Unit 2

Introduction to ASP.NET

ASP.NET

- ASP.NET is a web application framework designed and developed by Microsoft.
- a subset of the .NET Framework and successor of the classic ASP (Active Server Pages).
- With version 1.0 of the .NET Framework, it was first released in January 2002.
- before the year 2002 for developing web applications and services, there was Classic ASP.

.NET	ASP.NET
.NET is a software development framework aimed to develop Windows, Web and Server based applications.	ASP.NET is a main tool that present in the .NET Framework and aimed at simplifying the creation of dynamic webpages.
Server side and client side application development can be done using .NET framework.	You can only develop server side web applications using ASP.NET as it is integrated with .NET framework.
Mainly used to make business applications on the Windows platform.	It is used to make dynamic web pages and websites using .NET languages.
Its programming can be done using any language with CIL (Common Intermediate Language) compiler.	Its programming can be done using any .NET compliant language.

.NET Core

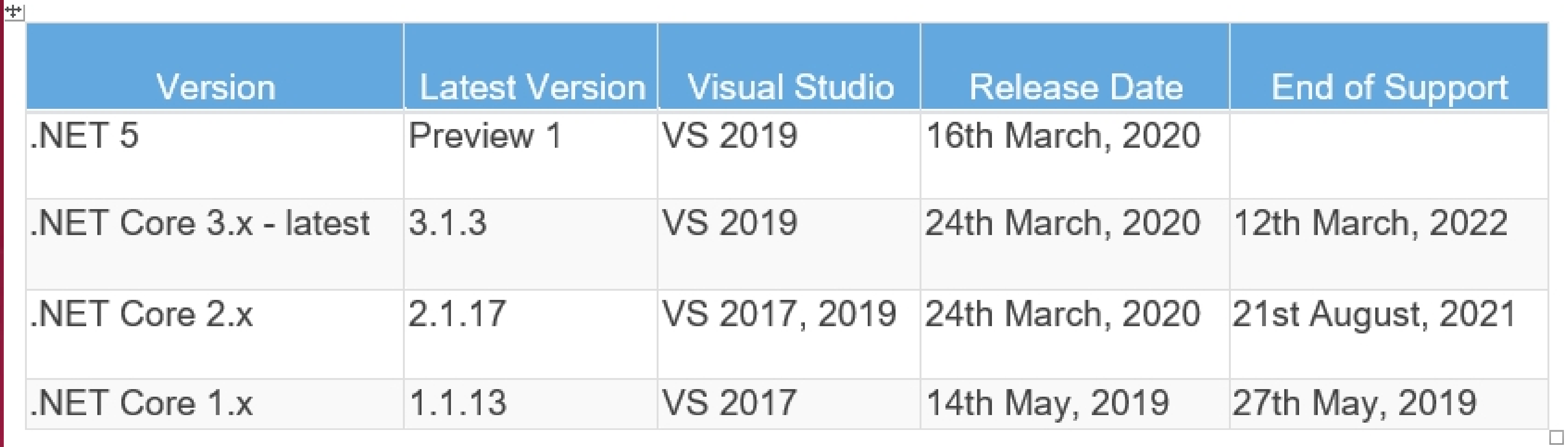
- .NET Core is a new version of .NET Framework
- general-purpose development platform maintained by Microsoft.
- It is a cross-platform framework that runs on Windows, macOS, and Linux operating systems, used to build different types of applications such as mobile, desktop, web, cloud, IoT, machine learning, microservices, game, etc.
- .NET Core is written from scratch to make it modular, lightweight, fast, and cross-platform Framework.
- It includes the core features that are required to run a basic .NET Core app. Other features are provided as NuGet packages, which you can add it in your application as needed. In this way, the .NET Core application speed up the performance, reduce the memory footprint and becomes easy to maintain.

.NET Core Characteristics

- **Open-source Framework:** .NET Core is an open-source framework maintained by Microsoft and available on GitHub under MIT and Apache 2 licenses. It is a .NET Foundation project.
- **Cross-platform:** .NET Core runs on Windows, macOS, and Linux operating systems. There are different runtime for each operating system that executes the code and generates the same output.
- **Consistent across Architectures**: Execute the code with the same behavior in different instruction set architectures, including x64, x86, and ARM.
- **Wide-range of Applications:** Various types of applications can be developed and run on .NET Core platform such as mobile, desktop, web, cloud, IoT, machine learning, microservices, game, etc.

- **Supports Multiple Languages:** You can use C#, F#, and Visual Basic programming languages to develop .NET Core applications. You can use your favorite IDE, including Visual Studio 2017/2019, Visual Studio Code, Sublime Text, Vim, etc.
- **Modular Architecture**: supports modular architecture approach using NuGet packages for various features that can be added to the .NET Core project as needed. Even the .NET Core library is provided as a NuGet package. The NuGet package for the default .NET Core application model is Microsoft.NETCore.App. It reduces the memory footprint, speeds up the performance, and easy to maintain.
- **CLI Tools:** .NET Core includes CLI tools (Command-line interface) for development and continuous-integration.
- **Flexible Deployment:** .NET Core application can be deployed user-wide or system-wide or with Docker Containers.
- **Compatibility:** Compatible with .NET Framework and Mono APIs by using .NET Standard specification

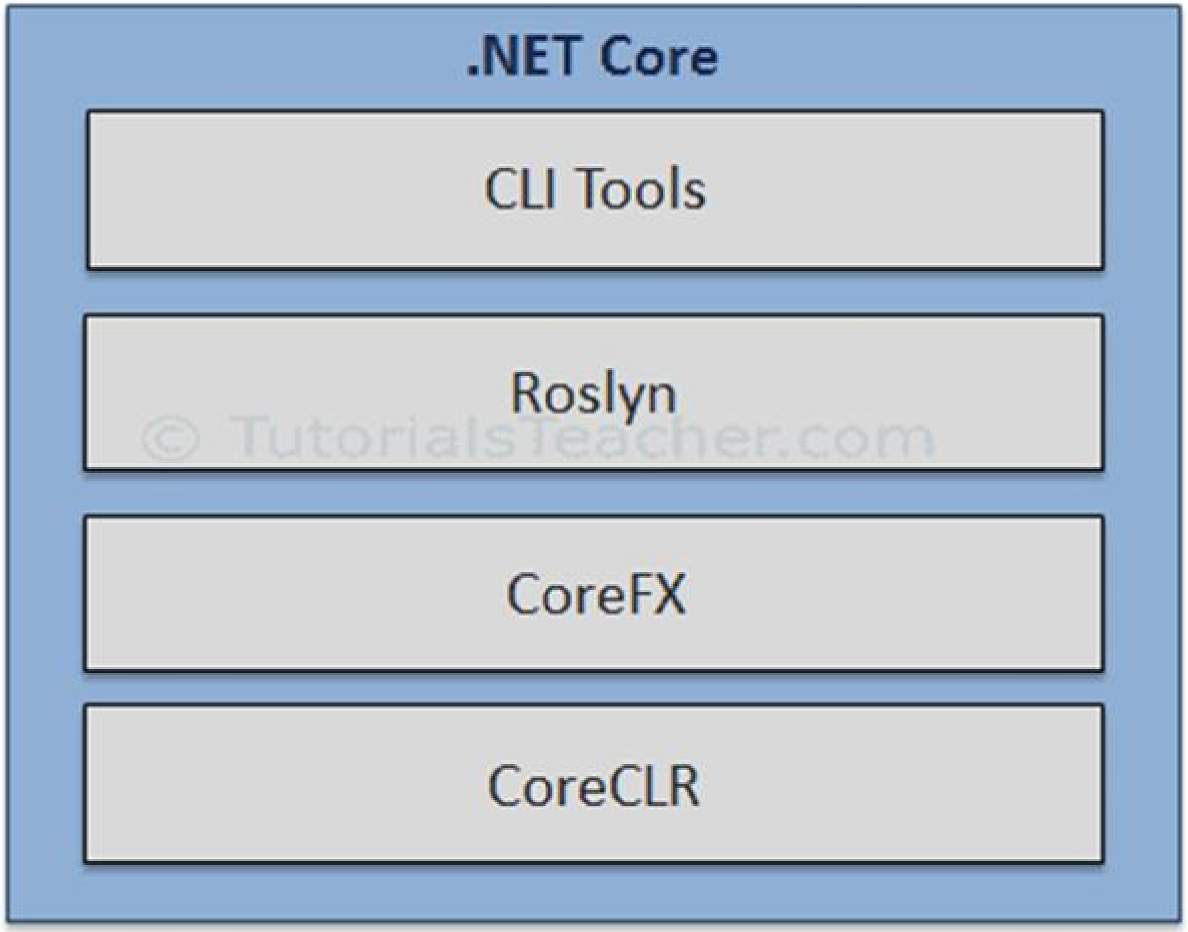
.NET Core Version History



Version	Latest Version	Visual Studio	Release Date	End of Support
.NET 5	Preview 1	VS 2019	16th March, 2020	
.NET Core 3.x - latest	3.1.3	VS 2019	24th March, 2020	12th March, 2022
.NET Core 2.x	2.1.17	VS 2017, 2019	24th March, 2020	21st August, 2021
.NET Core 1.x	1.1.13	VS 2017	14th May, 2019	27th May, 2019

.NET Core Framework parts

- **CLI Tools:** A set of tooling for development and deployment.
- **Roslyn:** Language compiler for C# and Visual Basic
- **CoreFX:** Set of framework libraries.
- **CoreCLR:** A JIT based CLR (Command Language Runtime).



