**A Complete Guide to Microsoft .NET Framework**

**.NET Framework**

Famous technology companies, as well as top Microsoft apps, are known to have been built with .NET and Windows CE technologies. .NET programming platform (with implementations such as .**NET Framework, .NET Core, Xamarin, Mono, Unity, or Godot**) allows developers to program in a single codebase, and create apps that run on Windows, Mac OS X, and Linux.

The .NET Framework is an open-source Microsoft development framework that offers a wide range of tools for building web and mobile apps. There are two .NET implementations used for creating applications: .NET Core and the .NET Framework. In this article, we will be focusing on the latter.

Developing web and mobile apps is our great passion, and we strive to deliver compelling projects. Using the .NET Framework, we have successfully deployed several [products](https://jelvix.com/technologies/net-development), and continue to create websites and apps relying on the [best software architecture](https://jelvix.com/technologies).

According to a KBVResearch report, the Global Application market is expected to reach [$27.3 billion](https://www.kbvresearch.com/application-server-market/) by 2025, with the highest sales in Asia-Pacific and Greater China. The prognosis seems fairly reasonable: there has never been a time for the application software market to enjoy as much money as today. The demand for Web services and peer-to-peer [mobile applications](https://jelvix.com/services/mobile-app-development) is extremely high, but the extent this demand will mark the advent of a new era for software technology is still to be seen.

Nevertheless, the emerging market opportunities and striking innovations in software development will help developers reach a wider audience as well as deliver large-scale products much faster. Meanwhile, our team will continue to monitor market dynamics to keep up-to-date and absorb relevant developments.

**What is the .NET Framework?**In the olden days, Microsoft programmers struggled with COM and multilayered architecture, and [peer-to-peer technology](https://www.encyclopedia.com/economics/encyclopedias-almanacs-transcripts-and-maps/peer-peer-technology-p2p) was considered a promising new era that was supposed to provide cutting-edge programming solutions, including Remoting, .NET Networking, encryption, etc.

Today, [.NET Framework](https://searchwindevelopment.techtarget.com/definition/NET) is essentially linked to Web Service and peer-to-peer applications creation, which are extremely difficult to implement right. With the variety of programming languages and libraries, it is possible to develop Web Services that are responsive and adaptive. That said, it is important to note that you will need a thorough understanding of .NET concepts (assemblies and types) to create and deploy applications that run across all types of devices.

So to get back to our earlier point about the original implementation of .NET – a framework that has revolutionized an open-source approach to software engineering – we’d better start off with the basics. The word “.NET” was chosen because it mirrored the domain suffix of (at that time) every ISP, so it was intended to remind users that “web-enabling your software” was the core goal of this project.

The term “framework” stands for APIs, and a shared library of prewritten code and routines, used to handle the lower-level coding for devs (e.g., parsing XML). That open-source library of predefined code is called the [Framework Class Library (FCL)](https://docs.microsoft.com/en-us/dotnet/standard/class-library-overview).

Like its competitor Java, .NET is an intermediate bytecode language that requires a runtime interpreter to implement .NET compilers into Common Intermediate Language (IL). IL runs on [Common Language Runtime (CLR)](https://www.geeksforgeeks.org/common-language-runtime-clr-in-c-sharp/), a platform that allows libraries and programming languages to work together; it provides services such as code compilation, security, database storage, and management.

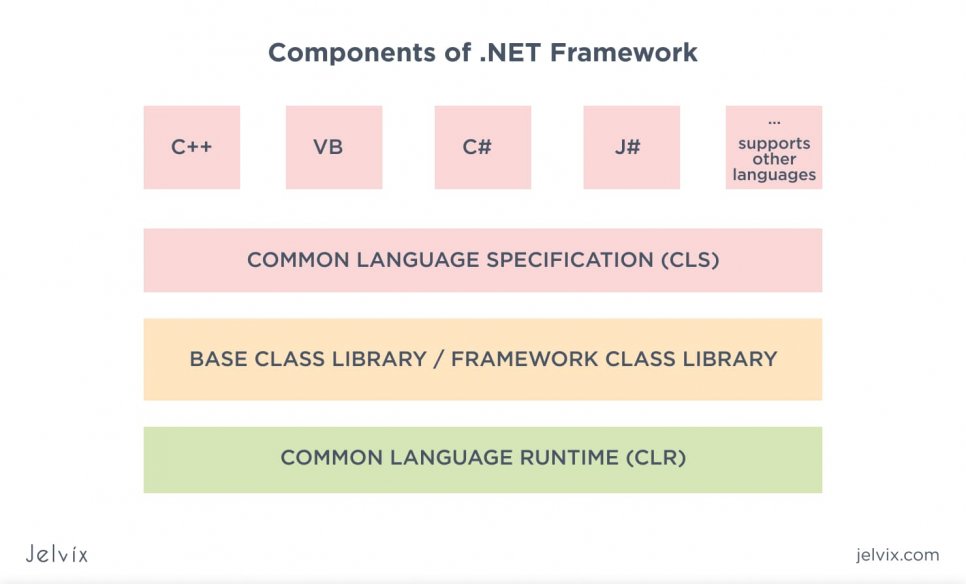
Therefore, we may conclude that the .NET Framework represents a set of pre-built classes that anyone can use to execute programs in an efficient manner. At the same time the runtime environment provides the basis for managing and deploying these programs easily.

There are many advantages of the .NET Framework, but the foremost benefit is [language interoperability](https://stackoverflow.com/questions/17242590/what-is-language-interoperability-basic-concept-in-net-framework). Because of CLR, languages like C#, C++, Visual Basic, F#, JS, and many others can be called to create fast and easily accessible applications.

Another important .NET implementation is [Mono](https://www.mono-project.com/docs/about-mono/dotnet-integration/). The original purpose behind this project was to create a compatible software framework and run .NET applications cross-platform (on Linux,  Unix, Mac OS X, Solaris, and even Windows).

Each implementation of .NET consists of the following major components:

* Common Language Runtime (CLR) for the .NET Framework (runs sites and apps on Windows);
* Core CLR for .NET Core (runs web services apps on Windows, Linux, macOS, etc.);
* Class Library;
* additional frameworks (if needed);
* development tools.

[](https://jelvix.com/wp-content/uploads/2020/07/components-of.net-framework.jpg)

**Advantages of using the .NET Framework: The advantages** of using the .NET Framework for application development are as follows:

* shared code libraries;
* no “writing from scratch”;
* code access security;
* open-source (free on [GitHub](https://github.com/dotnet));
* in-built UI controls;
* flexible/ highly demand apps and services;
* one of the top [Microsoft software frameworks.](https://pypl.github.io/PYPL.html)

**What is the .NET Framework used for?** So why is it considered one of the [top development frameworks](https://dottutorials.net/stats-surveys-about-net-core-future-2020/)? First and foremost, the biggest problem with software development is complexity. And this is what leads to buggy systems and long development cycles; this is why we need frameworks and other programming tools to solve this very problem of software complexity.

One of the most significant things about the .NET Framework is that it allows various versions of the CLR to exist on the same device. Further, coders have access to source code thanks to the open-source licenses; in computing, anyone who discloses computer software [should also release source code](https://www.linuxfoundation.org/resources/open-source-guides/using-open-source-code/) for that software. For example, the [C# compiler Roslyn](https://github.com/dotnet/roslyn) was made open source by Microsoft under the**Apache License**, and now is available for all users on GitHub.

What is more, with extensive Class Library, developers have straightforward access to simplifying objects. So, instead of wasting time on low-level optimizations, they use compilers that require little or no support. And these are only a few little examples of the commendable .NET features. The Framework also encompasses such things as the Common Type System (CTS) and the Common Type Specification (CTS).

These standards indicate how certain types are represented in computer memory. Thus, another key aspect that distinguishes the considerable benefit of using the full the .NET Framework is that it is language agnostic. The way devs program with it does not change depending on the language they are using. This is a huge productivity benefit as it creates a level playing field for all Windows developers.

Below you can find the .NET Framework objectives within the potential benefits and values list.

* **Platform Independence.**The .NET Framework is used to implement any software on multiple computing platforms. Most .NET projects are platform-independent applications that run under Windows.
* **Easy Development.**The Framework provides you with all the necessary resources for smooth application deployment. Features like language interoperability and extended Class Library allow programmers to quickly and easily produce software by combining their source code with Framework Libraries.
* **The Four “ities”**. The four “ities” are Reliability, Compatibility, Scalability, and Security. The .NET Framework is a highly *scalable* and *reliable*platform as it secures its applications by assigning access rights to specific users. What is more, the inbuilt security mechanism lets programmers “sign” the code before installing it into the system. The .NET Framework has also maximized its backward-compatibility; applications compiled for the earlier versions of .NET should now compile on the later versions.
* **High Performance.**Speed is the problem with most development circuits; Thanks to .NET Just-In-Time compiler**,**which performs a speedy and smooth code execution, the machine code can be obtained in a more efficient way.

