**rview**

1. **ASP.NET**

**ASP.NET** is an [open source](http://en.wikipedia.org/wiki/Open_source)[server-side](http://en.wikipedia.org/wiki/Server-side_scripting) [Web application framework](http://en.wikipedia.org/wiki/Web_application_framework) designed for [Web development](http://en.wikipedia.org/wiki/Web_development) to produce [dynamic Web pages](http://en.wikipedia.org/wiki/Dynamic_Web_pages). It was developed by [Microsoft](http://en.wikipedia.org/wiki/Microsoft) to allow [programmers](http://en.wikipedia.org/wiki/Programmer) to build dynamic [web sites](http://en.wikipedia.org/wiki/Web_site), [web applications](http://en.wikipedia.org/wiki/Web_application) and [web services](http://en.wikipedia.org/wiki/Web_service).

It was first released in January 2002 with version 1.0 of the [.NET Framework](http://en.wikipedia.org/wiki/.NET_Framework), and is the successor to Microsoft's [Active Server Pages](http://en.wikipedia.org/wiki/Active_Server_Pages) (ASP) technology. ASP.NET is built on the [Common Language Runtime](http://en.wikipedia.org/wiki/Common_Language_Runtime) (CLR), allowing programmers to write ASP.NET code using any supported [.NET language](http://en.wikipedia.org/wiki/List_of_CLI_languages). The ASP.NET [SOAP](http://en.wikipedia.org/wiki/SOAP) extension framework allows ASP.NET components to process SOAP messages.

ASP.NET is in the process of being re-implemented as a modern and modular [web framework](http://en.wikipedia.org/wiki/Web_framework), together with other frameworks like [Entity Framework](http://en.wikipedia.org/wiki/Entity_Framework). The new framework will make use of the new open-source [.NET Compiler Platform](http://en.wikipedia.org/wiki/.NET_Compiler_Platform) (code-name "Roslyn") and be[cross platform](http://en.wikipedia.org/wiki/Cross_platform). [ASP.NET MVC](http://en.wikipedia.org/wiki/ASP.NET_MVC), ASP.NET Web API, and ASP.NET Web Pages (a platform using only [Razor](http://en.wikipedia.org/wiki/ASP.NET_Razor_view_engine) pages) will merge into a unified MVC 6. The project is called "ASP.NET vNext".

1. **Microsoft SQL Server**

Microsoft SQL Server is a relational database management system developed by Microsoft. As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet). There are at least a dozen different editions of Microsoft SQL Server aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users. Its primary query languages are T-SQL and ANSI SQL.

1. **ADO.NET Entity Framework**

The Entity Framework is a set of technologies in ADO.NET that support the development of data-oriented software applications. Architects and developers of data-oriented applications have typically struggled with the need to achieve two very different objectives. They must model the entities, relationships, and logic of the business problems they are solving, and they must also work with the data engines used to store and retrieve the data. The data may span multiple storage systems, each with its own protocols; even applications that work with a single storage system must balance the requirements of the storage system against the requirements of writing efficient and maintainable application code.

The Entity Framework enables developers to work with data in the form of domain-specific objects and properties, such as customers and customer addresses, without having to concern themselves with the underlying database tables and columns where this data is stored. With the Entity Framework, developers can work at a higher level of abstraction when they deal with data, and can create and maintain data-oriented applications with less code than in traditional applications. Because the Entity Framework is a component of the .NET Framework, Entity Framework applications can run on any computer on which the .NET Framework (starting with version 3.5 SP1) is installed.

1. **.NET Framework**

NET Framework (pronounced dot net) is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library known as Framework Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for .NET Framework execute in a software environment (as contrasted to hardware environment), known as Common Language Runtime (CLR), an application virtual machine that provides services such as security, memory management, and exception handling. FCL and CLR together constitute .NET Framework.

FCL provides user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. Programmers produce software by combining their own source code with .NET Framework and other libraries. .NET Framework is intended to be used by most new applications created for Windows platform. Microsoft also produces an integrated development environment largely for .NET software called Visual Studio.

1. **Case study – developing the Student Informer Web Application**

The Student Informer Web Application has been created for helping improve the communication between students and professors. The application has a friendly interface, easy to use, but with high functionality and utility.

**1.ASP.NET**

**1.2 History**

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](http://en.wikipedia.org/wiki/.NET_Framework). Even prior to the release, dozens of books had been written about ASP.NET, and Microsoft promoted it heavily as part of its platform for Web services. [Scott Guthrie](http://en.wikipedia.org/wiki/Scott_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](http://en.wikipedia.org/wiki/Windows_Server_2003). ASP.NET is loosely based on HTML. This release focused on improving ASP.NET's support for [mobile devices](http://en.wikipedia.org/wiki/Mobile_devices).

**1.3. Characteristics**

ASP.NET Web pages, known officially as Web Forms, are the main building blocks for application development. Web forms are contained in files with a ".aspx" extension; these files typically contain static ([X](http://en.wikipedia.org/wiki/XHTML))[HTML](http://en.wikipedia.org/wiki/HTML) markup, as well as markup defining server-side Web Controls and User Controls where the developers place all the rc contentfor 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](http://en.wikipedia.org/wiki/PHP), [JSP](http://en.wikipedia.org/wiki/JavaServer_Pages), and [ASP](http://en.wikipedia.org/wiki/Active_Server_Pages). With [ASP.NET Framework 2.0](http://en.wikipedia.org/wiki/ASP.NET_3.5), Microsoft introduced a new *code-behind* model which allows static text to remain on the .aspx page, while dynamic code remains in an .aspx.vb or .aspx.cs or .aspx.fs file (depending on the programming language used).

**1.3.1 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](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio) and other [IDEs](http://en.wikipedia.org/wiki/Integrated_development_environment). 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 walkthrough of 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](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/Model%E2%80%93View%E2%80%93Controller) (MVC) frameworks.

### 1.3.2 Directives

A [directive](http://en.wikipedia.org/wiki/Directive_(programming)) is a special instruction on how ASP.NET should process the page. The most common directive is <%@ Page %> which can specify many attributes used by the ASP.NET page parser and compiler.

<!-- Web.Config Configuration File -->

<configuration>

<system.web>

<customErrors mode="On"/>

</system.web>

</configuration>

**1.3.3 User controls**

*User controls* are encapsulations of sections of pages which are registered and used as controls in ASP.NET, org,etc.

### 1.3.4 Custom controls

Programmers can also build *custom controls* for ASP.NET applications. Unlike user controls, these controls do not have an ASCX markup file, having all their code compiled into a[dynamic link library (DLL)](http://en.wikipedia.org/wiki/Dynamic-link_library) file. Such custom controls can be used across multiple Web applications and [Visual Studio](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio) projects.

### 1.3.5 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 the class instance members that are lost with every page request/response cycle.

### 1.3.6 State management

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

#### **Application**

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

#### **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 Sessioncollection, 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](http://en.wikipedia.org/wiki/HTTP_cookie) or by encoding the session ID in the URL itself.

### ASP.NET supports three modes of persistence for server-side session variables:

**In-process mode**

The session variables are maintained within the ASP.NET [process](http://en.wikipedia.org/wiki/Process_(computing)). This is the fastest way; however, in this mode the variables are destroyed when the ASP.NET process is recycled or shut down.

**State server mode**

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

**SQL Server mode**

State variables are stored in a [database](http://en.wikipedia.org/wiki/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.

ASP.NET session state enables you to store and retrieve values for a user as the user navigates ASP.NET pages in a Web application. HTTP is a stateless protocol. This means that a Web server treats each HTTP request for a page as an independent request. The server retains no knowledge of variable values that were used during previous requests. ASP.NET session state identifies requests from the same browser during a limited time window as a session, and provides a way to persist variable values for the duration of that session. By default, ASP.NET session state is enabled for all ASP.NET applications.

Alternatives to session state include the following:

* Application state, which stores variables that can be accessed by all users of an ASP.NET application.
* Profile properties, which persists user values in a data store without expiring them.
* ASP.NET caching, which stores values in memory that is available to all ASP.NET applications.
* View state, which persists values in a page.
* Cookies.
* The query string and fields on an HTML form that are available from an HTTP request.

For a comparison of different state-management options, see ASP.NET State Management Recommendations Session.

#### **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](http://en.wikipedia.org/wiki/GUI_widget). 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.

The main use for this is to preserve form information across postbacks. View state is turned on by default and normally [serializes](http://en.wikipedia.org/wiki/Serialization) 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](http://en.wikipedia.org/wiki/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.

#### **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](http://en.wikipedia.org/wiki/HTTP_cookie), [caching](http://en.wikipedia.org/wiki/Web_cache), and using the [query string](http://en.wikipedia.org/wiki/Query_string).

### 1.3.7 Template engine

When first released, ASP.NET lacked a [template engine](http://en.wikipedia.org/wiki/Template_engine_(web)). Because the [.NET Framework](http://en.wikipedia.org/wiki/.NET_Framework) is [object-oriented](http://en.wikipedia.org/wiki/Object-oriented) and allows for [inheritance](http://en.wikipedia.org/wiki/Inheritance_(computer_science)), many developers would define a new base class that inherits from "System.Web.UI.Page", write [methods](http://en.wikipedia.org/wiki/Method_(computer_science)) 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](http://en.wikipedia.org/wiki/Source_code) with [markup](http://en.wikipedia.org/wiki/Markup_language). Furthermore, this method can only be visually tested by running the application – not while designing it. Other developers have used [include files](http://en.wikipedia.org/wiki/Header_file) 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](http://en.wikipedia.org/wiki/Web_template)-based page development. A Web application can have one or more master pages, which, beginning with ASP.NET 2.0, can be nested. Master templates have place-holder controls, called *ContentPlaceHolders* to denote where the dynamic content goes, as well as[HTML](http://en.wikipedia.org/wiki/HTML) and [JavaScript](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/Mail_merge) in a [word processor](http://en.wikipedia.org/wiki/Word_processor). All markup and [server](http://en.wikipedia.org/wiki/Server_(computing)) 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.

### 1.3.8 Other files

Other [file extensions](http://en.wikipedia.org/wiki/File_extension) associated with different versions of ASP.NET include:

| **Extension** | **Introduced in version** | **Description** |
| --- | --- | --- |
| asax | 1.0 | This is the global application file.You can use this file to define global variables (Variable that can be accessed from any Web page in the Web application.) It is mostly used to define the overall application event related to application & session object.[Global.asax](http://en.wikipedia.org/wiki/Global.asax), used for application-level logic |
| ascx | 1.0 | User Control, used for User Control files logic |
| ashx | 1.0 | custom [HTTP handlers](http://en.wikipedia.org/wiki/HTTP_handler) Do not have a user interface. |
| asmx | 1.0 | [Web service](http://en.wikipedia.org/wiki/Web_service) pages. From version 2.0 a Code behind page of an asmx file is placed into the app\_code folder. |
| aspx | 1.0 | An ASP.NET Web Forms page that can contain Web controls and presentation and business logic. <http://msdn.microsoft.com/en-us/library/2wawkw1c.aspx> |
| axd | 1.0 | when enabled in [web.config](http://en.wikipedia.org/wiki/Web.config) requesting trace.axd outputs application-level [tracing](http://en.wikipedia.org/wiki/Tracing_(software)). 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 one file (an 'assembly') |
| browser | 2.0 | browser capabilities files stored in [XML](http://en.wikipedia.org/wiki/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 abilities, 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](http://en.wikipedia.org/wiki/Web.config) in ASP.NET 1.x. |
| config | 1.0 | [web.config](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/XML) format. |
| cs/vb/fs | 1.0 | Code files (cs indicates C#, vb indicates Visual Basic, fs indicates F#). Code behind files (see above) predominantly have the extension ".aspx.cs" or ".aspx.vb" for the 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. |
| cshtml | 4.1 | Views (mixed C# and HTML using [Razor syntax](http://en.wikipedia.org/wiki/ASP.NET_Razor_view_engine)) |
| dbml | 3.5 | [LINQ to SQL](http://en.wikipedia.org/wiki/Language_Integrated_Query) data classes file |
| edmx | 3.5 | [ADO.NET Entity Framework](http://en.wikipedia.org/wiki/ADO.NET_Entity_Framework) model |
| master | 2.0 | master page file. Default file name is Master1.master |
| resx | 1.0 | resource files for [internationalization and localization](http://en.wikipedia.org/wiki/Internationalization_and_localization). Resource files can be global (e.g. messages) or "local" which means specific for one 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](http://en.wikipedia.org/wiki/Windows_Communication_Foundation) service file |
| vbhtml | 4.1 | Views (mixed VB and HTML using [Razor syntax](http://en.wikipedia.org/wiki/ASP.NET_Razor_view_engine)) |

### 1.3.9 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

**App\_Code**

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

**App\_Data**

The App\_Data ASP.NET Directory is the default directory for any [database](http://en.wikipedia.org/wiki/Database) used by the ASP.NET Website. These databases might include Access (mdb) files or [SQL Server](http://en.wikipedia.org/wiki/Microsoft_SQL_Server)(mdf) files. The App\_Data is the only directory with Write Access enabled for the ASP.NET web application.:

**App\_LocalResources**

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

**App\_GlobalResources**

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

**App\_Themes**

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

**App\_WebReferences**

holds discovery files and [WSDL](http://en.wikipedia.org/wiki/Web_Services_Description_Language) files for references to [Web services](http://en.wikipedia.org/wiki/Web_service) 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.

