Guillaume Verfaille

Rapport de laboratoire : Application web avancées, Angular JS 2 et Symfony

Fonctionnement

L'application est constituée de deux grosses parties, le front-end réalisé avec AngularJs 2 et le back-end réalisé avec Symfony. Au lancement de l'application, l'utilisateur arrive sur la vue « Dashboard ». Cette vue est composée de plusieurs composant qui permettent chacun d'avoir une interaction avec l'application. Le premier composant permet de navigué entre la vue « Dashboard » et la vue « Auteurs » ensuite un autre composant permet à l'utilisateur d'accéder plus rapidement aux détails des auteurs affichés. Si l'utilisateur clique sur le bouton « Auteurs », il accède à la vue « Auteurs » qui est composée du composant de navigation entre les deux vues principales et d'un composant qui affiche la liste complète des Auteurs. L'utilisateur peut alors ajouter un auteur, en supprimer un ou accéder au détail d'un auteur. Quand l'utilisateur accède au détail d'un auteur, il peut modifier les informations concernant l'auteur en question. Du coté back-end, l'application envoi des requêtes qui sont reçue et qui reçoivent une réponse grâce au contrôleur. Les informations concernant les auteurs sont stockées dans la base de données.

Le code

Coté serveur(back-end) :

- L'entité : elle sert à définir la définition d'un auteur du coté serveur. Dans le cas présent, il définit qu'un auteur est définit par un id, un nom et un mail.

```php
<?php
// Cette entité déclare la classe Auteur du coté serveur de l'application
namespace ListeAuteurBundle\Entity;

/**
 * Auteur
 */
class Auteur
{
    /**
     * @var int
     */
    private $id;

    /**
     * @var string
     */
    private $nom;

    /**
     * @var string
     */
    private $mail;
```

```php
    /**
     * Get id
     *
     * @return int
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set nom
     *
     * @param string $nom
     *
     * @return Auteur
     */
    public function setNom($nom)
    {
        $this->nom = $nom;

        return $this;
    }

    /**
     * Get nom
     *
     * @return string
     */
    public function getNom()
    {
        return $this->nom;
    }

    /**
     * Set mail
     *
     * @param string $mail
     *
     * @return Auteur
     */
    public function setMail($adresse)
    {
        $this->mail = $adresse;

        return $this;
    }

    /**
     * Get adresse
     *
     * @return string
     */
    public function getMail()
    {
        return $this->mail;
    }
}
```

