

[Become a member](#)[Login](#)[Post ▼](#)[Ask Question](#)

ASP.NET Core Web API 5.0 Authentication Using JWT(JSON BASE TOKEN)



Kirtesh Shah



Updated date Jun 30, 2021



88.3k



21



19



[Download Free .NET & JAVA Files API](#)

[Try Free File Format APIs for Word/Excel/PDF](#)

Introduction

Authentication plays a very important role to keep an organization's data/network etc secure. There are many authentication approaches available in the market, but the most popular approach is "Token Based Authentication".

In this article, we are going to discuss and implement Token Based Authentication with Asp.net Core Web API 5.0 + JWT (JSON Web Token).

We will create a simple Web API to understand JWT. We can use postman or swagger to test our Web API once it's done.

I recommend to read this [article](#) first if you are not aware of JWT and then come back for implementation.



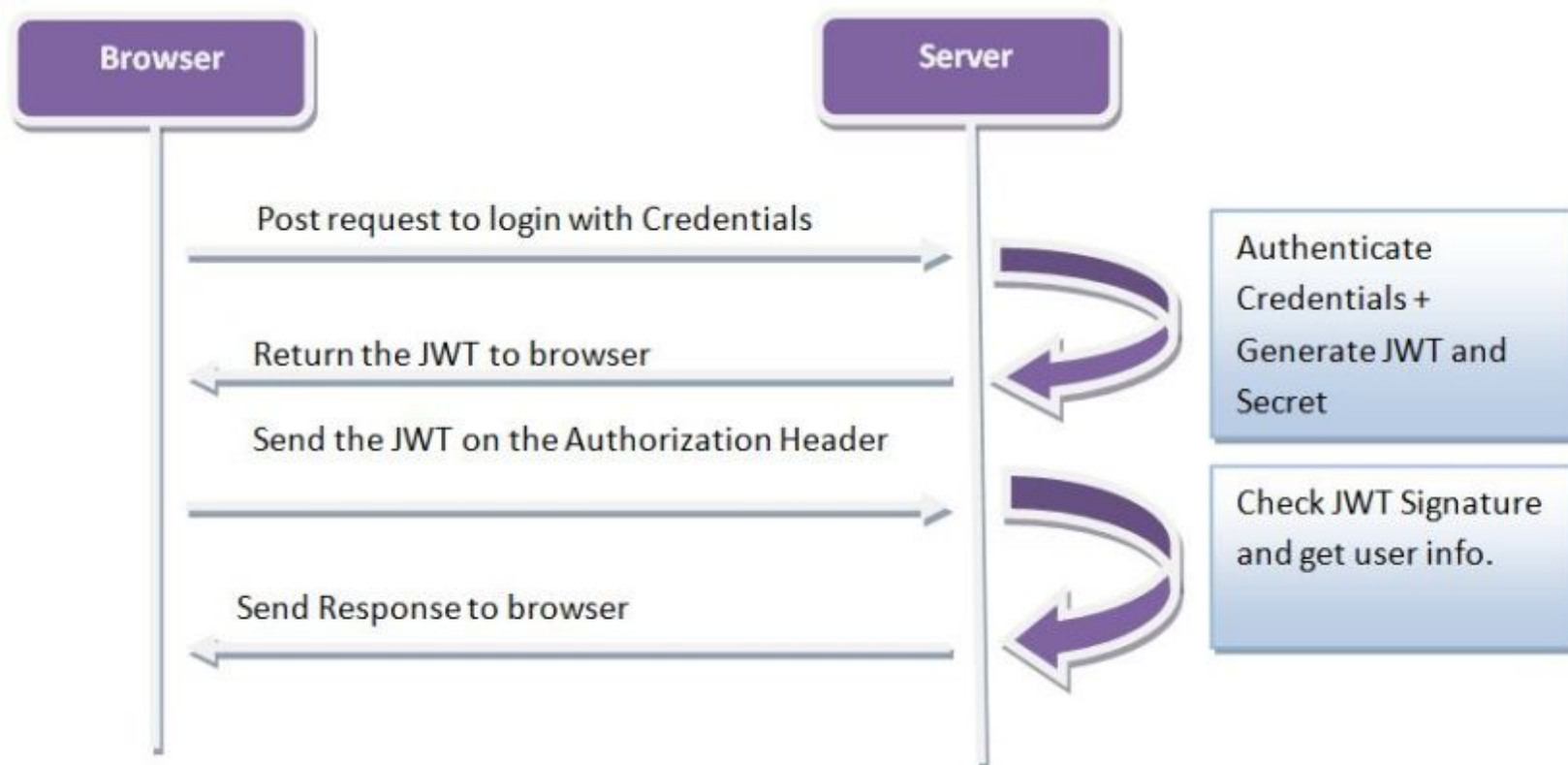
Before we start, we should know what Authentication is.

In simple words we can say, Authentication is validating the user with credentials or identity.

Now next question comes in the mind is “What is Token based authentication in Web API”?

Token Base Authentication processes,

1. The client sent a request to the server with credentials.
2. The server validates the credential and creates an Access token and sends it back to the client.
3. All subsequence requests content this token until its expired.



Steps to follow to create JWT Authentication in Web API

Web API Project has the below endpoints,

1. /authenticate – Authenticate Member Credential and send Access token for subsequence request.
2. /All members – Return List of Members.
3. / MemberByid /id - Members filter by id and return a single member.




Let us create the project using Visual Studio 2019.

Step 1

Create a new project.



Recent project templates

 Class library	C#
 ASP.NET Core Web API	C#
 ASP.NET Core with React.js	C#
 Console Application	C#
 ASP.NET Core Web App	C#

All languages

All platforms

All project types



An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

C# Linux macOS Windows Cloud Service Web



ASP.NET Core Empty

An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

F# Linux macOS Windows Cloud Service Web



ASP.NET Core Web API

A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

C# Linux macOS Windows Cloud Service Web



ASP.NET Core Web API

A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

F# Linux macOS Windows Cloud Service Web



Worker Service

An empty project template for creating a worker service.

