Table of Contents

[Basics 3](#_Toc117116212)

[Current Resources with Changes 3](#_Toc117116213)

[New Resources Added 3](#_Toc117116214)

[npm install / ng serve 3](#_Toc117116215)

[/core/enums 3](#_Toc117116216)

[zMessageType.cs 3](#_Toc117116217)

[/core/models/helpers 3](#_Toc117116218)

[page-params.model.ts 3](#_Toc117116219)

[message-params.model.ts 4](#_Toc117116220)

[like-params.model.ts 4](#_Toc117116221)

[user-params.model 4](#_Toc117116222)

[/core/models 5](#_Toc117116223)

[messageDto interface 5](#_Toc117116224)

[/core/services 5](#_Toc117116225)

[pagination.service.ts 5](#_Toc117116226)

[members.service.ts 6](#_Toc117116227)

[api-url.service.ts 7](#_Toc117116228)

[message.service.ts 7](#_Toc117116229)

[/core/resolvers 7](#_Toc117116230)

[member-detail.resolver.ts 7](#_Toc117116231)

[app-routing.moduel.ts 8](#_Toc117116232)

[Components 8](#_Toc117116233)

[/site/messages.component 8](#_Toc117116234)

[messages.component.ts 8](#_Toc117116235)

[messages.component.html 10](#_Toc117116236)

[messages.component.css 11](#_Toc117116237)

[/site/members/member-messages 11](#_Toc117116238)

[/core/modules/shared.module.ts 11](#_Toc117116239)

[member-messages.component.ts 11](#_Toc117116240)

[member-messages.component.html 12](#_Toc117116241)

[member-messages.component.html 13](#_Toc117116242)

[/site/members/member-detail 13](#_Toc117116243)

[member-detail.component.ts 13](#_Toc117116244)

[member-detail.component.html 16](#_Toc117116245)

[/site/members/member-card.component 16](#_Toc117116246)

[member-card.component.html 16](#_Toc117116247)

# 

# Basics

|  |  |
| --- | --- |
| WorkingFolder | Copy the content of “Site-03-Like Feature” in “Site-04-Message Feature” and issue   * npm install * ng serve |

# Current Resources with Changes

1. /core/enums/zMemberGetBy
2. /core/models/helpers/like-params.model
3. /core/models/helpers/user-params.model
4. /core/services/members.service
5. /core/services/api-url.service
6. /site/messages.component
7. /core/modules/shared.module.ts
8. /site/members/member-detail
9. /site/members/member-card
10. app-routing.module.ts

# New Resources Added

1. /core/enums/zMessageType
2. /code/models/helpers/page-params.model
3. /core/models/helpers/message-params.model
4. /core/models/messageDto interface
5. /core/service/pagination
6. /core/services/message
7. /site/members/member-messages
8. /core/resolvers/member-detail.resolver

# npm install / ng serve

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

# /core/enums

## zMessageType.cs

export enum zMessageType{

    inbox = 0,

    inboxUnread = 1,

    outbox = 2,

}

# /core/models/helpers

## page-params.model.ts

* ng g class /core/models/helpers/pageParams --type=model --skip-tests

This is the parent class which will be now used with user-params, likes-params and messages-params

import { HttpParams } from "@angular/common/http";

export class PageParams{

    pageNumber: number = 1;

    pageSize: number = 5;

    constructor() { }

    getPaginationSearchParams() : HttpParams {

        let params = new HttpParams();

        params = params.append('pageNumber', this.pageNumber.toString());

        params = params.append('pageSize', this.pageSize.toString());

        return params;

    }

}

## message-params.model.ts

* ng g class /core/models/helpers/messageParams --type=model --skip-tests

This extends PageParams.

import { HttpParams } from "@angular/common/http";

import { zMessageType } from "../../enums/zMessageType";

import { PageParams } from "./page-params.model";

export class MessageParams extends PageParams {

    messageType: zMessageType = zMessageType.inboxUnread;

    constructor() {

        super();

