Table of Contents

[Basics 3](#_Toc116512720)

[Current Resources with Changes 3](#_Toc116512721)

[New Resources Added 3](#_Toc116512722)

[npm install / ng serve 3](#_Toc116512723)

[Looking for Headers 3](#_Toc116512724)

[/core/constants/app-constants.ts 4](#_Toc116512725)

[/core/models 4](#_Toc116512726)

[userTokenDto.model.ts 4](#_Toc116512727)

[/core/models/helpers 4](#_Toc116512728)

[user-params.model.ts 4](#_Toc116512729)

[pagination.model.ts 5](#_Toc116512730)

[paginated-result.model.ts 5](#_Toc116512731)

[/core/enums 5](#_Toc116512732)

[zMemberGetBy.ts 5](#_Toc116512733)

[/core/services 5](#_Toc116512734)

[local-storage.service.ts 5](#_Toc116512735)

[http-client.service.ts 5](#_Toc116512736)

[members.service.ts 6](#_Toc116512737)

[ngx-bootstrap pagination 11](#_Toc116512738)

[ngx-bootstrap buttons 11](#_Toc116512739)

[ngx-timeago 12](#_Toc116512740)

[Component 12](#_Toc116512741)

[/site/members/member-detail 12](#_Toc116512742)

[member-detail.component.html 12](#_Toc116512743)

[/site/members/member-edit 12](#_Toc116512744)

[member-edit.component.html 12](#_Toc116512745)

[/site/members/members-list 12](#_Toc116512746)

[members-list.component.ts 12](#_Toc116512747)

[members-list.component.html 13](#_Toc116512748)

# Basics

|  |  |
| --- | --- |
| WorkingFolder | Copy the content of “Site-01-Basics” in “Site-02-PagingSortingFiltering” and issue   * npm install |

# Current Resources with Changes

1. /core/models/userTokenDto.model.ts
2. /core/services/local-storage.service.ts
3. /core/services/members.service
4. /core/services/http-client.service.ts
5. /core/constants/app-constants.ts
6. /core/enums
7. /site/members/member-detail
8. /site/members/member-edit.component

# New Resources Added

1. /core/models/helpers
2. ngx-bootstrap pagination
3. ngx-bootstrap buttons

# npm install / ng serve

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

Login and go to matches. You’ll see only 10 records (default) without even doing anything.

## Looking for Headers

* Go to dev tools (F12) and network and refresh page.
* Then go to users request and you’ll see the header.

Graphical user interface, application

Description automatically generated

# /core/constants/app-constants.ts

Add the genders and pagination header name

    public static readonly Female = "female";

    public static readonly Male = "male";

    public static readonly membersOrderByLastActive = 'lastActive';

    public static readonly membersOrderByCreated = 'created';

    //headers

    public static readonly PaginationHeader = 'pagination';

# /core/models

## userTokenDto.model.ts

Add gender since it is now getting returned after login or register.

export class UserTokenDto {

    constructor(public userName: string = "",

        public guId: string = "",

        public token: string = "",

        public mainPhotoUrl = "",

        public displayName = "",

        public gender = "") { }

}

# /core/models/helpers

Create a new “helpers” folder under “models” folder

## user-params.model.ts

This will be used to pass the parameters from the component to the service

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

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

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

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

export class UserParams {

    gender: string;

    minAge: number = 18;

    maxAge: number = 99;

    pageNumber: number = 1;

    pageSize: number = 5;

    orderBy: string = AppConstants.membersOrderByLastActive;

    constructor(user: UserTokenDto) {

        //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

    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

    getMemberSearchParams() : HttpParams {

        let params = this.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;

    }

}

## pagination.model.ts

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

//need to match exactally as Pagination response header

export class Pagination {

    constructor(public currentPage: number, public itemsPerPage: number, public totalItems: number, public totalPages: number){}

}

## paginated-result.model.ts

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

It will have two members to hold the users and pagination information. Result is generic type and will hold the users.

import { Pagination } from "./pagination.model";

export class PaginatedResult<T> {

    result!: T;

    pagination!: Pagination;

    constructor() {}

}

# /core/enums

## zMemberGetBy.ts

export enum zMemberGetBy{

    id = 1,

    userName = 2,

    guid = 3,

}

# /core/services

## local-storage.service.ts

Add the helper property to get the logged in users gender as well. Do logout and log back in so that gender is retrieved.

  getLoggedinUserGender: string = this.getLoggedInUser?.gender;

## http-client.service.ts

Here working with pagination so need the full response back.

The current get method is returning response body. Need a new method to return full http response.

