Atelier Angular :

Transmettre les données entre composants avec @Input() et @Output

On se propose dans cet atelier de développer une interface pour gérer les opération CRUD sur les catégories des produits en utilisant les décorateurs @Input() et @Output

Objectifs :

1. Créer un composant pour lister toutes les catégories,
2. Créer un composant pour ajouter et modifier une catégorie,
3. Transmettre une catégorie d'un composant à l'autre,
4. Ajouter une catégorie à l'aide de @Output() et EventEmitter,
5. Modifier une catégorie à l'aide de @Output() et EventEmitter.

Créer un composant pour lister toutes les catégories

1. Créer le composant Web ListeCategories en tapant :

   ng g c ListeCategories --skip-tests -s

```
export class ListeCategoriesComponent implements OnInit {

  categories! : Categorie[];

  constructor(private produitService : ProduitService) { }

  ngOnInit(): void {
    this.produitService.listeCategories().
      subscribe(cats => {this.categories = cats._embedded.categories;
      console.log(cats);
    });
  }
```

2. Modifier le fichier liste-categories.component.html comme suit :

```
<div class="container">
    <div class="card shadow mb-4">
        <div class="card-body">
            <table class="table table-striped">
                <thead>
                    <tr> <th>Id</th><th>Nom Catégorie</th> </tr>
                </thead>
            @for(cat of categories; track cat.idCat)
               {
                <tbody >
                    <tr>
                        <td>{{cat.idCat}}</td>
                        <td>{{cat.nomCat}}</td>
                     </tr>
                </tbody>
               }
            </table>
        </div>
    </div>
</div>
```

3. Ajouter le composant ListeCategories au fichier app.routes.ts et à la navbar

```
{path: "listeCategories", component : ListeCategoriesComponent},
```

```
<a class="dropdown-item" routerLink="/listeCategories" >Gestion des catégories</a>
```

4. Testez votre travail

Créer un composant pour ajouter et modifier une catégorie

5. Créer le composant updateCategorie :

   ng g c UpdateCategorie --skip-tests -s

6. Ajouter le composant au début du fichier liste-categories.component.html juste avant la table :

```
<div class="col-sm-4 col-md-4 col-lg-4 mb-4">
      <app-update-categorie  ></app-update-categorie>
 </div>
```

Importer le composant UpdateCategorieComponent dans le composant ListeCategoriesComponent

```
imports: [CommonModule,UpdateCategorieComponent],
```

7. Ajouter l'attribut *categorie* au composant UpdateCatgorie :

```
@Input()
categorie! : Categorie;
```

Transmettre une catégorie d'un composant à l'autre

8. Transmettre une catégorie en appelant le composant au niveau du fichier liste-categories.component.html :

```
   updatedCat:Categorie = {"idCat":0,"nomCat":""};
```

```
  <app-update-categorie   [categorie]="updatedCat" ></app-update-categorie>
```

9. Afficher la catégorie transmise au niveau de la méthode ngOnInit() du composant UpdateCatgorie :

```
ngOnInit(): void {
    console.log("ngOnInit du composant UpdateCategorie ",this.categorie);
  }
```

10. Modifier le fichier update-categorie.component.html comme suit :

```
<form>
    <div >
     <label>ID Catégorie</label>
     <input type="text" [(ngModel)]="categorie.idCat" name="idCategorie" class="form-control w-25">
    </div>
    <div>
      <label>Nom Catégorie</label>
     <input type="text" [(ngModel)]="categorie.nomCat" name="nomCategorie" class="form-control"
placeholder="Entrez un nom de catégorie...">
     </div>
     <div class="mt-2">
       <button type="submit" [disabled]="!categorie.nomCat" class="btn btn-success"
(click)="saveCategorie()" >Ajouter</button>
    </div>
</form>
```

Au niveau du fichier update-categorie.component.ts :

```
  imports: [FormsModule],
```

Ajouter une catégorie à l'aide de @Output() et EventEmitter

11. Ajout, à la classe UpdateCategorie l'attribut suivant :

```
  @Output()
  categorieUpdated = new EventEmitter<Categorie>();
```

12. Modifier la méthode saveCategorie() de la classe UpdateCategorieComponent comme suit :

```
saveCategorie(){
    this.categorieUpdated.emit(this.categorie);
  }
```

13. Modifier l'appel au comopsant UpdateCategorie au niveau du fichier liste-categories.component.html, pour récupérer l'event «categorieUpdated » emis par la méthode saveCategorie :

```
 <app-update-categorie
             [categorie]="updatedCat"
             (categorieUpdated)= "categorieUpdated($event)" >
 </app-update-categorie>
```

14. Ajouter la méthode ajouterCategorie à la classe ProduitService :

```
 ajouterCategorie( cat: Categorie):Observable<Categorie>{
      return this.http.post<Categorie>(this.apiURLCat, cat, httpOptions);
     }
```

15. Ajouter la méthode categorieUpdated à la classe ListeCategoriesComponent :

```
categorieUpdated(cat:Categorie){
    console.log("Cat updated event",cat);
    this.produitService.ajouterCategorie(cat).
      subscribe( ()=>  this.chargerCategories());
  }
```

```
chargerCategories(){
```

```
    this.produitService.listeCategories().
      subscribe(cats => {this.categories = cats._embedded.categories;
      console.log(cats);
    });
  }
```

Modifier une catégorie à l'aide de @Output() et EventEmitter

16. Ajouter le bouton « Editer » à la liste des catégories :

```
<td><a class="btn btn-success"  (click)="updateCat(cat)">Editer</a></td>
```

17. Ajouter la méthode updateCat :

```
updateCat(cat:Categorie) {
    this.updatedCat=cat;
  }
```

Modifier le label du bouton « Ajouter » ou « Modifier » selon le contexte

18. Ajouter l'attribut *ajout* à la classe UpdateCategorieComponent

```
 @Input()
 ajout!:boolean;
```

19. Ajouter l'attribut ajout à la classe ListeCategoriesComponent

```
  ajout:boolean=true;
```

20. Modifier la méthode updateCat de la classe ListeCategoriesComponent comme suit :

```
updateCat(cat:Categorie) {
    this.updatedCat=cat;
    this.ajout=false;
  }
```

21. Modifier l'appel au comopsant UpdateCategorie au niveau du fichier liste-categories.component.html

```
<app-update-categorie
              [categorie]="updatedCat"
              (categorieUpdated)= "categorieUpdated($event)"
              [ajout]="ajout">
 </app-update-categorie>
```

22. Modifier le code du bouton comme suit :

```
<button type="submit" [disabled]="!categorie.nomCat" class="btn btn-success"
          (click)="saveCategorie()" >{{ajout ? 'Ajouter' : 'Modifier'}}
</button>
```

Cacher l'id Catégorie lorsqu'il s'agit d'un ajout

```
<div [hidden]="ajout" >
      <label>ID Catégorie</label>
      <input type="text" [(ngModel)]="categorie.idCat" name="idCategorie" class="form-control w-25">
  </div>
```