ASP.NET Web API

# Overview to ASP.NET Core

# ASP.NET Core fundamentals overview

# Dependency injection in ASP.NET Core

# Create web APIs with ASP.NET Core

# Controller action return types in ASP.NET Core web API

# Format response data in ASP.NET Core Web API

# Web API Request/Response Data Formats

# ASP.NET Web API: Media-Type Formatters

# Custom formatters in ASP.NET Core Web API

# Model Validation in ASP.NET Web API

# Parameter Binding in ASP.NET Web API

# ASP.NET Core Middleware

# Options pattern in ASP.NET Core

# Use multiple environments in ASP.NET Core

# Logging in .NET Core and ASP.NET Core

# Routing in ASP.NET Core

# Error Handling in ASP.NET Web API

# ASP.NET Core web API documentation with Swagger / OpenAPI

# Filters in ASP.NET Core

# Overview of ASP.NET Core authentication

**What is Web API?**

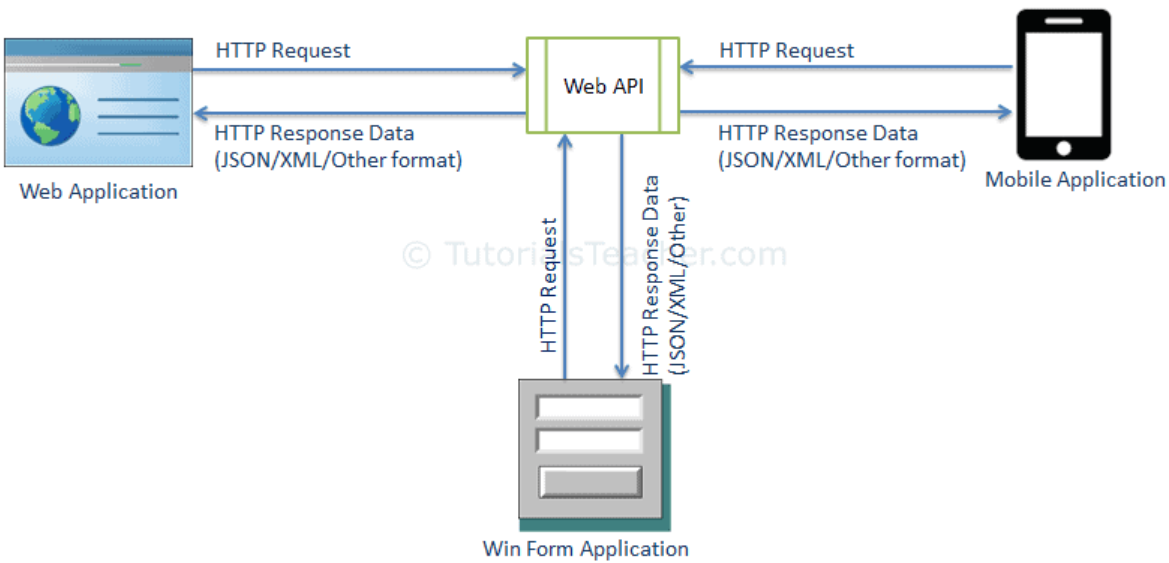
As per Wikipedia's Definition of API: In computer programming, an application programming interface (API) is a set of subroutine definitions, protocols, and tools for building software and applications.

To put it in simple terms, 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.

Web API as the name suggests, is an API over the web which can be accessed using HTTP protocol. It is a concept and not a technology. We can build Web API using different technologies such as Java, .NET etc. For example, Twitter's REST APIs provide programmatic access to read and write data using which we can integrate twitter's capabilities into our own application.

**What is ASP.NET Web API?**

The ASP.NET Web API is an extensible framework for building HTTP based services that can be accessed in different applications on different platforms such as web, windows, mobile etc. It works more or less the same way as ASP.NET MVC web application except that it sends data as a response instead of html view. It is like a webservice or WCF service but the exception is that it only supports HTTP protocol.



**What are the characteristics of ASP.NET Web API?**

1. ASP.NET Web API is an ideal platform for building RESTful services.
2. ASP.NET Web API is built on top of ASP.NET and supports ASP.NET request/response pipeline
3. ASP.NET Web API maps HTTP verbs to method names.
4. ASP.NET Web API supports different formats of response data. Built-in support for JSON, XML, BSON format.
5. ASP.NET Web API can be hosted in IIS, Self-hosted or other web server that supports .NET 4.0+.
6. 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.