**Why do I have it on my PC?**

Typically, the latest versions of the .NET Framework “come in package” with each version of Windows. But it wasn’t always like that. Windows used to include the newest version of .NET in each release of the operating system. The newer versions were designed to be backward-compatible, but things didn’t go as planned. Many applications did not work with them.

Below, you can see three ways of how any particular version of the .NET Framework would get installed on your PC:

* A particular version of the .NET Framework comes with your version of Windows;
* you are using an application that requires a particular version of the .NET Framework (the app has installed it by default);
* some applications send you a link to install the .NET Framework.

Today, we see a completely different picture in modern versions of Windows. Two significant developments occurred during **the old Windows Vista days** with respect to providing upgrades to the .NET Framework. First, Windows released [the .NET Framework 3.5](https://www.microsoft.com/en-us/download/details.aspx?id=21) that combined components from versions 2 and 3. Second, further updates to the Framework **started being delivered through Windows Update.**

These two factors have put questioning tech’s compatibility to an end – there was no longer a need for developers to be concerned about users performing additional installations.

The latest 4.7 version of the .NET Framework supports Windows Server 2016 and Windows 10. The framework is backward-compatible with its previous versions (the .NET Framework 4, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, and 4.7.1), and is available on Windows Update and Windows Server Update Service (WSUS).

—————————————————————————————————————-

*Jelvix team recommends you to avoid installing any particular version of the .NET Framework. The thing is that even if you already have one installed, there is always a slight chance of patch installation failure. If you have an older operating system, you do not need the .NET Framework, but if you have the latest shareware (written by professional companies), you might need it.*

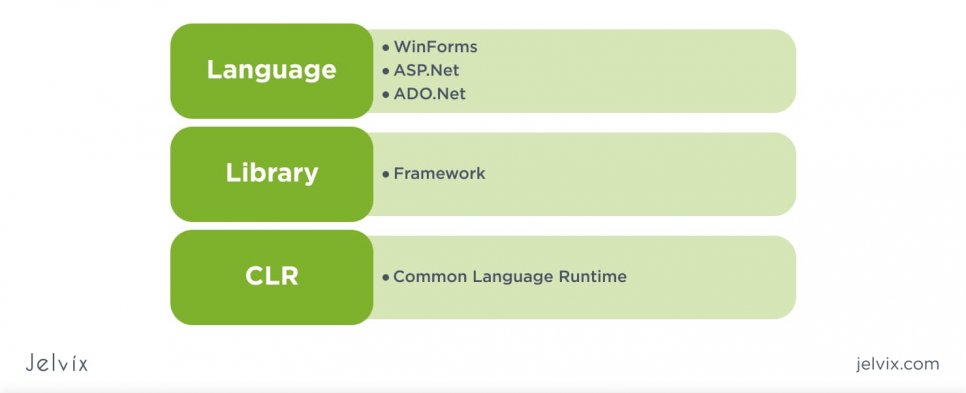
*—————————————————————————————————————-*

Most commonly, many small applications ask for a particular version to be installed, but it’s unnecessary. Since .NET packages are backward-compatible, apps that require you to download the .[NET Framework version 2.0](https://www.microsoft.com/en-us/download/details.aspx?id=1639" \t "_blank) are able to handle what’s packaged into the .[NET Framework 4.0.](https://www.microsoft.com/downloads/en/details.aspx?FamilyID=9cfb2d51-5ff4-4491-b0e5-b386f32c0992" \t "_blank)It is also important to make sure that your Windows is up-to-date with the latest Microsoft Windows update.

The only small downside is that .NET installations often require a lot of free hard disk space. In particular, the 4.0 version of .NET for standard 32-bit Windows systems needs 850 MB of space available on your Windows drive.

In contrast, a 64-bit Windows system needs 2 GB, and Windows is not always clear about whether you have enough space on the existing partition to install the Framework. In this case, check the amount of free space on your hard drive, and clear out the extra cruft if needed.

**The .NET Framework architecture in detail**

The diagram below illustrates the basic architecture of the .NET Framework.  
[](https://jelvix.com/wp-content/uploads/2020/07/language-library-crl.jpg)

As we mentioned before, the .NET Framework architecture consists of the following components:

* Common Language Runtime (CLR)
* Class Library
* Multiple app models

Now, let us proceed to consider the nature of these components.

**Common Language Runtime (CLR)**

This customary language runtime, developed for the .NET Framework, handles running apps and manages other system services, such as memory (garbage collection), security, exception handling, etc.

CLR key features:

**Exception Handling.** This is a process that helps to ensure there are no runtime errors (e.g., SQLException) in the code. Let’s see the examples of exceptions:

* An app tries to open a file with an appropriate app, but the file is not accessible or does not open.
* An app tries to make a server request (fetch) some records from a database, but the specified database is not valid.

**Garbage Collection.** The term describes the process of managing objects that are no longer in use to release unused memory of an app. Examples of garbage collection are:

* There is no longer a need to use a particular filehandle. After the application has finished working with a file, a filehandle will no longer be needed.
* Connection to the database is no longer required. After the application has handled all the database tasks, the connection must be closed, and all the allocated resources must be freed.

Below are the two different means a programming language is implemented by:

1. **Compiler**. This is a program that converts source code into machine code; every programming language has its own compiler. Accordingly, there is CSC for C# and VBC for VB.Net languages.
2. **Interpreter**. Contrary to the compiler, an interpreter does not convert the program into the machine language at once but rather concentrates on a single line at a time.

**Class Library**

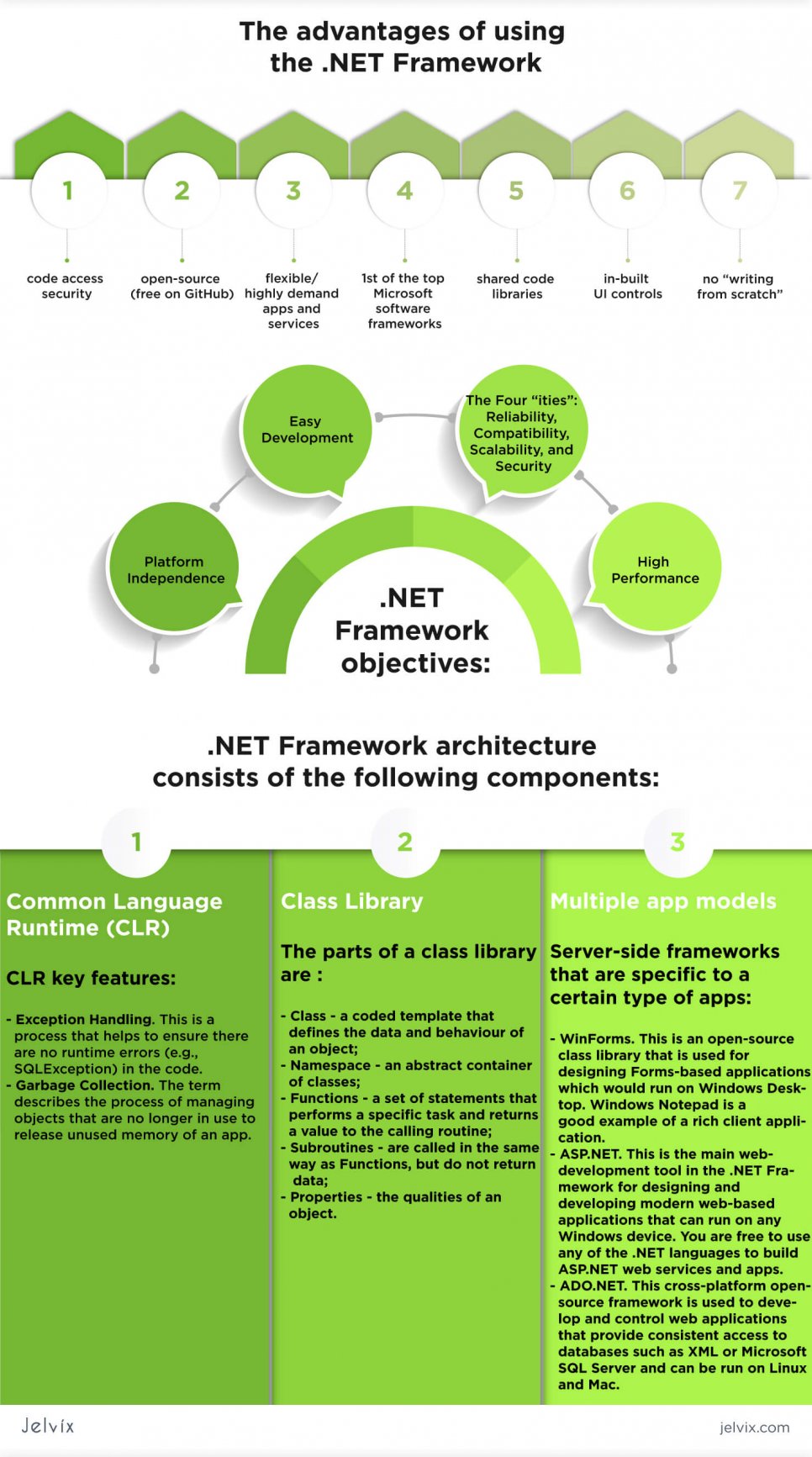
The .NET Framework includes subsets of reusable code (Class Libraries) that can be called by any .NET implementation to perform a specific task.

