Introduction to WEB Dev.

- There are two broad divisions of web development – front-end development (also called client-side development) and back-end development (also called server-side development)
- Front-end development refers to constructing what a user sees when they load a web application – the content, design and how you interact with it. This is done with three codes – HTML, CSS and JavaScript.
- HTMLis a special code for 'marking up' text in order to turn it into a web page. Every web page on the net is written in HTML, and it will form the backbone of any web application. CSS is a code for setting style rules for the appearance of web pages. CSS handles the cosmetic side of the web. Finally, JavaScript is a scripting language that's widely used to add functionality and interactivity to web pages.
- Back-end development controls what goes on behind the scenes of a web application. A back-end often uses a database to generate the front-end.
- Back-end scripts are written in many different coding languages and frameworks, such as

- PHP
- Ruby on Rails
- ASP.NET
- Perl
- Java
- Node.js
- Python

- **Web Forms**
- With ASP.NET Web Forms, you can build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

- **MVC**
- ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and that gives you full control over markup for enjoyable, agile development. ASP.NET MVC includes many features that enable fast, TDD-friendly development for creating sophisticated applications that use the latest web standards.

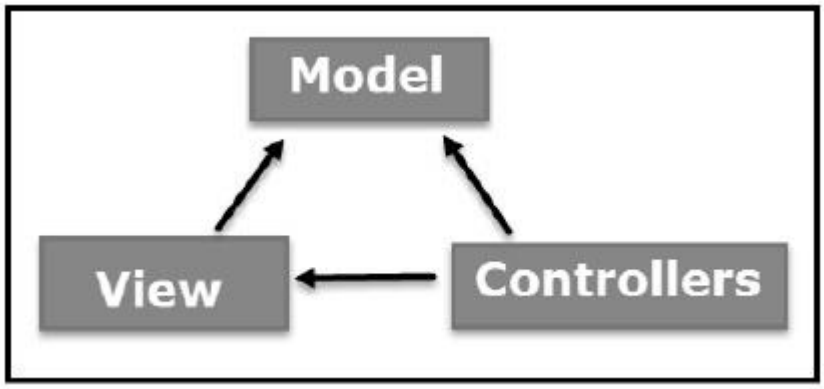
- **ASP.NET Web Pages**
- ASP.NET Web Pages and the Razor syntax provide a fast, approachable, and lightweight way to combine server code with HTML to create dynamic web content. Connect to databases, add video, link to social networking sites, and include many more features that help you create beautiful sites that conform to the latest web standards.

- **Web APIs**
- ASP.NET Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices. ASP.NET Web API is an ideal platform for building RESTful applications on the .NET Framework.

MVC

- The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application.
- MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

- MVC Components :



- Model : The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.
- View: The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.
- Controller : Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