Mono

- Mono is an example of a cross-platform framework available on Windows, macOS, Linux, and more. It was first designed as an open source implementation of the .NET Framework on Linux.
- Mono (like .NET) is tied heavily around the C# programming language, known for its high level of portability.
- For example, the Unity game engine uses C# as a cross-platform way of creating video games. This is in part due to the language's design. C# can be turned into CIL (Common Intermediate Language), which can either be compiled to native code (faster, less portable), or run through a virtual machine (slower, more portable).
- Mono provides the means to compile, and run C# programs, similar to the .NET Framework.

ASP.NET Web Forms

- a part of the ASP.NET web application framework and is included with Visual Studio.
- you can use to create ASP.NET web applications, the others are ASP.NET MVC, ASP.NET Web Pages, and ASP.NET Single Page Applications.
- Web Forms are pages that your users request using their browser. These pages can be written using a combination of HTML, client-script, server controls, and server code.
- When users request a page, it is compiled and executed on the server by the framework, and then the framework generates the HTML markup that the browser can render.

ASP.NET Web Forms

- An ASP.NET Web Forms page presents information to the user in any browser or client device.
- The Visual Studio (IDE) lets you drag and drop server controls to lay out your Web Forms page. You can then easily set properties, methods, and events for controls on the page or for the page itself. These properties, methods, and events are used to define the web page's behavior, look and feel, and so on
- Based on Microsoft ASP.NET technology, in which code that runs on the server dynamically generates Web page output to the browser or client device.

Features of ASP.NET Web Forms

- **Server Controls-** ASP.NET Web server controls are similar to familiar HTML elements, such as buttons and text boxes. Other controls are calendar controls, and controls that you can use to connect to data sources and display data.
- **Master Pages-** ASP.NET master pages allow you to create a consistent layout for the pages in your application. A single master page defines the look and feel and standard behavior for all of the pages (or a group of pages) in your application. You can then create individual content pages along with the master page to render the web page.
- **Working with Data**- ASP.NET provides many options for storing, retrieving, and displaying data in web page UI elements such as tables and text boxes and drop-down lists.

Features of ASP.NET Web Forms

- **Client Script and Client Frameworks -** You can write client-script functionality in ASP.NET Web Form pages to provide responsive user interface to users. You can also use client script to make asynchronous calls to the Web server while a page is running in the browser.
- **Routing -** URL routing allows you to configure an application to accept request URL. A request URL is simply the URL a user enters into their browser to find a page on your web site. You use routing to define URLs that are semantically meaningful to users and that can help with search-engine optimization (SEO).
- **State Management -** ASP.NET Web Forms includes several options that help you preserve data on both a per-page basis and an application-wide basis.
- **Security -** offer features to develop secure application from various security threats.

Features of ASP.NET Web Forms

- **Performance –** offers performance related to page and server control processing, state management, data access, application configuration and loading, and efficient coding practices.
- **Internationalization -** enables you to create web pages that can obtain content and other data based on the preferred language setting or localized resource for the browser or based on the user's explicit choice of language. Content and other data is referred to as resources and such data can be stored in resource files or other sources.
- **Debugging and Error Handling -** diagnose problems that might arise in application. Debugging and error handling are well so that applications compile and run effectively.
- **Deployment and Hosting-** Visual Studio, ASP.NET, Azure, and IIS provide tools that help you with the process of deploying and hosting your application

