**Running the Angular 2/4 JWT Login Example Locally**

1. Install NodeJS (> v4) and NPM (> v3) from <https://nodejs.org/en/download/>, you can check the versions you have installed by running node -v and npm -v from the command line.
2. Download the project source code from <https://github.com/cornflourblue/angular2-jwt-authentication-example>
3. Install all required npm packages by running npm install from the command line in the project root folder (where the package.json is located).
4. Start the application by running npm start from the command line in the project root folder.

**Angular 2/4 Project Structure**

I used the [Angular 2 quickstart](https://angular.io/docs/ts/latest/quickstart.html) project as a base for the application, it's written in TypeScript and uses systemjs for loading modules. If you're new to angular 2 I'd recommend checking out the quickstart as it provides details on the project tooling and configuration files which aren't covered in this post.

The project and code structure mostly follows the recommendations in the official [Angular 2 style guide](https://angular.io/styleguide), with my own tweaks here and there.

Each feature has it's own folder (home & login), other code such as services, models, guards etc are placed in folders prefixed with an underscore to easily differentiate them and group them together at the top of the folder structure.

Here's the project structure:

* app
  + \_guards
    - [auth.guard.ts](http://jasonwatmore.com/#auth-guard-ts)
    - index.ts
  + \_helpers
    - [fake-backend.ts](http://jasonwatmore.com/#fake-backend-ts)
    - index.ts
  + \_models
    - [user.ts](http://jasonwatmore.com/#user-ts)
    - index.ts
  + \_services
    - [authentication.service.ts](http://jasonwatmore.com/#authentication-service-ts)
    - index.ts
    - [user.service.ts](http://jasonwatmore.com/#user-service-ts)
  + home
    - [home.component.html](http://jasonwatmore.com/#home-component-html)
    - [home.component.ts](http://jasonwatmore.com/#home-component-ts)
    - index.ts
  + login
    - index.ts
    - [login.component.html](http://jasonwatmore.com/#login-component-html)
    - [login.component.ts](http://jasonwatmore.com/#login-component-ts)
  + [app.component.html](http://jasonwatmore.com/#app-component-html)
  + [app.component.ts](http://jasonwatmore.com/#app-component-ts)
  + [app.module.ts](http://jasonwatmore.com/#app-module-ts)
  + [app.routing.ts](http://jasonwatmore.com/#app-routing-ts)
  + [main.ts](http://jasonwatmore.com/#main-ts)
* app.css
* index.html
* package.json
* system.config.js
* tsconfig.json

Below are brief descriptions and the code for the main files of the example application, all files are available in the github project linked at the top of the post.

**Angular 2/4 Auth Guard**

**Path: /app/\_guards/auth.guard.ts**

The auth guard is used to prevent unauthenticated users from accessing restricted routes, it's used in app.routing.ts to protect the home page route. For more information about angular 2 guards you can check out [this post](http://blog.thoughtram.io/angular/2016/07/18/guards-in-angular-2.html) on the thoughtram blog.

[?](http://jasonwatmore.com/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | import { Injectable } from '@angular/core';  import { Router, CanActivate } from '@angular/router';    @Injectable()  export class AuthGuard implements CanActivate {        constructor(private router: Router) { }        canActivate() {          if (localStorage.getItem('currentUser')) {              // logged in so return true              return true;          }            // not logged in so redirect to login page          this.router.navigate(['/login']);          return false;      }  } |

[Back to top](http://jasonwatmore.com/#projectstructure)

**Angular 2/4 Fake Backend Provider**

**Path: /app/\_helpers/fake-backend.ts**

The fake backend provider enables the example to run without a backend / backendless, I created it so I could focus the example and tutorial just on the angular 2 code, and also so it works on Plunker.

It uses the Angular 2 MockBackend to replace the default backend used by the Http service, the MockBackend enables you to intercept http requests made within the application and provide fake responses, it's also used for unit testing.

