ASP.NET

Web API Token Based Authentication

37.5k 6 11

Files.zip

Introduction

In this article, we will learn how to implement Token Based Authentication in Web APIs to secure the data.

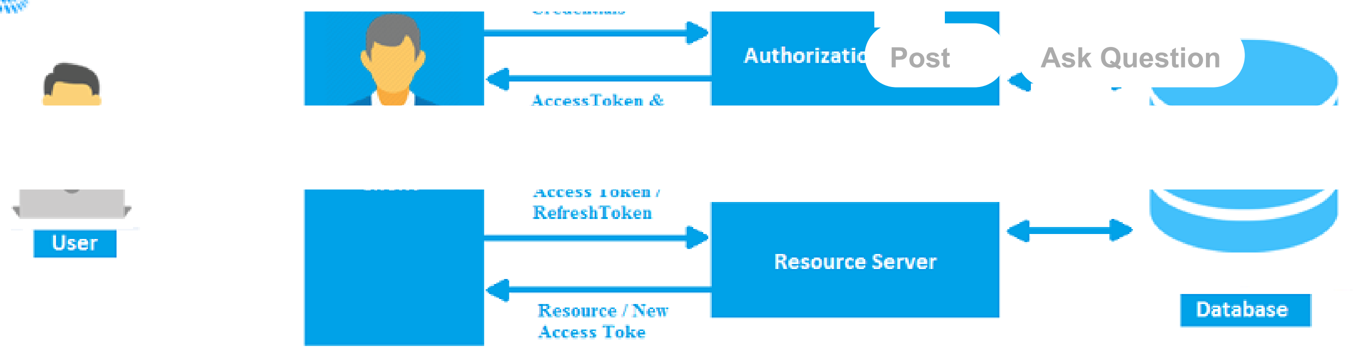
There are 4 common methods of Web API Authentication:

1. HTTP Authentication Schemes (Basic & Bearer)
2. API Keys
3. OAuth (2.0)
4. OpenID Connect

Here we will learn OAuth authentication. OAuth is an open standard for token based authentication and authorization on internet. By using OAuth we can create Token Based Authentication API.

What is Token Based Authentication in Web API?

Token-based authentication is a process where the client application first sends a request to Authentication server with a valid credentials. The Authentication server sends an Access token to the client as a response. This token contains enough data to identify a particular user and it has an expiry time. The client application then uses the token to access the restricted resources in the next requests until the token is valid. If the Access token is expired, then the client application can request for a new access token by using Refresh token.



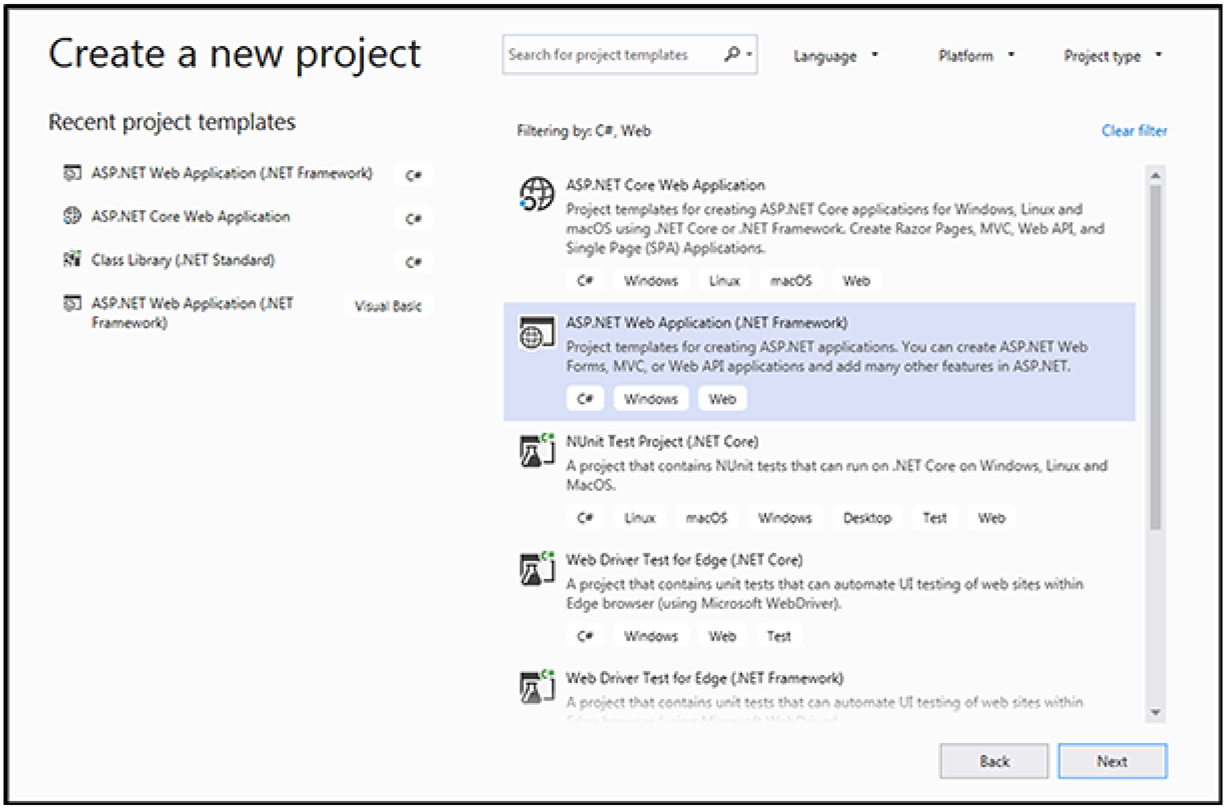
Advantages of Token Based Authentication

- Scalability of Servers
- Loosely Coupling
- Mobile-Friendly

Let's discuss the step by step procedure to create Token-Based Authentication,

Step 1 - Create ASP.NET Web Project in Visual Studio 2019

We have to create web project in Visual Studio as given in the below image. Choose ASP.Net Web Application from the menu.



Create a new project

Recent project templates

- ASP.NET Web Application (.NET Framework) C#
- ASP.NET Core Web Application C#
- Class Library (.NET Standard) C#
- ASP.NET Web Application (.NET Framework) Visual Basic

Search for project templates | Language | Platform | Project type

Filtering by: C#, Web — Clear filter

ASP.NET Core Web Application
Project templates for creating ASP.NET Core applications for Windows, Linux and macOS using .NET Core or .NET Framework. Create Razor Pages, MVC, Web API, and Single Page (SPA) Applications.
C# Windows Linux macOS Web

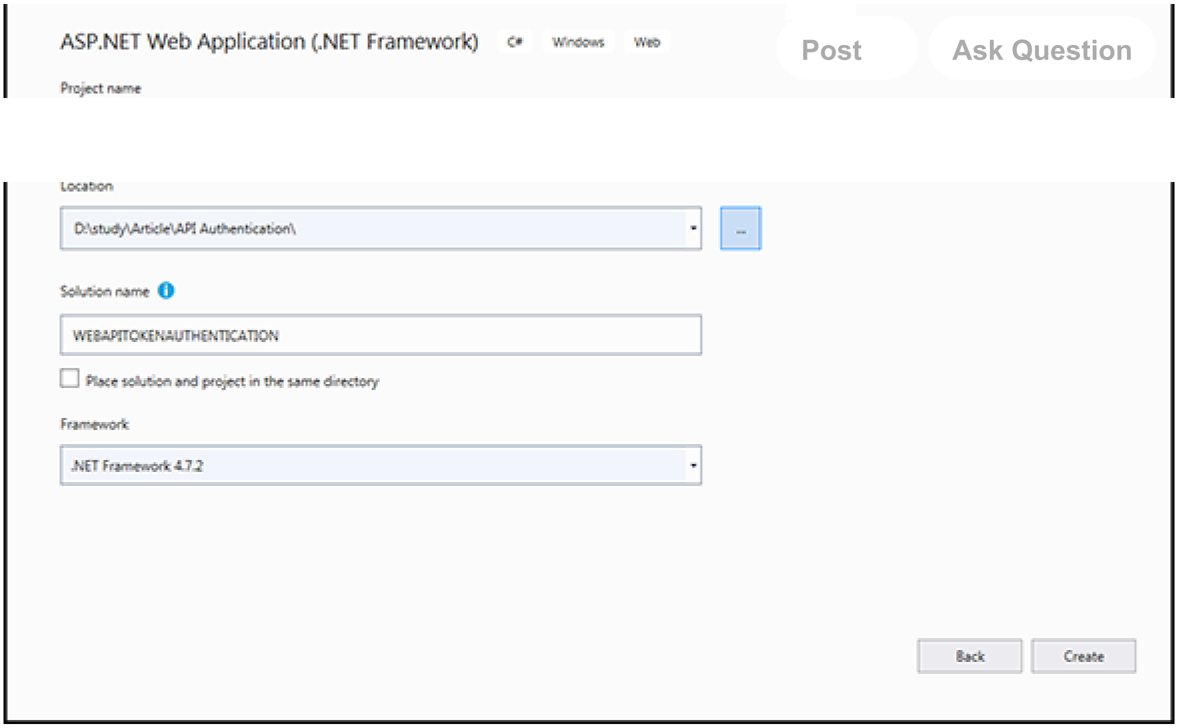
ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
C# Windows Web

NUnit Test Project (.NET Core)
A project that contains NUnit tests that can run on .NET Core on Windows, Linux and MacOS.
C# Linux macOS Windows Desktop Test Web

Web Driver Test for Edge (.NET Core)
A project that contains unit tests that can automate UI testing of web sites within Edge browser (using Microsoft WebDriver).
C# Windows Web Test

Web Driver Test for Edge (.NET Framework)
A project that contains unit tests that can automate UI testing of web sites within

Give the project name as:WEBAPITOKENAUTHENTICATION.