When writing local in any other service can do

let params = new HttpParams();

    if (page && itemsPerPage) {

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

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

    }

    return this.http.get<userDto[]>(url, { observe: 'response', params }).pipe(

      map(response => {

        this.paginatedResult.result = response.body!;

        if (response.headers.get(AppConstants.PaginationHeader) !== null) {

          this.paginatedResult.pagination = JSON.parse(response.headers.get(AppConstants.PaginationHeader)!);

        }

        return this.paginatedResult;

      })

    );

Since we have a generic service for the http calls, we need a new method to return full http response back. Even though when passing

    /\*\*

   \* A GET method

   \* @param url api url

   \* @param params pass empty meaning do not pass when not used, will cover stuff like ?x=1&y=2,

   \*        instead use HttpParams

   \*        let params = new HttpParams();

   \*        params = params.append('pageNumber', page.toString());

   \*        params = params.append('pageSize', itemsPerPage.toString());

   \* @returns rreturns the full HttpResponse

   \*/

  getWithFullResponse<TReturn>(ul: string, params = {}) : Observable<HttpResponse<TReturn>> {

    return this.httpClient

      .get<TReturn>(ul, { observe: 'response', params })

      .pipe(retry(this.retries));

  }

## members.service.ts

A lot has changed in this service. To see the changes compare it against “Site-01-Basics” folder

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

import { map, Observable, of, take } from 'rxjs';

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

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

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

import { LocalStorageService } from './local-storage.service';

import { AccountService } from './account.service';

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

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

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

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

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

import { UserParams } from '../models/helpers/user-params.model';

import { zMemberGetBy } from '../enums/zMemberGetBy';

@Injectable({

  providedIn: 'root'

})

export class MembersService {

  //to pass the auth token to the api, later will use interceptor

  private httpOptions;

  //state mangement. We dont want to pull the user info every time

  //little tricky since page size, filtering and page number are at play

  private members: userDto[] = []; //not used any more

  private memberCache = new Map();

  private userParams!: UserParams;

  private user!: UserTokenDto;

  constructor(private apiUrlService: ApiUrlService,

      private httpClientService: HttpClientService,

      private localStorageService: LocalStorageService,

      private http: HttpClient,

      private accountService: AccountService) {

    //to pass the auth token to the api, later will user interceptor

    //not using this any more, using the interceptor

    this.httpOptions = {

      headers: new HttpHeaders({

        Authorization: AppConstants.Bearer + localStorageService.getLoggedinToken

      })

    };

    this.retrieveUserInfo();

  }

  private retrieveUserInfo() {

    //get the logged in user

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

      next: (user: UserTokenDto) => {

        this.user = user;

        this.resetUserParams();

      },

      error: e => { },

      complete: () => { }

    });

  }

  resetUserParams() : UserParams {

    //filtering information is un userParams, check the class for more details.

    this.userParams = new UserParams(this.user);

    return this.userParams;

  }

  getUserParams() : UserParams {

    return this.userParams;

  }

  setUserParams(user\_params: UserParams) {

    this.userParams = user\_params;

  }

  getUser() : UserTokenDto {

    return this.user;

  }

  /\*\*

   \* A GET method

   \* @returns returns Observable userDto[]

   \*/

  /\*

  getMembers(): Observable<userDto[]> {

    //state mangement. when we habe members then return that

    if (this.members.length > 0) {

      //return observable

      return of(this.members);

    }

    const url = this.apiUrlService.usersAll;

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

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

    //check jwt interceptor for details

    //return this.httpClientService

    //  .get<userDto[]>(url, this.httpOptions);

    return this.httpClientService.get<userDto[]>(url).pipe(

        map(members => {

          //set the members for state management and return as well

          this.members = members;

          return members;

        })

      );

  }

  \*/

  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.getMemberSearchParams();

    //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.getPaginatedResult<userDto[]>(url, params).pipe(

      map(response => {

        //set the cache

        this.memberCache.set(cacheKey, response);

        return response;

      })

    );

  }

  private 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;

      })

    );

  }

  /\*\*

   \* A GET method

   \* @param guid guid to fetch

   \* @returns returns Observable userDto

   \*/

  getMemberByGuId(guid: string): Observable<userDto>{

    //state management. When the member is in members array then return from there other wise get from api

    const member = this.getMemberCache(guid, zMemberGetBy.guid);//this.getMember(guid, zMemberGetBy.guid);

    if (member !== undefined) return of(member);

    let url = this.apiUrlService.userByGuId;

    //replace [guid] with the guid in the url

    url = url.replace(this.apiUrlService.userByGuIdReplace, guid.toString());

    if (environment.displayConsoleLog) console.log(`Users by id url: ${url}`)

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

    //check jwt interceptor for details

    /\*

    return this.httpClientService

      .get<userDto>(url, this.httpOptions);

      \*/

    return this.httpClientService.get<userDto>(url);

