**Web Development**

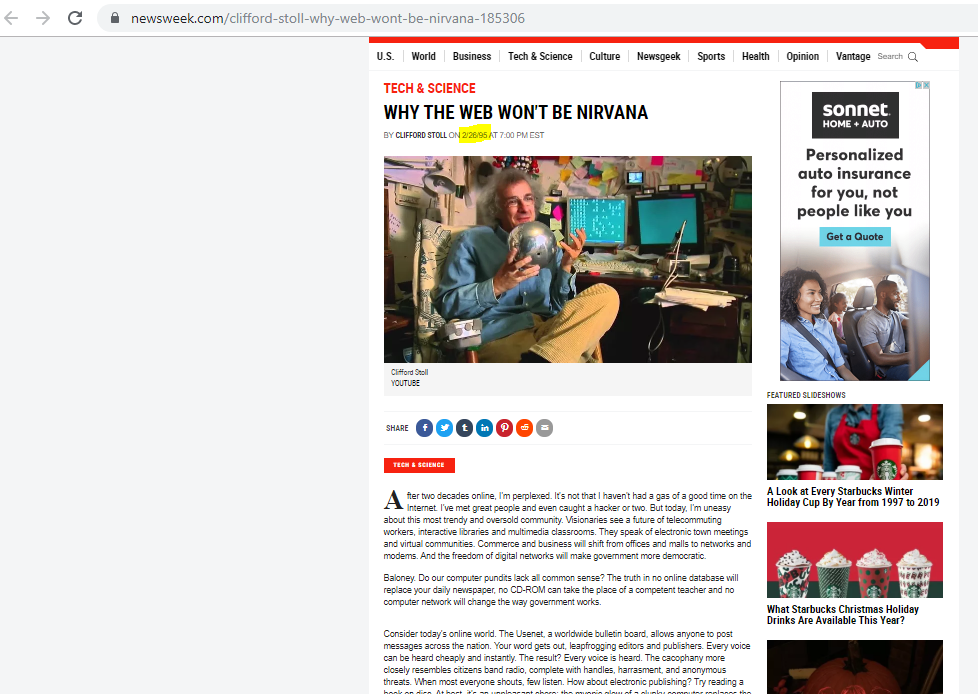
I am going to be teaching you how to build your own web application.

If you live in the modern world, you have probably noticed that has changed society in the last 50 years completely.

The Internet.

In the beginning there was doubt about it success. Newsweek even posted an article saying it would fail in 1995!

<https://www.newsweek.com/clifford-stoll-why-web-wont-be-nirvana-185306>



This was a year after banks first started to offer their services online!

Boy were they wrong. Ha! ha!

Furthermore, the dotcom bubble if you somehow missed it, was basically a period of time from 1995 – 2000 which the internet was widely adopted and grew in popularity massively!

Today, the internet is a part of our lives, and with it comes web sites and web applications. Furthermore, life now is significantly more complex, and the net knowledge level of people is much higher because of how widely accessible it is! The world the boomers grew up in vs the world of today was much more simpler.

Some people may want things to go back to how they used to be. But in reality, the internet has changed things and there is no going back. This hard reality was faced by Blockbuster going bankrupt. In fact, earlier they were approached with the concept of an online video rental store. They turned it down. Then Netflix came around and their business went crashing down like the tower of Babylon!

If you have a small business or have tried putting up a site of your own, you have probably heard of Wordpress, Wix and other such applications that help make putting up your own website simple and quick.

But that isn’t what we are going to be covering today. We want to be all star developers, right!? In order to do that, in my opinion we need to know the answer to the following questions:

1. What is a web site?
2. What is a web application and how does it differ from a web site?
3. Why use a web application vs a regular application vs a mobile application?
4. Why build your own website when you can utilize something like Wix or Wordpress which will be able to do the same thing as you with less work?
5. Why are web developers always switching up technologies and languages?
6. What are the benefits of one technology over another?
7. What does it take to think like a programmer or what makes programmers different from the general population?
8. How has the way web pages changed over time?
9. How is the way web pages are built changing in the future?
10. What are the expenses related to hosting a web application?

If you haven’t ever asked yourself those questions, you should. Asking the right questions, and finding accurate answers will make your life a lot easier. Furthermore, as we learn to develop web applications, I will bring up additional questions you should know the answers to which will make your development experience more pleasurable.

Remember stay engaged.

Understanding is a two way street, and I cannot know what you do not know or understand unless you ask the questions.

In order to succeed and become good at anything, you gotta have the three Ds:

1. Desire
2. Discipline
3. Drugs … kidding. Develop.

In my opinion knowledge of how to do things, always trumps definitions, however, knowledge of common terms and their meanings makes it easier to convey more complex information later on.

So lets get the answers to those question.

***Firstly, ‘What is a Web Site’?***

A web site is a page you go to that you cannot interact with. An example would be a blog without the ability to comment on it.

It can have images, multimedia, text, but whatever you do on that web site will not be recorded and will stay as is.

This leads to our next question, ‘What is a Web Application’?

A web application or dynamic web site is a web page that we can interact with.

