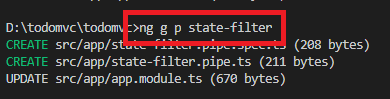
Pipe用法

1. 新增一個Pipe用法
   1. 輸入Ng g p [名稱]



1. 因為我有切Service，所以在data.service.ts加入一個變數，不然也可以在app.component.ts層加入變數即可

export class DataService {

state;

}

1. 在新建的state-filter.pipe.ts中寫入想要的方法
   1. 第一個value: any是回傳值
   2. state?: any是參數，要傳多個參數可以用逗點的方式繼續加下去

export class StateFilterPipe implements PipeTransform {

transform(value: any, state?: any): any {

if (state === 'active') {

return value.filter(x => x.isCompleted === false);

}

if (state === 'completed') {

return value.filter(x => x.isCompleted === true);

}

return value;

}

}

1. 在state-filter.pipe.ts設定
   1. 上方有加入pure: false就代表即時更新

@Pipe({

name: 'stateFilter',

pure: false

})

1. 之後在app.component.html中使用剛剛建立的Pipe
   1. 將dataService.todos回傳內容
   2. 轉換為stateFilter: dataService.state的型態

<li \*ngFor=" let todo of (dataService.todos | stateFilter: dataService.state); let idx = index " [ngClass]="{ completed: todo.isCompleted }" >

1. 最後就是要在section-footer.component.html加入click事件
   1. 加入Click事件(click)="dataService.state = 'active'"
   2. 同時加入CSS樣式，[class.selected]="dataService.state === 'active'"
   3. 上方跟[ngClass]="{ selected: dataService.state === 'active' }同樣意思

<ul class="filters">

<li>

<a [class.selected]="dataService.state === ''" href="#/" (click)="dataService.state = ''">All</a>

</li>

<li>

<a href="#/active" [class.selected]="dataService.state === 'active'" (click)="dataService.state = 'active'" >Active</a>

</li>

<li>

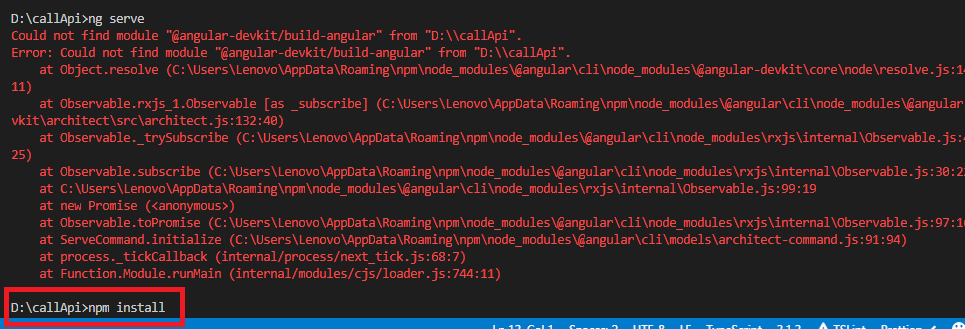
<a href="#/completed" [class.selected]="dataService.state === 'completed'" (click)="dataService.state = 'completed'" >Completed</a>

</li>

</ul>

重建專案失敗解決方案

1. 如果輸入ng serve時有遇到以下問題
2. 可以輸入npm install重新安裝module
3. 可能是電腦一開始沒有幫我們建立到module這部份



建立第一個API

1. 在app.module.ts加入import

import { HttpClientModule } from '@angular/common/http';

1. 同時在下方imports加入HttpClientModule

@NgModule({

declarations: [AppComponent],

imports: [BrowserModule, HttpClientModule],

providers: [],

bootstrap: [AppComponent]

})

1. 此時就可以在app.component.ts引用

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

1. Call API的方式
   * 1. Constructor一個剛剛引用的HttpClient
     2. 使用http.get方法，寫入對應URL
     3. 透過Subscribe回傳console.log(data)

export class AppComponent {

constructor(private http: HttpClient) {

http

.get('https://jsonplaceholder.typicode.com/posts')

.subscribe(data => {

console.log(data);

});

}

}

透過MAP方式做欄位對應

1. 在app.component.ts引用MAP

import { map } from 'rxjs/operators';

