

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

June 1, 2014 By [Taiseer Joudeh](#) — [1,383 Comments](#)

Be Sociable, Share!

Share 607

Tweet

SHARE

Post

Email

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.

# BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

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 where 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

# BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

As I mentioned before our back-end API should accept request coming from any origin, not only our front-end, so we'll be enabling CORS ([Cross Origin Resource Sharing](#)) in Web API as well for the OAuth bearer token provider.

## Use cases which will be covered in this application:

- 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.

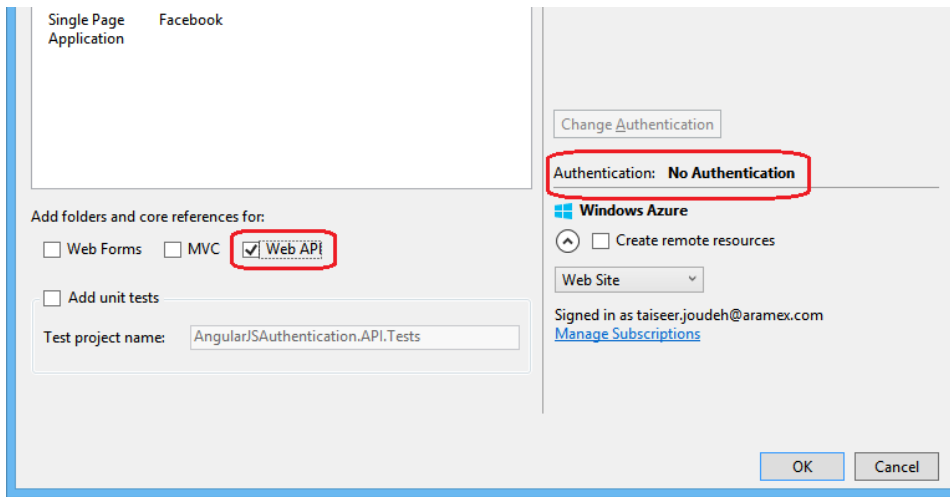
Enough theories let's get our hands dirty and start implementing the API!

## 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.



## Step 2: Installing the needed NuGet Packages:

Now we need to install the NuGet packages which are needed to setup our Owin server and configure ASP.NET Web API to be hosted within an Owin server, so open NuGet Package Manger Console and type the below:

```
1 Install-Package Microsoft.AspNet.WebApi.Owin -Version 5.1.2
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:

```
1 using Microsoft.Owin;
2 using Owin;
3 using System;
4 using System.Collections.Generic;
5 using System.Linq;
6 using System.Web;
7 using System.Web.Http;
8
9 [assembly: OwinStartup(typeof(AngularJSAuthentication.API.Startup))]
10 namespace AngularJSAuthentication.API
11 {
12     public class Startup
13     {
14         public void Configuration(IAppBuilder app)
15         {
16             HttpConfiguration config = new HttpConfiguration();
```

# BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

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:

```
1 public static class WebApiConfig
2 {
3     public static void Register(HttpConfiguration config)
4     {
5
6         // Web API routes
7         config.MapHttpAttributeRoutes();
8
9         config.Routes.MapHttpRoute(
10             name: "DefaultApi",
11             routeTemplate: "api/{controller}/{id}",
12             defaults: new { id = RouteParameter.Optional }
13         );
14
15         var jsonFormatter = config.Formatters.OfType<JsonMediaTypeFormatter>().First();
16         jsonFormatter.SerializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver();
17     }
18 }
```

## 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:

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

# BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

```
1 public class AuthContext : IdentityDbContext<IdentityUser>
2 {
3     public AuthContext()
4         : base("AuthContext")
5     {
6     }
7 }
8 }
```

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:

```
1 public class UserModel
2 {
3     [Required]
4     [Display(Name = "User name")]
5     public string UserName { get; set; }
6
7     [Required]
8     [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]
9     [DataType(DataType.Password)]
10    [Display(Name = "Password")]
11    public string Password { get; set; }
12
13    [DataType(DataType.Password)]
14    [Display(Name = "Confirm password")]
15    [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
16    public string ConfirmPassword { get; set; }
17 }
```

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:

