IAPF Blazor Toolset

Utility Library and Best Practices

This document covers the process of installing and getting to use a Blazor Net 8 Application in a development role. Basic Blazor concepts will be covered, as well as specific library functions available to the CVSHealth.IAM.IAPF.Tools.WebCoreUtility project which will be bundled as a NuGet package from Aetna Nexus.

Contents

[Setup Blazor Environment 2](#_Toc173760287)

[Creating a New Server Project 3](#_Toc173760288)

[App Settings JSON Setup 4](#_Toc173760289)

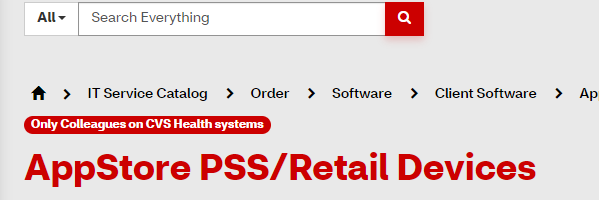
[Authentication & Authorization 6](#_Toc173760290)

[Web Core Utility Usage 7](#_Toc173760291)

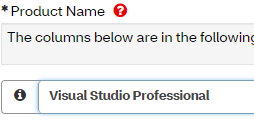
[Web Core Utility Class Definitions 8](#_Toc173760292)

# Setup Blazor Environment

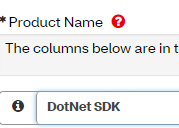
The local environment setup to be ready to work on a Blazor application is short with 2 main steps both done from [AppStore PSS/Retail Devices](https://itsm.cvs.com/sp_home?id=sc_cat_item&sys_id=051bdb7b1bf3f098ba5b77761a4bcb12&sysparm_category=ffb6395b6fcf150007a5f00dba3ee457)



1. **Install Visual Studio Professional 2022**



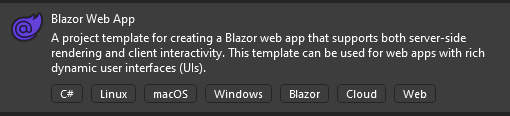
1. **Install DOTNET 8 SDK**



Alongside this connectivity to the IAPF GitHub after developer onboarding is necessary to work with existing projects.

# Creating a New Server Project

Visual Studio New Project uses Blazor Web App Template

Set Authentication Type to none, Interactive Render Mode to Server, and Interactivity to Global.



Create for Project a new Tests Project using the xUnit Template.

Add Reference Project dependencies:

**CVSHealth.IAM.IAPF.Tools.WebCoreUtility**

From Aetna Nexus NuGet Repository

# App Settings JSON Setup

Every application will need to have its app settings created with some factors in mind. NET Core applications use a JSON format file, named AppSettings.JSON, in order to configure themselves. In order to mount an application at a nested location on the site they are mounted in the format of: DomainUrl/ApplicationName . Examples:

iapf.tools.webdev.com (Application Shell is mounted at base site)

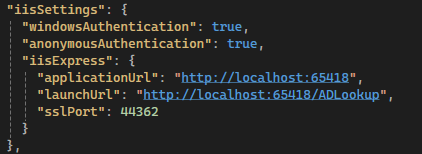
iapf.tools.webdev.com/EnableDisable (EnableDisable application would have its base HREF here)

When an application is given a base HREF other than the root domain, it needs to be setup in 4 locations so the application knows this sub folder is its root.

