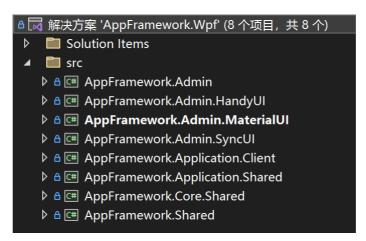
# WPF 项目部署

当您打开服务器端解决方案 (AppFramework.Wpf.sln) 使用 Visual Studio 2022, 您将看到解决方案结构如下:



右键单击任意一种 UI 框架,然后选择"**设置为启动项目**"。然后**生成**解决方案。在第一次生成期间可能需要更长的时间,因为将还原所有 **nuget** 包。

## 服务地址配置

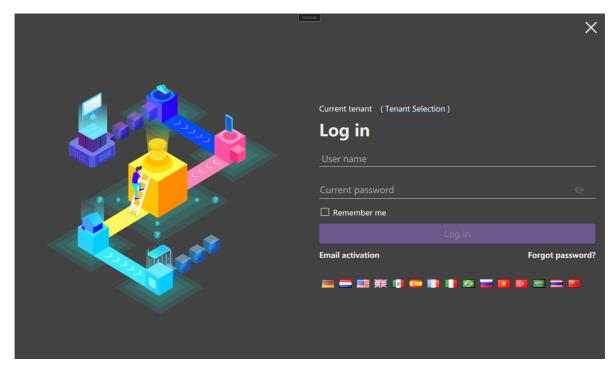
在 AppFramework.Application.Client 项目中打开 ApiUrlConfig 。并根据需要更改默认服务器地址:

```
nespace AppFramework.ApiClient
                                                                                                 ▶ 🗈 💷 AppFramework.Admin
                                                                                                 ▷ 🛎 🕮 AppFramework.Admin.HandyUI
                                                                                                 ▷ 🗈 🖾 AppFramework.Admin.MaterialUI
                                                                                                 ▷ 🛎 🖼 AppFramework.Admin.SyncUI
     public static string DefaultHostUrl => "https://localhost:44301/";
                                                                                                  public static string BaseUrl { get; private set; }
                                                                                                    🛮 角 🛅 ApiClient
                                                                                                      ▶ a C# AbpApiClient.cs
     static ApiUrlConfig()
                                                                                                      ▶ A C# AccessTokenManager.cs
                                                                                                      ▶ ≜ C# ApiUrlConfig.cs
         ResetBaseUrl();
                                                                                                      ▶ & C# ApplicationContext.cs
                                                                                                      ▶ â C# AuthenticationHttpHandler.cs
                                                                                                      b ≜ C# DebugServerlpAddresses.cs
     public static void ChangeBaseUrl(string baseUrl)
                                                                                                      ► 6 C# IAccessTokenManager.cs
         BaseUrl = ReplaceLocalhost(NormalizeUrl(baseUrl));
                                                                                                      ▶ ≜ C# ModernHttpClientFactory.cs
```

注: 默认情况下, 服务器地址为 https://localhost:44301/, 实际生产环境中, 请更改为实际的地址部署。

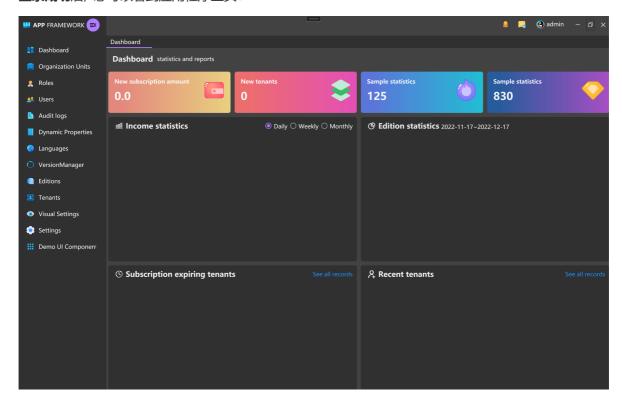
# 运行WPF项目

确保WebAPI启动的情况下,运行WPF项目。您将看到如下所示的登录页面:



注意: 默认的账号密码为 admin、123qwe

登录成功后, 您可以看到应用程序主页:



# 探索WPF项目

## 启动流程

- RegisterTypes 注册容器服务 (包含应用程序依赖的所有服务、页面等)
- ConfigureRegionAdapterMappings 注册适配器 (Prism适配器)
- OnInitialized 重写初始化流程
- 版本更新检查

位于App.xaml.cs 文件

```
var appVersionService = ContainerLocator.Container.Resolve<IUpdateService>();
await appVersionService.CheckVersion();
```

• 初始化屏幕(下载服务器资源/验收授权信息)

