Angular : pipe

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

Pipe

- a le rôle de formater et modifier l'affichage d'une donnée dans le `component.html`
- est une classe décorée par le décorateur `@pipe`
- doit implémenter la méthode `transform(value: any, args?: any)` de l'interface `PipeTransform`
- peut prendre un ou plusieurs paramètres

Angular

Pipe

- a le rôle de formater et modifier l'affichage d'une donnée dans le `component.html`
- est une classe décorée par le décorateur `@pipe`
- doit implémenter la méthode `transform(value: any, args?: any)` de l'interface `PipeTransform`
- peut prendre un ou plusieurs paramètres

Les pipes prédéfinis sont à importer de `CommonModule`

Angular

Exemple

```
<p>{{ "bonjour" | uppercase }}</p>
<!-- BONJOUR -->
```

Angular

Exemple

```
<p>{{ "bonjour" | uppercase }}</p>
<!-- BONJOUR -->
```

Certains pipes peuvent prendre des paramètres (déclarer `maDate = Date.now()` dans `component.ts`)

```
<p>{{ maDate | date:'d MMM y' }}</p>
<!-- affiche 19 oct 2018 -->
```

Angular

Exemple

```
<p>{{ "bonjour" | uppercase }}</p>
<!-- BONJOUR -->
```

Certains pipes peuvent prendre des paramètres (déclarer `maDate = Date.now()` dans `component.ts`)

```
<p>{{ maDate | date:'d MMM y' }}</p>
<!-- affiche 19 oct 2018 -->
```

Il est possible d'enchaîner les pipes

```
<p>{{ maDate | date:'d MMM y' | uppercase }}</p>
<!-- affiche 19 OCT 2018 -->
```

Angular

Considérons l'objet suivant défini dans un `.component.ts`

```
stagiaire: Stagiaire = new Stagiaire(100, 'Wick', 'John');
```

Angular

Considérons l'objet suivant défini dans un `.component.ts`

```
stagiaire: Stagiaire = new Stagiaire(100, 'Wick', 'John');
```

Avec `*ngFor`**, on ne peut itérer sur des objets**

```
<ul>
    <li *ngFor="let elt of stagiaire">
    </li>
</ul>
```

Angular

Depuis Angular 6, un pipe `keyvalue` a été introduit pour itérer sur les objets

```
<ul>
    <li *ngFor="let elt of stagiaire | keyvalue">
        {{ elt.key }} : {{ elt.value }}
    </li>
</ul>
```

Angular

Depuis Angular 6, un pipe `keyvalue` a été introduit pour itérer sur les objets

```
<ul>
    <li *ngFor="let elt of stagiaire | keyvalue">
        {{ elt.key }} : {{ elt.value }}
    </li>
</ul>
```

Ou

```
<ul>
  @for (elt of stagiaire | keyvalue; track $index) {
  <li>
    {{ elt.key }} : {{ elt.value }}
  </li>
  }
</ul>
```

Angular

Remarque

Angular propose dans `CommonModule` 13 pipes.

Angular

Pipes prédéfinis pour objets

- `json`
- `keyvalue`

Angular

Pipes prédéfinis pour objets

- `json`
- `keyvalue`

Pipes prédéfinis pour chaînes de caractères

- `titlecase`
- `uppercase`
- `lowercase`
- `slice` (aussi pour les tableaux)

Angular

Pipes qui s'adaptent avec la valeur `locale` (i18n)

- `currency`
- `percent`
- `number`
- `date`
- ...

Angular

Les pipes

- pipes prédéfinis
- pipes personnalisés

Angular

Pour créer un pipe

```
ng generate pipe pipe-name
```

Angular

Pour créer un pipe

```
ng generate pipe pipe-name
```

Ou le raccourci

```
ng g p pipe-name
```

Angular

Pour créer un pipe

```
ng generate pipe pipe-name
```

Ou le raccourci

```
ng g p pipe-name
```

Pour créer un pipe sans le fichier de test

```
ng g p pipe-name -skip-tests=true
```

Angular

Exemple : pour créer un nouveau pipe (`get-char`) qui retourne la première lettre d'une chaîne de caractères

```
ng g p get-char -skip-tests=true
```

Angular

Exemple : pour créer un nouveau pipe (`get-char`) qui retourne la première lettre d'une chaîne de caractères

```
ng g p get-char -skip-tests=true
```

Pour une meilleure structuration, plaçons le pipe dans un répertoire `pipes`

```
ng g p pipes/get-char -skip-tests=true
```

Angular

Constat

Un fichier créé : `get-char.pipe.ts`

Angular

Contenu du `get-char.pipe.ts`

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'getChar'
})
export class GetCharPipe implements PipeTransform {

  transform(value: unknown, ...args: unknown[]): unknown {
    return null;
  }

}
```

Angular

Contenu du `get-char.pipe.ts`

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'getChar'
})
export class GetCharPipe implements PipeTransform {

  transform(value: unknown, ...args: unknown[]): unknown {
    return null;
  }

}
```

Explication

`value` est la valeur sur laquelle le pipe va s'appliquer.

Angular

Renommons le pipe en lowercase

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'getchar'
})
export class GetCharPipe implements PipeTransform {

  transform(value: unknown, ...args: unknown[]): unknown {
    return null;
  }

}
```

Angular

Modifions `get-char.pipe.ts` pour retourner la première lettre

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'getchar'
})
export class GetCharPipe implements PipeTransform {

  transform(value: string, ...args: unknown[]): string {
    return value[0];
  }

}
```

Angular

Testons ce nouveau pipe dans `app.html`

```
<p>{{ "bonjour" | getchar }}</p>
<!-- affiche b -->

<p>{{ "wick" | getchar }}</p>
<!-- affiche w -->

<p>{{ "john" | getchar }}</p>
<!-- affiche j -->
```

Angular

Testons ce nouveau pipe dans `app.html`

```
<p>{{ "bonjour" | getchar }}</p>
<!-- affiche b -->

<p>{{ "wick" | getchar }}</p>
<!-- affiche w -->

<p>{{ "john" | getchar }}</p>
<!-- affiche j -->
```


Sans oublier de déclarer le pipe dans `app.ts`

```
imports: [GetCharPipe]
```

Angular

Modifions `get-char.pipe.ts` pour retourner un caractère à une position donnée (`pos`)

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'getchar'
})
export class GetCharPipe implements PipeTransform {

  transform(value: string, ...args: number[]): string {
    if (args[0] && args[0] < value.length) {
      return value[args[0]];
    }
    return value[0];
  }

}
```

Angular

Pour tester

```
<p>{{ "bonjour" | getchar:2 }}</p>
<!-- affiche n -->

<p>{{ "wick" | getchar:6 }}</p>
<!-- affiche w -->

<p>{{ "john" | getchar }}</p>
<!-- affiche j -->
```

Angular

Exercice 1

Créer un pipe `even-value` qui permet de retourner les valeurs paires d'un tableau, dans tableau.

Angular

Exercice 1

Créer un pipe `even-value` qui permet de retourner les valeurs paires d'un tableau, dans tableau.

Exemple

```
<ul>
  @for (elt of tab | evenvalue; track $index) {
  <li>
    {{ elt }}
  </li>
  }
</ul>
<!-- affiche 2 et 8
```

Angular

Exercice 2

Créer un pipe `str-to-array` qui permet de transformer une chaîne de caractère en tableau de caractères. Le pipe peut accepter 0, 1 ou deux paramètres.

- si aucun paramètre n'a été fourni, alors le pipe retourne un tableau contenant tous les caractères de la chaîne.
- si un seul paramètre a été passé, alors le pipe retourne les caractères de la sous-chaîne qui commence de la position indiquée par ce paramètre sous forme d'un tableau de caractère.
- si deux paramètres sont présents, alors le pipe retourne les caractères de la sous-chaîne située, entre les deux paramètres, sous forme d'un tableau de caractère.

Vous pouvez utilisez `Array.from(str)` ou `str.split("")` pour convertir la chaîne de caractères `str` en tableau de caractères.