Table of Contents

[Basics 3](#_Toc118039287)

[Current Resources with Changes 3](#_Toc118039288)

[New Resources Added 3](#_Toc118039289)

[npm install / ng serve 3](#_Toc118039290)

[Install Microsoft SignalR Package 3](#_Toc118039291)

[environment 3](#_Toc118039292)

[environment.ts 3](#_Toc118039293)

[environment.prod.ts 3](#_Toc118039294)

[/core/services 4](#_Toc118039295)

[api-url.service.ts 4](#_Toc118039296)

[/signalr/presence-hub.service.ts 4](#_Toc118039297)

[/signal/message-hub.service.ts 5](#_Toc118039298)

[account.service.ts 6](#_Toc118039299)

[Components 7](#_Toc118039300)

[/site/members/member-card 7](#_Toc118039301)

[member-card.component.ts 7](#_Toc118039302)

[member-card.component.html 8](#_Toc118039303)

[member-card.component.css 8](#_Toc118039304)

[/site/members/member-detail 8](#_Toc118039305)

[member-detail.component.ts 8](#_Toc118039306)

[member-detail.component.html 9](#_Toc118039307)

[/site/members/member-messages 9](#_Toc118039308)

[member-message.component.ts 9](#_Toc118039309)

[/site/members/member-messages 10](#_Toc118039310)

[member.messages.component.ts 10](#_Toc118039311)

[member-messages.component.html 11](#_Toc118039312)

[Run Site 11](#_Toc118039313)

# 

# Basics

|  |  |
| --- | --- |
| WorkingFolder | Copy the content of “Site-05-Identity Role Management” in “Site-06-SignalR” and issue   * npm install * ng serve |

# Current Resources with Changes

1. environment
2. /core/services/api-url.service.ts
3. /core/services/account.service.ts
4. /site/members/member-card
5. /site/members/member-detail
6. /site/memerts/member-messages

# New Resources Added

1. Install Microsoft SignalR Package
2. /core/services/signalr/presence-hub.service
3. /core/services/signalr/message-hub.service

# npm install / ng serve

Follow basics and then run “npm install” and “ng serve” commands.

# Install Microsoft SignalR Package

* npm install @microsoft/signalr

Check package.json for details

    "@microsoft/signalr": "^6.0.10",

# environment

add the hubs base url and also update the api base urls

## environment.ts

export const environment = {

  production: false,

  title: "MySocialConnect",

  displayConsoleLog: true,

  usebaseUrlHttps: true,

  webApiBaseUrlHttps: 'https://localhost:5000/api/',

  webApiBaseUrlHttp: 'http://localhost:5001/api/',

  webApiBaseHubsUrlHttps: 'https://localhost:5000/hubs/',

  webApiBaseHubsUrlHttp: 'http://localhost:5001/hubs/',

};

## environment.prod.ts

For now follow the file. Later it will get updated

export const environment = {

  production: true,

  title: "MySocialConnect",

  displayConsoleLog: false,

  usebaseUrlHttps: true,

  webApiBaseUrlHttps: 'api/',

  webApiBaseUrlHttp: 'api/',

  webApiBaseHubsUrlHttps: 'hubs/',

  webApiBaseHubsUrlHttp: 'hubs/',

};

# /core/services

## api-url.service.ts

delete the baseUrl

 //get the api base url from the environment. This url ends with a /

  private baseUrl: string = environment.usebaseUrlHttps ? environment.webApiBaseUrlHttps : environment.webApiBaseUrlHttp;

update the apiBaseUrl

  //api base url 🡺 comes with /api/

  private apiBaseUrl: string = environment.usebaseUrlHttps ? environment.webApiBaseUrlHttps : environment.webApiBaseUrlHttp;

add the hubBaseUrl

  //signalr hub 🡺 comes with /hub/

  private hubBaseUrl: string = environment.usebaseUrlHttps ? environment.webApiBaseHubsUrlHttps : environment.webApiBaseHubsUrlHttp;

Delete the printing lines from the constructor. These are just white noise

And then add the aignalr urls

  //signalr urls

  signalr\_presenceUrl = `${this.hubBaseUrl}presence`;

  signalr\_messageUrl = `${this.hubBaseUrl}message`;

## /signalr/presence-hub.service.ts

* ng g s /core/services/signalr/presenceHub --skip-tests

import { Injectable } from '@angular/core';

import { HubConnection, HubConnectionBuilder } from '@microsoft/signalr';

import { ToastrService } from 'ngx-toastr';

import { BehaviorSubject } from 'rxjs';

import { UserTokenDto } from '../../models/userTokenDto.model';

import { ApiUrlService } from '../api-url.service';

@Injectable({

  providedIn: 'root'

})

export class PresenceHubService{

  private hubConnection!: HubConnection;

