

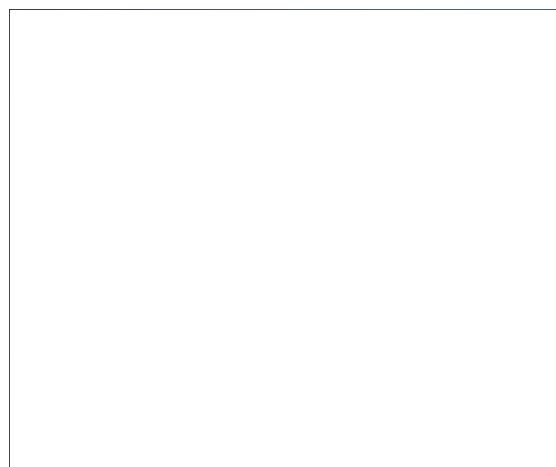
ABOUT ME	CONTACT US	PRIVACY POLICY	TERMS AND CONDITIONS	ADVERTISE / SPONSOR	
----------	------------	----------------	----------------------	---------------------	--

HOME	TUTORIALS	BOOKS	COURSES	CLOUD HOSTING	INTERVIEW QUESTIONS	
------	-----------	-------	---------	---------------	---------------------	--

YOU ARE HERE: [HOME](#) / [ASP.NET CORE](#) / ASP NET CORE 5 WEB API TOKEN BASED AUTHENTICATION EXAMPLE USING JWT

Asp Net Core 5 web API token based authentication example using JWT

NOVEMBER 29, 2020 BY [MEBAKAR1005](#) — [2 COMMENTS](#)



In this tutorial, we are going to cover a **web api token based authentication** example using **JWT** in **Asp.Net Core 5** using visual studio 2019. So, first-of-all, we will create a new Asp.Net Core 5 web API project and then we will see how to implement Microsoft Identity and then finally we will see how to implement token based authentication using JWT in Asp.Net Core 5 web API app.

Get Notifications *

If you have a project in **Asp.Net Core 3.1** and want to go back to the tutorial where we have discussed [here](#)

Prerequisites

There are some prerequisites for this tutorial. So, before going to next you must install these dependencies below.

- **Visual Studio 2019:** Click [here](#) to download it according to your machine and then install it. Make sure you have installed the **visual studio 2019 version 16.8** or later.
- **.NET Core 5:** Click [here](#) to download it according to your machine and install **.NET Core 5**.
- **Postman:** Click [here](#) to download and then install it.
- **SQL Server**

How to implement token based authentication (JWT) in Asp.Net Core 5 web API?

Let's start how to implement a web API token based authentication example using **Asp.Net Core 5**.



Step # 1: How to create Asp Net Core 5 web API using visual studio 2019?

First-of-all, now in this step we will create a new asp.net core 5 web api using visual studio 2019. So, go to **Visual Studio 2019** and then click on the **Create a new project** and then select **Asp.Net Core Web Application** and then enter the name in the **Project name** field and then set **location** of project directory and then click on the **Create** button.

Get Notifications *

Configure your new project

ASP.NET Core Web Application C# Linux macOS Windows Cloud Service Web

Project name

BookStore

Location

C:\Users\██████\Desktop\Dotnet Detail\.net 5\authentication and authorization using .net 5\

...

Solution name i

BookStore

Place solution and project in the same directory

Back

Create

After clicking on the **Create** button, then you will see a new popup as you do see below in the screenshot. Now, select **Asp.Net Core 5.0** from the dropdown and then select **Asp.Net Core Web API** template and then click on the **Create** button.

Create a new ASP.NET Core web application

.NET Core ASP.NET Core 5.0

- ASP.NET Core Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
- 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.
- ASP.NET Core Web App
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.
- ASP.NET Core Web App (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
- ASP.NET Core with Angular
A project template for creating an ASP.NET Core application with Angular
- ASP.NET Core with React.js

Authentication

No Authentication
[Change](#)

Advanced

Configure for HTTPS
 Enable Docker Support
(Requires Docker Desktop)
Linux
 Enable OpenAPI support

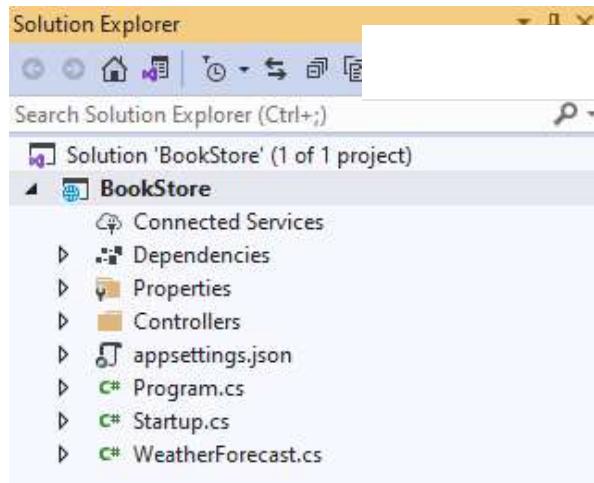
Author: Microsoft
Source: Templates 5.0.0

Get additional project templates

Back Create

After clicking on the **Create** button, then you will see a new **Asp.Net Core 5.0 Web API** project is created with the name of **BookStore**.

