# Project

* Site-06-Ng-error-handling-interceptors
  + Copied from: Site-05-Ng-routing-toast-route-guard-shared-module-misc-components
* For the “MySocialConnect-SPA” issue npm install command to install all packages
  + ng serve : to run the spa

# New & Updates

|  |  |
| --- | --- |
| New | Updates |
| /site/errors/sample | /core/services/helper.service.ts |
| /core/interceptors/errors | app-routing.module.ts |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Clear Database

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

dotnet ef database drop

dotnet ef database update

# Server Side – WebApi

* Server side - Web Api error handling has already been implemented via [Site 04](003%20Project4%20-WebApi-ExceptionHandlingMiddleware.docx).
* Sample controller updated via [Site 06](005%20Project6%20-WebApi-passing-back-errors.docx).

# core/services

## helper.service.ts

Updtaed buggy urls to sample urls

  //sampe controller urls for testing errors

  private urlSample = `${this.baseUrlApi}/sample`;

  public urlSampleAuth: string = `${this.urlSample}/auth`;

  public urlSampleUserNotFound: string = `${this.urlSample}/usernotfound`;

  public urlSampleServerError: string = `${this.urlSample}/servererror`;

  public urlSampleBadRequest: string = `${this.urlSample}/badrequest`;

  public urlSampleLogin: string = `${this.urlSample}/samplelogin`;

# SampleComponent

Create the component in shared module

> ng g c site/errors/sample --module=core/modules/shared --skip-tests

## app-routing.module.ts

* Add sample component, notlogged, notfound and servererror to app routing module
* Also change the invalid route to NotFound

const routes: Routes = [

  { path: '', component: HomeComponent },

  //dummy route to group secure resources together

  {

    path: '',

    runGuardsAndResolvers: 'always',

    canActivate: [authGuard],

    children: [

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

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

      { path: 'members/edit', component: MemberEditComponent },

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

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

    ]

  },

  { path: 'errors/sample', component: SampleComponent},

  { path: 'errors/notloggedin', component: NotLoggedInComponent },

  { path: 'errors/notfound', component: NotFoundComponent },

  { path: 'errors/servererror', component: ServerErrorComponent },

  { path: '\*\*', component: NotFoundComponent, pathMatch: 'full' } //invalid route

];

## nav.compoenent.html

Add the link to sample component

<!--logged in links section-->

        <ul class="navbar-nav me-auto mb-2 mb-md-0">

          <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>

          </ng-container>

          <li class="nav-item"><a class="nav-link" routerLink="/errors/sample" routerLinkActive="active">Sample</a></li>

        </ul>

## sample.component.ts

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

import { Subscription } from 'rxjs';

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

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

@Component({

  selector: 'app-sample',

  templateUrl: './sample.component.html',

  styleUrls: ['./sample.component.css']

})

export class SampleComponent implements OnInit, OnDestroy {

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

  subscription404!: Subscription;

  subscription400!: Subscription;

  subscription500!: Subscription;

  subscription401!: Subscription;

  subscription40val!: Subscription;

  ngOnDestroy(): void {

    if(this.subscription404) this.subscription404.unsubscribe();

    if(this.subscription400) this.subscription400.unsubscribe();

    if(this.subscription500) this.subscription500.unsubscribe();

    if(this.subscription401) this.subscription401.unsubscribe();

    if(this.subscription40val) this.subscription40val.unsubscribe();

  }

  ngOnInit(): void {

  }

  onGet404Error(){

    const url = this.helperService.urlSampleUserNotFound;

    this.subscription404 = this.makeCallGet(url, 'get404Error');

  }

  onGet400Error(){

    const url = this.helperService.urlSampleBadRequest;

    this.subscription400 = this.makeCallGet(url, 'get400Error');

  }

  onGet500Error(){

    const url = this.helperService.urlSampleServerError;

    this.subscription500 = this.makeCallGet(url, 'get500Error');

  }

  onGet401Error(){

    const url = this.helperService.urlSampleAuth;

    this.subscription401 = this.makeCallGet(url, 'get401Error');

  }

  onGet400ValidationErrors(){

    const url = this.helperService.urlSampleLogin;

    this.subscription40val = this.makeCallPost(url, 'get400ValidationErrors',  {})

  }

  private makeCallGet(url: string, title: string){

    return this.httpClientService.get<any>(url).subscribe({

      next: (resposne: any) => {

        this.helperService.logIfFrom(resposne, `Sample - ${title}`);

      },

      error: e => {

        this.helperService.logIfError(e, `Sample - ${title}`)

      },

      complete: () => {}

    });