  //create a generic subject for getting the online users, it will get the string array of our user names

  //initilize the array as well

  private onlineUsersSource = new BehaviorSubject<string[]>([]);

  onlineUsers$ = this.onlineUsersSource.asObservable();

  constructor(private apiUrlService: ApiUrlService, private toastrService: ToastrService) { }

  //create Hub Connection

  createHubConnection(user: UserTokenDto) {

    //build the connection

    this.hubConnection = new HubConnectionBuilder()

                              .withUrl(this.apiUrlService.signalr\_presenceUrl, {

                                accessTokenFactory: () => user.token

                              })

                              .withAutomaticReconnect()

                              .build();

    //start the connection

    this.hubConnection.start().catch(error => console.log(error));

    //listen for events UserIsOnline and UserIsOffline. both return userName

    this.hubConnection.on('UserIsOnline', userName => {

      this.toastrService.info(userName + ' has connected');

    });

    this.hubConnection.on('UserIsOffline', userName => {

      this.toastrService.warning(userName + ' has disconnected');

    });

    //listen for event GetOnlineUsers, returns string array of user names

    this.hubConnection.on('GetOnlineUsers', (userNames: string[]) => {

      this.onlineUsersSource.next(userNames);

    });

  }

  //stop hub Connection

  stopHubConnection() {

    if(this.hubConnection)

      this.hubConnection.stop().catch(error => console.log(error));

  }

}

## /signal/message-hub.service.ts

* ng g s /core/services/signalr/messageHub --skip-tests

import { Injectable } from '@angular/core';

import { HubConnection, HubConnectionBuilder } from '@microsoft/signalr';

import { BehaviorSubject, take } from 'rxjs';

import { MessageDto } from '../../models/messageDto';

import { UserTokenDto } from '../../models/userTokenDto.model';

import { ApiUrlService } from '../api-url.service';

@Injectable({

  providedIn: 'root'

})

export class MessageHubService {

  private hubConnection!: HubConnection;

  private messageThreadSource = new BehaviorSubject<MessageDto[]>([]);

  messageThread$ = this.messageThreadSource.asObservable();

  constructor(private apiUrlService: ApiUrlService) { }

  //create Hub Connection

  createHubConnection(user: UserTokenDto, otherUserName: string, otherUserId: number) {

    //query string params for the other user

    const otherParams = `?otherUserName=${otherUserName}&otherUserId=${otherUserId}`;

    const url = this.apiUrlService.signalr\_messageUrl + otherParams;

    //build the connection

    this.hubConnection = new HubConnectionBuilder()

                              .withUrl(url, {

                                accessTokenFactory: () => user.token

                              })

                              .withAutomaticReconnect()

                              .build();

    //start the connection

    this.hubConnection.start().catch(error => console.log(error));

    //listen for events UserIsOnline and UserIsOffline. both return userName

    this.hubConnection.on('ReceiveMessageThread', (messages: MessageDto[]) => {

      this.messageThreadSource.next(messages);

    });

    //on new message add

    this.hubConnection.on('NewMessage', (message: MessageDto) => {

      //add the new message to messageThreadSource

      this.messageThread$.pipe(take(1)).subscribe({

        next: (messages: MessageDto[]) => {

          if (message) {

            //without mutating the array add on the new message. this is a new array

            var newMessages = [...messages, message];

            this.messageThreadSource.next(newMessages);

          }

        }

      });

    });

  }

  //stop hub Connection

  stopHubConnection() {

    if(this.hubConnection)

      this.hubConnection.stop().catch(error => console.log(error));

  }

  //for sending the messages. Not using the message-service sendMessage function any more

  async sendMessage(receipentUserId: number, content: string) {

    return this.hubConnection

      .invoke('SendMessage', { receipientUserId: receipentUserId, content: content })

      .catch(error => console.log(error));

  }

}

## account.service.ts

inject presence service

  constructor(private apiUrlService: ApiUrlService,

    private httpClientService: HttpClientService,

    private localStorageService: LocalStorageService,

    private presenceService: PresenceHubService) { }

app.component.ts calls getAndFireCurrentUser so update that to make a connection. This is needed when the user refreshes the page

  //this is being called from inside app.component.ts

  getAndFireCurrentUser() {

    const user: UserTokenDto = this.localStorageService.getItem(this.localStorageService.\_keyUser);

    if (!user) return;

    this.fireCurrentUser(user);

    //signalr presence - create hub connection to be notified

    this.presenceService.createHubConnection(user);

  }

On login method set the presence

  //login the user. receives LoginDto and returns UserTokenDto

  login(loginDto: LoginDto) {

    var url = this.apiUrlService.accountLogin;

    if(environment.displayConsoleLog) console.log(`AccountService LoginUrl: ${url}`);

