



Angular 2+

Développer des applications Web

Support de cours

Réf. T44A-046





Module 7

Mise en place d'un service

Contenu du module

- Pourquoi mettre en place un service ?
- Création d'un service
- Déclarer le service dans **app.module.ts**
- Appeler le service depuis le composant principal
- Récupérer les données du service
- Modifier les données du service
- Appeler le service depuis un autre composant



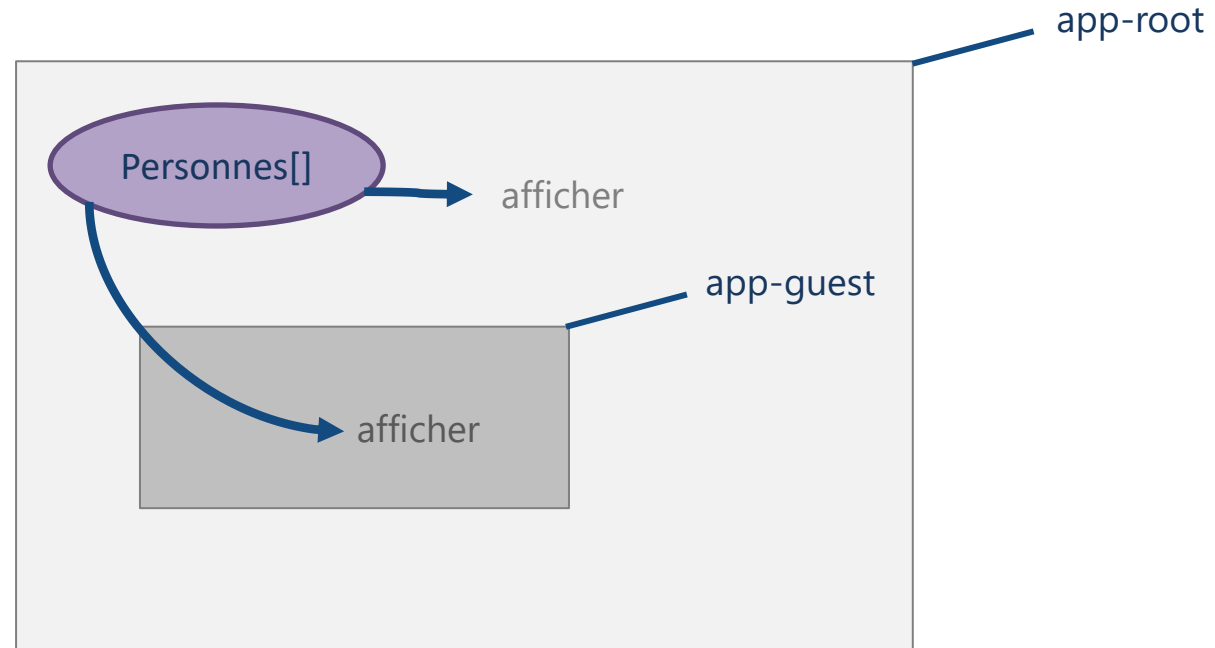
2 piments

Pourquoi créer un service ?

