Par NIDHAL JELASSI
nidhal.jelassi@fsegt.utm.tn

PROGRAMMATION WEB AVANCÉE

ANGULAR

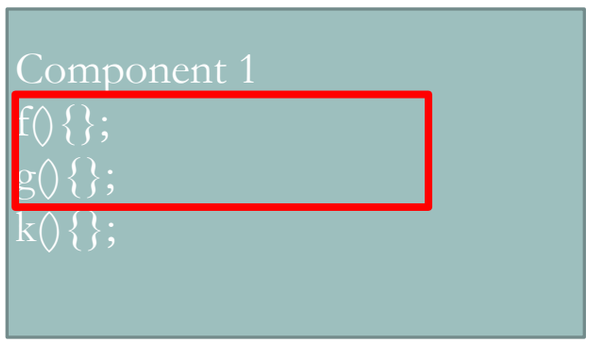
Chapitre 5 : Les Services et les injections de dépendances

OBJECTIFS

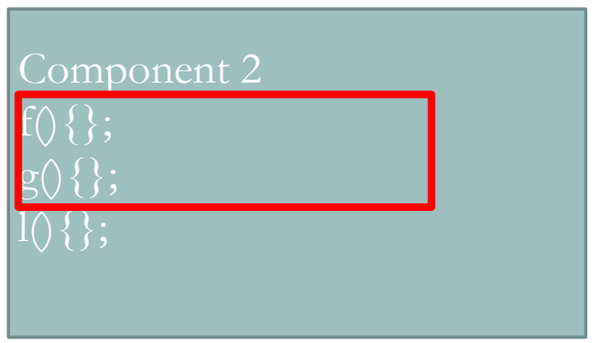
- Définir un service
- Définir ce qu'est l'injection de dépendance
- Injecter un service
- Définir la portée d'un service
- Réordonner son code en utilisant les services

INTÉRÊT DES SERVICES

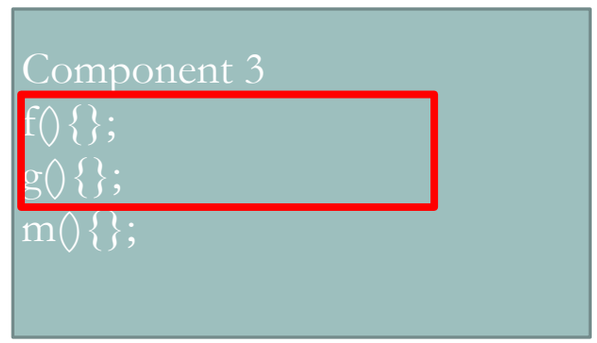
- Un service est une classe qui permet d'exécuter un traitement.
- Permet d'encapsuler des fonctionnalités redondantes permettant ainsi d'éviter la redondance de code.