For instance, the Framework Class Library APIs provide file-level operations, as well as methods to read and write from the files. So, there is a method that can be used to read the text from a file. Similarly, there is a method to write text to a file.

Most of the methods are organized by either the System.\* or Microsoft.\* namespaces. As opposed to a class library, which is a concrete collection of classes, namespaces represent a notional separation of classes; classes contain data members and functions, whereas namespaces are used to differentiate these functions.

The parts of a class library are :

* **Class**– a coded template that defines the data and behaviour of an object;
* **Namespace** – an abstract container of classes;
* **Functions** – a set of statements that performs a specific task and returns a value to the calling routine;
* **Subroutines** – are called in the same way as Functions, but do not return data;
* **Properties** – the qualities of an object.

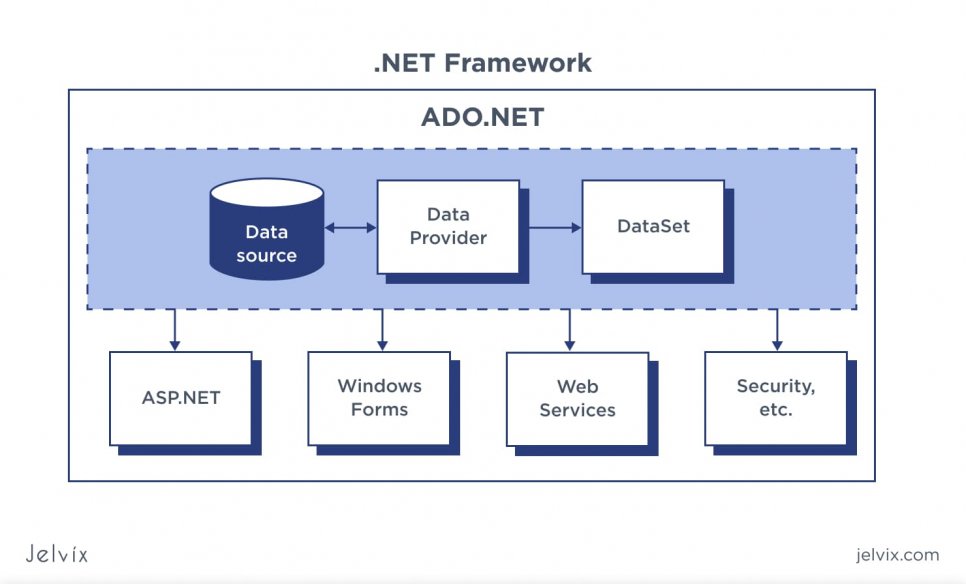
[](https://jelvix.com/wp-content/uploads/2020/07/overview-of-the-framework-1-scaled.jpg)

**?**Let's consider[best PHP Frameworks for Web Development in 2020](https://jelvix.com/blog/best-php-frameworks)

**App Models**

The .NET Framework provides app models for building astounding applications. The following sections illustrate server-side frameworks that are specific to a certain type of apps.

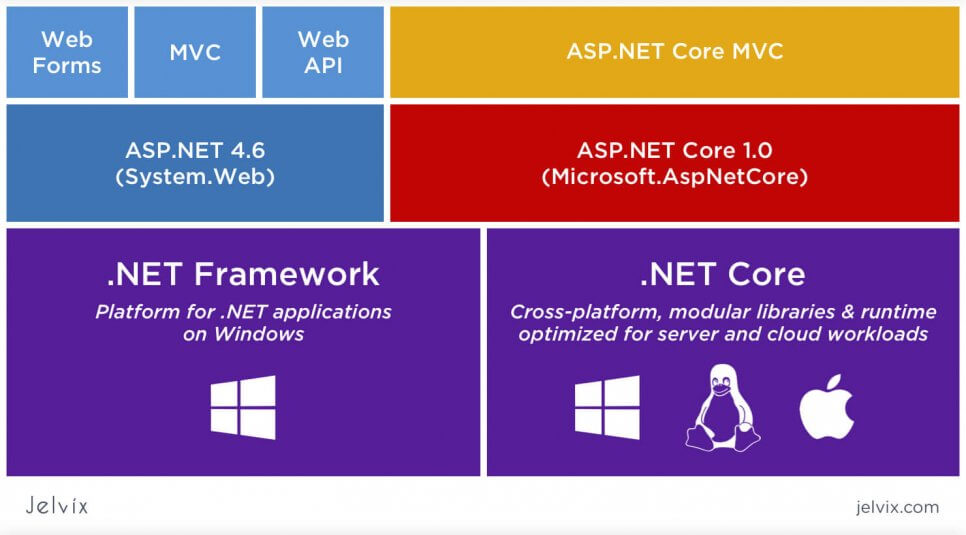
* **WinForms**. This is an open-source class library that is used for designing  Forms-based applications, which would run on Windows Desktop. Windows [Notepad](https://www.microsoft.com/en-us/p/windows-notepad/9msmlrh6lzf3?activetab=pivot:overviewtab) is a good example of a rich client application.
* [**ASP.NET**](https://dotnet.microsoft.com/apps/aspnet). This is the main web-development tool in the .NET Framework for designing and developing modern web-based applications that can run on any Windows device. You are free to use any of the .NET languages to build ASP.NET web services and apps.
* [**ADO.NET.**](https://www.tutorialspoint.com/asp.net/asp.net_ado_net.htm)This cross-platform open-source framework is used to develop and control web applications that provide consistent access to databases such as XML or Microsoft[SQL](https://www.guru99.com/sql.html)Server and can be run on Linux and Mac.

[](https://jelvix.com/wp-content/uploads/2020/07/data-source.jpg)

**ASP.NET vs ASP.NET Core**

If you are considering building software for PC or a cross-platform software, you most likely have heard of ASP.NET vs ASP.NET Core. Both derive from .[NET](https://dotnet.microsoft.com/" \t "_blank), a Microsoft-built framework compatible with C# and C++, but also with other less popular languages like F# and VB.NET.

The framework is available both under a Microsoft license and open-source. It used to be exclusive to PC development only but lately became applicable to OS as well. The reason for using .NET is in its simplicity. You can quickly comply with the outfit, use library components, and manage framework classes.

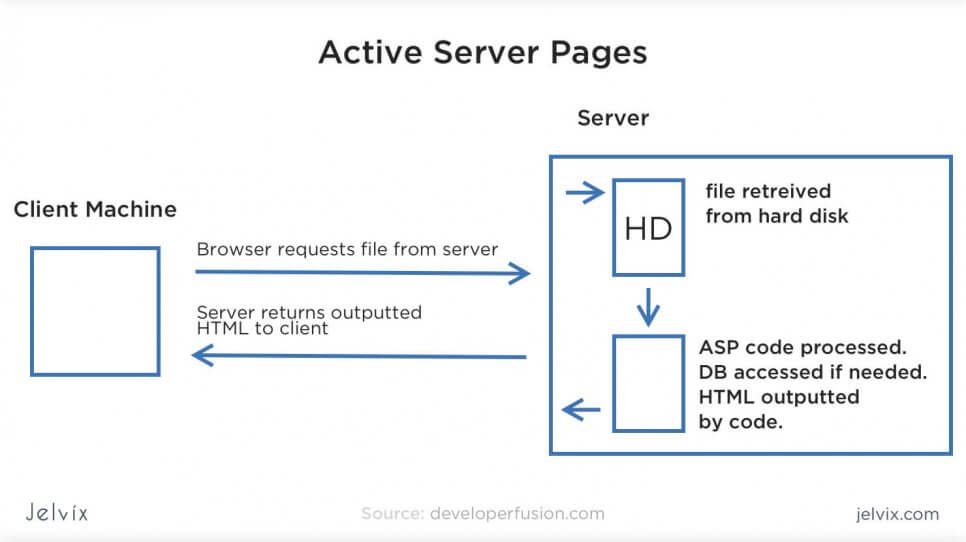
[](https://jelvix.com/wp-content/uploads/2020/09/microsoft-built-framework.jpg)

However, while .NET makes the development process simpler, graphing multiple concepts behind the infrastructure is challenging for many developers. This is especially true for distinguishing between ASP.NET and ASP.NET Core.

**Clearing Out Some Common Misunderstanding**

You already know what .NET is and understand its purpose. However, we still need to clarify the ASP part. Active Service Page is basically an HTML page that runs based on scripts and is used to build web applications.

[Active Service Pages](https://en.wikipedia.org/wiki/Active_Server_Pages) can be previewed from any browsers, including mobile versions. ASPs are typically used for the backend of a web application. They are used to connect inputs and outputs reserved on client and server sides.

