such information by any other party should be prevented via cryptographic means.

Transmitting state back and forth with every request is only practical when the size of the cookie is small. In essence, client-side sessions trade server disk space for the extra bandwidth that each web request will require. Moreover, web browsers limit the number and size of cookies that may be stored by a web site. To improve efficiency and allow for more session data, the server may compress the data before creating the cookie, decompressing it later when the cookie is returned by the client.

## HTTP session token

A session token is a unique identifier that is generated and sent from a server to a client to identify the current interaction session. The client usually stores and sends the token as an HTTP cookie and/or sends it as a parameter in GET or POST queries. The reason to use session tokens is that the client only has to handle the identifier—all session data is stored on the server (usually in a database, to which the client does not have direct access) linked to that identifier. Examples of the names that some programming languages use when naming their HTTP cookie include JSESSIONID (JSP), PHPSESSID (PHP), and ASPSESSIONID (ASP).

## See also

- Login session
- Session management
- Session fixation
- Session poisoning

## External links

- Understanding Sessions in PHP [1]
- Session tracking methods [2]
- Sessions by Doug Lea [3]

## References

[1] http://www.talkphp.com/showthread.php?t=1077
[2] http://javapapers.com/servlet-interview-questions/explain-the-methods-used-for-session-tracking/
[3] http://g.oswego.edu/dl/pats/session.html

# Base Class Library

The **Base Class Library** (**BCL**) is a standard library available to all languages using the .NET Framework. .NET includes the BCL in order to encapsulate a large number of common functions, such as file reading and writing, graphic rendering, database interaction, and XML document manipulation, which makes the programmer's job easier. It is much larger in scope than standard libraries for most other languages, including C++, and is comparable in scope to the standard libraries of Java. The BCL is sometimes incorrectly referred to as the Framework Class Library (FCL), which is a superset including the Microsoft.* namespaces.

The BCL is updated with each version of the .NET Framework.

## Namespaces

Some of the namespaces may or may not be officially considered part of the BCL by Microsoft, but all are included as part of the libraries that are provided with Microsoft's implementation of the .NET Framework.

### Standardized namespaces

These are the namespaces that are standardized as of the ECMA 335 and ISO/IEC 23271:2006 standards.[1] [2]

System

This namespace includes the core needs for programming. It includes base types like String, DateTime, Boolean, and so forth, support for environments such as the console, math functions, and base classes for attributes, exceptions, and arrays.

System.Collections

Defines many common containers or collections used in programming, such as lists, queues, stacks, hashtables, and dictionaries. It includes support for generics.[3]

System.Diagnostics

Provides the ability to diagnose applications. It includes event logging, performance counters, tracing, and interaction with system processes.[4]

System.Globalization

Provides help for writing internationalized applications. "Culture-related information, including the language, the country/region, the calendars in use, [and] the format patterns for dates, currency, and numbers" can be defined.[5]

System.IO

Enables reading from and writing to different streams, such as files or other data streams. Also provides a connection to the file system.[6]

System.Net

Provides an interface "for many of the protocols used on networks today",[7] such as HTTP, FTP, and SMTP. Secure communication is supported by protocols such as SSL.

System.Reflection

Provides an object view of types, methods, and fields; and "the ability to dynamically create and invoke types".[8] It exposes the API to access the Reflective programming capabilities of CLR.

System.Runtime

Allows management of the runtime behavior of an application or the CLR. Some of the included abilities are interoperable with with COM or other native code, writing distributed applications, and serializing objects into binary or SOAP.[9]

System.Security

"Provides the underlying structure of the common language runtime security system."[10] This namespace allows security to be built into applications based on policy and permissions. It provides services such as cryptography.

System.Text

Supports various encodings, regular expressions, and a more efficient mechanism for manipulating strings (StringBuilder).[11]

System.Threading

Helps facilitate multithreaded programming. It allows the synchronizing of "thread activities and access to data" and provides "a pool of system-supplied threads."[12]

System.Xml

"Provides standards-based support for processing XML,"[13] including reading, writing, schemas, serialization, searching, and transforming.

## Non standardized namespaces

These are the namespaces that are not standardized as of the ECMA and/or ISO standards, and are specific to Microsoft implementation. However, even if implementing them is not mandatory, some of them have been implemented completely or partially by other .NET implementations.

System.CodeDom

This library provides the ability to create code and run it, at runtime.[14]

System.ComponentModel

Provides the ability to implement the run-time and design-time behavior of components and controls. It contains the infrastructure "for implementing attributes and type converters, binding to data sources, and licensing components."[15]

System.Configuration

Provides the infrastructure for handling configuration data.[16]

System.Data

This namespace represents the ADO.NET architecture, which is a set of computer software components that can be used by programmers to access data and data services.[17]

System.Deployment

Allows customization of the way applications upgrade when using ClickOnce.[18] [19]

System.DirectoryServices

Provides easy access to Active Directory from managed code.[20]

System.Drawing

Provides access to GDI+ graphics functionality, including support for 2D and vector graphics, imaging, printing, and text services.[21]

System.EnterpriseServices

"Provides .NET objects with access to COM+ services making the .NET Framework objects more practical for enterprise applications."[22]

System.Linq

Defines the IQueryable<T> interface and related methods, that lets LINQ providers to be plugged in.[23]

System.Linq.Expressions

Allows Delegates and Lambda expressions to be represented as expression trees, so that the high-level code can be examined and processed at runtime.[24]

System.Management

Allows querying for system information, "such as how much free space is left on the disk, what is the current CPU utilization, which database a certain application is connected to, and much more."[25]

System.Media

Provides the ability to play system sounds and .wav files.[26]

System.Messaging

Provides the ability "to connect to, monitor, and administer message queues on the network and send, receive, or peek messages."[27] .NET Remoting is another name for some of the functionality provided. This namespace is being superseded by Windows Communication Foundation.

System.Resources

Allows management of resources in the application in order to internationalize an application for different cultures and languages.[28]

System.ServiceProcess

Allows the creation of applications that run as a service within Windows.[29]

System.Timers

"Allows you to raise an event on a specified interval."[30]

System.Transactions

Provides support for local or distributed transactions.[31]

System.Web

Provides various web related functionality. It enables browser-server communication and the creating XML Web services. Most or all of these libraries are referred to as the ASP.NET architecture.[32]

System.Windows.Forms

This namespace contains the Windows Forms architecture which provides access to the native Windows interface elements by wrapping the existing Windows API. This allows for writing graphical applications for Windows from within managed code.[33] This system is being superseded by the Windows Presentation Foundation.