Get Notifications *



Step # 2: How to setup Database and implement Microsoft Identity in Asp.Net Core 5 Web API project?

Now, in this step, we will setup a new database in **Asp.Net Core 5.0 web api** project. So, go to the project folder structure and then open **appsettings.json** file and then add a connection string as you do see in the below file code.

```

1  {
2      "ConnectionStrings": {
3          "DefaultConnection": "Data Source=DataSourceName; Initial Catalog=BookStoreDB;Integrated Se
4      },
5      "Logging": {
6          "LogLevel": {
7              "Default": "Information",
8              "Microsoft": "Warning",
9              "Microsoft.Hosting.Lifetime": "Information"
10         }
11     },
12     "AllowedHosts": "*"
13 }
```

connection string - asp.net core 5 Authentication hosted with ❤ by GitHub [view raw](#)

Now, go to the project folder structure and then enter a new folder with the name of **Data**.

Now, we will add some classes related to the databases like **ApplicationUser**, **BookStoreDbContext** and etc.

READ [How to add SignalR to Asp.Net Core 3.0 app using Visual Studio 2019](#)

Note: – don't forget to add libraries.

```

1  using Microsoft.AspNetCore.Identity;
2  using System;
```

Get Notifications *

```

3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Threading.Tasks;
6
7  namespace BookStore.Data
8  {
9      public class ApplicationUser : IdentityUser
10     {
11     }
12 }
```

Application User Class - Asp.Net Core 5 authentication hosted with ❤ by GitHub

[view raw](#)

Now, go to **NuGet Packages** and then search for

Microsoft.AspNetCore.Identity.EntityFrameworkCore and then select it and then install as you do see in the below file.

The screenshot shows the NuGet Package Manager interface. On the left, there's a search bar with 'Microsoft.AspNetCore.Identity.EntityFrameworkCore' typed in, along with a 'Include prerelease' checkbox. Below the search bar, several packages are listed:

- Microsoft.AspNetCore.Identity.EntityFrameworkCore** (by Microsoft, 45. v5.0.0) - Description: ASP.NET Core Identity provider that uses Entity Framework Core.
- Microsoft.AspNetCore.Identity.AspNetCoreCompat** (by Microsoft, 42.3K down) - Description: A compatibility layer for sharing identity databases between Microsoft.AspNetCore.Identity.EntityFrameworkCore and Microsoft.AspNetCore.Identity.EntityFra...
- MongoDB.AspNetCore.Identity** (by Jonathan Sheely, 11K downloads) - Description: ASP.NET Identity provider that users MongoDB for storage
- Redis.AspNetCore.Identity** (by aminjam, 2.29K downloads) - Description: ASP.NET Identity provider that uses Redis for storage and works with VS2013 Default SPA Template
- MongoLab.AspNetCore.Identity** (by MikeDevenney, 2.54K downloads) - Description: ASP.NET Identity provider that users MongoLab for storage

At the bottom left, there are two checkboxes: 'Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.' and 'Do not show this again'.

On the right, the details page for **Microsoft.AspNetCore.Identity.EntityFrameworkCore** is shown. It includes the package version (v5.0.0), a table for selecting projects (with 'BookStore' checked), and buttons for 'Uninstall' and 'Install'. There are also sections for 'Options' and 'Description'.

Now, add another class inside the **Data** folder with the name of **BookStoreDbContext** and then write the code as you do see below in the file.

Get Notifications *

```

1  using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
2  using Microsoft.EntityFrameworkCore;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Threading.Tasks;
7
8  namespace BookStore.Data
9  {
10     public class BookStoreDbContext : IdentityDbContext<ApplicationUser>
11     {
12         public BookStoreDbContext(DbContextOptions<BookStoreDbContext> options) : base(options)
13         {
14         }
15         protected override void OnModelCreating(ModelBuilder builder)
16         {
17             base.OnModelCreating(builder);
18         }
19     }
20 }
```

BookStore Db Context - Asp.Net Core 5 authentication hosted with ❤ by GitHub

[view raw](#)

Note: – don't forget to add libraries.

