



Navigation et mise en place des routes

- ▶ **Routage**
- ▶ **Configuration des routes dans un projet standalone**
- ▶ **Configuration des routes dans un projet classique (non-standalone)**
- ▶ **Les routes paramétrées**
- ▶ **Configuration de child route**
- ▶ **Le service `ActivatedRoute`**
- ▶ **Le service `Router`**
- ▶ **API de routing basé sur les signals**





Routage



Routage - Définition



Le routage est le fait de naviguer d'une vue à une autre selon la demande de l'utilisateur.

Un utilisateur peut:

- Taper l'url d'une vue dans le navigateur
- Cliquer sur un lien, bouton, etc..
- Cliquer sur les boutons de retour du navigateur

Routage - @angular/router



Le package **@angular/router** :

- Implémente le service Angular Router , qui permet la navigation d'une vue à l'autre
- Définit un ensemble d'objets, directives, modules et des services pour assurer le bon fonctionnement de la navigation.
- Il faut importer, depuis, le module RouterModule dans votre application pour utiliser le service Router

Un routeur n'a pas d'itinéraires jusqu'à ce que vous le configuriez.

Routage - <base href>



- Les applications routées mentionne au routeur comment composer l'url de navigation.
- Ceci est fait en utilisant la balise <base href> à placer comme premier nœud de la balise <head> dans le fichier index.html
- Si le répertoire app est la racine de l'application, il faut écrire:

```
<base href="/">
```



Configuration des routes dans un projet standalone

► Configuration des routes



Dans un projet standalone:

- Les routes sont à définir au niveau de la propriété **routes** du fichier **app.routes.ts** dont le contenu par défaut est :

```
import { Routes } from '@angular/router';  
export const routes: Routes = [];
```

- Exemple:

```
const routes: Routes = [  
  { path: 'first-component', component: FirstComponent },  
  { path: 'second-component', component: SecondComponent },  
];
```

- Dans le fichier **app.config.ts**, assurez vous que la propriété **routes** est passée à la fonction **provideRouter**

```
export const appConfig: ApplicationConfig = {  
  providers: [provideRouter(routes)]  
};
```


► Configuration des routes



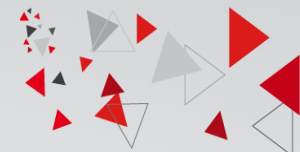
- Le path peut être associé à une url vide pour rediriger l'utilisateur s'il ne tape rien à un composant particulier. Il faut ajouter ainsi la propriété **pathMatch: 'full'** pour informer angular de faire la correspondance sur tout le mot

```
{path: '', redirectTo: 'home', pathMatch: 'full'}
```

- Pour envoyer des informations static et spécifique à la route appelée (exemple: titre de la page) nous utilisons la propriété "data"

```
{path: 'path', component: unComponent, data:{title : 'titrespecial' } }
```

► Configuration des routes



- Le path peut être associé à une url '**' pour désigner n'importe quel url tapé

`{path: '**', component: HomeComponent}`

- La route ayant comme path:** doit être en dernière position. Si le router ne trouve aucun path correspondant au path tapé dans l'url jusqu'au qu'il arrive à cette route alors il appliquera cette route.

L'ordre de la déclaration des routes est important

► Chargement des routes - RouterOutlet



- Cette directive permet de désigner l'emplacement de la vue associée à l'url demandé
- Utilisé dans le template du composant
- A définir dans la liste des imports d'un composant standalone

<router-outlet></router-outlet>

```
@Component({
  selector: 'app-root',
  standalone: true,
  imports: [CommonModule, RouterOutlet],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'routing-app';
}
```

► Création de lien - RouterLink



- La directive routerLink génère tout de même l'attribut href pour faciliter la compréhension de la page par les "browsers" ou "moteurs de recherche" et désactive le chargement de la page.

```
<a [routerLink]="['/search']">Search</a>
```