**1.4 Performance**

ASP.NET aims for performance benefits over other script-based technologies (including classic ASP) by compiling the server-side code the first time it is used to one or more [DLL](http://en.wikipedia.org/wiki/Dynamic_link_library)[files](http://en.wikipedia.org/wiki/Computer_file) on the [Web server](http://en.wikipedia.org/wiki/Web_server). These dll files or assemblies contain Microsoft Intermediate Language (MSIL) for running within the [common language runtime](http://en.wikipedia.org/wiki/Common_language_runtime); this provides a performance boost over pure scripted languages and is similar to the approach used by Python and not dissimilar to [java server pages](http://en.wikipedia.org/wiki/Java_server_pages).[]](http://en.wikipedia.org/wiki/ASP.NET#cite_note-macdonald7-8-18) This compilation happens automatically the first time a page is requested (which means the developer need not perform a separate compilation step for pages).

This feature provides the ease of development offered by scripting languages with the performance benefits of a compiled binary. However, the compilation might cause a noticeable but short delay to the Web user when the newly edited page is first requested from the Web server, but will not again unless the page requested is updated further.

The ASPX and other resource files are placed in a virtual host on an [Internet Information Services](http://en.wikipedia.org/wiki/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 Microsoft Visual Studio, eliminating the need for [just-in-time compilation](http://en.wikipedia.org/wiki/Just-in-time_compilation) in a production environment. This also eliminates the need of having the source code on the Web server. It also supports pre-compile text.

**1.5 Extension**

[Microsoft](http://en.wikipedia.org/wiki/Microsoft) has released some extension frameworks that plug into ASP.NET and extend its functionality. Some of them are:

**1.5.1** [**ASP.NET AJAX**](http://en.wikipedia.org/wiki/ASP.NET_AJAX)

ASP.NET AJAX is a set of extensions to [ASP.NET](http://en.wikipedia.org/wiki/ASP.NET) developed by [Microsoft](http://en.wikipedia.org/wiki/Microsoft) for implementing [Ajax](http://en.wikipedia.org/wiki/Ajax_(programming)) functionality. It is released under the [Microsoft Public License](http://en.wikipedia.org/wiki/Microsoft_Public_License) (Ms-PL).

**Clients supporting AJAX:**

ASP.NET AJAX runs on the following browsers:

* Microsoft Internet Explorer (>= 6.0)
* Mozilla Firefox (>= 1.5)
* Opera (>= 9.0)
* Apple Safari (>= 2.0)
* Google Chrome

**ASP.NET AJAX Suite**

At present, the ASP.NET AJAX suite consists of the following components and packages:

* Microsoft Now Launched Ajax Library 4.0, which supports Data Driven Web Applications.
* Microsoft Ajax Library 3.5, which is a JavaScript library that provides the client-side features of the ASP.NET AJAX framework. Integrated in ASP.NET 3.5, the library is also available as a separate download for use in other environments, such as [PHP](http://en.wikipedia.org/wiki/PHP).
* A server framework – included in ASP.NET 3.5 – for building Ajax-enabled ASP.NET server controls. These components are also available for ASP.NET 2.0 in a separate package called ASP.NET AJAX 1.0 Extensions.
* ASP.NET 2.0 AJAX Templates, a package with a set of Visual Studio templates for building ASP.NET AJAX applications with ASP.NET 2.0 and Visual Studio 2008.
* ASP.NET AJAX Preview, a package with the new features that will be part of the future versions of the framework.

**Microsoft Ajax Library**

The Microsoft Ajax Library is a JavaScript library that provides the features for the client portion of the ASP.NET AJAX framework. - Components – The library provides an infrastructure to build either visual or non-visual JavaScript components. A global JavaScript object – Sys.Application – is responsible for managing the lifecycle of client components.

- JavaScript extensions – An enhanced type system is introduced to emulate object-oriented constructs such as namespaces, classes and interfaces; and to perform reflection on client types.

- Abstraction API – Common operations on the DOM (retrieving elements, setting styles and other manipulations) are automatically translated by the library into browser-specific calls.

- Ajax – A set of client components is provided to handle Ajax requests and [web-service](http://en.wikipedia.org/wiki/Web_service) calls. - Application Services – The library allows accessing the ASP.NET Membership, Authentication, Roles and Profile services from the client side.

Recently, new features have been announced as part of the ASP.NET AJAX 4.0 release: - Template Engine – Allows displaying data on the client side by using HTML templates and a custom binding notation. This approach avoids performing page rendering on the server side.

- Declarative instantiation of client components – Allows registration, instantiation and configuration of client components using markup code, without writing any imperative JavaScript code.

- Live Bindings – Synchronize "element properties".

**The UpdatePanel Control**

The UpdatePanel is an ASP.NET server control that updates portions of a web page without reloading it. Through a mechanism called asynchronous [postback](http://en.wikipedia.org/wiki/Postback), the HTML for the region of the page wrapped by the control is sent by the server asynchronously through an Ajax request.

The ASP.NET controls that have been specified as content in an UpdatePanel are able to cause either synchronous (traditional) or asynchronous postbacks, by means of triggers.

A trigger is an event coming from an ASP.NET control that causes an UpdatePanel to refresh its contents. Through triggers, an asynchronous postback can be started also by controls that are declared outside the region of the ASP.NET page wrapped by the UpdatePanel control.

In the following code, only the content of the Update control (the span element that displays the current date and time) is re-rendered every time the button is clicked.

<asp:Button ID="Button1" runat="server" Text="Refresh" />

<asp:UpdatePanel ID="UpdatePanel1" runat="server"><Triggers>

<asp:AsyncPostBackTrigger ControlID="Button1" EventName="Click" />

</Triggers>

<ContentTemplate>

<span><%= DateTime{{Not a typo|.}}Now %></span>

</ContentTemplate>

</asp:UpdatePanel>

**Web-services and JSON**

ASP.NET AJAX framework brings [JSON](http://en.wikipedia.org/wiki/JSON) serialization features to the ASP.NET web-services and allows calling web-services from [client-side JavaScript](http://en.wikipedia.org/wiki/Client-side_JavaScript), even using third-party Javascript-libraries like [jQuery](http://en.wikipedia.org/wiki/JQuery).

**1.5.2 ASP.NET MVC Framework**

The **ASP.NET MVC** is an open source [web application framework](http://en.wikipedia.org/wiki/Web_application_framework) that implements the [model–view–controller](http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) (MVC) pattern.

In the latter versions of ASP.NET, ASP.NET MVC, ASP.NET Web API, and ASP.NET Web Pages (a platform using only [Razor](http://en.wikipedia.org/wiki/ASP.NET_Razor_view_engine) pages) will merge into a unified MVC 6. The project is called "ASP.NET vNext".

**Background**

Based on [ASP.NET](http://en.wikipedia.org/wiki/ASP.NET), ASP.NET MVC allows software developers to build a [web application](http://en.wikipedia.org/wiki/Web_application) as a composition of three roles: *Model*, *View* and *Controller*. The MVC model defines web applications with 3 logic layers:

* Model (business layer)
* View (display layer)
* Controller (input control)

A *model* represents the state of a particular aspect of the application. A *controller* handles interactions and updates the model to reflect a change in state of the application, and then passes information to the view. A *view* accepts necessary information from the controller and renders a user interface to display that information.

In April 2009, the ASP.NET MVC source code was released under the [Microsoft Public License](http://en.wikipedia.org/wiki/Microsoft_Public_License) (MS-PL).

ASP.NET MVC framework is a lightweight, highly testable presentation framework that is integrated with existing ASP.NET features. Some of these integrated features are master pages and membership-based authentication. The MVC framework is defined in the System.Web.Mvc assembly.

The ASP.NET MVC Framework couples the models, views, and controllers using [interface](http://en.wikipedia.org/wiki/Interface_(object-oriented_programming))-based contracts, thereby allowing each component to be tested independently.

### Apache License 2.0 release

In March 2012, [Scott Guthrie](http://en.wikipedia.org/wiki/Scott_Guthrie) announced on his blog that Microsoft had released part of their web stack (including ASP.NET MVC, [Razor](http://en.wikipedia.org/wiki/ASP.NET_Razor_view_engine) and Web API) under an open source license ([Apache License 2.0](http://en.wikipedia.org/wiki/Apache_License_2.0)).

Guthrie wrote that "Doing so will enable a more open development model where everyone in the community will be able to engage and provide feedback on code checkins, bug-fixes, new feature development, and build and test the products on a daily basis using the most up-to-date version of the source code and tests."

The source code now resides on [CodePlex](http://en.wikipedia.org/wiki/CodePlex). ASP.NET Web Forms was not included in this initiative for various reasons.

**View engines**

The view engines used in the ASP.NET MVC 3 and MVC 4 Frameworks are the [Razor View Engine](http://en.wikipedia.org/wiki/ASP.NET_Razor_view_engine) and the Web Forms view engine. Both view engines are part of the MVC 3 framework. By default, the view engine in the MVC framework uses Razor .cshtml and .vbhtml, or Web Forms .aspx pages to design the layout of the user interface pages onto which the data is composed. However, different view engines can be used. Additionally, rather than the default ASP.NET Web Forms [postback](http://en.wikipedia.org/wiki/Postback#In_web_development) model, any interactions are routed to the controllers using the ASP.NET [Routing](http://en.wikipedia.org/wiki/Rewrite_engine) mechanism. Views can be mapped to [REST](http://en.wikipedia.org/wiki/Representational_State_Transfer)-friendly [URLs](http://en.wikipedia.org/wiki/Uniform_Resource_Locator).

Other view engines:

* The [MVCContrib](http://www.codeplex.com/MVCContrib) library contains 8 alternate view engines. Brail, NDjango, NHaml, NVelocity, SharpTiles, Spark, StringTemplate and XSLT.
* The [StringTemplate View Engine](http://code.google.com/p/string-template-view-engine-mvc) utilizes a .NET port of the popular Java Templating engine, [StringTemplate](http://www.stringtemplate.org/).
* [Spark](http://sparkviewengine.codeplex.com/) is a view engine for the ASP.NET MVC (and the Castle Project MonoRail) frameworks.
* [NDjango](http://ndjango.org/index.php?title=NDjango_Home) is a port of the popular [Django templating engine](http://docs.djangoproject.com/en/dev/topics/templates/) to .NET. It is written in F# and comes with Visual Studio extension including full Intellisense support
* [Naked Objects MVC](http://en.wikipedia.org/wiki/Naked_Objects_MVC) - an implementation of the [naked objects](http://en.wikipedia.org/wiki/Naked_objects) pattern using ASP.NET MVC
* [Razor](http://weblogs.asp.net/scottgu/archive/2010/07/02/introducing-razor.aspx) is a view-engine [developed by Microsoft](http://www.asp.net/mvc/mvc3#BM_The_Razor_View_Engine) and released with MVC 3 that is optimized around [HTML](http://en.wikipedia.org/wiki/HTML) generation using a code-focused templating approach.

**1.5.3 ASP.NET Razor view engine**

**Razor** is an [ASP.NET](http://en.wikipedia.org/wiki/ASP.NET) programming syntax used to create [dynamic web pages](http://en.wikipedia.org/wiki/Dynamic_web_page) with the [C#](http://en.wikipedia.org/wiki/C_Sharp_(programming_language)) or [Visual Basic .NET](http://en.wikipedia.org/wiki/Visual_Basic_.NET) programming languages. Razor was in development in June 2010 and was released for Microsoft Visual Studio 2010 in January 2011. Razor is a simple-syntax view engine and was released as part of [ASP.NET MVC](http://en.wikipedia.org/wiki/ASP.NET_MVC) 3 and the [Microsoft WebMatrix](http://en.wikipedia.org/wiki/Microsoft_WebMatrix) tool set.

**Advantages of Razor**

The idea behind Razor is to provide an optimized syntax for HTML generation using a code-focused templating approach, with minimal transition between HTML and code. The design reduces the number of characters and keystrokes, and enables a more fluid coding workflow by not requiring explicitly denoted server blocks within the HTML code. Other advantages that have been noted:

* Supports [IntelliSense](http://en.wikipedia.org/wiki/IntelliSense) (statement completion support)
* [Unit Testable](http://en.wikipedia.org/wiki/Unit_testing)
* Supports "layouts" (an alternative to the "master page" concept in classic aspx pages)

**1.5.4 ASP.NET Dynamic Data**

**ASP.NET Dynamic Data** is a [Ruby on Rails](http://en.wikipedia.org/wiki/Ruby_on_Rails)-inspired [web application](http://en.wikipedia.org/wiki/Web_application) [scaffolding](http://en.wikipedia.org/wiki/Scaffold_(programming)) framework from [Microsoft](http://en.wikipedia.org/wiki/Microsoft), shipped as an extension to [ASP.NET](http://en.wikipedia.org/wiki/ASP.NET), that can be used to build data driven web applications. It exposes tables in a [database](http://en.wikipedia.org/wiki/Database) by encoding it in the [URI](http://en.wikipedia.org/wiki/URI) of the [ASP.NET](http://en.wikipedia.org/wiki/ASP.NET) web service, **and** the data in the table is automatically rendered to [HTML](http://en.wikipedia.org/wiki/HTML). The process of rendering can be controlled using custom design [templates](http://en.wikipedia.org/wiki/Template_processor). Internally, it discovers the [database schema](http://en.wikipedia.org/wiki/Database_schema) by using the database [metadata](http://en.wikipedia.org/wiki/Metadata).