Let’s create first ASP.NET Web Forms Project in Visual Studio 2017/2019

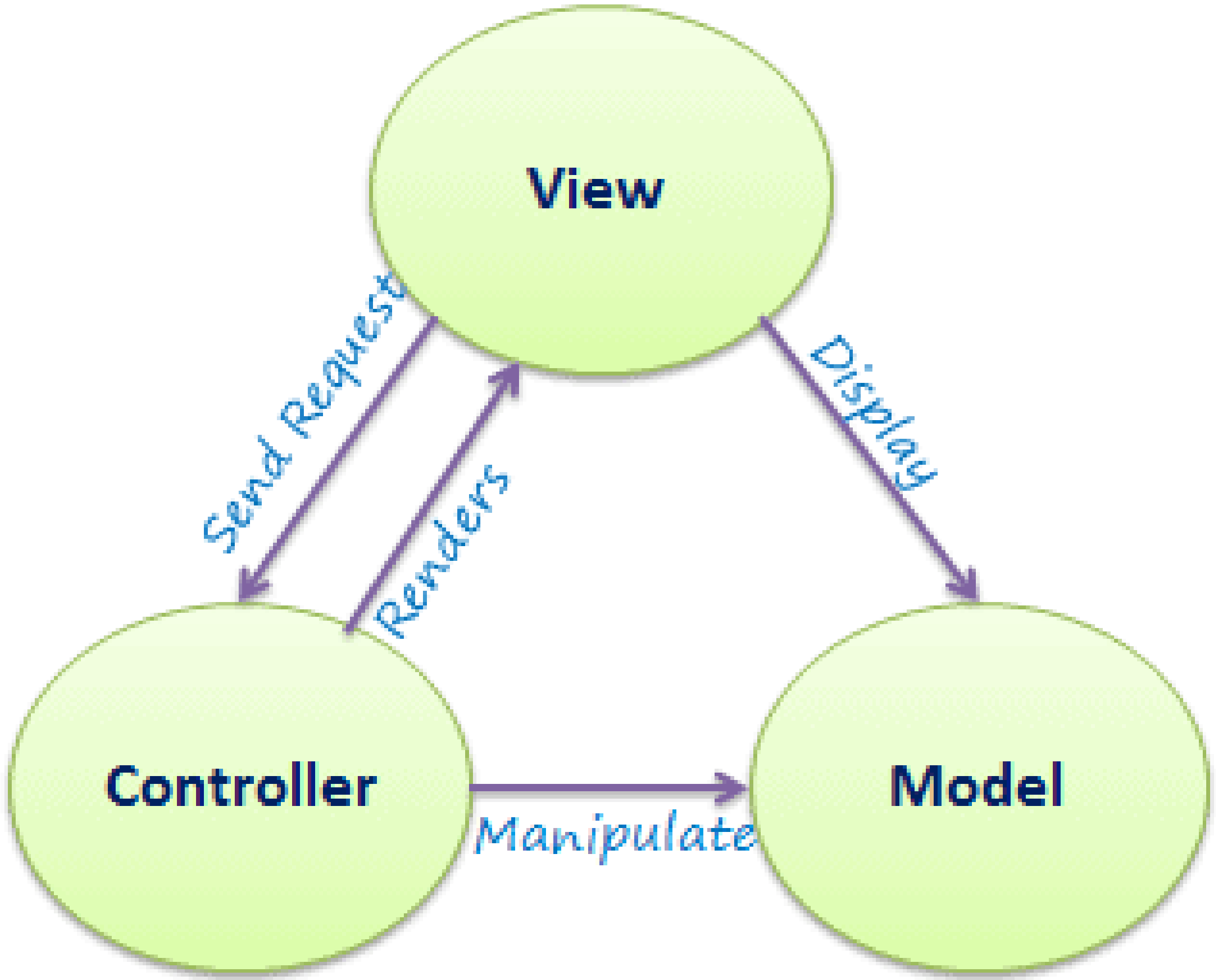
ASP.NET MVC

- ASP.NET MVC is an open source web development framework from Microsoft that provides a Model View Controller architecture.
- ASP.net MVC offers an alternative to ASP.net web forms for building web applications.
- It is a part of the .Net platform for building, deploying and running web apps.
- You can develop web apps and website with the help of HTML, CSS, jQuery, Javascript, etc.