  }

  private makeCallPost(url: string, title: string, payload: any){

    return this.httpClientService.post<any>(url, payload).subscribe({

      next: (resposne: any) => {

        this.helperService.logIfFrom(resposne, `Sample - ${title}`);

      },

      error: e => {

        this.helperService.logIfError(e, `Sample - ${title}`)

      },

      complete: () => {}

    });

  }

}

## sample.component.html

<h1>Sample Errors</h1>

<ng-container>

    <button class="btn btn-outline-primary me-3" (click)="onGet404Error()">Test 404</button>

    <button class="btn btn-outline-primary me-3" (click)="onGet400Error()">Test 400</button>

    <button class="btn btn-outline-primary me-3" (click)="onGet500Error()">Test 500</button>

    <button class="btn btn-outline-primary me-3" (click)="onGet401Error()">Test 401</button>

    <button class="btn btn-outline-primary me-3" (click)="onGet400ValidationErrors()">Test 400 Validation</button>

</ng-container>

## Run app

Run the app and check the console by clicking each button

# Error Interceptor

## Add interceptor – error.interceptor.ts

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

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

import {

  HttpRequest,

  HttpHandler,

  HttpEvent,

  HttpInterceptor,

  HttpErrorResponse

} from '@angular/common/http';

import { Observable, catchError, throwError } from 'rxjs';

import { NavigationExtras, Router } from '@angular/router';

import { ToastrService } from 'ngx-toastr';

@Injectable()

export class ErrorInterceptor implements HttpInterceptor {

  constructor(private router: Router, private toastr: ToastrService) {}

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

    return next.handle(request)

    .pipe(

      catchError((error: HttpErrorResponse) => {

        if(error){

          switch (error.status){

            case 400:

              if(error.error.errors){

                //model valdation

                const modelStateErrors = [];

                for(const key in error.error.errors){

                  if(error.error.errors[key]){

                    modelStateErrors.push(error.error.errors[key]);

                  }

                }

                throw modelStateErrors.flat();

              }

              else{

                this.toastr.error(error.error, error.status.toString());

              }

              break;

            case 401:

              const message = error.error ? error.error : 'Unauthorized';

              this.toastr.error(message, error.status.toString());

              this.router.navigateByUrl("/errors/notloggedin");

              break;

            case 404:

              this.toastr.error(`${error.error.title} (${error.status}) : url ${error.url}`, error.status.toString());

              this.router.navigateByUrl('/errors/notfound');

              break;

            case 500:

              this.toastr.error("An error has happened", error.status.toString())

              const navExtras: NavigationExtras = {state: {error: error.error}};

              this.router.navigateByUrl("/errors/servererror", navExtras);

              break;

            default:

              this.toastr.error("Something unexpcted went wrong", error.status.toString());

              const navExtras2: NavigationExtras = {state: {error: error.error}};

              this.router.navigateByUrl("/errors/servererror", navExtras2);

              break;

          }

        }

        //return throwError(() => new Error(error.error));

        throw error;

      })

    );

  }

}

## Add to app.module.ts

  providers: [

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

  ]

## sample.component.ts – after interceptor

modelValidationErrors: string[] = [];

  private makeCallPost(url: string, title: string, payload: any){

    this.modelValidationErrors = [];

    return this.httpClientService.post<any>(url, payload).subscribe({

      next: (resposne: any) => {

        this.helperService.logIfFrom(resposne, `Sample - ${title}`);

      },

      error: e => {

        this.helperService.logIfError(e, `Sample - ${title}`);

        this.modelValidationErrors = e;

      },

      complete: () => {}

    });

  }

## sample.component.html – after interceptor

<div class="row mt-5" \*ngIf="modelValidationErrors && modelValidationErrors.length > 0">

    <p>Test 400 validation error</p>

    <ul class="text-danger">

        <li \*ngFor="let error of modelValidationErrors">{{error}}</li>

    </ul>

</div>

## nav.component.ts – after interceptor

onLogin method, comment the error toast since this is now coming from the interceptor

# Error Components

## /site/errors/not-found.componenet

### not-found.componenet.ts

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

@Component({

  selector: 'app-not-found',

  templateUrl: './not-found.component.html',

  styleUrls: ['./not-found.component.css']

})

export class NotFoundComponent {

}

### not-found.component.html

<div class="container">

    <h1>Not found</h1>

    <button class="btn btn-info btn-lg" routerLink="/">Return to Home</button>

</div>

## /site/errors/not-logged-in.componenet

### not-logged-int.compoenent.ts

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

import { ActivatedRoute, ParamMap, Router } from '@angular/router';

import { Subscription, combineLatest, forkJoin } from 'rxjs';

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

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

@Component({

  selector: 'app-not-logged-in',

  templateUrl: './not-logged-in.component.html',

  styleUrls: ['./not-logged-in.component.css']

