Angular : service

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

Service

- classe **TypeScript** décorée par `@Injectable`
- singleton
- pouvant avoir le rôle de
  - l'intermédiaire avec la partie back-end
  - un moyen de communication entre composants
- injectable dans les classes où on a besoin de l'utiliser
- pouvant utiliser un ou plusieurs autres services

Angular

Quelques services **Angular** utilisés dans les chapitres précédents

- `ActivatedRoute`
- `Router`
- `FormBuilder`

Angular

Quelques services **Angular** utilisés dans les chapitres précédents

- `ActivatedRoute`
- `Router`
- `FormBuilder`

Autres services **Angular**

- `HttpClient`
- ...

Angular

Quelques services **Angular** utilisés dans les chapitres précédents

- `ActivatedRoute`
- `Router`
- `FormBuilder`

Autres services **Angular**

- `HttpClient`
- ...

Il est possible de créer nos services personnalisés.

Angular

Pour créer un service

```
ng generate service service-name
```

Angular

Pour créer un service

```
ng generate service service-name
```

Ou aussi

```
ng g s service-name
```

Angular

Pour créer un service

```
ng generate service service-name
```

Ou aussi

```
ng g s service-name
```

Pour créer un service dans un répertoire `services`

```
ng g s services/service-name
```

Angular

Pour créer un service sans générer le fichier de test

```
ng g s services/service-name -skip-tests=true
```

Angular

Pour créer un service sans générer le fichier de test

```
ng g s services/service-name -skip-tests=true
```

Exemple

```
ng g s services/personne -skip-tests
```

Angular

Pour créer un service sans générer le fichier de test

```
ng g s services/service-name -skip-tests=true
```

Exemple

```
ng g s services/personne -skip-tests
```

Constat

```
CREATE src/app/services/personne.service.ts (130 bytes)
```

Angular

Le contenu de `personne.service.ts` jusqu'à la version 5 d'Angular

```
import { Injectable } from '@angular/core';

@Injectable()
export class PersonneService {

  constructor() { }
}
```

Angular

Le contenu de `personne.service.ts` jusqu'à la version 5 d'Angular

```
import { Injectable } from '@angular/core';

@Injectable()
export class PersonneService {

  constructor() { }
}
```

Le contenu de `personne.service.ts` à partir de la version 6 d'Angular

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class PersonneService {

  constructor() { }
}
```

Remarque

- `providedIn: 'root'` : service déclaré dans le module principal (Singleton pour tous les composants qui le chargent depuis ce module)
- Autres valeurs de `providedIn` :
  - `any` :
    - chaque 'lazy-loaded' module a sa propre instance du service.
    - Tous les 'eager-loaded' modules partagent une instance fournie par le module racine.
  - `platform` :
    - Tous les modules utilisent la même instance du service, y compris les 'lazy-loaded'.
    - À la différence de `root`. on aura la même instance même dans le cas où on a deux modules principaux déclaré dans `index.html`.

Angular

Considérons l'interface `Personne` suivante

```
export interface Personne {
    id?: number;
    nom?: string;
    prenom?: string;
}
```

Angular

Mettons à jour le contenu de `personne.service.ts`

```
import { Injectable } from '@angular/core';
import { Personne } from '../interfaces/personne';

@Injectable({
  providedIn: 'root'
})
export class PersonneService {

  personnes: Personne[] = [];

  constructor() { }

  findAll(): Array<Personne> {
    return this.personnes;
  }

  save(p: Personne): void {
    this.personnes.push(p);
  }
}
```

Angular

Mettons à jour le contenu de `personne.service.ts`

```
import { Injectable } from '@angular/core';
import { Personne } from '../interfaces/personne';

@Injectable({
  providedIn: 'root'
})
export class PersonneService {

  personnes: Personne[] = [];

  constructor() { }

  findAll(): Array<Personne> {
    return this.personnes;
  }

  save(p: Personne): void {
    this.personnes.push(p);
  }
}
```

Il faut aussi créer les méthodes qui permettent de modifier et supprimer de personnes.

Angular

Pour commencer

- créer un composant `personne`
- ajouter une route avec le chemin `/personne` pour afficher ce composant

Angular

Pour créer le composant `personne`

```
ng g c components/personne
```

Angular

Pour créer le composant `personne`

```
ng g c components/personne
```

Résultat

```
CREATE src/app/components/personne/personne.spec.ts (565 bytes) \\
CREATE src/app/components/personne/personne.ts (204 bytes) \\
CREATE src/app/components/personne/personne.css (0 bytes) \\
CREATE src/app/components/personne/personne.html (24 bytes)
```

Angular

Contenu de `personne.html`

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

@Component({
  selector: 'app-personne',
  imports: [],
  templateUrl: './personne.html',
  styleUrl: './personne.css'
})
export class PersonneComponent {

  constructor() { }

}
```

Angular

Injectons `PersonneService` dans le constructeur de `PersonneComponent` pour pouvoir l'utiliser

```
import { Component } from '@angular/core';
import { PersonneService } from '../../services/personne.
   service';

@Component({
  selector: 'app-personne',
  templateUrl: './personne.html',
  styleUrls: ['./personne.css']
})
export class PersonneComponent {

  constructor(private personneService: PersonneService) { }

}
```

Angular

Depuis Angular 14, on peut utiliser la fonction `inject`

```
import { Component, inject } from '@angular/core';
import { PersonneService } from '../../services/personne.
   service';

@Component({
  selector: 'app-personne',
  templateUrl: './personne.html',
  styleUrls: ['./personne.css']
})
export class PersonneComponent {

  personneService = inject(PersonneService)


}
```

Angular

Remarque

- Maintenant on peut utiliser les méthodes définies dans le service.
- Il est à rappeler que le service et l'élément de notre application qui va communiquer avec un serveur **JSON** ou un Web-Service (pour persister ou récupérer des données).

Angular

Pour tester, on prépare la liste des personnes dans `personne.service.ts`

```
import { Injectable } from '@angular/core';
import { Personne } from '../interfaces/personne';

@Injectable({
  providedIn: 'root'
})
export class PersonneService {

  personnes: Personne[] = [];

  constructor() {
    this.personnes.push({ nom: 'wick', prenom: 'john' });
    this.personnes.push({ nom: 'green', prenom: 'bob' });
  }
  findAll(): Array<Personne> {
    return this.personnes;
  }
  save(p: Personne): void {
    this.personnes.push(p);
  }
}
```

Angular

Dans `personne.ts`, on appelle la méthode `getAll()` du service

```
import { Component, OnInit } from '@angular/core';
import { PersonneService } from '../../services/personne';
import { Personne } from '../../interfaces/personne';

@Component({
  selector: 'app-personne',
  imports: [],
  templateUrl: './personne.html',
  styleUrl: './personne.css'
})
export class PersonneComponent implements OnInit {

  personnes: Personne[] = [];

  constructor(private personneService: PersonneService) {  }

  ngOnInit() {
    this.personnes = this.personneService.findAll();
  }
}
```

Angular

Enfin, on affiche le résultat dans `personne.html`

```
<h2>Liste des personnes</h2>
<ul>
    @for (elt of personnes; track $index) {
    <li>
        {{ elt.prenom }} {{ elt.nom }}
    </li>
    }
</ul>
```