位于xxxUIStartService 文件

具体实现中 SplashScreenViewModel 文件中

```
public override async void OnDialogOpened(IDialogParameters parameters)
            await SetBusyAsync(async () =>
            {
               await Task.Delay(200);
               //加载本地的缓存信息
               DisplayText = LocalTranslationHelper.Localize("Initializing");
               accessTokenManager.AuthenticateResult =
dataStorageService.RetrieveAuthenticateResult();
               applicationContext.Load(dataStorageService.RetrieveTenantInfo(),
dataStorageService.RetrieveLoginInfo());
               //加载系统资源
               DisplayText = LocalTranslationHelper.Localize("LoadResource");
               await UserConfigurationManager.GetIfNeedsAsync();
               //如果本地授权存在,直接进入系统首页
               if (accessTokenManager.IsUserLoggedIn &&
applicationContext.Configuration != null)
                   OnDialogClosed();
               else if (applicationContext.Configuration!=null)
                   OnDialogClosed(ButtonResult.Ignore);
               else
                   OnDialogClosed(ButtonResult.No);
            });
       }
```

• 进入首页

加载用户资料,导航面板页,位于 MainTabViewModel

## 多UI框架

本套框架实现了三种UI框架: Syncfusion、MaterialDesign、HandyControl。

在每一种UI框架中,各自实现了基于不通UI框架的样式资源定义、特定的主题切换功能、视图定义。

启动时, 会将Admin模块中的ViewModel与各自UI中的 View 进行绑定注册。

注: Syncfusion 为商用框架,需授权使用,注册社区版本即可使用。

#### **MVVM**

该框架中,内置了几种MVVM基类,分别应用在不同的场景中使用。

- DialogViewModel 弹窗模块中使用,不具备阴影遮挡效果。
- HostDialogViewModel 弹窗模块中使用,具备阴影遮挡效果。
- NavigationCurdViewModel 具备导航功能的单个表的基础实现、新增、编辑、分页。
- NavigationViewModel 具备导航功能实现。
- ViewModelBase 实现绑定通知基类、包含实体映射、验证器、等待属性定义。

## 本地化多语言

多语言由 TranslateExtension 标记扩展实现,UI视图中绑定多语言转换的Key字符串,在TranslateExtension

的 ProvideValue 中, 通过 Local.Localize(string text) 静态方法转换成特定语言的字符串。

视图中定义:

```
<TextBox md:HintAssist.Hint="{extensions:Translate UserName}"

Text="{Binding UserName}" />
```

TranslateExtension 实现:

```
/// <summary>
   /// 字符串多语言转换扩展
   /// </summary>
   public class TranslateExtension : MarkupExtension
   {
      public TranslateExtension(string text)
      {
            Text = text;
      }
      /// <summary>
```

```
/// 本地化字符串
        /// </summary>
        public string Text { get; set; }
        /// <summary>
       /// 返回当前语言的对应文本
       /// </summary>
        /// <param name="serviceProvider"></param>
        /// <returns></returns>
        public override object ProvideValue(IServiceProvider serviceProvider)
            if (serviceProvider.GetService(typeof(IProvideValueTarget)) is
IProvideValueTarget target
               && target.TargetObject is FrameworkElement element
               && DesignerProperties.GetIsInDesignMode(element))
            {
               return Text;
            }
            if (string.IsNullOrWhiteSpace(Text)) return Text;
            return Local.Localize(Text);
       }
   }
```

注: Localize 静态方法中 通过使用 IApplicationContext 转换具体多语言字符串, 该接口中的多语言数据从后端的WebAPI中获得, 参考: UserConfigurationManager.GetAsync() 实现。

## 身份授权

客户端通过 IAccountService 接口进行身份授权,如下图所示:

```
→ AppFramework.Admin.Services.Account.AccountService
                                                                          - € dialog
                                                                                       ) A 🐙 🐚 + 与 🗐 🖆 🔼 + 🔑 🛋
    this.AuthenticateModel = authenticateModel;
                                                                                    搜索解决方案资源管理器(Ctrl+;)
                                                                                    â 🔜 解決方案 'AppFramework.Wpf' □(8 个项目, 共 8 个)
                                                                                       Solution Items
                                                                                      ▲ 6 © AppFramework.Admin

▶ 8日 依赖项
public AbpAuthenticateModel AuthenticateModel { get; set; }
                                                                                        ▶ a ■ Behaviors▶ a ■ Extensions
public AbpAuthenticateResultModel AuthenticateResultModel { get; set;
                                                                                        ▲ 🗈 🚞 Services
public async Task LoginUserAsync()
                                                                                            ▶ a C# AccountService.cs
    var result = await accessTokenManager.LoginAsync();
                                                                                            ▶ a C# ApplicationService.cs
    await AuthenticateSucceed(result);
                                                                                            ▶ a C# IApplicationService.cs
                                                                                          ▶ a m Features
public async Task LoginCurrentUserAsync(UserListModel user)
                                                                                          ▶ â ■ Notification
                                                                                          ▶ a ■ Permission
                                                                                          accessTokenManager.Logout();
                                                                                        applicationContext.ClearLoginInfo();
    dataStorageService.ClearSessionPersistance();
                                                                                      ▷ 🐧 📧 AppFramework.Admin.HandyUI
                                                                                       ▶ △ ☑ AppFramework.Admin.SyncUI
    await GoToLoginPageAsync();
```