1. 使用MAP的方式
   1. 使用.pipe的方式用MAP欄位對應
   2. 然後只return id跟title

constructor(private http: HttpClient) {

http

.get('https://jsonplaceholder.typicode.com/posts')

.pipe(

map((posts: any[]) => {

return posts.map((post: any) => {

return { id: post.id, title: post.title };

});

})

)

.subscribe(data => {

console.log(data);

});

}

1. 重構程式碼
   1. 建立posts陣列變數
   2. 改成只註冊constructor
   3. 建立apiSource直接Call API取值
   4. 利用loadData處理方法回傳結果
   5. 未來可以不要寫在component.ts，應該存在Service層

export class AppComponent {

posts = [];

constructor(private http: HttpClient) {}

apiSource = this.http.get('https://jsonplaceholder.typicode.com/posts').pipe(

map((posts: any[]) => {

return posts.map((post: any) => {

return { id: post.id, title: post.title };

});

})

);

loadData() {

this.apiSource.subscribe(data => {

console.log(data);

this.posts = data;

});

}

}

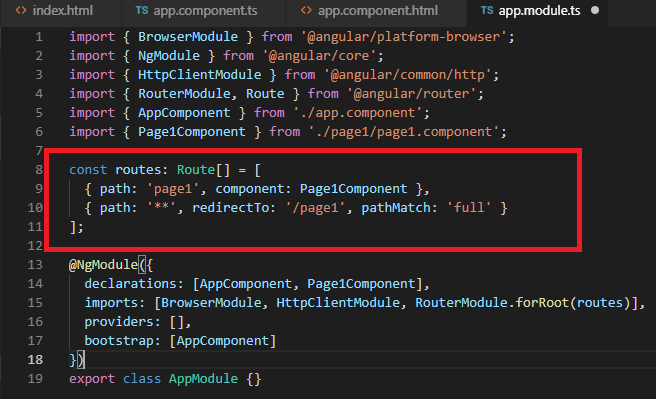
使用Router代替HTML href

1. 在app.module.ts內寫入imput Route內容
   1. 必須同時引用RouterModule,Route
   2. 而Page1Component是我們透過ng g c新建的Component位置

import { RouterModule,Route } from '@angular/router';

import { Page1Component } from './page1/page1.component';

1. 且在imports下方@Module中間，新增常數routes(第三步會解釋)



1. 修改常數內容為對應連結
   1. Path是前端輸入的routerLink連結名稱
   2. Component是剛剛引用的名稱
   3. Path=\*\*代表如果都不是上方輸入的情況
   4. redirectTo代表轉址到的地點
   5. pathMatch是代表位置要全部相符

const routes: Route[] = [

{ path: 'page1', component: Page1Component },

{ path: '\*\*', redirectTo: '/page1', pathMatch: 'full' }

];

1. 在Module加入中加入對應內容
   1. Declarations中需要加入要連結的Component
   2. Imports有分forRoot和forChild，通常使用forRoot即可

@NgModule({

declarations: [AppComponent,Page1Component],

imports: [BrowserModule, HttpClientModule, RouterModule.forRoot(routes)],

providers: [],

bootstrap: [AppComponent]

})

1. 最後就可以在app.component.html中設定
   1. routerLink轉址
   2. 並加入router-outlet，即可更改部份內容

<ul>

<li><a routerLink="/page1">Page1</a></li>

</ul>

<router-outlet></router-outlet>

加入子畫面項目

1. 在app.module.ts內寫入imput Route內容
   1. 必須同時引用RouterModule,Route
   2. 而Page1, Page2,Page2-detail是透過ng g c新建的Component位置

import { RouterModule, Route } from '@angular/router';

import { Page1Component } from './page1/page1.component';

import { Page2Component } from './page2/page2.component';

import { Page2DetailComponent } from './page2-detail/page2-detail.component';

1. 且在imports下方@Module中間，加入routes常數內容如下
   1. 這裡建立page2時，有在項下多建立children
   2. 若有子項目時，需要多設定傳入的內容，例如設定path:’id’

const routes: Route[] = [

{ path: 'page1', component: Page1Component },

{

path: 'page2',

component: Page2Component,

children: [{ path: ':id', component: Page2DetailComponent }]

