**Section08 OAuth Client**

**Notes:-**

**1-we learn about the JWT barer middleware that used to generate token and to authorize token if it’s valid or not and pass the JWT token through header attribute or query string**

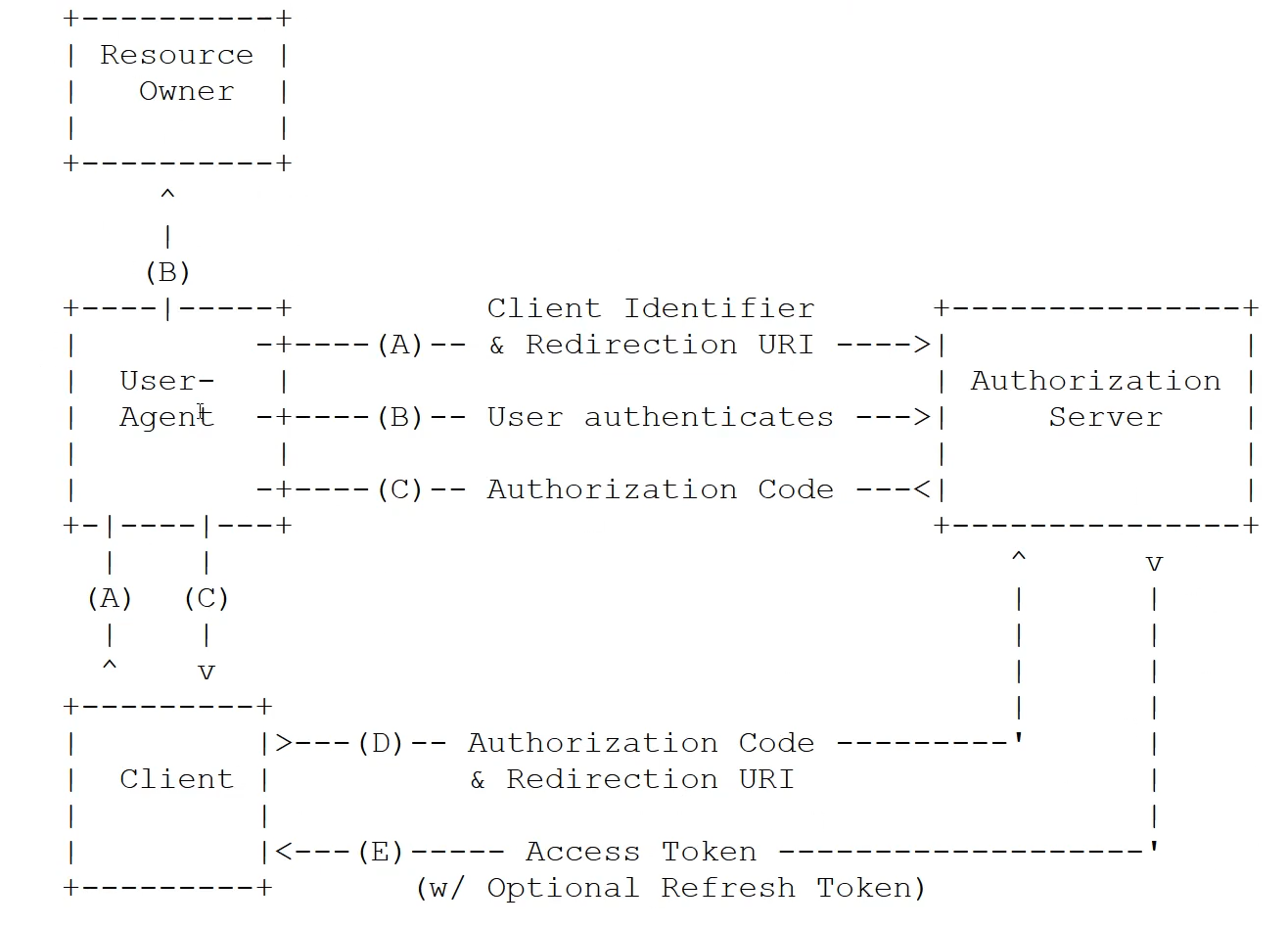
**(But we don’t store the JWT token as cookie in the browser, so we have to apply OAuth Client)**

**2-we will use OAuth client to store the JWT token on the browser and used to validate authorization**

**Note:-**

**That on the Microsoft.AspNetCore.Authentication the generate and register and validate and store the token all things are occurred on the same project like web app**

**While on the JWT we have to make two project client and server where the server is used to generate and validate only while on the client used to store the token**



**1-We have two project Client and Server which the Client is configured when the cookie is invalid or not exist or expired redirect to the view defined on the server project**

**2-on the server project when submit the information it will redirect again through sandbox to client and return again to the server to check if the client id is valid and then call Token action which generate Token on the browser.**

