Table of Contents

[Basics 3](#_Toc117549980)

[Current Resources with Changes 3](#_Toc117549981)

[New Resources Added 3](#_Toc117549982)

[npm install / ng serve 3](#_Toc117549983)

[Modal Module 3](#_Toc117549984)

[/core/enums 3](#_Toc117549985)

[zEnumType 3](#_Toc117549986)

[zRoles.ts 4](#_Toc117549987)

[/core/models 4](#_Toc117549988)

[userTokenDto.model.ts 4](#_Toc117549989)

[userDto.model.ts 4](#_Toc117549990)

[/core/services 4](#_Toc117549991)

[enum.service.ts 4](#_Toc117549992)

[api-url.service.ts 5](#_Toc117549993)

[local-storage.service.ts 5](#_Toc117549994)

[account.service.ts 5](#_Toc117549995)

[admin.service.ts 6](#_Toc117549996)

[/core/guards 7](#_Toc117549997)

[admin.guard.ts 7](#_Toc117549998)

[/core/directives 8](#_Toc117549999)

[has-role.directive.ts 8](#_Toc117550000)

[Components 8](#_Toc117550001)

[/site/nav/nav.component 8](#_Toc117550002)

[nav.component.ts 8](#_Toc117550003)

[nav.component.html 9](#_Toc117550004)

[/site/admin/admin-panel 9](#_Toc117550005)

[admin-panel.component.ts 9](#_Toc117550006)

[admin-panel.component.html 9](#_Toc117550007)

[/site/modals/roles-modal 9](#_Toc117550008)

[roles-modal.compoent.ts 9](#_Toc117550009)

[roles-modal.compoent.html 10](#_Toc117550010)

[/site/admin/user-management.component 10](#_Toc117550011)

[user-management.component.ts 10](#_Toc117550012)

[user.management.component.html 12](#_Toc117550013)

[/site/admin/photo-management.component 12](#_Toc117550014)

[photo-management.component.ts 12](#_Toc117550015)

[photo-managemnt.component.html 12](#_Toc117550016)

[app-routing.module.ts 13](#_Toc117550017)

# 

# Basics

|  |  |
| --- | --- |
| WorkingFolder | Copy the content of “Site-04-Message Feature” in “Site-05-Identity Role Management” and issue   * npm install * ng serve |

# Current Resources with Changes

1. /core/services/api-url.service
2. app-routing.module.ts
3. /site/nav/nav.component
4. /core/models/userTokenDto.model
5. /core/models/userDto.model

# New Resources Added

1. /core/enums/zEnumType
2. /core/enums/zRoles
3. /core/services/enum.service
4. /site/admin/admin-panel
5. /core/guards/admin.guard
6. /core/directive/has-role.directive
7. /site/admin/user-management
8. /site/admin/photo-management
9. /core/services/admin.service

# npm install / ng serve

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

# Modal Module

Add the ngx bootstrap modal module to shared module

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

and then add to the imports array

ModalModule.forRoot(),

And the exprots array

ModalModule,

# /core/enums

## zEnumType

enums are either standard number type or string type. This is used when using the enumService generic enum methods

export enum zEnumType{

  number,

  string

}

## zRoles.ts

export enum zRoles{

  Member = "Member",

  Admin = "Admin",

  Moderator = "Moderator"

}

# /core/models

## userTokenDto.model.ts

Add string array for getting the roles.

export class UserTokenDto {

    constructor(public userName: string = "",

        public guId: string = "",

        public token: string = "",

        public mainPhotoUrl = "",

        public displayName = "",

        public gender = "",

        public roles: string[]

        ) { }

}

## userDto.model.ts

Add roles to it

import { PhotoDto } from "./photoDto.model";

export class userDto {

    constructor(public id: number, public guId: string, public userName: string, public photoUrl: string, public age: number,

                public displayName: string, public gender: string, public introduction: string,

                public lookingFor: string, public interests: string, public city: string, public country: string,

                public photos: PhotoDto[],

                public lastActive: Date, public createdOn: Date, public updatedOn: Date,

                public roles: string[] = []

    ) { }

}

# /core/services

## enum.service.ts

This is a helper service to work with the enums. Check the comments for use

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

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

@Injectable({

  providedIn: 'root'

})

export class EnumService {

  constructor() { }

    /\*\*

   \* Use for the string enum

   \* @T : enum

   \* @param obj: pass the enum <T>

   \* @param enumType: is this number (default) or string enum

   \* @returns key value pair array

    To use

    let keys = this.enumService.enumToKeyValuePairs<zMemberGetBy>(zMemberGetBy, zEnumType.number);

