



Chat WebSocket ASP.NET Core

Artur Quaresma
12ºM
2019/2020

WEBSOCKET

Conceito
e sua aplicação

WebSocket

- Tecnologia que através de canais full-duplex permite haver comunicação bidirecional sobre um único socket TCP.
- Usado e criado principalmente em aplicações web
- Usos
 - *Webchats*
 - *Jogos browser*
 - *Sistema de senhas em fila*

LINGUAGENS DE PROGRAMAÇÃO

Linguagens
Ferramentas

ASP.NET Core

- Utilizei a framework ASP.NET Core para desenvolvimento do meu projeto.
- A linguagem de código é C#, e esta framework permite desenvolver aplicações com C# com Web (HTML + CSS + JS)



SignalR

- Utilizei esta biblioteca para a comunicacao em tempo-real entre utilizadores.



Font awesome



Usado para manipular alguns aspetos da pagina

<http://fontawesome.io> - @fontawesome

CÓDIGO

Explicação código
e
Funcionalidades aplicação

WebSocket-CHAT

- Connected Services
- Dependencies
- Properties
 - launchSettings.json
- wwwroot
 - css
 - font-awesome.min.css
 - site.css
 - style.css
 - js
 - chat.js
 - jquery-3.3.1.js
 - site.js
 - lib
 - favicon.ico
 - Hub
 - MessageHub.cs
 - Pages
 - Shared
 - _CookieConsentPartial.cshtml
 - _Layout.cshtml
 - _ValidationScriptsPartial.cshtml
 - _ViewImports.cshtml
 - _ViewStart.cshtml
 - Index.cshtml
 - appsettings.json
 - libman.json
 - Program.cs
 - Startup.cs

Aspeto do chat

WebChat Signlar

Utilizadores Online

Nome: ____

Username...

Username

Join Private Group

Everyone ▼

Mensagem

Startup.cs

Todo o projeto de asp.net tem este ficheiro que é criado automaticamente, mas tive que adicionar essas linha para referenciar as bibliotecas

Com isto chama a classe MessageHub para criar o Hub de mensagens central

```
namespace WebSocket_CHAT
{
    1 reference
    public class Startup
    {
        0 references
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddMvc();
            services.AddSignalR();
        }

        0 references
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.UseStaticFiles();
            app.UseSignalR(config => {
                config.MapHub<MessageHub>("/messages");
            });
            app.UseMvc();
        }
    }
}
```

MessageHub.cs

Classe que faz a gestão das mensagens entre utilizadores conectados

```
namespace WebSocket_CHAT
{
    1 reference
    public class MessageHub : Hub
    {
        3 references
        public static List<string> Utilizadores { get; set; } = new List<string>();

        0 references
        public async Task NewUser(string user) ...

        0 references
        public Task SendMessageToAll(string message) ...

        0 references
        public Task SendMessageToCaller(string message) ...

        0 references
        public Task JoinGroup(string group) ...

        0 references
        public Task SendMessageToGroup(string group, string message) ...

        1 reference
        public override async Task OnConnectedAsync() ...

        1 reference
        public override async Task OnDisconnectedAsync(Exception ex) ...
    }
}
```

MessageHub.cs

Recebe novo utilizador e adiciona à lista de utilizadores

```
public async Task NewUser(string user)
{
    foreach (var elem in Utilizadores)
    {
        await Clients.Caller.SendAsync("NewUser", elem);
    }

    if (!Utilizadores.Contains(user))
    {
        Utilizadores.Add(user);
        await Clients.All.SendAsync("NewUser", user);
    }
}
```

MessageHub.cs

Envia a mensagem para todos os
utilizadores



```
public Task SendMessageToAll(string message)
{
    return Clients.All.SendAsync("ReceiveMessage", message);
}
```

Envia apenas a mensagem para o
utilizador a escolha



0 references

```
public Task SendMessageToCaller(string message)
{
    return Clients.Caller.SendAsync("ReceiveMessage", message);
}
```