Now, we will add a new seed class which will add a demo data for the user account. So, right click on the **Data** folder and then click on the **Add** and then click on the **Class** and then enter the name of class (**E.g. SeedDB**) and then write the code as you do see below in the file.

```

1  using Microsoft.AspNetCore.Identity;
2  using Microsoft.Extensions.DependencyInjection;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Threading.Tasks;
7
8  namespace BookStore.Data
9  {
10     public class SeedDB
11     {
12         public static void Initialize(IServiceProvider serviceProvider)
13         {
14             var context = serviceProvider.GetRequiredService<BookStoreDbContext>();
15             var userManager = serviceProvider.GetRequiredService<UserManager< ApplicationUser >>();
16             context.Database.EnsureCreated();
17             if (!context.Users.Any())
18             {
19                 ApplicationUser user = new ApplicationUser()
20                 {
```

Get Notifications *

```

21     Email = "test@gmail.com",
22     UserName = "test",
23     SecurityStamp = Guid.NewGuid().ToString()
24   };
25   userManager.CreateAsync(user, "Test@123");
26 }
27 }
28 }
29 }
```

SeedDB - Asp.Net Core 5 authentication hosted with ❤ by GitHub

[view raw](#)

Now, go to **NuGet Packages** and then search for **Microsoft.EntityFrameworkCore.SqlServer** and then install it as you do see below in the screenshot.

The screenshot shows the NuGet Package Manager interface. On the left, there's a search bar with 'Microsoft.EntityFrameworkCore.SqlServer' typed in, and a dropdown menu showing 'nuget.org'. On the right, the 'Manage Packages for Solution' window is open for the 'BookStore' project. It shows the selected package 'Microsoft.EntityFrameworkCore.SqlServer' v5.0.0. Below it are other packages: Microsoft.EntityFrameworkCore v5.0.0, Microsoft.EntityFrameworkCore.SqlServer.Design v1.1.6, Microsoft.EntityFrameworkCore.SqlServer.NetTopologySuite v5.0.0, and Z.EntityFramework.Extensions.EFCore v5.1.4. At the bottom of the left pane, there's a note about package licensing and a 'Do not show this again' checkbox. The right pane has sections for 'Versions', 'Installed' (set to 'not installed'), 'Version' (set to 'Latest stable 5.0.0'), and 'Options'. The 'Install' button is highlighted with a yellow background.

Now, we will add services to **Startup.cs** class. So, go to the project folder structure and then open the **Startup.cs** class and then write the code as you do see in the below file's **line # 33 to 36** and **line # 52**.

```

1  using BookStore.Data;
2  using Microsoft.AspNetCore.Builder;
3  using Microsoft.AspNetCore.Hosting;
4  using Microsoft.AspNetCore.HttpsPolicy;
5  using Microsoft.AspNetCore.Identity;
6  using Microsoft.AspNetCore.Mvc;
7  using Microsoft.EntityFrameworkCore;
8  using Microsoft.Extensions.Configuration;
9  using Microsoft.Extensions.DependencyInjection;
10 using Microsoft.Extensions.Hosting;
11 using Microsoft.Extensions.Logging;
12 using Microsoft.OpenApi.Models;
```

Get Notifications *

```
13  using System;
14  using System.Collections.Generic;
15  using System.Linq;
16  using System.Threading.Tasks;
17
18  namespace BookStore
19  {
20      public class Startup
21      {
22          public Startup(IConfiguration configuration)
23          {
24              Configuration = configuration;
25          }
26
27          public IConfiguration Configuration { get; }
28
29          // This method gets called by the runtime. Use this method to add services to the container
30          public void ConfigureServices(IServiceCollection services)
31          {
32              services.AddControllers();
33              services.AddDbContext<BookStoreDbContext>(options => options.UseSqlServer(Configuration));
34              services.AddIdentity<ApplicationUser, IdentityRole>()
35                  .AddEntityFrameworkStores<BookStoreDbContext>()
36                  .AddDefaultTokenProviders();
37              services.AddSwaggerGen(c =>
38              {
39                  c.SwaggerDoc("v1", new OpenApiInfo { Title = "BookStore", Version = "v1" });
40              });
41          }
42
43          // This method gets called by the runtime. Use this method to configure the HTTP request pipeline
44          public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
45          {
46              if (env.IsDevelopment())
47              {
48                  app.UseDeveloperExceptionPage();
49                  app.UseSwagger();
50                  app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "BookStore v1"));
51              }
52              SeedDB.Initialize(app.ApplicationServices.GetRequiredService<IServiceScopeFactory>().CreateScope);
53              app.UseHttpsRedirection();
54              app.UseRouting();
55              app.UseAuthorization();
56              app.UseEndpoints(endpoints =>
57              {
58                  endpoints.MapControllers();
59              });
60          }
61      }
```

Get Notifications *

```
62 }
```

Startup.cs for services - asp.net core 5 authentication hosted with ❤ by GitHub

[view raw](#)

Let's understand the above code.

