

Angular : composants et routage

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

`elmouelhi.achref@gmail.com`



Plan

- 1 Création de composant
- 2 Routage
- 3 Création d'un module de routage
- 4 Paramètres de requête
 - `/chemin/param1/param2`
 - `/chemin?var1=value1&var2=value2`
- 5 Injection de dépendance avec `inject`
- 6 Création de liens avec paramètres
- 7 Récupération des paramètres avec `@Input`
- 8 Mise à jour du `title` en fonction de la route demandée
- 9 Navigation depuis `.component.ts`
- 10 Chemin vide et chemin inexistant

Angular

Pour créer un nouveau composant

```
ng generate component component-name
```

© Achref EL MOUELHI ©

Angular

Pour créer un nouveau composant

```
ng generate component component-name
```

Ou utiliser le raccourci

```
ng g c component-name
```

Angular

Pour créer un nouveau composant

```
ng generate component component-name
```

Ou utiliser le raccourci

```
ng g c component-name
```

Pour placer un composant dans un répertoire x

```
ng g c x/component-name
```

Angular

Autres options de la commande `ng g c`

- `--inline-style` : permet de définir les styles **CSS** du composant directement dans le fichier **TypeScript** du composant, plutôt que dans un fichier de style **CSS** séparé.
- `--inline-template` : permet de définir le template **HTML** du composant directement dans le fichier **TypeScript** du composant, plutôt que dans un fichier de template **HTML** séparé.
- `--skip-tests` : permet de créer un composant sans générer de fichiers de test unitaires.
- `--dry-run` : permet d'exécuter la commande **CLI** comme une simulation sans effectuer réellement de modifications sur le système de fichiers.
- `--standalone` [**Angular 16**] : permet de créer un composant de manière autonome, sans le lier à un module existant.
- ...

Angular

Pour appliquer certaines propriétés sur tous les composants générés, il faut modifier la section `schematics` de `angular.json`

```
"schematics": {  
  "@schematics/angular:component": {  
    "inlineTemplate": true,  
    "inlineStyle": true  
  }  
},
```

Angular

Pour la suite, on va créer 3 composants

- adresse
- stagiaire
- menu

Angular

Commençons par créer le premier composant `adresse` dans un répertoire `composants`

```
ng g c composants/adresse
```

© Achref EL MOUËL

Angular

Commençons par créer le premier composant `adresse` dans un répertoire `composants`

```
ng g c composants/adresse
```

Résultat

```
CREATE src/app/composants/adresse/adresse.component.html (22 bytes)
CREATE src/app/composants/adresse/adresse.component.spec.ts (635 bytes)
CREATE src/app/composants/adresse/adresse.component.ts (273 bytes)
CREATE src/app/composants/adresse/adresse.component.css (0 bytes)
UPDATE src/app/app.module.ts (490 bytes)
```

Angular

Ensuite stagiaire

```
ng g c composants/stagiaire
```

© Achref EL MOUETI

Angular

Ensuite stagiaire

```
ng g c composants/stagiaire
```

Résultat

```
CREATE src/app/composants/stagiaire/stagiaire.component.html (23 bytes)  
CREATE src/app/composants/stagiaire/stagiaire.component.spec.ts (642  
bytes) CREATE src/app/composants/stagiaire/stagiaire.component.ts (277  
bytes) CREATE src/app/composants/stagiaire/stagiaire.component.css (0  
bytes) UPDATE src/app/app.module.ts (591 bytes)
```

Angular

Et enfin menu

```
ng g c composants/menu
```

© Achref EL MOUËLTI

Angular

Et enfin menu

```
ng g c composants/menu
```

Résultat

```
CREATE src/app/composants/menu/menu.component.html (19 bytes) CREATE  
src/app/composants/menu/menu.component.spec.ts (614 bytes) CREATE  
src/app/composants/menu/menu.component.ts (267 bytes) CREATE  
src/app/composants/menu/menu.component.css (0 bytes) UPDATE  
src/app/app.module.ts (1044 bytes)
```

Angular

Constats

- Quatre fichiers créés pour chaque composant
 - `x.component.ts` avec `x` = stagiaire, adresse ou menu
 - `x.component.html`
 - `x.component.css`
 - `x.component.spec.ts`
- Trois déclarations effectuées dans la section `declarations` de `app.module.ts`

Angular

Nouveau contenu de `app.module.ts`

```
@NgModule({
  declarations: [
    AppComponent,
    ObservableComponent,
    AdresseComponent,
    StagiaireComponent,
    MenuComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Angular

Pour afficher le contenu de ces trois composants dans

`app.component.html`

```
<app-stagiaire></app-stagiaire>  
<app-adresse></app-adresse>  
<app-menu></app-menu>
```

© Achref EL MOUËL

Angular

Pour afficher le contenu de ces trois composants dans

`app.component.html`

