# Project

* Site-09-Api-Ng-templateform-candeactivate-spinner-intercepter-caching
  + Copied from: Site-08-Api-Ng-interceptor-membercards-photogallery
* For the “MySocialConnect-SPA” issue npm install command to install all packages
  + ng serve : to run the spa

# New & Updates

|  |  |
| --- | --- |
| New | Updates |
| /core/guards/prevent-unsaved-changes.guard.ts | /site/members/member-edit |
| /core/services/spinner-busy.service.ts | environment.ts |
| /core/interceptors/loading.interceptor.ts | environment.development.ts |
|  | /core/services/helper.service.ts |
|  | angular.json |
|  | /core/modules/shared.module.ts |
|  | App.module.ts |
|  | App.component.html |
|  | /core/services/member.service.ts |
|  | /site/members/member-list |

# Clear Database

If you want to clear database then drop it and recreate it

dotnet ef database drop

dotnet ef database update

# Preventing the User from moving away when have changes

## /core/Guards

This will only prevent when user will click other routes

### prevent-unsaved-changes.guards.ts - CanDeactivate

> ng g g core/guards/preventUnsavedChanges --skip-tests

? Which type of guard would you like to create? CanDeactivate

import { CanDeactivateFn } from '@angular/router';

import { MemberEditComponent } from '../../site/members/member-edit/member-edit.component';

//this will not prevent when user will type a different URL

//for different URL check member-edit.component.ts

//export const preventUnsavedChangesGuard: CanDeactivateFn<unknown> = (component, currentRoute, currentState, nextState) => {

export const preventUnsavedChangesGuard: CanDeactivateFn<MemberEditComponent> = (component) => {

  if(component.editForm?.dirty){

    return confirm('Are you sure you want to continue? Any unsaved changes will be lost.');

  }

  return true;

};

### app-routing.module.ts

      { path: 'members/edit', component: MemberEditComponent, canDeactivate: [preventUnsavedChangesGuard] },

## Prevent when user go to another website

### member-edit.component.ts

  //access the editForm

  @ViewChild('editForm') editForm: NgForm | undefined;

  //We have CanDeactiavte PreventUnsavedChanges guard. It will prevent the user from clicking away when have changes

  //However, the user can type a different url in the broweser

  @HostListener('window:beforeunload', ['$event']) unloadNotification($event: any){

    if(this.editForm?.dirty){

      $event.returnValue = true;

    }

  }

# Update Member

## /Core/services/member.service.ts

  updateMember(user: UserDto){

    const url = this.helperService.urlUserUpdate;

    this.helperService.logIfFrom(url, "memberService updateMember");

    return this.httpClientService.put(url, user);

  }

## /site/members/member-edit

### member-edit.componenet.css

.img-thumbnail{

    margin: 25px;

    width: 85%;

    height: 85%;

    border-radius: 5px;

}

.card-body{

    padding: 9 25px;

}

.card-footer{

    padding: 10px 15px;

    background-color: #fff;

    border-top: none;

}

### member-edit.componenet.ts

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

import { Subscription, take } from 'rxjs';

import { ToastrService } from 'ngx-toastr';

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

import { MemberService } from '../../../core/services/member.service';

import { LoggedInUserDto } from '../../../core/models-interfaces/logged-in-user-dto.model';

import { UserDto } from '../../../core/models-interfaces/user-dto.model';

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

@Component({

  selector: 'app-member-edit',

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

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

})

export class MemberEditComponent implements OnInit, OnDestroy {

  //access the editForm

  @ViewChild('editForm') editForm: NgForm | undefined;

  //We have CanDeactiavte PreventUnsavedChanges guard. It will prevent the user from clicking away when have changes

  //However, the user can type a different url in the broweser

  @HostListener('window:beforeunload', ['$event']) unloadNotification($event: any){

    if(this.editForm?.dirty){

      $event.returnValue = true;

    }

  }

  member: UserDto | undefined;

  user: LoggedInUserDto | null = null;

  memberSubscription!: Subscription;

  memberUpdateSubscription!: Subscription;

  constructor(private accountService: AccountService,

              private memberService: MemberService,

              private toastr: ToastrService

  ){

    //we can also get the user from the local storage service as well. every thing is there

    //no need to unsubscribe due to take(1)

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

      next: (user: LoggedInUserDto | null) => {

        this.user = user;

      }

    })

  }

  ngOnDestroy(): void {

    if(this.memberSubscription) this.memberSubscription.unsubscribe();

    if(this.memberUpdateSubscription) this.memberUpdateSubscription.unsubscribe();

  }

  ngOnInit(): void {

    this.loadMember();

