

Implémentation d'un système de navigation



Année universitaire
2023-2024



- ▶ **Routage**
- ▶ **Configuration des routes**
- ▶ **Routes paramétrées**
- ▶ **Le service Router**
- ▶ **Configuration de child root**



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 configurez.

Routage - <base href>



- Les applications routées mentionnent 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



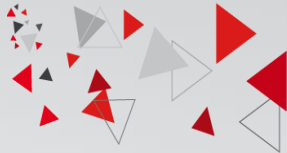
Configuration des routes - AppRoutingModuleModule



- 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 par YES à la première question posée lors de la création du projet.



Configuration des routes - RouterModule



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

Méthode	Rôle	Lieu de déclaration	Déclaration
forRoot	Créer 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	Créer 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: wildcard



- Le path peut être associé à une url '**' pour désigner n'importe quelle url tapée

```
{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
- Définie dans RouterModule
- Utilisé comme un composant

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

▶ 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>
```

▶ Création de lien dynamique



- Dans le fichier TS

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

- Dans le fichier HTML

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



Création de lien - RouterLinkActive



La directive **RouterLinkActive** associe une classe CSS (exemple : active) au lien une fois la route sollicitée devient active et retire la classe, quand la route devient inactive

```
<nav>  
<a routerLink="/path1" routerLinkActive="active">Accueil</a>  
  <a routerLink="/path2" routerLinkActive="active">About</a>  
</nav>  
<router-outlet></router-outlet>
```




Les routes paramétrées

▶ Les routes paramétrées



- Une route paramétrée est définie comme suit:

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

un paramètre est précédé par « : »

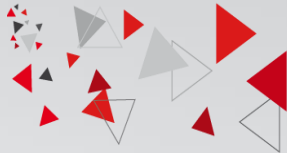
- Un paramètre peut être envoyé dans un lien :

```
<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



ActivatedRoute



- **ActivatedRoute** est un service qui fournit à chaque composant des informations spécifiques à la route tel que : url, data, paramMap, queryParamsMap, outlet, fragment, childre, firstChild,
- On peut récupérer les paramètres envoyés dans l'url grâce aux propriétés sapshot, params, paramMap, queryParams, queryParamsMap, ...
- Le service **Activated Route** 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 indexée 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>



Le service Router

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) {}
```

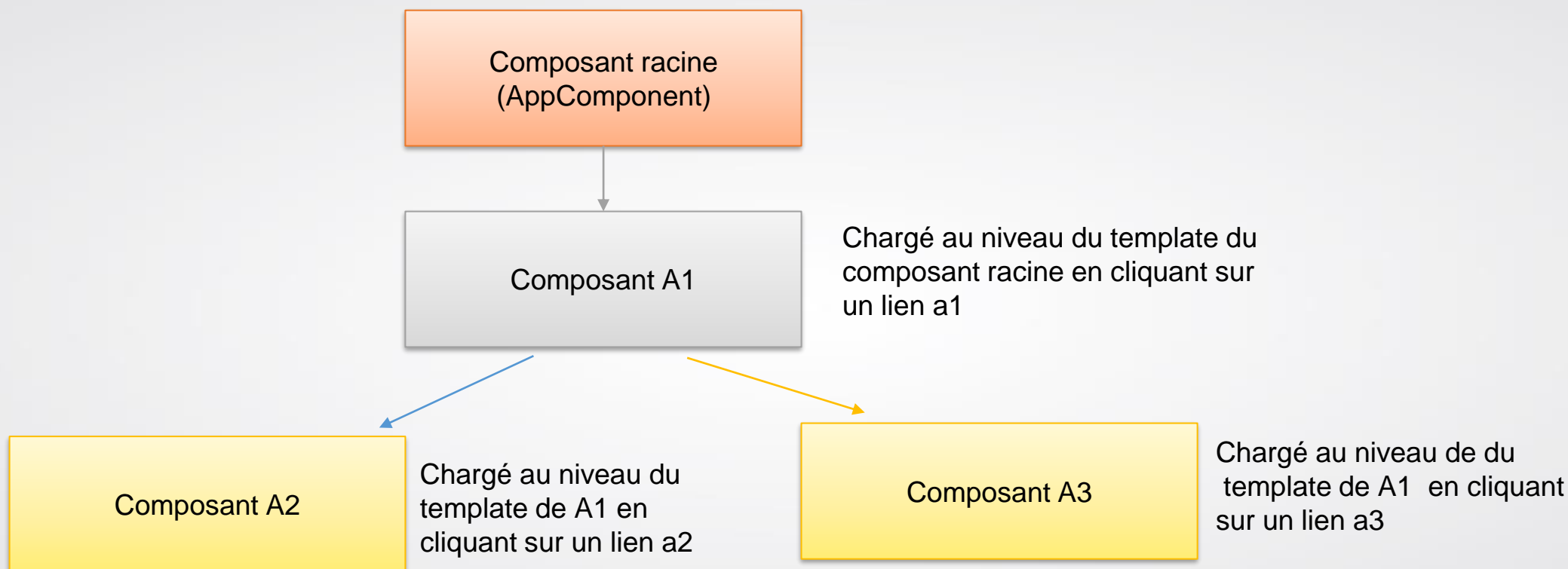
- Exploiter ses propriétés et ses méthodes

