# Project

* Site-15-Api-Ng-identity-role-management
  + Copied from: Site-14-Api-Ng-messaging-queryparams-routeresolvers
* For the “MySocialConnect-SPA” issue npm install command to install all packages
  + ng serve : to run the spa

# New & Updates

|  |  |
| --- | --- |
| New | Updates |
| /site/admin/admin-panel | /core/services/helper.service.ts |
| /core/enums/z-roles.ts | app-routing.module.ts |
| /core/guards/admin.guard.ts | /core/models/interfaces/logged-in-user-dto.model.ts |
| /core/directives/in-in-role.directive.ts | /core/services/account.service.ts |
| /core/services/admin.service.ts |  |
| /site/admin/user-management |  |
| /site/admin/photo-management |  |
| /core/models-interfaces/site-role.ts |  |
|  |  |
|  |  |

# Clear Database

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

dotnet ef database drop

dotnet ef database update

# /core/enums

## z-roles.ts

export enum ZRoles {

    Member = "Member",

    Admin = "Admin",

    Moderator = "Moderator"

}

# /core/models-interfaces

## logged-in-user-dto.model.ts

Add new property roles to it

export interface LoggedInUserDto {

    id: number;

    userName: string;

    guid: string;

    token: string;

    displayName: string;

    mainPhotoUrl: string;

    gender: string;

    roles: string[]

}

## user-dto.model.ts

Add a new property roles: string[] to it as well. This will be filled only when getting data in admincontroller when getting the roles

    roles: string[]; //adminService.getUsersWithRoles

## site-role.model.ts

> ng g class core/models-interfaces/siteRole --type=model --skip-tests

//filed by adminService.getSiteRoles

//used by the user-managment

export class SiteRole {

    constructor(public name: string, public value: string){}

}

# /core/services

## helper.service.ts

Add admin api links

  //admin

  private urlAdmin: string = `${this.baseUrlServer}admin`;

  public urlAdminGetUsersWithRoles: string = `${this.urlAdmin}/users-with-roles`;

  public urlAdminGetPhotosToModerate: string = `${this.urlAdmin}/photos-to-moderate`;

  public urlAdminEditRoles: string = `${this.urlAdmin}/edit-roles`;

## account.service.ts

Authen token has role clain in it as well. Will need to decode to get the roles out of it. Role claim could be a string or string[].

|  |  |
| --- | --- |
| {  "nameid": "12",  "unique\_name": "kevin",  "Guid": "0e66ed5b-ea78-4f3b-9e73-603c61ffa7da",  "DisplayName": "Kevin",  "role": "Member",  "nbf": 1715813975,  "exp": 1716418775,  "iat": 1715813975  } | {  "nameid": "11",  "unique\_name": "admin",  "Guid": "4ce8e019-d3d1-4744-8e49-9e97737867b7",  "DisplayName": "Admin",  "role": [  "Admin",  "Moderator"  ],  "nbf": 1715661515,  "exp": 1716266315,  "iat": 1715661515  } |

### getDecodedToken Method

Write a method to decode the token

  getDecodedToken(token: string) :any {

    //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;

  }

### setAndFireCurrentUser Method

Write roles to the user object being persisted in local storage

  setAndFireCurrentUser(user: LoggedInUserDto){

    //pick roles from the token, for this need to decode the token.

    //keep in mind users may only have single role as string or a string array of roles

    user.roles = [];

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

    if(roles){

      if(Array.isArray(roles))

        user.roles = roles;

      else

        user.roles.push(roles);

    }

    this.localStorageService.setUser(user);

    this.fireCurrentUser(user);

  }

### Persisted User Info in Local Storage

Login again to see the updated info in local storage

{

"id": 12,

"userName": "kevin",

"guid": "0e66ed5b-ea78-4f3b-9e73-603c61ffa7da",

"token": "eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJuYW1laWQiOiIxMiIsInVuaXF1ZV9uYW1lIjoia2V2aW4iLCJHdWlkIjoiMGU2NmVkNWItZWE3OC00ZjNiLTllNzMtNjAzYzYxZmZhN2RhIiwiRGlzcGxheU5hbWUiOiJLZXZpbiIsInJvbGUiOiJNZW1iZXIiLCJuYmYiOjE3MTU4MTczMTUsImV4cCI6MTcxNjQyMjExNSwiaWF0IjoxNzE1ODE3MzE1fQ.tcHeVmtmxqQOr5GrnMcXKCqi3CLDnwoEfiNLXwCkY5SHBJnu4baA\_Nl0mzkTpvxZ9rVAQtIPX2hToHpxW0DbUw",