ASP.NET MVC Architecture

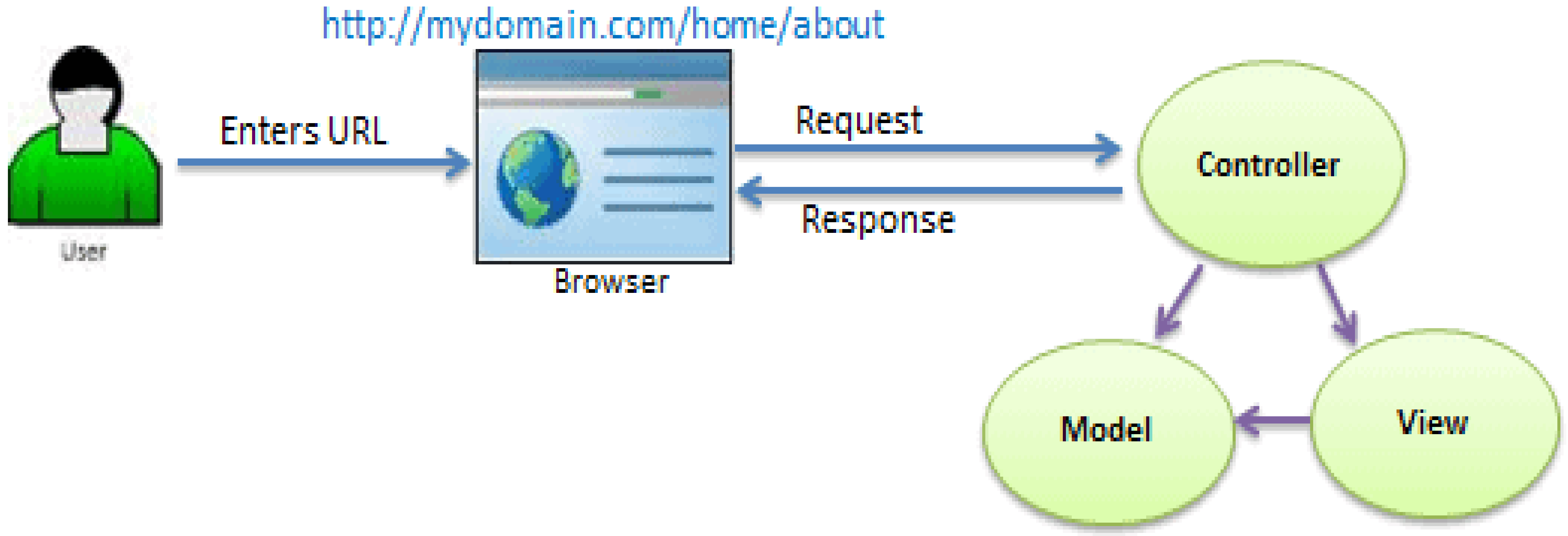
- MVC stands for Model, View, and Controller. MVC separates an application into three components - Model, View, and Controller.
- **Model**: represents the shape of the data. A class in C# is used to describe a model. Model objects store data retrieved from the database. Model represents the data.
- **View**: View in MVC is a user interface. View display model data to the user and also enables them to modify them. View in ASP.NET MVC is HTML, CSS, and some special syntax (Razor syntax) that makes it easy to communicate with the model and the controller.
- **Controller**: handles the user request. Typically, the user uses the view and raises an HTTP request. Controller processes request and returns the appropriate view as a response. Controller is the request handler.

ASP.NET MVC Architecture



Request Flow in MVC Architecture

The following figure illustrates the flow of the user's request in ASP.NET MVC.

