Travaux Dirigé pas à pas : Créez votre Portfolio avec Angular

Objectifs

Ce TD vous guide étape par étape pour créer un portfolio web avec Angular. Vous apprendrez à :

- Installer et configurer Angular
- Créer et organiser des composants
- Mettre en place le routage
- Gérer les styles avec SCSS
- Charger des données dynamiques

Prérequis

- Node.js et npm installés (télécharger ici)
- Visual Studio Code ou un éditeur équivalent

1. Création du projet Angular

1. Ouvrez un terminal et tapez :

   ```
   npx @angular/cli new portfolio --style=scss --routing=true
   ```

   - **Nom du projet** : mini_td_angular
   - **Style** : SCSS
   - **Routage** : Oui
2. Entrez dans le dossier du projet :

   ```
   cd mini_td_angular
   ```

3. Lancez le serveur de développement :

   ```
   npm start
   ```

Ouvrez votre navigateur sur http://localhost:4200

2. Création des composants

Nous allons créer les composants suivants :

- header (barre de navigation)
- homepage (page d'accueil)
- showroom (projets)
- resume (CV)
- resume-card-body (détail du CV)
- resume-presentation-card (présentation du CV)
- sliding-resume-textfield (champ animé)

Pour chaque composant, tapez dans le terminal :

```
ng generate component header
ng generate component homepage
ng generate component showroom
ng generate component resume
ng generate component resume-card-body
ng generate component resume-presentation-card
ng generate component sliding-resume-textfield
```

Chaque commande crée un dossier dans `src/app/` avec 4 fichiers :

- `.ts` (logique)
- `.html` (vue)
- `.scss` (styles)
- `.spec.ts` (tests)

3. Mise en place du routage

1. Ouvrez `src/app/app.routes.ts` (ou `app-routing.module.ts` selon la version Angular).
2. Importez vos composants :

```
import { HomepageComponent } from './homepage/homepage.component';
import { ShowroomComponent } from './showroom/showroom.component';
import { ResumeComponent } from './resume/resume.component';
// ...
```

3. Définissez les routes :

```
const routes: Routes = [
   { path: '', component: HomepageComponent },
   { path: 'showroom', component: ShowroomComponent },
   { path: 'resume', component: ResumeComponent },
];
```

4. Dans `app.component.html` , remplacez le contenu par :

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

Cela affichera la barre de navigation et la page courante.

4. Création de la barre de navigation (header)

Dans `src/app/header/header.component.html` :

```
<nav>
   <a href="homepage">SENKAYA Mikrail</a>
   <div>
      <ul>
         <li>
         <a href="resume">Mon CV</a>
         </li>
         <li>
         <a href="showroom">Mes réalisations</a>
         </li>
      </ul>
   </div>
   </div>
</nav>
```

On ne s'occupes pas du fichier de style `src/app/header/header.component.scss` .
C'est Bootstrap qui va s'en occuper à notre place.

Pour utiliser Bootstrap il faut déjà l'installer. Pour cela notre un terminal powershell nous allons taper la commande suivante :

```
npm install bootstrap@5.3.8
```

Après avoir installé Bootstrap il faudra indiquer au projet quelles fonctionnalités de Bootstrap on veu utiliser, dans notre cas ce sont les parties style et script qui nous intéresse.

Pour cela on va se rendre dans le fichier angular.json et ajouter l'indication suivante dans le fichier :

```
"styles": [
   "node_modules/bootstrap/dist/css/bootstrap.min.css",
   "src/styles.scss"
],
"scripts": [
   "node_modules/bootstrap/dist/js/bootstrap.min.js"
],
```

Maintenant Bootstrap peut gérer le style de nos page via les mot-clés que nous indiquons dans la classe nos balises :

```
<nav class="navbar navbar-expand-lg bg-body-tertiary">
   <div class="container-fluid">
   <a class="navbar-brand" href="homepage">SENKAYA Mikrail</a>
   <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navb
      <span class="navbar-toggler-icon"></span>
   </button>
   <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
         <li class="nav-item">
         <a class="nav-link" href="resume">Mon CV</a>
         </li>
         <li class="nav-item">
         <a class="nav-link" href="showroom">Mes réalisations</a>
         </li>
      </ul>
   </div>
   </div>
</nav>
```

5. Création de la page d’accueil (homepage)

Dans `src/app/homepage/homepage.component.html` :

```
<h1>Bienvenue sur mon portfolio !</h1>
<p>Découvrez mes projets et mon parcours.</p>
```

Vous pouvez personaliser cette page via Bootstrap ou vous pouvez le modifier directement depuis le fichier SCSS. Vous pouvez également ajouter un carousel d'image pour rendre votre page d'accueil plus dynamique si vous le souhaitez.

