Lab03 – Component Class

Shopping List

ในแลปครั้งนี้เราจะมาเรียนรู้การแก้ไขข้อมูลในแอพและการเชื่อมโยงแต่ละ Module ผ่านระบบ Navigation ด้วยการสร้างแอพสำหรับจัดการรายชื่อของที่จะซื้อ ก่อนอื่นเลยเราจะต้องสร้าง Class สำหรับเก็บข้อมูลสิ่งของหนึ่งชิ้น

1. เปิด stackblitz แล้วสร้างแอพ Angular ใหม่ขึ้นมา ตั้งชื่อ Repository เป็น รหัสนักศึกษา lab03 จากนั้นแก้ไขไฟล์ app.component.html ให้แสดงชื่อแอพใน Tag <h1> และแก้ไขตัวแปร name ในไฟล์ app.component.ts เป็นคำว่า Shopping List
2. เพิ่ม Component ชื่อ items ในแอพของเรา จากนั้นสร้าง Property ชื่อว่า item และให้ค่าเป็น Apple ใน itemsComponent
3. แก้ไขไฟล์ items.component.html ให้แสดงชื่อค่าของ Property item ที่สร้างไว้ข้างต้นใน Tag <p> จากนั้นเพิ่ม Tag <app-items> ไว้ด้านล่างของชื่อแอพในไฟล์ app.component.html

เนื่องจากสิ่งของที่เราจะซื้อ นอกจากจะต้องจดจำชื่อแล้วยังต้องจดจำปริมาณและหน่วยที่ซื้อด้วย เราสามารถทำได้ด้วยการสร้างคลาส Item ขึ้นมาแล้วกำหนด Property ที่ต้องจดจำทั้งหมดไว้ในคลาสนี้

1. สร้างไฟล์ใหม่ในแอพชื่อ item.ts จากนั้นประกาศคลาส Item ดังนี้

```
export class Item {
  id: number;
  name: string;
  quantity: number;
  unit: string;
}
```

สังเกตว่าในคลาสของเรามี Property id, quantity เก็บตัวเลขและ Property name, unit เก็บข้อความ การระบุประเภทของข้อมูลที่เก็บจะช่วยให้แอพทำงานได้อย่างมีประสิทธิภาพ

2. นำเข้าคลาส Item ไปยัง Component Items โดยการใช้คำสั่ง import {} from ''; จากนั้นแก้ไข Property item เดิมให้กลายเป็น Class Property item ดังรูปข้างล่าง

```
1   import { Component, OnInit } from '@angular/core';
2   import { Item } from '../item.ts';
3
4   @Component({
5     selector: 'app-items',
6     templateUrl: './items.component.html',
7     styleUrls: ['./items.component.css']
8   })
9   export class ItemsComponent implements OnInit {
10
11    item : Item = {
12      id: 1,
13      name: "Apple",
14      quantity: 1,
15      unit: "piece"
16    };
17
18    constructor() { }
19
20    ngOnInit() {
21    }
22
23  }
```

3. เราจะพบว่าชื่อ item เดิมที่แสดงผลอยู่กลายเป็นคำว่า [object Object] นั่นเพราะ Property item ได้กลายเป็นคลาสไปแล้ว ดังนั้นเราต้องแก้ไขการแสดงให้ตรงกับคลาส

ตอนนี้เรามีสิ่งของที่ต้องซื้อหนึ่งชิ้นในลิสต์แล้ว ถ้าเราเกิดอยากเปลี่ยนปริมาณหรือหน่วยในการซื้อ เราจะต้องมาเปลี่ยนค่าของ item ในแอพ ซึ่งผู้ใช้งานไม่มีทางเข้าถึงไฟล์นี้ได้ เพราะฉะนั้นเราจะต้องสร้างระบบการแก้ไขค่า Property จากตัวแอพโดยการใช้ Tag <input> เราเรียกวิธีนี้ว่า two-way binding

1. กลับไปยังไฟล์ items.component.html แล้วเปลี่ยนการแสดงผลแบบเดิมด้วยการใช้ Tag <input> และ ngModel ดังนี้

```
<div>
  <label>Name:
    <input [(ngModel)]="item.name" placeholder="name"/>
  </label>
</div>
```

การกำหนด Attribute ngModel ใน <input> เป็นการประกาศการใช้งาน two-way binding ใน Angular โดย item.name จะถูก bind ไว้กับ HTML Textbox ที่แสดงผล นั่นแปลว่าถ้า item.name เปลี่ยนค่า ค่าใน Textbox ก็จะเปลี่ยนไปด้วย และในทางกลับกันถ้าค่าใน Textbox เปลี่ยน ค่าของ item.name ก็จะเปลี่ยนไปด้วย

2. ทดสอบการทำงานด้วยการเพิ่ม {{item.name}} ไว้ด้านบน จากนั้นลองแก้ไขข้อความใน Textbox เราจะพบว่าข้อความด้านบนจะเปลี่ยนตาม