It can have everything a web site has plus more. Such as:

* The ability for users to interact with a page and modify it so that people can see the changes.
* The ability to log in and access protected content.
* The ability to make purchases.
* The ability to control cameras remotely.
* The ability to edit videos.
* Etc …

Clearly, utilizing a web application opens doors! In fact, I’d dare to say that web applications were the reason for the majority of small business exploding into multi milion dollar businesses!

But in recent years, you might have heard how the smart phone mobile industry has exploded and there are more people utilizing mobile devices than laptops or computers!

***This leads to our next question, “Why use a web application vs a regular application vs a mobile application!?”***

Have you ever opened a program and had to wait a very long time before it opened?

The main difference between web applications and regular application is that web applications distribute the web work between a computer called the website host which sends you the web application and your computer. What a web application has access to on your computer is severely limited by the browser. Furthermore, any activities you perform locally are usually double checked by the server to make sure nothing is going wrong.

With regular applications, most of the work is done on your computer or in the case of mobile applications on your phone. Furthermore, regular applications/mobile usually have access to your local files and higher level resources with fewer controls and consequently, can cause more damage!

That last point is arguable but I wont get into it.

Also, if regular applications/mobile applications are communicating over different networks and use unique communication methods, they can run into unique problems. They also tend to be limited to working on certain devices!

Web applications usually have a standardized method of communication, are limited by browsers version, moreso than your hardware and consequently, you have less complexities to deal with if it is built correctly.

Furthermore, with applications you have to worry about the different versions used by your different customers and the incompatibilities that might arise from it. You just got to upgrade the web application on your servers and boom, all users will receive the updated version whenever they come to your site!

Consequently, I’d suggest using web applications over the other types of mobile and regular applications when computational intensity isn’t an issue and you are not limited by resources because simply overall it is less work for development, upgrading, and scenarios you need to compensate for.

You might know more and feel free to throw in your two cents or counter arguments.

***Now a question I’ve often been asked by clients & heard from students in classes that I have taken is this. Why build a web application from scratch when you can use a framework like Wix, or Wordpress to build out your site?***

My answer is that in numerous scenarios it is actually a good idea to utilize a framework such as Wix or Wordpress. But the minute you hope to integrate it with other systems to automate processes if the data is not exposed right, your your team is not trained with the proper sets of tools a level of complexity is added.

Furthermore, when working with wordpress/wix, creating custom functionality or compensating for unique situations may be more complicated of difficult for cases they did not think of.

Sites which are data driven, utilized as applications (there is a lot of UI interaction) or have number of users running on the cloud may cause heavy financial hits or perform poorly if a framework like Wordpress is used as of 2019.

This might change in the future.

So for small sites like a landing page, or a e-commerce store it actually might be simpler to use the pre-built modules provided by Wix and Wordpress, however, as your needs become more complex and you want to truly utilize the full power of a web application to say provide a webpage for individual people to remotely control devices in their houses, or monitor massive amounts of equipment and send alerts if things go wrong, or scan members of your organization’s id cards into a database and bring up their information. In these scenarios and much more, Wix and Wordpress are currently in my opinion not at the point where it would be a simple task to implement these things.

The main reason for Wordpress and Wix’s massive success is that the majority of companies that do utilize them, have needs that can be quickly met by them without having to learn as much since they are rather intuitive to use. Because of this, Wordpress exploded and 30% of websites on the internet use it.

It is the financially intelligent choice.

***This leads to my next question, “why are web developers always switching up technologies?”***

Technology and programming languages run in waves. People bet on a concept that is sold to them, learn it and implement solutions based on their perceived notion of it. Furthermore, new technologies usually have less that you need to learn and more flexibility! This makes it easier for people to pick up!

What you will notice as you go from company to company is that they often use the same technology in different ways with different libraries. The way of building things is not very standardized and as they are built there is a lot of learning involved. More often than not, people learn new technologies and use them in self defense or without having a thorough understanding of how it works and consequently, build incomplete or overly complicated solutions to a problem or use the libraries in ways they were not meant to be used.

But this is OKAY in my opinion. You don’t have to build something perfectly in order to solve a problem!

HOWEVER, knowing how to use good tools and the full scope of technologies while implementing a solution using best practices and standard tools will allow you to build it with less difficulties, faster, and with less difficulties.

***This brings us to our next question, “What are the benefits of one technology/programming language over another?”***

The smart way to determine whether you will switch to a new technology is to ask yourself whether utilizing that technology will **reduce your teams overall workload and issues downstream**. With the switching comes retraining costs and time.

Furthermore, with new technologies there is no guarantee that the people training you will teach its use in an optimal and the correct way! Newly created systems and frameworks usually have major flaws that aren’t fixed for the first two years of its release! So as a rule of thumb, before switching to a new technology, make sure the market tests it for at least two years. Examples include:

* 1. Bitcoin
  2. Visa
  3. The Newer Coca Cola Vending Machines
  4. The telephone booth
  5. The internet
  6. Wireless car locks
  7. Analogue cellphones
  8. Health care systems
  9. Retail stores
  10. Etc …

As more people gain experience in technologies and make the mistakes better curriculum and lessons plans come into play!

Furthermore, different programming languages have different strengths for different scenarios.

