Token Based Authentication using ASP.NET Web API 2, Owin, and Identity

June 1, 2014 By Taiseer Joudeh — 1,450 Comments

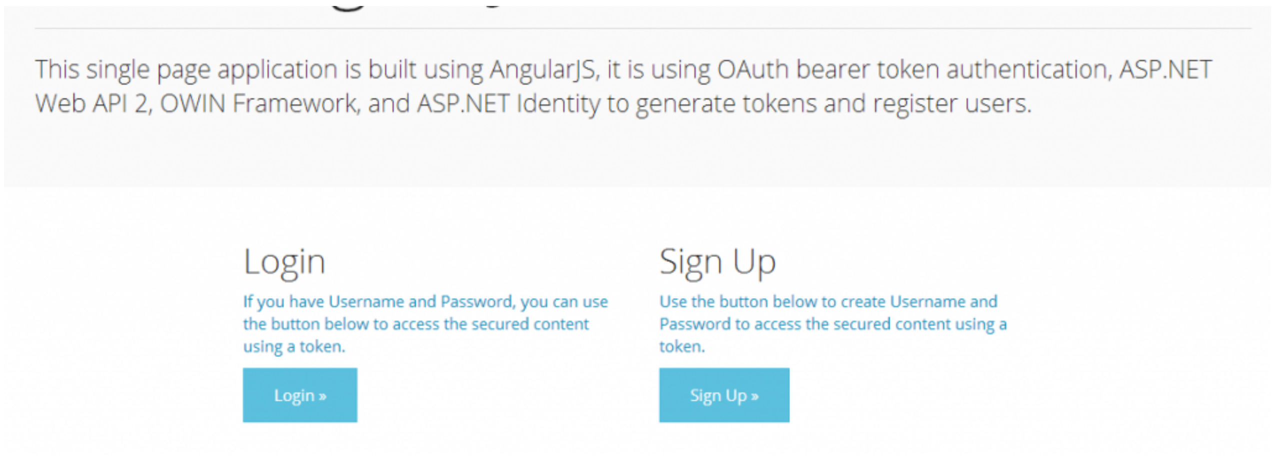
Last week I was looking at the top viewed posts on my blog and I noticed that visitors are interested in the authentication part of ASP.NET Web API, CORS Support, and how to authenticate users in single page applications built with AngularJS using token based approach.

So I decided to compile mini tutorial of ~~three~~ five posts which covers and connects those topics. In this tutorial we'll build SPA using AngularJS for the front-end, and ASP.NET Web API 2, Owin middleware, and ASP.NET Identity for the back-end.

- AngularJS Token Authentication using ASP.NET Web API 2, Owin, and ASP.NET Identity – Part 2.
- Enable OAuth Refresh Tokens in AngularJS App using ASP .NET Web API 2, and Owin – Part 3.
- ASP.NET Web API 2 external logins with Facebook and Google in AngularJS app – Part 4.
- Decouple OWIN Authorization Server from Resource Server – Part 5.

- **New post Aug-2016**: Secure ASP.NET Web API 2 Using Azure AD B2C (Business to Consumer).
- New post: Two Factor Authentication in ASP.NET Web API & AngularJS using Google Authenticator.

The demo application can be accessed on (http://ngAuthenticationWeb.azurewebsites.net). The back-end API can be accessed on (http://ngAuthenticationAPI.azurewebsites.net/) and both are hosted on Microsoft Azure, for learning purposes feel free to integrate and play with the back-end API with your front-end application. The API supports CORS and accepts HTTP calls from any origin. You can check the source code for this tutorial on Github.


This single page application is built using AngularJS, it is using OAuth bearer token authentication, ASP.NET Web API 2, OWIN Framework, and ASP.NET Identity to generate tokens and register users.

Login
If you have Username and Password, you can use the button below to access the secured content using a token.
Login »
Sign Up
Use the button below to create Username and Password to access the secured content using a token.
Sign Up »

Token Based Authentication

As I stated before we'll use token based approach to implement authentication between the front-end application and the back-end API, as we all know the common and old way to implement authentication is the cookie-based approach were the cookie is sent with each request from the client to the server, and on the server it is used to identify the authenticated user.

With the evolution of front-end frameworks and the huge change on how we build web applications nowadays the preferred approach to authenticate users is to use signed token as this token sent to the server with each request, some of the benefits for using this approach are:

- **Scalability of Servers:** The token sent to the server is self contained which holds all the user information needed for authentication, so adding more servers to your web farm is an easy task, there is no dependent on shared session stores.
- **Loosely Coupling:** Your front-end application is not coupled with specific authentication mechanism, the token is generated from the server and your API is built in a way to understand this token and do the authentication.
- **Mobile Friendly:** Cookies and browsers like each other, but storing cookies on native platforms (Android, iOS, Windows Phone) is not a trivial task, having standard way to authenticate users will simplify our life if we decided to consume the back-end API from native applications.

What we'll build in this tutorial?

The front-end SPA will be built using HTML5, AngularJS, and Twitter Bootstrap. The back-end server will be built using ASP.NET Web API 2 on top of Owin middleware not directly on top of ASP.NET; the reason for doing so that we'll configure the server to issue OAuth bearer token authentication using Owin middleware too, so setting up everything on the same pipeline is better approach. In addition to this we'll use ASP.NET Identity system which is built on top of Owin middleware and we'll use it to registe new users and validate their credentials before generating the tokens.

الخصوصية - البنود

- Allow users to signup (register) by providing username and password then store credentials in secure medium.
- Prevent anonymous users from viewing secured data or secured pages (views).
- Once the user is logged in successfully, the system should not ask for credentials or re-authentication for the next ~~24 hours~~ 30 minutes because we are using refresh tokens.

So in this post we'll cover step by step how to build the back-end API, and on the next post we'll cover how we'll build and integrate the SPA with the API.

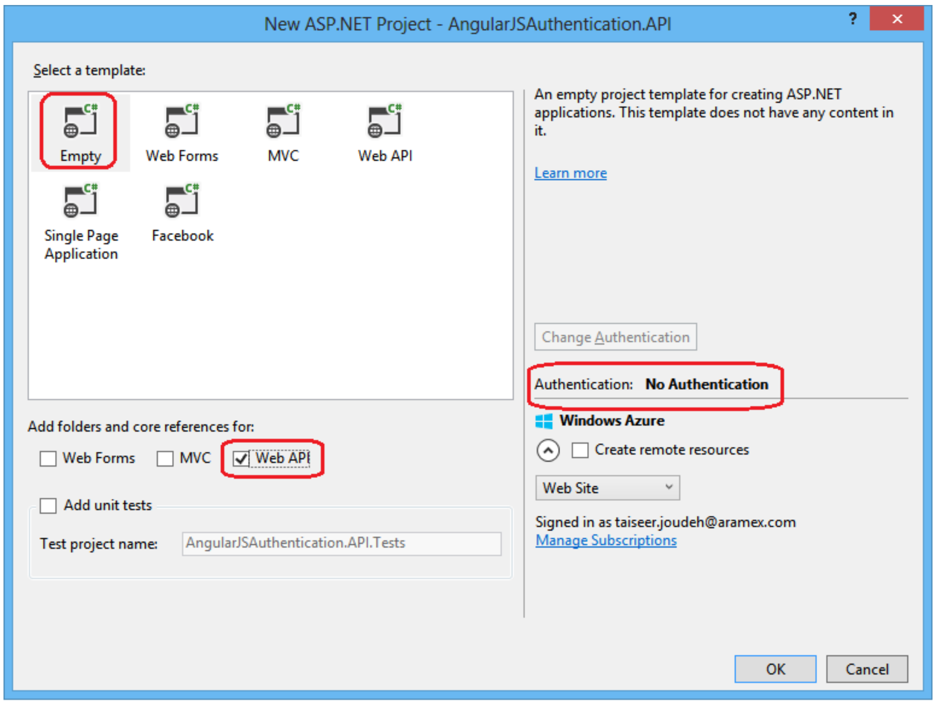
Enough theories let's get our hands dirty and start implementing the API!

