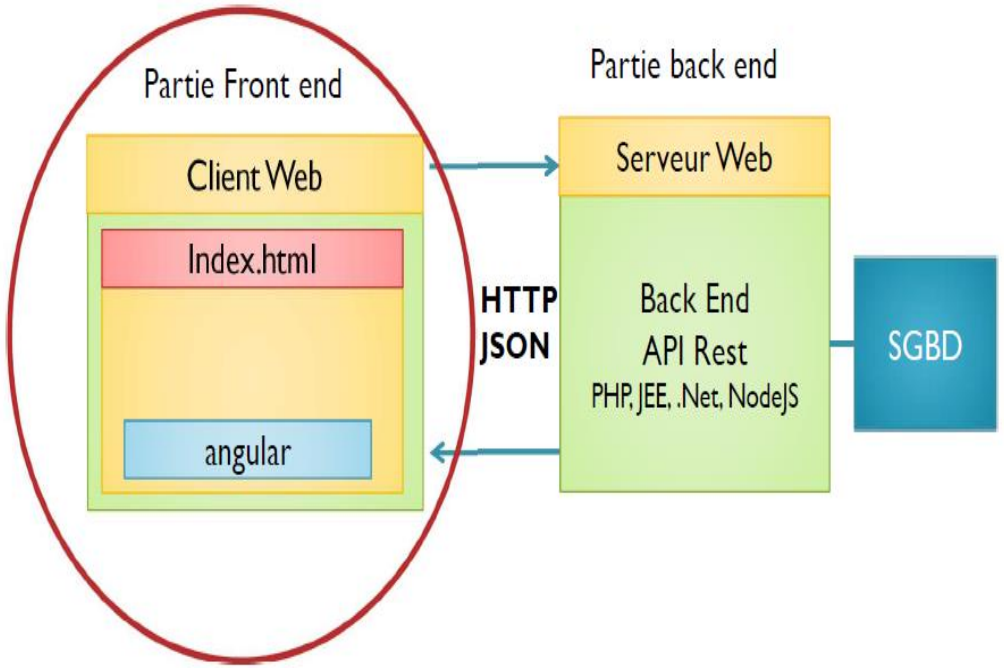
Formation

Introduction

- Angular est un Framework de developpement Javascript. Il permet de créer des applications client (Client-side application) en utilisant HTML, CSS et Javascript (TypeScript)
- Angular permet de créer des applications Web basées sur une seule page (Single Page Application)
- Une SPA est une application qui contient une seule page HTML (index.html) récupérée du serveur.



Historique

- **Angular 1 (Agular JS) :**
  - Première version de Angular qui est la plus populaire.
  - Elle est basée sur une architecture MVC coté client. Les applications Angular 1 sont écrite en Java Script.
- **Angular 2 (Angular) :**
  - Est une réécriture de Angular 1 qui est plus performante, mieux structurée et représente le futur de Angular.
  - Les applications de Angular2 sont écrites en Type Script qui est compilé et traduit en Java Script avant d'être exécuté par les Browsers Web.
  - Angular 2 est basée sur une programmation basée sur les Composants Web (Web Component)
- **Angular 4, 5, 6,7,8,… :** sont de simples mises à jour de Angular 2 avec des améliorations au niveau performances.

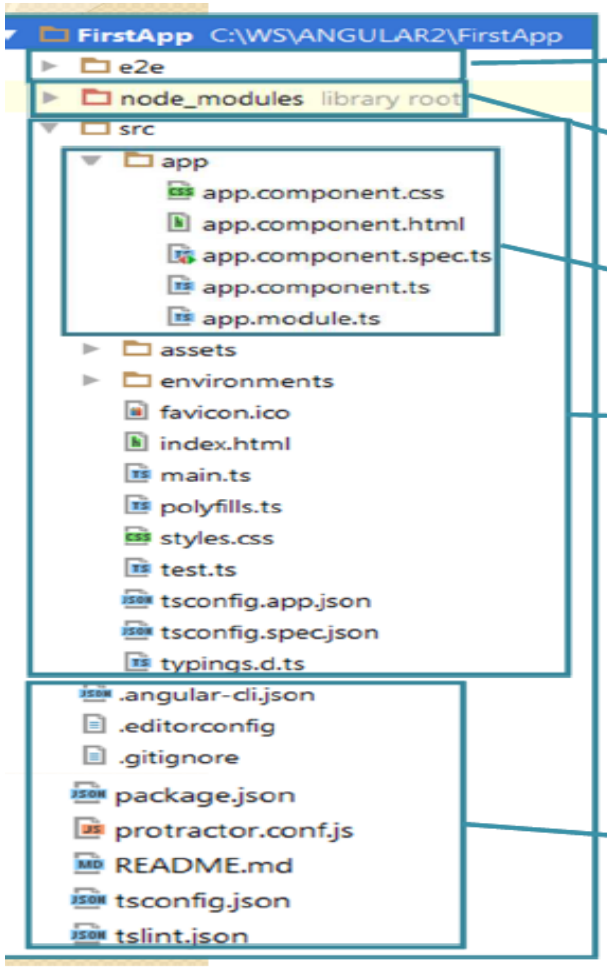
Préparation de l'environnement

- Installation de Node Js (www.NodeJs.org)
- Installation de Angular CLI : *npm install -g @angular/cli*
- Installation d'un IDE : Visual Studio code