ASP.NET Dynamic Data was originally shipped as part of the "ASP.NET 3.5 Extensions" package in 2007, and was incorporated into the [.NET Framework](http://en.wikipedia.org/wiki/.NET_Framework) 3.5 Service Pack 1, which was released August 11, 2008.

**1.5.5 ASP.NET Web API**

An HTTP API framework for exposing web services.

**1.5.6 ASP.NET SignalR**

ASP.NET SignalR is a library for [ASP.NET](http://en.wikipedia.org/wiki/ASP.NET) developers to add real-time web functionality to their applications. Real-time web functionality is the ability to have server-side code push content to the connected clients as it happens, in real-time.

SignalR takes advantage of several transports, automatically selecting the best available transport given the client's and server's best available transport. SignalR takes advantage of [WebSocket](http://en.wikipedia.org/wiki/WebSocket), an [HTML5](http://en.wikipedia.org/wiki/HTML5) API that enables bi-directional communication between the browser and server. SignalR will use WebSockets under the covers when it's available, and gracefully fall back to other techniques and technologies when it isn't, while the application code remains the same.

SignalR also provides a simple, high-level [API](http://en.wikipedia.org/wiki/API) for doing server-to-client [RPC](http://en.wikipedia.org/wiki/Remote_procedure_call) (call JavaScript functions in a client's browser from server-side .NET code) in your ASP.NET application, as well as adding useful hooks for connection management, such as connect/disconnect events, grouping connections, authorization.

**1.6 ASP.NET compared with classic ASP**

ASP.NET simplifies developers' transition from [Windows](http://en.wikipedia.org/wiki/Microsoft_Windows) application development to Web development by offering the ability to build pages composed of [controls](http://en.wikipedia.org/wiki/Widget_(computing)) similar to a Windows [user interface](http://en.wikipedia.org/wiki/User_interface). A Web control, such as a button or label, functions in very much the same way as its Windows counterparts: code can assign its properties and respond to its events. Controls know how to render themselves: whereas Windows controls draw themselves to the screen, Web controls produce segments of [HTML](http://en.wikipedia.org/wiki/HTML) and [JavaScript](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/Event-driven_programming) [GUI](http://en.wikipedia.org/wiki/Graphical_user_interface) model, rather than in conventional Web-[scripting](http://en.wikipedia.org/wiki/Scripting_programming_language) environments like ASP and [PHP](http://en.wikipedia.org/wiki/PHP). The framework combines existing technologies such as JavaScript with internal components like "[ViewState](http://en.wikipedia.org/wiki/ViewState)" to bring persistent (inter-request) state to the inherently [stateless](http://en.wikipedia.org/wiki/Stateless_server) Web environment.

Other differences compared to [classic ASP](http://en.wikipedia.org/wiki/Active_Server_Pages) 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](http://en.wikipedia.org/wiki/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, as well as user-defined controls, allow the rapid building of applications. Layout of these controls on a page is easier because most of it can be done visually in most editors.
* ASP.NET uses the multi-language abilities of the .NET [Common Language Runtime](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/ASP.NET#Code-behind_model) development model to separate business logic from presentation.
* Ability to use true [object-oriented](http://en.wikipedia.org/wiki/Object-oriented_programming) design for programming pages and controls
* If an ASP.NET application [leaks memory](http://en.wikipedia.org/wiki/Memory_leak), the ASP.NET runtime unloads the AppDomain hosting the erring application and reloads the application in a new AppDomain.
* [Session state](http://en.wikipedia.org/wiki/ASP.NET#Session_state) in ASP.NET can be saved in a [Microsoft SQL Server](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/World_Wide_Web_Consortium)/[ECMA](http://en.wikipedia.org/wiki/Ecma_International) standards. In addition, the framework's browser detection feature sometimes incorrectly identified Web browsers other than Microsoft's own[Internet Explorer](http://en.wikipedia.org/wiki/Internet_Explorer) as "downlevel" and returned HTML/JavaScript to these clients with some of the features removed, or sometimes crippled or broken. In version 2.0 however, all controls generate valid HTML 4.0, XHTML 1.0 (the default) or XHTML 1.1 output, depending on the site configuration. Detection of standards-compliant Web browsers is more robust and support for [Cascading Style Sheets](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/WYSIWYG)controls.

**1.7 Criticism**

On [IIS](http://en.wikipedia.org/wiki/Internet_Information_Services) 6.0 and lower, pages written using different versions of the ASP framework cannot share [session state](http://en.wikipedia.org/wiki/ASP.NET#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](http://en.wikipedia.org/wiki/Internet_Information_Services) 7. With IIS 7.0, modules may be run in an integrated pipeline that allows modules written in any language to be executed for any request.

**1.8 Development tools**

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

| **Software** | **Developer** | **Licensing** |
| --- | --- | --- |
| [Wiwet ASP.NET Templates](http://en.wikipedia.org/w/index.php?title=Wiwet_ASP.NET_Templates&action=edit&redlink=1) | [Avalon](http://en.wikipedia.org/wiki/Avalon) | Commercial |
| [ASP.NET Intellisense Generator](http://en.wikipedia.org/w/index.php?title=ASP.NET_Intellisense_Generator&action=edit&redlink=1) | BlueVision LLC | Free |
| [Microsoft Visual Studio](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio) | [Microsoft](http://en.wikipedia.org/wiki/Microsoft) | Free and commercial |
| Microsoft Visual Web Developer Express | [Microsoft](http://en.wikipedia.org/wiki/Microsoft) | Registerware |
| [CodeGear Delphi](http://en.wikipedia.org/wiki/CodeGear_Delphi) | [Embarcadero Technologies](http://en.wikipedia.org/wiki/Embarcadero_Technologies) | Commercial |
| [Macromedia HomeSite](http://en.wikipedia.org/wiki/Macromedia_HomeSite) | [Adobe Systems](http://en.wikipedia.org/wiki/Adobe_Systems) | Commercial |
| [Microsoft Expression Web](http://en.wikipedia.org/wiki/Microsoft_Expression_Web) | [Microsoft](http://en.wikipedia.org/wiki/Microsoft) | Free |
| [Microsoft SharePoint Designer](http://en.wikipedia.org/wiki/Microsoft_SharePoint_Designer) | [Microsoft](http://en.wikipedia.org/wiki/Microsoft) | Free |
| [MonoDevelop](http://en.wikipedia.org/wiki/MonoDevelop) | [Novell](http://en.wikipedia.org/wiki/Novell) and the Mono community | Free open source |
| [SharpDevelop](http://en.wikipedia.org/wiki/SharpDevelop) | ICSharpCode Team | Free open source |
| Eiffel for ASP.NET | [Eiffel Software](http://en.wikipedia.org/wiki/Eiffel_Software) | Free open source and commercial |
| [Adobe Dreamweaver](http://en.wikipedia.org/wiki/Adobe_Dreamweaver) | [Adobe Systems](http://en.wikipedia.org/wiki/Adobe_Systems) | Commercial |
|  |  |  |
| Stadium | Twenty57 | Commercial, free beta |

**1.9 Frameworks**

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

* [Base One Foundation Component Library](http://en.wikipedia.org/wiki/Base_One_Foundation_Component_Library) (BFC) is a [RAD](http://en.wikipedia.org/wiki/Rapid_application_development) framework for building .NET [database](http://en.wikipedia.org/wiki/Database) and [distributed computing](http://en.wikipedia.org/wiki/Distributed_computing) applications.
* [DotNetNuke](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/MonoRail_(software)), an open-source [MVC](http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) framework with an execution model similar to [Ruby on Rails](http://en.wikipedia.org/wiki/Ruby_on_Rails). The framework is commonly used with [Castle ActiveRecord](http://en.wikipedia.org/wiki/Castle_ActiveRecord), an ORM layer built on [NHibernate](http://en.wikipedia.org/wiki/NHibernate).
* [Spring.NET](http://www.springframework.net/), a port of the [Spring framework](http://en.wikipedia.org/wiki/Spring_framework) for Java.
* [Survey Project](http://en.wikipedia.org/w/index.php?title=Survey_Project&action=edit&redlink=1) is an open-source web-based survey and form engine framework written in ASP.NET and C#.
* [Carbon MVVM](http://carbonmvvm.codeplex.com/) is an open-source MVVM framework based on ASP.NET

**1.10 Versions**

The ASP.NET releases history tightly correlates with the [.NET Framework](http://en.wikipedia.org/wiki/.NET_Framework) releases:

| **Date** | **Version** | **Remarks** | **New ASP.NET related features** |
| --- | --- | --- | --- |
| January 16, 2002 | 1.0 | First version  released together with [Visual Studio .NET](http://en.wikipedia.org/wiki/Visual_Studio_.NET) | * [Object-oriented](http://en.wikipedia.org/wiki/Object-oriented) Web application development supporting [inheritance](http://en.wikipedia.org/wiki/Inheritance_(computer_science)),[polymorphism](http://en.wikipedia.org/wiki/Polymorphism_in_object-oriented_programming) and other standard OOP features   + Developers are no longer forced to use Server.CreateObject(...), so early-binding and type safety are possible. * Based on [Windows](http://en.wikipedia.org/wiki/Microsoft_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](http://en.wikipedia.org/wiki/Exception_handling)) |
| April 24, 2003 | 1.1 | released together with [Windows Server 2003](http://en.wikipedia.org/wiki/Windows_Server_2003)  released together with [Visual Studio .NET 2003](http://en.wikipedia.org/wiki/Visual_Studio_.NET_2003) | * Mobile controls * Automatic input validation |
| November 7, 2005 | 2.0 | codename [Whidbey](http://en.wikipedia.org/wiki/Microsoft_codenames) released together with [Visual Studio 2005](http://en.wikipedia.org/wiki/Visual_Studio_2005) and [Visual Web Developer Express](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio_Express) and [SQL Server 2005](http://en.wikipedia.org/wiki/Microsoft_SQL_Server) | * New data controls (GridView, FormView, DetailsView) * New technique for declarative data access (SqlDataSource, ObjectDataSource, XmlDataSource controls) * Navigation controls * [Master pages](http://en.wikipedia.org/wiki/ASP_master_pages) * Login controls * Themes * Skins * Web parts * Personalization services * Full pre-compilationa * New localization technique * Support for 64-bit processors * Provider class model |
| November 21, 2006 | 3.0 |  | * [Windows Presentation Foundation](http://en.wikipedia.org/wiki/Windows_Presentation_Foundation) (WPF) * [Windows Workflow Foundation](http://en.wikipedia.org/wiki/Windows_Workflow_Foundation) (WF) * [Windows Communication Foundation](http://en.wikipedia.org/wiki/Windows_Communication_Foundation) which can use ASP.NET to host services. * [Windows CardSpace](http://en.wikipedia.org/wiki/Windows_CardSpace) which uses ASP.NET for login roles. |
| November 19, 2007 | 3.5 | Released with [Visual Studio 2008](http://en.wikipedia.org/wiki/Visual_Studio_2008) and [Windows Server 2008](http://en.wikipedia.org/wiki/Windows_Server_2008) | * New data controls (ListView, DataPager) * [ASP.NET AJAX](http://en.wikipedia.org/wiki/ASP.NET_AJAX) included as part of the framework * Support for HTTP pipelining and syndication feeds. * WCF support for RSS, JSON, POX and Partial Trust * All the [.NET Framework 3.5](http://en.wikipedia.org/wiki/.NET_Framework_3.5) changes, like [LINQ](http://en.wikipedia.org/wiki/Language_Integrated_Query) etc. |
| August 11, 2008 | 3.5 Service Pack 1 | Released with Visual Studio 2008 Service Pack 1 | * Incorporation of [ASP.NET Dynamic Data](http://en.wikipedia.org/wiki/ASP.NET_Dynamic_Data) * Support for controlling browser history in an ASP.NET AJAX application * Ability to combine multiple JavaScript files into one file for more efficient downloading * New namespaces System.Web.Abstractions and System.Web.Routing |
| April 12, 2010 | 4.0 | Parallel extensions and other [.NET Framework 4](http://en.wikipedia.org/wiki/.NET_Framework_4) features | The two new properties added in the Page class are MetaKeyword and MetaDescription. |
| August 15, 2012 | **4.5** | Released with [Visual Studio 2012](http://en.wikipedia.org/wiki/Visual_Studio_2012) and [Windows Server 2012](http://en.wikipedia.org/wiki/Windows_Server_2012) for Windows 8  Parallel extensions and other [.NET Framework 4.5](http://en.wikipedia.org/wiki/.NET_Framework_4.5)features |  |
|  | | | |

**1.11 Other implementations**

The [Mono](http://en.wikipedia.org/wiki/Mono_(software)) Project supports "everything in .NET 4.5 except [WPF](http://en.wikipedia.org/wiki/Windows_Presentation_Foundation), [WF](http://en.wikipedia.org/wiki/Windows_Workflow_Foundation), limited [WCF](http://en.wikipedia.org/wiki/Windows_Communication_Foundation), and limited ASP.NET 4.5 async stack." ASP.NET can be run with Mono using one of three options: Apache hosting using the mod\_mono module, FastCGI hosting, and XSP.