Building the Back-End API

Step 1: Creating the Web API Project

In this tutorial I'm using Visual Studio 2013 and .Net framework 4.5, you can follow along using Visual Studio 2012 but you need to install Web Tools 2013.1 for VS 2012 by visiting this link.

Now create an empty solution and name it "AngularJSAuthentication" then add new ASP.NET Web application named "AngularJSAuthentication.API", the selected template for project will be as the image below. Notice that the authentication is set to "No Authentication" taking into consideration that we'll add this manually.


New ASP.NET Project - AngularJSAuthentication.API

Select a template: [✓ Empty] Web Forms MVC Web API Single Page Application Facebook

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

Learn more

Change Authentication

Authentication: No Authentication

Windows Azure

☐ Create remote resources

Web Site

Signed in as taiseer.joudeh@aramex.com

Manage Subscriptions

Add folders and core references for:

☐ Web Forms ☐ MVC ☑ Web API

☐ Add unit tests

Test project name: AngularJSAuthentication.API.Tests

OK Cancel

Step 2: Installing the needed NuGet Packages:

```
Install-Package Microsoft.AspNet.WebApi.Owin -Version 5.1.2
Install-Package Microsoft.Owin.Host.SystemWeb -Version 2.1.0
```

The package "Microsoft.Owin.Host.SystemWeb" is used to enable our Owin server to run our API on IIS using ASP.NET request pipeline as eventually we'll host this API on Microsoft Azure Websites which uses IIS.

Step 3: Add Owin "Startup" Class

Right click on your project then add new class named "Startup". We'll visit this class many times and modify it, for now it will contain the code below:

```
using Microsoft.Owin;
using Owin;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;

[assembly: OwinStartup(typeof(AngularJSAuthentication.API.Startup))]
namespace AngularJSAuthentication.API
{
    public class Startup
    {
        public void Configuration(IAppBuilder app)
        {
            HttpConfiguration config = new HttpConfiguration();
            WebApiConfig.Register(config);
            app.UseWebApi(config);
        }

    }
}
```

What we've implemented above is simple, this class will be fired once our server starts, notice the "assembly" attribute which states which class to fire on start-up. The "Configuration" method accepts parameter of type "IAppBuilder" this parameter will be supplied by the host at run-time. This "app" parameter is an interface which will be used to compose the application for our Owin server.

The "HttpConfiguration" object is used to configure API routes, so we'll pass this object to method "Register" in "WebApiConfig" class.

Lastly, we'll pass the "config" object to the extension method "UseWebApi" which will be responsible to wire up ASP.NET Web API to our Owin server pipeline.

Usually the class "WebApiConfig" exists with the templates we've selected, if it doesn't exist then add it under the folder "App_Start". Below is the code inside it:

```
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {

            // Web API routes
            config.MapHttpAttributeRoutes();

```

الخصوصية

```
    }
```

Step 4: Delete Global.asax Class

No need to use this class and fire up the Application_Start event after we've configured our "Startup" class so feel free to delete it.

Step 5: Add the ASP.NET Identity System

After we've configured the Web API, it is time to add the needed NuGet packages to add support for registering and validating user credentials, so open package manager console and add the below NuGet packages:

```
Install-Package Microsoft.AspNet.Identity.Owin -Version 2.0.1
Install-Package Microsoft.AspNet.Identity.EntityFramework -Version 2.0.1
```

The first package will add support for ASP.NET Identity Owin, and the second package will add support for using ASP.NET Identity with Entity Framework so we can save users to SQL Server database.

Now we need to add Database context class which will be responsible to communicate with our database, so add new class and name it "AuthContext" then paste the code snippet below:

```
public class AuthContext : IdentityDbContext<IdentityUser>
    {
        public AuthContext()
            : base("AuthContext")
        {

        }
    }
```

As you can see this class inherits from "IdentityDbContext" class, you can think about this class as special version of the traditional "DbContext" Class, it will provide all of the Entity Framework code-first mapping and DbSet properties needed to manage the identity tables in SQL Server. You can read more about this class on Scott Allen Blog.

Now we want to add "UserModel" which contains the properties needed to be sent once we register a user, this model is POCO class with some data annotations attributes used for the sake of validating the registration payload request. So under "Models" folder add new class named "UserModel" and paste the code below:

```
public class UserModel
    {
        [Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
```

Now we need to add new connection string named “AuthContext” in our Web.Config class, so open you web.config and add the below section:

```
<connectionStrings>
    <add name="AuthContext" connectionString="Data Source=.\sqlexpress;Initial Catalog=AngularJSAuth;Integrated Securit
  </connectionStrings>
```

Step 6: Add Repository class to support ASP.NET Identity System

Now we want to implement two methods needed in our application which they are: “RegisterUser” and “FindUser”, so add new class named “AuthRepository” and paste the code snippet below:

```
    public class AuthRepository : IDisposable
    {
        private AuthContext _ctx;

        private UserManager<IdentityUser> _userManager;

        public AuthRepository()
        {
            _ctx = new AuthContext();
            _userManager = new UserManager<IdentityUser>(new UserStore<IdentityUser>(_ctx));
        }

        public async Task<IdentityResult> RegisterUser(UserModel userModel)
        {
            IdentityUser user = new IdentityUser
            {
                UserName = userModel.UserName
            };

            var result = await _userManager.CreateAsync(user, userModel.Password);

            return result;
        }

        public async Task<IdentityUser> FindUser(string userName, string password)
        {
            IdentityUser user = await _userManager.FindAsync(userName, password);

            return user;
        }

        public void Dispose()
        {
            _ctx.Dispose();
            _userManager.Dispose();

        }
    }
```

What we’ve implemented above is the following: we are depending on the “UserManager” that provides the domain logic for working with user information. The “UserManager” knows when to hash a password, how and when to validate a user, and how to manage claims. You can read more about ASP.NET Identity System.

Step 7: Add our “Account” Controller

Now it is the time to add our first Web API controller which will be used to register new users, so under f “Controllers” add Empty Web API 2 Controller named “AccountController” and paste the code below:

```
            _repo = new AuthRepository();
        }

        // POST api/Account/Register
        [AllowAnonymous]
        [Route("Register")]
        public async Task<IHttpActionResult> Register(UserModel userModel)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            IdentityResult result = await _repo.RegisterUser(userModel);

            IHttpActionResult errorResult = GetErrorResult(result);

            if (errorResult != null)
            {
                return errorResult;
            }

            return Ok();
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                _repo.Dispose();
            }

            base.Dispose(disposing);
        }

        private IHttpActionResult GetErrorResult(IdentityResult result)
        {
            if (result == null)
            {
                return InternalServerError();
            }

            if (!result.Succeeded)
            {
                if (result.Errors != null)
                {
                    foreach (string error in result.Errors)
                    {
                        ModelState.AddModelError("", error);
                    }
                }

                if (ModelState.IsValid)
                {
                    // No ModelState errors are available to send, so just return an empty BadRequest.
                    return BadRequest();
                }

                return BadRequest(ModelState);
            }

            return null;
        }
    }
```

By looking at the “Register” method you will notice that we’ve configured the endpoint for this method to be “/api/account/register” so any user wants to register into our system must issue HTTP POST request to this URI and the pay load for this request will contain the JSON object as below:

```
{
```

point: http://ngauthenticationapi.azurewebsites.net/api/account/register if all went fine you will receive HTTP status code 200 and the database specified in connection string will be created automatically and the user will be inserted into table “dbo.AspNetUsers”.

Note: It is very important to send this POST request over HTTPS so the sensitive information get encrypted between the client and the server.

The “GetErrorResult” method is just a helper method which is used to validate the “UserModel” and return the correct HTTP status code if the input data is invalid.