Line # 33: In this line, we are adding the service for SQL server db.

Line # 34 to 36: In this line, we are adding service for Microsoft Identity.

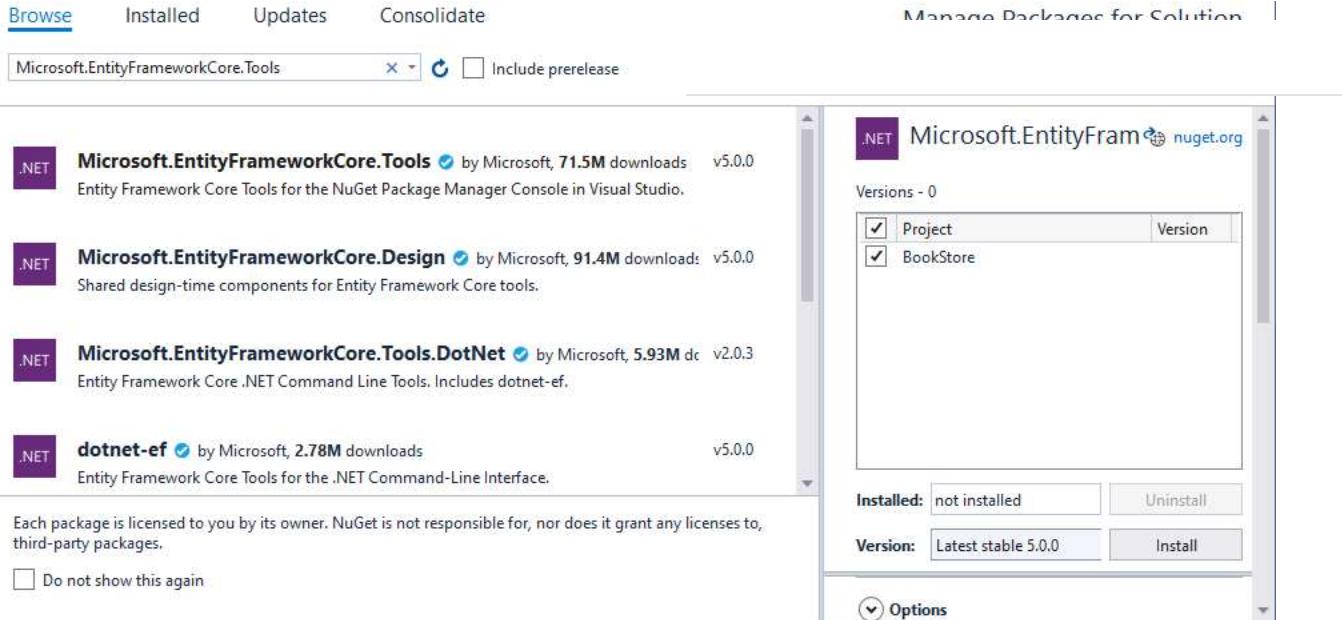
Line # 56: In this line, we are initializing the **SeedDB** class to create a new user.

Add Entity Framework Core Migrations

Now, we will run the migration. So, go to **Tools** and then **NuGet Package Manager** and then go to **Manage NuGet Packages for solutions...** and then search for **Microsoft.EntityFrameworkCore.Tools** and then install it as you do see below in the screenshot.

READ [How to add Authentication to React app using Asp.Net Core 3 in VS2019](#)

Get Notifications *



So, go to **Package Manager Console** and then run the below command and then you will see the migration folder will be created.

`add-migration initial`

```
PM> add-migration initial
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
```

Now, run below command.

`update-database`

```
PM> update-database
Build started...
Build succeeded.
Security Warning: The negotiated TLS 1.0 is an insecure protocol and is supported for backward compatibility only. The recommended protocol version is TLS 1.2 and later.
Security Warning: The negotiated TLS 1.0 is an insecure protocol and is supported for backward compatibility only. The recommended protocol version is TLS 1.2 and later.
Security Warning: The negotiated TLS 1.0 is an insecure protocol and is supported for backward compatibility only. The recommended protocol version is TLS 1.2 and later.
Done.
```

Now, the database and identity implementation part is completed.

Step # 3: How to implement token based authentication using JWT in Asp.Net Core 5 web api?

Now, in this step, we will see how to implement **token based authentication** using **JWT** in **Asp.Net Core 5.0 web API**.