[?](http://jasonwatmore.com/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57 | import { Http, BaseRequestOptions, Response, ResponseOptions, RequestMethod } from '@angular/http';  import { MockBackend, MockConnection } from '@angular/http/testing';    export function fakeBackendFactory(backend: MockBackend, options: BaseRequestOptions) {      // configure fake backend      backend.connections.subscribe((connection: MockConnection) => {          let testUser = { username: 'test', password: 'test', firstName: 'Test', lastName: 'User' };            // wrap in timeout to simulate server api call          setTimeout(() => {                // fake authenticate api end point              if (connection.request.url.endsWith('/api/authenticate') && connection.request.method === RequestMethod.Post) {                  // get parameters from post request                  let params = JSON.parse(connection.request.getBody());                    // check user credentials and return fake jwt token if valid                  if (params.username === testUser.username && params.password === testUser.password) {                      connection.mockRespond(new Response(                          new ResponseOptions({ status: 200, body: { token: 'fake-jwt-token' } })                      ));                  } else {                      connection.mockRespond(new Response(                          new ResponseOptions({ status: 200 })                      ));                  }              }                // fake users api end point              if (connection.request.url.endsWith('/api/users') && connection.request.method === RequestMethod.Get) {                  // check for fake auth token in header and return test users if valid, this security is implemented server side                  // in a real application                  if (connection.request.headers.get('Authorization') === 'Bearer fake-jwt-token') {                      connection.mockRespond(new Response(                          new ResponseOptions({ status: 200, body: [testUser] })                      ));                  } else {                      // return 401 not authorised if token is null or invalid                      connection.mockRespond(new Response(                          new ResponseOptions({ status: 401 })                      ));                  }              }            }, 500);        });        return new Http(backend, options);  }    export let fakeBackendProvider = {      // use fake backend in place of Http service for backend-less development      provide: Http,      useFactory: fakeBackendFactory,      deps: [MockBackend, BaseRequestOptions]  }; |

[Back to top](http://jasonwatmore.com/#projectstructure)

**Angular 2/4 User Model**

**Path: /app/\_models/user.ts**

The user model is a small class that defines the properties of a user.

[?](http://jasonwatmore.com/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | export class User {      username: string;      password: string;      firstName: string;      lastName: string;  } |

[Back to top](http://jasonwatmore.com/#projectstructure)

**Angular 2/4 JWT Authentication Service**

**Path: /app/\_services/authentication.service.ts**

The JWT authentication service is used to login and logout of the application, to login it posts the users credentials to the api and checks the response for a JWT token, if there is one it means authentication was successful so the user details are added to local storage and the token saved to the AuthenticationService.token property. The token property is used by other services in the application to set the authorization header of http requests made to secure api endpoints.

The logged in user details are stored in local storage so the user will stay logged in if they refresh the browser and also between browser sessions until they logout. If you don't want the user to stay logged in between refreshes or sessions the behaviour could easily be changed by storing user details somewhere less persistent such as session storage or in a property of the authentication service.

[?](http://jasonwatmore.com/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | import { Injectable } from '@angular/core';  import { Http, Headers, Response } from '@angular/http';  import { Observable } from 'rxjs';  import 'rxjs/add/operator/map'    @Injectable()  export class AuthenticationService {      public token: string;        constructor(private http: Http) {          // set token if saved in local storage          var currentUser = JSON.parse(localStorage.getItem('currentUser'));          this.token = currentUser && currentUser.token;      }        login(username: string, password: string): Observable<boolean> {          return this.http.post('/api/authenticate', JSON.stringify({ username: username, password: password }))              .map((response: Response) => {                  // login successful if there's a jwt token in the response                  let token = response.json() && response.json().token;                  if (token) {                      // set token property                      this.token = token;                        // store username and jwt token in local storage to keep user logged in between page refreshes                      localStorage.setItem('currentUser', JSON.stringify({ username: username, token: token }));                        // return true to indicate successful login                      return true;                  } else {                      // return false to indicate failed login                      return false;                  }              });      }        logout(): void {          // clear token remove user from local storage to log user out          this.token = null;          localStorage.removeItem('currentUser');      }  } |

[Back to top](http://jasonwatmore.com/#projectstructure)

**Angular 2/4 User Service**

**Path: /app/\_services/user.service.ts**

The user service contains a method for getting all users from the api, I included it to demonstrate accessing a secure api endpoint with the http authorization header set after logging in to the application. The secure endpoint in the example is a fake one implemented in the fake backend provider above.

[?](http://jasonwatmore.com/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | import { Injectable } from '@angular/core';  import { Http, Headers, RequestOptions, Response } from '@angular/http';  import { Observable } from 'rxjs';  import 'rxjs/add/operator/map'    import { AuthenticationService } from '../\_services/index';  import { User } from '../\_models/index';    @Injectable()  export class UserService {      constructor(          private http: Http,          private authenticationService: AuthenticationService) {      }        getUsers(): Observable<User[]> {          // add authorization header with jwt token          let headers = new Headers({ 'Authorization': 'Bearer ' + this.authenticationService.token });          let options = new RequestOptions({ headers: headers });            // get users from api          return this.http.get('/api/users', options)              .map((response: Response) => response.json());      }  } |

[Back to top](http://jasonwatmore.com/#projectstructure)

**Angular 2/4 Home Component Template**

**Path: /app/home/home.component.html**

The home component template contains html and angular 2 template syntax for displaying a simple welcome message, a list of users and a logout link.