ASP.NET Web Application (.NET Framework) C# Windows Web

Project name

Location

D:\study\Article\API Authentication\

Solution name

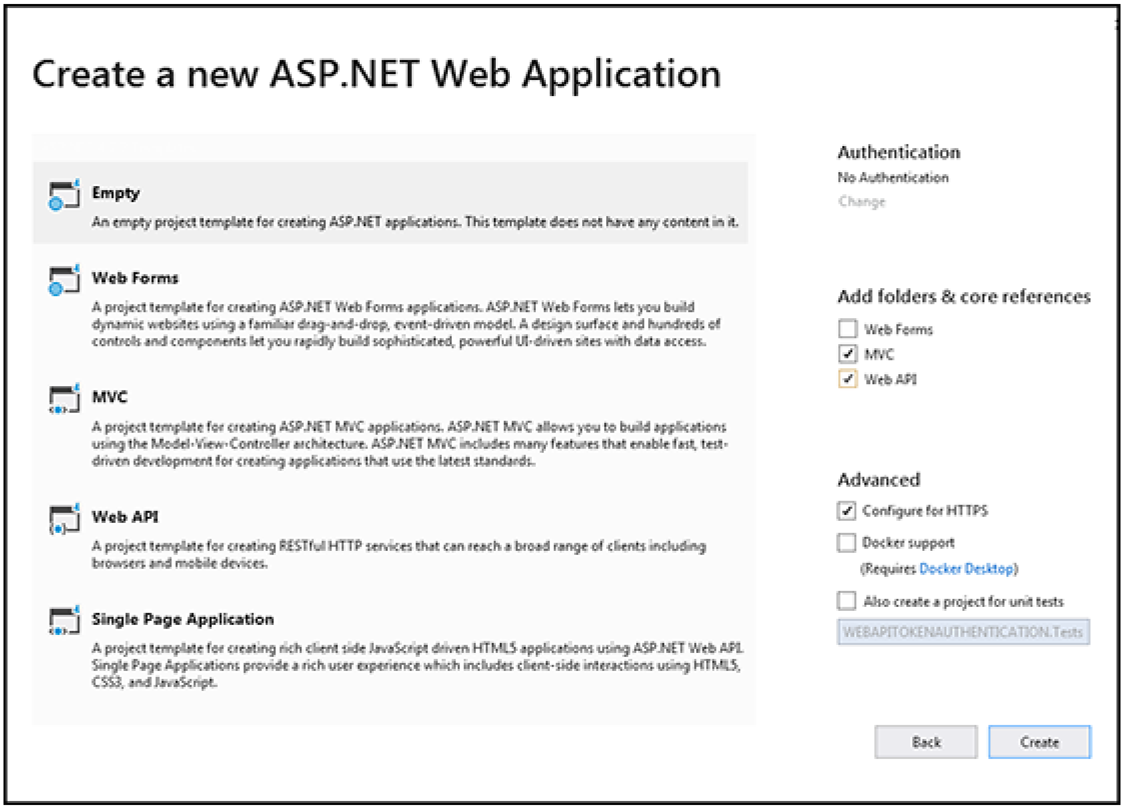
WEBAPITOKENAUTHENTICATION

☐ Place solution and project in the same directory

Framework

.NET Framework 4.7.2

Now choose the empty template and check the "MVC" and "Web API" on the right hand side.



Create a new ASP.NET Web Application

Empty
An empty project template for creating ASP.NET applications. This template does not have any content in it.

Web Forms
A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

MVC
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Web API
A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

Single Page Application
A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.

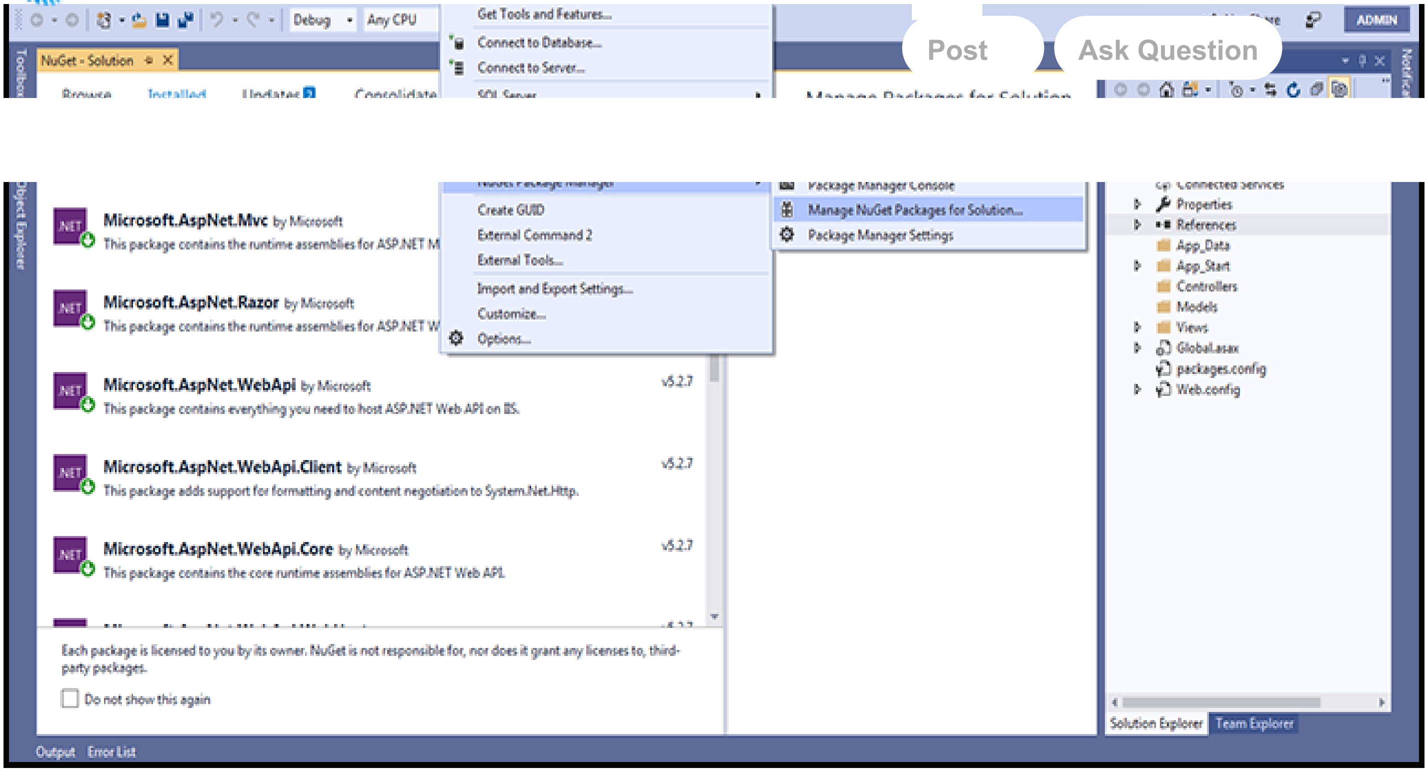
Authentication
No Authentication
Change

Add folders & core references
☐ Web Forms
☑ MVC
☑ Web API

Advanced
☑ Configure for HTTPS
☐ Docker support
(Requires Docker Desktop)
☐ Also create a project for unit tests
WEBAPITOKENAUTHENTICATION.Tests

Step 2 - Addition Of References

In this step,we have to add Nuget References like the below image,



Microsoft.AspNet.Mvc by Microsoft
This package contains the runtime assemblies for ASP.NET M

Microsoft.AspNet.Razor by Microsoft
This package contains the runtime assemblies for ASP.NET W

Microsoft.AspNet.WebApi by Microsoft v5.2.7
This package contains everything you need to host ASP.NET Web API on IIS.

Microsoft.AspNet.WebApi.Client by Microsoft v5.2.7
This package adds support for formatting and content negotiation to System.Net.Http.