- A définir dans la liste des imports d'un composant standalone

```
@Component({
  selector: 'app-root',
  standalone: true,
  imports: [CommonModule, RouterOutlet, RouterLink],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'routing-app';
}
```



Configuration des routes dans un projet classique (non-standalone)



Configuration des routes

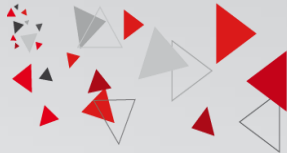


Dans un projet classique (non-standalone):

- Les routes sont à définir au niveau d'un module de routage.
- Chaque module possède son propre module de routage
- Le module AppRoutingModuleModule est le module de routage pour le module racine.
- AppRoutingModuleModule est généré automatiquement.



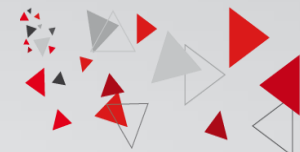
Configuration des routes - RouterModule



- Le module RouterModule permet de configurer les routes et propose deux méthodes:

Méthode	Rôle	Lieu de déclaration	Déclaration
forRoot	Crée un module qui contient toutes les directives, les routes et le service router.	Module de routage du module racine «AppRoutingModule»	<pre>@NgModule({ imports: [RouterModule.forRoot(routes, { options }] }) export class AppRoutingModule { }</pre>
forChild	Crée un module qui contient toutes les directives, les routes mais pas le service router.	Module de routage de Features Modules	<pre>@NgModule({ imports: [RouterModule.forChild(routes)]}) export class FeatureRoutingModule { }</pre>

► Configuration des routes



- Il faut importer **RouterModule** et **Routes** à partir de `@angular/router`
- Chaque route associe un path à un composant.
- Le paramètre ***routes*** définit un tableau de routes et est défini comme suit avant `@NgModule`

```
const routes : Routes = [  
  {path: 'textUrl1', component: nomDucompsant1},  
  {path: 'textUrl2', component: nomDucompsant2 },  
  {...}  
];
```


► Configuration des routes



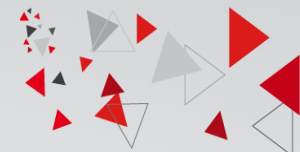
- Le path peut être associé à une url vide pour rediriger l'utilisateur s'il ne tape rien à un composant particulier. Il faut ajouter ainsi la propriété **pathMatch: 'full'** pour informer angular de faire la correspondance sur tout le mot

```
{path: '', redirectTo: 'home', pathMatch: 'full'}
```

- Pour envoyer des informations static et spécifique à la route appelée (exemple: titre de la page) nous utilisons la propriété "data"

```
{path: 'path', component: unComponent, data:{title : 'titrespecial' } }
```

► Configuration des routes



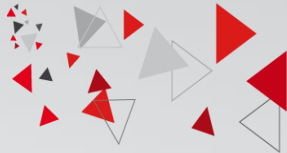
- Le path peut être associé à une url '**' pour désigner n'importe quel url tapé

`{path: '**', component: HomeComponent}`

- La route ayant comme path:** doit être en dernière position. Si le router ne trouve aucun path correspondant au path tapé dans l'url jusqu'au qu'il arrive à cette route alors il appliquera cette route.

L'ordre de la déclaration des routes est important

▶ Création de lien - RouterLink



- En utilisant des liens natifs le "browser" va produire une requête HTTP GET vers le serveur et recharger toute l'application.
- Pour éviter ce problème, le module de "Routing" Angular fournit la directive routerLink qui permet d'intercepter l'événement click sur lien et de changer de "route" sans recharger toute l'application
- La directive routerLink génère tout de même l'attribut href pour faciliter la compréhension de la page par les "browsers" ou "moteurs de recherche"

```
<a routerLink="/serach">Search</a>
```