[?](http://jasonwatmore.com/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <div class="col-md-6 col-md-offset-3">      <h1>Home</h1>      <p>You're logged in with JWT!!</p>      <div>          Users from secure api end point:          <ul>              <li \*ngFor="let user of users">{{user.firstName}} {{user.lastName}}</li>          </ul>      </div>      <p><a [routerLink]="['/login']">Logout</a></p>  </div> |

[Back to top](http://jasonwatmore.com/#projectstructure)

**Angular 2/4 Home Component**

**Path: /app/home/home.component.ts**

The home component defines an angular 2 component that gets all users from the user service and makes them available to the template via the 'users' property.

[?](http://jasonwatmore.com/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | import { Component, OnInit } from '@angular/core';    import { User } from '../\_models/index';  import { UserService } from '../\_services/index';    @Component({      moduleId: module.id,      templateUrl: 'home.component.html'  })    export class HomeComponent implements OnInit {      users: User[] = [];        constructor(private userService: UserService) { }        ngOnInit() {          // get users from secure api end point          this.userService.getUsers()              .subscribe(users => {                  this.users = users;              });      }    } |

[Back to top](http://jasonwatmore.com/#projectstructure)

**Angular 2/4 Login Component Template**

**Path: /app/login/login.component.html**

The login component template contains a login form with username and password fields. It displays validation messages for invalid fields when the submit button is clicked. On form submit the login() method is called.

[?](http://jasonwatmore.com/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | <div class="col-md-6 col-md-offset-3">      <div class="alert alert-info">          Username: test<br />          Password: test      </div>      <h2>Login</h2>      <form name="form" (ngSubmit)="f.form.valid && login()" #f="ngForm" novalidate>          <div class="form-group" [ngClass]="{ 'has-error': f.submitted && !username.valid }">              <label for="username">Username</label>              <input type="text" class="form-control" name="username" [(ngModel)]="model.username" #username="ngModel" required />              <div \*ngIf="f.submitted && !username.valid" class="help-block">Username is required</div>          </div>          <div class="form-group" [ngClass]="{ 'has-error': f.submitted && !password.valid }">              <label for="password">Password</label>              <input type="password" class="form-control" name="password" [(ngModel)]="model.password" #password="ngModel" required />              <div \*ngIf="f.submitted && !password.valid" class="help-block">Password is required</div>          </div>          <div class="form-group">              <button [disabled]="loading" class="btn btn-primary">Login</button>              <img \*ngIf="loading" src="data:image/gif;base64," />          </div>          <div \*ngIf="error" class="alert alert-danger">{{error}}</div>      </form>  </div> |

[Back to top](http://jasonwatmore.com/#projectstructure)

**Angular 2/4 Login Component**

**Path: /app/login/login.component.ts**

The login component uses the authentication service to login and logout of the application. It automatically logs the user out when it initializes (ngOnInit) so the login page can also be used to logout.

[?](http://jasonwatmore.com/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | import { Component, OnInit } from '@angular/core';  import { Router } from '@angular/router';    import { AuthenticationService } from '../\_services/index';    @Component({      moduleId: module.id,      templateUrl: 'login.component.html'  })    export class LoginComponent implements OnInit {      model: any = {};      loading = false;      error = '';        constructor(          private router: Router,          private authenticationService: AuthenticationService) { }        ngOnInit() {          // reset login status          this.authenticationService.logout();      }        login() {          this.loading = true;          this.authenticationService.login(this.model.username, this.model.password)              .subscribe(result => {                  if (result === true) {                      // login successful                      this.router.navigate(['/']);                  } else {                      // login failed                      this.error = 'Username or password is incorrect';                      this.loading = false;                  }              });      }  } |

[Back to top](http://jasonwatmore.com/#projectstructure)

**Angular 2/4 App Component Template**

**Path: /app/app.component.html**

The app component template is the root component template of the application, it contains a router-outlet directive for displaying the contents of each view based on the current route / path.