- Partager les données
- Centraliser les données
- Une seule source à jour

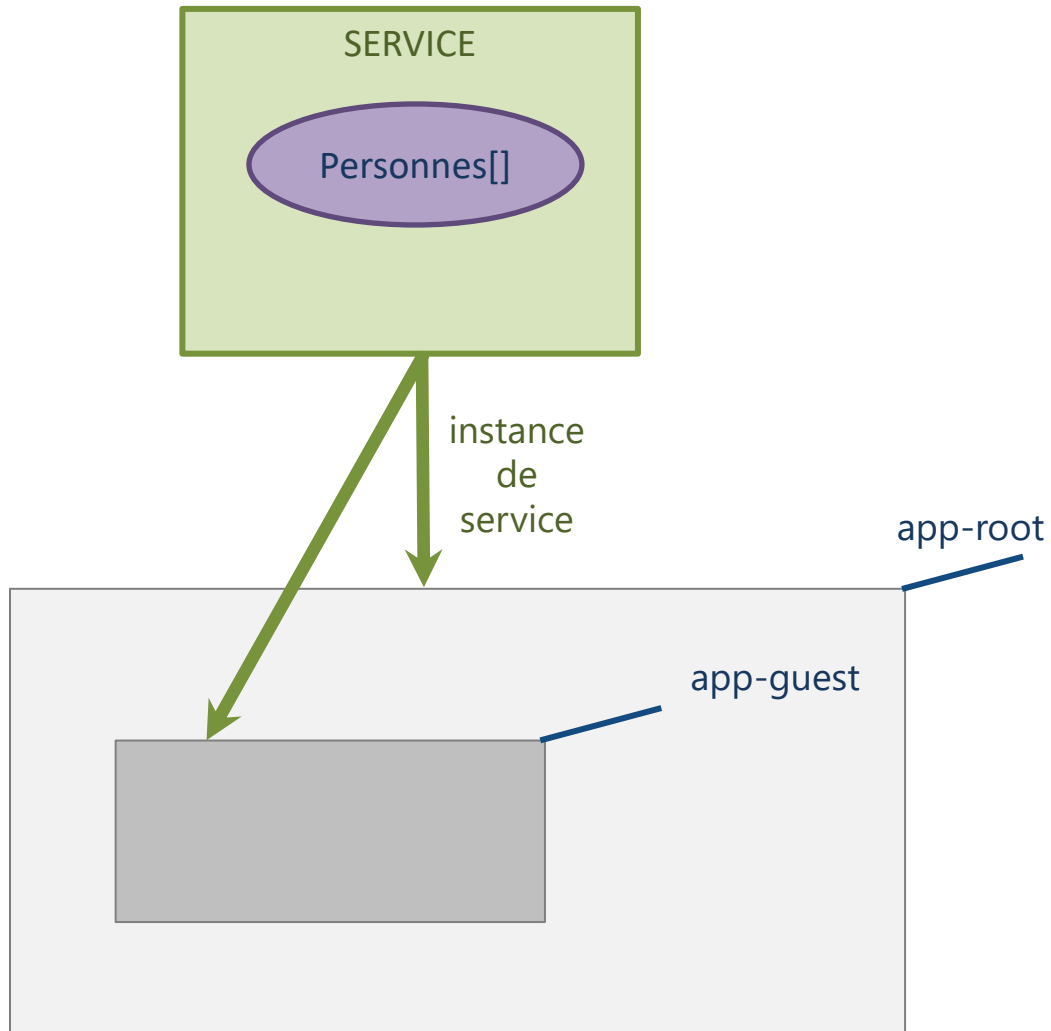
Représentation de l'application

- **AVANT** Le Service



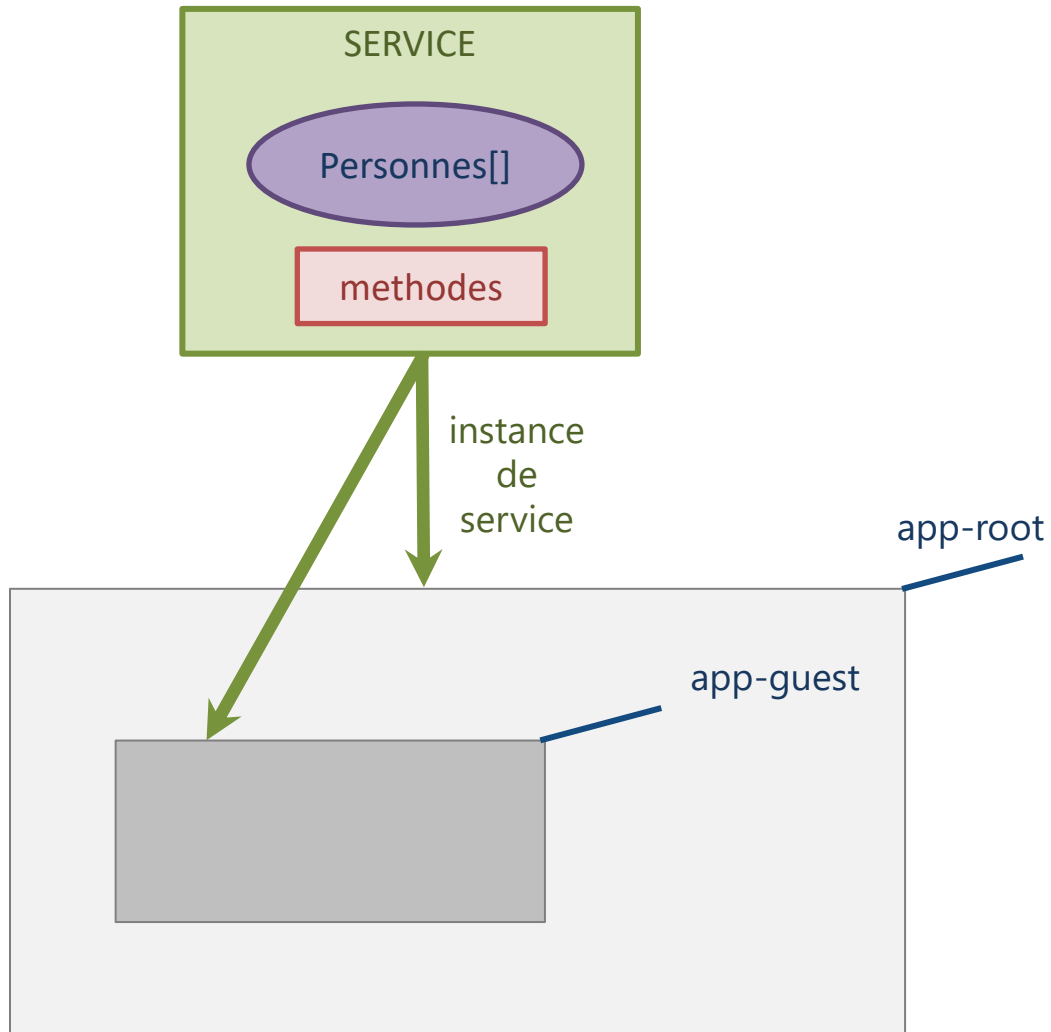
Représentation de l'application

