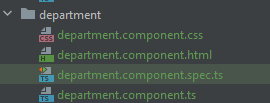
**ANGULAR**

1. **Tổng quan**
   * là một framework mã nguồn mở, JavaScript framework do Google tạo ra và phát triển để xây dựng nên các Single Page Application (SPA).
   * Được viết dựa trên typescript và được biên dịch sang javascript.
   * AngularJs lúc được tạo ra được viết dựa trên mô hình MVC
     + Model : Thành phần trung tâm, thể hiện hành vi của ứng dụng và quản lí dữ liệu.
     + View: được tạo ra dựa trên thông tin của model
     + Controller: đóng vai trò trung gian giữa model và view. Có vai trò xử lý logic.
   * Sau AngularJs thì Google đã phát triển và công bố thêm các version khác như :
     + Angular 2: 3/2015

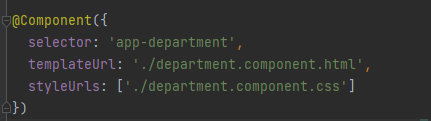
* thay thế nhiều thứ từ AngularJs
* Viết hoàn toàn bằng typescript
* Thay controller và $scope từ angular js thành component và directives
* Hỗ trợ đa nền tảng, đa trình duyệt , đơn giản, dễ học
* Angular 4: 3/2017
  + - Giảm thiểu code trong tệp, nhanh hơn so với angular 2
  + Angular 5: 1/11/2017 : có rất nhiều cải tiến so với angular 4
  + Angular 6: cập nhật CLI, element, multiple validator, Tree-shakeables providers, sử dụng RxJS 6 với syntax ko đổi
  + Angular 7: 18/10/2018 cập nhật Scrolling module , Drag and Drop
  + Angular 8: 28/5/2019 cập nhật CLI workflow improvements, Dynamic imports for lazy routers,..
  + Angular 9 : 6/2/2019.

**2.Components**

* Component là thành phần cơ bản của angular, mỗi component sẽ bao gồm các thành phần :
  + HTML template : 1 file HTML bao gồm phần hiển thị giao diện cho component đó.
  + CSS selector : đây là file css giúp trang trí và format giao diện cho HTML template của component
  + 1 class typescript: đây file được sử dụng để khai báo các thành phần của component và còn là file thực hiện chứa các biến dữ liệu của component.
  + Ngoài ra component còn có 1 số thành phần khác như:
    - Service : thực hiện xử lí logic cho component, thực thiện call api,…
    - Module : đây là nơi gói component, service,.. thành 1 phần 1 nhóm để có thể thực hiện export ra ngoài.
    - …

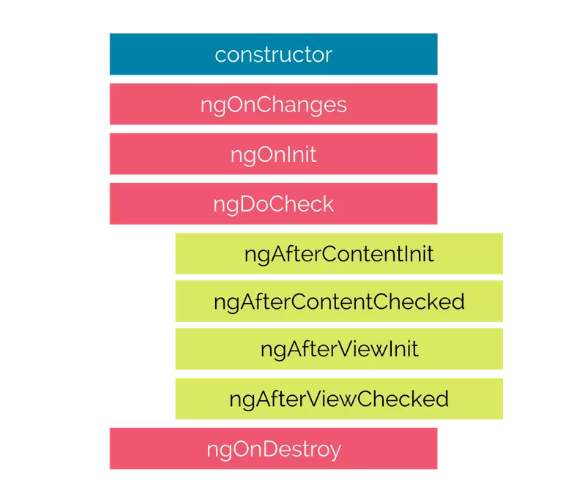
****

Trong đó file class.ts sẽ thực hiện khai báo các thông tin trong @Component



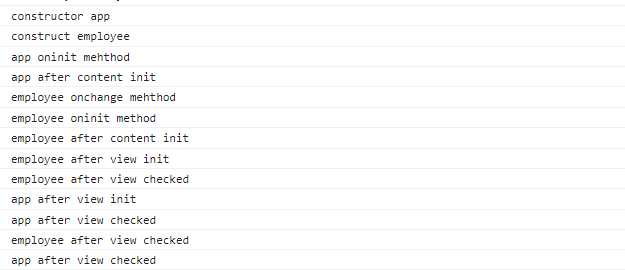
* Selector : tên định danh của component, sẽ được sử dụng để gọi đến ở HTML
* templateUrl : đường dẫn đến file html của component (có thể sử dụng template để thay thế)
* styleUrls : đường dẫn đến file css của component. (có thể sử dụng style để thay thế).
* **LifeCycle của component**