**What are the differences between ASP.NET Web API vs WCF?**

| Web API | WCF |
| --- | --- |
| Open source and ships with .NET framework. | Ships with .NET framework |
| Supports only HTTP protocol. | Supports HTTP, TCP, UDP and custom transport protocol. |
| Maps http verbs to methods | Uses attributes based programming model. |
| Uses routing and controller concept similar to ASP.NET MVC. | Uses Service, Operation and Data contracts. |
| Does not support Reliable Messaging and transaction. | Supports Reliable Messaging and Transactions. |
| Web API can be configured using HttpConfiguration class but not in web.config. | Uses web.config and attributes to configure a service. |
| Ideal for building RESTful services. | Supports RESTful services but with limitations. |

### When to choose WCF?

* Choose WCF if you use .NET Framework 3.5. Web API does not support .NET 3.5 or below.
* Choose WCF if your service needs to support multiple protocols such as HTTP, TCP, Named pipe.
* Choose WCF if you want to build service with WS-\* standards like Reliable Messaging, Transactions, Message Security.
* Choose WCF if you want to use Request-Reply, One Way, and Duplex message exchange patterns.

### When to choose ASP.NET Web API?

* Choose Web API if you are using .NET framework 4.0 or above.
* Choose Web API if you want to build a service that supports only HTTP protocol.
* Choose Web API to build RESTful HTTP based services.
* Choose Web API if you are familiar with ASP.NET MVC.

**How do you test Web API?**

We can use the following third party tools for testing Web API.

* [Fiddler](http://www.telerik.com/fiddler)
* [Postman](https://www.getpostman.com/)

# mvc vs api

# What changes for WCF to make restapi?

# Overview to ASP.NET Core

<https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0>

# ASP.NET Core fundamentals overview

<https://learn.microsoft.com/en-us/aspnet/core/fundamentals/?view=aspnetcore-6.0&tabs=windows>

# Want to send a mail in code, but in development and production mail ids are different. How will you handle?

# How to access data from appSettings.json in the code?

# In Startup.cs which is Optional? Configure or ConfigureServices?

# In ConfigureServices apart from injecting services what else can be done?

# Why is Host is required in .NetCore and Is host required in .NetFramework? Is IISExpress is a Host?

# What is Kestrel and what is WebListener?

# What is the use of Program.cs? Why are you starting WebListener? or Host?

# Want to send a mail in code, but in development and production mail ids are different. How will you handle?

# How to access data from appSettings.json in the code?

# How and when to design REST API asynchronous?

# Are rest API calls threadsafe? Do we have to make REST API threadsafe?

# Dependency injection in ASP.NET Core

<https://learn.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-6.0>

* What is dependency Injection? What is Singleton, Transient and Scoped? Is there any other way to achieve loosely coupled systems?
* Can you inject Scoped into a Singleton class?
* Add scoped vs add transient.
* Can we use transient inside singleton? What issue can happen? Will it throw any exception at compile time or runtime?
* Can we inject transient in singleton?
* Relatime use of DI(Singlton, Scoped and transient)
* How do you avoid injection in .Net core?

# Create web APIs with ASP.NET Core

<https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-6.0>

**What is a controller?**

ASP.NET Core supports creating web APIs using controllers or using minimal APIs. Controllers in a web API are classes that derive from [ControllerBase](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.controllerbase).

A controller-based web API consists of one or more controller classes that derive from [ControllerBase](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.controllerbase). The web API project template provides a starter controller:

[ApiController]

[Route("[controller]")]

public class WeatherForecastController : ControllerBase

Web API controllers should typically derive from [ControllerBase](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.controllerbase) rather from [Controller](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.controller). Controller derives from [ControllerBase](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.controllerbase) and adds support for views, so it's for handling web pages, not web API requests. If the same controller must support views and web APIs, derive from Controller.

**What are the methods and properties available in ControllerBase?**

The ControllerBase class provides many properties and methods that are useful for handling HTTP requests. For example, [CreatedAtAction](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.controllerbase.createdataction) returns a 201 status code:

[HttpPost]

[ProducesResponseType(StatusCodes.Status201Created)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public ActionResult<Pet> Create(Pet pet)

{

pet.Id = \_petsInMemoryStore.Any() ?