MessageHub.cs

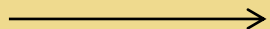
Serve para entrar num grupo (chat privado)



```
public Task JoinGroup(string group)
{
    return Groups.AddToGroupAsync(Context.ConnectionId, group);
}
```

0 references

Serve para enviar mensagem ao grupo, logo quem estiver ligado a este grupo irá receber as mensagens deste grupo



```
public Task SendMessageToGroup(string group, string message)
{
    return Clients.Group(group).SendAsync("ReceiveMessage", message);
}
```

MessageHub.cs

Ao entrar um utilizador avisa todos os outros
que entrou uma nova conexão



Ao sair um utilizador avisa todos os outros que
ocorreu o fecho de uma conexão



```
public override async Task OnConnectedAsync()
{
    await Clients.All.SendAsync("UserConnected", Context.ConnectionId);
    await base.OnConnectedAsync();
}

1 reference
public override async Task OnDisconnectedAsync(Exception ex)
{
    await Clients.All.SendAsync("UserDisconnected", Context.ConnectionId);
    await base.OnDisconnectedAsync(ex);
}
```


Referência deste ficheiro

<https://github.com/YarkoKhamar/Chat/blob/master/ChatAspNetCoreSignalR/Hubs/ChatHub.cs>

Utilizado na manipulação de utilizadores e mensagens,

<https://docs.microsoft.com/en-us/aspnet/signalr/overview/guide-to-the-api/working-with-groups>

Utilizado na manipulação de chat de grupos e sua criação

<https://youtu.be/oOd1NZTNv10>

chat.js

Para poder utilizar as funções da *MessageHub.cs* é preciso haver uma ligação entre a view do *Index.cshtml* com *MessageHub.cs*, utilizador JavaScript.

Com este script é possível controlar as conexões e enviar as devidas mensagens para os utilizadores.

```
"use strict";

var connection = new signalR.HubConnectionBuilder().withUrl("/messages").build();
var username;

function tempoMsg() { ...
}

function isEmptyOrSpaces(str) { ...
}

//registo do utilizador
$("#sendUserName").click(function () { ...
});

//receber mensagens
connection.on("ReceiveMessage", function (message) { ...
});

//utilizador conecta
connection.on("UserConnected", function (connectionId) { ...
});

//utilizador desconecta
connection.on("UserDisconnected", function (connectionId) { ...
});

//codigo alteracao DOM
$("#sendButton").click(function () { ...
});

$("#joinGroup").click(function (event) { ...
});
```

chat.js

São criadas estas variáveis globais para poder utilizar entre as varias funções.

var connection é a variável que constrói um *Hub* para conexão, utilizando a biblioteca SignlaR e junta a classe MessageHub.cs

var username é a variável para guardar o nome de utilizador.

```
var connection = new signalR.HubConnectionBuilder().withUrl("/messages").build();  
var username;
```

chat.js

Funções de manipulação de texto

tempoMsg serve para saber a hora



isEmptyOrSpaces serve para identificação de espaços ou vazio.



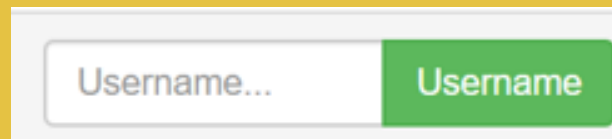
Referencia para tal funcao



<https://stackoverflow.com/questions/10232366/how-to-check-if-a-variable-is-null-or-empty-string-or-all-whitespace-in-javascript>

```
function tempoMsg() {  
  var data = new Date();  
  return " " + data.getHours() + ":" + data.getMinutes() + ":" + data.getSeconds() + " ";  
}  
  
function isEmptyOrSpaces(str) {  
  return str === null || str.match(/^ *$/) !== null;  
}
```

```
<div class="input-group-btn">
  <button class="btn btn-success" id="sendUserName">Username</button>
</div>
```



```
//registo do utilizador
$("#sendUserName").click(function () {
    username = $('#usernameInput:text').val();

    if (isEmptyOrSpaces(username))
        username = "Anonymous";

    $('#usernameSpan').html(username);

    //dá-se o início da conexão
    connection.start();


    //inserção do novo user
    connection.invoke("NewUser", username);

    //avisa a todos os user que um novo user entrou
    $connection.invoke("SendMessageToAll", username);
});
```

Utilizador escreve o seu username se preferir, se não entra como anónimo.

