Smit Vora

Learnings and Conclusion

Module – 5

Topics

25. **Introduction to Web Dev –** ASP.NET Core is a free, open-source web framework developed by Microsoft. It provides features that enable building the backend for modern web applications, as well as web APIs. The programming language that is used for the development of ASP.NET Core is C# or any other .NET-based programming language.

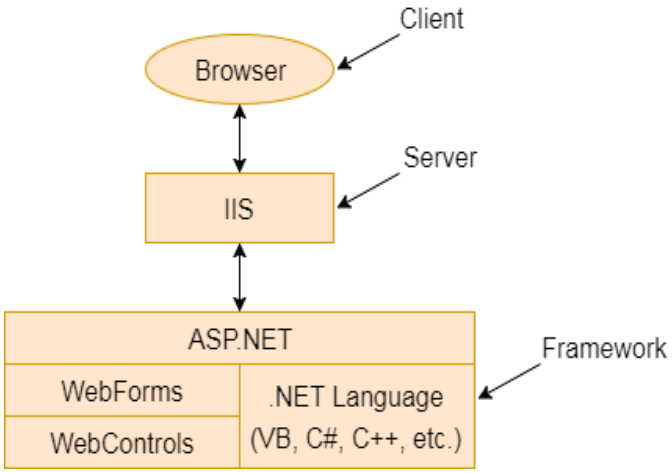
    ASP.NET Core is a redesign of the popular ASP.NET **Model View Controller** (**MVC**) and ASP.NET Web API frameworks.

    Both parts of the framework, MVC and Web API, help in creating modern web applications. MVC is for building traditional web applications in which rendering is done on the server-side, but also supports integration with modern JS libraries and client-side rendering.

    ASP.NET Core is a popular choice but is definitely not alone in the world of backend development. It is similar to other frameworks such as Laravel (PHP), Spring (Java), Ruby on Rails (Ruby), Django (Python), and others.

    25.1. ASP.Net Web Forms – Web Forms are the feature of Asp.Net framework which are used to build web pages. It is executed on a server and output is generated on a browser. It is compatible to any browser to any language supported by .NET common language runtime. Web Forms are made up of two components: the visual portion (the ASPX file), and the code behind the form, which resides in a separate class file.

    It's framework shown in the diagram below:



ASP.NET provides various controls like: server controls and HTML controls for the Web Forms.

25.1.1. ASP.Net Web Features:

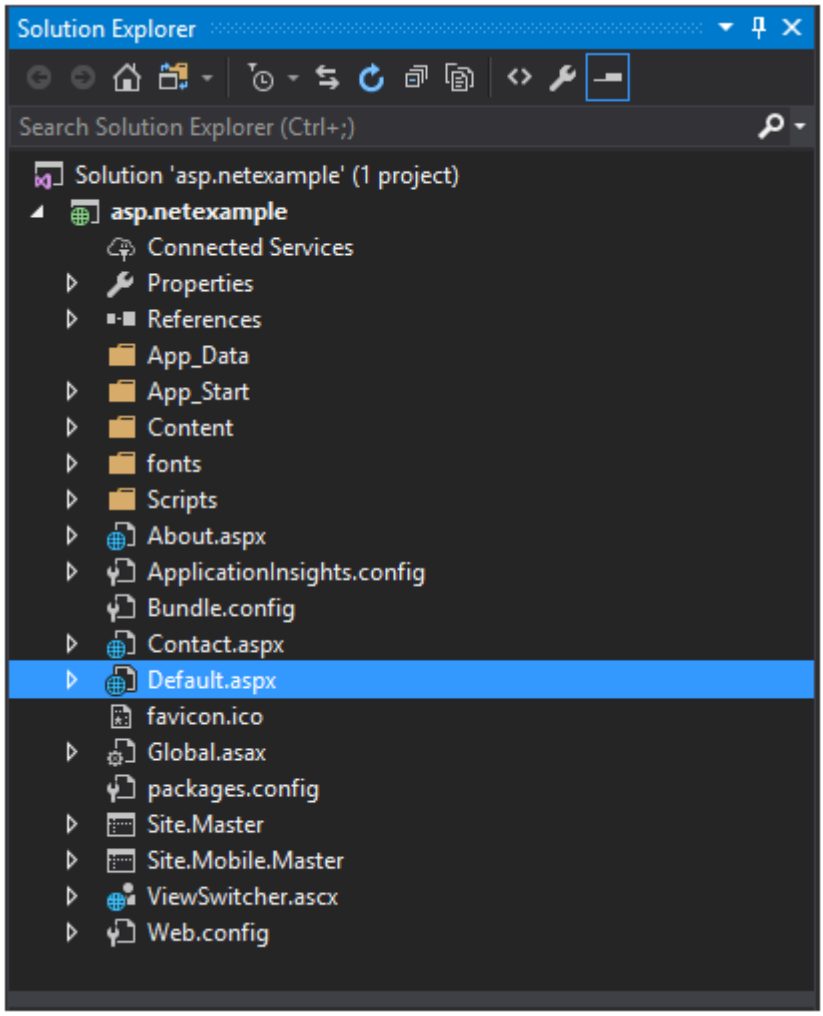
- Server Controls
- Master Pages
- Working with data
- Membership
- Client Script and Client Frameworks
- Routing
- State Management
- Security
- Performance
- Error Handling