\_petsInMemoryStore.Max(p => p.Id) + 1 : 1;

\_petsInMemoryStore.Add(pet);

return CreatedAtAction(nameof(GetById), new { id = pet.Id }, pet);

}

The following table contains examples of methods in ControllerBase.

|  |  |
| --- | --- |
|  |  |
|  |  |

For a list of all available methods and properties, see [ControllerBase](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.controllerbase).

**What are the uses of attributes?**

The [Microsoft.AspNetCore.Mvc](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc) namespace provides attributes that can be used to configure the behavior of web API controllers and action methods. The following example uses attributes to specify the supported HTTP action verb and any known HTTP status codes that could be returned:

[HttpPost]

[ProducesResponseType(StatusCodes.Status201Created)] [ProducesResponseType(StatusCodes.Status400BadRequest)]

public ActionResult<Pet> Create(Pet pet)

{

pet.Id = \_petsInMemoryStore.Any() ?

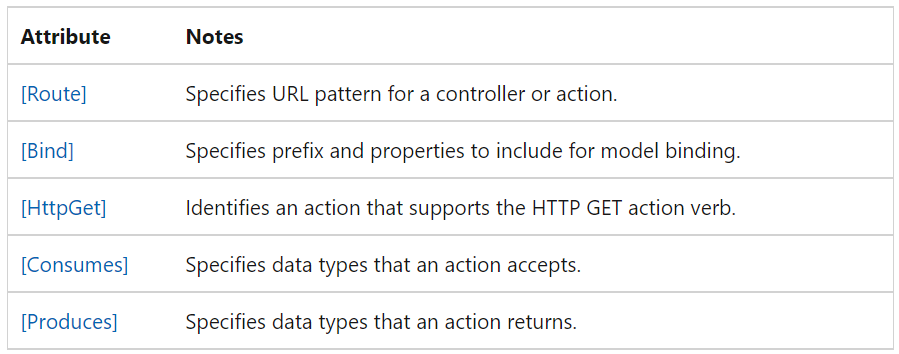
\_petsInMemoryStore.Max(p => p.Id) + 1 : 1;

\_petsInMemoryStore.Add(pet);

return CreatedAtAction(nameof(GetById), new { id = pet.Id }, pet);

}

Here are some more examples of attributes that are available.



For a list that includes the available attributes, see the [Microsoft.AspNetCore.Mvc](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc) namespace.

**What is the use of ApiController attribute?**

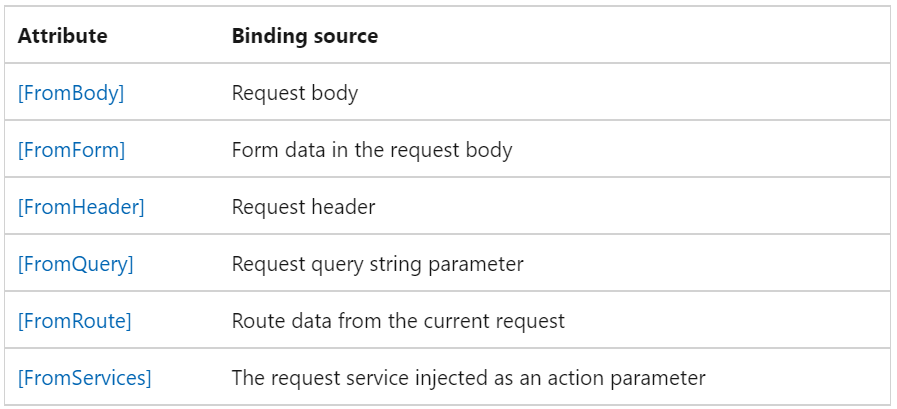
The [[ApiController]](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.apicontrollerattribute) attribute can be applied to a controller class to enable the following opinionated, API-specific behaviors:

* [Attribute routing requirement](https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-6.0#attribute-routing-requirement)
* [Automatic HTTP 400 responses](https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-6.0#automatic-http-400-responses)
* [Binding source parameter inference](https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-6.0#binding-source-parameter-inference)
* [Multipart/form-data request inference](https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-6.0#multipartform-data-request-inference)
* [Problem details for error status codes](https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-6.0#problem-details-for-error-status-codes)

More details can be found here. <https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-6.0>

**What is binding source attribute?**

A binding source attribute defines the location at which an action parameter's value is found. The following binding source attributes exist:



Without the [ApiController] attribute or binding source attributes like [FromQuery], the ASP.NET Core runtime attempts to use the complex object model binder. The complex object model binder pulls data from value providers in a defined order.

