Use ASP.NET Core SignalR with Blazor

Article • 11/17/2023

This tutorial provides a basic working experience for building a real-time app using SignalR with Blazor. This article is useful for developers who are already familiar with SignalR and are seeking to understand how to use SignalR in a Blazor app. For detailed guidance on the SignalR and Blazor frameworks, see the following reference documentation sets and the API documentation:

- Overview of ASP.NET Core SignalR
- ASP.NET Core Blazor
- .NET API browser

Learn how to:

✓ Create a Blazor app
✓ Add the SignalR client library
✓ Add a SignalR hub
✓ Add SignalR services and an endpoint for the SignalR hub
✓ Add a Razor component code for chat

At the end of this tutorial, you'll have a working chat app.

Prerequisites

Visual Studio

Visual Studio 2022 or later with the **ASP.NET and web development** workload

Sample app

Downloading the tutorial's sample chat app isn't required for this tutorial. The sample app is the final, working app produced by following the steps of this tutorial.

View or download sample code

Create a Blazor Web App

Follow the guidance for your choice of tooling:

Visual Studio

Note

Visual Studio 2022 or later and .NET Core SDK 8.0.0 or later are required.

Create a new project.

Select the **Blazor Web App** template. Select **Next**.

Type `BlazorSignalRApp` in the **Project name** field. Confirm the **Location** entry is correct or provide a location for the project. Select **Next**.

Confirm the **Framework** is .NET 8.0 or later. Select **Create**.

Add the SignalR client library

Visual Studio

In **Solution Explorer**, right-click the `BlazorSignalRApp` project and select **Manage NuGet Packages**.

In the **Manage NuGet Packages** dialog, confirm that the **Package source** is set to `nuget.org`.

With **Browse** selected, type `Microsoft.AspNetCore.SignalR.Client` in the search box.

In the search results, select the latest release of the Microsoft.AspNetCore.SignalR.Client package. Select **Install**.

If the **Preview Changes** dialog appears, select **OK**.

If the **License Acceptance** dialog appears, select **I Accept** if you agree with the license terms.

Add a SignalR hub

Create a `Hubs` (plural) folder and add the following `ChatHub` class (`Hubs/ChatHub.cs`) to the root of the app:

C#

```
using Microsoft.AspNetCore.SignalR;

namespace BlazorSignalRApp.Hubs;

public class ChatHub : Hub
{
    public async Task SendMessage(string user, string message)
    {
        await Clients.All.SendAsync("ReceiveMessage", user, message);
    }
}
```

Add services and an endpoint for the SignalR hub

Open the `Program` file.

Add the namespaces for Microsoft.AspNetCore.ResponseCompression and the `ChatHub` class to the top of the file:

C#

```
using Microsoft.AspNetCore.ResponseCompression;
using BlazorSignalRApp.Hubs;
```

Add Response Compression Middleware services:

C#

```
builder.Services.AddResponseCompression(opts =>
{
   opts.MimeTypes = ResponseCompressionDefaults.MimeTypes.Concat(
         new[] { "application/octet-stream" });
});
```

Use Response Compression Middleware at the top of the processing pipeline's configuration:

C#

```
app.UseResponseCompression();
```

Add an endpoint for the hub immediately after the line that maps Razor components (`app.MapRazorComponents<T>()`):

C#

```
app.MapHub<ChatHub>("/chathub");
```

Add Razor component code for chat

Open the `Components/Pages/Home.razor` file.

Replace the markup with the following code:

razor

```
@page "/"
@rendermode InteractiveServer
@using Microsoft.AspNetCore.SignalR.Client
@inject NavigationManager Navigation
@implements IAsyncDisposable

<PageTitle>Home</PageTitle>

<div class="form-group">
    <label>
        User:
        <input @bind="userInput" />
    </label>
</div>
<div class="form-group">
    <label>
        Message:
        <input @bind="messageInput" size="50" />
    </label>
</div>
<button @onclick="Send" disabled="@(!IsConnected)">Send</button>
```

```
<hr>

<ul id="messagesList">
    @foreach (var message in messages)
    {
        <li>@message</li>
    }
</ul>

@code {
    private HubConnection? hubConnection;
    private List<string> messages = new List<string>();
    private string? userInput;
    private string? messageInput;

    protected override async Task OnInitializedAsync()
    {
        hubConnection = new HubConnectionBuilder()
            .WithUrl(Navigation.ToAbsoluteUri("/chathub"))
            .Build();

        hubConnection.On<string, string>("ReceiveMessage", (user, message) =>
        {
            var encodedMsg = $"{user}: {message}";
            messages.Add(encodedMsg);
            InvokeAsync(StateHasChanged);
        });

        await hubConnection.StartAsync();
    }

    private async Task Send()
    {
        if (hubConnection is not null)
        {
            await hubConnection.SendAsync("SendMessage", userInput, messageIn-
put);
        }
    }

    public bool IsConnected =>
        hubConnection?.State == HubConnectionState.Connected;

    public async ValueTask DisposeAsync()
    {
        if (hubConnection is not null)
        {
            await hubConnection.DisposeAsync();
        }
    }
}
```

ⓘ Note

Disable Response Compression Middleware in the `Development` environment when using **Hot Reload**. For more information, see **ASP.NET Core Blazor SignalR guidance**.