* AppSettings.JSON: “BaseUrl” : “/ApplicationName/”



* LaunchSettings.JSON: Affix each profile with a launchUrl of localhost/ApplicationName

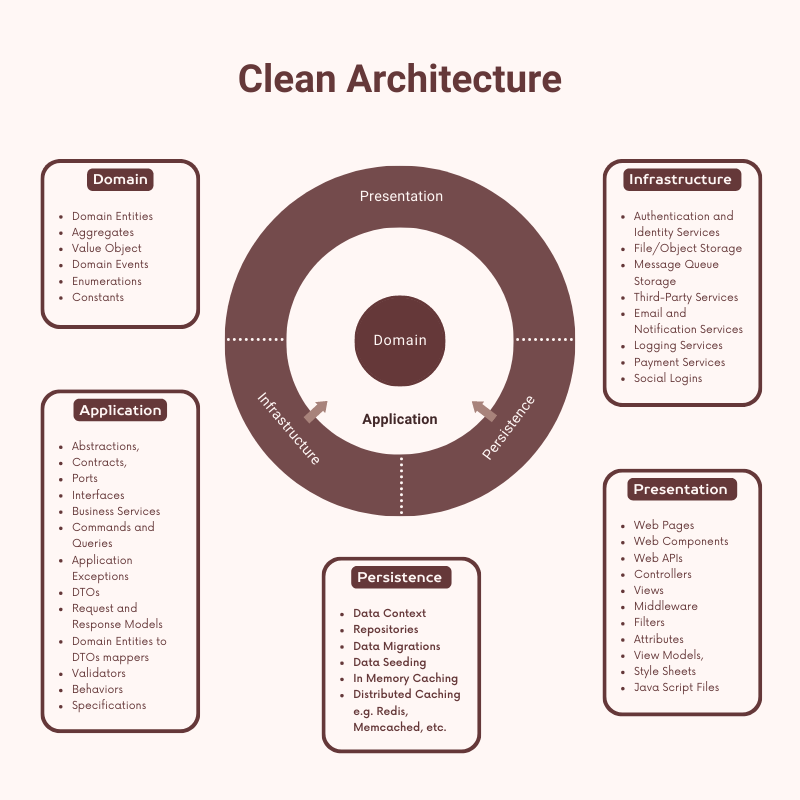


* app. UsePathBase(baseUrl) – These lines are added to DependencyInjection.cs to point the application to this base path.



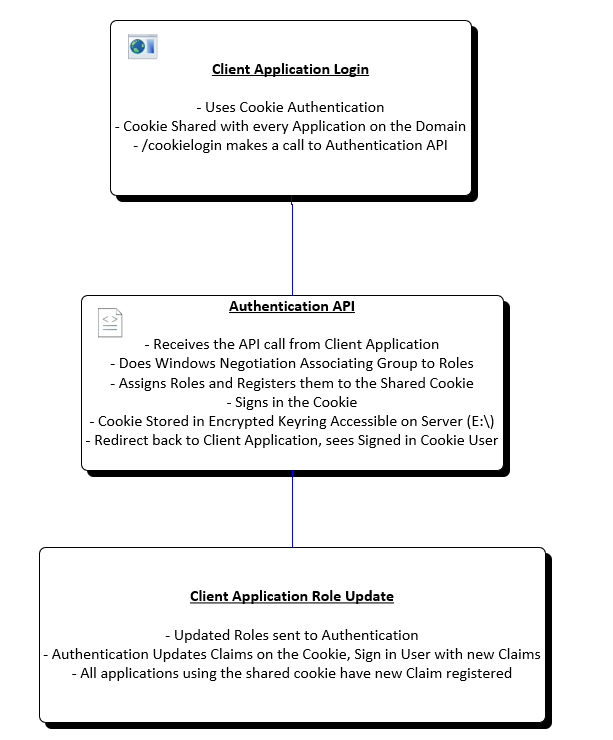
* App.Razor - <base href="@AppConfiguration!.BaseUrl" />





# Authentication & Authorization

All future projects and applications will use the Authentication API project to authenticate users to any application microservices. As soon as a user logs into any application it does a check against its current cookie to see if it is valid, if it is not a request goes to the Authentication API whose response is a set of user Roles and a login time stamp. If the time stamp is recent for the username, it remains valid across any application reducing load times and AD queries.



# Web Core Utility Usage

When adding a new project, it is important to understand the split of structure of what the Web Application (your project) and the structure of the WebCoreUtility project that every application will share.

**Host Application Structure**

│

├── \_Imports.Razor // Imports file for libraries and components.

