Table of Contents

[Basics 2](#_Toc116677565)

[Current Resources with Changes 2](#_Toc116677566)

[New Resources Added 2](#_Toc116677567)

[npm install / ng serve 2](#_Toc116677568)

[/core/enums 2](#_Toc116677569)

[zMemberGetBy.ts 2](#_Toc116677570)

[zUserLiketype.ts 2](#_Toc116677571)

[/core/models 2](#_Toc116677572)

[likeDto.ts 2](#_Toc116677573)

[/core/models/helpers 3](#_Toc116677574)

[like-params.models.ts 3](#_Toc116677575)

[/core/interceptors 3](#_Toc116677576)

[error.interceptor.ts 3](#_Toc116677577)

[/core/services 4](#_Toc116677578)

[api-url.services.ts 4](#_Toc116677579)

[member.service.ts 4](#_Toc116677580)

[Components 5](#_Toc116677581)

[/site/members/member-card 5](#_Toc116677582)

[member-card.component.ts 5](#_Toc116677583)

[member-card.component.html 6](#_Toc116677584)

[/site/members/list.component 6](#_Toc116677585)

[list.component.ts 6](#_Toc116677586)

[list.component.html 7](#_Toc116677587)

# 

# 

# Basics

|  |  |
| --- | --- |
| WorkingFolder | Copy the content of “Site-02-PagingSortingFiltering” in “Site-03-Like Feature” and issue   * npm install * ng serve |

# Current Resources with Changes

1. /core/enums/zMemberGetBy
2. /core/services/api-url.services
3. /core/services/member.service
4. /core/interceptors/error.interceptor
5. /site/members/member-card
6. /site/lists

# New Resources Added

1. /core/enums/zUserLikeType
2. /core/models/likeDto
3. /core/models/helpers/userParams

# npm install / ng serve

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

# /core/enums

## zMemberGetBy.ts

Updated the values

export enum zMemberGetBy{

    id = 0,

    userName = 1,

    guid = 2,

}

## zUserLiketype.ts

export enum zUserLikeType{

    none = 0,

    liked = 1,

    likedBy = 2

}

# /core/models

## likeDto.ts

We have been so far creating model classes. This one is an interface. We don’t need a constructor so created as an interface.

export interface LikeDto {

    id: number

    guId: string;

    userName: string;

    photoUrl: string;

    age: number;

    displayName: string;

    city: string;

}

# /core/models/helpers

## like-params.models.ts

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

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

export class LikeParams {

    pageNumber: number = 1;

    pageSize: number = 5;

    userLikeType: zUserLikeType = zUserLikeType.liked;

    constructor() { }

    //helper metod to build search params

    getPaginationSearchParams() : HttpParams {

        let params = new HttpParams();

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

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

        return params;

    }

    //helper metod to build search params

    getLikesSearchParams() : HttpParams {

        let params = this.getPaginationSearchParams();

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

        return params;

    }

}

# /core/interceptors

## error.interceptor.ts

Updated the 400 error part to handle where no message is returned with BadRequest from the api

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

import {

  HttpRequest,

  HttpHandler,

  HttpEvent,

  HttpInterceptor

} from '@angular/common/http';

import { catchError, Observable, throwError } from 'rxjs';

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

import { ToastrService } from 'ngx-toastr';

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

@Injectable()

export class ErrorInterceptor implements HttpInterceptor {

  constructor(private router: Router, private toastrService: ToastrService) {}

  intercept(request: HttpRequest<unknown>, next: HttpHandler): Observable<HttpEvent<unknown>> {

    return next.handle(request).pipe(

      //error is the http response that we see when writing the console.log

      catchError(error => {

        if (error) {

          switch (error.status) {

            case 400:

              if (error.error.errors) {

                //400 validation errors, it has error.error.errors array, start

                const modalStateErrors = [];

                for (const key in error.error.errors) {

                  if (error.error.errors[key])

                    modalStateErrors.push(error.error.errors[key]);

                }

                //modalStateErrors so that the errors could be displayed near the form

                throw modalStateErrors.flat();

                //400 validation errors, end

              }

              else if (typeof(error.error) === 'object') {

                //400 all other errors, start

                //will use the toastr service to display the error

                this.toastrService.error(`${error.statusText}: ${error.error}`, error.status);

                //400 all other errors, end

              }

              else {

                this.toastrService.error(error.error, error.status);

              }

              break;

            case 401:

              //will use the toastr service to display the error

              this.toastrService.error(`${error.statusText}: ${error.error}`, error.status);

              this.router.navigateByUrl('/');

              break;

            case 404:

              //redirect to the not found page

              this.toastrService.error(`${error.statusText}: ${error.error}`, error.status)

              this.router.navigateByUrl('/not-found');

              break;

            case 500:

              //the error object. It has the properties message and details that you would need to tap into

              const navigationExtras: NavigationExtras = { state: { error: error.error } };

              //redirect to the server error page

              this.router.navigateByUrl('/server-error', navigationExtras);

              break;

            default:

              this.toastrService.error('Something unexpected went wrong');

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

              break;

          }

        }

        //return the error

        return throwError(error);

