

TP de bases pour Angular 10

TABLE DES MATIERES

I.	Installation des outils pour réaliser le TP :	Erreur ! Signet non défini.
II.	Création d'un projet avec Angular CLI.....	3
III.	Création des modules et composants des pages	4
A.	Chacun des 3 modules représentent une page de notre application	5
1.	Ajouter les composants qui serviront à afficher chacune des pages :	6
IV.	Mise en place des Routes	8
A.	Dans le module principal, on a déjà le chargement des routes principales	8
1.	– Ajouter les routes aux modules enfants.....	9
2.	Indiquer le point où se chargera les templates des composants des pages	11
3.	Tester les routes :	11
V.	Ajout de Bootstrap	14
VI.	Ajout d'un menu.....	15
1.	Ajouter le composant au module shared.module.ts	16
2.	Vérifier que le composant navmenu s'affiche :	17
3.	Importer RouterModule à shared.module.ts	17
4.	Modifier le template navmenu.component.html pour y ajouter un menu avec les liens vers les différentes pages.....	18
VII.	Ajout d'un formulaire	19
A.	Créer un nouveau composant : ng g component site/films/searchform	19
1.	Ajouter le formulaire au template : films.component.html	19
B.	Créer un formulaire dans searchform.component.html	20
C.	Ajouter au module : films.module.ts	20
D.	Ajouter à searchform.component.ts :	21
VIII.	Création d'un service émettant une requête http.....	25
1.	Injecter le service à searchform.component.ts	27
2.	Tester si la requête se fait bien en lançant une recherche.....	29
3.	Fournir les valeurs entrées par l'utilisateur dans le formulaire au service. Utiliser pour cela l'attribut searchForm: FormGroup	30
IX.	Ajouter des validateurs à un formulaire	30
X.	Création d'une directive pour l'affichage des résultats	33
A.	Créer une directive : ng g directive site/films/directives/list.....	33
B.	Retoucher searchform.component.ts :	33
1.	Renommer le sélecteur de la directive en movieList (attention à la casse).....	34
2.	Appeler la directive depuis le template searchform.component.html	35

Formation Angular 10

3.	Modifier la directive pour qu'elle affiche les données	36
4.	Tester le résultat	37

I. Création d'un projet avec Angular CLI

1. Créer un dossier Angular10 dans votre dossier utilisateur.
2. Se placer grâce à la console de Windows dans ce répertoire
3. Taper dans la console : `ng new tpbinding` :

```
C:\Users\GERMAIN\angular7>ng new exercice
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
SCSS [ http://sass-lang.com/documentation/file.SASS_REFERENCE.html#syntax ]
Sass  [ http://sass-lang.com/documentation/file.INDENTED_SYNTAX.html       ]
Less  [ http://lesscss.org                                                  ]
Stylus [ http://stylus-lang.com                                              ]
```

Taper Entrée

```
CREATE exercice/README.md (1025 bytes)
CREATE exercice/tsconfig.json (470 bytes)
CREATE exercice/tslint.json (1985 bytes)
CREATE exercice/.editorconfig (246 bytes)
CREATE exercice/.gitignore (629 bytes)
CREATE exercice/browserslist (429 bytes)
CREATE exercice/karma.conf.js (1020 bytes)
CREATE exercice/tsconfig.app.json (210 bytes)
CREATE exercice/tsconfig.spec.json (270 bytes)
CREATE exercice/src/favicon.ico (5430 bytes)
CREATE exercice/src/index.html (295 bytes)
CREATE exercice/src/main.ts (372 bytes)
CREATE exercice/src/polyfills.ts (2838 bytes)
CREATE exercice/src/styles.css (80 bytes)
CREATE exercice/src/test.ts (642 bytes)
CREATE exercice/src/assets/.gitkeep (0 bytes)
CREATE exercice/src/environments/environment.prod.ts (51 bytes)
CREATE exercice/src/environments/environment.ts (662 bytes)
CREATE exercice/src/app/app-routing.module.ts (245 bytes)
CREATE exercice/src/app/app.module.ts (393 bytes)
CREATE exercice/src/app/app.component.html (1152 bytes)
CREATE exercice/src/app/app.component.spec.ts (1101 bytes)
CREATE exercice/src/app/app.component.ts (212 bytes)
CREATE exercice/src/app/app.component.css (0 bytes)
CREATE exercice/e2e/protractor.conf.js (810 bytes)
CREATE exercice/e2e/tsconfig.json (214 bytes)
CREATE exercice/e2e/src/app.e2e-spec.ts (637 bytes)
CREATE exercice/e2e/src/app.po.ts (251 bytes)
[.....] - fetchMetadata: sill pacote range manifest for @types/jasminewd2@~2.0.3 fetched in 3129ms
```