For example, languages like Matlab are better for algorithm and signal processing reasons.

Languages like CAD languages (AUTOCAD, VHDL etc…) are better for designing things.

Platforms like Unity are better for cross system game development.

Ask yourself what your goal is and choose the language appropriately. As time progresses new languages will come along to solve problems easily that were a lot of work to solve or code around previously.

So keep learning and finding faster, easier and better ways to do things.

***This brings me to my next question, “What does it take to think like a programmer or what makes programmers different from the general population?”***

Programming in my opinion is a new language that is simpler to learn than most spoken languages.

However, it does take grunt work. Just like you can’t learn the particularities of a language without using it as well as studying it in debt, the same applies moreso for programming languages since they tend to be more logical and have fewer contradictions.

It doesn’t take much to learn, in fact I encourage you to learn new languages on your own! However, in the process what you will learn is that you will build systems and only realise the things you could have done better in the aftermath!

Thus, while you can learn a programming language by trial and error and just doing it, just like you can with a spoken language, I suggest having a mentor that is knowledgeable and has made the mistakes so that you can learn from them rather than recreating the wheel. Furthermore, I believe it is beneficial to learn it as a whole on a high level before delving in and learning each languages particularities.

There is resistance to this view since programming as a science and being taught in schools is probably only 50 years old at most and has changed vastly since its inception.

Many old time programmers learn’t the language through older books and experiences they picked up by going from job to job. Consequently, if you look at many custom systems and how they are built at different companies you’ll notices differences in their implementations, programming patterns and approaches to solving similar problems differ based on how they implemented it. Furthermore, you will notice issues in design that tend to appear again and again.

With the advent of software engineering there has been a patchwork effort to rectify this. However, despite this, the majority of large systems in use built from antiquity were not been created by computer science or software engineering graduates but graduates of degrees predating them. People who learn’t from curriculums vastly different from what we have today.

Furthermore, in my opinion, many programmers and technicians love to over complicate things unnecessarily.

The industry is perfused why many people that use your ignorance to sell you on false concepts and ideas. This is why it is important to continuously learn.

The government is aware of this and as you have probably heard from the Snowden whistle blower controversy actively exploits these flaws on demand to monitor people of interest!

In general anything you send over the internet or through 3rd parties is usually not protected. Government wants control, and as a result owns the corporations that govern these things via the title crown corporation aka Canada Post, DNS etc ...

***This is linked to the next question,* *“How has website design changed over time”?***

Well in short, it has gotten simpler. Older web development languages required something called data sanitation, were susceptible by default to several security vulnerabilities like:

Cross site scripting, sql injection, directory traversal, web page defacing, privilege escalation, sniffing, remote code execution, over posting attacks, javascript hacks etc …

In fact, in order to build out a website without these issues, you’d have to have knowledge of them and compensate for them. This isn’t just web development languages, but older programming languages!

What protected a lot of infrastructures in the past was the general populations ignorance to how the technology worked. In reality, they were using systems in which their information was highly unprotected!

Even Microsoft is guilty of this! IIS a web serving platform prior to version 4 I believe was vulnerable to remote code execution and yet it was widely used by enterprise scale companies (big business like banks, insurance companies, software companies etc ...)

I chose Microsoft today, because I have often seen systems implemented with it over-complicated. Furthermore, there are a lot of misleading educational resources out there as well as high quality resources in existence. People learn’t the technologies or built things from scratch unnecessarily that were available out of the box.

But, if you know how to use it properly, you can build things faster and not run into as many issues.

Note that Microsoft has several web frameworks currently that vary in their use, their newest one being blazor. I will be teaching you Server side MVC coding.

The reason why I am not utilizing Blazor is because it doesn’t run currently with older browsers, only newer ones.

***So, knowing this, “where is web development headed in the future?”***

I believe that as time goes on, web development languages will converge into a common standard. One major dominating language that survives.

Most applications will migrate to web based platforms that can also run without a connection to the internet.

The language and frameworks which are easiest to learn, and in which you can build out products the fastest will dominate the market.

Web development will come back and overtake application developer jobs, data entry and repetitive jobs and mobile application development jobs. Companies that resist will be taken out of business.

I might be wrong, but this seems like the logical progression to me.

***This leads us to the next questions, “What are the expenses related to hosting a web application?”***

Now, if you plan on hosting a web application that will have many users, there are costs involved and you should know where you will be hit:

1. You pay for the server that you will store your web application on
2. You pay for space your website takes up
3. You pay for the domain name
4. You pay for the network usage
5. You pay for the software
6. You pay for the support
7. You pay for the processing power your application needs

The reasons often given for using the different frameworks/languages for building web applications are:

1. Reduced costs via:
   1. Reduced processing power
   2. Reduced software usage fees
   3. Reduced network bandwidth utilization
   4. Less hardware
   5. Licensing costs
2. Ease of integration with other systems
3. Reduced reliance on underlying technologies
4. Speed of software roll out

In the case of Software Engineers, they rarely state reduced workload or development overhead as a reason for this.