- Le controller : son rôle est de décrire les actions à effectuer lorsqu'on lui fait une requête. GET,POST, PUT, DELETE.

```php
<?php

// src/ListeAuteurBundle/Controller/AdvertController.php

namespace ListeAuteurBundle\Controller;
use ListeAuteurBundle\Entity\Auteur;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\HttpFoundation\Request;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Method;

class AdvertController extends Controller
{
    public function indexAction()
    {
        $content = $this->get('templating')->render('ListeAuteurBundle:Advert:index.html.twig');
        return new Response($content);
        $Auteurs = new Auteur;
    }
     /**
     * @Route("/api/auteur")
     * @Method("GET")
     */
     //cette fonction est la fonction qui est appelée lorsque l'API reçoit une requête GET
    public function getAction()
    {
        //récupération de tous les auteurs dans la variable $auteurs
        $em = $this->getDoctrine()->getManager();
        $auteurs = $em->getRepository('ListeAuteurBundle:Auteur')->findAll();
        //Construction et envoie de la réponse
        $results = array();
        foreach ($auteurs as $auteur) {
            $results[] = array(
                'id' => $auteur->getId(),
                'nom' => $auteur->getNom(),
                'mail' => $auteur->getMail()
            );
        }

        return new JsonResponse($results);
    }
    /**
     * @Route("/api/auteur/{id}")
     * @Method("GET")
     */
     //cette fonction est la fonction qui est appelée lorsque l'API reçoit une requête GET
    public function getAuteurAction($id)
    {
        //récupération de l'auteur dont l'ID est $id dans la variable $auteur
        $em = $this->getDoctrine()->getManager();
```

```
        $auteur = $em->getRepository('ListeAuteurBundle:Auteur')-
>findOneById($id);
        //Construction et envoie de la réponse
        $result = array();
        if ($auteur) {
            $result = array(
                'id' => $auteur->getId(),
                'nom' => $auteur->getNom(),
                'mail' => $auteur->getMail()
            );
        }

        return new JsonResponse($result);
    }

     /**
     * @Route("/api/auteur")
     * @Method("POST")
     */
     //cette fonction est la fonction qui est appelée lorsque l'API reçoit
une  POST
    public function postAction(Request $request)
    {
        //récupération du nom et du mail de l'auteur reçu via la requête
        $nom = $request->get('nom');
        $mail = $request->get('mail');
        //Création de l'auteur avec son nom et de son mail
        $auteur = new Auteur();
        $auteur->setNom($nom)
            ->setMail($mail);
        //stockage de l'auteur dans la base de données
        $em = $this->getDoctrine()->getManager();
        $em->persist($auteur);
        $em->flush();
        //Construction et envoie de la réponse
        return new JsonResponse(array(
            'id' => $auteur->getId(),
            'nom' => $auteur->getNom(),
            'mail' => $auteur->getMail()
        ));
    }

    /**
     * @Route("/api/auteur/{id}")
     * @Method("DELETE")
     */
    //cette fonction est la fonction qui est appelée lorsque l'API reçoit
une requête DELETE
    public function delAction($id)
    {
        //Récupération de l'auteur dont l'ID est $id dans la variable
$auteur
        $em = $this->getDoctrine()->getManager();
        $auteur = $em->getRepository('ListeAuteurBundle:Auteur')-
>findOneById($id);
        $em->remove($auteur);//Suppression des données de l'auteur $auteur
        $em->flush();
        //Construction et envoie de la réponse
        return new JsonResponse(array());
    }
```

```
    /**
     * @Route("/api/auteur")
     * @Method("PUT")
     */
    //cette fonction est la fonction qui est appelée lorsque l'API reçoit
une requête PUT
    public function putAction(Request $request)
    {
        //récupération de l'id, du nom et du mail de l'auteur reçu via la
requête
        $id = $request->get('id');
        $nom = $request->get('nom');
        $mail = $request->get('mail');
        //Modification du nom et du mail de l'auteur dont l'id est $id
        $em = $this->getDoctrine()->getManager();
        $auteur = $em->getRepository('ListeAuteurBundle:Auteur')-
>findOneById($id);
        $auteur->setNom($nom)
            ->setMail($mail);
        $em->flush();
        //Construction et envoie de la réponse
        return new JsonResponse(array(
            'id' => $auteur->getId(),
            'nom' => $auteur->getNom(),
            'mail' => $auteur->getMail()
        ));
    }

}
```

- Le routage : ce fichier définit que l'accès au chemin /auteur envoi vers le controller.

```
liste_auteur_homepage:
    path:     /
    defaults: { _controller: ListeAuteurBundle:Default:index }


hello_the_world:
    path:     /hello-world
    defaults: { _controller: ListeAuteurBundle:Advert:index }

auteur:
    path:     /auteur
    defaults: {_controller: ListeAuteurBundle:Advert:index }
```

Coté utilisateur(front-end) :

- Index.html : Il s'agit de la première page que l'utilisateur voit. Le reste de l'application viendra modifier le contenu de cette page dans la balise <my-app>

```
<!DOCTYPE html>
<html>
  <head>
    <script>document.write('<base href="' + document.location + '"
/>');</script>
```

```
    <title>Gestion des auteurs</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link rel="stylesheet" href="styles.css">

    <!-- Polyfills for older browsers -->
    <script src="https://unpkg.com/core-js/client/shim.min.js"></script>

    <script src="https://unpkg.com/zone.js@0.6.25?main=browser"></script>
    <script src="https://unpkg.com/reflect-metadata@0.1.8"></script>
    <script
src="https://unpkg.com/systemjs@0.19.39/dist/system.src.js"></script>

    <script
src="https://cdn.rawgit.com/angular/angular.io/b3c65a9/public/docs/_examples/_boilerplate/systemjs.config.web.js"></script>
    <script>
      System.import('app').catch(function(err){ console.error(err); });
    </script>
  </head>

  <body>
    <my-app><a>Chargement...</a></my-app>
  </body>
</html>


<!--
Copyright 2016 Google Inc. All Rights Reserved.
Use of this source code is governed by an MIT-style license that
can be found in the LICENSE file at http://angular.io/license
-->
```