"displayName": "Kevin",

"mainPhotoUrl": "",

"gender": "male",

"roles": [

"Member"

]

}

## admin.service.ts

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

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

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

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

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

import { SiteRole } from '../models-interfaces/site-role.model';

import { ZRoles } from '../enums/z-roles';

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

@Injectable({

  providedIn: 'root'

})

export class AdminService {

  private \_siteRoles: SiteRole[] = [];

  constructor(private helperService: HelperService, private httpClientService: HttpClientService) { }

  //build static list of site roles

  getSiteRoles(): SiteRole[]{

    if(this.\_siteRoles && this.\_siteRoles.length > 0)

      return this.\_siteRoles;

    this.\_siteRoles = [

      new SiteRole(ZRoles.Admin, ZRoles.Admin, false),

      new SiteRole(ZRoles.Moderator, ZRoles.Moderator, false),

      new SiteRole(ZRoles.Member, ZRoles.Member, false)

    ];

    return this.\_siteRoles;

  }

  //mark site roles as selected or not selected per the user roles

  translateUserRolesToSiteRoles(userRoles: string[]) : SiteRole[]{

    const roles: SiteRole[] = [];

    const siteRoles: SiteRole[] = [...this.getSiteRoles()];

    siteRoles.forEach(role => {

      let isSelected = false;

      if(userRoles && userRoles.length > 0)

        isSelected = userRoles.includes(role.name);

      role.isSelected = isSelected;

      roles.push(role);

    })

    return roles;

  }

  /\*

  Pass back an array of object so you Partial<UserDto>

  [{"id": 11,"userName": "admin","displayName": "Admin","guId": "4ce8e019-d3d1-4744-8e49-9e97737867b7","roles": ["Admin","Moderator"]}]

  \*/

  getUsersWithRoles()

  {

    const url = this.helperService.urlAdminGetUsersWithRoles;

    this.helperService.logIfFrom(url, "AdminService getUsersWithRoles");

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

  }

  //passing query string / query parameters with post

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

    let url = this.helperService.replaceKeyValue(this.helperService.urlAdminEditRoles, this.helperService.keyGuid, userGuid);

    this.helperService.logIfFrom(url, "AdminService updateUserRoles")

    this.helperService.logIf(roles);

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

    const paramsSet = new HttpParams().set("roles", roles.join(","));

    let httpOptions = { params: paramsSet };

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

  }

}

# /core/guards

## admin.guard.ts

> ng g guard core/guards/admin

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

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

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

import { map } from 'rxjs';

import { ToastrService } from 'ngx-toastr';

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

import { ZRoles } from '../enums/z-roles';

export const adminGuard: CanActivateFn = (route, state) => {

  const accountService = inject(AccountService);

  const toastr = inject(ToastrService);

  //account service is firing the logged in user

  //inside the component/interceptor we subscribe

  //inside guards

  return accountService.currentLoggedInUser$.pipe(

    map(user => {

      if(!user) return false;

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

        return true;

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

      return false;

    })

  );

  return true;

};

# /core/directives

## is-in-role.directive.ts

> ng g d core/directives/isInRole --skip-tests

* Make sure that it is added to the shared module or app module
  + Add it to shared module both declarations and export
  + Since the nav is declared in app module
* This will be used as a structural directive which will be removing items from the DOM

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

import { take } from 'rxjs';

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

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

/\*

this needs to be structural directive

\*appHasRole='["Admin", "Moderator"]'

\*/

@Directive({

  selector: '[appIsInRole]'

})

export class IsInRoleDirective implements OnInit {

  @Input() appIsInRole: string[] = [];

  user: LoggedInUserDto = <LoggedInUserDto>{}; //or {} as LoggedInUserDto

  constructor(private viewContainerRef: ViewContainerRef,

              private templateRef: TemplateRef<any>,

              private accountService: AccountService) {

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

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

        if(user)

          this.user = user;

      }

    });

  }

  //remove the item from the DOM

  ngOnInit(): void {

    //when no user or no roles for user then clear

    //when no roles passed in then clear

    if(!this.user || !this.user?.roles || !this.appIsInRole || this.appIsInRole.length <= 0){

      this.viewContainerRef.clear();

      return;

    }

    //if the user has roles and passed ones exist in the roles then keep the item

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

      this.viewContainerRef.createEmbeddedView(this.templateRef);

      return;

    }

    //clear

    this.viewContainerRef.clear();

  }

}

# /site/admin

## admin-panel

Create new component

> ng g c site/admin/adminPanel --skip-tests --module=core/modules/shared

The declarations will get added to the shared module as well

### admin-panel.component.ts

Add ZRoles enum as property to be used inside the html

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

import { ZRoles } from 'src/app/core/enums/z-roles';

