



Angular 2+

Développer des applications Web

Support de cours

Réf. T44A-046





Module 6

Mon premier component

Contenu du module

- Création d'un composant
- Appeler un composant depuis le composant principal (root)
- Envoyer des données à un composant
- Recevoir des données depuis un composant
- Afficher les données envoyées
- Appeler plusieurs composants depuis un tableau d'objet
- Créer une classe BO **B**usiness **O**bject avec TypeScript

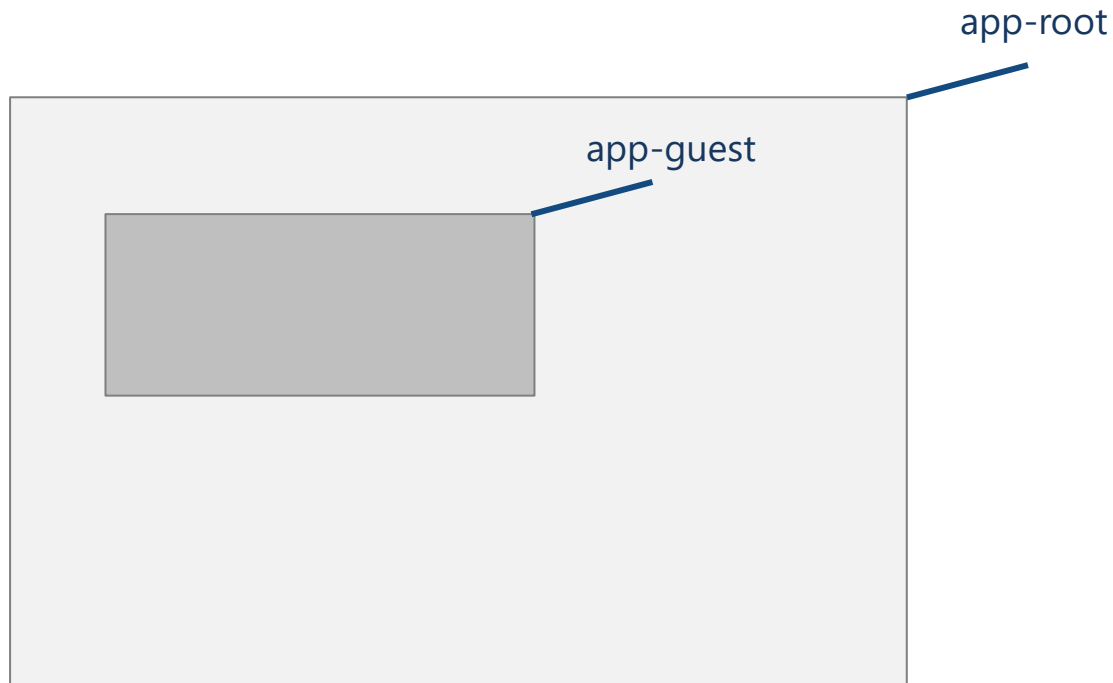
difficulté



1 piment

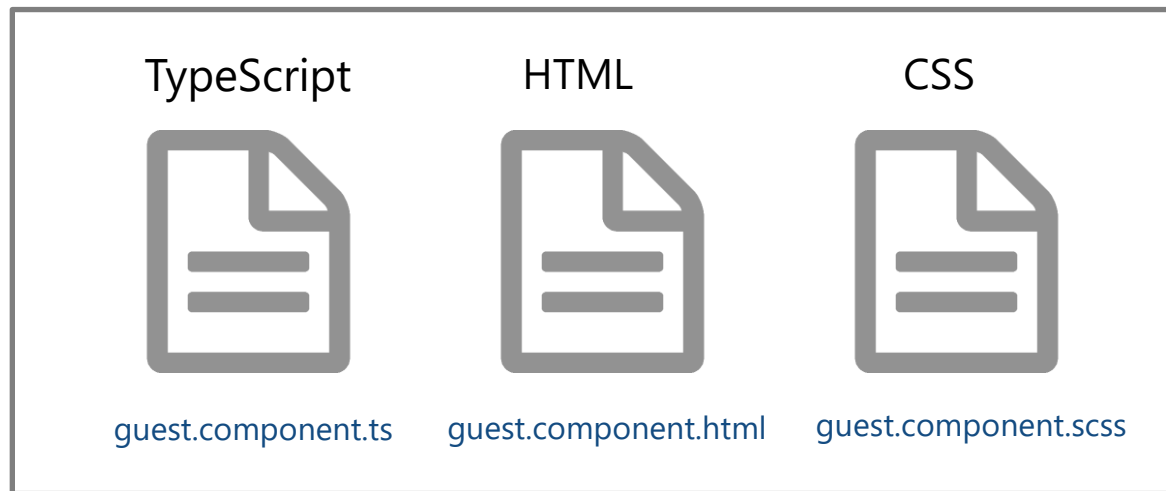
Création d'un composant

- Dans le composant principale nous allons créer et appeler un composant enfant
- Nom du composant enfant guest : **app-guest**



Création d'un composant

- Architecture du composant **guest**



le composant
guest-app

Création d'un composant

- Pour créer un nouveau **composant** : **guest** dans le terminal
- il faut arrêter le serveur : **CTRL + C**

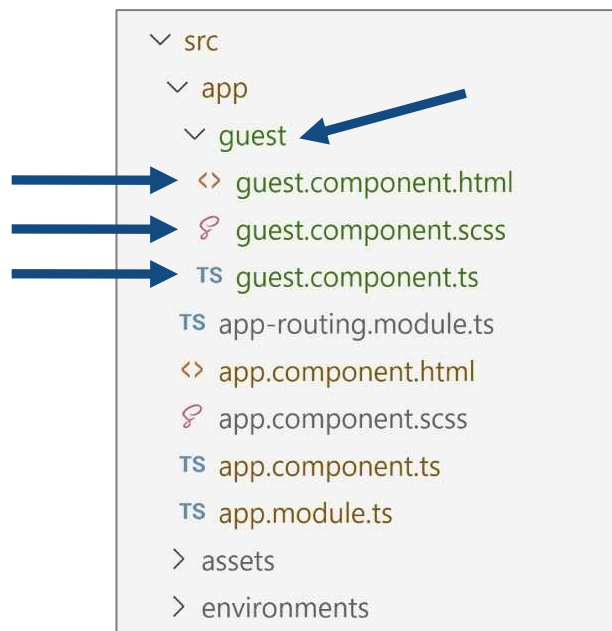
```
ng generate component guest
```

- Ou plus rapide avec la première lettre de **g**enerate et **c**omponent !

```
ng g c guest
```

Création d'un composant

- Un nouveau répertoire est créé : **guest** dans le répertoire **app**
- Les 3 fichiers du composant sont aussi créés
- Le composant est déclaré dans **app.module.ts** au niveau des **declarations**



Appeler un composant depuis le composant principal

- Depuis la vue (HTML) du composant principal **app.component.html**
