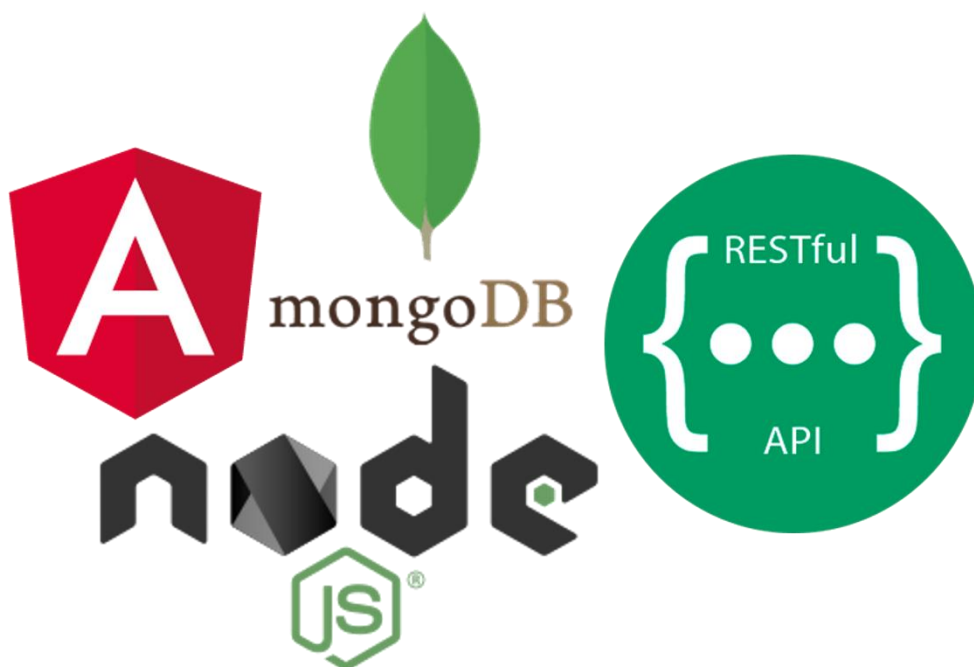


# Hướng Dẫn Thực Hành

## Phát triển Web Kinh Doanh nâng cao

Khối: Đại Học



### Hướng dẫn:

- Bài tập thực hành được chia làm nhiều Module
- Mỗi Module được thiết kế cho thời lượng là 3 hoặc 6 tiết thực hành tại lớp với sự hướng dẫn của Giảng viên.
- Tùy theo số tiết phân bổ, mỗi tuần học có thể thực hiện nhiều Module.
- Sinh viên phải làm tất cả các bài tập trong các Module ở tuần tương ứng. Những sinh viên chưa hoàn tất phần bài tập tại lớp có trách nhiệm tự làm tiếp tục ở nhà.
- Các bài có dấu (\*) là các bài tập nâng cao dành cho sinh viên khá giỏi.

Biên soạn: **TS. Trần Duy Thanh**

## MỤC LỤC

Module 1: AngularJS .....	3
Bài 1 – Kiến trúc thành phần của Angular.....	3
Bài 2 – Cài đặt và lập trình Angular .....	3
Bài 3 - Binding .....	4
Bài 4 – Binding Property .....	4
Bài 5 – Binding Class.....	4
Bài 6 – Binding Style .....	5
Bài 7 – Binding Event.....	6
Bài 8 – Binding Two-Way .....	7
Bài 9 - Binding Two-Way, model for Quadratic Equation .....	7
Bài 10- Binding Two-Way, model for Lunar Year (*) .....	8
Bài 11- Json Object Model - Product .....	9
Bài 12- Json Array Model - Product.....	9
Bài 13- Json Array Model – Product Event (*).....	10
Bài 14- Json Array Model – Product - Catalog.....	14
Bài 15- Json Array Model – Product– Http Service(*).....	15
Bài 16- Json Array Model – Product– Http Service Handle Error (*) .....	17
Bài 17- Json Object Model – Customer – Service.....	19
Bài 18- Json Array Model – Group Customers (*) .....	19
Bài 19 – Routing for Page Not Found .....	21