  }

  /\*\*

   \* A GET method

   \* @param id id to fetch

   \* @returns returns Observable userDto

   \*/

  getMemberById(id: number): Observable<userDto>{

    //state management. When the member is in members array then return from there other wise get from api

    const member = this.getMemberCache(id, zMemberGetBy.id);//this.getMember(id, zMemberGetBy.id);

    if (member !== undefined) return of(member);

    let url = this.apiUrlService.userById;

    //replace [id] with the id in the url

    url = url.replace(this.apiUrlService.userByIdReplace, id.toString());

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

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

    //check jwt interceptor for details

    /\*

    return this.httpClientService

      .get<userDto>(url, this.httpOptions);

      \*/

      return this.httpClientService.get<userDto>(url);

  }

  /\*\*

   \* A GET method

   \* @param userName userName to fetch

   \* @returns returns Observable userDto

   \*/

  getMemberByUserName(userName: string): Observable<userDto>{

    //state management. When the member is in members array then return from there other wise get from api

    const member = this.getMemberCache(userName, zMemberGetBy.userName);//this.getMember(userName, zMemberGetBy.userName);

    if (member !== undefined) return of(member);

    let url = this.apiUrlService.userByName;

    //replace [name] with the userName in the url

    url = url.replace(this.apiUrlService.userByNameReplace, userName);

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

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

    //check jwt interceptor for details

    /\*return this.httpClientService

      .get<userDto>(url, this.httpOptions);

    \*/

      return this.httpClientService.get<userDto>(url);

  }

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

    let member: userDto | undefined;

    if (!by || !key || this.members.length <= 0) return member;

    switch (by) {

      case zMemberGetBy.id:

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

        break;

      case zMemberGetBy.userName:

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

        break;

      case zMemberGetBy.guid:

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

        break;

    }

    return member;

  }

  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) {

      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;

  }

  /\*\*

   \* A PUT method

   \* @param member member to update.

   \* Only updating introduction, lookingFor, interests, city, country so still can send the full member model

   \* @returns returns Observable noContent 204

   \*/

  updateMember(member: userDto) {

    let url = this.apiUrlService.userUpdate;

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

    return this.httpClientService.put(url, member).pipe(

      map(() => {

        const index = this.members.indexOf(member);

        this.members[index] = member;

      })

    );

  }

  /\*\*

 \* A PUT method to make the photo main

 \* @param photoId photo id to make active

 \*

 \* @returns returns nocontent

 \*/

  setMainPhoto(photoId: number) {

    let url = this.apiUrlService.userSetPhotoMain;

    url = url.replace(this.apiUrlService.userPhotoIdReplace, photoId.toString());

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

    //pass an empty object since it is a put request

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

  }

  /\*\*

 \* A DELETE method to delete the photo

 \* @param photoId photo id to delete

 \*

 \* @returns returns Ok

 \*/

  deletePhoto(photoId: number) {

    let url = this.apiUrlService.userDeletePhoto;

    url = url.replace(this.apiUrlService.userPhotoIdReplace, photoId.toString());

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

    return this.httpClientService.delete(url);

  }

}

# ngx-bootstrap pagination

Link: <https://valor-software.com/ngx-bootstrap/#/components/pagination?tab=overview>

ngx-bootstrap is already added so go to /core/modules/shared.module.ts

**import**

import { PaginationModule } from 'ngx-bootstrap/pagination';

Then add to the imports array

PaginationModule.forRoot(),

And finally add to the export as well

PaginationModule

Will be using the custom link so look at the following

<https://valor-software.com/ngx-bootstrap/#/components/pagination?tab=overview#custom-links-content>

1. <pagination [boundaryLinks]="true" [totalItems]="77"
2. previousText="&lsaquo;" nextText="&rsaquo;" firstText="&laquo;" lastText="&raquo;">
4. </pagination>

# ngx-bootstrap buttons

Link: <https://valor-software.com/ngx-bootstrap/#/components/buttons?tab=overview>

ngx-bootstrap is already added so go to /core/modules/shared.module.ts

**import**

import { ButtonsModule } from 'ngx-bootstrap/buttons';

Then add to the imports array

ButtonsModule.forRoot(),

And finally add to the export as well

ButtonsModule,

And then use it as

1. <pre class="card card-block card-header">{{checkModel | json}}</pre>
3. <div class="btn-group">
4. <label class="btn btn-primary" [(ngModel)]="checkModel.left"
5. btnCheckbox tabindex="0" role="button">Left</label>
6. <label class="btn btn-primary" [(ngModel)]="checkModel.middle"
7. btnCheckbox tabindex="0" role="button">Middle</label>
8. <label class="btn btn-primary" [(ngModel)]="checkModel.right"
9. btnCheckbox tabindex="0" role="button">Right</label>
10. </div>