├── App.Razor // Entry point for application HTTP DOM.

├── Routes.Razor // Defines application routing.

├── appsettings.json // Holds configuration settings in JSON format.

├── runtimeconfig.template.json // Enforces .NET Core versioning for host runtime.

├── Program.cs // Application startup point.

├── DependencyInjection.cs // Defines services and configuration for the app.

├── NLog.Config // Define NLog configuration for the app

├── Configuration

│ └── Web.config // Configuration for IIS hosting settings.

│

└── Components // UI components specific to the application.

├── Account // Pages related to login and authentication.

└── Pages // All other application routable pages.

**Web Core Utility Structure**

│

├── \_Imports.Razor // Library Imports for UI Components

│

├── wwwroot // Static web files (CSS, JS, etc.).

│

├── Domain // Models and structure tied to the core of the app.

│

├── Application // Files and logic specific to the application.

│

├── Infrastructure // Shared logic for interacting external services.

│

└── UIComponents // Presentation layer with shared UI elements

├── Layouts // Layouts used by any application.

├── Pages // Routable pages, unique from Host routes.

└── Components // Single components for use in any page.

# Web Core Utility Class Definitions

**Domain**

* **Entities**
  + **ApiPermissions**
    - Represents an API permission with its name and associated scopes.
  + **ApplicationUser**
    - Represents an application user, extending the IdentityUser class.
  + **AzureAdOptions**
    - Represents the Azure AD configuration options.
  + **DistributedCacheEntry**
    - Represents an entry in a distributed cache system.
    - This entity defines the structure for the DistributedCacheEntry table in the database.
  + **LDAPServer**
    - Gets or sets the name of the LDAP server table in the database.
  + **RemoteScript**
    - Represents remote script information.
    - This entity defines the structure for the RemoteScripts table in the database.
  + **ServiceAccount**
    - Represents configuration for service accounts in the application.
    - This entity defines the structure for the ServiceAccounts table in the database.
  + **ToolsConfiguration**
    - Represents configuration settings for various tools in the application.
    - This entity defines the structure for the ToolsConfiguration table in the database.
  + **UserSettings**
    - Represents user settings in the application.
    - This entity defines the structure for the UserSettings table in the database.

**Application**

* **Models**
  + **LoginModel**
    - Represents the login information for user authentication.
  + **LoginResponse**
    - Represents the response model for a login operation.
    - This class encapsulates all relevant information returned after a user authentication attempt.
  + **NavLinkInfo**
    - Represents information for a navigation link.
  + **RoleUpdateRequest**
    - Represents a request to update the roles and groups of a user.
    - This class is used in the application layer to encapsulate the data required for a role update operation.
  + **UserInfoResponse**
    - Represents the response model containing user information returned to the client.
    - This class encapsulates all relevant information about a user after a successful authentication or user info request.
  + **ViewModule**
    - Represents a view module with its associated URLs.