```
1 <connectionStrings>
2     <add name="AuthContext" connectionString="Data Source=.\\sqlexpress;Initial Catalog=AngularJSAuth;Integrated Security=SSPI;" />
3 </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:

# BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

```
8      {
9          _ctx = new AuthContext();
10         _userManager = new UserManager<IdentityUser>(new UserStore<IdentityUser>(_ctx));
11     }
12
13     public async Task<IdentityResult> RegisterUser(UserModel userModel)
14     {
15         IdentityUser user = new IdentityUser
16         {
17             UserName = userModel.UserName
18         };
19
20         var result = await _userManager.CreateAsync(user, userModel.Password);
21
22         return result;
23     }
24
25     public async Task<IdentityUser> FindUser(string userName, string password)
26     {
27         IdentityUser user = await _userManager.FindAsync(userName, password);
28
29         return user;
30     }
31
32     public void Dispose()
33     {
34         _ctx.Dispose();
35         _userManager.Dispose();
36     }
37 }
38 }
```

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 file "Controllers" add Empty Web API 2 Controller named "AccountController" and paste the code below:

```
1  [RoutePrefix("api/Account")]
2  public class AccountController : ApiController
3  {
4      private AuthRepository _repo = null;
5
6      public AccountController()
7      {
8          _repo = new AuthRepository();
9      }
10
11      // POST api/Account/Register
12      [AllowAnonymous]
13      [Route("Register")]
14      public async Task<IHttpActionResult> Register(UserModel userModel)
15      {
16          if (!ModelState.IsValid)
17          {
18              return BadRequest(ModelState);
19          }
20      }
```

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

```
27         return BadRequest();
28     }
29
30     return Ok();
31 }
32
33 protected override void Dispose(bool disposing)
34 {
35     if (disposing)
36     {
37         _repo.Dispose();
38     }
39
40     base.Dispose(disposing);
41 }
42
43 private IHttpActionResult GetErrorResult(IdentityResult result)
44 {
45     if (result == null)
46     {
47         return InternalServerError();
48     }
49
50     if (!result.Succeeded)
51     {
52         if (result.Errors != null)
53         {
54             foreach (string error in result.Errors)
55             {
56                 ModelState.AddModelError("", error);
57             }
58         }
59
60         if (ModelState.IsValid)
61         {
62             // No ModelState errors are available to send, so just return an empty BadRequest.
63             return BadRequest();
64         }
65
66         return BadRequest(ModelState);
67     }
68
69     return null;
70 }
71 }
```

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:

```
1 {
2   "userName": "Taiseer",
3   "password": "SuperPass",
4   "confirmPassword": "SuperPass"
5 }
```

Now you can run your application and issue HTTP POST request to your local URI:

“http://localhost:port/api/account/register” or you can try the published API using this end

point: <http://ngauthenticationapi.azurewebsites.net/api/account/register> if all went fine you will receive



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:

```
1 [RoutePrefix("api/Orders")]
2     public class OrdersController : ApiController
3     {
4         [Authorize]
5         [Route("")]
6         public IHttpActionResult Get()
7         {
8             return Ok(Order.CreateOrders());
9         }
10    }
11
12    #region Helpers
13
14    public class Order
15    {
16        public int OrderID { get; set; }
17        public string CustomerName { get; set; }
18        public string ShipperCity { get; set; }
19        public Boolean IsShipped { get; set; }
20
21        public static List<Order> CreateOrders()
22        {
23            List<Order> OrderList = new List<Order>
24            {
25                new Order {OrderID = 10248, CustomerName = "Taiseer Joudeh", ShipperCity = "Amman", IsShipped = true },
26                new Order {OrderID = 10249, CustomerName = "Ahmad Hasan", ShipperCity = "Dubai", IsShipped = false},
27                new Order {OrderID = 10250, CustomerName = "Tamer Yaser", ShipperCity = "Jeddah", IsShipped = false },
28                new Order {OrderID = 10251, CustomerName = "Lina Majed", ShipperCity = "Abu Dhabi", IsShipped = false},
29                new Order {OrderID = 10252, CustomerName = "Yasmeen Rami", ShipperCity = "Kuwait", IsShipped = true}
30            };
31
32            return OrderList;
33        }
34    }
35
36    #endregion
37
```