ตอนนี้ลิสต์ซื้อของของเรามีแค่ 1 ชิ้น เราต้องการให้มีการเพิ่มสิ่งของในลิสต์แต่ item ของเราสามารถเก็บได้แค่ค่าเดียว ดังนั้นเราจึงจำเป็นต้องเปลี่ยน item ให้กลายเป็น Array ชื่อ items ดังตัวอย่างด้านล่าง

```
items : Item[] = [
  {id: 1, name: "Apple", quantity: 1, unit: "piece"},
  {id: 2, name: "Orange", quantity: 2, unit: "piece"}
];
```

จากนั้นให้เราแสดงข้อมูลใน items ออกมาในรูปแบบลิสต์ (ดูตัวอย่างจากแลปที่ผ่านมา) สังเกตว่าตอนนี้แอพของเรามีสองส่วน ส่วนแสดงสิ่งของที่ต้องซื้อ และส่วนที่แสดงรายละเอียดของสิ่งของนั้น ทั้งสองส่วนยังไม่มีการเชื่อมต่อกัน เป้าหมายของเราคือเมื่อคลิกสิ่งของชิ้นไหนแล้วเราจะต้องสามารถแก้ไขได้

1. ก่อนอื่นเลยเราจะต้องตัวแปรมาจดจำว่าตอนนี้เรากำลังเลือกสิ่งของชิ้นใดอยู่ ให้เราสร้างตัวแปรใหม่ที่ชื่อ selectedItem ใน ItemsComponent อย่าลืมกำหนดให้เป็นคลาส Item ด้วย
2. เปลี่ยนตัวแปร item ในส่วนที่เป็นรายละเอียดของสินค้าให้เป็น selectedItem
3. เราจะสังเกตเห็นว่าเกิด error ในการแสดงผล เนื่องจากตอนเริ่มแอพ ตัวแปร selectedItem ยังไม่มีค่าอะไรกำหนดไว้ วิธีการป้องกันคือการใช้โครงสร้าง If ครอบดังตัวอย่างด้านล่าง

```
 9    <div *ngIf="selectedItem">
10    <h2>{{selectedItem.name | uppercase}} Details</h2>
11    <div>
12      <label>name:
13        <input [(ngModel)]="selectedItem.name" placeholder="name"/>
14      </label>
15    </div>
16    </div>
```

4. กลับที่ส่วนที่แสดงลิสต์สิ่งของ ให้เราเพิ่ม Attribute (click) ให้กับ <li> แล้วให้ค่าเป็นฟังก์ชัน onSelect(item) ซึ่ง item ตัวนี้จะต้องเป็นชื่อเดียวกันกับ item ใน *ngFor="let item of items" เมื่อทำเสร็จแล้วแอฟเราก็จะเรียกใช้ฟังก์ชัน onSelect() เมื่อมีการคลิกที่ของชิ้นนั้น
5. ลำดับต่อไปคือการเพิ่มฟังก์ชันใน items.component.ts ตามตัวอย่างด้านล่าง

```
onSelect(item : Item): void {
  this.selectedItem = item;
}
```

ลำดับต่อไปคือการเพิ่มสิ่งของในลิสต์ เราสามารถทำได้โดยการสร้างปุ่ม Add item แล้วผูกกับฟังก์ชัน addItem() โดยไม่ต้องใส่ argument ข้างใน

1. สร้างปุ่ม Add item แล้วเพิ่ม Attribute (click) ให้ไปเรียกฟังก์ชัน addItem()
2. เพิ่มฟังก์ชัน addItem() ใน items.component.ts ตามตัวอย่างด้านล่าง สังเกตว่าเวลาเราประกาศตัวแปรใหม่ภายในฟังก์ชันเราจะต้องใช้คำสั่ง let ฟังก์ชัน push เป็นการเพิ่มข้อมูลใน Array ไปไว้หลังสุด

```
addItem(): void {
  let tmp : Item = {id:0, name:"" , quantity:0,
   unit:""};
  this.items.push(tmp);
}
```

แบบฝึกหัด

1. ในส่วนที่แสดงรายละเอียดของสิ่งของ ให้แสดง id, quantity และ unit แต่จะสามารถแก้ไขข้อมูลได้เฉพาะ quantity และ unit
2. เพื่อป้องกันไม่ให้ id ซ้ำตอนเพิ่มสิ่งของใหม่ เราให้เพิ่ม Property counter ใน ItemsComponent และให้ค่าเป็น 3 (เนื่องจากลิสต์เริ่มต้นเรามีสิ่งของ 2 ชิ้น) จากนั้นแก้ไขฟังก์ชัน addItem ให้มีการใช้ค่า counter แทน id:0 และมีการเพิ่มค่า counter หลังจากเพิ่มสิ่งของเสร็จแล้ว (ใช้ ++ ได้)
3. สร้างปุ่ม Remove Item แล้วผูกการคลิกปุ่มกับฟังก์ชัน removeItem โดยส่งค่า argument เป็นสิ่งที่ของที่กำลังเลือกอยู่

```
removeItem(item : Item): void {
  let idx : number = this.items.indexOf(item);
  if (idx !== -1) {
    this.items.splice(idx,1);
  }
}
```