25.1.2. Creating a Project in Web Forms

- Open Visual Studio.
- Select New Project from the File menu in Visual Studio.
- Create the Project - New Project Menu Item.
- Select the Templates -> Visual C# -> Web templates group on the left.
- Choose the ASP.NET Web Application template in the center column.
- Name your project and choose the OK button.
- Click the Change Authentication button. Select Individual User Accounts and click the OK button.

- Select the Web Forms template and click the OK button.

25.1.3. Project View:

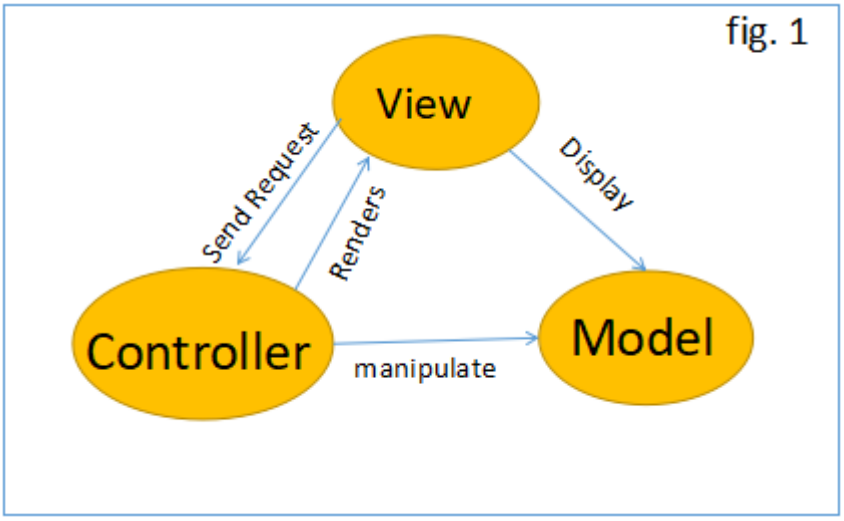


25.1.4. Some File And Their Uses:

File	Purpose
Default.aspx	Typically the first page displayed when the application is run in a browser.
Site.Master	A page that allows you to create a consistent layout and use standard behavior for pages in your application.
Global.asax	An optional file that contains code for responding to application-level and session-level events raised by ASP.NET or by HTTP modules.
Web.config	The configuration data for an application.

25.2. MVC – MVC is a framework which is used to decouple user-interface(**View**), Database(**Model**), application logic(**Controller**). It is used for separation of whole logic in these 3 categories and files.

25.2.1. Working –



25.2.2. Model – It is a c# class which represent the datatables in database as they are binded with the database. They even have validation rules if required for the fields resembling the datatable formation in database.

25.2.3. View – View is a component that forms the user interface. It is used to create web pages and it interacts with controller to show the data of model.

25.2.4. Controller - Controller is the component which handles user interaction. It works with the model and selects the view to render the web page. In an MVC application, the view only displays information whereas the controller handles and responds to the user input and requests. It separates the business logic from design logic and data logic.

