DaffittTech Network Status

Overview

DaffittTech.NetworkStatus is a Blazor utility package for checking and monitoring Internet connectivity in Blazor applications. It leverages JSInterop to interact with browser APIs to communicate with JavaScript functions that do the connection status workload and then relay the results to the NetworkConnection component.

This README file is relevant to this NuGet package's current version, so older versions may vary. To take advantage of this Blazor utility package, follow the instructions below.

Software Dependencies

- **NuGet Package:** DaffittTech.NetworkStatus
- **Latest Release:** The current release version is 2.0.0
- **Target Framework:** .NET 9.0
- **Dependencies:** Microsoft.ASPNetCore.Components.Web (>= 8.0.6)
- **License:** This package is licensed under the MIT License. See the LICENSE.txt file for more information.

Features

- Detects online/offline status in real time or on demand.
- Blazor component for UI integration.
- Service for programmatic access to network status.
- Customizable status check intervals.
- Easy integration with Blazor projects.

Setting It Up

There are several steps to set up and use DaffittTech.NetworkStatus package.

1. Install the NuGet package
2. Register the package
3. Add the directive to the _Imports.razor file
4. Application entry point
5. Inject the NetworkService into your Blazor code
6. Add some properties
7. Add the NetworkConnection component to your UI

Install NuGet Package

Depending on your chosen Blazor project configuration, you will likely need to add this package to the application's Client-side (UI) and server-side based on where your pages and components are running from. However, you may need to experiment, as Microsoft likes to change technologies on a moment-to-moment basis.

1. The most straightforward way to add this service to your Blazor application is to use the Package Manager in Visual Studio. Search for Daffitt Technologies and select DaffittTech.NetworkStatus package, choose the projects you wish to install it into (client-side and/or server-side), and click install.

2. You can also go to https://www.nuget.org/packages/DaffittTech.NetworkStatus and follow the directions for your favorite method of installing NuGet packages.

Register The NetworkService

Add the using statement and the NetworkService to the `Programs.cs` file. This will need to be added to both server-side and client-side if your project uses both. We are using AddScoped rather than AddSingleton or AddTransiant because this package uses JSInterop which requires the component to be registered as Scoped.

```csharp
using DaffittTech.NetworkStatus;
// Other stuff goes here...
builder.Services.AddScoped<NetworkService>();
```

_Imports.razor

To prevent having to add this next line at the top of every .razor file where you need it, you can add this line to the `_Imports.razor` file.

```html
@using DaffittTech.NetworkStatus
```

If you separate your *.razor* page from its *.cs* page, you may also need to add the line above to the using section in the *.cs* page (without the "@" symbol).

Application Entry Point

Add the following line to your end of the section if your `App.razor` or `_host.cshtml` or `index.html` file.

```html
<body>
    <!-- Other stuff goes here -->
    <script src="_content/DaffittTech.NetworkStatus/NetworkService.js"></script>
</body>
```

Inject NetworkService Into Your Blazor Code

If you keep your *.razor* code and backend *.cs* code together in the same razor file, add this line at the top of the razor page.

```html
@inject NetworkService NetworkService
```

If you separate your *.razor* page from its *.cs* page, add this line to the partial class of the page. You'll also want to add a private property to hold the value of the network status.

```csharp
[Inject] protected NetworkService NetworkService { get; set; }
```

Add Some Properties

To set the Interval and the Timeout parameters of the component, you'll need to add a couple of properties to hold those values. You could enter them directly, but that's not good practice. To the code section or the partial .cs file add these properties.

```csharp
private int? Interval { get; set; } = null; // Seconds between auto status checking.
private double? NetworkStatusTimeout { get; set; } = null; // Time allowed in seconds for checking status
```

Add NetworkConnection Component To The UI

To use the component, just add it as an HTML element to the `.razor` page. My example here allows the component to live near the bottom of it's container in a project that uses bootstrap.

```html
<div class="d-flex flex-column position-absolute bottom-0 px-3 text-center w-100">
    <NetworkConnection Interval="@Interval" Timeout="@NetworkStatusTimeout" />
</div>
```

If you are not using bootstrap, you may get the same affect with this line.

```html
<div style="display: flex; flex-direction: column; position: absolute; bottom: 0; padding: 0 16px; width: 100%">
    <NetworkConnection Interval="@Interval" Timeout="@NetworkStatusTimeout" />
</div>
```

Parameters

Set the `Interval` property (in seconds) above zero to check network status regularly; set it to null to disable automatic checks. *Note:* This setting will repeatedly send checks until the NetworkStatus component is disposed.

The `NetworkStatusTimeout` property sets how long (as a nullable double in seconds) to wait for a response before timing out. Values greater than zero are converted to milliseconds for the timeout. If null or zero, a default of 1.0 second applies.

Code Section Example

Finally, set up an event listener and handler to capture the action event of the network status change. You'll also need to trigger updates, respond to the change using the OnAfterRenderAsync and UpdateNetworkStatus methods, and finally call a dispose to clean it up.

```csharp
using DaffittTech.NetworkStatus;
using Microsoft.AspNetCore.Components;

namespace DaffittSampleBlazorApp.Client.Layout
{
    public partial class NavMenu : ComponentBase, IDisposable
    {
        [Inject] protected NetworkService _networkStatusService { get; set; }

        private int? Interval { get; set; } = null; // Seconds between auto
status checking.
        private double? NetworkStatusTimeout { get; set; } = null; // Time
allowed in seconds for checking status

        protected override async Task OnInitializedAsync()
        {
            _networkStatusService.OnNetworkStatusChanged += UpdateNetworkStatus;
            await base.OnInitializedAsync();
        }

        protected override async Task OnAfterRenderAsync(bool firstRender)
        {
            if (firstRender)
            {
                // This step is optional, as the component will automatically
initialize and
                // conduct an initial status check upon being rendered within
the application.
                await
_networkStatusService.GetStatusAsync(NetworkStatusTimeout);
            }
            await base.OnAfterRenderAsync(firstRender);
        }

        private void UpdateNetworkStatus(bool status)
        {
            // Do some stuff here with the "status" if you are so inclined...
        }

        public void Dispose()
        {
            _networkStatusService.OnNetworkStatusChanged -= UpdateNetworkStatus;
            _networkStatusService.Dispose();
        }
```

```
    }
}
```

If you are placing your UI content and @code content in the same .razor page remember to add the reference to IDisposable at the top like so...

```csharp
@implements IDisposable
```

Sample Screen Shot
