*Asp.Net Core Identity API’s & Secure JWT \ Refresh HttpOnly Cookies within a Clean Architecture Solution*

|  |  |
| --- | --- |
| Bert O’Neill | |
| Department: | Code Project - IT\Software Support |
| Document Title: | Asp.Net Core Identity API & Secure JWT \ Refresh HttpOnly Cookies Within a Clean Architecture solution |
| Author: | Bert O’Neill |
| Date: | 03-Sept-2024 |
| Version | 1.0 |
| Email | BertONeill@Yahoo.com |
| Document Goals | Provide a complete reference for all the Asp.Net Core Identity API’s within .Net Core environment. But also securing them with JWT & Refresh Bearer Tokens using HttpOnly cookies, within a Clean Architecture solution. |

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 03-Sept-2024 | 1.0 | Initial draft | Bert O’Neill |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[Revision History 1](#_Toc176134947)

[Introduction 3](#_Toc176134948)

[Purpose 3](#_Toc176134949)

[Scope 3](#_Toc176134950)

[Prerequisites 3](#_Toc176134951)

[Miscellaneous 4](#_Toc176134952)

[Acronym 4](#_Toc176134953)

# Introduction

This document a detailed overview on how to securely implement the various Asp.Net Core Identity Minimal API’s, using HttpOnly Cookies, incorporating JWT and Refresh Bearer tokens within a Clean Architecture approach.

## Purpose

Online you will find numerous articles, tutorials and YouTube videos detailing the various Identity APIs, but they are either disjointed (only detailing several API’s) or wanting you to subscribe to their patron site to gain access to their code!!!

I wanted to provide a one-stop-shop tutorial where you can pick and choose the API you are interested in learning more about, but also how to securely (HTTS) implement them using Bearer (JWT\Refresh) Tokens – and convey this within a Clean Architecture approach.

But also, to incorporate as many standard project components as possible (like Logging Middleware, Mappers, Fluent Validation, API Caching, Error Handling Middleware and the various Identity configurations, that need to be implemented for Asp.Net Core Identity to function smoothly).

## Scope

The scope of this document is to quickly convey the implementation and configuration steps, needed to securely incorporate the Identity APIs into your application – to authentication and authorization to your project APIs, like production code by the major multinationals).

# Prerequisites

* An understanding of [Asp.Net Core Identity API’s](https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio)
* An understanding of [Clean Architecture](https://www.codeproject.com/Articles/5351235/Clean-Architecture-Incorporating-Repository-Patter)
* An understanding of [Minimal API’s](https://learn.microsoft.com/en-us/aspnet/core/tutorials/min-web-api?view=aspnetcore-8.0&tabs=visual-studio)
* An understanding of [Entity Framework Core](https://learn.microsoft.com/en-us/ef/) (EFC)
* An understanding of [EFC Data Migrations](https://learn.microsoft.com/en-us/ef/core/managing-schemas/migrations/managing?tabs=dotnet-core-cli)
* An understanding of [Microsoft’s SQL Server](https://learn.microsoft.com/en-us/sql/linux/new-to-sql-learning-resources?view=sql-server-ver16)
* An understanding of [Auto Mappers](https://automapper.org/)
* An understanding of [Fluent Validation](https://fluentvalidation.net/)
* An understanding of the [HttpOnly](https://owasp.org/www-community/HttpOnly) flag
* Install [Visual Studio 22 Community](https://visualstudio.microsoft.com/vs/community/)
* Install [SQL Server Developer](https://www.microsoft.com/en-us/sql-server/sql-server-downloads) and [SSMS](https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16)

# Project Structure

Below, you can see that I have incorporated a Clean Architecture approach to implementing ASP Net Core Identity security APIs. The API layer has been designed using .Net8 Minimal APIs, *I am testing the API’s at this stage with Swagger only* – but a Blazor WASM (client) project has been included in the solution for completeness – which references a Shared project of DTO’s to communicate with the API layer efficiently.

A screenshot of a computer

Description automatically generated

# Setup And Component Configurations

## Register With Mailosaur to Send (Confirmation) Emails

## Setup Identity Database Using Data Migration

## Why Stateless Identity Session - No Caching

## Why use JWT & Refresh Bearer Tokens

### View JWT & Refresh Tokens in Browser

## Why set HttpOnly Flag

## Extra Security with Policy Authentication Against API’s – Role Based

## Performance With API Caching

## Clean Code - Using Endpoint Extracted Extensions

# APIs

## Configurations

### Extend Identity Model\Database Table

### JWT

#### Bearer JWT Options

#### Bearer JWT Authentication

### Identity Options

#### Password

#### Lockout

#### Default provider

#### Confirmation Email When Registering

### 2FA Token Timespan

### Authorization Policy’s

### Capture JWT Bearer in Pipeline

### Cookie Options

#### HttpOnly

#### Remember Me Timespan

### CORS Policy

### API Output Caching

### Swagger Setup with JWT Security Requirement

## List of APIs Implemented and Scenarios on Usage

## Register | Login | Logout | Confirm Registration | Verify Registration | Resend Registration Confirm

## Verify 2FA | Enable 2FA | Disable 2FA | Resend 2FA

## JWT & Refresh Bearer - Revoke Tokens | Refresh Tokens (IMemory Cache)

## Remember Me Option (Persist user after browser closed – not Logout)

## Update User Details

## Request Password Update | Update Password

## Lockout | Unlock Account

# Miscellaneous

## Acronym

|  |  |
| --- | --- |
| **Abbreviation** | **Meaning** |
| EFC | Entity Framework Core |
| SQL | Structured Query Language |
| API | Application Programming Interface |
|  |  |
|  |  |