Table of Contents

[Current Resources with Changes 3](#_Toc113404445)

[New Resources Added 3](#_Toc113404446)

[Models 3](#_Toc113404447)

[/core/models.userDto.model.ts 3](#_Toc113404448)

[Services 3](#_Toc113404449)

[api-url.services.ts 3](#_Toc113404450)

[Components 3](#_Toc113404451)

[not-found 3](#_Toc113404452)

[not-found.component.ts 3](#_Toc113404453)

[Not-found.component.html 4](#_Toc113404454)

[Add to app-routing.module.ts 4](#_Toc113404455)

[server-error 4](#_Toc113404456)

[server-error.component.ts 4](#_Toc113404457)

[server-error.component.html 4](#_Toc113404458)

[Add to app-routing.module.ts 5](#_Toc113404459)

[test-errors 5](#_Toc113404460)

[test-errors.component.ts 5](#_Toc113404461)

[test-errors.component.html 7](#_Toc113404462)

[Add Route 8](#_Toc113404463)

[Add link to nav.component.html 8](#_Toc113404464)

[Test 8](#_Toc113404465)

[Interceptor 9](#_Toc113404466)

[Error 9](#_Toc113404467)

[Basic Structure By command 9](#_Toc113404468)

[Updates to the Error Interceptor 9](#_Toc113404469)

[Add the Error Interceptor to app.module.ts 11](#_Toc113404470)

[Clean Up 11](#_Toc113404471)

[nav.component.ts 11](#_Toc113404472)

[login 11](#_Toc113404473)

[/site/site-register.component.ts 11](#_Toc113404474)

[/site/site-register.component.html 12](#_Toc113404475)

# Current Resources with Changes

1. /core/services/api-url.services.ts
2. app.module.ts
3. app-routing.module.ts
4. /site/nav/nav.component.html
5. /site/nav/nav.component.ts

# New Resources Added

1. /core/models/userDto.model.ts
2. /site/errors/test-error component
3. /core/interceptors/error.interceptor.cs
4. /site/errors/not-found component
5. /site/errors/server-error component
6. /site/site-register.component.ts
7. /site/site-register.component.html

# Models

## /core/models.userDto.model.ts

export class userDto {

    constructor(public id: number, public username: string){}

}

# Services

## api-url.services.ts

Add the url for the webapi buggyController urls in this service.

  //buggy controller urls

  private buggyBaseUrl = `${this.apiBaseUrl}buggy/`;

  buggyAuth: string = `${this.buggyBaseUrl}auth`;

  buggyNotFound: string = `${this.buggyBaseUrl}not-found`;

  buggyServerError: string = `${this.buggyBaseUrl}server-error`;

  buggyBadRequest: string = `${this.buggyBaseUrl}bad-request`;

# Components

## not-found

create a new component

* ng g c /site/errors/not-found --skip-tests

This will also add the NotFound to the declarations array of the app.module.ts file.

### not-found.component.ts

No custom code here

### Not-found.component.html

<div class="container">

    <h1>Not Found</h1>

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

</div>

### Add to app-routing.module.ts

  { path: 'not-found', component: NotFoundComponent  },

## server-error

Create a new component

* ng g c /site/errors/server-error --skip-tests

This will also add the TestErrors to the declarations array of the app.module.ts file.

### 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 {

  //gets error as navigation extras from the error interceptor

  error: any;

  constructor(private router: Router) {

    const navigation = this.router.getCurrentNavigation();

    //doesnt work in ng13, either use as below or suppress the error

    //see: https://stackoverflow.com/questions/70106472/property-fname-comes-from-an-index-signature-so-it-must-be-accessed-with-fn

    //this.error = navigation?.extras?.state?.error;

    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">Error: {{ error.message }}</h5>

    <p>Following is the stack trace</p>

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

        {{ error.details }}

    </code>

</ng-container>

### Add to app-routing.module.ts

  { path: 'server-error', component: ServerErrorComponent },

## test-errors

Create a new component

* ng g c /site/errors/test-errors --skip-tests

This will also add the TestErrors to the declarations array of the app.module.ts file.

### test-errors.component.ts

This will have the methods to call the buggy web api controller methods just like postman.