```
<app-stagiaire></app-stagiaire>  
<app-adresse></app-adresse>  
<app-menu></app-menu>
```

Remarques

- En général, on préfère ne pas afficher tous les composants dans le composant principal
- On associe plutôt un chemin à chaque composant
- Le composant sera affiché dans le composant principal si son chemin apparaît dans l'URL de la requête **HTTP**

Module de routage

- À la création du projet, on a demandé la génération d'un fichier de routage : `app-routing.module.ts`
- Ce fichier permet d'assurer le mapping chemin/composant
- Il contient un tableau vide de type `Routes`
- Chaque route peut avoir comme attributs (`path`, `component`, `redirectTo`, `children...`)

Contenu de `app-routing.module.ts`

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

const routes: Routes = [];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Angular

`RouterModule` a deux méthodes statiques qui prennent en paramètre un tableau de `Routes`

- `.forRoot(tableau)` : pour le module principal (racine)
- `.forChild(tableau)` : pour les sous-modules inclus dans le module principal

Définissons des routes dans ce tableau de routes de `app-routing.module.ts`

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { StagiaireComponent } from '../composants/stagiaire/stagiaire.
  component';
import { AdresseComponent } from '../composants/adresse/adresse.
  component';

const routes: Routes = [
  { path: 'stagiaire', component: StagiaireComponent },
  { path: 'adresse', component: AdresseComponent },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Définissons des routes dans ce tableau de routes de `app-routing.module.ts`

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { StagiaireComponent } from '../composants/stagiaire/stagiaire.
  component';
import { AdresseComponent } from '../composants/adresse/adresse.
  component';

const routes: Routes = [
  { path: 'stagiaire', component: StagiaireComponent },
  { path: 'adresse', component: AdresseComponent },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Pas de route pour le composant `menu` car on veut l'afficher quelle que soit le chemin demandé.

Remarques

- L'URL saisies auront la forme suivante
`localhost:4200/adresse` **ou** `localhost:4200/stagiaire`
- Faut-il ajouter `/` comme préfix aux valeurs de l'attribut `path` ?
- Non, car `/` a été défini dans `index.html` dans la balise `<base href="/">`

Angular

`{ enableTracing: true }` permet de garder une trace de la recherche d'un chemin (pour le débogage).

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { StagiaireComponent } from '../composants/stagiaire/stagiaire.
  component';
import { AdresseComponent } from '../composants/adresse/adresse.
  component';

const routes: Routes = [
  { path: 'stagiaire', component: StagiaireComponent },
  { path: 'adresse', component: AdresseComponent },
];

@NgModule({
  imports: [RouterModule.forRoot(routes, { enableTracing: true })],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Angular

Comment faire pour tester ?

Il faut saisir

- `localhost:4200/adresse` **ou**
- `localhost:4200/stagiaire`

dans la barre d'adresse du navigateur

© Achre

Angular

Comment faire pour tester ?

Il faut saisir

- `localhost:4200/adresse` **ou**
- `localhost:4200/stagiaire`

dans la barre d'adresse du navigateur

Où le composant sera affiché dans `app.component.html` ?

Il faut lui indiquer l'emplacement **mais pas en ajoutant le sélecteur du composant**

Angular

Il faut indiquer l'emplacement du composant à charger dans `app.component.html` en ajoutant la balise suivante

`app-component`

`app-header`

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

`app-footer`

Remarques

- Pour accéder à un composant, l'utilisateur doit connaître son chemin défini dans le tableau de routes (or ceci n'est pas vraiment très pratique)
- On peut plutôt définir un menu contenant des liens vers nos différents composants

Commençons par définir le menu suivant dans `menu.component.html`

```
<ul>
  <li><a href=''> Accueil </a></li>
  <li><a href='stagiaire'> Stagiaire </a></li>
  <li><a href='adresse'> Adresse </a></li>
</ul>
```

© Achref EL MOUELHI ©

Commençons par définir le menu suivant dans `menu.component.html`

```
<ul>
  <li><a href=''> Accueil </a></li>
  <li><a href='stagiaire'> Stagiaire </a></li>
  <li><a href='adresse'> Adresse </a></li>
</ul>
```

Dans `app.component.html`, on ajoute le menu et on indique l'emplacement des composants à afficher

```
<app-menu></app-menu>
<router-outlet></router-outlet>
```

Commençons par définir le menu suivant dans `menu.component.html`

```
<ul>
  <li><a href=''> Accueil </a></li>
  <li><a href='stagiaire'> Stagiaire </a></li>
  <li><a href='adresse'> Adresse </a></li>
</ul>
```

Dans `app.component.html`, on ajoute le menu et on indique l'emplacement des composants à afficher