     }

    //helper metod to build search params

    getSearchParams() : HttpParams {

        let params = super.getPaginationSearchParams();

        params = params.append('messageType', zMessageType[this.messageType]);

        return params;

    }

}

## like-params.model.ts

import { HttpParams } from "@angular/common/http";

import { zUserLikeType } from "../../enums/zUserLikeType";

import { PageParams } from "./page-params.model";

export class LikeParams extends PageParams {

    userLikeType: zUserLikeType = zUserLikeType.liked;

    constructor() {

        super();

    }

    //helper metod to build search params

    getSearchParams() : HttpParams {

        let params = super.getPaginationSearchParams();

        params = params.append('userLikeType', zUserLikeType[this.userLikeType]);

        return params;

    }

}

## user-params.model

import { HttpParams } from "@angular/common/http";

import { AppConstants } from "../../constants/app-constants";

import { UserTokenDto } from "../userTokenDto.model";

import { PageParams } from "./page-params.model";

export class UserParams  extends PageParams {

    gender: string;

    minAge: number = 18;

    maxAge: number = 99;

    orderBy: string = AppConstants.membersOrderByLastActive;

    constructor(user: UserTokenDto) {

        super();

        //when logged in user is male then get the female members otherwise male members

        this.gender = user.gender === AppConstants.Female ? AppConstants.Male : AppConstants.Female;

    }

    //helper metod to build search params

    getSearchParams() : HttpParams {

        let params = super.getPaginationSearchParams();

        params = params.append('gender', this.gender);

        params = params.append('minAge', this.minAge.toString());

        params = params.append('maxAge', this.maxAge.toString());

        params = params.append('orderBy', this.orderBy);

        return params;

    }

}

# /core/models

## messageDto interface

This one created as interface and not class since won’t be using the constructor.

export interface MessageDto {

    id: number;

    senderId: number;

    senderGuid: string,

    senderUsername: string;

    senderPhotoUrl: string;

    receipientId: number;

    receipientGuid: string;

    receipientUsername: string;

    receipientPhotoUrl: string;

    messageContent: string;

    dateMessageRead?: Date;

    dateMessageSent: Date;

}

# /core/services

## pagination.service.ts

New pagination service to be used by the members, likes and messages

* ng g service /core/services/pagination --skip-tests

import { HttpParams } from '@angular/common/http';

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

import { map } from 'rxjs';

import { AppConstants } from '../constants/app-constants';

import { HttpClientService } from './http-client.service';

import { PaginatedResult } from '../models/helpers/paginated-result.model';

@Injectable({

  providedIn: 'root'

})

export class PaginationService {

  constructor(private httpClientService: HttpClientService) { }

  getPaginatedResult<T>(url: string, params: HttpParams) {

    const paginatedResult: PaginatedResult<T> = new PaginatedResult<T>();

    return this.httpClientService.getWithFullResponse<T>(url, params).pipe(

      map(response => {

        paginatedResult.result = response.body!;

        const paginationHeader = response.headers.get(AppConstants.PaginationHeader);

        if (paginationHeader !== null) {

          paginatedResult.pagination = JSON.parse(paginationHeader);

        }

        return paginatedResult;

      })

    );

  }

}

## members.service.ts

Add pagination service to the constructor

private pageService: PaginationService

Remove the private method: getPaginatedResult since this has been moved to the paginatedService

Update the getMembers method

  getMembers(user\_params: UserParams) {

    //we'll use the key using the full param object properties we have in place for searching

    const cacheKey = Object.values(user\_params).join('-');

    if (environment.displayConsoleLog) console.log(`MemberKey: ${cacheKey}`);

    let cacheReuslt = this.memberCache.get(cacheKey);

    if (cacheReuslt)

      return of(cacheReuslt);

    //add the pagination and filtering parameters

    const params = user\_params.getSearchParams();

    //url

    const url = this.apiUrlService.usersAll;

    if (environment.displayConsoleLog) console.log(`Users allUrl: ${url} params: ${params}`);

    //users end point is protected by authentication so need to send the token as well