**2. Microsoft SQL Server**

**2.1 History**

**2.1.1 Genesis**

**SQL Server Release History table**

Prior to version 7.0 the code base for MS SQL Server was sold by Sybase SQL Server to Microsoft, and was Microsoft's entry to the enterprise-level database market, competing against Oracle, IBM, and, later, Sybase. Microsoft, Sybase and Ashton-Tate originally worked together to create and market the first version named SQL Server 1.0 for OS/2 (about 1989) which was essentially the same as Sybase SQL Server 3.0 on Unix, VMS, etc. Microsoft SQL Server 4.2 was shipped around 1992 (available bundled with IBM OS/2 version 1.3). Later Microsoft SQL Server 4.21 for Windows NT was released at the same time as Windows NT 3.1. Microsoft SQL Server v6.0 was the first version designed for NT, and did not include any direction from Sybase.

About the time Windows NT was released in July 1993, Sybase and Microsoft parted ways and each pursued its own design and marketing schemes. Microsoft negotiated exclusive rights to all versions of SQL Server written for Microsoft operating systems. (In 1996 Sybase changed the name of its product to Adaptive Server Enterprise to avoid confusion with Microsoft SQL Server.) Until 1994, Microsoft's SQL Server carried three Sybase copyright notices as an indication of its origin.

SQL Server 7.0 and SQL Server 2000 included modifications and extensions to the Sybase code base, adding support for the IA-64 architecture. By SQL Server 2005 the legacy Sybase code had been completely rewritten.

Since the release of SQL Server 2000, advances have been made in performance, the client IDE tools, and several complementary systems that are packaged with SQL Server 2005.

These include:

* an extract-transform-load (ETL) tool (SQL Server Integration Services or SSIS)
* a Reporting Server
* an OLAP and data mining server (Analysis Services)
* several messaging technologies, specifically Service Broker and Notification Services

**2.1.2 SQL Server 2005**

SQL Server 2005 (formerly codenamed "Yukon") released in October 2005. It included native support for managing XML data, in addition to relational data. For this purpose, it defined an xml data type that could be used either as a data type in database columns or as literals in queries. XML columns can be associated with XSD schemas; XML data being stored is verified against the schema. XML is converted to an internal binary data type before being stored in the database. Specialized indexing methods were made available for XML data. XML data is queried using XQuery; SQL Server 2005 added some extensions to the T-SQL language to allow embedding XQuery queries in T-SQL. In addition, it also defines a new extension to XQuery, called XML DML, that allows query-based modifications to XML data. SQL Server 2005 also allows a database server to be exposed over web services using Tabular Data Stream (TDS) packets encapsulated within SOAP (protocol) requests. When the data is accessed over web services, results are returned as XML.

Common Language Runtime (CLR) integration was introduced with this version, enabling one to write SQL code as Managed Code by the CLR. For relational data, T-SQL has been augmented with error handling features (try/catch) and support for recursive queries with CTEs (Common Table Expressions). SQL Server 2005 has also been enhanced with new indexing algorithms, syntax and better error recovery systems. Data pages are checksummed for better error resiliency, and optimistic concurrency support has been added for better performance. Permissions and access control have been made more granular and the query processor handles concurrent execution of queries in a more efficient way. Partitions on tables and indexes are supported natively, so scaling out a database onto a cluster is easier. SQL CLR was introduced with SQL Server 2005 to let it integrate with the .NET Framework.

SQL Server 2005 introduced Multi-Version Concurrency Control. User facing features include new transaction isolation level called SNAPSHOT and a variation of the READ COMMITTED isolation level based on statement-level data snapshots.

SQL Server 2005 introduced "MARS" (Multiple Active Results Sets), a method of allowing usage of database connections for multiple purposes.

SQL Server 2005 introduced DMVs (Dynamic Management Views), which are specialized views and functions that return server state information that can be used to monitor the health of a server instance, diagnose problems, and tune performance.

Service Pack 1 (SP1) of SQL Server 2005 introduced Database Mirroring,a high availability option that provides redundancy and failover capabilities at the database level. Failover can be performed manually or can be configured for automatic failover. Automatic failover requires a witness partner and an operating mode of synchronous (also known as high-safety or full safety).

**2.1.3 SQL Server 2008**

SQL Server 2008 (formerly codenamed "Katmai") was released on August 6, 2008 and aims to make data management self-tuning, self organizing, and self maintaining with the development of SQL Server Always On technologies, to provide near-zero downtime. SQL Server 2008 also includes support for structured and semi-structured data, including digital media formats for pictures, audio, video and other multimedia data. In current versions, such multimedia data can be stored as BLOBs (binary large objects), but they are generic bitstreams. Intrinsic awareness of multimedia data will allow specialized functions to be performed on them. According to Paul Flessner, senior Vice President, Server Applications, Microsoft Corp., SQL Server 2008 can be a data storage backend for different varieties of data: XML, email, time/calendar, file, document, spatial, etc as well as perform search, query, analysis, sharing, and synchronization across all data types.

Other new data types include specialized date and time types and a Spatial data type for location-dependent data. Better support for unstructured and semi-structured data is provided using the new FILESTREAM data type, which can be used to reference any file stored on the file system. Structured data and metadata about the file is stored in SQL Server database, whereas the unstructured component is stored in the file system. Such files can be accessed both via Win32 file handling APIs as well as via SQL Server using T-SQL; doing the latter accesses the file data as a BLOB. Backing up and restoring the database backs up or restores the referenced files as well.SQL Server 2008 also natively supports hierarchical data, and includes T-SQL constructs to directly deal with them, without using recursive queries.

The Full-text search functionality has been integrated with the database engine. According to a Microsoft technical article, this simplifies management and improves performance.

Spatial data will be stored in two types. A "Flat Earth" (GEOMETRY or planar) data type represents geospatial data which has been projected from its native, spherical, coordinate system into a plane. A "Round Earth" data type (GEOGRAPHY) uses an ellipsoidal model in which the Earth is defined as a single continuous entity which does not suffer from the singularities such as the international dateline, poles, or map projection zone "edges". Approximately 70 methods are available to represent spatial operations for the Open Geospatial Consortium Simple Features for SQL, Version 1.1.

SQL Server includes better compression features, which also helps in improving scalability. It enhanced the indexing algorithms and introduced the notion of filtered indexes. It also includes Resource Governor that allows reserving resources for certain users or workflows. It also includes capabilities for transparent encryption of data (TDE) as well as compression of backups. SQL Server 2008 supports the ADO.NET Entity Framework and the reporting tools, replication, and data definition will be built around the Entity Data Model. SQL Server Reporting Services will gain charting capabilities from the integration of the data visualization products from Dundas Data Visualization, Inc., which was acquired by Microsoft. On the management side, SQL Server 2008 includes the Declarative Management Framework which allows configuring policies and constraints, on the entire database or certain tables, declaratively. The version of SQL Server Management Studio included with SQL Server 2008 supports IntelliSense for SQL queries against a SQL Server 2008 Database Engine. SQL Server 2008 also makes the databases available via Windows PowerShell providers and management functionality available as Cmdlets, so that the server and all the running instances can be managed from Windows PowerShell.

The final SQL Server 2008 service pack (10.00.6000, Service Pack 4) was released on September 30, 2014.

**2.1.4 SQL Server 2008 R2**

SQL Server 2008 R2 (10.50.1600.1, formerly codenamed "Kilimanjaro") was announced at TechEd 2009, and was released to manufacturing on April 21, 2010. SQL Server 2008 R2 adds certain features to SQL Server 2008 including a master data management system branded as Master Data Services, a central management of master data entities and hierarchies. Also Multi Server Management, a centralized console to manage multiple SQL Server 2008 instances and services including relational databases, Reporting Services, Analysis Services & Integration Services.

SQL Server 2008 R2 includes a number of new services, including PowerPivot for Excel and SharePoint, Master Data Services, StreamInsight, Report Builder 3.0, Reporting Services Add-in for SharePoint, a Data-tier function in Visual Studio that enables packaging of tiered databases as part of an application, and a SQL Server Utility named UC (Utility Control Point), part of AMSM (Application and Multi-Server Management) that is used to manage multiple SQL Servers.

The first SQL Server 2008 R2 service pack (10.50.2500, Service Pack 1) was released on July 11, 2011.

The second SQL Server 2008 R2 service pack (10.50.4000, Service Pack 2) was released on July 26, 2012.

The final SQL Server 2008 R2 service pack (10.50.6000, Service Pack 3) was released on September 26, 2014.

**2.1.5 SQL Server 2012**

At the 2011 Professional Association for SQL Server (PASS) summit on October 11, Microsoft announced that the next major version of SQL Server (codenamed "Denali"), would be SQL Server 2012. It was released to manufacturing on March 6, 2012. SQL Server 2012 Service Pack 1 was released to manufacturing on November 9, 2012, and Service Pack 2 was released to manufacturing on June 10, 2014.

It was announced to be the last version to natively support OLE DB and instead to prefer ODBC for native connectivity.

SQL Server 2012's new features and enhancements include AlwaysOn SQL Server Failover Cluster Instances and Availability Groups which provides a set of options to improve database availability, Contained Databases which simplify the moving of databases between instances, new and modified Dynamic Management Views and Functions, programmability enhancements including new spatial features, metadata discovery, sequence objects and the THROW statement, performance enhancements such as ColumnStore Indexes as well as improvements to OnLine and partition level operations and security enhancements including provisioning during setup, new permissions, improved role management, and default schema assignment for groups.

**2.1.6 SQL Server 2014**

SQL Server 2014 was released to manufacturing on March 18, 2014, and released to the general public on April 1, 2014. Until November 2013 there were two CTP revisions, CTP1 and CTP2. SQL Server 2014 provides a new in-memory capability for tables that can fit entirely in memory (also known as Hekaton). Whilst small tables may be entirely resident in memory in all versions of SQL Server, they also may reside on disk, so work is involved in reserving RAM, writing evicted pages to disk, loading new pages from disk, locking the pages in RAM while they are being operated on, and many other tasks. By treating a table as guaranteed to be entirely resident in memory much of the 'plumbing' of disk-based databases can be avoided.

For disk-based SQL Server applications, it also provides the SSD Buffer Pool Extension, which can improve performance by cache between DRAM and spinning media.

SQL Server 2014 also enhances the AlwaysOn (HADR) solution by increasing the readable secondaries count and sustaining read operations upon secondary-primary disconnections, and it provides new hybrid disaster recovery and backup solutions with Windows Azure, enabling customers to use existing skills with the on-premises version of SQL Server to take advantage of Microsoft's global datacenters. In addition, it takes advantage of new Windows Server 2012 and Windows Server 2012 R2 capabilities for database application scalability in a physical or virtual environment.

Microsoft provides three versions of SQL Server 2014 for downloading: the one that runs on Microsoft Azure, the SQL Server 2014 CAB, and SQL Server 2014 ISO.

**2.2. Builds**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Version** | **RTM (no SP)** | **SP1** | **SP2** | **SP3** | **SP4** | **SP5** |
| SQL Server 2014 | 12.00.2000.8 |  |  |  |  |  |
| SQL Server 2012 | 11.00.2100.60 | 11.00.3000.0 | 11.00.5058.0 |  |  |  |
| SQL Server 2008 R2 | 10.50.1600.1 | 10.50.2500.0 / 10.51.2500.0 | 10.50.4000.0 / 10.52.4000.0 | 10.50.6000.34 |  |  |
| SQL Server 2008 | 10.00.1600.22 | 10.00.2531.0 | 10.00.4000.0 | 10.00.5500.0 | 10.00.6000.29 |  |
| SQL Server 2005 | 9.00.1399.06 | 9.00.2047 | 9.00.3042 | 9.00.4035 | 9.00.5000 |  |
| SQL Server 2000 | 8.00.194 | 8.00.384 | 8.00.532 | 8.00.760 | 8.00.2039 |  |
| SQL Server 7.0 | 7.00.623 | 7.00.699 | 7.00.842 | 7.00.961 | 7.00.1063 |  |
| SQL Server 6.5 | 6.50.201 | 6.50.213 | 6.50.240 | 6.50.258 (SP3a) | 6.50.281 | 6.50.416 (SP5a) |
| SQL Server 6.0 | 6.00.121 | 6.00.124 | 6.00.139 | 6.00.151 |  |  |

**2.3. Editions**

Microsoft makes SQL Server available in multiple editions, with different feature sets and targeting different users. These editions are:

**2.3.1 Mainstream editions**

**Datacenter**

SQL Server 2008 R2 Datacenter is the full-featured edition of SQL Server and is designed for datacenters that need the high levels of application support and scalability. It supports 256 logical processors and virtually unlimited memory. Comes with StreamInsight Premium edition.The Datacenter edition has been retired in SQL Server 2012, all its features are available in SQL Server 2012 Enterprise Edition.

**Enterprise**

SQL Server Enterprise Edition includes both the core database engine and add-on services, with a range of tools for creating and managing a SQL Server cluster. It can manage databases as large as 524 petabytes and address 2 terabytes of memory and supports 8 physical processors. SQL Server 2012 Enterprise Edition supports 160 physical processors.

**Standard**

SQL Server Standard edition includes the core database engine, along with the stand-alone services. It differs from Enterprise edition in that it supports fewer active instances (number of nodes in a cluster) and does not include some high-availability functions such as hot-add memory (allowing memory to be added while the server is still running), and parallel indexes.

**Web**

SQL Server Web Edition is a low-TCO option for Web hosting.