- App.module.ts : Ce script permet de définir les modules et les composants qui vont être utilisé par l'application.

```
import './rxjs-extensions';

import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule }   from '@angular/forms';
import { HttpModule }    from '@angular/http';

import { AppRoutingModule } from './app-routing.module';

// Imports for loading & configuring the in-memory web api
import { InMemoryWebApiModule } from 'angular-in-memory-web-api';
import { InMemoryDataService }  from './in-memory-data.service';

import { AppComponent }         from './app.component';
import { DashboardComponent }   from './dashboard.component';
import { HeroesComponent }      from './heroes.component';
import { HeroDetailComponent }  from './hero-detail.component';
import { HeroService }          from './hero.service';


@NgModule({
  imports: [
```

```
    BrowserModule,
    FormsModule,
    HttpModule,
    AppRoutingModule
  ],
  declarations: [
    AppComponent,
    DashboardComponent,
    HeroDetailComponent,
    HeroesComponent
  ],
  providers: [ HeroService ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }


/*
Copyright 2016 Google Inc. All Rights Reserved.
Use of this source code is governed by an MIT-style license that
can be found in the LICENSE file at http://angular.io/license
*/
```

- App.component.ts : voici le composant qui gère l'affichage du titre et des deux boutons (Menu et Auteurs)

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

@Component({
  moduleId: module.id,
  selector: 'my-app',
  template: `
    <h1>{{title}}</h1>
    <nav>
      <a routerLink="/dashboard" routerLinkActive="active">Menu</a>
      <a routerLink="/heroes" routerLinkActive="active">Auteurs</a>
    </nav>
    <router-outlet></router-outlet>
  `,
  styleUrls: ['app.component.css']
})
export class AppComponent {
  title = 'Gestion des auteurs';
}


/*
Copyright 2016 Google Inc. All Rights Reserved.
Use of this source code is governed by an MIT-style license that
can be found in the LICENSE file at http://angular.io/license
*/
```

- Dashboard.component.ts : Ce composant gère l'affichage des utilisateurs mis en valeur quand on arrive sur la première page de l'application.

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

import { Hero }        from './hero';
import { HeroService } from './hero.service';

@Component({
  moduleId: module.id,
  selector: 'my-dashboard',
  templateUrl: 'dashboard.component.html',
  styleUrls: [ 'dashboard.component.css' ]
})
export class DashboardComponent implements OnInit {
  heroes: Hero[] = [];

  constructor(private heroService: HeroService) { }

  ngOnInit(): void {
    this.heroService.getHeroes()
      .then(heroes => this.heroes = heroes.slice(1, 5));
  }
}


/*
Copyright 2016 Google Inc. All Rights Reserved.
Use of this source code is governed by an MIT-style license that
can be found in the LICENSE file at http://angular.io/license
*/
```

- Hero.ts : cette classe définit l'auteur du coté client.

```
export class Hero {
  id: number;
  nom: string;
  mail: string;
}

/*
Copyright 2016 Google Inc. All Rights Reserved.
Use of this source code is governed by an MIT-style license that
can be found in the LICENSE file at http://angular.io/license
*/
```

- Hero.service.ts : Ce script définit les fonctions qui concernent les auteurs. C'est dans ce script que l'on retrouve les requêtes vers l'API

```
import { Injectable }     from '@angular/core';
import { Headers, Http, URLSearchParams } from '@angular/http';
```

```
import 'rxjs/add/operator/toPromise';

import { Hero } from './hero';

@Injectable()
export class HeroService {

  private headers = new Headers({'Content-Type': 'application/x-www-form-
urlencoded'});
  private heroesUrl =
'http://localhost/my_project/web/app_dev.php/api/auteur';  // URL vers
l'API de symfony

  constructor(private http: Http) { }

  getHeroes(): Promise<Hero[]> { // cette fonction fait une requète GET
pour récupérer tous les auteurs
        return this.http
            .get(this.heroesUrl)
            .toPromise()
            .then(response => response.json() as Hero)
            .catch(this.handleError);
    }

  getHero(id: number): Promise<Hero> { // cette fonction fait une requète
GET en préçisant l'id de l'auteur que l'on souhaite récupérer
    const url = `${this.heroesUrl}/${id}`;
  return this.http.get(url)
    .toPromise()
    .then(response => response.json() as Hero)
    .catch(this.handleError);
  }

  delete(id: number): Promise<void> { //Cette fonction envoie une requète
DELETE pour effacer les informations concernant l'auteur ciblé par son id
    const url = `${this.heroesUrl}/${id}`;
    return this.http.delete(url, {headers: this.headers})
      .toPromise()
      .then(() => null)
      .catch(this.handleError);
  }

  create(nom: string,mail: string): Promise<Hero> { // Cette fonction envoi
une requète POST qui permet de créer un ateur
    let form = new URLSearchParams();

        form.set('nom', nom);
        form.set('mail', mail);
        return this.http
            .post(this.heroesUrl, form.toString(), {headers: this.headers})
            .toPromise()
            .then(res => res.json())
            .catch(this.handleError);
  }

  update(hero: Hero): Promise<Hero> { // Cette fonction envoi une requète
PUT qui permet de modifier les informations concernant un auteur
    let form = new URLSearchParams();

        form.set('id', hero.id.toString());
```

```
        form.set('nom', hero.nom);
        form.set('mail', hero.mail);

        return this.http
            .put(this.heroesUrl, form.toString(), {headers: this.headers})
            .toPromise()
            .then(res => res.json())
            .catch(this.handleError);
  }

  private handleError(error: any): Promise<any> {
    console.error('ERROR', error); // for demo purposes only
    return Promise.reject(error.message || error);
  }
}

/*
Copyright 2016 Google Inc. All Rights Reserved.
Use of this source code is governed by an MIT-style license that
can be found in the LICENSE file at http://angular.io/license
*/
```