[](https://jelvix.com/wp-content/uploads/2020/09/active-server-pages.jpg)

To make .NET more applicable for web application development, Microsoft chose to release the ASP.NET framework – a version of .NET that’s adapted specifically to the web backend.

**What are the Benefits of Building Web Apps as an ASP.NET?**

Microsoft enabled the ASP support to let developers use Microsoft tools, ecosystem, and C++ and C# for web development as well as for desktop. As a result, they achieved the goal of delivering a 100% cross-platform development environment. The main advantages of using .NET-based development tools as web development tools are:

* **The possibility to write backend with C#**: front-end, as usual, is written with JavaScript and its frameworks. However, for backend, developers can choose among multiple tech stacks, and Microsoft’s goal was to make C-based backend development easier. Developers can use [JET packages](https://visualstudio.microsoft.com/) and work in [Visual Studio](https://en.wikipedia.org/wiki/Microsoft_Visual_Studio).
* **A single development environment for JS, HTML, CSS, and C# development**: developers can build syntax for dynamic web applications and immediately preview the app’s outputs sent to a user.
* **.NET framework allows seamless integration of C#** with JavaScript and provides libraries, templates, ready-to-use plugins for progressive web application, and single page application development.

Another question here might be what’s the reason for using a framework for C#-based backend development? What are the advantages that specifically .NET delivers?

.NET just does what all good development frameworks are supposed to do. It provides developers with a ready-to-use application architecture, libraries with patterns and page templates, ready [web app](https://jelvix.com/blog/guide-to-web-application-architecture) functionality (like authentication, access control, page display, etc.)

Additionally, developers work in code editors that automatically complete code fragments, highlight code, sort through different sections efficiently, and support real-time editing.

**How do ASP.NET Core vs ASP.NET Relate?**

You already understand the distinction between .NET and ASP.NET. Just to recap:

NET is a C#, C++, F# framework that makes desktop development easier and faster.

ASP.NET is a framework that has the same functionality as .NET, only it’s adapted to writing backend for web pages and web apps.

Developers can use the same tools, libraries, and infrastructure to build web and desktop projects. This way, a desktop development team can easily learn backend development, development takes less time, and cross-platform development becomes more effortless.

However, it’s not the only distinction in the .NET ecosystem. ASP.NET is only one of the versions of the framework. Another one is called [ASP.NET Core](https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet-core).

**Are ASP.NET and ASP.NET Core the Same?**

ASP.NET was the first version of the web-adapted .NET framework. ASK.NET Core is an improved version with richer functionality, a more comfortable interface, new libraries, and other distinctions. We’ll talk about their side-by-side comparison just in a minute, but here are the key points for now:

* ASP.NET Core is the continuation of ASP.NET, an improved one;
* ASP.NET Core is available as an open-source tool;
* ASP.NET Core is cross-platform and runs on macOS, Linux, and Windows, unlike ASP.NET that works only on Windows;
* Just like ASP.NET, ASP.NET Core is based on the Model-View-Controller framework, like most web development frameworks;
* ASP.NET Core has robust [Cloud support](https://jelvix.com/blog/aws-vs-google-cloud-vs-azure), and it supports modular architecture better than ASP.NET does.

So, ASP.NET Core is a newer, more comfortable version of ASP.NET. Microsoft’s team understood what things could be improved in regards to user experience and improved the functionality.

**Are .NET Core and .NET Framework the Same?**

Similarly, as ASP.NET was broken down into a traditional .NET version and a newer Core edition, a classical .NET framework underwent the same transformation. This is why now developers see demand for two tools – .[NET Core](https://dotnet.microsoft.com/download" \t "_blank) and .[NET Framework](https://dotnet.microsoft.com/download/dotnet-framework" \t "_blank).

**What is the difference between the two?**

.NET framework has been on the market longer, which is why it has a lot more elaborate ecosystem. It’s not just about patches, library updates, and guides – even in the job market .NET framework skills are still highly regarded, even though it technically isn’t the newest version.

**Scenarios for using .NET framework**

So, many teams prefer to work with .NET if they are supporting tools developed before introducing the Core. Another common scenario is when a team used to .NET doesn’t want to spend time on learning Core functionality because they need to meet deadlines fast.

**?**

For more information read our [Complete Guide to Microsoft .NET Framework](https://jelvix.com/blog/microsoft-net-framework).

The issue with *.*NET Framework: the latest version of .NET framework is 4.8. and there will not be a new one. Sooner or later, you will have to migrate to the Core version – there isn’t much doubt about it.

**Scenarios for using .NET Core**

If you are starting a new project and assembling a team from scratch, we definitely recommend going for a newer, core version. For one thing, it’s indeed improved. Developers who migrate from the .NET version to the Core all agree that the interface became a lot better. Even though getting used to the new feel takes some time, it’s definitely worth the struggle.

Also, though .NET is still highly demanded on the market, the situation will shift for sure soon. Microsoft might discontinue the support of all versions altogether. It’s definitely better to be moving in sync with the official updates, which means using the Core version.

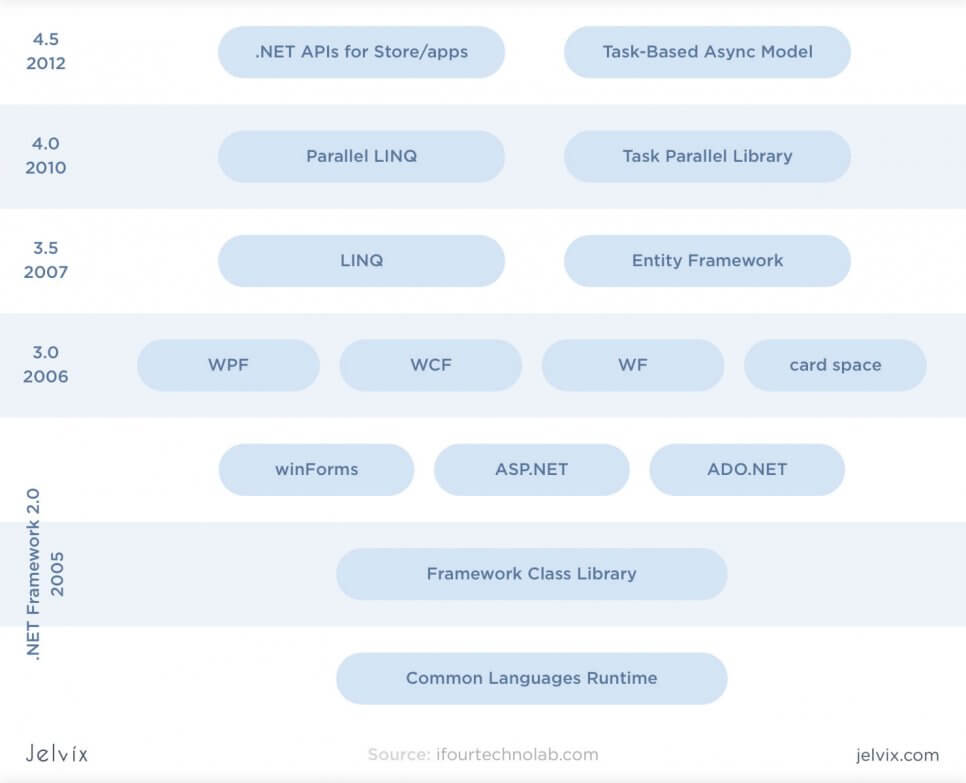
The issue with the .NET Core: although this version will definitely soon become dominant on the market, it’s not the case yet. For now, the ecosystem is poorer, there are not that many educational resources, [open-source add-ons](https://www.mono-project.com/docs/about-mono/dotnet-integration/), and recruiting an experienced .NET Core developer is a challenge.

**ASP.NET Core vs ASP.NET**

So, across this comparison, we mentioned multiple times that Core is an updated version of the network, which is supposedly better. To see if that’s really the case, let’s zoom in on both frameworks’ technical characteristics.

**ASP.NET**

A platform for developing desktop, web, and cross-platform applications. The first version of the platform was published in 2002 and based on the .NET framework. The platform is based on [HTTP](https://jelvix.com/blog/are-you-secured-the-simplest-terms-about-https-and-ssl) protocol for requests and communication.

[](https://jelvix.com/wp-content/uploads/2020/09/asp.net-development-modes.-0A.jpg)

ASP.NET has three major development modes.

* [WebForms](https://dotnet.microsoft.com/apps/aspnet/web-forms): ASP.NET allows building event-driven applications, managing user inputs in forms, and enabling real-time user interactions.
* [MVC:](https://dotnet.microsoft.com/apps/aspnet/mvc) the ASP.NET is based on the model-view-controller to connect application logic, data, and interface. Developers can clearly separate the application’s three components and preview how the entire app changes once Model, View, or Controller are updated.
* [Web pages](https://en.wikipedia.org/wiki/Web_page): the platform allows developers to build dynamic HTML web pages. You can connect the backend code to the front-end functionality to review its actual performance and interface.

