

Cours 5 : Frontend – partie 2



Applications web et mobiles

Christophe Gonzales

Quelques directives



Afficher une liste de cours dans l'appli :

- 1 Stocker la liste des cours dans le composant `Courses`
 - 2 Dans le template de `Courses`, itérer l'insertion des cours avec la directive `*ngFor`
- ⚠ Toutes les directives (`*ngFor`, `*ngIf`, *etc.*) débutent par une *
- ⚠ Peut nécessiter l'inclusion de `CommonModule` dans les `imports` de `courses.component.ts`
- ▶ `*ngIf` : le composant est utilisé si et seulement si `ngIf=true`

La directive *ngFor dans le TypeScript

```
File Edit View Navigate Code Refactor Run Tools Git Window Help
obile_cours > src > app > courses > courses.component.ts Angular CLI Server Git:
Project courses.component.ts x
  courses
    courses.component.html
    courses.component.scss
    courses.component.spec.ts
    courses.component.ts
  services
    courses.service.spec.ts
    courses.service.ts
  app.component.html
  app.component.scss
  app.component.spec.ts
  app.component.ts
  app.config.ts
  app.routes.ts
  assets
    favicon.ico
    index.html
    main.ts
    styles.scss
  .editorconfig
  .gitignore
  angular.json
  getCourses.php
  package.json
  package-lock.json
  README.md

import {Component, OnInit} from '@angular/core';
import {FormsModule} from "@angular/forms";
import {Course, CoursesService} from "../services/courses.service";
import {CommonModule} from "@angular/common";

@Component({
  selector: 'app-courses',
  standalone: true,
  // ici, ne pas oublier d'inclure le CommonModule :
  imports: [FormsModule, CommonModule],
  templateUrl: './courses.component.html',
  styleUrls: ['./courses.component.scss']
})
export class CoursesComponent implements OnInit {
  titre = 'liste de cours';
  UE : Course[] = []; // on n'initialise avec une liste vide!

  constructor(private service: CoursesService) {}

  ngOnInit() {
    this.service.getCourses().subscribe(
      courses => {

```

La directive *ngFor dans le template HTML

```
File Edit View Navigate Code Refactor Run Tools Git Window Help
le_cours > src > app > courses > courses.component.html Angular CLI Server
courses.component.html x
<p>{{getTitle()}}</p>
<ul>
  <!-- directive *ngFor : on parcourt tous les éléments du tableau UE. Chaque élément
    est stocké dans la variable cours. On peut alors utiliser cette variable,
    notamment dans les interpolations et property bindings -->
  <li *ngFor="let cours of UE">
    {{cours.nom}} : {{cours.nb_etuds}} étuds
  </li>
</ul>
<!-- Attention : on ne peut pas appliquer 2 directives (par exemple *ngIf et *ngFor
à la même balise. Il faut choisir : soit utiliser *ngIf, soit *ngFor.
Si on a besoin des 2, l'astuce est de créer un composant ou une balise
supplémentaire (par exemple un span). On appliquera à l'un des composant le
*ngFor et au composant supplémentaire le *ngIf. -->
```

Le résultat

XmobileCours 127.0.0.1:4200

Ceci est mon composant App

liste de cours

- c1 : 2 étuds
- c2 : 5 étuds
- c3 : 6 étuds

liste de cours

- c1 : 2 étuds
- c2 : 5 étuds
- c3 : 6 étuds

Console

```
[vite] connecting...
[vite] connected.
Angular is running in development mode.
```

client:229
client:324
core.mjs:29995

La directive *ngIf

The screenshot shows a code editor with the following content:

```
<p>{{getTitle()}}</p>
<ul>
  <li *ngFor="let cours of UE">
    {{cours.nom}} : {{cours.nb_etuds}} étuds :
    <!-- Ici, la page web ne contiendra qu'un seul des deux span ci-dessous.
    Si cours contient moins de 5 étudiants, ce sera le premier span, sinon
    ce sera le 2ème. Quand je dis "la page web ne contiendra...", je parle
    du DOM : le DOM ne contiendra effectivement qu'un seul span et non deux,
    comme cela aurait été le cas si on avait utilisé des hidden. -->
    <span *ngIf="cours.nb_etuds < 5">(petit cours)</span>
    <span *ngIf="cours.nb_etuds >= 5">(gros cours)</span>
  </li>
</ul>
```

The interface includes a menu bar (File, Edit, View, Navigate, Code, Refactor, Run, Tools, Git, Window, Help), a toolbar with icons for running, debugging, and Git, and a sidebar with Project, Commit, Bookmarks, and npm views. The status bar at the bottom shows 'Externally added files can be added to Git // View Files // Always... (today 14:01)', '5:68 LF UTF-8', 'TypeScript 5.2.2 2 spaces*', and 'main'.

