Angular : cycle de vie d'un composant

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

Rappel

Tous les composants sauf, `AppComponent`, ont deux méthodes générées : `constructor` et `ngOnInit()`.

Angular

Question 1

Les deux méthodes sont-elles différentes ? ou font-elles la même chose ?

Angular

Question 1

Les deux méthodes sont-elles différentes ? ou font-elles la même chose ?

`constructor`

- Fonction **ES6**,
- Permettant d'initialiser les attributs d'une classe,
- Utilisée par **Angular** pour l'injection de dépendance,
- Exécutée en premier pour permettre d'instancier les classes injectées (qui pourraient être utilisées par `ngOnInit()`).

`ngOnInit()`

- Fonction **Angular**,
- Définie dans l'interface `OnInit`,
- Appartenant au cycle de vie d'un composant **Angular**,
- Exécutée après le constructeur et après chargement de `Input`...

Angular

Question 2

Pourquoi `ngOnInit()` n'a pas été généré dans le code par défaut de `AppComponent` ?

Angular

Question 2

Pourquoi `ngOnInit()` n'a pas été généré dans le code par défaut de `AppComponent` ?

Réponse

- Le composant principal ne peut être l'enfant d'un autre composant.
- La route du composant principal ne prend souvent pas de paramètre.

Angular

Question 3

Peut-on ajouter `ngOnInit()` dans `AppComponent` ?

Angular

Question 3

Peut-on ajouter `ngOnInit()` dans `AppComponent` ?

Réponse

Oui.

Angular

Lifecycle hooks

- Tout composant `Angular` a un cycle de vie qui commence à l'instanciation.
- `Angular` nous a préparé une méthode pour chaque phase du cycle de vie : **Lifecycle hooks**.

Différentes méthodes (hooks) de cycle de vie d'un composant **Angular** (dans l'ordre)

- `constructor` (~~hook~~) : appelé pour préparer les éléments injectés et les attributs à initialiser.
- `ngOnChanges` : appelée lorsqu'un `@Input` est défini ou modifié de l'extérieur.
- `ngOnInit` : appelée une seule fois et permet d'initialiser le composant.
- `ngDoCheck` : appelée après chaque détection de changements.
- `ngAfterContentInit` : appelée une seule fois après initialisation du composant avec injection du contenu externe (transclusion).
- `ngAfterContentChecked` : appelé chaque fois qu'**Angular** a fini d'exécuter la détection de changement sur un composant et ses enfants
- `ngAfterViewInit` : appelée une seule fois après initialisation de la vue du composant ainsi que celle de ses enfants (après `ngAfterContentInit`).
- `ngAfterViewChecked` : appelée chaque fois qu'**Angular** a fini d'exécuter le change detection sur un composant et ses enfants.
- `ngOnDestroy` : appelée juste avant la destruction du composant par **Angular**.

Angular

Créons un composant `cycle-vie` pour explorer ces différentes méthodes

```
ng g c modules/cours/cycle-vie
```

Angular

Créons un composant `cycle-vie` pour explorer ces différentes méthodes

```
ng g c modules/cours/cycle-vie
```


Remarque

N'oublions pas d'associer une route à ce nouveau composant et de l'intégrer dans le menu.

Ajoutons toutes les méthodes `hooks` **dans** `cycle-vie.component.ts`

```
export class CycleVieComponent implements OnInit, OnChanges, OnDestroy, DoCheck, AfterViewInit,
    AfterViewChecked, AfterContentInit, AfterContentChecked {
  constructor() {
    console.log('constructor');
  }
  ngOnChanges(changes: SimpleChanges): void {
    console.log('ngOnChanges');
  }
  ngAfterViewChecked(): void {
    console.log('ngAfterViewChecked');
  }
  ngAfterContentChecked(): void {
    console.log('ngAfterContentChecked');
  }
  ngAfterContentInit(): void {
    console.log("ngAfterContentInit");
  }
  ngAfterViewInit(): void {
    console.log("ngAfterViewInit");
  }
  ngOnInit(): void {
    console.log("ngOnInit");
  }
  ngOnDestroy(): void {
    console.log("ngOnDestroy");
  }
  ngDoCheck(): void {
    console.log("ngDoCheck");
  }
}
```

Angular

Allez à la route associée au composant `CycleVieComponent` et vérifier l'affichage des messages suivants

```
constructor
ngOnInit
ngDoCheck
ngAfterContentInit
ngAfterContentChecked
ngAfterViewInit
ngAfterViewChecked
```

Angular

Allez à la route associée au composant `CycleVieComponent` et vérifier l'affichage des messages suivants

```
constructor
ngOnInit
ngDoCheck
ngAfterContentInit
ngAfterContentChecked
ngAfterViewInit
ngAfterViewChecked
```


Changez de composant et vérifiez l'affichage du message suivant

```
ngOnDestroy
```

Angular

Rappel

Le message `ngOnChanges` ne s'affiche pas car `CycleVieComponent` ne reçoit pas de valeur depuis un composant parent avec le décorateur `@Input()`.

Angular

Modifions le constructeur et la méthode `ngOnDestroy` pour vérifier les méthodes qui s'exécutent après chaque changement

```
export class CycleVieComponent implements OnInit, OnChanges, OnDestroy,
    DoCheck, AfterViewInit, AfterViewChecked, AfterContentInit,
   AfterContentChecked {

  value = 0;
  interval: any;

  constructor() {
    console.log('constructor');
    this.interval = setInterval(() => console.log(this.value++), 2000)
  }

  ngOnDestroy(): void {
    clearInterval(this.interval);
    console.log("ngOnDestroy");
  }

  // contenu précédent
}
```

Angular

Allez à la route associée au composant `CycleVieComponent` et vérifier l'affichage répétitif des messages suivants

```
// la valeur de value
ngDoCheck
ngAfterContentChecked
ngAfterViewChecked
```

Angular

Allez à la route associée au composant `CycleVieComponent` et vérifier l'affichage répétitif des messages suivants

```
// la valeur de value
ngDoCheck
ngAfterContentChecked
ngAfterViewChecked
```


Changez de composant, vérifiez que le seul message affiché et le suivant

```
ngOnDestroy
```