- Ceci va créer un nouveau projet Angular
 - Selon l'ordinateur, cela peut prendre du temps.
4. Toujours dans la **console**, se placer dans le répertoire `tpbinding` (`cd tpbinding`) et taper :

ng serve -o ou **modifier le fichier package.json pour lancer automatiquement cette commande par : npm start**

- Cela permet d'exécuter un serveur http qui va pouvoir fournir l'application et de lancer le navigateur par défaut à l'adresse :

<http://localhost:4200>

Welcome to jour01!



Here are some links to help you start:

- [Tour of Heroes](#)
- [CLI Documentation](#)
- [Angular blog](#)

II. Création des modules et composants des pages

Suivez bien les instructions pour éviter tout soucis de chemin physique et de quiproquos dans les noms de fichiers.

1. Créer un **répertoire site** dans **src/app**
2. Avec la **console**, se placer à la racine du projet et taper :

ng g module site/home

ng g module site/about

ng g module site/films

A. Chacun des 3 modules représentent une page de notre application

- Ajouter ces 3 modules dans le module principal qui se trouve dans le répertoire **app** et se nomme **app.module.ts** :

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModuleModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { AboutModule } from './site/about/about.module';
import { FilmsModule } from './site/films/films.module';
import { HomeModule } from './site/home/home.module';
@NgModule({  declarations: [
    AppComponent
],
  imports: [
    BrowserModule,
    AboutModule,
    FilmsModule,
    HomeModule,
    AppRoutingModuleModule
],
  providers: [],
  bootstrap: [AppComponent]
}) export class AppModule { }
```

Formation Angular 10

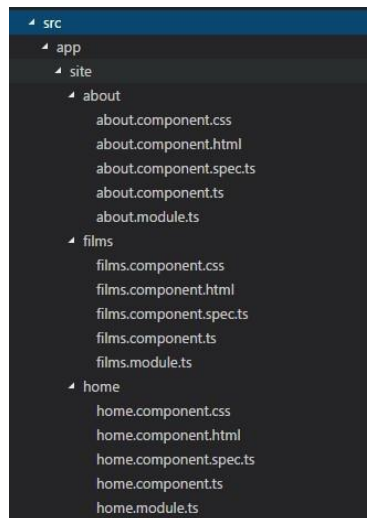
1. Ajouter les composants qui serviront à afficher chacune des pages :

ng g component site/home

ng g component site/about

ng g component site/films

a) *Nous devons avoir la structure ci-dessous*



Les fichiers modules sont mis à jour :

 **home.module.ts**

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { HomeComponent } from './home.component';

@NgModule({
  declarations:
[HomeComponent],
  imports: [
    CommonModule
  ] }) export class
HomeModule { }
```

films.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FilmsComponent } from './films.component';
@NgModule({
  declarations: [FilmsComponent],
  imports: [
    CommonModule
  ] }) export class
FilmsModule { }
```

about.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { AboutComponent } from './about.component';
@NgModule({
  declarations: [AboutComponent],
  imports: [
    CommonModule
  ] }) export class AboutModule { }
```

III. Mise en place des Routes

A. Dans le module principal, on a déjà le chargement des routes principales

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModuleModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { AboutModule } from './site/about/about.module';
import { FilmsModule } from './site/films/films.module';
import { HomeModule } from './site/home/home.module';

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

Voir le fichier suivant **app-routing.module.ts** :

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [];

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


1. – Ajouter les routes aux modules enfants
 - Le module racine ne contiendra pas les routes ceci pour séparer au maximum les composants/modules
 - Le module racine chargera les routes indiquées par ses enfants.
 - Pour configurer des routes dans des modules autre que le module racine nous devons utiliser la méthode **forChild** :

Exemple : **RouterModule.forChild([{ path: 'ROUTE_VERS_COMPOSANT', component: COMPOSANT_A_CHARGER }])**

home.module.ts

```
import { NgModule, Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { HomeComponent } from './home.component';
import { Routes,RouterModule } from '@angular/router';

const routes: Routes = [
  {
    path: '',component:HomeComponent
  }
];

@NgModule({
  declarations: [HomeComponent],
  imports: [
    CommonModule,
    RouterModule.forChild(routes)
  ] }) export class HomeModule { }
```

films.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FilmsComponent } from './films.component';
import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [
  {
    path: 'films', component: FilmsComponent
  }
];

