

ASPNET Core and Angular Course changes

Module 1

No changes

Module 2

Installed .Net Core 2.1, Angular CLI 6.0.8 and Bootstrap 4. Also the GIT repo is now a single Git repository for both the projects.

Migrating to .Net Core 2.1

Check which versions you have installed to see what you will need to upgrade. To check the version of dotnet use:

```
dotnet --info
```

If the version of the .Net Core SDK is lower than 2.1.300 then you will need to upgrade. The steps to upgrade the project are:

1. Download and install 2.1.300 from the Microsoft.Net site. This will add this runtime in addition to the previous one.
2. Review the following document to see changes between projects to .Net 2.1.
https://docs.microsoft.com/en-us/aspnet/core/migration/20_21?view=aspnetcore-2.1
3. You will need to update the .csproj file to the following:
<https://github.com/TryCatchLearn/DatingApp/blob/8c4c42013189bd69d6dda4c2e0b96efa004f4e47/DatingApp.API/DatingApp.API.csproj>
4. Update the program.cs file to the following:
<https://github.com/TryCatchLearn/DatingApp/blob/8c4c42013189bd69d6dda4c2e0b96efa004f4e47/DatingApp.API/Program.cs>
5. Create a folder called Properties
6. Create a file called launchSettings.json and paste the following code in:
<https://github.com/TryCatchLearn/DatingApp/blob/8c4c42013189bd69d6dda4c2e0b96efa004f4e47/DatingApp.API/Properties/launchSettings.json>
7. Run dotnet build
8. Test the application to ensure it still works.

Migrating to Angular v6

To check which version you have installed use:

```
ng --version
```

If on less than version 6 will need to upgrade. The steps to do so are as follows:

First, install the angular cli globally on your computer as follows (the course is re-recorded using v6.0.8 of the Angular CLI so would recommend this:

```
npm install -g @angular/cli@6.0.8
```

The rest of these commands need to be run in the context of the DatingApp.SPA project use:

```
npm install @angular/cli@6.0.8
```

You may need to do this more than once if it still shows as less than v6.0.8 in the output of the terminal window. Next run:

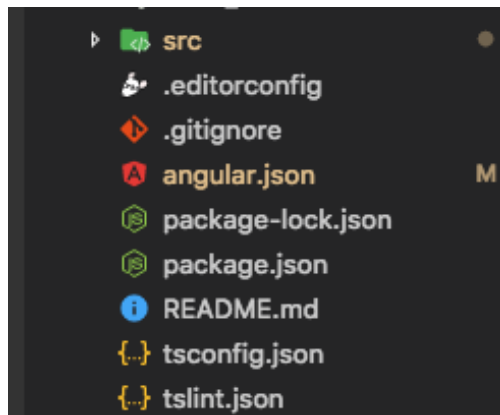
```
ng update @angular/core
```

Next run `ng --version` again and ensure that you see the following output:

```
Angular CLI: 6.0.8
Node: 8.11.2
OS: darwin x64
Angular: 6.0.9
... animations, common, compiler, compiler-cli, core, forms
... http, language-service, platform-browser
... platform-browser-dynamic, router

Package                                  Version
-----
@angular-devkit/architect                0.6.8
@angular-devkit/build-angular            0.6.8
@angular-devkit/build-optimizer          0.6.8
@angular-devkit/core                     0.6.8
@angular-devkit/schematics               0.6.8
@angular/cli                             6.0.8
@ngtools/webpack                         6.0.8
@schematics/angular                      0.6.8
@schematics/update                       0.6.8
rxjs                                     6.2.1
typescript                              2.7.2
webpack                                  4.8.3
```

You should also see an `angular.json` file in the solution explorer rather than `angular-cli.json`.



Migrating to Bootstrap 4.

