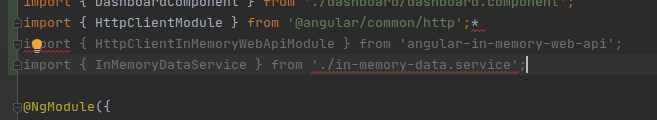
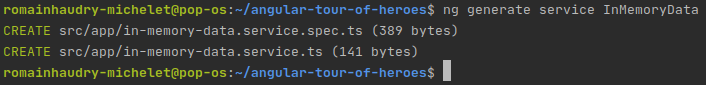


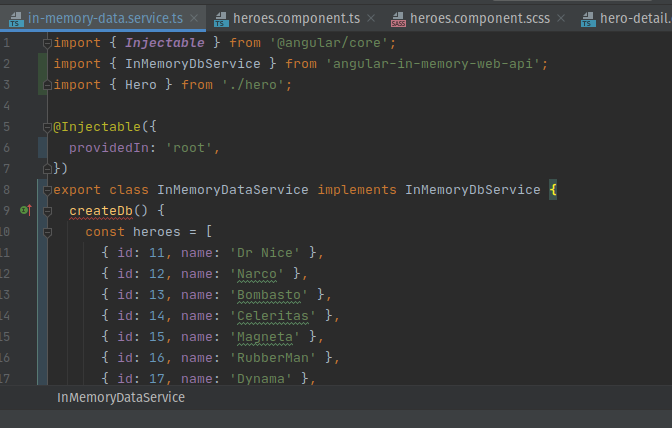
Importation API WEB en mémoire :



Création du In Memory Data



Mise a jour des fichiers :



Mise a jour Hero.service.ts

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

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Observable, of } from 'rxjs';

import { Hero } from './hero';

import { ***HEROES*** } from './mock-heroes';

import { MessageService } from './message.service';

@Injectable({

providedIn: 'root',

})

export class HeroService {

constructor(

private http: HttpClient,

private messageService: MessageService) { }

private heroesUrl = 'api/heroes'; // URL to web api

getHeroes(): Observable<Hero[]> {

// *TODO: send the message \_after\_ fetching the heroes*

this.messageService.add('HeroService: fetched heroes');

return of(***HEROES***);

}

getHero(id: number): Observable<Hero> {

// *TODO: send the message \_after\_ fetching the hero*

this.messageService.add(`HeroService: fetched hero id=${id}`);

return of(***HEROES***.find(hero => hero.id === id));

}

private log(message: string) {

this.messageService.add(`HeroService: ${message}`);

}

}

Mise a jour 2 :

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

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Observable, of } from 'rxjs';

import { catchError, map, tap } from 'rxjs/operators';

import { Hero } from './hero';

import { ***HEROES*** } from './mock-heroes';

import { MessageService } from './message.service';

@Injectable({

providedIn: 'root',

})

export class HeroService {

constructor(

private http: HttpClient,

private messageService: MessageService) { }

private heroesUrl = 'api/heroes'; // URL to web api

*/\*\* GET heroes from the server \*/*

getHeroes(): Observable<Hero[]> {

return this.http.get<Hero[]>(this.heroesUrl)

.pipe(

tap(\_ => this.log('fetched heroes')),

catchError(this.handleError<Hero[]>('getHeroes', []))

);

}

getHero(id: number): Observable<Hero> {

// *TODO: send the message \_after\_ fetching the hero*

this.messageService.add(`HeroService: fetched hero id=${id}`);

return of(***HEROES***.find(hero => hero.id === id));

}

// tslint:disable-next-line:typedef

private log(message: string) {

this.messageService.add(`HeroService: ${message}`);

}

*/\*\**

*\* Handle Http operation that failed.*

*\* Let the app continue.*

*\** ***@param*** *operation - name of the operation that failed*

*\** ***@param*** *result - optional value to return as the observable result*

*\*/*

// tslint:disable-next-line:typedef

private handleError<T>(operation = 'operation', result?: T) {

return (error: any): Observable<T> => {

// *TODO: send the error to remote logging infrastructure*

***console***.error(error); // log to console instead

// *TODO: better job of transforming error for user consumption*

this.log(`${operation} failed: ${error.message}`);