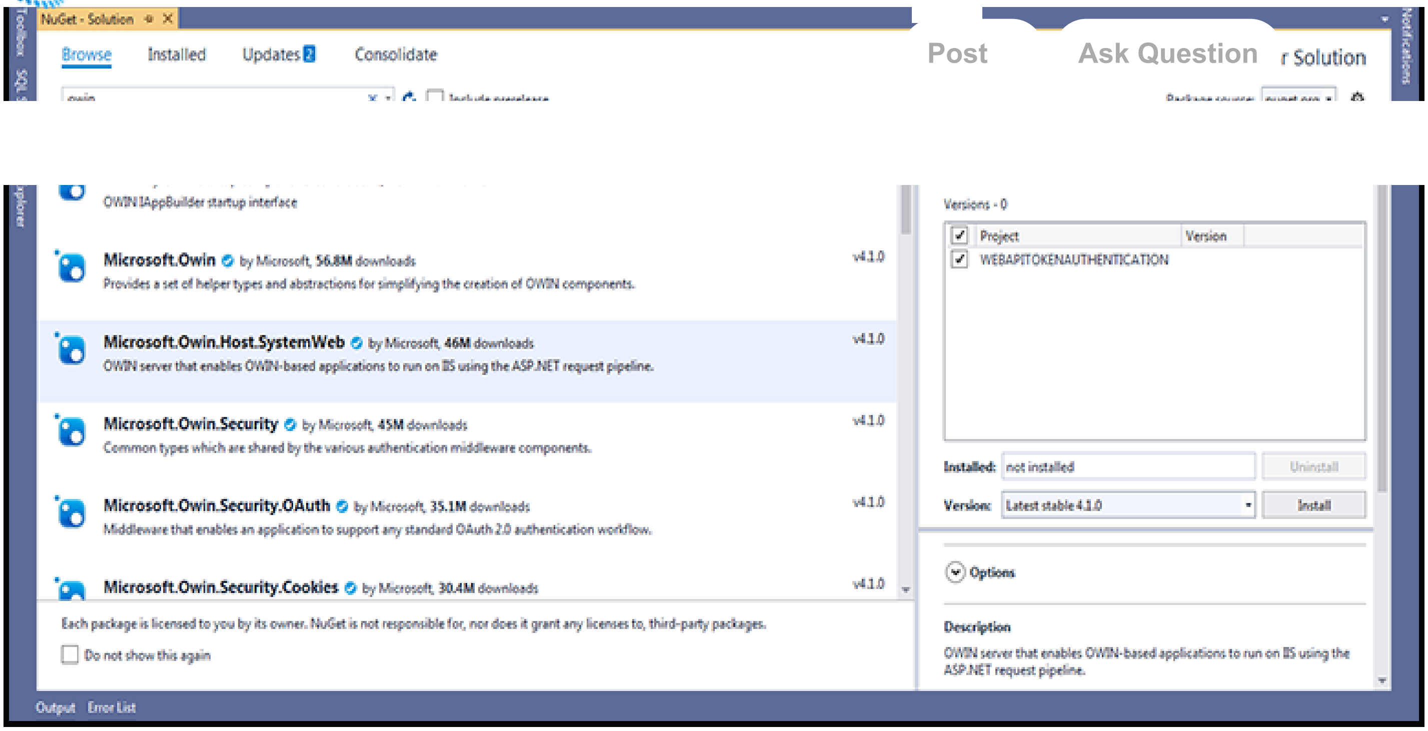
Microsoft.AspNet.WebApi.Core by Microsoft v5.2.7
This package contains the core runtime assemblies for ASP.NET Web API.

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

□ Do not show this again

Here we have to add the following references :

- Microsoft.Owin.Host.SystemWeb
- Microsoft.Owin.Security.OAuth
- Microsoft.Owin.Cors



OWIN IAppBuilder startup interface

Microsoft.Owin by Microsoft, 56.8M downloads v4.1.0
Provides a set of helper types and abstractions for simplifying the creation of OWIN components.

Microsoft.Owin.Host.SystemWeb by Microsoft, 46M downloads v4.1.0
OWIN server that enables OWIN-based applications to run on IIS using the ASP.NET request pipeline.

Microsoft.Owin.Security by Microsoft, 45M downloads v4.1.0
Common types which are shared by the various authentication middleware components.

Microsoft.Owin.Security.OAuth by Microsoft, 35.1M downloads v4.1.0
Middleware that enables an application to support any standard OAuth 2.0 authentication workflow.

Microsoft.Owin.Security.Cookies by Microsoft, 30.4M downloads v4.1.0

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

□ Do not show this again

Versions - 0

☑ Project	Version
☑ WEBAPITOKENAUTHENTICATION	

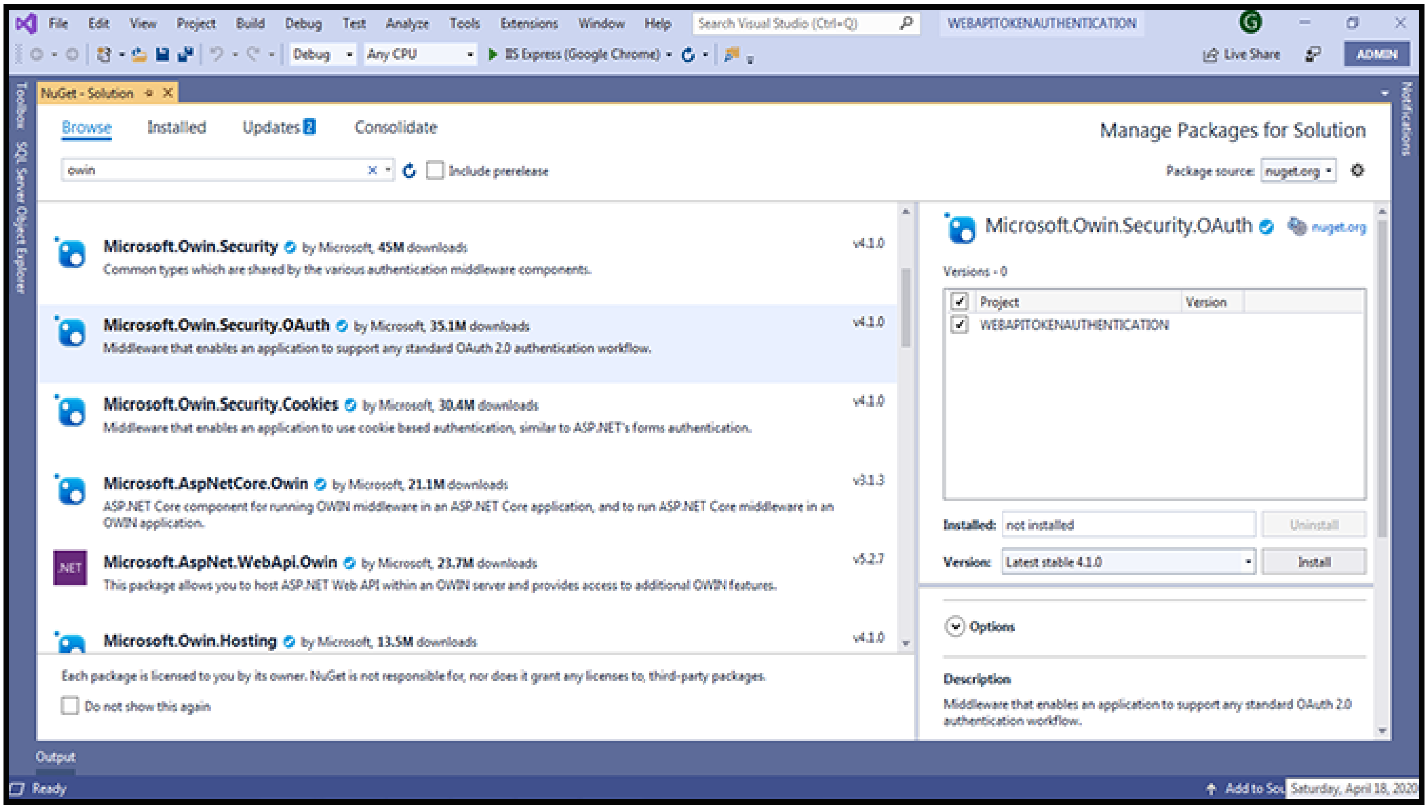
Installed: not installed Uninstall

Version: Latest stable 4.1.0 Install

Options

Description

OWIN server that enables OWIN-based applications to run on IIS using the ASP.NET request pipeline.



Manage Packages for Solution

owin Include prerelease

Package source: nuget.org

Microsoft.Owin.Security by Microsoft, 45M downloads v4.1.0
Common types which are shared by the various authentication middleware components.

Microsoft.Owin.Security.OAuth by Microsoft, 35.1M downloads v4.1.0
Middleware that enables an application to support any standard OAuth 2.0 authentication workflow.

Microsoft.Owin.Security.Cookies by Microsoft, 30.4M downloads v4.1.0
Middleware that enables an application to use cookie based authentication, similar to ASP.NET's forms authentication.

Microsoft.AspNetCore.Owin by Microsoft, 21.1M downloads v3.1.3
ASP.NET Core component for running OWIN middleware in an ASP.NET Core application, and to run ASP.NET Core middleware in an OWIN application.

Microsoft.AspNet.WebApi.Owin by Microsoft, 23.7M downloads v5.2.7
This package allows you to host ASP.NET Web API within an OWIN server and provides access to additional OWIN features.

