**6. Web Development**

**6.1 Introduction to Web Development**

* In C# ASP.NET, a web application refers to a dynamic and interactive software solution that incorporates server-side logic to process user requests and generate dynamic content.
* It typically follows a multi-tier architecture, involving separate layers for presentation, business logic, and data access.
* Web applications often have complex functionality, user authentication, database integration, and can provide services such as e-commerce, social networking, or productivity tools.
* They are built using frameworks like ASP.NET MVC or ASP.NET Core, and they often involve the use of server-side technologies such as C# for coding the logic.
* Key components :
  + **ASP.NET Core :**

ASP.NET Core is an open-source, cross-platform framework for building modern, cloud-based, and internet-connected applications. It's highly modular and allows developers to choose the components they need, making it more lightweight and flexible.

* + **MVC (Model-View-Controller) Architecture :**

ASP.NET Core follows the MVC architectural pattern, which separates the application into three main components: Model (data and business logic), View (user interface), and Controller (handles user input and manages communication between Model and View).

* + **Languages :**

These include C#, VB.NET and J#, as well as compilers.

* + **A base framework for processing web requests :**

As part of the ASP.NET framework, servers evaluate the code that web developers write with C# and send HTML to the user. Client-side code is written in JavaScript.

* + **Razor Pages and Views :**

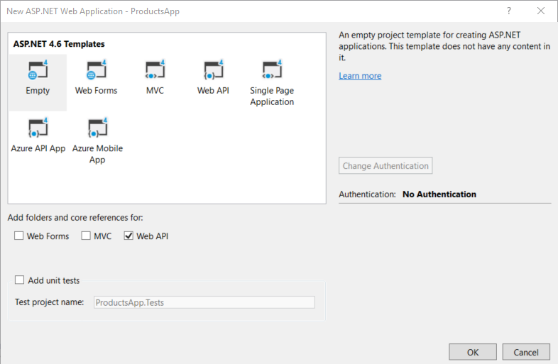
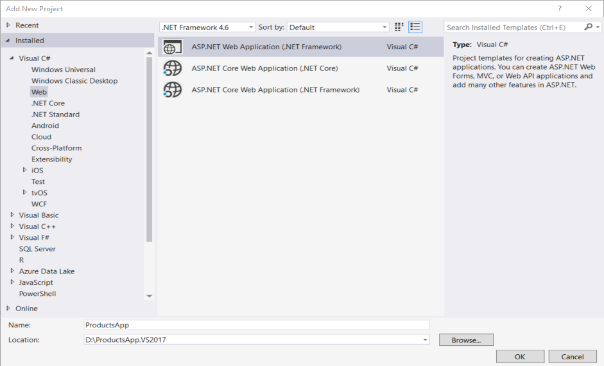
Razor is a view engine that enables the creation of dynamic and expressive views in ASP.NET Core. Razor Pages provide a simpler page-based programming model, while Razor Views are used within the MVC framework for rendering HTML.

**6.2 Web API project**

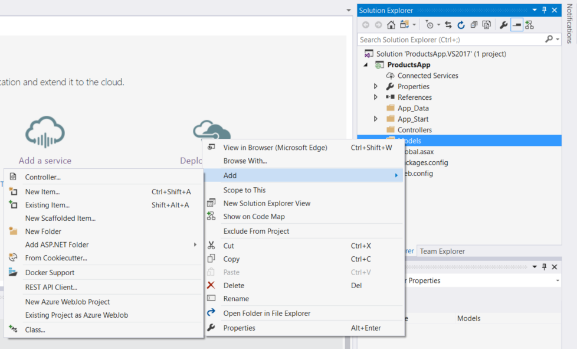
* Web API is an application programming interface (API) that is used to enable communication or interaction with software components with each other. ASP.NET Web API is a framework that makes it easy to build HTTP Service that reaches a broad range of clients, including browsers and mobile devices. Using ASP.NET, web API can enable communication by different devices from the same database.
* Main Components of web API project :
  + Controllers :
    - In Web API, a controller is an object that handles HTTP requests.
  + Models :
    - A model is an object that represents the data in your application. ASP.NET Web API can automatically serialize your model to JSON, XML, or some other format, and then write the serialized data into the body of the HTTP response message. As long as a client can read the serialization format, it can deserialize the object. Most clients can parse either XML or JSON. Moreover, the client can indicate which format it wants by setting the Accept header in the HTTP request message.

**6.3 Building Web API**

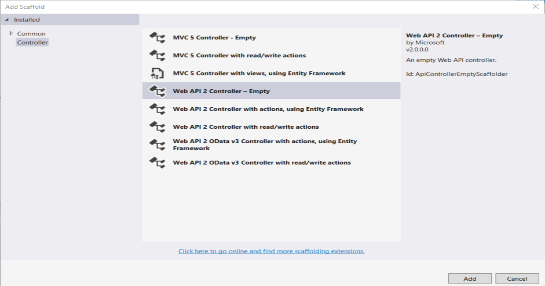
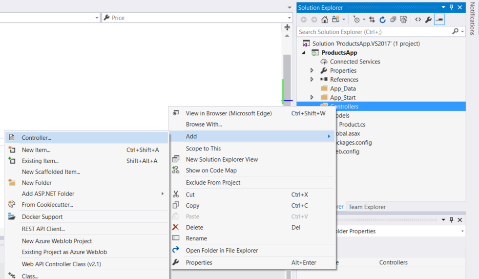
* Steps for building Web API is as follows :
  1. Create a Web API project



* 1. Add Model



* 1. Add Controller



* 1. Build project and fetch data with appropriate URL

**6.4 Action Method Response**

* Return types of action methods will be embedded in the Web API response sent to the client.
* The Web API action method can have following return types.