# ngx-timeago

Link: <https://www.npmjs.com/package/ngx-timeago>

* npm install ngx-timeago

Add it to the shared module

Import it

import { TimeagoModule } from 'ngx-timeago';

and then add to the imports array

   TimeagoModule.forRoot(),

Add it to the exports array as well

    TimeagoModule,

# Component

## /site/members/member-detail

### member-detail.component.html

Format the dates, use time ago for lastActive

                <div>

                    <strong>Last Active:</strong>

                    <p>{{ member.lastActive | timeago }}</p>

                </div>

                <div>

                    <strong>Member Since:</strong>

                    <p>{{ member.createdOn | date:'dd MMMM yyyy' }}</p>

                </div>

## /site/members/member-edit

### member-edit.component.html

                <div>

                    <strong>Last Active:</strong>

                    <p>{{ member.lastActive | timeago }}</p>

                </div>

                <div>

                    <strong>Member Since:</strong>

                    <p>{{ member.createdOn | date:'dd MMMM yyyy' }}</p>

                </div>

## /site/members/members-list

### members-list.component.ts

A lot has changed, compare it against “Site-01-Basics” folder to see the changes

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

import { Observable, take } from 'rxjs';

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

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

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

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

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

import { UserParams } from '../../../core/models/helpers/user-params.model';

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

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

import { AccountService } from '../../../core/services/account.service';

@Component({

  selector: 'app-member-list',

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

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

})

export class MemberListComponent implements OnInit {

  members: userDto[] = [];

  pagination!: Pagination;

  userParams!: UserParams;

  //filtering

  genderList = [

    { value: AppConstants.Male, display: 'Males' },

    { value: AppConstants.Female, display: 'Females' }

  ];

  constructor(private membersService: MembersService) {

    this.userParams = this.membersService.getUserParams();

  }

  ngOnInit(): void {

    this.loadMembers();

  }

  loadMembers() {

    this.membersService.setUserParams(this.userParams);

    this.membersService.getMembers(this.userParams).subscribe({

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

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

        this.members = response.result;

        this.pagination = response.pagination;

      }, error: e => {

        //error will get displayed by the error interceptor

      }, complete: () => {}

    });

  }

  onResetFilters() {

    this.userParams = this.membersService.resetUserParams();

    this.loadMembers();

  }

  onGenderSelect() {

    //reset the page number to 1 since the page the user is on my not exist when the gender is changed

    this.userParams.pageNumber = 1;

  }

  onPageChanged(event: any) {

    if (this.userParams)

      this.userParams.pageNumber = event.page;

    //get next batch of members

    this.loadMembers();

  }

}

### members-list.component.html

A lot has changed, compare it against “Site-01-Basics” folder to see the changes

<!--result count-->

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

    <h2>Your matches - {{ pagination?.totalItems }} found</h2>

</div>

<!--filtering-->

<div class="container mt-3 mb-3">

    <form #form="ngForm" class="d-flex flex-row align-items-center flex-wrap" (ngSubmit)="loadMembers()" autocomplete="off">

        <label for="minAge" class="my-1 m2-2 input-group-text">Age from:</label>

        <input type="number" class="form-control form-input my-1 me-sm-2 w-auto" style="width: 70px !important;" name="minAge" [(ngModel)]="userParams.minAge">

        <label for="maxAge" class="my-1 m2-2 input-group-text">Age to:</label>

        <input type="number" class="form-control form-input my-1 me-sm-2 w-auto" style="width: 70px !important;" name="maxAge" [(ngModel)]="userParams.maxAge">

        <label for="gender" class="my-1 m2-2 input-group-text">Show:</label>

        <select name="gender" class="form-control form-select my-1 me-sm-2 w-auto" [(ngModel)]="userParams.gender" (change)="onGenderSelect()">

            <option \*ngFor="let gender of genderList" [value]="gender.value">{{gender.display}}</option>

        </select>

        <!--

             style="width: 130px;"

         -->

        <button class="btn btn-small btn-primary my-1 me-1" type="submit">Apply Filters</button>

        <button class="btn btn-small btn-secondary my-1 me-1" type="button" (click)="onResetFilters()">Reset Filters</button>

        <!--order by bottons-->

        <div class="col">

            <div class="btn-group float-end">

                <label type="button" class="btn btn-small btn-info my-1 me-1" name="orderBy" [(ngModel)]="userParams.orderBy" btnRadio="lastActive" tabindex="0" role="button" (click)="loadMembers()">Last Active</label>

                <label type="button" class="btn btn-small btn-info my-1" name="orderBy" [(ngModel)]="userParams.orderBy" btnRadio="created" tabindex="1" role="button" (click)="loadMembers()">Newest Members</label>

            </div>

        </div>

    </form>

</div>

<!--result-->

<div class="row">

    <div \*ngFor="let member of members" class="col-sm-4 col-md-2 col-12">

        <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>