Vòng đời của 1 component trong angular được biểu diễn theo:

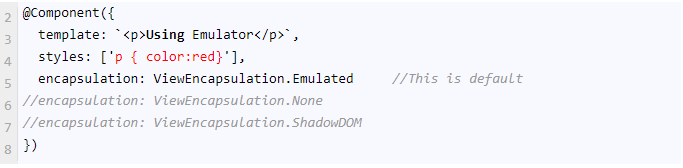


1. Constructor : hàm khởi tạo của component, hàm được gọi khi component được khởi tạo.
2. ngOnChanges : hàm được gọi khi component phát hiện có giá trị nào đó được biding vào component (thông qua html,…). Được gọi sau constructor.s
3. ngOnInit : được gọi khi component được tạo lần đầu tiền. được gọi sau contructor và ngOnChanges
4. ngDoCheck ; được gọi mỗi khi component phát hiện có ự thay đổi dữ liệu trong nó.
5. ngAfterContentInit: được gọi khi compoenent được tạo thành công
6. ngAfterContentChecked : được gọi khi angular kiếm tra xong nội dung của component.
7. ngAfterViewInit: tương tự AfterContentInit nhưng được gọi khi component đó và các component con của nó được tạo thành công.
8. ngAfterViewChecked : tương tự ngAfterContentChecked nhưng được gọi khi component đó và các component con của nó được kiểm tra xong.
9. ngOnDestroy: được gọi đến khi component hoặc derective bị phá bởi angular.

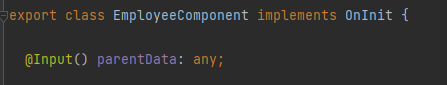
Vd : app component là component cha còn employee là component con.



* View Encapsulation : cung cấp các cách thức để angular có thể thực hiện đóng gói các CSS được xác định trong 1 component để nó không ảnh hưởng đến các component khác (các component cha hoặc component anh em). Gồm 3 lựa chọn :
  + None : không thực hiện đóng gói
  + Emulated (mặc định) : tạch biệt css giữa các thành phần với nhau
  + Shadow : đóng gói css của component. Tách biệt với các component khác.

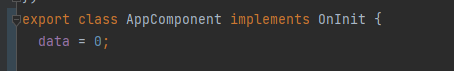


* Component Interaction : thực hiện tương tác giữa các component với nhau, xảy ra 4 trường hợp:
  + Component cha 🡺 component con : sử dụng @Input.
    - Component con :



* + - Component cha :

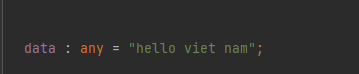




* + Component con 🡺 component cha : sử dụng ViewChild và AfterViewInit hoặc @Output và EventEmitter.
    - Sử dụng ViewChild và AfterViewInit

Component con:

Component.ts:



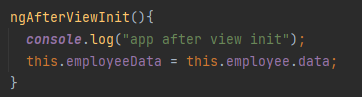
Component cha:

Html :



Component.ts:



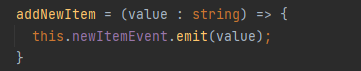


* + - Sử dụng @Ouput và EventEmitter

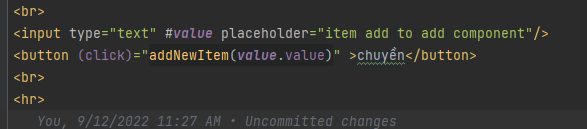
Component Con :

Component.ts





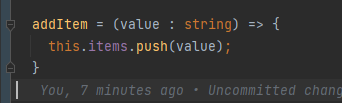
Html



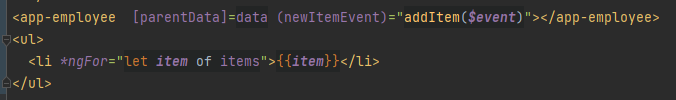
Component cha :