Step 8: Add Secured Orders Controller

Now we want to add another controller to serve our Orders, we’ll assume that this controller will return orders only for Authenticated users, to keep things simple we’ll return static data. So add new controller named “OrdersController” under “Controllers” folder and paste the code below:

```
[RoutePrefix("api/Orders")]
    public class OrdersController : ApiController
    {
        [Authorize]
        [Route("")]
        public IHttpActionResult Get()
        {
            return Ok(Order.CreateOrders());
        }

    }

    #region Helpers

    public class Order
    {
        public int OrderID { get; set; }
        public string CustomerName { get; set; }
        public string ShipperCity { get; set; }
        public Boolean IsShipped { get; set; }

        public static List<Order> CreateOrders()
        {
            List<Order> OrderList = new List<Order>
            {
                new Order {OrderID = 10248, CustomerName = "Taiseer Joudeh", ShipperCity = "Amman", IsShipped = true }
                new Order {OrderID = 10249, CustomerName = "Ahmad Hasan", ShipperCity = "Dubai", IsShipped = false},
                new Order {OrderID = 10250,CustomerName = "Tamer Yaser", ShipperCity = "Jeddah", IsShipped = false },
                new Order {OrderID = 10251,CustomerName = "Lina Majed", ShipperCity = "Abu Dhabi", IsShipped = false},
                new Order {OrderID = 10252,CustomerName = "Yasmeen Rami", ShipperCity = "Kuwait", IsShipped = true}
            };

            return OrderList;
        }
    }

    #endregion
```

Notice how we added the “Authorize” attribute on the method “Get” so if you tried to issue HTTP GET request to the end point “http://localhost:port/api/orders” you will receive HTTP status code 401 unauthorized because the request you send till this moment doesn’t contain valid authorization header. Y can check this using this end point: http://ngauthenticationapi.azurewebsites.net/api/orders

الخصوصية - البنود

```
Install-Package Microsoft.Owin.Security.OAuth -Version 2.1.0
```

After you install this package open file "Startup" again and call the new method named "ConfigureOAuth" as the first line inside the method "Configuration", the implemntation for this method as below:

```
public class Startup
    {
        public void Configuration(IAppBuilder app)
        {
            ConfigureOAuth(app);
        //Rest of code is here;
        }

        public void ConfigureOAuth(IAppBuilder app)
        {
            OAuthAuthorizationServerOptions OAuthServerOptions = new OAuthAuthorizationServerOptions()
            {
                AllowInsecureHttp = true,
                TokenEndpointPath = new PathString("/token"),
                AccessTokenExpireTimeSpan = TimeSpan.FromDays(1),
                Provider = new SimpleAuthorizationServerProvider()
            };

            // Token Generation
            app.UseOAuthAuthorizationServer(OAuthServerOptions);
            app.UseOAuthBearerAuthentication(new OAuthBearerAuthenticationOptions());

        }
    }
```

Here we've created new instance from class "OAuthAuthorizationServerOptions" and set its option as the below:

- The path for generating tokens will be as :"http://localhost:port/token". We'll see how we will issue HTTP POST request to generate token in the next steps.
- We've specified the expiry for token to be 24 hours, so if the user tried to use the same token for authentication after 24 hours from the issue time, his request will be rejected and HTTP status code 401 is returned.
- We've specified the implementation on how to validate the credentials for users asking for tokens in custom class named "SimpleAuthorizationServerProvider".

Now we passed this options to the extension method "UseOAuthAuthorizationServer" so we'll add the authentication middleware to the pipeline.

Step 10: Implement the "SimpleAuthorizationServerProvider" class

Add new folder named "Providers" then add new class named "SimpleAuthorizationServerProvider", paste the code snippet below:

```
public class SimpleAuthorizationServerProvider : OAuthAuthorizationServerProvider
    {
        public override async Task ValidateClientAuthentication(OAuthValidateClientAuthenticationContext context)
        {
            context.Validated();
        }
```

الخصوصية

```
                IdentityUser user = await _repo.FindUser(context.UserName, context.Password);

                if (user == null)
                {
                    context.SetError("invalid_grant", "The user name or password is incorrect.");
                    return;
                }
            }

            var identity = new ClaimsIdentity(context.Options.AuthenticationType);
            identity.AddClaim(new Claim("sub", context.UserName));
            identity.AddClaim(new Claim("role", "user"));

            context.Validated(identity);

        }
    }
```

As you notice this class inherits from class “OAuthAuthorizationServerProvider”, we’ve overridden two methods “ValidateClientAuthentication” and “GrantResourceOwnerCredentials”. The first method is responsible for validating the “Client”, in our case we have only one client so we’ll always return that its validated successfully.

The second method “GrantResourceOwnerCredentials” is responsible to validate the username and password sent to the authorization server’s token endpoint, so we’ll use the “AuthRepository” class we created earlier and call the method “FindUser” to check if the username and password are valid.

If the credentials are valid we’ll create “ClaimsIdentity” class and pass the authentication type to it, in our case “bearer token”, then we’ll add two claims (“sub”,”role”) and those will be included in the signed token. You can add different claims here but the token size will increase for sure.

Now generating the token happens behind the scenes when we call “context.Validated(identity)”.

To allow CORS on the token middleware provider we need to add the header “Access-Control-Allow-Origin” to Owin context, if you forget this, generating the token will fail when you try to call it from your browser. Not that this allows CORS for token middleware provider not for ASP.NET Web API which we’ll add on the next step.

Step 11: Allow CORS for ASP.NET Web API

First of all we need to install the following NuGet package manger, so open package manager console and type:

```
Install-Package Microsoft.Owin.Cors -Version 2.1.0
```

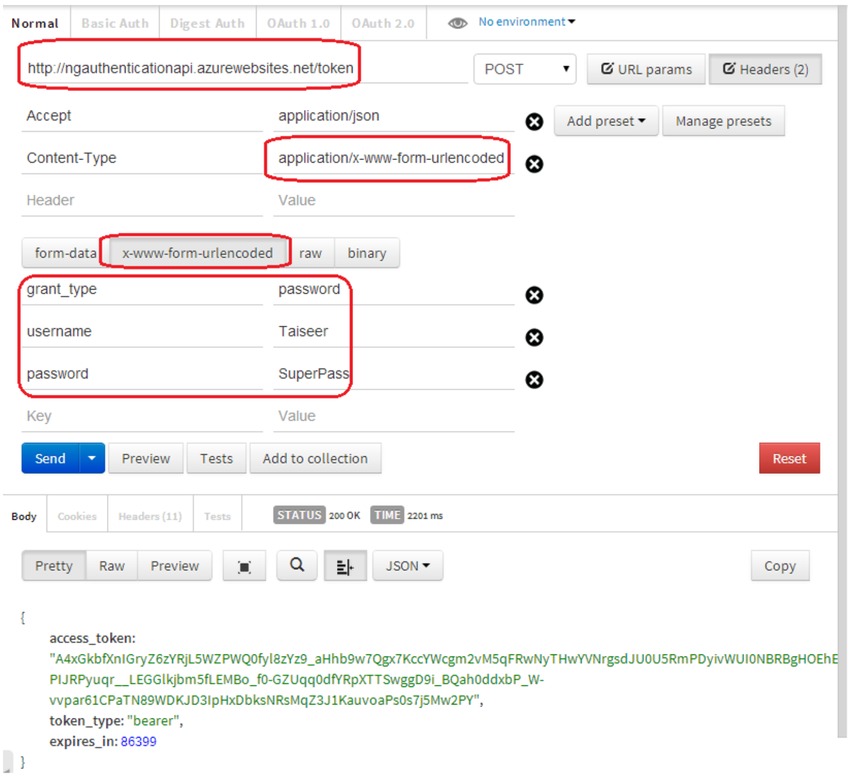
Now open class “Startup” again and add the highlighted line of code (line 8) to the method “Configuration” as the below:

```
1  public void Configuration(IAppBuilder app)
2          {
3              HttpConfiguration config = new HttpConfiguration();
4
```

Step 12: Testing the Back-end API

Assuming that you registered the username "Taiseer" with password "SuperPass" in the step below, we'll use the same username to generate token, so to test this out open your favorite REST client application in order to issue HTTP requests to generate token for user "Taiseer". For me I'll be using PostMan.