Depois entra na lista de utilizadores

Agora a melhor parte, a conexão.

- 
1. A conexão é iniciada.
 2. Depois é utilizador é adiciona a lista
 3. Finalmente avisa todos os utilizadores conectados que utilizador juntou-se

Utilizadores Online

- artur

Nome: artur

artur ON

Everyone ▼

Todos os utilizadores recebem a mensagem, depende do tipo de envio, e a mensagem é adiciona no chat

```
//receber mensagens
connection.on("ReceiveMessage", function (message) {
    var msg = message.replace(/&/g, "&amp;").replace(/</g, "&lt;").replace(/>/g, "&gt;");
    var div = document.createElement("div");
    div.innerHTML = msg + "<hr/>";
    document.getElementById("messages").appendChild(div);
    div.scrollIntoView();
});
```

```
public Task SendMessageToAll(string message)
{
    return Clients.All.SendAsync("ReceiveMessage", message);
}
```

Neste caso é enviado uma mensagem para todos,

Utilizador conecta e faz avisa todos os utilizadores online e adiciona da lista de utilizadores



```
//utilizador conecta
connection.on("UserConnected", function (connectionId) {

    connection.invoke("SendMessageToAll", ("" + username + " ON"));

    //adiciona User nos User ON
    var userTag = document.createElement("li");
    userTag.textContent = username;
    document.getElementById("userList").appendChild(userTag);
});
```

Utilizador desconecta e faz avisa todos os utilizadores online e remove lista de utilizadores



```
//utilizador desconecta
connection.on("UserDisconnected", function (connectionId) {

    connection.invoke("SendMessageToAll", ("" + username + " OFF"));

    //remove User nos User ON
    var userTag = document.createElement("li");
    userTag.textContent = username;
    document.getElementById("userList").appendChild(userTag + "off");
});
```

ligação entre *MessageHub.cs* com o *chat.js*, e depois a manipulação no *Index.cshtml*

Join Private Group

arturEsquerdo ON

arturEsquerdo ON

Anonymous ON

arturEsquerdo (12:20:31)-- > Olá malta

Anonymous (12:20:35)-- > td bem?

Everyone ▾

Olá malta

Join Private Group

arturEsquerdo ON

Anonymous ON

arturEsquerdo (12:20:31)-- > Olá malta

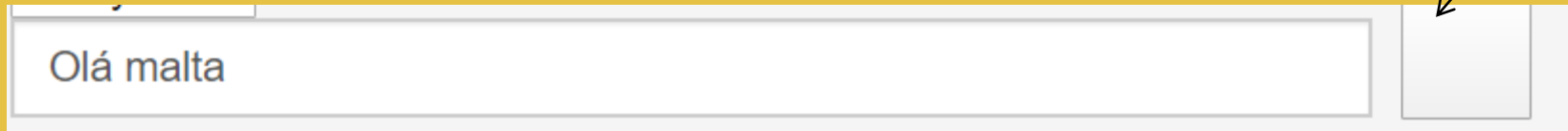
Anonymous (12:20:35)-- > td bem?