[?](http://jasonwatmore.com/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <!-- main app container -->  <div class="jumbotron">      <div class="container">          <div class="col-sm-8 col-sm-offset-2">              <router-outlet></router-outlet>          </div>      </div>  </div> |

[Back to top](http://jasonwatmore.com/#projectstructure)

**Angular 2/4 App Component**

**Path: /app/app.component.ts**

The app component is the root component of the application, it defines the root tag of the app as <app></app> with the selector property.

The moduleId property is set to allow a relative path to be used for the templateUrl.

[?](http://jasonwatmore.com/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | import { Component } from '@angular/core';    @Component({      moduleId: module.id,      selector: 'app',      templateUrl: 'app.component.html'  })    export class AppComponent { } |

[Back to top](http://jasonwatmore.com/#projectstructure)

**Angular 2/4 App Module**

**Path: /app/app.module.ts**

The app module defines the root module of the application along with metadata about the module. For more info about angular 2 modules check out [this page](https://angular.io/docs/ts/latest/guide/ngmodule.html) on the official docs site.

This is where the fake backend provider is added to the application, to switch to a real backend simply remove the providers located under the comment "// providers used to create fake backend".

[?](http://jasonwatmore.com/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | import { NgModule }      from '@angular/core';  import { BrowserModule } from '@angular/platform-browser';  import { FormsModule }    from '@angular/forms';  import { HttpModule } from '@angular/http';    // used to create fake backend  import { fakeBackendProvider } from './\_helpers/index';  import { MockBackend, MockConnection } from '@angular/http/testing';  import { BaseRequestOptions } from '@angular/http';    import { AppComponent }  from './app.component';  import { routing }        from './app.routing';    import { AuthGuard } from './\_guards/index';  import { AuthenticationService, UserService } from './\_services/index';  import { LoginComponent } from './login/index';  import { HomeComponent } from './home/index';    @NgModule({      imports: [          BrowserModule,          FormsModule,          HttpModule,          routing      ],      declarations: [          AppComponent,          LoginComponent,          HomeComponent      ],      providers: [          AuthGuard,          AuthenticationService,          UserService,            // providers used to create fake backend          fakeBackendProvider,          MockBackend,          BaseRequestOptions      ],      bootstrap: [AppComponent]  })    export class AppModule { } |

[Back to top](http://jasonwatmore.com/#projectstructure)

**Angular 2/4 App Routing**

**Path: /app/app.routing.ts**

The app routing file defines the routes of the application, each route contains a path and associated component. The home route is secured by passing the AuthGuard to the canActivate property of the route.

[?](http://jasonwatmore.com/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | import { Routes, RouterModule } from '@angular/router';    import { LoginComponent } from './login/index';  import { HomeComponent } from './home/index';  import { AuthGuard } from './\_guards/index';    const appRoutes: Routes = [      { path: 'login', component: LoginComponent },      { path: '', component: HomeComponent, canActivate: [AuthGuard] },        // otherwise redirect to home      { path: '\*\*', redirectTo: '' }  ];    export const routing = RouterModule.forRoot(appRoutes); |

[Back to top](http://jasonwatmore.com/#projectstructure)

**Angular 2/4 Main (Bootstrap) File**

**Path: /app/main.ts**

The main file is the entry point used by angular to launch and bootstrap the application.

[?](http://jasonwatmore.com/)

|  |  |
| --- | --- |
| 1  2  3  4  5 | import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';    import { AppModule } from './app.module';    platformBrowserDynamic().bootstrapModule(AppModule); |

[Back to top](http://jasonwatmore.com/#projectstructure)

**Recommended Books on Angular 2/4**

* [Angular 2 Cookbook - Second Edition](https://www.amazon.com/gp/product/1785881922/ref=as_li_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=1785881922&linkCode=as2&tag=jasonwatmore-20&linkId=62e30a6d3d356a2ae92c6e1ac85f9ba4) by Matt Frisbie (a developer at Google)http://ir-na.amazon-adsystem.com/e/ir?t=jasonwatmore-20&l=am2&o=1&a=1785881922
* [Angular 2 Development with TypeScript](https://www.amazon.com/gp/product/1617293121/ref=as_li_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=1617293121&linkCode=as2&tag=jasonwatmore-20&linkId=9733d68da7d94b5334324509b264c25e)http://ir-na.amazon-adsystem.com/e/ir?t=jasonwatmore-20&l=am2&o=1&a=1617293121 by Yakov Fain and Anton Moiseev