```
<app-menu></app-menu>
<router-outlet></router-outlet>
```

Remarque

Chaque fois qu'on clique sur un lien la page est rechargée : ce n'est pas le but d'une application mono-page

Solution : remplacer l'attribut href par routerLink

```
<ul>
  <li><a routerLink=''> Accueil </a></li>
  <li><a routerLink='stagiaire'> Stagiaire </a></li>
  <li><a routerLink='adresse'> Adresse </a></li>
</ul>
```

Angular

Pour afficher la route active en gras

```
<ul>
  <li><a routerLink=''> Accueil </a></li>
  <li routerLinkActive=active>
    <a routerLink='stagiaire'> Stagiaire </a>
  </li>
  <li routerLinkActive=active>
    <a routerLink='adresse'> Adresse </a>
  </li>
</ul>
```

© Act

Angular

Pour afficher la route active en gras

```
<ul>
  <li><a routerLink=''> Accueil </a></li>
  <li routerLinkActive=active>
    <a routerLink='stagiaire'> Stagiaire </a>
  </li>
  <li routerLinkActive=active>
    <a routerLink='adresse'> Adresse </a>
  </li>
</ul>
```

Dans `menu.component.css`, il faut définir la classe `active`

```
.active {
  font-weight: bold;
}
```

Angular

Si on ajoute `routerLinkActive=active`, il sera en gras quelle que soit la page visitée, pour cela on ajoute `[routerLinkActiveOptions]="{ exact: true }"` pour que la classe soit uniquement ajoutée lorsque la route correspond exactement à la valeur de `routerLink`

```
<ul>
  <li routerLinkActive=active [routerLinkActiveOptions]="{
    exact: true }">
    <a routerLink=''> Accueil </a>
  </li>
  <li routerLinkActive=active>
    <a routerLink='stagiaire'> Stagiaire </a>
  </li>
  <li routerLinkActive=active>
    <a routerLink='adresse'> Adresse </a>
  </li>
</ul>
```

Angular

Pour créer un module de routage (Si on n'a pas accepté qu'il soit généré à la création du projet)

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

© Achref EL MOUELHI ©

Angular

Pour créer un module de routage (Si on n'a pas accepté qu'il soit généré à la création du projet)

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

Comprenons la commande

- `ng generate module app-routing` : pour générer un module de routage appelé `app-routing`.
- `--flat` : pour placer le fichier dans `src/app` et éviter de créer un répertoire propre à ce module.
- `--module=app` : pour enregistrer ce module dans le tableau `imports` de `AppModule`.

Angular

La section `imports` du fichier `app.module.ts`

```
imports: [
  BrowserModule,
  RouterModule.forRoot([
    { path: 'stagiaire', component: StagiaireComponent },
    { path: 'adresse', component: AdresseComponent },
  ]),
  AppRoutingModuleModule
],
```

Le fichier `app-routing.module.ts`

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';

@NgModule({
  imports: [
    CommonModule
  ],
  declarations: []
})
export class AppRoutingModuleModule { }
```

Angular

La section `imports` du fichier `app.module.ts`

```
imports: [  
  BrowserModule,  
  AppRoutingModule  
],
```

Modifions le contenu du fichier `app-routing.module.ts`

```
import { NgModule } from '@angular/core';  
import { RouterModule, Routes } from '@angular/router';  
// + les autres imports de composants  
  
const routes: Routes = [  
  { path: 'stagiaire', component: StagiaireComponent },  
  { path: 'adresse', component: AdresseComponent }  
];  
  
@NgModule({  
  imports: [RouterModule.forRoot(routes)],  
  exports: [RouterModule]  
})  
export class AppRoutingModule { }
```

Angular

Pour activer le traçage

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
// + les autres imports de composants

const routes: Routes = [
  { path: 'stagiaire', component: StagiaireComponent },
  { path: 'adresse', component: AdresseComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes, { enableTracing: true })],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Angular

Deux formes de paramètres de requête

- `/chemin/param1/param2`
- `/chemin?var1=value1&var2=value2`

© Achref EL MOUELHI

Angular

Deux formes de paramètres de requête

- `/chemin/param1/param2`
- `/chemin?var1=value1&var2=value2`

Pour ces deux formes de paramètres

- Deux manières différentes de définir les routes
- Deux objets différents permettant de récupérer les valeurs respectives
 - `paramMap` **et** `params` **pour** `/chemin/param1/param2`
 - `queryParamMap` **et** `queryParams` **pour** `/chemin?var1=value1&var2=value2`

Angular

Définissons une route de la forme /chemin/param1/param2 dans

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { StagiaireComponent } from '../composants/stagiaire/stagiaire.
  component';
import { AdresseComponent } from '../composants/adresse/adresse.
  component';

const routes: Routes = [
  { path: 'stagiaire', component: StagiaireComponent },
  { path: 'stagiaire/:nom/:prenom', component: StagiaireComponent },
  { path: 'adresse', component: AdresseComponent },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Angular

