Lab05 – HTTP Get and Observables

Intro

Front-end application เชื่อมต่อกับ Backend service ผ่าน HTTP Protocol ซึ่ง Web Browser ในปัจจุบันสามารถใช้งาน API ได้สองแบบคือ XMLHttpRequest และ fetch() API โดยใน Angular มีการบรรจุ HttpClient สำหรับไว้ใช้งาน XMLHttpRequest แบบง่าย ในแลปนี้เราจะจำลอง Backend service หรือ Data server ด้วย Angular in-memory web-api เนื่องจากการเชื่อมต่อไปยัง Backend service จะต้องเป็นแบบ Async เราจึงจำเป็นต้องครอบข้อมูลที่ส่งกลับคืนมาในรูปแบบของ Observable object

Setup

1. Fork repository ของ lab04 แล้วตั้งชื่อเป็น lab05
2. Import HttpClientModule ใน AppModule
3. Import HttpClientInMemoryWebApiModule ใน AppModule

```
import { HttpClientModule } from '@angular/common/http';
import { HttpClientInMemoryWebApiModule } from
'angular-in-memory-web-api';
```

4. สร้าง Service ใหม่ชื่อ InMemoryData
5. ใน NgModule ของ AppModule เพิ่ม HttpClientModule และ HttpClientInMemoryWebApiModule โดยใน Root กำหนดเป็น InMemoryDataService

```
HttpClientModule,
HttpClientInMemoryWebApiModule.forRoot(
  InMemoryDataService, { dataEncapsulation: false }
)
```

6. แก้ไข product.ts โดยการเปลี่ยนการประกาศ Array มาเป็นการประกาศคลาส Product (เราจะย้ายข้อมูลใน Array ไปยัง InMemoryData)

```
export class Product {
  id : number;
  name : string;
  price : number;
}
```

Simulate a server

1. แก้ไขไฟล์ InMemoryData ตามตัวอย่างด้านล่าง
    a. providedIn เป็นการกำหนดให้มีแค่ Service เดียวใช้ร่วมกันทั้งแอพ
    b. implements InMemoryDbService คือการ Inherit DbService ที่มีอยู่แล้ว เราไม่จำเป็นต้องเขียนโค้ดดึงข้อมูลจากฐานข้อมูลเพิ่มเติม
    c. createDB คือการสร้างฐานข้อมูลภายใน Service ในขณะนี้จะเป็น Array ที่เราทำหลอกไว้ ในอนาคตมันจะกลายเป็นข้อมูลจากฐานข้อมูลภายนอกจริง ๆ การเพิ่ม id จะช่วยให้เราสามารถค้นหาสินค้าได้ง่ายขึ้น

```
import { Injectable } from '@angular/core';
import { InMemoryDbService } from 'angular-in-memory-web-api';
import { Product } from 'product';

@Injectable({
  providedIn: 'root',
})
export class InMemoryDataService implements InMemoryDbService {

  createDb() {
    const products = [
      {
        id : 0,
        name: 'iPhone 11',
        price: 24900,
      },
      {
        id : 1,
        name: 'iPhone 11 Pro',
        price: 35900,
      },
      {
        id : 2,
        name: 'iPhone 11 Pro Max',
        price: 39900,
      }
    ];
    return {products};
  }

}
```

2. สร้าง Service ชื่อ Product สำหรับไว้ใช้ติดต่อกับฐานข้อมูลชั่วคราวที่เราพึ่งสร้างขึ้นมา
    a. Import HttpClient, HttpHeaders, Product และ Observable
    b. Inject HttpClient ไว้ที่ constructor
    c. สร้างตัวแปร private ชื่อ productUrl เพื่อเตรียมเรียกใช้ Web api
    d. สร้างฟังก์ชัน getProducts เพื่อดึงข้อมูลสินค้าทุกชิ้นมาจาก Server โดยเรียกใช้งาน Http get ฟังก์ชัน ซึ่งส่งผ่าน Url ของสินค้าทั้งหมด
    เนื่องจากเราต้องการให้ฟังก์ชันนี้มีการทำงานแบบ Async เราต้องครอบ Array ของ Product ด้วย Observable
    e. สร้างฟังก์ชัน getProduct by id โดยเริ่มจากการสร้าง Url ตาม id ที่ได้รับมา จากนั้นส่ง Url ผ่าน Http get เพื่อดึงข้อมูลสินค้าเฉพาะ id ที่เราต้องการ (หมายเหตุ: เครื่องหมาย ` คือ Grave accent ซึ่งต่างจาก Single quote ที่เราเคยใช้)

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Product } from 'product';
import { Observable } from 'rxjs';

@Injectable()
export class ProductService {

  constructor(
    private http: HttpClient,
  ) { }

  private productUrl = 'api/products';

  getProducts(): Observable<Product[]> {
    return this.http.get<Product[]>(this.productUrl);
  }

  getProduct(id: number): Observable<Product> {
    const url = `${this.productUrl}/${id}`;
    return this.http.get<Product>(url);
  }

}
```

Reimplement the app

1. ตอนนี้แอพเราไม่แสดงผลสินค้าอะไรเลย เนื่องจากเราย้ายที่อยู่ของข้อมูลไปแล้ว ดังนั้นเราจำเป็นต้องปรับการรับข้อมูลใน Product list component
    a. เริ่มจากการ Import Product และ Product Service
    b. Inject Product Service ใน constructor
    c. Subscribe ไปยังฟังก์ชัน getProducts() ของ Product Service เราจะสังเกตว่าฟังก์ชัน getProducts() ส่งค่าตัวแปรในรูปแบบ Observable ซึ่งเราจะต้องไป Subscribe เพื่อให้การเชื่อมต่อเป็นไปในแบบ Asynchonous (ไม่ต้องรอจนกว่าจะได้ข้อมูลครบ)
    d. ในหน้าเพจ ให้ลบ index as productId ออกแล้วใช้ product.id แทน

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

@Component({
  selector: 'app-product-list',
  templateUrl: './product-list.component.html',
  styleUrls: ['./product-list.component.css']
})
export class ProductListComponent {
  products : Product[];

  constructor(private productService: ProductService) { }

  ngOnInit() {
    this.productService.getProducts()
    .subscribe(products => this.products = products);
  }

}
```

```
<div *ngFor="let product of products">

  <h3>
    <a [title]="product.name + ' details'" [routerLink]="
    ['/product',product.id]">
      {{ product.name }}
    </a>
  </h3>

</div>
```

Finish up (แบบฝึกหัด)

1. แก้ไข Product detail component ในลักษณะเดียวกันกับ Product list component ใช้คำสั่งด้านล่างในการดึงค่า id จากหน้าเพจ

   const id = +this.route.snapshot.paramMap.get('id');