Angular : directive

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

Deux types de directives **Angular**

- Directives structurelles : ayant comme but de modifier le **DOM** (`ngIf`, `ngFor`...)
- Directives attributes : ayant comme but de modifier l'apparence ou le comportement d'un élément (`ngStyle`, `ngClass`...)

Angular

Deux types de directives **Angular**

- Directives structurelles : ayant comme but de modifier le **DOM** (`ngIf`, `ngFor`...)
- Directives attributes : ayant comme but de modifier l'apparence ou le comportement d'un élément (`ngStyle`, `ngClass`...)

Remarque

Il est possible de créer sa propre directive attribute.

Angular

Démarque

- Créer une classe **TypeScript**
- Utiliser le décorateur `@Directive` qui a une propriété `selector`
- La valeur de la propriété `selector` peut-être utilisée comme attribut de balise dans les templates

Angular

Pour créer une directive

```
ng generate directive directive-name
```

Angular

Pour créer une directive

```
ng generate directive directive-name
```

Ou le raccourci

```
ng g d nom-directive
```

Angular

Pour créer une directive

```
ng generate directive directive-name
```

Ou le raccourci

```
ng g d nom-directive
```

Et pour ne pas générer le fichier de test

```
ng g d nom-directive --skip-tests=true
```

Angular

Exemple : définissons une directive (`mouvement`) qui modifie l'apparence d'un élément lorsqu'il est survolé par la souris

```
ng g d directives/mouvement --skip-tests=true
```

Angular

Exemple : définissons une directive (`mouvement`) qui modifie l'apparence d'un élément lorsqu'il est survolé par la souris

```
ng g d directives/mouvement --skip-tests=true
```

Résultat

```
CREATE src/app/directives/mouvement.directive.ts (147 bytes)
UPDATE src/app/app.module.ts (1035 bytes)
```

Angular

Exemple : définissons une directive (`mouvement`) qui modifie l'apparence d'un élément lorsqu'il est survolé par la souris

```
ng g d directives/mouvement --skip-tests=true
```

Résultat

```
CREATE src/app/directives/mouvement.directive.ts (147 bytes)
UPDATE src/app/app.module.ts (1035 bytes)
```


Explication

- Un fichier généré : `mouvement.directive.ts`
- Une mise à jour effectuée : déclaration de la directive dans `app.module.ts`

Angular

Contenu du `mouvement.directive.ts`

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

@Directive({
  selector: '[appMouvement]'
})
export class MouvementDirective {

  constructor() { }

}
```

Angular

Pour référencer les éléments concernés par la directive, on fait une injection de dépendance de `ElementRef`

```
import { Directive, ElementRef } from '@angular/core
    ';

@Directive({
  selector: '[appMouvement]'
})
export class MouvementDirective {

  constructor(private el: ElementRef) {
  }

}
```

Angular

Pour changer la couleur de fond d'un élément utilisant cette directive

```
import { Directive, ElementRef } from '@angular/core';

@Directive({
  selector: '[appMouvement]'
})
export class MouvementDirective {

  constructor(private el: ElementRef) {
    el.nativeElement.style.backgroundColor = 'red';
  }

}
```

Angular

Pour changer la couleur de fond d'un élément utilisant cette directive

```
import { Directive, ElementRef } from '@angular/core';

@Directive({
  selector: '[appMouvement]'
})
export class MouvementDirective {

  constructor(private el: ElementRef) {
    el.nativeElement.style.backgroundColor = 'red';
  }

}
```

Pour tester, ajouter le code suivant dans `app.component.html` **(par exemple)**

```
<p appMouvement> mon texte </p>
```

Angular

Hypothèse

- Si on voulait modifier la couleur de fond d'un élément s'il est survolé
- Et remettre sa couleur à la sortie du curseur
- Il faut attacher des évènements aux changements de couleur
- On peut utiliser un décorateur **Angular** : `@HostListener` qui prend comme paramètre le nom d'un évènement

Solution

```
import { Directive, ElementRef, HostListener } from '@angular/
   core';

@Directive({
  selector: '[appMouvement]'
})
export class MouvementDirective {

  constructor(private el: ElementRef) {  }

  @HostListener('mouseenter') onMouseEnter(): void {
    this.changerCouleur('skyblue');
  }
  @HostListener('mouseleave') onMouseLeave(): void {
    this.changerCouleur('white');
  }
  changerCouleur(couleur: string): void {
    this.el.nativeElement.style.backgroundColor = couleur;
  }
}
```

Angular

Hypothèse

- Si on voulait que la couleur affichée soit passée comme valeur de notre attribut `appMouvement`
- On peut utiliser `@Input`

Angular

Hypothèse

- Si on voulait que la couleur affichée soit passée comme valeur de notre attribut `appMouvement`
- On peut utiliser `@Input`

Par exemple

