

Angular permet de créer la partie front end des applications web de type SPA basé sur les composants web

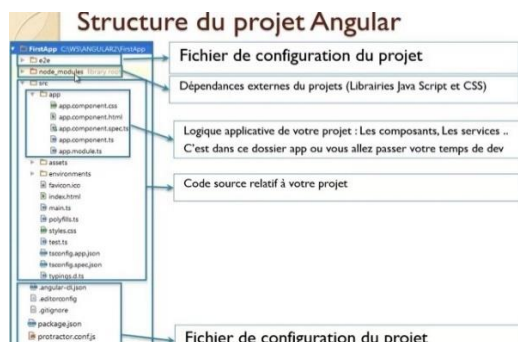
Définition du SPA : Simple Page Application (toute application angular contient une seule page index.html, pas de rechargement dans les pages routées), mise en pratique dans la formation 1 au niveau des routerlink

## Préparation de l'environnement de développement

Cli : command line interface (ng new, ng serve ...) permet de créer, compiler, tester et le déployer un projet angular : installation de @angular/cli contient la logique ng « mot » selon ce que nous voulons faire. npm : node package manager permet de télécharger ou d'inclure des dépendances dans le projet

```
package.json > {} dependencies
{
  "name": "app-coll-etiq",
  "version": "0.0.0",
  "scripts": {
    "start": "ng serve",
    "build": "ng build",
    "watch": "ng build --watch --configuration development",
    "test": "ng test"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^16.1.0",
    "@angular/cdk": "^16.2.3",
    "@angular/common": "^16.1.0",
    "@angular/compiler": "^16.1.0",
    "@angular/core": "^16.1.0",
    "@angular/forms": "^16.1.0",
    "@angular/material": "^16.2.3",
    "@angular/platform-browser": "^16.1.0",
    "@angular/platform-browser-dynamic": "^16.1.0",
    "@angular/router": "^16.1.0",
    "@fortawesome/fontawesome-free": "^6.4.2",
    "bootstrap": "^5.3.1",
    "rxjs": "^7.8.0",
    "tslib": "^2.3.0",
    "zone.js": "^0.13.0"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "^16.1.1",
    "@angular/cli": "^16.1.1",
    "@angular/compiler-cli": "^16.1.0",
    "@types/jasmine": "~4.3.0",
    "jasmine-core": "~4.6.0",
    "karma": "~6.4.0",
    "karma-chrome-launcher": "~3.2.0",
    "karma-coverage": "~2.2.0",
    "karma-jasmine": "~5.1.0",
    "karma-jasmine-html-reporter": "~2.1.0",
    "typescript": "~5.1.3"
  }
}
```

## Création du projet : ng new nouveau\_projet



Intallation de style bootstrap dans nouveau\_projet : npm install bootstrap@version --save: Flag(save) permet de l'ajouter au fichier json du projet. Dans angular.json : "node\_modules/bootstrap/dist/css/bootstrap.css",

## Création d'un composant : ng g c nom\_composant

Décorateur :

**Component** importer depuis **@angular/core**, **@component** contient trois attributs, le sélecteur est une balise unique liée à un component, l'identifiant de la logique du component

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
```

**NgModule** importer depuis **@angular/core**, **@NgModule** contient 04 attributs : la déclaration : toute composant d'un module doit être déclarer dans le module du composant, Imports : se sont des librairies connues dans angular qu'on import pour un cas d'usage, providers : les services de chaque composant seront déclarés dans ce décorateur, Bootstrap : indique la logique qui doit être afficher

```
app > ts app.module.ts > ...
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { BrowserModule } from '@angular/platform-browser';
import { AppareilComponent } from './dossier/appareil.component';
import { FormsModule } from '@angular/forms';

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

la balise **appareil-root** geree dans **App.component.html** affiche la logique **AppareilComponent**

```
<div class=container>
  <h2>Mes appareils</h2>
  <ul class=list-group>
    <appareil-root></appareil-root>
    <appareil-root></appareil-root>
    <appareil-root></appareil-root>
    <appareil-root></appareil-root>
  </ul>
</div>
```

la balise **app-root** affiche la logique **AppComponent** au navigateur

```
<body class="mat-typography">
  <app-root></app-root>
</body>
```