Run the following command:

```
npm install bootstrap@4.1.1
```

Then in the angular.json file remove the references to this in the “styles” array. The styles in the Angular.json should then look as follows:

```
"assets": [  
  "src/favicon.ico",  
  "src/assets"  
],  
"styles": [  
  "src/styles.css"  
],
```

Then instead of importing in the angular.json file we use the styles.css file to import the bootstrap and font-awesome styles into our application as follows:

```
inchSettings.json  styles.css x  
1  /* You can add global styles to this file, and also import other style files */  
2  @import '../node_modules/bootstrap/dist/css/bootstrap.min.css';  
3  @import '../node_modules/bootswatch/dist/united/bootstrap.min.css';
```

Copy and paste the following:

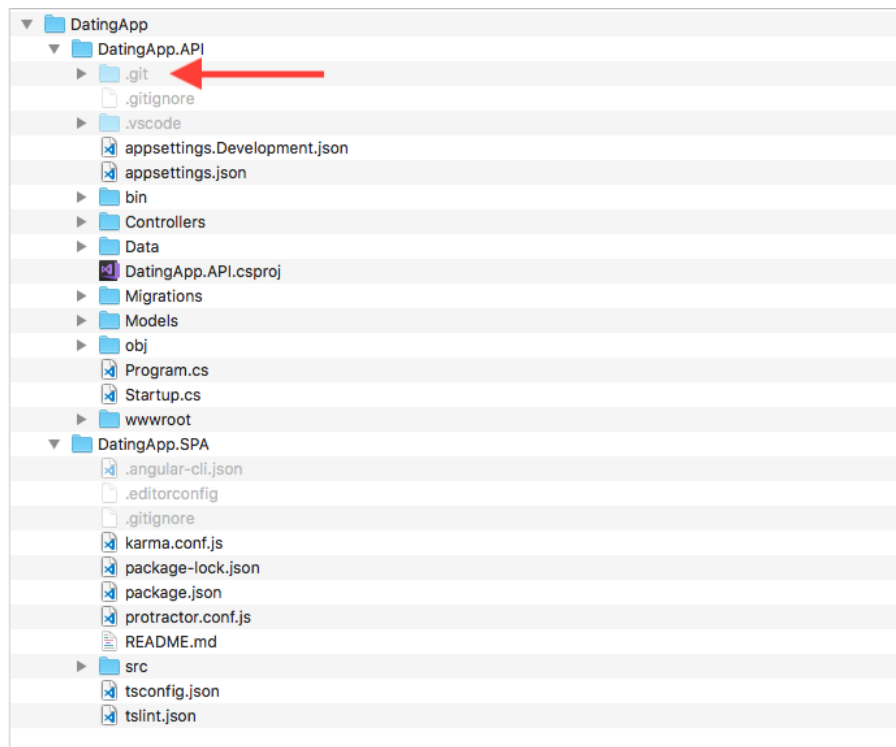
```
@import '../node_modules/bootstrap/dist/css/bootstrap.min.css';  
@import '../node_modules/bootswatch/dist/united/bootstrap.min.css';
```

Git changes

This part is optional but since the lessons in the course all refer to a single repository now please make the following changes to enable this.

Firstly, we need to remove the Git repos from the API project and the SPA project. To do this please remove the .git folder from both these project folders. These are hidden folders so to see this in file explorer or finder you will need to show hidden folders. On a Mac you can do this by pressing SHIFT+CMD+Period together.

Delete both the .git folders:



Ensure you see the following responses in each project folder for the following command:

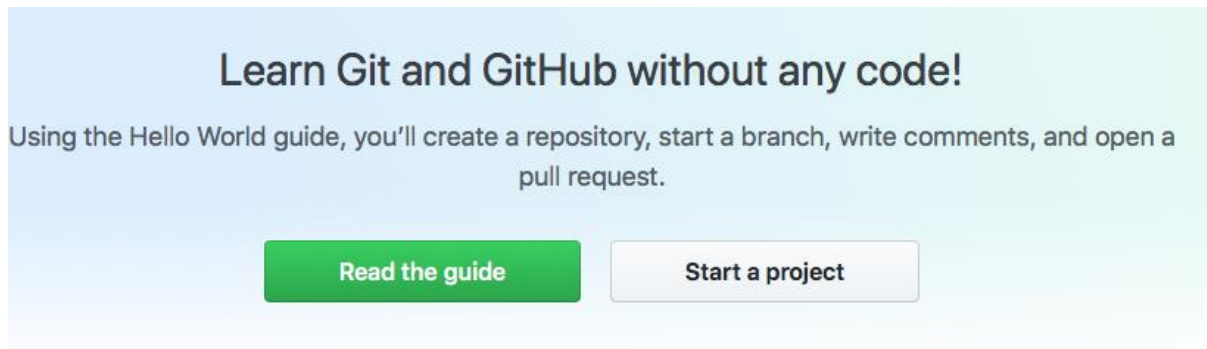
```
MacBook-Pro:DatingApp neil$ git status
fatal: Not a git repository (or any of the parent directories): .git
MacBook-Pro:DatingApp neil$ cd DatingApp.API/
MacBook-Pro:DatingApp.API neil$ git status
fatal: Not a git repository (or any of the parent directories): .git
MacBook-Pro:DatingApp.API neil$ cd ..
MacBook-Pro:DatingApp neil$ cd DatingApp.SPA/
MacBook-Pro:DatingApp.SPA neil$ git status
fatal: Not a git repository (or any of the parent directories): .git
MacBook-Pro:DatingApp.SPA neil$
```