  }

  loadMember(){

    if(!this.user) return;

    this.memberSubscription = this.memberService.getMemberByGuid(this.user.guid).subscribe({

      next: (member: UserDto) => {

        this.member = member;

      }

    })

  }

  onUpdateMember()

  {

    if(!this.member){

      this.toastr.error("Member issue");

      return;

    }

    this.memberUpdateSubscription = this.memberService.updateMember(this.member).subscribe({

      next: \_ => {

        this.toastr.success("Profile updated successfully!");

        this.editForm?.reset(this.member);

      },

      error: e => {},

      complete: () => {}

    });

  }

}

### member-edit.componenet.html

<div class="row" \*ngIf="member">

    <div class="col-4">

        <h1>Your profile</h1>

    </div>

    <div class="col-8">

        <div class="alert alert-danger pb-0" \*ngIf="editForm.dirty">

            <p><strong>Information: </strong> You have made changes. Any unsaved changes will be lost</p>

        </div>

    </div>

    <div class="col-4">

        <!--left start-->

        <div class="card">

            <img src="{{member.photoUrl || './assets/user.png'}}" alt="{{member.displayName | titlecase}}"

                class="card-img-top img-thumbnail">

            <div class="card-body">

                <div>

                    <strong>Location:</strong>

                    <p>{{member.city}}, {{member.country}}</p>

                </div>

                <div>

                    <strong>Age:</strong>

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

                </div>

                <div>

                    <strong>Last active:</strong>

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

                </div>

                <div>

                    <strong>Member since:</strong>

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

                </div>

            </div>

            <div class="card-footer">

                <button class="btn btn-success col-12" form="editForm" type="submit" [disabled]="!editForm.dirty">Save changes</button>

            </div>

        </div>

         <!--left end-->

    </div>

    <div class="col-8">

        <!--right start-->

        <tabset class="member-tabset">

            <tab heading="About {{member.displayName | titlecase }}">

                <form #editForm="ngForm"  id="editForm" (ngSubmit)="onUpdateMember()">

                    <h4 class="mt-2">Description</h4>

                    <textarea class="form-control" [(ngModel)]="member.introduction" name="introduction" id="introduction" rows="6"></textarea>

                    <h4 class="mt-2">Looking for</h4>

                    <textarea class="form-control" [(ngModel)]="member.lookingFor" name="lookingFor" id="lookingFor" rows="6"></textarea>

                    <h4 class="mt-2">Interests</h4>

                    <textarea class="form-control" [(ngModel)]="member.interests" name="interests" id="interests" rows="6"></textarea>

                    <h4 class="mt-2">Location Details:</h4>

                    <div class="d-flex flex-row align-items-center">

                        <label for="city">City:</label>

                        <input type="text" [(ngModel)]="member.city" class="form-control mx-2" name="city" id="city">

                        <label for="country">Country:</label>

                        <input type="text" [(ngModel)]="member.country" class="form-control mx-2" name="country" id="country">

                    </div>

                </form>

            </tab>

            <tab heading="Edit Photos">

                <p>Photo edit will go here</p>

            </tab>

        </tabset>

        <!--right end-->

    </div>

</div>

# Adding Loading Indicator

## ngx-spinner

* <https://napster2210.github.io/ngx-spinner/>
* <https://github.com/Napster2210/ngx-spinner>

> npm install ngx-spinner

Or if an issues happens then install per the ng version of the app

>npm install ngx-spinner@16 --legacy-peer-deps

### Angular.json

And then add the style for the spinner you want to use to angular.json styles array

./node\_modules/ngx-spinner/animations/line-scale-party.css

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

Add it to the imports and exports array

NgxSpinnerModule.forRoot({type: 'line-scale-party'})

## environment.ts & environment.development.ts

Add a new property to put in delay to show the spinner.

Change to 0 to not show delay.

loadingSpinnerDelayMiliSec: 1000

## /core/services

### helper.service.ts

Create a property to return the spinner delay

  private loadingSpinnerDelayMiliSec: number = environment.loadingSpinnerDelayMiliSec;

  public LoadingSpinnerDelayMiliSec: number = this.loadingSpinnerDelayMiliSec;

### spinner-busy.ts

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

import { NgxSpinnerService } from 'ngx-spinner';

@Injectable({

  providedIn: 'root'

})

export class SpinnerBusyService {

  //as the request is taken place will increment the busyCount

  busyRequestCount = 0;

  constructor(private spinnerService: NgxSpinnerService) { }

  busy(){

    this.busyRequestCount++;

    this.spinnerService.show(undefined, {

      type: 'line-scale-party',

      bdColor: 'rgba(255, 255, 255, 0)',

      color: '#333333'

    });

  }

  idle(){

    this.busyRequestCount--;

    if(this.busyRequestCount <= 0){

      this.busyRequestCount = 0;

      this.spinnerService.hide();