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

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

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

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

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

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

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

@Component({

  selector: 'app-test-errors',

  templateUrl: './test-errors.component.html',

  styleUrls: ['./test-errors.component.css']

})

export class TestErrorsComponent implements OnInit {

  validationErrors: string[] = [];

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

  ngOnInit(): void {

  }

  getNotFoundError() {

    var url = this.apiUrlService.buggyNotFound;

    if (environment.displayConsoleLog) console.log(`NotFoundURL: ${url}`);

    this.httpClientService.get<userDto>(url).subscribe({

      next: (user: userDto) => {

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

      }, error: e => {

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

      }, complete: () => {

        if (environment.displayConsoleLog) console.log("NotFoundComplete");

      }

    });

  }

  getBadRequestError() {

    var url = this.apiUrlService.buggyBadRequest;

    if (environment.displayConsoleLog) console.log(`BadRequestURL: ${url}`);

    this.httpClientService.get<string>(url).subscribe({

      next: response => {

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

      }, error: e => {

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

      }, complete: () => {

        if (environment.displayConsoleLog) console.log("BadRequestComplete");

      }

    });

  }

  getServerError() {

    var url = this.apiUrlService.buggyServerError;

    if (environment.displayConsoleLog) console.log(`ServerErrorURL: ${url}`);

    this.httpClientService.get<string>(url).subscribe({

      next: response => {

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

      }, error: e => {

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

      }, complete: () => {

        if (environment.displayConsoleLog) console.log("ServerErrorComplete");

      }

    });

  }

  getAuthError(){

    var url = this.apiUrlService.buggyAuth;

    if (environment.displayConsoleLog) console.log(`AuthURL: ${url}`);

    this.httpClientService.get<string>(url).subscribe({

      next: response => {

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

      }, error: e => {

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

      }, complete: () => {

        if (environment.displayConsoleLog) console.log("AuthComplete");

      }

    });

  }

  getValidationError() {

    const register: SiteRegisterDto = { userName: "", password: "pas" };

    var url = this.apiUrlService.accountRegisterUser;

    if (environment.displayConsoleLog) {

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

      console.log(`RegisterModel: ${register}`);

    }

    this.httpClientService.post<UserTokenDto>(url, register).subscribe({

      next: (user: UserTokenDto) => {

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

      }, error: e => {

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

        //put this line only after implementing the error interceptor

        this.validationErrors = e;

      }, complete: () => {

        if (environment.displayConsoleLog) console.log("ValidationComplete");

      }

    });

  }

}

### test-errors.component.html

This will have the buttons to call the methods in the component

<ng-container>

    <button class="btn btn-outline-primary mt-3" (click)="getNotFoundError()">Test NotFound Error</button><br>

    <button class="btn btn-outline-primary mt-3" (click)="getBadRequestError()">Test BadRequest Error</button><br>

    <button class="btn btn-outline-primary mt-3" (click)="getServerError()">Test Server Error</button><br>

    <button class="btn btn-outline-primary mt-3" (click)="getAuthError()">Test Auth Error</button><br>

    <button class="btn btn-outline-primary mt-3" (click)="getValidationError()">Test Validation Error</button>

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

        <ul class="text-danger">

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

        </ul>

    </div>

</ng-container>

### Add Route

Add the route to app-routing.module.ts

{ path: 'test-errors', component: TestErrorsComponent},

### Add link to nav.component.html

Add link to the nav.component.html below the ng-container links

<li class="nav-item">

  <a class="nav-link" routerLink="/test-errors" routerLinkActive="active" (click)="onNavBarItemClickCloseNavBar()">Test Errors</a>

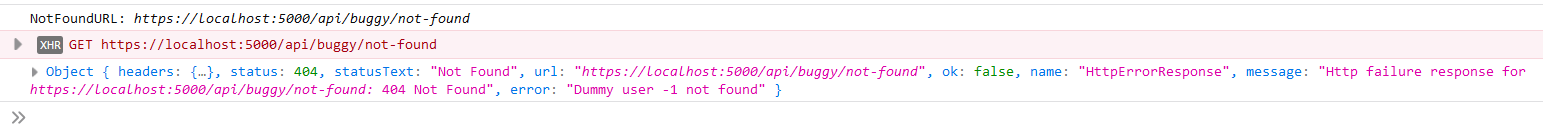
</li>

### Test

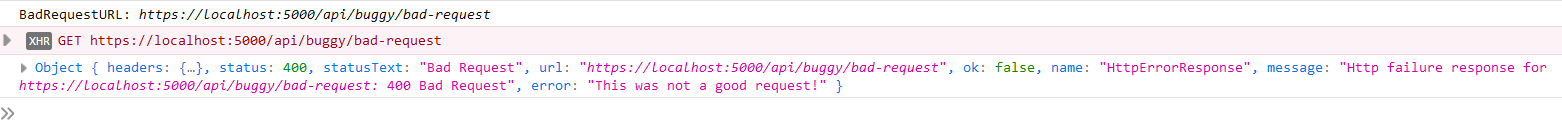
Run the app with ng serve and then click on test error link.