- Le composant enfant **guest** va être appelé

app.component.html

```
<h1>Personnes :</h1>  
<app-guest></app-guest>  
<app-guest></app-guest>  
<app-guest></app-guest>
```

localhost:4200

Personnes :
guest works !
guest works !
guest works !

- 3 composants enfant sont appelés

Envoyer des données à un composant

- Création d'attributs dans la balise du component enfant
- Les données sont envoyées via les attributs


app.component.html

```
<h1>Personnes :</h1>  
<app-guest nom="Brad" prenom="PITT"></app-guest>  
<app-guest nom="Bruce" prenom="Willis"></app-guest>  
<app-guest nom="Angelina" prenom="JOLIE"></app-guest>
```

Recevoir des données depuis un composant

- A l'aide du décorateur **@Input()** on accuse réception des données
- Afin d'injecter une donnée depuis l'extérieur

guest.component.ts



```
import { Component, Input, OnInit } from '@angular/core';

@Component({
  selector: 'app-guest',
  templateUrl: './guest.component.html',
  styleUrls: ['./guest.component.scss']
})
export class GuestComponent implements OnInit {
  @Input() prenom='';
  @Input() nom='';
  constructor() { }
  ngOnInit(): void {
  }
}
```



```
src
├── app
│   ├── guest
│   │   ├── guest.component.html
│   │   ├── guest.component.scss
│   │   └── TS guest.component.ts
│   ├── TS app-routing.module.ts
│   ├── <> app.component.html
│   ├── & app.component.scss
│   ├── TS app.component.ts
│   └── TS app.module.ts
├── > assets
└── > environments
```

