Click sig up

Select account type : Personal or Company

CallBack Url: is where Auth0 sends the users back to the moment they are logged in

Same thing with the log out url

Upon scaffolfing of your project run it so as to be sure that everything is working as planned

<https://localhost:44360/>



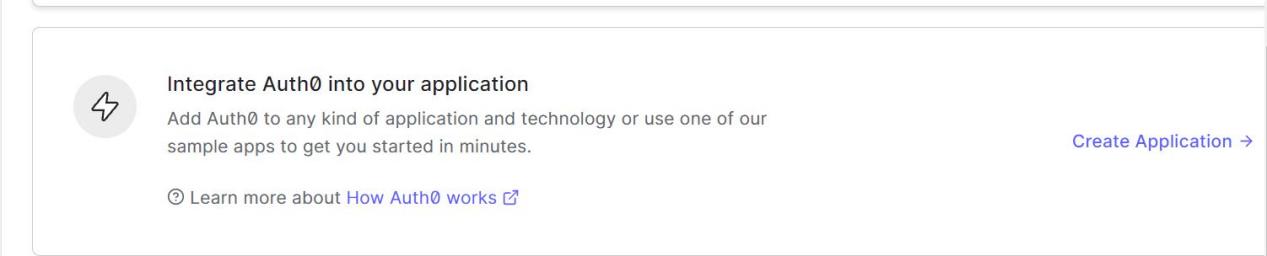
Copy the entire text and numbers on the url (local host)

Create and account or signup on Auth0 (https://auth0.com/signup?place=header&type=button&text=sign%20up)

Upon successful sign up, a tenant domain name is generated for you by Auth0

Choose a Region where you want the app to be hosted then click the Create account button

Then select Create Application



Go to ypur dash board upon signing up

Next you have to make Auth0 to be aware of your Asp.net app, by supplying the following parameters

Name: This is asking you to fill in the name of your Dotnet application

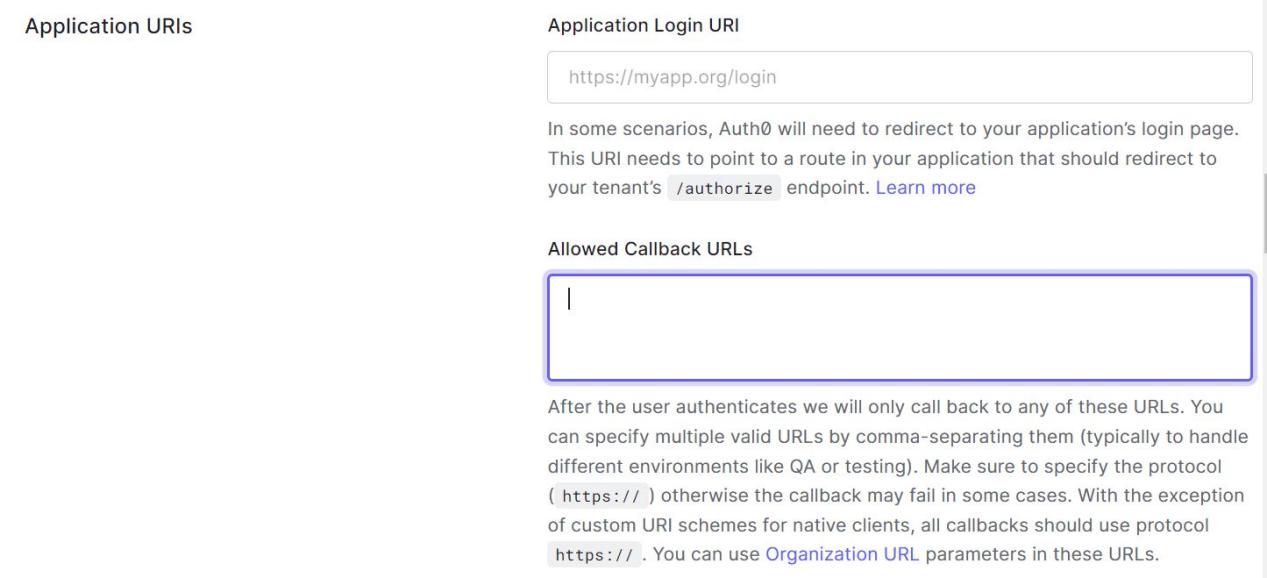
Application Type: Select the type of application you want to build