Chargement des routes - RouterOutlet



- Cette directive permet de désigner l'emplacement de la vue associée à l'url demandé
- Définie dans RouterModule
- Utilisé dans le template du composant

```
<router-outlet></router-outlet>
```

▶ Création de lien dynamique



- Dans le fichier TS

```
getLink(){  
  this.route="/serach";  
  This.routeName="search";}
```

- Dans le fichier HTML

```
<a [routerLink]="route">{{routeName}}</a>
```



Les routes paramétrées



Les routes paramétrées - Définition



- Une route paramétrée est définie comme suit:

```
{ path: 'path/id', component: oneComponent},
```

un paramètre est précédé par « : »

- Un paramètre peut être envoyé dans un lien :

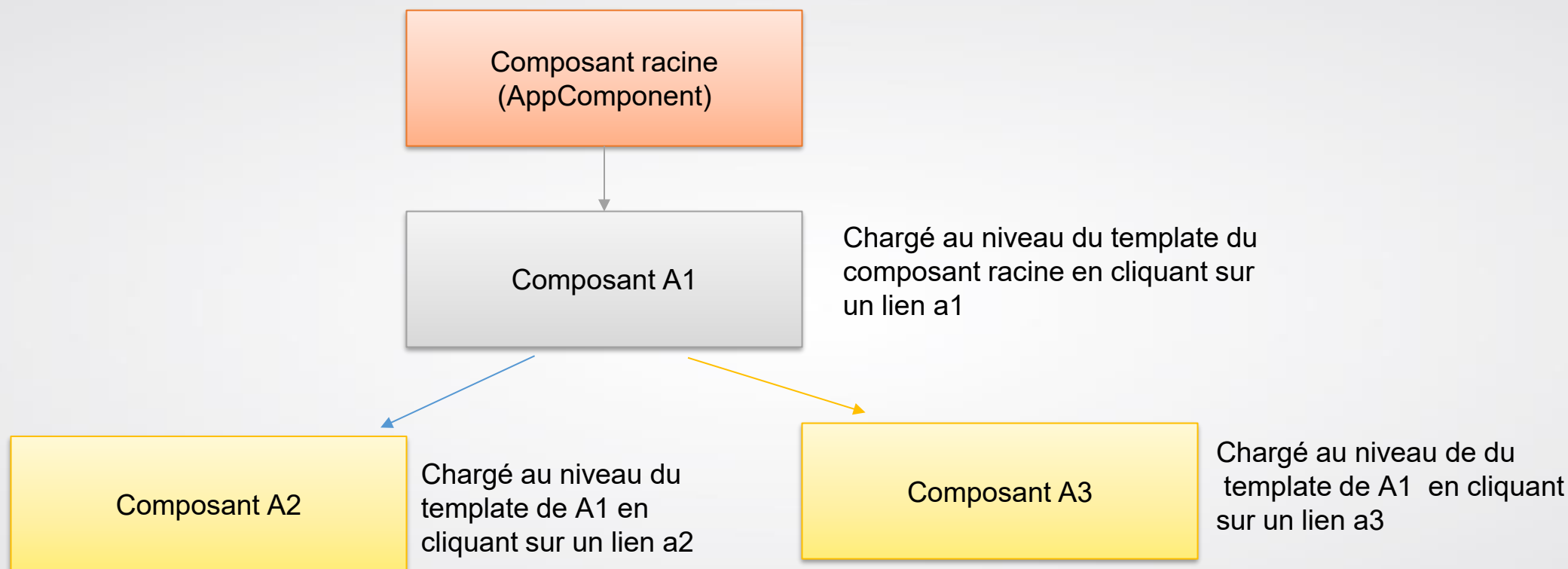
```
<a [routerLink]="['/path',id']">Lien</a>
```