Microsoft.Owin.Hosting by Microsoft, 13.5M downloads v4.1.0

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

□ Do not show this again

Microsoft.Owin.Security.OAuth nuget.org

Versions - 0

☑ Project	Version
☑ WEBAPITOKENAUTHENTICATION	

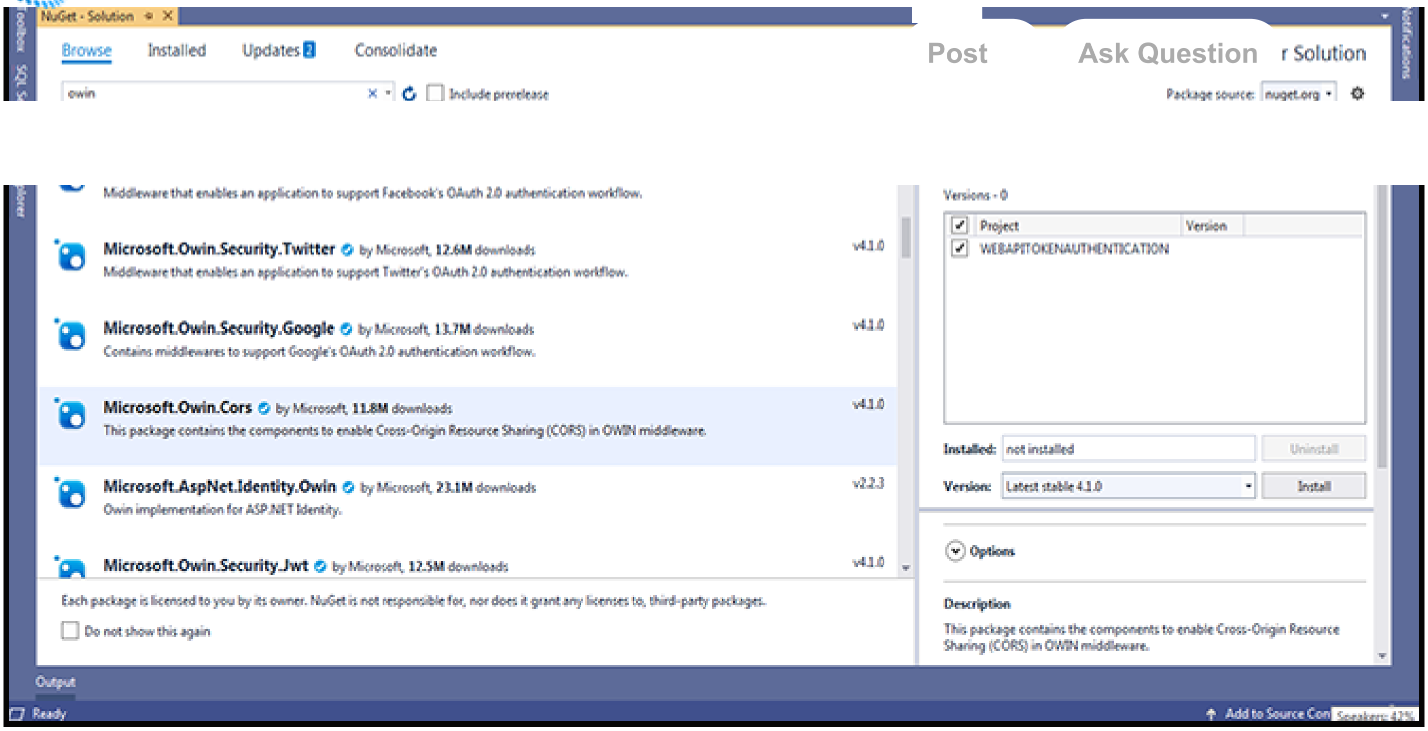
Installed: not installed Uninstall

Version: Latest stable 4.1.0 Install

Options

Description

Middleware that enables an application to support any standard OAuth 2.0 authentication workflow.



Middleware that enables an application to support Facebook's OAuth 2.0 authentication workflow.

Microsoft.Owin.Security.Twitter by Microsoft, 12.6M downloads v4.1.0
Middleware that enables an application to support Twitter's OAuth 2.0 authentication workflow.

Microsoft.Owin.Security.Google by Microsoft, 13.7M downloads v4.1.0
Contains middlewares to support Google's OAuth 2.0 authentication workflow.

Microsoft.Owin.Cors by Microsoft, 11.8M downloads v4.1.0
This package contains the components to enable Cross-Origin Resource Sharing (CORS) in OWIN middleware.

Microsoft.AspNet.Identity.Owin by Microsoft, 23.1M downloads v2.2.3
Owin implementation for ASP.NET Identity.

Microsoft.Owin.Security.Jwt by Microsoft, 12.5M downloads v4.1.0

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

□ Do not show this again

Versions - 0

☑ Project	Version
☑ WEBAPITOKENAUTHENTICATION	

Installed: not installed [Uninstall]

Version: Latest stable 4.1.0 [Install]

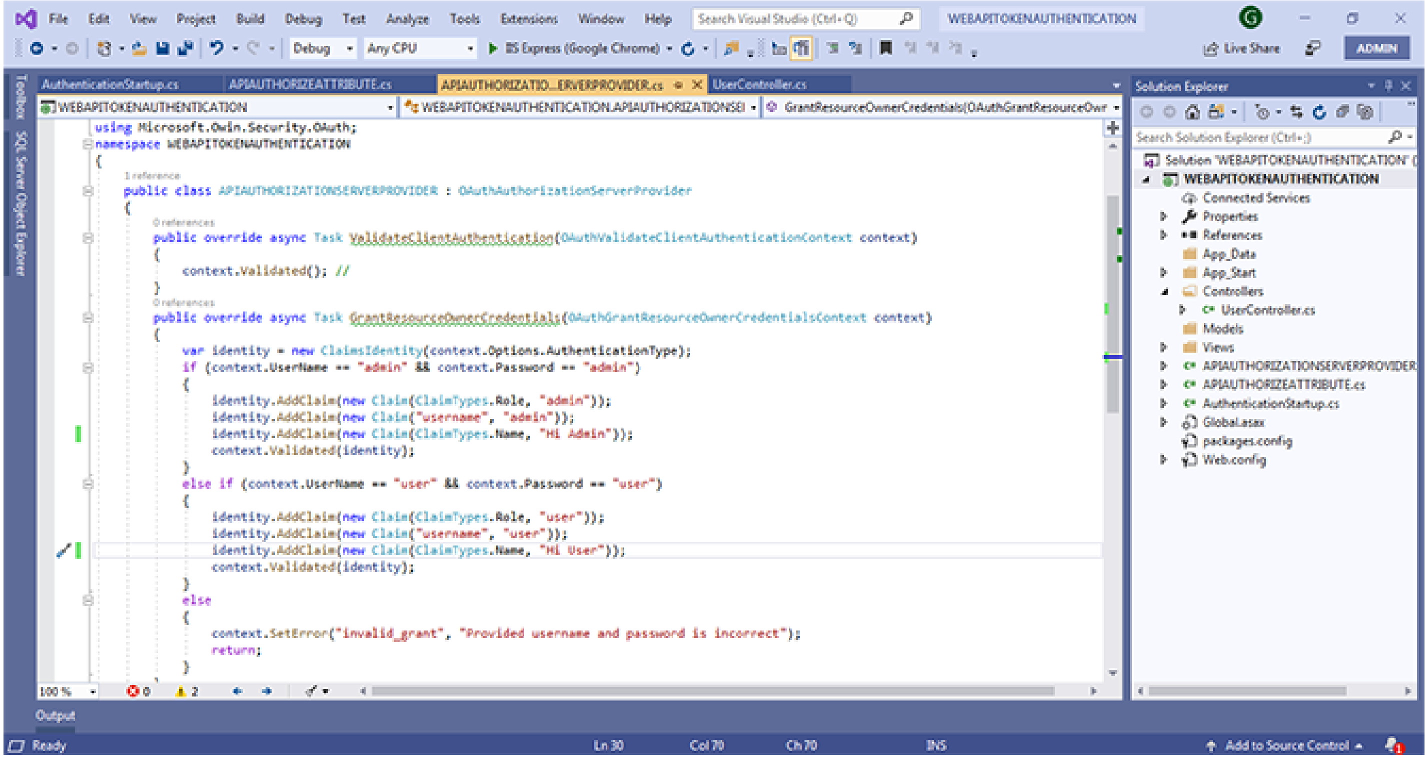
Options

Description

This package contains the components to enable Cross-Origin Resource Sharing (CORS) in OWIN middleware.

Step 3 - Create APIAUTHORIZATIONSERVERPROVIDER.cs Class File

Now, let's create the class file to provide credentials to access data depending on username, password, and roles.