## Module 1: AngularJS

### Nội dung kiến thức thực hành:

- + Cấu hình và cài đặt NodeJS, AngularJS
- + Thực hành các lệnh của AngularJS: Tạo dự án, chạy dự án, tạo component, tạo service
- + Tìm hiểu cấu trúc thành phần của một Project AngularJS
- + Lập trình Binding: Binding, Binding Property, Binding Style, Binding Event, Binding Two Way
- + Lập trình Type Script
- + Lập trình built-in directives: ngIf, ngSwitch, ngFor
- + Lập trình Component
- + Lập trình Service
- + Lập trình Routing
- + Lập trình Form, handling error
- + Và các kỹ năng thực hành khác

### ***Bài 1 – Kiến trúc thành phần của Angular***

#### **Yêu cầu:**

Hãy trình bày kiến trúc thành phần của Angular, giải thích chức năng của từng thành phần (ví dụ App, Module, Component, Service...).

### ***Bài 2 – Cài đặt và lập trình Angular***

#### **Yêu cầu:**

Hãy trình bày cách thức cài đặt AngularJS

Trình bày các lệnh tạo dự án, chạy dự án, tạo component, tạo service

### **Bài 3 - Binding**

#### **Yêu cầu:**

Tạo một Component tên “MyComponent”. File Type Script (.ts) chứa một biến tên là “myVar” có giá trị “Hello Angular”, hãy viết các lệnh binding để hiển thị dữ liệu lên giao diện (html) bằng 2 cách:

- Truy suất trực tiếp vào biến “myVar”
- Viết hàm getMyVar() trong (.ts) trả giá trị của “myVar”

File (html) truy suất và hiển thị giá trị của “myVar” trong (.ts) như sau:

- Hiển thị dữ liệu gốc của biến “myVar”
- Hiển thị toàn bộ chữ in hoa của “myVar”
- Hiển thị toàn bộ chữ in thường của “myVar”

### **Bài 4 – Binding Property**

#### **Yêu cầu:**

Tạo một “BindingPropertyComponent”, Type Script (.ts) chứa các Property (Attribute/Variable) sau:

```
export class BindingPropertyComponent {  
  public name:string="Trần Duy Thanh"  
  public email:string="thanhtd@uel.edu.vn"  
  public nameid:string="nameid"  
  public emailid:string="emailid"  
  public isDisabled:boolean=true  
}
```

HTML sử dụng 2 cách: tạo cấu trúc HTML trong .ts và cấu trúc HTML trong file HTML để binding các thuộc tính này. Sử dụng cú pháp `[]` để binding cho name và `{{}}` để binding cho email. Thuộc tính isDisabled sẽ thiết lập cho Email, sinh viên thay đổi true hoặc false để quan sát kết quả.

### **Bài 5 – Binding Class**

#### **Yêu cầu:**

Tạo một “BindingClassComponent”.

File “binding-class.component.css” định nghĩa các class sau:

```
.text-success{  
    color:darkgreen  
}  
.text-error{  
    color:darkred  
}  
.text-primary{  
    color:navy  
}  
.text-normal{  
    font-style: italic;  
}
```

File “binding-class.component.ts” khai báo:

```
export class BindingClassComponent {  
    public success="text-success"  
    public error="text-error"  
    public primary="text-primary"  
    public normal="text-normal"  
    public multiClass={  
        "text-primary":true,  
        "text-normal":true,  
        "text-error":true  
    }  
}
```

Yêu cầu, file “binding-class.component.html” tạo 5 thẻ <h1> dùng cú pháp [] để binding các class, tạo 5 thẻ <h2> dùng cú pháp {{ }} để binding các class. Nội dung hiển thị trong mỗi <h1> và <h2> là tùy ý.

Sau đó định nghĩa một class “.text-complexity” định dạng chữ Đậm, In Nghiêng, Font chữ “Cambria”, cỡ chữ 18px, màu chữ Blue. Tiến hành định nghĩa tên biến trong .ts và binding trong html để sử dụng class “.text-complexity” này.

## **Bài 6 – Binding Style**

### **Yêu cầu:**

Tạo một “BindingStyleComponent” component.