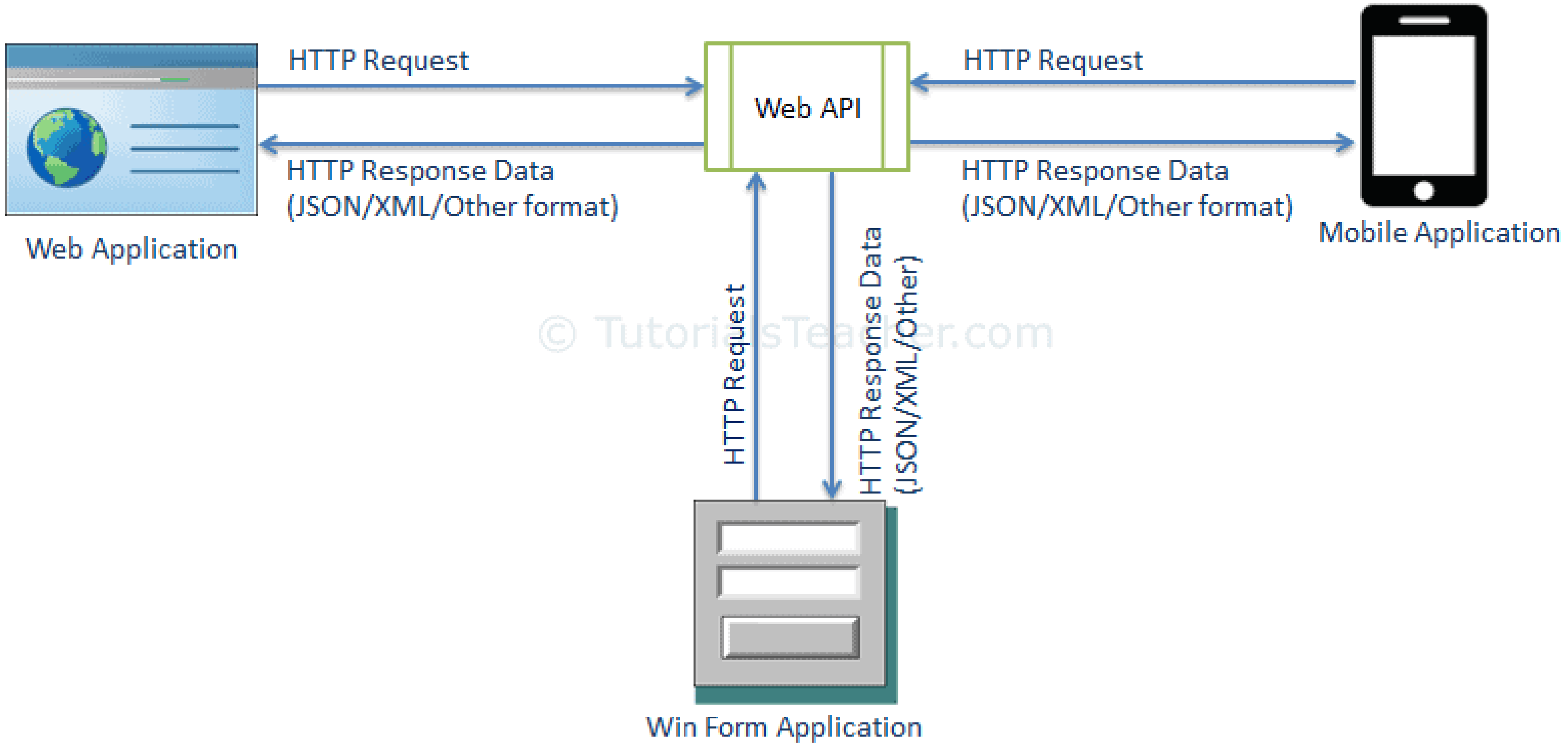


Let’s create first ASP.NET MVC Project in Visual Studio 2017/2019

ASP.NET Web API

- ASP.NET Web API is a framework for building HTTP services that can be accessed from any client including browsers and mobile devices.
- It is an ideal platform for building RESTful applications on the .NET Framework.
- It works more or less the same way as ASP.NET MVC web application except that it sends data as a response instead of html view.
- like a webservice or WCF service but the exception is that it only supports HTTP protocol.

ASP.NET Web API



ASP.NET Web API Characteristics

- a framework for building HTTP services that can be accessed from any client including browsers and mobile devices.
- ideal for building RESTful applications on the .NET Framework.
- The ASP.NET Web API is an extensible framework for building HTTP based services that can be accessed in different applications on different platforms such as web, windows, mobile etc.
- It works more or less the same way as ASP.NET MVC web application except that it sends data as a response instead of html view.
- like a webservice or WCF service but the exception is that it only supports HTTP protocol.

ASP.NET Web API Project

You can create a Web API project in two ways.

- Web API with MVC Project
- Stand-alone Web API Project

ASP.NET Core

- new version of the ASP.NET web framework
- free, open-source, and cross-platform framework
- ASP.NET Core applications can run on Windows, Linux, and Mac. So you don't need to build different apps for different platforms using different frameworks.
- allows you to use and manage modern UI frameworks such as AngularJS, ReactJS, Umber, Bootstrap, etc. using Bower (a package manager for the web).

.NET Core Vs ASP.NET Core

.NET Core	ASP.NET Core
Open-source and Cross-platform	Open-source and Cross-platform
.NET Core is a runtime to execute applications build on it.	ASP.NET Core is a web framework to build web apps, IoT apps, and mobile backends on the top of .NET Core or .NET Framework.
Install .NET Core Runtime to run applications and install .NET Core SDK to build applications.	There is no separate runtime and SDK are available for ASP.NET Core. .NET Core runtime and SDK includes ASP.NET Core libraries.
.NET Core 3.1 - latest version	ASP.NET Core 3.1 There is no separate versioning for ASP.NET Core. It is the same as .NET Core versions.

ASP.NET Core

- **Supports Multiple Platforms**
- **Hosting:** ASP.NET Core web application can be hosted on multiple platforms with any web server such as IIS, Apache etc. It is not dependent only on IIS as a standard .NET Framework.
- **Fast** - This reduces the request pipeline and improves performance and scalability.
- **IoC Container:** It includes the built-in IoC container for automatic dependency injection which makes it maintainable and testable.
- **Integration with Modern UI Frameworks**
- **Code Sharing:** allow to build a class library that can be used with other .NET frameworks such as .NET Framework 4.x or Mono. Thus a single code base can be shared across frameworks.

ASP.NET Core

- **Side-by-Side App Versioning:** ASP.NET Core runs on .NET Core, which supports the simultaneous running of multiple versions of applications.
- **Smaller Deployment Footprint:** ASP.NET Core application runs on .NET Core, which is smaller than the full .NET Framework. So, the application which uses only a part of .NET CoreFX will have a smaller deployment size. This reduces the deployment footprint.