Redondance de code

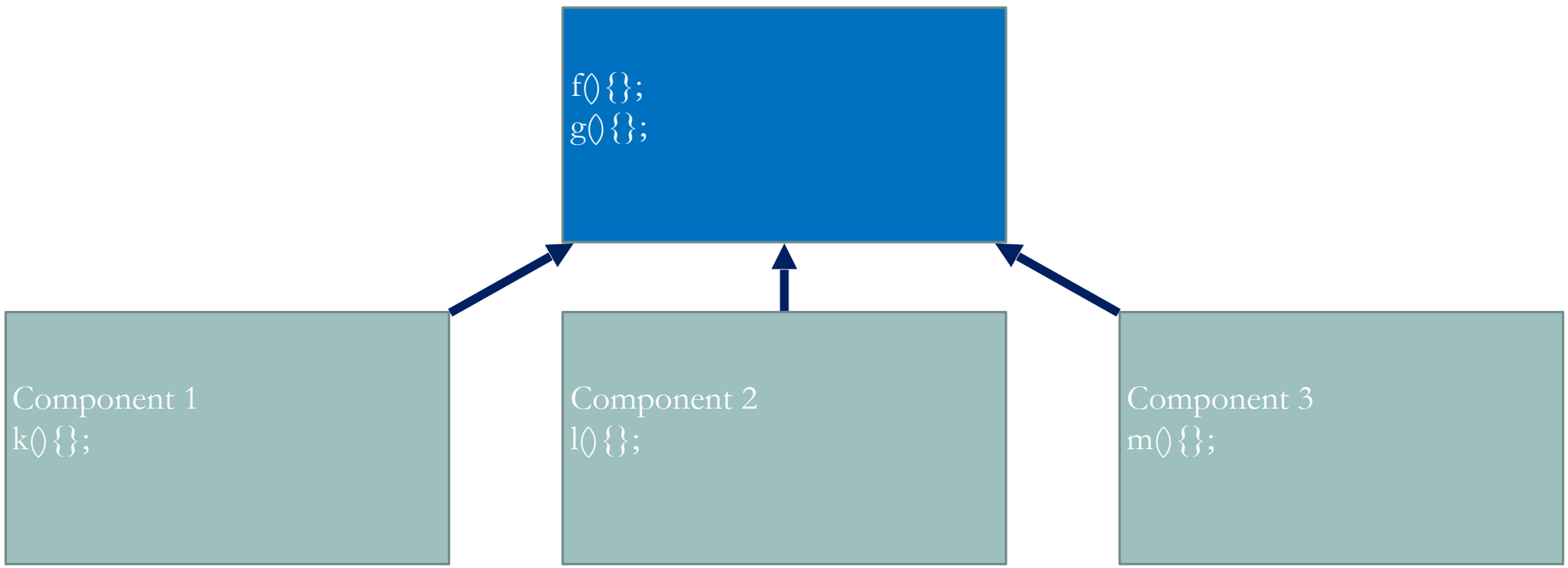


Maintenabilité difficile

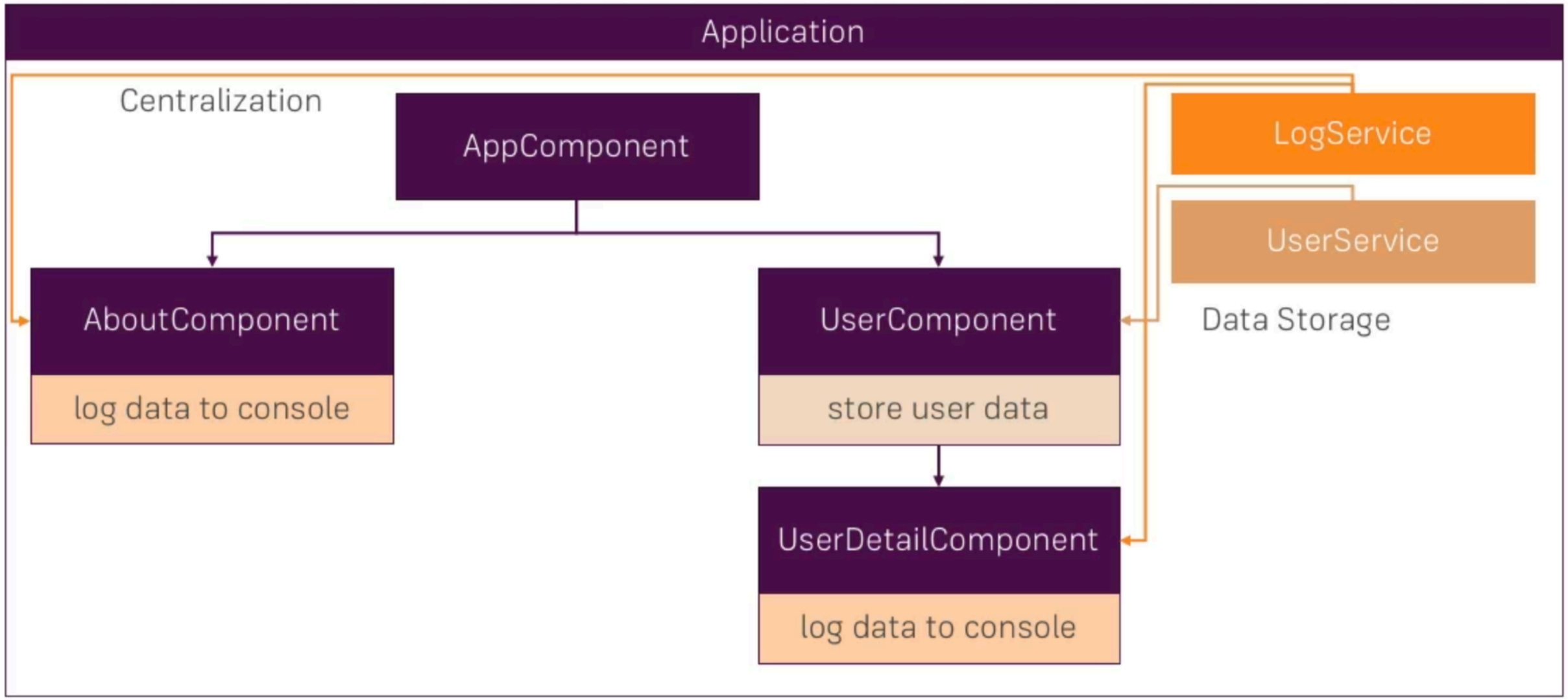


Indisponibilité

INTÉRÊT DES SERVICES



EXEMPLE


Application
Centralization
AppComponent
AboutComponent
log data to console
UserComponent
store user data
UserDetailComponent
log data to console
LogService
UserService
Data Storage

QU'EST CE QU'UN SERVICE ?

- Un service est une sorte d'intermédiaire entre la vue et la logique
- Ce qui n'est pas trivial doit être écrit sous forme d'un composant
- Un service est associé à un composant en utilisant l'injection de dépendance (DI).