Pour récupérer les paramètres d'une route ayant la forme `stagiaire/:nom/:prenom`, il faut :

- aller dans le composant concerné (ici, `stagiaire.component.ts`)
- faire une injection de dépendance de la classe `ActivatedRoute` (comme paramètre de constructeur)
- utiliser un objet de cette classe dans la méthode `ngOnInit()`
 - soit par l'intermédiaire d'un objet `paramMap` (`Map`)
 - soit par l'intermédiaire d'un objet `params` (`Object`)
- choisir une des deux solutions suivantes
 - solution asynchrone avec les observables
 - solution synchrone avec les snapshots

Angular

Pour récupérer les paramètres d'une route de la forme `stagiaire/:nom/:prenom`, dans `stagiaire.component.ts`

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

@Component({
  selector: 'app-stagiaire',
  templateUrl: './stagiaire.component.html',
  styleUrls: ['./stagiaire.component.css']
})
export class StagiaireComponent implements OnInit {

  nom = "";
  prenom = "";

  constructor(private route: ActivatedRoute) { }

  ngOnInit() {
    this.route.paramMap.subscribe(res => {
      this.nom = res.get('nom') ?? "";
      this.prenom = res.get('prenom') ?? "";
      console.log(res);
    });
  }
}
```

Angular

Contenu de stagiaire.component.html

```
<h2>Stagiaire</h2>  
<p> Bonjour {{ prenom }} {{ nom }} </p>
```

Angular

Pour récupérer les paramètres d'une route dans un objet JavaScript, on utilise `params`

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

@Component({
  selector: 'app-stagiaire',
  templateUrl: './stagiaire.component.html',
  styleUrls: ['./stagiaire.component.css']
})
export class StagiaireComponent implements OnInit {

  nom = "";
  prenom = "";

  constructor(private route: ActivatedRoute) { }

  ngOnInit() {
    this.route.params.subscribe(res => {
      this.nom = res['nom']
      this.prenom = res['prenom']
      console.log(res)
    })
  }
}
```

Angular

La deuxième solution avec snapshot et paramMap

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

@Component({
  selector: 'app-stagiaire',
  templateUrl: './stagiaire.component.html',
  styleUrls: ['./stagiaire.component.css']
})
export class StagiaireComponent implements OnInit {

  nom = "";
  prenom = "";

  constructor(private route: ActivatedRoute) { }

  ngOnInit() {
    this.nom = this.route.snapshot.paramMap.get('nom') ?? ''
    this.prenom = this.route.snapshot.paramMap.get('prenom') ?? ''
  }
}
```

Angular

Ou snapshot et params

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

@Component({
  selector: 'app-stagiaire',
  templateUrl: './stagiaire.component.html',
  styleUrls: ['./stagiaire.component.css']
})
export class StagiaireComponent implements OnInit {

  nom = "";
  prenom = "";

  constructor(private route: ActivatedRoute) { }

  ngOnInit() {
    this.nom = this.route.snapshot.params['nom'];
    this.prenom = this.route.snapshot.params['prenom'];
    console.log(this.nom + ' ' + this.prenom);
  }
}
```

Angular

Pour récupérer les paramètres d'une route ayant la forme `adresse?ville=value1&rue=value2&codepostal=value3`, il faut :

- aller dans le composant concerné (ici, `adresse.component.ts`)
- faire une injection de dépendance de la classe `ActivatedRoute`
- utiliser un objet cette classe dans la méthode `ngOnInit()`
 - soit par l'intermédiaire d'un objet `queryParamMap (Map)`
 - soit par l'intermédiaire d'un objet `queryParams (Object)`
- choisir une des deux solutions suivantes
 - solution asynchrone avec les observables
 - solution synchrone avec les snapshot

Angular

Pas besoin de définir une route pour récupérer les paramètres `rue`, `codepostal` et `ville`

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

@Component({
  selector: 'app-adresse',
  templateUrl: './adresse.component.html',
  styleUrls: ['./adresse.component.css']
})
export class AdresseComponent implements OnInit {

  rue = '';
  codePostal = '';
  ville = '';

  constructor(private route: ActivatedRoute) { }

  ngOnInit(): void {
    this.route.queryParamMap.subscribe(
      res => {
        this.ville = res.get('ville') ?? '';
        this.rue = res.get('rue') ?? '';
        this.codePostal = res.get('codepostal') ?? '';
      }
    );
  }
}
```

Angular

Dans `adresse.component.html`