Now we'll issue a POST request to the endpoint http://ngauthenticationapi.azurewebsites.net/token the request will be as the image below:



http://ngauthenticationapi.azurewebsites.net/token POST

Accept	application/json
Content-Type	application/x-www-form-urlencoded
Header	Value

grant_type	password
username	Taiseer
password	SuperPass
Key	Value

STATUS 200 OK TIME 2201 ms

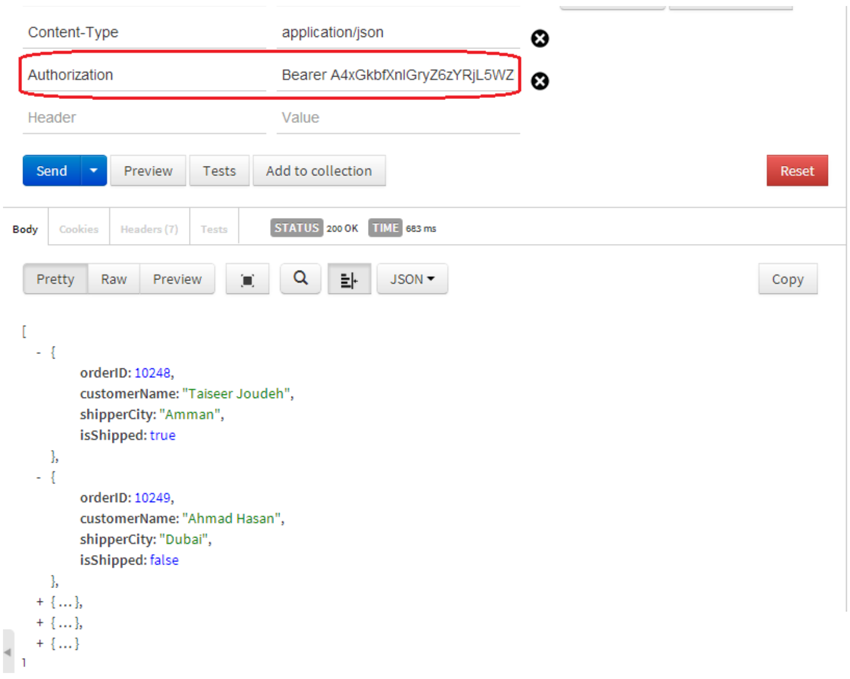
```
{
    access_token:
    "A4xGkbfXnIGryZ6zYRjL5WZPWQ0fyl8zYz9_aHhb9w7Qgx7KccYWcgm2vM5qFRwNyTHwYVNrgsdJU0U5RmPDyivWUI0NBRBgHOEhE
    PIJRPyuqr__LEGGlkjbm5fLEMBo_f0-GZUqq0dfYRpXTTSwggD9i_BQah0ddxbP_W-
    vvpar61CPaTN89WDKJD3IpHxDbksNRsMqZ3J1KauvoaPs0s7j5Mw2PY",
    token_type: "bearer",
    expires_in: 86399
}
```

Notice that the content-type and payload type is "x-www-form-urlencoded" so the payload body will be on form (grant_type=password&username="Taiseer"&password="SuperPass"). If all is correct you'll notice that we've received signed token on the response.

As well the "grant_type" Indicates the type of grant being presented in exchange for an access token, in our case it is password.

Now we want to use this token to request the secure data using the end point http://ngauthenticationapi.azurewebsites.net/api/orders so we'll issue GET request to the end point and will pass the bearer token in the Authorization header, so for any secure end point we've to pass this bearer token along with each request to authenticate the user.

Note: that we are not transferring the username/password as the case of Basic authentication.



```
[
  - {
        orderID: 10248,
        customerName: "Taiseer Joudeh",
        shipperCity: "Amman",
        isShipped: true
    },
  - {
        orderID: 10249,
        customerName: "Ahmad Hasan",
        shipperCity: "Dubai",
        isShipped: false
    },
  + {...},
  + {...},
  + {...}
]
```

If all is correct we'll receive HTTP status 200 along with the secured data in the response body, if you try to change any character with signed token you directly receive HTTP status code 401 unauthorized.

Now our back-end API is ready to be consumed from any front end application or native mobile app.

Update (2014-08-11) Thanks for Attila Hajdrik for forking my repo and updating it to use MongoDb instead of Entity Framework, you can check it here.

You can check the demo application, play with the back-end API for learning purposes (http://ngauthenticationapi.azurewebsites.net), and check the source code on Github.

Follow me on Twitter @tjoudeh

References

- 10 Things You Should Know about Tokens by Matias Woloski
- SPA Authentication Example by David Antaramian

Be Sociable, Share!

Share 3 Tweet SHARE Email WhatsApp

Related Posts

Integrate Azure AD B2C with ASP.NET MVC Web App – Part 3
Secure ASP.NET Web API 2 using Azure AD B2C – Part 2
Azure Active Directory B2C Overview and Policies Management – Part 1

الخصوصية - البنود

Comments

Sok Heang says
July 25, 2017 at 11:23 am

It always response invalid grant type when I post to get the token

Reply

Nipesh says
January 31, 2020 at 4:57 pm

If you add grant_type & credentials in Params then it will give this error.
You need to use > Body > x-www-form-urlencoded > Key Pair then that will work like a charm.
I have also face the same issue initially.

Reply

Amir says
July 27, 2017 at 1:46 am

Hi Taiseer,

Is it possible to use OWIN only to validate the request Bearer token that has been issued out of web api ? In my case the client gets Bearer from Auth0 and I only need to set up OWIN to validate the Token not issuing it. If so can you provide some code snippet or link please?

Reply

Danny says
July 15, 2020 at 2:55 am

Did you ever figure this one out? Similar situation here, the bearer token is created in another app, and we just need to validate it for Web Api.

Reply

Syed Abiddin says
August 2, 2017 at 10:05 pm

Great job, please keep up the good work.

Reply

btduoc71 says
August 8, 2017 at 6:06 am

Why not having logout?

Reply

Alex says
August 13, 2017 at 11:46 pm

Great tutorial, thanks so much!

Only feedback I could give would be that there's a version / dependency issue with installing all the shown versions of the NuGet packages, but choosing newer versions seems to work fine.

And I was a bit confused by this:

//Rest of code is here;

Hoping one of the next tutorials will include how to use your own tables for user storage, using these ASPNET tables just doesn't have that real-world feel. Do production systems really use these default tables?

Reply

Paul says
August 16, 2017 at 3:34 pm

Hello Taiseer,
Nice article !

I have a doubt that you haven't mentioned the user role in authorize attribute i.e "user" (but u have added it in claim).

Tino says
August 17, 2017 at 3:50 pm

Great post. I have one question.

What is the preferred way of token based authentication when publishing your API on Azure API Management Portal? Azure API Management is by default having a key based authentication. Should we have the authentication at two levels(one at the azure portal and another within code) ?

— Tino

Reply

Bryan says
August 18, 2017 at 3:26 am

Thank you for your post.
It helped a lot.
But one thing I can't find out myself.
Where does this owin/asp.net identity save token informations. I think there should be aspNetUser, aspNetRole, and the like..
I guess my currently working server works well. but I can't get where it handle all login related stuffs.
I have no those tables in my database even in master, msdb, model, temp db (MSSQL 2008)
I suppose that entity framework maybe is doing all those things.

Reply

Tadeusz says
August 19, 2017 at 3:58 pm

How to get UserID (UserName) from token? I want to return orders assigned to signed in user name.

Reply

Mario Guajardo says

can you help me with this issue.

Thanks in advance.

Reply

Vaibhav says
September 19, 2019 at 1:17 pm

I had some analysis on this, Here's my finding:

The ConfigureOAuth needs to be initialized in Startup.cs -> Configuration method.