Component.ts





Html



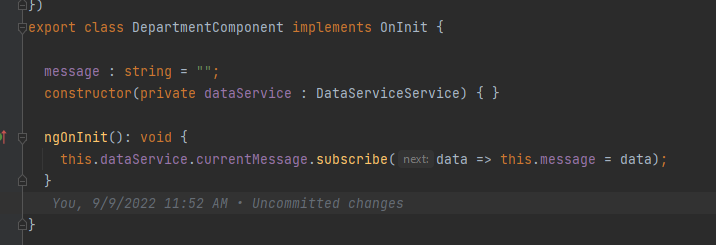
* + Component 🡺 component : sử dụng dataservice

Vd : tạo component dataservice : “ng g c dataservice”

Component.ts :

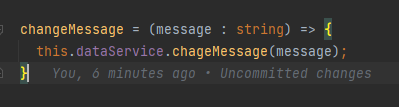


Component A :



Component B:

Component.ts



Html



* + Trong phạm vi component (từ class.ts sang file html) : sử dụng Interpolation (“ {{}} ”) để có thể truyền data.

Vd: component.ts

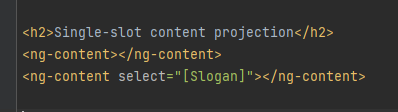


Html



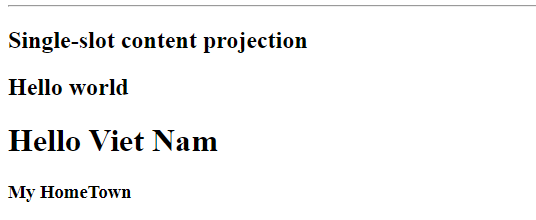
* + - Content Projection : nhúng các template HTML từ các component bên ngoài
      * Ng-content : truyền các html template từ component cha vào component con thì html template đó sẽ được render ở vị trí ng-content được đặt. Có thể dụng select để có thể chọn vị trí render khi thực hiện truyền nhiều template.

Vd : component con : html



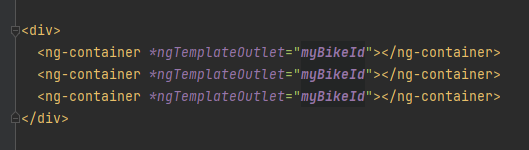
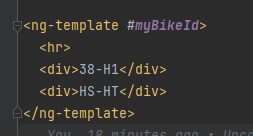
Component cha : html





* + - * Ng-template: là đối tượng có thể dụng với các directives (ngIf, ngFor,..). Chỉ hoạt động khi được chỉ định với directive điều kiện hiển thị.

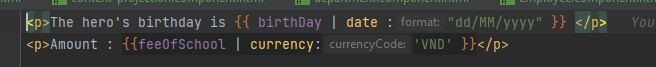


* + - * Ng-container là 1 template HTML được sử dụng để render tránh extra tag và thường được sử dụng cùng ngTemplateOutLet
      * ngTemplateOutlet : được sử dụng để đánh dấu các template html và có thể sử dụng lại nhiều lần.
    - Dynamic content : cung cấp cách để có thể thực hiện tải lên những component mới trong ứng dụng angular. Lập trình viên có thể xác định nơi mà component mới được chèn vào, và thường sử dụng <ng-template>.
    - Angular Elements : là các gói component angular hay còn được gọi là custom elements (web component). Tính năng này giúp tạo các thẻ html với nội dung được tạo và quản lí bơi r javascript.