Back

Next

Step 2

Select the "Asp.Net Core Web API" template and click on the Next button.



ASP.NET Core Web API

[C#](#)[Linux](#)[macOS](#)[Windows](#)[Cloud](#)[Service](#)[Web](#)

Project name

Location

Solution name 

☐ Place solution and project in the same directory

[Back](#)[Next](#)

Step 3

Configure Project Name, location as per the above screen. Click on the Next button.



ASP.NET Core Web API

[C#](#)[Linux](#)[macOS](#)[Windows](#)[Cloud](#)[Service](#)[Web](#)

Target Framework

.NET 5.0 (Current)

Authentication Type

None

☒ Configure for HTTPS

☐ Enable Docker

Docker OS

Linux

☒ Enable OpenAPI support

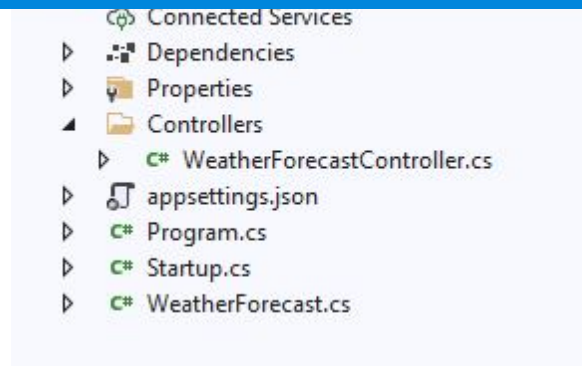
Back

Create

Step 4

Provide Target Framework (.Net 5.0) and click on create button to create a Web API project.



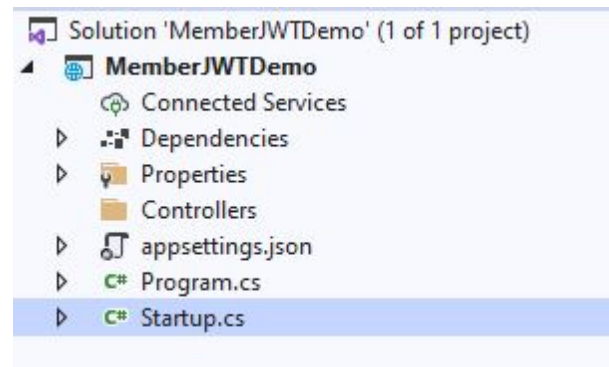


"MemberJWTDemo" Project has been created successfully, and default solution should look like the above image.

The initial Project setup is ready, now follow the below steps to implement JWT Authentication.

Step 5

Remove default controller "WeatherForecastController.cs" and "WeatherForecast.cs" files from the project.



Step 6

We need to Enable Authentication in middleware to validate members. To do so, open the Startup.cs file and add the below code,



```
3 using Microsoft.AspNetCore.HttpsPolicy;
4 using Microsoft.AspNetCore.Mvc;
5 using Microsoft.Extensions.Configuration;
6 using Microsoft.Extensions.DependencyInjection;
7 using Microsoft.Extensions.Hosting;
8 using Microsoft.Extensions.Logging;
9 using Microsoft.OpenApi.Models;
10 using System;
11 using System.Collections.Generic;
12 using System.Linq;
13 using System.Threading.Tasks;
14
15 namespace MemberJWTDemo
16 {
17     public class Startup
18     {
19         public Startup(IConfiguration configuration)
20         {
21             Configuration = configuration;
22         }
23
24         public IConfiguration Configuration { get; }
25
26         // This method gets called by the runtime. Use this method to add services to the container.
27         public void ConfigureServices(IServiceCollection services)
28         {
29
30             services.AddControllers();
```




```
33         c.SwaggerDoc("v1", new OpenApiInfo { Title = "MemberJWTDemo", Version = "v1" });
34     });
35 }
36
37 // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
38 public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
39 {
40     if (env.IsDevelopment())
41     {
42         app.UseDeveloperExceptionPage();
43         app.UseSwagger();
44         app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "MemberJWTDemo v1"));
45     }
46
47     app.UseHttpsRedirection();
48
49     app.UseRouting();
50
51     app.UseAuthentication();
52
53     app.UseAuthorization();
54
55     app.UseEndpoints(endpoints =>
56     {
57         endpoints.MapControllers();
58     });
59 }
60
```

