Module 5 Theory

1) **Web Forms** are web pages built on the ASP.NET Technology. It executes on the server and generates output to the browser. It is compatible to any browser to any language supported by .NET common language run time. It is flexible and allows us to create and add custom controls.

We can use Visual Studio to create ASP.NET Web Forms. It is an IDE (Integrated Development Environment) that allows us to drag and drop server controls to the web forms. It also allows us to set properties, events and methods for the controls. To write business logic, we can choose any .NET language like: Visual Basic or Visual C#.

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.

2) The **MVC** (Model-View-Controller) is an application development pattern or design pattern which separates an application into three main components:

- Model: Model is a part of the application which implements the logic for the data domain of the application. It is used to retrieve and store model state in a database such as SQL Server database. It also used for business logic separation from the data in the application.
- View: View is a component that forms the application's user interface. It is uses to create web pages for the application. An example would be an edit view of a Products table that displays text boxes, drop-down lists and check boxes based on the current state of a Product object.
- 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.

This design pattern is a lightweight framework which is integrated with various features such as master pages and membership based authentication. It is defined in the System.Web.Mvc assembly.

This approach provides the following advantages.

- It manages application complexity by dividing an application into the model, view and controller.
- It does not use view state or server-based forms. This makes the MVC framework ideal for developers who want full control over the behavior of an application.
- It provides better support for test-driven development.
- It is suitable for large scale developer team and web applications.
- It provides high degree of control to the developer over the application behavior.

3) **REST** is the acronym that stands for: Representational State Transfer. REST is an architectural style of distributed system. It is based upon the set of principles that describes how network resources are defined and addressed. These set of principles was first described by "Roy Fielding" in 2000. REST is bigger than Web Services.

RESTful services uses HTTP (Hyper Text Transfer Protocol) to communicate. REST system interface with external systems as web resources identified by URIs (Uniform Resource Identifiers).

4) [Reference : https://www.tutorialsteacher.com/webapi]

Action methods in Web API controller can have one or more **parameters** of different types. It can be either primitive type or complex type. Web API binds action method parameters either with URL's query string or with request body depending on the parameter type. By default, if parameter type is of .NET primitive type 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 complex type then Web API tries to get the value from request body by default.

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.

5) The Web API **serialize** all the public properties into JSON. In the older versions of Web API, the default serialization property was in PascalCase. When we are working with .NET based applications, the casing doesn't matter. But when the API is consumed by another application, such as Angular JS or any other application, then most of the cases are CamelCase as per the JavaScript standard. Moreover, most of the developers are familiar with this type of naming.

ASP.NET Core becomes the camelCase serialization as default. If we have migrated the application from Web API 2.0 to .NET core, the application may not be working. However, we can configure the serialization to PascalCase.

[refer : https://docs.microsoft.com/en-us/aspnet/web-api/overview/formats-and-model-binding/json-and-xml-serialization
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/serialization/]

6) **Web API routing** is similar to ASP.NET MVC 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.

When we created Web API project using WebAPI template in the Create Web API Project section, it also added WebApiConfig class in the App_Start folder with default route.

Attribute Routing - Attribute 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.

[Reference : https://www.tutorialsteacher.com/webapi]

7) 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.

[Reference : https://www.tutorialsteacher.com/webapi]