在 accessTokenManager 实现中, 包含授权接口地址以及刷新身份令牌的接口地址:

```
private const string LoginUrlSegment = "api/TokenAuth/Authenticate";
private const string RefreshTokenUrlSegment = "api/TokenAuth/RefreshToken";
```

授权代码如下所示:

创建一个api客户端,然后构造请求资源地址,通过Post传递一个JSON的授权信息,其中包含用户名、密码等参数,最终返回一个授权结果,该结果当中,包含AccessToken 用于其它受权限的资源访问。

在框架中, 封装了用于Web请求的操作类, 在 Application. Client项目中可以查看:

```
→ 🌣 AppFramework.ApiClient.AbpApiClient
                                                                                                      ) 슈 🚜 ष - 与 🗇 🗗 🍢 - 🗡 🛎
                                                                                                  搜索解决方案资源管理器(Ctrl+;)
                                                                                                  â 🛜 解决方案 'AppFramework.Wpf' 🛭 (8 个项目 , 共 8 个
ublic class AbpApiClient : ISingletonDependency, IDisposable
                                                                                                   Solution Items
                                                                                                   ▲ 🛅 src
                                                                                                    ▶ ≜ □ AppFramework.Admin
   private readonly IAccessTokenManager _accessTokenManager;
                                                                                                     ▶ å ■ AppFramework.Admin.HandyUI
   private readonly IApplicationContext _applicationContext;

    ▶ â ■ AppFramework.Admin.MaterialUI
    ▶ â ■ AppFramework.Admin.SyncUI

   private readonly IMultiTenancyConfig _multiTenancyConfig;
                                                                                                     ▲ 6 回 AppFramework.Application.Client
→ 路 依赖项
   private static FlurlClient _client;
                                                                                                       ▲ 6  ApiClient

▶ 6  Models
   public static int? TimeoutSeconds { get; set; } = 30;
                                                                                                       ▶ å C# AbpApiClient.cs
                                                                                                         ▶ a C# AccessTokenManager.cs
                                                                                                         ▶ A C# ApiUrlConfig.cs
                                                                                                         ▶ ♠ C# ApplicationContext.cs
   public AbpApiClient(
                                                                                                         ▶ ≜ C# AuthenticationHttpHandler.cs
        IAccessTokenManager accessTokenManager,
                                                                                                         ▶ ≜ C# DebugServerIpAddresses.cs
         IApplicationContext applicationContext,
                                                                                                         ▶ A C# IAccessTokenManager.cs
        IMultiTenancyConfig multiTenancyConfig)...
                                                                                                         ▶ & C# ModernHttpClientFactory.cs
                                                                                                       Da 🛅 Common
                                                                                                       ▶ a  Configuration
                                                                                                       ▶ 🖴 🚞 Edition
```

AbpApiClient封装了 Get、Post、Put、Delete、文件下载等方法,其中还包含了创建API客户端的静态方法,由于构造Web服务请求的Header,如: 当前的访问资源语言、租户信息、授权令牌等设置,如下所示:

#### 自动更新

自动更新功能通过 IUpdateService 接口实现, 其中包含检查当前应用程序最新版本,如果存在新的版本信息,将在首次登录时进行通知提示,更新过程需中断应用程序下载,完成后自动打开应用程序。

该功能需要配合 上传应用程序的版本信息实现,首先上传最新的版本的压缩包至服务器。上传的版本号需大于当前应用程序的版本号,即可收到更新通知,自动更新功能通过开源组件: AutoUpdaterDotNET 实现。

