Table of Contents

[Current Resources with Changes 2](#_Toc112109862)

[New Resources Added 2](#_Toc112109863)

[Models 2](#_Toc112109864)

[siteRegisterDto 2](#_Toc112109865)

[Account Service 2](#_Toc112109866)

[Add Registration Method 2](#_Toc112109867)

[Add Register component 3](#_Toc112109868)

[site-register.component.ts 3](#_Toc112109869)

[site-register.component.html 4](#_Toc112109870)

[Add Site Home component 5](#_Toc112109871)

[site-home.component.ts 5](#_Toc112109872)

[site-home.component.html 6](#_Toc112109873)

[app.component.html 7](#_Toc112109874)

# Current Resources with Changes

1. app.component.html
2. /core/services/account.service.ts

# New Resources Added

1. /core/models/siteRegisterDto
2. /site/site-home component added
3. /site/site-register component added

# Models

## siteRegisterDto

Create a new model in /core/models folder with name siteRegisterDto.

export class SiteRegisterDto {

    constructor(public userName: string = "", public password: string = "") { }

}

# Account Service

## Add Registration Method

* Add registration method to account service to handle the registration.
* Api is sending back the user name/token just like login on successful login. The register method shouldlogin the user as well

  register(registerDto: SiteRegisterDto) {

    var url = this.apiUrlService.accountRegisterUser;

    if(environment.displayConsoleLog)

      console.log(`AccountService RegisterUrl: ${url}`);

    return this.httpClientService

      .post<UserTokenDto>(url, registerDto)

      .pipe(

        map((respone: UserTokenDto) => {

          const user = respone;

          if (user) {

            //store the user in local storage

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

            this.currentUserSource.next(user);

          }

          return user;

        })

      );

  }

# Add Register component

Run the following command

* ng g c /site/siteRegister --skip-tests

It will also add the siteRegister to the app.module.ts as well

1. Will be using template driven form
2. Submit button should not be enabled when following fails
   * Username is required and minLength should be 5
   * Password is required
3. On Cancel click , the site home (below) should get informed about it.
   * Site home will toggle between registration section and the text section. This will show child to parent communication.
   * It should clear the form
4. Call the account service register method to register
   * on success call the cancel method to hide/clear the form

## site-register.component.ts

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

import { Subscription } from 'rxjs';

import { SiteRegisterDto } from '../../core/models/siteRegisterDto.model';

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

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

import { ErrorMessageService } from '../../core/services/error-message.service';

@Component({

  selector: 'app-site-register',

  templateUrl: './site-register.component.html',

  styleUrls: ['./site-register.component.css']

})

export class SiteRegisterComponent implements OnInit, OnDestroy {

  //to tell the site home to hide the register form since cancel has been clicked

  @Output() cancelRegister = new EventEmitter();

  siteRegister: SiteRegisterDto = <SiteRegisterDto>{};

  //note use of ! or will see a compiler error

  registerSubscription!: Subscription;

  constructor(private accountService: AccountService, private errorMsgService: ErrorMessageService) { }

  ngOnInit(): void {

  }

  ngOnDestroy(): void {

    if (this.registerSubscription) this.registerSubscription.unsubscribe();

  }

  onRegister() {

    if (environment.displayConsoleLog) console.log(this.siteRegister);

    this.registerSubscription = this.accountService.register(this.siteRegister).subscribe({

      next: r => {

        if (environment.displayConsoleLog) {

          console.log("RegisterUserBack: ");

          console.log(r);

          //cancel the form

          this.onCancel();

        }

      }, error: e => {

        this.displayError(e, "Registeration");

      }, complete: () => {

      }

    });

  }

  displayError(error: any, from: string) {

    const errormsg = this.errorMsgService.getHttpErrorMessage(error);

    if(environment.displayConsoleLog) console.log(`displayError-${from} Error: ${errormsg}`);

    alert(`displayError-${from} Error: ${errormsg}`);

  }

  onCancel() {

    if (environment.displayConsoleLog) console.log('cancelled');

    this.cancelRegister.emit(false);

    //reset form

    this.siteRegister = <SiteRegisterDto>{};

