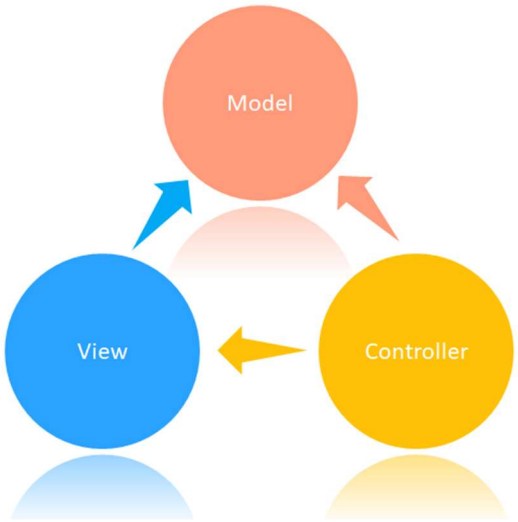
MODULE 1.1

What Is the MVC Pattern?



- The Model-View-Controller MVC architectural pattern divides an application into three major component groups: Models, Views, and Controllers.
- The principle behind developing MVC is the separation of concerns. This principle implies that software should be classified based on the type of task it performs.
- Separation ensures that the business model is simple to test and can change without becoming too involved in low-level implementation details.
- Using this MVC Pattern, user requests are routed to a Controller, who is in charge of interacting with the Model to conduct user actions and receive query results. The Controller selects the View to be displayed to the user and supplies it with any Model data it requires

What Is ASP.NET Core MVC?

- The ASP.NET Core MVC framework is a lightweight, open source, and robustly configurable presentation framework for ASP.NET Core.
- It is an Application Framework that allows you to create modern web apps using the MVC (Model View Controller) architectural pattern.
- It provides a pattern-based approach to developing dynamic websites of particular concerns. It allows you complete markup control, facilitates (Test-driven development)TDD-friendly development, and corresponds to the latest web standards.

Responsibilities/Functions of MVC

Model Responsibilities

- In an MVC application, the model represents the application's state and any other business logic or activities that should be executed by it.
- Business logic, and any implementation logic for maintaining the application's state, should be wrap
-
- ed in the model.
- ViewModel classes intended to contain the data to display on that view are often used in strongly-typed views. The controller uses the model to construct and populate these ViewModel instances.

View Responsibilities

- Views are in charge of displaying content through the user interface. The embed .NET code in HTML markup using the razor view engine.
- Views should have minimal logic, and any logic should be related to presenting content. Consider using a view component, view model, or view template to simplify the view if you need to perform logic in view files to display data from a complex model.

Controller Responsibilities

- Controllers are the components that manage user input, interact with the model, and choose which view to render.
- The view displays information in an MVC application; the controller processes and responds to user input and interaction.

- The controller is the initial entry point in the MVC architecture, and it is responsible for deciding which model types to interact with and which view to present. Hence its name is the controller as it responds to a given request.

The All Time Best Tutorial to Understand ASP.NET Core MVC

Lesson 11 of 15By Ravikiran A S

Last updated on Jul 26, 202455040

PreviousNext

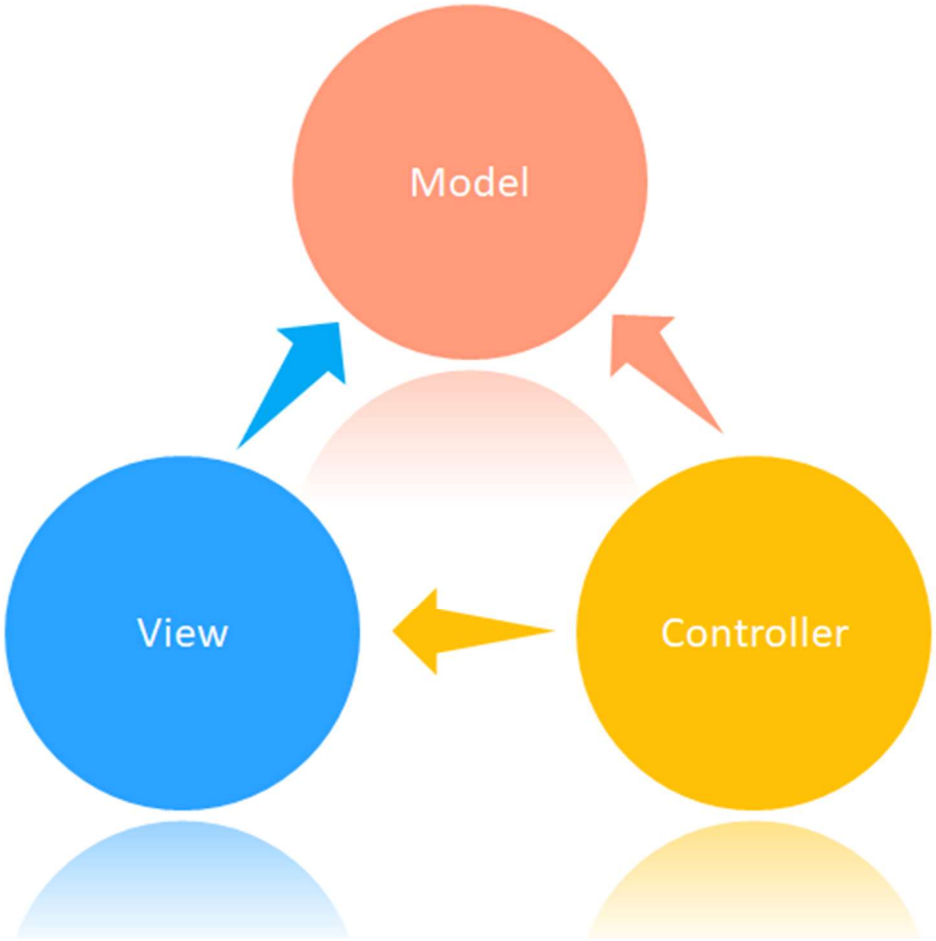
Table of Contents

The ASP.NET Core framework is growing in popularity among developers and is anticipated to continue to do so in the future. On the .NET platform, ASP.NET is a well-known web development framework for constructing web applications. ASP.NET Core MVC is a robust framework for building online programs and APIs using the Model-View-Controller architectural pattern. You've come to the right place if you want to learn about ASP.NET Core. This article will provide you with a quick overview of ASP.NET Core MVC.

Want a Top Software Development Job? Start Here!

Full Stack Developer - MERN StackExplore Program

What Is the MVC Pattern?



- The Model-View-Controller MVC architectural pattern divides an application into three major component groups: Models, Views, and Controllers.
- The principle behind developing MVC is the separation of concerns. This principle implies that software should be classified based on the type of task it performs.
- Separation ensures that the business model is simple to test and can change without becoming too involved in low-level implementation details.
- Using this MVC Pattern, user requests are routed to a Controller, who is in charge of interacting with the Model to conduct user actions and receive query results. The Controller selects the View to be displayed to the user and supplies it with any Model data it requires.

Want a Top Software Development Job? Start Here!

Full Stack Developer - MERN StackExplore Program

What Is ASP.NET Core MVC?

Responsibilities/Functions of MVC

Model Responsibilities

- In an MVC application, the model represents the application's state and any other business logic or activities that should be executed by it.
- Business logic, and any implementation logic for maintaining the application's state, should be wrapped in the model.
- ViewModel classes intended to contain the data to display on that view are often used in strongly-typed views. The controller uses the model to construct and populate these ViewModel instances.

View Responsibilities

- Views are in charge of displaying content through the user interface. The embed .NET code in HTML markup using the razor view engine.
- Views should have minimal logic, and any logic should be related to presenting content. Consider using a view component, view model, or view template to simplify

Components of ASP.NET MVC

Routing

- ASP.NET Core MVC is built on top of ASP.NET Core routing, a robust URL-mapping component that enables applications with logical and searchable URLs.
- This allows us to establish URL naming patterns for applications that perform well for search engine optimization (SEO) and link generation, regardless of how the files on your web server are arranged.
- You may define your routes with a simple route template syntax that considers route value limitations, defaults, and optional values.

Model Binding

- Model binding in ASP.NET MVC transforms client request data (form values, route data, query string parameters, HTTP headers) into objects that the controller can handle.
- As a result, your controller logic does not have to do the job of determining the incoming request data; instead, the data is passed as parameters to its action methods.

Model Validation

Validation is supported in ASP.NET Core by decorating your model object with data annotation validation attributes. Before values are sent to the server, the validation attributes are checked on the client-side. The framework validates request data on both the client and the server.

Dependency Injection

- ASP.NET Core includes dependency injection capabilities. Controllers in ASP.NET Core can request required services through their constructors, allowing them to follow the Explicit Dependencies Principle.
- Dependency injection (DI) is a software design technique that achieves Inversion of Control (IoC) between classes and their dependencies.

Filters

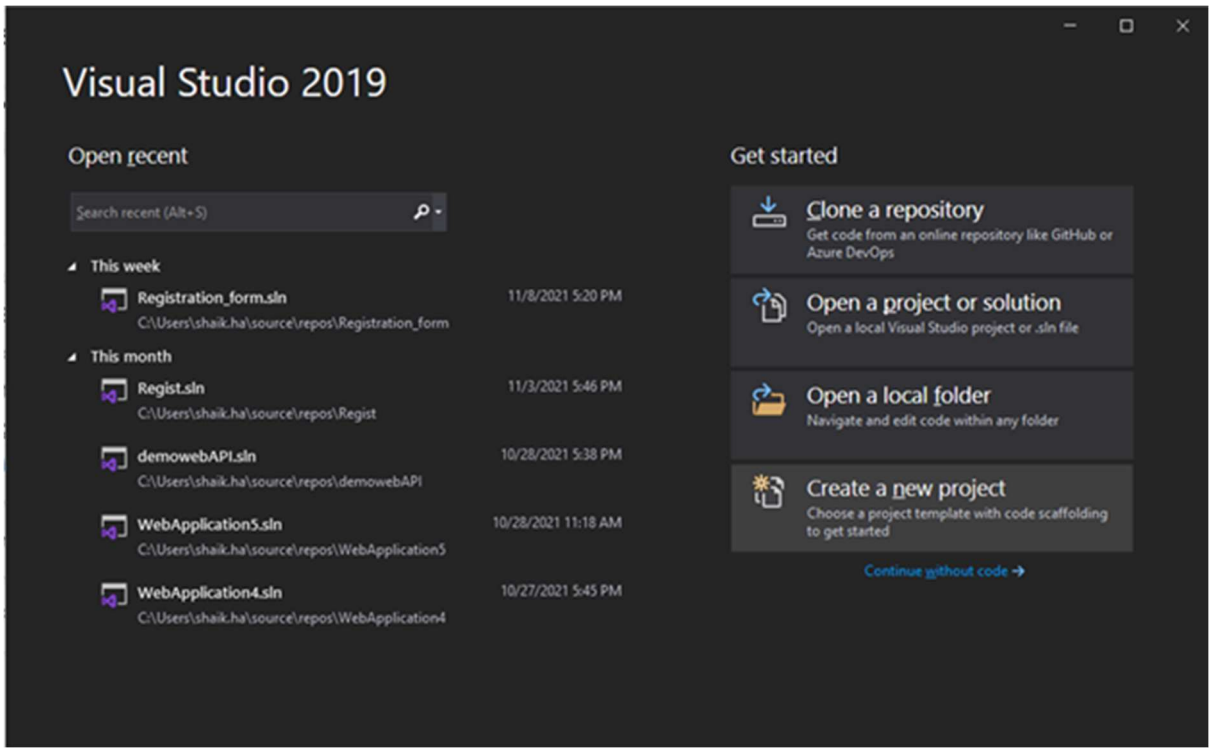
- Filters in ASP.NET Core enable code to be executed before or after particular stages of the request processing pipeline. Filters can be used as attributes on controllers or actions.
- Filters manage duties such as authorization, restricting unauthorized access to resources, and response caching.
- Filters allow for the execution of custom pre and post-processing logic for action methods, and they may be set to run at specific points in the execution pipeline for a given request.