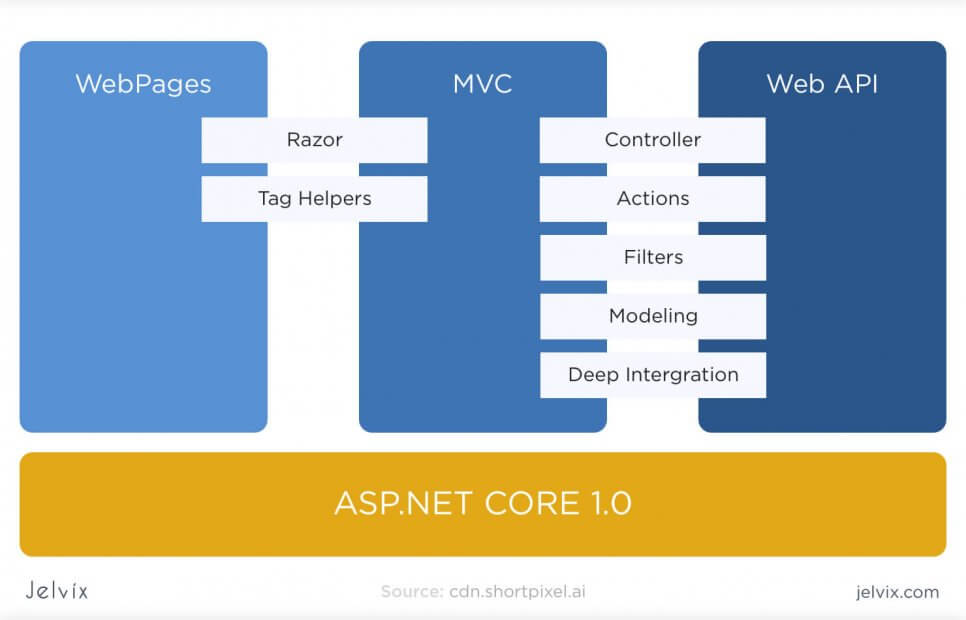
By combining these three development components, the team can easily write, preview, and manage all web application architecture components. On top of that, ASP.NET offers a variety of development styles.

These three development models provide developers with a versatile stack for SPA, cross-platform, and [PWA development](https://jelvix.com/blog/pwa-vs-native-app-benefits-for-users-and-developers). We like to combine all three to get full visibility.

**ASP.NET Architecture and its Components:**

ASP.NET is based on the .NET framework, which is why it shares most of its key components with .NET.

* Supports multiple languages: the primary language for .NET development is C#, but it also supports F#, VB.NET, C++, and others;
* Library: ASP.NET provides reusable code components, interfaces, classes, and values for .NET-based web development.
* Language runtime: developers can detect code issues, handle exceptions, and in-depth editing of the codebase itself.

[](https://jelvix.com/wp-content/uploads/2020/09/asp.net-core-1.0.jpg)

**Development Styles and Models**

[](https://jelvix.com/wp-content/uploads/2020/09/development-styles-and-models.jpg)

**Major Benefits**

When ASP.NET just appeared, it impressed .NET developers with multiple useful benefits – now, however, they sound somewhat less promising, because Core inherited most of them. Still, let’s take a look at the main positive aspects of .NET as they will also help us understand Core’s functionality better.

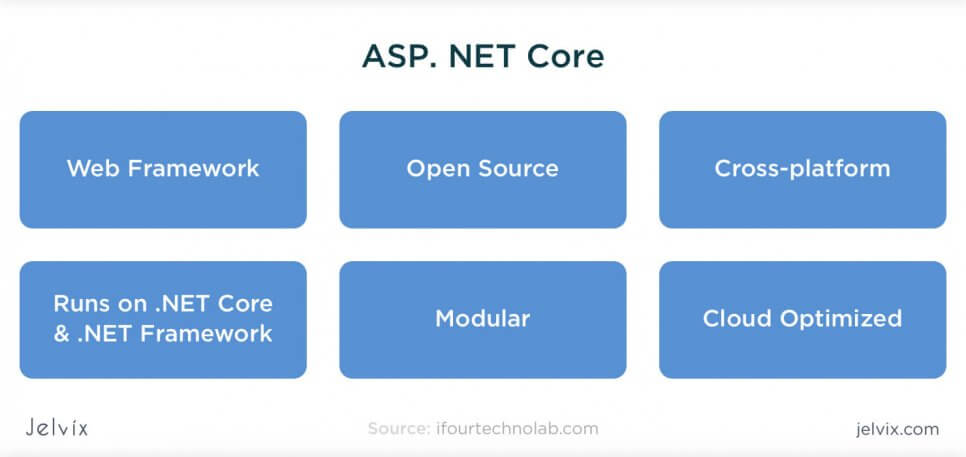
* Security and access control: the codebase is protected with Microsoft authentication; developers can also configure safety settings early on during development;
* Fewer lines of code, more functionality: ASP offers a convenient view of code lines, grouping the codebase and shortening the syntax;
* Built-in HTML generator. ASP.NET supports HTML and allows previewing web pages automatically;
* Support of multiple languages: although ASP.NET is mainly used with C#, it’s not an obligatory requirement; you can use any language, including Java or [Ruby](https://jelvix.com/blog/red-or-blue-pill-neo-ruby-vs-python-which-will-you-choose-for-your-backend).
* Realtime Windows server monitoring: the server checks the number of components, pages, and integrations;
* Garbage collection and security alerts: the framework detects memory leaks, flawed loops, etc;
* Built-in caching system;
* Separation of the application internal logic and content.

Overall, ASP.NET provides a lot of great features for code editing, management, and monitoring. It connects the backend of the application to other components, giving developers a full idea of their app’s functionality and interface.

**ASP.NET Core**

Just like we reviewed ASP.NET, let’s take a look at the Core’s architecture, components, development styles, and benefits.

ASP Core is a new version of ASP that became available open-source and can be executed not only on Windows but also on Linux and Mac. With this update, ASP.NET became increasingly more available to teams who usually work outside of the Microsoft ecosystem.

[](https://jelvix.com/wp-content/uploads/2020/09/asp.net-core.jpg)

**Architecture**

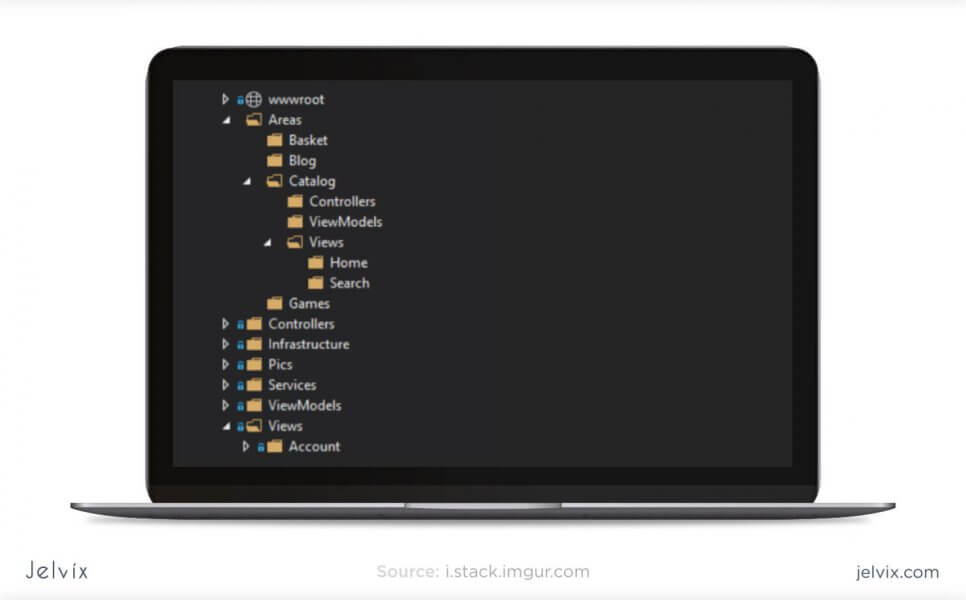
ASP.NET Core is a Cloud platform where teams can build web and cross-platform apps from any OS. This version is a combination of Model-View Controller and [WEB API](https://docs.microsoft.com/en-us/aspnet/web-api/) within a unified development system.

With ASP Core’s MVC, users can do model binding and validation for their requests, automatically establishing parameters and converting values. Additionally, the framework supports filtering that automatically detects invalid models and blocks them for future sessions.