Now in the DatingApp folder run git init to create a new git repository:

```
MacBook-Pro:DatingApp neil$ git init
Initialized empty Git repository in /Users/neil/Documents/Projects/DA-Course-Final/Exercise Files/Module 2/Code/After/DatingApp/.git/
MacBook-Pro:DatingApp neil$
```


The .gitignore files in both the DatingApp.SPA and the DatingApp.API will be respected so no further changes will be necessary.

Create a GitHub repository at GitHub.com to publish this repo. Create new account if necessary and use the Start a project button to create a new repository:



Give it the name DatingApp (or whatever you prefer) and create the repository:

Owner **Repository name**

 TryCatchLearn ▾ / DatingAppp ✓

Great repository names are short and memorable. Need inspiration? How about **musical-sniffle**.

Description (optional)

☒ **Public**
Anyone can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.


☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾ | Add a license: **None** ▾ ⓘ

Create repository

Then on the next screen, copy the line starting git remote into your clipboard

Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** **SSH** <https://github.com/TryCatchLearn/test.git>

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/TryCatchLearn/test.git
git push -u origin master
```

Then paste this into your terminal window where you initialized the repository.

Then use the 'push to...' command to push the repository up to the GitHub repo.

Module 3 Changes

No changes in this module

Module 4 Changes

nav.component.html ([link](#))

This has been updated for Bootstrap 4

auth.service.ts

As the Angular Http module is deprecated and will be removed from Angular v7 we have removed this from our application and it is replaced with the HttpClient module. The two services are quite similar but there are some differences to be aware of. Also, since we are using Angular 6 this means we are using RxJS 6 for our code and there are a few changes here also.

The Angular v5 version ([link](#)):

auth.service.tsRaw

```
1 import { Injectable } from '@angular/core';
2 import { Http, RequestOptions, Headers, Response } from '@angular/http';
3 import 'rxjs/add/operator/map';
4
5 @Injectable()
6 export class AuthService {
7   baseUrl = 'http://localhost:5000/api/auth/';
8   userToken: any;
9
10  constructor(private http: Http) {}
11
12  login(model: any) {
13    return this.http.post(this.baseUrl + 'login', model, this.requestOptions()).map((response: Response) => {
14      const user = response.json();
15      if (user && user.tokenString) {
16        localStorage.setItem('token', user.tokenString);
17      }
18    });
19  }
20
21  register(model: any) {
22    return this.http.post(this.baseUrl + 'register', model, this.requestOptions());
23  }
24
25  private requestOptions() {
26    const headers = new Headers({ 'Content-type': 'application/json' });
27    return new RequestOptions({ headers: headers });
28  }
29 }
```

Is now updated to the following ([link](#)):

28 lines (23 sloc) 634 BytesRawBlameHistory

```
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { map } from 'rxjs/operators';
4
5 @Injectable({
6   providedIn: 'root'
7 })
8 export class AuthService {
9   baseUrl = 'http://localhost:5000/api/auth/';
10
11  constructor(private http: HttpClient) {}
12
13  login(model: any) {
14    return this.http.post(this.baseUrl + 'login', model).pipe(
15      map((response: any) => {
16        const user = response;
17        if (user) {
18          localStorage.setItem('token', user.token);
19        }
20      })
21    );
22  }
23
24  register(model: any) {
25    return this.http.post(this.baseUrl + 'register', model);
26  }
27 }
```

Changes:

- The 'requestOptions' method has been removed. The default content-type is application/json so we don't need to add this.
- The login method is now using the new RxJs syntax for the map method. This means we need to chain the 'pipe' method to the http.post request, and then use the 'map' operator in there.
- The import for the 'map' operator now uses the import {map} from 'rxjs/operators' syntax rather than the import 'rxjs/add/operator/map' syntax.

home.component.html

This has been updated to use Bootstrap 4 classes ([link](#))

register.component.html

This has been updated to use Bootstrap 4 classes ([link](#))

Module 5 Changes

The angular application is now using the HttpInterceptor class as since we are using the HttpClient module now we have access to the http interceptor which means we can intercept http errors globally so can remove any error handling from our services. So the handleError method in the auth.service.ts is no longer required.

So the auth.service.ts file is the same as the one used in Module 4 ([link](#)) and there is a new ts file created called error.interceptor.ts.

Changes:

Create a new file in the _services folder in DatingApp-SPA called error.interceptor.ts and use the code available from here ([link](#)):