Le module a été créé dans le dossier **app** qui contient **AppModule**, le module **AuthModule** est un sous module. Création de module : **ng g m nom\_module**

```
app-routing.module.ts M X
app-coll-etiq > src > app > ts app-routing.module.ts > ...
1 | import { NgModule } from '@angular/core';
2 | import { RouterModule, Routes } from '@angular/router';
3 | import { AuthComponent } from './auth/auth.component';
4 | import { PresentationComponent } from './auth/presentation/presentation.component';
5 | import { LoginComponent } from './auth/login/login.component';
6 | import { SignUpComponent } from './auth/sign-up/sign-up.component';
7 | import { AccueilComponent } from './auth/accueil/accueil.component';
8 |
9 |
10 | const routes: Routes = [
11 |   {path: '', component: PresentationComponent},
12 |   {path: 'auth', component: AuthComponent, children: [
13 |     {path: 'presentation', component: PresentationComponent},
14 |     {path: 'login', component: LoginComponent},
15 |     {path: 'sign', component: SignUpComponent},
16 |     {path: 'accueil', component: AccueilComponent},
17 |   ]},
18 | ],
19 |
20 | @NgModule({
21 |   imports: [RouterModule.forRoot(routes)],
22 |   exports: [RouterModule]
23 | })
24 | export class AppRoutingModule { }
```

**AppRoutingModule** qui gère le chemin de l'ensemble des modules, doit être importer

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { RouterModule } from '@angular/router';
import { AuthModule } from './auth/auth.module';
import { EtiquetageModule } from './etiquetage/etiquetage.module';
import { ProjectModule } from './project/project.module';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';
import { HttpClientModule } from '@angular/common/http';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatTableModule } from '@angular/material/table';
import { MatSortModule } from '@angular/material/sort';
import { DataSetsModule } from './datasets/datasets.module';
import { DataModule } from './data/data.module';

@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule, RouterModule, AuthModule, EtiquetageModule, ProjectModule, FormsModule, CommonModule, HttpClientModule, ReactiveFormsModule, BrowserAnimationsModule,
    MatTableModule, MatSortModule, DataSetsModule, DataModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

Nous affichons AppModule dans **main.ts** comme le module qui va gerer les autres modules

```

TS main.ts > ...
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

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

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));

```

**Liaison de données** : permet la communication des informations de TS vers le Template

## Code bootstrap :

```
<div class=container>
<ul class="list-group">
  <li class="list-group-item" >bootstrap 1</li>
  <li class='list-group-item'> bootstrap 2</li>
</ul>
</div>
```



## String interpolation :

Appareil.component.ts

```
appareilName='machine a laver';
appareilStatut='eteint';
```

Appareil.component.html

```
<li class='list-group-item'>
  appareil:{{appareilName}} --- statut: {{appareilStatut}}
</li>
```

App.component.html

```
<div class="container">
  <h2> Mes appareils :</h2>
  <ul class="list-group">
    <appareil-root></appareil-root>
    <appareil-root></appareil-root>
  </ul>
</div>
```

## Objet

```
appareil={
  appareilName:'machine a laver',
  appareilStatut:'eteint'
}
```

## Rendu html

### Mes appareils

appareil:machine a laver -- status:eteint
appareil:machine a laver -- status:eteint
appareil:machine a laver -- status:eteint

## Code Bootstrap :

```
<button class="btn btn-primary">tout allumer </button>
```

tout allumer

```
<button class="btn btn-success">tout allumer </button>
```

tout allumer

## Property binding : on affecte à une propriété une condition

### Mes appareils :

appareil:machine a laver ---statut:eteint
appareil:machine a laver ---statut:eteint
appareil:machine a laver ---statut:eteint
tout allumer

```
isAuth=false;
```

```
<button class="btn btn-primary"
[disabled]='!isAuth'>tout allumer </button>
```

```
constructor(){
  setTimeout(()=>{this.isAuth=true}, 5000)
}
```

appareil:machine a laver -- status:eteint

tout allumer

**Event binding** : évènement click, ....

App.component

```
<button class="btn btn-primary"
(click)="onToutAllumer()"> tout
allumer </button>
```

```
onToutAllumer(){
  console.log('tout est allumer')
}
```