6. Création de la page Showroom

La page Showroom est la page qui vous permettra de mettre en avant vos divers projet. Pour cela nous alons utiliser un carousel grâce à Bootstrap afin d'afficher tous vos projets. Chaque projet correspondra à une page du carousel.

Vu que l'on a pas de serveur pouvant stocker les informations toutes les informations seront écrites en dur

Dans `src/app/showroom/showroom.component.html` :

```
<div id="carouselExample" class="carousel slide">
    <div class="carousel-inner">
        <div class="carousel-item active">
            <div class="card text-center">
                <div class="card d-flex align-items-center justify-content-center">
                    <div class="card" style="width: 18rem;">
                        <div class="card-body">
                            <h5 class="card-title">TD Angular</h5>
                            <p class="card-text"></p>
                            <a href="#" class="btn btn-primary">Voir mon projet</a>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <div class="carousel-item">
            <div class="card text-center">
                <div class="card d-flex align-items-center justify-content-center">
                    <div class="card" style="width: 18rem;">
                        <div class="card-body">
                            <h5 class="card-title">Mon dessin</h5>
                            <p class="card-text"></p>
                            <a href="#" class="btn btn-primary">Voir ma création</a>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <button class="carousel-control-prev bg-primary" type="button" data-bs-target="#carouselExa
        <span class="carousel-control-prev-icon" aria-hidden="false"></span>
        <span class="visually-hidden">Previous</span>
    </button>
    <button class="carousel-control-next bg-primary" type="button" data-bs-target="#carouselExa
        <span class="carousel-control-next-icon " aria-hidden="false"></span>
        <span class="visually-hidden">Next</span>
    </button>
</div>
```

7. Création de la page CV (resume)

On arrive à la page CV.

Cette page va vous permettre de mettre un petit résumé de vous, puis d'afficher chacune des activités professionnelles que vous avez déjà eu.

Pour cela la page se découpe en sous composant afin que nous puissions voir comment passer les données d'un composant parent à ses enfants.

Pour cela modifions notre composant Resume.

Commençons par la partie html `src/app/resume/resume.component.html` :

```
<sliding_resume_textfield
   header_presentation_text="Mon CV : Présentation"
   header_scholar_text="Mon CV : Scolarité"
   header_pro_text="Mon CV : Experience Professionnelle"
   [personalInformation]="personalInformation"
   [scholarParcours]="scholarParcours"
   [professionalXp]="professionalXp"
></sliding_resume_textfield>
```

Nous retrouvons le `sliding_resume_textfield` que nous avons généré au début. Nous reviendrons sur lui un peu après. Ce que l'on peut noter c'est que nous lui passons divers données. Allons récupérer ces données

8. Ajout de données dynamiques pour le CV

1. Créez un fichier `resume.json` dans `src/app/resume/JSON/` :

```
{
"information_personelle": {
    "nom": "SENKAYA Mikrail",
    "date_naissance": "30/10/1998",
    "ville" : "Dijon",
    "hobbies" : "Jeux Vidéos, Maquette, Manga"
},
"parcours_scolaire": [
    {
        "nom_ecole": "ESIREM",
        "nom_formation": "ILC",
        "plage_annee_cursus": "2019-2022"
    },
    {
        "nom_ecole": "DUT",
        "nom_formation": "GEII",
        "plage_annee_cursus": "2018-2019"
    }
],
"experience_pro" : [
    {
        "poste": "Ingénieur en développement",
        "entreprise": "Alteca",
        "plage_annee_travaillee": "2022-2025"
    },
    {
        "poste": "Stagiaire",
        "entreprise": "Alteca",
        "plage_annee_travaillee": "2022"
    }
]
}
```

2. Dans `resume.component.ts` , chargez ce fichier avec HttpClient :

```
import { HttpClient } from '@angular/common/http';
// ...
export class ResumeComponent implements OnInit {
     json = require('./JSON/resume.json');

     personalInformation = this.json["information_personelle"];
     scholarParcours = this.json["parcours_scolaire"];
     professionalXp = this.json["experience_pro"];
}
```

Maintenant que nous avons les données allons voir comment sont géré les données dans notre `sliding_resume_textfield`