```
<h2>Adresse</h2>
<ul>
  <li>Rue : {{ rue }} </li>
  <li>Code Postal : {{ codePostal }} </li>
  <li>Ville : {{ ville }} </li>
</ul>
```

Angular

Ou avec l'objet `queryParams`

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

@Component({
  selector: 'app-adresse',
  templateUrl: './adresse.component.html',
  styleUrls: ['./adresse.component.css']
})
export class AdresseComponent implements OnInit {

  rue = '';
  codePostal = '';
  ville = '';

  constructor(private route: ActivatedRoute) { }

  ngOnInit(): void {
    this.route.queryParams.subscribe(res => {
      this.rue = res['rue']
      this.codePostal = res['codePostal']
      this.ville = res['ville']
    })
  }
}
```

Angular

Une première solution synchrone avec `snapshot` et `queryParams`

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

@Component({
  selector: 'app-adresse',
  templateUrl: './adresse.component.html',
  styleUrls: ['./adresse.component.css']
})
export class AdresseComponent implements OnInit {

  rue = '';
  codePostal = '';
  ville = '';

  constructor(private route: ActivatedRoute) { }

  ngOnInit(): void {
    this.rue = this.route.snapshot.queryParamMap.get('rue') ?? ''
    this.ville = this.route.snapshot.queryParamMap.get('ville') ?? ''
    this.codePostal = this.route.snapshot.queryParamMap.get('codePostal') ?? ''
  }
}
```

Angular

Ou snapshot et queryParams

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

@Component({
  selector: 'app-adresse',
  templateUrl: './adresse.component.html',
  styleUrls: ['./adresse.component.css']
})
export class AdresseComponent implements OnInit {

  rue = '';
  codePostal = '';
  ville = '';

  constructor(private route: ActivatedRoute) { }

  ngOnInit(): void {
    this.ville = this.route.snapshot.queryParams.ville;
    this.rue = this.route.snapshot.queryParams.rue;
    this.codePostal = this.route.snapshot.queryParams.codepostal;
  }
}
```

Angular

Depuis Angular 14, `inject` a été introduit pour injecter les dépendances

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

@Component({
  selector: 'app-adresse',
  templateUrl: './adresse.component.html',
  styleUrls: ['./adresse.component.css']
})
export class AdresseComponent implements OnInit {

  rue = '';
  codePostal = '';
  ville = '';

  route = inject(ActivatedRoute)

  ngOnInit(): void {
    this.ville = this.route.snapshot.queryParams.ville;
    this.rue = this.route.snapshot.queryParams.rue;
    this.codePostal = this.route.snapshot.queryParams.codepostal;
  }
}
```

Angular

Injection de dépendance par constructeur

- Design pattern connu dans les frameworks frontend et backend.
- Pratique standard dans **Angular**, largement utilisée dans la communauté.
- Facilite les tests unitaires car les dépendances peuvent être facilement remplacées par des mocks.
- Les dépendances sont déclarées explicitement, ce qui améliore la lisibilité du code.
- La déclaration des dépendances dans le constructeur peut être moins intuitif dans le cas où les dépendances ne sont pas immédiatement utilisées.

Utilisation de `inject()`

- Permet d'injecter des dépendances directement dans le corps d'une méthode ou d'un hook.
- Peut être utilisé dans des services fonctionnels.
- Moins répandu et donc moins connu, ce qui peut rendre le code moins accessible à d'autres développeurs.
- Peut compliquer les tests unitaires et rendre plus difficile le remplacement par des mocks.

Angular

Exercice

- Considérons un composant `calcul` (à créer)
- Ce composant est accessible via le chemin `calcul/:op` (à définir dans `app-routing.module.ts`)
- Les valeurs possibles de `op` sont `plus`, `moins`, `fois` et `div`
- Si l'adresse saisie dans la barre d'adresse contient `/calcul/plus?value1=2&value2=5`, la réponse attendue dans le template est `2 + 5 = 7`

Angular

Création de liens avec paramètres : deux méthodes

- On prépare le lien dans la classe et on l'affiche dans le template.
- On prépare le lien et on l'affiche dans le template.

Angular

Commençons par ajouter le lien suivant dans `app.component.html`

```
<a routerLink='{ { lienStagiaire } }'> Stagiaire avec params 1 </a>
```

© Achref EL MOUELHI ©

Angular

Commençons par ajouter le lien suivant dans `app.component.html`

```
<a routerLink='{{ lienStagiaire }}'> Stagiaire avec params 1 </a>
```

Explication

Logiquement, `{{ lienStagiaire }}` est un attribut de la classe `AppComponent`.

Angular

Commençons par ajouter le lien suivant dans `app.component.html`

```
<a routerLink='{ { lienStagiaire } }'> Stagiaire avec params 1 </a>
```

Explication

Logiquement, `{ { lienStagiaire } }` est un attribut de la classe `AppComponent`.

Dans `app.component.ts`, on ajoute un attribut `lienStagiaire`

```
param1 = 'wick';  
param2 = 'john';  
lienStagiaire = '';  
constructor() {  
    this.lienStagiaire = `/stagiaire/${this.param1}/${this.param2}`;  
}
```

Angular

Modifions les liens dans le menu pour les styler si seulement si les routes sans paramètres sont actives

