Angular

Les Composants

Par Rafael Da Silva Mesquita

Qu'est ce qu'un composant ?

Dans Angular, un composant est essentiellement un bloc de construction pour les interfaces utilisateur, encapsulant la logique, la présentation, et le style d'une partie spécifique de votre application web. Il est constitué d'une classe TypeScript décorée avec @Component, où vous définissez les comportements, couplées à un template HTML qui décrit la vue, et optionnellement un fichier CSS pour les styles. Les composants permettent une approche modulaire du développement web, facilitant la réutilisation, la maintenance et le test des parties de l'interface utilisateur. Ils interagissent entre eux et avec les services

pour créer des applications dynamiques et réactives, en structurant l'application en petites unités de gestion facile qui travaillent ensemble pour présenter une application riche et interactive.

Prérequis pour pouvoir mettre en place des composants :

Pour créer des composants, les utilisateurs doivent remplir quelques conditions préalables.

1. Pour l'utilisation des composants, il est impératif d'avoir une version à jour de Node.js ainsi que de npm, ce dernier étant essentiel pour la gestion des packages et des dépendances au sein des projets Angular. Il incombe à l'utilisateur de veiller à installer la version la plus récente et stable de Node.js, laquelle comprendra automatiquement npm (pour obtenir la dernière version : https://nodejs.org/en/download).
2. Pour pouvoir créer et gérer des composants, il est nécessaire de disposer d'un projet Angular ainsi que de l'interface en ligne de commande (CLI) d'Angular. Si vous n'avez pas encore de projet, référez-vous à la procédure "comment créer son projet" où tout est expliqué en détail.

Les composants Angular sont situés dans le répertoire src/app de votre projet.

Étape 1 : Modification dans le répertoire app

Pour commencer, modifiez le fichier app.components.ts de la manière suivante :

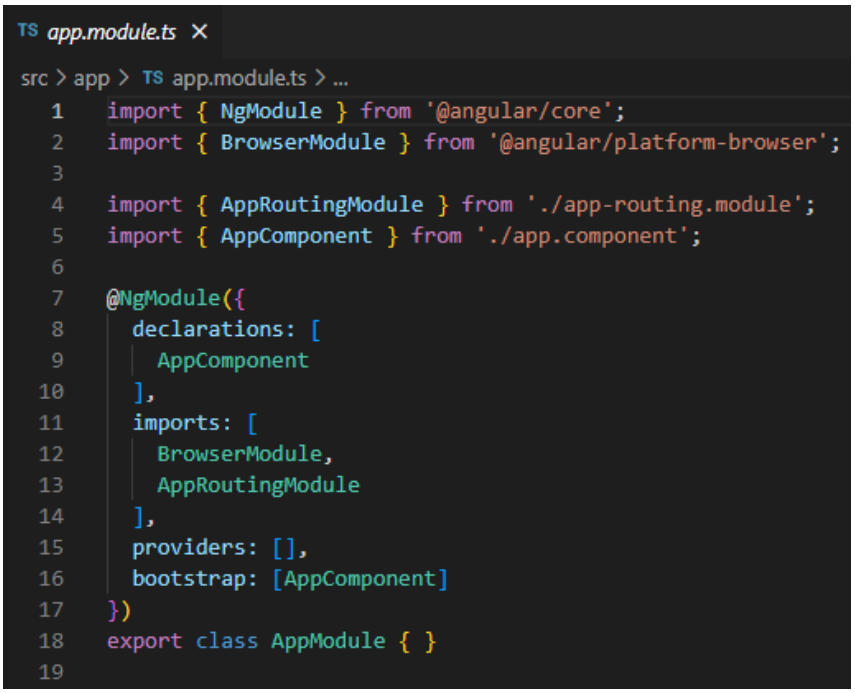
app.component.ts M

src > app > app.component.ts > AppComponent

```ts
import { Component } from '@angular/core'; // importation du coeur d'Angular

@Component({
  selector: 'app-root', // Le nom de la balise Html du composant
  template: `<h1>Welcome to {{title}}! </h1>` //Le contenu du composant contenu entre des backticks (`)
})
export class AppComponent { // "export" permet de rendre accessible le composant depuis ailleurs dans l'application
  title = 'nom_du_projet'; // Propriété du composant qui est ensuite utilisé dans le template entre accolades plus haut
}
```