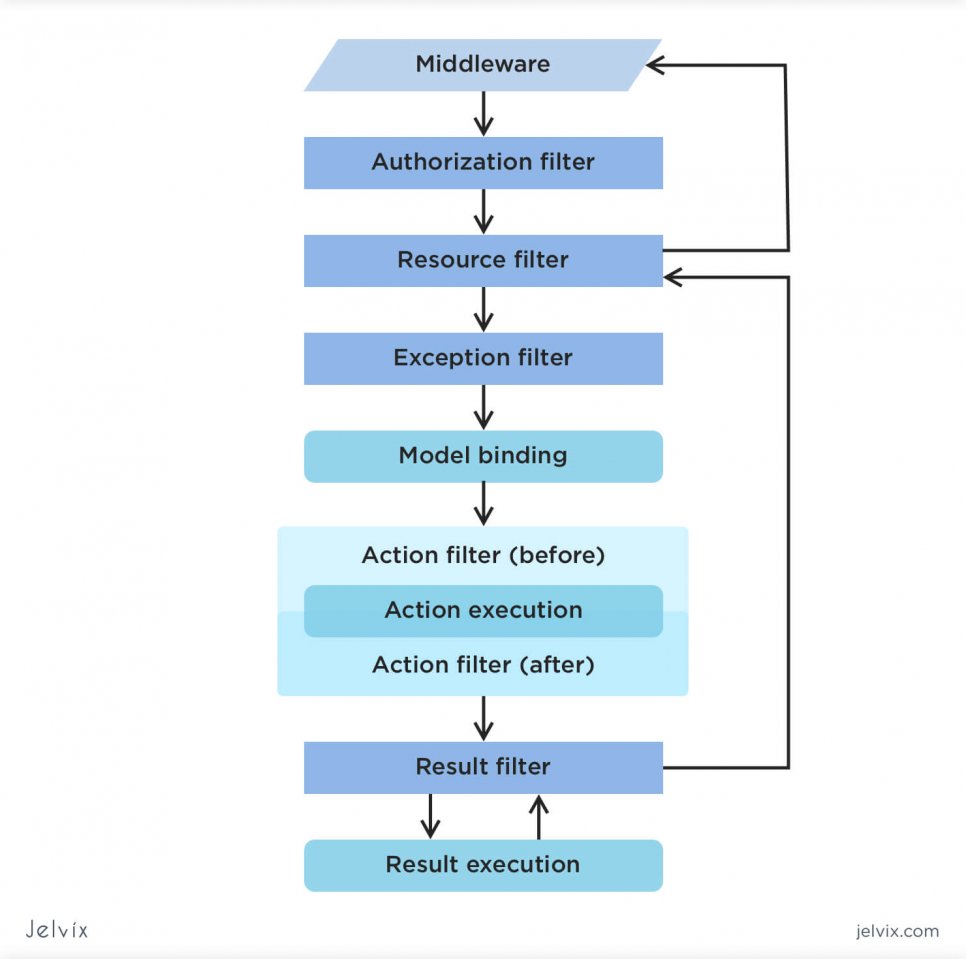
What used to be “References” in ASP.NET, became a Dependencies folder in ASP.NET Core. Right as you start creating the application, the platform demonstrates all default dependencies and allows you to track their increase or decrease.

**Benefits of Using ASP.NET Core**

* Clear structure: the platform allows breaking applications into multiple layers, using clear architecture templates, manage business logic, and UI files.
* Features are structured in [Views and Controllers](https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/overview/understanding-models-views-and-controllers-cs). Developers can organize functionality by feature folders, create separate Areas with standalone sets of Views and Controllers.

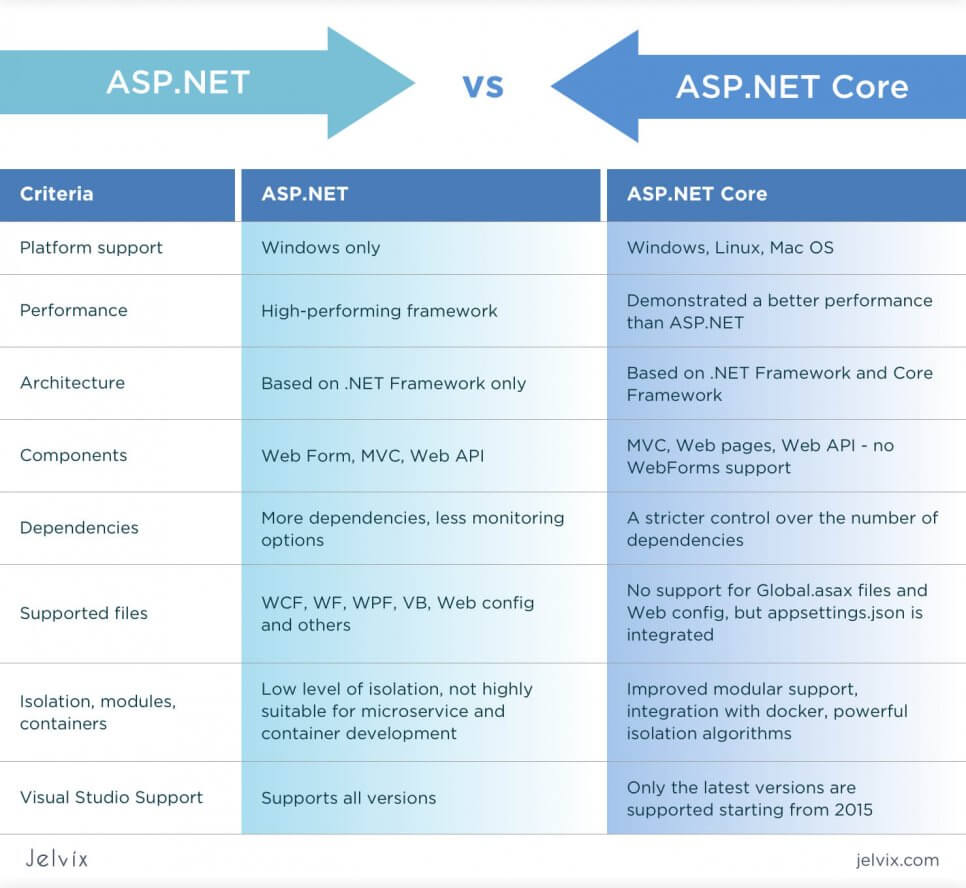
[](https://jelvix.com/wp-content/uploads/2020/09/storage-folders.jpg)

* Filters: in ASP.NET Core, developers can easily track middleware and duplication by assigning filters that would detect unwanted characteristics and clean up the codebase. You can filter models, exceptions, actions, resources, and authorization. Importantly, these filters can be structured in an organized hierarchy.

[](https://jelvix.com/wp-content/uploads/2020/09/filters-in-asp.net-core.jpg)

ASP.NET Core offers multiple other updates at the levels of security, token-based authentication, client communication, update control. Most importantly, this additional functionality is very well structured in the application – it’s obvious that the team focused on minimizing dependencies. You can isolate components and make standalone changes across the entire codebase.

Lastly, ASP.NET Core can be hosted in Docker, HTTP.sys, Apache, Kestrel, IIS, Nginx, and other systems. It’s a great option for Cloud and microservices development, especially well-suited for enterprise-level projects.

[](https://jelvix.com/wp-content/uploads/2020/09/aps.net-asp.net-core.jpg)