    //check jwt interceptor for details

    return this.pageService.getPaginatedResult<userDto[]>(url, params).pipe(

      map(response => {

        //set the cache

        this.memberCache.set(cacheKey, response);

        return response;

      })

    );

  }

Update getLikes method

  getLikes(like\_parms: LikeParams) {

    //add the pagination and filtering parameters

    const params = like\_parms.getSearchParams();

    var url = this.apiUrlService.likesForUser;

    if (environment.displayConsoleLog) console.log(`getLikes url: ${url} params: ${params}`);

    return this.pageService.getPaginatedResult<Partial<userDto[]>>(url, params);

  }

## api-url.service.ts

Add the message controller urls

  //message controller urls

  private messagesBaseUrl = `${this.apiBaseUrl}messages/`;

  messageRecipIdReplace = "[recpid]";

  messageDelIdReplace = "[msgid]";

  messageSend = `${this.messagesBaseUrl}send/message`;

  messagesGet = `${this.messagesBaseUrl}user/get/messages`;

  messageThread = `${this.messagesBaseUrl}message/thread/${this.messageRecipIdReplace}`; //replace [recpid] with receipent id

  messageDelete = `${this.messagesBaseUrl}delete/message/${this.messageDelIdReplace}`; //replace [msgid] with the msg id to delete

## message.service.ts

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

import { Observable } from 'rxjs';

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

import { HttpClientService } from './http-client.service';

import { PaginationService } from './pagination.service';

import { MessageParams } from '../models/helpers/message-params.model';

import { environment } from '../../../environments/environment';

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

@Injectable({

  providedIn: 'root'

})

export class MessageService {

  constructor(private apiUrlService: ApiUrlService,

    private httpClientService: HttpClientService,

    private pageService: PaginationService) { }

  getMessages(msgParams: MessageParams) {

    //add the pagination and filtering parameters

    const params = msgParams.getSearchParams();

    var url = this.apiUrlService.messagesGet;

    if (environment.displayConsoleLog) console.log(`getMessages url: ${url} params: ${params}`);

    return this.pageService.getPaginatedResult<MessageDto[]>(url, params);

  }

  getMessageThread(otherUserId: number) : Observable<MessageDto[]> {

    var url = this.apiUrlService.messageThread.replace(this.apiUrlService.messageRecipIdReplace, otherUserId.toString());

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

    return this.httpClientService.get<MessageDto[]>(url);

  }

  sendMessage(receipentUserId: number, content: string) {

    var url = this.apiUrlService.messageSend;

    return this.httpClientService.post<MessageDto>(url, { receipientUserId: receipentUserId, content: content });

  }

}

# /core/resolvers

## member-detail.resolver.ts

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

import { ActivatedRouteSnapshot, Resolve, RouterStateSnapshot } from "@angular/router";

import { Observable } from "rxjs";

import { userDto } from "../models/userDto.model";

import { MembersService } from "../services/members.service";

@Injectable({

    providedIn: 'root'

})

//user for /site/members/member-detail

//passing the query param tab which means to make the tab active

//recolver will help with that

//look at the routing module as well

export class MemberDetailResolver implements Resolve<userDto>{

    constructor(private membersService: MembersService){}

    /\*

    resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): userDto | Observable<userDto> | Promise<userDto> {

        throw new Error("Method not implemented.");

    }

    \*/

    resolve(route: ActivatedRouteSnapshot): Observable<userDto> {

        //get the route params

        const guid = route.paramMap.get('guid');

        const name = route.paramMap.get('name');

        return this.membersService.getMemberByGuId(guid!);

    }

}

### app-routing.moduel.ts

Assign the resolver to the members detail route

      //member detail is using resolver. member is key and MemberDetailResolver  is the value

      { path: 'members/detail/:guid/:name', component: MemberDetailComponent, resolve: { member: MemberDetailResolver } },

# Components

## /site/messages.component

### messages.component.ts

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

import { ToastrService } from 'ngx-toastr';

import { Subscription } from 'rxjs';

import { zMessageType } from '../../core/enums/zMessageType';

import { PaginatedResult } from '../../core/models/helpers/paginated-result.model';

import { MessageParams } from '../../core/models/helpers/message-params.model';

import { Pagination } from '../../core/models/helpers/pagination.model';