**Business Intelligence**

Introduced in SQL Server 2012 and focusing on Self Service and Corporate Business Intelligence. It includes the Standard Edition capabilities and Business Intelligence tools: PowerPivot, Power View, the BI Semantic Model, Master Data Services, Data Quality Services and xVelocity in-memory analytics.

**Workgroup**

SQL Server Workgroup Edition includes the core database functionality but does not include the additional services. Note that this edition has been retired in SQL Server 2012.

**Express**

SQL Server Express Edition is a scaled down, free edition of SQL Server, which includes the core database engine. While there are no limitations on the number of databases or users supported, it is limited to using one processor, 1 GB memory and 10 GB database files (4 GB database files prior to SQL Server Express 2008 R2). It is intended as a replacement for MSDE. Two additional editions provide a superset of features not in the original Express Edition. The first is SQL Server Express with Tools, which includes SQL Server Management Studio Basic. SQL Server Express with Advanced Services adds full-text search capability and reporting services.

**2.3.2 Specialized editions**

**Azure**

Microsoft SQL Azure Database is the cloud-based version of Microsoft SQL Server, presented as software as a service on Azure Services Platform.

**Compact (SQL CE)**

The compact edition is an embedded database engine. Unlike the other editions of SQL Server, the SQL CE engine is based on SQL Mobile (initially designed for use with hand-held devices) and does not share the same binaries. Due to its small size (1 MB DLL footprint), it has a markedly reduced feature set compared to the other editions. For example, it supports a subset of the standard data types, does not support stored procedures or Views or multiple-statement batches (among other limitations). It is limited to 4 GB maximum database size and cannot be run as a Windows service, Compact Edition must be hosted by the application using it. The 3.5 version includes support for ADO.NET Synchronization Services. SQL CE does not support ODBC connectivity, unlike SQL Server proper.

**Developer**

SQL Server Developer Edition includes the same features as SQL Server 2012 Enterprise Edition, but is limited by the license to be only used as a development and test system, and not as production server. This edition is available to download by students free of charge as a part of Microsoft's DreamSpark program.

**Embedded (SSEE)**

SQL Server 2005 Embedded Edition is a specially configured named instance of the SQL Server Express database engine which can be accessed only by certain Windows Services.

**Evaluation**

SQL Server Evaluation Edition, also known as the Trial Edition, has all the features of the Enterprise Edition, but is limited to 180 days, after which the tools will continue to run, but the server services will stop.

**Fast Track**

SQL Server Fast Track is specifically for enterprise-scale data warehousing storage and business intelligence processing, and runs on reference-architecture hardware that is optimized for Fast Track.

**LocalDB**

Introduced in SQL Server Express 2012, LocalDB is a minimal, on-demand, version of SQL Server that is designed for application developers. It can also be used as an embedded database.

**Parallel Data Warehouse (PDW)**

A massively parallel processing (MPP) SQL Server appliance optimized for large-scale data warehousing such as hundreds of terabytes.

**Datawarehouse Appliance Edition**

Pre-installed and configured as part of an appliance in partnership with Dell & HP base on the Fast Track architecture. This edition does not include SQL Server Integration Services, Analysis Services, or Reporting Services.

**2.4 Architecture**

The protocol layer implements the external interface to SQL Server. All operations that can be invoked on SQL Server are communicated to it via a Microsoft-defined format, called Tabular Data Stream (TDS). TDS is an application layer protocol, used to transfer data between a database server and a client. Initially designed and developed by Sybase Inc. for their Sybase SQL Server relational database engine in 1984, and later by Microsoft in Microsoft SQL Server, TDS packets can be encased in other physical transport dependent protocols, including TCP/IP, Named pipes, and Shared memory. Consequently, access to SQL Server is available over these protocols. In addition, the SQL Server API is also exposed over web services.

**2.5 Data storage**

Data storage is a database, which is a collection of tables with typed columns. SQL Server supports different data types, including primary types such as Integer, Float, Decimal, Char (including character strings), Varchar (variable length character strings), binary (for unstructured blobs of data), Text (for textual data) among others. The rounding of floats to integers uses either Symmetric Arithmetic Rounding or Symmetric Round Down (Fix) depending on arguments: SELECT Round(2.5, 0) gives 3.

Microsoft SQL Server also allows user-defined composite types (UDTs) to be defined and used. It also makes server statistics available as virtual tables and views (called Dynamic Management Views or DMVs). In addition to tables, a database can also contain other objects including views, stored procedures, indexes and constraints, along with a transaction log. A SQL Server database can contain a maximum of 231 objects, and can span multiple OS-level files with a maximum file size of 260 bytes (1 exabyte). The data in the database are stored in primary data files with an extension .mdf. Secondary data files, identified with a .ndf extension, are used to allow the data of a single database to be spread across more than one file, and optionally across more than one file system. Log files are identified with the .ldf extension.

Storage space allocated to a database is divided into sequentially numbered pages, each 8 KB in size. A page is the basic unit of I/O for SQL Server operations. A page is marked with a 96-byte header which stores metadata about the page including the page number, page type, free space on the page and the ID of the object that owns it. Page type defines the data contained in the page - data stored in the database, index, allocation map which holds information about how pages are allocated to tables and indexes, change map which holds information about the changes made to other pages since last backup or logging, or contain large data types such as image or text. While page is the basic unit of an I/O operation, space is actually managed in terms of an extent which consists of 8 pages. A database object can either span all 8 pages in an extent ("uniform extent") or share an extent with up to 7 more objects ("mixed extent"). A row in a database table cannot span more than one page, so is limited to 8 KB in size. However, if the data exceeds 8 KB and the row contains Varchar or Varbinary data, the data in those columns are moved to a new page (or possibly a sequence of pages, called an Allocation unit) and replaced with a pointer to the data.

For physical storage of a table, its rows are divided into a series of partitions (numbered 1 to n). The partition size is user defined; by default all rows are in a single partition. A table is split into multiple partitions in order to spread a database over a computer cluster. Rows in each partition are stored in either B-tree or heap structure. If the table has an associated, clustered index to allow fast retrieval of rows, the rows are stored in-order according to their index values, with a B-tree providing the index. The data is in the leaf node of the leaves, and other nodes storing the index values for the leaf data reachable from the respective nodes. If the index is non-clustered, the rows are not sorted according to the index keys. An indexed view has the same storage structure as an indexed table. A table without a clustered index is stored in an unordered heap structure. However, the table may have non-clustered indices to allow fast retrieval of rows. In some situations the heap structure has performance advantages over the clustered structure. Both heaps and B-trees can span multiple allocation units.

**2.5.1 Buffer management**

SQL Server buffers pages in RAM to minimize disc I/O. Any 8 KB page can be buffered in-memory, and the set of all pages currently buffered is called the buffer cache. The amount of memory available to SQL Server decides how many pages will be cached in memory. The buffer cache is managed by the Buffer Manager. Either reading from or writing to any page copies it to the buffer cache. Subsequent reads or writes are redirected to the in-memory copy, rather than the on-disc version. The page is updated on the disc by the Buffer Manager only if the in-memory cache has not been referenced for some time. While writing pages back to disc, asynchronous I/O is used whereby the I/O operation is done in a background thread so that other operations do not have to wait for the I/O operation to complete. Each page is written along with its checksum when it is written. When reading the page back, its checksum is computed again and matched with the stored version to ensure the page has not been damaged or tampered with in the meantime.

**2.5.2 Concurrency and locking**

SQL Server allows multiple clients to use the same database concurrently. As such, it needs to control concurrent access to shared data, to ensure data integrity—when multiple clients update the same data, or clients attempt to read data that is in the process of being changed by another client. SQL Server provides two modes of concurrency control: pessimistic concurrency and optimistic concurrency. When pessimistic concurrency control is being used, SQL Server controls concurrent access by using locks. Locks can be either shared or exclusive. Exclusive lock grants the user exclusive access to the data—no other user can access the data as long as the lock is held. Shared locks are used when some data is being read—multiple users can read from data locked with a shared lock, but not acquire an exclusive lock. The latter would have to wait for all shared locks to be released. Locks can be applied on different levels of granularity—on entire tables, pages, or even on a per-row basis on tables. For indexes, it can either be on the entire index or on index leaves. The level of granularity to be used is defined on a per-database basis by the database administrator. While a fine grained locking system allows more users to use the table or index simultaneously, it requires more resources. So it does not automatically turn into higher performing solution. SQL Server also includes two more lightweight mutual exclusion solutions—latches and spinlocks—which are less robust than locks but are less resource intensive. SQL Server uses them for DMVs and other resources that are usually not busy. SQL Server also monitors all worker threads that acquire locks to ensure that they do not end up in deadlocks—in case they do, SQL Server takes remedial measures, which in many cases is to kill one of the threads entangled in a deadlock and rollback the transaction it started. To implement locking, SQL Server contains the Lock Manager. The Lock Manager maintains an in-memory table that manages the database objects and locks, if any, on them along with other metadata about the lock. Access to any shared object is mediated by the lock manager, which either grants access to the resource or blocks it.

SQL Server also provides the optimistic concurrency control mechanism, which is similar to the multiversion concurrency control used in other databases. The mechanism allows a new version of a row to be created whenever the row is updated, as opposed to overwriting the row, i.e., a row is additionally identified by the ID of the transaction that created the version of the row. Both the old as well as the new versions of the row are stored and maintained, though the old versions are moved out of the database into a system database identified as Tempdb. When a row is in the process of being updated, any other requests are not blocked (unlike locking) but are executed on the older version of the row. If the other request is an update statement, it will result in two different versions of the rows—both of them will be stored by the database, identified by their respective transaction IDs.

**2.6 Data retrieval**

The main mode of retrieving data from an SQL Server database is querying for it. The query is expressed using a variant of SQL called T-SQL, a dialect Microsoft SQL Server shares with Sybase SQL Server due to its legacy. The query declaratively specifies what is to be retrieved. It is processed by the query processor, which figures out the sequence of steps that will be necessary to retrieve the requested data. The sequence of actions necessary to execute a query is called a query plan. There might be multiple ways to process the same query. For example, for a query that contains a join statement and a select statement, executing join on both the tables and then executing select on the results would give the same result as selecting from each table and then executing the join, but result in different execution plans. In such case, SQL Server chooses the plan that is expected to yield the results in the shortest possible time. This is called query optimization and is performed by the query processor itself.

SQL Server includes a cost-based query optimizer which tries to optimize on the cost, in terms of the resources it will take to execute the query. Given a query, then the query optimizer looks at the database schema, the database statistics and the system load at that time. It then decides which sequence to access the tables referred in the query, which sequence to execute the operations and what access method to be used to access the tables. For example, if the table has an associated index, whether the index should be used or not - if the index is on a column which is not unique for most of the columns (low "selectivity"), it might not be worthwhile to use the index to access the data. Finally, it decides whether to execute the query concurrently or not. While a concurrent execution is more costly in terms of total processor time, because the execution is actually split to different processors might mean it will execute faster. Once a query plan is generated for a query, it is temporarily cached. For further invocations of the same query, the cached plan is used. Unused plans are discarded after some time.

SQL Server also allows stored procedures to be defined. Stored procedures are parameterized T-SQL queries, that are stored in the server itself (and not issued by the client application as is the case with general queries). Stored procedures can accept values sent by the client as input parameters, and send back results as output parameters. They can call defined functions, and other stored procedures, including the same stored procedure (up to a set number of times). They can be selectively provided access to. Unlike other queries, stored procedures have an associated name, which is used at runtime to resolve into the actual queries. Also because the code need not be sent from the client every time (as it can be accessed by name), it reduces network traffic and somewhat improves performance. Execution plans for stored procedures are also cached as necessary.

**2.6.1 SQL CLR**

Microsoft SQL Server 2005 includes a component named SQL CLR ("Common Language Runtime") via which it integrates with .NET Framework. Unlike most other applications that use .NET Framework, SQL Server itself hosts the .NET Framework runtime, i.e., memory, threading and resource management requirements of .NET Framework are satisfied by SQLOS itself, rather than the underlying Windows operating system. SQLOS provides deadlock detection and resolution services for .NET code as well. With SQL CLR, stored procedures and triggers can be written in any managed .NET language, including C# and VB.NET. Managed code can also be used to define UDT's (user defined types), which can persist in the database. Managed code is compiled to CLI assemblies and after being verified for type safety, registered at the database. After that, they can be invoked like any other procedure. However, only a subset of the Base Class Library is available, when running code under SQL CLR. Most APIs relating to user interface functionality are not available.

When writing code for SQL CLR, data stored in SQL Server databases can be accessed using the ADO.NET APIs like any other managed application that accesses SQL Server data. However, doing that creates a new database session, different from the one in which the code is executing. To avoid this, SQL Server provides some enhancements to the ADO.NET provider that allows the connection to be redirected to the same session which already hosts the running code. Such connections are called context connections and are set by setting context connection parameter to true in the connection string. SQL Server also provides several other enhancements to the ADO.NET API, including classes to work with tabular data or a single row of data as well as classes to work with internal metadata about the data stored in the database. It also provides access to the XML features in SQL Server, including XQuery support. These enhancements are also available in T-SQL Procedures in consequence of the introduction of the new XML Datatype (query,value,nodes functions).