Định nghĩa Type Script (.ts) như sau:

```
export class BindingStyleComponent {  
    public isError:boolean=false  
    public textStyle={
```

```
color: 'darkorange',  
fontSize: '26px'  
}  
}
```

Trong html binding style theo cú pháp sau:

```
<h1 style="color:red;">Trần Duy Thanh</h1>  
<h1 [style.color]="isError? 'red': 'darkgreen'">Trần Phạm Thanh Trà</h1>  
<h1 [style]="{'color': 'purple', 'font-size': '26px'}">Trần Phạm Mẫn Nhi</h1>  
<h1 [style]="textStyle">Phạm Thị Xuân Diệu</h1>
```

Chạy và quan sát kết quả.

Mở rộng: Bổ sung style cho chữ tự động in Hoa toàn bộ. Gán binding và kiểm tra kết quả.

## **Bài 7 – Binding Event**

### **Yêu cầu:**

Tạo một “BindingEventComponent” component.

File Type Script “binding-event.component.ts” định nghĩa 2 sự kiện:

```
export class BindingEventComponent {  
  onClick(event:any){  
    alert(event.pointerId)  
  }  
  onSubmit(value:string){  
    alert(value)  
  }  
}
```

File “binding-event.component.html” định nghĩa và binding event:

```
<p>binding-event works!</p>  
<button (click)="onClick($event)">Click Me</button>  
Name:  
<input type="text" #myName>  
<button (click)="onSubmit(myName.value)">Submit</button>
```

Chạy component và quan sát kết quả.

Mở rộng: Chỉnh sửa giao diện cho người dùng nhập 2 số a, b bất kỳ và 5 button:

Button cộng: xử lý xuất kết quả tổng a+b

Button trừ: xử lý xuất kết quả hiệu a-b

Button nhân: xử lý xuất kết quả tích  $a*b$

Button chia: xử lý xuất kết quả thương  $a/b$ , lưu ý xử lý mẫu số  $=0$

Button xóa trắng: Xóa dữ liệu trên giao diện.

Chạy component để kiểm tra kết quả sau khi chỉnh sửa.

## **Bài 8 – Binding Two-Way**

### **Yêu cầu:**

Tạo một component “BindingTwoWayComponent”.

File “binding-two-way.component.ts” định nghĩa:

```
export class BindingTwoWayComponent {  
  public name:string=''  
  public address:string=''  
  setDefaultAddress(){  
    this.address='13 đường Hùng Vương'  
  }  
}
```

File “binding-two-way.component.html” định nghĩa:

```
<p>binding-two-way works!</p>  
<p>Name: <input [(ngModel)]="name" type="text"></p>  
<p>Your name:{{name}}</p>  
<p>Address:<input [(ngModel)]="address" type="text"></p>  
<p>Your address:{{address}}</p>  
<button (click)="setDefaultAddress()">Set default</button>
```

Chạy component và quan sát kết quả.

Mở rộng bổ sung email, số điện thoại, tiến hành binding và chạy lại component, quan sát kết quả sau khi thay đổi.

## **Bài 9 - Binding Two-Way, model for Quadratic Equation**

### **Yêu cầu:**

Tạo một component có giao diện HTML như sau:

Giải phương trình bậc 2 online	
Hệ số a:	<input type="text" value="1"/>
Hệ số b:	<input type="text" value="3"/>
Hệ số c:	<input type="text" value="-4"/>
Kết quả:	<b>x1=-4 ; x2=1</b>
<input type="button" value="Giải phương trình"/> <input type="button" value="Xóa trắng"/>	

- Dùng Binding Two-way để mapping giá trị cho các biến trong Type Script và Html.
- Tạo một class tên là “Quadratic” nhận vào 3 hệ số a, b,c. và có một phương thức “findSolution()” để xử lý giải phương trình bậc 2
- Khi nhấn nút “Giải phương trình” thì Type Script sẽ áp dụng cơ chế binding để nhận các giá trị nhập vào từ giao diện Html, sau đó dùng lớp “Quadratic” để giải phương trình, sau khi giải xong thì trả về kết quả lên giao diện html.
- Khi nhấn nút “Xóa trắng” thì xóa các dữ liệu trên các ô nhập liệu và dòng kết quả.