25.2.5. Advantages of MVC are –

25.2.5.1. Loose Coupling

25.2.5.2. Lightweight code

25.2.5.3. Effective look

25.2.5.4. Testing is Very Easy

25.2.5.5. Rapid application integrated development.

25.2.6. Create an MVC project –

25.2.6.1. Start Visual Studio and select Create a new project.

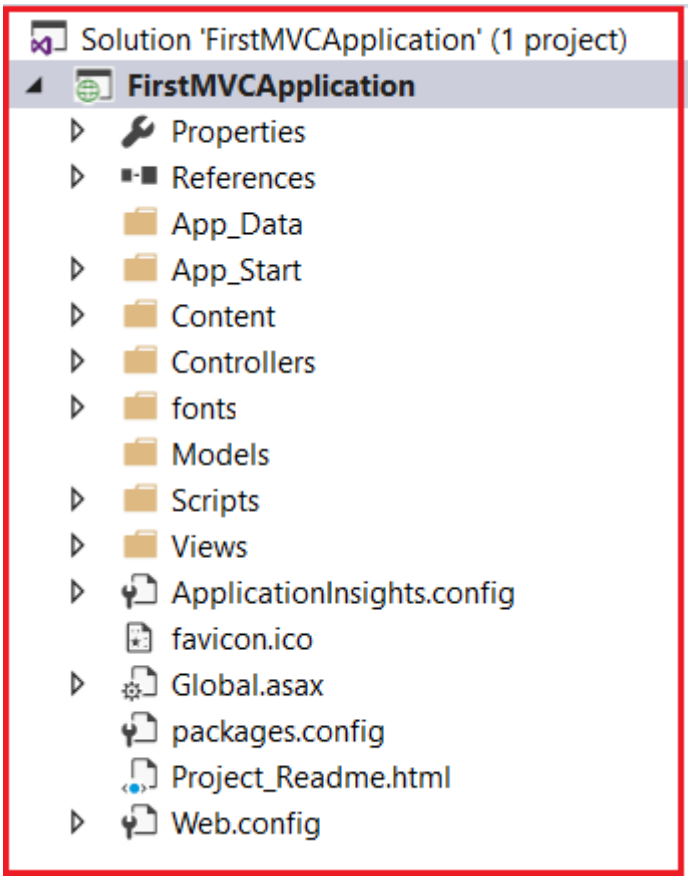
25.2.6.2. In the Create a new project dialog, select ASP.NET Core Web Application > Next.

25.2.6.3. In the Configure your new project dialog, enter **FirstMVCApplication** for Project name. It's important to name the project **FirstMVCApplication**. Capitalization needs to match each namespace matches when code is copied.

25.2.6.4. Select Create.

- In the Create a new ASP.NET Core web application dialog, select:
- .NET Core and ASP.NET Core 5.0 in the dropdowns.
- ASP.NET Core Web App (Model-View-Controller).
- Create.

25.2.7. Project Structure –



25.3. Rest Web API - 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.

- 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.

25.3.1. REST API or RESTFul services -

- Web API may or may not be RESTful services, but they are always HTTP based services.
- REST stands for Representational State Transfer.
- In REST API, only the state of the object is sent to the server to find the desired result.
- REST is an architectural pattern for developing an API that uses HTTP as its underlying communication method.
- As the name suggests it transfers state of the data at the time of its representation.
- It is different from SOAP as it uses just nouns to deal with the resources and specific functions are taken care of by HTTP protocol itself on which the concept of REST is based on.

25.3.2. Principles Of REST API –

25.3.2.1.1. **Stateless** – It means client requests contains all the data necessary for access and server need not need to save any data of request(state).

25.3.2.1.2. **Client-Server –** It follows a client-server model for communication.

25.3.2.1.3. **Uniform Interface –** Every Equivalent request receives the same response from the server interface.

25.3.2.1.4. **Cacheable –** For repeated requests data is saved in client browser so it doesn't need to fetch data every time from the server.

25.3.2.1.5. **Layered System –** There are several intermediaries(layers) in between client and server from which data flows

25.3.2.1.6. **Code on demand –** Data received may be in the form of a program which needs to be executed by client when received.

26. Start With Project

26.1. Create An API Project –

26.1.1. From the File menu, select New > Project.

26.1.2. Select the ASP.NET Core Web API template and click Next.

26.1.3. Name the project MyFirstWebAPIProject and click Create.

26.1.4. In the Create a new ASP.NET Core Web Application dialog, confirm that .NET Core and ASP.NET Core 5.0 are selected. Select the API template and click Create.