## Tow way data binding

```
import {FormsModule} from '@angular/forms';
```

Appareil.component.html



```
<input [(ngModel)]='Appareils[0].appareilName'>
```

appareil:je suis ici ce matin -- status:eteint

je suis ici ce matin

appareil:machine a laver -- status:eteint

machine a laver

appareil:machine a laver -- status:eteint

machine a laver

tout allumer

## Property binding personalized

```
import {Input} from "@angular/core";
```

Appareil.component

```
@Input() val=' '
<input [(ngModel)]='val'>
```

App.component

```
appareilOne=' '
<appareil-root [val]='appareilOne' ></appareil-root>
```

## **result**

appareil:je suis ici ce matin -- status:eteint

je suis ici ce matin

appareil:machine a laver -- status:eteint

machine a laver

appareil:machine a laver -- status:eteint

machine a laver

tout allumer

## Les directives structurales

Contact.component

### Objects

```
infos=[{last_name:'NAPON', second_name:'Emile',tel:'+226 63 09 31 31',niveau_etude:'Master 2'},
|      {last_name:'KABORE',second_name:'Leticia',tel:'+226 54 - - -',niveau_etude:'3ieme'
|    }
];
```

```
<ul class='list-group' *ngFor='let j of infos'>
|   <li class='list-group-item' >{{j.last_name}} {{j.second_name}}</li>
|
| </ul>
```

NAPON Emile

KABORE Leticia

### Comments objects

```
comments=[{date:new Date(),message: ''},{date:new Date(),message: ''}];
```

```
comment=[{date:new Date(),message: 'boojour Emile'},{date:new Date(),message: 'Oui Bonjour Leticia'}]
```

```
<ul class='list-group' *ngFor='let j of infos'>
|   <div class='dispo'>
|     <li class='list-group-item' >{{j.last_name}} {{j.second_name}}</li>
|     <li class='list-group-item' id='li' *ngIf='j.last_name=="NAPON"'>{{comments[1].message}} ---{{comments[1].date}}</li>
|     <li class='list-group-item' id='li' *ngIf='j.last_name=="KABORE"'>{{comments[0].message}} ---{{comments[0].date}}</li>
|   </div>
```

NAPON Emile	Oui Bonjour Leticia ---Thu Mar 07 2024 14:38:08 GMT+0000 (heure moyenne de Greenwich)
KABORE Leticia	boojour Emile ---Thu Mar 07 2024 14:38:08 GMT+0000 (heure moyenne de Greenwich)

## Directives par attribut : [ngStyle]


## Vue

appareil:Machine a laver --- statut: allume

---

appareil:Television --- statut: allume

---



appareil:Ordinateur --- statut: eteint

---

tout allumer

```
appareils=[{
  appareilName:'Machine a laver',
  appareilStatut:'eteint'
},{
  appareilName:'Television',
  appareilStatut:'allume'
},{
  appareilName:'Ordinateur',
  appareilStatut:'eteint'
}]

getColor(v:string){
  if(v=='eteint'){return 'red';} else if( v=="allume"){return "green"} else{return ''}}
}
```

*dom*

```
<li class='list-group-item' *ngFor='let appareil of appareils'>
<div>
<p style='width:20px; height:20px; background-color:red;' *ngIf="appareil.appareilStatut === 'eteint'" ></p>
<p [ngStyle]='{color:getColor(appareil.appareilStatut)}'>appareil:{{appareil.appareilName}}
  --- statut: {{appareil.appareilStatut}}</p>
</div>
<p><input [(ngModel)]='appareil.appareilName'> </p>
</li>
```