## See also

- ADO.NET
- ASP.NET
- Windows Forms
- Java Class Library
- Standard library

## Other capabilities of the .NET framework

- Windows Presentation Foundation
- Windows Communication Foundation
- Windows Workflow Foundation
- Windows CardSpace

## References

[1] "ECMA C# and Common Language Infrastructure Standards" (http://msdn.microsoft.com/en-us/netframework/aa569283.aspx). Microsoft. 2009. . Retrieved 2009-02-21.
[2] "Common Language Infrastructure (CLI)" (http://www.ecma-international.org/publications/standards/Ecma-335.htm). Ecma International. June 2006. . Retrieved 2009-02-21.
[3] MSDN Documentation - System.Collections Namespace (http://msdn.microsoft.com/en-us/library/system.collections.aspx)
[4] MSDN Documentation - System.Diagnostics Namespace (http://msdn.microsoft.com/en-us/library/system.diagnostics.aspx)
[5] MSDN Documentation - System.Globalization Namespace (http://msdn.microsoft.com/en-us/library/system.globalization.aspx)
[6] MSDN Documentation - System.IO Namespace (http://msdn.microsoft.com/en-us/library/system.io.aspx)
[7] MSDN Documentation - System.Net Namespace (http://msdn.microsoft.com/en-us/library/system.net.aspx)
[8] MSDN Documentation - System.Reflection Namespace (http://msdn.microsoft.com/en-us/library/system.reflection.aspx)
[9] MSDN Documentation - System.Runtime Namespace (http://msdn.microsoft.com/en-us/library/system.runtime.aspx)
[10] MSDN Documentation - System.Security Namespace (http://msdn.microsoft.com/en-us/library/system.security.aspx)
[11] MSDN Documentation - System.Text Namespace (http://msdn.microsoft.com/en-us/library/system.text.aspx)
[12] MSDN Documentation - System.Threading Namespace (http://msdn.microsoft.com/en-us/library/system.threading.aspx)
[13] MSDN Documentation - System.Xml Namespace (http://msdn.microsoft.com/en-us/library/system.xml.aspx)
[14] MSDN Documentation - System.CodeDom Namespace (http://msdn.microsoft.com/en-us/library/system.codedom.aspx)
[15] MSDN Documentation - System.ComponentModel Namespace (http://msdn.microsoft.com/en-us/library/system.componentmodel.aspx)
[16] MSDN Documentation - System.Configuration Namespace (http://msdn.microsoft.com/en-us/library/system.configuration.aspx)
[17] MSDN Documentation - System.Data Namespace (http://msdn.microsoft.com/en-us/library/system.data.aspx)
[18] MSDN Documentation - System.Deployment.Application Namespace (http://msdn.microsoft.com/en-us/library/system.deployment.application.aspx)
[19] MSDN Documentation - System.Deployment.Internal Namespace (http://msdn.microsoft.com/en-us/library/system.deployment.internal.aspx)
[20] MSDN Documentation - System.DirectoryServices Namespace (http://msdn.microsoft.com/en-us/library/system.directoryservices.aspx)
[21] MSDN Documentation - System.Drawing Namespace (http://msdn.microsoft.com/en-us/library/system.drawing.aspx)
[22] MSDN Documentation - System.EnterpriseServices Namespace (http://msdn.microsoft.com/en-us/library/system.enterpriseservices.aspx)
[23] MSDN Documentation - System.Linq Namespace (http://msdn.microsoft.com/en-us/library/system.linq.aspx)
[24] MSDN Documentation - System.Linq.Expressions Namespace (http://msdn.microsoft.com/en-us/library/system.linq.expressions.aspx)
[25] MSDN Documentation - System.Management Namespace (http://msdn.microsoft.com/en-us/library/system.management.aspx)
[26] MSDN Documentation - System.Media Namespace (http://msdn.microsoft.com/en-us/library/system.media.aspx)
[27] MSDN Documentation - System.Messaging Namespace (http://msdn.microsoft.com/en-us/library/system.messaging.aspx)
[28] MSDN Documentation - System.Resources Namespace (http://msdn.microsoft.com/en-us/library/system.resources.aspx)
[29] MSDN Documentation - System.ServiceProcess Namespace (http://msdn.microsoft.com/en-us/library/system.serviceprocess.aspx)
[30] MSDN Documentation - System.Timers Namespace (http://msdn.microsoft.com/en-us/library/system.timers.aspx)
[31] MSDN Documentation - System.Transactions Namespace (http://msdn.microsoft.com/en-us/library/system.transactions.aspx)
[32] MSDN Documentation - System.Web Namespace (http://msdn.microsoft.com/en-us/library/system.web.aspx)
[33] MSDN Documentation - System.Windows.Forms Namespace (http://msdn.microsoft.com/en-us/library/system.windows.forms.aspx)

## External links

- .NET Framework Developer Center (http://msdn.microsoft.com/netframework/) Microsoft Developer Network (MSDN).
- Base Class Libraries Community (http://msdn.microsoft.com/en-us/netframework/aa569603.aspx) More information and FAQs about the BCL.
- .NET Framework 3.5 namespaces (http://download.microsoft.com/download/4/a/3/4a3c7c55-84ab-4588-84a4-f96424a7d82d/NET35_Namespaces_Poster_LORES.pdf)
- MSDN BCL Documentation (http://msdn.microsoft.com/en-us/library/aa388745.aspx)
- BCLTeam's WebLog (http://blogs.msdn.com/bclteam/)

