Angular : 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

Angular

Pour créer un nouveau composant

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

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

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
```

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
```

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
```

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>
```

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 méthode 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.

Angular

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

Angular

Comment faire pour tester ?

Il faut saisir

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

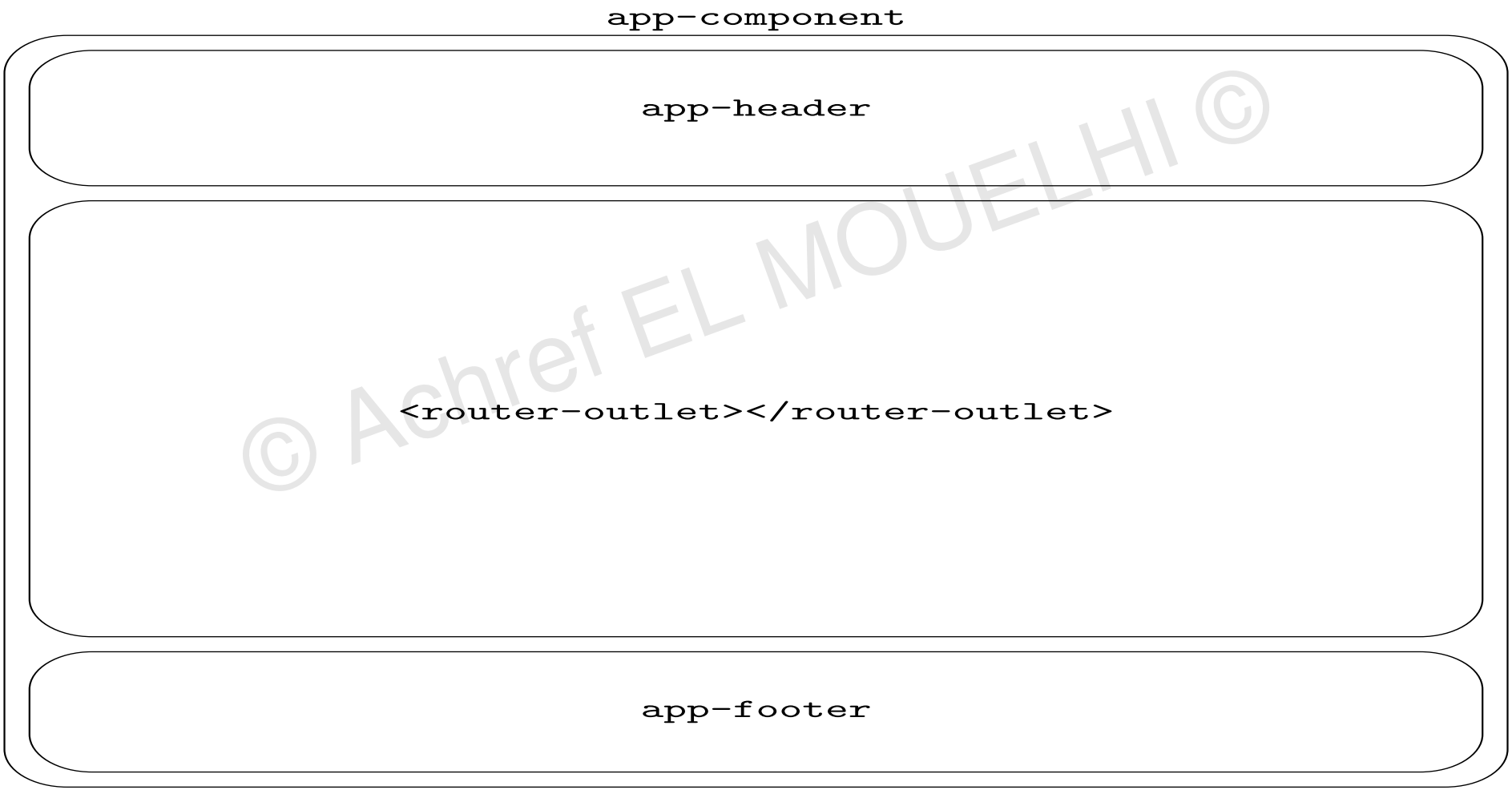
dans la barre d'adresse du navigateur

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**



Angular

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>
```

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

Angular

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>
```