Commandes CLI à utiliser fréquemment

Commandes	Besoin
ng version	afficher la version de angular Cli
ng new firstApp	Créer un projet angular
ng serve	Exécuter un projet
ng g component componentName (ng g c componentName), *(ng g c ComponentName - -skip-tests=true)*	Créer un composant
ng g service serviceName	Créer un service
ng g class className	Créer une classe
ng g i interfaceName	Créer une interface

Structure du Projet Angular



Dossier des scripts des tests unitaires

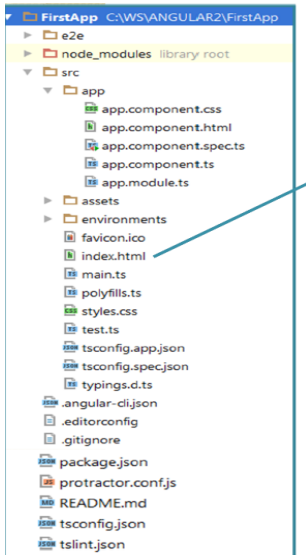
Dépendances externes du projets (Librairies Java Script et CSS)

Logique applicative de votre projet : Les composants, Les services ..
C'est dans ce dossier app ou vous allez passer votre temps de dev

Code source relatif à votre projet

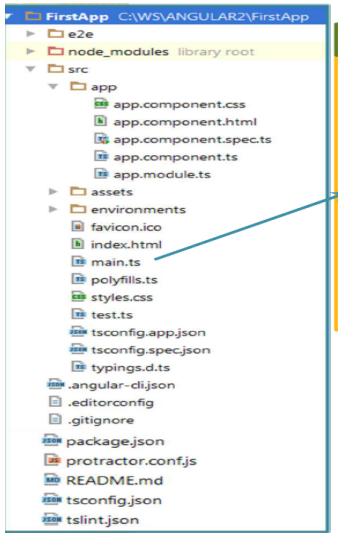
Fichier de configuration du projet

Structure du Projet Angular



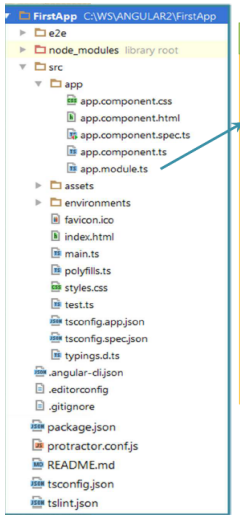
index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>FirstApp</title>
  <base href="/">

  <meta name="viewport" content="width=device-
width, initial-scale=1">
  <link rel="icon" type="image/x-icon"
href="favicon.ico">
</head>
<body>

  <app-root> </app-root>

</body>
</html>
```



main.ts

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from
'@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from
'./environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule);
```



app.module.ts

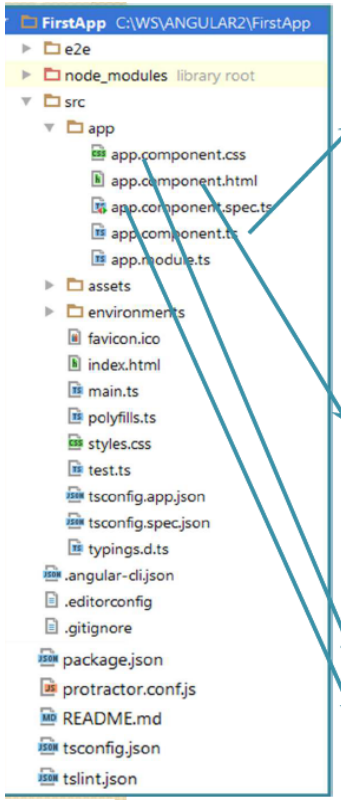
```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Declarations : C'est le tableau de composants. Si un nouveau composant est créé, il sera importé en premier et la référence sera incluse dans les déclarations

Imports : C'est un tableau de modules requis pour être utilisé dans l'application providers**.** Cela va contenir tous les services crées

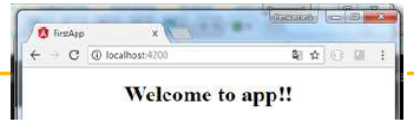
Bootstrap : Cela inclut le composant principal de l'application pour démarrer l'exécution

Structure du Projet Angular



app.component.ts

```ts
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app';
}
```



app.component.html