```
<ul>
  <li routerLinkActive=active [routerLinkActiveOptions]="{exact: true}">
    <a routerLink=''> Accueil </a>
  </li>
  <li routerLinkActive=active [routerLinkActiveOptions]="{exact: true}">
    <a routerLink='stagiaire'> Stagiaire </a>
  </li>
  <li routerLinkActive=active [routerLinkActiveOptions]="{exact: true}">
    <a routerLink='adresse'> Adresse </a>
  </li>
</ul>
```

Angular

Une deuxième écriture avec le one way binding (property binding)

```
<a [routerLink]='lienStagiaire'> Stagiaire avec params 2 </a>
```

© Achref EL MOUËL

Angular

Une deuxième écriture avec le one way binding (property binding)

```
<a [routerLink]='lienStagiaire'> Stagiaire avec params 2 </a>
```

Remarque

Rien à changer dans la `app.component.html`.

Angular

Une troisième écriture

```
<a [routerLink]="['/stagiaire',param1, param2]">  
  Stagiaire avec params 3  
</a>
```

© Achref EL MOU

Angular

Une troisième écriture

```
<a [routerLink]="['/stagiaire',param1, param2]">  
  Stagiaire avec params 3  
</a>
```

Remarque

Rien à changer dans la `app.component.html`.

Angular

Pour construire un chemin de la forme `/chemin?param1=value1¶m2=value2`

```
<a [routerLink]="['/adresse']" [queryParams]="{ ville: 'Marseille',  
  codepostal: '13000', rue: 'paradis'}">  
  Adresse avec params  
</a>
```

Angular

Exercice

- Dans `stagiaire.component.html`, construisez un lien vers la route `/stagiaire` avec deux paramètres
- Dans `stagiaire.component.ts`, utilisez la solution snapshot puis observable pour récupérer les paramètres. Dans `stagiaire.component.html`, on affiche les paramètres.
- Vérifier, en cliquant sur le lien, que les nouveaux paramètres sont affichés

Angular

Conclusion

- Si la valeur initiale de paramètre est utilisée seulement à l'initialisation du composant et ne risque pas de changer, utilisez les snapshot.
- Si la route risque de changer tout en restant dans le même composant, utilisez les observables. L'initialisation du composant (`ngOnInit()`) ne serait donc pas appelée à nouveau, l'observateur sera notifié lorsque l'URL est modifiée.

Angular

Exercice

- Dans un nouveau composant `tableau`, déclarez un tableau `numbers = [2, 3, 8, 1]`.
- Définissez une route `tableau/:id` dans `app-routing.module.ts` (`id` étant l'indice de l'élément à afficher).
- Ajoutez deux liens `suivant` et `précédent` qui permettent de naviguer respectivement sur l'élément suivant et précédent de `numbers`.
- Les deux liens `suivant` et `précédent` doivent permettre une navigation circulaire.

Angular

Angular 16

- a simplifié la récupération des paramètres
- ~~a introduit~~ le décorateur `@Input` pour cet effet

Angular

Simplifions `stagiaire.component.ts` avec @Input

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

@Component({
  selector: 'app-stagiaire',
  templateUrl: './stagiaire.component.html',
  styleUrls: ['./stagiaire.component.css']
})
export class StagiaireComponent {

  @Input() nom?: string;
  @Input() prenom?: string;

}
```

Angular

Simplifions stagiaire.component.ts **avec** @Input

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

@Component({
  selector: 'app-stagiaire',
  templateUrl: './stagiaire.component.html',
  styleUrls: ['./stagiaire.component.css']
})
export class StagiaireComponent {

  @Input() nom?: string;
  @Input() prenom?: string;

}
```

Pas de changement dans stagiaire.component.html

Angular

De même pour `adresse.component.ts`

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

@Component({
  selector: 'app-adresse',
  templateUrl: './adresse.component.html',
  styleUrls: ['./adresse.component.css']
})
export class AdresseComponent {

  @Input() rue?: string;
  @Input() codePostal?: string;
  @Input() ville?: string;

}
```

Angular

De même pour `adresse.component.ts`

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

@Component({
  selector: 'app-adresse',
  templateUrl: './adresse.component.html',
  styleUrls: ['./adresse.component.css']
})
export class AdresseComponent {

  @Input() rue?: string;
  @Input() codePostal?: string;
  @Input() ville?: string;