**How the Internet Works**

Whenever, you browse the internet you have imagined the internet to look something like the following:

|  |
| --- |
|  |

However, it is a bit more complicated than that and knowing the particularities will simplify your web development experience. The cool thing about the internet is that it is self documented, and each technical aspect of it is documented in something called a RFC or Request for comment. Just google it RFC 1 and you will see what I mean.

The internet on a software level is a bunch of tables on computers that act as traffic signs on where the data should go.

So there are a few key terms you should know before I show you the layout:

1. Network & Subnets

2. Packets

3. IP Address & DNS Tables

4. Switches, Routers and Gateways

5. Machine address & ARP Tables

6. Protocol (http, ftp, ...)

7. Firewall

8. Network traffic monitors (OWASP, Wireshark, etc …)

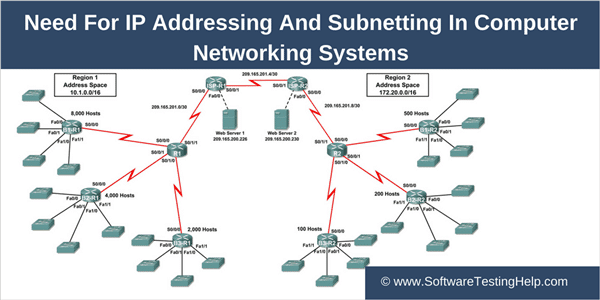
9. HTML & CSS & Front End Languages (Angular, React, Jquery, JavaScript, Web Assembly)

10. Back End Languages (SQL, .NET, Java, NodeJs, Python, etc ...)

11. Databases

**1. Network & Subnets**

A network is a bunch of interconnected computers. A subnet is a smaller grouping of computers that connects to the larger group of computers.



There is more to it, but if you want to know more learn about networking.

**2. Packets**

Everything sent over a network even a message is broken up into smaller data segments called a packet which is arranged in such a way according to a language called a protocol that is understood by the sending and receiving computer.

A packet is a bunch of ones and zeros and varies in size up to 64000 bits I believe, though I might be wrong.

In summary it is a unit of data used for communicating between two computers.

**3. Ip Address & DNS Servers**

An IP Address is like your home address, except it is the address of your computer on a network!

An Internet Protocol address is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication. An IP address serves two main functions: host or network interface identification and location addressing.

So whenever you enter a web address in your browser, a request or inquiry for the ip address is sent to something called DNS which is basically a data table that specifies what that web addresses actual ip address is!

If you change this or intercept this request responding to it instead of allowing the true the DNS server to, or incorrectly update the DNS table on the web, you can redirect the user to a page of your choice! A fake page that can be used to steal a person’s information without them knowing!

DNS servers exist within a business level, a ISP level, a webhost and even a country wide level. Often times they are called name servers and these name servers communicate with other name servers to update them.

**4. Switches, Routers & Gateway IP**

Switches are devices used to connect computers within a network and routers are devices used to connect computers to a network. They do this based on the IP Address.

A gateway ip is basically a term used to describe the ip address that data all data within a network must flow through in order to reach a larger network.

Think of it like an exit/entrance to a room that only has one door.

**5. Machine addresses & ARP Tables**

This isn’t a term you often hear. But they are important.

Your computer’s unique identifier is something called a Machine address. Your router is actually what determines its IP Address based on how you decide to set it up!

An ARP table is basically a table that maps your IP Address to Machine address for communications sakes!

So the internet uses IP Addresses & DNS servers, while your home network of computers relies mainly on a ARP table local and on the switch/router to determine whether data is meant for you or someone else!

If you forceably update this information on another computer to make it think that your computer is the gateway, via an exploit called ARP poisoning (a type of man in the middle attack), you can monitor all the data coming in and out of that computer!

6. **Protocols**

|  |
| --- |
|  |

A protocol is basically a standardized method of communication like English with a set of communication rules. There are levels of protocols, and the higher levels use the lower levels to communicate and compensate for things automatically that the lower level protocols do not.

This is all you need to know. Often times people will use the OSI model to teach how these protocols work, but I wont get into that.

Examples of higher level protocols are: http, ftp, …

Yes, you specify the protocol in your browser when trying to access websites!

**7. Firewall**

A firewall is a network security system that monitors/controls incoming and outgoing traffic via a set of rules. It can run on your computer or on routers and switches.

If your website is unable to be reached when installed on a server from another computer, it is usually either a problem with the firewall configuration or DNS server (unless someone is ARP poisoning your network).

This is more common than you’d think since security often changes configurations or the network team may also change stuff.

8. **Network Traffic Monitors**

You can use tools to view the packets that are sent over a network if your modem is in promiscuous mode.

These tools are called Network Traffic Monitors or Packet sniffers.

The most popular of these is Wireshark.

You can use this to identify website issues.

**9. HTML & CSS & Front End Languages (Bootstrap, Angular, React, JQuery, JavaScript, Web Assembly)**

The above are a bunch of standardized languages utilized by your browser.

HTML or Hypertext markup language is pretty universal on the internet and is a markup language sent to browser to specify data and its attributes. It predates CSS and was one of the first languages used by browsers.

CSS or Cascading Style Sheets is a language used for defining the appearance of your site. Initially, HTML was the only thing available but it proved inadequate for more complex graphical layouts. So in 1994, CSS was introduced and since has developed to allow for more user friendly website layouts!

