WPF应用中融入ASP.NET Core：实操指南及代码示例

作者：微信公众号：【架构师老卢】 12-25 18:3 134

概述：本文详细阐述了如何在WPF应用程序中宿主ASP.NET Core服务的方法及步骤。首先创建一个ASP.NET Core服务项目，然后在WPF应用中启动该服务。同时，提供了详细的代码实例以及注释，让读者能更易理解和实践。

在WPF中宿主一个ASP.NET Core Web API服务，并实现WPF与Web API的交互涉及到将两者整合在一起，以便实现数据的传输与交互。以下是详细的步骤和示例代码：

1. 原理

在WPF中宿主一个ASP.NET Core Web API服务的原理是通过WPF应用程序启动时创建一个内嵌的Web服务器，该服务器托管ASP.NET Core Web API。WPF应用程序通过HTTP协议与Web API进行通信，实现数据的传输与交互。

2. 方法

实现这一目标的方法主要包括以下几个步骤：

1. **创建ASP.NET Core Web API项目**：使用Visual Studio创建一个ASP.NET Core Web API项目，定义API接口以及相应的控制器。
2. **配置Web API服务**：确保Web API能够在WPF应用程序中访问，配置CORS策略以允许WPF应用程序进行跨域请求。
3. **在WPF中集成Web API服务**：在WPF应用程序中集成ASP.NET Core的WebHostBuilder，并在应用程序启动时启动Web API服务。
4. **实现WPF与Web API的交互**：使用HttpClient等工具在WPF中发送HTTP请求，与Web API进行通信，实现数据的获取和更新。

3. 步骤

步骤一：创建ASP.NET Core Web API项目

在Visual Studio中创建一个ASP.NET Core Web API项目，并定义一些API接口和控制器。示例代码如下：

```
// UsersController.cs
[Route("api/[controller]")]
[ApiController]
public class UsersController : ControllerBase
{
    [HttpGet]
    public ActionResult<IEnumerable<string>> Get()
    {
        return new string[] { "user1", "user2" };
    }
}
```

步骤二：配置Web API服务

在`Startup.cs`文件中配置CORS策略，允许WPF应用程序进行跨域请求：

```
// Startup.cs
public void ConfigureServices(IServiceCollection services)
{
    services.AddCors(options =>
    {
        options.AddPolicy("AllowWpfApp",
            builder => builder.WithOrigins("http://localhost:YourWpfPort")
                              .AllowAnyHeader()
                              .AllowAnyMethod());
    });
    // Other configurations...
}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.UseCors("AllowWpfApp");
    // Other configurations...
}
```

步骤三：在WPF中集成Web API服务

在WPF应用程序中使用`WebHostBuilder`启动Web API服务：

```
// App.xaml.cs
public partial class App : Application
{
    private IWebHost _webHost;

    protected override void OnStartup(StartupEventArgs e)
    {
        _webHost = new WebHostBuilder()
            .UseKestrel()
            .UseStartup<Startup>()
            .UseUrls("http://localhost:YourWpfPort")
            .Build();

        _webHost.Start();

        base.OnStartup(e);
    }

    protected override void OnExit(ExitEventArgs e)
    {
        _webHost?.Dispose();
        base.OnExit(e);
    }
}
```

步骤四：实现WPF与Web API的交互

在WPF应用程序中使用`HttpClient`等工具进行HTTP请求，与Web API进行通信。以下是一个简单的示例：

```
// MainWindow.xaml.cs
public partial class MainWindow : Window
{
    private const string ApiBaseUrl = "http://localhost:YourWpfPort/api/users";

    public MainWindow()
    {
        InitializeComponent();
        LoadUserData();
    }

    private async void LoadUserData()
    {
        using (var httpClient = new HttpClient())
        {
            var response = await httpClient.GetStringAsync(ApiBaseUrl);
            var users = JsonConvert.DeserializeObject<IEnumerable<string>>(response);

            foreach (var user in users)
            {
                // Handle user data in WPF application
                listBoxUsers.Items.Add(user);
            }
        }
    }
}
```

4. 示例源代码

以下是一个简化的示例源代码，实现了上述步骤：

```
// ASP.NET Core Web API Controller
[Route("api/[controller]")]
[ApiController]
public class UsersController : ControllerBase
{
    [HttpGet]
    public ActionResult<IEnumerable<string>> Get()
    {
        return new string[] { "user1", "user2" };
    }
}

// ASP.NET Core Startup
public class Startup
{
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddCors(options =>
        {
            options.AddPolicy("AllowWpfApp",
                builder => builder.WithOrigins("http://localhost:YourWpfPort")
                                  .AllowAnyHeader()
                                  .AllowAnyMethod());
        });
        // Other configurations...
    }

    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        app.UseCors("AllowWpfApp");
        // Other configurations...
    }
}

// WPF App
public partial class App : Application
{
    private IWebHost _webHost;

    protected override void OnStartup(StartupEventArgs e)
    {
        _webHost = new WebHostBuilder()
            .UseKestrel()
            .UseStartup<Startup>()
            .UseUrls("http://localhost:YourWpfPort")
            .Build();

        _webHost.Start();

        base.OnStartup(e);
    }

    protected override void OnExit(ExitEventArgs e)
    {
        _webHost?.Dispose();
        base.OnExit(e);
    }
}

// WPF MainWindow
public partial class MainWindow : Window
{
    private const string ApiBaseUrl = "http://localhost:YourWpfPort/api/users";

    public MainWindow()
    {
        InitializeComponent();
        LoadUserData();
    }

    private async void LoadUserData()
    {
        using (var httpClient = new HttpClient())
        {
            var response = await httpClient.GetStringAsync(ApiBaseUrl);
            var users = JsonConvert.DeserializeObject<IEnumerable<string>>(response);

            foreach (var user in users)
            {
                // Handle user data in WPF application
                listBoxUsers.Items.Add(user);
            }
        }
    }
}
```

请注意替换示例代码中的`YourWpfPort`为实际的WPF应用程序端口。此示例仅为演示目的，实际应用中需要根据具体情况进行更详细的配置和优化。