Token base Authenection

Server (WebApi/Empty+Api):

1. הוספת חבילות nuget:

Microsoft.Owin

Microsoft.Owin.Host.SystemWeb

Microsoft.Owin.Security.OAuth

Microsoft.Owin.Security

Microsoft.AspNet.Identity.Owin

Microsoft.Owin.Cors

Microsoft.AspNet.WebApi.Cors

1. יצירת class בשם startup , הוא יראה כך:

using Microsoft.Owin;

using Microsoft.Owin.Security;

using Microsoft.Owin.Security.OAuth;

using Newtonsoft.Json;

using Owin;

using Server.Models;

using System;

using System.Collections.Generic;

using System.Security.Claims;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Cors;

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

namespace Server

{

public class Startup

{

[EnableCors(origins: "\*", headers: "\*", methods: "\*")]

public void Configuration(IAppBuilder app)

{

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

var myProvider = new MyAuthProvider();

OAuthAuthorizationServerOptions options = new OAuthAuthorizationServerOptions

{

AllowInsecureHttp = true,

TokenEndpointPath = new PathString("/token"),

AccessTokenExpireTimeSpan = TimeSpan.FromMinutes(180),

Provider = myProvider

};

app.UseOAuthAuthorizationServer(options);

app.UseOAuthBearerAuthentication(new OAuthBearerAuthenticationOptions());

HttpConfiguration config = new HttpConfiguration();

WebApiConfig.Register(config);

}

}

public class MyAuthProvider : OAuthAuthorizationServerProvider

{

public override Task ValidateClientAuthentication(OAuthValidateClientAuthenticationContext context)

{

context.Validated();

return Task.FromResult<object>(null);

}

public override Task GrantResourceOwnerCredentials(OAuthGrantResourceOwnerCredentialsContext context)

{

var dict = new Dictionary<string, string>();

var identity = new ClaimsIdentity(context.Options.AuthenticationType);

identity.AddClaim(new Claim(ClaimTypes.Name, context.UserName));

context.Validated(identity);

return Task.FromResult<object>(null);

}

}

}

1. הוספת שני קבצי classים בתיקיית המודלים:

LoginData.cs:

namespace Server.Models

{

public class LoginData

{

public string TokenJson { get; set; }

public User User { get; set; }

}

}

==========================================================

UserLoggedAttribute.cs:

using System.Linq;

using System.Security.Claims;

using System.Threading;

using System.Threading.Tasks;

using System.Web;

using System.Web.Http;

using System.Web.Http.Controllers;

using System.Web.Http.Metadata;

namespace Server.Models

{

public class UserLoggedAttribute : ParameterBindingAttribute

{

WolfProjectsEntities db = new WolfProjectsEntities();

public override HttpParameterBinding GetBinding(HttpParameterDescriptor parameter)

{

return new UserLoggedParameterBinding(parameter);

}

}

public class UserLoggedParameterBinding : HttpParameterBinding

{

WolfProjectsEntities db = new WolfProjectsEntities();

public UserLoggedParameterBinding(HttpParameterDescriptor parameter)

: base(parameter)

{

}

public override Task ExecuteBindingAsync(ModelMetadataProvider metadataProvider,

HttpActionContext actionContext, CancellationToken cancellationToken)

{

var user = HttpContext.Current.User as ClaimsPrincipal;

var identity = user.Identity as ClaimsIdentity;

var claim = identity.Claims.Where(c => c.Type == ClaimTypes.Name).Select(s=>s.Value).SingleOrDefault();

actionContext.ActionArguments[Descriptor.ParameterName] = db.Users.Where(w => w.name == claim).FirstOrDefault();

return Task.FromResult<object>(null);

}

}

}

1. הוספת קונטרולר בשם Account :

using Server.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Http;

using System.Security.Claims;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Cors;

namespace Server.Controllers

{

[EnableCors(origins: "\*", headers: "\*", methods: "\*")]

public class AccountController : ApiController

{

WolfProjectsEntities db = new WolfProjectsEntities();

[HttpGet]

[Route("logout")]

public WebResult<string> logout([UserLogged] User user)