**2.7 Services**

SQL Server also includes an assortment of add-on services. While these are not essential for the operation of the database system, they provide value added services on top of the core database management system. These services either run as a part of some SQL Server component or out-of-process as Windows Service and presents their own API to control and interact with them.

**2.7.1. Service Broker**

Used inside an instance, programming environment. For cross instance applications, Service Broker communicates over TCP/IP and allows the different components to be synchronized together, via exchange of messages. The Service Broker, which runs as a part of the database engine, provides a reliable messaging and message queuing platform for SQL Server applications.

**2.7.2 Replication Services**

SQL Server Replication Services are used by SQL Server to replicate and synchronize database objects, either in entirety or a subset of the objects present, across replication agents, which might be other database servers across the network, or database caches on the client side. Replication follows a publisher/subscriber model, i.e., the changes are sent out by one database server ("publisher") and are received by others ("subscribers"). SQL Server supports three different types of replication:

**Transaction replication**

Each transaction made to the publisher database (master database) is synced out to subscribers, who update their databases with the transaction. Transactional replication synchronizes databases in near real time.

**Merge replication**

Changes made at both the publisher and subscriber databases are tracked, and periodically the changes are synchronized bi-directionally between the publisher and the subscribers. If the same data has been modified differently in both the publisher and the subscriber databases, synchronization will result in a conflict which has to be resolved - either manually or by using pre-defined policies. rowguid needs to be configured on a column if merge replication is configured.

**Snapshot replication**

Snapshot replication publishes a copy of the entire database (the then-snapshot of the data) and replicates out to the subscribers. Further changes to the snapshot are not tracked.

**2.7.3 Analysis Services**

SQL Server Analysis Services adds OLAP and data mining capabilities for SQL Server databases. The OLAP engine supports MOLAP, ROLAP and HOLAP storage modes for data. Analysis Services supports the XML for Analysis standard as the underlying communication protocol. The cube data can be accessed using MDX and LINQ queries. Data mining specific functionality is exposed via the DMX query language. Analysis Services includes various algorithms - Decision trees, clustering algorithm, Naive Bayes algorithm, time series analysis, sequence clustering algorithm, linear and logistic regression analysis, and neural networks - for use in data mining.

**2.7.4 Reporting Services**

SQL Server Reporting Services is a report generation environment for data gathered from SQL Server databases. It is administered via a web interface. Reporting services features a web services interface to support the development of custom reporting applications. Reports are created as RDL files.

Reports can be designed using recent versions of Microsoft Visual Studio (Visual Studio.NET 2003, 2005, and 2008) with Business Intelligence Development Studio, installed or with the included Report Builder. Once created, RDL files can be rendered in a variety of formats including Excel, PDF, CSV, XML, TIFF (and other image formats), and HTML Web Archive.

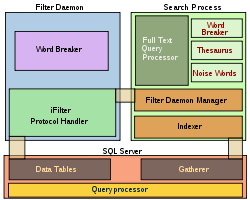
**2.7.5 Notification Services**

Originally introduced as a post-release add-on for SQL Server 2000, Notification Services was bundled as part of the Microsoft SQL Server platform for the first and only time with SQL Server 2005. SQL Server Notification Services is a mechanism for generating data-driven notifications, which are sent to Notification Services subscribers. A subscriber registers for a specific event or transaction (which is registered on the database server as a trigger); when the event occurs, Notification Services can use one of three methods to send a message to the subscriber informing about the occurrence of the event. These methods include SMTP, SOAP, or by writing to a file in the filesystem. Notification Services was discontinued by Microsoft with the release of SQL Server 2008 in August 2008, and is no longer an officially supported component of the SQL Server database platform.

**2.7.6 Integration Services**

SQL Server Integration Services (SSIS) provides ETL capabilities for SQL Server for data import, data integration and data warehousing needs. Integration Services includes GUI tools to build workflows such as extracting data from various sources, querying data, transforming data—including aggregation, de-duplication, de-/normalization and merging of data—and then exporting the transformed data into destination databases or files.

**2.7.7 Full Text Search Service**

[](http://en.wikipedia.org/wiki/File:SQL_Server_FTS.svg)

The SQL Server Full Text Search service architecture

SQL Server Full Text Search service is a specialized indexing and querying service for unstructured text stored in SQL Server databases. The full text search index can be created on any column with character based text data. It allows for words to be searched for in the text columns. While it can be performed with the SQL LIKE operator, using SQL Server Full Text Search service can be more efficient. Full allows for inexact matching of the source string, indicated by a Rank value which can range from 0 to 1000 - a higher rank means a more accurate match. It also allows linguistic matching ("inflectional search"), i.e., linguistic variants of a word (such as a verb in a different tense) will also be a match for a given word (but with a lower rank than an exact match). Proximity searches are also supported, i.e., if the words searched for do not occur in the sequence they are specified in the query but are near each other, they are also considered a match. T-SQL exposes special operators that can be used to access the FTS capabilities.

The Full Text Search engine is divided into two processes - the Filter Daemon process (msftefd.exe) and the Search process (msftesql.exe). These processes interact with the SQL Server. The Search process includes the indexer (that creates the full text indexes) and the full text query processor. The indexer scans through text columns in the database. It can also index through binary columns, and use iFilters to extract meaningful text from the binary blob (for example, when a Microsoft Word document is stored as an unstructured binary file in a database). The iFilters are hosted by the Filter Daemon process. Once the text is extracted, the Filter Daemon process breaks it up into a sequence of words and hands it over to the indexer. The indexer filters out noise words, i.e., words like A, And etc., which occur frequently and are not useful for search. With the remaining words, an inverted index is created, associating each word with the columns they were found in. SQL Server itself includes a Gatherer component that monitors changes to tables and invokes the indexer in case of updates.

When a full text query is received by the SQL Server query processor, it is handed over to the FTS query processor in the Search process. The FTS query processor breaks up the query into the constituent words, filters out the noise words, and uses an inbuilt thesaurus to find out the linguistic variants for each word. The words are then queried against the inverted index and a rank of their accurateness is computed. The results are returned to the client via the SQL Server process.

**2.7.8 SQLCMD**

SQLCMD is a command line application that comes with Microsoft SQL Server, and exposes the management features of SQL Server. It allows SQL queries to be written and executed from the command prompt. It can also act as a scripting language to create and run a set of SQL statements as a script. Such scripts are stored as a .sql file, and are used either for management of databases or to create the database schema during the deployment of a database.

SQLCMD was introduced with SQL Server 2005 and this continues with SQL Server 2012 and 2014. Its predecessor for earlier versions was OSQL and ISQL, which is functionally equivalent as it pertains to TSQL execution, and many of the command line parameters are identical, although SQLCMD adds extra versatility.

**2.7.9 Visual Studio**

Microsoft Visual Studio includes native support for data programming with Microsoft SQL Server. It can be used to write and debug code to be executed by SQL CLR. It also includes a data designer that can be used to graphically create, view or edit database schemas. Queries can be created either visually or using code. SSMS 2008 onwards, provides intellisense for SQL queries as well.

**2.7.10 SQL Server Management Studio**

SQL Server Management Studio is a GUI tool included with SQL Server 2005 and later for configuring, managing, and administering all components within Microsoft SQL Server. The tool includes both script editors and graphical tools that work with objects and features of the server. SQL Server Management Studio replaces Enterprise Manager as the primary management interface for Microsoft SQL Server since SQL Server 2005. A version of SQL Server Management Studio is also available for SQL Server Express Edition, for which it is known as SQL Server Management Studio Express (SSMSE).

A central feature of SQL Server Management Studio is the Object Explorer, which allows the user to browse, select, and act upon any of the objects within the server. It can be used to visually observe and analyze query plans and optimize the database performance, among others. SQL Server Management Studio can also be used to create a new database, alter any existing database schema by adding or modifying tables and indexes, or analyze performance. It includes the query windows which provide a GUI based interface to write and execute queries.

**2.7.11 Business Intelligence Development Studio**

Business Intelligence Development Studio (BIDS) is the IDE from Microsoft used for developing data analysis and Business Intelligence solutions utilizing the Microsoft SQL Server Analysis Services, Reporting Services and Integration Services. It is based on the Microsoft Visual Studio development environment but is customized with the SQL Server services-specific extensions and project types, including tools, controls and projects for reports (using Reporting Services), Cubes and data mining structures (using Analysis Services).

**2.8 Programmability**

**2.8.1 T-SQL**

T-SQL (Transact-SQL) is the Secondary means of programming and managing SQL Server. It exposes keywords for the operations that can be performed on SQL Server, including creating and altering database schemas, entering and editing data in the database as well as monitoring and managing the server itself. Client applications that consume data or manage the server will leverage SQL Server functionality by sending T-SQL queries and statements which are then processed by the server and results (or errors) returned to the client application. SQL Server allows it to be managed using T-SQL. For this it exposes read-only tables from which server statistics can be read. Management functionality is exposed via system-defined stored procedures which can be invoked from T-SQL queries to perform the management operation. It is also possible to create linked Server using T-SQL. Linked server allows operation to multiple server as one query.

**2.8.2 SQL Native Client (aka SNAC**)

SQL Native Client is the native client side data access library for Microsoft SQL Server, version 2005 onwards. It natively implements support for the SQL Server features including the Tabular Data Stream implementation, support for mirrored SQL Server databases, full support for all data types supported by SQL Server, asynchronous operations, query notifications, encryption support, as well as receiving multiple result sets in a single database session. SQL Native Client is used under the hood by SQL Server plug-ins for other data access technologies, including ADO or OLE DB. The SQL Native Client can also be directly used, bypassing the generic data access layers.

On November 28, 2011 a preview release of the SQL Server ODBC driver for Linux was released.

**3. ADO.NET Entity Framework**

**3.2 History**

The first version of Entity Framework (EFv1) was included with .NET Framework 3.5 Service Pack 1 and Visual Studio 2008 Service Pack 1, released on 11 August 2008. This version was widely criticized, even attracting a 'vote of no confidence' signed by approximately one thousand developers.

The second version of Entity Framework, named Entity Framework 4.0 (EFv4), was released as part of .NET 4.0 on 12 April 2010 and addressed many of the criticisms made of version 1.

A third version of Entity Framework, version 4.1, was released on April 12, 2011, with Code First support.

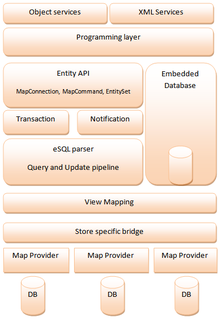
A refresh of version 4.1, named Entity Framework 4.1 Update 1, was released on July 25, 2011. It includes bug fixes and new supported types.

The version 4.3.1 was released on February 29, 2012. There were a few updates, like support for migration.

Version 5.0.0 was released on August 11, 2012 and is targeted at .NET framework 4.5. Also, this version is available for .Net framework 4, but without any runtime advantages over version 4.

Version 6.0 was released on October 17, 2013 and is now an open source project licensed under Apache License v2. Like ASP.NET MVC Framework, its source code is hosted at CodePlex using Git. This version has a number of improvements for code-first support.

**3.3 Architecture**

[](http://en.wikipedia.org/wiki/File:ANEF.PNG)

ADO.NET Entity Framework stack

The architecture of the ADO.NET Entity Framework, from the bottom up, consists of the following:

**Data source specific providers**, which abstract the ADO.NET interfaces to connect to the database when programming against the conceptual schema.

**Map provider**, a database-specific provider that translates the Entity SQL command tree into a query in the native SQL flavor of the database. It includes the Store-specific bridge, which is the component responsible for translating the generic command tree into a store-specific command tree.

**EDM parser and view mapping**, which takes the SDL specification of the data model and how it maps onto the underlying relational model and enables programming against the conceptual model. From the relational schema, it creates views of the data corresponding to the conceptual model. It aggregates information from multiple tables in order to aggregate them into an entity, and splits an update to an entity into multiple updates to whichever table(s) contributed to that entity.

**Query and update pipeline**, processes queries, filters and updates requests to convert them into canonical command trees which are then converted into store-specific queries by the map provider.

**Metadata services**, which handle all metadata related to entities, relationships and mappings.

**Transactions**, to integrate with transactional capabilities of the underlying store. If the underlying store does not support transactions, support for it needs to be implemented at this layer.

**Conceptual layer API**, the runtime that exposes the programming model for coding against the conceptual schema. It follows the ADO.NET pattern of using Connection objects to refer to the map provider, using Command objects to send the query, and returning EntityResultSets or EntitySets containing the result.

**Disconnected components**, which locally cache datasets and entity sets for using the ADO.NET Entity Framework in an occasionally connected environment.

**Embedded database**: ADO.NET Entity Framework includes a lightweight embedded database for client-side caching and querying of relational data.

**Design tools**, such as Mapping Designer, are also included with ADO.NET Entity Framework, which simplifies the job of mapping a conceptual schema to the relational schema and specifying which properties of an entity type correspond to which table in the database.

**Programming layer**, which exposes the EDM as programming constructs which can be consumed by programming languages.

**Object services**, automatically generate code for CLR classes that expose the same properties as an entity, thus enabling instantiation of entities as .NET objects.

**Web services**, which expose entities as web services.

**High-level services**, such as reporting services which work on entities rather than relational data.

**3.4 Entity Data Model**

The Entity data model (EDM) specifies the conceptual model (CSDL) of the data, using a modelling technique that is itself called Entity Data Model, an extended version of the Entity-Relationship model.The data model primarily describes the Entities and the Associations they participate in. The EDM schema is expressed in the Schema Definition Language (SDL), which is an application of XML. In addition, the mapping (MSL) of the elements of the conceptual schema (CSDL) to the storage schema (SSDL) must also be specified. The mapping specification is also expressed in XML.