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

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

@Component({

  selector: 'app-messages',

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

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

})

export class MessagesComponent implements OnInit, OnDestroy {

  isGettingMessage: boolean = false;

  messages!: MessageDto[];

  pagination!: Pagination;

  msgParams!: MessageParams;

  msgTypeString!: string;

  zMessageType = zMessageType;

  messageSubscription!: Subscription;

  constructor(private msgService: MessageService, private toastrService: ToastrService) {

    this.msgParams = new MessageParams();

    this.msgTypeString = zMessageType[this.msgParams.messageType];

  }

  ngOnInit(): void {

    this.loadMessages();

  }

  ngOnDestroy(): void {

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

  }

  loadMessages() {

    this.pagination = <Pagination>{};

    this.isGettingMessage = true;

    this.messages = [];

    const msgTypeString = zMessageType[this.msgParams.messageType];

    //if the user is asked for a different type then reset the number

    if (this.msgTypeString !== msgTypeString) {

      this.msgTypeString = msgTypeString;

      this.msgParams.pageNumber = 1;

    }

    this.messageSubscription = this.msgService.getMessages(this.msgParams).subscribe({

      next: (response: PaginatedResult<MessageDto[]>) => {

        this.messages = response.result;

        if(this.messages && this.messages.length > 0)

          this.pagination = response.pagination;

      },

      error: e => { },

      complete: () => {

        this.isGettingMessage = false;

      }

    });

  }

  isOutBox() : boolean {

    return this.msgParams.messageType === zMessageType.outbox;

  }

  memberDetailLink(message: MessageDto): string{

    let url = '/members/detail/';

    if (this.isOutBox())

      url += message.receipientGuid + '/' + message.receipientUsername;

    else

      url += message.senderGuid + '/' + message.senderUsername;

    return url;

  }

  memberPhotoUrl(message: MessageDto): string{

    console.log(this.isOutBox());

    let url = message.senderPhotoUrl;

    if (this.isOutBox())

      url = message.receipientPhotoUrl;

    return url;

  }

  memberUserName(message: MessageDto): string{

    let name = message.senderUsername;

    if (this.isOutBox())

      name = message.receipientUsername;

    return name;

  }

  onPageChanged(event: any) {

    if (this.msgParams && this.msgParams.pageNumber !== event.page) {

      this.msgParams.pageNumber = event.page;

      this.loadMessages();

    }

  }

  onDeleteMessage(msgId: number) {

    this.msgService.deleteMessage(msgId).subscribe({

      next: () => {

        //remove the message from the messages by passing the index and total number of messages to delete which is one in this case

        this.messages.splice(this.messages.findIndex(x => x.id === msgId), 1);

        this.toastrService.success("Message deleted", 'Delete Result');

      },

      error: e => { },

      complete: () => { }

    });

  }

}

### messages.component.html

<!--btns to pull information-->

<div class="row my-4">

    <div class="btn-group" name="messageType">

        <button class="btn btn-primary btn-small me-1"

            [ngClass]="{'btn-success': msgTypeString == zMessageType[zMessageType.inboxUnread] }"

            [(ngModel)]="msgParams.messageType"

            btnRadio="{{ zMessageType.inboxUnread }}"

            (click)="loadMessages()">

            <i class="fa fa-envelope"></i> Unread

        </button>

        <button class="btn btn-primary btn-small me-1"

            [ngClass]="{'btn-success': msgTypeString == zMessageType[zMessageType.inbox] }"

            [(ngModel)]="msgParams.messageType"

            btnRadio="{{ zMessageType.inbox }}"

            (click)="loadMessages()">

            <i class="fa fa-envelope-open"></i> Inbox

        </button>

        <button class="btn btn-primary btn-small me-1"

            [ngClass]="{'btn-success': msgTypeString == zMessageType[zMessageType.outbox] }"