- **APRES** AVEC Le Service
- Le service contient désormais le tableau



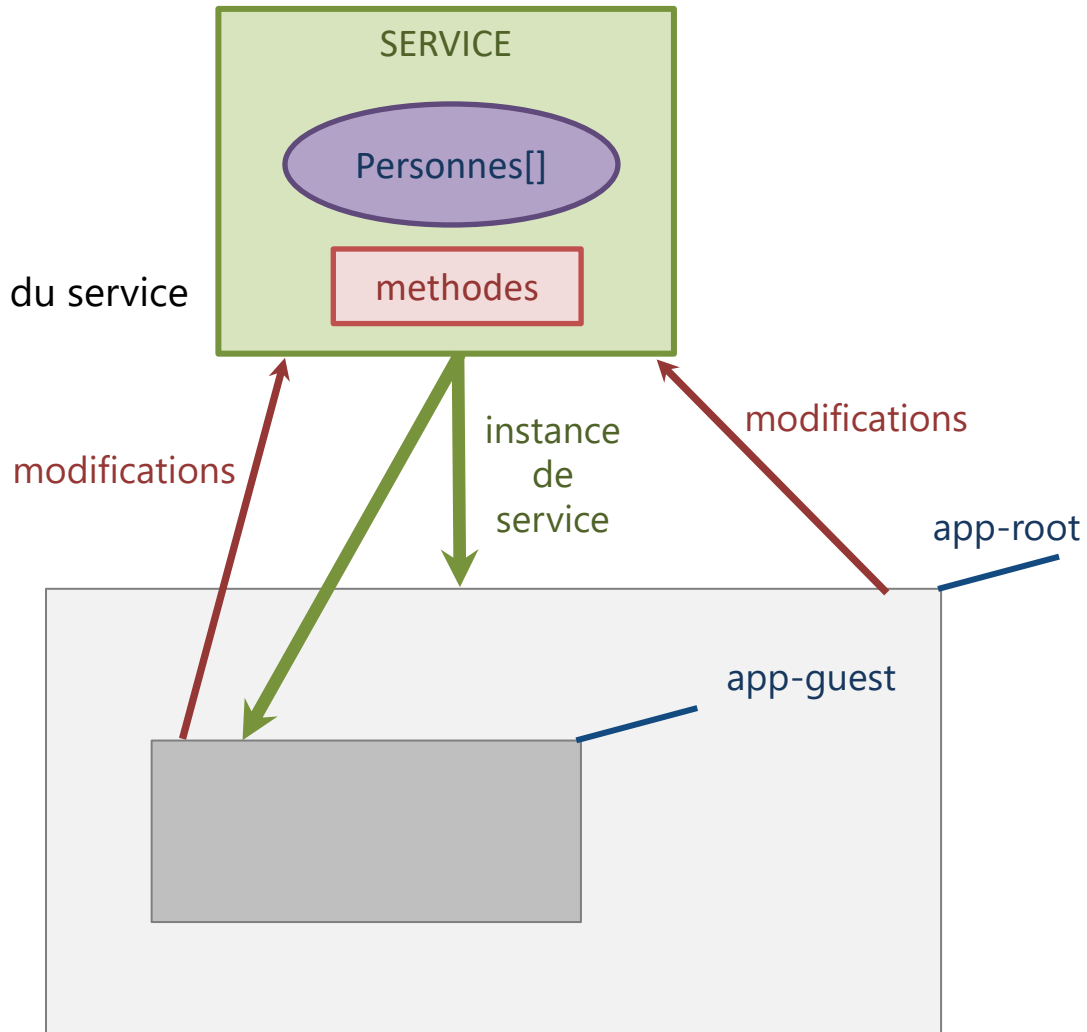
Représentation de l'application

- **APRES** AVEC Le Service
- Les méthodes qui modifient les données
 - Se trouvent **dans** le service