```html
<div style="text-align:center">
  <h1>
    Welcome to {{title}}!!
  </h1>
</div>
```

app.component.css

Les fichiers spec sont des tests unitaires pour vos fichiers source. La convention pour les applications Angular2 est d'avoir un fichier .spec.ts pour chaque fichier .ts. Ils sont exécutés à l'aide du framework de test javascript Jasmine via le programme de tâches Karma lorsque vous utilisez la commande '**ng test**'.

Chaque composant se compose principalement des éléments suivants :

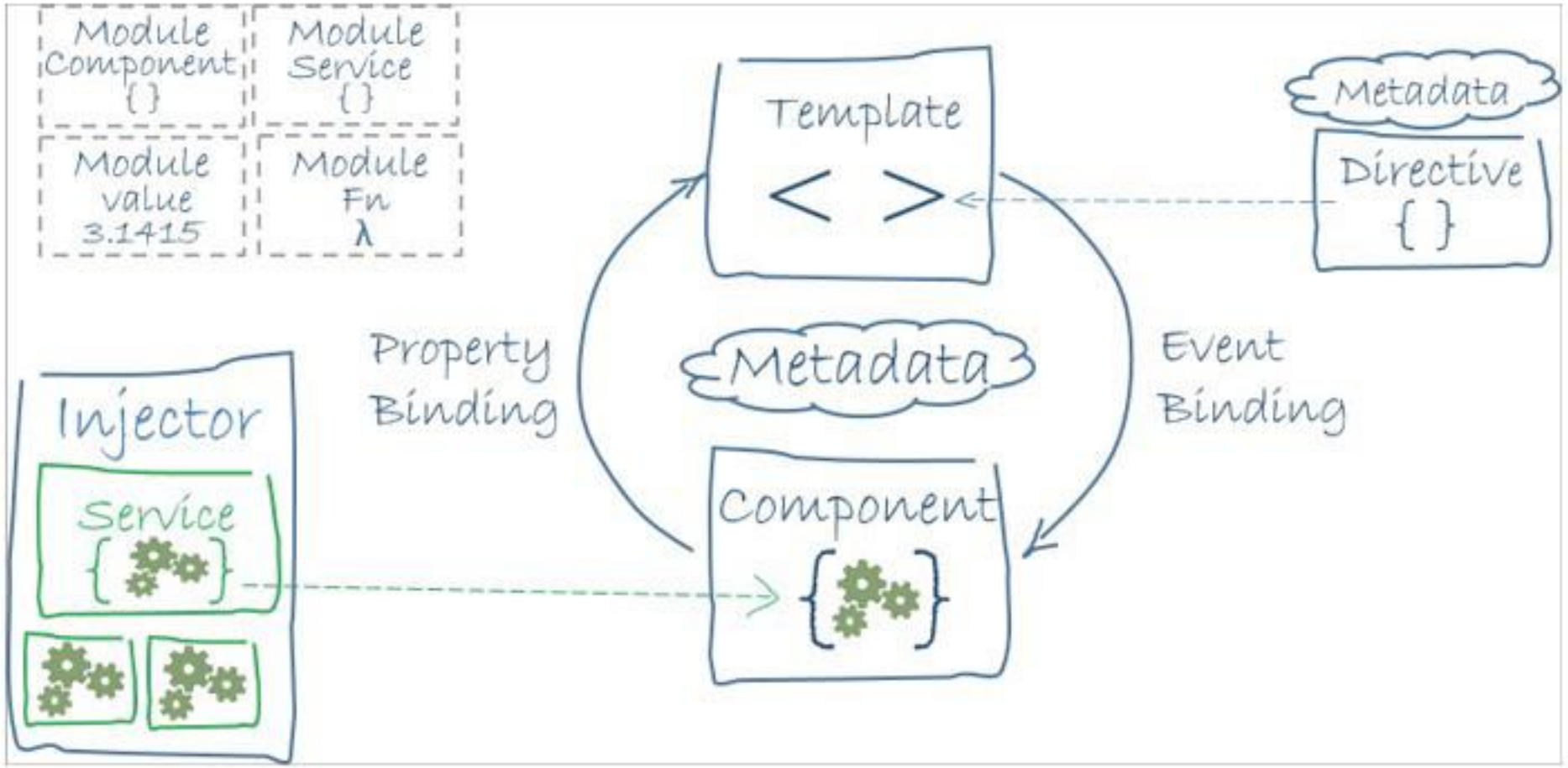
- HTML Template : représentant sa vue
- Une classe représentant sa logique métier
- Une feuille de style CSS
- Un fichier spec sont des tests

Un composant peut être inséré dans n'importe quelle partie HTML de l'application en utilisant son sélecteur associé.

Un composant est une classe qui possède le décorateur ***@Component***. Ce décorateur possède les propriétés suivantes :

- **selector** : permet de spécifier le tag (nom de la balise) associé à ce composant.
- **templateUrl** : permet d'associer un fichier externe HTML contenant la structure de la vue du composant
- **styleUrls** : spécifier les feuilles de styles CSS associées à ce composant

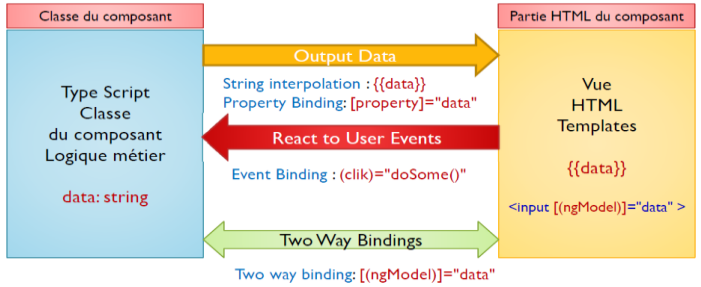
Architecture de base d'une application Angular



DataBinding

- Pour insérer dynamiquement des données de l'applications dans les vues des composants, Angular définit des techniques pour assurer la liaison des données.

- **Data Binding = Communication**



String interpolation:est une technique de One Way Binding, elle utilise l'expression {{ }} pour afficher les données du composant dans la vue.
Exemple:

```
export class AppComponent {
  title = 'TP 1 Angular';

}
```

app.component.ts

```
<h2>{{title}}</h2>
```

app.component.html

Property Binding : est une autre technique One Way Binding. Elle permet de lier une propriété de la vue avec une propriété définie dans le composant.

Exemple:

```
urlImg="./assets/images/pc portable.jfif"
```

```
<img [src]=urlImg>
```

Event Binding : dans Angular , event binding est utilisé pour gérer les événements déclenchés comme le clic de bouton, le déplacement de la souris, etc. Lorsque l'événement se produit, il appelle la méthode spécifiée dans le composant.

Exemple:

```
afficher()
{
  console.log("hello");
}
```

```
<button (click)="afficher()">Afficher</button>
```

Two-way binding : nous avons vu que dans la le One Way Binding, tout changement dans la vue n'était pas reflété dans le composant. Pour résoudre ce problème, Angular Two Way Binding.