## Directives par attribut : [ngClass]

```
<li *ngFor='let appareil of appareils'

[ngClass]="{
  'list-group-item':true,
  'list-group-item-danger':appareil.appareilStatut=='eteint',
  'list-group-item-success':appareil.appareilStatut=='allume'
}" > |


<p style='width:20px; height:20px; background-color:red;' *ngIf="appareil.appareilStatut === 'eteint'" ></p>
<p [ngStyle]='{color:getColor(appareil.appareilStatut)}'>appareil:{{appareil.appareilName}}
  --- statut: {{appareil.appareilStatut}}</p>
<input [(ngModel)]='appareil.appareilName'>
</li>
```

appareil:Machine a laver --- statut: allume

---

appareil:Television --- statut: allume

---



appareil:Ordinateur --- statut: eteint

---

tout allumer

## Promesses, pipes

Date Pipe



`<span [ngStyle]='{color:"red" }'>{{comments[1].date | date}}</span> --->`

NAPON Emile	---Thu Mar 07 2024 14:38:08 GMT+0000 (heure moyenne de Greenwich)	Oui Bonjour Leticia --- Mar 14, 2024
KABORE Leticia	---Thu Mar 07 2024 14:38:08 GMT+0000 (heure moyenne de Greenwich)	boojour Emile--- Mar 14, 2024

Pipe async

```
lastUpdate = new Promise((resolve, reject) => {
  const date = new Date();
  setTimeout(() => { resolve(date); }, 4000);
});
```

## Personalized Pipe

```
@NgModule({
  declarations: [
    AppComponent,
    HotelListComponent,
    ReplaceComma
  ],
```

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'removeComma'
})
export class RemoveCommaPipe implements PipeTransform {

  transform(value: string): string {
    // logique du pipe
  }
}
```

Importer l'interface

Ajouter le décorateur

Implémenter l'interface

Implémenter la méthode de l'interface

```
import { Pipe, PipeTransform } from '@angular/core';
@Pipe({
  name: 'replacecomma'
})
export class replaceCommaPipe implements PipeTransform{

  transform(value: any):string {
    if(!value){
      return value.replace(/,/g, ".");
    }else{
      return '';
    }
  }
}
```

Remplace d'une virgule par un point, dans l'affichage du prix dans une application de e-commerce

List of pipes: ngx-pipes sur github

## Services



Création du service et Injection du service

```
providers: [AppareilService, CouleurAppareilComponent]
```

Logique du service

```
export class AppareilService{
  contact=[{appareilName: 'machine a laver',appareilStatut:'eteint'},
    {appareilName : 'Television',appareilStatut:'eteint'},
    {appareilName: 'telephone',appareilStatut: 'allume'}]
  AllumerAll(): void{
    for(let i of this.contact){
      if(i.appareilStatut=='eteint'){
        i.appareilStatut='allume'; }}
  }
}
```

Injection dans le component et implémentation du service

```
export class AppComponent implements OnInit {

  constructor(private service: AppareilService){
```

L'appel des éléments du service

Déclarer un tableau vide dans le composant et utiliser la méthode ngOnInit pour dire que les données dans le service sous forme de tableau correspondant au tableau déclaré dans le component.

Objets

```
appareils:any[]=[]
```

```
ngOnInit(): void {
  this.appareils=this.service.contact;
}
```

Dom

Méthodes

```
ngToutAllumer(){
  this.service.AllumerAll();
}
ngToutEteindre(){
  this.service.ToutEteindre() ;
}
```

```
<div >
  <button class= 'btn btn-primary' [disabled]='!isAuth' (click)='ngToutAllumer()'>tout allumer</button>
  <button class= 'btn btn-danger' [disabled]='!isAuth' (click)='ngToutEteindre()'>tout eteindre</button>
</div>
```

**NB :** Un component : contient un ou plusieurs services, Les objets et les méthodes de chaque service sont escortes dans le service.

Implémenter une méthode qui va permettre d'éteindre tous les appareils et une autre méthode qui va permettre d'allume tous les appareils d'un coup.

Les méthodes interpolées dans le dom sont les méthodes qui implémentent les méthodes du service dans le component



L'indice let i=index dans le service

La navigation