# How to use websocket with ASP.NET Core

## Introduction

This sample demonstrates how to use websocket with ASP.NET Core.

We will make a simple chat web application in ASP.NET Core.

## Sample prerequisites

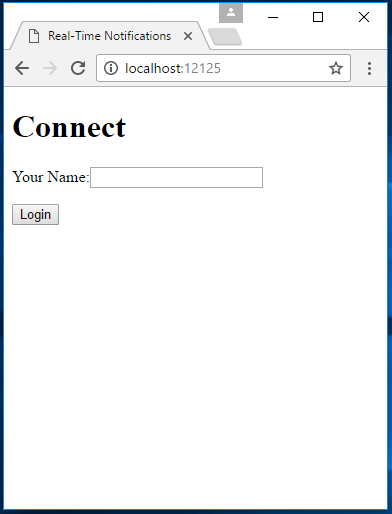
* Visual Studio 2017 or above.[[Visual Studio Home Page](https://www.visualstudio.com/)]
* Visual Studio enabled ASP.NET Core develop component.
* A modern browser that support web socket.

## Building the sample

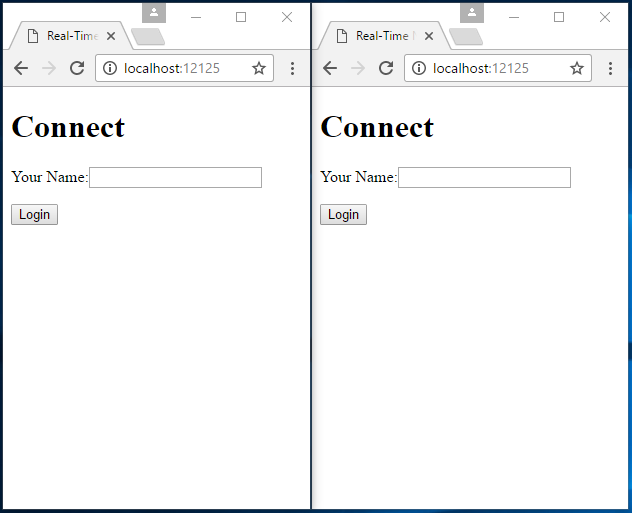
Use Visual studio open the sample solution, then press **F6 Key** to building the sample project.

## Running the sample

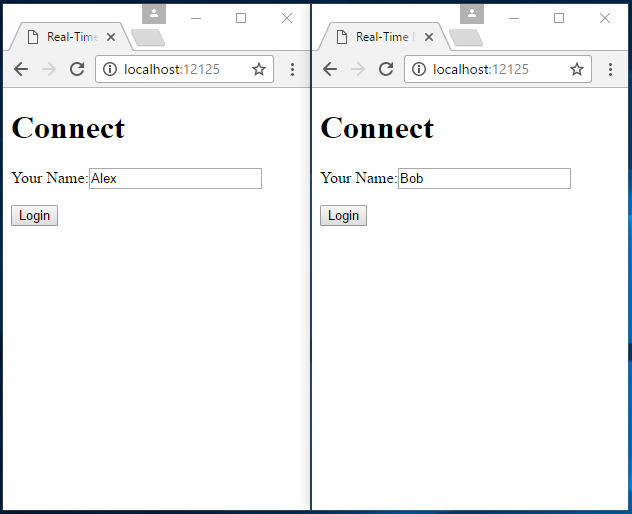
* Use Visual Studio open the sample solution, then press **F5 Key** or select **Debug -> Start Debugging** from the menu.
* When the project is running, you can see below page in browser.

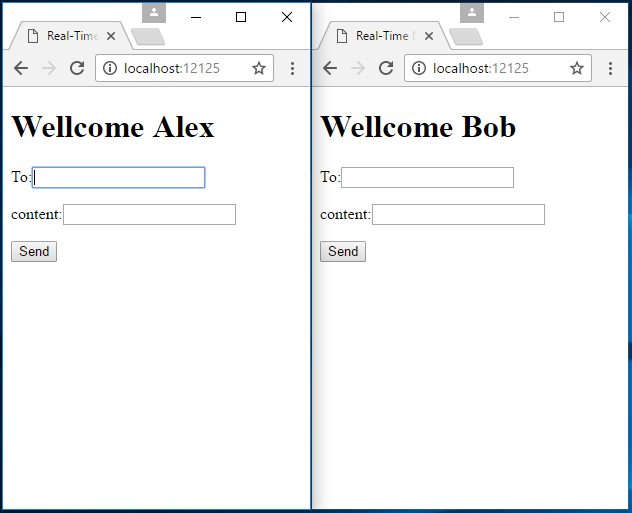


* Duplicate this tab.