- Un service peut donc :
  - Interagir avec les données (fournit, supprime et modifie)
  - Interagir entre classes et composants
  - Effectuer tout traitement métier (calcul, tri, extraction …)

CRÉATION D'UN SERVICE

- **Manuellement**
- A noter qu'un Service est comme un composant sauf qu'il n'a pas de template.
- **Via CLI**

  - **ng generate service nomDuService**
  - **ng g s nomDuService**

EXEMPLE DE CRÉATION D'UN SERVICE

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

@Injectable()
export class FirstService {

  constructor() { }

}
```

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

@Injectable(
providerIn : 'root'
)
export class FirstService {

  constructor() { }

}
```

```
//…Other import
import {FirstService} from "./
first.service";
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule
  ],
  providers: [FirstService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

L'INJECTION DE DÉPENDANCE

- L'injection de dépendance est un patron de conception (Pattern Design).

```
Classe A1{
 ClasseB b;
 ClasseC c;
 …
}
```

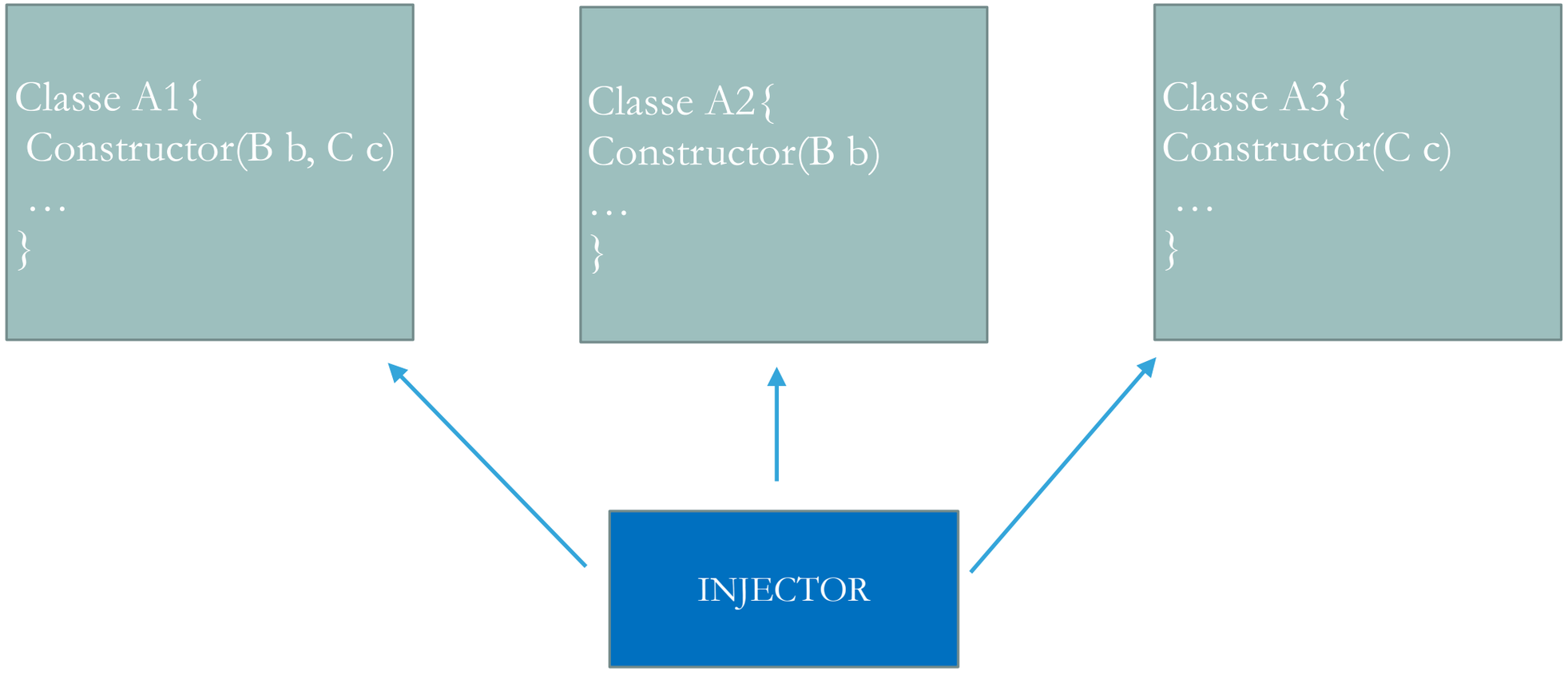
```
Classe A2{
 ClasseB b;
…
}
```