**ASP.NET Core vs .NET Core: Side-to-Side Comparison**

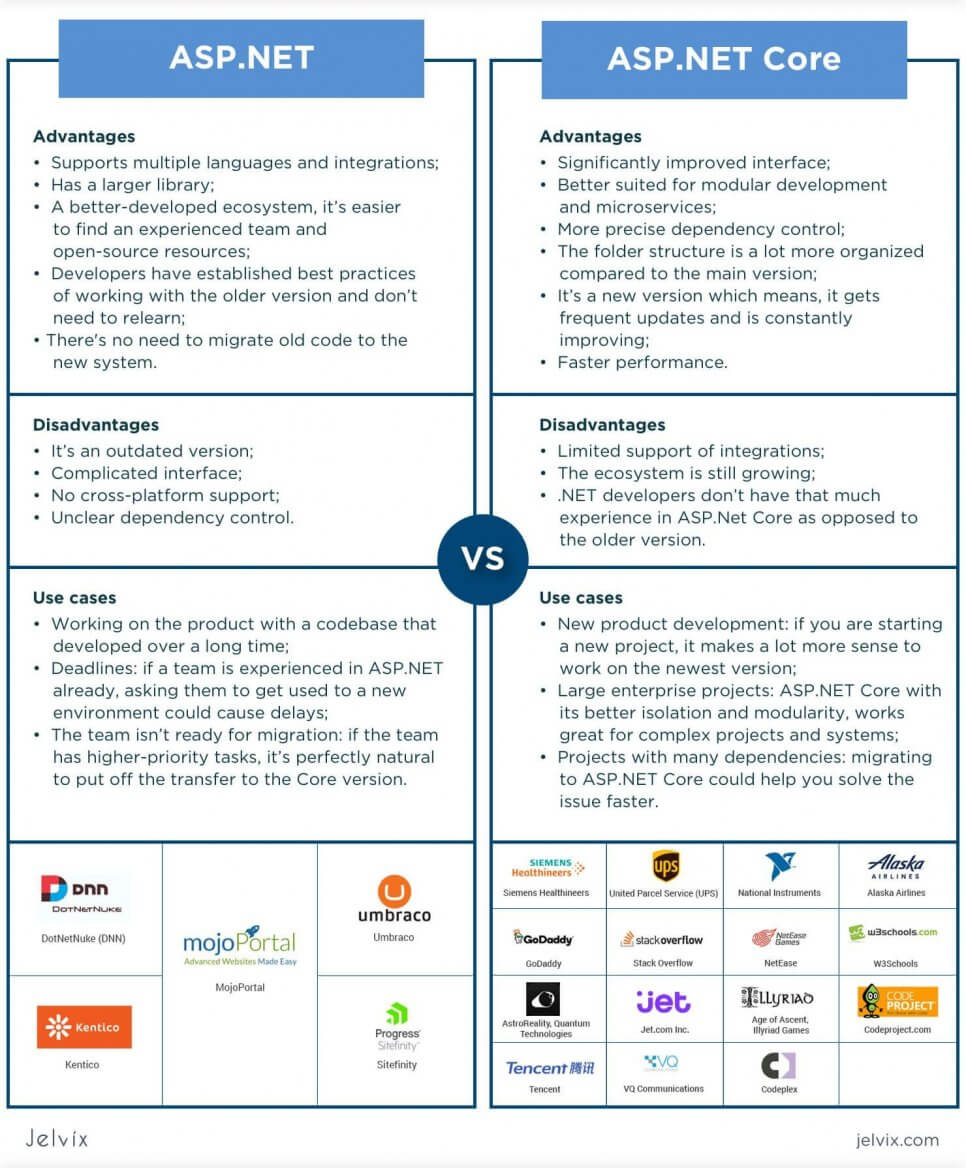
Overall, the major advantage of ASP.NET vs ASP.NET Core is the maturity of the ecosystem. It supports more plugins, has more powerful integration, and can be opened in old versions of Microsoft Visual Studio. On the other hand, this edition will likely get discontinued eventually, considering that it’s an outdated version of the platform.

.NET Core, on the other hand, stands out with its improved interface and compatibility for Windows, macOS, Linux, but it still lacks maturity in terms of the ecosystem.

**Choosing Between ASP.NET Core vs .NET Core for Web Application Development**

Let’s recap the main advantages and disadvantages of both versions before committing to a final option.

What is the difference between ASP.NET and ASP.NET Core?

[](https://jelvix.com/wp-content/uploads/2020/09/difference-between-asp.net-and-asp.core_.jpeg)

**Porting ASP.NET MVC to ASP.NET Core**

Teams who migrate from ASP.NET to ASP.NET Core need to go through the following stages:

* Setup;
* Migrating views and controllers;
* Transferring static content;
* Managing dependencies.

To start the migration, you need to have a [NET.Core SDK](https://dotnet.microsoft.com/download" \t "_blank), Visual Studio (at least 2019 version), and an ASP.NET project.

**Step 1 – Migrating MVC**

Although both ASP.NET Core vs ASP.NET MVC feature an MVC, they approach it differently. The MVC will be automatically transferred to the new project, but you still need to re-register it. Here’s the list of commands that are used to upload all the MVC file to the Core.

* AddControllersWithViews: command used to let ASP.NET Core recognize APIs, controllers, and views of the old application;
* UseStaticFiles makes static files recognized by the system;
* UseRouting allows uploading routing paths from your .NET projects.

**Step 2 – Testing End-Points**

You need to check end-points to see if application content and styles were moved correctly. The official documentation recommends rendering transferred views to the browser to check their correct display.

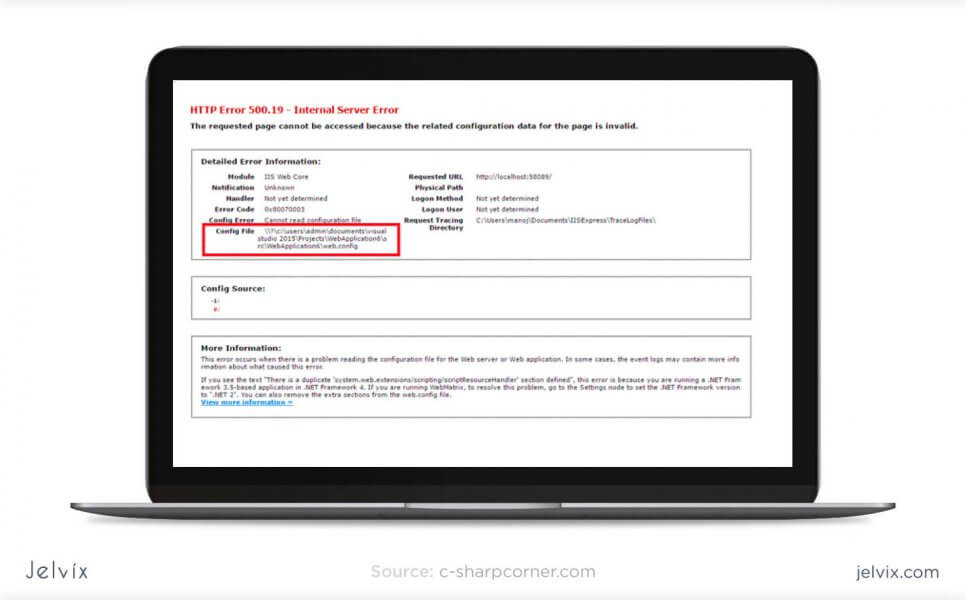
**Step 3 – Transferring Static Content**

ASP.NET MVC vs ASP.NET Core store static files in different directories. In the first version, files were stored together with server-side content, whereas in the Core version, static files aren’t mixed with other backend content anymore.

So, you need to change a directory from *wwwroot* to ASP.NET Core WebApp1 projects.

**Step 4 – Checking Migration Errors**

For now, ASP.NET Core doesn’t support built-in bundling configuration. However, Microsoft documentation suggests using open-source plugins like WebOptimizer. If your project shows 500 errors (that’s another common consequence of the move), you can integrate a UseDeveloperExceptionPage to the ASP.NET Core.

[](https://jelvix.com/wp-content/uploads/2020/09/migration-errors.jpg)

**Conclusion**

The best practice is to use ASP.NET Core, since it’s a new version that presents more opportunities to the team. Migrating to ASP Core form ASP.NET is a long-term investment – this way, the team is in sync with official Microsoft’s updates and can benefit from continuous improvements.

While there are use cases for remaining on ASP.NET, it’s generally not a good idea. You’ll be setting your business back and cutting the team off from official resources. It’s better not to wait till ASP.NET becomes entirely outdated and possibly discontinued. Instead, show an active approach and start migration.

**NET vs. .NET Framework for server apps**

* Article
* 05/13/2022
* 5 minutes to read
* 24 contributors

There are two supported [.NET implementations](https://docs.microsoft.com/en-us/dotnet/standard/glossary#implementation-of-net) for building server-side apps.

| **Implementation** | **Included versions** |
| --- | --- |
| .NET | .NET Core 1.0 - 3.1, .NET 5, and later versions of .NET. |
| .NET Framework | .NET Framework 1.0 - 4.8 |

Both share many of the same components, and you can share code across the two. However, there are fundamental differences between the two and your choice depends on what you want to accomplish. This article provides guidance on when to use each.

Use .NET for your server application when:

* You have cross-platform needs.
* You're targeting microservices.
* You're using Docker containers.
* You need high-performance and scalable systems.
* You need side-by-side .NET versions per application.

Use .NET Framework for your server application when:

* Your app currently uses .NET Framework (recommendation is to extend instead of migrating).
* Your app uses third-party libraries or NuGet packages not available for .NET.
* Your app uses .NET Framework technologies that aren't available for .NET.
* Your app uses a platform that doesn't support .NET.

**When to choose .NET**

The following sections give a more detailed explanation of the previously stated reasons for picking .NET over .NET Framework.

**Cross-platform needs**

If your web or service application needs to run on multiple platforms, for example, Windows, Linux, and macOS, use .NET.

.NET supports the previously mentioned operating systems as your development workstation. Visual Studio provides an Integrated Development Environment (IDE) for Windows and macOS. You can also use Visual Studio Code, which runs on macOS, Linux, and Windows. Visual Studio Code supports .NET, including IntelliSense and debugging. Most third-party editors, such as Sublime, Emacs, and VI, work with .NET. These third-party editors get editor IntelliSense using [Omnisharp](https://www.omnisharp.net/). You can also avoid any code editor and directly use the [.NET CLI](https://docs.microsoft.com/en-us/dotnet/core/tools/), available for all supported platforms.

**Microservices architecture**

A microservices architecture allows a mix of technologies across a service boundary. This technology mix enables a gradual embrace of .NET for new microservices that work with other microservices or services. For example, you can mix microservices or services developed with .NET Framework, Java, Ruby, or other monolithic technologies.