```
texte:string="hello";
```

```
<input type="text" [(ngModel)]=texte> {{texte}}
```

<u>Remarque :</u> il faut ajouter « ***FormsModule***» dans le tableau ***imports*** du fichier ***app.module.ts***

Les directives

- Les directives sont des instructions intégrées dans le DOM. Quand Angular lit le template et rencontre une directive qu'il reconnait, il suit les instructions correspondantes.

- **La directive *ngFor** : elle est très utile dans le cas d'un tableau et qu'on a besoin de répéter un traitement donné.

  Exemple *: <ul> <li *ngFor="let p of produits">{{ p.nom }}
*

- La directive *ngIf : est utilisée lorsque vous souhaitez afficher ou supprimer un élément en fonction d'une condition. Prend un booléen en paramètre. Tout comme nous sommes habitués à d'autres langages de programmation, la directive angular ngIf nous permet également de déclarer un bloc *else*. Ce bloc est affiché si l'instruction définie dans le bloc principal put être fausse.

  Exemple : *<div *ngIf="x%2 != 0 ;else sinon "> impair …* ***<ng-****template #sinon>pair</ng-template>*

- **La directive [ngStyle] :** Cette directive permet d'appliquer des styles à un objet du DOM de manière dynamique.

  Exemple: <h4 [ngStyle]="{color: getColor()}">{{produits[0].nom}}</h4>

  La fonction getColor() retourne la couleur du titre h4. Cette méthode est développée dans le fichier .ts.

Les pipes

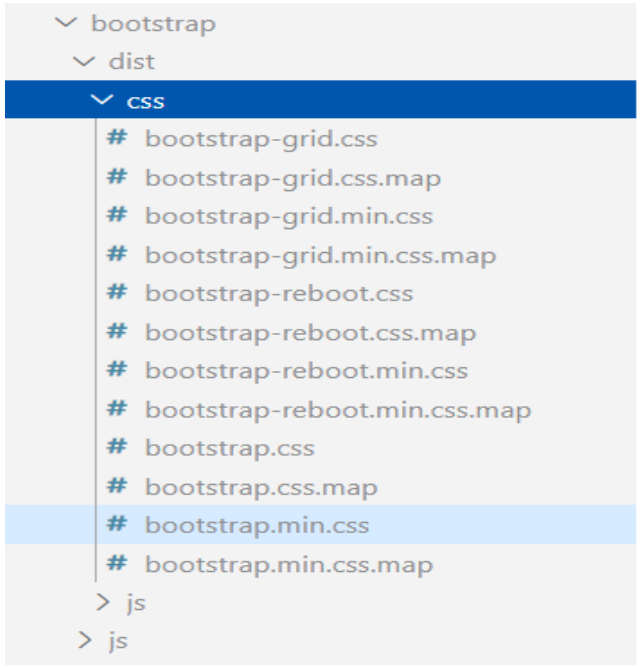
- Les Pipes sont des filtres utilisables directement depuis la vue afin de transformer les valeurs lors du "binding". Il existe plusieurs pipes natives dans Angular comme par exemple DecimalPipe, UpperCasePipe, CurrencyPipe, etc.

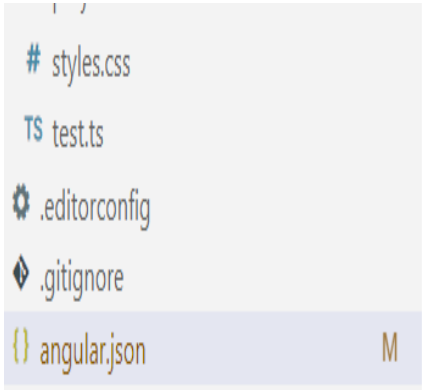
- Exemple: *{{nom|uppercase}}, {{prix|currency:'TND':'symbol':'2.2-2'}}*

- Vous trouverez dans ce lien les pipes natives d'Angular : https://angular.io/api?type=pipe

Ajouter le framework Bootsrap au projet

- Il faut tout d'abord installer Bootstrap avec npm : *npm install bootstrap –save*
- *Ensuite, ajouter le chemin de bootstrap.min.css dans le fichier angular-cli.json*






```
25        ],
26        "styles": [
27          "src/styles.css",
28          "node_modules/bootstrap/dist/css/bootstrap.min.css"
29        ],
30        "scripts": []
```

Exercice d’application

Rechercher un produit

Mot Clé:

cacher	id	nom	prix	quantite
	1	PC PORTABLE	1,200.54	100
	2	IMPRIMANTE	356.54	70
	3	SMART PHONE	700.40	0

Routage et Navigation

- Le routeur angulaire permet la navigation d'une vue à l'autre lorsque les utilisateurs exécutent des tâches d'application.
- Le routeur angulaire est un service facultatif qui présente une vue de composant particulière pour une URL donnée.
- Il ne fait pas partie du noyau angulaire.
- C'est dans son propre paquet de bibliothèque, @angulaire/router.

Remarque : *Angular-CLI nous propose le module applicatif AppRoutingModule permettant d'ajouter la fonctionnalité de routage à une application.*

un module regroupant les directives et les services paramétrables permettant de remplir la fonctionnalité de routage

un Array contenant la déclaration des routes