参考资源: AutoUpdater

## 即时通讯

该功能支持应用程序内的注册用户进行文字、图像、文件上传等交流,该功能主要通过ASP.NET Core Signalr实现。

• Web服务实现

Web服务实现在 Web.Core 项目中可以找到,如下所示:

```
using Castle.Windsor;
                                                                                    搜索解决方案资源管理器(Ctrl+;)
using AppFramework.Chat;
                                                                                      Solu

¬namespace AppFramework.Web.Chat.SignalR

    ▶ â ■ AppFramework.Core.Shared
    ▶ â ■ AppFramework.EntityFramework

     public class ChatHub : OnlineClientHubBase
                                                                                      private readonly IChatMessageManager _chatMessageManager;
         private readonly ILocalizationManager _localizationManager;
         private readonly IWindsorContainer _windsorContainer;
                                                                                        ▷ and 依赖项
▷ a ■ Authentication
         private bool _isCallByRelease;
                                                                                        ▲ â 🛅 Chat
                                                                                          ▲ â ■ SignalR

▷ â C# ChatHub.cs
         private IAbpSession ChatAbpSession { get; }
                                                                                            ▶ a C# SendChatMessageInput.cs

    A C# SendFriendshipRequestInput.cs
    A C# SignalRChatCommunicator.cs

         /// Initializes a new instance of the <see cref="ChatHub"/> class
                                                                                        ▶ â 🛅 Common
                                                                                        public ChatHub(
              IChatMessageManager chatMessageManager,
                                                                                        a ☐ HealthCheck
              ILocalizationManager localizationManager,
              IWindsorContainer windsorContainer,
                                                                                        ▶ a light IdentityServer
              IOnlineClientManager<ChatChannel> onlineClientManager,
                                                                                        ▶ A ■ Models
              IOnlineClientInfoProvider clientInfoProvider) : base(onlineCli
```

即时通讯功能需要验证用户的授权信息,当授权信息通过后,并且检测用户连接至Signalr服务。

```
▼ 😭 Configure(IServiceCollection services, IConfiguratior ▼ 🛨
                 → 
<sup>A</sup>g AppFramework.Web.Startup.AuthConfigurer
                                                                                                       ^ ^ 세 🏿 · → 🗏 @ 🖫 · 🎤 ≕
                                                                                                   搜索解决方案资源管理器(Ctrl+;
public static class AuthConfigurer
                                                                                                   å 🗔 解决方案 'AppFramework.Web' 🛚 (10 个项目,共 10
                                                                                                      Solution Items
     public static void Configure(IServiceCollection services, IConfigura_
                                                                                                      ▶ ≜ ☐ AppFramework.Application.Shared
                                                                                                      var authenticationBuilder = services.AddAuthentication();
                                                                                                      ▶ a 	 AppFramework.Core.Shared
                                                                                                      ▷ 🗗 💷 AppFramework.EntityFrameworkCore
          if (bool.Parse(configuration["Authentication:JwtBearer:IsEnabled")
                                                                                                      D a ☐ AppFramework.Migrator
D a ☐ AppFramework.Web.Core
                authenticationBuilder.AddAbpAsyncJwtBearer(options =>
                                                                                                      🗸 🐧 🎒 AppFramework.Web.Host
                                                                                                          Connected Services
                                                                                                        D A ● Properties
D A ● www.root
D AP 依赖项
                     options.TokenValidationParameters = new TokenValidationP
                                                                                                        ▶ a  Controllers
                     options.SecurityTokenValidators.Clear();
                                                                                                        ▶ a ■ Models
                     options.AsyncSecurityTokenValidators.Add(new AppFramewor

▲ A ■ Startup

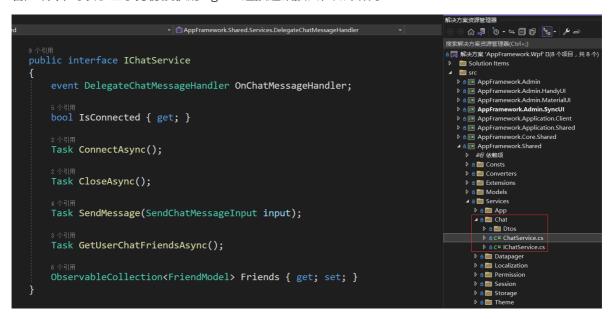
    D △ □ ExternalLoginInfoProviders
    D △ C = AppFrameworkWebHostModule.cs

                     options.Events = new JwtBearerEvents
                                                                                                         ▶ a C# AuthConfigurer.cs
                                                                                                          OnMessageReceived = QueryStringTokenResolver
                                                                                                        Þ ≜ 🛅 Url
                     };
                });
                                                                                                        app.config
appsettings.json
                                                                                                        ▶ a □ Dockerfile
```

如上图所示, 在接受到授权信息后, QueryStringTokenResolver 会检查请求的资源地址是否包含Signalr相关信息。

• WPF客户端实现

客户端中, 封装了基于身份授权的Signalr连接通讯接口, 如下所示:



如上图所示, 包含了连接服务、断开服务、发送消息、以及获取好友信息等实现。 具体实现可以参考ChatService ,实现库: <u>Microsoft.AspNetCore.SignalR.Client</u> 参考资料:

使用 SignalR 进行实时 ASP.NET

SignalR Learn