// Let the app keep running by returning an empty result.

return of(result as T);

};

}

}

on modifie :

/\*\* GET hero by id. Will 404 if id not found \*/

getHero(id: number): Observable<Hero> {

const url = `${this.heroesUrl}/${id}`;

return this.http.get<Hero>(url).pipe(

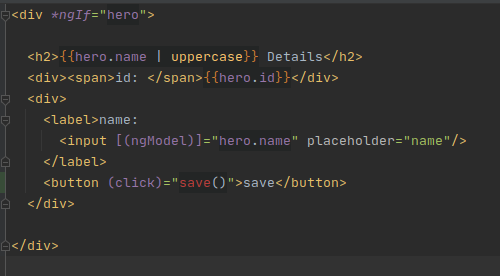
tap(\_ => this.log(`fetched hero id=${id}`)),

catchError(this.handleError<Hero>(`getHero id=${id}`))

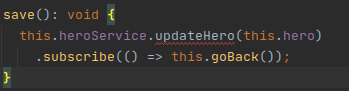
);

}

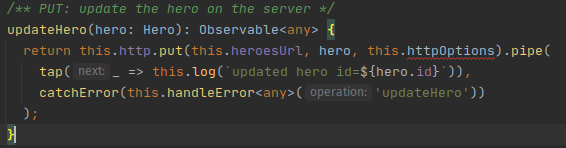
Ajout du bouton save :



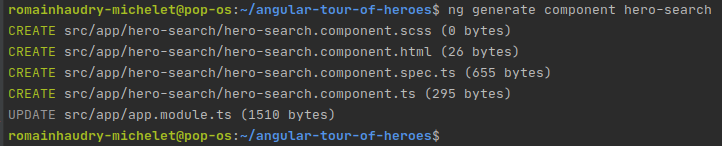
Ajout de la fonction save :



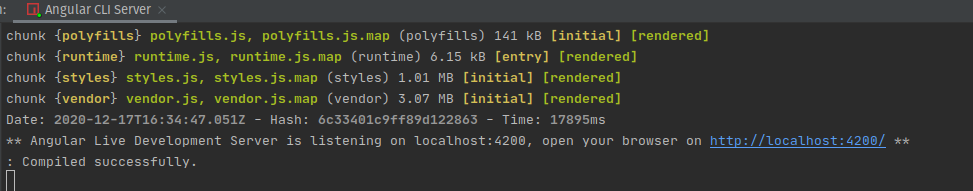
Ajout de hero update :



Creation du component hero search

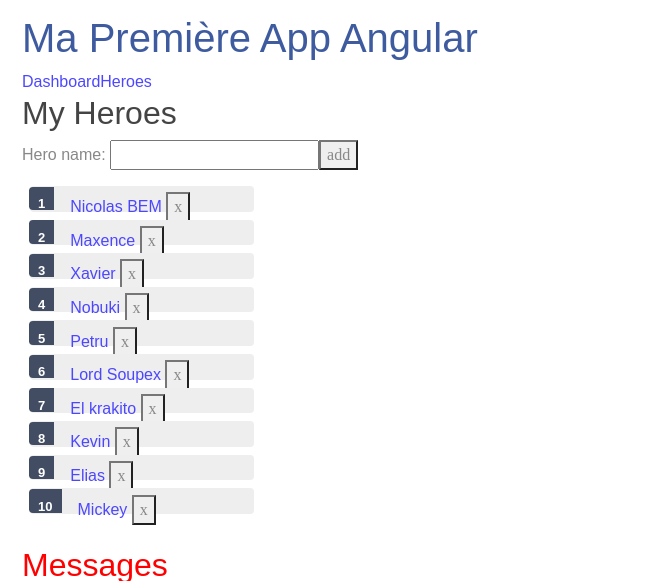


Après quelques corrections :



Compilation sans soucis

Test de l’application :



Tout fonctionne parfaitement,

Il est possible que par la suite j’améliore le SCSS.

Fin du TP 5h48,début 13h30