4. Composants

Introduction

Un composant Angular est un élément permettant d'afficher un élément graphique dans une application :

- vue dans une application,
- élement dans un formulaire,
- composant graphique avancée (chart par exemple),
- etc.

Un composant angular est composé de 3 sous éléments principaux :

- le contrôleur : fichier Typescript (ex : `chart.component.ts`)
- le template : fichier HTML (ex : `chart.component.html`),
- la feuille de style : fichier CSS (ou SCSS) (ex : `chart.component.scss`),

Un composant peut être également complété par un fichier de :

- test unitaire : fichier Typescript (ex : `chart.component.spec.ts`)
- test d'intégration (ex : `chart.component.e2e-spec.ts`)

Décorateur @Component

Le décorateur @Component permet de décrire un composant Angular. Il est défini en entête de classe du composant :

```
@Component({
  selector: "app-home",
  templateUrl: "./home.component.html",
  styleUrls: ["./home.component.css"],
})
export class HomeComponent implements OnInit {}
```

Un composant Angular contient les méta données principales suivantes :

- selector : Sélecteur à utiliser dans les template HTML pour ajouter le composant au DOM : `<app-home></app-home>` ,
- templateUrl : Page HTML servant de template au composant graphique,
- styleUrls : Liste des fichiers CSS associées au composant,

Il existent de nombreuses autres propriétés pour définir le comportement d'un composant.

- animations : Liste d'animations CSS permettant de définir des transitions lors d'un changement d'état du composant,
- changeDetection : Stratégie de détections des changements sur les propriétés d'un contrôleur.
  - ChangeDetectionStrategy.Default : Détection automatiquement (valeur par défaut),
  - ChangeDetectionStrategy.OnPush : Détéction uniquement sur les objets immutables.(changement non détecté sur la modification d'une propriété d'un objet)
- encapsulation : Défini la politique d'utilisation du Shadow Dom dans la portée des CSS.
  - ViewEncapsulation.None - Pas de Shadow DOM, les CSS déclarée ont une portée globale.
  - ViewEncapsulation.Emulated - Pas de Shadow DOM, mais les CSS ont une portée locale.
  - ViewEncapsulation.Native - Shadow DOM, les CSS ont une portée locale.
- entryComponents : Liste de composants insérée dynamiquement dans le DOM au runtime du composant,
- exportAs : Défini le nom qui pourra être utilisé dans la template pour l'assignée à une variable dans le contrôleur,
- host : permet de rattacher des évènements ou des états au composant HTML host du contrôleur,
- inputs : défini les variables qui sont accessibles depuis le contrôleur (c'est variables peuvent porter un nom différent de la variable déclarée dans le contrôleur),
- interpolation : permet d'overrider l'interpolation dans la template du contrôleur (redéfinir `{{` et `}}`)
- moduleId : identifiant du module. Il permet de résoudre les URLs relatives de la template et de la CSS associée.
- outputs : défini les évènements qui sont susceptibles d'être propagés par le contrôleur,
- providers : liste des objets injectés visible uniquement pour ce composant,
- queries : Injection de queries, équivaut aux décorateurs @ViewChild, @ViewChildren, @ContentChild, @ContentChildren
- styles : Définition des styles directement dans le contrôleur,
- template : Définition de la template directement dans le contrôleur,
- viewProviders : liste des objets injectés visible pour ce composant et ces enfants

Documentation officielle

Exercice

Présentation

Application Météo.

L'objectif de cet excercice est de créer les composants d'affichage de la météo.

Etapes de création d'un composant

1. Créer un composant `meteo-view` dans votre dossier d'exercice. Exploiter pour le cela commande de angular-cli.

> ng generate component meteo-view

Référencez ce composant dans la template HTML de l'application principale `app.component.html`, et n'oublier pas de référencer ce nouveau composant dans le module `meteo.module`.

Ce composant va contenir les élements de la vue Météo :

- Moteur de recherche
- Informations sur la ville Recherchée
- La liste de résultats.

1. Créer un composant `meteo-search`. Exploiter pour le cela commande de angular-cli.

> ng generate component meteo-search

Ce composant va permettre de rechercher une ville.

2. Créer un composant `meteo-itemrenderer`. Exploiter pour le cela commande de angular-cli.

> ng generate component meteo-itemrenderer

Ce composant va permettre d'afficher un item de météo.

3. Référencer ces composants dans le module `meteo.module.ts` pour qu'ils soient "compilés" et donc exploitables dans `meteo-view`.
4. Référencer dans la template HTML de `app.component.html` ces 2 composants.
5. Tester l'application, ces composants devraient s'afficher sur la page principale.
6. Désormais compléter le fichier TS et HTML de ces 2 composants. Vous pouvez reprendre en partie ce qui a été fait dans le projet Meteo JS/TS pour compléter la template HTML, et identifier les fonctions disponibles dans le contrôleur. Nous allons les utiliserer plus tard.

Vous avez créé vos premiers composants Angular ! Nous allons les compléter et les exploiter.

Correction disponible ici