# Article Sources and Contributors

**ASP.NET** *Source*: http://en.wikipedia.org/w/index.php?oldid=405030622 *Contributors*: 28bytes, 31stCenturyMatt, ABF, ASK, Aabbramkumaraabb, Abhinav777, Ahoerstemeier, Akhristov, Ale jrb, Alexf, Alexius08, Alfie66, Ammar shaker, Ancheta Wis, Anclation, AnimalFriend, Anishmbait, Anna Lincoln, Anonymous anonymous, Another-anomaly, Aranel, Archer3, AreJay, ArtVandelay13, Artw, Asad1981, Aspandphp, Atraxani, Audiohifi, Avicennasis, Avijit6399, Az1568, BRSQUIRRL, BWCNY, BankingBum, Bennelliott, Benrick, Bevo, Bhopkins, Bill, Bill.albing, Blowdart, Bluemoose, Booles, Borgx, Bovlb, Bristol, Brock491, Btx40, Bunnyhop11, Butros, CWenger, Cactus.man, CaliforniaAliBaba, Calvinit21, Cam8001, Can't sleep, clown will eat me, CaptainCat, CatherineMunro, Cburnett, Centrx, Cerowyn, Chmod007, Chowells, Christopher G Lewis, Chriswiki, Clayoquot, Cmbay, Cmehtarahul, Cnash11, Coldfire82, Composeme, Connelly, Cremepuff222, CrizCraig, Cryptoid, DARTH SIDIOUS 2, DRogers, Dah Cn, Danakil, Dankstick, Darknode, Davidshq, Deiaemeth, Dekay46, Deon, DerHexer, Deryck Chan, Developer38, DigitalEnthusiast, DizzyITTech, Dockurt2k, Dreadstar, ED-tech, Earle Martin, ElationAviation, Ellenaz, Emurphy42, Eng.ahmedm, Enjarcher, Enric Naval, Epbr123, Excirial, FayssalF, Fiftyquid, Fishnet37222, Frankenpuppy, Frap, Fred Bradstadt, Frencheigh, FritzLaurel, Frosted14, Funkendub, Furrykef, Fæ, GDonato, George Leung, Gfoley4, Gholson, Giraffedata, Goa103, GraemeL, GreedyCapitalist, Guanaco, Gurch, Gurchzilla, Halex56, Harri J. Talvitie, Hede2000, Heinzi.at, Heirpixel, Here, Hu12, I already forgot, ILikeThings, Iamion, Ian Bailey, Ian Dalziel, Iggymwangi, Igoro1975, Imroy, Irishguy, IstvanWolf, It Is Me Here, Ixfd64, J.delanoy, JLaTondre, Jamierumbelow, Japonicus, Jaq316, Jaraics, Javiguillen, Jazbeat, Jazzycat, Jeff G., Jeffreymcmanus, Jeremy T. Fuller, Jhenry2075, Jhertel, Jigarrpatel, Jim182, Jimmytharpe, Jjmontalbo, John Vandenberg, Jonik, Joshua Issac, Jsalcedo, Julesd, Jutiphan, Jzucchetto, K. Annoyomous, Kaliramanjs, Kayau, Kenyon, Kibbled bits, Kmng, Kovyrin, Kseg, Kseguin, Ktpenrose, Kukini, Kypr8, LOL, Larrymcp, Lear's Fool, LeaveSleaves, Leeshm, Lightmouse, Livitup, Lofote, Luca001, Luke C, Lukes123, MC10, MER-C, Mandar1, Mandarax, Manishearth, Marek69, Marius, Martarius, Matithyahu, MattJubb, MattTM, Matthew Yeager, Matthew.townsend, MattieTK, McSly, Mebden, Meiskam, Michielvo, MightyWarrior, Mike Payne, Mike Schwartz, Mild Bill Hiccup, Minghong, Mitch Ames, Mlaffs, Mohamed A Meligy, Moreati, Mortense, Muntuwandi, NawlinWiki, Nburden, NeilFraser, NickBush24, Nigelj, Ninly, Nivix, NoTime, Novasource, Nuggetboy, OVERM1ND, Oblivious, Ochib, Ohnoitsjamie, Okanoksak, Onastic, Ondrejsv, OwenBlacker, Oxymoron83, PatrikR, Pedant17, PerLundberg, Pharos, Phgao, Pmerson, Pne, Ppalani, Prabhu2mail, QuWiki, Ramir, Random contributor, Ranganh, Rasmus Faber, Ravibodake, Razorflame, Rchandra, RedWolf, Rettetast, RexNL, Rhobite, Richi, Rickyp, Riki, Rjoshi s, Rjwilmsi, Robert-Parmelee, Roboblob, Rodasmith, Rodrigostrauss, Ronz, Root4(one), Rotem.E, Rror, Rsharp 59, Rupali1000, S.Örvarr.S, Sagaciousuk, Sahtesh, Sander Säde, Sanjaykunjam, Scott In SD, Seek1, Serlin, Shubhranshu, Sietse Snel, Sleepwiththefishes, Sloman, Snramkumar, SonicAD, Soumyasch, Squids and Chips, Ssangwan, Stormie, Strangnet, Strebe, Studchirag, Sturm55, Sujathalavi, SummerWithMorons, Sundippankaj, Sunridin, Suprotim, Suraj.mehare, Susurrus, Svick, Tagus, Taka, Tcncv, TechEditor2010, Technoverma, Tedickey, The Rambling Man, TheGuern, Tibti, Timdew, Timneu22, Tlusťa, Tomtheeditor, Tonydent, Tonyfaull, Toussaint, TravisTX, Trevor MacInnis, Tufflaw, Turnstep, Typofixer76, Useerup, VaneWimsey, Versus22, Vgribok, Visor, W3bbo, Waggers, Warren, WayneMokane, Weregerbil, Whomp, Winchelsea, Wknight94, Woohookitty, XXpress, Xpclient, Xtreme.msen, Y9s4gee, Yacht, Yaronf, Ytram99, Yug1rt, Ziya Suzen, Zzuuzz, 977 anonymous edits

**Web application framework** *Source*: http://en.wikipedia.org/w/index.php?oldid=405021134 *Contributors*: 4th-otaku, Aashu.dwivedi, Abab99, Abukaspar, Amr.eladawy, AndrewAllen, Atgreen, Axlq, Biggus jimus, Blonkm, Bluecarbon, Bob.firth, Brian Fenton, ChaoticXSinZ, Christopher Mahan, Cleared as filed, CloD, CortlandKlein, CosineKitty, Ctoofwiki, Dandv, Dianna m, Diletante, DojoWarrior, Edburns, Edoe, Ekspiulo, Ercillatechnologies, Fiftyquid, Florian Sening, Generalcomplex386, Gilliam, Gioto, GregChant, Haakon, Hairi, Hanenkamp, Henricchen, Hirsulhim, Ian Bailey, Ido50, Intgr, Isaacdealey, IslandHopper973, JCLately, JHP, JJC1138, JLaTondre, Jasondburkert, JasperRB, Jleedev, Jlundell, JoeCotellese, Josephgrossberg, Jpmagalhaes, JuJube, Juliehamwood, Karl Dickman, Kozuch, Leafman, LiDaobing, LittleBenW, Lkcl, Lohith.mk, MER-C, Martijn faassen, Massimodipierro, Matt Schwartz, Mbeck, Meryn, Mfedyk, Mivsek, Mjancuska, Moreati, Mwjmurphy, NathanBeach, Nchauvat, Nusoftware, Ojw, Oli Filth, Opolat, Peak, PhilKnight, Qtgeo, RHaworth, Robert K S, Roikonen, Saheek213, Shappy, Slark, Sohdubom, Sonjaaa, Stevietheman, Sumeet.chhetri, Tedickey, The Thing That Should Not Be, Tiago simoes, Tom W.M., Tonymarston, Tonyshan, Tylerl, UmassThrower, V. berus, Vanished user 04, Vl'hurg, Vrenator, Webmistress25, Wesley, Wongm, Wwwwolf, Zimbabwer, 230 anonymous edits