1. Void

It's not necessary that all action methods must return something. It can have void return type.

1. Primitive type or Complex type

An action method can return primitive or other custom complex types as other normal methods.

For example custom class object,etc.

1. HttpResponseMessage

Web API controller always returns an object of HttpResponseMessage to the hosting infrastructure. The following figure illustrates the overall Web API request/response pipeline.

1. IHttpActionResult

An action method in Web API 2 can return an implementation of IHttpActionResult class

You can create your own class that implements IHttpActionResult or use various methods of ApiController class that returns an object that implement the IHttpActionResult.

The following table lists all the methods of ApiController class that returns an object of a class that implements IHttpActionResult interface.

|  |  |
| --- | --- |
| ApiController Method | Description |
| BadRequest() | Creates a BadRequestResult object with status code 400. |
| Conflict() | Creates a ConflictResult object with status code 409. |
| Content() | Creates a NegotiatedContentResult with the specified status code and data. |
| Created() | Creates a CreatedNegotiatedContentResult with status code 201 Created. |
| CreatedAtRoute() | Creates a CreatedAtRouteNegotiatedContentResult with status code 201 created. |
| InternalServerError() | Creates an InternalServerErrorResult with status code 500 Internal server error. |
| NotFound() | Creates a NotFoundResult with status code404. |
| Ok() | Creates an OkResult with status code 200. |
| Redirect() | Creates a RedirectResult with status code 302. |
| RedirectToRoute() | Creates a RedirectToRouteResult with status code 302. |
| ResponseMessage() | Creates a ResponseMessageResult with the specified HttpResponseMessage. |
| StatusCode() | Creates a StatusCodeResult with the specified http status code. |
| Unauthorized() | Creates an UnauthorizedResult with status code 401. |

**6.5 Security**

* Security can achieve in Web API by following measure techniques :
* CORS

Browser security prevents a web page from making AJAX requests to another domain. This restriction is called the same-origin policy, and prevents a malicious site from reading sensitive data from another site. However, sometimes you might want to let other sites call your web API.

[Cross Origin Resource Sharing](http://www.w3.org/TR/cors/) (CORS) is a W3C standard that allows a server to relax the same-origin policy. Using CORS, a server can explicitly allow some cross-origin requests while rejecting others.

* Authentication

Authentication is used to protect our applications and websites from unauthorized access and also, it restricts the user from accessing the information from tools like postman and fiddler.

To access the web API method, we have to pass the user credentials in the request header. If we do not pass the user credentials in the request header, then the server returns 401 (unauthorized) status code indicating the server supports Basic Authentication.

* Authorization

Authorization allows a website user to grant and restrict permissions on Web pages, functionality, and data. Authorization checks whether a user is allowed to perform an action or has access to some functionality. For example, having the permission to get data and post data is a part of authorization.

Web API uses authorization filters to implement authorization.  The Authorization filters run before the controller action. If the request is not authorized, the filter returns an error response, and the action is not invoked.  
Web API provides a built-in authorization filter, Authorize Attribute. This filter checks whether the user is authenticated. If not then it returns the HTTP status code 401 (Unauthorized), without invoking the action.

* JWT Token

JWTs or JSON Web Tokens are most commonly used to identify an authenticated user. They are issued by an authentication server and are consumed by the client-server (to secure its APIs).

A JWT contains three parts:

Header:

Consists of two parts:

1. The signing algorithm that’s being used.

2. The type of token, which, in this case, is mostly “JWT”.

Payload:

The payload contains the claims or the JSON object.

Signature:

A string that is generated via a cryptographic algorithm that can be used to verify the integrity of the JSON payload.

**6.6 HTTP Caching**

* Caching is a technique of storing frequently used data or information in a local memory, for a certain time period. So, next time, when the client requests the same information, instead of retrieving the information from the database, it will give the information from the local memory. The main advantage of caching is that it improves the performance by reducing the processing burden.
* Common Cache-Control directives are shown in the following table.

|  |  |
| --- | --- |
| Directive | Action |
| [public](https://www.rfc-editor.org/rfc/rfc9111#cache-response-directive.public) | A cache may store the response. |
| [private](https://www.rfc-editor.org/rfc/rfc9111#cache-response-directive.private) | The response must not be stored by a shared cache. A private cache may store and reuse the response. |
| [max-age](https://www.rfc-editor.org/rfc/rfc9111#cache-response-directive.max-age) | The client doesn't accept a response whose age is greater than the specified number of seconds. Examples: max-age=60 (60 seconds), max-age=2592000 (1 month) |
| [no-cache](https://www.rfc-editor.org/rfc/rfc9111#cache-response-directive.no-cache) | On requests: A cache must not use a stored response to satisfy the request. The origin server regenerates the response for the client, and the middleware updates the stored response in its cache.  On responses: The response must not be used for a subsequent request without validation on the origin server. |
| [no-store](https://www.rfc-editor.org/rfc/rfc9111#cache-response-directive.no-store) | On requests: A cache must not store the request.  On responses: A cache must not store any part of the response. |
| [Age](https://www.rfc-editor.org/rfc/rfc9111#field.age) | An estimate of the amount of time in seconds since the response was generated or successfully validated at the origin server. |
| [Expires](https://www.rfc-editor.org/rfc/rfc9111#field.expires) | The time after which the response is considered stale. |

**6.7 Versoning**

* [Web API](https://www.dotnettricks.com/learn/webapi/what-is-web-api-and-why-to-use-it)Versioning is required as the business grows and business requirement changes with the time. As Web API can be consumed by multiple clients at a time, Versioning of Web API will be necessarily required so that Business changes in the API will not impact the client that are using/consuming the existing API.
* Web API Versioning can be done by using the following methods:
* URI
* QueryString parameter
* Custom Header parameter
* Accept Header parameter