id est la valeur de l'id à envoyer dans l'url et récupéré au niveau de l'html

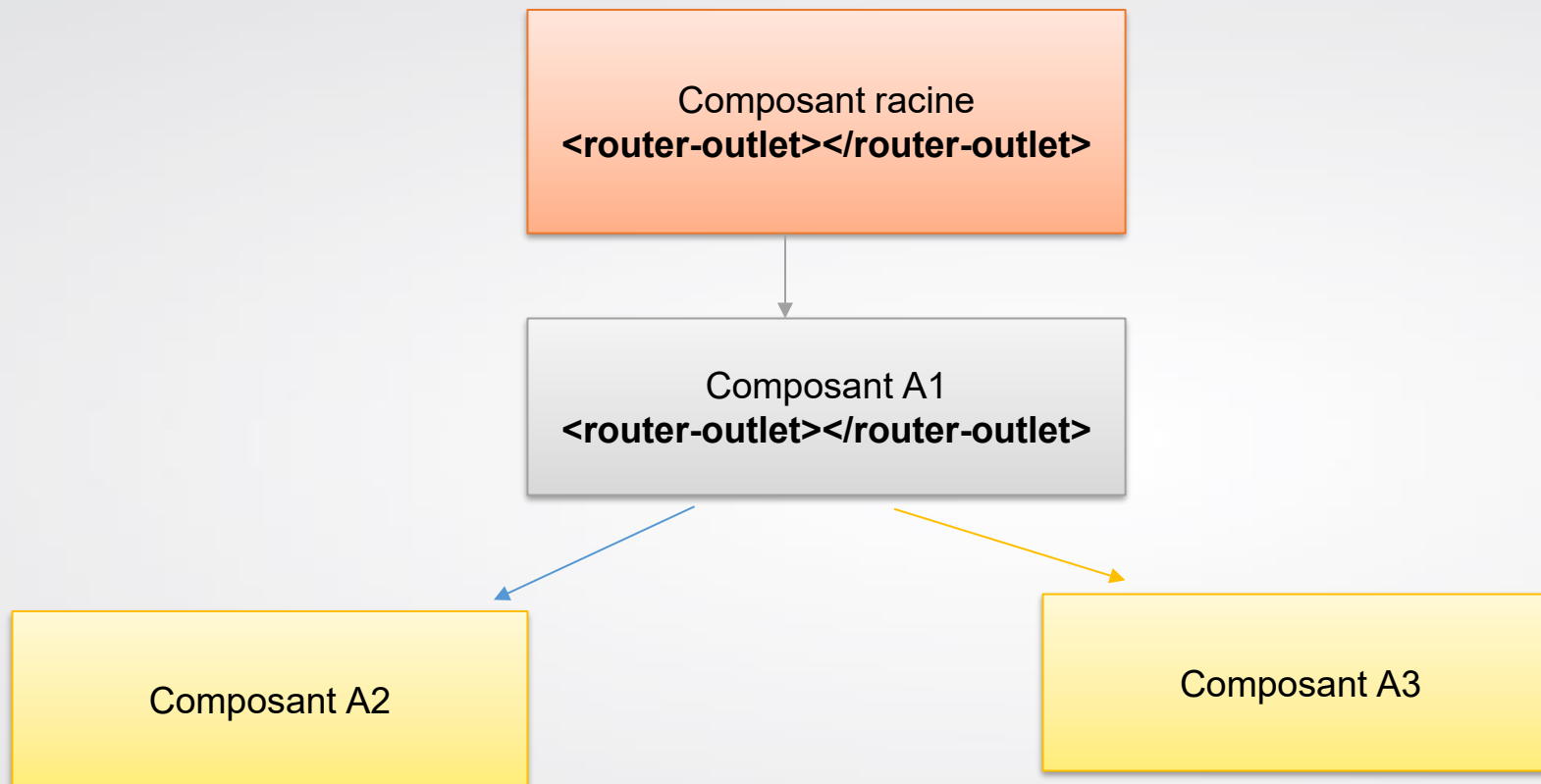


Configuration de child route

► Configuration de « child routes »



► Configuration de « child routes »