Change the following (Woking):

```
public void Configuration(IAppBuilder app)
{
HttpConfiguration config = new HttpConfiguration();
WebApiConfig.Register(config);
//app.UseCors(Microsoft.Owin.Cors.CorsOptions.AllowAll);
app.UseWebApi(config);

ConfigureOAuth(app); // This needs to be added
}
```

Reply

shekar says
August 31, 2017 at 10:32 am

sir as u said above i implemented.i have one question. how to invalidate token before expires ? because am giving 15 days as expire time.

Reply

gate says
September 6, 2017 at 5:44 am

الخصوصية - البنود

mohsen says
September 7, 2017 at 10:31 am

where should i login and log out and register user? I hav and mvc app and i used idetity .i login and logout useing cookie in that app.but i have web api controller and i want to write login action in my user contorller how should i login?

Reply

User1 says
September 7, 2017 at 12:22 pm

First of all, thanks for this great tutorial.
I've got a question: How does the service now if the token is valid (is not expired)? Where is this validated?

Reply

Sunny says
September 13, 2017 at 8:54 am

Hi Taiseer,
Thanks for such a nice article.

I am new and don't know how to use it.

I have one more doubt, can I use this code for both web application authentication and for those users which only require API result (like in Postman Chrome)?

What should be the first page to start with as I am getting "HTTP Error 403.14 – Forbidden".

Reply

Hưng Nguyễn Văn says
September 19, 2017 at 1:04 pm

ichungcuhathanh.info says
September 20, 2017 at 5:04 am

Also you can search through foreclosures, auctions, and homes which might be preparing to access foreclosure being
a great supply of bargains. The police will be able to offer you all the information regarding crime
rates. The requirement for real estate will invariably outstrip that of supply and
thereby always fueling a rise in home based prices.

Reply

Ikram says
September 21, 2017 at 9:31 am

Hi Taiseer,

Thank you for great article, it helped a lot.

Reply

Mertovski says
September 27, 2017 at 9:16 am

Dear friend, thanks for the tutorial.

I did come across a CORS problem. The next morning (after screaming at my screen for a couple hours and trying different things) my colleague found out what the issue was.

context.OwinContext.Response.Headers.Add("Access-Control-Allow-Origin", new[] { "*" });

and

app.UseCors(Microsoft.Owin.Cors.CorsOptions.AllowAll);

in Startup.cs were interfering with each other, I suppose it happened because you're allowing CORS twice? Not quite confident about that

But removing the following line solved my problem:

Hooman says
October 3, 2017 at 10:17 am

Hi Taiseer, well done
I just wanted to say thank you
It works like a charm.
Especially I should add that this topic is very rare to find

Reply

Surajit Shah says
October 6, 2017 at 9:02 am

Hi,
Can you please prove the the Angular 2 client side code for loogin.

Reply

Karthik Nagendiran says
October 19, 2017 at 3:44 pm

How can I return user's First name and Last name in the response along with the token details ?

Reply

Mohammed Rashid says
October 24, 2017 at 4:37 pm

Hello Taiseer,
Nice article .
I learned a lot from this.
Now I want to connect to my database and read my table instead of static order detail.
So at which place I have to connect and read my db.

Thanks

Asa chan says
October 26, 2017 at 12:00 pm

Thank you machan(friend)

Reply

RDG says
November 7, 2017 at 3:01 pm

First of all….. Thanq for this great article, really very helpful and even I've successfully implemented in my API, but now i'm in a big confusion with refresh-tokens. Please help me with this Refresh tokens and how to implement it

Reply

Mark Reyes says
November 9, 2017 at 2:09 am

Do you have an Angular 2 version of this token based authorization technique?

Reply

Debbs says
November 16, 2017 at 2:54 pm

Simple and on point. Thanks

Reply

Naveen says
November 20, 2017 at 4:13 pm

Hi,
I followed your post and created application, I was able to register with user name and password, but I'm not able

}

Reply

Jose Guadalupe Rodriguez Flores says
February 13, 2018 at 8:07 pm

hi, your url is bad, if in startup configure TokenEndpointPath = new PathString("/token") then call http://localhost:53227/token no need "Api"

Reply

nandita says
February 28, 2018 at 11:34 pm

I faced the same issue. And it was due to two (very silly) reason.
1) AllowInsecureHttp = true was missing in the intialization of OAuthorizationServerOption
2) ConfigureoAuth(app) was missing in the Configuration(IAppBuilder app).

Hope it helps.

Reply

Nick says
March 7, 2018 at 1:17 am

You have probably realised by now that you should drop the api. http://localhost:53227/token'

Reply

Vinnie says
March 8, 2018 at 7:58 pm

This localhost:53227 should match with your current iis/express settings. Go to project properties, click on web tab on left and on right side, see whether you have set iis as local iis or iis express and replace your url in the code appropriately.

Xibalba says
March 16, 2018 at 4:32 pm

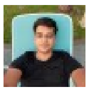
Drop the api/ from your path.

Reply

Mukul says
April 24, 2018 at 4:43 am

You are trying to hit the wrong URL, it will be only http://localhost:*****/token instead of http://localhost:*****/api/token

Reply



Himanshu says
November 23, 2017 at 2:25 pm

Hi,

Thanks for this great article. Still useful in 2017. Ended up using a few other stack overflow answers to implement OWIN token based authentication for our software that does not use EntityFramework. (y)

Reply

Peter says
December 1, 2017 at 1:06 pm

Hello Taiseer,

Thanks, good article!

Reply

الخصوصية - البنود

Thank you for putting this together, this really filled in many holes in my knowledge of WebApi auth and identity.

Reply

Elior says
December 7, 2017 at 12:12 am

Hi,
Thank you very much, your tutorial is a great help for me!
I was able to register a user, but when I try to call the '/token' end point using the following js call, I get back the html of the default file in the 'data' variable … any suggestions?
Thanks !
Elior

this is the js code:

$.ajax({url: "http://localhost:53118/token/",data:
{"username":"Taiseer","password":"SuperPass","grant_type":"password"},cache: false,type: 'post'}).done(function (data) {debugger;});

Reply

Elior says
December 7, 2017 at 12:44 pm

ok, it turns out that I should omit the last slash in the url parameter in the ajax call..

$.ajax({url: "http://localhost:53118/token",data:
{"username":"Taiseer","password":"SuperPass","grant_type":"password"},cache: false,type: 'post'}).done(function (data) {debugger;});

thanks for the great tutorial !

Reply

Tullio Tesorone says
December 10, 2017 at 11:57 am

Hello Taiseer , first chapeau.

الخصوصية - البنود

Tullio

Reply

Vinnie says
March 8, 2018 at 8:01 pm

If you are using postman, check the request params should exactly match with the model variable name, in this case, check the register user model and make sure, both matches.

Hope this solves your problem.

Reply

fabricio says
December 13, 2017 at 9:03 pm

Hi, nice tutorial! One question though, what if I want to add some properties to the user model?
For instance, I want to store address, country or whatever..

Reply

Vinnie says
March 8, 2018 at 8:02 pm

You can add without hesitation, it will work, let me know, in case it doesn't work.

Hope that solves your query.

Reply

Edison says
January 15, 2018 at 12:00 pm

Thank you very much Sir .....:)

Thierry says
January 21, 2018 at 6:48 am

Hi Taiseer,

Great for this very detailed article. Can't wait to start the next part with AngularJS!! 🙂

Quick question: Is there a way to achieve the same without using Entity Framework? I'm not a fan of it to be honest and prefer creating my own data layer using SqlConnection, etc…

Thanks.

Reply

Vinnie says
March 8, 2018 at 8:03 pm

You can very well replace entity framework with your own service that serves bridge between repository and data layer and it will work flawless. Let me know, in case, of any problems you face.

Hope that answers your query.

Reply

Thierry says
November 18, 2018 at 9:21 pm

Hi Vinnie,

Apologies for the late reply but I never got any notification of a reply to my question.

