# Angular

*Université de Montpellier*

**Abstract.**

## 1 Introduction

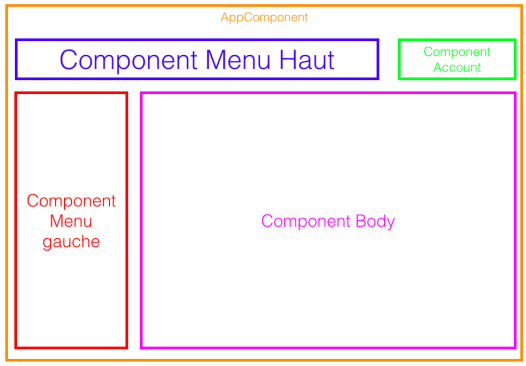
Angular est un framework Javascript côté client qui permet de réaliser des applications de type ”Single Page Application”. Il est basé sur le concept de l’architecture MVC (Model View Controller) qui permet de séparer les données, les vues et les différentes actions que l’on peut effectuer.

## 2 Installation

Avant de créer un projet, il faut installer Angular CLI : npm install -g @angular/cli
Pour creer un nouveau projet : ng new monProjet
Pour lancer le serveur de développement : ng serve –open
Pour inclure bootstrap au projet Angular : npm install bootstrap –save
Ensuite, il faut inclure dans le tableau ”style” du fichier angular.json la ligne :
”../node_modules/bootstrap/dist/css/bootstrap.css”

## 3 La structure d’une page avec Angular

Une page web créée avec Angular se divise en plusieurs component (composant) qui peuvent s’encapsuler les uns dans les autres :




Chacun de ces composants correspond à un dossier dans le dossier source de notre espace de travail. Ce dossier peut être créé avec la commande : ng g monComposant.
Dans chaque dossier de composant, on à trois fichier :

- monComposant.component.html : contient le code html
- monComposant.component.css : contient une feuille de style css monComposant.component.ts : contient notamment le nom du selecteur (balise) correspondant au component ainsi que la déclaration de la classe permettant de créer le component. C'est ici que nous pourrons écrire le code typeScript (fonctions, objets, variables).

Le composant de base de l'application est app et son sélécteur app-root est appelé dans le fichier index.html.

## 4 Interaction entre le code TypeScript et le Html

### 4.1 typescript -> html

Il est possible d'exploiter du code du fichier .ts dans la partie Html.
On déclare un attribut dans la classe. Exemple : maVariable: number = 4
On peut déclarer une méthode. Exemple : maMethode() return this.maVariable
Pour appeler un attribut ou une méthode dans le code html, il faut l'écrire entre . Exemple:  maVariable 
On peut associer la valeur d'un attribut de balise à une variable. Il faut mettre l'attribut entre []. Exemple <button [disabled]="monBool">

### 4.2 html -> typescript et écouteur d'évènement

Pour exploiter dans le code typescript les évènements provenant des interactions de l'utiliateur avec la page web, on utilise les ecouteurs d'évènements.
Par exemple, si on veut appeler une méthode lorsque l'utilisateur clique sur un bouton.

On créé la maMethode() dans la classe puis on associe l'évènement click du boutton à maMethode: <button (click)=”maMethode()”>

### 4.3 creation d'attribut personnalisé

Il est possible de créer des attributs personnalisés pour nos composants.
Dans la classe, il faut préfixer la declaration de l'attribut par @Input()
Exemple :
dans la classe de monComposant: @Input nom:string
dans le code html appelant monComposant:
<app-monComposant [nom]=”’toto”’></app-monComposant>

## 5 structure de controle

### 5.1 boucler avec *ngFor

On peut utiliser la directive *ngFor pour exploiter une collection déclarée dans la classe de notre composant. On créera donc autant d'élément qu'il y à d'éléments dans notre collection.
Exemple:
Dans la déclaration de notre classe myComponent, on créé un tableau.
myArray=[
nom:’jean’, mdp:’mdp’ ,
nom:’jacques’, mdp:’pdm’ ]
On peut créer une liste avec autant d'éléments qu'il y a d'objet dans le tableau.
<li *ngFor=”let item of myArray”> item.nom </li>
On obtient donc une liste avec les attributs noms de chaque objet.

### 5.2 *ngIf=”condition”

On utilise *ngIf de la même manière que *ngFor. L'élément sera affiché seulement si la condition est remplie.

## 6 pipe

Les pipes prennent une donnés en entrée et l'affiche d'une certaine manière en sortie souvent plus lisible. Par exemple si on à un attribut maDate de type Date dans notre classe et qu'on souhaite l'afficher de anière élégante, on utilise:
date: maDate | date 
On peut paramétrer les pipes ou même en créer.
Pour plus d'info : https://angular.io/guide/pipes

## 7 Centraliser les données

On utilise les ”services” pour mettre en commun des données ou du code entre plusieurs composants. C’est un fichier typescript que l’on peut générer par la commande :
ng g service monService
Pour utiliser les données de ce service, il faut l’importer en haut du fichier app.module.ts:
import monService from ’./monService.service’
Puis ajouter dans le tableau des providers de ce même fichier : monService
Pour qu’il soit utilsables par un composant, il faut

- passer le service en paramètre du constructeur:
  constructor(private proprieteService : monService)...
- implementer l’interface ngOnInit : export class ... implements OnInit
- implementer la méthode ngOnInit()this.myArray=proprieteService.myArray

## 8 Le routage

Gros avantage d’Angular : charger une seule page et modifiez les components côté client pour modifier l’affichage plutôt que de charger une page à chaque fois. On utilise donc le routage pour naviguer entre nos différentes vues.
La première étape est d’importer 2 moodules dans le tableau Imports de app.module.td : AppRoutingModule et RouterModule.forRoot(chemins)
chemin est un tableau d’objet de la forme path:”,component:monComponent
On associe en fait un composant (donc une page) à un chemin.
La seconde étape est de préciser où est ce que l’on veut afficher les composants concernés: c’est le rôle de la balise <router-outlet>// Dernière étape, il faut qu’on définisse les manières de naviguer entre les vues. Pour se faire, on utilise des ancres <a> avec une propriété [router-link]=”[’path’]”. Lorsque l’on va cliquer sur l’ancre, le composant correspondant au chemin s’affichera à l’endroit où est placé la balise <router-outlet> est placée.

Pour aller plus loin : https://angular.io/guide/router