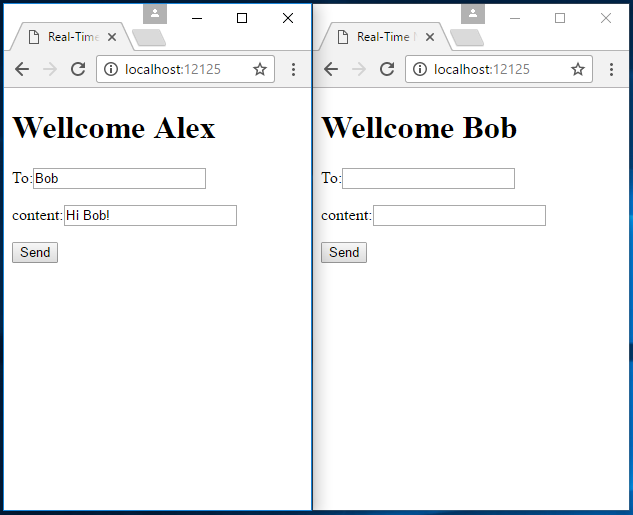


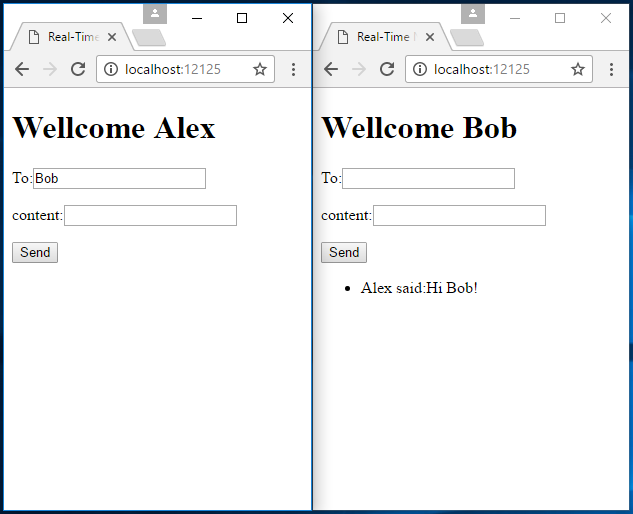
* Type user name and click login.





* Fill fields as below screen and click **Send** button.





## Using the code

This section only show how to use library, code is too many, please download package and view code.

**ChartHandler.cs**

public class ChartHandler : WebSocketHandler

{

protected override int BufferSize { get => 1024 \* 4; }

public override async Task<WebSocketConnection> OnConnected(HttpContext context)

{

var name = context.Request.Query["Name"];

if (!string.IsNullOrEmpty(name))

{

var connection = Connections.FirstOrDefault(m => ((ChartConnection)m).NickName == name);

if (connection == null)

{

var webSocket = await context.WebSockets.AcceptWebSocketAsync();

connection = new ChartConnection(this)

{

NickName = name,

WebSocket = webSocket

};

Connections.Add(connection);

}

return connection;

}

return null;

}

}

**ChartConnection.cs**

public class ChartConnection : WebSocketConnection

{

public ChartConnection(WebSocketHandler handler) : base(handler)

{

}

public string NickName { get; set; }

public override async Task ReceiveAsync(string message)

{

var receiveMessage = JsonConvert.DeserializeObject<ReceiveMessage>(message);

var receiver = Handler.Connections.FirstOrDefault(m => ((ChartConnection)m).NickName == receiveMessage.Receiver);

if (receiver != null)

{

var sendMessage = JsonConvert.SerializeObject(new SendMessage

{

Sender = NickName,

Message = receiveMessage.Message

});

await receiver.SendMessageAsync(sendMessage);

}

else

{

var sendMessage = JsonConvert.SerializeObject(new SendMessage

{

Sender = NickName,

Message = "Can not seed to " + receiveMessage.Receiver

});

await SendMessageAsync(sendMessage);

}

}

private class ReceiveMessage

{

public string Receiver { get; set; }

public string Message { get; set; }

}

private class SendMessage

{

public string Sender { get; set; }

public string Message { get; set; }

}

}

**Startup.cs**

public class Startup

{

public void ConfigureServices(IServiceCollection services)

{

services.AddMvc();

services.AddWebSocketManager();

}

public void Configure(IApplicationBuilder app, IHostingEnvironment env, IServiceProvider serviceProvider)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseWebSockets();

app.MapWebSocketManager("/LiveChat", serviceProvider.GetService<ChartHandler>());

app.UseMvcWithDefaultRoute();

}

}

## More information

Aspnet/WebSockets on GitHub.

<https://github.com/aspnet/WebSockets>

Introduction SignalR

<https://www.asp.net/signalr>