Compilation and Execution of .NET applications: CLI, MSIL and CLR

- C# programs run on the .NET Framework, which includes the common language runtime (CLR) and a unified set of class libraries. The CLR is the commercial implementation by Microsoft of the common language infrastructure (CLI), an international standard that is the basis for creating execution and development environments in which languages and libraries work together seamlessly.
- Source code written in C# is compiled into an Microsoft Intermediate Language (MSIL) or simply(IS) that conforms to the CLI specification. The IL code are stored on disk in an executable file called an assembly, typically with an extension of .exe or .dll.
- CLR performs just in time (JIT) compilation to convert the IL code to native machine instructions. The CLR also provides other services related to automatic garbage collection, exception handling, and resource management.

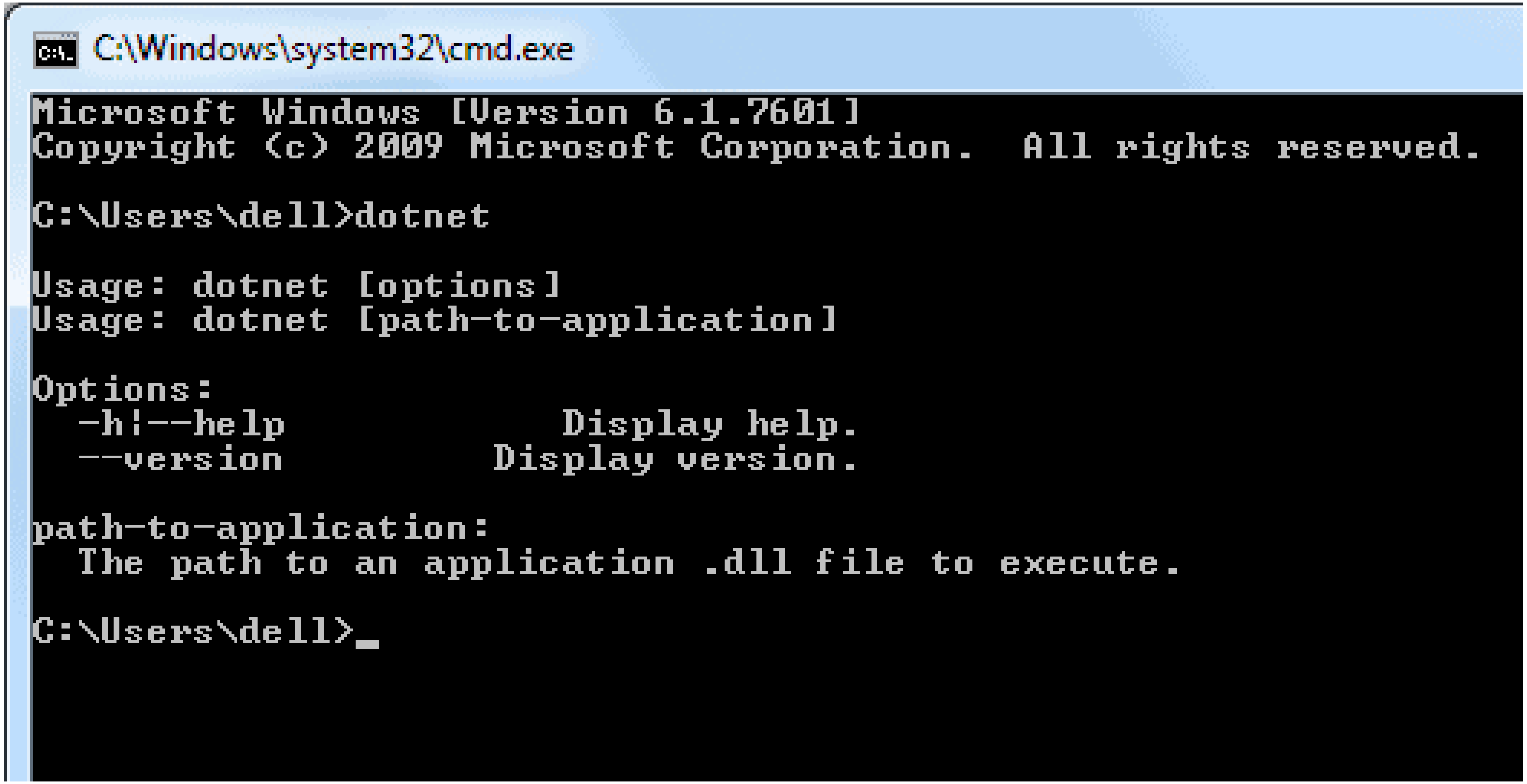
Compilation and Execution of .NET applications: CLI, MSIL and CLR

- Code that is executed by the CLR is sometimes referred to as "managed code," in contrast to "unmanaged code" which is compiled into native machine language that targets a specific system.
- Language interoperability is a key feature of the .NET Framework. Because the IL code produced by the C# compiler conforms to the Common Type Specification (CTS), IL code generated from C# can interact with code that was generated from the .NET versions of Visual Basic, Visual C++, or any of more than 20 other CTS-compliant languages. A single assembly may contain multiple modules written in different .NET languages, and the types can reference each other just as if they were written in the same language.