Make sure that the wep api is running.

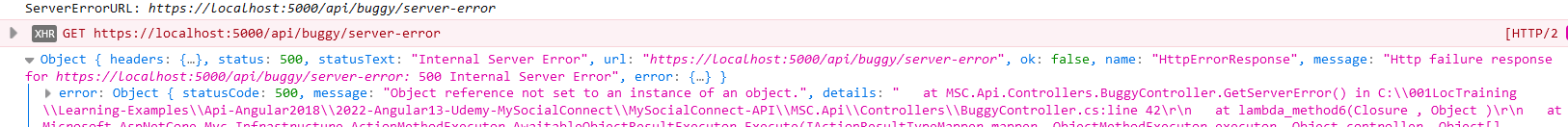
#### Not Found



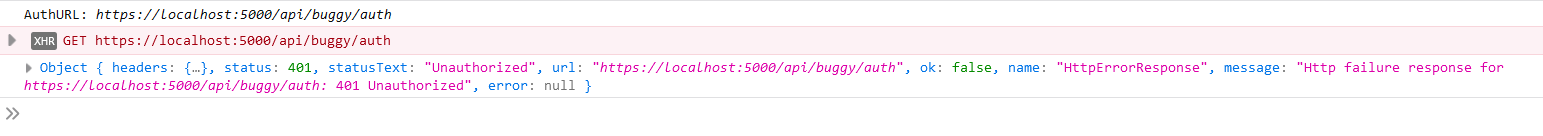
#### Bad Request



#### Server Error



#### Auth Error



#### Validation Error

Text

Description automatically generated

# Interceptor

## Error

Create a global error interceptor to intercept the http errors.

Use following command

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

### Basic Structure By command

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

import {

  HttpRequest,

  HttpHandler,

  HttpEvent,

  HttpInterceptor

} from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable()

export class ErrorInterceptor implements HttpInterceptor {

  constructor() {}

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

    return next.handle(request);

  }

}

### Updates to the Error Interceptor

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

import {

  HttpRequest,

  HttpHandler,

  HttpEvent,

  HttpInterceptor

} from '@angular/common/http';

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

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

import { ToastrService } from 'ngx-toastr';

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

@Injectable()

export class ErrorInterceptor implements HttpInterceptor {

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

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

    return next.handle(request).pipe(

      //error is the http response that we see when writing the console.log

      catchError(error => {

        if (error) {

          switch (error.status) {

            case 400:

              if (error.error.errors) {

                //400 validation errors, it has error.error.errors array, start

                const modalStateErrors = [];

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

                  if (error.error.errors[key])

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

                }

                //modalStateErrors so that the errors could be displayed near the form

                throw modalStateErrors;

                //400 validation errors, end

              }

              else {

                //400 all other errors, start

                //will use the toastr service to display the error

                this.toastrService.error(`${error.statusText}: ${error.error}`, error.status);

                //400 all other errors, end

              }

              break;

            case 401:

              //will use the toastr service to display the error

              this.toastrService.error(`${error.statusText}: ${error.error}`, error.status);

              break;

            case 404:

              //redirect to the not found page

              this.router.navigateByUrl('/not-found');

              break;

            case 500:

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

              //redirect to the server error page

              this.router.navigateByUrl('/server-error', navigationExtras);

              break;

            default:

              this.toastrService.error('Something unexpected went wrong');

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

              break;

          }

        }

        //return the error

        return throwError(error);

      })

    );

  }

}

### Add the Error Interceptor to app.module.ts

  providers: [

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

  ]

# Clean Up

## nav.component.ts

### login

Not displaying the error message when the login error happens. In this case the error is getting displayed by the error interceptor so comment out the error display call.

## /site/site-register.component.ts

Declare variable

  //after implementation on error interceptor

  validationErrors: string[] = [];

and then update the register method as

  onRegister() {

    this.validationErrors = [];//reset

    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 => {

        //due to error intercepter we are getting a flat array of validation items so for modal validation need to check that

        //check array and length > 0

        //other cases the error interceptor is displaying the error

        if (e?.length) {

          if (environment.displayConsoleLog) console.log("\*\*\*inside model validation errors\*\*\*");

          this.validationErrors = e;

        }

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

        //this.displayError(e, "Registeration");

      }, complete: () => {

      }

    });

  }

## /site/site-register.component.html

Display the validation error under the form

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

        <ul class="text-danger">

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

        </ul>

    </div>