**.NET Framework** *Source*: http://en.wikipedia.org/w/index.php?oldid=405497338 *Contributors*: 10metreh, 1exec1, A Quest For Knowledge, AKeen, Abdulsami20, Abhishek2617, Abrech, Academic Challenger, Acc3ss, Adamralph, Adrian, Adrian Sampson, Adrianwn, AdunaicLayman, Aecy, Ahoerstemeier, Akanemoto, Akhristov, Akshaykumardwivedi, Alansohn, Aldenrw, Alex.muller, AlexOvShaolin, Alexf, Alexignatiou, Alfio, Alg8662, AlistairMcMillan, Alkarex, Allstarecho, Almogo, Alnokta, Altacus, Altenmann, Amire80, Amplitude101, Anaraug, Ancheta Wis, Andrewtechhelp, Anil235, Anna Lincoln, Annaellis, Anon lynx, Anthall1991, Apoltix, ArglebargleIV, Arjunsridharur, Arnon Chaffin, Arthena, Asa Winstanley, Askari Mark, Atlant, Audriusa, B3virq3b, BMT, BWCNY, Bahder, Basil.bourque, Behun, Bento00, Bevo, Bgpaulus, Bijoysr, Billy9999, Bizzako, BjarkeDahlEbert, Blakkandekka, Bluemoose, Bo98, Bobblewik, Bobo192, Booler80, Borgdylan, Borgx, Brandon, Brian Tvedt, Brianreading, Brijen911, Bruce89, Bsadowski1, BurntSky, Buxley Hall, CSharp-Online.NET Editor, CWenger, Cagliost, Calmer Waters, Caltas, Calton, Calwiki, CambridgeBayWeather, Campoftheamericas, Camw, Can't sleep, clown will eat me, CapitalSasha, Capricorn42, Carlo.milanesi, CecilWard, Ceyockey, Cflm001, ChangChienFu, Chanting Fox, Cheungpat, Chininazu12, Chip Zero, Chitomcgee, Chris Burrows, Christopher G Lewis, ChristopherM, Christopherlin, CiudadanoGlobal, ClinkingDog, Cnu2k5, CodeCaster, Codepro, Cody-7, Coldfire82, Compie, Computerjoe, ConCompS, ConfusedSushi, Conversion script, Corti, Craig Bolon, Cresvi, CritterNYC, Crpietschmann, Cryoboy, Cst17, Ctrautman1, Cwolfsheep, CyberSkull, Cybercobra, Cyberjessy, Cynical, Cyril, DARTH SIDIOUS 2, DStoykov, Da monster under your bed, Damian Yerrick, Dan Atkinson, Dan.tsafrir, Darin, Daverocks, DavidBlackwell, Dbnull, Ddon, Deh, Demono, DenisBach, DerHexer, Destructionator, Diberri, Digvijaypundir1, DiiCinta, Dina, Discospinster, Dispenser, Djg2006, Dlunch, Dmb, DonToto, Donama, Dondemuth, Donsez, DopefishJustin, Doug Bell, Dratman, Drilnoth, Drjonesgp, Drysart, Dtobias, Dvanderboom, Dwo, Dycedarg, Dylanfromthenorth, Dysprosia, EJSawyer, EagleOne, Eamon914, Eazulay, Ed Brey, EdJogg, Edaelon, ElTyrant, Elcobbola, Eldenf, Elfguy, Ellmist, Eluchil404, Elykyllek, Enchanter, Engfelt, Epbr123, Eric Burnett, Erik Swanson, Erpe, EscapeTechwriter, Escla, Eviktor, Evilandi, Evildeathmath, Ewlyahoocom, FatalError, Fayssal FERTAKH, FayssalF, Fduch, Feedmecereal, Felyza, Fibonacci, Findling67, Flashmob, FleetCommand, Fleminra, Flewis, Flib0, Foofy, Frank Hileman, Frecklefoot, Fred Bradstadt, Furrykef, Futurix, GFHandel, Gadfium, Gaijin42, Galestar, Gary King, Garzo, Gbjbaanb, Gbleem, GeorgeStepanek, Gggh, Gilliam, Gingko, Glane23, Gnj, GoddersUK, Googl, Gracefool, Gracenotes, Graham87, Granburguesa, Greenleaf, Gwernol, Hadal, Haeleth, Halaelkholy, Halestock, Hamtechperson, Harborsparrow, Harriv, Harryboyles, Hasruf, Heath007, Helpsloose, Hephaestos, Heron, Hervegirod, Himanet, Hippy deluxe, Hirzel, HorseShoe, Hu12, Hut 8.5, Hyad, Hydrogen Iodide, I dream of horses, IByte, IRP, Ian Dalziel, Ideaclimber, Idolescent, Iggymwangi, Ijustam, Ike-bana, Illyria05, Ilyaroz, Imasleepviking, Immunize, Imnotminkus, Inquisitus, Intersofia, InvaderJim42, Ioakar, Ipapasa, Iridescent, J Di, J M Rice, J.delanoy, JAn Dudík, JLaTondre, Jacek Kendysz, Jack's Revenge, Jacques Bergeron, Jake-helliwell, James Michael 1, JamesNK, Jamesontai, Jamestochter, Janadore, Jay, Jayrvox, Jaysmith2108, Jcarroll, Jclemens, Jdog78889, Jehjoa, Jeltz, Jeremy Visser, Jerryobject, Jesdisciple, JesseHogan, Jgyanendra, Jic, Jim McKeeth, Jimbluedog, Jimmyzimms, Jjegers, Joakim Ziegler, JohnOwens, Johndci, Johnuniq, Jonmmorgan, Jonnyapple, Josh the Nerd, Jsnover, Jtalledo, Judik, Julesd, JunkDNA, JustSomeKid, Justpossible, Jutiphan, KAMiKAZOW, KJK::Hyperion, Kai, Karpouzi, Kartano, KartoumHero, Kasutt, Kathryn NicDhàna, Kcargile, Kellen`, Kenb215, Kephart, Kerotan, Kesla, Ketiltrout, KevM, Khishig.s, Killiondude, Kocio, Konstantin Veretennicov, Kovyrin, Kozuch, Kprobst, Kr-val, Kraenar, Kumioko, Kuru, Kutulu, Kvdveer, Kypr8, Lashiec, Laudaka, LauriO, Leaflord, Legotech, Leirith, Leotohell, Leotohill, LibLord, Lightmouse, Ligulem, Linkspamremover, LittleDan, Liujiang, Localzuk, Longhair, Lovely Chris, Lradrama, Lyml, MER-C, Mac, Magioladitis, MagnaMopus, Magnus.de, Mahanga, Mailtojp, Malekith, Manish014, Manpal78, ManuelGR, Marc Kupper, Marc44, Marek69, Marius, Mark5677, Markcrosbie, Markhurd, Marknew, MartinRinehart, Martyn Lovell, Masonwheeler, Masterhomer, Mathiastck, Matt Crypto, Maury Markowitz, Mav, Maw136, MaxSem, Mbjencyclopedia, McSly, Mckaysalisbury, Mcmatter, McoreD, Mdupont, Mebden, Member, MementoVivere, Memodude, Mendaliv, Mfb52, Mfloryan, Mgiann, Michael Hodgson, Michael R Bax, Michelleem, Miguel.de.Icaza, Mike.rooke, Millahnna, Minesweeper, Minghong, Minooch, Mintleaf, Mitch Ames, Mitchello, Mitchumch, Mlk, Mmmready, Mmortal03, Mnikoo, Modster, Mormegil, Morte, Mortense, MrBlueSky, MrJones, MrOllie, Mrcarter8, Mrh30, Myanw, Naerii, Natalie Erin, Nblumhardt, Ndenison, NeilSantos, Neilc, Nevalex, Nevilley, Ngyikp, Nikiforov, Nimrand, Nistra, Nixdorf, Nk, Nma wiki, Nn123645, Noldoaran, Noosentaal, Northgrove, Notinasnaid, Noypi380, Nsaa, Nshuks7, Nuggetboy, Numberp, Nysin, ObitheWan, OlivierNallet, Ondrejsv, Onebit, Ongar the World-Weary, Op12, Optakeover, Optikos, Orderud, Ortzinator, OwenBlacker, Oxymoron83, PCrew00, PDH, PL290, PatrikR, Paul Magnussen, Pelister, Perfecto, Petermarcu, Petrb, PetrochemicalPete, Petrtom, Philip Trueman, PhilipMay, Philmatic, Piano non troppo, Plasticup, Plugwash, PonThePony, Porchcrop, Porqin, Ppntori, Pps, Proofreader77, Puhfyn, Quux, Qxz, RAM, RJSampson, RainbowOfLight, Raja99, RamirBorja, Ramu50, Random user 39849958, Rao mocharla, Rapidshare, Ratarsed, Ravichandar84, Rchandra, Reach Out to the Truth, RedWolf, Reedy, Remember the dot, Remon sherin, RexNL, Rexmorgan, Rgsiii, Rich Farmbrough, Richard Allen, Richard.szalay, Richardsonke, RickK, Rickz0rz, Riki, Rjwilmsi, Rlee0001, RobbieAB, Robert.maclean, Robertbrichter, Robogun, Rodasmith, Ronark, Ronz, Rsatter, RtpHarry, RussPitcher, Rwalker, Ryan Norton, S.Örvarr.S, SMC, SNIyer12, ST47, SVG, Saiswa, Saksham, Samrolken, Santiago Roza (Kq), SchuminWeb, Scohoust, Sdaconsulting, Sean Whitton, Sean.woodward, SebastianHelm, Sega381, Shadowjams, Shalvin P D, Shinji14, Shinsuk, Shirulashem, Shizhao, Sietse Snel, Silvergoat, Sitethief, SixSix, Skarebo, Skunkboy74, SkyWalker, Skyfiler, Slakr, Small island, Smitty, Snailwalker, Snoutholder, Snowolf, Some Wiki Editor, Somercet, Soumyasch, SpaceFlight89, SpeedyGonsales, Staka, StaticVision, Stephenb, Stephenchou0722, Stimpy77, Storm Rider, Stormie, Stwalkerster, Sundström, Superm401, Superrcat, Surachit, Suruena, Susurrus, Svick, Sycthos, Sydius, TEB728, Tagus, Tbrill5, Tbsdy lives, Teacurran, Techeeee, Terence, Tfl, The Anome, The Thing That Should Not Be, The locster, The1physicist, TheFattest, TheRasIsBack, Theking9794, Themfromspace, Thijs!, TiagoDias, Tide rolls, Tiger Khan, Tim1357, Timc, Timwi, Tobias Bergemann, Tobiasly, Tomabuct, Tommy2010, Tomy9510, Torc2, Toussaint, Triggersite, Triona, Triwbe, Tvarnoe, Tyomitch, Typhoonhurricane, Uncle G, Uncle.bungle, UncleDouggie, Unfree, Uzume, Vamsankar, Venkat 9009, Versus22, Viajero, Vicky4ud, Vikasgoyal77, Vinh1000, Vinitfichadia, Voidxor, Volvov70, Walter, Warren, Watson Ladd, Wei.cs, Werdna, Wereon, Widefox, WikHead, Wiki fanatic, Wikimsd, Wikipedia brown, Willardjuice, William Avery, Winterst, Wizzardmr42, Wjw, Wmahan, WulfTheSaxon, Wwwwolf, XDanielx, XMog, XP1, Xcentaur, Xpclient, Xtv, Yacht, Yoenit, Youssefsan, Ysangkok, Yurik, ZacBowling, Zachary, ZealousCoder, Zenohockey, Zhengjp, Zidler, Ziggurism, Zout, Zzuuzz, 1925 anonymous edits