Anyway, I'm glad to hear this but you do have any pointers/links that I can read an investigate some more. I found another very good article which showed how to create the token, add claims to it, etc… and while this part works very well, the article is missing 2 important points how to write this token to a cookie when signing in (not using UserManager/EF/Entity) and how to read and pass the token with every request made?

So I ended up googling things again and found this article again and feel I still don't understand the ins and outs of how to get this to work without using EF.

Any help would be appreciated.

Thanks.

Reply

sujith says
February 5, 2018 at 7:50 pm

have one question about this Security token that you described in this lesson . i followed your method and created same security type token in my live project but if you look at the page source you can see the token .any hakers can take my token and skip login page authentication.

Can you please suggest some other approach so i can implemented in my project.

Reply

Vinnie says
March 8, 2018 at 8:05 pm

You can use JWT instead, which is very well described by Taiseer in subsequent tutorials. Let me know, if you need any help.

Hope that solves your query.

Reply

Ishwor says
February 10, 2018 at 5:07 am

I wonder do we need to set Mime type in the server as .json for file extension and application/json for file type because in my case it shows that 406 NotAcceptable and nothing happens.

Reply

Ishwor says
February 10, 2018 at 5:15 pm

Nice article. I tried all your way but used SimpleMemebrshipProvider instead Asp.Net Identity and I was able to access bearer access token using localhost but was unable in DevServer while testing on PostMan. I always get 500 status code. Can you please help me out ? Thank you.

Reply

and copied your code from the article, but I get a 400 Bad Request after ValidateClientAuthentication is called from authService.js This appears to be CORS-related, but I have verified that the app.UseCors(Microsoft.Owin.Cors.CorsOptions.AllowAll); is in Startup.cs and the context.OwinContext.Response.Headers.Add("Access-Control-Allow-Origin", new[] { "*" }); is in the override GrantResourceOwnerCredentials.

When I debug the code, the app calls the ValidateClientAuthentication method, which has only context.Validated(); in the method. It appears to succeed here, but immediately after, it returns to the client with the 400 message.
There is a second error as follows:

Failed to load http://localhost:46497/token: Response to preflight request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://localhost:2464' is therefore not allowed access. The response had HTTP status code 400. These errors seem CORS-related, but I have followed your instructions and I have verified that there is no other CORS code in the app, as I have seen that this can occur if one enables CORS in multiple places

Reply

Lee1 says
April 29, 2019 at 2:53 pm

I will just leave this for anyone that has had the same issues…

It seems that Moving the line "app.UseCors(Microsoft.Owin.Cors.CorsOptions(.AllowAll);" to being the first line in the Configuration method fixes it for me.

Reply

Greg Knierim says
March 15, 2018 at 5:54 pm

I have implemented this and after upgrading to the latest Nuget packages, it works great.

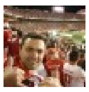
The only question is, is there a way to test the API either by navigating to the URL and entering credentials or from another application (other than the web application) to make sure the responses are as needed?

Thanks,
Greg

Reply

الخصوصية - البنود

Reply


Alexandre Pereira says
March 20, 2018 at 3:01 pm

public override async Task ValidateClientAuthentication(OAuthValidateClientAuthenticationContext context)
{
context.Validated();
}

I am getting an error:
This async method lacks 'await' operators and will run syncchronously.....

Reply

Dave says
November 5, 2018 at 7:59 am

This is not an error but a warning, telling your 'Async' statement is useless unless you set the 'await' for that.

Reply

vvr says
March 24, 2018 at 1:39 pm

Is it possible to use Windows authentication and Authorization using custom tables and then to generate a token?

Reply

Janos says
April 14, 2018 at 11:47 pm

When trying to access the current user's ID by User.Identity.GetUserId() I always get null. Why is that?

الخصوصية - البنود

To answer to the questions asked by myself, in SimpleAuthorizationServerProvider.GrantResourceOwnerCredentials I did the following:

```
public override async Task GrantResourceOwnerCredentials(OAuthGrantResourceOwnerCredentialsContext context)
{

context.OwinContext.Response.Headers.Add(“Access-Control-Allow-Origin”, new[] { “*” });

using (AuthRepository _repo = new AuthRepository())
{
IdentityUser user = await _repo.FindUser(context.UserName, context.Password);

if (user == null)
{
context.SetError(“invalid_grant”, “The user name or password is incorrect.”);
return;
}
else
{
var identity = _repo.CreateIdentity(user, context.Options.AuthenticationType);
identity.AddClaim(new Claim(“sub”, context.UserName));
identity.AddClaim(new Claim(“role”, “user”));

context.Validated(identity);
}
}
}
```

And in AuthRepository:

```
public ClaimsIdentity CreateIdentity(IdentityUser user, string defaultAuthenticationType)
{
var userIdentity = _userManager.CreateIdentity(user, defaultAuthenticationType);

return userIdentity;
}
```

Reply

Harry Vu says
May 5, 2018 at 1:44 am

A great article even to this day (May 2018). Thank you very much.

Reply

Someoby copied your entire article without any attribution: http://www.geeksblood.com/token-based-authentication-using-asp-net-web-api-2-owin-identity/

Reply

Taiseer Joudeh says
May 23, 2018 at 2:37 pm

Thanks Jake for the heads-up, I do not know what to do with guys who steal other efforts, sad story 🙁

Reply

Bp says
May 25, 2018 at 5:40 am

This is really great post. Thanks for breaking this concept.

I implemented this architecture for frontend MVC app (instead of Angular as demoed in this tutorial). It works great

I have a second mvc app which also uses the same web api for the authentication purposes, can SSO (single sign-on) be implemented on these 2 apps with this architecture and be deployed to azure

webapp1.azurewebsites.net -> uses authserver.azurewebsites.net (Web API) for authentication
webapp2.azurewebsites.net -> uses authserver.azurewebsites.net (Web API) for authentication

These both applications have same users and roles. Is SSO Possible?

Any pointers are greatly appreciated

Thanks

Reply

David says
July 17, 2018 at 5:53 pm

Great article, thanks a lot for all the ideas I took from here.

الخصوصية - البنود

I've tried to change the Claim in SimpleAuthorizationServerProvider.GrantResourceOwnerCredentials, but it does not seem to be that …

Do you have any idea of where the problem is ?

Reply

Anshu Dahiya says
July 19, 2018 at 1:59 pm

Had found 1 issue. Please correct if I had done wrong.
1st Create token with user – Abc and Password is abc(suppose).
Use that token to get Orders Detail.
Again create new token with same credentials.
When I try to fetch orders using 1st token , still it return data.
In my opinion,1st token should be invalid for orders request.
Please suggest.

Reply

Jack Lavallet says
August 3, 2018 at 11:42 pm

Hi Taiseer,

I understand how the token is getting created but I'm a bit confused at how the token is used to authenticate the user.

When a request comes in to one of my controllers, with the Bearer token as a header, what exactly performs the Authentication?

One reason I ask is that I would like to tap into this authentication pipeline and set a "LastIndication" property on my User object. Can you think of an elegant way to do that?

Thank you for your consideration.

Reply

nunya says

Reply

Eyal says
September 3, 2018 at 6:05 pm

Hi,
is there a way to keep rolling the expiration on the access token ?
for example, I used the user/pass flow to get access_token and refresh_token.
Ideally if the client keeps making calls i want to roll the expiration on the access token to another X time.
the alternative is actively sending refresh token requests from the client, or a new login request for new set of refresh and access token .
Thanks.

Reply

Jeremy Wells says
September 28, 2018 at 11:20 pm

Great tutorial! I'm working through this in September 2018 and I have installed all the latest packages (mostly at 4.0.0) instead of the versions in this article. I'm not sure what the differences are, but one thing I noticed is I had to change a line in Web.config to get the Owin Middleware to startup. In , there was a key – – that was keeping my api from finding the token generator. when I changed it to everything worked. I checked the repo for this tutorial and saw that this field isn't in the Web.config . I'm not sure why mine is different. Anyway, I thought I'd share this fix in case anyone else runs into a problem with the /token url.