There are many infrastructure platforms available. [Azure Service Fabric](https://azure.microsoft.com/services/service-fabric/) is designed for large and complex microservice systems. [Azure App Service](https://azure.microsoft.com/services/app-service/) is a good choice for stateless microservices. Microservices alternatives based on Docker fit any kind of microservices approach, as explained in the [Containers](https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server#containers) section. All these platforms support .NET and make them ideal for hosting your microservices.

For more information about microservices architecture, see [.NET Microservices. Architecture for Containerized .NET Applications](https://docs.microsoft.com/en-us/dotnet/architecture/microservices/).

**Containers**

Containers are commonly used in conjunction with a microservices architecture. Containers can also be used to containerize web apps or services that follow any architectural pattern. .NET Framework can be used on Windows containers, but the modularity and lightweight nature of .NET makes it a better choice for containers. When creating and deploying a container, the size of its image is much smaller with .NET than with .NET Framework. Because it's cross-platform, you can deploy server apps to Linux Docker containers, for example.

Docker containers can be hosted in your own Linux or Windows infrastructure, or in a cloud service such as [Azure Kubernetes Service](https://azure.microsoft.com/services/kubernetes-service/). Azure Kubernetes Service can manage, orchestrate, and scale container-based applications in the cloud.

**High-performance and scalable systems**

When your system needs the best possible performance and scalability, .NET and ASP.NET Core are your best options. The high-performance server runtime for Windows Server and Linux makes ASP.NET Core a top performing web framework on [TechEmpower benchmarks](https://www.techempower.com/benchmarks/" \l "hw=ph&test=plaintext).

Performance and scalability are especially relevant for microservices architectures, where hundreds of microservices may be running. With ASP.NET Core, systems run with a much lower number of servers/Virtual Machines (VM). The reduced servers/VMs save costs in infrastructure and hosting.

**Side by side .NET versions per application level**

To install applications with dependencies on different versions of .NET, we recommend .NET. This implementation supports side-by-side installation of different versions of the .NET runtime on the same machine. This side-by-side installation allows multiple services on the same server, each of them on its own version of .NET. It also lowers risks and saves money in application upgrades and IT operations.

Side-by-side installation isn't possible with .NET Framework. It's a Windows component, and only one version can exist on a machine at a time. Each version of .NET Framework replaces the previous version. If you install a new app that targets a later version of .NET Framework, you might break existing apps that run on the machine, because the previous version was replaced.

**When to choose .NET Framework**

.NET offers significant benefits for new applications and application patterns. However, .NET Framework continues to be the natural choice for many existing scenarios, and as such, .NET Framework isn't replaced by .NET for all server applications.

**Current .NET Framework applications**

In most cases, you don't need to migrate your existing applications to .NET. Instead, a recommended approach is to use .NET as you extend an existing application, such as writing a new web service in ASP.NET Core.

**Third-party libraries or NuGet packages not available for .NET**

.NET Standard enables sharing code across all .NET implementations, including .NET Core/5+. With .NET Standard 2.0, a compatibility mode allows .NET Standard and .NET projects to reference .NET Framework libraries. For more information, see [Support for .NET Framework libraries](https://docs.microsoft.com/en-us/dotnet/standard/whats-new/whats-new-in-dotnet-standard#support-for-net-framework-libraries).

You need to use .NET Framework only in cases where the libraries or NuGet packages use technologies that aren't available in .NET Standard or .NET.

**.NET Framework technologies not available for .NET**

Some .NET Framework technologies aren't available in .NET. The following list shows the most common technologies not found in .NET:

* ASP.NET Web Forms applications: ASP.NET Web Forms are only available in .NET Framework. ASP.NET Core cannot be used for ASP.NET Web Forms.
* ASP.NET Web Pages applications: ASP.NET Web Pages aren't included in ASP.NET Core.
* Workflow-related services: Windows Workflow Foundation (WF), Workflow Services (WCF + WF in a single service), and WCF Data Services (formerly known as "ADO.NET Data Services") are only available in .NET Framework.
* Language support: Visual Basic and F# are currently supported in .NET, but not for all project types. For a list of supported project templates, see [Template options for dotnet new](https://docs.microsoft.com/en-us/dotnet/core/tools/dotnet-new#arguments).

For more information, see [.NET Framework technologies unavailable in .NET](https://docs.microsoft.com/en-us/dotnet/core/porting/net-framework-tech-unavailable).

**Platform doesn't support .NET**

Some Microsoft or third-party platforms don't support .NET. Some Azure services provide an SDK not yet available for consumption on .NET. In such cases, you can use the equivalent REST API instead of the client SDK.

.Net vs .Net Core

POST PUBLISHED:DECEMBER 1, 2020

.NET has two supported implementations for server-side app development: .NET Framework and .NET Core.

If you ask, they both share several similar mechanisms. Nonetheless, there are important differences amidst the two and that’s precisely what you will accomplish in this article.

Theoretically, both share the same philosophy of using CLR, CTS, JIT, GC, etc., however, the implementation of these in both frameworks are very different.

How is a framework different from a language? Or how is .NET Framework different from .NET Core?

You will get to know about both, .Net vs .Net Core.

So, stay tuned.

.Net vs .Net Core 1

Learn the Difference, It Could Save Your Life

Are you anxious about how a framework is different from a language? Feeling overwhelmed?

Don’t worry, because you are going to learn the difference in a few seconds.

Stay with us on .Net vs .Net Core.

‘A programming language is a specific human-readable way of telling a computer what to do.’

For example, if you said, ‘hey computer, plus 5 with 5’ it wouldn’t know what to do. But if you use a programming language to send an instruction to the computer, it would understand instantly.

A language is crafted using serious symbols () ++ & @ for \* {} <> // : = ; but each symbol represents a meaning, an instruction for the computer. When combined, these symbols create a computer program, which becomes a set of instructions for the computer to follow.

To understand a framework, think of a physical building.

The basic building block of that physical building can be a framework; the supporting structure of the building.

In the software development context, think of it as some software that’s already been developed and can help you create your application.

So, a framework simply is a reusable code that provides you a specific functionality for you to use in your program.

We gave you the following summary for sake of clarity. Anyhow, let’s move onward.

.NET Vs .NET Core

.Net core and .NET framework, what to choose for your desktop application? You are going to learn about .NET Core and how it compares to .NET Framework?

.NET Framework: Learn the Basics

This framework was developed by Microsoft, and initially, it was only operational on Windows devices. It still targets the apps for Windows. Also, you can develop web applications too.

Currently. the .NET framework is most commonly used to code and run software applications.

.NET framework isn’t cross-platform. It only supports applications deployed on Windows.

.NET framework does not offer a few but several sorts of applications to build.

You can create a website, desktop apps, development productivity tools using the Integrated Development Environment (IDE), and much more.

The applications designed in the .NET framework are coded in F#, C#, or Visual Basic.

There are two chief components of the .NET framework:

The Common Language Runtime (CLR)

The .NET Framework Class Library

The CLR component exclusively provides the API integration functionality and the class library is used to execute the engine that handles running applications.

.NET framework is not open source. Developers can use it for free for research and study purposes. However, they need to purchase a license of Visual Studio IDE for commercial use.

.NET Framework 4.8 is the current release and in most cases, you must use this version.

.NET Core: Learn the Basics

The .NET core is open-source, a free to use cross-platform software framework. It is for Microsoft Windows, Linux OS, Apple macOS, a successor to the .NET Framework.

.Net vs .Net Core 2

The framework is a blend of the ASP.NET MVC and ASP.NET Web API, but it also includes Xamarin, Win Forms, and WPF (Core 3).

It was developed by the .NET foundation. MIT Licensed and written in C++ and C#.

You can design multiple applications such as desktop, mobile, web, IoT, cloud, machine learning, game, and microservices.

Although you can design desktop apps in .net core, they are not capable to run on MAC OS or Linux.

As it is the new form of the .NET framework; the need was to make it light in weight, much faster than the current version, and operational across platforms.

.NET core has the following characteristics that make it a legendary framework:

Cross-platform

Reliable Across Architectures

Multiple Application Development Ease

Multilingual Support

Modular Architecture Approach

CLI Tools

Flexible Deployment

Compatibility

.NET Vs .NET Core

.NET FRAMEWORK .NET CORE

Net Stack1- ASP.Net2- WPF 3- Windows Forms Net Stack1- ASP .Net v52- .Net Native

A Platform A Platform

Not Open-source Open-source

Majorly used to develop Windows based applications. Build once, and it runs on limited platforms and then obviously cross applications types is also limited to handful.

Does not support multi-languages or micro-services. .NET framework also supports multiple languages and micro-services.

Too heavy for command line interface Too smooth & light for command line interface

Designed and maintained by Microsoft. Designed by the .NET community & Microsoft.

Closing Statement

There are certain boundaries with the .NET framework as you perceive. The trend that follows these days says that the applications much support multiple platforms.

So, considering this, the .NET core wins here.

But, there are features that only the .NET framework as the latest version can support. Overall, the .NET Core is the modernized and reshaped version of the .NET Framework.