```
<p [appMouvement]="'red'"> mon texte </p>
```

Angular

Hypothèse

- Si on voulait que la couleur affichée soit passée comme valeur de notre attribut `appMouvement`
- On peut utiliser `@Input`

Par exemple

```
<p [appMouvement]="'red'"> mon texte </p>
```

Dans ce cas, `maCouleur` **est un attribut défini dans la classe associée**

```
<p [appMouvement]="maCouleur"> mon texte </p>
```

Angular

Solution

```
import { Directive, ElementRef, HostListener, Input } from '@angular/core';

@Directive({
  selector: '[appMouvement]'
})
export class MouvementDirective {
  @Input('appMouvement') couleur = '';
  constructor(private el: ElementRef) { }

  @HostListener('mouseenter') onMouseEnter(): void {
    this.changerCouleur(this.couleur);
  }
  @HostListener('mouseleave') onMouseLeave(): void {
    this.changerCouleur('white');
  }
  changerCouleur(couleur: string): void {
    this.el.nativeElement.style.backgroundColor = couleur;
  }
}
```

Angular

On peut utiliser le décorateur `@HostBinding` et lui passer le nom d'un attribut HTML de l'élément ayant l'attribut que l'on souhaite modifier

```
import { Directive, HostListener, Input, HostBinding } from '@angular/core';

@Directive({
  selector: '[appMouvement]'
})
export class MouvementDirective {
  @HostBinding('style.backgroundColor') background = '';
  @Input('appMouvement') couleur = '';

  constructor() { }

  @HostListener('mouseenter') onMouseEnter(): void {
    this.changerCouleur(this.couleur);
  }
  @HostListener('mouseleave') onMouseLeave(): void {
    this.changerCouleur('white');
  }
  changerCouleur(color: string): void {
    this.background = color;
  }
}
```

Angular

Question

Pourquoi ne pas tout faire dans les fichiers **CSS** ?

Angular

Question

Pourquoi ne pas tout faire dans les fichiers **CSS** ?

Réponse

- Éviter d'écrire des sélecteurs **CSS** compliqués
- Réutilisation dans plusieurs composants

Angular

Remarque

La directive `appMouvement` est uniquement utilisable dans le module où elle a été déclarée.

Angular

Remarque

La directive `appMouvement` est uniquement utilisable dans le module où elle a été déclarée.

Question

Comment l'utiliser dans plusieurs modules ?

Angular

Remarque

La directive `appMouvement` est uniquement utilisable dans le module où elle a été déclarée.

Question

Comment l'utiliser dans plusieurs modules ?

Solution

- Utiliser le module `shared`.
- Dans `shared` on déclare et exporte tous les éléments à partager entre plusieurs modules.
- Importer le module `shared` dans les modules où on a besoin d'utiliser la directive.

Angular

Contenu précédent de `shared.module.ts`

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { GetCharPipe } from 'src/app/pipes/get-char.pipe';

@NgModule({
  declarations: [
    GetCharPipe
  ],
  imports: [
    CommonModule,
  ],
  exports: [
    GetCharPipe
  ]
})
export class SharedModule { }
```

Angular

Déclarons et exportons la directive `appMouvement`

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { GetCharPipe } from 'src/app/pipes/get-char.pipe';
import { MouvementDirective } from 'src/app/directives/mouvement.
   directive';

@NgModule({
  declarations: [
    GetCharPipe,
    MouvementDirective,
  ],
  imports: [
    CommonModule,
  ],
  exports: [
    GetCharPipe,
    MouvementDirective,
  ]
})
export class SharedModule { }
```

Angular

N'oublions pas de supprimer la déclaration de `appMouvement` dans `app.module.ts`

Angular

N'oublions pas de supprimer la déclaration de `appMouvement` dans `app.module.ts`

Remarque

Pour utiliser la directive `appMouvement` dans un module `x`, il suffit d'importer `SharedModule` dans la section `imports` de `x`.

Angular

Exercice

- Créez un composant `peintre`
- Dans `peintre.component.ts`, déclarez un attribut `couleurs` avec les valeurs données ci-dessous.
- Créez un composant fils nommé `peinture`.
- Le composant `peinture` a un attribut `value` recevant sa valeur du tableau `couleurs` défini dans `peintre.component.ts`.
- Le nombre de composants `peinture` = taille du tableau `couleurs`.
- Chaque composant `peinture` affiche le nom de la couleur qu'il a reçu de son parent.
- Créez une directive `pinceau` qui modifie la couleur de fond des composants `peinture` survolés. La couleur de fond est la valeur de l'attribut `value` du composant `peinture` survolé.

```
couleurs = ['red', 'skyblue', 'gray', 'green', 'yellow', 'teal']
```