In the following example, the [FromQuery] attribute indicates that the discontinuedOnly parameter value is provided in the request URL's query string:

[HttpGet]

public ActionResult<List<Product>> Get([FromQuery] bool discontinuedOnly = false)

{

List<Product> products = null;

if (discontinuedOnly)

{

products = \_productsInMemoryStore.Where(p => p.IsDiscontinued).ToList();

}

else

{

products = \_productsInMemoryStore;

}

return products;

}

The [ApiController] attribute applies inference rules for the default data sources of action parameters. These rules save you from having to identify binding sources manually by applying attributes to the action parameters. The binding source inference rules behave as follows:

* [FromBody] is inferred for complex type parameters not registered in the DI Container. An exception to the [FromBody] inference rule is any complex, built-in type with a special meaning, such as [IFormCollection](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.http.iformcollection) and [CancellationToken](https://learn.microsoft.com/en-us/dotnet/api/system.threading.cancellationtoken). The binding source inference code ignores those special types.
* [FromForm] is inferred for action parameters of type [IFormFile](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.http.iformfile) and [IFormFileCollection](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.http.iformfilecollection). It's not inferred for any simple or user-defined types.
* [FromRoute] is inferred for any action parameter name matching a parameter in the route template. When more than one route matches an action parameter, any route value is considered [FromRoute].
* [FromQuery] is inferred for any other action parameters.

FromBody in Httpget?

Can we update using Post? Yes? then what is need of Put? Why Put for Update?

<https://stackoverflow.com/questions/43258725/use-post-instead-of-put-rest>

Use post for update? (bulk update and insert)

# Controller action return types in ASP.NET Core web API

<https://learn.microsoft.com/en-us/aspnet/core/web-api/action-return-types?view=aspnetcore-6.0>

**What are the different return types of Web API?**

ASP.NET Core offers the following options for web API controller action return types:

* [Specific type](https://learn.microsoft.com/en-us/aspnet/core/web-api/action-return-types?view=aspnetcore-6.0#specific-type)
* [IActionResult](https://learn.microsoft.com/en-us/aspnet/core/web-api/action-return-types?view=aspnetcore-6.0#iactionresult-type)
* [ActionResult<T>](https://learn.microsoft.com/en-us/aspnet/core/web-api/action-return-types?view=aspnetcore-6.0#actionresultt-type)

**When to use IActionResult and HttpResponse? What is the diff?**

**What is the difference between IActionResult and ActionResult?**

ASP.NET Core includes the [ActionResult<T>](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.actionresult-1) return type for web API controller actions. It enables you to return a type deriving from [ActionResult](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.actionresult) or return a [specific type](https://learn.microsoft.com/en-us/aspnet/core/web-api/action-return-types?view=aspnetcore-6.0#specific-type). ActionResult<T> offers the following benefits over the [IActionResult type](https://learn.microsoft.com/en-us/aspnet/core/web-api/action-return-types?view=aspnetcore-6.0" \l "iactionresult-type):

* The [[ProducesResponseType]](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.producesresponsetypeattribute) attribute's Type property can be excluded. For example, [ProducesResponseType(200, Type = typeof(Product))] is simplified to [ProducesResponseType(200)]. The action's expected return type is instead inferred from the T in ActionResult<T>.
* [Implicit cast operators](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/implicit) support the conversion of both T and ActionResult to ActionResult<T>. T converts to [ObjectResult](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.objectresult), which means return new ObjectResult(T); is simplified to return T;.

### Synchronous action

Consider a synchronous action in which there are two possible return types:

[HttpGet("{id}")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

public ActionResult<Product> GetById(int id)

{

if (!\_repository.TryGetProduct(id, out var product))

{

return NotFound();

}

return product;

}

In the preceding action:

* A 404 status code is returned when the product doesn't exist in the database.
* A 200 status code is returned with the corresponding Product object when the product does exist.