Après cela, en consultant le fichier app.module.ts également présent dans le répertoire app, on constate que le composant est importé à la ligne 5 :



Étape 2 : Création d'actions au chargement de la page avec ngOnInit

'ngOnInit' est une méthode d'interface dans Angular qui est appelé immédiatement après qu'Angular ait initialisé les données liées au composant. C'est un endroit idéal pour effectuer des initialisations supplémentaires telles que le chargement de données à partir d'un service ou l'initialisation des variables.

Vous devez modifier la classe app.component.ts en y ajoutant ce qui suit dans l'exemple ci-dessous :

app.component.ts

src > app > app.component.ts > ...

```ts
import { Component, OnInit } from '@angular/core';// On ajoute l'import de OnInit

@Component({
  selector: 'app-root',
  template: `<h1>Welcome to {{ utilisateurs[0] }}! </h1>`
})// La ligne 5 affiche le premier utilisateur du tableau, il est également possible
  // d'utiliser l'option templateUrl pour choisir l'emplacement du fichier template
export class AppComponent implements OnInit {
  utilisateurs = ['Aymeric', 'Utilisateur1', 'Utilisateur2'];// On créer une liste d'utilisateur

  ngOnInit() {
    console.table(this.utilisateurs);// Lors de l'initialisation du composant, on affiche le tableau d'utilisateurs
  }
}

```

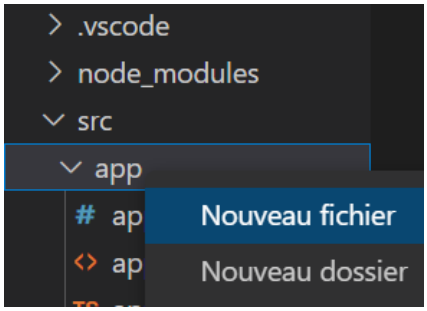
Il existe d'autres cycle de vie liés aux composants tels que :

- ngOnChange
- ngDoCheck
- ngAfterContentInit
- ngAfterContentChecked
- ngAfterViewInit
- ngAfterViewChecked
- ngOnDestroy

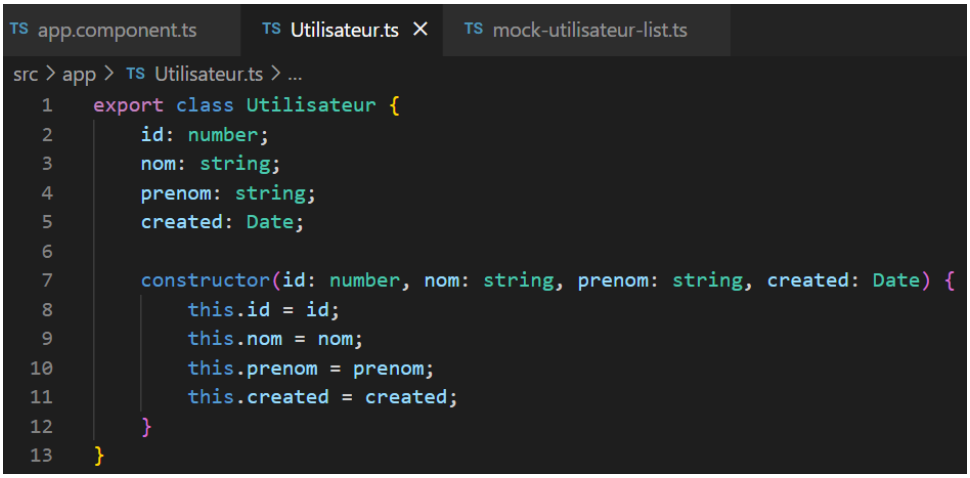
Si vous souhaitez plus d'informations sur les cycle de vie d'un composant Angular, vous pouvez directement aller sur la documentation officielle : https://angular.io/guide/lifecycle-hooks