    //persist the user in the local storage

    return this.httpClientService

      .post<UserTokenDto>(url, loginDto)

      .pipe(

        map((respone: UserTokenDto) => {

          const user = respone;

          if(environment.displayConsoleLog) console.log(user);

          if (user) {

            this.setAndFireCurrentUser(user);

            //signalr presence - create hub connection to be notified

            this.presenceService.createHubConnection(user);

          }

          return user;

        })

      );

  }

Set presence for the register

  register(registerDto: SiteRegisterDto) {

    var url = this.apiUrlService.accountRegisterUser;

    if(environment.displayConsoleLog)

      console.log(`AccountService RegisterUrl: ${url}`);

    return this.httpClientService

      .post<UserTokenDto>(url, registerDto)

      .pipe(

        map((respone: UserTokenDto) => {

          const user = respone;

          if (user) {

            this.setAndFireCurrentUser(user);

            //signalr presence - create hub connection to be notified

            this.presenceService.createHubConnection(user);

          }

          return user;

        })

      );

  }

And stop the connection on logout

  logout() {

    //remove the user from local storage

    this.localStorageService.removeItem(this.localStorageService.\_keyUser);

    this.fireCurrentUser(null!);

    //signalr presence - stop hub connection

    this.presenceService.stopHubConnection();

  }

# Components

## /site/members/member-card

### member-card.component.ts

Add the properties

  //signalr - presence service - online users

  isUserOnline: boolean = false;

  onlineUsers: string[] = [];

  onlineUsersSubscription!: Subscription;

Add PresenceService to the constructor

  constructor(private memberService: MembersService,

              private toastrService: ToastrService,

              private presenceService: PresenceHubService) { }

Create a method get online users and mark as the current user as online if in the list

  //signalR - presence service

  getOnlineUsers() {

    this.isUserOnline = false;

    this.onlineUsersSubscription = this.presenceService.onlineUsers$.subscribe({

      next: (userNames: string[]) => {

        this.onlineUsers = userNames;

        if (userNames && userNames.length > 0) {

          this.isUserOnline = userNames.includes(this.member.userName);

        }

      }

    });

  }

Call the method in ngOnInit

  ngOnInit(): void {

    this.getOnlineUsers();

  }

Also, unsubscribe in ngOnDestroy

  ngOnDestroy(): void {

    if (this.addSubscription) this.addSubscription.unsubscribe();

    if (this.onlineUsersSubscription) this.onlineUsersSubscription.unsubscribe();

  }

### member-card.component.html

Update the user icons and add class when the user is online

<!--show users online/offline-->

            <span [class.is-online]="isUserOnline">

              <i class="fa fa-user mr-2"></i>

            </span>

### member-card.component.css

Finally add the css needed

@keyframes fa-blink {

  0% {opacity: 1;}

  100% {opacity: 0.4;}

}

.is-online{

  animation: fa-blink 1.5s linear infinite;

  color: rgb(1, 189, 42);

}

## /site/members/member-detail

### member-detail.component.ts

Add the properties

  //signalr - presence service - online users

  isUserOnline: boolean = false;

  onlineUsers: string[] = [];

  onlineUsersSubscription!: Subscription;

  //signalr - messageHub service - messages

  messagesHubSubscription!: Subscription

  //logged in user

  user: UserTokenDto = <UserTokenDto>{};

Inject presenceHubService, messageHubService and AccountService via the constructor

Also get the logged in user via the constructor

  constructor(private memberService: MembersService,

    private route: ActivatedRoute,

    private router: Router,

    private toastrService: ToastrService,

    private messageService: MessageService,

    private presenceHubService: PresenceHubService,

    private messageHubService: MessageHubService,

    private accountService: AccountService) {

    this.accountService.currentUser$.pipe(take(1)).subscribe({

      next: (user: UserTokenDto) => {

        this.user = user;

      }

    });

  }

Add a method to get the onlineUsers

    getOnlineUsers() {

      this.isUserOnline = false;

      this.onlineUsersSubscription = this.presenceHubService.onlineUsers$.subscribe({

        next: (userNames: string[]) => {

          this.onlineUsers = userNames;

          if (userNames && userNames.length > 0) {

            this.isUserOnline = userNames.includes(this.member.userName);

          }

        }

      });

    }

Call the getOnlineUSers in ngOnInit

    this.getOnlineUsers();

Update onTabActive method to load the messages from messageHub and to also stop it when the user moves away from the tab

  onTabActivate(data: TabDirective) {

    this.activeTab = data;

    //check that the active tab is messages and only load messages when the messages have not already been loaded

    if (this.activeTab.heading === 'Messages' && this.messages.length === 0) {

      //rather than loading the messages using message service now gettig the messages via MessageHub

      //this.loadMessages();