}
```

Pas de changement dans `adresse.component.html`

Angular

Il nous reste d'activer ce mode de récupération de paramètre dans `app-routing.module.ts`

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes, withComponentInputBinding } from '@angular/router';
import { PereComponent } from '../components/pere/pere.component';
import { AdresseComponent } from '../components/adresse/adresse.component';
import { StagiaireComponent } from '../components/stagiaire/stagiaire.component';

const routes: Routes = [
  { path: 'pere', component: PereComponent },
  { path: 'adresse', component: AdresseComponent },
  { path: 'stagiaire', component: StagiaireComponent },
  { path: 'stagiaire/:nom/:prenom', component: StagiaireComponent },
];

@NgModule({
  imports: [RouterModule.forRoot(routes, {bindToComponentInputs: true})],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Angular

Remarque

La récupération des paramètres avec @Input permet une récupération asynchrone des données.

Angular

Pour modifier le titre en fonction du composant demandé, on utilise le service `Title`

```
import { Component, Input, OnInit } from '@angular/core';
import { Title } from '@angular/platform-browser';
import { ActivatedRoute, Router } from '@angular/router';

@Component({
  selector: 'app-adresse',
  templateUrl: './adresse.component.html',
  styleUrls: ['./adresse.component.css']
})
export class AdresseComponent implements OnInit {

  @Input() rue?: string;
  @Input() codePostal?: string;
  @Input() ville?: string;

  constructor(private router: Router, private title: Title) { }

  ngOnInit(): void {
    this.title.setTitle('Adresse')
  }
}
```

Angular

On peut aussi le faire dans le fichier de routage

```
{ path: 'stagiaire', component: StagiaireComponent, title: 'Trainee' },  
{ path: 'adresse', component: AdresseComponent, title: 'Address' },
```

Angular

Pour remettre un titre par défaut pour tous les autres composants (code à placer dans le fichier de routage)

```
routes
  .filter(elt => !elt.title)
  .map(elt => elt.title = 'CoursAngular')
```

Angular

Pour tester

- visitez la route `/adresse` et vérifiez que le `title` affiché est Adresse.
- visitez la route `/stagiaire` et vérifiez que le `title` affiché est CoursAngular.

Angular

Étapes

- Injecter la classe `Router` dans le constructeur de notre classe
- Utiliser l'objet de cette classe injectée dans n'importe quelle méthode de notre classe (`.component.ts`) pour réorienter vers un autre chemin

Angular

Dans `adresse.component.ts`

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

@Component({
  selector: 'app-adresse',
  templateUrl: './adresse.component.html',
  styleUrls: ['./adresse.component.css']
})
export class AdresseComponent {

  @Input() rue?: string;
  @Input() codePostal?: string;
  @Input() ville?: string;
  nom = 'wick';
  prenom = 'john';

  constructor(private router: Router) { }

  goToStagiaire(): void {
    this.router.navigateByUrl('/stagiaire/' + this.nom + '/' + this.prenom);
  }
}
```

Angular

Dans `adresse.component.ts`

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

@Component({
  selector: 'app-adresse',
  templateUrl: './adresse.component.html',
  styleUrls: ['./adresse.component.css']
})
export class AdresseComponent {

  @Input() rue?: string;
  @Input() codePostal?: string;
  @Input() ville?: string;
  nom = 'wick';
  prenom = 'john';

  constructor(private router: Router) { }

  goToStagiaire(): void {
    this.router.navigateByUrl('/stagiaire/' + this.nom + '/' + this.prenom);
  }
}
```

En appelant la méthode `goToStagiaire()`, on sera redirigé vers `/stagiaire/john/wick`.

Angular

On peut aussi utiliser la méthode `navigate()`

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

@Component({
  selector: 'app-adresse',
  templateUrl: './adresse.component.html',
  styleUrls: ['./adresse.component.css']
})
export class AdresseComponent {

  @Input() rue?: string;
  @Input() codePostal?: string;
  @Input() ville?: string;
  nom = 'wick';
  prenom = 'john';

  constructor(private router: Router) { }

  goToStagiaire() {
    this.router.navigate(['/stagiaire', this.nom , this.prenom]);
  }
}
```

Angular

Pour rediriger vers une route acceptant des `queryParams`

```
gotoMarseille() {  
  this.router.navigate(  
    ['/adresse'],  
    { queryParams: { ville: 'Marseille', codePostal: '13008' } }  
  )  
}
```

Angular

On peut rediriger vers un chemin existant

```
const routes: Routes = [  
  { path: 'stagiaire', component: StagiaireComponent },  
  { path: 'stagiaire/:param1/:param2', component: StagiaireComponent },  
  { path: 'adresse', component: AdresseComponent },  
  { path: 'trainee', redirectTo: '/stagiaire' }  
];
```

Angular

Ou pour un lien avec paramètres

```
const routes: Routes = [  
  { path: 'stagiaire', component: StagiaireComponent },  
  { path: 'stagiaire/:param1/:param2', component: StagiaireComponent },  
  { path: 'adresse', component: AdresseComponent },  
  { path: 'trainee', redirectTo: '/stagiaire' },  
  { path: 'trainee/:param1/:param2', redirectTo: '/stagiaire/:param1/:  
    param2' },  
];
```

