C# Web Api Bearer Token Tabanlı Authentication İşlemi

Öncelikle Nuget kütüphanelerinin kurulması gerekir:

```
Install-Package Microsoft.AspNet.WebApi.Owin -Version 5.2.2
Install-Package Microsoft.Owin.Host.SystemWeb -Version 2.1.0
Install-Package Microsoft.AspNet.Identity.Owin -Version 2.0.1
Install-Package Microsoft.AspNet.Identity.EntityFramework -Version 2.0.1
Install-Package Microsoft.Owin.Security.OAuth -Version 2.1.0
Install-Package Microsoft.Owin.Cors -Version 2.1.0
```

Daha sonra **Startup.cs** sınıfı eklenir:

```
using System;
using System.Web.Http;
using MapToner.Helpers;
using Microsoft.AspNet.Identity;
using Microsoft.Owin;
using Microsoft.Owin.Cors;
using Microsoft.Owin.Security.Cookies;
using Microsoft.Owin.Security.OAuth;
using Owin;

[assembly: OwinStartup(typeof(MapToner.Startup))]

namespace MapToner
{

     public class Startup
     {
          public void Configuration(IAppBuilder app)
          {
               var config = new HttpConfiguration();

               WebApiConfig.Register(config);
               ConfigureAuth(app);

               app.UseCors(CorsOptions.AllowAll);
               app.UseWebApi(config);
          }
          public void ConfigureAuth(IAppBuilder app)
          {

               var oAuthOptions = new OAuthAuthorizationServerOptions
               {
                    TokenEndpointPath = new PathString("/token"),
                    Provider = new AppOAuthProvider(),
                    AccessTokenExpireTimeSpan = TimeSpan.FromMinutes(30),
                    AllowInsecureHttp = true //todo Don't do this in production ONLY FOR
DEVELOPING: ALLOW INSECURE HTTP!
               };

               app.UseOAuthAuthorizationServer(oAuthOptions);//Oauth2 ayarlarinin yapilmasini
saglar
               //Http request header'daki bearer token olup olmadigini anlamak icin kullanilir.
               app.UseOAuthBearerAuthentication(new OAuthBearerAuthenticationOptions());
          }
     }
}
```

WebApiConfig.cs sınıfı aşağıdaki gibi olursa çok iyi olur:

```

public static class WebApiConfig
{
```

```
        public static void Register(HttpConfiguration config)
        {
            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(
            name: "DefaultApi",
            routeTemplate: "api/v1/{controller}/{id}",
            defaults: new {id = RouteParameter.Optional}
            );

            ConfigureJsonResponse(config);
        }

        private static void ConfigureJsonResponse(HttpConfiguration config)
        {
            //Removes XML response, return JSON response
            var contractResolver = new DefaultContractResolver
            {
                NamingStrategy = new SnakeCaseNamingStrategy()
            };
            var jsonFormatter = config.Formatters.OfType<JsonMediaTypeFormatter>().First();
            jsonFormatter.SerializerSettings.ContractResolver= contractResolver;
            // Adding JSON type web api formatting.
            config.Formatters.Clear();
            config.Formatters.Add(jsonFormatter);
        }
    }
```

Son olarak **AppOAuthProvider.cs** sınıfı tanımlanır:

```
using System.Collections.Generic;
using System.Security.Claims;
using System.Threading.Tasks;
using MapToner.Models;
using Microsoft.Owin.Security;
using Microsoft.Owin.Security.OAuth;

namespace MapToner.Helpers
{
     /// <summary>
     /// Application OAUTH Provider class.
     /// </summary>
     public class AppOAuthProvider : OAuthAuthorizationServerProvider
     {

         public override async Task
GrantResourceOwnerCredentials(OAuthGrantResourceOwnerCredentialsContext context)
         {
             // Initialization.
             var username = context.UserName;
             var password = context.Password;
             var user = new User(); //todo read from db
             user.Username= username;
             user.Password= password;
             // Verification.
             if (user == null)
             {
                 // Settings.
                 context.SetError("invalid_grant", "The user name or password is incorrect.");

                 // Retuen info.
                 return;
             }

             // Initialization.
             var claims = new List<Claim>();
```

```
            // Setting
            var claims = new List<Claim>
            {
                new Claim(ClaimTypes.Name, context.UserName),
                new Claim(ClaimTypes.NameIdentifier, user.id.ToString())
            };

            // Setting Claim Identities for OAUTH 2 protocol.
            var oAuthClaimIdentity = new ClaimsIdentity(claims, OAuthDefaults.AuthenticationType);
            // Setting user authentication.
            var ticket = new AuthenticationTicket(oAuthClaimIdentity,
CreateProperties(user.Username));
            // Grant access to authorize user.
            context.Validated(ticket);

        }
        /// <summary>
        /// Accesss token generate olurken yani /token linkine post edildiginde CreateProperties
metodunun donderdigi Map degeri json response da ekstra
        /// parametre olarak gonderilir.
        /// </summary>
        /// <param name="userName"></param>
        /// <returns></returns>
        public static AuthenticationProperties CreateProperties(string userName)
        {
            IDictionary<string, string> data = new Dictionary<string, string>
            {
                {"userName", userName}
            };
            return new AuthenticationProperties(data);
        }

        /// <summary>
        /// CreateProperties metodundaki parametrelere ek olarak baska parametreler eklemek icin
kullanilabilir. Yine ayni sekilde /token linkine post edilince
        /// post sonucunda donen json da bu parametreler yer alacaktir.
        /// </summary>
        /// <param name="context"></param>
        /// <returns></returns>
        public override Task TokenEndpoint(OAuthTokenEndpointContext context)
        {
            foreach (var property in context.Properties.Dictionary)
                context.AdditionalResponseParameters.Add(property.Key, property.Value);

            return Task.FromResult<object>(null);
        }
        /// <summary>
        /// Client Id leri validate etmek icin kullanilir. Bizim uygulamamızda 1 tane olduğu için
context.Validated() demek yeterli olacaktır.
        /// </summary>
        /// <param name="context"></param>
        /// <returns></returns>
        public override async Task
ValidateClientAuthentication(OAuthValidateClientAuthenticationContext context)
        {
            context.Validated();
        }
    }
}
```

Daha fazla bilgi için:

https://blogs.perficient.com/2017/06/11/token-based-authentication-in-web-api-2-via-owin/

https://www.c-sharpcorner.com/article/asp-net-mvc-oauth-2-0-rest-web-api-authorization-using-database-first-approach/