```
Classe A3{
ClasseC c;
 …
}
```

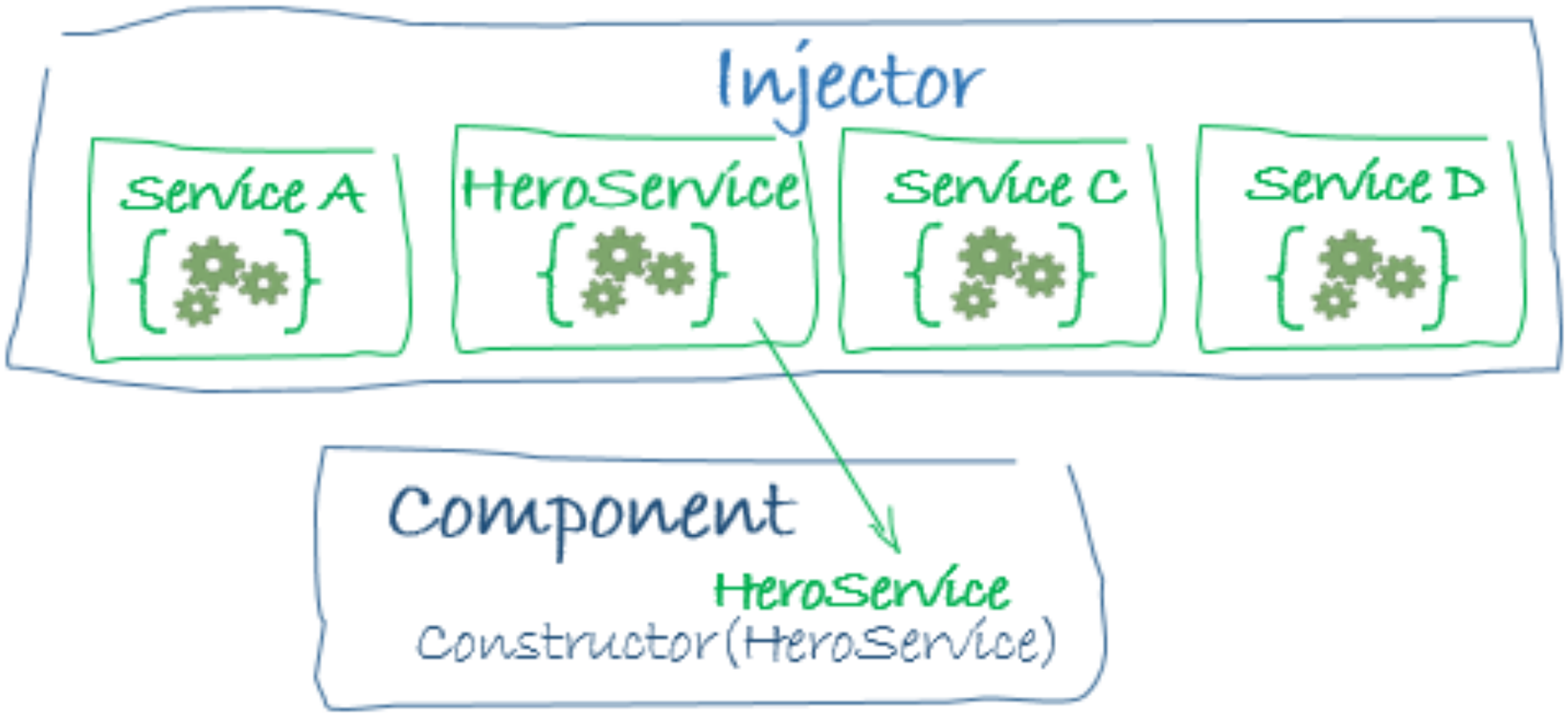
- *Que se passera t-il si on change quelque chose dans le constructeur de B ou C ?*
- *Qui va modifier l'instanciation de ces classes dans les différentes classes qui en dépendent?*

SOLUTION

- Déléguer cette tache à une entité tierce.
- Il s'agit de l'**INJECTOR**



INJECTOR



- Comment les injecter ?
- Comment spécifier à l'injecteur quel service injecter et où l'injecter ?

L'INJECTION DE DÉPENDANCE

- Deux grandes questions se posent alors :

- Comment injecter ces services ?

- Comment spécifier à l'injecteur de dépendances quel service injecter et où sera-t-il visible ?

L'INJECTION DE DÉPENDANCE

- L'injection de dépendance utilise les étapes suivantes :

1. Déclarer le service dans le provider du module ou du composant
2. Passer le service comme paramètre du constructeur de l'entité qui en a besoin.

SYNTAXE

Marque une classe comme étant disponible pour l'Injector, au moment de sa création.

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

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

@Component, @Pipe, et @Directive sont des sous classes de @Injectable(), ceci explique le fait qu'on peut y injecter directement des dépendances.