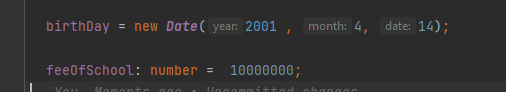
**3. Templates**

* là file html trong component, được sử dụng để biểu diễn giao diện, cá dữ liệu của trang web.
* Interpolation : các thức truyền dữ liệu trong phạm vi component (từ class.ts đến file html).
* Biding : Thực hiện truyền dữ liệu giữa file class.ts và file html. Có 2 loại biding dữ liệu chính dó là one way binding và two way biding.
  + One way biding : Thực hiện truyền dữ liệu 1 chiều, có thể là từ file html về file ts hoặc ngược lại. Mỗi khi dữ liệu thay đổi thì trang web sẽ refresh. Bao gồm các loại như
    - Test inpolations (thực hiện truyền dữ liệu)
    - Property biding (thực hiện truyền thuộc tính)
    - Event biding (thực hiện truyền 1 sự kiện nào đó , có thể là click, change , …).
  + Two way biding: thực hiện truyền dữ liệu 2 chiều, giá trị ở html gắn chặt với biến trong file component.ts. Khi giá trị ở file html thay đổi sẽ kéo theo sự thay đổi ở file ts và ngược lại mà web không refresh
* Pipe : là 1 tính năng giúp biến dữ liệu đầu vào/ra theo đúng chuẩn mà người lập trình chỉ định. (vd : DateTime, uppercase, …)
  + Pipe có múc độ ưu tiên cao hơn toán tử ternary (… ? … : …)
  + Có thể thực hiện sử dụng nhiều kiểu format cho pipe bằng cách cung cấp các kiểu định dạng sau dấu “ | “

**Vd :** html



Component.ts



* SVG Template: có thể sử dụng file SVG và thực hiện biding tương tự html template.

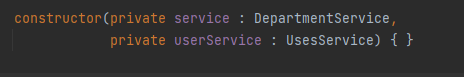
**4. Directives (chỉ thị)**

* Được sử dụng để quản lí form, list, style,…
* Có sử dụng để quản lí, viết các câu lệnh, điều kiện, vòng lặp,… với html
* Gồm 3 loại chính:
  1. Built-in Directives:
  + Built-in attribute directives:
    - Thực hiện nghe và sửa đổi các hành vi của các thành phần html, thuộc tính, và cả component.
    - Một số loại thường được sử dụng như :
      * ngClass: thêm , xóa css class
      * ngStyle: thêm , xóa css cho html
      * ngModel : thực hiện quản lí two way biding (form, input,…)
  + Structural directives:
    - Quản lí về hiển thị của html. (thực hiện thêm, xóa, thao tác với các thành phần html).
    - Một số loại thường được sử dụng:
      * ngIf : áp dụng khi cần viết câu lệnh if
      * ngFor : áp dụng khi cần sử dụng vòng for cho html
      * ngSwitch : áp dụng cho trường hợp sử dụng switch case trong html.
  1. Attribute directives:
* **@**Directive() : định nghĩa 1 directive thành css và có thể gán nó vào 1 tag html = selector của nó
* Cú pháp : ng generate directive <<ten directive>>
* Thực hiện thêm 1 directive vào 1 tag html tương tự 1 thuộc tính.
* Thực hiện set attribute directive với @Input thì có thể sử dụng nó như 1 thuộc tính html
  1. Structural Directive:
     + Đối với các ng-template thì sẽ sử dụng các directive trong dấu []
     + Còn đối với cá thẻ html bình thường khác sẽ được sử dụng sau dấu “\*”.

**5. Dependent Injection.**

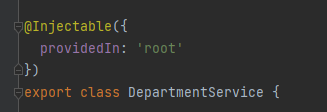
* Nguyên tắc Solid (single responsibility ,open/close , liskov subtitution , interface segregation, dependence inversion )
* Dependence Injection (DI) : là 1 design pattern cho phép chương trình có thể tránh được sự phụ thuộc giữa các thành phần với nhau. Đảm bảo tính chất thứ 5 của nguyên tắc solid (dependence inversion)
* DI sẽ giúp cho component có thể thực hiện inject các thành phần từ bên ngoài vào để có thể giúp cho component có thêm tính năng để sử dụng.
* Sử dụng contructor của component để có thể DI có thể thực hiện inject thêm các thành phần bên ngoài vào

Vd : trong componnet department



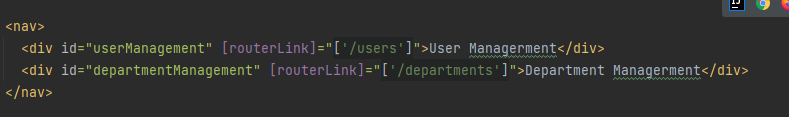
Sử dụng construct của department.component.ts để có thể inject thêm 2 thành phần bên ngoài là department.service.ts và user.serivce.ts để deparment có thêm các tính năng để sử dụng

Sử dụng @Injectable để có thể cung cấp cho DI biết được thành phần được đánh dấu đó có thể thực hiện inject vào 1 component khác.



