Creating a Blazor Project

Introduction

This guide will help you set up a Blazor WebAssembly project in Visual Studio Code. You'll learn how to create the project, configure essential settings, manage dependencies, and use `appsettings.json` for flexible configuration. These steps provide a solid foundation for building Blazor applications with custom settings and external dependencies.

Steps

Step 1: Create a New Blazor Project

1. Open Visual Studio Code and launch the terminal.
2. Type the following command to create a new Blazor WebAssembly project:

```
dotnet new blazorwasm -o MyFirstBlazorApp
```

3. Press Enter. This command creates a new Blazor app in a folder named "MyFirstBlazorApp."

Step 2: Open and Edit the `.csproj` File

1. Locate the `.csproj` file in the project's root directory (inside the "MyFirstBlazorApp" folder).
2. Open the file in Visual Studio Code. This file controls build settings and dependencies for your Blazor project.
3. To change the target framework:

- Find the `<TargetFramework>` element within `<PropertyGroup>` and update its value, for example:

```
<TargetFramework>net6.0</TargetFramework>
```

4. To set the output type and root namespace:

- Add the following lines in the `<PropertyGroup>` section:

```
<OutputType>Exe</OutputType>
```

```
<RootNamespace>MyBlazorApp</RootNamespace>
```

Step 3: Add Dependencies

1. Inside the `.csproj` file, locate or add an `<ItemGroup>` section.
2. To add a NuGet package:

- Add a `<PackageReference>` entry for each package you need. For example, to add `Newtonsoft.Json`, use:

```
<ItemGroup>
  <PackageReference Include="Newtonsoft.Json" Version="13.0.1" />
</ItemGroup>
```

3. Save the file to allow Visual Studio Code to download and include the new package in your project.

Step 4: Use `appsettings.json` for Application Settings

1. Check if an `appsettings.json` file exists in your project root. If not, create one.
2. Open `appsettings.json` and add key-value pairs for your configuration. For example, to add an API base URL:

```
{
  "ApiSettings": {
    "BaseUrl": "https://api.myapp.com"
  }
}
```

3. Save the file. This file stores settings outside of the code, making updates easier.

Step 5: Access Settings in Code

1. Create a Service Class:

- In your project, create a folder named `Services`.
- Inside `Services`, create a new file named `ApiService.cs.`
- Add the following code to `ApiService.cs` to retrieve settings from `appsettings.json`:

```
using Microsoft.Extensions.Configuration;

public class ApiService
```

```
{
    private readonly IConfiguration _configuration;
           public ApiService(IConfiguration configuration)
    {
        _configuration = configuration;
    }
    public string GetApiUrl()
    {
        return _configuration["ApiSettings:BaseUrl"];
    }
}
```

2. Register the Service:

- Open `Program.cs` and add your new service to the service container:

```
builder.Services.AddSingleton<ApiService>();
```

Conclusion

Following these steps, you have created a Blazor WebAssembly project, configured project settings, managed dependencies, and set up appsettings.json for centralized configuration. This setup is ready for further development and customizations, providing a solid base for your Blazor application’s needs.