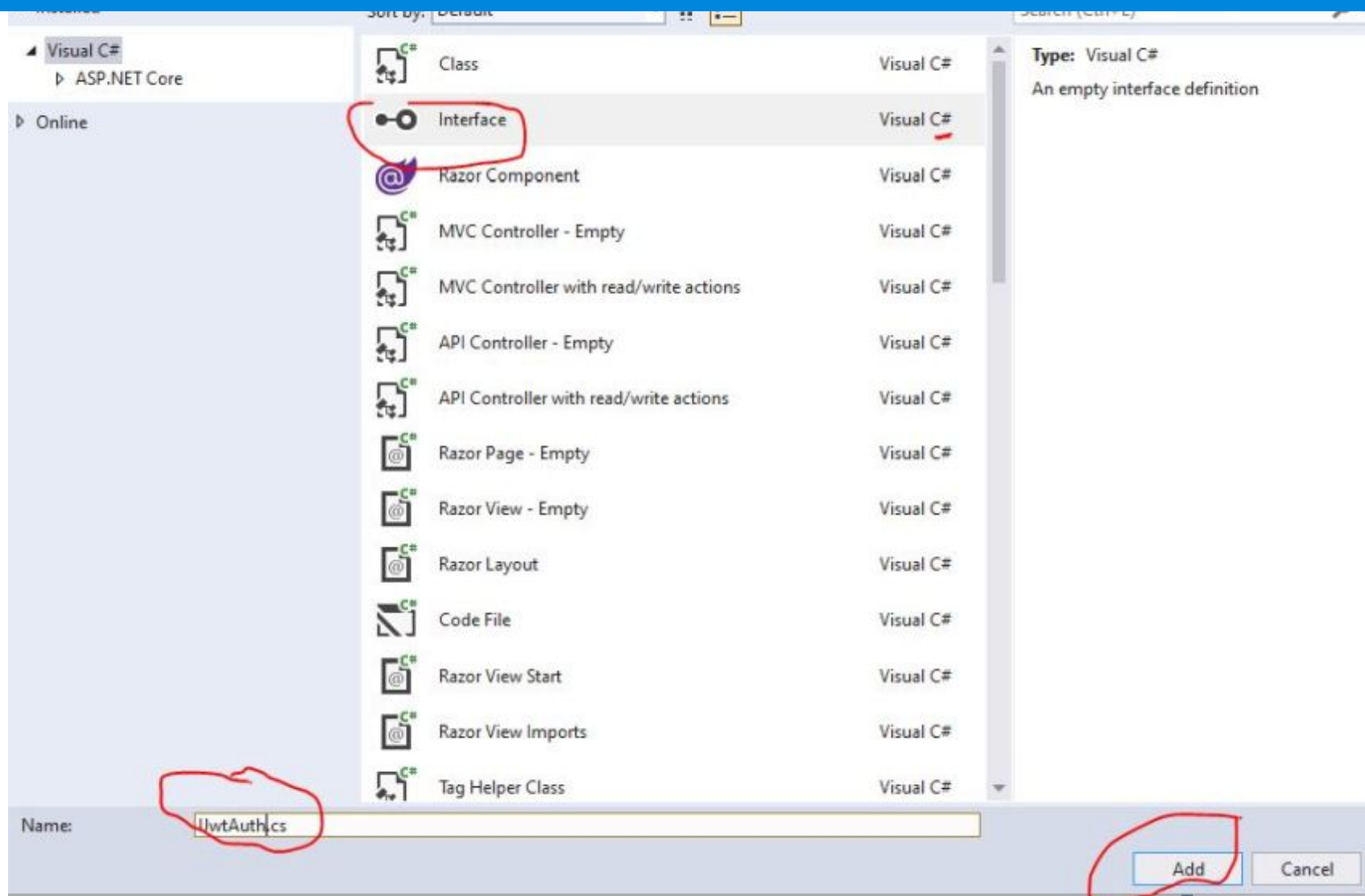


To validate member's credentials and generate JWT tokens, we need a custom authentication class.

Step 7

First, we will create an interface called the "IJwtAuth.cs" file,





```
1 namespace MemberJWTDemo
2 {
3     public interface IJwtAuth
4     {
```



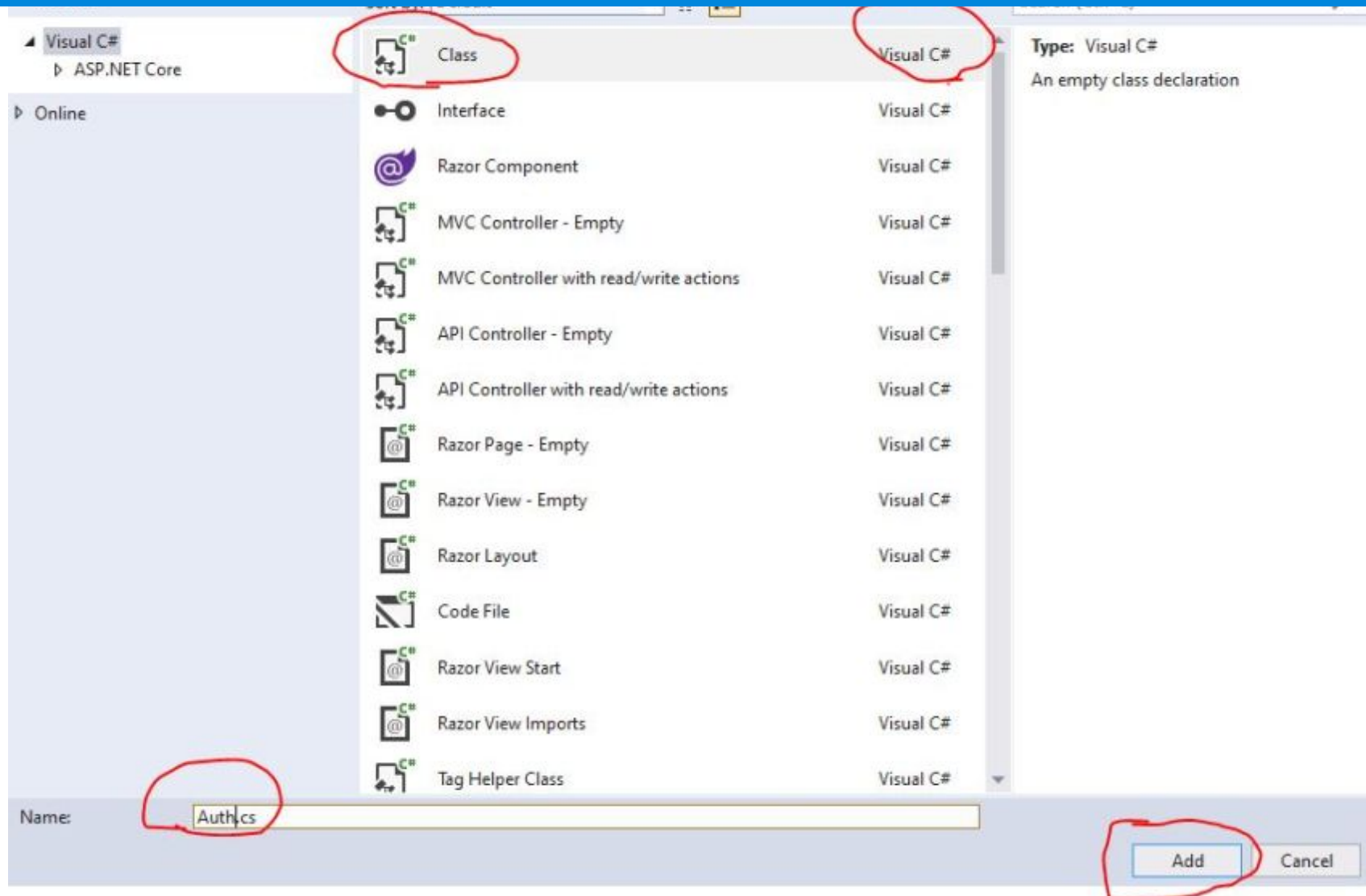
7

}

Step 8

Now add a new class called "Auth" and Implement the "IAuth" interface.