► Configuration de « child routes »



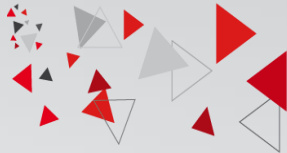
```
const routes: Routes = [  
  { path: 'A1', component: A1Component, children: [  
    { path: 'A2', component: A2Component,  
      { path: 'A3', component: A3Component }  
    ]  
  }  
];
```

Le service `ActivatedRoute`





ActivatedRoute



- **ActivatedRoute** est un service qui fournit à chaque composant des informations spécifiques à la route tel que : url, data, paramMap, queryParams, outlet, fragment, children, firstChild,
- On peut récupérer les paramètres envoyés dans l'URL grâce aux propriétés snapshot, params, paramMap, queryParams, queryParamsMap, ...
- Le service **ActivatedRoute** est importé depuis **@angular/router** et injecté au niveau du composant/service dans lequel on veut l'utiliser.

```
import {ActivatedRoute} from '@angular/router';  
.....  
constructor( private activatedroute : ActivatedRoute) { }  
.....
```

▶ ActivatedRoute - Propriétés



Propriété	Description	Type de retour
snapshot	Retourne une capture sur la route initiale	<u>ActivatedRouteSnapshot</u>
url	Retourne les segments d'URL correspondant à cette route.	Observable<UrlSegment[]>
params http://localhost:4200/profiles/ 5	Retourne un objet contenant les paramètres de cette route indexés par leurs noms.	Observable<Params>
queryParams http://localhost:4200/profiles? id=5	Retourne un objet contenant les query parameters de cette route indexés par leurs noms	Observable<Params>
data	Retourne les données statiques	Observable<Data>
paramMap	Retourne les paramètres obligatoires et facultatifs spécifiques à l'itinéraire. On peut récupérer depuis la valeur de chaque paramètre.	Observable<ParamMap>
queryParamMap	Retourne les paramètres de requête disponibles pour toutes les routes. On peut récupérer depuis la valeur de chaque paramètre.	Observable<ParamMap>

RQ: il y a encore d'autres propriétés

Voir le lien : <https://angular.io/api/router/ActivatedRoute#properties>

ActivatedRoute – Capture sur la route



<http://localhost:4200/profiles?min=45&max=50&location=29293>

Configuration de la route	{path:"list/:id/:name", component:ListUserComponent},
Lien de la route	<a [routerLink]="['/list', 5,'ahmed']" [queryParams]="{min:45,max:50,location:29293}">List
Capture sur la route avec Snapshot	constructor(private ac:ActivatedRoute) {} ngOnInit(){console.log(this.ac.snapshot); }



```
▼ ActivatedRouteSnapshot {url: Array(3), params: {...}, quer
  {...}, ...} ⓘ
  ► component: class ListUserComponent
  ► data: {}
    fragment: null
    outlet: "primary"
  ► params: {id: '5', name: 'ahmed'}
  ► queryParams: {min: '45', max: '50', location: '29293'}
  ► routeConfig: {path: 'list/:id/:name', component: f}
  ▼ url: Array(3)
    ► 0: UrlSegment {path: 'list', parameters: {...}}
    ► 1: UrlSegment {path: '5', parameters: {...}}
    ► 2: UrlSegment {path: 'ahmed', parameters: {...}}
    length: 3
  ► [[Prototype]]: Array(0)
```

```
▼ _paramMap: ParamsAsMap
```

```
▼ params:
```

```
  id: "5"
```

```
  name: "ahmed"
```

```
  ► [[Prototype]]: 0
```

```
  keys: (...)
```

```
  ► [[Prototype]]: Obj
```

```
  ► _resolve: {}
```

```
  ► _resolvedData: {}
```

```
  ► _routerState: RouterStateSnapsh
```

```
  ► _urlSegment: UrlSegmentGroup {s
```

```
▼ queryParams: ParamsAsMap
```

```
  ► params: {min: '45', max: '50', location: '29293'}
```

```
  ▼ keys: Array(3)
```

```
    0: "min"
```

```
    1: "max"
```

```
    2: "location"
```

```
    length: 3
```