### Asynchronous action

Consider an asynchronous action in which there are two possible return types:

[HttpPost]

[Consumes(MediaTypeNames.Application.Json)]

[ProducesResponseType(StatusCodes.Status201Created)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public async Task<ActionResult<Product>> CreateAsync(Product product)

{

if (product.Description.Contains("XYZ Widget"))

{

return BadRequest();

}

await \_repository.AddProductAsync(product);

return CreatedAtAction(nameof(GetById), new { id = product.Id }, product);

}

In the preceding action:

* A 400 status code ([BadRequest](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.controllerbase.badrequest)) is returned by the ASP.NET Core runtime when:
  + The [[ApiController]](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.apicontrollerattribute) attribute has been applied and model validation fails.
  + The product description contains "XYZ Widget".
* A 201 status code is generated by the [CreatedAtAction](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.controllerbase.createdataction) method when a product is created. In this code path, the Product object is provided in the response body. A Location response header containing the newly created product's URL is provided.

**Can a WebAPI return a string, List<Employee>, etc? If not why? Why is it good to have object wrapper around your list when returning?**

**The answer is to make it backward compatible.**

**Where we configure return types in webapi – mediatype formatter**

**Status code what is return code : 202**

* 201 response code?
* 400 Bad Request – client sent an invalid request, such as lacking required request body or parameter
* 401 Unauthorized – client failed to authenticate with the server
* 403 Forbidden – client authenticated but does not have permission to access the requested resource
* 404 Not Found – the requested resource does not exist
* 412 Precondition Failed – one or more conditions in the request header fields evaluated to false
* 500 Internal Server Error – a generic error occurred on the server
* 503 Service Unavailable – the requested service is not available

# Format response data in ASP.NET Core Web API

[**https://learn.microsoft.com/en-us/aspnet/core/web-api/advanced/formatting?view=aspnetcore-6.0**](https://learn.microsoft.com/en-us/aspnet/core/web-api/advanced/formatting?view=aspnetcore-6.0)

# Web API Request/Response Data Formats

https://www.tutorialsteacher.com/webapi/request-response-data-formats-in-web-api

**How Web API handles different formats of request and response data?**

Media type (aka MIME type) specifies the format of the data as type/subtype e.g. text/html, text/xml, application/json, image/jpeg etc.

In HTTP request, MIME type is specified in the request header using **Accept** and **Content-Type** attribute. The Accept header attribute specifies the format of response data which the client expects and the Content-Type header attribute specifies the format of the data in the request body so that receiver can parse it into appropriate format.

For example, if a client wants response data in JSON format then it will send following GET HTTP request with Accept header to the Web API.

HTTP GET Request:

GET http://localhost:60464/api/student HTTP/1.1

User-Agent: Fiddler

Host: localhost:1234

**Accept: application/json**

The same way, if a client includes JSON data in the request body to send it to the receiver then it will send following POST HTTP request with Content-Type header with JSON data in the body.

HTTP POST Request:

POST http://localhost:60464/api/student?age=15 HTTP/1.1

User-Agent: Fiddler

Host: localhost:60464

**Content-Type: application/json**

Content-Length: 13

{

id:1,

name:'Steve'

}

# ASP.NET Web API: Media-Type Formatters

[**https://www.tutorialsteacher.com/webapi/web-api-formatters**](https://www.tutorialsteacher.com/webapi/web-api-formatters)

**What are media type formatters?**

Media type formatters are classes responsible for serializing request/response data so that Web API can understand the request data format and send data in the format which client expects.

**What are the different built-in media type formatters?**

# Custom formatters in ASP.NET Core Web API

[**https://learn.microsoft.com/en-us/aspnet/core/web-api/advanced/custom-formatters?view=aspnetcore-6.0**](https://learn.microsoft.com/en-us/aspnet/core/web-api/advanced/custom-formatters?view=aspnetcore-6.0)

## When to use a custom formatter?

## Use a custom formatter to add support for a content type that isn't handled by the built-in formatters.

## How to create a custom formatter?

To create a custom formatter:

* For serializing data sent to the client, create an output formatter class.
* For deserializing data received from the client, create an input formatter class.
* Add instances of formatter classes to the [InputFormatters](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.mvcoptions.inputformatters) and [OutputFormatters](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.mvcoptions.outputformatters) collections in [MvcOptions](https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.mvcoptions).