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

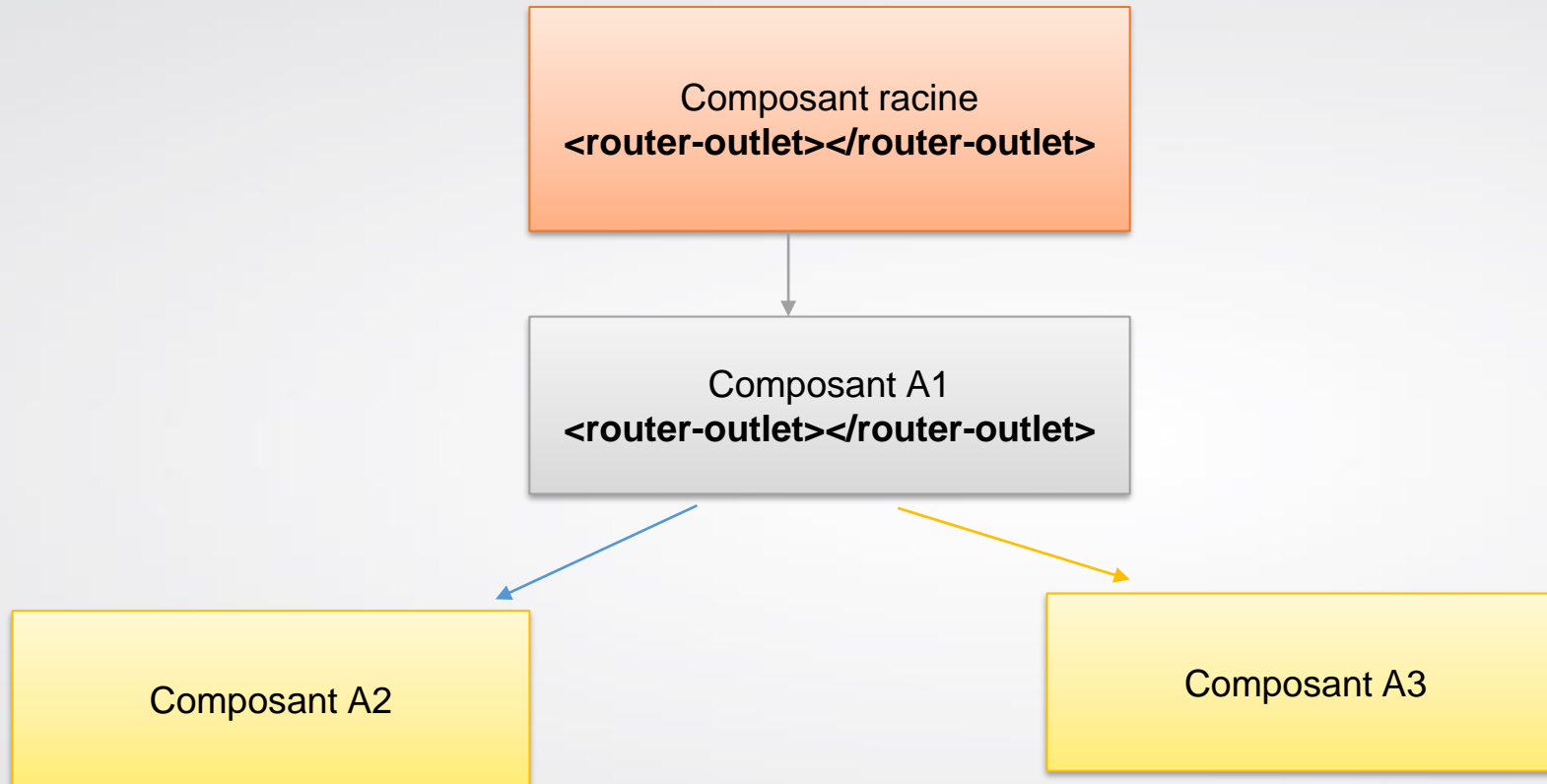


Configuration de child root

► 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 }  
    ]  
  }  
];
```



► **Merci de votre attention**