Cours 3 : Angular

Mardi 17 Novembre 2020

Antoine Flotte (aflotte@excilys.com)

Excilys
Développeurs de passion

SOMMAIRE

AngularJS = Angular 1.x

Angular = Angular 2+

Angular

Configuration

Configuration

Windows

- Télécharger nodeJs avec le lien ci-dessous
- https://nodejs.org/en/download/
- `npm install -g @angular/cli`

- ng new ProjectName
- ng serve / npm start

Configuration

Ubuntu

```
apt update
apt install nodejs npm
npm install -g @angular/cli
ng new ProjectName
ng serve / npm start
```

Code .ts

Angular utilise le TypeScript. C’est un langage semblable au Java.

app.module.ts

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


import { AppComponent } from './app.component';
import { MonComponent } from './mon-premier/mon-premier.component';


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

ts - lien avec le html

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

@Component({
  selector: 'app-appareil',
  templateUrl: './appareil.component.html',
  styleUrls: ['./appareil.component.scss']
})
export class AppareilComponent implements OnInit {

  appareilName: string = 'Machine à laver';

  constructor() { }

  ngOnInit() {
  }

}
```

html - lien avec le js

```
<li class="list-group-item">
  <h4>Appareil : {{ appareilName }}</h4>
</li>
```

Two Way Binding

Dans votre class app.module.ts ajouter l'import :

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

et dans la partie imports ajouter :

```
FormsModule
```

Puis vous pourrez utiliser dans votre html :

```
<input type="text" class="form-control"
[(ngModel)]="attribut">
```

Directive ngIf

```
<div *ngIf="condition"></div>
```

Attention, on fait les conditions avec ===

Directive ngFor

```
<app-appareil  *ngFor="let element of list"
                    [appareilName]="element.name"

[appareilStatus]="appareil.status"></app-appareil>
```

Angular

Projet

ng new frontFilmList (ou le nom que vous voulez)

Y pour le routing et choisir CSS

Puis dans src/app/app-component.html tout supprimer sauf la dernière ligne :

<router-outlet></router-outlet>

ng generate component films

Dans le app.module.ts vous pouvez voir que le FilmsComponent a été ajouté dans les declarations.

Un package vous a été créé avec 4 fichiers.

Rajoutez y votre modèle film.model.ts

Film

```
export class Film {
    id: number;
    titre: String;
    duration: number;
}
```

Et l'importer dans film.component.ts

```
import { Film } from './film.model';
```

Dans app-rooting.module.ts importer FilmComponent et rajouter dans la constante routes :

```
{ path: 'film', component: FilmsComponent }
```

Vous pouvez maintenant lancer le serveur et acceder à localhost:4200/film et voir un films works!

Appel au back

Pour appeler le back on utilise HttpsClient :

```
import { HttpClient } from '@angular/common/http';
```

On va pouvoir le mettre dans les imports du app.module.ts et l'importer aussi dans notre composant.

Ensuite dans le FilmComponent on crée une sous fonction :

```
 getFilms(): Observable<Film[]> {
    return
this.http.get<Film[]>('http://localhost:8080/film');
  }
```

Appel au back

Pour que le code fonctionne, on a besoin de modifier le constructeur :

```
constructor(private http: HttpClient) { }
```

c'est de l'injection de dépendance comme on l'a déjà vue dans le back

et on a un attribu :

```
 films : Film[];
```

Et enfin dans le OnInit :

```
this.getFilms().subscribe(
      data => {this.films = data;}
)
```

C'est un code inédit qui fonctionne avec un Observable, comme c'est asynchrone on met à jour la liste de film quand on la reçoit

Enfin dans le html :

```
<ul>
    <li *ngFor="let film of films">{{ film.titre }}</li>
</ul>
```

Cross-Origin Back

```
@CrossOrigin(origins = "http://localhost:4200")
```

Ajoutez la ligne ci-dessus en haut des controllers du back pour reglez vos problème de CORS.

Pour passer un id dans le routid :

/:id (ou le nom que vous voulez pour l'attribut)

```
 id : number;

 constructor(private route: ActivatedRoute) { }

 ngOnInit(): void {
   this.route.params.subscribe(params => {
     this.id = +params['id'];
     console.warn(params['id']);
   });
 }
```

Balise Redirection

```
[routerLink]="['/film', film.id]"
```

À ajouter à la balise qui doit faire la redirection.

Dernier conseil

Dans le html, ne repartez pas de zero, utilisez Angular Material, par exemple les list, ou même les table

Faite ce qui vous semble le plus logique, vous êtes libre pour le fonctionnement du site, adaptez votre back à vos choix front !

Et le Swagger est un bon outils pour savoir les appels à faire

console.warn(“blabla”); // info,error,debug

popup : MatDialog

```
let params = new HttpParams()
      .set('page', page
      .set('nbParPage', 10);


return this.http.get<Film[]>(`localhost:8080/film`,
{params});
```

Events

```
<button (click)="send()">Send</button>
```

Il existe d’autres event Angular, celui-ci étant indispensable