      //create the hub connection

      this.messageHubService.createHubConnection(this.user, this.member.userName, this.member.id);

    }

    else {

      //when the user is not on the messages tab then disconnect from the hub as well

      this.messageHubService.stopHubConnection();

    }

  }

Finally unsubscribe on ngOnDestroy and also stop the messageHub connection

  ngOnDestroy(): void {

    if (this.paramSubscription) this.paramSubscription.unsubscribe();

    if (this.membersSubscription) this.membersSubscription.unsubscribe();

    if (this.messageSubscription) this.messageSubscription.unsubscribe();

    if (this.queryParamSubscription) this.queryParamSubscription.unsubscribe();

    if (this.onlineUsersSubscription) this.onlineUsersSubscription.unsubscribe();

    if (this.messagesHubSubscription) this.messagesHubSubscription.unsubscribe();

    //when the user moves away from the component then stop the message connection

    this.messageHubService.stopHubConnection();

  }

### member-detail.component.html

Add the icon and a text online now

<div class="row" >

    <!--left side quick info start-->

    <div class="col-4">

        <div class="card">

            <img src="{{ member.photoUrl || './assets/user.png' }}" alt="{{ member.displayName}}" class="card-img-top img-thumbnail">

            <div class="card-body">

                <div class="my-2" \*ngIf="isUserOnline">

                  <i class="fa fa-user-circle text-success"></i> Online now

                </div>

## /site/members/member-messages

### member-message.component.ts

Messages are not being passed to the component any more after MessageHub implementation

Add properties

  messagesHubSubscription!: Subscription;

Inject messageHubService

  constructor(private messageService: MessageService, private messageHubService: MessageHubService) { }

on ngOnInit call a method to load the messages using hub

  ngOnInit(): void {

    this.loadMessagesFromMessageHub();

  }

Write the load messages method

  loadMessagesFromMessageHub() {

    this.messagesHubSubscription = this.messageHubService.messageThread$.subscribe({

      next: (messages: MessageDto[]) => {

        this.messages = messages;

      }

    });

  }

## /site/members/member-messages

### member.messages.component.ts

load messages and send message using the MessageHub Service

import { Component, Input, OnDestroy, OnInit, ViewChild } from '@angular/core';

import { NgForm } from '@angular/forms';

import { MessageService } from '../../../core/services/message.service';

import { MessageHubService } from '../../../core/services/signalr/message-hub.service';

import { MessageDto } from '../../../core/models/messageDto';

import { Subscription } from 'rxjs';

@Component({

  selector: 'app-member-messages',

  templateUrl: './member-messages.component.html',

  styleUrls: ['./member-messages.component.css']

})

export class MemberMessagesComponent implements OnInit, OnDestroy {

  @ViewChild('messageForm') messageForm!: NgForm;

  @Input() userId!: number;

  @Input() messages: MessageDto[] = [];

  messageContent!: string;

  messageSubscription!: Subscription;

  messagesHubSubscription!: Subscription;

  messageHubSendSubscription!: Subscription;

  constructor(private messageService: MessageService, private messageHubService: MessageHubService) { }

  ngOnInit(): void {

    this.loadMessagesFromMessageHub();

  }

  ngOnDestroy(): void {

    if (this.messageSubscription) this.messageSubscription.unsubscribe();

    if (this.messagesHubSubscription) this.messagesHubSubscription.unsubscribe();

    if (this.messageHubSendSubscription) this.messageHubSendSubscription.unsubscribe();

  }

  loadMessagesFromMessageHub() {

    this.messagesHubSubscription = this.messageHubService.messageThread$.subscribe({

      next: (messages: MessageDto[]) => {

        this.messages = messages;

      }

    });

  }

  //this not being used any more. below method using hub is being used now

  onSendMessage() {

    this.messageSubscription = this.messageService.sendMessage(this.userId, this.messageContent).subscribe({

      next: (message: MessageDto) => {

        this.messages.push(message);

        this.messageForm.reset();

      },

      error: e => { },

      complete: () => { }

    });

  }

  onSendMessageUsingHub() {

    //this is returning a promise. Then new message added will be show via loadMessagesFromMessageHub

    this.messageHubService.sendMessage(this.userId, this.messageContent).then(() => {

      this.messageForm.reset();

    });

  }

}

### member-messages.component.html

User the onSendMessageUsingHub method to send the messages.

        <form #messageForm="ngForm" (ngSubmit)="onSendMessageUsingHub()" autocomplete="off">

# Run Site

View the site and then go to console, web socket info will be displaying



Then open another browser and login with a different user . You’ll see the notification

Graphical user interface, text, application

Description automatically generated

Member card will show a green user icon which will be animating

Graphical user interface, text, application, chat or text message

Description automatically generated

Member detail online now and icon displaying

Graphical user interface, text, application

Description automatically generated