```
41 lines (38 sloc) | 1.62 KB
Raw Blame History

1 import {Injectable} from '@angular/core';
2 import {HttpInterceptor, HttpRequest, HttpHandler, HttpEvent, HttpResponse, HTTP_INTERCEPTORS} from '@angular/common/http';
3 import { Observable, throwError } from 'rxjs';
4 import { catchError } from 'rxjs/operators';
5
6 @Injectable()
7 export class ErrorInterceptor implements HttpInterceptor {
8     intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
9         return next.handle(req).pipe(
10             catchError(error => {
11                 if (error instanceof HttpResponse) {
12                     if (error.status === 401) {
13                         return throwError(error.statusText);
14                     }
15                     const applicationError = error.headers.get('Application-Error');
16                     if (applicationError) {
17                         console.error(applicationError);
18                         return throwError(applicationError);
19                     }
20                     const serverError = error.error;
21                     let modalStateErrors = '';
22                     if (serverError && typeof serverError === 'object') {
23                         for (const key in serverError) {
24                             if (serverError[key]) {
25                                 modalStateErrors += serverError[key] + '\n';
26                             }
27                         }
28                     }
29                     return throwError(modalStateErrors || serverError || 'Server Error');
30                 }
31             })
32         );
33     }
34 }
35
36 export const ErrorInterceptorProvider = {
37     provide: HTTP_INTERCEPTORS,
38     useClass: ErrorInterceptor,
39     multi: true
40 };
```

Recommend watching the lesson in Section 5 called “Setting up the global error handler in Angular” to get an understanding of what we are doing here.

Module 6 Changes

Due to the angular.json file not respecting the ordering of the imported css files we need to import them directly into the styles.css file.

So remove the imports from the angular.json file styles array so the only import is the styles.css file:

```

20     "tsConfig": "src/tsconfig.app.json",
21     "assets": [
22       "src/favicon.ico",
23       "src/assets"
24     ],
25     "styles": [
26       "src/styles.css"
27     ],
28     "scripts": [
29       "node_modules/alertifyjs/build/alertify.min.js"
30     ]
31   },

```

The import for alertify.min.js can stay in here without an issue so please ensure the angular.json file looks like above ([link](#))

The styles.css should look like the following with the alertify css files imported ([link](#))

```

6 lines (6 sloc) | 414 Bytes
Raw Blame History
1  /* You can add global styles to this file, and also import other style files */
2  @import '../node_modules/bootstrap/dist/css/bootstrap.min.css';
3  @import '../node_modules/bootswatch/dist/united/bootstrap.min.css';
4  @import '../node_modules/font-awesome/css/font-awesome.min.css';
5  @import '../node_modules/alertifyjs/build/css/alertify.min.css';
6  @import '../node_modules/alertifyjs/build/css/themes/bootstrap.min.css';

```

Angular 2 jwt.

Since we are now using the HttpClient module we need to update the version of @auth0/angular-jwt.

Run the following command in the context of the DatingApp-SPA

```
npm install @auth0/angular-jwt@2.0.0
```

auth.service.ts

Update the **auth.service.ts** file to use this newly installed service. The 'isTokenExpired' and 'DecodeToken' is now provided as a service and we need to create a new instance of this to use it. The auth.service.ts file should look as follows after this module ([link](#)):

```
38 lines (32 sloc) | 979 Bytes
Raw Blame History

1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { map } from 'rxjs/operators';
4 import { JwtHelperService } from '@auth0/angular-jwt';
5
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class AuthService {
10   baseUrl = 'http://localhost:5000/api/auth/';
11   jwtHelper = new JwtHelperService();
12   decodedToken: any;
13
14   constructor(private http: HttpClient) {}
15
16   login(model: any) {
17     return this.http.post(this.baseUrl + 'login', model).pipe(
18       map((response: any) => {
19         const user = response;
20         if (user) {
21           localStorage.setItem('token', user.token);
22           this.decodedToken = this.jwtHelper.decodeToken(user.token);
23           console.log(this.decodedToken);
24         }
25       })
26     );
27   }
28
29   register(model: any) {
30     return this.http.post(this.baseUrl + 'register', model);
31   }
32
33   loggedIn() {
34     const token = localStorage.getItem('token');
35     return !this.jwtHelper.isTokenExpired(token);
36   }
37 }
```

We also need to update the **app.component.ts** file with this change ([link](#)):

