**Tran Cao Minh PS18817 – LAB 8 – Angular**

**Contents**

[# 1. Summary ~ Angular Framework 1](#_Toc106461042)

[1. Angular? 1](#_Toc106461043)

[2. Environment 1](#_Toc106461044)

[3. Install 1](#_Toc106461045)

[4. Organize 1](#_Toc106461046)

[5. Structure 1](#_Toc106461047)

[6. Style 2](#_Toc106461048)

[7. Angular from Typescript? 2](#_Toc106461049)

[# 2. Component 2](#_Toc106461050)

[1. Component? 2](#_Toc106461051)

[2. Data Binding 3](#_Toc106461052)

[3. Directive 3](#_Toc106461053)

[# 3. Databinding 3](#_Toc106461054)

[1. Binding 3](#_Toc106461055)

[2. Component mount 3](#_Toc106461056)

[3. Lifecycle 4](#_Toc106461057)

[# 4. Routing 4](#_Toc106461058)

[1. Overview 4](#_Toc106461059)

[2. Setup 4](#_Toc106461060)

[3. Declare and use router 4](#_Toc106461061)

[4. Protect Router 4](#_Toc106461062)

[# 5. Form 4](#_Toc106461063)

[1. Template-driven 4](#_Toc106461064)

[2. Reactive 5](#_Toc106461065)

[# 6. Service - HTTP Service Module 6](#_Toc106461066)

[1. Overview 6](#_Toc106461067)

[2. HTTP Service Module 6](#_Toc106461068)

[# 7. Authentication 6](#_Toc106461069)

[1. How it active? 6](#_Toc106461070)

[2. Use Guard in Authentication 6](#_Toc106461071)

[# 8. Module 6](#_Toc106461072)

[1. Overview 6](#_Toc106461073)

[2. Lazy Loaded Module 7](#_Toc106461074)

# # 1. Summary ~ Angular Framework

## 1. Angular?

\_ JS framework

\_ build app - client-side (Front-End)

\_ use HTML, CSS, JS

\_ advantage:

  \_ separated HTML

  \_ good binding

  \_ module structure

  \_ work well ~ Back-End

## 2. Environment

\_ VS Code

\_ Windows Terminal

\_ node JS -> npm

## 3. Install

\_ in Terminal

\_ > npm i -g @angular/cli

\_ > ng new projectName

\_ > ng server --o

## 4. Organize

\_ folderName/src: app source code

\_ src/app: components

\_ src/assets: resources like images, videos,...

## 5. Structure

--------------    -----------------

| index.html | -> | App Component |

--------------    -----------------

                    ^      |

         |----------|      |

         |                 V

         |         --------------------

         |   |---> | Child Components |

         |   |     --------------------

         |   |               |

--------------------         |

| Service, Guard,...|         |

--------------------         V

           |      ------------------------

           |----> | Sub-child Components |

                  ------------------------

## 6. Style

\_ config: folderName/angular.json

~ {}project > {}folderName >

  {}architect > {}build >

  {}option > []style

\_ # can install style libraries and

    define path from node\_modules folder

## 7. Angular from Typescript?

\_ more extend function than JavaScript

  \_ type for variant

  \_ interface

\_ combine to JS

**# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

# # 2. Component

## 1. Component?

\_ = Template + (Properties, Methods) + Metadata

\_ make view layout

\_ create with HTML (directives + data binding)

\_ ex:

\_--------------------------------------------------

  import { Component } from '@angular/core';

  @Component({

    selector: 'app-root',

    templateUrl: './app.component.html',

    styleUrls: ['./app.component.css'],

  })

  export class AppComponent {

    pageTitle: string = 'Tran Cao Minh';

  }

\_--------------------------------------------------

## 2. Data Binding

\_ component <=> template

\_ event: (eventName)="funcName(var)"

\_ value: {{ varName/function/expression => return value }}

\_ data mapping: [(ngModel)]="mappingVar"

## 3. Directive

\_ expand attributes (Angular provide)

\_ help develop website easier

\_ ex:

\_--------------------------------------------------

  \*ngIf='condition'

  \*ngFor='let item of itemList'