{

//get user data

var identity = User.Identity as ClaimsIdentity;

var claim = identity.Claims.Where(c => c.Type == ClaimTypes.Name).SingleOrDefault();

//remove user data

identity.RemoveClaim(claim);

return new WebResult<string>

{

Success = true,

Message = $"{user.name} התנתק בהצלחה",

Value = null

};

}

[HttpGet]

[Route("login")]

public async Task<WebResult<LoginData>> Login(string username, string password)

{

var user = db.Users.Where(w => w.name == username && w.password == password).FirstOrDefault();

if (user != null)//אם המשתמש קיים במאגר המשך לקבלת טוקן, אחרת החזר שגיאה שהמתשמש לא קיים

{

var accessToken = await GetTokenDataAsync(username, password);

if (!string.IsNullOrEmpty(accessToken))

{

return new WebResult<LoginData>

{

Success = true,

Message = "התחברת בהצלחה",

Value = new LoginData

{

TokenJson = accessToken,

User = user

}

};

}

}

return new WebResult<LoginData>

{

Success = false,

Message = "אין משתמש רשום בשם וסיסמא זו....",

Value = null

};

}

[HttpPost]

[Route("register")]

public async Task<WebResult<LoginData>> Register(User user)

{

db.Users.Add(user);

if (db.SaveChanges() > 0)//בדיקה שהמידע נשמר

{

var accessToken = await GetTokenDataAsync(user.name, user.password);

if (!string.IsNullOrEmpty(accessToken))

{

return new WebResult<LoginData>

{

Success = true,

Message = "התחברת בהצלחה",

Value = new LoginData

{

TokenJson = accessToken,

User = user

}

};

}

}

return new WebResult<LoginData>

{

Success = false,

Message = "אין משתמש רשום בשם וסיסמא זו....",

Value = null

};

}

private async Task<string> GetTokenDataAsync(string username, string password)

{

HttpClient httpClient = new HttpClient();

httpClient.BaseAddress = new Uri(Request.RequestUri.Scheme + "://" + Request.RequestUri.Authority + "/token");

var postData = new List<KeyValuePair<string, string>>();

postData.Add(new KeyValuePair<string, string>("UserName", username));

postData.Add(new KeyValuePair<string, string>("Password", password));

postData.Add(new KeyValuePair<string, string>("grant\_type", "password"));//don't dare to change that!!!

HttpContent content = new FormUrlEncodedContent(postData);

HttpResponseMessage response = await httpClient.PostAsync("token", content);

response.EnsureSuccessStatusCode();

return await response.Content.ReadAsStringAsync();

}

}

}

בקליינט (אנגולר) הוספת service:

import { Injectable } from '@angular/core';

import { User } from '../models';

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Router } from '@angular/router';

@Injectable()

export class UserService {

public errors: any = {};

public errorMsg: string = "אירעה שגיאה, נסה שוב מאוחר יותר";

public user: User;

httpOptions;

url = "http://localhost:51396/";

constructor(private http: HttpClient, private router: Router) { this.resetToken(); }

changeToken(token: string) {

this.httpOptions.headers =

new HttpHeaders({ 'Content-Type': 'application/json', 'Authorization': "Bearer " + token });

}

resetToken() {

this.httpOptions = { headers: new HttpHeaders({ 'Content-Type': 'application/json', 'Authorization': '' }) };

}

login(username, password) {

return this.http.get(this.url + `login?username=${username}&password=${password}`)

.subscribe((res: any) => { this.afterLogin(res); });

}

register(username, password) {

this.http.post(`${this.url}/register`, { name: username, password: password })

.subscribe((res: User) => { this.afterLogin(res); });

}

logout() {

this.http.get(`${this.url}/logout`, this.httpOptions).subscribe((res: any) => {

if (res.Success) {

this.user = null;

this.resetToken();

this.router.navigate(["/account"]);

alert(res.Message);

}

});

}

afterLogin(res) {

if (res.Success) {

var token = JSON.parse(res.Value.TokenJson)

this.changeToken(token.access\_token);

this.user = res.Value.User;

this.router.navigate([""]);

}

else alert(res.Message);

}

}