```
22 lines (18 sloc) | 582 Bytes
Raw Blame History

1 import { Component, OnInit } from '@angular/core';
2 import { AuthService } from '../_services/auth.service';
3 import { JwtHelperService } from '@auth0/angular-jwt';
4
5 @Component({
6   selector: 'app-root',
7   templateUrl: './app.component.html',
8   styleUrls: ['./app.component.css']
9 })
10 export class AppComponent implements OnInit {
11   jwtHelper = new JwtHelperService();
12
13   constructor(private authService: AuthService) {}
14
15   ngOnInit() {
16     const token = localStorage.getItem('token');
17     if (token) {
18       this.authService.decodedToken = this.jwtHelper.decodeToken(token);
19     }
20   }
21 }
```

```
38 lines (32 sloc) | 979 Bytes
Raw Blame History

1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { map } from 'rxjs/operators';
4 import { JwtHelperService } from '@auth0/angular-jwt';
5
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class AuthService {
10   baseUrl = 'http://localhost:5000/api/auth/';
11   jwtHelper = new JwtHelperService();
12   decodedToken: any;
13
14   constructor(private http: HttpClient) {}
15
16   login(model: any) {
17     return this.http.post(this.baseUrl + 'login', model).pipe(
18       map((response: any) => {
19         const user = response;
20         if (user) {
21           localStorage.setItem('token', user.token);
22           this.decodedToken = this.jwtHelper.decodeToken(user.token);
23           console.log(this.decodedToken);
24         }
25       })
26     );
27   }
28
29   register(model: any) {
30     return this.http.post(this.baseUrl + 'register', model);
31   }
32
33   loggedIn() {
34     const token = localStorage.getItem('token');
35     return !this.jwtHelper.isTokenExpired(token);
36   }
37 }
```

ngx-bootstrap

This works with Bootstrap 3 and 4 but you might want to update the version used to the version used in the demonstrations:

```
npm install ngx-bootstrap@3.0.1
```

The HTML for the dropdown we add in this module has changed slightly, please update the nav.component.html ([link](#)) for how this file should look:

```

17 <div *ngIf="loggedIn()" class="dropdown" dropdown>
18   <a class="dropdown-toggle text-light" dropdownToggle>
19     Welcome {{authService.decodedToken?.unique_name | titlecase}}
20   </a>
21
22   <div class="dropdown-menu mt-3" *dropdownMenu>
23     <a class="dropdown-item" href="#">
24       <i class="fa fa-user"></i> Edit Profile</a>
25     <div class="dropdown-divider"></div>
26     <a class="dropdown-item" (click)="logout()">
27       <i class="fa fa-sign-out"></i> Logout</a>
28   </div>
29 </div>
30
31 <form *ngIf="!loggedIn()" #loginForm="ngForm" class="form-inline my-2 my-lg-0" (ngSubmit)="login()">
32   <input class="form-control mr-sm-2" type="text" name="username" placeholder="Username" required [(ngModel)]="model.username">
33   <input class="form-control mr-sm-2" type="password" name="password" placeholder="Password" required [(ngModel)]="model.password">
34   <button [disabled]="!loginForm.valid" class="btn btn-success my-2 my-sm-0" type="submit">Login</button>
35 </form>
36 </div>
37

```

bootswatch

This needs to match the version of Bootstrap installed. Please update this:

npm install bootswatch@4.1.1

The only other change needed for this is to change the background color of the **nav.component.html** navbar class ([link](#)):

```

39 lines (33 sloc) | 1.48 KB
Raw Blame History
1 <nav class="navbar navbar-expand-md navbar-dark bg-primary">
2   <div class="container">
3     <a class="navbar-brand" href="#">Dating App</a>

```

Module 7 Changes

No changes, other than a small fix to the routes.ts file to resolve a bug that prevented the routes from activating when browsing to localhost:4200 on a new tab when remaining logged in to the app.

routes.ts ([link](#))

routes.ts

Raw