So, go to project folder structure and then add a new folder with the name of **Models**. Then right click on the **Models** folder and then click on the **Add** and then click on the **Get Notifications***

then enter the name of class (**E.g. LoginModel**) and some properties as you do see below in the file.

```

1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel.DataAnnotations;
4  using System.Linq;
5  using System.Threading.Tasks;
6
7  namespace BookStore.Models
8  {
9      public class LoginModel
10     {
11         [Required(ErrorMessage = "User Name is required")]
12         public string Username { get; set; }
13
14         [Required(ErrorMessage = "Password is required")]
15         public string Password { get; set; }
16     }
17 }
```

Login Model - Asp.Net Core 5 authentication hosted with ❤ by GitHub

[view raw](#)

Now, go to the project folder structure and then right click on the **Controllers** folder and then select **Add** and then click on the **Controller...** and then select the **MVC Controller – Empty** from the middle pane and then click on the **Add** button and then enter the name of the controller (**E.g. AuthenticateController**) and then click the **Add** button. So, it will create a new controller class with the name of **AuthenticateController.cs**.

Now, write the code as you do see below in the file.

```

1  using BookStore.Data;
2  using BookStore.Models;
3  using Microsoft.AspNetCore.Identity;
4  using Microsoft.AspNetCore.Mvc;
```

Get Notifications *

```

5   using Microsoft.IdentityModel.Tokens;
6   using System;
7   using System.Collections.Generic;
8   using System.IdentityModel.Tokens.Jwt;
9   using System.Linq;
10  using System.Security.Claims;
11  using System.Text;
12  using System.Threading.Tasks;
13
14 namespace BookStore.Controllers
15 {
16     [Route("api/[controller]")]
17     public class AuthenticateController : Controller
18     {
19         private UserManager<ApplicationUser> userManager;
20
21         public AuthenticateController(UserManager<ApplicationUser> userManager)
22         {
23             this.userManager = userManager;
24         }
25
26         [HttpPost]
27         [Route("login")]
28         public async Task<IActionResult> Login([FromBody] LoginModel model)
29         {
30             var user = await userManager.FindByNameAsync(model.Username);
31             if (user != null && await userManager.CheckPasswordAsync(user, model.Password))
32             {
33                 var authClaims = new[]
34                 {
35                     new Claim(JwtRegisteredClaimNames.Sub, user.UserName),
36                     new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())
37                 };
38                 var authSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("7S79jvOkEdwoR
39                 var token = new JwtSecurityToken(
40                     issuer: "https://dotnetdetail.net",
41                     audience: "https://dotnetdetail.net",
42                     expires: DateTime.Now.AddDays(5),
43                     claims: authClaims,
44                     signingCredentials: new Microsoft.IdentityModel.Tokens.SigningCredentials(authS
45                 );
46
47                 return Ok(new
48                 {
49                     token = new JwtSecurityTokenHandler().WriteToken(token),
50                     expiration = token.ValidTo
51                 });
52
53             }
54             return Unauthorized();
55         }
56     }
57 }
```

Get Notifications *

```
54  
55     }  
56 }
```

Authenticate Controller - AspNet Core 5 Authentication hosted with ❤ by GitHub

[view raw](#)

Let's understand the above code.

Line # 16: In this line, we are declaring the route path for API.

Line # 19: In this line, we are injecting the User Manager.

Line # 30: In this line, we are getting the specific user.

Line # 31: In this line, we are checking the user value and password. If the condition is true, then we will generate the token otherwise it will return an Unauthorized response.

Line # 33 to 37: In this block of code, we are creating claims.

Line # 38: In this line, we are creating the signing key.

[READ Entity Framework Core 3.0 Bulk Insert Update and Delete - Asp.Net Core 3.0](#)

Line # 39 to 45: In this block of code, we are generating the token using JWT.

Line # 46 to 50: In this block of code, we are returning the status with token and expiration time after generating the token successfully.

[Get Notifications *](#)

Now, go to the project folder structure and then go

WeatherForecastController.cs and then just put th

you do see below in the file's **line # 11**.

```

1  using Microsoft.AspNetCore.Authorization;
2  using Microsoft.AspNetCore.Mvc;
3  using Microsoft.Extensions.Logging;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Threading.Tasks;
8
9  namespace BookStore.Controllers
10 {
11     [Authorize]
12     [ApiController]
13     [Route("[controller]")]
14     public class WeatherForecastController : ControllerBase
15     {
16         private static readonly string[] Summaries = new[]
17         {
18             "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering",
19         };
20
21         private readonly ILogger<WeatherForecastController> _logger;
22
23         public WeatherForecastController(ILogger<WeatherForecastController> logger)
24         {
25             _logger = logger;
26         }
27
28         [HttpGet]
29         public IEnumerable<WeatherForecast> Get()
30         {
31             var rng = new Random();
32             return Enumerable.Range(1, 5).Select(index => new WeatherForecast
33             {
34                 Date = DateTime.Now.AddDays(index),
35                 TemperatureC = rng.Next(-20, 55),
36                 Summary = Summaries[rng.Next(Summaries.Length)]
37             })
38             .ToArray();
39         }
40     }
41 }
```