},

{ path: '\*\*', redirectTo: '/page1', pathMatch: 'full' }

];

1. 在Module加入中加入對應內容

@NgModule({

declarations: [

AppComponent,Page1Component,Page2Component,Page2DetailComponent

],

imports: [BrowserModule, HttpClientModule, RouterModule.forRoot(routes)],

providers: [],

bootstrap: [AppComponent]

})

1. 最後就可以在app.component.html中設定跳轉內容
   1. 也可以像下面在routerLink後面可以加入id

<ul>

<li><a routerLink="/page1">Page 1</a></li>

<li><a routerLink="/page2">Page 2</a></li>

<li><a routerLink="/page2/1">Page 2 Detail with ID 1</a></li>

<li><a routerLink="/page2/2">Page 2 Detail with ID 2</a></li>

<li><a routerLink="/page2/3">Page 2 Detail with ID 3</a></li>

</ul>

轉址的方式2

1. 在page2.component.ts中加入引用

import { Router } from '@angular/router';

1. page2.component.ts寫入gotoPage1方法

export class Page2Component implements OnInit {

constructor(private router: Router) {}

gotoPage1() {

this.router.navigate(['/page1']);

}

}

1. 最後在畫面上寫入click

<button (click)="gotoPage1()">Go to Page1</button>

表單(開啟Form練習檔)

1. 在app.module.ts中import表單

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

1. 且在app.module.ts下方import部份也要加入FormsModule

imports: [

FormsModule,

],

1. 接下來在signup.component.html指定form表單

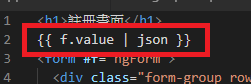
<form #f="ngForm">

1. 且修改姓名input內的內容
   1. 加入ngModel
   2. 給予一個name，例如此處為personName

<input name="personName" ngModel type="text" class="form-control" placeholder="姓名 (必填)" required/>

1. 將剛剛設定的#f表單內容，轉換為Json格式顯示在畫面上

{{ f.value | json }}



表單欄位驗證

1. 在signup.component.html修改姓名的input內容
   1. 加入#personName="ngModel"

<input name="personName" ngModel #personName="ngModel" type="text" class="form-control" placeholder="姓名 (必填)" required/>

1. 並在下面寫下，如果驗證失敗將會顯示false

{{ personName.errors | json }}

<div>invalid: {{ personName.invalid }}</div>

<span \*ngIf="personName.hasError('required')">必須輸入姓名欄位</span>

表單全部合法驗證

1. 全部要驗證的欄位都設定完後，可以在送出時搭配f.invalid

<button type="submit" class="btn btn-primary" [disabled]="f.invalid">送出</button>

重新表單內容

1. 可以透過reset方法讓表單內容初始化
   1. 使用f.reset()就可以初始化
   2. 也可以在裡面放Object，將值初使化代入

<button (click)="f.reset({ personName: 'hello' })">reset</button>

取得表單資訊

1. 在signup.component.ts中引用ViewChild

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

1. 可以用ViewChild方式取得form

export class SignupComponent implements OnInit {

//form盡量不要用ViewChild的方式取得

@ViewChild('f') form;

submitForm(f) {

// 使用 ViewChild 取得 Form 實體

// console.log(this.form);

// 透過傳值的方式取得 Form實體

console.log(f);

}

1. 前端signup.component.html加入

<button type="submit" class="btn btn-primary" [disabled]="f.invalid" (click)="submitForm(f)" >

或在form表單上面加入click事件

<form #f="ngForm" (ngSubmit)="submitForm(f)">

1. 之後將內容顯示在畫面上看是否成功

{{ f.form.getRawValue() | json }}

選擇核選方塊

1. 在核選方塊中加入屬性
   1. 加入ngModel
   2. 給予一個name
   3. 並且指定#send=ngModel

<input class="form-check-input" type="checkbox" value="" id="defaultCheck1" ngModel name="send" #send="ngModel" />

1. 之後若要勾選後將某區域顯示/隱藏時可透過If判斷send.value

<ng-container \*ngIf="send.value">

<div class="form-group row">

<label for="staticEmail" class=" col-form-label">地址</label>

<div ><input type="text" class="form-control" /></div>

</div>

</ng-container>

解析JSON資料

1. 首先看一下資料樣子，但我們希望只得到縣市就好



1. 因此這時候我們需要建立一個變數在signup.component.ts

cityTownAreaSource;