Reply

Kamal Kumar says
October 2, 2018 at 5:28 am

Hi TAISEER,

Thanks for posting this nice article.
Actually i followed the same steps mentioned in this article.
But one problem i am facing.
when i am making call to http://localhost:58901/token using postman, i am getting the response with bearer token and values.

However when i am trying the same thing from angular 4 app i am getting the following issue.
"Failed to load http://localhost:58901/token: No 'Access-Control-Allow-Origin' header is present on the

Thanks in advance

Kamal

Reply

Kumar says
October 3, 2018 at 1:06 pm

Hi

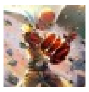
I am login in this application after that i click welcome then click login page this page my page resetting crendentials
how to avoid this

Reply

MAzhar says
October 8, 2018 at 8:55 pm

Please Help how can i get Username , name and role with token at client side ?

Reply



Dorian says
October 11, 2018 at 3:53 pm

When adding an API on a web project which already have a classic authentication system, in the startup configuration function, you have to use

app.Map("/api", inner =>
{
config = new HttpConfiguration();

// this will disable default web authentication (meaning your api requests won't be redirected to login)
config.SuppressDefaultHostAuthentication();
// this line set authentication type to oauth
config.Filters.Add(new HostAuthenticationFilter(OAuthDefaults.AuthenticationType));

Hermes Condez says
October 16, 2018 at 5:34 pm

Hello Taiseer,

How can I configure this in IIS as a subsite. It's parent site is using Windows Authentication.
What is the IIS configuration. Your solution is working without issue in my local machine but when I deploy it in IIS as a subsite, I can't make it work. Please help.

Many thanks in advance.

Hermz

Reply

adeep says
May 6, 2020 at 5:10 pm

Have you found any solution for this. same issue I am also facing in my application.

Reply

Mr R M Watton says
October 17, 2018 at 9:46 pm

Very helpful, everything worked perfectly. I needed authorized endpoints which I could securely access data from using the same login details as for the MVC site. This way the site can be used to create data and a mobile app used to access / change the data securely with the users added to the site when needed.
Everything worked perfectly, I just needed to change IdentityUser to ApplicationUser as this is the name of the class created during setup, which derives from IdentityUser (since ApplicationDbContext derives from IdentityDbContext. Many thanks.

Reply

Mauri Mendoza says
October 18, 2018 at 1:04 am

Bobby says
October 18, 2018 at 6:20 pm

Thank you so much for this thorough and detailed tutorial. Has really helped my understanding of how to create a secure API.

One thing I noticed, the http://localhost:port/api/account/register endpoint doesn't work for my setup. I had remove the api/account/ part to get a 200 status returned. http://localhost:port/register

Hope this will help someone else down the road. Thanks again for the help!

Reply

Taiseer Joudeh says
October 18, 2018 at 9:07 pm

Thanks Bobby, happy to hear post is useful after all this years 🙂

Reply

Shalom says
October 24, 2018 at 11:38 am

Thank you for the great Tutorial. Works!!!

Reply

Dorian says
October 24, 2018 at 4:01 pm

why are comments filtered ?
I posted a useful one some weeks ago and it never showed up

Reply

This is great post .Thanks for writing this article. I followed this article and implemented the customauthprovider.

I was wondering if you would be able to guide me in implemeting ConfigureOAuthTokenConsumption for validating the bearer token similar to what you implemented in your JWT auth implementation.

Thanks again

Reply

BBA Dev says
October 31, 2018 at 2:44 am

Thanks for posting this article. It is still very relevant today after so many years. I have a question regards to verification of Bearer token .. something similar to what you did in your JWT consumption article

https://bitoftech.net/2015/02/16/implement-oauth-json-web-tokens-authentication-in-asp-net-web-api-and-identity-2/

how would you verify the Bearer token that is issued by SimpleAuthorizationServerProvider when the client sends one in request header?

I implemented SimpleAuthorizationServerProvider in various environments(dev,QA) and dev bearer token is also accepted in QA and viceversa.

your inputs will be greatly appreciated.

Reply

Jimi says
November 8, 2018 at 6:31 pm

I got the token generation part working.
But with the generated token in the Authorization header if I make a GET request
http://localhost:/prt/api/orders
with headers:
Content-Type: application/json
Accept: application/json
Authorization: Bearer {my token}

it gives the ouput:

الخصوصية - البنود

Reply

kathiravan says
November 11, 2018 at 11:22 am

Role-based authorization not working how to fix that please share some sample

Reply

Reham Mohamed says
November 14, 2018 at 1:47 pm

Hello, I'd like to thank you about this great and helpful topic 🙂
but I need to know, How can I change the "expires in" attribute of access token ?
I need it for my application

Reply

Christian Sandoval says
November 26, 2018 at 9:23 pm

Hello Taiseer,

Is it possible to add a validation just for application tokens.
I mean, to restrict access to those token where the scope is "api_client"
and not "user".

Thanks

Reply

Jaime says
December 1, 2018 at 2:57 am

Why is it safe to remove Global.asax?

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
RouteConfig.RegisterRoutes(RouteTable.Routes);
BundleConfig.RegisterBundles(BundleTable.Bundles);
}

As you see, applocation is being configured, for example, registering areas, filters, routes and bundles.

While Startup class only configures authentication.

I think it is a bad advice to remove it.

Regards
Jaime

Reply

Green says
December 2, 2018 at 9:59 am

Thanks, the article is very helpful.

But there's one thing still bothering me.
Is there any reason for wrapping UserManager inside AuthRepository ? Can we just use a UserManager directly inside the Controller instead of instantiating an AuthRepository ?

Reply

Jaime says
December 3, 2018 at 11:41 pm

Hello.... I am having a problem calling a Web Api method when user is authenticated.

The fact is that when I call a Web Api method when user is authenticated, it is not recorgnized.

First, I had problem with Logout method in AccountController. I solved it by using AllowAnonymous attribute.

However, now I am facing the same problem with ChangePassword method. I cannot add AllowAnonymous attribute in this case because when I do that I cannot retrieve the logged in user Id.

if I don't add AllowAnonymous attribute, system throws a forbidden error.

I am stuck here.... how can i solve it?

Dharmender says
January 14, 2019 at 2:30 pm

Here from postman we are sending user credentials like username and password? What should I change in the code so that user can pass clientId and client_secret to get the access token

Reply

muhammad farooq says
February 15, 2019 at 12:56 am

i would like to know how we can add the functionality to not only Authorize but add authorization plus also check Role. so in order i would like to say
[Authorize(Roles = “Administrator”)]
can someone please help.

Reply

tc says
March 14, 2019 at 5:56 am

Hi there this is great article. I’m working thru it I seem to have gotten a 404 for http://localhost:56834/token

I can seem to figure out whats happening…
startup file

```
public void ConfigureOAuth(IAppBuilder app)
{
OAuthAuthorizationServerOptions OAuthServerOptions = new OAuthAuthorizationServerOptions()
{
AllowInsecureHttp = true,
TokenEndpointPath = new PathString(“/token”),
AccessTokenExpireTimeSpan = TimeSpan.FromDays(1),
Provider = new SimpleAuthorizationServerProvider()
};
```

webconfig file

…’

Reply

I followed your post and created application, I was able to register with user name and password, but I'm not able
insert data from the database. there are following expection.

" Error The entity type IdentityUser is not part of the model for the current context"

my api: http://localhost:59867/Api/Account/Register
{"UserName":"Nareshrawat""Password":"Naresh@123″,"ConfirmPassword":"Naresh@123″}

Please help me.

Reply

Ethan says
March 26, 2019 at 9:03 am

Hi. Thank you for this excellent article. I have used this code as a starting point. Now I want to implement the authorization code grant, but the OnCreate of my AuthorizationCodeProvider never runs. What needs to happen to make the oauth server call the OnCreate?