@NgModule({
  declarations: [FilmsComponent],
  imports: [
    CommonModule,
    RouterModule.forChild(routes)
  ] }) export class FilmsModule { }
```

about.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { AboutComponent } from './about.component';
import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [
  {
    path: 'about', component: AboutComponent
  }
];

@NgModule({
  declarations: [AboutComponent],
  imports: [
    CommonModule,
    RouterModule.forChild(routes)
  ] }) export class AboutModule { }
```

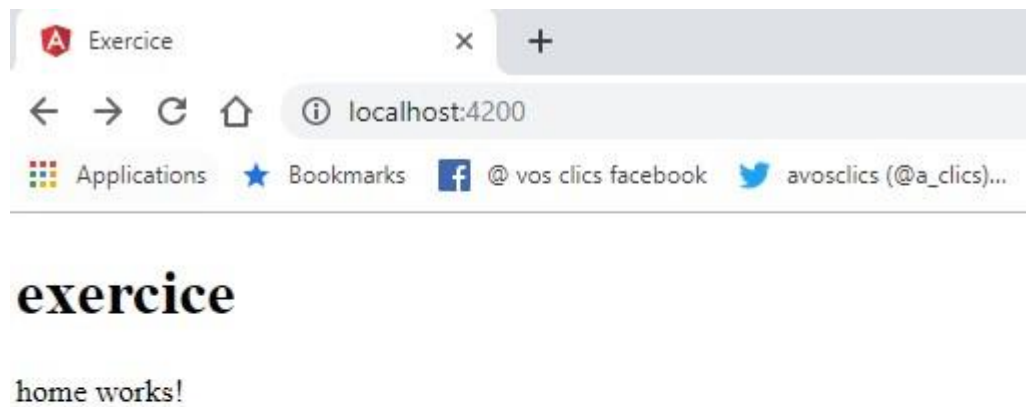
2. Indiquer le point où se chargera les templates des composants des pages

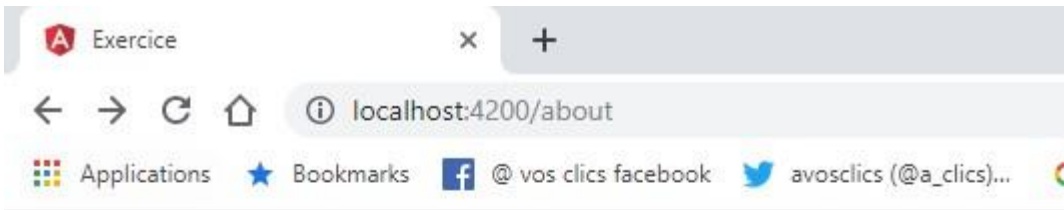
✚ Lorsque le module Root charge les sous-modules, il prend en compte les routes indiquées dans celles-ci.

Fichier app.component.html :

```
<div class="container">
  <h1>{{title}}</h1>
</div>
<router-outlet></router-outlet>
```

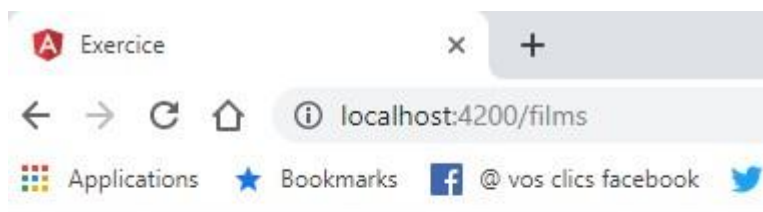
3. Tester les routes :





exercice

about works!



exercice

films works!

- Reprendre ce qui a été fait et ajouter un module qui gère les pages pour les erreurs (404, 401, etc.) :
 - ng g module site/errors
 - ng g c site/errors

✚ Exemple de configuration du module pour les erreurs d'URL

Dans le fichier errors.module.ts :

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { ErrorsComponent } from './errors.component';
import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [
  {
    path: '404', component: ErrorsComponent
  }
];

@NgModule({
  declarations: [ErrorsComponent],
  imports: [
    CommonModule,
    RouterModule.forChild(routes)
  ] }) export class ErrorsModule { }
```

✚ Il est possible de rediriger toutes les adresses inexistantes vers les pages d'erreurs.

✚ Dans ce dernier cas, le module erreur doit être le dernier à être chargé par le module root
!!!

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { ErrorsComponent } from './errors.component';
import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [
  {
    path: '404', component: ErrorsComponent
  },
  {
    path: '**', redirectTo: '/404'
  }
];

@NgModule({
  declarations: [ErrorsComponent],
  imports: [
    CommonModule,
    RouterModule.forChild(routes)
  ] }) export class ErrorsModule { }
```