Front end languages, are languages that run on the computer viewing a website. Often times they are used to reduce processing and work done by the computer hosting the website and pushing it off onto the computer that is supposed to view it!

Bootstrap, Angular, React, Jquery, JavaScript and Web Assembly are all examples of frameworks that allow for this to be done.

There are higher level front end languages and lower level, most front end languages are built on JavaScript although this might change to Web Assembly in the foreseeable future. The higher level languages also compensate for things the lower level languages do not compensate for automatically, like cross browser compatibility, responsively designed (works on different sized screens and devices), and more.

Another short coming of front end languages is that they are insecure by design. Anything that is sent to a computer can be modified or manipulated prior to being sent back, so you can’t trust anything from the client.

**10.** **Backend Languages**

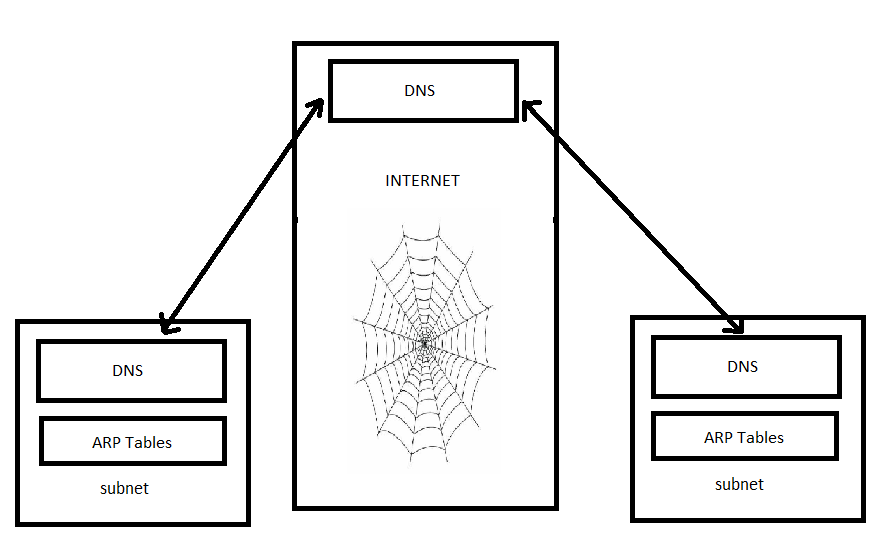
Backend languages are the languages that run on the server!

C#.NET is an example of this!

**11. Databases**

A database is a repository of data. A bunch of tables that hold useful information. This is where our data will be stored!

So knowing this, the internet looks more like that following:



With many subnets, connecting to the main internet. A tree of DNS servers that communicate and update each other, and a web which represents a web of routers that redirects data to their proper location based on ip addresses!

Your computer and the computer hosting the website will be located in different subnets.

But all of that isn’t that important with relation to what it coming up next. I just wanted you to understand these things because you will likely come across them in your web development career.

Now onto ASP.NET Core!

**The structure of a website in ASP.NET Core**

.NET is a library created by Microsoft for Web and Application development. There are several versions of it and it wasn’t always called the .NET library!

It has changed a lot over time and if you look at a program’s structure from its inception to current day, you’ll notice vast differences in its layouts.

In fact when you hear .NET4 or .NET 3 or .NET 2.

The number after .NET is stating a changing of the language specification or standard of the framework.

Furthermore, with Microsoft products, very rarely will you see the people that using it, using the full power of the suite. Why? Because they want you to pay for everything!

For example for older .NET builds, you could rapidly build out data driven server side Windows Forms applications if you knew how to use SSIS, SSRS, and SSAS. But you had to pay for all those licenses. Furthermore, they have a testing suite for automated testing, performance testing tools, database migration and comparison tools and much more!

They are a force to recon with if you know how to use their products efficiently and can afford to. But even in big businesses with the older tools, they rarely did. But the time of these tools is passing.

.NET Core is a rebuild of the .NET framework and it is free to use and runs on Windows and Linux. For how long? I don’t know. The developers of the .NET framework found it had been getting too bulky and hard to keep updating. It was slow because of a lot of unnecessarily bulky code in it. Furthermore, the developers realized the poor implementations that they had done while building it out.

Consequently, they redid it so it is less bulky, faster and free!

It has some things that the older framework had, but not all since it is still relatively new.

In this class we will be using .NET Core 3.0 and building a C# MVC Application with Entity Framework. Note the version because as I said, the language specifications and site structure change with each implementation.

What is MVC you may ask?

It stands for Model-View-Controller and is one way of structuring and organizing your web application.

Coincidently, this is also the three main types of files types this kind of project uses for structuring itself.

Views files are the files that contain actual website layout and markup. If you have used HTML before, these are the equivalent of HTML files, but with C#.NET they are CSHTML files.