```
providedIn: Type<any> | 'root' | null
```

Détermine quels injecteurs fourniront l'injectable, en l'associant à un @NgModule ou à un autre InjectorType, ou en spécifiant que cet injectable doit être fourni dans l'injecteur 'root', qui sera l'injecteur au niveau de l'application dans la plupart des applications.

EXEMPLE

```
import { Component, OnInit } from '@angular/core';
import {Cv} from './cv';
import {CvService} from "../cv.service";
@Component({
  selector: 'app-cv',
  templateUrl: './cv.component.html',
  styleUrls: ['./cv.component.css'],
  providers:[CvService] // on peut aussi l'importer ici
})
export class CvComponent implements OnInit {
  selectedCv : Cv;
  constructor(private monPremierService:CvService) { }
  ngOnInit() {
  }
}
```

CHARGEMENT AUTOMATIQUE DU SERVICE

- A partir de Angular 6 vous pouvez ne plus utiliser le provider du module afin de charger votre service mais le faire directement au niveau du service à travers l'annotation **@Injectable** et sa propriété **providedIn**. Vous pouvez charger le service dans toute l'application via le mot clé root.

- Si vous voulez charger le service dans un module particulier vous l'importer et vous le mettez à la place de 'root'.

AVANTAGES

- On peut citer deux avantages au providedIn :

- Lazy loading : Ne charger le code des services qu'à la première injection

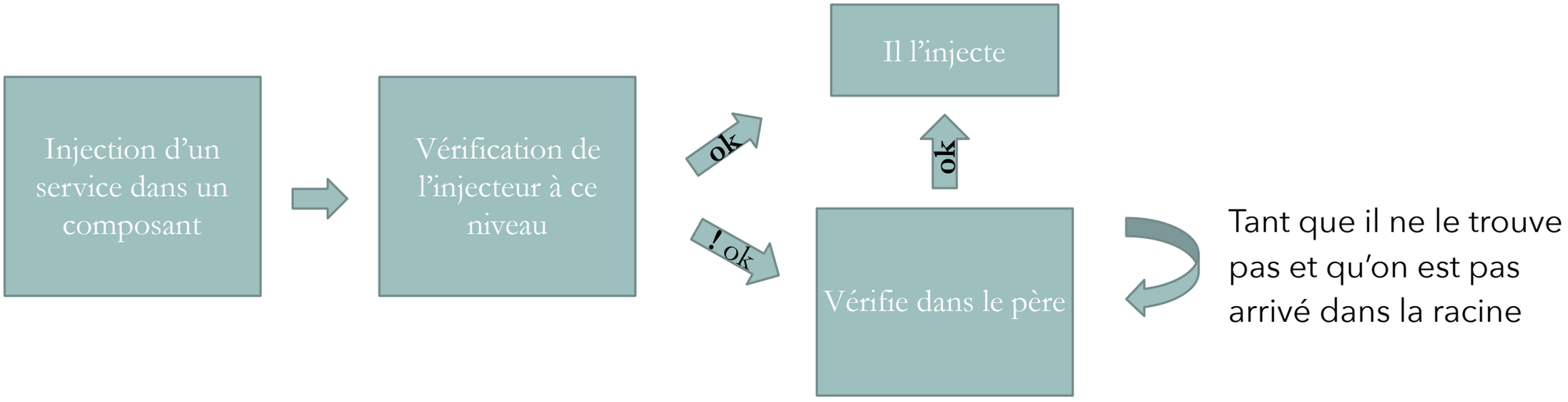
- Permettre le Tree-Shaking des services non utilisés : Si le service n'est jamais utilisé, son code ne sera entièrement retiré du build final.

@INJECTABLE

- C'est un décorateur permettant de rendre une classe injectable
- Une classe est dite injectable si on peut y injecter des dépendances
- @Component, @Pipe, et @Directive sont des sous classes de @Injectable(), ceci explique le fait qu'on peut y injecter directement des dépendances.
- Si vous n'allez injecter aucun service dans votre service, cette annotation n'est plus nécessaire.
- Remarque : Angular conseille de toujours mettre cette annotation.

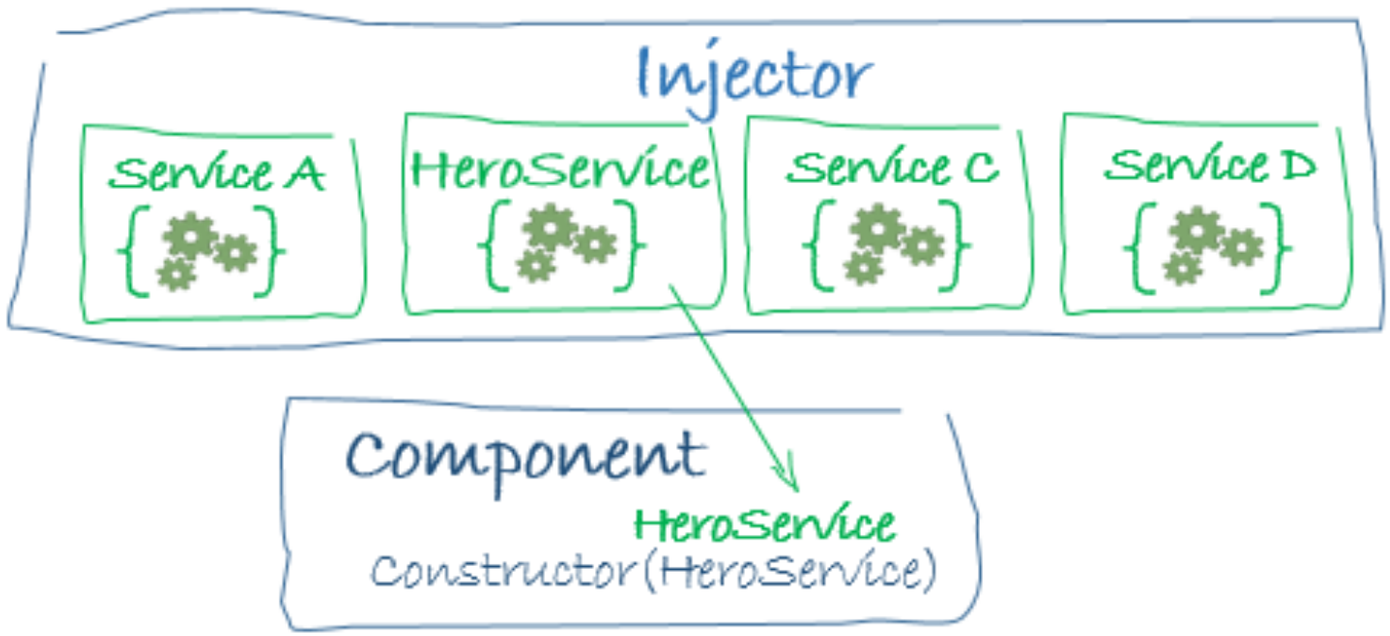
DI HIÉRARCHIQUE

- Le système d'injection de dépendance d'Angular est hiérarchique.
- Un arbre d'injecteur est crée. Cet arbre est parallèle à l'arbre de composant.
- L'algorithme suivant permet la détection de l'injecteur adéquat selon le diagramme suivant :



L'INJECTION DE DÉPENDANCE

- Si un service est déclaré au niveau du Module et qu'il est déclaré dans le provider d'un composant c'est la déclaration la plus spécifique qui l'emporte.