IV. Ajout de Bootstrap

npm install bootstrap@4.0

Puis mettre à jour le fichier : angular.json

```
"styles": [
  "node_modules/bootstrap/dist/css/bootstrap.min.css",
  "src/styles.css"
],
```

V. Ajout d'un menu

1. Dans la console taper : **ng g module site/shared**
2. Dans la console taper : **ng g component site/shared/navmenu**
3. Le répertoire /shared/ contiendra les éléments partagés
4. Ajouter **shared.module.ts** au module Root.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { AboutModule } from './site/about/about.module';
import { FilmsModule } from './site/films/films.module';
import { HomeModule } from './site/home/home.module';
import { ErrorsModule } from './site/errors/errors.module';
import { SharedModule } from './site/shared/shared.module';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AboutModule,
    FilmsModule,
    HomeModule,
    ErrorsModule,
    SharedModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
}) export class AppModule { }
```

1. Ajouter le composant au module `shared.module.ts`
- Nous ajoutons aussi : `exports: [NavmenuComponent]`
 - `exports: [composant, module, ...]` Indique que le module exporte des éléments vers le module parent. Le module parent pourra donc utiliser directement les éléments exportés.

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { NavmenuComponent } from '../navmenu/navmenu.component';

@NgModule({
  declarations: [NavmenuComponent],
  imports: [
    CommonModule
  ],
  exports : [NavmenuComponent]
})

export class SharedModule { }
```

Dans `app.component.html` :

```
<div class="container">
  <h1>{{title}}</h1>
</div>
<app-navmenu></app-navmenu>
<router-outlet></router-outlet>
```


2. Vérifier que le composant navmenu s'affiche :



3. Importer RouterModule à shared.module.ts

- RouterModule embarque des directives pour faire des menus

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { NavmenuComponent } from './navmenu/navmenu.component';
import { RouterModule } from '@angular/router';
@NgModule({
  declarations: [NavmenuComponent],
  imports: [
    CommonModule,
    RouterModule
  ],
  exports : [NavmenuComponent]
}) export class SharedModule { }
```

4. Modifier le template **navmenu.component.html** pour y ajouter un menu avec les liens vers les différentes pages

- Les classes utilisées ci-dessous sont des classes Bootstrap qui permettent de mettre facilement en forme un menu

```
<div class="container">
  <nav class="navbar navbar-light bg-light">
    <ul class="navbar-nav">
      <li>
        <a class="nav-link" [routerLink]="['/']">Accueil</a>
      </li>
      <li>
        <a class="nav-link" [routerLink]="['/films']">Films</a>
      </li>
      <li>
        <a class="nav-link" [routerLink]="['/about']">About</a>
      </li>
    </ul>
  </nav>
</div>
```



VI. Ajout d'un formulaire

A. Créer un nouveau composant : ng g component site/films/searchform

- Ce composant est un formulaire pour rechercher des films
- Angular CLI a ajouté ce composant au module films.module.ts

1. Ajouter le formulaire au template : **films.component.html**

```
<p> films  
works!  
</p>  
<app-searchform></app-searchform>
```

exercice

Accueil

Films

About

films works!

searchform works!

B. Créer un formulaire dans `searchform.component.html`.

Ce formulaire doit contenir un champ de **type texte** et un autre de type **number**.

Les valeurs du champ de type number devront être comprises entre **1900** et **2022**.

```
<div class="container">
  <form class="form-inline well well-in">
    <div class="form-group col-md-5">
      <label>Veuillez entrer un titre de film</label>
      <input type="text" class="form-control" placeholder="Titre du film">
    </div>
    <div class="form-group col-md-5">
      <label>Veuillez choisir une année</label>
      <input type="number" min="1900" max="2022" step="1" value="2018"
        class="form-control">
    </div>
    <button type="submit" class="btn btn-primary">
      Rechercher
    </button>
  </form>
</div>
```

C. Ajouter au module : `films.module.ts`

- Les module FormsModule, ReactiveFormsModule du package @angular/forms

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FilmsComponent } from './films.component';
import { ReactiveFormsModule, FormsModule } from '@angular/forms';
import { Routes, RouterModule } from '@angular/router';
import { SearchformComponent } from './searchform/searchform.component';

const routes: Routes = [
  {
    path: 'films', component: FilmsComponent
  }
];

@NgModule({
  declarations: [FilmsComponent, SearchformComponent],
  imports: [
    CommonModule,
    FormsModule,
    ReactiveFormsModule,
    RouterModule.forChild(routes)
  ] })
export class FilmsModule { }
```