```
import ...
import { RouterModule, Routes } from '@angular/router';
const appRoutes:Routes=[
      {path:"products",component:ListeProduitsComponent},
    {path:"accueil",component:AccueilComponent},
    {path:"products/:id",component:ProduitDetailsComponent},
    {path:"",redirectTo:"/accueil",pathMatch:'full'},
    {path:"**", component:PageNotFoundComponent }
]
@NgModule({
  declarations: [
    AppComponent,ListeProduitsComponent,AccueilComponent,
    ProduitDetailsComponent],
  imports: [
    BrowserModule,FormsModule,RouterModule.forRoot(appRoutes) ],
  providers: [],
  bootstrap: [AppComponent]})
export class AppModule { }
```

url du composant

la classe du composant concerné

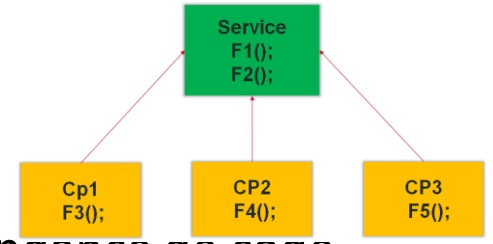
Application du routage

- Pour appliquer un routage dans une vue (HTML Template), on a besoin des directives suivantes:
- **routerLink**: c'est une directive prenant en valeur le chemin (path) indiqué dans la table routes.
- **<router-outlet>**: Elle précise l'emplacement ou la vue du composant appelé sera affichée.
- Quand l'utilisateur tape http://localhost:4200/products , le routeur cherche et charge le composant *ListeproduitsComponent* et l'affiche dans un élément

  <router-outlet></router-outlet> .

- Cet élément est sensé se trouver dans la vue du composant racine.

Les 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.
- Un service est une classe, qui peut contenir du code réutilisable, ou des données qu'on veut partager entre plusieurs composants.
  - Service = conteneur de code réutilisable
  - Service = conteneur de données partagées
  - Création d'un service : **ng g s services/test**

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

@Injectable({  providedIn: 'root'})

export class TestService {

 constructor() { }}
```

- Un service est associé à un composant en utilisant **l'injection de dépendance** (un "design pattern" qui consiste à séparer l'instanciation d'une dépendance et son utilisation)
- Un service peut donc :
  - Interagir avec les données (fournit, supprime et modifie)
  - Faire tout traitement métier (calcul, tri, extraction …)

Les formulaires

Angular propose deux types de forms :

Template driven forms (piloté par le template) :

- Utilisant FormsModule
- Basé sur les directives (ng model)
- Écrit en HTML
- Facile et conseillé pour les formulaires simples

Reactive forms :

- Basé sur ReactiveFormsModule
- Crée dans le composant,
- Basé sur les validations par les fonctions et les patterns.
- Conçu pour les applications nécessitant des contrôles particuliers.

Reactive Forms

Ajouter ***ReactiveFormsModule*** dans la section imports de app.module.ts.

```
.....
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
.....
export class AddProduct2Component implements OnInit {
produitForm: FormGroup;
constructor(private service:ProduitSerService,private fb:FormBuilder) { }
```

commençons par définir un objet de type FormGroup (Composée de plusieurs objets de type FormControl)

FormBuilder : classe service défini par **Angular**, pour l'utiliser, il faut l'injecter dans le constructeur. Elle permet de simplifier la création d'un objet FormGroup (avec la méthode ***group()***) en évitant les répétitions de FormControl

Pour définir une règle de validation, on peut utiliser la classe **Angular** ***Validators*** contenant plusieurs règles de validation

```
ngOnInit() {
  this. produitForm =this.fb.group({
    nom:['',[Validators.required,Validators.minLength(4),
             Validators.pattern('[a-zA-Z _]+')]],
    prix:[0,[Validators.required,Validators.min(2)]],
    quantite:[0,[Validators.required,Validators.min(10)]],
  });
}
onsave()
{let p:Produit=this.produitForm.value;
```

Sur la balise <form> , vous utilisez le property binding pour lier l'objet ***produitForm*** à l'attribut formGroup du formulaire, créant la liaison pour Angular entre le template et le TypeScript.

```
<form [formGroup]="produitForm" (ngSubmit)="onsave()" >
 <div class="form-group">
   <label for="nom">Nom:</label>
   <input type="text"  id="nom" class="form-control"
          placeholder="nom du produit"
          formControlName="nom"
 [ngClass]="{'is-invalid':
            produitForm.get('nom').touched  && !produitForm.get('nom').valid}"
   >
 </div >
 ……
   <button class="btn btn-success" [disabled]=! produitForm.valid>
       Ajouter
  </button>
```