    }

  }

}

## /core/inteceptors/loading-interceptor.ts

> ng g interceptor core/interceptors/loading --skip-tests

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

import {

  HttpRequest,

  HttpHandler,

  HttpEvent,

  HttpInterceptor

} from '@angular/common/http';

import { Observable, delay, finalize } from 'rxjs';

import { SpinnerBusyService } from '../services/spinner-busy.service';

import { HelperService } from '../services/helper.service';

@Injectable()

export class LoadingInterceptor implements HttpInterceptor {

  constructor(private spinnerBusyService: SpinnerBusyService,

              private helperService: HelperService

  ) {}

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

    this.spinnerBusyService.busy();

    //return next.handle(request);

    return next.handle(request).pipe(

      delay(this.helperService.LoadingSpinnerDelayMiliSec),

      finalize(() => {

        this.spinnerBusyService.idle();

      })

    );

  }

}

### app.module.ts

Add the interceptor to the providers array of the app module

    { provide: HTTP\_INTERCEPTORS, useClass: LoadingInterceptor, multi: true },

## app.componeent.html

Add the ngx spinner here

<ngx-spinner type="line-scale-party">

    <h3>Loading...</h3>

</ngx-spinner>

# Service To Store State

## /core/services/member.service.ts

Using the members array now

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

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

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

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

import { HelperService } from './helper.service';

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

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

import { UserDto } from '../models-interfaces/user-dto.model';

@Injectable({

  providedIn: 'root'

})

export class MemberService {

  //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

  members: UserDto[] = [];

  constructor(private localStorageService: LocalStorageService,

            private helperService: HelperService,

            private httpClientService: HttpClientService) {

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

    this.httpOptions = {

      headers: new HttpHeaders({

        Authorization: AppConstants.Bearer + this.localStorageService.getUserToken

      })

    };

  }

  /\*\*

   \* A GET users method

   \* @returns Observable userDto[]

   \*/

  getMembers(): Observable<UserDto[]>{

    //state

    if(this.members.length > 0){

      return of(this.members);

    }

    const url = this.helperService.urlUsersAll;

    this.helperService.logIfFrom(url, "memberService getMembers");

    //return this.httpClientService.get<UserDto[]>(url, this.httpOptions)

    return this.httpClientService.get<UserDto[]>(url, this.httpOptions) //jwt interceptor

    .pipe(

      map((members: UserDto[]) => {

        this.members = members;

        return members;

      })

    );

  }

  /\*\*

   \* A GET user by guid method

   \* @returns Observable userDto

   \*/

  getMemberByGuid(guid: string): Observable<UserDto>{

    //state

    if(this.members.length > 0){

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

      if(member){

        return of(member);

      }

    }

    const url = this.helperService.replaceKeyValue(this.helperService.urlUserGetByGuid, this.helperService.keyGuid, guid, "");

    this.helperService.logIfFrom(url, "memberService getMemberByGuid");

    //return this.httpClientService.get<UserDto>(url, this.httpOptions)

    return this.httpClientService.get<UserDto>(url, this.httpOptions) //jwt interceptor

      .pipe(

        map((member: UserDto) => {

          return member;

        })

      );

  }

  updateMember(user: UserDto){

    const url = this.helperService.urlUserUpdate;

    this.helperService.logIfFrom(url, "memberService updateMember");

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

      map(() => {

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

        //this.members[index] = user;

        this.members[index] = {...this.members[index], ...user};

      })

    );

  }

}

## /site/members/member-list

Using async pipe

### member-list.component.ts

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

import { Observable, Subscription } from 'rxjs';

import { UserDto } from 'src/app/core/models-interfaces/user-dto.model';

import { MemberService } from 'src/app/core/services/member.service';

@Component({

  selector: 'app-member-list',

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

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

})

export class MemberListComponent implements OnInit, OnDestroy {

  members: UserDto[] = [];

  membersAsync$: Observable<UserDto[]> | undefined;

  membersSubscription!: Subscription;

  constructor(private memberService: MemberService){}

  ngOnDestroy(): void {

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

  }

  ngOnInit(): void {

    //not using load members any more but using membersAsync$

    //this.loadMembers();

    this.membersAsync$ = this.memberService.getMembers();

  }

  loadMembers(){

    this.membersSubscription = this.memberService.getMembers().subscribe({

      next: (members: UserDto[]) => {

        this.members = members;

      },

      error: e => {},

      complete: () => {}

    })

  }

}

### member-list.component.html

<div class="row">

    <!--

    //using the async

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

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

    </div>

    -->

    <div class="col-2" \*ngFor="let member of membersAsync$ | async">

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

    </div>

</div>