Rest Web API

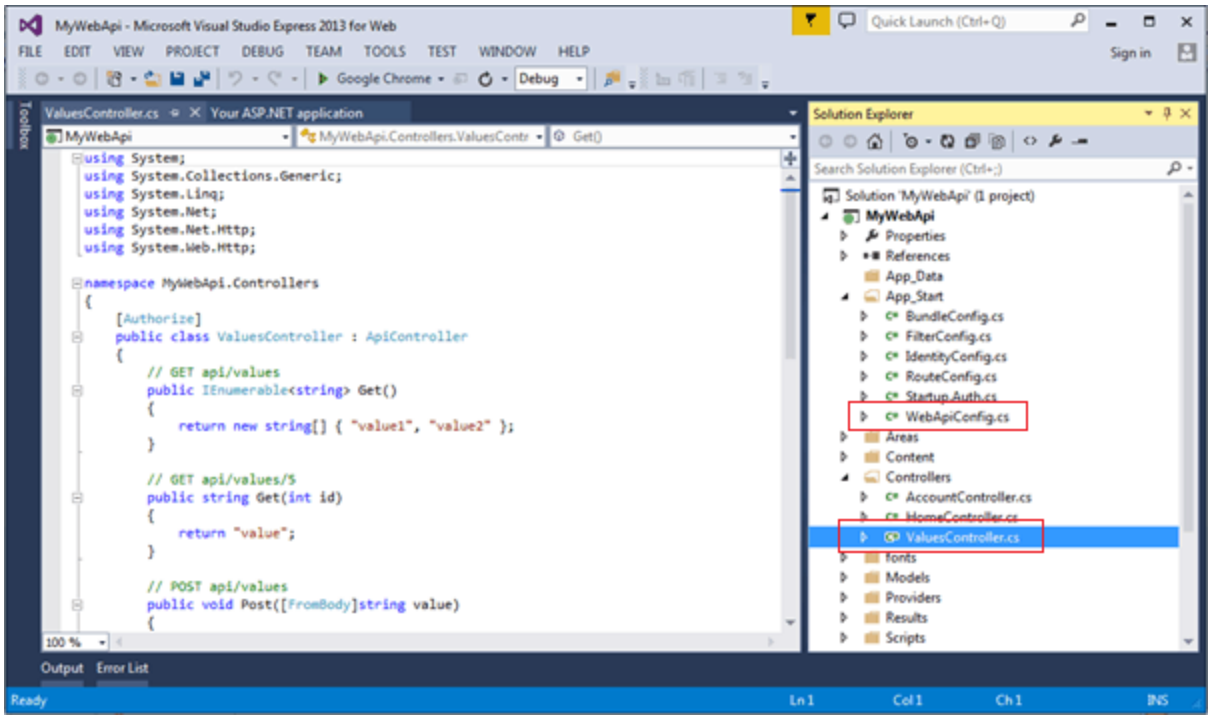
- ASP.NET Web API is a framework for building HTTP services that can be accessed from any client including browsers and mobile devices. It is an ideal platform for building RESTful applications on the .NET Framework.
- In computer programming, an application programming interface (API) is a set of subroutine definitions, protocols, and tools for building software and applications.

- To put it in simple terms, API is some kind of interface which has a set of functions that allow programmers to access specific features or data of an application, operating system or other services.
- 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
- ASP.NET Web API is an ideal platform for building RESTful services.
- ASP.NET Web API is built on top of ASP.NET and supports ASP.NET request/response pipeline
- ASP.NET Web API maps HTTP verbs to method names.
- ASP.NET Web API supports different formats of response data. Built-in support for JSON, XML, BSON format.
- ASP.NET Web API can be hosted in IIS, Self-hosted or other web server that supports .NET 4.0+.
- ASP.NET Web API framework includes new HttpClient to communicate with Web API server. HttpClient can be used in ASP.MVC server side, Windows Form application, Console application or other apps.

Start With Project

Create New Web API Project

- Web APIs are simple Non-SOAP-based HTTP services. We have client-side and server-side, when it comes to server-side, data stored on a database and WEB API acts as an intermediary that fetches data or inserts data to the DB. On the client-side, the actions take place on the user computer. To communicate between client-side and server-side we need REST Apis. ASP.Net Web API is a framework for building HTTP services. We can do different operations using APIs like update invoice details, Insert new user, delete customer order, etc.
- Go to File => New Porject => Select ASP.NET Core Web Application.
- Add Project Name and Location. Click on Create.
- From the dropdown, select .NET Core and select API. Click on Create.
- This project is same as default MVC project with two specific files for Web API, WebApiConfig.cs in App_Start folder and ValuesController.cs in Controllers folder as shown below.



- The WebApiConfig.cs is configuration file for Web API. You can configure routes and other things for web API, same like RouteConfig.cs is used to configure MVC routes. It also creates Web API controller ValuesController.cs by default.