Angular

On peut créer un chemin vide pour que l'URL `localhost:4200` soit accessible

```
const routes: Routes = [  
  { path: 'stagiaire', component: StagiaireComponent },  
  { path: 'stagiaire/:param1/:param2', component: StagiaireComponent },  
  { path: 'adresse', component: AdresseComponent },  
  { path: 'trainee', redirectTo: '/stagiaire' },  
  { path: '', redirectTo: '/stagiaire', pathMatch: 'full' }  
];
```

© Achref EL MOU

Angular

On peut créer un chemin vide pour que l'URL `localhost:4200` soit accessible

```
const routes: Routes = [  
  { path: 'stagiaire', component: StagiaireComponent },  
  { path: 'stagiaire/:param1/:param2', component: StagiaireComponent },  
  { path: 'adresse', component: AdresseComponent },  
  { path: 'trainee', redirectTo: '/stagiaire' },  
  { path: '', redirectTo: '/stagiaire', pathMatch: 'full' }  
];
```

Remarque

- Sans la partie `pathMatch: 'full'` (pour les chemins vides), toutes les routes déclarées après cette dernière ne seront pas accessibles.
- `pathMatch: 'full'` ne laisse donc passer que les requêtes dont le chemin correspond exactement au chemin vide
- La deuxième valeur possible pour `pathMatch` est `'prefix'`

Angular

On peut créer un composant `error` et l'afficher en cas de chemin inexistant

```
const routes: Routes = [  
  { path: 'stagiaire', component: StagiaireComponent },  
  { path: 'stagiaire/:nom/:prenom', component: StagiaireComponent },  
  { path: 'adresse', component: AdresseComponent },  
  { path: 'trainee', redirectTo: '/stagiaire' },  
  { path: 'error', component: ErrorComponent },  
  { path: '**', redirectTo: '/error' }  
];
```

© Achre

Angular

On peut créer un composant `error` et l'afficher en cas de chemin inexistant

```
const routes: Routes = [  
  { path: 'stagiaire', component: StagiaireComponent },  
  { path: 'stagiaire/:nom/:prenom', component: StagiaireComponent },  
  { path: 'adresse', component: AdresseComponent },  
  { path: 'trainee', redirectTo: '/stagiaire' },  
  { path: 'error', component: ErrorComponent },  
  { path: '**', redirectTo: '/error' }  
];
```

Le chemin `**` doit être le dernier. Autrement, toutes les requêtes seront redirigées vers le composant `error`.

Angular

Remarques

- En cliquant sur `Accueil` dans le menu, c'est le composant `Error` qui s'affiche.
- La route vide n'existe plus.

© Achref EL MOUELHI ©

Angular

Remarques

- En cliquant sur `Accueil` dans le menu, c'est le composant `Error` qui s'affiche.
- La route vide n'existe plus.

Problème

Si on définit la route `{ path: '', component: AppComponent }`, le composant principal sera affiché deux fois en demandant la route vide.

Angular

Remarques

- En cliquant sur `Accueil` dans le menu, c'est le composant `Error` qui s'affiche.
- La route vide n'existe plus.

Problème

Si on définit la route `{ path: '', component: AppComponent }`, le composant principal sera affiché deux fois en demandant la route vide.

Solution

Créer un composant `Home` et l'associer à la route vide.

Angular

Commençons par créer le composant `home`

```
ng g c composants/home
```

© Achref EL MOUËLHI

Angular

Commençons par créer le composant `home`

```
ng g c composants/home
```

Résultat

```
CREATE src/app/composants/home/home.component.html (19 bytes) CREATE  
src/app/composants/home/home.component.spec.ts (612 bytes) CREATE  
src/app/composants/home/home.component.ts (267 bytes) CREATE  
src/app/composants/home/home.component.css (0 bytes) UPDATE  
src/app/app.module.ts (1048 bytes)
```

Angular

Ensuite

Déplaçons les contenus respectifs de `app.component.ts` et `app.component.html` dans `home.component.ts` et `home.component.html`.

Dans `app.component.html`, **gardons seulement le contenu suivant**

```
<app-menu></app-menu>  
<router-outlet></router-outlet>
```

Angular

Assurez vous d'avoir le tableau routes suivant dans `app-routing.module.ts` avant de tester l'URL `localhost:4200`

```
const routes: Routes = [  
  { path: '', component: HomeComponent },  
  { path: 'stagiaire', component: StagiaireComponent },  
  { path: 'stagiaire/:nom/:prenom', component: StagiaireComponent },  
  { path: 'adresse', component: AdresseComponent },  
  { path: 'trainee', redirectTo: '/stagiaire' },  
  { path: 'error', component: ErrorComponent },  
  { path: '**', redirectTo: '/error' }  
];
```