esprit Se former autrement	Année Universitaire : 2020-2021 ANGULAR

Workshop n°2 :

Imbrication de composant

Objectifs :

- Appliquer les différents types de data-binding (interpolation, eventBinding, propertybinding et two way databinding)
- Imbriquer deux composants parent et fils
- Appliquer le two-way data-binding
- Manipuler les décorateurs @ input et @output

Etude de cas:

Nous souhaiterions créer une application E-Commerce en utilisant le framework Angular côté frontEnd. Sachant que la partie opérationnelle (la couche métier) est déjà développée et elle est exposée sous forme des services web côté backend.

Créer une SPA (Single Page Application) permettant à l'utilisateur de:

a. parcourir une liste de produits
b. commander un produit,
c. liker un produit,
d. de chercher un produit par prix.

Travail à faire :

1. Afin de rendre l'interface de l'application plus attractive, nous allons installer bootstrap (https://getbootstrap.com/).
   a. Taper la commande npm install bootstrap dans votre terminal
   b. Dans le fichier **src/styles.css**, Ajouter la ligne suivante :

```
@import "~bootstrap/dist/css/bootstrap.min.css";
```

2. En utilisant bootstrap, améliorer le rendu de votre page comme le montre la figure suivante :

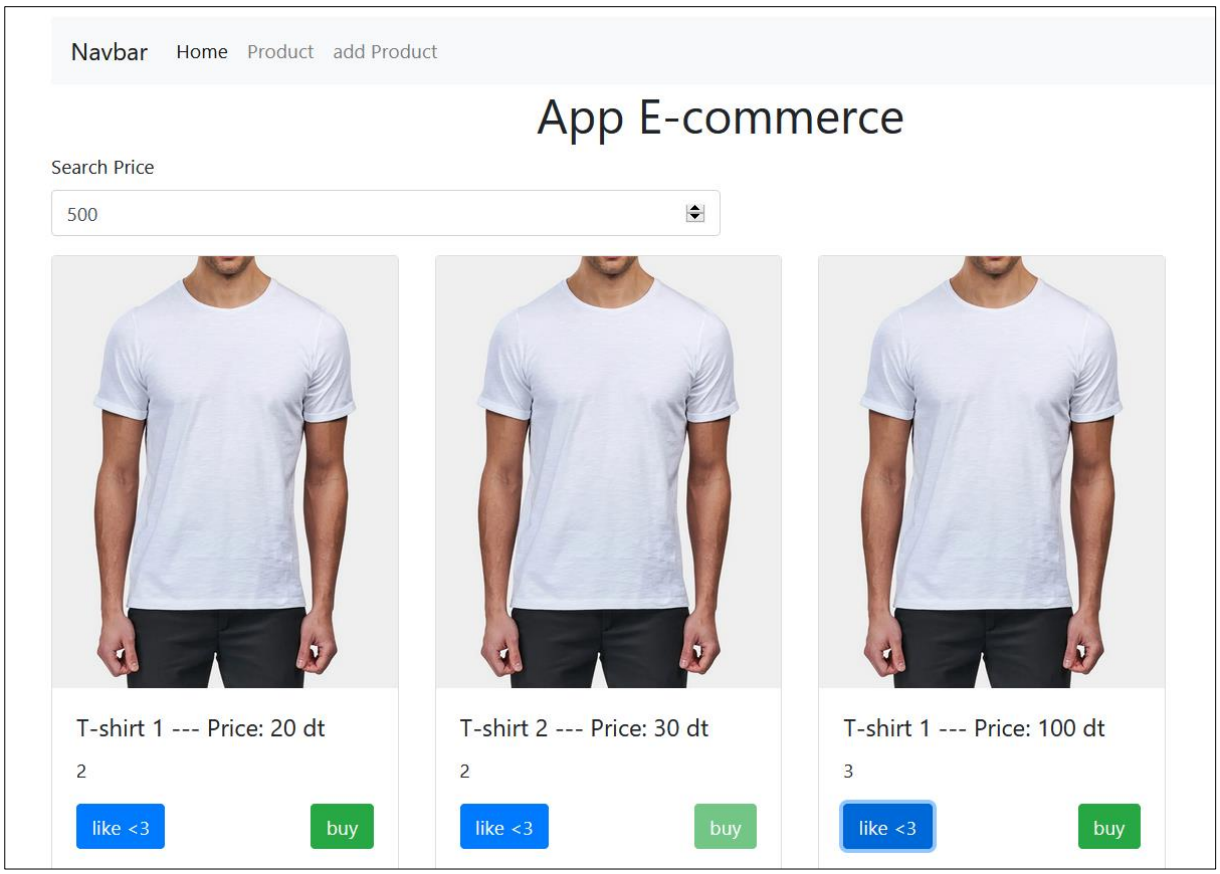

Figure 1 : la page index.html

3. Créer un composant appelé **ProductComponent** en tapant la commande :
   => ng g c Product
4. Le composant « product » sera dédié à l'affichage d'un produit aussi que l'implémentation des méthodes qui sont liées à un produit.

5. Afin d'afficher la liste des produits dans le composant **parent** « productsComponent », on va ajouter le composant **fils** dans la boucle for.

```
<div class="row" >
  <app-product *ngFor="let p of listProduct;let i= index">
  </app-product>
</div>
```

6. Sachant que la liste des produits est déclarée dans le composant parent, comment vous allez assurer le passage des informations du chaque produit au composant fils ?
   a. Au niveau du composant fils, déclarer l'objet produit en tant qu'input

```
export class ProductComponent implements OnInit {
  @Input() product: Product;
   …….
```

   b. Au niveau de composant parent ProductsComponent nous allons faire le lien entre la property input et la liste des produits en utilisant le property binding.

```
<div class="row" >
  <app-product *ngFor="let p of listProduct;let i= index"
               [product]="p">
  </app-product>
</div>
```

7. Faire les mêmes étapes pour le champ input de recherche par prix « priceMax »

8. Le clique sur le bouton « like », qui est affiché dans le composant fils, permet d'incrémenter le nombre de like dans la liste des produits selon le choix de l'utilisateur. Donc notre objectif est de faire le lien entre l'évènement clique qui sera déclencher dans le composant fils et l'action d'incrémentation qui sera implémenter dans le composant parent.

    a. Nous allons créer une « property » dans le composant fils en tant que output de type « EventEmitter() » :

```
export class ProductComponent implements OnInit {
  @Input() product: Product;
  @Input() priceMaxInput: number;
  @Output() incrementEvent = new EventEmitter<Product>() ;
……
```

    b. Donc l'évènement clique sur le bouton « like » dans le composant fils permet d'excuter la méthode sendNotif()

```
<button class="btn btn-primary" (click)="sendNotif()" >
  like <3
</button>
```

    c. La méthode sendNotif() va déclencher un autre évènement qui est « **incrementEvent** » .

```
sendNotif(){
  this.incrementEvent.emit(this.product);
}
```

    d. En utilisant l'**event binding** dans le composant parent, nous allons faire le lien entre l'évènement **incrementEvent** et la méthode **incrementLike()**

```
<app-product *ngFor="let p of listProduct;let i= index"
             [product]="p"
             [priceMaxInput]="priceMax"
             (incrementEvent)="incrementLike($event)">
</app-product>
```

```
incrementLike(product: Product){
  let i = this.listProduct.indexOf(product);
  this.listProduct[i].like++;
}
```