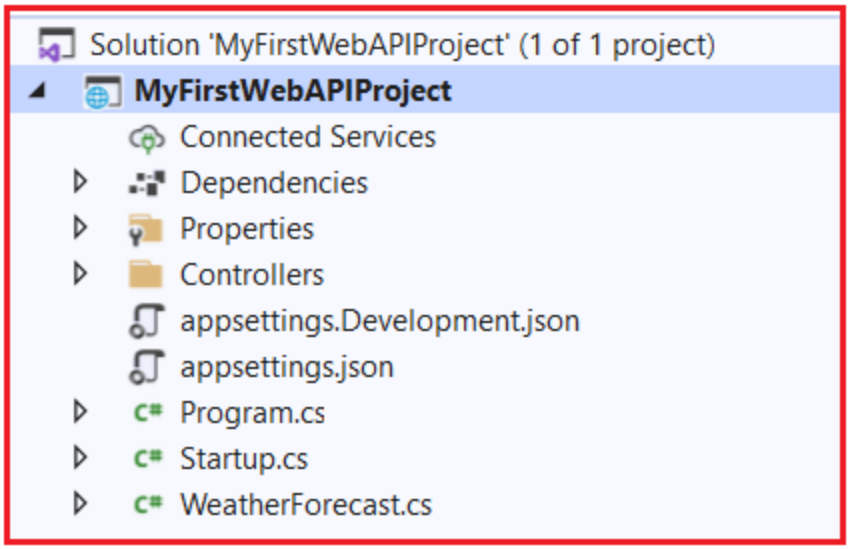
26.1.5. It can be created in XML, JSON or BSON format.

26.1.6. A model, view and controller can also be incorporated in the project like mvc.

26.1.7. A controller in web API returns just data and not the view.

26.1.8. The link of API which is written in the browser to get data includes an URI (Uniform Resource Identifier) to identify the resource via names or locations.

26.1.9. Project Structure –



26.2. Create Controller And Model -

26.2.1. Creating Controller

26.2.2. We right click on controller folder and click on Add then Add Controller.

26.2.3. Then we name the controller and it is created.

26.2.4. A controller can be created in both MVC and Web API.

26.2.5. In MVC it is executed in accordance to following URI.

26.2.6. URI : {Controller}/{Action}/{Id}

26.2.7. In Web API URI: API/{Controller}/{Id}

26.2.8. Controller gets requests from the client and then it shows them the data according to the action performed.

26.2.9. Creating Model

26.2.10. Models are the classes in the c# with fields and constraints mapped into the database as datatables.

26.2.11. We right click on model folder and click on Add then Add class.

26.2.12. Then we name the class and it is created.

26.2.13. We can write an ApplicationDbContext file with references of all our model classes and then we configure it to the database to connect it to the database.

26.2.14. Then we create object of that class in the controller to use it's properties.

26.3. Parameter Passing (From URI and from Body) – When Web API calls a method on the controller it must set parameters by a process called parameter binding.

26.3.1. So for simple datatypes like int, bool, double, decimal, string Web API tries to get values from URI by default.

26.3.2. For complex datatypes like user defined objects it takes Web API tries to read value from message body.

26.3.3. These are default cases but these methods can be overridden by [FromUri] for getting complex data from URI itself and [FromBody] for getting simple data from the message body.

26.3.4. These are used just to override in given conditions otherwise in default case other methods can be used.

26.4. Serialization – It is a process of storing objects in physical storage or send over a connection as a stream of bytes which after getting received by the recipient can be then be deserialized into an object again or when we want to read that object from the physical storage.

26.4.1. The reverse process is deserialization.

26.4.2. It is an technology that enables an object to be converted into a stream of data so they can be easily passes across the system or machine.

26.4.3. This format should be understandable by both end of a communication channel.

26.4.4. It is used by the web services , remoting for transmitting data between server and a client.

26.4.5. The namespace of serialization contain Iformatter interface which contain the methods serialize and de_serialize that can used to save and load data to and from a stream.

26.4.6. In order to implement serilization in .NET ,we basically require a stream and a formatter.

26.4.7. Stream act as a container for serializes object.

26.4.8. Formatter is used to serialize these objects onto the stream.

26.4.9. Namespaces used for Serialization -

26.4.9.1. System.Runtime.Serialization

26.4.9.2. System.Xml.Serialization

26.4.9.3. System.Text.Json

26.4.10. Types of Serialization in .Net

26.4.10.1. Json Serialization

26.4.10.2. XML and SOAP Serialization

26.4.10.3. Binary Serialization

26.4.11. SerializeTest class has two methods SerializeNow() and DeSerializeNow() which perform the task of serialization and deserialization respectively.