6

D. Ajouter à searchform.component.ts :

- **import { FormControl, FormBuilder, FormGroup, Validators } from '@angular/forms';**
- **FormBuilder** permet de créer plus facilement des « **FormControl** » (objet lié à un champ) et des « **FormGroup** » (objet lié à un formulaire).
- En utilisant l'injection par le constructeur de **searchform.component.ts**, récupérer un objet de type **FormBuilder**.
- Grâce à l'objet **FormBuilder** (qui est un **service**), fabriquer un objet de type **FormGroup** qui contiendra deux **FormControl**, l'un pour le titre, l'autre pour l'année qui aura pour valeur par défaut : 2018.

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({
  selector: 'app-searchform',
  templateUrl: './searchform.component.html',
  styleUrls: ['./searchform.component.css']
})
export class SearchformComponent implements OnInit {
  searchForm : FormGroup;
  constructor(private fb:FormBuilder) { }
  ngOnInit() {
    this.searchForm = this.fb.group({
      title:'',
      year:''
    });
  }
}
```

- Lier le formulaire **searchform.component.html** au composant **searchform.component.ts** en utilisant l'objet de type **FormGroup**.
- Ajouter une méthode (fonction) **startSearch** au composant qui sera appelé à chaque soumission du formulaire. La méthode **startSearch** affiche les valeurs du formulaire en utilisant l'objet **FormGroup**.

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({
  selector: 'app-searchform',
  templateUrl: './searchform.component.html',
  styleUrls: ['./searchform.component.css']
})
export class SearchformComponent implements OnInit {
  searchForm : FormGroup;
  constructor(private fb:FormBuilder) { }
  ngOnInit() {
    this.searchForm = this.fb.group({
      title:'',
      year:2018
    });
  }
  startSearch() {
    console.log("Recherche Lancée");
  }
}
```

```
<div class="container">
  <form (ngSubmit)="startSearch()" [formGroup]="searchForm"
    class="form-inline well wellin">
    <div class="form-group col-md-5">
      <label>Veuillez entrer un titre de film</label>
      <input type="text" class="form-control"
        placeholder="Titre du film"
        FormControlName="title">
    </div>
    <div class="form-group col-md-5">
      <label>Veuillez choisir une année</label>
      <input type="number" min="1900" max="2022" step="1" value="2018"
        class="form-control" FormControlName="year">
    </div>
    <button type="submit" class="btn btn-primary">
      Rechercher
    </button>
  </form>
</div>
```

- Tester si la fonction **startSearch** est enclenchée lors du clic sur le bouton rechercher.

exercice

Accueil

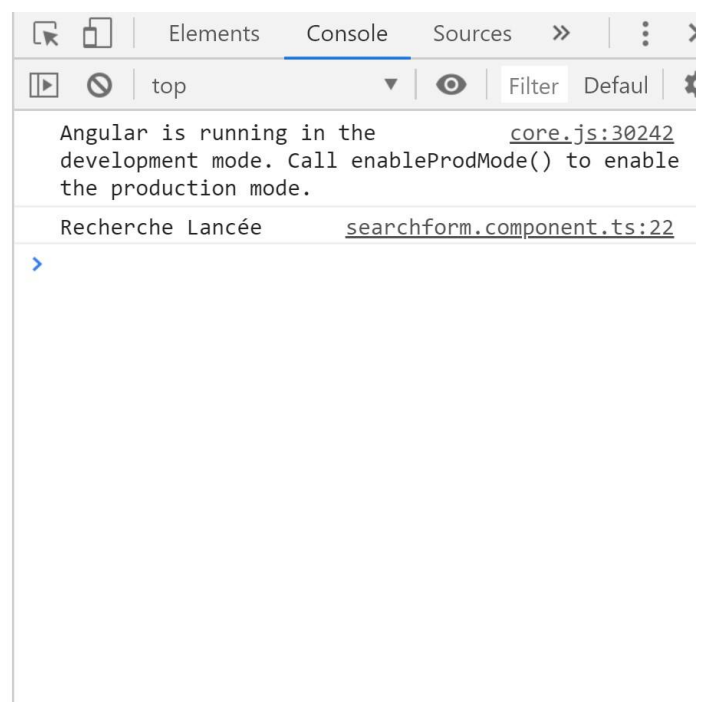
Films

About

ilms works!