            [(ngModel)]="msgParams.messageType"

            btnRadio="{{ zMessageType.outbox }}"

            (click)="loadMessages()">

            <i class="fa fa-regular fa-paper-plane"></i> Outbox

        </button>

    </div>

</div>

<!--when no messages are found then displaying conditional messages-->

<div class="row" \*ngIf="!isGettingMessage && messages?.length === 0" [ngSwitch]="msgTypeString">

    <div \*ngSwitchCase="zMessageType[zMessageType.inbox]">Inbox no messages</div>

    <div \*ngSwitchCase="zMessageType[zMessageType.inboxUnread]">No unread messages</div>

    <div \*ngSwitchDefault>Nothing in outbox</div>

</div>

<!--displaying messages-->

<div class="row" \*ngIf="messages!.length > 0">

    <table class="table" style="cursor: pointer">

        <tr>

            <th style="width: 40%">Message</th>

            <th style="width: 20%">From / To</th>

            <th style="width: 20%">Sent / Received</th>

            <th style="width: 20%"></th>

        </tr>

        <tr [hidden]="isGettingMessage" \*ngFor="let message of messages" routerLink='{{ memberDetailLink(message) }}' [queryParams]="{tab: 3}">

            <td>{{message.messageContent}}</td>

            <td>

                <div>

                    <img class="img-circle rounded-circle me-2" src="{{ memberPhotoUrl(message)  || './assets/user.png' }}" alt="{{ memberUserName(message) }}">

                    <strong>{{ memberUserName(message) | titlecase }}</strong>

                </div>

            </td>

            <td>{{ message.dateMessageSent | timeago }}</td>

            <td>

                <!--there is link on the row. add second link to stop propagation -->

                <button type="button" class="btn btn-danger btn-sm" (click)="$event.stopPropagation()" (click)="onDeleteMessage(message.id)">Delete</button>

            </td>

        </tr>

    </table>

</div>

<!--pagination-->

<div class="d-flex justify-content-center" \*ngIf="pagination && !isGettingMessage && messages!.length > 0">

    <pagination

            [boundaryLinks]="true"

            [totalItems]="pagination.totalItems"

            [itemsPerPage]="pagination.itemsPerPage"

            [(ngModel)]="pagination.currentPage"

            (pageChanged)="onPageChanged($event)"

            previousText="&lsaquo;"

            nextText="&rsaquo;"

            firstText="&laquo;"

            lastText="&raquo;">

    </pagination>

</div>

### messages.component.css

.img-circle{

    max-height: 30px;

}

## /site/members/member-messages

Create a new component

* ng g c /site/members/memberMessages --skip-tests

### /core/modules/shared.module.ts

Add the component to the shared module declarations array.

If it gets added to the app.module.ts then remove it from there.

### member-messages.component.ts

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

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

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

  constructor(private messageService: MessageService) { }

  ngOnInit(): void {

  }

  ngOnDestroy(): void {

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

  }

  onSendMessage() {

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

      next: (message: MessageDto) => {

        this.messages.push(message);

        this.messageForm.reset();

      },

      error: e => { },

      complete: () => { }

    });

  }

}

### member-messages.component.html

<div class="card">

    <!--display messages start-->

    <div class="card-body">

        <!--no messages section-->

        <div \*ngIf="!messages || messages.length === 0">No messages yet... say hi by using the message box below</div>

        <!--messages start-->

        <ul \*ngIf="messages && messages.length > 0" class="chat">

            <li \*ngFor="let message of messages!">

                <div>

                    <span class="chat-image float-end">

                        <img src="{{ message.senderPhotoUrl || './assets/user.png' }}" alt="{{ message.senderUsername | titlecase }}" class="rounded-circle">

                    </span>

                    <div class="chat-body">

                        <div class="header">

                            <small class="text-muted">

                                <span class="fa fa-clock-o">{{ message.dateMessageSent | timeago }}</span>

                                <!--for the other user, not the logged in user-->

                                <span class="text-danger ms-1" \*ngIf="!message.dateMessageRead && message.senderId !== userId">(unread)</span>