Authorize in weatherforecast controller - asp.net core 5 authentication hosted with ❤ by GitHub

[view raw](#)

Get Notifications *

Now, we will add authentication service in the **Startup.cs**

line # 40 to 58 and **line # 77.**

```

1  using BookStore.Data;
2  using Microsoft.AspNetCore.Authentication.JwtBearer;
3  using Microsoft.AspNetCore.Builder;
4  using Microsoft.AspNetCore.Hosting;
5  using Microsoft.AspNetCore.HttpsPolicy;
6  using Microsoft.AspNetCore.Identity;
7  using Microsoft.AspNetCore.Mvc;
8  using Microsoft.EntityFrameworkCore;
9  using Microsoft.Extensions.Configuration;
10 using Microsoft.Extensions.DependencyInjection;
11 using Microsoft.Extensions.Hosting;
12 using Microsoft.Extensions.Logging;
13 using Microsoft.IdentityModel.Tokens;
14 using Microsoft.OpenApi.Models;
15 using System;
16 using System.Collections.Generic;
17 using System.Linq;
18 using System.Text;
19 using System.Threading.Tasks;
20
21 namespace BookStore
22 {
23     public class Startup
24     {
25         public Startup(IConfiguration configuration)
26         {
27             Configuration = configuration;
28         }
29
30         public IConfiguration Configuration { get; }
31
32         // This method gets called by the runtime. Use this method to add services to the container
33         public void ConfigureServices(IServiceCollection services)
34         {
35             services.AddControllers();
36             services.AddDbContext<BookStoreDbContext>(options => options.UseSqlServer(Configuration
37             services.AddIdentity<ApplicationUser, IdentityRole>()
38                 .AddEntityFrameworkStores<BookStoreDbContext>()
39                 .AddDefaultTokenProviders();
40             services.AddAuthentication(options =>
41             {
42                 options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
43                 options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
44                 options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
45             })
46             .AddJwtBearer(options =>

```

Get Notifications *

```

47
48     options.SaveToken = true;
49
50     options.RequireHttpsMetadata = false;
51
52     options.TokenValidationParameters = new Microsoft.IdentityModel.Tokens.TokenValidat
53     {
54         ValidateIssuer = true,
55         ValidateAudience = true,
56         ValidAudience = "https://dotnetdetail.net",
57         ValidIssuer = "https://dotnetdetail.net",
58         IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("7S79jv0kEdw
59     );
60
61     services.AddSwaggerGen(c =>
62     {
63
64         c.SwaggerDoc("v1", new OpenApiInfo { Title = "BookStore", Version = "v1" });
65
66     });
67
68 }
69
70 // This method gets called by the runtime. Use this method to configure the HTTP request pi
71 public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
72 {
73
74     if (env.IsDevelopment())
75     {
76
77         app.UseDeveloperExceptionPage();
78
79         app.UseSwagger();
80
81         app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "BookStore v1")
82
83     }
84
85 }

```

startup with authorization - asp.net core 5 authentication hosted with  by GitHub

[view raw](#)

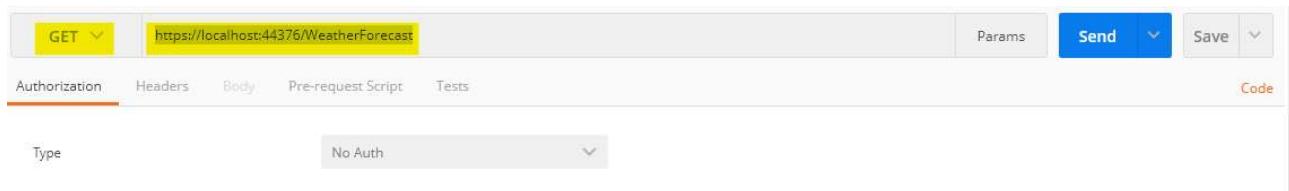
Test Project

Now, in this step, we will test our project and we will see the output using the **postman**. So, run your project by pressing **f5** or just clicking on the **IIS Express**.