  ( for-of  get value / for-in get attribute )

\_--------------------------------------------------

**# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

# # 3. Databinding

## 1. Binding

\_ Property

  <tag [attrName]='componentVar'></tag>

\_ Event

  <tag (eventName)='funcName(var)'></tag>

\_ Two-Way

  <tag [(ngModel)]='componentVar'></tag>

\_ Value

  {{ componentVar | pipesFunc[uppercase, date,...] }}

## 2. Component mount

\_ Transfer data -> components

  in child component use: @Input

\_ components -> data

  in child component use: @Output + EventEmitter

\_ parent -> child

  in parent use: @ViewChild('name') referenceVar

## 3. Lifecycle

\_ some common angular lifecycle function

\_--------------------------------------------------

  ngOnChanges: execute every time input data change

  ngOnInit: execute once time when init component

  ngDoCheck: execute when discover change

  ...

\_--------------------------------------------------

**# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

# # 4. Routing

## 1. Overview

- feature allow navigate URL -> specific page in app

- in Angular -> a component

## 2. Setup

\_ ng generate module app-routing --flat --module=app

\_ or when create

  ng new appName --routing --defaults

## 3. Declare and use router

\_ in app-routing.module.ts declare routes

\_ navigate with routerLink attr

\_ view content in <router-outlet></router-outlet>

## 4. Protect Router

\_ use Service and Guard

\_ ng g s serviceName + ng g g guardName

\_ declare guard in app-routing.module.ts

\_ guard return true/false

**# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

# # 5. Form

\_ Two common way

## 1. Template-driven

  \_---------------------------------------------------------------

  \_ create form by input code HTML and directive angular in view

  \_ step

    \_ dip formsModule in app.modules.ts

    \_ create form with HTML

    \_ declare ngForm and mount ngModel in form controls (name is required)

    \_ get form value (form.value.field)

    \_ validation by HTML validate attributes (required, minlength, ...)

    \_ declare check error var #errVar="ngModel"

    \_ view error {{ errVar.invalid }} {{ errVar.errors?.['required

    ] | json }}

    \_ some angular helper class - ng-valid / ng-invalid

    \_ submit by (ngSubmit)="handleFunc(formName.value)"

  \_---------------------------------------------------------------

## 2. Reactive

  \_---------------------------------------------------------------

  \_ import module reactiveFormsModule in app.module.ts

  \_ element in form reactive

    \_ formControl (HTML form control - input, select, textarea)

    \_ formGroup (group objects formControl, formGroup, formArray)

    \_ formArray (array of objects)

  \_ import FormControl, FormGroup

  \_ binding control [formGroup], formControlName="formControlName"

  \_ get value like template-driven form

  \_ syntax (validate here):

    \_ new FormGroup({

        username: new FormControl('defaultValue', [

          Validators.minLength(3),

          Validators.maxLength(10),

        ]),

      })

  \_---------------------------------------------------------------

**# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

# # 6. Service - HTTP Service Module

## 1. Overview

\_ Angular communicate with Back-End via Service

\_ clear, clean code

\_ way to share information through separated class

\_ create: ng g s serviceName

## 2. HTTP Service Module

\_ restful APIs: GET, POST, PUT, DELETE

\_ use .subscribe to get result

**# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

# # 7. Authentication

## 1. How it active?

\_ step 1: client send authentication data

\_ step 2: server get data and handle

\_ step 3: if data is verify -> send TOKEN

\_ step 4: client save and use TOKEN

## 2. Use Guard in Authentication

\_ decide depend on TOKEN expires time

\_ use Moment to code faster (check expires time)

\_ import and use in app-routing.module.ts

**# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

# # 8. Module

## 1. Overview

\_ manage app, feature

\_ characteristic modules

  \_ built-in: module in app, ready to use

  \_ lazy-loaded: improve performance and user experience

  \_ routing: navigate, add guard

  \_ shared: directive, pipe

## 2. Lazy Loaded Module

\_ only load needed module

\_ use like call back function

\_ ex:

  {

    path: 'admin',

    loadChildren: () =>

      import('./admin.module').then(m => m.AdminModule)

  },

**# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**