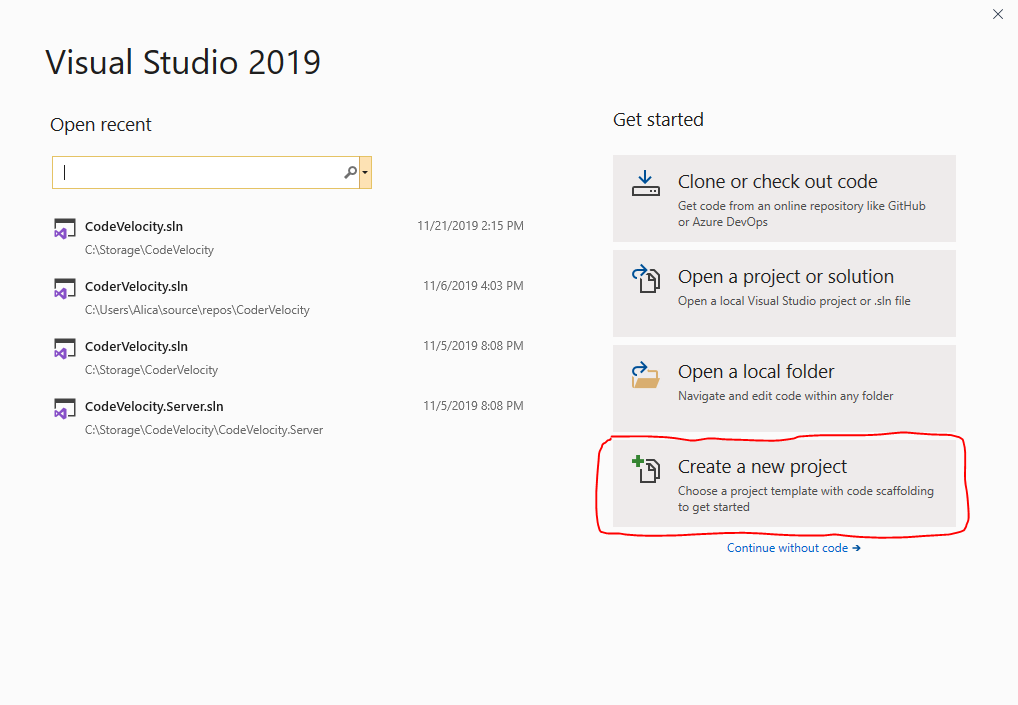
Controllers are files where you process information and you use to communicate with the database. These files contain the business logic and processing code.

Lastly, models are used to describe the structure of your database!

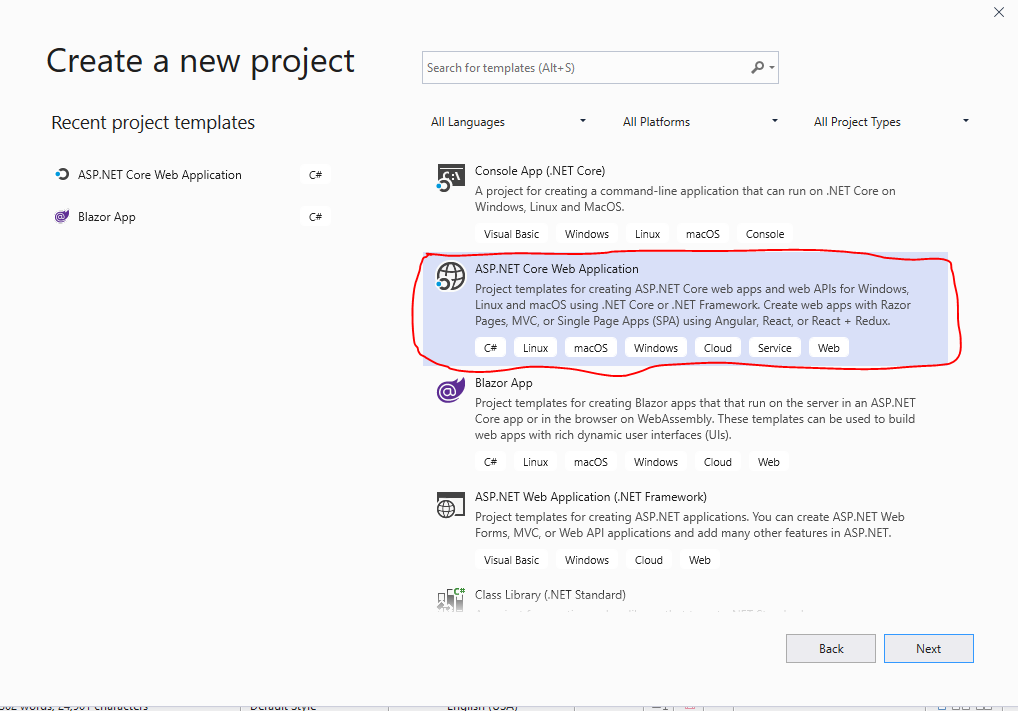
Furthermore, we will be using a library called Identity Framework in order to control people’s roles and access to the different pages!

The benefit of using a framework versus building your own website is that it often compensates for things without us having to about them or knowing the details, like Facebook and Instagram integration, or security, or time/region based access controls. Knowing how to use a framework and using reusable components saves times. But the drawback is that if the developers made a mistake & it is publicized, the bug will be widely known & many people could exploit it if it is a customer facing situation in worst case scenarios!