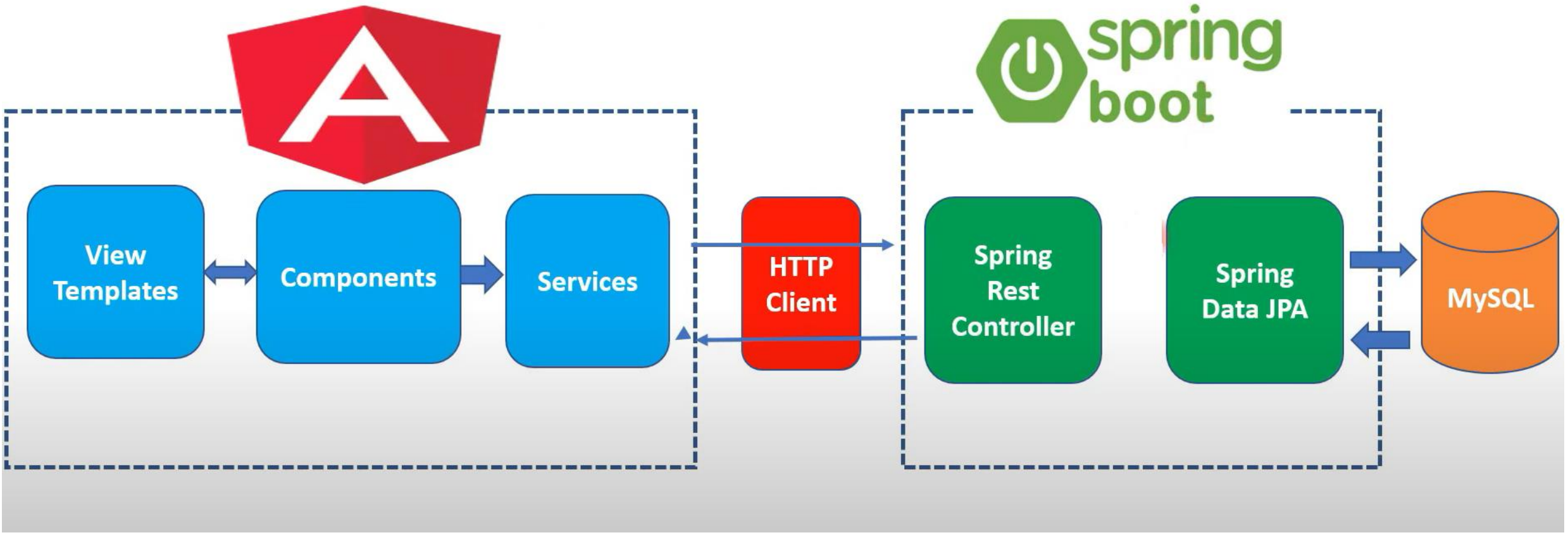
vous n'avez plus besoin de passer le formulaire comme argument puisque vous y avez déjà accès par l'objet ***produitForm*** que vous avez créé.

Sur chaque <input> qui correspond à un control du formulaire, vous ajoutez l'attribut ***formControlName*** où vous passez un string correspondant au nom du control dans l'objet TypeScript.

Pour récupérer le ***FormControl*** associé à nom

Partie Front-End de l’application Gestion de Produits

Architecture d’une application Spring boot-Angular



Model (ng g i model/**)

categorie.ts

```
import { Produit } from "./produit";

export interface Categorie {
      id:number;
      nom:string;
     }
```

Produit.ts

```
import { Categorie } from "./categorie";

export interface Produit {

      id:number;
      nom:string;
      prix:number;
      quantite:number;
      photo:string;
      categorie:Categorie;
}
```

CategorieService (ng g s services/categorie)

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { Categorie } from '../categorie';

@Injectable({
  providedIn: 'root'
})
export class CategorieService {
  host="http://localhost:8080/api/";
  constructor(private client:HttpClient) { }

  public getAllCategories():Observable<Categorie[]>
  {
    return this.client.get<Categorie[]>(this.host+"categories");
  }
  public getCategorie(id):Observable<Categorie>
  {
    return this.client.get<Categorie>(this.host+"categorie/"+id);
  }

}
```

ProduitService (ng g s services/produit)

```
export class ProduitService {
 host="http://localhost:8080/api/";
  constructor(private client:HttpClient) { }

  public getAllProducts():Observable<Produit[]>
  {
    return this.client.get<Produit[]>(this.host+"products");
  }

  public getProductsByCategorie(idcat:number):Observable<Produit[]>
  {
    return this.client.get<Produit[]>(this.host+"productsByCat/"+idcat);
  }
addProduct(formData:FormData):Observable<void>
  {
    return this.client.post<void>(this.host+"addProduct",formData);
  }
  /*public getProductById(id:number):Observable<Produit>
  {
    return this.client.get<getResponse>(this.host+"produits/"+id).pipe(
      map(data=>data._embedded.produit)
    )
  }*/
```

```
public getProductById(id:number):Observable<Produit>
  {
    return this.client.get<Produit>(this.host+"product/"+id)
  }

public supprimer(id:number):Observable<void>
  {
    return this.client.delete<void>(this.host+"deleteProduct/"+id);
  }
    public updateProduct(formData:FormData,id:number):Observable<void>
  {console.log("update1");
    return this.client.put<void>(this.host+"update/"+id,formData);
  }
  public updateProduct2(p:Produit,id:number):Observable<void>
  {console.log("update2");
    return this.client.put<void>(this.host+"update2/"+id,p);
  }


}
interface getResponse{
  _embedded:{
    produit:Produit;
  }

}
```

App-routing.module.ts

```
const routes: Routes = [
  {path:"products",component:ListeProduitComponent},
  {path:"categories/:id",component:ListeProduitComponent},
  {path:"addproduct",component:AjouterProduitComponent},
  {path:"products/:id",component:DetailsComponent},
  {path:"updateproducts/:id",component:UpdateComponent},
  {path:"",redirectTo:"/products",pathMatch:'full'}

];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

App.Component.html (bootstrap 3)

```
<nav class="navbar navbar-inverse">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-
brand" href="#">WebSiteName</a>
    </div>
    <ul class="nav navbar-nav">
      <li class="active"><a href="#">Home</a></
li>
      <li><a routerLink="/">liste Produits</a><
/li>
      <li><a routerLink="/addproduct">ajouter P
roduit</a></li>


    </ul>
  </div>
</nav>
```

```
<div class=container-fluid>
  <div class=row>
        <div class=col-lg-3>
          <app-liste-categories></app-liste-
categories>
        </div>
        <div class=col-lg-9>
           <router-outlet></router-outlet>
        </div>
  </div>
</div>
```