**Steps:-**

**1-on the Client we install the following nugget packages**

**Microsoft.AspNetCore.Authentication.JwtBearer**

**Newtonsoft**

**2-on the Server we install the following nugget packages**

**Microsoft.AspNetCore.Authentication.JwtBearer**

**Microsoft.AspNetCore.Authentication.JwtBearer**

**Newtonsoft**

**3-on the client we set the following code on the HomeController.cs as below**

**using Microsoft.AspNetCore.Authentication;**

**using Microsoft.AspNetCore.Authorization;**

**using Microsoft.AspNetCore.Mvc;**

**using Microsoft.Extensions.Logging;**

**namespace Client.Controllers{**

**public class HomeController : Controller{**

**private readonly ILogger<HomeController> \_logger;**

**public HomeController(ILogger<HomeController> logger){\_logger = logger;}**

**public IActionResult Index(){return View();}**

**[Authorize]**

**public IActionResult Secret(){**

**var token = HttpContext.GetTokenAsync("access\_token");**

**return View();}}}**

**4-on the client > startup.cs we set the following code as below**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Security.Claims;**

**using System.Text;**

**using System.Text.Json.Serialization;**

**using System.Threading.Tasks;**

**using Microsoft.AspNetCore.Authentication.OAuth;**

**using Microsoft.AspNetCore.Builder;**

**using Microsoft.AspNetCore.Hosting;**

**using Microsoft.AspNetCore.Http;**

**using Microsoft.Extensions.DependencyInjection;**

**using Microsoft.Extensions.Hosting;**

**using Newtonsoft.Json;**

**namespace Client{**

**public class Startup{**

**public void ConfigureServices(IServiceCollection services){**

**//we add authentication middleware with set the cookie and shcema name and**

**services.AddAuthentication(config => {**

**// We check the cookie to confirm that we are authenticated**

**config.DefaultAuthenticateScheme = "EnbehCookie";**

**// When we sign in we will deal out a cookie**

**config.DefaultSignInScheme = "EnbehCookie";**

**// use this to check if we are allowed to do something.**

**config.DefaultChallengeScheme = "OurServer";})**

**.AddCookie("EnbehCookie")**

**.AddOAuth("OurServer", config => {**

**config.ClientId = "client\_id";**

**config.ClientSecret = "client\_secret";**

**//the request path within the application path which pass through it**

**config.CallbackPath = "/oauth/callback";**

**//to map the redirect link from oAuth client to the Auth server**

**config.AuthorizationEndpoint = "https://localhost:44382/oauth/authorize";**

**//to map the redirect link from Auth server to oAuth Client (4)**

**config.TokenEndpoint = "https://localhost:44382/oauth/token";**

**//to allow save token on the browser**

**config.SaveTokens = true;**

**//when the token generate we can caputre it and make operation**

**config.Events = new OAuthEvents(){**

**OnCreatingTicket = context =>{**

**var accessToken = context.AccessToken;**

**var base64payload = accessToken.Split('.')[1];**

**var bytes = Convert.FromBase64String(base64payload);**

**var jsonPayload = Encoding.UTF8.GetString(bytes);**

**var claims = JsonConvert.DeserializeObject<Dictionary<string, string>>(jsonPayload);**

**foreach (var claim in claims){context.Identity.AddClaim(new Claim(claim.Key, claim.Value));}**

**return Task.CompletedTask;}};});**

**services.AddHttpClient();**

**services.AddControllersWithViews()**

**.AddRazorRuntimeCompilation();}**

**public void Configure(IApplicationBuilder app, IWebHostEnvironment env){**

**if (env.IsDevelopment()){app.UseDeveloperExceptionPage();}**

**app.UseHttpsRedirection();**

**app.UseStaticFiles();**

**app.UseRouting();**

**app.UseAuthentication();**

**app.UseAuthorization();**

**app.UseEndpoints(endpoints =>{endpoints.MapDefaultControllerRoute();});}}}**

**5-on the Server > OAuthController.cs we set the following code**

**using Microsoft.AspNetCore.Authorization;**

**using Microsoft.AspNetCore.Http.Extensions;**

**using Microsoft.AspNetCore.Mvc;**

**using Microsoft.IdentityModel.Tokens;**

**using Newtonsoft.Json;**

**using Server.Models;**

**using System;**

**using System.IdentityModel.Tokens.Jwt;**

**using System.Security.Claims;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace Server.Controllers{**

**public class OAuthController : Controller{**

**[HttpGet]**

**public IActionResult Authorize(**

**string response\_type, // authorization flow type**

**string client\_id, // client id**

**string redirect\_uri,**

**string scope, // what info I want = email,grandma,tel**

**string state) // random string generated to confirm that we are going to back to the same client{**

**var query = new QueryBuilder();**