Veuillez entrer un titre de film

Veuillez choisir une année



VII. Création d'un service émettant une requête http

1. Créer un répertoire **services** dans le répertoire **app/site/films**
2. Dans la console taper : **ng g service site/films/services/searchmovie**
 - A. Ce service est une classe qui permettra de faire différentes requêtes pour la recherche de films.
3. Ajouter le service au module : **films.module.ts**
4. Ajouter le module HttpClientModule au module **films.module.ts**

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FilmsComponent } from '../films.component';
import { ReactiveFormsModule, FormsModule } from '@angular/forms';
import { Routes, RouterModule } from '@angular/router';
import { HttpClientModule } from '@angular/common/http';
import { SearchformComponent } from '../searchform/searchform.component';
import { SearchmovieService } from '../services/searchmovie.service';
const routes: Routes = [
  {
    path: 'films', component: FilmsComponent
  }
];

@NgModule({
  declarations: [FilmsComponent, SearchformComponent],
  providers : [SearchmovieService],
  imports: [
    CommonModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule,
    RouterModule.forChild(routes)
  ] }) export class FilmsModule { }
```

- Ajouter une méthode **search(action: (data: **Object**) , title: string, year: number = 0): void** à **SearchmovieService**.

Cette méthode doit faire une requête de type **get** en utilisant le service http fournit par Angular. L'URL est la suivant :

- ``http://www.omdbapi.com/?t=${title}&y=${year}&plot=full``
- Cette méthode récupérera les titres de film ainsi que l'année du film
- La valeur par défaut de l'année est 0, si aucune année est fournie. Dans ce cas, l'URL sera :
- ``http://www.omdbapi.com/?t=${title}&plot=full``
- action: (data: **Object**)** indique qu'une fonction pouvant recevoir 1 paramètre de type Object est attendu

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpResponse } from '@angular/common/http';
@Injectable({
  providedIn: 'root'
})
export class SearchmovieService {
  constructor(private http:HttpClient) { }
  search(action:any,title:string,year:number=0) : void {
    let results ={};
    let y = year? `&y=${year}`:'';
    this.http.
      get(`http://www.omdbapi.com/?apikey=b267f2ad&t=${title}${y}&plot=full`)
      .subscribe(
        (response) =>{
          action(response);
        }
      )
  }
}
```



1. Injecter le service à `searchform.component.ts`

Lancer la recherche en appelant la méthode **search** de ce service lors de la soumission du formulaire, pour cela utiliser la méthode **startSearch** de **searchform.component.ts**

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormBuilder, FormGroup, Validators } from '@angular/forms';
import { SearchmovieService } from '../services/searchmovie.service';

@Component({
  selector: 'app-searchform',
  templateUrl: './searchform.component.html',
  styleUrls: ['./searchform.component.css']
})
export class SearchformComponent implements OnInit {
  searchForm : FormGroup;

  constructor(private fb:FormBuilder,
    private searchMovie:SearchmovieService) { }

  ngOnInit() {
    this.searchForm = this.fb.group({
      title:'',
      year:2018
    });
  }

  startSearch() {
    let action= (data:Object) =>{
      console.log(data);
    }
    this.searchMovie.search(action,"test");
  }
}
```

2. Tester si la requête se fait bien en lançant une recherche.

Regarder dans la console du navigateur si une réponse apparaît.

exercice

Accueil

Films

About

lms works!

Veuillez entrer un titre de film

Veuillez choisir une année

Rechercher

Elements Console Sources >> ⋮ ✕

▶ ⏸ top ▼ 🔍 Filter Default ⚙

Angular is running in the [core.js:30242](#) development mode. Call enableProdMode() to enable the production mode.

[searchform.component.ts:25](#)
`{Title: "Beta Test", Year: "2016", Rated: "Not Rated", Released: "22 Jul 2016", Runtime: "88 min", ...}`

>

Elements Console Sources Network Performance Memory Application >> ⋮ ✕									
⏸ ⏹ 🔍	View:	Group by frame			Preserve log		Disable cache		Offline No thrott
Filter Clear	<input type="text"/>		<input type="checkbox"/> Hide data URLs		All	XHR	JS	CSS	Img Media Font Doc WS Manifest Other
10 ms	20 ms	30 ms	40 ms	50 ms	60 ms	70 ms	80 ms	90 ms	100 ms 110
Name	Status	Type	Initiator	Size	Time	Waterfall ▲			
polyfills.js	304	script	films	212 B	10 ms				
styles.js	304	script	films	213 B	12 ms				
vendor.js	304	script	films	213 B	18 ms				
main.js	304	script	films	211 B	20 ms				
engines.json	200	xhr	mcafee wa conte...	19.4 KB	5 ms				
info?t=1560905969285	200	xhr	zone-evergreen.js...	368 B	2 ms				
websocket	101	websocket	sockjs.js:1683	0 B	Pending				
?apikey=b267f2ad&t=testj&plot=f...	200	xhr	zone-evergreen.js...	1.5 KB	18 ms				

3. Fournir les valeurs entrées par l'utilisateur dans le formulaire au service. Utiliser pour cela l'attribut **searchForm: FormGroup**