**Active Server Pages** *Source*: http://en.wikipedia.org/w/index.php?oldid=405098712 *Contributors*: 12.21.224.xxx, 16x9, 1812ahill, 217.99.105.xxx, 2mcm, Access Denied, Achew22, Agateller, Ahoerstemeier, Airwot4, Alex.muller, AlexKarpman, Allo002, Alyssa3467, AndrewHowse, Angela, AnnaFrance, Ashitaka96, BCube, Bemoeial, Beno1000, Benrick, Bevo, Blahma, Bluemoose, Borgx, Brighterorange, Bsadowski1, BurntSky, CWii, Campustr, Capricorn42, Carbuncle, CatherineMunro, Chipp, Christian List, Christopher G Lewis, Claude.mercury, Conversion script, Craigoliver, Darksbane, Davepusey, Dfgriggs, Discospinster, Dockingman, Dotancohen, DraQue Star, Dreftymac, Dwo, Dze27, E0steven, ESkog, Eitheladar, ElfWarrior, Erkekjetter, EvanCarroll, Evoluzion, Fieldday-sunday, Fkhandy, Flockmeal, Frecklefoot, Fredrik, Ftiercel, Fvw, Gakusha, Greeneto, Greenleaf, Grenbert, Greyskinnedboy, Guanaco, Hairy Dude, Hetzer, Highclimber, Hu12, I already forgot, Irishguy, JCLately, JLaTondre, Jebba, Jeepday, Jeltz, Jkid, Johnleemk, Jose Icaza, Julesd, Justpossible, Kaare, Karada, Kmaster, Kooshal, Kuru, LFaraone, Largoplazo, Laura SIMMONS, Leafyplant, Lianmei, Liftarn, LilHelpa, Litany42, Locke Cole, Logixoul, M0tah, MER-C, MISIIM, Mac, Mad Scientist, Majid ahmadi, Meetfazan, MichaelBillington, Michal.Pohorelsky, Minghong, Mipadi, Moeron, Moreati, Mormat, Mpnolan, MrVibrating, Mram80, Mrwojo, Mschlindwein, Mz5starchick, Mütze, Nate Silva, Neilc, Nigelj, Nma wiki, Novasource, Omicronpersei8, Patricktride, Phantom784, Pharos, PhilKnight, Prickus, Quadra23, Quarl, R00m c, Rbonvall, RedWolf, Rhobite, RickK, Rising, Rje, Rjwilmsi, Roadkill69, Rob Cranfill, Rogerhc, Rokaszil, Root4(one), SDC, SadanYagci, Sander Säde, Scott In SD, Sealican, Shreevatsa, Sidhekin, SimonD, Sirlearnsalot, SlayerDork, Sleepyhead81, Smartweb, Snowboarder, Sopoforic, Spalding, Starnestommy, Stevertigo, Stevietheman, Szlam, TOReilly, TYelliot, Ta bu shi da yu, Timdew, Timo Honkasalo, Tomchiukc, Trusilver, Tyomitch, Tznkai, Ump2152, UnitedStatesian, Unyoyega, Utcursch, Vinhtantran, Winterst, Wknight94, Xezbeth, Xpclient, Yurik, Yyy, Zeus, Zfr, Zirnevis, Zondor, Zundark, Zzuuzz, Александр Сигачёв, Милан Јелисавчић, 431 anonymous edits