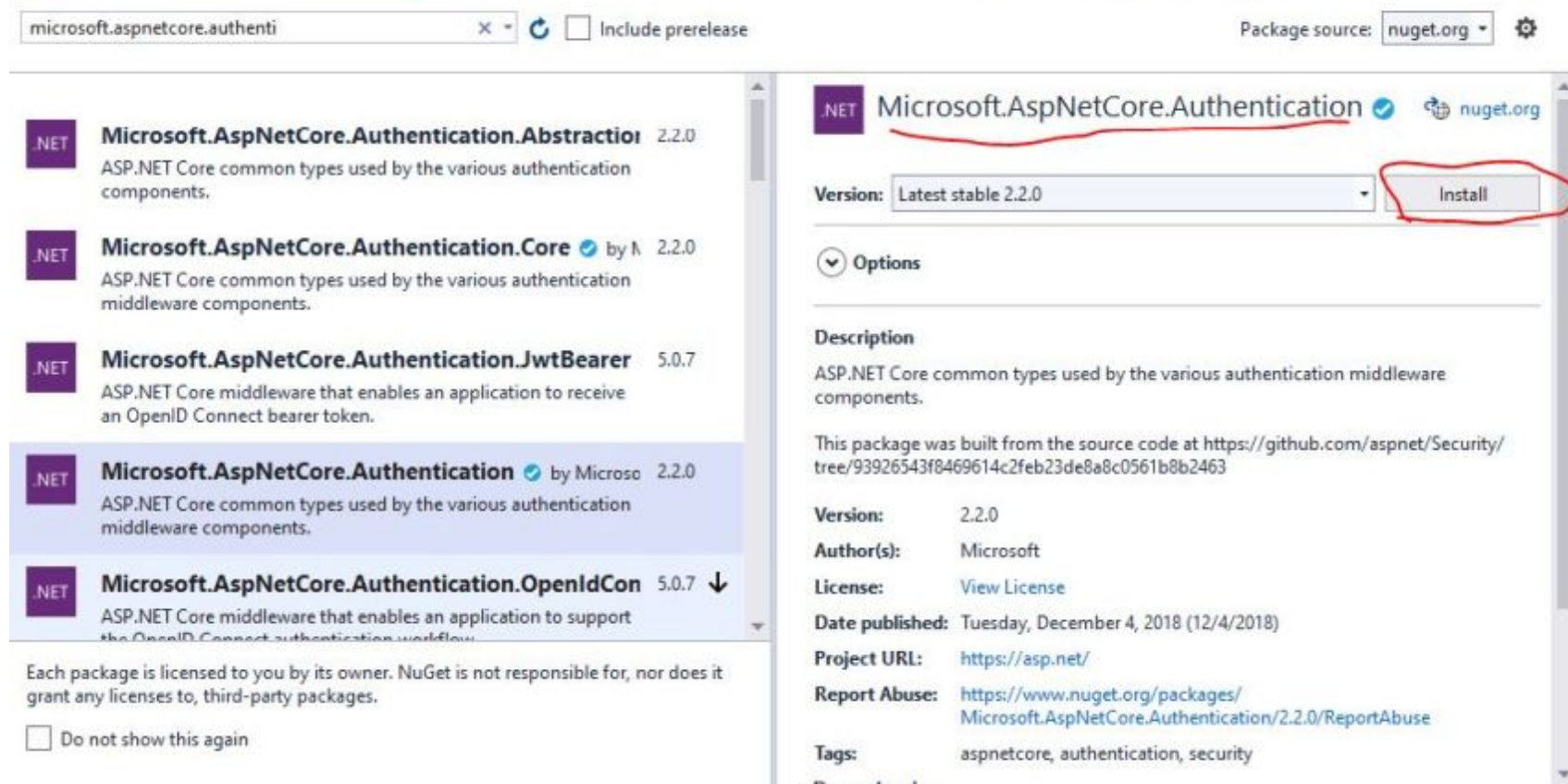




Step 9

We need to add Nuget package "Microsoft.AspNetCore.Authentication" for Authentication.



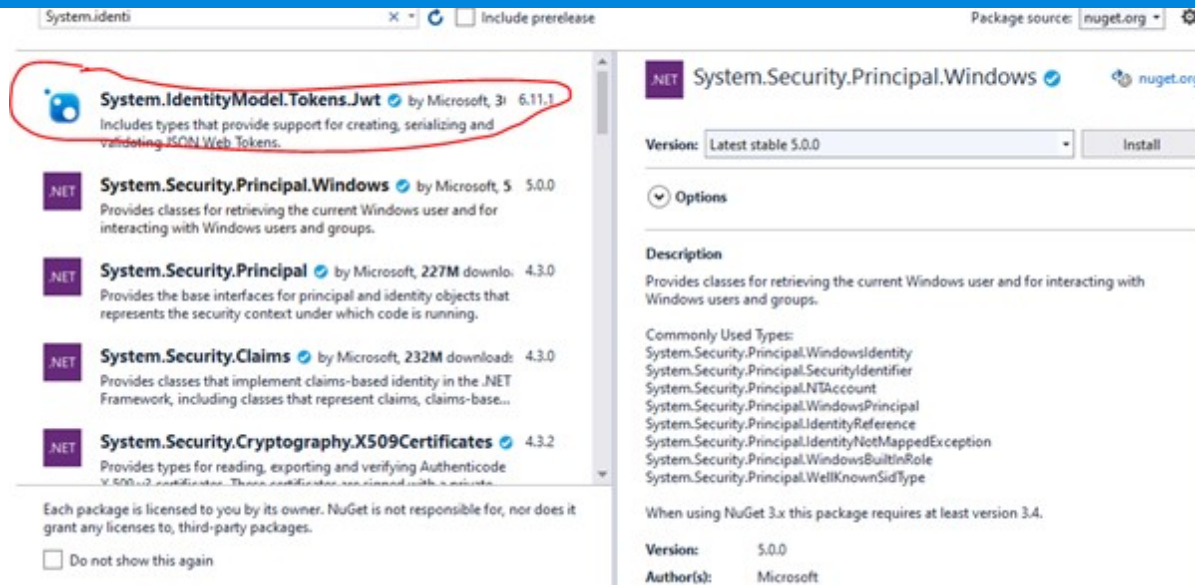


We will keep the username and password hardcoded for demo purposes. In the real-world scenario, it will come from a database. If the user is authenticated successfully then we will create a JWT token.