Visual Studio also provides Entity Designer, for visual creation of the EDM and the mapping specification. The output of the tool is the XML file (\*.edmx) specifying the schema and the mapping. Edmx file contains EF metadata artifacts (CSDL/MSL/SSDL content). These 3 files (csdl, msl, ssdl) can also be created or edited by hand.

**3.4.1 Mapping**

Entity Data Model Wizard in Visual Studio initially generates a 1:1 (one to one) mapping between the database schema and the conceptual schema in most of the cases. In the relational schema, the elements are composed of the tables, with the primary and foreign keys gluing the related tables together. In contrast, the Entity Types define the conceptual schema of the data.

The entity types are an aggregation of multiple typed fields – each field maps to a certain column in the database - and can contain information from multiple physical tables. The entity types can be related to each other, independent of the relationships in the physical schema. Related entities are also exposed similarly – via a field whose name denotes the relation they are participating in and accessing which, instead of retrieving the value from some column in the database, traverses the relationship and returns the entity (or a collection of entities) it is related with.

Entity Types form the class of objects entities conform to, with the Entities being instances of the entity types. Entities represent individual objects which form a part of the problem being solved by the application and are indexed by a key. For example, converting the physical schema described above, we will have two entity types:

* *CustomerEntity*, which contains the customer's name from the Customers table, and the customer's address from the Contacts table.
* *OrderEntity*, which encapsulates the orders of a certain customer, retrieving it from the Orders table.

The logical schema and its mapping with the physical schema is represented as an **Entity Data Model** (EDM), specified as an XML file. ADO.NET Entity Framework uses the EDM to actually perform the mapping letting the application work with the entities, while internally abstracting the use of ADO.NET constructs like DataSet and RecordSet. ADO.NET Entity Framework performs the joins necessary to have entity reference information from multiple tables, or when a relationship is traversed. When an entity is updated, it traces back which table the information came from and issues SQL update statements to update the tables in which some data has been updated. ADO.NET Entity Framework uses eSQL, a derivative of SQL, to perform queries, set-theoretic operations, and updates on entities and their relationships. Queries in eSQL, if required, are then translated to the native SQL flavor of the underlying database.

Entity types and entity sets just form the logical EDM schema, and can be exposed as anything. ADO.NET Entity Framework includes Object Service that presents these entities as Objects with the elements and relationships exposed as properties. Thus Entity objects are just front-end to the instances of the EDM entity types, which lets Object Oriented languages access and use them. Similarly, other front-ends can be created, which expose the entities via web services (e.g., WCF Data Services) or XML which is used when entities are serialized for persistence storage or over-the-wire transfer.

**3.4.2 Entities**

Entities are instances of EntityTypes; they represent the individual instances of the objects (such as customer, orders) to which the information pertains. The identity of an entity is defined by the entity type it is an instance of; in that sense an entity type defines the class an entity belongs to and also defines what properties an entity will have. Properties describe some aspect of the entity by giving it a name and a type. The properties of an entity type in ADO.NET Entity Framework are fully typed, and are fully compatible with the type system used in a DBMS system, as well as the Common Type System of the .NET Framework. A property can be SimpleType, or ComplexType, and can be multi-valued as well. All EntityTypes belong to some namespace, and have an EntityKey property which uniquely identifies each instance of the entity type. The different property types are distinguished as follows:

* **SimpleType**, corresponds to primitive data types such as Integer, Characters and Floating Point numbers.
* **ComplexType**, is an aggregate of multiple properties of type SimpleType, or ComplexType. Unlike EntityTypes, however, ComplexTypes cannot have an EntityKey. In Entity Framework v1 ComplexTypes cannot be inherited.

All entity instances are housed in EntityContainers, which are per-project containers for entities. Each project has one or more named EntityContainers, which can reference entities across multiple namespaces and entity types. Multiple instances of one entity type can be stored in collections called EntitySets. One entity type can have multiple EntitySets.

|  |  |
| --- | --- |
| **EDM type** | **CLR type mapping** |
| Edm.Binary | Byte[] |
| Edm.Boolean | Boolean |
| Edm.Byte | Byte |
| Edm.DateTime | DateTime |
| Edm.DateTimeOffset | DateTimeOffset |
| Edm.Decimal | Decimal |
| Edm.Double | Double |
| Edm.Guid | Guid |
| Edm.Int16 | Int16 |
| Edm.Int32 | Int32 |
| Edm.Int64 | Int64 |
| Edm.SByte | SByte |
| Edm.Single | Single |
| Edm.String | String |
| Edm.Time | TimeSpan |

**3.4.3 Relationships**

Any two entity types can be related, by either an Association relation or a Containment relation. For example, a shipment is billed to a customer is an association whereas an order contains order details is a containment relation. A containment relation can also be used to model inheritance between entities. The relation between two entity types is specified by a Relationship Type, instances of which, called Relationships, relate entity instances. In future releases, other kinds of relationship types such as Composition, or Identification, may be introduced. Relationship types are characterized by their degree (arity) or the count of entity types they relate and their multiplicity. However, in the initial release of ADO.NET Entity Framework, relationships are limited to a binary (of degree two) bi-directional relationship. Multiplicity defines how many entity instances can be related together. Based on multiplicity, relationships can be either one-to-one, one-to-many, or many-to-many. Relationships between entities are named; the name is called a Role. It defines the purpose of the relationship.

A relationship type can also have an Operation or Action associated with it, which allows some action to be performed on an entity in the event of an action being performed on a related entity. A relationship can be specified to take an Action when some Operation is done on a related entity. For example, on deleting an entity that forms the part of a relation (the OnDelete operation) the actions that can be taken are:

* Cascade, which instructs to delete the relationship instance and all associated entity instances.
* None.

For association relationships, which can have different semantics at either ends, different actions can be specified for either end.

**3.4.4 Schema definition language**

ADO.NET Entity Framework uses an XML based Data Definition Language called Schema Definition Language (SDL) to define the EDM Schema. The SDL defines the SimpleTypes similar to the CTS primitive types, including String, Int32, Double, Decimal, Guid, and DateTime, among others. An Enumeration, which defines a map of primitive values and names, is also considered a simple type. Enumerations are unsupported in the current version of the framework. ComplexTypes are created from an aggregation of other types. A collection of properties of these types define an Entity Type. This definition can be written in EBNF grammar as:

EntityType ::= ENTITYTYPE entityTypeName [BASE entityTypeName]

[ABSTRACT true|false] KEY propertyName [, propertyName]\*

{(propertyName PropertyType [PropertyFacet]\*) +}

PropertyType ::= ((PrimitiveType [PrimitiveTypeFacets]\*)

| (complexTypeName) | RowType

PropertyFacet ::= ( [NULLABLE true | false] |

[DEFAULT defaultVal] | [MULTIPLICITY [ 1|\*] ] )

PropertyTypeFacet ::= MAXLENGTH | PRECISION | SCALE | UNICODE | FIXEDLENGTH | COLLATION

| DATETIMEKIND | PRESERVESECONDS

PrimitiveType ::= BINARY | STRING | BOOLEAN

| SINGLE | DOUBLE | DECIMAL | GUID

| BYTE | SBYTE | INT16 | INT32 | INT64

| DATETIME | DATETIMEOFFSET | TIME )

Facets are used to describe metadata of a property, such as whether it is nullable or has a default value, as also the cardinality of the property, i.e., whether the property is single valued or multi valued. A multiplicity of “1” denotes a single valued property; a “\*” means it is a multi-valued property. As an example, an entity can be denoted in SDL as:

**<ComplexType** Name="Addr"**>**

**<Property** Name="Street" Type="String" Nullable="false" **/>**

**<Property** Name="City" Type="String" Nullable="false" **/>**

**<Property** Name="Country" Type="String" Nullable="false" **/>**

**<Property** Name="PostalCode" Type="Int32" **/>**

**</ComplexType>**

**<EntityType** Name="Customer"**>**

**<Key>**

**<PropertyRef** Name="Email" **/>**

**</Key>**

**<Property** Name="Name" Type="String" **/>**

**<Property** Name="Email" Type="String" Nullable="false" **/>**

**<Property** Name="Address" Type="Addr" **/>**

**</EntityType>**

A relationship type is defined as specifying the end points and their multiplicities. For example, a one-to-many relationship between *Customer* and *Orders* can be defined as

**<Association** Name="CustomerAndOrders"**>**

**<End** Type="Customer" Multiplicity="1" **/>**

**<End** Type="Orders" Multiplicity="\*"**>**

**<OnDelete** Action="Cascade" **/>**

**</End>**

**</Association>**

**3.5 Querying data**

**3.5.1 Entity SQL**

ADO.NET Entity Framework uses a variant of the Structured Query Language, named Entity SQL, which is aimed at writing declarative queries and updates over entities and entity relationships – at the conceptual level. It differs from SQL in that it does not have explicit constructs for joins because the EDM is designed to abstract partitioning data across tables. Querying against the conceptual model is facilitated by EntityClient classes, which accepts an Entity SQL query. The query pipeline parses the Entity SQL query into a command tree, segregating the query across multiple tables, which is handed over to the EntityClient provider. Like ADO.NET data providers, an EntityClient provider is also initialized using a Connection object, which in addition to the usual parameters of data store and authentication info, requires the SDL schema and the mapping information. The EntityClient provider in turn then turns the Entity SQL command tree into an SQL query in the native flavor of the database. The execution of the query then returns an Entity SQL ResultSet, which is not limited to a tabular structure, unlike ADO.NET ResultSets.

Entity SQL enhances SQL by adding intrinsic support for:

* Types, as ADO.NET entities are fully typed.
* EntitySets, which are treated as collections of entities.
* Composability, which removes restrictions on where subqueries can be used.

**Entity SQL canonical functions**

Canonical functions are supported by all Entity Framework compliant data providers. They can be used in an Entity SQL query. Also, most of the extension methods in LINQ to Entities are translated to canonical functions. They are independent of any specific database. When ADO.NET data provider receives a function, it translates it to the desired SQL statement.

But not all DBMSs have equivalent functionality and a set of standard embedded functions. There are also differences in the accuracy of calculations. Therefore, not all canonical functions are supported for all databases, and not all canonical functions return the same results.

|  |  |
| --- | --- |
| **Group** | **Canonical functions** |
| Aggregate functions | Avg, BigCount, Count, Max, Min, StDev, StDevP, Sum, Var, VarP |
| Math functions | Abs, Ceiling, Floor, Power, Round, Truncate |
| String functions | Concat, Contains, EndsWith, IndexOf, Left, Length, LTrim, Replace, Reverse, Right, RTrim, Substring, StartsWith, ToLower, ToUpper, Trim |
| Date and Time functions | AddMicroseconds, AddMilliseconds, AddSeconds, AddMinutes, AddHours, AddNanoseconds, AddDays, AddYears, CreateDateTime, AddMonths, CreateDateTimeOffset, CreateTime, CurrentDateTime, CurrentDateTimeOffset, CurrentUtcDateTime, Day, DayOfYear, DiffNanoseconds, DiffMilliseconds, DiffMicroseconds, DiffSeconds, DiffMinutes, DiffHours, DiffDays, DiffMonths, DiffYears, GetTotalOffsetMinutes, Hour, Millisecond, Minute, Month, Second, TruncateTime, Year |
| Bitwise functions | BitWiseAnd, BitWiseNot, BitWiseOr, BitWiseXor |
| Other functions | NewGuid |

The LINQ to Entities provider allows LINQ to be used to query various RDBMS data sources. Several database server specific providers with Entity Framework support are available.

**3.5.2 Native SQL**

In the Entity Framework v4 new methods ExecuteStoreQuery() and ExecuteStoreCommand() were added to the class ObjectContext.

**3.5.3 Visualizers**

Visual Studio has a feature called Visualizer. A LINQ query written in Visual Studio can be viewed as Native SQL using a Visualizer during debug session. A Visualizer for LINQ to Entities (Object Query) targeting all RDBMS is available via VisualStudioGallery

**4. .NET Framework**

.

**4.1 History**

Microsoft started development of .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.