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>
    <a routerLink='' routerLinkActive=active [
      routerLinkActiveOptions]="{ exact: true }"> 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
```

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 },
  ]),
  AppRoutingModule
],
```

Le fichier `app-routing.module.ts`

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

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

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 components

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 components

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`

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` pour `/chemin/param1/param2`
  - `queryParamMap` 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 paramètre d'une route de 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 cette classe dans la méthode `ngOnInit()`
  - soit par l'intermédiaire d'un objet `paramMap` pour récupérer les paramètres (solution avec les observables (asynchrone))
  - soit par l'intermédiaire d'un objet `params` pour récupérer les paramètres (solution avec les snapshot (instantanée))

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

```
import { ActivatedRoute } from '@angular/router';
// + les autres imports de components

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

  nom: any;
  prenom: any;

  constructor(private route: ActivatedRoute) { }

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

Angular

Dans `stagiaire.component.html`

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

Angular

`constructor` et `ngOnInit`

- `constructor` : fonction JavaScript qui sert à initialiser les attributs d'une classe
- `constructor` avec Angular sert seulement à faire les injections de dépendances
- `ngOnInit` : méthode exécutée quand Angular a fini d'initialiser le composant (charger `@Input()` ...)

Angular

La deuxième solution avec `snapshot`

```
import { ActivatedRoute } from '@angular/router';
// + les autres imports de components

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

  nom: any;
  prenom: any;

  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 de 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` pour récupérer les paramètres (solution avec les observables)
  - soit par l'intermédiaire d'un objet `queryParams` pour récupérer les paramètres (solution avec les snapshot)

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

```
import { ActivatedRoute } from '@angular/router';
// + les autres imports de components

@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>
```

La deuxième solution avec `snapshot`

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

// + les autres imports de components

@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

Une première méthode classique en HTML

```
<ul>
  <li><a routerLink=''> Accueil </a></li>
  <li>
    <a routerLink='{{ lienStagiaire }}'> Stagiaire </a>
  </li>
  <li><a routerLink='/adresse'> Adresse </a></li>
</ul>
```


On comprend de `{{ lienStagiaire }}` qu'il existe un attribut `lienStagiaire` dans `menu.component.ts`

Angular

Dans `menu.component.ts`

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

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

  lienStagiaire = '';
  param1 = 'john';
  param2 = 'wick';

  constructor() {
    this.lienStagiaire = '/stagiaire/' + this.param1 + '/' + this.
      param2;
  }
  ngOnInit(): void {

  }
}
```

Angular

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

```
<ul>
  <li><a routerLink=''> Accueil </a> </li>
  <li><a [routerLink]='lienStagiaire'> Stagiaire </a
    ></li>
  <li><a routerLink='/adresse'> Adresse </a></li>
</ul>
```

Angular

Une troisième écriture

```
<ul>
  <li>
    <a routerLink=''> Accueil </a>
  </li>
  <li>
    <a [routerLink]="['/stagiaire',param1,param2]">
       Stagiaire </a>
  </li>
  <li>
    <a routerLink='/adresse'> Adresse </a>
  </li>
</ul>
```

Angular

Pour construire un chemin de la forme `/chemin?param1=value1&param2=value2`

```
<ul>
  <li>
    <a routerLink=''> Accueil </a>
  </li>

  <li>
    <a [routerLink]="['/stagiaire',param1,param2]"> Stagiaire </a>
  </li>

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

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 a été modifiée.

Angular

Exercice

- Créez un composant `calcul` et définissez une route `calcul/:op` dans `app-routing.module.ts` (`op` peut contenir principalement les valeurs `plus`, `moins`, `fois` et `div`)
- Si l'adresse saisie dans la barre d'adresse est `/calcul/plus?value1=2&value2=5`, la réponse attendue est donc `2 + 5 = 7`

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 { Router } from '@angular/router';
// + les autres imports de components

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

  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 { Router } from '@angular/router';
// + les autres imports de components

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

  nom = 'wick';
  prenom = 'john';

  constructor(private router: Router) { }

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

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

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: '/stagiaire', pathMatch: 'full'},
  { path: '**', redirectTo: '/error' }
];
```

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: '/stagiaire', pathMatch: 'full'},
  { path: '**', redirectTo: '/error' }
];
```

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

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.