Step 10

To create JWT Token we need to install Nuget package "System.IdentityModel.Tokens.Jwt".





Step 11

Now we will write the below code in Auth class - Authentication method to create token after authenticating.

```

1  using Microsoft.IdentityModel.Tokens;
2  using System;
3  using System.IdentityModel.Tokens.Jwt;
4  using System.Text;
5  using System.Security.Claims;
6
7  namespace MemberJWTDemo
8  {
9      public class Auth : IJwtAuth
10     {
11         private readonly string username = "kirtesh";

```



```
14     public Auth(string key)
15     {
16         this.key = key;
17     }
18     public string Authentication(string username, string password)
19     {
20         if (!(username.Equals(username) || password.Equals(password)))
21         {
22             return null;
23         }
24
25         // 1. Create Security Token Handler
26         var tokenHandler = new JwtSecurityTokenHandler();
27
28         // 2. Create Private Key to Encrypted
29         var tokenKey = Encoding.ASCII.GetBytes(key);
30
31         //3. Create JETdescriptor
32         var tokenDescriptor = new SecurityTokenDescriptor()
33         {
34             Subject = new ClaimsIdentity(
35                 new Claim[]
36                 {
37                     new Claim(ClaimTypes.Name, username)
38                 }
39             ),
40             Expires = DateTime.UtcNow.AddHours(1),
41             SigningCredentials = new SigningCredentials(
42                 new SymmetricSecurityKey(tokenKey), SecurityAlgorithms.HmacSha256Signature)
```




```
44         var token = tokenHandler.CreateToken(tokenDescriptor);
45
46         // 5. Return Token from method
47         return tokenHandler.WriteToken(token);
48     }
49 }
50 }
```

In the above code,

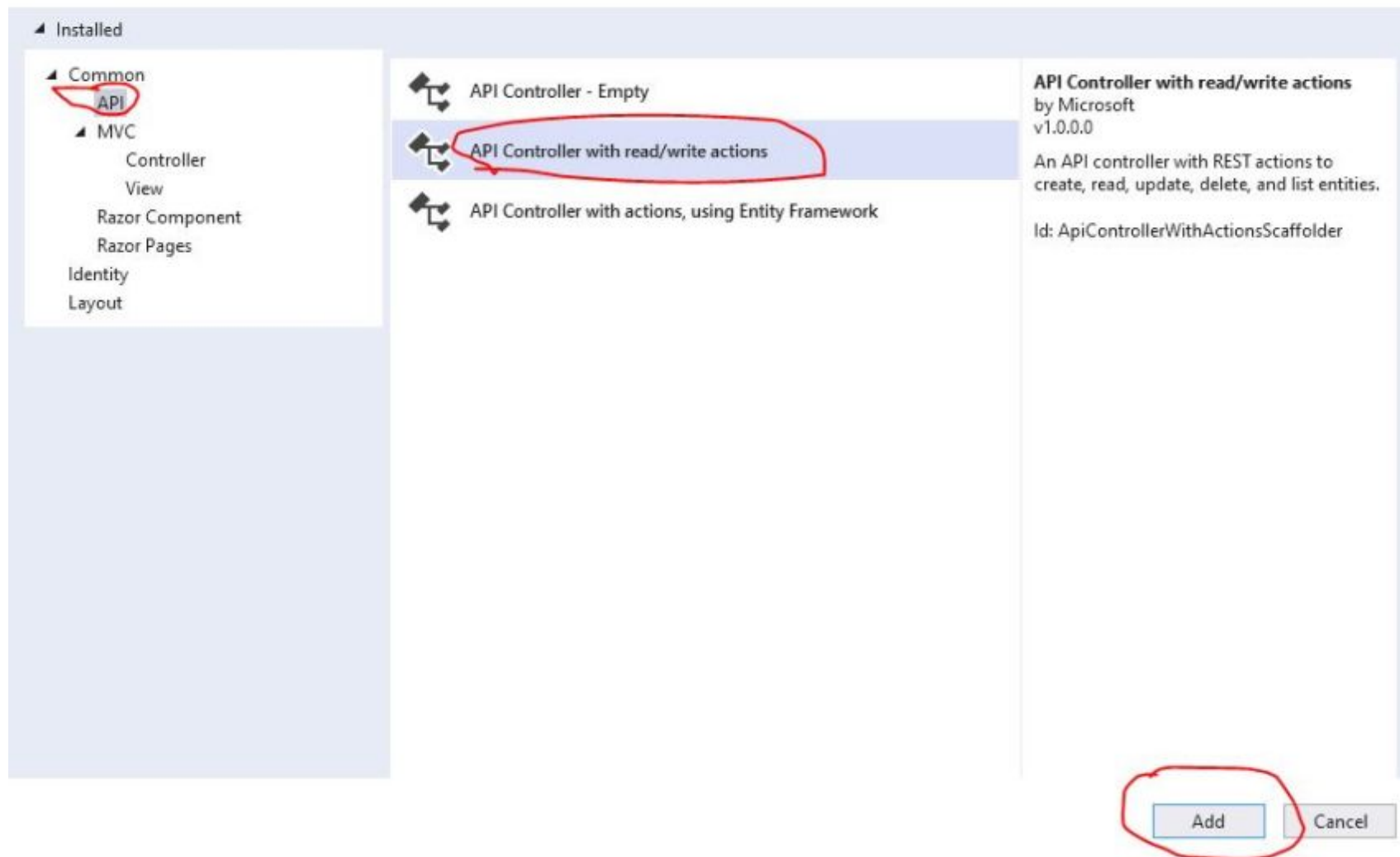
- Create Security Handler – “token handler”.
- Once Token Handler is created, we will encrypt the key into bytes.
- Now we will create a token descriptor.
 - **Subject** – New Claim identity
 - **Expired** – When it will be expired.
 - **SigningCredentical** – Private key + Algorithm
- Now we will create a token using the “create token” method of the token handler.
- Return token from Authentication method.