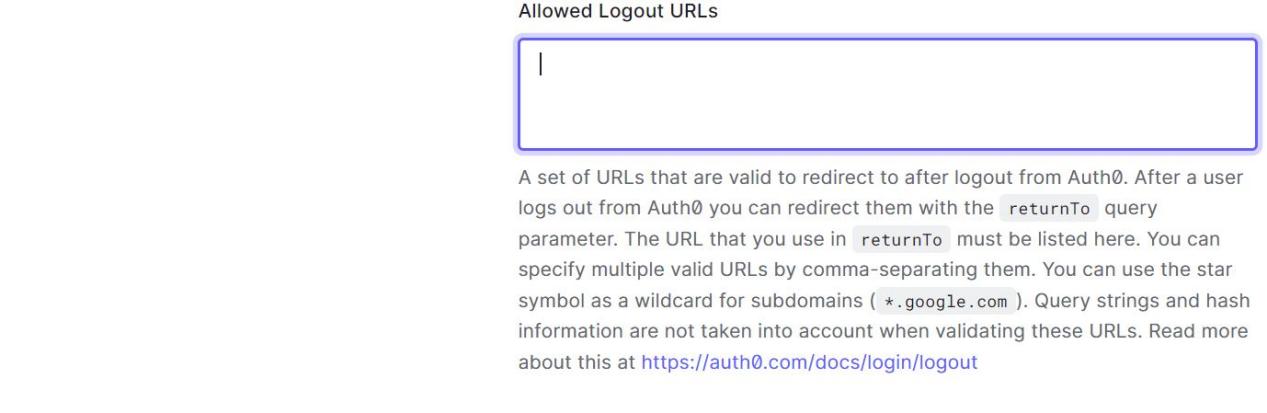
Hit the Create Button

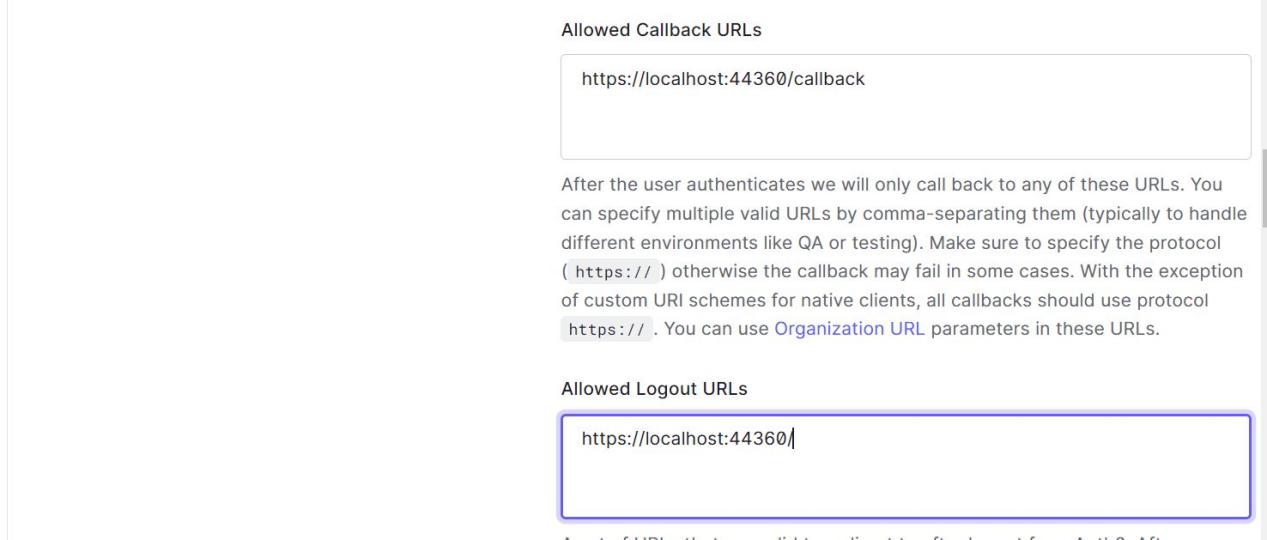
What Technology are using to build your project :Select Asp.Net Core v3

A client Id number will be automatically generated for you copy this and keep it in a safe place

Next go to the Application Url section of the settings since we are just setting up our applcation, we will use the local host address of our dotnet core applicxation that we copied awhile back as the “Allowed Callback url” then we will just add a callback







Next click on save changes and then we are done for now setting up Auth0 on the Auth0 website.

Next move to the appsettings section of your dotnet core application.

We store the Auth0 details we copied awhile back in the appsettings.json file so that we can assess it later in the configuration.

"Auth0": {

"Domain": "",

"ClientId": "",

"ClientSecret": ""

},

"AllowedHosts": "\*"

Since we are building a traditional web app we are going to be using Cookies as part of the authentication

Next fire up the nugget package manager of the Web App project (ImageGallery) and install the latest version of the following

Microsoft.Asp.NetCore.Authentication.Cookies 2.2.0

Microsoft.Asp.NetCore.Authentication.OpenIdConnect 5.0.10

Basically, the nugget packages above is what will inform our DotNet Core project that we want to use Auth0

Next register the Auth0 in the configure method of the StartUp class by calling the services.AddAuthentication() method

You have to tell the Auth0 how you want it to work or how you want it to be set up by configuring various options on it. So next we have to configure options on the Authentication method by adding the codes below to the StarUp class.

services.AddAuthentication( options =>

{

options.DefaultAuthenticateScheme = CookieAuthenticationDefaults.AuthenticationScheme;

options.DefaultSignInScheme = CookieAuthenticationDefaults.AuthenticationScheme;

options.DefaultChallengeScheme = CookieAuthenticationDefaults.AuthenticationScheme;

})

Next chain the above with the AddCookies and the AddOpenIdConnect method respectively.

Now, we are set to configure options on the AddOpenIdConnect method.

Doc for the configuration (https://auth0.com/docs/quickstart/webapp/aspnet-core/01-login#configure-your-application-to-use-auth0) link back to this

Next go to the controllers folder and add an empty controller class which will be responsible for handling all login, log out requests. You can give this controller class any name you want but by convention it is usually called AccountController

Lets just head to the documentation and copy the codes for setting up the Login and Logout methods in the AccountController:

public class AccountController : Controller

{

public async Task Login(string returnUrl = "/")

{

await HttpContext.ChallengeAsync("Auth0", new AuthenticationProperties() { RedirectUri = returnUrl });

}

public async Task LogOut(string returnUrl = "/")

{

await HttpContext.SignOutAsync("Auth0", new AuthenticationProperties()

{

RedirectUri = Url.Action("Index", "Gallery")

});

await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);

}

}

What is simply happening here is that we have two methods Login and Logout

For the Login method:

We it takes a parameter string and then we call the ChallengeAsync function and then telling it to use the Auth0 scheme.

For the Logout method:

This is also similar to what we have in the login function except that we invoke the SignoutAsync method and then tell it to use the Auth0 scheme.

Next we now configure an event options in the configure method of the Startup class just after the options.CliamsIssuer method which is what responsible for call the Access Token Api calls.

Next we need to add the button for login and logout to our application Navbar to do this, go to the following folder in your web app /Views/Shared/\_Layout.cshtml . Basically what we want to do is just to add a Login/Logout button link to the Navbar. So the user can click login if not authenticated and then logout if authenticated.