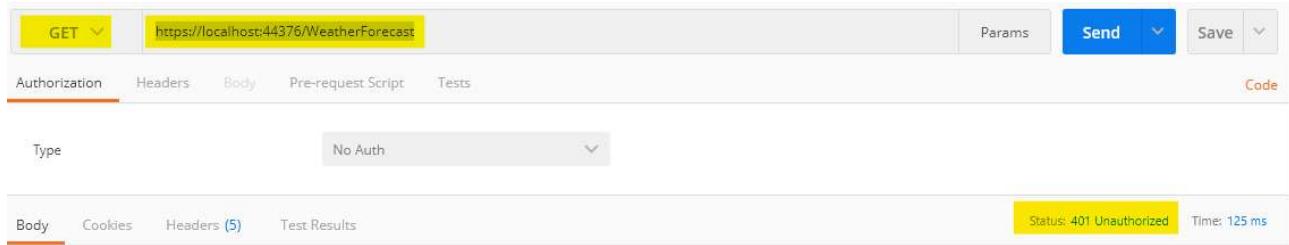
Get Notifications *

Now, open the postman and then enter the API path (**E.g.**

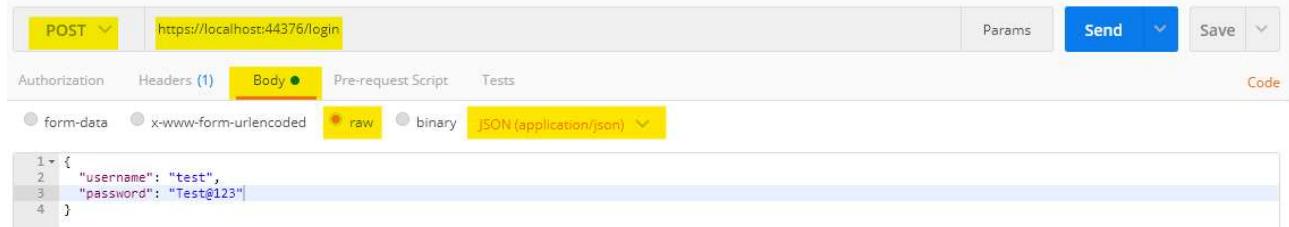
https://localhost:44376/WeatherForecast) using the get request and then click on the **Send** button as you do see below in the screenshot.



After clicking on the **Send** button, then you will see the output with **unauthorized** status as you do see below in the screenshot.



Now, we will generate the token using the login method by passing the **username** and **password** as you do see below in the screenshot.



If the above login request is generated successfully, then you will see the output as you do see below in the screenshot.

Get Notifications *

```

1 "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
2 .eyJzdWIiOiJ0ZXN0IiwianRpIjoiN2ZIMjIzDQtOTZhZS00ZDYwLThjMzAtMjYzN2U5NTJmNmUvIiwizXhwIjoxNjA3MDMyMDA1LCJpc3N10iJodHRwczovL2RvdG51dGR1dGFpbC5uZXQiLCJhdWQi
3 OiJodHRwczovL2RvdG51dGR1dGFpbC5uZXQiQ.cNK-nxSSrsjzhw1k1YXCW7e48WgJa8eBssEQBK0ckM4",
4 "expiration": "2020-12-03T21:46:45Z"
      
```

Now, copy the token value and then enter the url (<https://localhost:44376/weatherforecast>) with **Get** method and then go to **Headers** tab and then enter the **key (Authorization)** and then **Value(Bearer <Token Value>)** and then click on the **Send** button.

Key	Value	Description	Bulk Edit	Presets
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ0ZXN0IiwianRpIjoiN2ZIMjIzDQtOTZhZS00ZDYwLThjMzAtMjYzN2U5NTJmNmUvIiwizXhwIjoxNjA3MDMyMDA1LCJpc3N10iJodHRwczovL2RvdG51dGR1dGFpbC5uZXQiLCJhdWQiOiJodHRwczovL2RvdG51dGR1dGFpbC5uZXQiQ.cNK-nxSSrsjzhw1k1YXCW7e48WgJa8eBssEQBK0ckM4			

After clicking on the **Send** button, then you will see the output with 200 status code as you do see below in the screenshot.