**Common Language Runtime** *Source*: http://en.wikipedia.org/w/index.php?oldid=404147475 *Contributors*: Abrahami, Ahy1, Akhristov, AlistairMcMillan, Alksub, Allstarecho, Ancheta Wis, Andreas Kaufmann, Andrevan, Attardi, Bhadani, Borgx, Brighterorange, Bryan Derksen, BurntSky, Caesura, Chuunen Baka, CodeMonk, Courcelles, Cwolfsheep, Cybercobra, Darguz Parsilvan, Dynaxis, Eng.ahmedm, Ettlz, Evyn, FayssalF, Frap, Furrykef, Get It, Gioto, Gskuse, Hirzel, Hu12, InvaderJim42, Jaavaaguru, Jensfiederer, Jsmethers, Jutiphan, Lastorset, Leotohill, M4gnum0n, MER-C, MacTed, Markhurd, MattieTK, Maviles, Michal Jurosz, Mike Rosoft, Moa3333, MrOllie, Myron.walker, Nilmerg, Onebit, Pelister, Phaedrus86, RedWolf, Rjwilmsi, Rmorrish, Rodasmith, Rursus, Saimhe, Saiyedistiyak, Scorpiuss, Sdrtirs, Skagedal, Soumyasch, Sundström, Svick, Tagus, TakuyaMurata, The Rambling Man, Tommy2010, Torc2, Toussaint, Troy 07, Uzume, Vina, VineetKumar, Warren, Yug1rt, Zegoma beach, 125 anonymous edits

**SOAP** *Source*: http://en.wikipedia.org/w/index.php?oldid=404960278 *Contributors*: 21655, Aaronbrick, Abhinav316, Access Denied, Acroterion, Adam J. Sporka, Aeolian, Alansohn, Aldie, Ale2006, Alejandro.imass, Aleksander Zawitkowski, Alex.muller, Algae, AliveFreeHappy, Andreas Kaufmann, Andrisi, Antonycarthy, Apparati, Average Earthman, Bachrach44, Bahaba, BeakerK44, Ben Babcock, Bezenek, Bijee, Bkonrad, Blanchardb, Blanghals, Blueboy96, Bluetulip, Bobcat43, Bobo192, Bongwarrior, Bovineone, BrEyes, Brainix, Brossow, Bveratudela, CIreland, Caltas, Calton, Can't sleep, clown will eat me, CanisRufus, Cartque, Christopher Mahan, Chuunen Baka, Cjolley101, Cliffhanger, Coasterlover1994, Conversion script, Coolhandscot, Cybercobra, DAud IcI, DKEdwards, DMacks, DSosnoski, Daeley, Daev, DanMS, Dangiankit, Daniel Olsen, Danlev, Darkimmortal, Darwinek, David Gerard, DavidABraun, Dawidl, Dawn Bard, Dekisugi, Derek, Desmoh, Destro, Diberri, Dockurt2k, Dpm64, Drappel, Dreftymac, Dtwong, Dtynan, Duk, Dysprosia, Eclectic star, Edjanx, Edmund.vonderburg, Ehn, Elf, Eliyahu S, Eltener, Enjoi4586, Epbr123, Ernstdehaan, Ethilien, Ezra Wax, FatalError, Femto, Ferrans, Fg, Flammifer, GaborLajos, Gamer007, Ghettoblaster, GoClick, Gogo Dodo, Gorpik, Grafen, Grshiplett, HalfShadow, Hannes Hirzel, Hayabusa future, Helix84, Herostratus, HiDrNick, Hu12, HughJorgan, IAMTHEORIGINALTED, Iamfool, Ianaf4you, Intelliproject, Iridescence, Ironwolf, Isaac Rabinovitch, J.delanoy, JForget, JamesBWatson, JamieS93, Jamiecampbell, Janom e, Japo, JaredWBurt, Javidjamae, Jenny MacKinnon, Jetekus, Jnc, Joakim Ziegler, John Vandenberg, JohnicholasHines, Joy, Jpaulm, Jsawiuk, Jusdafax, Jzhang2007, KSiimson, Kadavralol, Kaimiddleton, KarlPihlblad, Kbdank71, Kbrose, Kgaughan, Khalid hassani, Khym Chanur, Kinema, Kowey, Krellis, Kricxjo, Kristof vt, Kustere, Lasombra, Laurentdk01, LeaveSleaves, Leebo, Lengyeltom, Liassic, Liftarn, Lights, Lloy0076, Loadmaster, LodeRunner, MBisanz, MC10, Maebmij, Majkiw, Mange01, Manoj.aimca2, Manop, Marcuswittig, Mav, Mcmohd20, Menew222, Mephistophelian, Michael Hardy, Midnight Madness, Mindmatrix, Minghong, Mkosmul, Modster, Mooglemoogle, Mr link, Munahaf, MurrayRM, MyFavoriteMartin, Mzajac, Nahtanoj04, Nattelsker, Nessymonster, Netan'el, Netmonger, Netsnipe, Nigelj, Nixdorf, Nixeagle, Nohat, Nux, OliD, Pavel Vozenilek, Peterl, Phils, Piano non troppo, Prickus, Pseudomonas, Psiphiorg, Pyrofork, QuickBrownFox, RJaguar3, Rami R, Reach Out to the Truth, Rednblu, Reggaloza, Reinhardheydt, Ridvany, Rillian, RlyehRising, Roberto999, Robmanson, Robsinden, Rogerd, Ronz, RoySmith, Rrjanbiah, Rsankarg39, Ryan, Samisa.abeysinghe, Samuelsen, Satchy, Saturnight, Saucepan, Scharer, SchreyP, Schzmo, Scientus, Scope creep, Sderose, Seashorewiki, Securiger, SemiEvolved, Shawisland, Silver Spoon Sokpop, Simon80, Slartibartfast1992, Snanda85, Soaguy, Soap, Soumyasch, SpikeToronto, SpuriousQ, Stabbyboyo, Sterremix, Superandoni, Suruena, Svick, Syberguru, Sytone, TXiKi, Tagishsimon, Tbhotch, Tedickey, The Anome, Thumperward, Thv, Tide rolls, TimNelson, Todd Vierling, Tommy2010, Toussaint, Traroth, Triona, Truthandsanity, UninvitedCompany, Unixer, Vald, Veghead, Versus22, Vitz-RS, Warren, Waxy Protector, Waycool27, Wickethewok, WickieNcf, Wiesel 99, Wiki alf, Wildthing61476, Wj32, Wjburnett, Wrathchild, Wwwwolf, Xi3nt, Yaxh, Yongxinge, Yurik, Zearin, Zero sharp, Zmydlarz, Zserghei, 641 anonymous edits