**Infrastructure**

* **Authentication**
  + **CookieHandler** -> Infrastructure.Authentiation.Interfaces.ICookieHandler
    - Handles the creation and addition of a cookie header for communication between Domain endpoints.
  + **CustomAuthenticationStateProvider**
    - Custom authentication state provider for handling user authentication and authorization.
    - This class extends RevalidatingServerAuthenticationStateProvider to provide additional functionality for managing user authentication state, caching, and user settings.
  + **ClaimsTransformation**
    - Provides custom claims transformation functionality for authenticated users.
  + **DomainUserGroupService** -> Infrastructure.Authentiation.Interfaces.IDomainUserGroupService
    - Provides services for retrieving user groups and roles from Active Directory.
  + **LdapAuthenticationService** -> Infrastructure.Authentiation.Interfaces.ILdapAuthenticationService
    - Provides LDAP authentication services, including credential validation and role fetching.
* **Authorization**
  + **ClaimsPrincipalExtensions**
    - Provides extension methods for ClaimsPrincipal to enhance authorization capabilities.
* **Common**
  + **CustomDistributedCache**
    - Provides a custom implementation of IDistributedCache that uses IDistributedCacheRepository.
  + **Configuration**
    - **ConfigurationInitializationService**
      * A hosted service that initializes configuration data on application startup.
    - **LdapRoleMappingConfig**
      * Manages the mapping of LDAP roles to application roles.
    - **LdapServerList**
      * Manages the list of LDAP servers.
    - **NavLinksInfoList**
      * Manages the list of navigation menu links.
    - **RemoteScriptList**
      * Manages the list remote scripts.
    - **ServerAppConfiguration** -> Application.Common.Configuration.Interfaces.IAppConfiguration
      * Retrieves and manages configuration values from appsettings.json and the registry.
    - **ServiceAccountList**
      * Manages the list of service account configurations.
    - **UserGroupsCache**
      * Provides caching functionality for user groups and roles.
    - **ViewModuleList**
      * Manages a list of modules for the NavMenu lookup.
  + **Data**
    - **Repositories**
      * **DistributedCacheRepository** -> Application.Interfaces.IDistributedCacheRepository
        + Repository implementation for distributed cache operations using Entity Framework Core.
      * **LDAPServerRepository** -> Application.Interfaces.ILDAPServerRepository
        + Repository implementation for LDAPServer operations using Entity Framework Core.
      * **RemoteScriptRepository** -> Application.Interfaces.IRemoteScriptRepository
        + Repository implementation for RemoteScript operations using Entity Framework Core.
      * **ServiceAccountRepository** -> Application.Interfaces.IServiceAccountRepository
        + Repository implementation for ServiceAccount operations using Entity Framework Core.
      * **ToolsConfigurationRepository** -> Application.Interfaces.IToolsConfigurationRepository
        + Repository implementation for ToolsConfiguration operations using Entity Framework Core.
      * **UserSettingsRepository** -> Application.Interfaces.IUserSettingsRepository
        + Repository implementation for UserSettings operations using Entity Framework Core.
        + This class handles the data access logic for user settings in the application.
    - **AppDbContext**
      * Initializes a new instance of the AppDbContext class.
  + **Navigation**
    - **IdentityRedirectManager**
      * Manages redirects for identity-related operations in the application.
  + **Services**
    - **AzureAdOptionsService**
      * Service for retrieving and managing Azure AD configuration options.
    - **ClaimsUpdateService** -> Infrastructure.Services.Interfaces.IClaimsUpdateService
      * Service for handling real-time updates to user claims via SignalR.
    - **InitializationService** -> Infrastructure.Services.Interfaces.IInitializationService
      * Provides thread-safe initialization services.
    - **RegistryHelperService** -> Infrastructure.Services.Interfaces.IRegistryHelperService
      * Service for retrieving values from Windows Registry Configuration Paths.
    - **RequestRefresh** -> Infrastructure.Services.Interfaces.IRequestRefresh
      * Service to trigger a refresh in one component from another.
    - **ThemeService** -> Infrastructure.Services.Interfaces.IThemeService
      * Service for managing theme settings and dark mode preferences.
    - **UserSettingsService** -> Infrastructure.Services.Interfaces.IUserSettingsService
      * Service for managing user preferences and settings.
  + **Themes**
    - **PaletteThemes**
      * Provides pre-defined color palettes for use with MudBlazor themes.

**UIComponents**

* **Components**
  + **AppBar.Razor**
    - Represents the application bar component.
  + **MenuBar.Razor**
    - Represents the menu bar component of the application.
  + **NavMenu.Razor**
    - Represents the navigation menu component of the application.
* **Layouts**
  + **MainLayout**
    - Represents the main layout component of the application.
  + **ModuleLayout**
    - Represents the module layout component of the application.
  + **PublicLayout**
    - Represents the public layout component of the application.
* **Pages**
  + **Counter**
    - Represents the sample Counter page component.
  + **Weather**
    - Represents the sample Weather page component.
  + **Error**
    - Represents the Error page component.