So now, open Visual Studio 2019 and click Create New Project:



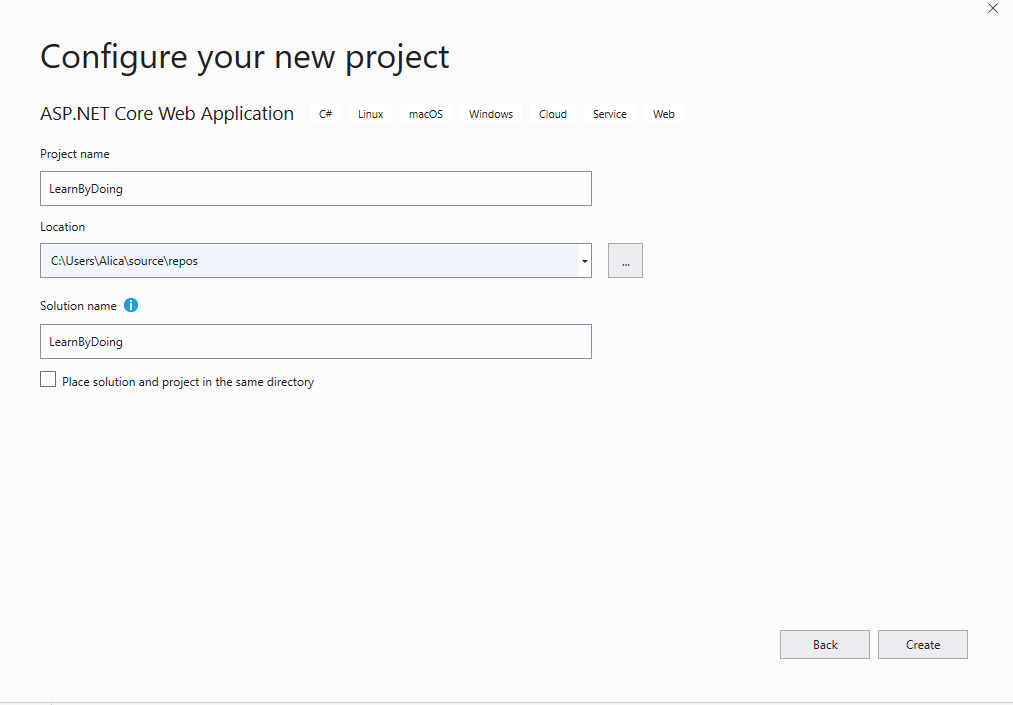
Next, click ASP.NET Core Web Application:



The way you access this is different based on the version of Visual Studio you utilize.

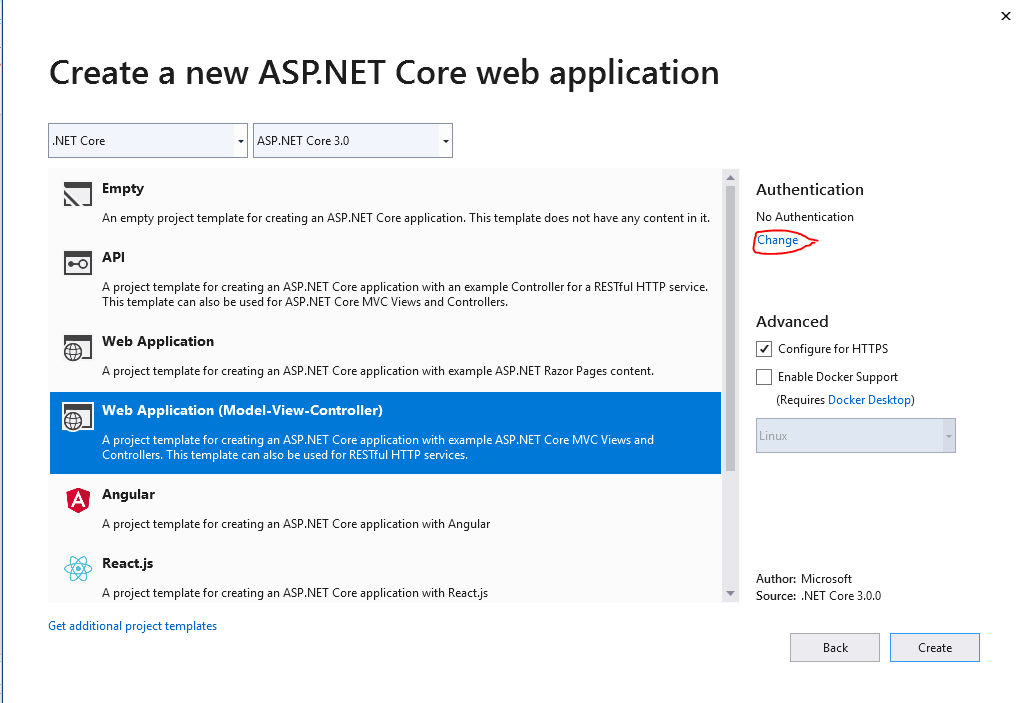
Select it and click Next!