### **Bài 10- Binding Two-Way, model for Lunar Year (\*)**

#### **Yêu cầu:**

Tạo một component có giao diện như dưới đây:

Chuyển đổi Dương lịch thành Âm lịch	
<input type="text" value="15"/> Ngày	<input type="text" value="5"/> Tháng <input type="text" value="1986"/> Năm <input type="button" value="Chuyển đổi"/>
Âm Lịch	
Thứ trong tuần	Ngày thứ 5
Ngày tháng năm âm	7/4/1986
Năm	Bính Dần
Tháng	Quý Tỵ
Ngày	Kỷ Mùi

Người dùng lựa chọn Ngày tháng năm sinh trên giao diện (dùng select/dropdownlist). Bấm nút “Chuyển đổi”, chuyển ngày dương thành ngày âm.

Yêu cầu dùng cơ chế binding để nạp dữ liệu lên dropdownlist. Viết class “LunarYear” nhận vào 3 thông số (ngày, tháng, năm dương lịch) và hàm



“findLunarYearDetail()” trả về thông tin chi tiết như trên giao diện (nên tạo 4 thuộc tính để lưu dữ liệu chi tiết cho LunarYear).

### **Hướng dẫn:**

Google search để biết cách chuyển 1 năm Dương lịch thành Âm lịch.

Binding dữ liệu từ mảng lên dropdownlist. Ví dụ ta có:


```
days=["1","2","3"]
<select class="form-select" id="days" name="days">
  <option *ngFor="let day of days">{{day}}</option>
</select>
```

Google Search thêm để biết cách binding lấy dữ liệu trên dropdownlist mà người dùng chọn.

## **Bài 11- Json Object Model - Product**

### **Yêu cầu:**




Tạo một Component để hiển thị sản phẩm như hình bên dưới. Yêu cầu dùng JsonObject và cơ chế binding. Dữ liệu khai báo trong Type Script.

Product ID:	123
Product Name:	Thuốc Trị Xâm
Price:	300
	

## **Bài 12- Json Array Model - Product**

### **Yêu cầu:**

Tạo một Component để hiển thị danh sách sản phẩm như hình bên dưới. Yêu cầu dùng JsonArray và cơ chế binding. Dữ liệu khai báo trong Type Script.

Ma San Pham	Ten San Pham	Gia San Pham	Picture
p1	Coca	100	
p2	Pepsi	300	
p3	Sting	200	

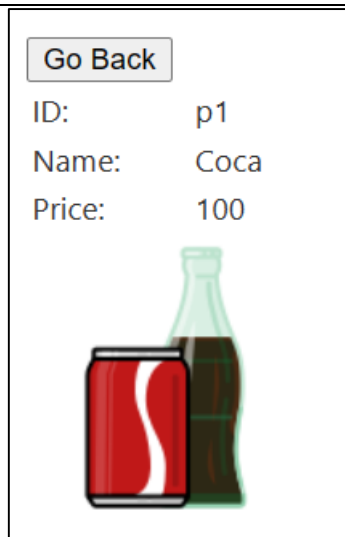
### ***Bài 13- Json Array Model – Product Event (\*)***

#### **Yêu cầu:**

Tạo một Component để hiển thị danh sách sản phẩm như hình bên dưới. Yêu cầu dùng JsonArray và cơ chế binding. Dữ liệu khai báo trong Service.

Ma San Pham	Ten San Pham	Gia San Pham	Picture	#
p1	Coca	100		Details
p2	Pepsi	300		Details
p3	Sting	200		Details

Nhấn vào details sẽ hiển thị chi tiết của sản phẩm đó:



Nhấn “Go Back” sẽ quay lại màn hình danh sách sản phẩm.

### Hướng dẫn:

- Tạo ProductService:

```
export class ProductService {
  productsImage=[
    {"ProductId": "p1", "ProductName": "Coca", "Price": 100, "Image": "assets/h1.png"},
    {"ProductId": "p2", "ProductName": "Pepsi", "Price": 300, "Image": "assets/h2.png"},
    {"ProductId": "p3", "ProductName": "Sting", "Price": 200, "Image": "assets/h3.png"},
  ]
  constructor() { }
  getProductsWithImages()
  {
    return this.productsImage
  }
  getProductDetail(id:any){

    return this.productsImage.find(x=>x.ProductId==id)
  }
}
```

- Các hình ảnh sao chép vào assets
- Tạo component “service-product-image-event”:

+ Viết lệnh cho “service-product-image-event.component.ts”:

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { ProductService } from '../product.service';