Reply

mareeomy says
April 7, 2019 at 4:57 pm

hi so where is the login method ??

Reply

Muhammad Kamran says
September 10, 2020 at 12:35 pm

It's right there in startup api/token here you can authenticate user and get token in return.

Reply

recommended way to deploy the SQL Server environment? The SQL user login appears to need CREATE DATABASE permission. My hosting service would not allow this, what other options do I have instead of changing hosting providers?

Reply

Nirjhar Jain says
June 3, 2019 at 2:11 pm

Hi, Great article. But I need to pass some extra parameters like IP and device Id etc in the context with Username & password. How that can be achieved?

Will those parameters also be the part of x-www-form-urlencoded body like username & password? If Yes, Then How can we get those values on API side?

Reply

John says
June 6, 2019 at 10:12 am

This article series is legendary! Is there a PDF version?

Reply

Mattis says
June 20, 2019 at 2:17 pm

HI,

2019 now and still highly relevant. I want to use this in a new solution I am making for a customer. Everything worked perfectly until i wanted to get the token.

Using Fiddler I tried calling /token on both your API and on my own on localhost, but get:
{“error”:”invalid_grant”,”error_description”:”The user name or password is incorrect.”}

I know the user/password combination is correct, and are able to log in to your demo site with the same user/pass.

In fiddler I do:

Body:
grant_type=password&username="johnny123456"&password="donald"

As mentioned I can log on: http://ngauthenticationweb.azurewebsites.net/#/login using the credentials abowe.

What am I doing wrong – please help me. Really need to get this working. Please please.

Thank you!

Reply

Rutuja says
June 26, 2019 at 4:27 pm

Hi,

Your article helped me a lot. Thanks for that.
I have one query, can we implement two factor authentication in web api using owin and asp .net Identity?
If yes, please guide me. I have already reviewed your article 'https://bitoftech.net/2014/10/15/two-factor-authentication-asp-net-web-api-angularjs-google-authenticator/', but it's not what I'm looking for.

Also, I don't want to implement two factor authentication using phone number, email should be sent to the user with the verification code.

Any help will be appreciated. Thanks.

Reply

Kaushik Rathod says
June 27, 2019 at 9:32 am

How I get 'Authorize Code' from this code using '/authorize' in 'AuthorizeEndpointPath'? and I also want to implement 'Authorization_Code' grant and 'Client_Credential' grant. Please suggest me how to implement it.

Thank you.

Reply

lt says
July 15, 2019 at 11:03 am

Jackie says
July 16, 2019 at 12:40 pm

Hi
Thanks a lot for sharing your knowledge.
How can I set Email service to get a verify link after register a user?

Reply

Bernhard P. says
August 24, 2019 at 9:47 am

top content! thanks

Reply

Sourabh says
September 11, 2019 at 10:39 am

Please share your database structure.

Reply

Srinivas says
October 15, 2019 at 1:23 pm

Hi,

How this token based authentication will work with Load Balance (Two diff servers) ?

What are the options to solve the prob ?

Reply

SLay says

Steve

Reply

Aamer Saeed says
December 25, 2019 at 7:32 pm

If you add namespaces in your code snippet it would be of great help. No doubt a great post and Github source code link is also there but its my suggestion. Another thing is right now I am stuck on Startup.cs class and code line Provider = new SimpleAuthorizationServerProvider giving me error "
Error CS0266 Cannot implicitly convert type 'POSApi.Providers.SimpleAuthorizationServerProvider' to 'Microsoft.Owin.Security.OAuth.IOAuthAuthorizationServerProvider'. An explicit conversion exists (are you missing a cast?)"

Reply

payam says
January 1, 2020 at 12:45 pm

Thank's so much
you learning token base authentication web api in the simplest way in this tutorial

Reply

tejal says
January 20, 2020 at 12:56 pm

Great article. thanks a lot. save my time

Reply

mike says
February 27, 2020 at 6:20 pm

الخصوصية - البنود

Nilanjan says
March 5, 2020 at 7:53 pm

Hi Sir, I have created a back end internal API by following your instructions.
Is it possible to call this API from any console or winsows service which is running under a service account?
How would I generate a token for the service account as the service needs to be running always.
Please shed some lights on this topic.

Reply

BOUZIDI says
July 5, 2020 at 3:30 pm

Hi
Thanks a lot for sharing your knowledge.
[Authorize(Roles = "Admin")] is not working for me 🙁 it always returns a 401

Can you help me please.

Thank you.

Reply

BOUZIDI. says
July 5, 2020 at 3:48 pm

Hi
Thanks a lot for sharing your knowledge.
[Authorize(Roles = "Admin")] is not working for me 🙁 it always returns a 401

Can you help me please.

Thank you.

Reply

Ivan says

Reply

SGA says
September 4, 2020 at 6:10 pm

Thanks a lot! Very good tutorial to migrate quickly to token authentication!!

Reply

Muhammad Kamran says
September 10, 2020 at 12:30 pm

Hi Sir I am following your blog its awesome I have done a bit of modification to change to 3 tier architecture using using dependency pattern and it is working like a charm.

Reply

« Older Comments

Leave a Reply

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

Comment

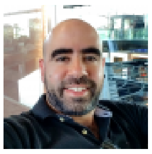
Name *

Email *

الخصوصية - البنود

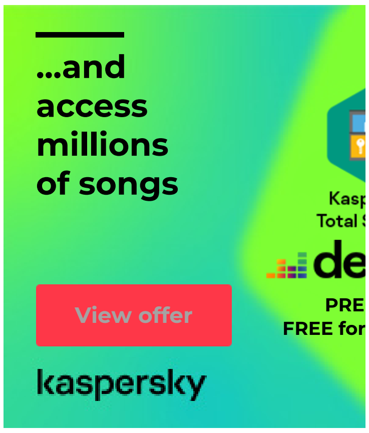
POST COMMENT

ABOUT TAISEER



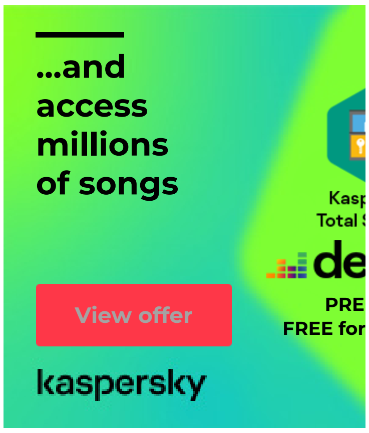
Husband, Father, Consultant @ MSFT, Life Time Learner... Read More...

Buy me a coffee



BIT OF TECHNOLOGY

ARCHIVE ABOUT ME SPEAKING CONTACT



RECENT POSTS

- Integrate Azure AD B2C with ASP.NET MVC Web App – Part 3
- Secure ASP.NET Web API 2 using Azure AD B2C – Part 2
- Azure Active Directory B2C Overview and Policies Management – Part 1
- ASP.NET Web API Claims Authorization with ASP.NET Identity 2.1 – Part 5
- ASP.NET Identity 2.1 Roles Based Authorization with ASP.NET Web API – Part 4

BLOG ARCHIVES

Select Month

RECENT POSTS

- Integrate Azure AD B2C with ASP.NET MVC Web App – Part 3
- Secure ASP.NET Web API 2 using Azure AD B2C – Part 2

TAGS

AJAX AngularJS API API Versioning ASP.NET Authentication Authorization Server Azure Active Directory B2C Azure AD B2C

الخصوصية - البنود

BIT OF TECHNOLOGY

ARCHIVE ABOUT ME SPEAKING CONTACT

ASP.NET Identity 2.1 Roles Based Authorization with ASP.NET Web API – Part 4

Tokens JWT Model Factory Ninject OAuth OData Pagination Resources Association Resource Server REST RESTful Single Page Applications SPA Token Authentication Tutorial Web API Web API 2 Web API Security Web Service wordpress.com

SEARCH

Search this website