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. You can check this using this end point: <http://ngauthenticationapi.azurewebsites.net/api/orders>

## Step 9: Add support for OAuth Bearer Tokens Generation

# BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

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 implementation for this method as below:

```
1 public class Startup
2 {
3     public void Configuration(IApplicationBuilder app)
4     {
5         ConfigureOAuth(app);
6         //Rest of code is here;
7     }
8
9     public void ConfigureOAuth(IApplicationBuilder app)
10    {
11        OAuthAuthorizationServerOptions OAuthServerOptions = new OAuthAuthorizationServerOptions()
12        {
13            AllowInsecureHttp = true,
14            TokenEndpointPath = new PathString("/token"),
15            AccessTokenExpireTimeSpan = TimeSpan.FromDays(1),
16            Provider = new SimpleAuthorizationServerProvider()
17        };
18
19        // Token Generation
20        app.UseOAuthAuthorizationServer(OAuthServerOptions);
21        app.UseOAuthBearerAuthentication(new OAuthBearerAuthenticationOptions());
22
23    }
24 }
```

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:

```
1 public class SimpleAuthorizationServerProvider : OAuthAuthorizationServerProvider
```

## BIT OF TECHNOLOGY

ARCHIVE

ABOUT ME

SPEAKING

CONTACT

```
11 context.OwinContext.Response.Headers.Add("Access-Control-Allow-Origin", new[] { "*" });
12
13 using (AuthRepository _repo = new AuthRepository())
14 {
15     IdentityUser user = await _repo.FindUser(context.UserName, context.Password);
16
17     if (user == null)
18     {
19         context.SetError("invalid_grant", "The user name or password is incorrect.");
20         return;
21     }
22 }
23
24 var identity = new ClaimsIdentity(context.Options.AuthenticationType);
25 identity.AddClaim(new Claim("sub", context.UserName));
26 identity.AddClaim(new Claim("role", "user"));
27
28 context.Validated(identity);
29
30 }
31 }
```

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:

# BIT OF TECHNOLOGY

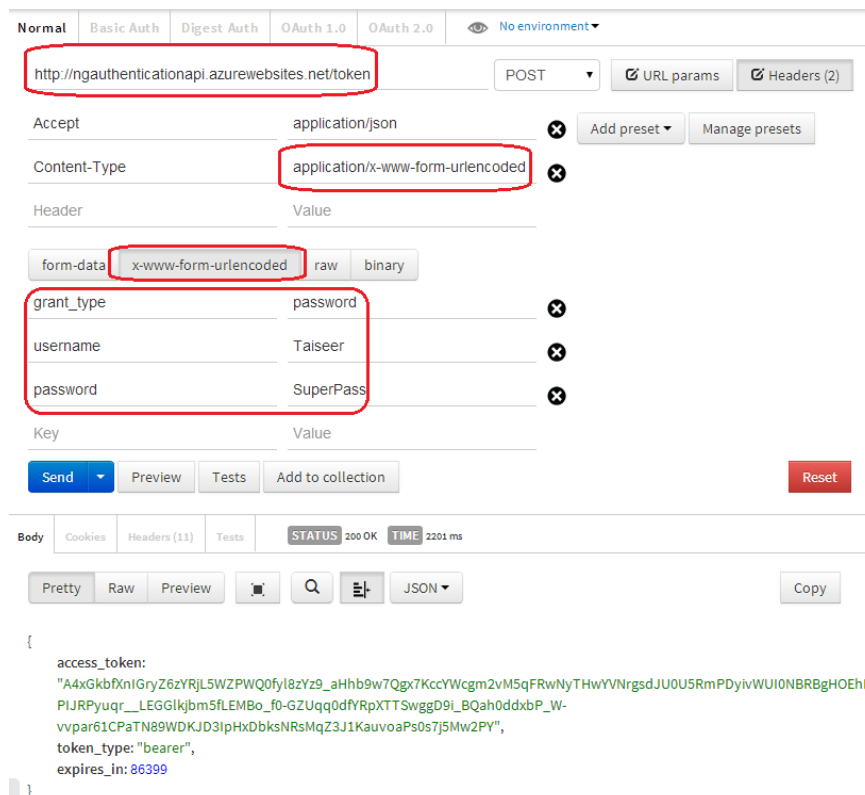
[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

```
1 public void Configuration(IAppBuilder app)
2     {
3         HttpConfiguration config = new HttpConfiguration();
4
5         ConfigureOAuth(app);
6
7         WebApiConfig.Register(config);
8         app.UseCors(Microsoft.Owin.Cors.CorsOptions.AllowAll);
9         app.UseWebApi(config);
10
11     }
```