    <select placeholder="mode" [(ngModel)]="mode" name="mode">

      <option \*ngFor="let c of keys" [value]="c.value">{{ c.name }}</option>

    </select>

  \*/

    enumToKeyValuePairs<T>(obj: any, enumType: zEnumType = zEnumType.number): { name: string, value: string}[] {

      if(enumType === zEnumType.string)

        return Object.keys(obj as keyof T).map(key => ({ name: key, value: obj[key] }));

      return Object.keys(obj as keyof T).filter((v) => isNaN(Number(v))).map(key => ({ name: key, value: obj[key] }));

    }

    /\*\*

     \* get the string description for enum

     \* @T : enum

     \* @param obj: pass the enum <T>

     \* @param enumVal: the enaum value

     \* @returns enum string description or undefined

      To use

      let key = this.enumService.enumToString<zRoles>(zRoles, zRoles.Moderator)

    \*/

    enumToString<T>(obj: any, enumVal: T) : string | undefined {

      for (var item in obj) {

        if (obj[item] === enumVal) return item;

      }

      return undefined;

    }

    /\*\*

     \* get the rnum from string description

     \* @T : enum

     \* @param obj: pass the enum <T>

     \* @param val: the description

     \* @returns enum or undefined

      To use

      let key = this.enumService.stringToEnum<zRoles>(zRoles, "Moderator")

    \*/

    stringToEnum<T>(obj: any, val: string,): T | undefined{

      for (var item in obj) {

        if (item === val) return obj[item];

      }

      return undefined;

    }

}

## api-url.service.ts

Add the admin controller actions url

  //admin controller urls

  private adminBaseUrl = `${this.apiBaseUrl}admin/`;

  adminRolesGuidReplace = "[guid]";

  adminUsersWithroles = `${this.adminBaseUrl}users-with-roles`;

  adminEditRoles = `${this.adminBaseUrl}edit-roles/${this.adminRolesGuidReplace}`; //replace [guid] with user guids and pass in the new roles as params

## local-storage.service.ts

Account service below will add the roles to user and then will write to local storage. Add a new property to get the roles as well now

  getLoggedinUserRoles: string[] = this.getLoggedInUser?.roles;

## account.service.ts

Add a method to decode the token.

|  |  |
| --- | --- |
| Token when logged in: | Go to <https://jwt.io/> and take a look at the token |

Create a method to decode middle part of the token

  getDecodedToken(token: string) : string[] {

    //token is not encypted, the signature is.

    //get the user roles from the token

    //The atob() method decodes a string that has been encoded by the btoa() method

    //token comes in three parts seperated by the . It is Header, Payload and signature

    //interested in the middle part

    var parsedToken = token.split(".")[1];

    var decoded = JSON.parse(atob(parsedToken));

    return decoded;

  }

After that go to setAndFireCurrentUser and call decode on token and assign role to user.roles.

  setAndFireCurrentUser(user: UserTokenDto) {

    //decode token and add roles to the user. Keep in mind some users may only have single role so that is not a string[]

    user.roles = [];

    const roles = this.getDecodedToken(user.token)?.role;

    if (roles) {

      Array.isArray(roles) ? user.roles = roles : user.roles.push(roles);

    }

    //store the user in local storage

    this.localStorageService.setItem(this.localStorageService.\_keyUser, user);

    this.fireCurrentUser(user);

  }

|  |  |
| --- | --- |
| Go to local storage and look at the user saved |  |

## admin.service.ts

* ng g s /core/services/admin --skip-tests

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

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

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

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

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

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

@Injectable({

  providedIn: 'root'

})

export class AdminService {

  constructor(private apiUrlService: ApiUrlService, private httpClientService: HttpClientService) { }

  /\*

  this pass back an array object of followng type

  {"id":12,"userName":"bobmed","displayName":"Bobmed","guId":"cfad1be7-d0ad-4137-b95b-5d4a5b92991a","roles":["Member"]}

  \*/

  getUsersWithRoles() {

    var url = this.apiUrlService.adminUsersWithroles;

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

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

  }

  updateUserRoles(userGuid: string, roles: string[]) {

    var url = this.apiUrlService.adminEditRoles.replace(this.apiUrlService.adminRolesGuidReplace, userGuid);

    if (roles && roles.length > 0) url += "?roles=" + roles;

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

    return this.httpClientService.post<string[]>(url, {});

  }

}

# /core/guards

## admin.guard.ts

Create a new CanActivate guard

* ng g guard /core/guards/admin --skip-tests

? Which interfaces would you like to implement? CanActivate

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

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

import { map, Observable } from 'rxjs';

import { ToastrService } from 'ngx-toastr';

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

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

@Injectable({

  providedIn: 'root'

})

export class AdminGuard implements CanActivate {

  constructor(private accountService: AccountService, private toastrService: ToastrService) { }

  canActivate(): Observable<boolean> {

    return this.accountService.currentUser$.pipe(

      map(user => {

        if (user.roles && (user.roles.includes(zRoles.Admin) || user.roles.includes(zRoles.Moderator)))

          return true;

        this.toastrService.error('You cannot enter this area', 'Restricted Area');

        return false;