```
1 import {Routes} from '@angular/router';
2 import { HomeComponent } from './home/home.component';
3 import { MemberListComponent } from './member-list/member-list.component';
4 import { MessagesComponent } from './messages/messages.component';
5 import { ListsComponent } from './lists/lists.component';
6 import { AuthGuard } from './guards/auth.guard';
7
8 export const appRoutes: Routes = [
9   {path: 'home', component: HomeComponent},
10   {
11     path: '',
12     runGuardsAndResolvers: 'always',
13     canActivate: [AuthGuard],
14     children: [
15       {path: 'members', component: MemberListComponent},
16       {path: 'messages', component: MessagesComponent},
17       {path: 'lists', component: ListsComponent},
18     ]
19   },
20   {path: '**', redirectTo: 'home', pathMatch: 'full'},
21 ];
```

Module 8 Changes

No changes in this module

Module 9 Changes

The user.service.ts is now making use of the HttpClient module.

In the early part of this module we send up the Jwt token within the user.service.ts as shown here ([link](#))

```
29 lines (23 sloc) | 737 Bytes
Raw Blame History

1 import { Injectable } from '@angular/core';
2 import { environment } from '../../environments/environment';
3 import { HttpClient, HttpHeaders } from '@angular/common/http';
4 import { Observable } from 'rxjs';
5 import { User } from '../_models/user';
6
7 const httpOptions = {
8   headers: new HttpHeaders({
9     'Authorization': 'Bearer ' + localStorage.getItem('token')
10   })
11 };
12
13 @Injectable({
14   providedIn: 'root'
15 })
16 export class UserService {
17   baseUrl = environment.apiUrl;
18
19   constructor(private http: HttpClient) {}
20
21   getUsers(): Observable<User[]> {
22     return this.http.get<User[]>(this.baseUrl + 'users', httpOptions);
23   }
24
25   getUser(id): Observable<User> {
26     return this.http.get<User>(this.baseUrl + 'users/' + id, httpOptions);
27   }
28 }
```

Later on in this module we change this to use the `HttpInterceptor` and a module provided by the `@auth0/angular-jwt` so that we can send up the token automatically for every request, except for those requests that are calling the 'auth controller' methods in our API.

app.module.ts ([link](#))

To use this we import the module:

```
7 import { JwtModule } from '@auth0/angular-jwt';
```

Then we configure it in the imports section of the **app.module.ts** file:

```
43     ],
44     imports: [
45         BrowserModule,
46         HttpClientModule,
47         FormsModule,
48         BsDropdownModule.forRoot(),
49         TabsModule.forRoot(),
50         RouterModule.forRoot(appRoutes),
51         NgxGalleryModule,
52         JwtModule.forRoot({
53             config: {
54                 tokenGetter: tokenGetter,
55                 whitelistedDomains: ['localhost:5000'],
56                 blacklistedRoutes: ['localhost:5000/api/auth']
57             }
58         })
59     ],
```

The route listed in the blacklistedRoutes array means that we do not send up the token for requests that go to api/auth.

We can then remove the HttpOptions that were added to the user.service.ts to send up the token manually. The version of the user.service.ts at the end of this module looks like this ([link](#)):


```
user.service.ts
1
2 import { Injectable } from '@angular/core';
3 import { environment } from '../../environments/environment';
4 import { HttpClient, HttpHeaders } from '@angular/common/http';
5 import { Observable } from 'rxjs';
6 import { User } from '../models/user';
7
8 const httpOptions = {
9   headers: new HttpHeaders({
10     'Authorization': 'Bearer ' + localStorage.getItem('token')
11   })
12 };
13
14 @Injectable({
15   providedIn: 'root'
16 })
17 export class UserService {
18   baseUrl = environment.apiUrl;
19
20   constructor(private http: HttpClient) {}
21
22   getUsers(): Observable<User[]> {
23     return this.http.get<User[]>(this.baseUrl + 'users', httpOptions);
24   }
25
26   getUser(id): Observable<User> {
27     return this.http.get<User>(this.baseUrl + 'users/' + id, httpOptions);
28   }
29 }
```

As we are now using Bootstrap 4 the following files have been updated to use the new bootstrap classes. Please overwrite your current files with these ones.

member-list.component.html ([link](#))
member-detail.component.html ([link](#))
member-detail.component.css ([link](#))
member-card.component.html ([link](#))
member-card.component.css ([link](#))

The route resolvers are updated to use the new RxJS import and syntax. The 'catch' operator has been renamed to 'catchError' and now needs to be defined inside the 'pipe' method from RxJS. So the resolvers look as follows:

```
24 lines (21 sloc) | 884 Bytes
Raw Blame History

1 import {Injectable} from '@angular/core';
2 import {User} from '../models/user';
3 import {Resolve, Router, ActivatedRouteSnapshot} from '@angular/router';
4 import { UserService } from '../services/user.service';
5 import { AlertifyService } from '../services/alertify.service';
6 import { Observable, of } from 'rxjs';
7 import { catchError } from 'rxjs/operators';
8
9 @Injectable()
10 export class MemberListResolver implements Resolve<User[]> {
11     constructor(private userService: UserService, private router: Router,
12                 private alertify: AlertifyService) {}
13
14     resolve(route: ActivatedRouteSnapshot): Observable<User[]> {
15         return this.userService.getUsers().pipe(
16             catchError(error => {
17                 this.alertify.error('Problem retrieving data');
18                 this.router.navigate(['/home']);
19                 return of(null);
20             })
21         );
22     }
23 }
```

both the following files need to be updated:

member-list.resolver.ts ([link](#))

member-detail.resolver.ts ([link](#))

Module 10 Changes

member-edit.resolver.ts is updated to use the new RxJS import and syntax. The ‘catch’ operator has been renamed to ‘catchError’ and now needs to be defined inside the ‘pipe’ method from RxJS. So the resolver now looks as follows ([link](#)):

```
25 lines (22 sloc) | 1014 Bytes
Raw Blame History

1 import {Injectable} from '@angular/core';
2 import {User} from '../models/user';
3 import {Resolve, Router, ActivatedRouteSnapshot} from '@angular/router';
4 import { UserService } from '../services/user.service';
5 import { AlertifyService } from '../services/alertify.service';
6 import { Observable, of } from 'rxjs';
7 import { catchError } from 'rxjs/operators';
8 import { AuthService } from '../services/auth.service';
9
10 @Injectable()
11 export class MemberEditResolver implements Resolve<User> {
12     constructor(private userService: UserService, private router: Router,
13                 private alertify: AlertifyService, private authService: AuthService) {}
14
15     resolve(route: ActivatedRouteSnapshot): Observable<User> {
16         return this.userService.getUser(this.authService.decodedToken.nameid).pipe(
17             catchError(error => {
18                 this.alertify.error('Problem retrieving your data');
19                 this.router.navigate(['/members']);
20                 return of(null);
21             })
22         );
23     }
24 }
```

member-edit.component.html ([link](#))

member-edit.component.css ([link](#))

This above files have been updated for Bootstrap 4.

Also, updated for this version of the course is not something specific to Angular 6, but is something that has come up in the Q&A about how to prevent a user from losing their changes if they quit the browser. This can be accomplished via the `HostListener` decorator as follows ([link](#)):

```
6 import { UserService } from '../_services/user.service';
7 import { AuthService } from '../_services/auth.service';
8
9 @Component({
10   selector: 'app-member-edit',
11   templateUrl: './member-edit.component.html',
12   styleUrls: ['./member-edit.component.css']
13 })
14 export class MemberEditComponent implements OnInit {
15   @ViewChild('editForm') editForm: NgForm;
16   user: User;
17   @HostListener('window:beforeunload', ['$event'])
18   unloadNotification($event: any) {
19     if (this.editForm.dirty) {
20       $event.returnValue = true;
21     }
22   }
23 }
```

Module 11 Changes

The **photo-editor.component.html** ([link](#)) has been updated to use Bootstrap 4 classes.

Previously in the API for the CORS settings we had the following configuration:

```
// seeder.SeedUsers();
app.UseCors(x => x.AllowAnyOrigin().AllowAnyHeader().AllowAnyMethod().AllowCredentials())
```

We don't need the `AllowCredentials` as this is a setting to allow cookie authentication to be sent to the server, we are not using this type of authentication. This was needed to get around an issue with NG2 File upload. However there is a better solution to this and the following can be added to the `photo.editor.component.ts` ([link](#))