- Hero-detail.component.ts : Ce composant gère l'affichage de la vue détaillé des auteurs.

```
import 'rxjs/add/operator/switchMap';
import { Component, OnInit }      from '@angular/core';
import { ActivatedRoute, Params } from '@angular/router';
import { Location }               from '@angular/common';

import { Hero }        from './hero';
import { HeroService } from './hero.service';
@Component({
  moduleId: module.id,
  selector: 'my-hero-detail',
  templateUrl: 'hero-detail.component.html',
  styleUrls: [ 'hero-detail.component.css' ]
})
export class HeroDetailComponent implements OnInit {
  hero: Hero;

  constructor(
    private heroService: HeroService,
    private route: ActivatedRoute,
    private location: Location
  ) {}

  ngOnInit(): void {
    this.route.params
      .switchMap((params: Params) =>
this.heroService.getHero(+params['id']))
      .subscribe(hero => this.hero = hero);
  }

  save(): void {
    this.heroService.update(this.hero)
      .then(() => this.goBack());
  }
```

```
  goBack(): void {
    this.location.back();
  }
}

/*
Copyright 2016 Google Inc. All Rights Reserved.
Use of this source code is governed by an MIT-style license that
can be found in the LICENSE file at http://angular.io/license
*/
```

- Heroes.component.ts : Ce composant gère l'affichage des auteurs dans la vue auteurs.

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

import { Hero }                from './hero';
import { HeroService }         from './hero.service';

@Component({
  moduleId: module.id,
  selector: 'my-heroes',
  templateUrl: 'heroes.component.html',
  styleUrls: [ 'heroes.component.css' ]
})
export class HeroesComponent implements OnInit {
  heroes: Hero[];
  selectedHero: Hero;

  constructor(
    private heroService: HeroService,
    private router: Router) { }

  getHeroes(): void {
    this.heroService
        .getHeroes()
        .then(heroes => this.heroes = heroes);
  }

  add(nom: string, mail: string): void {
    nom = nom.trim();
    if ((!nom)&&(!mail)) { return; }
    this.heroService.create(nom, mail)
      .then(hero => {
        this.heroes.push(hero);
        this.selectedHero = null;
      });
  }

  delete(hero: Hero): void {
    this.heroService
        .delete(hero.id)
        .then(() => {
          this.heroes = this.heroes.filter(h => h !== hero);
          if (this.selectedHero === hero) { this.selectedHero = null; }
        });
```

```
  }

  ngOnInit(): void {
    this.getHeroes();
  }

  onSelect(hero: Hero): void {
    this.selectedHero = hero;
  }

  gotoDetail(): void {
    this.router.navigate(['/detail', this.selectedHero.id]);
  }
}