Areas

- Areas allow you to divide a large ASP.NET Core MVC Web app into smaller functional groups. An area is an MVC structure that exists within an application.
- Logical components such as Model, Controller, and View are kept in separate folders in an MVC project, and MVC uses naming conventions to build the relationship between these components.
- It may be good to partition a large app into different high-level functional regions.
- For example, consider an e-commerce app with numerous business units such as checkout, billing, and search, among others. Each of these units has its own set of logical component views, controllers, and models.

Demo: Creating an ASP.NET Core MVC Web Application

Step 1: Open Visual Studio 2019 and Select Create a New Project

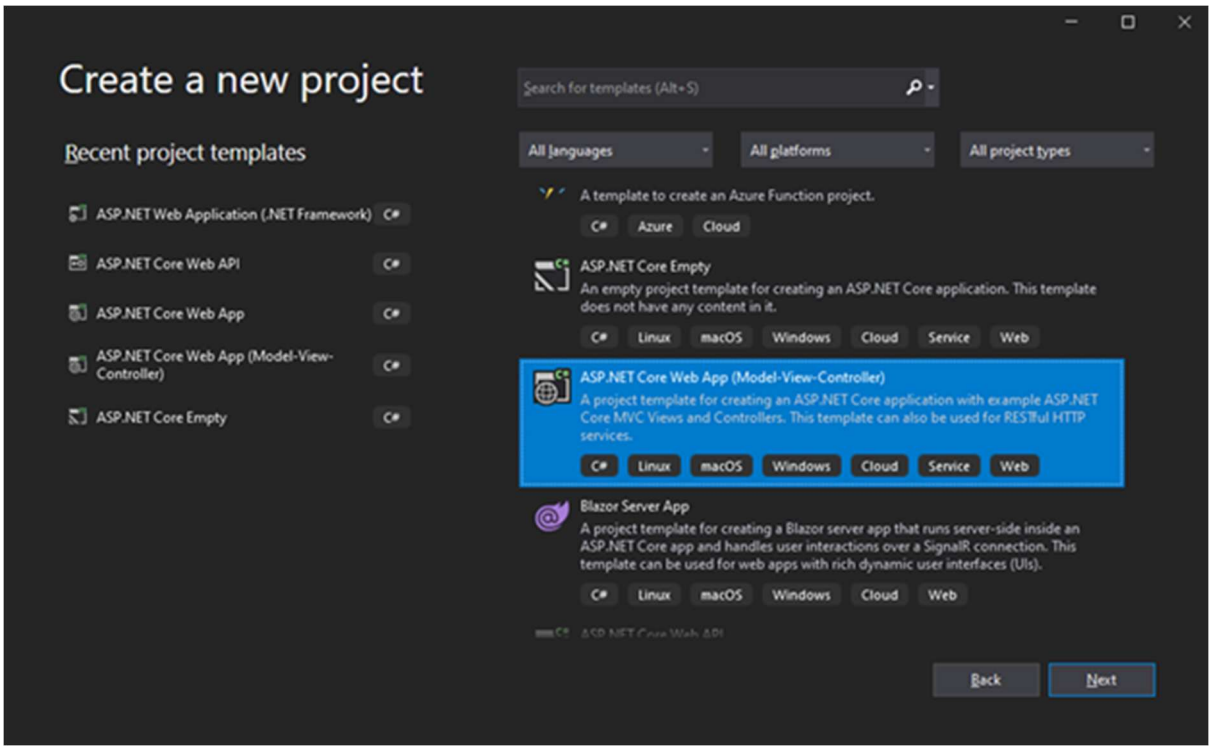


Once the visual studio is opened on the left-hand side, you can access the recently opened files.

And on the right-hand side, you have the following options:

- Clone a Repository: Get code from online repositories like Github and Azure
- Open a Project: Open a local visual studio project or .sln file
- Open a local folder: Navigate and edit code within any folder
- Create a new Project: Choose a project template to get started
- Select the option to create a new project.

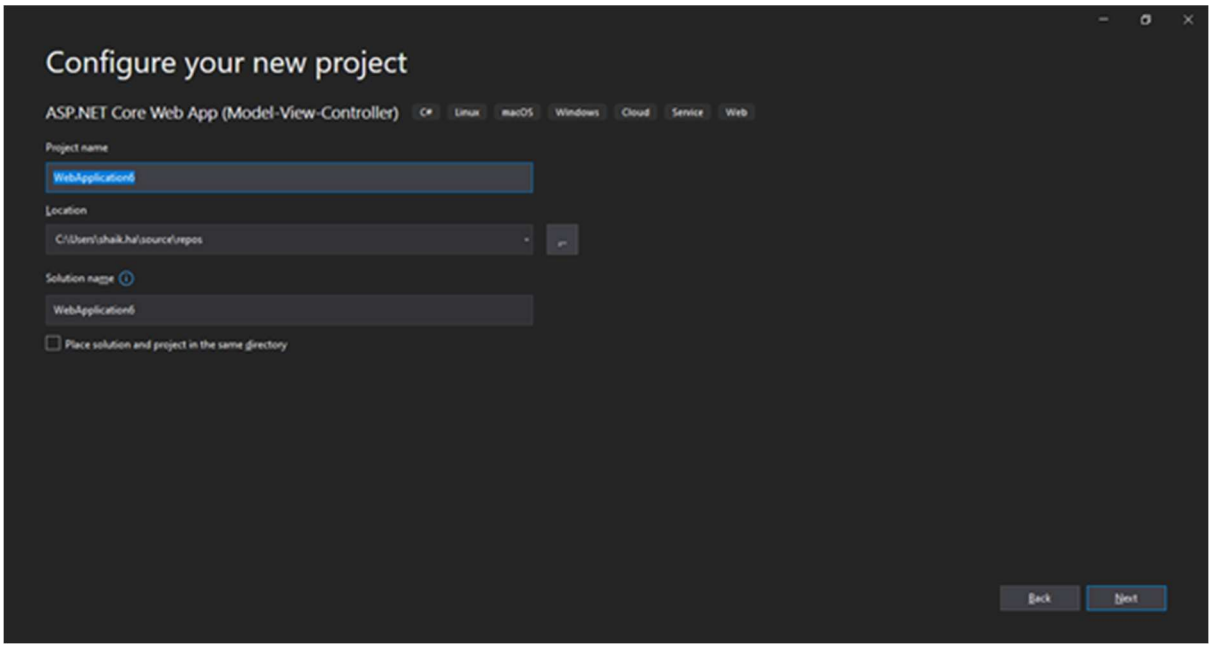
Step 2: Choose the Project Template



You will see a lot of templates available. But for creating an ASP.NET Core MVC Web application, select ASP.NET Core Web App(Model-View-Controller).

ASP.NET Core Web App(Model-View-Controller) is a project template for creating an ASP.NET Core application with an example, ASP.NET Core Views and Controllers. This template is also used for RESTful HTTP services.

Step 3: Fill in Project Name and Location



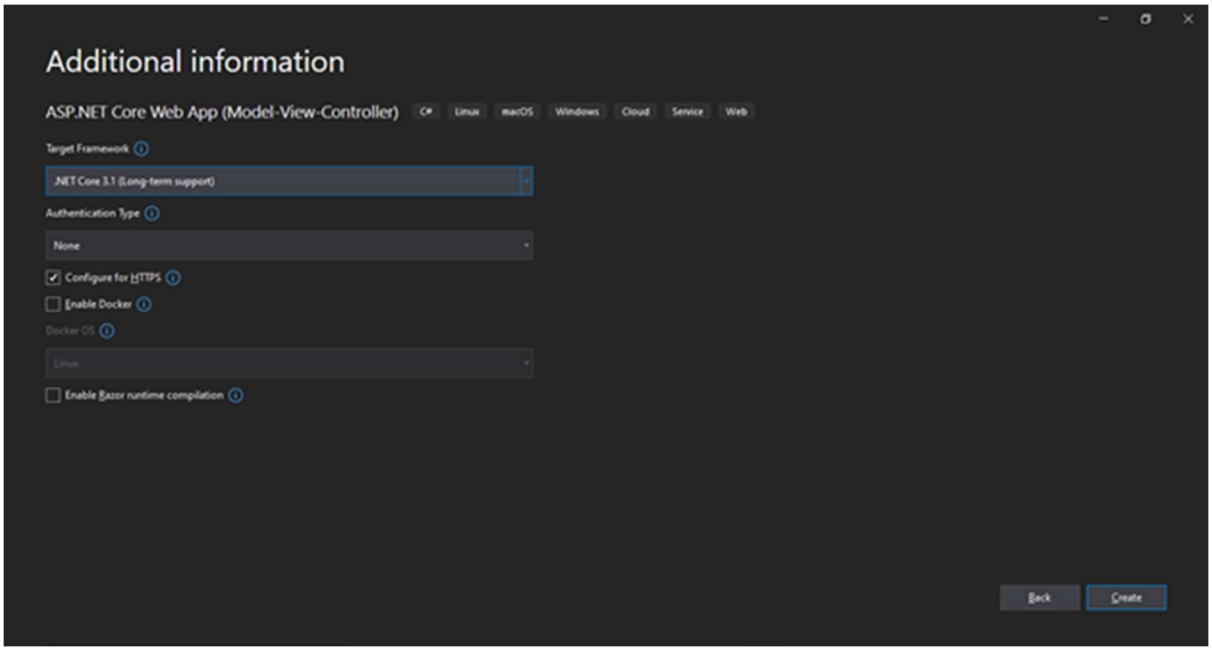
Configure your new project window consists of three options, namely

Project Name: Enter the required project name

Location: Select the location where you want to save your project

Solution Name: By default, the solution name is the same as the project name, But we can change it if required.

