UNDERSTANDINGS & CONCLUSIONS

NANDINI BHATTACHARYA

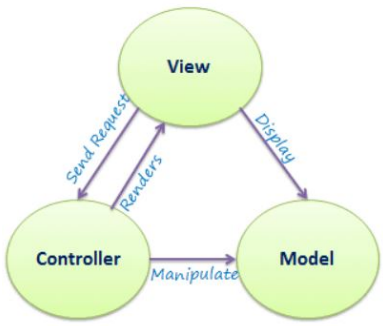
MODULE – 5

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.

25.2 MVC

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



- Let's see significance of each folder:
- **App_Data**: The App_Data folder can contain application data files like LocalDB, .mdf files, XML files, and other data related files. IIS will never serve files from App_Data folder.
- **App_Start:** The App_Start folder can contain class files that will be executed when the application starts. Typically, these would be config files like AuthConfig.cs, BundleConfig.cs, FilterConfig.cs, RouteConfig.cs etc.
- **Content**:The Content folder contains static files like CSS files, images, and icons files. MVC 5 application includes bootstrap.css, bootstrap.min.css, and Site.css by default.
- **Controllers:** The Controllers folder contains class files for the controllers. A Controller handles users' request and returns a response. MVC requires the name of all controller files to end with "Controller".
- **fonts**: The Fonts folder contains custom font files for your application.
- **Models:** The Models folder contains model class files. Typically model class includes public properties, which will be used by the application to hold and manipulate application data.
- **Scripts:** The Scripts folder contains JavaScript or VBScript files for the application.

- **Views:** The Views folder contains HTML files for the application. Typically view file is a .cshtml file where you write HTML and C# or VB.NET code.
- The Views folder includes a separate folder for each controller..
- The Shared folder under the View folder contains all the views shared among different controllers e.g., layout files.

25.3 REST Web API

- **REST** is an acronym for **RE**presentational **S**tate **T**ransfer.
- A Web API (or Web Service) conforming to the REST architectural style is a REST API.
- Restful Web Service, expose API from your application in a secure, uniform, stateless manner to the calling client. The calling client can perform predefined operations using the Restful service.
- The underlying protocol for REST is HTTP.
- REST is a way to access resources which lie in a particular environment. For example, you could have a server that could be hosting important documents or pictures or videos. All of these are an example of resources. If a client, say a web browser needs any of these are resources, it has to send a request to the server to access these resources. Now REST services define a way on how these resources can be accessed.

- The key elements of a RESTful implementation are as follows: **Resources, Request Verbs (POST, PUT, GET, DELETE), Request Headers, Request Body, Response Body, Response Status codes.**

- **RESTful Methods –**
  1. POST – This would be used to create a new data using the RESTful web service
  2. GET – This would be used to get a list of all data using the RESTful web service
  3. PUT – This would be used to update all data using the RESTful web service
  4. DELETE – This would be used to delete all data using the RESTful services

- **RESTful Architecture –**
  1. **State and functionality are divided into distributed resources** – This means that every resource should be accessible via the normal HTTP commands of GET, POST, PUT, or DELETE
  2. **The architecture is client/server, stateless, layered, and supports caching –**
     Client-server is the typical architecture where the server can be the web server hosting the application, and the client can be as simple as the web browser.
     Stateless means that the state of the application is not maintained in REST. For example, if you delete a resource from a server using the DELETE command, you cannot expect that delete information to be passed to the next request.

- **Principles Of REST API** –
  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).
  2. **Client-Server** – It follows a client-server model for communication.
  3. **Uniform Interface** – Every Equivalent request receives the same response from the server interface.
  4. **Cacheable** – For repeated requests data is saved in client browser so it doesn't need to fetch data every time from the server.
  5. **Layered System** – There are several intermediaries(layers) in between client and server from which data flows
  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 New Web API Project

1. From the File menu, select New -> Project.
2. Select the ASP.NET Core Web API template and click Next.

3. Name the project and click Create.
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.
5. It can be created in XML, JSON or BSON format.
6. A model, view and controller can also be incorporated in the project like MVC.
7. A controller in web API returns just data and not the view.
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.

26.2 Create Controller, Model

Creating Controller

1. Right click on controller folder and click on Add then Add Controller.
2. Then we name the controller and it is created.
3. A controller can be created in both MVC and Web API.
4. In MVC it is executed in accordance to following URI.
   URI : {Controller}/{Action}/{Id}
5. In Web API URI: API/{Controller}/{Id}
6. Controller gets requests from the client and then it shows them the data according to the action performed

Creating Model

7. Models are the classes in the c# with fields and constraints mapped into the database as datatables.
8. Right click on model folder and click on Add then Add class.
9. Then we name the class and it is created.
10. We can write a file with references of all our model classes and then we configure it to the database to connect it to the database.
11. Then we create object of that class in the controller to use it’s properties.

26.3 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.
- By default, if the parameter type is of .NET primitive types such as int, bool, double, string, GUID, DateTime, decimal, or any other type that can be converted from string type, then it sets the value of a parameter from the query string.
- And if the parameter type is the complex type, then Web API tries to get the value from the request body by default.
- 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.

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.
- The reverse process is deserialization.
- 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.

- 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.
- In order to implement serilization in .NET ,we basically require a stream and a formatter.
- Stream act as a container for serializes object. Formatter is used to serialize these objects onto the stream
- **Namespaces used for Serialization –**
  1. System.Runtime.Serialization
  2. System.Xml.Serialization
  3. System.Text.Json
- **SerializeTest** class has two methods **SerializeNow()** and **DeSerializeNow()** which perform the task of serialization and deserialization respectively.
- **Json Serialization**
  1. As the name suggest it is used to convert object in to the JSON stream format.
  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.
  3. The quickest method of converting between JSON text and a .NET object is using the JsonSerializer.
  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.
- **XML Serialization** –
  1. It converts only the public fields and properties of object or parameters and return the values of methods into an XML stream.
  2. XML serialization result in strongly typed classes with public properties.
  3. Fields that is converted in serial for storage purpose or transport purpose.
  4. As XML is an open source standard , XML stream can be processed by any application as need.
  5. Implementing XML serialization in .NET is quite simple.
  6. The basic class we need is xmlserializer for both serialization and de-serialization. It much slower compare to binary serialization.
- **Binary Serialization** –
  1. It is mechanism which writes the data to the output stream such as it can be used to re construct the object automatically.
  2. Binary refers to that the necessary information that is required to create object is saved onto the storage media.
  3. It also preserves the instance identity.
  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.
- When an URL’s Request matches any registered route patterns the routing engine exacts it to the corresponding handler.
- If it doesn’t match then engine will generate corresponding error on the web page.
  URL for API : api/{controller}/{id}
  URL for MVC : {controller}/{action}/{id}

- **Conventional Routing :** Conventional or Traditional Routing also is a pattern matching system for URL that maps incoming request to the particular controller and action method.
- We set all the routes in the RouteConfig file.
- RouteConfig file is available in the App_Start folder.
- We need to register all the routes to make them operational.
- 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.
- 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.
- **Attribute Routing** : As it uses attribute [Route()] to write the the routing so it's called attribute routing.
- It can be applied to any Controller and Action method.
- It helps customize an URL pattern according to the needs and that holds a major advantage over conventional routing.
- 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.
- 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.

26.5 Config()

- 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