@Component({

  selector: 'app-admin-panel',

  templateUrl: './admin-panel.component.html',

  styleUrls: ['./admin-panel.component.css']

})

export class AdminPanelComponent {

  zRole = ZRoles;

}

### admin-panel.component.html

Add the tab set with two tabs to call below created two components, [user management](#_user-management) and [photo management](#_photo-management)

<h2>Admin Panel</h2>

<div class="tab-panel">

    <tabset class="member-tabset">

        <tab heading="User Management" \*appIsInRole="[zRole.Admin]">

            <div class="container">

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

            </div>

        </tab>

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

            <div class="container">

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

            </div>

        </tab>

    </tabset>

</div>

### app-routing.module.ts

* Add the route to the admin panel
* Apply adminGuard to it as canActivate

  //dummy route to group secure resources together

  {

    path: '',

    runGuardsAndResolvers: 'always',

    canActivate: [authGuard],

    children: [

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

      //resolver used, property member will get the data so use 'member' in member-detail page

      { 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] },

    ]

  },

### /site/nav

#### nav.component.ts

Create a variable for ZRole enum to be used inside the html

  //to be used with appIsInRole directive on admin link. The link will be hidden

  zRole = ZRoles;

#### nav.component.html

* Add the link to Admin under the messages
* Apply the appIsInRole guar to it as well

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

            <li class="nav-item"><a class="nav-link" routerLink="/members" routerLinkActive="active">Members </a></li>

            <li class="nav-item"><a class="nav-link" routerLink="/lists" routerLinkActive="active">Lists</a></li>

            <li class="nav-item"><a class="nav-link" routerLink="/messages" routerLinkActive="active">Messages</a></li>

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

              <a class="nav-link" routerLink="/admin" routerLinkActive="active">Admin</a>

            </li>

          </ng-container>

## photo-management

> ng g c site/admin/photoManagement --skip-tests --module=core/modules/shared

Also add to the shared module.

We’ll get back to this component later in upcoming lessons

## user-management

> ng g c site/admin/userManagement --skip-tests --module=core/modules/shared

Also add to the shared module

### Modals

User management is using modals. For this enable the modal module in the shared module. Add it to bpth imports and exports array.

Check for details:

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

### user-management.component.ts

* The component is using a modal componenet
* This is created [site/modals/roles-modal component below](#_/site/modals/roles-modal).

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

import { Subscription } from 'rxjs';

import { ToastrService } from 'ngx-toastr';

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

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

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

import { SiteRole } from '../../../core/models-interfaces/site-role.model';

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

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;

  rolesUpdateSubscription!: Subscription;

  //modal reference for modal window

  bsModalRef: BsModalRef<RolesModalComponent> = new BsModalRef<RolesModalComponent>();

  constructor(private helperService: HelperService,

              private adminService: AdminService,

              private toastr: ToastrService,

              private modalService: BsModalService){

  }

  ngOnDestroy(): void {

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

    if(this.rolesUpdateSubscription) this.rolesUpdateSubscription.unsubscribe();

  }

  ngOnInit(): void {

    this.loadUserWithRoles();

  }

  loadUserWithRoles(){

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

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

        this.users = users;

      }

    });

  }

  openRolesModal(user?: UserDto) {

    if(!user){

      this.toastr.error("Unable to get user", "Error");

      return;

    }

    //convert the user assigned roles to site roles and mark isSelected=true for the roles that are assigned

    var roles = this.adminService.translateUserRolesToSiteRoles(user.roles);

    this.helperService.logIfFrom(user.displayName, "openRolesModal");

    this.helperService.logIf(roles);

    this.handleRolesModal(user, roles);

  }

  private handleRolesModal(user: UserDto, roles: SiteRole[]){

    //default is overwritten check followig for the defaults

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

    /\*

    const modalOptions: ModalOptions = {

      initialState: {

        roles: roles,

        user: user

      }

    }

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

    \*/

    //spread operator doesnt work, deep copy not happening

    //const newRolesObject = [...roles.slice()];

    const newRolesObject = JSON.parse(JSON.stringify(roles)) as typeof roles;

    const config = {

      class: 'modal-dialog-center',

      initialState: {

        roles: newRolesObject,

        user: user

      }

    }

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

    //this.bsModalRef.content!.closeBtnName = 'Close'; just local button

    //on modal hide/save do the stuff

    //subscribe for updatedRoles, in this case do not do onHide subscription

    this.bsModalRef.content!.updatedRoles.subscribe({

      next: (rolesFromModal: SiteRole[]) => {

        this.performRoleUpdate(rolesFromModal, user, roles);

      }

    })