```
startSearch() {  
  let action= (data:Object) =>{  
    console.log(data);  
  }  
  this.searchMovie.search(action,this.searchForm.value.title,this.searchForm.value.year); }  
}
```

VIII. Ajouter des validateurs à un formulaire

- ✚ Dans le fichier **searchform.component.ts**, importer **FormControl** et **Validators** du package **@angular/forms** et mettre des validations aux champs de recherche :
 - Le **titre** est requis, aura **30 caractères maximum** et ne devra avoir que les caractères suivants : **a-z, A-Z, 0-9, espace, virgule et point**
 - La **date** sera un entier **optionnel** compris **entre 1900 et 2022**
 - Modifier l'initialisation de **this.searchForm = fb.group(...)** ngOnInit() pour que cela soit pris en compte
 - Créer un **validateur personnalisé** pour valider les dates entre 1900 et 2022, celui-ci se trouvera dans un nouveau fichier **formvalidators.ts** placé dans le **répertoire searchform**. Il faudra l'importer dans **searchform.component.ts**
 - `ng g class site/films/FormValidators`
 - <https://regex101.com/> permet de tester les expressions régulières (regex)

```
import { AbstractControl, ValidatorFn } from '@angular/forms';

export class FormValidators {

  static integerBetween(min: number, max: number) : ValidatorFn {

    return (c: AbstractControl) => {

      if (!Number.isInteger(c.value)) {

        //si la valeur n'est pas un entier

        return {

          integer : {

            valid : false

          }

        }

      }

      else if ((c.value < min) || (c.value > max)) {

        //si la valeur est en dehors des limites

        return {

          limit : {

            valid : false

          }

        };

      }

      //Si tout est ok

      return null;

    }

  };

}
```

```

import { Component, OnInit } from '@angular/core';
import { FormControl, FormBuilder, FormGroup, Validators } from '@angular/forms';
import { SearchmovieService } from '../services/searchmovie.service';
import { FormValidators } from '../form-validators';

@Component({
  selector: 'app-searchform',
  templateUrl: './searchform.component.html',
  styleUrls: ['./searchform.component.css']
})
export class SearchformComponent implements OnInit {
  searchForm : FormGroup;
  title : FormControl;
  year : FormControl;
  constructor(private fb:FormBuilder,
    private searchMovie:SearchmovieService) { }
  ngOnInit() {
    let titlePattern = '[ a-zA-Z0-9,\.\.]+';
    let yearPattern = '[0-9]{4}';
    this.title = this.fb.control(
      '',[Validators.required,Validators.maxLength(30),Validators.pattern(titlePattern)]
    );
    this.year = this.fb.control(
      '2018',[Validators.pattern(yearPattern),FormValidators.integerBetween(1900,2022)]
    );
    this.searchForm = this.fb.group({
      title:this.title,
      year:this.year
    });
  }
  startSearch() {
    let title = this.title.valid ? this.title.value:null;
    let year = this.year.valid ? this.year.value : null;
    let action= (data:Object) =>{
      console.log(data);
    }
    if(title)
      this.searchMovie.search(action,title,year);
  }
}


```


IX. Création d'une directive pour l'affichage des résultats

A. Créer une directive : `ng g directive site/films/directives/list`

- Cette directive servira à afficher les résultats
- Angular CLI a ajouté cette directive à : **`films.module.ts`**

B. Retoucher `searchform.component.ts` :

 POUR QU'IL ENREGISTRE LES RESULTATS DANS UN ATTRIBUT privé : `results` et les erreurs dans un attribut privé `errors`

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormBuilder, FormGroup, Validators } from '@angular/forms';
import { SearchmovieService } from '../services/searchmovie.service';
import { FormValidators } from '../form-validators';
@Component({
  selector: 'app-searchform',
  templateUrl: './searchform.component.html',
  styleUrls: ['./searchform.component.css']
})
export class SearchformComponent implements OnInit {
  searchForm : FormGroup;
  title : FormControl;
  year : FormControl;
  results:any;
  errors:string;
  constructor(private fb:FormBuilder,
    private searchMovie:SearchmovieService) { }
  ngOnInit() {
    let titlePattern = '[ a-zA-Z0-9,\.\.]+';
    let yearPattern = '[0-9]{4}';
    this.title = this.fb.control(
      '',[Validators.required,Validators.maxLength(30),Validators.pattern(titlePattern)]
    );
    this.year = this.fb.control(
      '2018',[Validators.pattern(yearPattern),FormValidators.integerBetween(1900,2022)]
    );
    this.searchForm = this.fb.group({
      title:this.title,
      year:this.year
    });
  }
}
```

```

startSearch() {
  let title = this.title.valid ? this.title.value:null;
  let year = this.year.valid ? this.year.value : null;
  let that=this;

  /*      référence vers l'objet courant : à cause      de la portée des variables dans
les Arrow fucntions      et les fin de bloc {} (closures)
  */
  let action= (data:Object) =>{
    if(data['Error']) {
      that.errors = data['Error'];
      that.results='';
    }
    else
    {
      that.errors='';
      that.results=data;
    }
  }
  if(title)
    this.searchMovie.search(action,title,year);
  else
    this.errors = 'Titre non obligatoire !';
}
}