## 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:



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.

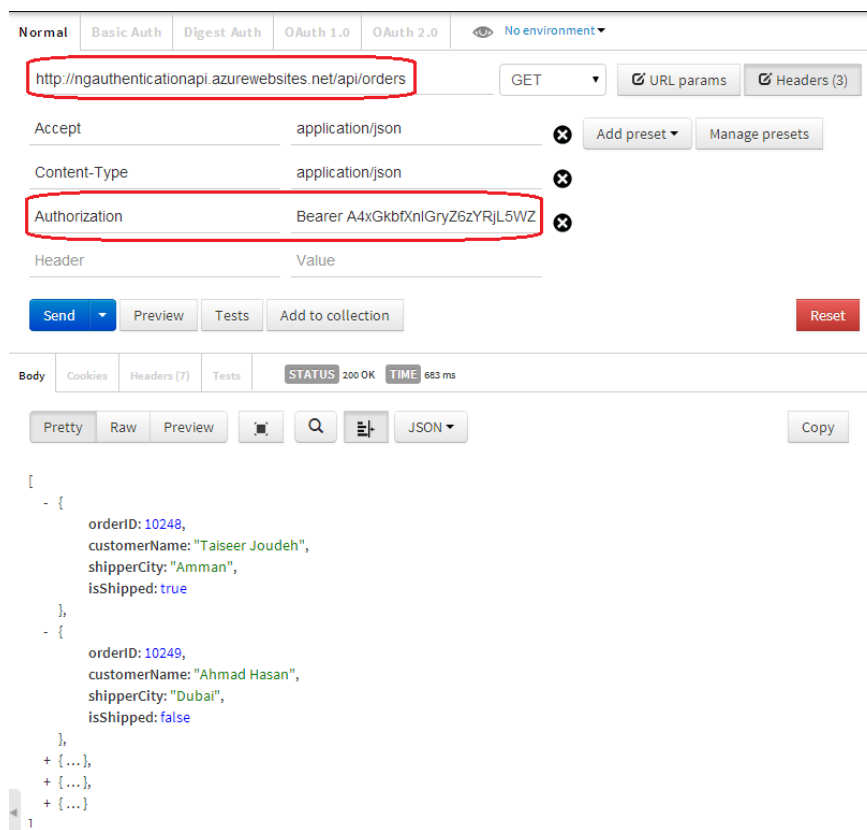
# BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

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.

The GET request will be as the image below:



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.

# BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)[Token Authentication Example](#) by [David Arslan](#)**Be Sociable, Share!**

Share 607

Tweet

SHARE

Post

 Email

## 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](#)[ASP.NET Web API Claims Authorization with ASP.NET Identity 2.1 – Part 5](#)[ASP.NET Identity 2.1 Accounts Confirmation, and Password Policy Configuration – Part 2](#)

Filed Under: [AngularJS](#), [ASP.NET](#), [ASP.NET Identity](#), [ASP.Net Web API](#), [RESTful API](#), [Single Page Applications](#), [Web API Tutorial](#)

Tagged With: [AngularJS](#), [ASP.NET](#), [SPA](#), [Token Authentication](#), [Tutorial](#), [Web API](#)

## Comments



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 to generate token,

When I try to request for token in Postman, I'm getting following jSon data, Please help me to resolve this.

```
{  
  "message": "No HTTP resource was found that matches the request URI 'http://localhost:53227/api/token'",  
}
```

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

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 initialization of OAuthAuthorizationServerOption
- 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.