Everyone ▾

td bem?

arturEsquerdo OFF

Ao clicar no botão para enviar é executada esta função



A screenshot of a web form. It features a text input field containing the text "Olá malta" and a light gray button to its right. An arrow from the text above points to the button.

```
//codigo alteracao DOM
$("#sendButton").click(function () {
    var message = $('#message').val();

    if(isEmptyOrSpaces(message))
        message = "----";

    var groupElement = document.getElementById("group");
    var groupValue = groupElement.options[groupElement.selectedIndex].value;

    if (groupValue === "All")
        connection.invoke("SendMessageToAll", username + " (" + tempoMsg() + ")-- > " + message);
    else
        connection.invoke("SendMessageToGroup", groupValue, " (" + tempoMsg() + ")-- > " + message);

    e.preventDefault();
});
```

```
//codigo alteracao DOM
$("#sendButton").click(function () {
    var message = $('#message').val();

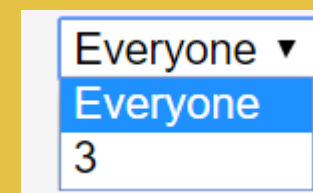
    if(isEmptyOrSpaces(message))
        message = "----";

    var groupElement = document.getElementById("group");
    var groupValue = groupElement.options[groupElement.selectedIndex].value;

    if (groupValue === "All")
        connection.invoke("SendMessageToAll", username + " (" + tempoMsg() + ")-- > " + message);
    else
        connection.invoke("SendMessageToGroup", groupValue, " (" + tempoMsg() + ")-- > " + message);

    e.preventDefault();
});
```

Identifica em que grupo o utilizador está e enviar para esses grupos



```
//codigo alteracao DOM
$("#sendButton").click(function () {
    var message = $('#message').val();

    if(isEmptyOrSpaces(message))
        message = "----";

    var groupElement = document.getElementById("group");
    var groupValue = groupElement.options[groupElement.selectedIndex].value;

    if (groupValue === "All")
        connection.invoke("SendMessageToAll", username + " (" + tempoMsg() + ")-- > " + message);
    else
        connection.invoke("SendMessageToGroup", groupValue, " (" + tempoMsg() + ")-- > " + message);

    e.preventDefault();
});
```

Identifica em que grupo o utilizador está e enviar para esses grupos

```
public Task SendMessageToGroup(string group, string message)
{
    return Clients.Group(group).SendAsync("ReceiveMessage", message);
}
```

Everyone ▼
Everyone
3

```

$("#joinGroup").click(function () {

    var nomeGrupo = $('#nomeGrupo:text').val();

    var option = document.createElement("option");
    option.text = nomeGrupo;
    option.value = nomeGrupo;
    document.getElementById("group").add(option); //grupo de elementos

    connection.invoke("JoinGroup", nomeGrupo);

    e.preventDefault();
});

```

Identifica um grupo e cria ele e adiciona a listbox

Everyone ▼

Everyone

3

```

public Task JoinGroup(string group)
{
    ...
    return Groups.AddToGroupAsync(Context.ConnectionId, group);
}

```

Join Private Group

arturEsquerdo (12:20:31)-- > Olá malta

Anonymous (12:20:35)-- > td bem?

arturEsquerdo OFF

arturEsquerdo ON

teste123 ON

Everyone ▼

Olá malta

grupo secreto

Join Private Group

arturEsquerdo ON

teste123 ON

(12:31:39)-- > ola secretos

grupo secreto ▼

ola secretos

Referencia chat.js

<https://github.com/YarkoKhamar/Chat/blob/master/ChatAspNetCoreSignalR/wwwroot/js/chat.js>

Simplificado e melhorado.

REFERÊNCIAS



Referências

<https://pt.wikipedia.org/wiki/WebSocket>

<https://docs.microsoft.com/en-us/aspnet/signalr/overview/guide-to-the-api/working-with-groups>

<https://docs.microsoft.com/en-us/aspnet/signalr/>

<https://github.com/YarkoKhamar/Chat>

<https://tinyurl.com/CodeOpinioChannel>

http://www.icons101.com/icon/id_67331/setid_2086/Flat_Round_by_Roundicons/Chat

<https://stackoverflow.com/questions/10232366/how-to-check-if-a-variable-is-null-or-empty-string-or-all-whitespace-in-javascr>



Sala de CHAT