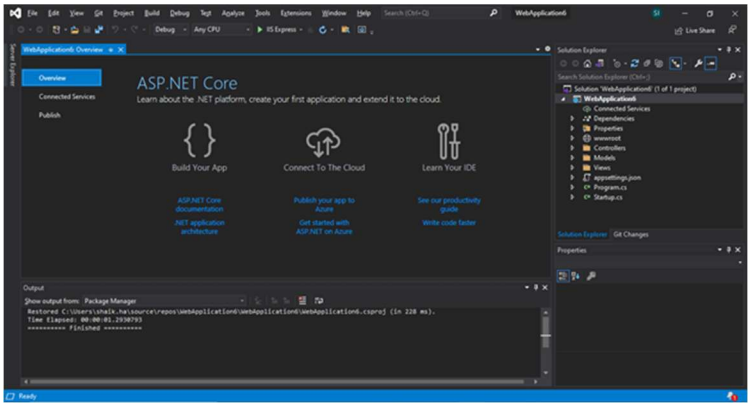
Step 4: Choose the Target Framework and Authentication Type



Choose the target framework .Net 2.1, .Net 3.1 or .NET 5.0 the latest version installed in your device. Also, select the authentication type and make the other changes as your project requirement. After entering the details, click the create option.

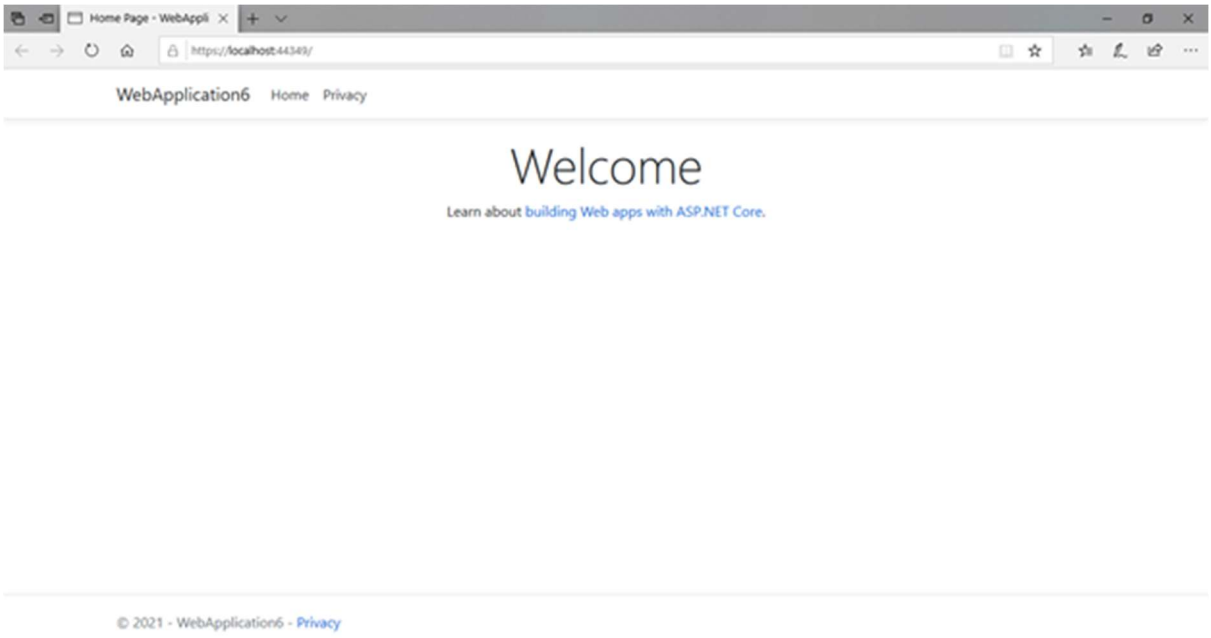
Step 5: The ASP.NET Core Web Application MVC is Created

Once the application is successfully created, click on solution explorer. The following folder list response is seen as shown in the figure below.



Step 6: Run the Application

Now run the application with default contents or put some contents, thereby opening the index.cshtml file. Now press the run option to run the application in the browser. Once the application is successfully built, it will show in the browser, as shown in the image below.



Next Steps

This article on 'ASP.NET Core MVC' covers what MVC Pattern and ASP.NET MVC are. Following which it gives a detailed explanation of responsibilities and architecture of ASP.NET Core. Lastly, we have a Demo to get started with our first ASP.NET MVC Application.

Take the Simplilearn Full Stack Developer - MERN Stack if you wish to learn.NET. This course will teach you how to learn the principles of.NET programming and how to create.NET projects. The.NET programming certification course will introduce you to the.NET space and C# coding, Visual Studio and Webforms, which will help you succeed in your profession.

Please visit our YouTube channel for a thorough explanation of ASP.NET Core MVC if you require additional information.

If you have any questions about the article ASP.NET Core MVC, please post them in the "ASP.NET Core MVC" Tutorial's comments section. We'll get back to you right away with a response from one of our experts!

MODULE 1.2

Why ASP.NET?

- ASP.NET reduces all the issues that come up while building a web application like speed, cost, and language.

- ASP.NET provides multiple development modes, which help to develop applications in an easy and better way.

- ASP.NET works on an HTTP protocol and uses HTTP commands.

- ASP.NET provides a platform that allows writing a code in a text editor program and Visual Studio .NET.

- If you are building an application, ASP.NET could be the best choice as it is faster and more efficient than other technologies.

Components of ASP.NET

ASP.NET depends on three major components:

1. Language

Language helps in the creation of web applications. Some languages that ASP.NET uses for development are VB.Net and C#.

2. Library

There are different types of libraries with all the components to help developers and create applications.

3. Common Language Runtime

It is a platform that helps to execute the .Net programs. It is used for running key activities like exception handling and garbage collection.

Tools Used for ASP.NET

1. Microsoft Web Platform Installer

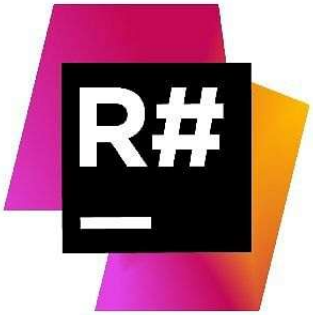
Microsoft Web Platform Installer is a free tool that can efficiently run web applications and get the latest components like SQL Server Express, .NET Framework, and Visual Studio.

2. Web Essentials for Visual Studio

Web Essentials is a tool, that opens up the inventory of CSS, Html, JavaScript, TypeScript. It makes developers easier to build by extending Visual Studio.

3. ReSharper

Resharper is a tool that provides an absolute productivity boost in code quality analysis and helps to reduce time by identifying errors.



4. LINQPad

LINQPad is a tool that helps in testing LINQ queries. It also provides instant feedback. It comes up with built-in features like the debugger and autocomplete.

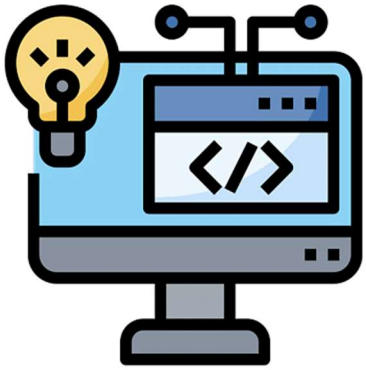
5. NDepend

NDepend is used for the optimization of code and measuring the code quality. It is mainly used for static code analysis. This tool also provides a custom querying language for examining the application.

Advantages of Using ASP.NET

- Applications that are built using ASP.NET can perform better by taking advantage of early binding.

- ASP.NET is an independent platform where you can use any language according to your comfort.



- Applications built using ASP.NET are used across the world.

- ASP.NET enables developers to work more efficiently.

Disadvantages of using ASP.NET

- Security is the chief advantage and the major disadvantage because you should take more care to protect the applications.
- Asp.Net does not provide 100% Data Access for the applications.



- Applications built using ASP.NET are more expensive because it requires SQL Server licenses, Visual Studio licenses, Windows Server licenses, etc.

- Changes may not work in ASP.NET Applications. If you want to make changes for the applications, then you have to take help from GitHub.

Conclusion

By now, you would have learned what ASP.NET is, and its importance and relevance in the present world. The hands-on demo would have helped you to understand more about ASP.NET.

If you are planning to learn .NET, consider enrolling in Simplilearn's Full Stack Developer - MERN StackThis comprehensive bootcamp is designed to help you master the fundamentals of .NET programming and how to create .NET projects. In the .NET programming certification course, you will be introduced to .NET space and coding with C#, including Visual Studio and Winforms, which will help you excel in

your career. Check out our ASP.Net MCV interview questions to prepare for the job interview.

If you have any queries on this “What is ASP.NET” tutorial, do let us know them in the comment section. We will have our experts answer it for you as soon as possible.

Happy learning!

ACTIVITY 1.1

1. What is the importance of ASP.NET in Programming?

2. What are the advantages of ASP.NET? Explain.

3. What are the three components of ASP.NET?

MODULE 1.3

All You Need to Know about an ASP File

A .asp file or an active server page file is an ASP.NET typed webpage or web document that contains HTML codes, text, graphics, and XML. A Microsoft IIS server processes all the scripts within the file and generates HTML code to display the page in a web browser.

ASP files were also known as classic ASP files because earlier, they used VB Script Language and had an extension .asp, but nowadays, they use C# as a programming/scripting language and have an extension as .aspx.

Some common scenarios where you might find a .asp or a .aspx file is when your browser sends you a .asp file by accident when you were trying to download something, or you might see a “.asp” at the end of a URL of a webpage which points to an ASP.NET webpage.

The Active Server Page files use source code compilation along with data encoding standards. The Microsoft IIS application uses these files and the Active Server Page engine to store content in the ASP file format.

How to Open Downloaded ASP Files?

If you got an asp file while trying to download another file, there are high chances that the file wasn't named correctly, and hence it served you the webpage that the file was on.

An example problem: Suppose you were trying to download a document such as your bank statement, but the downloaded file didn't open correctly, and when you checked the file name, it had a .asp or a .aspx extension. In this case, the most probable reason for this error is that the server added a .asp extension by mistake instead of .pdf, although the file was in the pdf format.

Solution #1

Just rename the file. Remove ".asp" and add a ".pdf" at the end of the file to view the file with any document viewer installed on your device. You are just doing something that the server forgot to do, adding "pdf" after the filename and the period.