                                <span class="text-success ms-1" \*ngIf="message.dateMessageRead && message.senderId !== userId">(read {{ message.dateMessageRead | timeago }})</span>

                            </small>

                        </div>

                    </div>

                    <p>{{ message.messageContent }}</p>

                </div>

            </li>

        </ul>

        <!--messages end-->

    </div>

    <!--display messages end-->

    <!--send message start -->

    <div class="card-footer">

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

            <div class="input-group">

                <input type="text" name="messageContent" required class="form-control" placeholder="Send a private message" [(ngModel)]="messageContent">

                <button [disabled]="!messageForm.valid" class="btn btn-primary ms-1 me-1" type="submit">Send</button>

                <button class="btn btn-danger" type="reset">Clear</button>

            </div>

        </form>

    </div>

    <!--send message end -->

</div>

### member-messages.component.html

.card{

    border: none;

}

.chat{

    list-style: none;

    margin: 0;

    padding: 0;

}

.chat li{

    margin-bottom: 10px;

    padding-bottom: 10px;

    border-bottom: 1px dotted #B3A9A9;

}

.rounded-circle{

    max-height: 50px;

}

## /site/members/member-detail

A lot has changed in this component.

### member-detail.component.ts

A lot has happened here due to message. Full type script is here with property comments for removed and added items.

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

import { ActivatedRoute, Router } from '@angular/router';

import { Subscription } from 'rxjs';

import { ToastrService } from 'ngx-toastr';

import {NgxGalleryOptions} from '@kolkov/ngx-gallery';

import {NgxGalleryImage} from '@kolkov/ngx-gallery';

import {NgxGalleryAnimation} from '@kolkov/ngx-gallery';

import { TabDirective, TabsetComponent } from 'ngx-bootstrap/tabs';

import { MembersService } from '../../../core/services/members.service';

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

import { userDto } from '../../../core/models/userDto.model';

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

import { environment } from '../../../../environments/environment';

@Component({

  selector: 'app-member-detail',

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

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

})

export class MemberDetailComponent implements OnInit, OnDestroy {

  //getting the reference to memberTabs

  //adding static since now we are using route resolver to get memebr data

  //also removed the check for members in the html

  @ViewChild('memberTabs', {static: true}) memberTabs!: TabsetComponent;

  //each tab placed in the html

  activeTab!: TabDirective;

  //message subscription

  messageSubscription!: Subscription;

  //messages

  messages:MessageDto[] = []

  member: userDto = <userDto>{};

  guid: string = "";

  name: string = "";

  //ngx gallery

  galleryOptions!: NgxGalleryOptions[];

  galleryImages!: NgxGalleryImage[];

  paramSubscription!: Subscription;

  membersSubscription!: Subscription;

  queryParamSubscription!: Subscription;

  constructor(private memberService: MembersService,

    private route: ActivatedRoute,

    private router: Router,

    private toastrService: ToastrService,

    private messageService: MessageService) { }

  ngOnInit(): void {

    //gallery optioins for the images

    this.setGalleryOptions();

    //we have a route resolver so member is coming from there

    /\*

    //there are two ways to get the id and name from the route

    ////we can get the param using snapshot, but this usually breaks when the item is clicked from inside the page and the url is the same as already showing

    //#1

    //this.route.snapshot.paramMap.get('id');

    //#2 way, preferred

    this.paramSubscription = this.route.params.subscribe(params => {

      this.guid = params['guid'];

      this.name = params['name'];

      if (environment.displayConsoleLog) {

        console.log(`Param guid: ${this.guid} name: ${this.name}`);

      }

      if (!this.guid) {

        this.toastrService.error("GUID not received on the page", "User Error")

        this.router.navigate(['/members/list'])

      }

      //we have the id so get the data

      this.initData();

    });

    \*/

    this.route.data.subscribe({

      next: data => {

        this.member = data['member'];

      }

    });

    this.queryParamSubscription = this.route.queryParams.subscribe({

      next: params => {

        //tab is being passed in as query param

        let tab: number = +params['tab'];

        if (isNaN(tab) || tab > 3) tab = 0;

        this.selectTab(tab);