Liste des categories

Liste-categories.componenet.ts

```
import { Component, OnInit } from '@angular/core';
import { Categorie } from '../categorie';
import { CategorieService } from '../services/categorie.service';

@Component({
  selector: 'app-liste-categories',
  templateUrl: './liste-categories.component.html',
  styleUrls: ['./liste-categories.component.css']
})
export class ListeCategoriesComponent implements OnInit {
  constructor(private service:CategorieService) { }
  categories:Categorie[];
  getAllCategories()
  {
    this.service.getAllCategories().subscribe(
      data=>this.categories=data)
  }
  ngOnInit() {
    this.getAllCategories()
  } }
```

Liste-categories.componenet.html

```
<div class="card" style="width: 18rem;">
      <ul class="list-group list-group-flush" >
       <li *ngFor="let c of categories" class="list-group-
item" > <a [routerLink]="['/categories',c.id]" >{{c.nom}}</a></li>

      </ul>
    </div>
```

Liste des produits(Liste-produit.componenet.ts)

```
export class ListeProduitComponent implements OnInit {
 produits:Produit[]
 tab_filtre:Produit[];
  set ff(a:string)
  { this.tab_filtre =this.filtrer(a);}
  filtrer(x:string)
  {return this.produits.filter(elt=>elt.nom.indexOf(x)!=-1);}
  constructor(private serviceproduit:ProduitService,
    private ar:ActivatedRoute,private router:Router) { }
  getAllProduct()
  {  this.serviceproduit.getAllProducts().subscribe(
      data=>{console.log(data);
      this.produits=data;
      this.tab_filtre=this.produits;}) }
  getProductByCat()
  {let idcat=this.ar.snapshot.params.id;
    console.log("idcat="+idcat)
    this.serviceproduit.getProductsByCategorie(idcat).subscribe(

      data=>{console.log("data"+data);
      this.produits=data;
      this.tab_filtre=this.produits;}
    )
  }
```

```
  ngOnInit() {
     this.ar.paramMap.subscribe(()=>
       {let hasId=this.ar.snapshot.paramMap.has('id');
          if(hasId==true)
           {this.getProductByCat();}
          else
          {this.getAllProduct();} })
      }

   delete(id:number)
   {let valid=confirm("voulez vous supprimer ce produit?");
   if(valid==true){
     this.serviceproduit.supprimer(id).subscribe(

       data=>{this.getAllProduct(); } ) }
   }
}
```

Liste des produits(Liste-produit.componenet.ts)

```
<div class=container-fluid>
<div class=row>
      <div class=col-lg-3>
<app-liste-categories></app-liste-categories>
</div>

<div class=col-lg-9>
      <div class=card style="width:500px">


              <div class="card-body">
                    <div class=row>
                          <label class="col-md-2">Mot cle:</label>
                          <input type="text" [(ngModel)]=ff placeholder=
"taper le nom du produit">
                    </div>
              </div>
            </div>
            <br>
```

```
button{
      margin: 10px;
    }
```

```
<div *ngIf="produits">
<div class=row>
      <div class=col-lg-4 *ngFor="let p of tab_filtre">
            <div class="card" >
                  <img class="card-img-
top" src="http://localhost:8080/api/images/{{p.id}}" width=200 height=150>
                <div class="card-body">
                  <h5 class="card-title">{{p.nom}}:{{p.prix}} DT </h5>
                  <p class="card-
text" *ngIf="p.quantite>0;else non"><span style="color:green">en stock</span></p>


                  <a routerLink="/products/{{p.id}}" class="btn btn-
primary">details</a>
                  <button class="btn btn-
danger" (click)="delete(p.id)"><span class="fa fa-trash"> </span></button>
                  <a routerLink="/updateproducts/{{p.id}}" class="btn btn-
warning" ><span class="fa fa-edit"> </span></a>


                </div>
              </div>
      </div>
</div>
 <ng-template #non ><span style="color:red">hors stock</span></ng-template>

</div>
</div>
</div>
</div>
```

Ajouter Produit (.ts)

```
export class AjouterComponent implements OnInit {
  produitGroupe:FormProduit;
  constructor(private fb:FormBuilder,
    private service:ProduitService,
    private router:Router) { }
public categories:Categorie[];

  ngOnInit() {
    this.service.getAllCategories().subscribe(
     data=>{this.categories=data
           console.log(this.categories)
     }
    )
    this.produitProduit=this.fb.group(
     {nom:['',[Validators.required,Validators.minLength(5)]],
     prix:[0,[Validators.required,Validators.min(5),Validators.max(10000)]
],
     quantite:[0,[Validators.required,Validators.min(1),Validators.max(100
0)]],
     categorie:['',Validators.required],
     photo:['',Validators.required]})
   }
```

```
file:File;
imgUrl:any
selectImage(event)
{
  this.file=event.target.files[0];
  let f=new FileReader();
  f.readAsDataURL(this.file);
  f.onload=(event)=>this.imgUrl=f.result

}
save()
  {
    let fd:FormData=new FormData();
    fd.append("produit",JSON.stringify(this.produitProduit.value))
    fd.append("file",this.file)
    this.service.addProduct(fd).subscribe(
      ()=>{alert("produit ajouter avec succes");
           this.router.navigateByUrl("/products");
          })
  }
}
```