})

export class NotLoggedInComponent implements OnInit, OnDestroy {

  returnUrl: string = "";

  isLoggedIn: boolean = false;

  querySubscription!: Subscription;

  loggedInUserSubscription!: Subscription;

  forkjoinSubscription!: Subscription;

  combineLatestSubscription!: Subscription;

  constructor(private activateRoute: ActivatedRoute,

            private accountService: AccountService,

            private helperService: HelperService,

            private router: Router){}

  ngOnDestroy(): void {

    if(this.querySubscription) this.querySubscription.unsubscribe();

    if(this.loggedInUserSubscription) this.loggedInUserSubscription.unsubscribe();

    if(this.forkjoinSubscription) this.forkjoinSubscription.unsubscribe();

    if(this.combineLatestSubscription) this.combineLatestSubscription.unsubscribe();

  }

  ngOnInit(): void {

    //this.individual();

    this.together();

  }

  individual(){

    this.loggedInUserSubscription = this.accountService.currentLoggedInUser$.subscribe({

      next: r => {

        this.isLoggedIn = !!r;

        this.getParams();

      },

      error: e => {

        this.helperService.logIfError(e, "getCurrentLoggedInUser")

      }

    });

  }

  getParams(){

    //one way of getting the query param

    /\*

    const returnUrl = this.activateRoute.snapshot.queryParamMap.get("returnUrl");

    if(returnUrl)

      this.returnUrl = returnUrl;

    \*/

    //second way of getting the query param - use this

    this.querySubscription = this.activateRoute.queryParamMap.subscribe({

      next: (params: ParamMap) => {

        const returnUrl = params.get("returnUrl");

        if(returnUrl)

          this.returnUrl = returnUrl;

        if(this.isLoggedIn){

          if(this.returnUrl)

            this.router.navigateByUrl(this.returnUrl);

          else

            this.router.navigateByUrl('/members');

        }

      }

    });

  }

  together(){

    const user$ = this.accountService.currentLoggedInUser$;

    const param$ = this.activateRoute.queryParamMap;

    /\*

    forkjoin the subscription must end

    this.forkjoinSubscription = forkJoin([user$, param$]).subscribe({

      next: result => {

        this.doAction(result);

      },

      error: e => {

        console.log(e);

      },

      complete: () => {

        console.log('complete');

      }

    });

    \*/

    this.combineLatestSubscription = combineLatest([user$,param$]).subscribe({

      next: r => {

        this.doAction(r);

      }

    });

  }

  doAction(r: any){

    const userResult = r[0];

    const paramResult = r[1];

    this.isLoggedIn = !!userResult;

    const temp = paramResult.get('returnUrl');

    if(temp) this.returnUrl = temp;

    if(this.isLoggedIn){

      let redirectTo = "/members";

      if(temp)

        redirectTo = temp;

      this.router.navigateByUrl(redirectTo);

    }

  }

}

### not-logged-in.compoenet.html

<div class="container mt-5" style="text-align: center">

    <h2 class="title"><i class="fa fa-lock"></i> Unauthorized</h2>

    <p>You have accessed a secure page. Please login at the top of this page.</p>

    <p \*ngIf="returnUrl">URL: {{ returnUrl }}</p>

</div>

## /site/errors/server-error.component

### server-error.component.ts

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

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

@Component({

  selector: 'app-server-error',

  templateUrl: './server-error.component.html',

  styleUrls: ['./server-error.component.css']

})

export class ServerErrorComponent implements OnInit {

  error: any;

  constructor(private router: Router){

    //get the current navigation (for extras we put in error interceptor).

    //ngOnInit is very late getting the item

    const navigation = this.router.getCurrentNavigation();

    this.error = navigation?.extras?.state?.['error'];

  }

  ngOnInit(): void {

  }

}

### server-error.component.html

<h4>Internal server error</h4>

<p>Refreshing the page will make the error disapper</p>

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

    <h5 class="text-danger" \*ngIf="error.message">Error: {{error.message}}</h5>

    <p class="font-weight-bold">

        Note: if you are seeing this then this is probably API error/issue

    </p>

    <p>What to do next?</p>

    <ol>

        <li>Open chrome dev tools. Then check the failing request in the network tab</li>

        <li>Examine the request URL - what API end point is being requested?</li>

        <li>Reproduce the problem in Postman - if we get same error Angular is not the problem</li>

    </ol>

    <ng-container \*ngIf="error.details">

        <p style="font-weight: bold">Following is the stack trace</p>

        <code class="mt-5" style="background-color: whitesmoke;">

            {{error.details}}

        </code>

    </ng-container>

</ng-container>