NET CLI: build, run, test and deploy .NET Core Applications

- The .NET Core command-line interface (CLI) is a new cross-platform tool for creating, restoring packages, building, running and publishing .NET applications.
- Visual Studio internally uses this CLI to restore, build and publish an application. Other higher level IDEs, editors and tools can use CLI to support .NET Core applications.
- The .NET Core CLI is installed with .NET Core SDK for selected platforms. So we don't need to install it separately on the development machine. We can verify whether the CLI is installed properly by opening command prompt in Windows and writing dotnet and pressing Enter. If it displays usage and help as shown below then it means it is installed properly.

NET CLI: build, run, test and deploy .NET Core Applications



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\dell>dotnet

Usage: dotnet [options]
Usage: dotnet [path-to-application]

Options:
  -h|--help                     Display help.
  --version                  Display version.

path-to-application:
  The path to an application .dll file to execute.

C:\Users\dell>_
```

Creating and running the Hello World console application

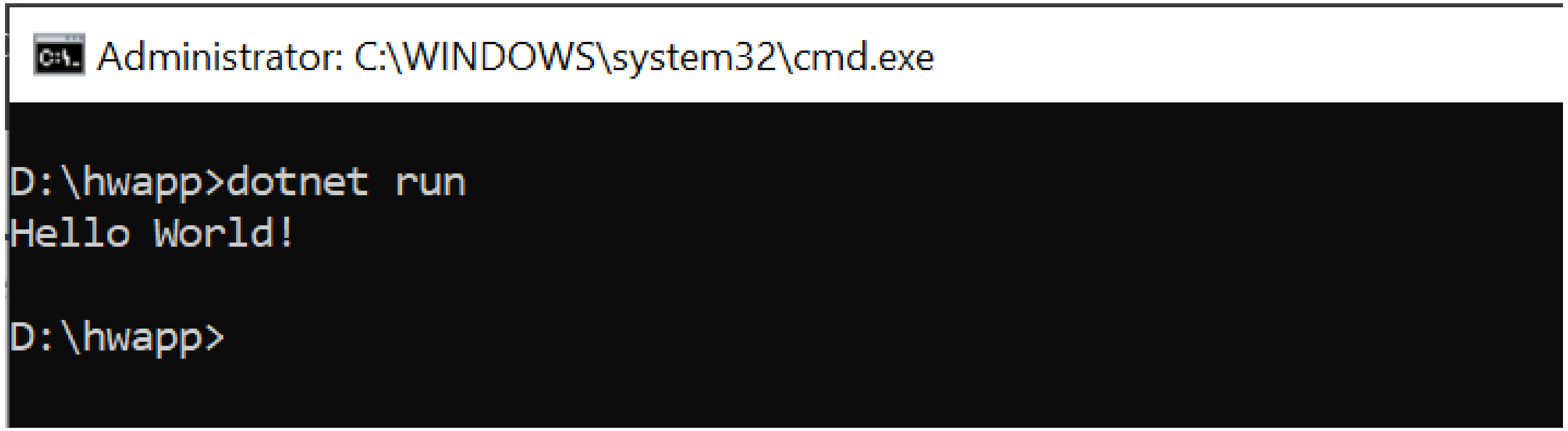
- Execute the following commands on the command line or terminal:
  - `mkdir hwapp`
  - `cd hwapp`
  - `dotnet new console`
- The command **dotnet new console** creates a new Hello World console application in the current folder.
- The dotnet new console command creates two files:
  - Program.cs and
  - hwapp.csproj

Program.cs should look similar to the following listing

```
using System;
namespace hwapp
{
  public class Program
  {
    public static void Main(string[] args)
    {
      Console.WriteLine("Hello World");
    }
  }
}
```

Running the Hello World console application

- When you’re using the .NET Core SDK, your application will be built automatically when needed. There’s no need to worry about whether or not you’re executing the latest code.
- Try running the Hello World application by executing *dotnet run* at the command line or terminal.



```
D:\hwapp>dotnet run
Hello World!

D:\hwapp>
```