      })

    );

  }

}

# /core/directives

## has-role.directive.ts

* ng g directive /core/directives/hasRole --skip-tests

Make sure to place the **directive in declarations and export arrays of the shared.module.ts** and not app.module.ts.

import { Directive, Input, OnInit, TemplateRef, ViewContainerRef } from '@angular/core';

import { take } from 'rxjs';

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

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

@Directive({

  selector: '[appHasRole]'

})

export class HasRoleDirective implements OnInit {

  @Input() appHasRole: string[] = [];

  user: UserTokenDto = <UserTokenDto>{};

  constructor(private viewContaineRef: ViewContainerRef,

    private templateRef: TemplateRef<any>,

    private accountService: AccountService) {

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

      next: (user: UserTokenDto) => {

        this.user = user;

      },

      error: e => { },

      complete: () => { }

    });

  }

  ngOnInit(): void {

    //clear the view if no roles

    if (!this.user?.roles || this.user == null) {

      this.viewContaineRef.clear();

      return;

    }

    //if the user has roles and are in passed then keep it other wise clear it

    if (this.user?.roles.some(r => this.appHasRole.includes(r))) {

      this.viewContaineRef.createEmbeddedView(this.templateRef);

    }

    else {

      this.viewContaineRef.clear();

    }

  }

}

# Components

## /site/nav/nav.component

### nav.component.ts

  //to be used for the appHasRole directive with Admin link

  zRoles = zRoles;

Add the property zRoles for enum zRoles so that it could be used in html

### nav.component.html

Add the admin link under the messages link and apply the custom appHasRole directive to it. Admin link is only going to display for Admin and Moderator roles.

<li \*appHasRole="[zRoles.Admin,zRoles.Moderator]" class="nav-item">

   <a class="nav-link" routerLink="/admin" routerLinkActive="active" (click)="onNavBarItemClickCloseNavBar()">Admin</a>

</li>

## /site/admin/admin-panel

Create new admin panel component in folder/site/admin

* ng g c /site/admin/adminPanel --module=/core/modules/shared.module.ts --skip-tests

Also add it the component to the shared module

### admin-panel.component.ts

Add the zRoles as property so that we can use it inside the html

  zRoles = zRoles;

### admin-panel.component.html

<h2>Admin Panel</h2>

<div class="tab-panel">

  <tabset class="member-tabset">

    <tab heading="User Management" \*appHasRole="[zRoles.Admin]">

      <!--tab will only display for the Admin roles-->

      <div class="container">

        <app-user-management></app-user-management>

      </div>

    </tab>

    <tab heading="Photo Management" \*appHasRole="[zRoles.Admin, zRoles.Moderator]">

      <!--tab will only display for the Admin roles-->

      <div class="container">

        <app-photo-management></app-photo-management>

      </div>

    </tab>

  </tabset>

</div>

## /site/modals/roles-modal

* ng g c /site/modals/roles-modal --skip-tests

Check following to use the modals in other components

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

### roles-modal.compoent.ts

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

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

import { zRoles } from 'src/app/core/enums/zRoles';

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

@Component({

  selector: 'app-roles-modal',

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

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

})

export class RolesModalComponent implements OnInit {

  /\*

  following are the default, will over write for our use case. For the default use see: https://valor-software.com/ngx-bootstrap/#/components/modals?tab=overview

  title: string = "";

  list: any[] = [];

  closeBtnName: string = "";

  \*/

  @Input() updateSelectedRoles = new EventEmitter();

  user: userDto = <userDto>{};

  roles: any[] = [];

  zRoles = zRoles;

  constructor(public bsModalRef: BsModalRef) { }

  ngOnInit(): void {

  }

  onUpdateSelectedRoles() {

    this.updateSelectedRoles.emit(this.roles);

    this.bsModalRef.hide();

  }

}

### roles-modal.compoent.html

<div class="modal-header">

  <h4 class="modal-title pull-left">Edit roles for {{user?.displayName}}</h4>

  <button type="button" class="btn-close close pull-right" aria-label="Close" (click)="bsModalRef.hide()">

    <span aria-hidden="true" class="visually-hidden">&times;</span>

  </button>

</div>

<div class="modal-body">

  <form #rolesForm="ngForm" id="rolesForm">

    <!--make sure that the admin can't change the admin role so disble it-->

    <div class="form-check" \*ngFor="let role of roles">

      <input type="checkbox"

        class="form-check-input"

        [checked]="role.checked"

        value="role.name"

        (change)="role.checked = !role.checked"

        [disabled]="role.name === zRoles.Admin && user?.userName === 'admin'"

      >

      <label>{{ role.name }}</label>

    </div>

  </form>

</div>

<div class="modal-footer">

  <button type="button" class="btn btn-danger" (click)="bsModalRef.hide()">Cancel</button>

  <button type="button" class="btn btn-success" (click)="onUpdateSelectedRoles()">Save</button>

</div>

## /site/admin/user-management.component

* ng g c /site/admin/userManagement --module=/core/modules/shared.module.ts --skip-tests

It will get added to the shared module declarations array as well.