/*
Copyright 2016 Google Inc. All Rights Reserved.
Use of this source code is governed by an MIT-style license that
can be found in the LICENSE file at http://angular.io/license
*/
```

Capture d'écran

Figure 1 Vue "Dashboard"

Figure 2 Vue "Auteurs"

Figure 3 Allons voir les détails de gigi en cliquant sur gigi puis sur Vue détaillé

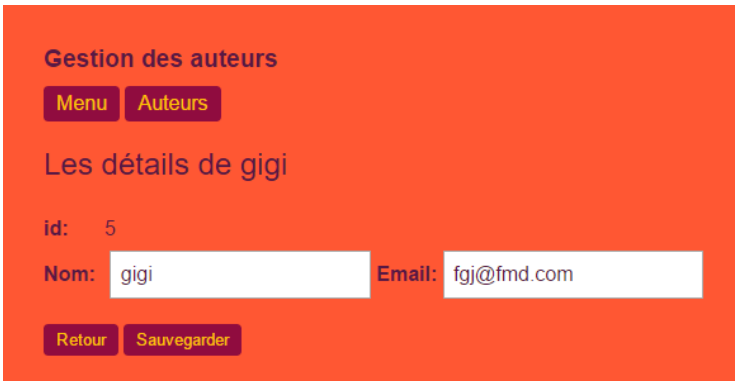

Figure 4 Vue détaillé de gigi

Figure 5 Création d'un auteur

Figure 6 L'auteur à été ajoutée avec succès

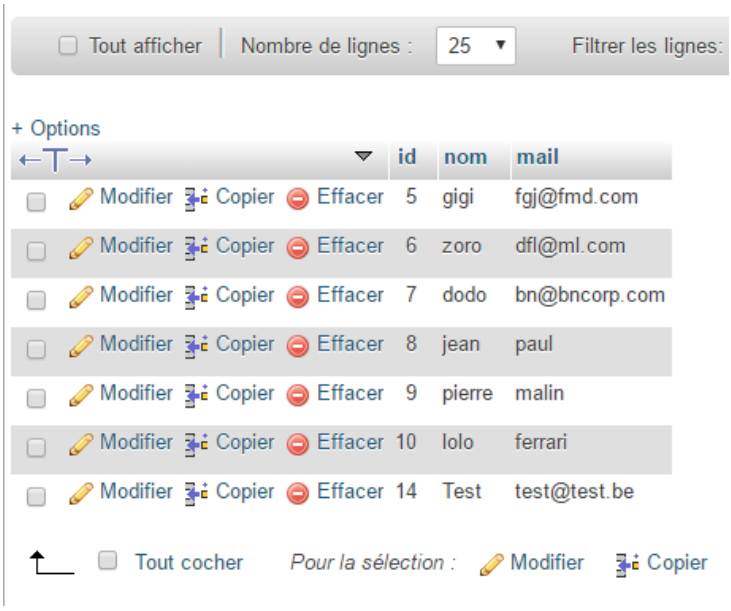

Figure 7 Etat de la base de données avant suppression de l'auteur Test

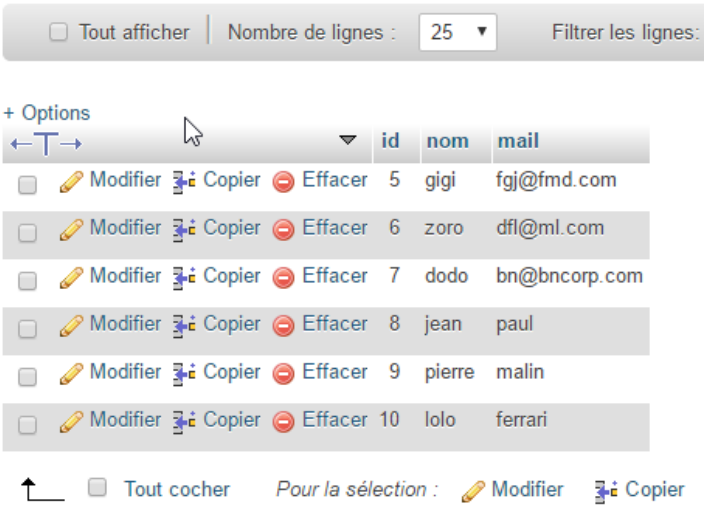

Figure 8 Basse de données après suppression de l'auteur Test

Conclusion

L'application est fonctionnelle et elle utilise bien la configuration front-end back-end demandé. Il reste néanmoins quelques détails à régler. Notamment j'aurai pu ajouté un prénom pour l'auteur. Il faudrait également vérifier les champs de texte lors de la création d'un auteur. Si l'utilisateur actualise sa page, l'application va planter il faut donc la relancé depuis la première page. J'ai essayé sans succès de trouver une solution à ce dernier problème. Pour les autres problèmes, il ne s'agit que d'amélioration et ils ne m'ont pas empêcher d'avoir compris le principe de fonctionnement de ce type d'application.

Lien vers github :

Le projet Angular se trouve à la racine tandis que la partie Symfony se trouve dans le dossier my_project. Et utilise le bundle « ListeAuteurBundle ».

https://github.com/guillaumeVerfaille/Angulars2JSApp