    //check for isSave=true if subscribing onHide. this case do not do above updatedRoles.subscribe

    /\*

    this.bsModalRef.onHide?.subscribe({

      next: () => {

        //get the roles from modal

        const rolesFromModal = this.bsModalRef.content?.roles;

        //get the isSave flag as well

        const isSaveFromModal = this.bsModalRef.content?.isSave;

        if(!isSaveFromModal) return;

        this.performRoleUpdate(rolesFromModal!, user, roles);

      }

    });

    \*/

  }

  private performRoleUpdate(rolesFromModal: SiteRole[], user: UserDto, roles: SiteRole[]){

    if(!rolesFromModal || rolesFromModal.length <= 0){

      this.toastr.error("No data found!", "No data")

      return;

    }

    if(this.arrRolesEqual(roles, rolesFromModal!)){

      this.toastr.info("No changes detected!", "No change")

      return;

    }

    //pick the selected roles

    var selected = [...rolesFromModal!.filter(r => r.isSelected === true).map(m => m.name)];

    if(!selected || selected.length <= 0){

      this.toastr.info("No selected roles detected!", "No selection")

      return;

    }

    //do update

    this.doRoleUpdate(user, selected);

  }

  private arrRolesEqual(rolesInitial: SiteRole[], rolesAfter: SiteRole[]){

    return JSON.stringify(rolesInitial.sort()) === JSON.stringify(rolesAfter);

  }

  private doRoleUpdate(user: UserDto, selectedRoles: string[]){

    this.rolesUpdateSubscription = this.adminService.updateUserRoles(user.guId, selectedRoles).subscribe({

      next: (roles: string[]) => {

        //update the user roles

        if(roles){

          user.roles = [...roles];

          user.roles.sort();

        }

        this.toastr.success("Roles updated successfully!", "Success");

      }

    })

  }

}

### user-management.component.html

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

    <div class="row">

        <table class="table">

            <thead>

                <tr>

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

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

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

                </tr>

            </thead>

            <tbody>

                <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>

            </tbody>

        </table>

    </div>

</ng-container>

# /site/modals/roles-modal

This is used inside the [/site/admin/user-management component above](#_user-management).

## role-modal.component.ts

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

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

import { ToastrService } from 'ngx-toastr';

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

import { SiteRole } from 'src/app/core/models-interfaces/site-role.model';

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

import { ZRoles } from 'src/app/core/enums/z-roles';

@Component({

  selector: 'app-roles-modal',

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

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

})

export class RolesModalComponent {

  //when changes have been made and X/Cancel clicked, the roles are still getting updated.

  //In this case no update should happen.

  //Two ways, to do this. either set a boolean flag on Save or emit an input

  isSave: boolean = false;

  @Input() updatedRoles = new EventEmitter();

  user: UserDto = <UserDto>{};

  roles: SiteRole[] = [];

  hasChanges: boolean = false;

  zRole = ZRoles;

  constructor(public bsModalRef: BsModalRef, private toastr: ToastrService){

  }

  onCheckedChangeRole(roleValue: any){

    this.roleChange(roleValue);

    this.hasChanges = true;

  }

  private roleChange(roleValue: any){

    const role = this.roles.find(x => x.value == roleValue);

    if(!role) return;

    role.isSelected = !role.isSelected;

  }

  onUpdateSelectedRoles(){

    if(!this.hasChanges){

      this.toastr.info("No changes made!", "No Changes");

      this.bsModalRef.hide();

    }

    const selectedItems = this.roles.filter(x => x.isSelected === true);

    if(!selectedItems || selectedItems.length <= 0){

      this.toastr.error(`Atleast member role must be added for ${this.user.displayName}`, "No Roles");

      return;

    }

    //all users must have a member role

    const memberRole = selectedItems.find(x => x.name === this.zRole.Member);

    if(!memberRole){

      this.toastr.info(`Every user must be a ${this.zRole.Member} also so this will be added automatically.`);

      selectedItems.push(new SiteRole(this.zRole.Member, this.zRole.Member, true));

      this.roleChange(this.zRole.Member);

    }

    //either set the flag to say Save was clicked

    this.isSave = true;

    //or emit an input

    this.updatedRoles.emit(this.roles);

    this.bsModalRef.hide();

    //after hide the rest of the work will be performed by the user-management.component.ts

  }

}

## role-modal.component.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">

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

            <input type="checkbox"

                class="form-check-input"

                [checked]="role.isSelected"

                value="role.value"

                (change)="onCheckedChangeRole(role.value)"

                [disabled]="role.name === zRole.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()" [disabled]="!roles || roles.length <= 0 || !hasChanges">Save</button>

</div>