Solution #2

Although it works most of the time, renaming isn't converting one file form to another. But sometimes, it fails to convert the file correctly, which corrupts the content of the file. In this case, use a third-party application or just search, for example, "asp to pdf converter" on any browser and use any online converter to convert the file from one form to another.

How to Open Other ASP Files?

In layman's tongue, ASP files are text files, i.e., they are acceptable to any text editors like Notepad + +, Brackets, Sublime Text, VS Code, etc. Some alternatives for ASP file editors include Visual Studio and Dreamweaver.

An asp file is almost the same as a web page. It just ends with .asp extension at the end.

ASP files need parsing before we open them in a web browser. A local .asp file won't open properly with a browser. It might appear as just plain text. To open a .asp file properly, you must run Microsoft IIS in the background and then open the file as localhost.

What Are the Applications Used to Open a .asp File?

Some of the most common applications to open .asp files are:

- Opera Web Browser or Any Web Browser

- Sublime Text or Any Text Editor
- Adobe Dreamweaver CS6
- Microsoft Visual Web Developer
- ES-Computing Edit Plus

How to Convert ASP Files?

Converting ASP files would mean that the file would not work the way it needs to because the server needs it to be in the proper format to display web pages correctly.

For example, if you were to convert an asp file into an HTML or a pdf, the file would open in a browser or a pdf viewer, but then it would also become obsolete if tried to use as an ASP file on the server.

But if you do need to convert an asp file, you can use Microsoft Visual Studio or Adobe Dreamweaver to convert ASP to formats like HTML, ASPX, VBS, ASMX, JS, SRF, etc.

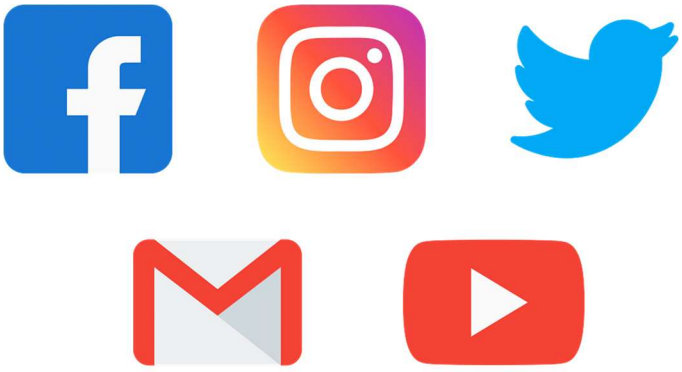
You can also convert it to .php using online converters.

ASP.NET Life Cycle: An Overview on Life Cycle of ASP.NET

ASP.NET is the latest version of Active Server Pages, which Microsoft developed to build websites. It is a web application framework released in 2002 and has an extension of .aspx.

Some of the major companies that use ASP.NET are:

- Facebook
- Instagram
- Twitter
- Email
- Youtube



Since you now have been shown a little about the basics of ASP.NET, it is time to investigate the life cycle of ASP.NET.

Life Cycle of ASP.NET

The ASP.NET life cycle is very crucial to developing applications. It includes various stages that help to produce dynamic output.

The Life Cycle of ASP.NET is divided into two categories:

1. Application Life Cycle

2. Page Life Cycle

Application Life Cycle

The application life cycle contains the following steps:

1. Application Start

The webserver executes the application start when a user requests an application for access. In this method, it sets all global variables to default.

2. Object Creation

Object creation holds all the HTTP Context, HTTP Request, and HTTP Response by the webserver. It also contains information about the request, cookies, and browsing information.

3. HTTP Application

HTTP Application is an object created by the webserver. It helps to process all the subsequent information that is sent to the user.

4. Dispose

Dispose is an event that is called before the application is destroyed. It also helps to release manually unmanaged resources when the objects are no longer needed.

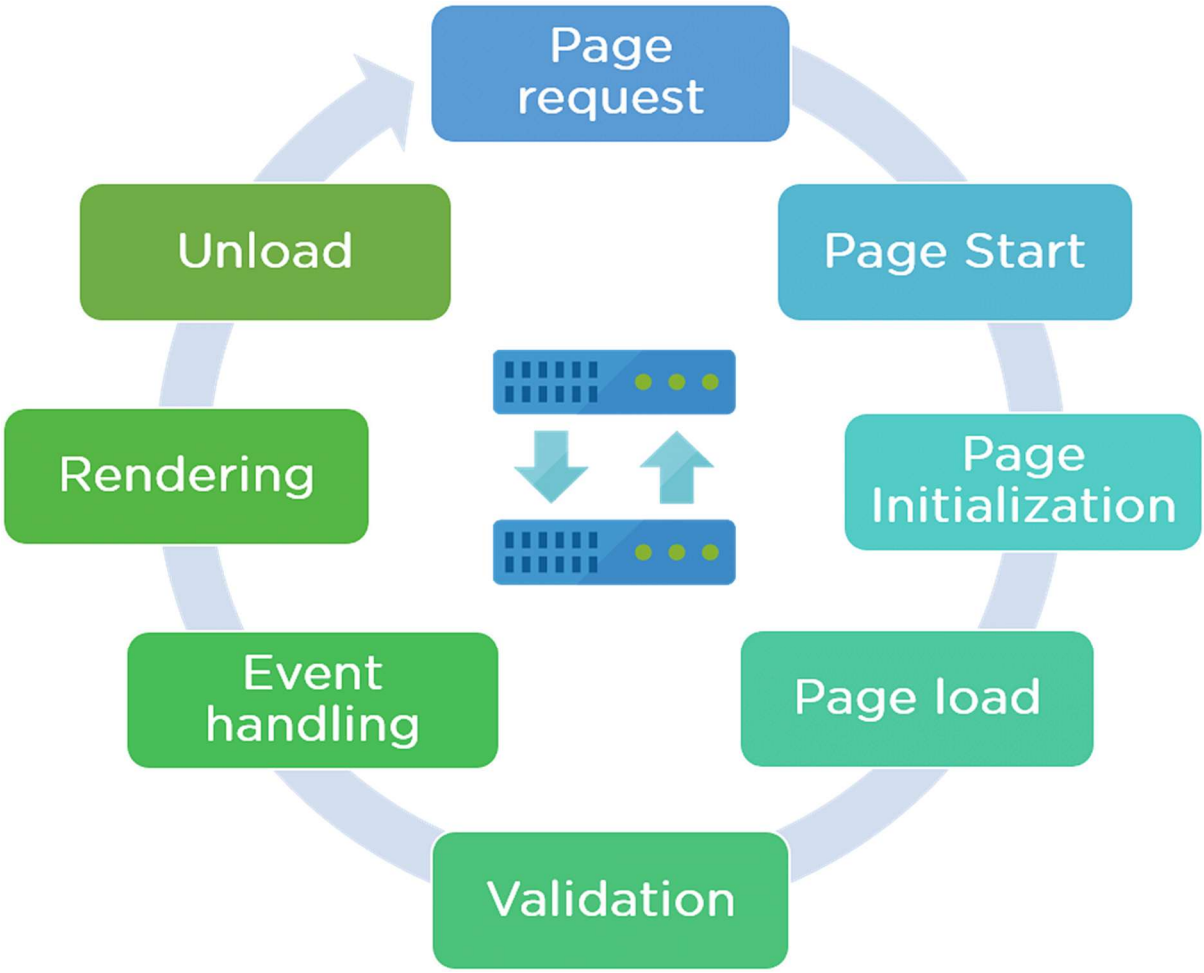
5. Application End

Application End is the last stage of the application life cycle. It helps to unload the memory.

Page Life Cycle

The Page Life Cycle has certain phases that help in writing custom controls and initializing an application.

Following are the different phases of the Page Life Cycle:



1. Page Request

Page Request is the first step of the page life cycle. When a user request is made, the server checks the request and compiles the pages. Once the compilation is done, the request is sent back to the user.

2. Page Start

Page Start helps in creating two objects: request and response. The request holds all the information which the user sent, while the response contains all the information that is sent back to the user.

3. Page Initialization

Page Initialization helps to set all the controls on the pages. It has a separate ID, and it applies themes to the pages in this step.

4. Page Load

Page Load helps to load all the control properties of an application. It also helps to provide information using view state and control state.

5. Validation

Validation happens when the execution of an application is successful. It returns two conditions: true and false. If execution is successful, it returns true, otherwise false.

6. Event Handling

Event Handling takes place when the same pages are loaded. It is a response for the validation. When the same page is loaded, a Postback event is called, which checks the page's credentials.

7. Rendering

Rendering happens before it sends all the information back to the user. And all this information is stored before being sent back.

8. Unload

Unload is a process that helps in deleting all unwanted information from the memory once the output is sent to the user.

Page Life Cycle Events

Here's a collection of typical page life cycle events you may encounter.

- PreInit: This event occurs when the start stage is done but before the initialization stage.

- Init: This event reads and initializes control properties and happens after all the controls are initialized.
- InitComplete: This happens at the conclusion of the page's initialization stage. The event makes changes to the view state that we want to ensure are persisted at the end of the subsequent postback.
- PreLoad: PreLoad occurs at the conclusion of the event-handling stage, and is used for tasks that need all other page controls to be loaded.
- Load: Load is raised initially for the page, then recursively for the child controls.
- Control events: This takes care of handling specific control events like button control clicks.
- LoadComplete: This event is used for many tasks that need other controls found on the page to be loaded. LoadComplete occurs at the conclusion of the event-handling stage.
- PreRender: This event kicks in after the page object has created the controls necessary to render the page.
- PreRenderComplete: This event happens once the pre-render stage is complete.
- SaveStateComplete: This event is raised when the view and control states are saved for both the page and all the controls.
- Render: Render isn't actually an event; rather, it's a method that the Page object calls for each control.
- Unload: This event is first raised for each control, then for the page itself.

Catch-Up Events for Added Controls

Should the controls be created dynamically at run time or created declaratively within the data-bound control templates, their respective events aren't initially synchronized with the rest of the page's controls. So, dynamically added controls and controls in templates must raise their events sequentially until all of them have caught up to the event present during its addition to the controls collection.

Data Binding Events for Data-Bound Controls

Here's a chart that shows data-related events within the context of data-bound controls.

Data-Related Control Event	Usual Use

DataBinding	Marks the start of the process which binds the data with the control. This event is typically used to dynamically set parameter values or manually open the database connections.
RowCreated (for GridView only) or ItemCreated (for DataList, DetailsView, SiteMapPath, DataGrid, FormView, Repeater, and ListView controls)	This event gets raised after the control's DataBinding event and used to manipulate any content that isn't dependent on data binding.
RowDataBound (for GridView only) or ItemDataBound (for DataList, SiteMapPath, DataGrid, Repeater, and ListView controls)	This event is raised either after the control's ItemCreated or RowCreated events. It means the data is now available in the row or the item, so it can be formatted or have the FilterExpression property set on a child data source.
DataBound	You use this event to format any data-bound content or start data binding in other controls that rely on values derived from the current control's content.