Now we will create a controller and use this authentication method.

Step 12

Create New Controller “Members” with HTTP POST endpoint called Authentication.

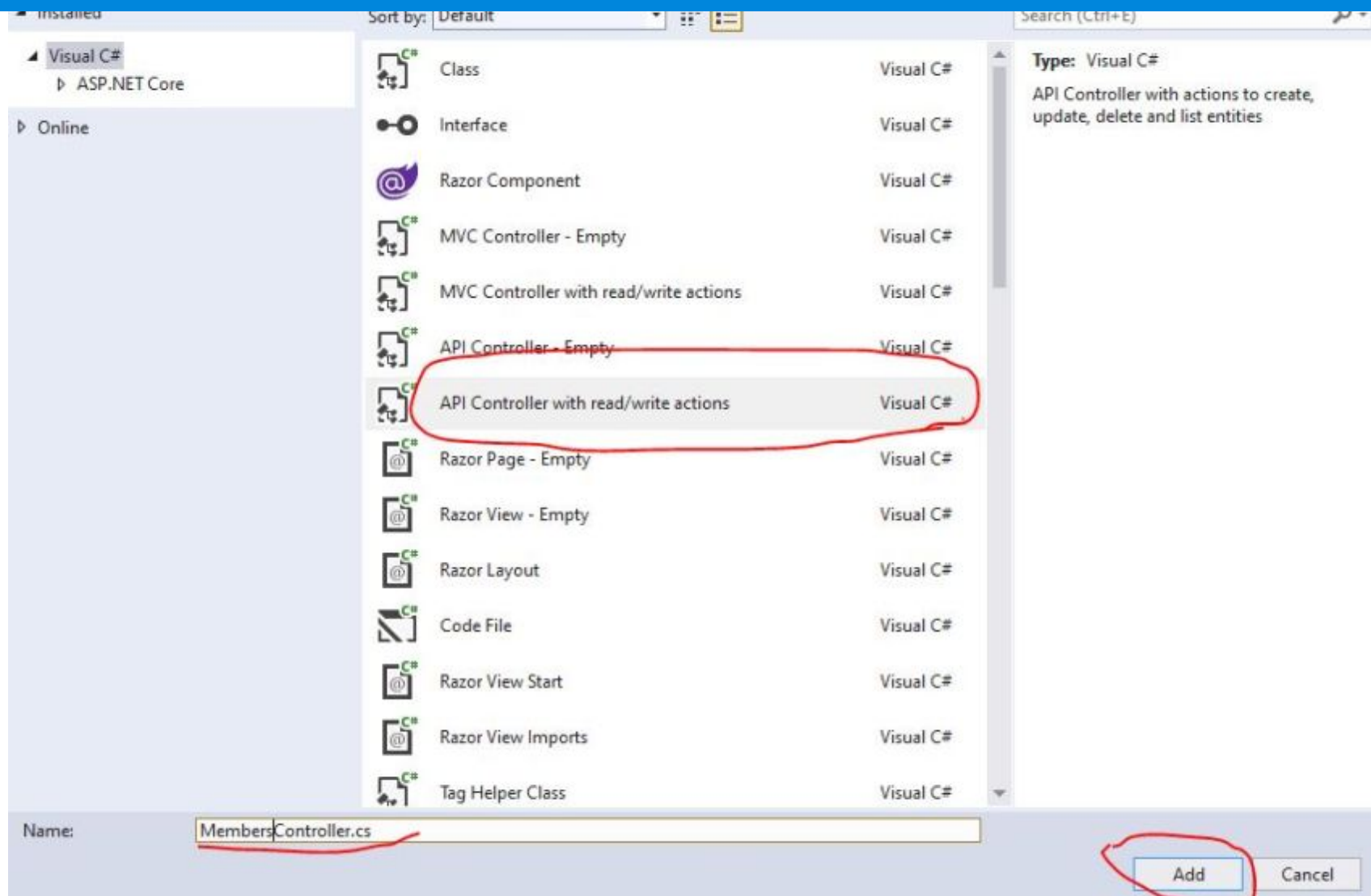




Step 13

Click on Add and Provide controller name in the below screen,





```
1 using Microsoft.AspNetCore.Authorization;  
2 using Microsoft.AspNetCore.Mvc;  
3 using System.Collections.Generic;  
4
```



```
7 namespace MemberJWTDemo.Controllers
8 {
9     [Authorize]
10    [Route("api/[controller]")]
11    [ApiController]
12    public class MembersController : ControllerBase
13    {
14        private readonly IJwtAuth jwtAuth;
15
16        private readonly List<Member> lstMember = new List<Member>()
17        {
18            new Member{Id=1, Name="Kirtesh" },
19            new Member {Id=2, Name="Nitya" },
20            new Member{Id=3, Name="pankaj"}
21        };
22        public MembersController(IJwtAuth jwtAuth)
23        {
24            this.jwtAuth = jwtAuth;
25        }
26        // GET: api/<MembersController>
27        [HttpGet]
28        public IEnumerable<Member> AllMembers()
29        {
30            return lstMember;
31        }
32
33        // GET api/<MembersController>/5
34        [HttpGet("{id}")]
```



```

37         return lstMember.Find(x => x.Id == id);
38     }
39
40     [AllowAnonymous]
41     // POST api/<MembersController>
42     [HttpPost("authentication")]
43     public IActionResult Authentication([FromBody]UserCredential userCredential)
44     {
45         var token = jwtAuth.Authentication(userCredential.UserName, userCredential.Password);
46         if (token == null)
47             return Unauthorized();
48         return Ok(token);
49     }
50
51
52     }
53 }