Afficher les données envoyées

- A l'aide de l'interpolation, il est possible d'afficher les attributs du composant

guest.component.html

```
<!-- <p> guest works !</p> -->
<p>
  {{prenom }}
  <strong>{{nom}}</strong>
</p>
```


localhost:4200

Personnes :

PITT **Brad**

Willis **Bruce**

JOLIE **Angelina**



```

  ▾ src
    ▾ app
      ▾ guest
        <> guest.component.html
        🔗 guest.component.scss
        TS guest.component.ts
        TS app-routing.module.ts
        <> app.component.html
        🔗 app.component.scss
        TS app.component.ts
        TS app.module.ts
      > assets
      > environments
```

Appeler plusieurs composants depuis un tableau d'objet

- Plutôt que de "mettre en dur" les données dans la vue HTML
- Il est préférable d'envoyer un tableau d'objets

app.component.ts

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

app.component.html

```
<h1>Personnes :</h1>  
<app-guest  
  *ngFor="let personne of personnes"  
  [nom]="personne.nom"  
  [prenom]="personne.prenom"></app-guest>
```

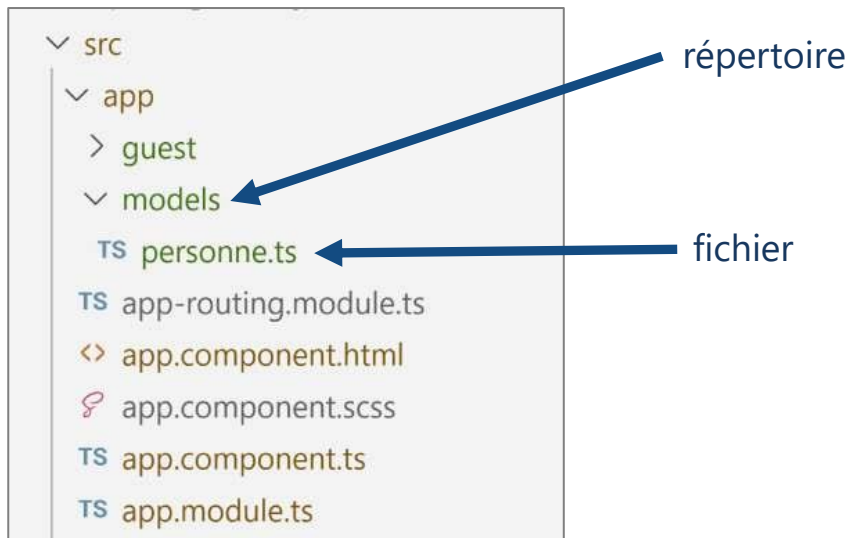


prenom devient [prenom]

Utilisation de la liaison par attribut **attribute binding**
pour envoyer des variables plutôt que des données brutes

Création d'une classe Business Object

- Créer un répertoire **models** dans le répertoire **app**
- Dans ce répertoire créer le fichier **personne.ts**



personne.ts

```
export class Personne{  
    public prenom:string;  
    public nom:string;  
    constructor( prenom:string, nom:string){  
        this.prenom=prenom;  
        this.nom=nom;  
    }  
}
```

personne.ts

```
export class Personne{  
    constructor(  
        public prenom:string,  
        public nom:string){  
    }  
}
```

Utilisation du Business Object

- On peut désormais **typer** notre tableau de personne
- Il faut importer la classe **personne.ts**

```
import { Component } from '@angular/core';
import { Personne } from './models/personne';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  personnes: Personne[] = [
    { prenom: 'Brad', nom: 'PITT' },
    { prenom: 'Bruce', nom: 'Willis' },
    { prenom: 'Angelina', nom: 'JOLIE' },
  ];
}
```

import

typage