Run the app

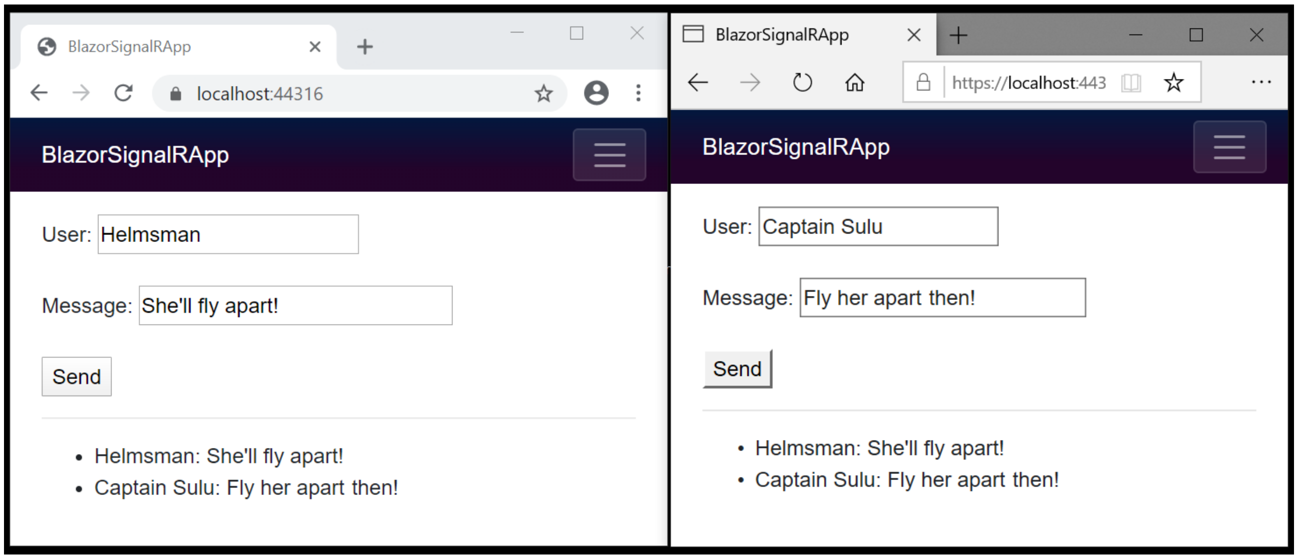
Follow the guidance for your tooling:

Visual Studio

Press `F5` to run the app with debugging or `Ctrl`+`F5` (Windows)/`⌘`+`F5` (macOS) to run the app without debugging.

Copy the URL from the address bar, open another browser instance or tab, and paste the URL in the address bar.

Choose either browser, enter a name and message, and select the button to send the message. The name and message are displayed on both pages instantly:



- Helmsman: She'll fly apart!
- Captain Sulu: Fly her apart then!

- Helmsman: She'll fly apart!
- Captain Sulu: Fly her apart then!

Quotes: *Star Trek VI: The Undiscovered Country* ©1991 Paramount

Next steps

In this tutorial, you learned how to:

- ✓ Create a Blazor app
- ✓ Add the SignalR client library
- ✓ Add a SignalR hub
- ✓ Add SignalR services and an endpoint for the SignalR hub
- ✓ Add a Razor component code for chat

For detailed guidance on the SignalR and Blazor frameworks, see the following reference documentation sets:

Overview of ASP.NET Core SignalR | ASP.NET Core Blazor

Additional resources

- Bearer token authentication with Identity Server, WebSockets, and Server-Sent Events
- Secure a SignalR hub in hosted Blazor WebAssembly apps
- SignalR cross-origin negotiation for authentication
- SignalR configuration
- Debug ASP.NET Core Blazor apps
- Threat mitigation guidance for ASP.NET Core Blazor static server-side rendering
- Threat mitigation guidance for ASP.NET Core Blazor interactive server-side rendering
- Blazor samples GitHub repository (dotnet/blazor-samples)