**Server-side scripting** *Source*: http://en.wikipedia.org/w/index.php?oldid=405008301 *Contributors*: (:Julien:), 12.21.224.xxx, 217.35.153.xxx, AIOS, AdjustShift, Alphanis, Anuj999, Avochelm, Backpackadam, BadSeed, BigFatBuddha, Bonadea, Brainix, Burchard, BurntSky, Conversion script, DNewhall, DSRH, Da monster under your bed, DanMS, Danski14, Diberri, Dmerrill, Donovanbrooke, Dpm64, Ed Poor, ElTyrant, Greyskinnedboy, Guaka, Helicon, Hu12, IFlashie, INic, Jclemens, Josemiguel93, Josh Parris, Lee Daniel Crocker, Light current, LittleOldMe, Longhair, Mark T, Metagraph, Minghong, Mother Russia, Mrwojo, NurAzije, Ohnoitsjamie, Pgk, Phantombantam, Pharos, Plugwash, QVanillaQ, RHaworth, Robsomebody, Simul, Sketch051, SkyWalker, Skyrail, Socialery, Sonjaaa, Taka, Taw, Teles, Thejoshwolfe, Thingg, Tide rolls, Tokek, ViRaveNuS, WikHead, Wilfrednilsen, Windharp, Wixardy, Zenohockey, Zundark, 139 anonymous edits

**Active Scripting** *Source*: http://en.wikipedia.org/w/index.php?oldid=404418025 *Contributors*: 231O, Akadruid, AlistairMcMillan, Anphanax, CSWarren, Digita, El Cubano, FreeRangeFrog, Fritz Saalfeld, Ghettoblaster, Greyskinnedboy, Helpsloose, JLaTondre, John Vandenberg, Michael Bednarek, Modster, Philu, Reisio, RexNL, Ruud Koot, Sdfisher, Sikon, Socrates2008, Soumyasch, Toddintr, Torc2, Tyomitch, Wvnobody, Xpclient, 17 anonymous edits

**Dynamic web page** *Source*: http://en.wikipedia.org/w/index.php?oldid=405321970 *Contributors*: 16@r, Alansohn, AlastairIrvine, Alexius08, Alfrin, Anna Lincoln, Bearcat, Bonadea, Caiaffa, Cappu115, CesarB, Chasingsol, ChrisLoosley, Ciphers, Click99, Curious1i, DARTH SIDIOUS 2, Dave Runger, David Haslam, Digisus, DoriSmith, Elonka, Fmccown, Furrykef, Gkremen, Greg Tyler, Hbent, Herbythyme, Immunize, Josh Parris, Joshuagross, Kbrose, Kdakin, Kghose, Kku, KnowBuddy, Krauss, LapoLuchini, Laurenceskoster, Llywelyn, Lubino2000, Madalino, Malapati, Mann jess, Manop, Mboverload, Mets501, Michael Hardy, Muntuwandi, NYKevin, Niaz Ally, Nihiltres, NuclearWarfare, Orphan Wiki, Patrick, PhilKnight, Pxma, Q Canuck, Quilokos, Ronitw, Sasoriza, Scole01, Somerut, Tbonge, Timneu22, Topbanana, Tregoweth, Turnstep, TutterMouse, VMS Mosaic, Vadmium, Wangi, Wowdezign, Xibe, Zzuuzz, 184 anonymous edits

**Internet Information Services** *Source*: http://en.wikipedia.org/w/index.php?oldid=405289463 *Contributors*: 1pezguy, Abune, Acanon, Adam Zivner, Aegicen, Alansohn, AlexKarpman, AlistairMcMillan, Angela, Another-anomaly, AriPernick, ArmadilloFromHell, Avocado27, BJB, Baldwinmathew, BankingBum, BenWibking, Benc, Bewildebeast, Borgx, Bovineone, Bryan Derksen, Calton, Cernatcristi, Cgknowlton, Chadders, Chaos Llama, Christopher G Lewis, Cleared as filed, Courcelles, Crossland, Cybercobra, DKEdwards, Daeken, Damieng, DanielPharos, Davecox77, DeadlyAssassin, DigitalEnthusiast, Dirkbb, Disk Crasher, Drilnoth, Drmonocle, Dysii, EagleOne, Earthbike, Eddpayne, ElBenevolente, Ellipsis, Evert, Fabio.Firmo, Faisal.akeel, Ferdinand Vimes, Fjm2, Fleminra, Frap, Frenziedsloth, Fsmadi, GB fan, GSMR, Gaius Cornelius, Gladmax, Gogo Dodo, Gopy333, Group29, Guess Who, Gulliveig, Hede2000, Hetzer, J.delanoy, JLaTondre, Jeremy Visser, JeroenMassar, Jghaines, Jkl, Jvstein, Kaal, Killiondude, Klingoncowboy4, Kolobukhova, Kurowoofwoof111, KurtRaschke, Lerneaen Hydra, LilHelpa, Lofote, Logicwiki, LordKenTheGreat, Lowellian, Mahanga, Mandarax, Mark T, Martarius, Martin451, Max613, Mephiles602, Mineralè, Minghong, Misi91, Mmmready, Mmxx, Motherofinvention, MrOllie, Mrt doulaty, Mschlindwein, Msiebuhr, Mysidia, N1cholson, Neilstuartcraig, Neo139, Niteowlneils, Njan, Nopetro, Notbyworks, Noypi380, Octahedron80, Oculus Spectatoris, Offerman, OlEnglish, Ondrejsv, Oscar Bravo, PPGMD, Pankkake, Peasea, Phaldo, Phallonz, PhilKnight, Piano non troppo, Promethean, Psym, QofASpiewak, Quadra23, R00m c, R4u, Rayward, Razorx, Rcsheets, Rdubois IIS, Reach Out to the Truth, RedWolf, Rhetoricalwater, Riki, Rkelch, Robert, Robertwharvey, Ronark, Ronz, Rovastar, Rsrikanth05, SJP, SSTwinrova, Sabri76, SciCorrector, Sdgjake, Sega381, Simoncpu, Sl, Sloman, Socrates2008, Soumyasch, Specs112, Stewartadcock, SunCreator, Ta bu shi da yu, Tas50, Techmdrn, Thalter, The Anome, Tjdw, Toutoune25, Trjumpet, UU, Uzume, Veratien, Veraxus, Vgribok, Vivio Testarossa, VxP, WarFox, Warren, Wereon, Wernher, WideArc, Wiki Raja, Wikiolap, Wikitiddler, Wine Guy, Wnissen, Wordwizz, Wtmitchell, Wuser10, Xpclient, Yug1rt, ZeroOne, Zollerriia, Ὁ οἶστρος, 虞海, 325 anonymous edits

