Hajek Valentin TP7 Angular info2

Objectif, nouvelles fonctionnalités :

* Ajout d’une vue DASHBOARD
* Ajoutez la possibilité de naviguer entre les vues Heroes et Dashboard.
* Lorsque les utilisateurs cliquent sur un nom de héros dans l'une des vues, accédez à une vue détaillée du héros sélectionné.
* Lorsque les utilisateurs cliquent sur un lien profond dans un e-mail, ouvrez la vue détaillée d'un héros particulier.

Ajout du appRoutingModule :

Dans src/app :

--flat place le fichier dans src/app au lieu de son propre dossier.

--module=app demande à la CLI de l'enregistrer dans le tableau imports de l'AppModule.

ng generate module app-routing --flat --module=app

nouvelle version module.ts:

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

import { RouterModule, Routes } from '@angular/router';

@NgModule({

exports: [ RouterModule ]

})

export class AppRoutingModule {}

2. On va ajouter des routes :

Path signifie le chemin et composant (crée lors de la navigation vers la route par le routeur)

import { HeroesComponent } from './heroes/heroes.component';

const routes: Routes = [

{ path: 'heroes', component: HeroesComponent }

];

Nous avons porté le HerosComponent pour pouvoir référencer dans un fichier Route. Définit un tableau de routes avec un seul route élément de ce composant

Maintenant le routeur associera cette URL au chemin : ‘heroes’ et affichera le HeroesComponent

3. RouterModule.forRoot()

On configure le rooter à la racine de l’application :

@NgModule({

imports: [ RouterModule.forRoot(routes) ],

4.

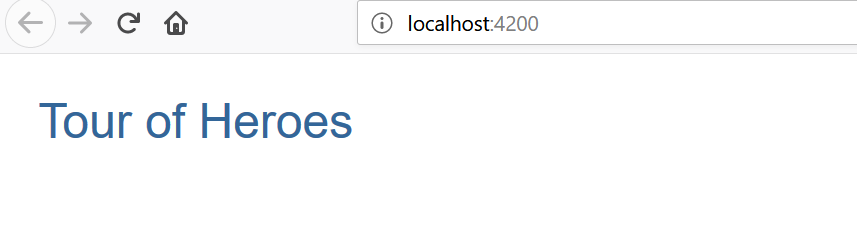
Le <routeur-outlet> indique au routeur où afficher les vues routées.

RouterOutlet est l'une des directives de routeur qui sont devenues disponibles pour AppComponent car AppModule importe le AppRoutingModule qui a exporté RouterModule.

<h1>{{title}}</h1>

<[router-outlet](https://angular.io/api/router/RouterOutlet)></[router-outlet](https://angular.io/api/router/RouterOutlet)>

<app-messages></app-messages>



5.

<nav>

<a routerLink="/heroes">Heroes</a>

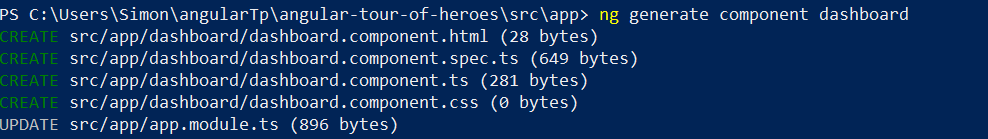
</nav>

On ajoute un lien de navigation vers Heroes, <a> </a> est la ligne permettant de cliquer sur Heroes vers /heroes.



6.

La CLI génère les fichiers pourle DashboardComponent et le déclare dans AppModul



7.)

On importe le composant Dashboard dans le module de routage d'application (src/app/app-routing.module.ts)

import { HeroDetailComponent } from './hero-detail/hero-detail.component';

On ajoute une route au tableau AppRoutingModule.routes qui correspond à un chemin d'accès au composant DashboardComponent.

{ path: 'dashboard', component: DashboardComponent },

8.

{ path: '', redirectTo: '/dashboard', pathMatch: 'full' },

Cette route redirige une URL qui correspond au chemin vide vers la route dont le chemin est '/dashboard'.

9.

On rajoute un lien vers le dashboard dans appcomponent.html

<h1>{{title}}</h1>

<nav>

<[a](https://angular.io/api/router/RouterLinkWithHref) [routerLink](https://angular.io/api/router/RouterLink)="/dashboard">Dashboard</[a](https://angular.io/api/router/RouterLinkWithHref)>

<[a](https://angular.io/api/router/RouterLinkWithHref) [routerLink](https://angular.io/api/router/RouterLink)="/heroes">Heroes</[a](https://angular.io/api/router/RouterLinkWithHref)>

</nav>

<[router-outlet](https://angular.io/api/router/RouterOutlet)></[router-outlet](https://angular.io/api/router/RouterOutlet)>

<app-messages></app-messages>

10.

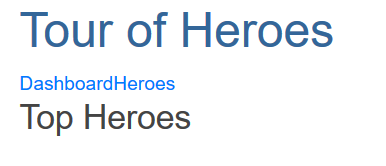
L'utilisateur devrait pouvoir accéder à ces informations de trois façons.

- En cliquant sur un héros dans le tableau de bord.

- En cliquant sur un héros dans la liste des héros.

- En collant une URL "lien profond" dans la barre d'adresse du navigateur qui identifie le héros à afficher.

Dans cette section, on va activer la navigation vers HeroDetailsComponent et la libérer du composant HeroesComponent.



12.