Version 3.0 of .NET Framework is included with Windows Server 2008 and Windows Vista. Version 3.5 is included with Windows 7 and Windows Server 2008 R2, and can also be installed on Windows XP and Windows Server 2003. Version 4.5 is included with Windows 8 and Windows Server 2012. Windows XP was delivered without any versions of .NET Framework

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

**Overview of .NET Framework release history**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Version number** | [**CLR**](http://en.wikipedia.org/wiki/Common_Language_Runtime) **version** | **Release date** | **Included in** | | | **Replaces** |
| **Development tool** | **Windows** | **Windows Server** |
| [1.0](http://en.wikipedia.org/wiki/.NET_Framework_version_history#.NET_Framework_1.0) | 1.0 | 2002-02-13 | [Visual Studio .NET](http://en.wikipedia.org/wiki/Visual_Studio_.NET) | N/A | N/A | N/A |
| [1.1](http://en.wikipedia.org/wiki/.NET_Framework_version_history#.NET_Framework_1.1) | 1.1 | 2003-04-24 | [Visual Studio .NET 2003](http://en.wikipedia.org/wiki/Visual_Studio_.NET_2003) | N/A | [2003](http://en.wikipedia.org/wiki/Windows_Server_2003) | 1.0 |
| [2.0](http://en.wikipedia.org/wiki/.NET_Framework_version_history#.NET_Framework_2.0) | 2.0 | 2005-11-07 | [Visual Studio 2005](http://en.wikipedia.org/wiki/Visual_Studio_2005) | N/A | [2003](http://en.wikipedia.org/wiki/Windows_Server_2003), [2003 R2](http://en.wikipedia.org/wiki/Windows_Server_2003_R2), [2008 SP2](http://en.wikipedia.org/wiki/Windows_Server_2008_SP2), [2008 R2 SP1](http://en.wikipedia.org/wiki/Windows_Server_2008_R2_SP1) | N/A |
| [3.0](http://en.wikipedia.org/wiki/.NET_Framework_version_history#.NET_Framework_3.0) | 2.0 | 2006-11-06 | [Expression Blend](http://en.wikipedia.org/wiki/Microsoft_Blend)[[a]](http://en.wikipedia.org/wiki/.NET_Framework#endnote_a1none) | [Vista](http://en.wikipedia.org/wiki/Windows_Vista) | [2008 SP2](http://en.wikipedia.org/wiki/Windows_Server_2008_SP2), [2008 R2 SP1](http://en.wikipedia.org/wiki/Windows_Server_2008_R2_SP1) | 2.0 |
| [3.5](http://en.wikipedia.org/wiki/.NET_Framework_version_history#.NET_Framework_3.5) | 2.0 | 2007-11-19 | [Visual Studio 2008](http://en.wikipedia.org/wiki/Visual_Studio_2008) | [7](http://en.wikipedia.org/wiki/Windows_7), [8](http://en.wikipedia.org/wiki/Windows_8),[[b]](http://en.wikipedia.org/wiki/.NET_Framework#endnote_b1none) [8.1](http://en.wikipedia.org/wiki/Windows_8.1)[[b]](http://en.wikipedia.org/wiki/.NET_Framework#endnote_b2none) | [2008 R2 SP1](http://en.wikipedia.org/wiki/Windows_Server_2008_R2_SP1) | 2.0, 3.0 |
| [4.0](http://en.wikipedia.org/wiki/.NET_Framework_version_history#.NET_Framework_4) | 4 | 2010-04-12 | [Visual Studio 2010](http://en.wikipedia.org/wiki/Visual_Studio_2010) | N/A | N/A | N/A |
| [4.5](http://en.wikipedia.org/wiki/.NET_Framework_version_history#.NET_Framework_4.5) | 4 | 2012-08-15 | [Visual Studio 2012](http://en.wikipedia.org/wiki/Visual_Studio_2012) | [8](http://en.wikipedia.org/wiki/Windows_8) | [2012](http://en.wikipedia.org/wiki/Windows_Server_2012) | 4.0 |
| [4.5.1](http://en.wikipedia.org/wiki/.NET_Framework_version_history#.NET_Framework_4.5.1) | 4 | 2013-10-17 | [Visual Studio 2013](http://en.wikipedia.org/wiki/Visual_Studio_2013) | [8.1](http://en.wikipedia.org/wiki/Windows_8.1) | [2012 R2](http://en.wikipedia.org/wiki/Windows_Server_2012_R2) | 4.0, 4.5 |
| [4.5.2](http://en.wikipedia.org/wiki/.NET_Framework_version_history#.NET_Framework_4.5.2) | 4 | 2014-05-05 | N/A | N/A | N/A | 4.0, 4.5, 4.5.1 |

**Notes:**

a.[**^**](http://en.wikipedia.org/wiki/.NET_Framework#ref_a1none) [Expression Blend](http://en.wikipedia.org/wiki/Microsoft_Blend) only covers the [Windows Presentation Foundation](http://en.wikipedia.org/wiki/Windows_Presentation_Foundation) part of .NET Framework 3.0.

b.[**^**](http://en.wikipedia.org/wiki/.NET_Framework#ref_b1none) [**^**](http://en.wikipedia.org/wiki/.NET_Framework#ref_b2none) .NET Framework 3.5 is not installed along with Windows 8 or Windows 8.1. It must be installed either from a Windows installation media or from the [Internet](http://en.wikipedia.org/wiki/Internet). [Control Panel](http://en.wikipedia.org/wiki/Control_Panel_(Windows)) always attempts the latter

**4.2 Design**

**Interoperability**

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

**Common Language Runtime engine**

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

**Language independence**

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

**Framework Class Library**

Framework Class Library (FCL) is a library of functionality available to all languages using .NET Framework. FCL provides classes that encapsulate a number of common functions, including file reading and writing, graphic rendering, database interaction, XML document manipulation, and so on. It consists of classes, interfaces of reusable types that integrates CLR.

**Simplified deployment**

.NET Framework includes design features and tools which 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.

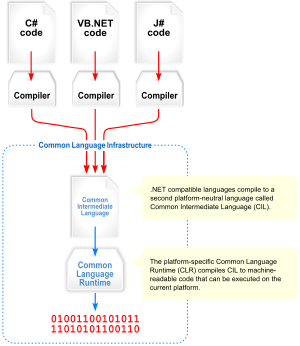
**Security**

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

**Portability**

While Microsoft has never implemented the full framework on any system except Microsoft Windows, it has engineered the framework to be platform-agnostic, and cross-platform implementations are available for other operating systems (see Silverlight and § Alternative implementations). Microsoft submitted the specifications for CLI (which includes the core class libraries, CTS, and the Common Intermediate Language), C#, and C++/CLI to both ECMA and ISO, making them available as official standards. This makes it possible for third parties to create compatible implementations of the framework and its languages on other platforms.

**4.3 Architecture**

[](http://en.wikipedia.org/wiki/File:Overview_of_the_Common_Language_Infrastructure.svg)

Visual overview of the Common Language Infrastructure (CLI)

**4.3.1 Common Language Infrastructure (CLI)**

The purpose of 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 .NET Framework within the scope of 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 CLI is CLR.

Common Intermediate Language (CIL) code is housed in CLI assemblies. As mandated by the specification, assemblies are stored in Portable Executable (PE) format, common on 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. Assemblies are considered equivalent if they share the same complete name, excluding the revision of the version number. A private key can also be used by the creator of the assembly for strong naming. The public key token identifies which private key an assembly is signed with. Only the creator of the keypair (typically .NET developer signing the assembly) can sign assemblies that have the same strong name as a previous version assembly, since the creator is in possession of the private key. Strong naming is required to add assemblies to Global Assembly Cache.

**4.3.2 Security**

.NET has its own security mechanism with two general features: Code Access Security (CAS), and validation and verification. CAS 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). CAS uses evidence to determine the permissions granted to the code. Other code can demand that calling code be granted a specified permission. The demand causes 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.

**4.3.3 Class library**

.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. .NET class libraries are available to all CLI compliant languages. .NET Framework class library is divided into two parts: FCL and Base Class Library (BCL).

FCL includes a small subset of the entire class library and is the core set of classes that serve as the basic API of CLR. Classes in mscorlib.dll and some classes in System.dll and System.core.dll are part of FCL. FCL classes are available in .NET Framework as well as its alternative implementations including .NET Compact Framework, Microsoft Silverlight and Mono.

BCL is a superset of FCL 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. BCL is much larger in scope than standard libraries for languages like C++, and comparable in scope to standard libraries of Java.

**4.3.4 Memory management**

CLR frees the developer from the burden of managing memory (allocating and freeing up when done); it handles memory management itself by detecting when memory can be safely freed. Instantiations of .NET types (objects) are allocated from the managed heap; a pool of memory managed by 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. When there is no reference to an object, and it cannot be reached or used, it becomes garbage, eligible for collection. .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.

.NET Garbage Collector (GC) is a non-deterministic, compacting, mark-and-sweep garbage collector. 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, 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. When 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 CLI 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. This is the mark phase. 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. Any reference to an object invalidated by moving the object is updated by GC to reflect the new location.The application is resumed after the garbage collection is over.

GC used by .NET Framework is also generational. 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. .NET Framework uses up to Generation 2 objects. 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 longer lifetime than newer objects. Thus, by eliminating 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.

**4.4 Standardization and licensing**

In August 2000, Microsoft, Hewlett-Packard, and Intel worked to standardize CLI and C#. By December 2001, both were ratified ECMA standards. ISO followed in April 2003. The current version of ISO standards are ISO/IEC 23271:2012 and ISO/IEC 23270:2006.

While Microsoft and their partners hold patents for 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 did not apply for the part of .NET Framework not covered by ECMA/ISO standards, which included Windows Forms, ADO.NET, and ASP.NET. Patents that Microsoft holds in these areas may have deterred non-Microsoft implementations of the full framework.

On 3 October 2007, Microsoft announced that much of the source code for .NET Framework Framework 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. 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 shared-source Microsoft Reference License means this source code was made available for debugging purpose only, primarily to support integrated debugging of FCL in Visual Studio.

On 12 November 2014, Microsoft announced it would open source .NET Core. The effort will include cross-platform support for .NET, the source release of Microsoft's CoreCLR implementation, source for the "entire [...] library stack" for .NET Core, and the adoption of a conventional ("bazaar"-like) open source development model under the stewardship of the .NET Foundation. Miguel de Icaza describes .NET Core as a "redesigned version of .NET that is based on the simplified version of the class libraries",and Microsoft's Immo Landwerth explained that .NET Core would be "the foundation of all future .NET platforms".

At the time of the announcement, the initial release of the .NET Core project had been seeded with a subset of the libraries' source code and coincided with the relicensing of Microsoft's existing .NET reference source away from the restrictions of the Microsoft Reference License. Both projects are made available under the MIT License. Landwerth explained that one of the disadvantages of the previously selected shared source license was that it "made Rotor [the Microsoft reference implementation] a non-starter" as a community-developed open source project because it did not meet the criteria of an OSI-approved license.

Microsoft also produced an update to its patent grants, which further extends the scope beyond its previous pledges. Whereas before projects like Mono existed in a legal grey area because Microsoft's earlier grants applied only to the technology in "covered specifications", including strictly the 4th editions each of ECMA-334 and ECMA-335, the new patent promise places no ceiling on the specification version and even extends to any .NET runtime technologies documented on MSDN that haven't been formally specified by the ECMA group, if a project chooses to implement them. This permits Mono and other projects to maintain feature parity with modern .NET features that have been introduced since the 4th edition was published without being at risk of patent litigation over the implementation of those features. The new grant does maintain the restriction that any implementation must maintain minimum compliance with the mandatory parts of the CLI specification.

Microsoft's press release highlights that the cross-platform commitment would now allow for a fully open source server-side .NET stack. However, Microsoft does not plan to release the source for Windows Presentation Foundation or Windows Forms.

**4.5 Alternative implementations**

.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." It is more difficult to develop alternatives to FCL, which is not described by an open standard and may be subject to copyright restrictions. Additionally, parts of FCL 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.

.NET Micro Framework is a .NET platform for extremely resource-constrained devices. It includes a small version of CLR and supports development in C# (though some developers were able to use VB.NET, albeit with an amount of hacking, and with limited functionalities) and debugging (in an emulator or on hardware), both using Microsoft Visual Studio. It also features a subset of .NET Framework Class Library (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 implementation of CLI and FCL, and provides additional functionality. It is dual-licensed under free software and proprietary software licenses. It includes support for ASP.NET, ADO.NET, and Windows Forms libraries for a wide range of architectures and operating systems. It also includes C# and VB.NET compilers.

Portable.NET (part of DotGNU) provides an implementation of CLI, portions of FCL, and a C# compiler. It supports a variety of CPUs and operating systems.

Microsoft Shared Source Common Language Infrastructure is a non-free implementation of CLR. However, the last version only runs on Microsoft Windows XP SP2, and was not updated since 2006, therefore it does not contain all features of version 2.0 of .NET Framework.

CrossNet is an implementation of CLI and portions of FCL. It is free software using the open source MIT License.

**4.5.1 Performance**

The garbage collector, which is integrated into the environment, can introduce unanticipated delays of execution over which the developer has little direct control. "In large applications, the number of objects that the garbage collector needs to deal with can become very large, which means it can take a very long time to visit and rearrange all of them."

.NET Framework provides support for calling Streaming SIMD Extensions (SSE) via managed code from April 2014 in Visual Studio 2013 Update 2. However, Mono has provided support for SIMD Extensions as of version 2.2 within the Mono.Simd namespace; before. Mono's lead developer Miguel de Icaza has expressed hope that this SIMD support will be adopted by CLR's ECMA standard. 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.

**4.5.2 Security**

Unobfuscated managed CIL bytecode can often be easier to reverse-engineer than native code. NET decompiler programs enable developers with no reverse-engineering skills to view the source code behind unobfuscated .NET assemblies (DLL/EXE). In contrast, applications built with Visual C++ are much harder to reverse-engineer and source code is almost never produced successfully, mainly due to compiler optimizations and lack of reflection.One concern is over possible loss of trade secrets and the bypassing of license control mechanisms. To mitigate this, Microsoft has included Dotfuscator Community Edition with Visual Studio .NET since 2002. Third-party obfuscation tools are also available from vendors such as vmware, V.i. Labs, Xenocode, Red Gate Software. Method-level encryption tools for .NET code are available from vendors such as SafeNet.