On the new screen, fill out the text boxes as follows:

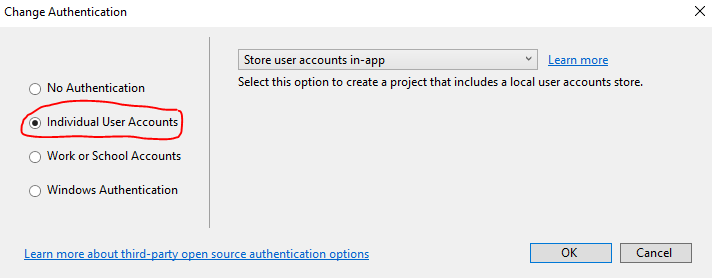


Then press **Create**.

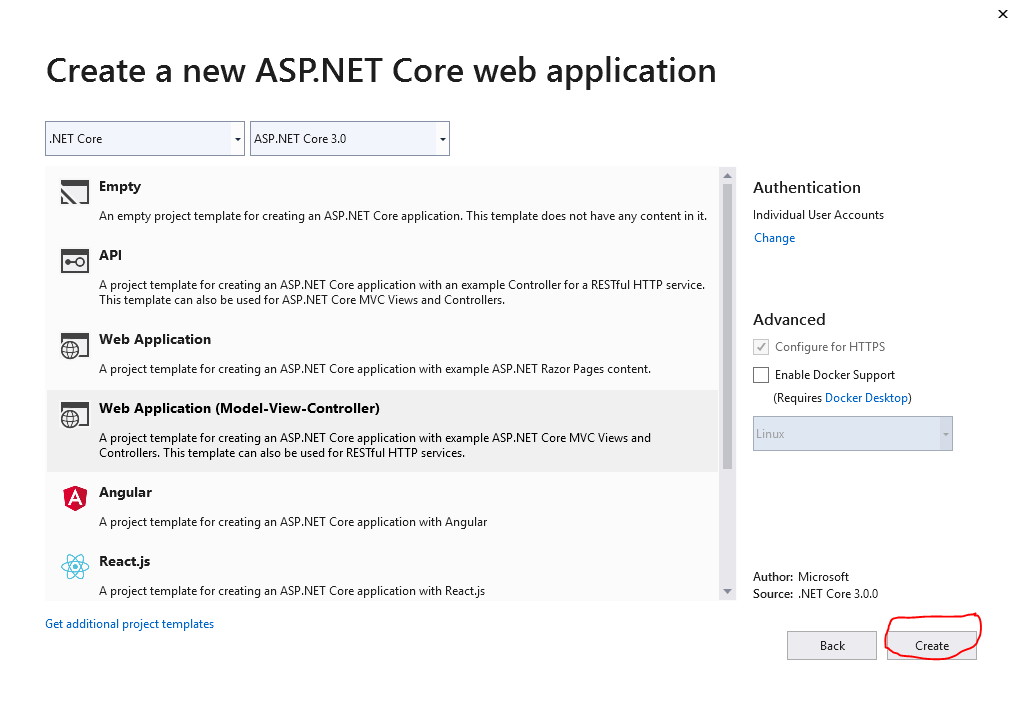
Next you will be taken to this screen:



Click **Change**.



On this screen, select Individual Accounts and click **OK**

****

Then press **Create**.

Visual studios will create your basic files, folders, logins and a sample web page using the MVC structure!

The file structure will be shown in the Solution Explorer panel as shown below!

|  |
| --- |
|  |

We see the following folders and files:

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Startup.cs | File | Startup contains your web application configurations, services and middleware. We will examine it closer later. |
| Program.cs | File | First file to run when your web application starts up. This fires off Startup. Rarely is this modified, so we won’t be going into this but the main function in this file is the first thing to be fired off. |
| Appsettings.json | File | This contains configuration information for your application like database connection strings, and other data you don’t want in the code. |
| Views | Folder | This folder will contain your view files. |
| Models | Folder | This folder will contain your models files. |
| Controllers | Folder | This folder will contain your controllers files. |
| Data | Folder | This folder will contain your database related files. |
| Areas | Folder | This folder is created because we are using the identity framework. |

Startup.cs

This file is where a lot of the magic happens. Here we define our middleware, services, and configurations.

You might be asking yourself what middleware is. Put simply, it is basically a piece of code that runs whenever you make a web page request to a server. It simplifies security, logging, and a host of other things that you may have otherwise had to implement in several pages.

|  |
| --- |
| An example of a browser request |

Next what you might be asking is, “What is a service?”

Well services in ASP.NET Core are a pretty cool thing. You might not get them now, but it will make sense when we actually look at code.

Services are used to define the life span of objects created in ASP.NET core.

There are three types:

1. Transient
2. Singleton
3. Scoped

When creating a web application, there might be times when you will want to create an independent object that is shared amongst all people that connect to your website, or that is reused as long as a person is accessing your website or is unique for every request that is made to your website.

|  |  |
| --- | --- |
| **Service Type** | **Use** |
| Transient | They are newly created each time a service is requested |
| Singleton | This is only created once, and is reused every time it is requested. |
| Scoped | This is created once within a session and is unique for each session. However, it is reused within that session. |

Using services correctly can save your computer memory and