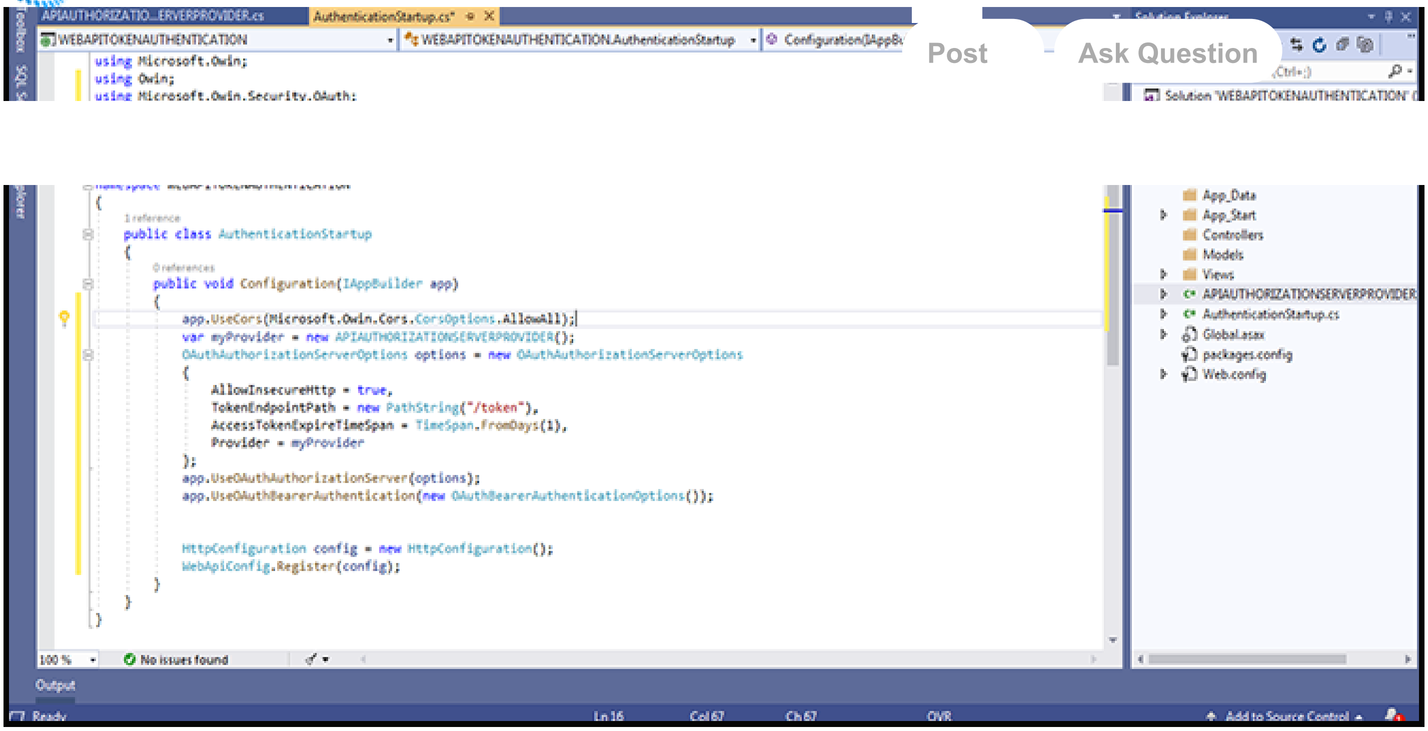


```
using Microsoft.Owin.Security.OAuth;
namespace WEBAPITOKENAUTHENTICATION
{
    1 reference
    public class APIAUTHORIZATIONSERVERPROVIDER : OAuthAuthorizationServerProvider
    {
        0 references
        public override async Task ValidateClientAuthentication(OAuthValidateClientAuthenticationContext context)
        {
            context.Validated(); //
        }
        0 references
        public override async Task GrantResourceOwnerCredentials(OAuthGrantResourceOwnerCredentialsContext context)
        {
            var identity = new ClaimsIdentity(context.Options.AuthenticationType);
            if (context.UserName == "admin" && context.Password == "admin")
            {
                identity.AddClaim(new Claim(ClaimTypes.Role, "admin"));
                identity.AddClaim(new Claim("username", "admin"));
                identity.AddClaim(new Claim(ClaimTypes.Name, "Hi Admin"));
                context.Validated(identity);
            }
            else if (context.UserName == "user" && context.Password == "user")
            {
                identity.AddClaim(new Claim(ClaimTypes.Role, "user"));
                identity.AddClaim(new Claim("username", "user"));
                identity.AddClaim(new Claim(ClaimTypes.Name, "Hi User"));
                context.Validated(identity);
            }
            else
            {
                context.SetError("invalid_grant", "Provided username and password is incorrect");
                return;
            }
```

```
public class APIAUTHORIZATIONSERVERPROVIDER : OAuthAuthorizationServerProv
    {
        public override async Task ValidateClientAuthentication(OAuthVali
        }
        public override async Task GrantResourceOwnerCredentials(OAuthGran
        {
            var identity = new ClaimsIdentity(context.Options.Authenticati
            if (context.UserName == "admin" && context.Password == "admin"
            {
                identity.AddClaim(new Claim(ClaimTypes.Role, "admin"));
                identity.AddClaim(new Claim("username", "admin"));
                identity.AddClaim(new Claim(ClaimTypes.Name, "Hi Admin"));
                context.Validated(identity);
            }
            else if (context.UserName == "user" && context.Password == "us
            {
                identity.AddClaim(new Claim(ClaimTypes.Role, "user"));
                identity.AddClaim(new Claim("username", "user"));
                identity.AddClaim(new Claim(ClaimTypes.Name, "Hi User"));
                context.Validated(identity);
            }
            else
            {
                context.SetError("invalid_grant", "Provided username and p
                return;
            }
        }
    }
```

Step 4 - Create a AuthenticationStartup.cs Class File

Here, we need to create a new class file to implement the code configuration provider and create an instance of class APIAUTHORIZATIONSERVERPROVIDER.



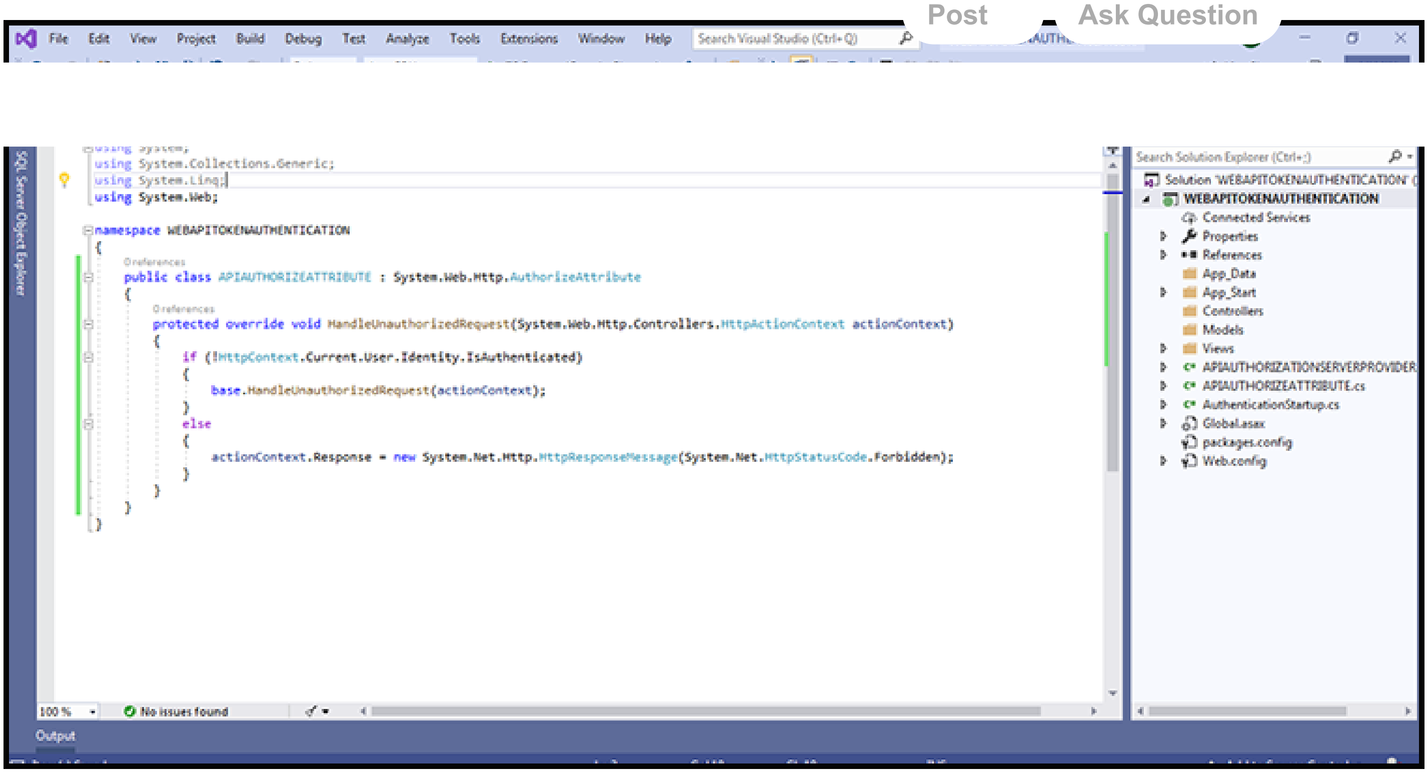
```
using Microsoft.Owin;
using Owin;
using Microsoft.Owin.Security.OAuth;

namespace WEBAPITOKENAUTHENTICATION
{
    1 reference
    public class AuthenticationStartup
    {
        0 references
        public void Configuration(IAppBuilder app)
        {
            app.UseCors(Microsoft.Owin.Cors.CorsOptions.AllowAll);
            var myProvider = new APIAUTHORIZATIONSERVERPROVIDER();
            OAuthAuthorizationServerOptions options = new OAuthAuthorizationServerOptions
            {
                AllowInsecureHttp = true,
                TokenEndpointPath = new PathString("/token"),
                AccessTokenExpireTimeSpan = TimeSpan.FromDays(1),
                Provider = myProvider
            };
            app.UseOAuthAuthorizationServer(options);
            app.UseOAuthBearerAuthentication(new OAuthBearerAuthenticationOptions());


            HttpConfiguration config = new HttpConfiguration();
            WebApiConfig.Register(config);
        }
    }
}
```