- ***L'injection d'un service dans un autre est la même que pour un composant.***

EXERCICE

- Créons un service de Todo, le Model et le composant qui va avec. Un Todo est caractérisé par un nom et un contenu.
- Ce service permettra de faire les fonctionnalités suivantes :
  - Logger un Todo
  - Ajouter un Todo
  - Afficher la liste des Todos
  - Supprimer un Todo

EXERCICE

- Reprenez le service Logger en y ajoutant un getter qui retourne le tableau de logs
- Créer deux composants compo1 et compo2
- Injecter y le service logger
- Dans les deux composants créer un input texte et un bouton. Au click sur le bouton il va logger le contenu de l'input.
- Déclarer le directement dans le provider des deux composants.
- Vérifier est ce qu'il travaille sur la même instance du service ou non.

EXERCICE

- Ajouter un troisième composant compo3.
- Provider le service dans le composant 3
- Reprenez les mêmes fonctionnalités du composant 1 dans le composant 3
- Testez.
- Que remarquez vous à présent ?

INJECTER UN SERVICE DANS UN SERVICE

- L'injection d'un service dans un autre est la même que pour un composant.

- Les seuls différences sont :
  - Le service à injecter doit être visible pour le service cible.
  - Le service cible doit obligatoirement avoir la décoration @Injectable.

RETOUR AU PROJET CV…

- Ajouter aussi un composant pour afficher la liste des cvs recrutés ainsi qu'un service **RecruteService** qui gérer les personnes recrutés.
- Au click sur le bouton **Recruter** d'un Cv, la personne est ajouté à la liste des personnes embauchées et **une liste des recrutés** apparait.
- Dans cette liste, chaque personne recrutée est représentée par son image, son nom et son prénom.

Les deux captures d'écrans des prochains diapos vous donneront un aperçu du résultat attendu…

CAPTURE 1


Bienvenue dans l'application de gestion de CV
Switch
Ajouter Personne
Nidhal Jelassi
Yasmine Jelassi
Senda Jelassi
Samira Jelassi
Nader Jelassi
Life is Beautiful
Elève en 2eme année
maths, français, anglais
235
Followers
114
Following
35
Projects
Recruter
+ Details
Listes des recrutés

CAPTURE 2


Bienvenue dans l'application de gestion de CV
Switch
Ajouter Personne
Nidhal Jelassi
Yasmine Jelassi
Senda Jelassi
Samira Jelassi
Nader Jelassi
Nidhal Jelassi
Enseignant
35
Life is Beautiful
Auto Rotation
Listes des recrutés
Yasmine Jelassi
Nidhal Jelassi