On importe HeroDetailComponent (src/app/app-routing.module.ts)

import { HeroDetailComponent } from './hero-detail/hero-detail.component';

On ajoute ensuite une route paramétrée au tableau AppRoutingModule.

{ path: 'detail/:id', component: HeroDetailComponent },

13.

On utilise une liaison d'interpolation dans le répéteur \*ngFor pour insérer le hero.id de l'interation courante dans chaque routeurLink.

<a \*ngFor="let hero of heroes" class="col-1-4"

routerLink="/detail/{{hero.id}}">

14.

On Réduis le <li> juste à son \*ngFor, puis on enveloppe le badge et le nom dans un élément <a>, et on ajoute un attribut routerLink à l'ancre qui est le même que dans le template du tableau de bord

<h2>My Heroes</h2>

<ul class="heroes">

<li \*ngFor="let hero of heroes"

[class.selected]="hero === selectedHero"

(click)="onSelect(hero)">

<span class="badge">{{hero.id}}</span> {{hero.name}}

</li>

</ul>

15. On supprime le code facultatif

16.

HeroDetailComponent a besoin d'une nouvelle façon d'obtenir le héros à afficher.

- On Obtient l'itinéraire qui l'a créé,

- On Extraie l'identifiant de la route

- On Acquérie le héros avec cet id du serveur via le HeroService

import { ActivatedRoute } from '@angular/router';

import { Location } from '@angular/common';

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

Puis on les construit:

constructor(

private route: [ActivatedRoute](https://angular.io/api/router/ActivatedRoute),

private heroService: HeroService,

private [location](https://angular.io/api/core/ComponentRef#location): [Location](https://angular.io/api/common/Location)

) {}

L'objet ActivatedRoute contient des informations sur la route vers cette instance de HeroDetailComponent.

Le HeroService obtient des données de héros du serveur distant et ce composant l'utilisera pour obtenir le héros à afficher.

Location est un service d'Angular pour interagir avec le navigateur. Vous l'utiliserez plus tard pour revenir à la vue précédente.

17.

Dans le hook de cycle de vie ngOnInit (), appelez getHero () et définissez-le comme suit :

ngOnInit(): void {

this.getHero();

}

getHero(): void {

const id = +this.route.snapshot.paramMap.get('id');

this.heroService.getHero(id)

.subscribe(hero => this.hero = hero);

}

route.snapshot est une image statique des informations de la route peu après la création du composant.

Le paramMap est un dictionnaire de valeurs de paramètres de route extraites de l'URL. La clé "id" renvoie l'identifiant du héros à récupérer.

Les paramètres d'une route sont toujours des chaînes. L'opérateur JavaScript (+) convertit la chaîne en un nombre, ce que doit être un identifiant de héros.

Le navigateur se rafraîchit et l'application se bloque avec une erreur de compilation. HeroService n'a pas de méthode getHero (). Ajoutez-la maintenant.

18.

les backticks (`) définissent un littéral de template JavaScript pour l'intégration de l'ID.

Comme getHeroes (), getHero () a une signature asynchrone. Il renvoie un héros simulé en tant qu'observable, en utilisant la fonction RxJS de ().

Nous sommes en mesure de ré-implémenter getHero () comme une vraie demande Http sans avoir à changer le composant HeroDetailComponent qui l'appelle.

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

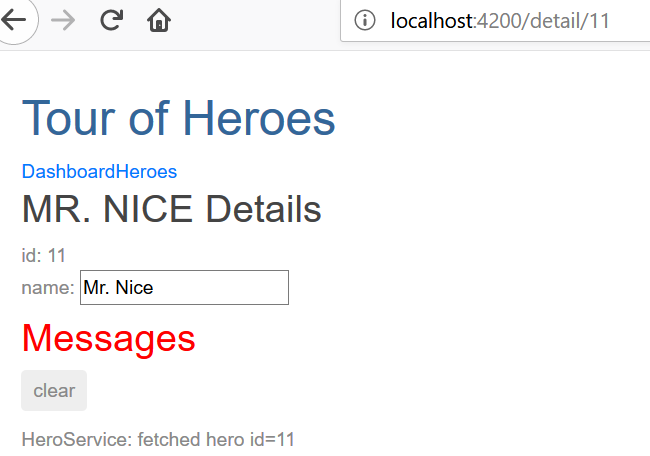
// TODO: send the message \_after\_ fetching the hero

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

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

}

Test url : http://localhost:4200/detail/11



19.

On ajoute un bouton sur la vue HeroDetail.

<button (click)="goBack()">go back</button>

On Ajoute une méthode goBack(, qui recule d'une étape dans la pile d'historique du navigateur à l'aide du service

goBack(): void {

this.location.back();

}

6. Conclusion

- On a ajouté le routeur d'Angular pour pouvoir naviguer parmi les différents composants.

- Nous avons transformé AppComponent en un shell de navigation avec des liens <a> et un <routeur-outlet>.

- Nous avons configuré le routeur dans un AppRoutingModule

- Nous avons défini des routes simples, une route de redirection et une route paramétrée.

- Nous avons utilisé la directive routerLink dans les éléments <a>.

- Nous avons refaçonné une vue maître/détail étroitement couplée en une vue détaillée routée.

- Nous avons utilisé les paramètres de lien du routeur pour accéder à la vue détaillée d'un héros sélectionné par l'utilisateur.

- Nous avons partagé le HeroService à plusieurs composants.

Ajout du code css des différentes pages html :