APIAUTHORIZATIO...ERVERPROVIDER.cs
AuthenticationStartup.cs*
WEBAPITOKENAUTHENTICATION
WEBAPITOKENAUTHENTICATION.AuthenticationStartup
Configuration(IAppBuilder
Solution Explorer
Solution 'WEBAPITOKENAUTHENTICATION'
App_Data
App_Start
Controllers
Models
Views
APIAUTHORIZATIONSERVERPROVIDER
AuthenticationStartup.cs
Global.asax
packages.config
Web.config
100 %
No issues found
Output
Ready
Ln 16
Col 67
Ch 67
OVR
Add to Source Control

Code is given below:

```
public class AuthenticationStartup
   {
       public void Configuration(IAppBuilder app)
       {
           app.UseCors(Microsoft.Owin.Cors.CorsOptions.AllowAll);
           var myProvider = new APIAUTHORIZATIONSERVERPROVIDER();
           OAuthAuthorizationServerOptions options = new OAuthAuthorizati
           {
               AllowInsecureHttp = true,
               TokenEndpointPath = new PathString("/token"),
               AccessTokenExpireTimeSpan = TimeSpan.FromDays(1),
               Provider = myProvider
           };
           app.UseOAuthAuthorizationServer(options);
           app.UseOAuthBearerAuthentication(new OAuthBearerAuthentication(


           HttpConfiguration config = new HttpConfiguration();
           WebApiConfig.Register(config);
       }
   }
```

Step 5 - Create a APIAUTHORIZEATTRIBUTE.cs Class File



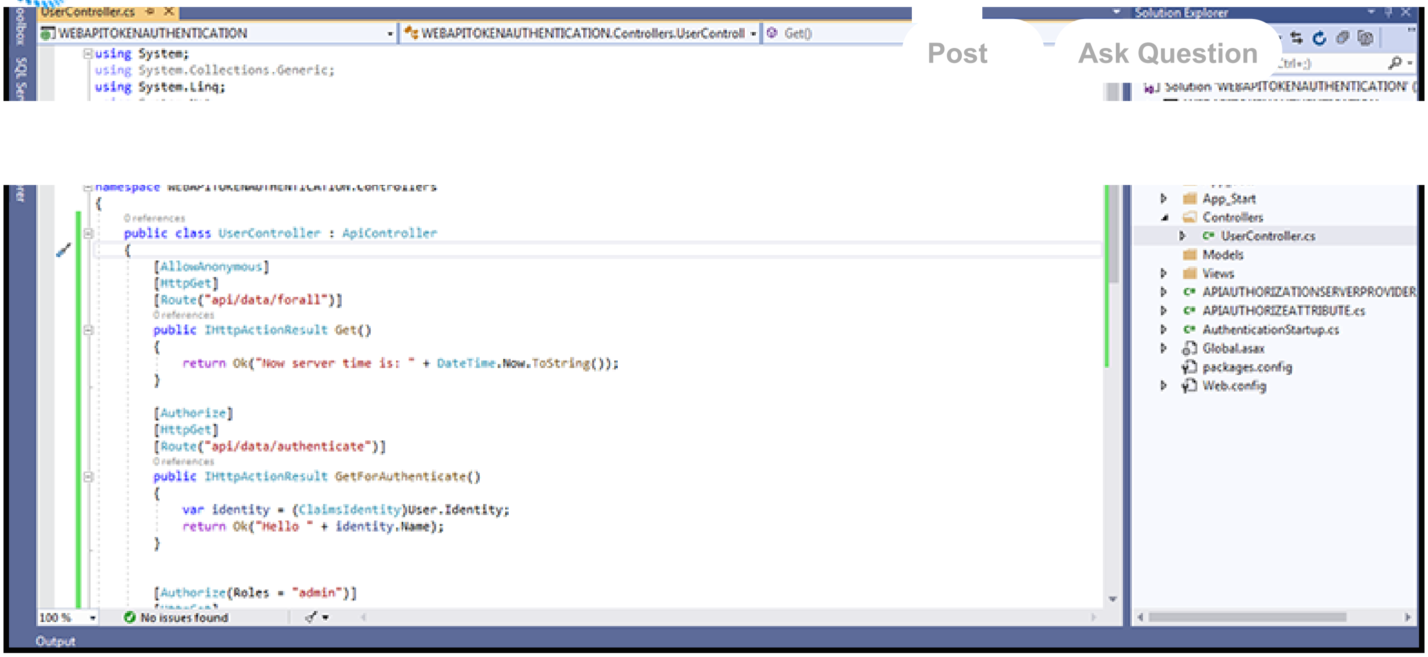
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace WEBAPITOKENAUTHENTICATION
{
    0 references
    public class APIAUTHORIZEATTRIBUTE : System.Web.Http.AuthorizeAttribute
    {
        0 references
        protected override void HandleUnauthorizedRequest(System.Web.Http.Controllers.HttpActionContext actionContext)
        {
            if (!HttpContext.Current.User.Identity.IsAuthenticated)
            {
                base.HandleUnauthorizedRequest(actionContext);
            }
            else
            {
                actionContext.Response = new System.Net.Http.HttpResponseMessage(System.Net.HttpStatusCode.Forbidden);
            }
        }
    }
}
```

Code is given below:

```
public class APIAUTHORIZEATTRIBUTE : System.Web.Http.AuthorizeAttribute
  {
      protected override void HandleUnauthorizedRequest(System.Web.Http.C
      {
          if (!HttpContext.Current.User.Identity.IsAuthenticated)
          {
              base.HandleUnauthorizedRequest(actionContext);
          }
          else
          {
              actionContext.Response = new System.Net.Http.HttpResponseMe
          }
      }
  }
```

Step 6 - Create a controller with name UserController

Now we have to create an empty web api controller with name usercontroller. In this controller, we will write the actions with different authorization and roles.



```
using System;
using System.Collections.Generic;
using System.Linq;

namespace WEBAPITOKENAUTHENTICATION.Controllers
{
    0 references
    public class UserController : ApiController
    {
        [AllowAnonymous]
        [HttpGet]
        [Route("api/data/forall")]
        0 references
        public IHttpActionResult Get()
        {
            return Ok("Now server time is: " + DateTime.Now.ToString());
        }

        [Authorize]
        [HttpGet]
        [Route("api/data/authenticate")]
        0 references
        public IHttpActionResult GetForAuthenticate()
        {
            var identity = (ClaimsIdentity)User.Identity;
            return Ok("Hello " + identity.Name);
        }


        [Authorize(Roles = "admin")]
```

The first action "Get" will be available for anonymous users .No authetication or token is needed for this.