# Model Validation in ASP.NET Web API

[**https://learn.microsoft.com/en-us/aspnet/web-api/overview/formats-and-model-binding/model-validation-in-aspnet-web-api**](https://learn.microsoft.com/en-us/aspnet/web-api/overview/formats-and-model-binding/model-validation-in-aspnet-web-api)

# Parameter Binding in ASP.NET Web API

[**https://learn.microsoft.com/en-us/aspnet/web-api/overview/formats-and-model-binding/parameter-binding-in-aspnet-web-api**](https://learn.microsoft.com/en-us/aspnet/web-api/overview/formats-and-model-binding/parameter-binding-in-aspnet-web-api)

# ASP.NET Core Middleware

[**https://learn.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?source=recommendations&view=aspnetcore-6.0**](https://learn.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?source=recommendations&view=aspnetcore-6.0)

* Custom middleware(Global Exception handling) - > IapplicationBuilder and extensions method
* Order of middleware
* What is the purpose of Middleware? Instead of Middleware can I have private method and have all the functionality there?
* Why do you need Middleware? without middleware can we run the application?
* Middleware execiutes on very request , if authentication middleware is there, how we can skip on every request ?
* Terminal middleware, mapWhen, useWhen

# Options pattern in ASP.NET Core

[**https://learn.microsoft.com/en-us/aspnet/core/fundamentals/configuration/options?view=aspnetcore-6.0**](https://learn.microsoft.com/en-us/aspnet/core/fundamentals/configuration/options?view=aspnetcore-6.0)

# Use multiple environments in ASP.NET Core

[**https://learn.microsoft.com/en-us/aspnet/core/fundamentals/environments?view=aspnetcore-6.0**](https://learn.microsoft.com/en-us/aspnet/core/fundamentals/environments?view=aspnetcore-6.0)

**Want to send a mail in code, but in development and production mail ids are different. How will you handle?**

# Logging in .NET Core and ASP.NET Core

[**https://learn.microsoft.com/en-us/aspnet/core/fundamentals/logging/?view=aspnetcore-6.0**](https://learn.microsoft.com/en-us/aspnet/core/fundamentals/logging/?view=aspnetcore-6.0)

# Routing in ASP.NET Core

[**https://learn.microsoft.com/en-us/aspnet/core/fundamentals/routing?view=aspnetcore-6.0**](https://learn.microsoft.com/en-us/aspnet/core/fundamentals/routing?view=aspnetcore-6.0)

# Error Handling in ASP.NET Web API

<https://learn.microsoft.com/en-us/aspnet/web-api/overview/error-handling/>

<https://learn.microsoft.com/en-us/aspnet/core/web-api/handle-errors?view=aspnetcore-6.0>

exception handling in web api?

# ASP.NET Core web API documentation with Swagger / OpenAPI

<https://learn.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-6.0>

# Filters in ASP.NET Core

<https://learn.microsoft.com/en-us/aspnet/core/mvc/controllers/filters?source=recommendations&view=aspnetcore-6.0>

# Overview of ASP.NET Core authentication

<https://learn.microsoft.com/en-us/aspnet/core/security/authentication/?view=aspnetcore-6.0>

**Difference between authentication and autherization?**

**2. Different types of AccessToken? Jwt Token? Structure of Token.**

**16.Difference between AccessToken and ResreshToken**

**22. what is difference between OAuth, OAuth2 and OpenId?**

**23.What are different Claims?**

**14. what is difference between OAuth and OpenId?**

**What are different Claims?**

https://curity.io/resources/learn/what-are-claims-and-how-they-are-used/

**15. RefreshToken and AccessToken**

https://www.oclc.org/developer/news/2013/authentication-and-authorization-refresh-tokens.en.html

https://stackoverflow.com/questions/3487991/why-does-oauth-v2-have-both-access-and-refresh-tokens

**13. Refresh Token :**

[**https://auth0.com/blog/refresh-tokens-what-are-they-and-when-to-use-them/**](https://auth0.com/blog/refresh-tokens-what-are-they-and-when-to-use-them/)

**2 factor authentication**

**Web SSO**

**Secrets management**

**How to avoid tokens to appear newtrok tab? is it secure?**

**jwt middeleware - what is the use of it? Authorize attribute?**

**token is saved in local storage? any other way?**

**Auth guard, routing, http interceptor?**

**user clicks on login button, authService.login () -> http interceptor transform outgoing request before passing to next interceptor,**

**it is used to add the add the jwt token if token available and logged in-> web api sends back token in response, client can save in local storage**

**-> any request after that should send the bearer token in header authorization field.**

**What are claims? uses of claims?**

**custom middleware - jwtmiddleware - uses of it**