ACTIVITY 1.2

Groupings. 5 members per group
Create a 3-5minutes video clip explaining the life cycle, flows, uses, and advantage nor disadvantages of ASP.NET. Use your creativity when making a video clip.

Activity 1.3

1. It helps in creating two objects: request and response. The request holds all the information which the user sent, while the response contains all the information that is sent back to the user.
2. This event is raised when the view and control states are saved for both the page and all the controls.
3. This event is raised either after the control's ItemCreated or RowCreated events. It means the data is now available in the row or the item, so it can be formatted or have the FilterExpression property set on a child data source
4. It happens when the execution of an application is successful. It returns two conditions: true and false. If execution is successful, it returns true, otherwise false.
5. An event that is called before the application is destroyed. It also helps to release manually unmanaged resources when the objects are no longer needed.

An Ultimate One-Stop Solution Guide to C# Web Services With Examples

Google Cloud Platform (GCP), Azure Web app services, and Amazon Web Services (AWS) come to mind whenever the word "web services" is mentioned these days. But there's a strong rationale behind it. In the face of this demand, these IT titans have set a new standard for the industry. Scalability from Tech Giants like Google and Amazon is really what renders current web services feasible.

The term "Web" in the name of a Web Service is an unfair assessment. However, Web Services do not use the World Wide Web (WWW), an Internet user interface, but a machine-to-machine service that uses the WWW standards to function on the Net. Web technologies like HTTP transport machine-readable forms like XML and JSON in a web service.

In this "C# Web Services" tutorial, you will learn the crucial technical aspects of the web services pattern and fundamentals involving SQL Server.

What Are C# Web Services?

A web service is a type of online application consisting of a collection of procedures other services may invoke. Although it lacks a user interface, it follows a script design similar to ASP.NET web pages. A web service is an internet feature retrieved through internet interfaces and used by online applications.

The .NET Framework isn't the only Framework that can use Web Services. Before .NET came out, the ideas were already in place, and companies other than Microsoft supported them. Web Services are cross-platform, which means that an application developed in one language may call a service built in another. The sole condition for using a service has internet access to send an HTTP request.

Because a web service is cross-platform, there should be a common language for requesting services and receiving responses from them. XML is an example of a universal common language. As a result, Web Services are based on XML-based data exchange protocols.

We can now go on to the next topic to learn about various technologies that support Web services now that we have a basic understanding of Web Services.

Technologies That Support Web Services

There are various technologies available that support Web Services. There are four prominent technologies:

- XML

An XML Web service is nothing more than a piece of programming code made available to other programs through the Internet. An XML Web service can be as basic as a piece of code or an object that publishes data over the Internet using HTTP or another standard protocol.

- SOAP

Soap is an acronym for "Simple Object Access Protocol." SOAP is an XML based protocol. SOAP is used on any system and in any programming language. You can communicate with different programming languages by utilizing SOAP.

- WSDL

Web Services Description Language is the abbreviation for WSDL. WSDL is the standard format for describing a web service and is widely used. In the development of WSDL, both Microsoft and IBM worked together. Sharing data in decentralized and distributed systems is made possible via the WSDL, an XML-based protocol.

- UDDI

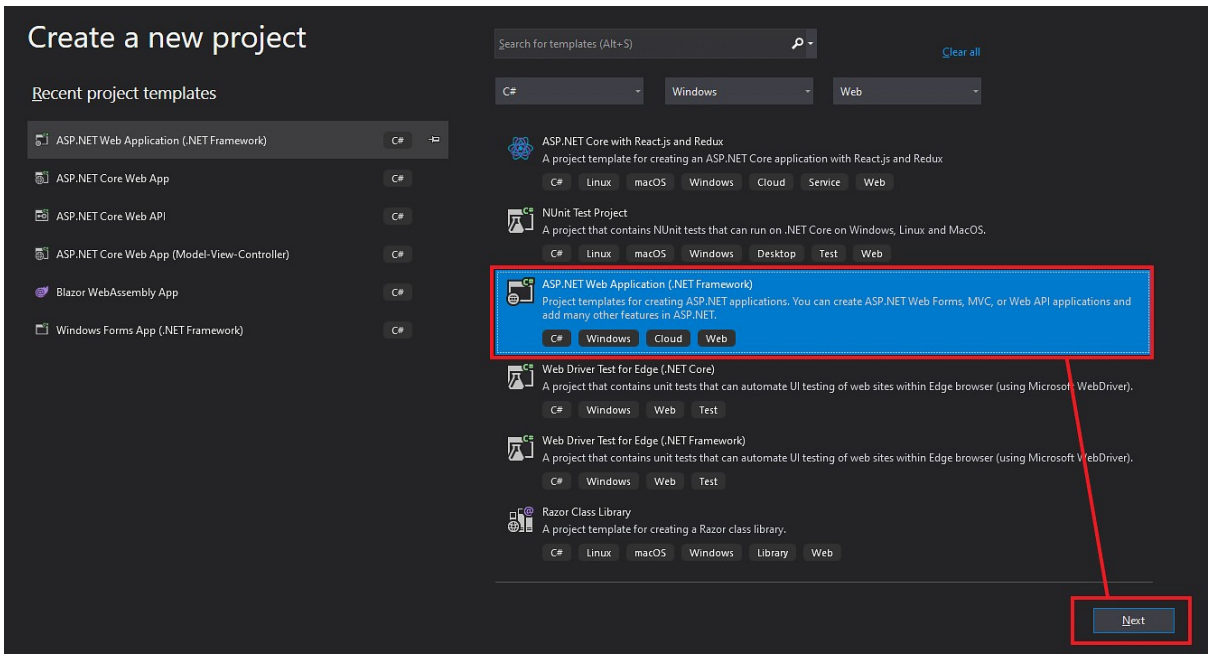
The "Universal Description, Discovery, and Integration (UDDI)" specification establishes a standard for publishing and discovering web service information. UDDI registries use the UDDI standard to provide web service directories. Web services are also linked to Technical models in the UDDI definition. A UDDI registry user can search for a kind of service using these patterns, or general groups, instead of knowing the login information for a particular service.

Now that we have a good understanding of the technologies that support Web Services. Let's try to implement Web Services in the visual studio.