```
[AllowAnonymous]
[HttpGet]
[Route("api/data/forall")]
public IHttpActionResult Get()
{
    return Ok("Now server time is: " + DateTime.Now.ToString());
}
```

The second action "GetForAuthenticate" only allows the authorized user to access it.

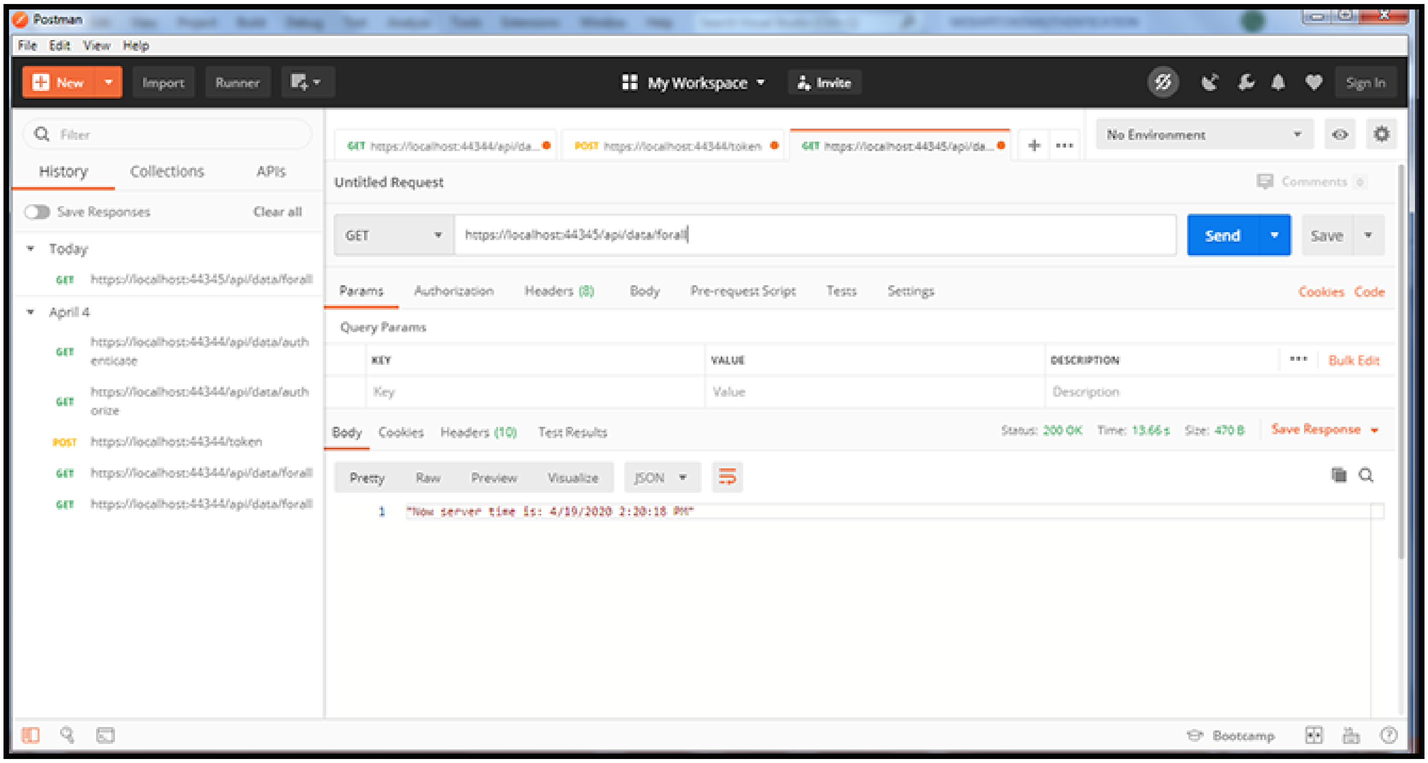
```
[Authorize]
[HttpGet]
[Route("api/data/authenticate")]
public IHttpActionResult GetForAuthenticate()
{
    var identity = (ClaimsIdentity)User.Identity;
    return Ok("Hello " + identity.Name);
}
```

The third action "GetForAdmin" checks authorization and allows only admins to access it.

```
[Authorize(Roles = "admin")]
[HttpGet]
[Route("api/data/authorize")]
public IHttpActionResult GetForAdmin()
{
    var identity = (ClaimsIdentity)User.Identity;
    var roles = identity.Claims
```

```
        return Ok("Hello " + identity.Name + " Ro    " + string.Join(",", role
}
```

Now we have to access the controller. For that purpose, we are using postman to get data.I n case ofthe first action "Get",we can access the data without generating a token as no authorization is needed for it, by just sending GET request for route "api/data/forall".

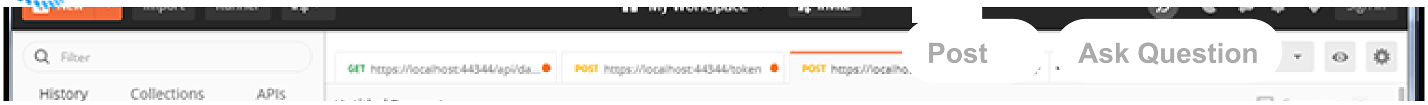


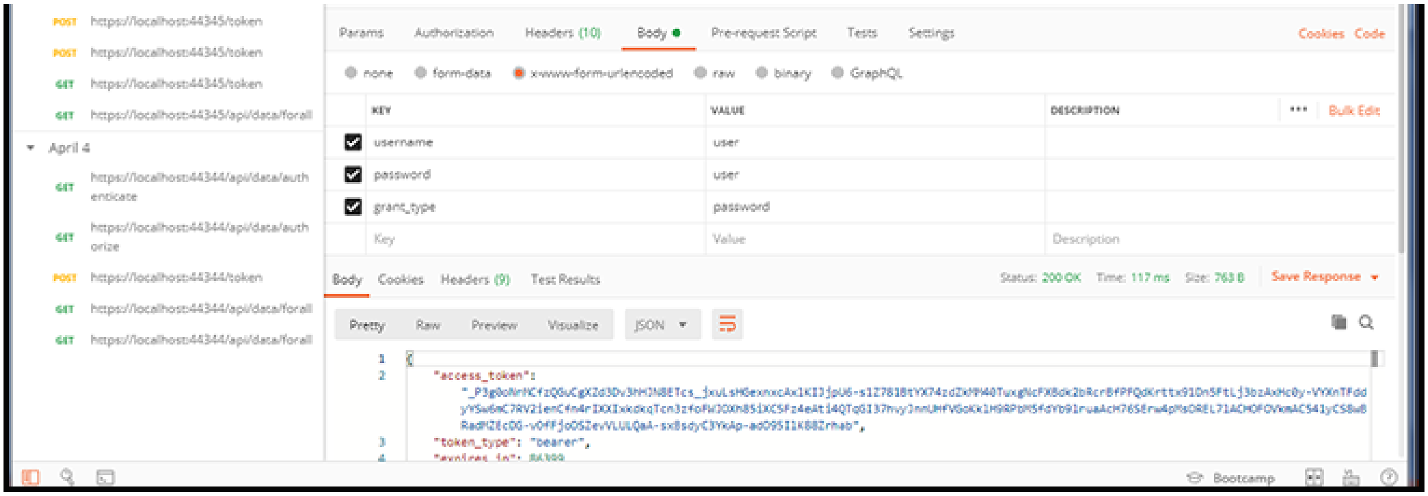
GET https://localhost:44345/api/data/forall

Status: 200 OK Time: 13.66 s Size: 470 B

```
1   "Now server time is: 4/19/2020 2:20:18 PM"
```

But for the rest of the actions, we have to generate a token using credentials. So we have to do post request for the token.





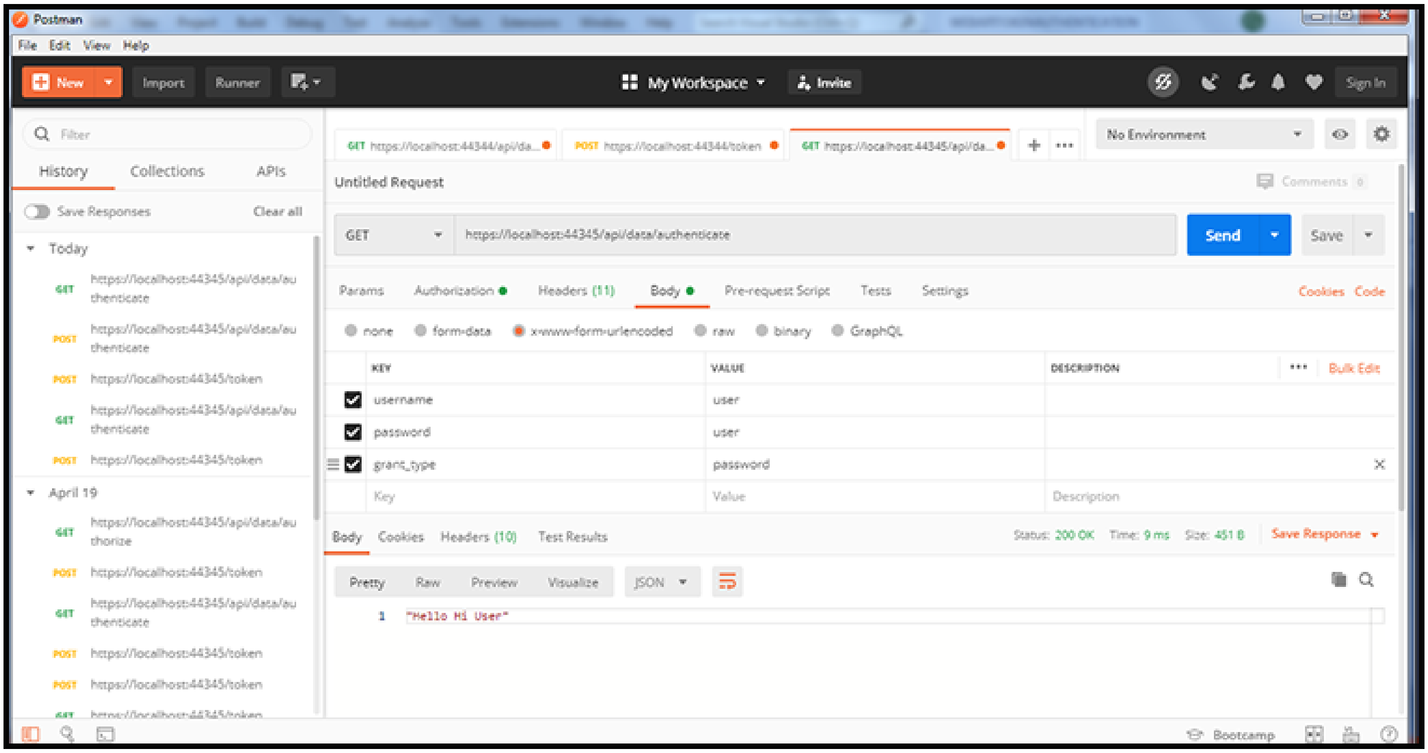
Body: x-www-form-urlencoded

KEY	VALUE	DESCRIPTION
☑ username	user	
☑ password	user	
☑ grant_type	password	

Status: 200 OK Time: 117 ms Size: 763 B