      })

    );

  }

}

# /core/services

## api-url.services.ts

add the properties for likes controller

  private likesBaseUrl = `${this.apiBaseUrl}likes/`;

  likeUserIdReplace = "[id]";

  likeUserNameReplace = "[name]";

  likesForUser = `${this.likesBaseUrl}user/likes`;

## member.service.ts

Update due to zMemberGetBy changes

  private getMemberCache(key: any, by: zMemberGetBy): userDto | undefined{

    let member: userDto | undefined;

    if (!key || !by) return member;

    //using map and reduce to create a single array out of key value array pairs

    const members = [...this.memberCache.values()].reduce((arr, elem) => arr.concat(elem.result), []);

    if (environment.displayConsoleLog) {

      console.log("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

      console.log(members);

      console.log("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

    }

    if (members.length <= 0) return member;

    switch (by as zMemberGetBy) {

      case zMemberGetBy.id:

        member = members.find((x: userDto) => x.id === +key);

        break;

      case zMemberGetBy.userName:

        member = members.find((x: userDto) => x.userName === key);

        break;

      case zMemberGetBy.guid:

        member = members.find((x: userDto) => x.guId === key);

        break;

    }

    return member;

  }

Also, added new methods for the “likes”

  /\*\*

   \* A add method to add likes

   \* @param id id of the user getting liked

   \*

   \* @returns returns Ok

   \*/

  addLike(id: number) : Observable<any> {

    var url = this.apiUrlService.likeAdd

      .replace(this.apiUrlService.likeUserIdReplace, id.toString())

      .replace(this.apiUrlService.likeUserNameReplace, this.user.userName);

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

    return this.httpClientService.post(url, {});

  }

  //not using likeDto instead using partial userDto since the properties are the same

  getLikes(like\_parms: LikeParams) {

    //add the pagination and filtering parameters

    const params = like\_parms.getLikesSearchParams();

    var url = this.apiUrlService.likesForUser;

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

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

  }

# Components

## /site/members/member-card

### member-card.component.ts

added onAddLike method

  onAddLike(member: userDto) {

    this.addSubscription = this.memberService.addLike(member.id).subscribe({

      next: () => {

        this.toastrService.success(`You have liked ${member.displayName}`);

      },

      error: e => {

        //no need to do something here since interceptor will display

      },

      complete: () => {

        //no need to do something here since interceptor will display

      }

    });

  }

### member-card.component.html

Add the method call to the like “thumbsup” icon

<li class="list-inline-item"><button class="btn btn-primary" (click)="onAddLike(member)"><i class="fa fa-light fa-thumbs-up"></i></button></li>

## /site/members/list.component

### list.component.ts

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

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

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

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

import { LikeDto } from '../../core/models/likeDto';

import { LikeParams } from '../../core/models/helpers/like-params.model';

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

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

@Component({

  selector: 'app-lists',

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

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

})

export class ListsComponent implements OnInit {

  //members!: Partial<userDto[]>;

  members: Partial<userDto[]> = [];

  //userLikeType: zUserLikeType = zUserLikeType.liked;

  likeParams!: LikeParams;

  pagination!: Pagination;

  userTypeString!: string;

  //to be used in the component

  zUserLikeType = zUserLikeType;

  constructor(private memberService: MembersService) {

    this.likeParams = new LikeParams();

    this.userTypeString = zUserLikeType[this.likeParams.userLikeType];

  }

  ngOnInit(): void {

    this.loadLikes();

  }

  loadLikes() {

    const userTypeString = zUserLikeType[this.likeParams.userLikeType];

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

    if (this.userTypeString != userTypeString) {

      this.userTypeString = userTypeString;

      this.likeParams.pageNumber = 1;

    }

    this.memberService.getLikes(this.likeParams).subscribe({

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

        this.members = response.result;

        this.pagination = response.pagination;

      },

      error: e => { },

      complete: () => {}

    });

  }

  onPageChanged(event: any) {

    if (this.likeParams)

      this.likeParams.pageNumber = event.page;

    //get next batch of members

    this.loadLikes();

  }

}

### list.component.html

<div class="text-center mt-3">

    <h2>{{ userTypeString === zUserLikeType[zUserLikeType.liked] ? 'Members I like' : 'Members who like me' }}</h2>

</div>

<div class="container mt-3">

    <div class="row">

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

            <label type="button" class="btn btn-small btn-info my-1 me-1" [(ngModel)]="likeParams.userLikeType" btnRadio="{{ zUserLikeType.liked }}" tabindex="0" role="button" (click)="loadLikes()">Members I like</label>

            <label type="button" class="btn btn-small btn-info my-1" [(ngModel)]="likeParams.userLikeType" btnRadio="{{ zUserLikeType.likedBy }}" tabindex="1" role="button" (click)="loadLikes()">Members who like me</label>

        </div>

    </div>

</div>

<div class="row mt-3" \*ngIf="members">

    <div class="col-2" \*ngFor="let member of members">

        <app-member-card [member]="member!"></app-member-card>

    </div>

</div>

<div class="d-flex justify-content-center" \*ngIf="pagination">

    <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>