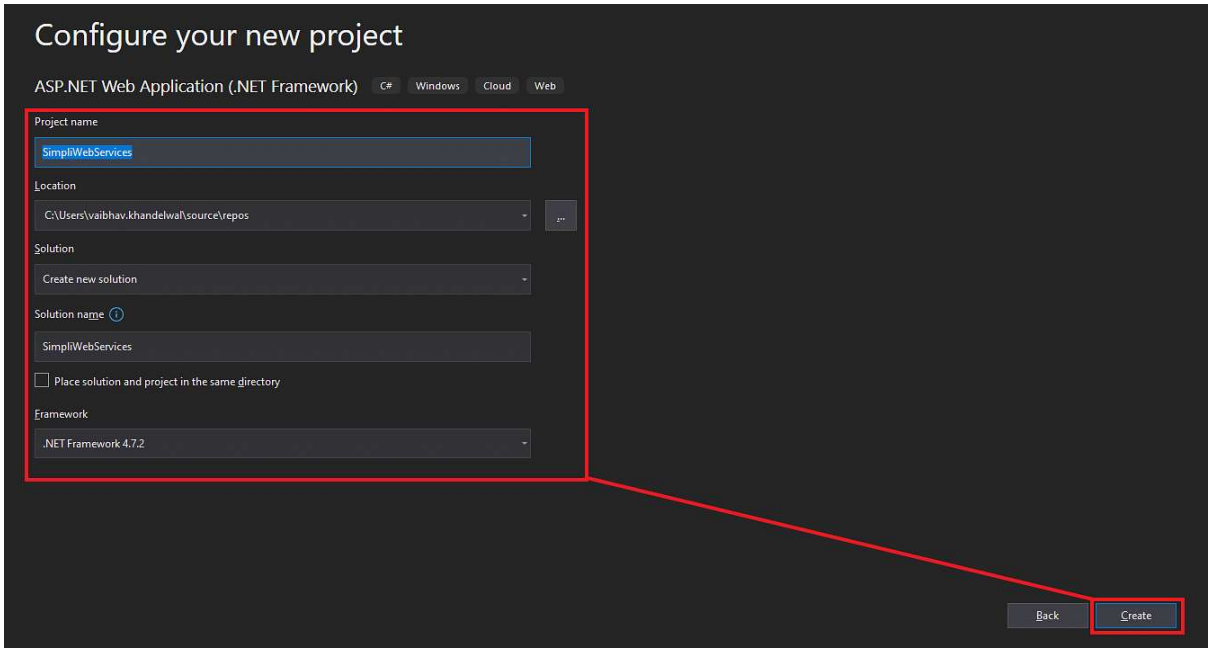
Implementation of Web Service

ACTIVITY 1.4

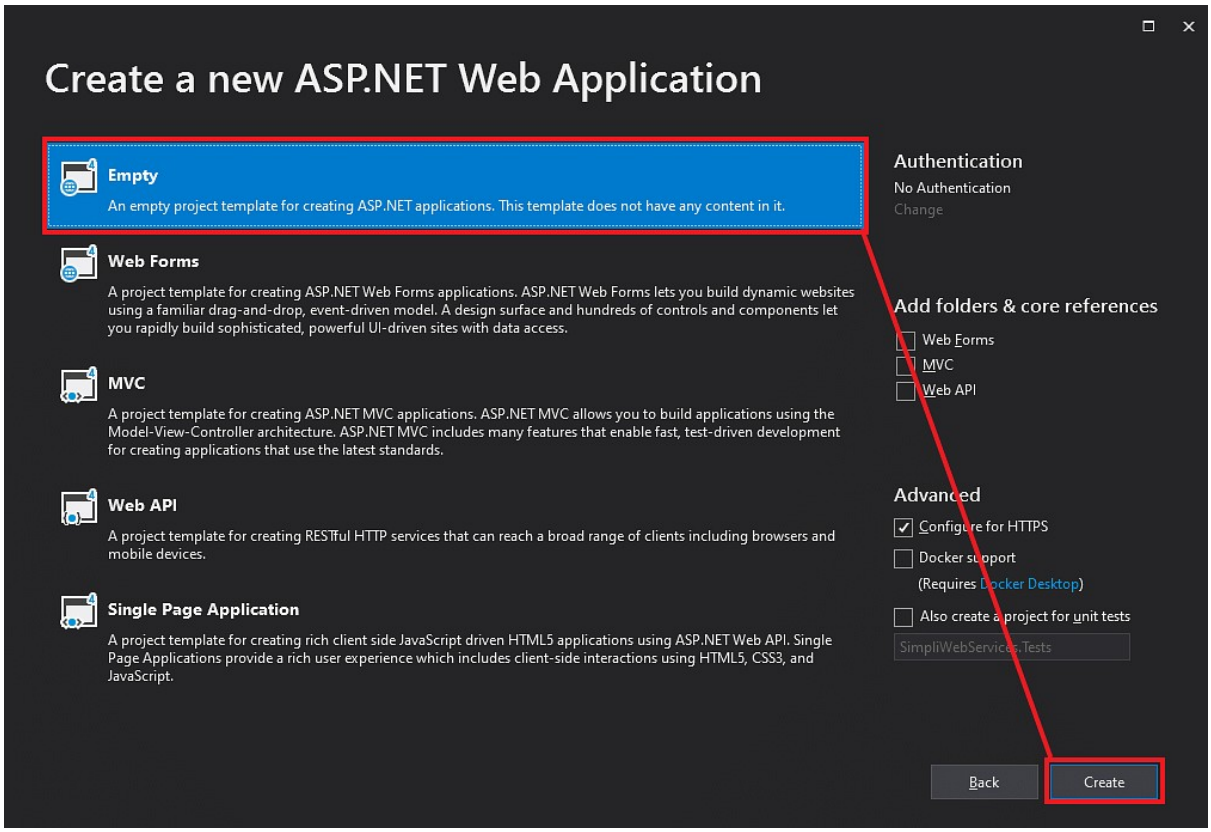
Let's start with creating a C# web services Project.



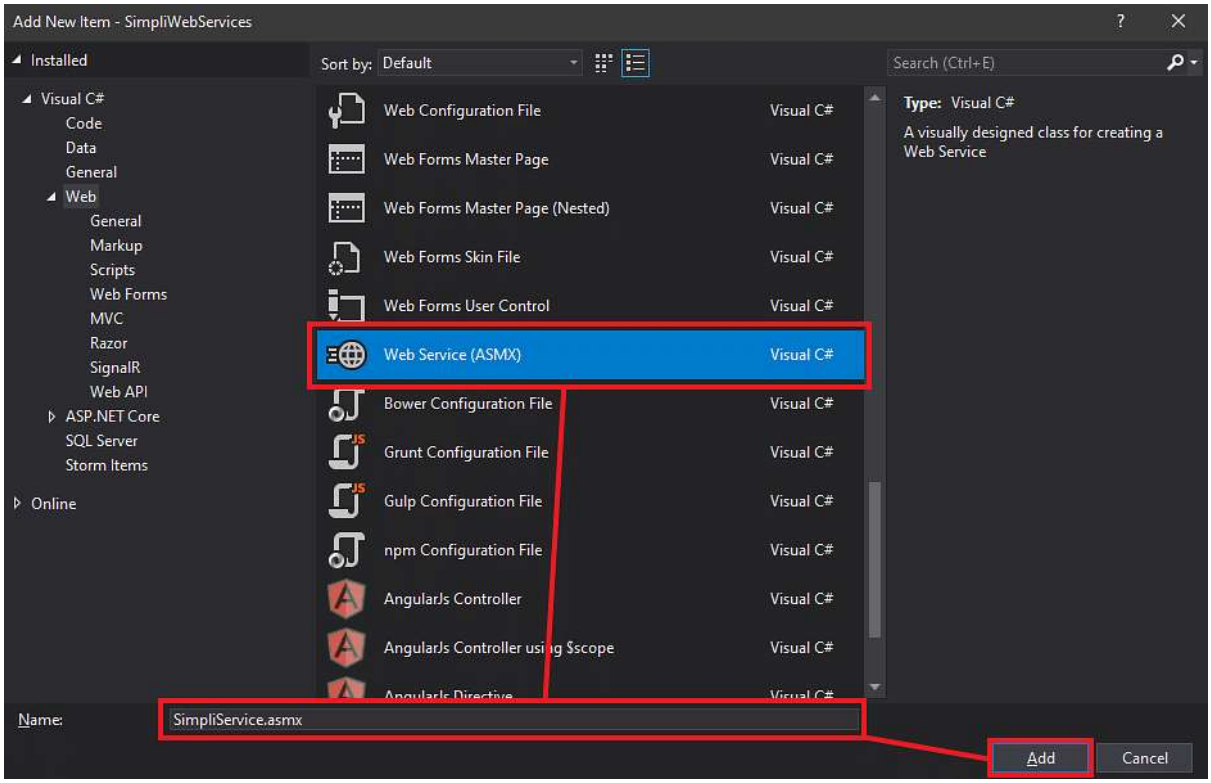
Let's open visual studio 2019; let's create a new project. It will be an asp.net web application (.net framework). We will click next.



Let's name it "SimpliWebServices" and hit create.



Next, we will be offered the template for our project, and we will choose Empty and hit create.



Now we will right-click on the project name, select add, and choose a new item. We create a web service (ASMX) file. We will name it SimpliService

We will uncomment the given below line.

[System.Web.Script.Services.ScriptService]

Now let's make a few functions.

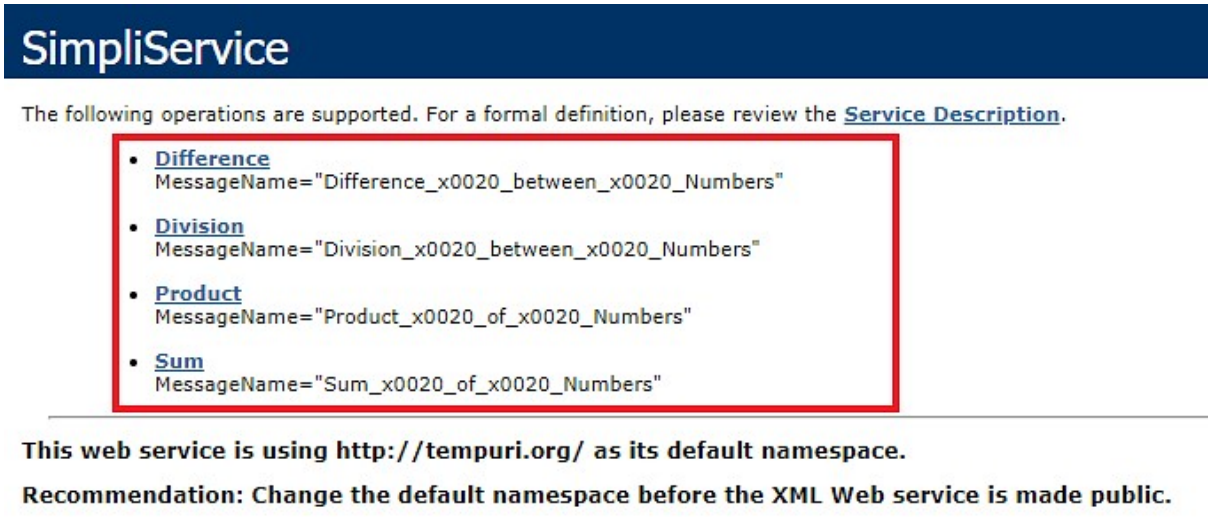
Code:

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Services;

namespace SimplilearnWebServices

{

    /// <summary>

    /// Summary description for SimplilearnService

    /// </summary>
```

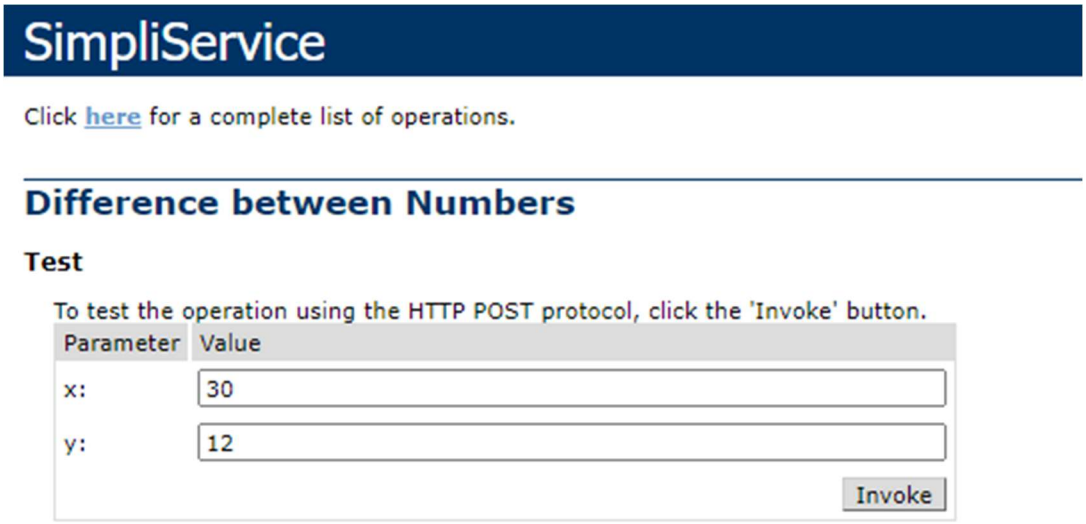
```
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[System.ComponentModel.ToolboxItem(false)]
// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
[System.Web.Script.Services.ScriptService]
public class SimplilearnService : System.Web.Services.WebService
{
    [WebMethod(MessageName ="Sum of Numbers")]
    public int Sum(int x, int y)
    {
        return x + y;
    }
    [WebMethod(MessageName = "Difference between Numbers")]
    public int Difference(int x, int y)
    {
        return x - y;
    }
    [WebMethod(MessageName = "Product of Numbers")]
    public int Product(int x, int y)
    {
        return x * y;
    }
    [WebMethod(MessageName = "Division between Numbers")]
    public int Division(int x, int y)
   {
        return x / y;
    }
```