```
{
    "access_token":
        "_P3g0oNnHCfzQGuCgXZd3Dv3hHJN8ETcs_jxuLsHGexnxcAx1KIJjpU6-s1Z7818tYX74zdZkMM40TuxgNcFX8dk2bRcrBfPFQdKrttx91Dn5FtLj3bzAxHc0y-VYXnTFdd
        yYSw6mC7RV2ienCfn4rIXXIxkdkqTcn3zfoFWJOXh85iXC5Fz4eAti4QTqGI37hvyJnnUMfVGoKk1H9RPbM5fdYb91ruaAcH765Erw4pMsOREL71ACHOFOVkmAC541yCS8w8
        RadMZEcDG-vOffjoOSZevVLULQaA-sxBsdyC3YkAp-adO95I1K88Zrhab",
    "token_type": "bearer",
    "expires_in": 86399
```

Now once the token is generated for the "user" now we can easily access the actions by using the user token.



Untitled Request

GET https://localhost:44345/api/data/authenticate

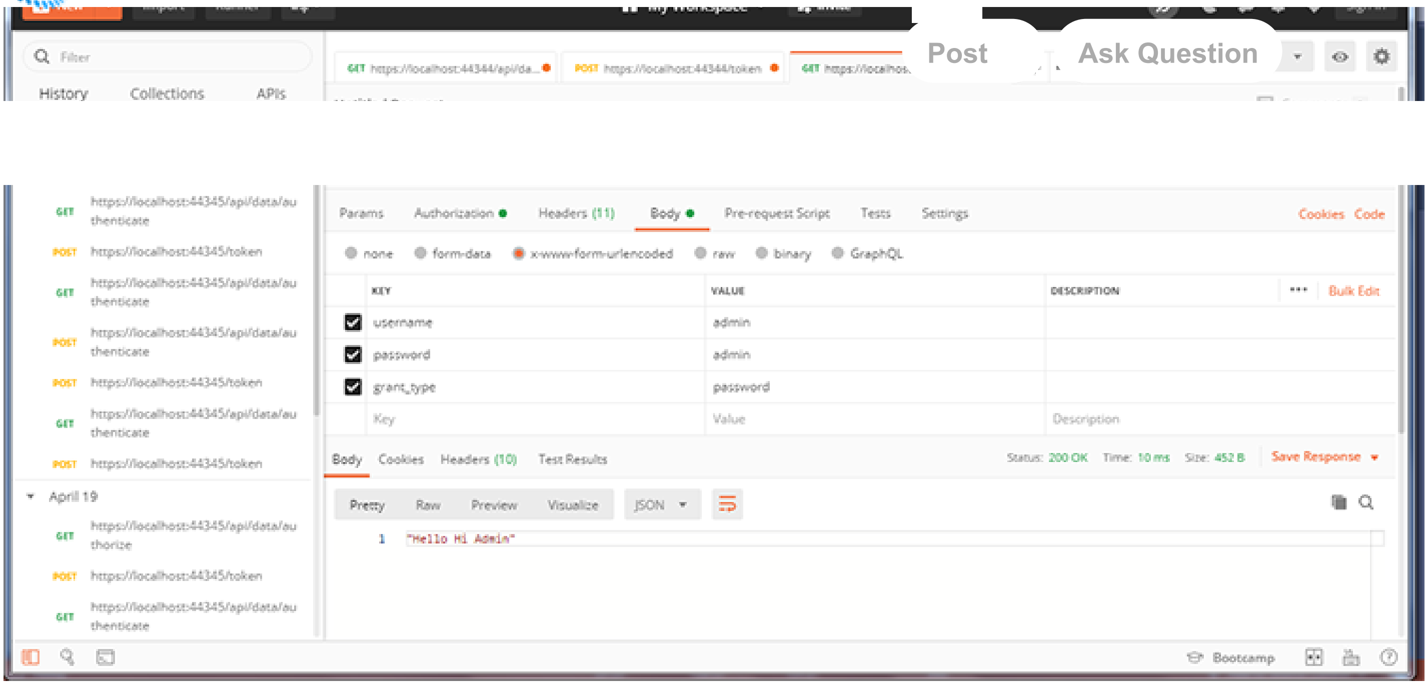
Body: x-www-form-urlencoded

KEY	VALUE	DESCRIPTION
☑ username	user	
☑ password	user	
☑ grant_type	password	

Status: 200 OK Time: 9 ms Size: 451 B

```
"Hello Hi User"
```

Similarly, we can access the action as "admin" by generating token as admin and then using it.



Filter
History
Collections
APIs
GET https://localhost:44344/api/da...
POST https://localhost:44344/token
GET https://localhos...
GET https://localhost:44345/api/data/authenticate
POST https://localhost:44345/token
GET https://localhost:44345/api/data/authenticate
POST https://localhost:44345/api/data/authenticate
POST https://localhost:44345/token
GET https://localhost:44345/api/data/authenticate
POST https://localhost:44345/token
April 19
GET https://localhost:44345/api/data/authorize
POST https://localhost:44345/token
GET https://localhost:44345/api/data/authenticate
Params
Authorization
Headers (11)
Body
Pre-request Script
Tests
Settings
Cookies
Code
Bulk Edit
Body
Cookies
Headers (10)
Test Results
Save Response
Pretty
Raw
Preview
Visualize
JSON
Bootcamp

○ none ○ form-data ● x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	KEY	VALUE	DESCRIPTION
☑	username	admin	
☑	password	admin	
☑	grant_type	password	
	Key	Value	Description

Status: 200 OK Time: 10 ms Size: 452 B

```
1   "Hello Hi Admin"
```

In the same way, the other actions can be accessed by user or admin depending on the way the token is generated.

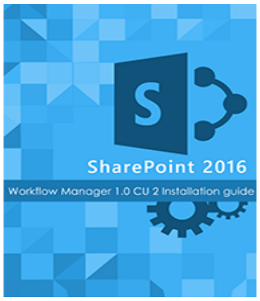
Authentication Authoriation OAUTH OWIN OWIN OAuth Token

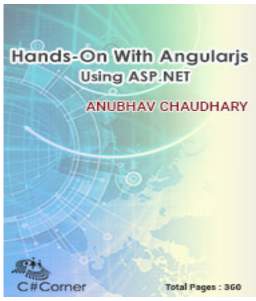
Web Api Token Authentication

Next Recommended Article

Token Based Authentication Using ASP.Net Web API, OWIN and Identity With Entity Framework

OUR BOOKS





Gaurav Karroy

Working on Microsoft Technologies more than 5 year

1849 37.5k

11 6

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

How to add expiry time of token or validation one token can use only one time

sourabh singh Oct 21, 2020

1971 3 0 0 0 Reply

I had generated the token. however when I select Authorization as OAuth2.0, I did not see generated token to there

Dattatray Arsule Jul 22, 2020

1956 18 0 0 1 Reply

Hi Dattatray, can please mail me in details in codegroom@gmail.com regarding the problem you are facing.

Gaurav Karroy Aug 15, 2020

1849 125 37.5k 0

When I select the GET, body section disabled. How it is enabled in you screenshot?

Dattatray Arsule Jul 22, 2020

1956 18 0 0 0 Reply

Just a tip, run your example code through StyleCop to make sure it adhears to common coding standards. For example this is incorrect casing for a class name: public class APIAUTHORIZATIONSERVERPROVIDER.

David Mccarter Jun 08, 2020

117 18.8k 870.7k 1 1 Reply

Hi David. Thanks for your tip. It will be surely used for the next article. Thanks.

Gaurav Karroy Aug 15, 2020

1849 125 37.5k 1

FEATURED ARTICLES

Getting Started Wth MongoDb Atlas

What Is a Full Stack Developer

View All

TRENDING UP

01 Clean Architecture End To End In .NET 5

02 How To Add A Document Viewer In Angular 10

03 Prototype Design Pattern With Java

04 Getting Started With Azure Service Bus Queues And ASP.NET Core - Part 1

05 C# 9 Cheat Sheet

06 Use Entity Framework Core 5.0 In .NET Core 3.1 With MySQL Database By Code-First Migration On Visual Studio 2019 For RESTful API Application

07 What Is a Full Stack Developer

08 Setup A Billing Alarm In AWS

09 Six Types Of Regression | Detailed Explanation

10 Getting Started With Amazon Redshift

View All