La directive *ngIf : le résultat

XmobileCours 127.0.0.1:4200

Ceci est mon composant App

liste de cours

- c1 : 2 étuds : (petit cours)
- c2 : 5 étuds : (gros cours)
- c3 : 6 étuds : (gros cours)

liste de cours

- c1 : 2 étuds : (petit cours)
- c2 : 5 étuds : (gros cours)
- c3 : 6 étuds : (gros cours)

Console

```
[vite] connecting...
[vite] connected.
Angular is running in development mode.
```

client:229
client:324
core.mjs:29995

La navigation

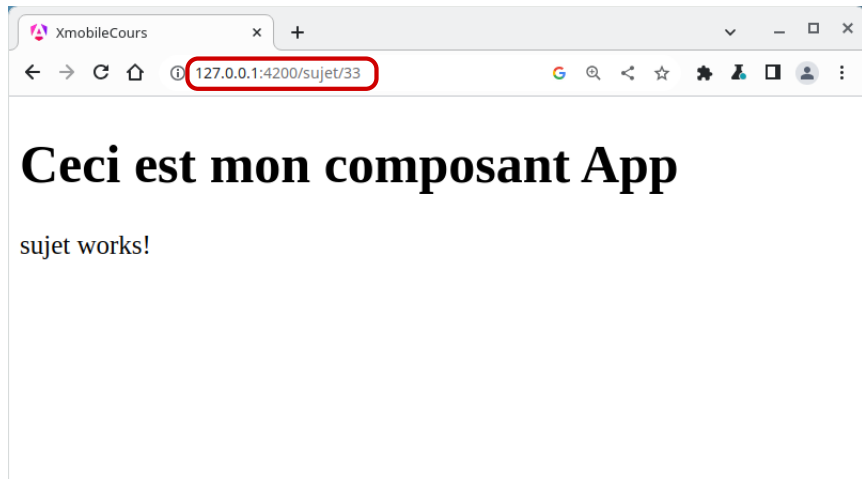
Navigation : les routes dans app.routes.ts

The screenshot shows an IDE window with the following content:

- File Explorer:** xmobile_cours > src > app > app.routes.ts
- Code Editor:**

```
import { Routes } from '@angular/router';
import { CoursesComponent } from "../courses/courses.component";
import { SujetComponent } from "../sujet/sujet.component";

// ici, on indique les routes de l'application.
// path = l'URL pour accéder au composant
export const routes: Routes = [
  {path: 'cours', component: CoursesComponent},
  // si, dans le path, il y a des ":", cela indique que ce sont
  // des paramètres, dont on pourra récupérer la valeur. Ainsi,
  // si l'utilisateur accède à l'URL /sujet/33, le paramètre id
  // aura la valeur 33
  {path: 'sujet/:id', component: SujetComponent}
];
```
- Structure:**
 - app.routes.ts
 - sujet.component.html
 - <p>sujet works!</p>
 - app.component.html
 - <h1>Ceci est mon composant App</h1>
 - <router-outlet></router-outlet>
- Terminal/Services:** Git, TODO, Problems, Terminal, Services
- Status Bar:** 7:32 LF UTF-8 TypeScript 5.2.2 2 spaces* main



RouterLink : le lien de navigation entre les pages

The image shows a code editor with the following HTML code in `courses.component.html`:

```
<!-- on rajoute ici une barre de navigation. Surtout, NE PAS UTILISER href pour
passer d'une page à l'autre : cela rechargerait toute l'application. Ici, il
faut utiliser la propriété "routerLink" qui permet à Angular de modifier
la vue sans rechargement. -->
<nav>
  <a routerLink="/sujet/33">Mes sujets</a>
</nav>

<p>{{getTitle()}}</p>
<ul>
  <li *ngFor="let cours of UE">
    {{cours.nom}} : {{cours.nb_etuds}} étuds :
    <span *ngIf="cours.nb_etuds < 5">(petit cours)</span>
    <span *ngIf="cours.nb_etuds >= 5">(gros cours)</span>
  </li>
</ul>
```

Two browser screenshots are overlaid on the code editor:

- The top browser screenshot shows the URL `127.0.0.1:4200/cours` and the page content: **Ceci est mon composant**, a link [Mes sujets](#), and a list of courses: **liste de cours** with items `c1 : 2 étuds : (petit cours)` and `c2 : 5 étuds : (gros cours)`.
- The bottom browser screenshot shows the URL `127.0.0.1:4200/sujet/33` and the page content: **Ceci est mon composant App** and `sujet works!`.

Les liens paramétrés

The image shows a development environment with two files open: `sujet.component.ts` and `sujet.component.html`.

```
export class SujetComponent implements OnInit {
  my_id!: number;

  // le constructeur récupère par dependency injection la route qui a mené à lui
  // => on va pouvoir récupérer les paramètres passés dans l'URL
  constructor(private route: ActivatedRoute) {}

  ngOnInit() {
    // ici, on récupère le paramètre de la route
    this.my_id = this.route.snapshot.params['id'];
  }
}
```

```
<!-- Ici, par interpolation, on affiche l'id qui a été passé en paramètre de l'URL. Pour
cela, le Typescript doit récupérer l'information via une instance de ActivatedRoute -->
<p>mon sujet {{my_id}}</p>
```

The browser preview shows the rendered output of the HTML template:

XmobileCours

127.0.0.1:4200/sujet/33

Ceci est mon composant App

mon sujet 33