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

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](http://localhost:*****/token) instead of [http://localhost:\\*\\*\\*\\*\\*/api/token](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,



## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

Alex says

December 6, 2017 at 2:06 am

Awesome tutorial!

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 !

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)**Tullio Tesorone** says

December 10, 2017 at 11:57 am

Hello Taiseer , first chapeau.

Then , in my sample project ( exact duplicate of this ) the register method fails because usermodel is null.

At controller time it is already null.

But i am sure that json body is correctly received on the server side.

Any idea?

Thanks in advance

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

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)[Reply](#)

Edison says

January 15, 2018 at 12:00 pm

Thank you very much Sir .....)

[Reply](#)

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](#)

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

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 and 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 hackers 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.

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

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 SimpleMembershipProvider 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](#)

Ronald Kincaid says

March 7, 2018 at 7:01 am

I have been fighting with this for some time and cannot get the app to run locally. I followed each of your steps 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

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

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](#)

Bp says

May 25, 2018 at 5:44 am

you could use POSTMAN client / fiddler or similar open source tools to test as shown in step 12 by Taiseer

[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 synchronously.....

[Reply](#)

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)[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?

[Reply](#)

Janos says

April 15, 2018 at 1:59 am

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)
        {

```

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

```
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](#)

Jake says

May 22, 2018 at 10:49 pm

Somebody 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



## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)[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](#)

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

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

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 credentials  
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();
```

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)[Reply](#)

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](#)

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

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

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

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

BD Ank says

October 31, 2018 at 2:42 am

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

<http://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

# BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

Authorization: Bearer {my token}

it gives the output:

```
{  
  "message": "Authorization has been denied for this request."  
}
```

Anyone facing similar issue or any solution?

[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,

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)[Reply](#)

Jaime says

December 1, 2018 at 2:57 am

Why is it safe to remove Global.asax?

my global.asax.cs has this code:

```
protected void Application_Start()
{
    AreaRegistration.RegisterAllAreas();
    GlobalConfiguration.Configure(WebApiConfig.Register);
    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 ?

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

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 recognized.

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?

Thanks

Jaime

[Reply](#)

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](#)[« Older Comments](#)

## Leave a Reply

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

Comment



# BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

Name \*

Email \*

Website

POST COMMENT

## ABOUT TAISEER



Husband, Father,  
Consultant @ MSFT,  
Life Time Learner...  
[Read More...](#)

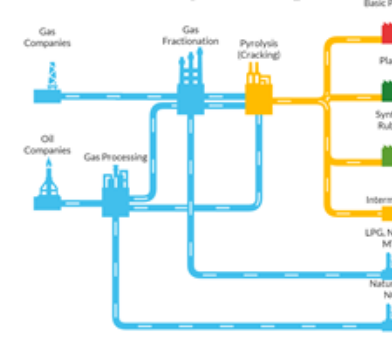


*Buy me a coffee*

# BIT OF TECHNOLOGY

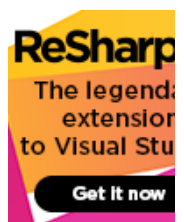
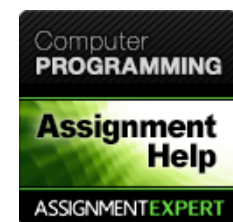
[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

**JavaScript Diagrams**



Click Here for GoJS

Advertise Here



Advertise Here

## BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

عايز تعرف  
نتيجة الماتش؟

اسأل Google

## 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

TAGS

# BIT OF TECHNOLOGY

[ARCHIVE](#)[ABOUT ME](#)[SPEAKING](#)[CONTACT](#)

[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](#)

[Basic authentication](#) [OWIN](#) [Cache](#) [COW](#) [Client Side Templating](#) [CORS](#)

[First Dependency Injection](#) [Entity](#)

[Framework](#) [ETag](#) [Foursquare API](#) [HTTP Caching](#)

[HTTP Verbs](#) [IMDB API](#) [IoC](#) [Javascript](#) [jQuery](#) [JSON](#) [JSON Web](#)

[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

Copyright © 2019 · eleven40 Pro Theme · Genesis Framework by StudioPress · WordPress · [Log in](#)