1. 並且製作一個方法

loadData() {

this.http.get('/assets/data/cityarea.json').subscribe(data => {

console.log(Object.keys(data));

this.cityTownAreaSource = Object.keys(data);

});

}

1. 最後在去令初使化時執行該方法

ngOnInit() {

this.loadData();

}

1. 而在signup.component.html畫面上可以使用\*egFor來撰寫

<select id="inputState" class="form-control">

<option selected>請選擇縣市</option>

<option \*ngFor="let city of cityTownAreaSource">{{city}}</option>

</select>

使用ReactiveForms做表單

1. 要在app.module.ts引用ReactiveFormsModule(FormsModule可以不用)

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

1. 並且在下方import註冊

imports: [

ReactiveFormsModule,

],

1. 同時到需要用到的地方(如：singup-advance.component.ts)內引用

import { FormGroup, FormControl, FormBuilder, Validators} from '@angular/forms';

1. 最後就可以繼續將我們要驗證的條件寫入formData
   1. FormControl內第一個參數是初始值
   2. 後面是放其他條件

export class SingupAdvanceComponent implements OnInit {

formData = new FormGroup({

name: new FormControl('12333', [Validators.required]),

email: new FormControl(),

confirmEmail: new FormControl({ value: '222', disabled: true })

});

}

或下方兩者選一即可

formData = this.fb.group({

name: ['12333', [Validators.required]],

email: '',

confirmEmail: [{ value: '222', disabled: true }]

});

1. 最後在singup-advance.component.html畫面部份加入我們設定的表單
   1. 透過[formGroup]屬性即可設定

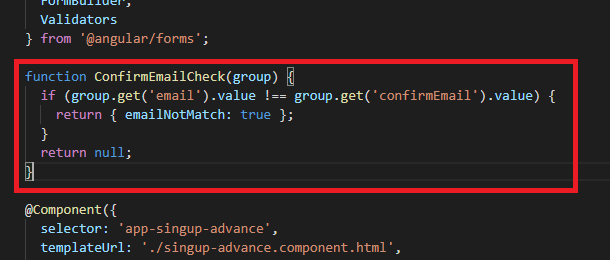
<form [formGroup]="formData">

1. 將Form內的驗證規則導入input中
   1. 在要加驗證的地方使用formControlName屬性
   2. 等於我們剛剛在formData中設定的名稱

<input type="text" class="form-control" placeholder="姓名 (必填)" formControlName="name" />

使用Group驗證方式 – 例如二次驗證信箱

1. 先加入比對方法
   1. 可以寫在最外層，但要用function



1. 方法內容是比對email跟confirmEmail是否相等的方法

function ConfirmEmailCheck(group) {

if (group.get('email').value!==group.get('confirmEmail').value) {

return { emailNotMatch: true };

}

return null;

}

1. 在formData中寫各種驗證條件
   1. 但這邊要注意的是emailGroup是可以包含了email和confirmEmail
   2. 而驗證條件validator就是第二步驟寫的[ConfirmEmailCheck]

export class SingupAdvanceComponent implements OnInit {

formData = this.fb.group({

name: ['', [Validators.required]],

emailGroup: this.fb.group(

{

email: '',

confirmEmail: [{ value: '', disabled: false }]

},

{ validator: [ConfirmEmailCheck] }

)

});

}

1. 此時前端singup-advance.component.html
   1. 前端只需要用formGroupName="emailGroup"來指定Group
   2. 其他內容部份就不用做多餘的更改，因為是在後端Group的

<div formGroupName="emailGroup">

<div class="form-group row">

<label for="staticEmail" class="col-sm-2 col-form-label">E-mail</label>

<div class="col-sm-10">

<input type="text" class="form-control" placeholder="Email (必填)" formControlName="email"/>

</div>

</div>

<div class="form-group row">

<label for="staticEmail" class="col-sm-2 col-form-label"

>Confirm E-mail</label

>

<div class="col-sm-10">

<input type="text" class="form-control" placeholder="Email (必填)" formControlName="confirmEmail"/>

</div>

</div>

</div>

1. 顯示部份就可以使用

{{ formData.get('emailGroup').errors | json }}