```
    }
}
```

Now let's save and run it.

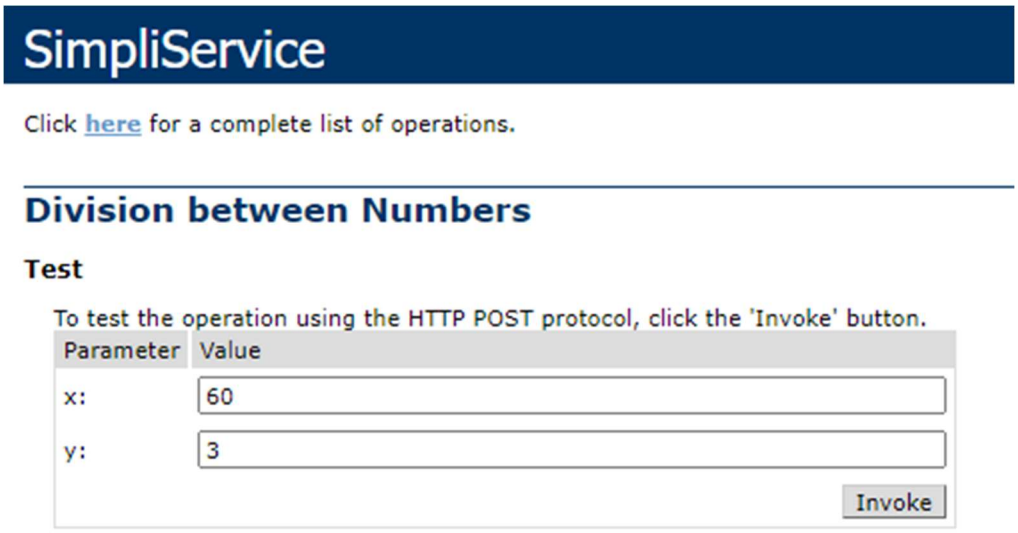


Here we have all four functions. Let's try them one by one



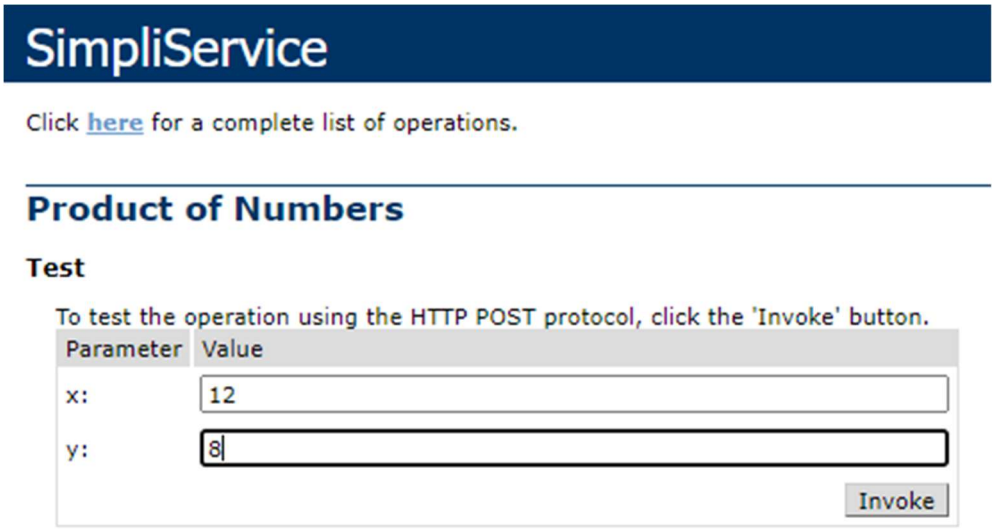
First, let's try the Difference function. Let's give the values like 30 and 12.

Here is the output which comes out to be 18.



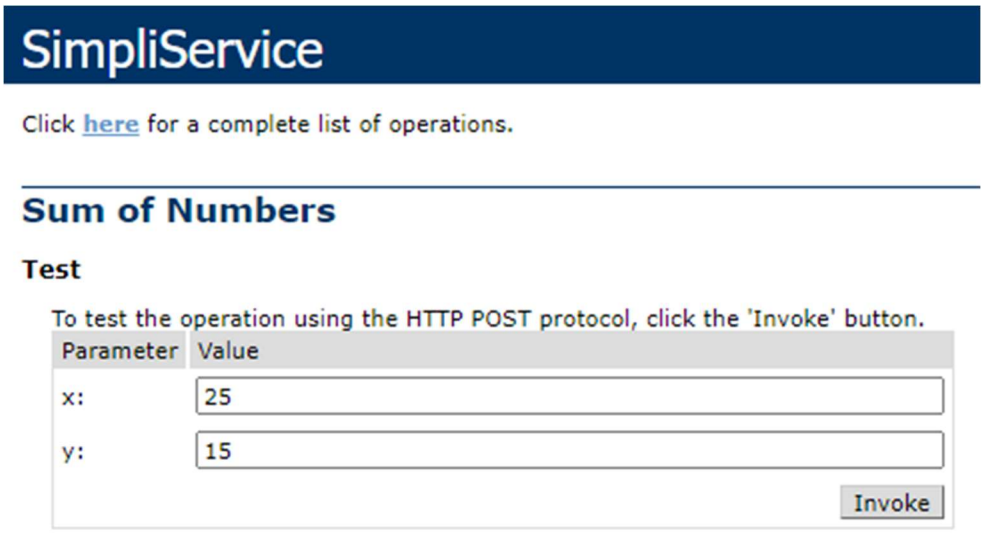
Next, let's try the Division function. Let's give the values like 60 and 3.

Here is the output which comes out to be 20.



Then, let's try the Product function. Let's give the values like 12 and 8.

Here is the output which comes out to be 96.



At last, let's try the Sum function. Let's give the values like 25 and 15.

Here is the output which comes out to be 40.

Now that we have successfully implemented the Web services. Let's glance over a few benefits of Web services.

Advantages of Web Services

There are many Advantages of C# Web Services.

- Web Services are simple to install but expensive to maintain since they use existing infrastructure. Most programs may be repurposed as Web Services.
- Web Services employ a text-based interface. Web services employ all elements, even if they're built-in in many languages and platforms.
- The cost of business application integration and B2B communications is reduced using Web Services.
- Web services encourage a modular development style, allowing many businesses to interface with the same web service.
- Web Services is an interoperable organization that brings together over a hundred manufacturers to promote interoperability.

These are the advantages of working with Web Services. Now let's have a look at the Disadvantages of Web services.

Disadvantages of Web Services

There are various Disadvantages of Web Services.

- One of the Web Services' constraints is that SOAP, WSDL, and UDDI need maintenance.
- The utilization of a web service enhances the platform's demand.
- We use a consistent process to assess the performance of certain online services.
- The web service's constraint is that the distributor retains the copyrights of the particular standard.
- The Web Service's constraint also contributes to interoperability.
- If we wish to use web services in a high-performance setting, web services will be sluggish in such circumstances.

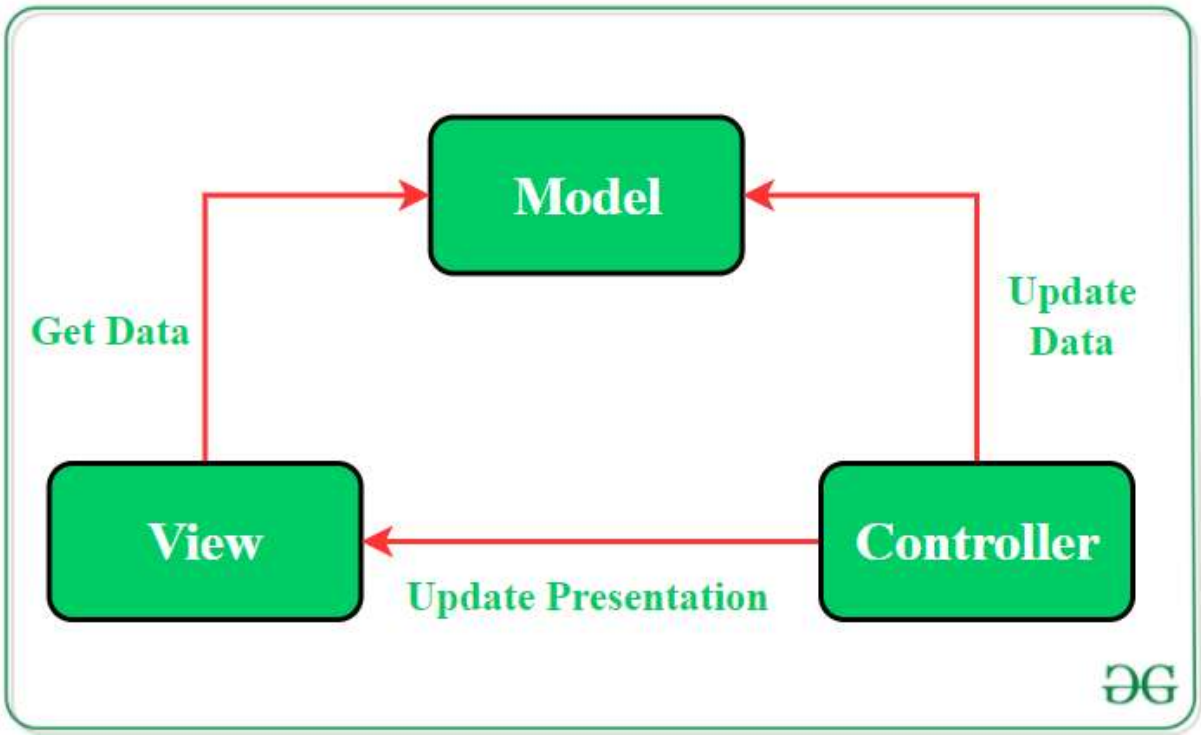
By now, you have a good understanding of the technical and theoretical side of Web Services.

MODULE 2

MVC (Model View Controller) Architecture Pattern

The **MVC pattern** suggests splitting the code into 3 components. While creating the class/file of the application, the developer must categorize it into one of the following three layers:

- **Model:** This component stores the application data. It has no knowledge about the interface. The model is responsible for handling the domain logic(real-world business rules) and communication with the database and network layers.
- **View:** It is the UI(User Interface) layer that holds components that are visible on the screen. Moreover, it provides the visualization of the data stored in the Model and offers interaction to the user.
- **Controller:** This component establishes the relationship between the **View** and the **Model.** It contains the core application logic and gets informed of the user's behavior and updates the Model as per the need.

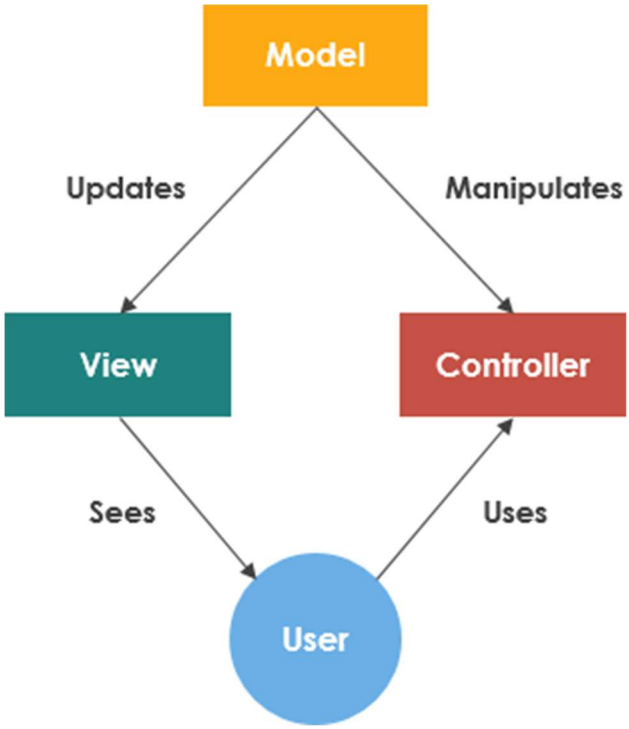


What is Model-View and Control?

MVC (Model-View-Controller) is an architectural design pattern that encourages improved application organization through a separation of concerns. It divides an interactive application into three components: Model / View and Controller. It enforces the isolation of

business data (Models) from user interfaces (Views), with a third component (Controllers) traditionally managing logic, user-input and coordinating both the models and views. The goal of MVC is to help structure the separate the concerns of an application into three parts:

- **Model** is responsible for managing the data of the application. It receives user input from the controller.
- **View** means the presentation of the model in a particular format.
- **Controller** responds to the user input and performs interactions on the data model objects. The controller receives the input, optionally validates it and then passes the input to the model.



The Purpose of MVC Framework

As mentioned above, MVC software framework helps us to separate the different aspects of the application (input logic, business logic, and GUI), while providing a loose coupling between these elements. Thus, the information (most reusable) logic belongs in the model, the GUI belongs in the view. Input logic belongs in the controller. This separation helps you manage complexity when you build an application because it enables you to focus on one aspect of the implementation at a time. MVC Framework is a good idea for a number of reasons, including:

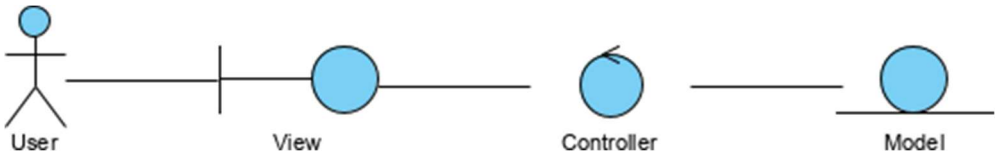
- Simultaneous development – Because MVC decouples the various components of an application, developers are able to work in parallel on different components without affecting or blocking one another.

- Reusability – The same (or similar) view for one application can be refactored for another application with different data because the view is simply handling how the data is being displayed to the user.
- improved scalability – if your application begins experiencing performance issues because database access is slow, you can upgrade the hardware running the database without other components being affected
- Low coupling — the very nature of the MVC framework is such that there is low coupling among models, views or controllers.
- better extendibility – as the components have a low dependency on each other, making changes to one (to fix bugs or change functionality) does not affect another

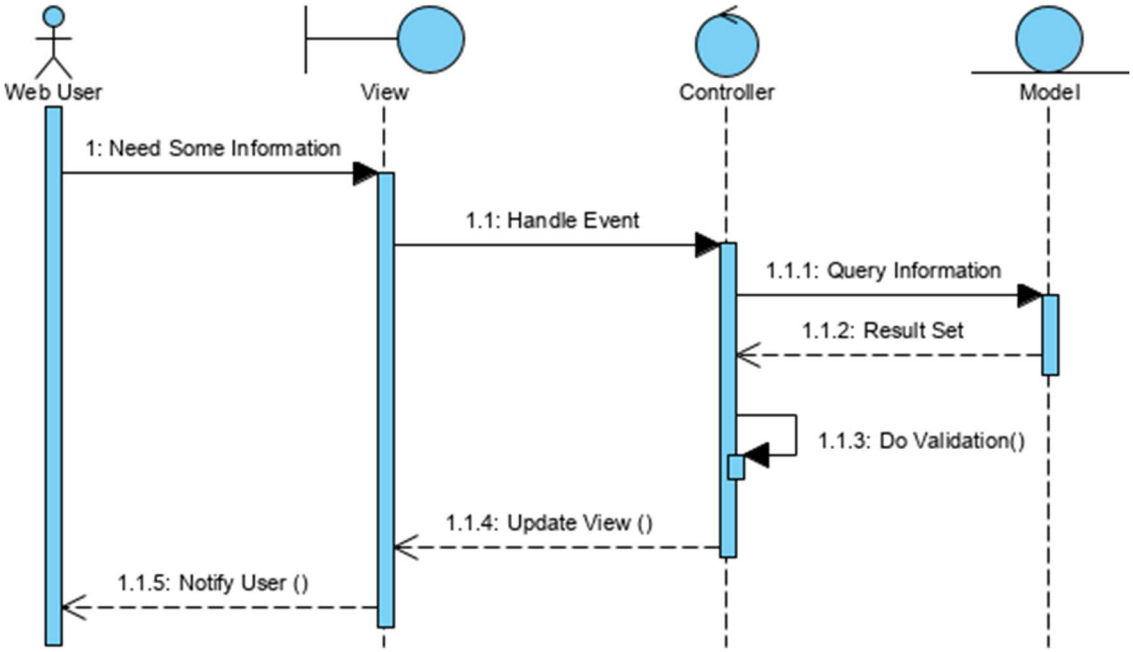
Develop Use Case Scenario Using MVC Sequence Diagrams

You can use stereotypes for the lifeline in the MVC sequence diagram to make visually clear what type of objects you are using in the MVC. An MVC Sequence diagram has interface objects, controller objects and entity objects:

- Entities are objects representing system data: Customer, Product, Transaction, Cart, etc.
- Boundaries are objects that interface with system actors: UserInterface, DataBaseGateway, ServerProxy, etc.
- Controls are objects that mediate between boundaries and entities. A controller object often correspond to use cases scenario and often represented by a sequence diagram.



And here is the simplistic and hypothetical sequence diagram for MVC. What you see in this diagram, a web-user initiated a query and an event is generated that is handled by the controller and gets information that is needed from the model, validates the information and passes back the result set to the view.

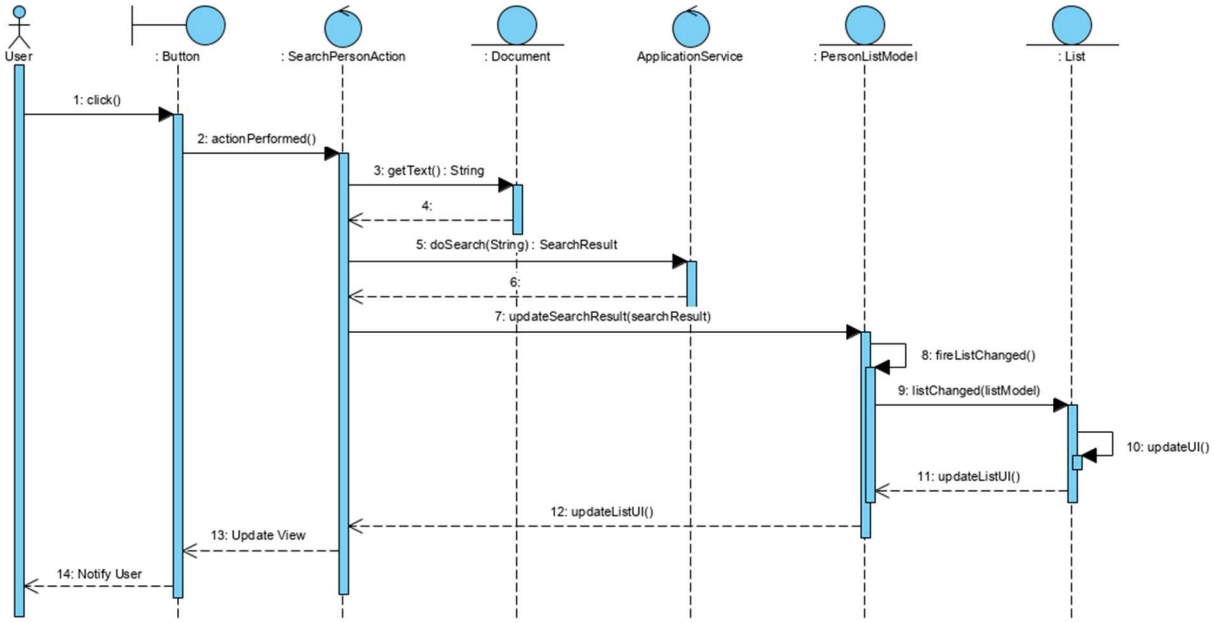


Example MVC Sequence Diagram

Suppose an application that let you search for persons. The UI must have a text field where the user can enter a search string and it might have a button to start the search. Finally it must have an area where the search results are displayed. In our case, it is implemented with a list component.

The “Search for Persons” use case Scenario is:

- The user enters a search string in the text field
- The user clicks the search button.
- The search result is displayed in the result list.



The sequence diagram above shows how the user's button click moves through the application until the result gets finally displayed in the list component.

Communication between the Components

This below communication flow ensures that each component is responsible for a specific aspect of the application's functionality, leading to a more maintainable and scalable architecture

- **User Interaction with View:** The user interacts with the View, such as clicking a button or entering text into a form.
- **View Receives User Input:** The View receives the user input and forwards it to the Controller.
- **Controller Processes User Input:** The Controller receives the user input from the View. It interprets the input, performs any necessary operations (such as updating the Model), and decides how to respond.
- **Controller Updates Model:** The Controller updates the Model based on the user input or application logic.
- **Model Notifies View of Changes:** If the Model changes, it notifies the View.
- **View Requests Data from Model:** The View requests data from the Model to update its display.
- **Controller Updates View:** The Controller updates the View based on the changes in the Model or in response to user input.
- **View Renders Updated UI:** The View renders the updated UI based on the changes made by the Controller.

ACTIVITY 1.5
SCENARIO. YELLOWPAD

1. If you are an Web Developer what will be your changes or features that you will contribute in view of the topic MVC ARCHITECTURE?

2. In 100 words what are your understanding about MVC Architecture?

ACTIVITY 1.6

Create a sample student directory program that indicate the Model, View and Control.

1. **Model (Student class)**
   Represents the data (student's name and roll number) and provides methods to access and modify this data.
2. **View (StudentView class)**
   Represents how the data (student details) should be displayed to the user. Contains a method (**printStudentDetails**) to print the student's name and roll number.
3. **Controller (StudentController class)**
   Acts as an intermediary between the Model and the View. Contains references to the Model and View objects. Provides methods to update the Model (e.g., **setStudentName**, **setStudentRollNo**) and to update the View (**updateView**).

EXAMPLE OUTPUT:

```
Student:
Name: Lokesh Sharma
Roll No: 15UCS157
Student:
Name: Vikram Sharma
Roll No: 15UCS157
```