### user-management.component.ts

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

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

import { ToastrService } from 'ngx-toastr';

import { Subscription, take } from 'rxjs';

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

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

import { AdminService } from '../../../core/services/admin.service';

import { RolesModalComponent } from '../../modals/roles-modal/roles-modal.component';

@Component({

  selector: 'app-user-management',

  templateUrl: './user-management.component.html',

  styleUrls: ['./user-management.component.css']

})

export class UserManagementComponent implements OnInit, OnDestroy {

  users?: Partial<userDto[]>;

  usersSubscription?: Subscription;

  //modal reference

  bsModalRef?: BsModalRef;

  constructor(private adminService: AdminService, private modalService: BsModalService, private toastrService: ToastrService) { }

  ngOnDestroy(): void {

    if (this.usersSubscription) this.usersSubscription.unsubscribe();

  }

  ngOnInit(): void {

    this.loadUsersWithRoles();

  }

  loadUsersWithRoles() {

    this.usersSubscription = this.adminService.getUsersWithRoles().subscribe({

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

        this.users = users

      }

    });

  }

  openRolesModal(user?: userDto) {

    //default is overwritten check followig for the defaults

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

    const config = {

      class: 'modal-disalog-center',

      initialState: {

        user: user,

        roles: this.getRolesArray(user)

      }

    };

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

    this.bsModalRef.content.updateSelectedRoles.subscribe((values: any[]) => {

      //get the roles that are checked on modal

      const rolesToUpdate = {

        roles: [...values.filter(el => el.checked === true).map(el => el.name)]

      };

      //make sure that user has some roles selcted

      if (rolesToUpdate && rolesToUpdate.roles) {

        //update the user roles in DB

        this.adminService.updateUserRoles(user!.guId, rolesToUpdate.roles).pipe(take(1)).subscribe({

          next: () => {

            //update the user roles

            user!.roles = [...rolesToUpdate.roles];

            this.toastrService.success('Roles updates successfully!', 'Success');

          }

        });

      }

      else {

        this.toastrService.show('No roles selected!', 'confirmation');

      }

    });

  }

  private getRolesArray(user?: userDto) : any[] {

    const roles: any[] = [];

    const userRoles = user?.roles;

    const availableRoles: any[] = [

      { name: zRoles.Admin, value: zRoles.Admin },

      { name: zRoles.Moderator, value: zRoles.Moderator },

      { name: zRoles.Member, value: zRoles.Member },

    ];

    availableRoles.forEach(role => {

      let isMatch = false;

      for (const userRole of userRoles!) {

        if (role.name === userRole) {

          //role found for user

          isMatch = true;

          role.checked = true;

          roles.push(role);

          break;

        }

      }

      if (!isMatch) {

        //role not found for user

        role.checked = false;

        roles.push(role);

      }

    });

    return roles;

  }

}

### user.management.component.html

<ng-container \*ngIf="users">

  <div class="row">

    <table class="table">

      <tr>

        <th style="width: 30%;">User</th>

        <th style="width: 40%;">Active Roles</th>

        <th style="width: 30%;"></th>

      </tr>

      <tr \*ngFor="let user of users">

        <td>{{ user?.displayName }}</td>

        <td>{{ user?.roles?.join(", ") }}</td>

        <td><button class="btn btn-info" (click)="openRolesModal(user)">Edit Roles</button></td>

      </tr>

    </table>

  </div>

</ng-container>

## /site/admin/photo-management.component

* ng g c /site/admin/photoManagement --module=/core/modules/shared.module.ts --skip-tests

It will get added to the shared module declarations array as well

### photo-management.component.ts

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

@Component({

  selector: 'app-photo-management',

  templateUrl: './photo-management.component.html',

  styleUrls: ['./photo-management.component.css']

})

export class PhotoManagementComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

}

### photo-managemnt.component.html

<p>photo-management works!</p>

# app-routing.module.ts

Add the admin route to under the lists route. User must be signed in to get to this component.

  {

    path: '',

    runGuardsAndResolvers: 'always',

    canActivate: [AuthGuard],

    children: [

      { path: 'members/list', component: MemberListComponent },

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

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

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

      { path: 'lists', component: ListsComponent },

      { path: 'messages', component: MessagesComponent },

      { path: 'admin', component: AdminPanelComponent, canActivate: [AdminGuard] },

    ]

  }