26.4.12. The general steps for serializing are –

26.4.12.1. Create an instance of File that will store serialized object.

26.4.12.2. Create a stream from the file object.

26.4.12.3. Create an instance of BinaryFormatter.

26.4.12.4. Call serialize method of the instance passing it stream and object to serialize.

26.4.12.5. The steps for de-serializing the object are similar. The only change is that you need to call deserialize method of BinaryFormatter object.

26.4.13. Json Serialization –

26.4.13.1. As the name suggest it is used to convert object in to the JSON stream format.

26.4.13.2. We can convert object in JSON stream format and also can get back that object from JSON stream using the concept of the serialization and de-serialization.

26.4.13.3. The quickest method of converting between JSON text and a .NET object is using the JsonSerializer.

26.4.13.4. The JsonSerializer converts .NET object into their JSON equivalent and back again by mapping the .NET object property names from the JSON property names and copies the values for you.

26.4.14. XML and SOAP Serialization –

26.4.14.1. It converts only the public fields and properties of object or parameters and return the values of methods into an XML stream.

26.4.14.2. XML serialization result in strongly typed classes with public properties.

26.4.14.3. Fields that is converted in serial for storage purpose or transport purpose.

26.4.14.4. As XML is an open source standard , XML stream can be processed by any application as need.

26.4.14.5. Implementing XML serialization in .NET is quite simple.

26.4.14.6. The basic class we need is xmlserializer for both serialization and de-serialization.

26.4.14.7. It much slower compare to binary serialization.

26.4.15. Binary Serialization –

26.4.15.1. It is mechanism which writes the data to the output stream such as it can be used to re construct the object automatically.

26.4.15.2. Binary refers to that the necessary information that is required to create object is saved onto the storage media.

26.4.15.3. It also preserves the instance identity.

26.4.15.4. In other word the binary serialization the entire object state is saved where in XML only some of the object data is saved.

26.5. Routing – Routing is a process of mapping the browser request to the required controller's action method. Each MVC Application has default route for the HomeController. We can set custom routing for other newly made Controllers.

26.5.1. Working :

26.5.1.1. When an URL's Request matches any registered route patterns the routing engine exacts it to the corresponding handler.

26.5.1.2. If it doesn't match then engine will generate corresponding error on the web page.

26.5.2. URL for API : api/{controller}/{id}

26.5.3. URL for MVC : {controller}/{action}/{id}

26.5.4. Types Of Routing

26.5.4.1. Conventional Routing :

26.5.4.2. Conventional or Traditional Routing also is a pattern matching system for URL that maps incoming request to the particular controller and action method.

26.5.4.3. We set all the routes in the RouteConfig file.

26.5.4.4. RouteConfig file is available in the App_Start folder.

26.5.4.5. We need to register all the routes to make them operational.

26.5.4.6. By giving this sort of routing every URL will match the default pattern given in the RouteConfig file to show the view if they don't then 404 error page is shown.

26.5.4.7. Advantage of it over attribute routing is that all the routing pattern is at one place which is generalized and mandatory for all paths to follow it makes routing simple.

26.5.4.8. Attribute Routing :

26.5.4.9. As it uses attribute [Route()] to write the the routing so it's called attribute routing.

26.5.4.10. It can be applied to any Controller and Action method.

26.5.4.11. It helps customize an URL pattern according to the needs and that holds a major advantage over conventional routing.

26.5.4.12. As Conventional routing pattern is all at one place and rules need to be followed by all controllers and actions so there is no room if we require a different URL pattern for an action method. But with attribute routing we may generalize the portion which is common for a controller over a controller and then give routes to action methods according to the requirement.

26.5.4.13. **WebApiConfig.Register()** method, **config.MapHttpAttributeRoutes()** enables attribute routing .The **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.

26.6. Config():

26.6.1. Web API configuration process starts when the application starts. It calls GlobalConfiguration.Configure(WebApiConfig.Register) in the Application_Start method.

26.6.2. The Configure() method requires the callback method where Web API has been configured in

code. By default this is the static WebApiConfig.Register() method.

26.6.3. WebApiConfig.Register() method includes a parameter of HttpConfiguration type which is then used to configure the Web API.

26.6.4. The HttpConfiguration is the main class which includes following properties using which you can override the default behaviour of Web API.