Table of Contents

[Basics 2](#_Toc119191853)

[Current Resources with Changes 2](#_Toc119191854)

[New Resources Added 2](#_Toc119191855)

[npm install / ng serve 2](#_Toc119191856)

[ngx Modals / Confirm 2](#_Toc119191857)

[/core/services 2](#_Toc119191858)

[ConfirmService 2](#_Toc119191859)

[/core/guards 3](#_Toc119191860)

[prevent-unsaved-changes.guard.ts 3](#_Toc119191861)

[Components 4](#_Toc119191862)

[/site/admin/user-management 4](#_Toc119191863)

[user-management.component.ts 4](#_Toc119191864)

[/site/modals/confirm-modal 4](#_Toc119191865)

[confirm-modal.component.ts 4](#_Toc119191866)

[confirm-modal.component.html 4](#_Toc119191867)

[/site/messages 4](#_Toc119191868)

[messages.component.ts 4](#_Toc119191869)

[/site/members/member-messages 5](#_Toc119191870)

[member-messages.component.ts 5](#_Toc119191871)

[member-messages.component.html 5](#_Toc119191872)

# Basics

|  |  |
| --- | --- |
| WorkingFolder | Copy the content of “Site-06-SignalR” in “Site-07-UnitOfWork FinishingTouces” and issue   * npm install * ng serve |

# Current Resources with Changes

1. /site/admin/user-management
2. /site/modals/confirm-modal
3. /core/guards/prevent-unsaved-changes.guard
4. /site/messages
5. /site/members/member-messages

# New Resources Added

1. ngx modals/confirm
2. /core/services/confirm.service

# npm install / ng serve

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

# ngx Modals / Confirm

Take a look at the following for more details

<https://valor-software.com/ngx-bootstrap/#/components/modals?tab=overview#confirm-window>

# /core/services

## ConfirmService

* ng g s /core/services/confirm --skip-tests
* BsModalRef has been used in site/admin/user-management/user-management.component.ts. We’ll be using the same technique here to create a confirm modal.
* This is using confirmModal, check [components](#_/site/modals/confirm-modal) below.

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

import { Observable } from 'rxjs';

import { BsModalRef, BsModalService } from 'ngx-bootstrap/modal';

import { ConfirmModalComponent } from '../../site/modals/confirm-modal/confirm-modal.component';

//also check site/admin/user-management/user-management.component.ts for another modal use

@Injectable({

  providedIn: 'root'

})

export class ConfirmService {

  private bsModalRef!: BsModalRef;

  private defaultTitle: string = 'Confirmation';

  private defaultMessage: string = 'Are you sure you want to do this?';

  private defaultButtonOkText: string = 'Ok';

  private defaultButtonCancelText: string = 'Cancel';

  constructor(private modalService: BsModalService) { }

  confirm(title = this.defaultTitle,

                  message = this.defaultMessage,

                  btnOkText = this.defaultButtonOkText,

                  btnCancelText = this.defaultButtonCancelText) : Observable<boolean> {

    const config = {

      initialState: { title, message, btnOkText, btnCancelText }

    };

    this.bsModalRef = this.modalService.show(ConfirmModalComponent, config);

    return new Observable<boolean>(this.getResult());

  }

  private getResult() {

    return (observer: any) => {

      const subscription = this.bsModalRef.onHidden?.subscribe(() => {

        observer.next(this.bsModalRef.content?.result);

        observer.complete();

      });

      return {

        unsubscribe() {

          subscription?.unsubscribe();

        }

      };

    }

  }

}

# /core/guards

## prevent-unsaved-changes.guard.ts

* Update the guard canDeactivate method to to use the confirm service
* Keep note that this guard is applied in app-routing.module.ts, it will be automatically subscribed to now

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

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

import { Observable } from 'rxjs';

import { ConfirmService } from '../services/confirm.service';

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

@Injectable({

  providedIn: 'root'

})

export class PreventUnsavedChangesGuard implements CanDeactivate<unknown> {

  constructor(private confirmService: ConfirmService){}

  /\*

  canDeactivate(component: MemberEditComponent): boolean {

    if (component.editForm.dirty)

    {

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

    }

    return true;

  }

  \*/

  canDeactivate(component: MemberEditComponent): Observable<boolean> | boolean {

    if (component.editForm.dirty)

    {

      return this.confirmService.confirm();

    }

    return true;

  }

}

# Components

## /site/admin/user-management

### user-management.component.ts

Only added the comment to take a look at the ConfirmService for another use of the modal

## /site/modals/confirm-modal

* ng g c /site/modals/confirmModal --module=/core/modules/shared.module

This will also add the component to the /modules/shared.module

### confirm-modal.component.ts

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

import { BsModalRef } from 'ngx-bootstrap/modal';

@Component({

  selector: 'app-comfirm-modal',

  templateUrl: './confirm-modal.component.html',

  styleUrls: ['./confirm-modal.component.css']

})

export class ConfirmModalComponent implements OnInit {

  title!: string;

  message!: string;

  btnOkText!: string;

  btnCancelText!: string;

  result!: boolean;

  constructor(public bsModalRef: BsModalRef) { }

  ngOnInit(): void {

  }

  confirm() {

    this.result = true;

    this.bsModalRef.hide();

  }

  decline() {

    this.result = false;

    this.bsModalRef.hide();

  }

### confirm-modal.component.html

<div class="modal-header">

  <h4 class="modal-title pull-left">{{title}}</h4>

</div>

<div class="modal-body">

  <p>{{message}}</p>

</div>

<div class="modal-footer">

  <button type="button" class="btn btn-success" (click)="confirm()">{{btnOkText}}</button>

  <button type="button" class="btn btn-danger" (click)="decline()">{{btnCancelText}}</button>

</div>

## /site/messages

### messages.component.ts

* Add the confirmation to delete.
* Add the confirm message service first
* Rename the onDeleteMessage to performDelete and also mark it as private.
* Write a new onDeleteMessage method to confirm and when true only then delete.

  onDeleteMessage(msgId: number) {

    //call the confirmService and overwrite the defaults

    const title = 'Confirm Delete';

    const message = 'Are you sure to delete the message? This cannot be undone.';

    const btnDelete = 'Delete';

    this.confirmService.confirm(title, message, btnDelete).subscribe({

      next: result => {

        //only delete when true is returned

        if (result) this.performDelete(msgId);

      },

      error: e => { },

      complete: () => {}

    })

  }

  private performDelete(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: () => { }

    });

  }

## /site/members/member-messages

Add the scrolly to the messages section

### member-messages.component.ts

Add the change detention strategy as OnPush

@Component({

  changeDetection: ChangeDetectionStrategy.OnPush,

  selector: 'app-member-messages',

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

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

})

### member-messages.component.html

Update ul tag to add style scrolling

        <ul #scrollMe

            style="overflow: scroll; height: 500px"

            [scrollTop]="scrollMe.scrollHeight"

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