Ajouter Produit (.html)

```
.car{
      width:500px;
      margin:auto;
}
.st{
      font-size: 20px;
      color:coral;
}
```

```
<div class="card car">
      <div class="card-header st">Nouveau  produit</div>
<div class=card-body>
<div *ngIf="formProduit">
<form [formGroup]="formProduit" (ngSubmit)=save()>

      <div class="form-group">
            <label for="nom">Nom:</label>
            <input type="text" id="nom" class="form-control"
            placeholder="nom du produit"
            formControlName="nom"
            [ngClass]="{'is-invalid': formProduit.get('nom').touched
                  && !formProduit.get('nom').valid}" >
       </div >
       <div class="form-group">
            <label for="prix">
               prix:
            </label>
            <input type="number" id="prix" class="form-control"
            placeholder="prix du produit"
            formControlName="prix"
            [ngClass]="{'is-invalid': formProduit.get('prix').touched
                  && !formProduit.get('prix').valid}"
            >
</div >
      <div class="form-group">
            <label for="quantite"> Quantite:</label>
            <input type="number" id="quantite" class="form-control"
            placeholder="quantite du produit"
            formControlName="quantite"
            [ngClass]="{'is-invalid': formProduit.get('quantite').touched
                  && !formProduit.get('quantite').valid}" >
      </div >
      <div>
            <label >Choisir une categorie:</label>
               <select formControlName=categorie class="form-control form-
control-lg" >
   <option *ngFor="let c of categories" value={{c.id}} >{{c.nom}}</option>
               </select>

      </div> <br><div>
 <input  type=file formControlName="photo"(change)=selectImge($event) >
      </div>
<div *ngIf="imgUrl"><img src={{imgUrl}} width=50 height=50></div>
      <br>
      <div>
      <button class="btn btn-success" [disabled]=!formProduit.valid>
            Ajouter
     </button></div></form></div></div></div>
```

Update Produit (.ts)

```
export class UpdateComponent implements OnInit {
  formProduit:FormGroup;
categories:Categorie[];
  constructor(private service:ProduitService,private build:FormBuilder,
    private servcat:CategorieService,private router:Router,
    private ar:ActivatedRoute) { }
getCategories()
{  this.servcat.getAllCategories().subscribe(
   data=> this.categories=data )}
id:number
  ngOnInit() {
    this.id=this.ar.snapshot.params.id;
    this.getCategories();
    this.service.getProductById(this.id).subscribe(
      data=>{this.produit=data;
      this.formProduit=this.build.group(
        {nom:[this.produit.nom,[Validators.required,Validators.minLength(5)]],
        prix:[this.produit.prix,[Validators.required,Validators.min(5)]],
        quantite:[this.produit.quantite,[Validators.required,Validators.min(0)]],
        categorie:[this.produit.categorie],
        photo:['']  } );  } )  }
```

```
save()
  {  if(this.file){
    let formdata:FormData=new FormData();
    formdata.append('produit',JSON.stringify(this.formProduit.value));
    formdata.append('file',this.file);
    this.service.updateProduct(
      formdata,this.id ).subscribe(
        ()=>{alert(« le produit a été modifié");
          this.router.navigate(['/products']);})}
      else{ let p:Produit=this.formProduit.value;
          p.photo=this.produit.photo;
      this.service.updateProduct2(
        p,this.id ).subscribe(
          ()=>{this.router.navigateByUrl("/products");}) }}
  file:File;  imgUrl:any
  selectImge(event){
    this.file=event.target.files[0];
    let fr=new FileReader();
    fr.readAsDataURL(this.file)
    fr.onload=(event)=>this.imgUrl=fr.result}
produit:Produit;
  compareCat(c:Categorie,cc:Categorie):boolean
{  return c && cc?c.id===cc.id:c===cc}}
```

Update Produit (.html)

```
<div *ngIf="produit">
<div class="card car">
      <div class="card-header st">Modifier le produit</div>
<div class=card-body>
<form [formGroup]="formProduit" (ngSubmit)=save()>
      <div class="form-group">
            <label for="nom">Nom:</label>
            <input type="text" id="nom" class="form-control"
            placeholder="nom du produit"
            formControlName="nom"
            [ngClass]="{'is-invalid': formProduit.get('nom').touched
                  && !formProduit.get('nom').valid}" > </div >
       <div class="form-group">
            <label for="prix">prix:</label>
            <input type="number" id="prix" class="form-control"
            placeholder="prix du produit"
            formControlName="prix"
            [ngClass]="{'is-invalid': formProduit.get('prix').touched
                  && !formProduit.get('prix').valid}" ></div >
       <div class="form-group">
            <label for="quantite"> Quantite: </label>
            <input type="number" id="quantite" class="form-control"
            placeholder="quantite du produit"
            formControlName="quantite"
            [ngClass]="{'is-invalid': formProduit.get('quantite').touched
```

```
<div> <label >Choisir une categorie:</label>
      <select formControlName=categorie [compareWith]="compareCat"
class="form-control form-control-lg"  >
      <option [ngValue]="null" disabled >select categorie</option>
      <option *ngFor="let c of categories" [ngValue]="c">{{c.nom}}
</option> </select></div>
      <br>
 <div *ngIf="imgUrl;else autre">
 <img  src={{imgUrl}} width=50 height=50>
     </div>
     <ng-template #autre>
        <img src="http://localhost:8080/api/images/{{produit.id}}"
 width=50 height=50>
     </ng-template>
    <div>
  <input  type=file formControlName="photo" accept="image/*"
           (change)=selectImge($event) > </div>
      <br>
      <div>
      <button class="btn btn-
success" [disabled]=!formProduit.valid>
            valider
     </button></div>
</form>
</div></div></div>
```