```

In the above code,

1. The authentication method took the user name and password from the body.
2. Pass credential to the jwtAuth. Authentication method to get token.
3. Return token.
4. Add attributes [AllowAnonymous] as this method can be handled by any user.
5. Add [Authorize] attributes to Member controller.
6. Add "jwtAuth" in the constructor.

```
1 namespace MemberJWTDemo.Controllers
2 {
3     public class UserCredential
4     {
5         public string UserName { get; set; }
6         public string Password { get; set; }
7     }
8 }
```

Step 15

Create Member model class as below,

```
1 namespace MemberJWTDemo
2 {
3     public class Member
4     {
5         public int Id { get; set; }
6         public string Name { get; set; }
7     }
8 }
```

We need to add dependency in the Startup.cs file. Also, we will add JETBearer to decrypt the key.

Step 16

Install "Microsoft.AspNetCore.Authentication.JwtBearer" NuGet package.

Step 17

Add the below code in the startup.cs file,



```
3 using Microsoft.AspNetCore.Hosting;
4 using Microsoft.Extensions.Configuration;
5 using Microsoft.Extensions.DependencyInjection;
6 using Microsoft.Extensions.Hosting;
7 using Microsoft.IdentityModel.Tokens;
8 using Microsoft.OpenApi.Models;
9 using System.Text;
10
11 namespace MemberJWTDemo
12 {
13     public class Startup
14     {
15         public Startup(IConfiguration configuration)
16         {
17             Configuration = configuration;
18         }
19
20         public IConfiguration Configuration { get; }
21
22         // This method gets called by the runtime. Use this method to add services to the container.
23         public void ConfigureServices(IServiceCollection services)
24         {
25
26             services.AddControllers();
27             var key = "This is my first Test Key";
28             services.AddAuthentication(x =>
29             {
30                 x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
```



```

33     {
34         x.RequireHttpsMetadata = false;
35         x.SaveToken = true;
36         x.TokenValidationParameters = new Microsoft.IdentityModel.Tokens.TokenValidationParameters
37         {
38             ValidateIssuerSigningKey = true,
39             ValidateIssuer = false,
40             ValidateAudience = false,
41             IssuerSigningKey = new SymmetricSecurityKey(Encoding.ASCII.GetBytes(key))
42         };
43     });
44
45     services.AddSingleton<IJwtAuth>(new Auth(key));
46     services.AddSwaggerGen(c =>
47     {
48         c.SwaggerDoc("v1", new OpenApiInfo { Title = "MemberJWTDemo", Version = "v1" });
49     })
50 }
51
52 // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
53 public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
54 {
55     if (env.IsDevelopment())
56     {
57         app.UseDeveloperExceptionPage();
58         app.UseSwagger();
59         app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "MemberJWTDemo v1"))
60     }

```



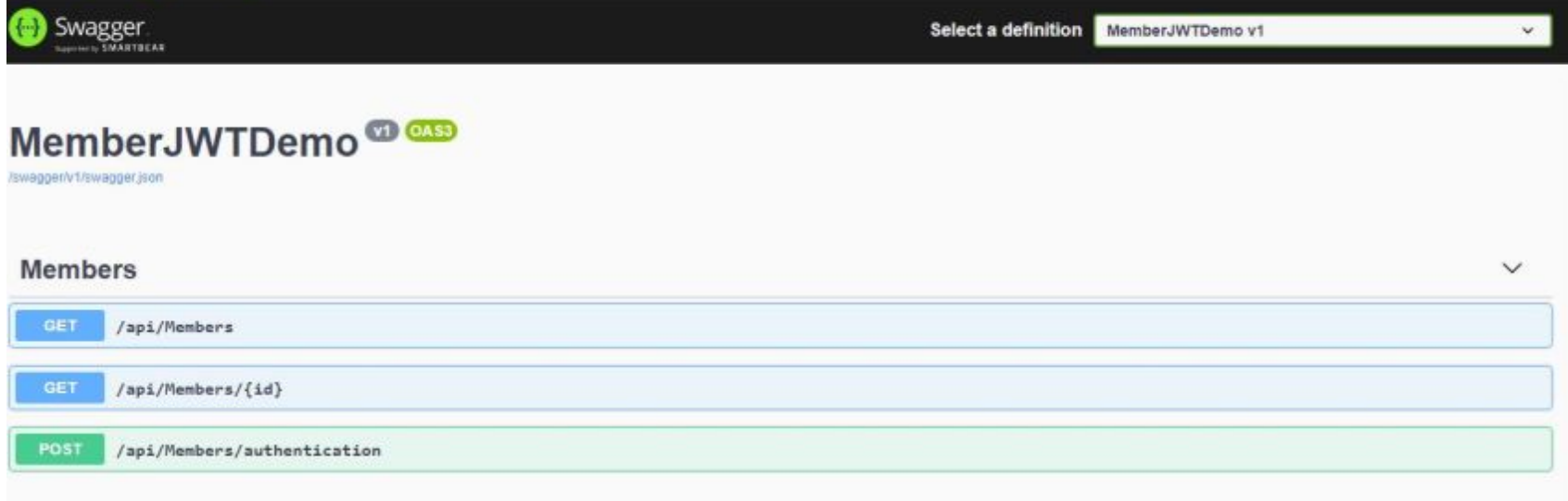

```
63
64     app.UseRouting();
65
66     app.UseAuthentication();
67
68     app.UseAuthorization();
69
70     app.UseEndpoints(endpoints =>
71     {
72         endpoints.MapControllers();
73     });
74 }
75 }
76 }
```

We are done with code.

Step 18

Now we will call API endpoint using Postman or swagger etc.





Step 19

First, we will call post method /API/Members/authentication,





I hope you enjoy this article and find it useful.