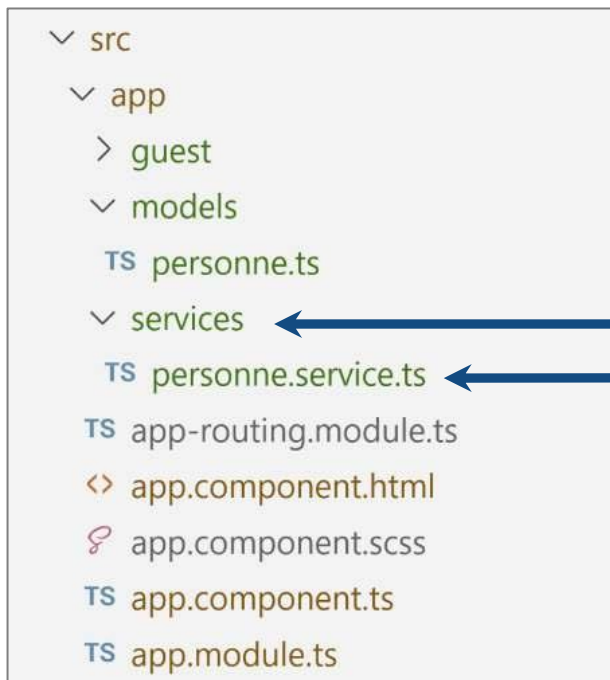
Représentation de l'application

- **APRES** AVEC Le Service
- Pour modifier les données :
 - Les composants appellent les méthodes du service



Création de la classe service

- Création du répertoire **services** dans le répertoire **app**
- Création du fichier **personne.service.ts**



personne.service.ts

```
import { Personne } from "../models/personne";

export class PersonneService{
    personnes:Personne[]=[
        {prenom:'Brad',nom:'PITT'},
        {prenom:'Bruce',nom:'Willis'},
        {prenom:'Angelina',nom:'JOLIE'},
    ];
}
```

Création de la classe service : ajout des méthodes

- Ajouter les méthodes pour modifier les données

personne.service.ts

```
export class PersonneService{  
    personnes:Personne[]=[  
        {prenom:'Brad',nom:'PITT'},  
        {prenom:'Bruce',nom:'Willis'},  
        {prenom:'Angelina',nom:'JOLIE'},  
    ];  
    ajouter (p:Personne):void{  
        this.personnes.push(p);  
    }  
    enlever(i:number){  
        this.personnes.splice(i,1);  
    }  
}
```

← Méthode du service

← Méthode du service

Déclarer le service

- Dans le fichiers **app.module.ts**
- Le service est déclaré dans la partie **providers**
- Vérifier que le la classe du service est bien importée

app.module.ts

```
providers: [  
  |  PersonneService  
],
```

Appeler le service depuis le composant principal

- Dans le fichier TypeScript du composant principal
 - Le service est récupéré par injection de dépendance
 - C'est-à-dire qu'il est instancié dans le constructeur
- Vérifier que la classe du service est bien importée

app.component.ts

```
export class AppComponent {  
  personnes:Personne[]=[];  
  constructor(private personneService:PersonneService){}  
}
```

Ne pas oublier **private** pour permettre l'instanciation automatique

Récupérer les données du service

- Pour agir au chargement du composant
 - Nous allons implémenter la méthode **ngOnInit()** →
 - Elle est définie dans une l'interface **OnInit** →

app.component.ts

```
import { Component ,OnInit} from '@angular/core';
import { Personne } from '../models/personne';
import { PersonneService } from '../services/personne.service';
@Component({
  selector: 'app-root',
  templateUrl: '../app.component.html',
  styleUrls: ['../app.component.scss']
})
export class AppComponent implements OnInit{
  personnes:Personne[]=[];
  constructor(private personneService:PersonneService){}
  ngOnInit(){
    this.personnes = this.personneService.personnes;
  }
}
```


Le tableau du service est récupéré

Appeler les méthodes du service

- Avec l'instance du service
 - Il est possible d'appeler les méthodes du service

app.component.ts

```
ngOnInit(){  
  this.personnes = this.personneService.personnes;  
}  
onAjouter(){  
  let p = new Personne(this.prenom,this.nom);  
  this.prenom = this.nom = ''; // vider les champs input  
  this.personneService.ajouter(p); // appeler une méthode du service  
}
```



Appeler le service depuis un autre composant

- Avec l'instance du service
 - Il est possible d'appeler les méthodes du service

guest.component.ts

```
export class GuestComponent implements OnInit {  
  @Input() p:Personne = new Personne();  
  @Input() i:number=0;  
  constructor(private personneService:PersonneService) {  
  }  
  ngOnInit(): void {  
  }  
  onEnlever(){  
    this.personneService.enlever(this.i);  
  }  
}
```

Avec l'injection de dépendance
L'instance du service est récupérée

A partir de l'instance du service
Les méthodes sont appelées