ActivatedRoute - ParamMap VS Params



- La propriété **Params** est un tableau des valeurs de paramètres, indexé par nom. Vous pouvez toujours l'utiliser mais il est désormais obsolète et remplacé par le ParamMap
- Le **ParamMap** facilite le travail avec les paramètres. Nous pouvons utiliser les méthodes `get` ou `getAll` pour récupérer la valeur des paramètres dans le composant. Utilisez la méthode `has` pour vérifier si un certain paramètre existe.

```
interface ParamMap {  
    keys: string[]  
    has(name: string): boolean  
    get(name: string): string | null  
    getAll(name: string): string[]  
}
```


ActivatedRoute - pathParam



http://localhost:4200/profiles/5

Configuration de la route	{path:"list/:id", component:ListUserComponent},
Lien de la route	<a [routerLink]="['/list', 5]">List

Action	Résultat	Récupération de l'id
console.log(this.ac.snapshot.paramMap.get("id"))	5	this.id=this.ac.snapshot.paramMap.get("id");
this.ac. paramMap .subscribe(params => { console.log(params); });	▼ ParamsAsMap {params: {...}} ⓘ ► params: {id: '5'} ► keys: (...) ► [[Prototype]]: Object	
this.ac. paramMap .subscribe(params => { console.log(params.get('id')); });	5	this.ac. paramMap .subscribe(params => { this.id=params.get('id'); });
this.ac. params .subscribe(params => { console.log(params); });	{id: '5'}	
this.ac. params .subscribe(params => { console.log(params['id']); });	5	this.ac. params .subscribe(params => { this.id=params['id']; });

ActivatedRoute - queryParams

http://localhost:4200/profiles?min=45&max=50&location=29293



Configuration de la route	{path:"profiles", component:ListUserComponent}
Lien de la route	<a [routerLink]="['/profiles']" [queryParams]="{min:45,max:50,location:29293}">

Action	Résultat	Récupération de min
console.log(this.ac.snapshot.queryParamMap.get("min"));	45	this.min= this.ac.snapshot.queryParamMap.get("min"))
this.ac.queryParamMap.subscribe(params => { console.log(params); });	<div>▼ ParamsAsMap {params: {...}} ⓘ</div> <div>▼ params: location: "29293" max: "50" min: "45" ► [[Prototype]]: Object keys: (...) ► [[Prototype]]: Object</div>	
this.ac.queryParamMap.subscribe(params => { console.log(params.get('min')); });	45	this.ac.queryParamMap.subscribe(params => { this.min=params.get('min'); });
this.ac.queryParams.subscribe(params => { console.log(params); });	{min: '45', max: '50', location: '29293'}	
this.ac.queryParams.subscribe(params => { console.log(params['min']); });	45	this.ac.queryParams.subscribe(params => { this.min=params['min']; });

Router - Présentation



- Le Router est un service qui fournit une navigation parmi les vues et des capacités de manipulation d'URL
- Il possède des propriétés qui permettent de suivre l'état de chaque route (routerState, errorHandler, navigated, ...)
- Il possède des méthodes pour manipuler les URLs (createUrlTree(), navigateByUrl(), navigate(),....)
- Il faut l'injecter dans un composant pour l'utiliser
- Provided in dans RouterModule et RouterTestingModule



Router - Utilisation



- Importer le service Router depuis @angular/router

```
import { ActivatedRoute, Router } from '@angular/router';
```

- l'injecter dans le composant/services

```
constructor(private _router:Router) {}
```

- Exploiter ses propriétés et ses méthodes

```
this._router.navigate(['/profiles', id], {queryParams: {}})
```



► **Merci de votre attention**