  }

}

## site-register.component.html

<form #registerForm="ngForm" (ngSubmit)="onRegister()" autocomplete="off">

    <h2 class="text-center text-primary">Sign up</h2>

    <hr>

    <div class="form-group mt-2">

        <input [(ngModel)]="siteRegister.userName" #username="ngModel" required minlength="5" placeholder="username" id="username" name="username" type="text" class="form-control">

    </div>

    <div class="form-group mt-2">

        <input [(ngModel)]="siteRegister.password" #password="ngModel" required placeholder="password" id="password" name="password" type="password" class="form-control">

    </div>

    <div class="form-group text-center mt-2">

        <button class="btn btn-success mr-2" type="submit" [disabled]="username.invalid || password.invalid">Register</button>&nbsp;

        <button class="btn btn-danger" (click)="onCancel()" type="button">Cancel</button>

    </div>

</form>

# Add Site Home component

Run the following command

* ng g c /site/siteHome --skip-tests

It will also add the siteHome to the app.module.ts as well

1. We’ll have a single site home page which will have the register form as well.
   1. When the register button will be clicked then the text will disappear and register form will display
   2. Will need to create a toggle for switch
   3. For now this will have only the place holders and we’ll fill with register section below
2. Place the site-home-page in app.component.html
   1. And pass the title as @Input decorator to the site home page
3. User will be automatically logged in so subscribe to the currentUser observable and

## site-home.component.ts

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

import { Subscription } from 'rxjs';

import { ErrorMessageService } from '../../core/services/error-message.service';

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

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

@Component({

  selector: 'app-site-home',

  templateUrl: './site-home.component.html',

  styleUrls: ['./site-home.component.css']

})

export class SiteHomeComponent implements OnInit, OnDestroy {

  //getting passed in from app.component.html

  @Input() title = '';

  registerMode: boolean = false;

  isLoggedIn: boolean = false;

  currentUserSubscription!: Subscription;

  constructor(private accountService: AccountService, private errorMsgService: ErrorMessageService) { }

  ngOnInit(): void {

    this.getCurrentUser();

  }

  ngOnDestroy(): void {

    if (this.currentUserSubscription) this.currentUserSubscription.unsubscribe();

  }

  onRegisterToggle() {

    this.registerMode = !this.registerMode;

  }

  onCancelRegisterMode(event: boolean) {

    this.registerMode = event;

  }

  //when the user is logged in then not displaying the register form

  getCurrentUser() {

    //subscribe to the observable being fired from the account service

    this.currentUserSubscription = this.accountService.currentUser$.subscribe({

      next: user => {

        this.isLoggedIn = !!user;

      }, error: e => {

        this.displayError(e, "getCurrentUser SiteHome");

      }, complete: () => {

        //do something on complete

      }

    });

  }

  displayError(error: any, from: string) {

    const errorMsg = this.errorMsgService.getHttpErrorMessage(error);

    if(environment.displayConsoleLog) console.log(`displayError-${from} Error: ${errorMsg}`);

    alert(`displayError-${from} Error: ${errorMsg}`);

  }

}

## site-home.component.html

<div class="container mt-5">

    <h1><span class="title">{{title}}</span> Home</h1>

</div>

<div \*ngIf="!isLoggedIn" class="container mt-5">

    <div \*ngIf="!registerMode" style="text-align: center">

        <h3>Find your <span class="title">{{title}}</span> matches!</h3>

        <p class="lead">Come on in to see who you know or want to connect with... all you need to do is sign up!</p>

        <div class="text-center">

            <button (click)="onRegisterToggle()" class="btn btn-primary btn-lg mr-2">Register</button>&nbsp;

            <button class="btn btn-info btn-lg ml-2">Learn More</button>

        </div>

    </div>

    <div \*ngIf="registerMode" class="conatiner">

        <div class="row justify-content-center">

            <div class="col-4">

                <!--cancelRegister is output in site-register, child is telling the parent that cancel button got clicked-->

                <!--pass the event to a function to reset the register mode-->

                <app-site-register (cancelRegister)="onCancelRegisterMode($event)"></app-site-register>

            </div>

        </div>

    </div>

</div>

# app.component.html

add the site-home to app.component.html

<app-nav [title]="title"></app-nav>

<div class="container main-container" >

    <app-site-home [title]="title"></app-site-home>

</div>

<footer><small><span class="title">{{ title }}</span> WebApiURL: <span class="api-url">{{webApiUrl}}</span></small></footer>