Create Controller, Model

- Web API Controller is similar to ASP.NET MVC controller. It handles incoming HTTP requests and send response back to the caller.
- Web API controller is a class which can be created under the Controllers folder or any other folder under your project's root folder. The name of a controller class must end with "Controller" and it must be derived from System.Web.Http.ApiController class. All the public methods of the controller are called action methods.
- ValuesController class is derived from ApiController and includes multiple action methods whose names match with HTTP verbs like Get, Post, Put and Delete.

- Based on the incoming request URL and HTTP verb (GET/POST/PUT/PATCH/DELETE), Web API decides which Web API controller and action method to execute e.g. Get() method will handle HTTP GET request, Post() method will handle HTTP POST request, Put() mehtod will handle HTTP PUT request and Delete() method will handle HTTP DELETE request for the above Web API.
- If you want to write methods that do not start with an HTTP verb then you can apply the appropriate http verb attribute on the method such as HttpGet, HttpPost, HttpPut etc. same as MVC controller.

- **Web API Controller Characteristics**

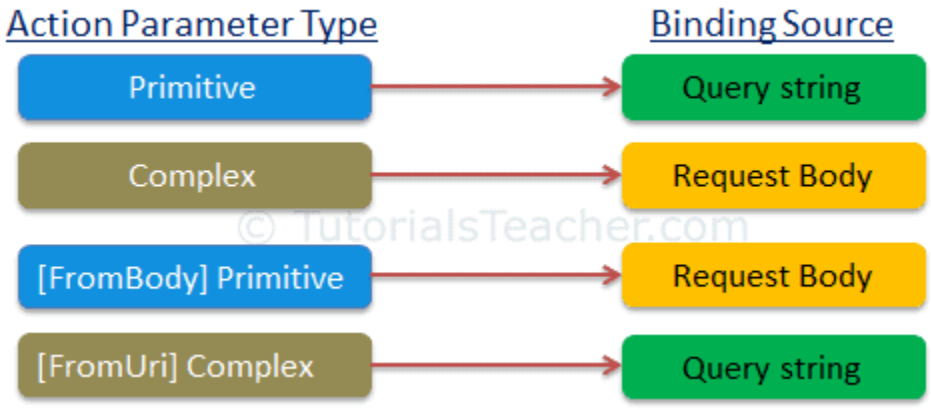
- It must be derived from System.Web.Http.ApiController class.
- It can be created under any folder in the project's root folder. However, it is recommended to create controller classes in the Controllers folder as per the convention.
- Action method name can be the same as HTTP verb name or it can start with HTTP verb with any suffix (case in-sensitive) or you can apply Http verb attributes to method.
- Return type of an action method can be any primitive or complex type.
- **Add Model**
- To add the model right-click on the model folder and add a class with the name
- After that add properties to the class.

Parameters (From URI, From Body)

- Action methods in Web API controllers can have one or more parameters of different types. It can be either primitive type or complex type. Web API binds action method parameters with the URL's query string or with the request body depending on the parameter type.
- Consider the following example of the GET action method that includes a single primitive type parameter.
- HTTP GET action method includes the id parameter of the int type. So, Web API will try to extract the value of id from the query string of the requested URL, convert it into int and assign it to the id parameter of the GET action

method. For example, if an HTTP request is http://localhost/api/student?id=1 then the value of the id parameter will be 1.

- Query string parameter name and action method parameter name must be the same (case-insensitive). If names do not match, then the values of the parameters will not be set. The order of the parameters can be different.
- Post action method cannot include multiple complex type parameters because, at most, one parameter is allowed to be read from the request body.
- You have seen that by default, ASP.NET Web API gets the value of a primitive parameter from the query string and a complex type parameter from the request body. But, what if we want to change this default behavior
- Use [FromUri] attribute to force Web API to get the value of complex type from the query string and [FromBody] attribute to get the value of primitive type from the request body, opposite to the default rules.
- The name of the complex type's properties and the query string parameters must match.
- The [FromBody] attribute can be applied on only one primitive parameter of an action method. It cannot be applied to multiple primitive parameters of the same action method.