3. Analysons les données dans le HTML de notre composant `sliding_resume_textfield` :

```
<div id="carouselExample" class="carousel slide">
  <div class="carousel-inner">
     <div class="carousel-item active">
           <div class="card d-flex align-items-center justify-content-center">
              <div class="card-body">
                 <resume_presentation_card
                       [body_title_text]="header_presentation_text"
                       [body_name_text]="personalInformation.nom"
                       [body_birthdate_text]="personalInformation.date_naissance"
                       [body_living_text]="personalInformation.ville"
                       [body_hobbies_text]="personalInformation.hobbies"
                 ></resume_presentation_card>
              </div>
           </div>
     </div>

     <div class="carousel-item">
           <div class="card d-flex align-items-center justify-content-center">
              <div class="card-header">
                 {{header_scholar_text}}
              </div>
              <div class="card-body" *ngFor="let scholarInfo of scholarParcours">
                 <resume_card_body
                       [body_title_text]="scholarInfo.nom_ecole"
                       [body_main_text]="scholarInfo.nom_formation"
                       [body_footer_text]="scholarInfo.plage_annee_cursus"
                 ></resume_card_body>
              </div>
           </div>
     </div>

     <div class="carousel-item">
           <div class="card d-flex align-items-center justify-content-center">
              <div class="card-header">
                 {{header_pro_text}}
              </div>
              <div class="card-body" *ngFor="let proXp of professionalXp">
                 <resume_card_body
                       [body_title_text]="proXp.entreprise"
                       [body_main_text]="proXp.poste"
                       [body_footer_text]="proXp.plage_annee_travaillee"
                 ></resume_card_body>
              </div>
           </div>
     </div>
  </div>
```

```
        <button class="carousel-control-prev bg-primary" type="button" data-bs-target="#carousel
            <span class="carousel-control-prev-icon" aria-hidden="false"></span>
            <span class="visually-hidden">Previous</span>
        </button>
        <button class="carousel-control-next bg-primary" type="button" data-bs-target="#carousel
            <span class="carousel-control-next-icon " aria-hidden="false"></span>
            <span class="visually-hidden">Next</span>
        </button>
    </div>
```

Nous pouvons voir que notre composant fait appel à deux autres composants que nous avons générés :

- resume_card_body
- resume_presentation_card

Ils prennent des variables en entrée de la même manière que notre composant. On détaillera ces composants juste après.

Les variables que ce composant passe aux composants qu'ils appellent sont bien les variables que notre composant `resume` lui a passé. Cependant comment les récupèrent-ils ?

C'est très simple, mais pour le comprendre il faut aller dans le `sliding_resume_textfield.ts`

```
export class sliding_resume_textfield{
    @Input()
    header_presentation_text : string = "Card Header";
    @Input()
    header_scholar_text : string = "Card Header";
    @Input()
    header_pro_text : string = "Card Header";
    @Input()
    personalInformation!: any;
    @Input()
    scholarParcours!: any[];
    @Input()
    professionalXp!: any[];
}
```

Comme nous pouvons le voir, il suffit d'indiquer `@Input()` avant notre variable pour indiquer aux composants qui intègre notre `sliding_resume_textfield` qu'il y a des champs d'entrée.

Ces variables pouvant soit être obligatoire, dans quel cas on met une valeur par defaut obligatoirement. Sinon il faut indiquer que notre variable est optionnelle en mettant un `!` juste après

le nom de notre variable.

C'est ainsi que nous récupérons nos valeurs du composant `resume` et que nous les renvoyons sur nos composants `resume_card_body` et `resume_presentation_card`

Pour finaliser le portfolio, nous avons besoin de nos deux derniers composants, voici le code :

resume_card_body.html

```
<div class="card text-center">
    <div class="card-body">
        <h5 class="card-title">{{body_title_text}}</h5>
        <p class="card-text">{{body_main_text}}</p>
        <p class="card-text">{{body_footer_text}}</p>
    </div>
</div>
```

resume_card_body.ts

```
export class resume_card_body{
    @Input()
    body_title_text : string = "Title";
    @Input()
    body_main_text : string = "Body";
    @Input()
    body_footer_text : string = "Footer";
}
```

resume_presentation_card.html

```
<div class="card text-center">
    <div class="card-body">
        <h5 class="card-title">{{body_title_text}}</h5>
        <p class="card-text">{{body_name_text}}</p>
        <p class="card-text">{{body_birthdate_text}}</p>
        <p class="card-text">{{body_living_text}}</p>
        <p class="card-text">{{body_hobbies_text}}</p>
    </div>
</div>
```

resume_presentation_card.ts

```
export class resume_presentation_card{
    @Input()
    body_title_text : string = "Title";
    @Input()
    body_name_text : string = "Name";
    @Input()
    body_birthdate_text : string = "Birthdate";
    @Input()
    body_living_text : string = "Addresses";
    @Input()
    body_hobbies_text : string = "Hobbies";
}
```

9. Tester et personnaliser

- Ajoutez du contenu et des styles à chaque composant.
- Testez l’affichage des images et des données dynamiques.