**6.Router**

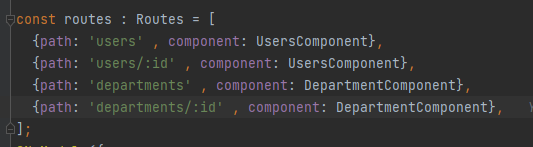
* Luôn là 1 thành phần quan trọng của 1 ứng dụng hoặc web vì nó cung cấp cách để ứng dụng có thể thực hiện chuyển hướng giữa các component
* Bên cạnh đó, việc truyền dữ liệu giữa các component cũng phải dựa vào router.
* Angular cung cấp rất nhiều thành phần có thể giúp cho ứng dụng có thể thực hiện việc router dễ hơn (Router , RouterModule)
* Sử dụng RouterLink để có thể thực hiện định dạng cho các thẻ html biết đường dẫn đi đến các component.

****

* Sử dụng Router-outlet để khai báo vị trí mà component được chi định bởi đường dẫn chèn vào

****

* Thực hiện config path trước để có thể thực hiện chuyển đường dẫn đến đúng component và truyền dữ liệu (id,…)

****

* Có thể sử dụng snapshot, paramMap, … để có thể lấy ra được dữ liệu khi thực hiện truyền qua url
* Mỗi path nên có 1 unique title đẻ có thể dễ dàng phân biệt trong lịch sử trình duyệt
* Lazy loading : là 1 tính năng có thể set cho angular chỉ load các module cần thiết.
* Route guards : là 1 tính năng thực hiện nang các user có thể truy cập vào app mà chưa có anthorization
* RouterModule.forRoot() : là 1 singleTon services đảm bảo rằng trong ứng dụng chỉ tồn tại 1 instance của routerModule.
* Ngoài sử dụng routerLink để khai báo đường dẫn chuyển trang và thực hiện chuyển thì còn có thể sử dụng navigate để thực hiện chuyển trang bằng function.

**7. Form**

* Gồm 2 loại form chính :
  + Reactive : đây là dạng thường được áp dụng cho form lớp, có nhiều tính năng , đồng bộ
  + Template-driven: phù hợp với các loại form nhỏ , đơn giản (đăng nhập,…), bất đồng bộ
  + Để thao tác với form , chúng ta thực có thể sử dụng :
    - Sử dụng FormBuilder để khởi tạo form, và định các giá trị của nó trong component.
    - FormControl : theo dõi các giá trị trong form và thực hiện validation form
    - FormGroup : theo dõi các giá trị giống nhau và trạng thái của các collection trong form
    - FormArray : theo dõi các giá trị giống nhau và trạng thái của các mảng trong form
    - ControlValueAccessor: tạo cầu nối giữa formcontrol và thẻ form dom.

Data Flow của reactive form:

* + - 1. User truyền dữ liệu vào thẻ input trong form
      2. Thẻ input sẽ gửi đi giá trị nhập vào cuối cũng đến controlValueAccessor
      3. controlValueAccessor sẽ gửi sự 1 sự kiện chữa giá trị inout đến formControl.
      4. FormControl sẽ thực hiện setValue
      5. Và formControl sẽ gửi giá trị đó đén với component sử dụng

Data Flow của template-driven form

1. User truyền dữ liệu vào thẻ input trong form
2. Thẻ input gửi sự kiện chứa giá trị nhập vào đén controlValueAccessor.
3. controlValueAccessor sẽ thực hiện gán giá trị vào hàm setValue của formControl.
4. component đăng kí nhận giá trị sẽ vào valueChange để nhận giá trị

* có thể thực hiện validation form bằng cách validation từng trường input trong form bằng các validation built-in (leng, required, max, min,…) hoặc tự define để sử dụng

1. controlValueAccessor sẽ gọi đến ngModel.viewToModelUpdate để thực hiện cập nhật giá trị input và gửi đến component nhận

Đối với template-driven form sử dụng two waybiding nên giá trị cũng sẽ cập nhật theo ngModel