Serialization

- Serialization is converting object to stream of byte
- We can serialize to any encoding format. Xml & Json
- In output. You will find that the property names have automatically converted into their camel case equivalent. For example, BookID became bookID .This means by default Web API serializes JSON data using camel casing.
- Open the Startup class and modify the ConfigureServices() method as shown below:

```
services.AddControllers()
.AddJsonOptions(options =>
{
options.JsonSerializerOptions .PropertyNamingPolicy = null;
});
```

- If you want to explicitly set the casing to camel casing then you can write :

```
options.JsonSerializerOptions.
ropertyNamingPolicy = JsonNamingPolicy.CamelCase;
```

- In ASP.NET Core MVC you can use Json() method to serialize data JSON format.
- In order to instruct MVC to stop using camel casing you can write first code in the ConfigureServices() or you can write You can set PropertyNamingPolicy property to JsonNamingPolicy.CamelCase to explicitly set the property casing to camel casing.
- You can also change this behavior for individual Json() calls as shown below:

```
var options = new JsonSerializerOptions(){
```

```
PropertyNamingPolicy = JsonNamingPolicy.CamelCase

};

return Json(empList, options);
```

Routing

- It routes an incoming HTTP request to a particular action method on a Web API controller.
- Web API supports two types of routing:
- **Convention-based Routing**
- In the convention-based routing, Web API uses route templates to determine which controller and action method to execute. At least one route template must be added into route table in order to handle various HTTP requests.
- In the above WebApiConfig.Register() method, config.MapHttpAttributeRoutes() enables attribute routing
- he config.Routes is a route table or route collection of type HttpRouteCollection. The "DefaultApi" route is added in the route table using MapHttpRoute() extension method. The MapHttpRoute() extension method internally creates a new instance of IHttpRoute and adds it to an HttpRouteCollection. However, you can create a new route and add it into a collection manually as shown below.
- The following table lists parameters of MapHttpRoute() method.

Parameter	Description
name	Name of the route
routeTemplate	URL pattern of the route
defaults	An object parameter that includes default route values
constraints	Regex expression to specify characteristic of route values
handler	The handler to which the request will be dispatched.

- **Attribute Routing**

- Attribue routing is supported in Web API 2. As the name implies, attribute routing uses [Route()] attribute to define routes. The Route attribute can be applied on any controller or action method.
- In order to use attribute routing with Web API, it must be enabled in WebApiConfig by calling config.MapHttpAttributeRoutes() method.

Config

- Web API supports code based configuration. It cannot be configured in web.config file. We can configure Web API to customize the behaviour of Web API hosting infrastructure and components such as routes, formatters, filters, DependencyResolver, MessageHandlers, ParamterBindingRules, properties, services etc.
- We created a simple Web API project in the Create Web API Project section. Web API project includes default WebApiConfig class in the App_Start folder and also includes Global.asax
- Web API configuration process starts when the application starts. It calls GlobalConfiguration.Configure(WebApiConfig.Register) in the Application_Start method. The Configure() method requires the callback method where Web API has been configured in code. By default this is the static WebApiConfig.Register() method.
- WebApiConfig.Register() method includes a parameter of HttpConfiguration type which is then used to configure the Web API. The HttpConfiguration is the main class which includes following properties using which you can override the default behaviour of Web API.

Property	Description
DependencyResolver	Gets or sets the dependency resolver for dependency injection.
Filters	Gets or sets the filters.
Formatters	Gets or sets the media-type formatters.
IncludeErrorDetailPolicy	Gets or sets a value indicating whether error details should be included in error messages.
MessageHandlers	Gets or sets the message handlers.
ParameterBindingRules	Gets the collection of rules for how parameters should be bound.

Property	Description
Properties	Gets the properties associated with this Web API instance.
Routes	Gets the collection of routes configured for the Web API.
Services	Gets the Web API services.