Étape 3 : Création des fichiers Utilisateur

Vous devez désormais créer le fichier 'Utilisateur.ts' dans le dossier src/app en intégrant le contenu suivant l'exemple ci-dessous :



Pour créer un nouveau fichier



Petite information : un constructeur (constructor en anglais) est un mécanisme fourni par les langages de programmation orienté objet pour initialiser de nouveaux objets. Il joue un rôle crucial dans les patterns de conception orienté objet, permettant de garantir que les objets sont dans un état valide dès leur création.

Ensuite, vous devez créer le fichier 'mock-utilisateur-list.ts' toujours dans le dossier src/app, ce fichier va contenir la liste de tous nos utilisateurs :

app.component.ts | Utilisateur.ts | mock-utilisateur-list.ts

src > app > mock-utilisateur-list.ts > ...

```ts
import { Utilisateur } from "./Utilisateur"; // On importe le modèle utilisé

export const UTILISATEURS: Utilisateur[] = [ // On définit la constante UTILISATEURS qui
                                             //sera égale à la liste d'éléments quelle
                                             //contient (de Type Utilisateur)
    {
        id: 1,
        nom: "Cucherousset",
        prenom : "Aymeric",
        created: new Date()
    },
    {
        id: 2,
        nom: "Utilisateur2",
        prenom: "deTest2",
        created: new Date()
    },
    {
        id: 3,
        nom: "Utilisateur3",
        prenom: "deTest3",
        created: new Date()
    }
]

```

Puis vous allez modifier le fichier 'app.component.ts' (toujours dans le dossier src/app) afin de l'adapter à votre classe Utilisateur ainsi que la liste d'utilisateur :

app.component.ts | Utilisateur.ts | mock-utilisateur-list.ts

src > app > app.component.ts > ...

```ts
import { Component, OnInit } from '@angular/core';
import { UTILISATEURS } from './mock-utilisateur-list'; // Importe tous les utilisateurs
import { Utilisateur } from './Utilisateur'; // Importe la classe Utilisateur

@Component({
  selector: 'app-root',
  template: `<h1>Liste des utilisateurs</h1>`
})
export class AppComponent implements OnInit {
  utilisateurs: Utilisateur[] = UTILISATEURS; // On définit utilisateurs qui est un tableau d'objets
                                              //Utilisateur qui est égale à la constante UTILISATEURS
  ngOnInit() {
    console.table(this.utilisateurs);
    this.selectUtilisateur(this.utilisateurs[0]); // On sélectionne le premier utilisateur de la liste
  }

  selectUtilisateur(utilisateur : Utilisateur){
    alert(`L'utilisateur ${utilisateur.nom} ${utilisateur.prenom} a été sélectionné !`)
  }
}

```

Lors de l'exécution de cette page dans le navigateur, vous recevrez une alerte JS pour vous informer que le premier utilisateur de la liste a été sélectionné, et vous pouvez voir dans la console le tableau de données de tous les utilisateurs.

Il est possible d'ajouter l'URL d'un fichier CSS dans un composant, avec la commande suivante :

styleUrls: ['./component-overview.component.css']

Vous devez la placer après le template dans la section @Component({}).

Il est également possible d'ajouter directement le CSS, pour donner un autre style au texte dans la balise <h1> :

```
@Component({
  selector: 'app-root',
  template: `<h1>Liste des utilisateurs</h1>`,
  styles: ['h1 { font-weight: normal; }']
})
```

Étape Bonus : Créer un composant Angular avec la CLI

Un composant peut être aussi créé à l'aide de la CLI Angular à l'aide de cette commande que vous pouvez exécuter dans votre terminal :

```
PS C:\Users\r_dasilvam\mon_projet_test> ng generate component nom_du_composant
```

```
PS C:\Users\r_dasilvam\mon_projet_test> ng g c nom_du_composant
```
Version de la commande abrégé

Source :

https://angular.io/docs