Key	Value	Description	Bulk Edit	Presets
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ0ZXN0IiwianRpIjoiN2ZIMjIzDQtOTZhZS00ZDYwLThjMzAtMjYzN2U5NTJmNmUvIiwizXhwIjoxNjA3MDMyMDA1LCJpc3N10iJodHRwczovL2RvdG51dGR1dGFpbC5uZXQiLCJhdWQiOiJodHRwczovL2RvdG51dGR1dGFpbC5uZXQiQ.cNK-nxSSrsjzhw1k1YXCW7e48WgJa8eBssEQBK0ckM4			

Congratulation, we have successfully created a **web api token based authentication** using **JWT** in **Asp.Net Core 5** with Visual Studio 2019

Note:- If you found this blog helpful then:



Buy me a coffee

« **How to upgrade Asp.Net Core 3.1 App to Asp.Net Core 5**

Build a Microservice using Asp Net Core 5 and Docker »

FILED UNDER: [ASP.NET CORE](#), [ASP.NET CORE 5](#), [AUTHENTICATION AND AUTHORIZATION](#), [SECURITY](#), [WEB API](#)
TAGGED WITH: [.NET CORE 5](#), [ASP.NET CORE 5](#), [ASP.NET CORE AUTHENTICATION](#), [AUTHENTICATION](#), [AUTHENTICATION AND AUTHORIZATION](#), [WEB API](#)

Get Notifications *

Comments



Sándor Hatvani says

FEBRUARY 3, 2021 AT 1:41 PM

Hi,

Thank you for your article. By the way I would ask you if authentication and authorization can happen by on-premise AD instead of EF?

Maybe you could suggest how to do it, please? 😊

[Reply](#)



Nick Taglianetti says

APRIL 16, 2021 AT 12:27 AM

Thank you for this tutorial. This is the first token based authentication tutorial that I've successfully implemented for a .NET Core 3.1 Web API solution to actually get a token back after building the required authentication controller. I was able to adapt it simply by installing the specified Nuget packages with the latest 3.1.x targeted framework.

HOWEVER, I am getting a 401 error when trying to send a request to one of my endpoints whose controller is decorated with the [Authorize] attribute. Any ideas what I could be missing? I'm getting the error when testing with both the Swagger (swashbuckle) UI and via Postman. I followed this guide exactly other than using the .NET Core 3.1 Nuget packages. Any advice would be greatly appreciated!

[Get Notifications *](#)

[Reply](#)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Notify me of follow-up comments by email.

Notify me of new posts by email.

POST COMMENT

Get Notifications *

[report this ad](#)[report this ad](#)[Buy me a coffee](#)

JOBS

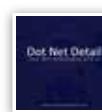
[Dotnet Jobs](#)

BEST ASP.NET HOSTING

[Get Notifications *](#)



report this ad



Dotnet Detail
1,324 likes

[Like Page](#)[Contact Us](#)

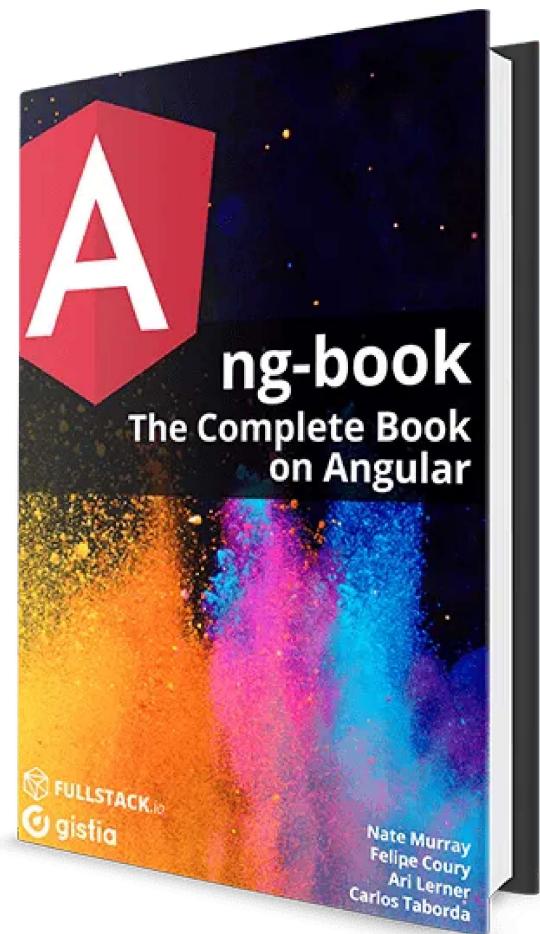
SUBSCRIBE TO BLOG VIA EMAIL

Enter your email address to subscribe.

Email Address

Get Notifications *

ONE OF THE BEST MATERIAL TO LEARN
ANGULAR



Get Notifications *

Copyright © 2021



report this ad

RECENT POSTS

- > [Top LINQ interview questions and answers.](#)

- > [Difference between .NET Core and .NET Framework](#)

- > [Top 10 Programming Languages of the Future.](#)

- > [Top MVC Interview Questions & Answers](#)

- > [20 best Data Science Books: Beginner to Advanced Level](#)

- > [10 Best Machine Learning Books](#)

Get Notifications *

