**Web API**

https://docs.microsoft.com/en-us/aspnet/web-api/overview/web-api-routing-and-actions/routing-in-aspnet-web-api

**Cross-Site Request Forgery (CSRF)** is an attack where a malicious site

sends a request to a vulnerable site where the user is currently logged in.

To help prevent CSRF attacks, ASP.NET MVC uses **anti-forgery tokens**,

also called request verification tokens.

1. The client requests an HTML page that contains a form.

2. The server includes two tokens in the response. One token is sent as a cookie.

The other is placed in a hidden form field.

The tokens are generated randomly so that an adversary cannot guess the values.

3. When the client submits the form, it must send both tokens back to the server.

The client sends the cookie token as a cookie, and it sends the form token

inside the form data.

(A browser client automatically does this when the user submits the form.)

4. If a request does not include both tokens, the server disallows the request.

**Cross Origin Resource Sharing (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.

config.EnableCors(); in webapi config

[EnableCORS] on controller

This flow includes both authentication and authorization steps.

Authentication proves the identity of the client.

Authorization determines whether the client can access a particular resource.

[RequireHttps] for ssl authentication

**Routing in web api**

Once a matching route is found, Web API selects the controller and the action:

To find the controller, Web API adds "Controller" to the value of the

{controller} variable.

To find the action, Web API looks at the HTTP verb,

and then looks for an action whose name begins with that HTTP verb name.

For example, with a GET request, Web API looks for an action prefixed with

"Get", such as "GetContact" or "GetAllContacts".

This convention applies only to GET, POST, PUT, DELETE, HEAD, OPTIONS,

and PATCH verbs.

You can enable other HTTP verbs by using attributes on your controller.

We'll see an example of that later.

Other placeholder variables in the route template, such as {id},

are mapped to action parameters.

**Attribute routing:**

Add [Route("customers/{customerId}/order")] to the HTTP method

[Route("~/api/authors/{authorId}/books")]

Also to enable attribute routing, call MapHttpAttributeRoutes during configuration.

You can set a common prefix for an entire controller by using the

[RoutePrefix] attribute: