**User Login with Azure AD in Blazor Assembly Project**

**Introduction**

This guide outlines the steps to enable user authentication using Azure Active Directory (Azure AD) in a Blazor assembly project. Azure AD authentication allows you to secure your Blazor application by leveraging the authentication capabilities provided by Azure AD.

**Prerequisites**

Before you begin, ensure you have the following prerequisites:

* An Azure subscription
* A Blazor assembly project set up
* Access to Azure Active Directory

**Steps**

**Step 1: Register your application in Azure AD**

1. Log in to the Azure portal.
2. Navigate to Azure Active Directory.
3. Select "App registrations" and click on "New registration."
4. Enter a name for your application, choose the supported account types, and specify the redirect URI.
5. After registration, note down the Application (client) ID and tenant ID.

**Step 2: Configure authentication in your Blazor application**

Install the necessary NuGet packages:

dotnet add package Microsoft.Authentication.WebAssembly.Msal

dotnet add package Microsoft.AspNetCore.Components.Authorization

dotnet add package Microsoft.AspNetCore.Components.WebAssembly.Authentication

Configure authentication in the **Program.cs** file:

builder.Services.AddMsalAuthentication(options =>

{

builder.Configuration.Bind("AzureAd", options.ProviderOptions.Authentication);

});

Add Azure AD settings to your **appsettings.json** file:

"AzureAd": {

"ClientId": "<your\_client\_id",

"Authority": "https://login.microsoftonline.com/<tenant\_id> ",

"ValidateAuthority": true

}

### Step 3: Secure your application routes

Add the **[Authorize]** attribute to the routes you want to protect in your Blazor components.

@attribute [Authorize]

Alternatively, you can apply authorization globally in **App.razor**:

<**CascadingAuthenticationState**>

<**Router** **AppAssembly**="@typeof(App).Assembly">

<**Found** **Context**="routeData">

<**AuthorizeRouteView** **RouteData**="@routeData" **DefaultLayout**="@typeof(MainLayout)">

<**NotAuthorized**>

@if (context.User.Identity?.IsAuthenticated != true)

{

<**RedirectToLogin** />

}

else

{

<p role="alert">You are not authorized to access this resource.</p>

}

</**NotAuthorized**>

</**AuthorizeRouteView**>

<**FocusOnNavigate** **RouteData**="@routeData" **Selector**="h1" />

</**Found**>

<**NotFound**>

<**PageTitle**>Not found</**PageTitle**>

<**LayoutView** **Layout**="@typeof(MainLayout)">

<p role="alert">Sorry, there's nothing at this address.</p>

</**LayoutView**>

</**NotFound**>

</**Router**>

</**CascadingAuthenticationState**>

### Step 4: Implement Login and Logout functionality

1. Create login and logout components **LoginDispaly**.
2. Use the **NavigateToLogout** methods from the **NavigationManager** service to handle user log out.

@using Microsoft.AspNetCore.Components.WebAssembly.Authentication

@inject NavigationManager Navigation

<AuthorizeView>

<Authorized>

Hello, @context.User.Identity?.Name!

<button class="nav-link btn btn-link" @onclick="BeginLogOut">Log out</button>

</Authorized>

<NotAuthorized>

<a href="authentication/login">Log in</a>

</NotAuthorized>

</AuthorizeView>

@code{

public void BeginLogOut()

{

Navigation.NavigateToLogout("authentication/logout");

}

}

If user is not login create a Authentication component and use **RemoteAuthenticatorView** to show Microsoft login view.

@page "/authentication/{action}"

@using Microsoft.AspNetCore.Components.WebAssembly.Authentication

<**RemoteAuthenticatorView** **Action**="@Action" />

@code{

[Parameter] public string? Action { get; set; }

}

## Conclusion

Congratulations! You have successfully configured user authentication with Azure AD in your Blazor assembly project. Users will now be able to authenticate using their Azure AD credentials.