**Common Language Infrastructure** *Source*: http://en.wikipedia.org/w/index.php?oldid=393480055 *Contributors*: 0, 15.67, AWendt, Abrahami, Altenmann, Ancheta Wis, Anon lynx, Bevo, Brianski, BurntSky, CarrerCrytharis, Ceyockey, Cgs, Compie, Conversion script, DARTH SIDIOUS 2, David Gerard, Derbeth, FayssalF, Firien, Foofy, Fred Bradstadt, G0rn, Ghettoblaster, Gracefool, GreatWhiteNortherner, Gronky, Hannes Hirzel, Hervegirod, Hu12, Ioakar, Irbanjit, Jan Hidders, K.menin, Karl-Henner, Kbolino, KellyCoinGuy, Konstantin Veretennicov, Kwaku, Lasombra, Leotohill, LilHelpa, Melchoir, MrOllie, Onebit, Peter Winnberg, Pgan002, Phil Boswell, Plasticup, Puhfyn, Rangsynth, RedWolf, Rich Farmbrough, Rjwilmsi, Ronduck, Snaxe920, Soumyasch, Tagus, The Anome, Thejoshwolfe, Ti89TProgrammer, Toussaint, Tyomitch, Waeswaes, Warren, Yug1rt, Yworo, ZacBowling, Zundark, 73 anonymous edits

**Common Intermediate Language** *Source*: http://en.wikipedia.org/w/index.php?oldid=398949306 *Contributors*: Aranel, Bevo, Blairsh, Borgx, Bryan Derksen, Cetinsert, Chip Zero, Danakil, Eliz81, Enmlil Ninlilbb1979, FatalError, FayssalF, Foofy, Frecklefoot, Freefrag, Furrykef, Gurch, Habbit, Hu12, Icktoofay, Int19h, Jjalexand, Jon186, Karl-Henner, LilHelpa, Macmcrae, Mahanga, Michael B. Trausch, Michaelkvance, MrOllie, Onebit, Pgan002, Philip Trueman, Quamaretto, Radagast83, Rickyp, Ronark, S.Örvarr.S, Salagtas, Saros136, Shajan, Soumyasch, Sundström, Superjordo, The Thing That Should Not Be, Ti89TProgrammer, Tide rolls, TimBentley, Tizio, Tobias Bergemann, Tocharianne, Tonybuaa, Toussaint, TutterMouse, Tyomitch, UU, Ultramandk, VisualMelon, Wipe, ZacBowling, 85 anonymous edits

**Software framework** *Source*: http://en.wikipedia.org/w/index.php?oldid=403940967 *Contributors*: AMackenzie, Abdull, Absolutehegemony, AlistairMcMillan, Alteregoz, Amipierro, Andreas Kaufmann, Another Stickler, Avargasm, BZer0, Beetstra, Beland, Blakkandekka, Brettright, Ceceliadid, DRogers, Digomez, EoGuy, Ervinn, Evalowyn, Fabartus, Friendlydata, Gachet, Gary King, Georgefairbanks, Giles65, Gletiecq, Gridrebel, Ham Pastrami, Harborsparrow, Hede2000, Ian Bailey, Ikip, Intgr, Ivantodorov, JLaTondre, Jamelan, JesseHogan, Joriki, King of Hearts, Kjramesh, Kucing, LedgendGamer, Leibniz, Lionart, Maliebelt, Manop, Marudubshinki, Materialscientist, Matt Schwartz, Mdd, Minna Sora no Shita, Minority Report, Mirosamek, Mjb, Mokhov, MusicScience, Mx.std, Nailbiter, NeilSantos, Newtonv2, Nusoftware, Onesimplehuman, Only2sea, PL290, Pearle, PoisonedQuill, Prakash Nadkarni, Prolog, Quiliro, Radagast83, Radical-Dreamer, Randomalious, Raysonho, S.Örvarr.S, Salam32, Satya61229, Shanes, Spsoni, TFOWR, THEN WHO WAS PHONE?, Tonyshan, TreveX, TseiTsei, Wikidrone, Wikievil666, Winterst, Wjgilmore, Zundark, سعی, 134 anonymous edits

**Template (software engineering)** *Source*: http://en.wikipedia.org/w/index.php?oldid=405261628 *Contributors*: Andreas Kaufmann, BenAveling, Dreftymac, Egil, Fortdj33, Krauss, Kuru, Nick Number, Srikant.sharma, 3 anonymous edits

**Session (computer science)** *Source*: http://en.wikipedia.org/w/index.php?oldid=403368713 *Contributors*: 16@r, Agony, Appleseed, Beland, Blaufish, Bultro, Caltas, Danny Bierek, Dariusz Peczek, Download, Euchiasmus, Franquomètre, Furrykef, Gardar Rurak, Gsp8181, Gurch, Hede2000, Hu12, Infrangible, Jay, Jcv hdjur, JoaoRicardo, JonHarder, Jonathan de Boyne Pollard, Jsf, Kulandai, Kumardinesha, LOL, Lannm, Lights, LoveEncounterFlow, Maghnus, Mange01, Pete142, Phatom87, Pnm, Project2501a, Ptdecker, R'n'B, Radagast83, Ruud Koot, Ryan Roos, Skrapion, Stevietheman, Timneu22, Uncle G, ZooFari, ^demon, 54 anonymous edits

**Base Class Library** *Source*: http://en.wikipedia.org/w/index.php?oldid=404629893 *Contributors*: 16@r, Acdx, Ak1jain, AndrewWTaylor, Borgdylan, C777, Calwiki, Etphonehome, Fred Bradstadt, GabrielBurt, Gioto, Gugale, Hervegirod, MarcoLittel, Michael Hardy, Pizzachicken, Pomte, Romanc19s, Ronark, S.Örvarr.S, Soumyasch, Toussaint, Uzume, Yug1rt, 22 anonymous edits

# Image Sources, Licenses and Contributors

# License