@Component({
  selector: 'app-service-product-image-event',
  templateUrl: './service-product-image-event.component.html',
  styleUrls: ['./service-product-image-event.component.css']
})
```

```
export class ServiceProductImageEventComponent {
  public products:any
  constructor(pservice: ProductService,private router:Router){
    this.products=pservice.getProductsWithImages()
  }
  viewDetail(f:any)
  {
    this.router.navigate(['service-product-image-event',f.ProductId])
  }
}
```

+Viết lệnh cho “service-product-image-event.component.html”:

```
<p>service-product-image-event works!</p>
<table border="1">
  <tr>
    <td>Ma San Pham</td>
    <td>Ten San Pham</td>
    <td>Gia San Pham</td>
    <td>Picture</td>
    <td>#</td>
  </tr>
  <tbody *ngFor="let p of products">
    <tr>
      <td>{{p.ProductId}}</td>
      <td>{{p.ProductName}}</td>
      <td>{{p.Price}}</td>
      <td>
        
      </td>
      <td>
        <a style="cursor:pointer;" (click)="viewDetail(p)" title="Click
here to view detail">Details</a>
      </td>
    </tr>
  </tbody>
</table>
```

- Tạo component “service-product-image-event”:

+ viết lệnh cho “service-product-image-event-detail.component.ts”:

```
import { Component } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { ProductService } from '../product.service';

@Component({
  selector: 'app-service-product-image-event-detail',
  templateUrl: './service-product-image-event-detail.component.html',
  styleUrls: ['./service-product-image-event-detail.component.css']
})
```

```
export class ServiceProductImageEventDetailComponent {
  selectedProduct:any
  constructor(private activateRoute:ActivatedRoute,private _fs:ProductService,
private router:Router)
  {
    activateRoute.paramMap.subscribe(
      (param)=>{
        let id=param.get('id')

        if(id!=null)
        {
          this.selectedProduct=_fs.getProductDetail(id)
        }
      }
    )
  }
  goBack(){
    this.router.navigate(['service-product-image-event'])
  }
}
```

+ viết lệnh cho “service-product-image-event-detail.component.html”:

```
<p>service-product-image-event-detail works!</p>
<button (click)="goBack()">Go Back</button>
<table>
  <tr>
    <td>ID:</td>
    <td>{{selectedProduct.ProductId}}</td>
  </tr>
  <tr>
    <td>Name:</td>
    <td>{{selectedProduct.ProductName}}</td>
  </tr>
  <tr>
    <td>Price:</td>
    <td>{{selectedProduct.Price}}</td>
  </tr>
  <tr>
    <td colspan="2">
      <img [src]="selectedProduct.Image"/>
    </td>
  </tr>
</table>
```

- Cấu hình Routing:

```
{path:'service-product-image-event',
component:ServiceProductImageEventComponent},
{path:'service-product-image-event/:id',
component:ServiceProductImageEventDetailComponent},
```

## **Bài 14- Json Array Model – Product - Catalog**







### **Yêu cầu:**

Cho CatalogService có cấu trúc như dưới đây:

```
export class CatalogService {
  datas=[
    {"Cateid":"cate1","CateName":"nuoc ngot",
      "Products":[
        {"ProductId":"p1","ProductName":"Coca","Price":100,
          "Image":"assets/h1.png"},
        {"ProductId":"p2","ProductName":"Pepsi","Price":300,
          "Image":"assets/h2.png"},
        {"ProductId":"p3","ProductName":"Sting","Price":200,
          "Image":"assets/h3.png"},
      ]
    },
    {"Cateid":"cate2","CateName":"Bia",
      "Products":[
        {"ProductId":"p4","ProductName":"Heleiken","Price":500,
          "Image":"assets/h4.png"},
        {"ProductId":"p5","ProductName":"333","Price":400,
          "Image":"assets/h5.png"},
        {"ProductId":"p6","ProductName":"Sai Gon","Price":600,
          "Image":"assets/h6.png"},
      ]
    },
  ]

  constructor() { }
  getCategories()
  {
    return this.datas
  }
}
```

- Tạo component và dùng ngFor lồng nhau để hiển thị danh mục và sản phẩm như hình dưới đây:

Ma Danh Muc		Ten Danh Muc	
cate1		nuoc ngot	
Ma San Pham	Ten San Pham	Gia San Pham	Picture
p1	Coca	100	
p2	Pepsi	300	
p3	Sting	200	
cate2		Bia	
Ma San Pham	Ten San Pham	Gia San Pham	Picture
p4	Heleiken	500	
p5	333	400	
p6	Sai Gon	600	




### Bài 15- Json Array Model – Product– Http Service(\*)

#### Yêu cầu:

Tạo một cơ sở dữ liệu json trong “assets/data/products.json” có cấu trúc mẫu:

```
[
  {"ProductId": "p1", "ProductName": "Coca", "Price": 100, "Image": "assets/h1.png"},
  {"ProductId": "p2", "ProductName": "Pepsi", "Price": 300, "Image": "assets/h2.png"},
  {"ProductId": "p3", "ProductName": "Sting", "Price": 200, "Image": "assets/h3.png"}
]
```

Hãy dùng cơ chế Service dạng Http để hiển thị dữ liệu như hình:

Ma San Pham	Ten San Pham	Gia San Pham	Picture
p1	Coca	100	
p2	Pepsi	300	
p3	Sting	200	

## Hướng dẫn:

-Tạo Interface:

```
export interface IProduct{  
  ProductId:string,  
  ProductName:string,  
  Price:number,  
  Image:string  
}
```

(Có thể để trống toàn bộ các property):

```
export interface IProduct{  
  
}
```

-Tạo “ProductHttpService”:

```
export class ProductHttpService {  
  
  private _url:string="./assets/data/products.json";  
  constructor(private _http: HttpClient) { }  
  
  getProducts():Observable<IProduct[]>{  
    return this._http.get<IProduct[]>(this._url)  
  }  
}
```

-Tạo component “service-product-http”:

+ File “service-product-http.component.ts”:

```
export class ServiceProductHttpComponent {  
  products:any;  
  constructor(private _service: ProductHttpService){  
    this._service.getProducts().subscribe({  
      next:(data)=>{this.products=data}  
    })  
  }  
}
```

+ File “service-product-http.component.html”:

```
<p>service-product-http works!</p>  
<table border="1">  
  <tr>  
    <td>Ma San Pham</td>  
    <td>Ten San Pham</td>  
    <td>Gia San Pham</td>
```



```
<td>Picture</td>
</tr>
<tbody>
  <tr *ngFor="let p of products">
    <td>{{p.ProductId}}</td>
    <td>{{p.ProductName}}</td>
    <td>{{p.Price}} </td>
    <td>
      
    </td>
  </tr>
</tbody>
</table>
```

### ***Bài 16- Json Array Model – Product– Http Service Handle Error (\*)***

#### **Yêu cầu:**

Tương tự như bài 15, tuy nhiên bài 16 xử lý lỗi ngoại lệ xảy ra. Ví dụ như cơ sở dữ liệu không tồn tại.

#### **Hướng dẫn:**

-Bổ sung “ProductHttpService”, và cố tình sửa sai đường dẫn dữ liệu:

```
export class ProductHttpService {

  private _url:string="./assets/data/productsXXX.json";
  constructor(private _http: HttpClient) { }

  getProducts():Observable<IProduct[]>{
    return this._http.get<IProduct[]>(this._url)
  }
  getProductsHandleError()
  {
    return this._http.get<IProduct[]>(this._url)
      .pipe(retry(3),
        catchError(this.handleError))
  }

  handleError(error:HttpErrorResponse){
    return throwError(()=>new Error(error.message))
  }

}
```

-Tạo component “service-product-http-handle-error”:

+ File “service-product-http-handle-error.component.ts”:

```
export class ServiceProductHttpHandleErrorComponent {
  products:any
  errMessage:string=''
  constructor(_service:ProductHttpService){
    _service.getProductsHandleError().subscribe({
      next:(data)=>{this.products=data},
      error:(err)=>{this.errMessage=err}
    })
  }
}
```

+ File “service-product-http-handle-error.component.html”:

```
<p>service-product-http-handle-error works!</p>
{{errMessage}}
<table border="1">
  <tr>
    <td>Ma San Pham</td>
    <td>Ten San Pham</td>
    <td>Gia San Pham</td>
    <td>Picture</td>
  </tr>
  <tbody>
    <tr *ngFor="let p of products">
      <td>{{p.ProductId}}</td>
      <td>{{p.ProductName}}</td>
      <td>{{p.Price}}</td>
      <td>
        