.Net core authentication

Asp.net core Authentication

Asp.net Core Web API 5.0 Authentication using JWTJSON BASE TOKEN

JSON Base token

JWT

JWT Authentication

Token Base authentication

Next Recommended Reading

[JWT Authentication In ASP.NET Core](#)

OUR BOOKS



Kirtesh Shah *TOP 500*





171 724.9k 2

[View Previous Comments](#)

19 21



Type your comment here and press Enter Key (Minimum 10 characters)



Hello i can't install Microsoft.AspNetCore.Authentication.JwtBearer => error (Microsoft.AspNetCore.Authentication.JwtBearer 6.0.1 is not compatible with netstandard2.0 (.NETStandard,Version=v2.0). Package Microsoft.AspNetCore.Authentication.JwtBearer 6.0.1 supports: net6.0)

[Triết Nguyễn](#)

2099 5 0

Jan 21, 2022

1 1 Reply



Hello, The JWT middleware is tied to ASP.NET Core, so it does not support .NET Standard 2.0 (since ASP.NET Core does not support .NET Standard 2.0). You can still reference it from a class library if your class library targets .NET Core 3.0 or higher.

[Kirtesh Shah](#)

171 13.5k 724.9k

Jan 21, 2022

0



When I execute get member request It doesn't ask for JWT Token to use Authorize Filter?? How do I ask user on Swagger api for JWT Token??

[Ashfaq Ali](#)

2102 2 0

Nov 13, 2021

0 0 Reply



Thanks for the guide,How do i apply "role" to token? Such as admin or manager or client, etc.. ?

[Moshe Azraf](#)

2102 2 0

Oct 30, 2021

0 0 Reply



I'm able to generate token but Why I'm getting unauthorized error even I'm passing authorized:"value" in header ? I'm using postman to consume the API

[piyush pallaw](#)





Hey did you find any solution for this. I'm also stuck with same issue. Please help Pls text 9888111795

Raj Aryan

2101 3 0

Feb 05, 2022



Did you got soln for above issue

ananth ganapathy

2039 65 23.3k

Feb 25, 2022



Make sure you are using the "Bearer Token" settings in the "Auth" tab of Postman. Also, note that the token is only good for 1 hour - see the expiration value in the Auth class.

brendan daunt

1982 122 0

Feb 27, 2022



If we are using the custom authentication then what is the use of defaultapp.UseAuthentication(); app.UseAuthorization(); in startup.cs class and is there any conflict occur internally?

ANIYAN C

1905 199 0

Sep 24, 2021



Sir please make a blog on facebook n google authentication in same above application

ishwar giri

1392 730 0

Sep 08, 2021



Sure. I will create one

Kirtesh Shah

171 13.5k 724.9k

Sep 09, 2021



Thank you for the tutorial. I was debugging and looking at line 20 in the Auth.cs. "if (!(username.Equals(username) || password.Equals(password)))". Won't this always be true?

Chuck Frank

2095 9 0

Aug 25, 2021



Yes true - has there been any solution to this ?





@Kritesh Shah - please guide

Arun K

2095 9 0

Oct 22, 2021



You just have to either change the spelling of the private "username" and "password" properties or change the spelling of the parameters. Because they are currently both spelled the same so the comparison is not valid.

brendan daunt

1982 122 0

Feb 27, 2022



Thanks Vilas

Kirtesh Shah

171 13.5k 724.9k

Aug 25, 2021



0 0 Reply



Nice Article Kirtesh

Vilas Sagar

1728 376 3.1k

Aug 25, 2021



0 0 Reply



Thanks Pranam

Kirtesh Shah

171 13.5k 724.9k

Jun 30, 2021



0 0 Reply



Embed Legal Contract in .NET

BoldSign

Open



FEATURED ARTICLES

SPFx Form Customizer Extension To Customize SharePoint New/Edit/Display Form Of List/Libraries

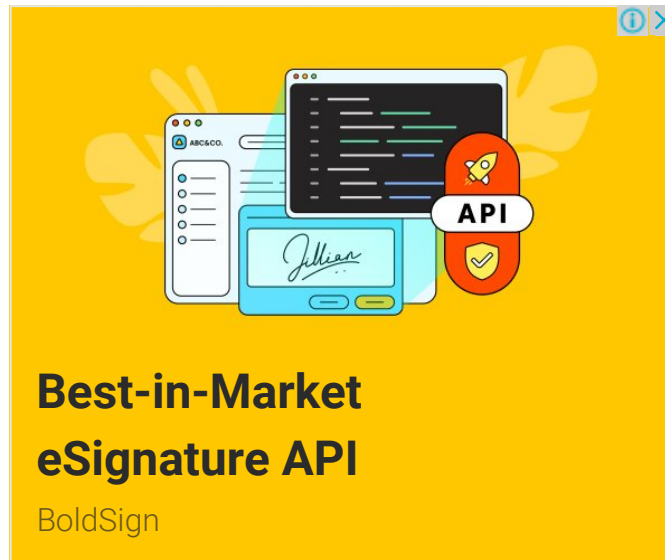
Azure Front Door



Change Data Capture - Another Way To Implement The Incremental Load

Kaggle Machine Learning

[View All](#) ➔



TRENDING UP

- 01 Azure Durable Functions - An Overview
- 02 How To Upload Files Into Azure Blob Storage Using Azure Functions In C#
- 03 Change Data Capture - Another Way To Implement The Incremental Load
- 04 Easily Use Redis Cache In ASP.NET 6.0 Web API
- 05 Understanding About Delegates In C#
- 06 Easily Understand Azure API Management



08 Rockin' The Code World with dotNetDave ft. Swizec Teller Ep. 51

09 Rockin' The Code World with dotNetDave ft. Steve Jones Ep. 53

10 Using Async/Await With Disposable Objects

[View All](#) 



Software for Enterprises

- Simpler team document management
- Supports complex signature workflows
- Configurable user permissions
- Legal compliance with ESIGN and eIDAS
- Single Sign-On (SSO)
- Customize email domain
- Branding



BoldSign
By Syncfusion

[LEARN MORE](#)



[About Us](#) [Contact Us](#) [Privacy Policy](#) [Terms](#) [Media Kit](#) [Sitemap](#) [Report a Bug](#) [FAQ](#) [Partners](#)