**query.Add("redirectUri", redirect\_uri);**

**query.Add("state", state);**

**return View(model: query.ToString());}**

**[HttpPost]**

**public IActionResult Authorize(**

**string username,**

**string redirectUri,**

**string state){**

**const string code = "BABAABABABA";**

**var query = new QueryBuilder();**

**query.Add("code", code);**

**query.Add("state", state);**

**return Redirect($"{redirectUri}{query.ToString()}");}**

**//here is the redirect in sandbox from OAuth client to the Authorization server**

**public async Task<IActionResult> Token(**

**string grant\_type, // flow of access\_token request**

**string code, // confirmation of the authentication process**

**string redirect\_uri,**

**string client\_id,**

**string refresh\_token){**

**var claims = new[]{**

**new Claim(JwtRegisteredClaimNames.Sub, "some\_id"),**

**new Claim("granny", "cookie")};**

**var secretBytes = Encoding.UTF8.GetBytes(JWTSettings.Secret);**

**var key = new SymmetricSecurityKey(secretBytes);**

**var algorithm = SecurityAlgorithms.HmacSha256;**

**var signingCredentials = new SigningCredentials(key, algorithm);**

**var token = new JwtSecurityToken(**

**JWTSettings.Issuer,**

**JWTSettings.Audiance,**

**claims,**

**notBefore: DateTime.Now,**

**expires: grant\_type == "refresh\_token"**

**? DateTime.Now.AddMinutes(5)**

**: DateTime.Now.AddMilliseconds(1),**

**signingCredentials);**

**var access\_token = new JwtSecurityTokenHandler().WriteToken(token);**

**var responseObject = new{**

**access\_token,**

**token\_type = "Bearer",**

**raw\_claim = "oauthTutorial",**

**refresh\_token = "RefreshTokenSampleValueSomething77"};**

**var responseJson = JsonConvert.SerializeObject(responseObject);**

**var responseBytes = Encoding.UTF8.GetBytes(responseJson);**

**await Response.Body.WriteAsync(responseBytes, 0, responseBytes.Length);**

**return Redirect(redirect\_uri);}**

**[Authorize]**

**public IActionResult Validate(){**

**if (HttpContext.Request.Query.TryGetValue("access\_token", out var accessToken)){**

**return Ok();}**

**return BadRequest();}}}**

**6-on the Views > OAuth > Authorize.cshtml we set the following code**

**@model string**

**@{var url = $"/OAuth/Authorize{Model}";}**

**@Model**

**<form action="@url" method="post">**

**<input type="text" name="username" value=""/>**

**<input type="submit" value="Submit"/>**

**</form>**

**7-on the server > startup.cs > the code as same in the previous section**

**using Microsoft.AspNetCore.Authentication.JwtBearer;**

**using Microsoft.AspNetCore.Builder;**

**using Microsoft.AspNetCore.Hosting;**

**using Microsoft.Extensions.DependencyInjection;**

**using Microsoft.Extensions.Hosting;**

**using Microsoft.IdentityModel.Tokens;**

**using Server.Models;**

**using System;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace Server{**

**public class Startup{**

**public void ConfigureServices(IServiceCollection services){**

**services.AddAuthentication("OAuth")**

**.AddJwtBearer("OAuth", config =>{**

**var secretBytes = Encoding.UTF8.GetBytes(JWTSettings.Secret);**

**var key = new SymmetricSecurityKey(secretBytes);**

**config.Events = new JwtBearerEvents(){**

**OnMessageReceived = context =>{**

**if (context.Request.Query.ContainsKey("access\_token")){**

**context.Token = context.Request.Query["access\_token"];}**

**return Task.CompletedTask;}};**

**config.TokenValidationParameters = new TokenValidationParameters(){**

**ClockSkew = TimeSpan.Zero,**

**ValidIssuer = JWTSettings.Issuer,**

**ValidAudience = JWTSettings.Audiance,**

**IssuerSigningKey = key,};});**

**services.AddControllersWithViews()**

**.AddRazorRuntimeCompilation();}**

**public void Configure(IApplicationBuilder app, IWebHostEnvironment env){**

**if (env.IsDevelopment()){app.UseDeveloperExceptionPage();}**

**app.UseHttpsRedirection();**

**app.UseStaticFiles();**

**app.UseRouting();**

**app.UseAuthentication();**

**app.UseAuthorization();**

**app.UseEndpoints(endpoints =>{endpoints.MapDefaultControllerRoute();});}}}**

**Steps:-**

**1-on the Client (SPA) when click on any authorized action and not have cookie it will redirect into the OAuth/Authorize that exist on the Server project (Authorization server)**

**2-on the server when click fill the information and submit it will redirect again into the client (SPA) to make sure that the client id is valid and then redirect to the OAuth/Token through internal sandbox and then generate Token and saving it through set SaveToken option on the client (SPA) Authentication layer**

**(So the JWT barer is installed only on the server and the authentication configuration redirection all operation on the client)**