```

initializeUploader() {
  this.uploader = new FileUploader({
    url: this.baseUrl + 'users/' + this.authService.decodedToken.nameid + '/photos',
    authToken: 'Bearer ' + localStorage.getItem('token'),
    allowedFileType: ['image'],
    removeAfterUpload: true,
    autoUpload: false,
    maxFileSize: 10 * 1024 * 1024
  });

  this.uploader.onAfterAddingFile = (file) => {file.withCredentials = false; };

  this.uploader.onSuccessItem = (item, response, status, headers) => {
    if (response) {

```

The dependency on underscore.js has been removed. This has been replaced by usage of the array Filter method and splice method instead in this component. The code for the setMainPhoto and Delete method is now ([link](#)):

```

59  setMainPhoto(photo: Photo) {
60    this.userService.setMainPhoto(this.authService.decodedToken.nameid, photo.id).subscribe(() => {
61      this.currentMain = this.photos.filter(p => p.isMain === true)[0];
62      this.currentMain.isMain = false;
63      photo.isMain = true;
64      this.authService.changeMemberPhoto(photo.url);
65      this.authService.currentUser.photoUrl = photo.url;
66      localStorage.setItem('user', JSON.stringify(this.authService.currentUser));
67    }, error => {
68      this.alertify.error(error);
69    });
70  }
71
72  deletePhoto(id: number) {
73    this.alertify.confirm('Are you sure you want to delete this photo?', () => {
74      this.userService.deletePhoto(this.authService.decodedToken.nameid, id).subscribe(() => {
75        this.photos.splice(this.photos.findIndex(p => p.id === id), 1);
76        this.alertify.success('Photo has been deleted');
77      }, error => {
78        this.alertify.error('Failed to delete the photo');
79      });
80    });
81  }

```

The other files updated in this module are:

nav.component.html ([link](#)) – Just adding the member photo to the nav.

Module 12 Changes

The HTML in the **member-edit.component.html** ([link](#)) has been updated to use Bootstrap 4 ([link](#))

Module 13 Changes

No changes

Module 14 Changes

user.service.ts ([link](#))

getUsers method updated to use the httpClient module and instead of adding querystring directly onto the Url the query string parameters are passed in a HttpParams object as follows:

```
getUsers(page?, itemsPerPage?, userParams?): Observable<PaginatedResult<User[]>> {  
    const paginatedResult: PaginatedResult<User[]> = new PaginatedResult<User[]>();  
  
    let params = new HttpParams();  
  
    if (page != null && itemsPerPage != null) {  
        params = params.append('pageNumber', page);  
        params = params.append('pageSize', itemsPerPage);  
    }  
  
    if (userParams != null) {  
        params = params.append('minAge', userParams.minAge);  
        params = params.append('maxAge', userParams.maxAge);  
        params = params.append('gender', userParams.gender);  
        params = params.append('orderBy', userParams.orderBy);  
    }  
  
    return this.http.get<User[]>(this.baseUrl + 'users', { observe: 'response', params })  
        .pipe(  
            map(response => {  
                paginatedResult.result = response.body;  
                if (response.headers.get('Pagination') != null) {  
                    paginatedResult.pagination = JSON.parse(response.headers.get('Pagination'))  
                }  
                return paginatedResult;  
            })  
        );  
}
```

member-list.resolver.ts ([link](#))

No major changes, but the RxJs pipe method is used in this file.

member-list.component.html ([link](#))

The member-list.component.html file is updated to use Bootstrap 4 classes for pagination alignment on the page:

```
53 <div class="d-flex justify-content-center">
54   <pagination
55     [boundaryLinks]="true"
56     [totalItems]="pagination.totalItems"
57     [(ngModel)]="pagination.currentPage"
58     [itemsPerPage]="pagination.itemsPerPage"
59     (pageChanged)="pageChanged($event)"
60     previousText="&lsaquo;" nextText="&rsaquo;" firstText="&laquo;" lastText="&raquo;">
61
62   </pagination>
63 </div>
```

Module 15 Changes

The following elements are updated:

lists.resolver.ts ([link](#)) - updated to use RxJs 6 methods.

lists.component.html ([link](#)) – updated to use Bootstrap 4 classes.

Module 16 Changes

user.service.ts ([link](#)) – updated to use HttpClientModule and RxJS methods

messages.component.html ([link](#)) – updated for bootstrap 4

member-messages.component.html ([link](#)) – updated for bootstrap 4

member-messages.component.css ([link](#)) – updated for bootstrap 4

member-messages.component.ts ([link](#)) – updated loadMessages() method to use the ‘tap’ rxjs operator.