      },

      error: e => { },

      complete: () => {}

    });

    //moved it here from initData since route resolver is now being used to pcik the member

    //image gallery

    this.galleryImages = this.getImages();

  }

  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();

  }

  //check https://www.npmjs.com/package/@kolkov/ngx-gallery

  setGalleryOptions() {

    this.galleryOptions = [

      {

        width: '500px',

        height: '500px',

        imagePercent: 100,

        thumbnailsColumns: 4,

        imageAnimation: NgxGalleryAnimation.Slide,

        preview: false

      }

    ]

  }

  //check https://www.npmjs.com/package/@kolkov/ngx-gallery

  getImages(): NgxGalleryImage[] {

    const imageUrls: NgxGalleryImage[] = [];

    if (!this.member || !this.member.photos) return imageUrls;

    for (const photo of this.member.photos) {

      imageUrls.push({

        small: photo?.url,

        medium: photo?.url,

        big: photo?.url

      })

    }

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

    return imageUrls;

  }

  //not using due to route resolver now. moved this.galleryImages = this.getImages(); to ngOnInit

  /\*

  initData() {

    this.membersSubscription = this.memberService.getMemberByGuId(this.guid).subscribe({

      next: (member: userDto) => {

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

        if (member.userName.toLowerCase() != this.name.toLowerCase()) {

          this.toastrService.error('Mismatched user');

          this.router.navigate(['/members/list']);

        }

        this.member = member;

        //image gallery

        this.galleryImages = this.getImages();

      },

      error: e => { },

      complete: () => { }

    });

  }

  \*/

  loadMessages() {

    this.messageSubscription = this.messageService.getMessageThread(this.member.id).subscribe({

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

        this.messages = messages;

      },

      error: e => { },

      complete: () => { }

    });

  }

  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) {

      this.loadMessages();

    }

  }

  //make message tab active when the user clicks on Message button in the left nav

  selectTab(tabId: number) {

    this.memberTabs.tabs[tabId].active = true;

  }

}

### member-detail.component.html

removed member check since static is being used for the view child and also route resolver is being used to get the data

<!--

    removed \*ngIf="member" since now using route resolver and

    and need to select the messages tab if it has been passed in

-->

<div class="row" >

Update the Message button and add a selectTab to it

  <button class="btn btn-success ms-1" (click)="selectTab(3)">Message</button>

* Add a local reference on tabset and add onTabActivate on each tab
* Add the above created member-messages component to the member details, messages tab

        <tabset class="member-tabset" #memberTabs>

            <tab heading="About {{ member.displayName }}" (selectTab)="onTabActivate($event)">

                <h4>Description</h4>

                <p>{{ member.introduction }}</p>

                <h4>Looking for</h4>

                <p>{{ member.lookingFor }}</p>

            </tab>

            <tab heading="Interests" (selectTab)="onTabActivate($event)">

                <h4>Interests</h4>

                <p>{{ member.interests }}</p>

            </tab>

            <tab heading="Photos" (selectTab)="onTabActivate($event)">

                <div \*ngIf="galleryImages">

                    <!--applying local style rather than via a styles.css-->

                    <ngx-gallery [options]="galleryOptions" [images]="galleryImages" style="display: inline-block; margin-bottom: 20px;"></ngx-gallery>

                </div>

            </tab>

            <tab heading="Messages" (selectTab)="onTabActivate($event)">

                <h4>Messages</h4>

                <app-member-messages [messages]="messages" [userId]="member.id"></app-member-messages>

            </tab>

        </tabset>

## /site/members/member-card.component

### member-card.component.html

Take the user to the members detail and pass the query parameter to show messages tab

            <!--take the user to the members detail tab 3-->

            <li class="list-inline-item"><button class="btn btn-primary"

                routerLink="/members/detail/{{member.guId}}/{{member.userName}}"

                [queryParams]="{tab: 3}"

                routerLinkActive="router-link-active" ><i class="fa fa-light fa-envelope"></i></button></li>