      </td>
    </tr>
  </tbody>
</table>
```

Chạy lên ta có kết quả (trường hợp ta đã sửa sai đường dẫn cơ sở dữ liệu):

service-product-http-handle-error works!

Error: Http failure response for http://localhost:4200/assets/data/products1.json: 404 Not Found

Ma San Pham	Ten San Pham	Gia San Pham	Picture
-------------	--------------	--------------	---------

Nếu sửa lại cho đúng thì ta có kết quả như bài 15.

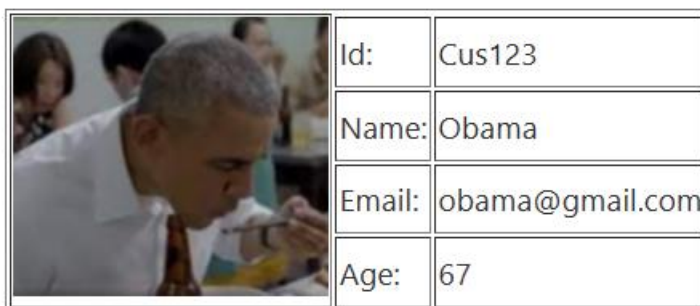
## **Bài 17- Json Object Model – Customer – Service**

### **Yêu cầu:**

Tạo một Component để hiển thị Khách hàng như hình bên dưới. Yêu cầu dùng JsonObject và cơ chế binding. Dữ liệu khai báo trong Type Script.

```
customer={
  "Id":"Cus123",
  "Name":"Obama",
  "Email":"obama@gmail.com",
  "Age":67,
  "Image":"assets/avatars/obama-avatar.png"
}
```

Giao diện khi chạy component:



## **Bài 18- Json Array Model – Group Customers (\*)**

### **Yêu cầu:**

Tạo 1 file Json lưu danh sách Khách hàng trong “assets/data/customers.json” có cấu trúc mẫu như dưới đây:

```
[
  {
    "CustomerId":1, "CustomerType": "VIP",
    "Customers": [{
      "Id": "Cus123",
      "Name": "Obama",
      "Email": "obama@gmail.com",
      "Age": 67,
      "Image": "assets/avatars/obama-avatar.png"
    }],
  },
  {
    "Id": "Cus456",
    "Name": "Kim jong Un",
  }
]
```

```
"Email": "unun@gmail.com",
"Age": 38,
"Image": "assets/avatars/unun-avatar.png"
},
{
  "Id": "Cus789",
  "Name": "Putin",
  "Email": "putin@gmail.com",
  "Age": 77,
  "Image": "assets/avatars/putin-avatar.png"
}]
},
{"CustomerId": 2, "CustomerTypeName": "Normal",
"Customers": [{
  "Id": "Cus000",
  "Name": "Hồ Cẩm Đào",
  "Email": "hodao@gmail.com",
  "Age": 16,
  "Image": "assets/avatars/hodao-avatar.png"
}],
{
  "Id": "Cus111",
  "Name": "Tap Can Binh",
  "Email": "binhbinh@gmail.com",
  "Age": 67,
  "Image": "assets/avatars/binhbinh-avatar.png"
}],
]
}
```

Áp dụng bài 14 (nhóm ) và bài 15 (service http) để làm bài 18 này.

Id	Name	Email	Age	#
1 - VIP				
Cus123	Obama	obama@gmail.com	67	
Cus456	Kim jong Un	unun@gmail.com	38	
Cus789	Putin	putin@gmail.com	77	
2 - Normal				
Cus000	Hồ Cẩm Đào	hodao@gmail.com	16	
Cus111	Tap Can Binh	binhbinh@gmail.com	67	

### **Bài 19 – Routing for Page Not Found**

Để điều hướng qua Page Not found khi người dùng chọn sai, ta cấu hình dòng cuối trong routes:

```
const routes: Routes = [
  {path: 'product', component: ProductComponent},
  {path: 'list-product', component: ListProductComponent},
  {path: 'service-product', component: ServiceProductComponent},
  {path: "**", component: PageNotFoundComponent},
];
```