```

1. Renommer le **sélecteur** de la **directive** en **movieList** (attention à la casse).

- Ajouter une **entrée** à la directive pour qu'elle puisse recevoir des données sous forme d'une liste d'objets.
- L'entrée se nommera aussi **movieList** et enregistrera la valeur dans **un attribut privé**, nommé **_list**

```
import { Directive, Input } from '@angular/core';
@Directive({
  selector: '[movieList]'
})
export class ListDirective {
  private _list:Object;
  constructor() { }

  @Input()
  set movieList(value) {
    console.log("Valeur reçu par la directive"+value);
  }
}
```

2. Appeler la directive depuis le template **searchform.component.html**

- L'appel se fera sur un élément <div>
- La **directive recevra** la valeur de l'attribut **results** du composant SearchformComponent
- Nous exécuterons la directive que s'il n'y a pas d'erreurs et que des résultats sont présents

```
<div class="container">
  <form (ngSubmit)="startSearch()" [formGroup]="searchForm"
    class="form-inline well wellin">
    <div class="form-group col-md-5">
      <label>Veuillez entrer un titre de film</label>
      <input type="text" class="form-control"
        placeholder="Titre du film" formControlName="title">
    </div>
    <div class="form-group col-md-5">
      <label>Veuillez choisir une année</label>
      <input type="number" min="1900" max="2022" step="1" value="2018"
        class="form-control" formControlName="year">
    </div>
    <button type="submit" class="btn btn-primary">
      Rechercher
    </button>
  </form>
</div>
<div>
  <div *ngIf="errors" class="alert alert-danger">
    <strong>Aucun résultat !</strong>
    {{errors}}
  </div>
  <div *ngIf="results" class="list-group"
    [movieList]="results">
  </div>
</div>
```

3. Modifier la directive pour qu'elle affiche les données

- **ElementRef** contient un attribut **nativeElement** qui est l'élément **DOM** de base.

```
import { Directive, Input, ElementRef } from '@angular/core';
@Directive({
  selector: '[movieList]'
})
export class ListDirective {

  private _list:Object;  constructor(private el:ElementRef) { }
  @Input()
  set movieList(movie) {
    let temp = `<a class="list-group-item list-group-item-action">`;
    temp +=
      `Titre : ${movie.Title} Année: ${movie.Year} Réalisateur : ${movie.Director}`;
    temp += '</a>';
    this.el.nativeElement.innerHTML = temp;

  }
}
```

4. Tester le résultat

Veuillez entrer un titre de film

star

Veuillez choisir une année

Rechercher

Titre : Star Wars: Episode IV - A New Hope Année: 1977 Réalisateur : George Lucas

Veuillez entrer un titre de film

Veuillez choisir une année

Rechercher

Aucun résultat ! Movie not found!

Veuillez entrer un titre de film

the kid

Veuillez choisir une année

2001

Rechercher

Titre : The Kid Année: 1921 Réalisateur : Charles Chaplin