

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО»**

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Звіт по лабораторній роботі № 4____
Сервіси. Створення локальних та глобальних сервісів.
з дисципліни: «Реактивне програмування»

Студент: Головня Олександр Ростиславович
Група: ІП-11
Дата захисту роботи: _____
Викладач: доц. Полупан Юлія Вікторівна
Захищено з оцінкою: _____

Київ, 2024

Зміст

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»	1
Вправа 1:	3
Вправа 2:	7
Впровадження сервісу в інший сервіс	7
Вправа 3:	8
Опціональні сервіси	8
Вправа 4:	10
Один сервіс для всіх компонентів	10
Ієрархія сервісів	14
Кореневий рівень	15
Рівень модуля	16
Рівень компоненту	17
Вправа 5: Виконати самостійно:	17
Виконання	17
Вправа 6: Виконати самостійно:	21
Висновок:	25
Список використаних джерел:	26

Мета: Мета: Навчитися створювати та використовувати сервіси у Angular.

Завдання: Завдання: Створити два Angular-додатки під назвою Service1 та Service2.

1. Для Angular-додатку Service1 виконати вправи 1-4;
2. Для Angular-додатку Service2 виконати самостійно вправу 5.
3. Для Angular-додатку Service3 виконати самостійно вправу 6.
4. Зробити звіт по роботі.
5. Angular-додаток Service2 та Service3 розгорнути на платформі Firebase у проєкті з ім'ям «ПрізвищеГрупаLaba4-2» та «ПрізвищеГрупаLaba4-3», наприклад «KovalenkoIP01Laba4-2» та «KovalenkoIP01Laba4-3».

Вправа 1:

Допустимо, у нас є наступна базова структура проєкту:

Додамо до папки src/app новий файл, який назвемо phone.ts. Визначимо у цьому файлі наступний клас:

```
HolovniaIP11Lab4 > service1 > src > app > TS phone.ts > Phone
1  export class Phone {
2      constructor(public name: string, public price: number) {}
3  }
```

Цей клас представлятиме дані, з якими ми працюватимемо.

Далі додамо до папки src/app новий файл data.service.ts. Цей файл міститиме код сервісу. Згідно з прийнятими в Angular умовностями в назві файлу після назви самого сервісу має йти підрядок .service. Якщо назва сервісу складається з кількох слів, які починаються з великої літери, то у назві файлу всі ці слова пишуться з маленької літери та поділяються дефісами. Наприклад, якщо сервіс називається SpecialSuperHeroService, то файл сервісу буде мати назву special-super-hero.service.ts.

Визначимо у цьому файлі наступний сервіс:

```
HolovniaIP11Lab4 > service1 > src > app > TS data.service.ts > DataService
1  import { Phone } from './phone'
2  export class DataService {
3      private data: Phone[] = [
4          { name: 'Apple iPhone 7', price: 36000 },
5          { name: 'HP Elite x3', price: 38000 },
6          { name: 'Alcatel Idol S4', price: 12000 },
7      ]
8      getData(): Phone[] {
9          return this.data
10     }
11     addData(name: string, price: number) {
12         this.data.push(new Phone(name, price))
13     }
14 }
```

У сервісі визначено масив даних та методи для роботи з ним. У реальному додатку ці дані можна отримувати з сервера або будь-якого зовнішнього сховища.

Тепер визначимо код компонента:

```
HolovniaIP11Lab4 > service1 > src > app > TS app.component.ts > AppComponent
1  import { Component, OnInit } from '@angular/core'
2  import { DataService } from './data.service'
3  import { Phone } from './phone'
4  @Component({
5      selector: 'my-app',
6      template: `
7      <div class="row">
8          <input class="form-control cardinput"
9              [(ngModel)]="name" placeholder="Модель"/>
10         <input type="number" class="form-control cardinput"
11             [(ngModel)]="price" placeholder="Ціна"/>
12         <button class="btn btn-default cardinput"
13             (click)="addItem(name, price)">
14             Додати
15         </button>
16     </div>
17     <table>
18     <thead>
19     <tr>
20     <th class="cardinput">Модель</th>
21     <th class="cardinput">Ціна</th>
22     </tr>
23     </thead>
24     <tbody>
25     <tr *ngFor="let item of items">
26     <td class="cardinput">{{item.name}}</td>
27     <td class="cardinput">{{item.price}}</td>
28     </tr>
29     </tbody>
30     </table>
31     `,
```

```

31 ,
32   styleUrls: ['./app.component.css'],
33
34   providers: [DataService],
35 })
36
37 export class AppComponent implements OnInit {
38   name: string = '';
39   price: number;
40   items: Phone[] = []
41   constructor(private dataService: DataService) { }
42   addItem(name: string, price: number) {
43     this.dataService.addData(name, price)
44   }
45   ngOnInit() {
46     this.items = this.dataService.getData()
47   }
48 }

```

Щоб задіяти сервіс у компоненті, його не тільки треба імпортувати:

```
import { DataService } from './data.service'
```

Але також необхідно його додати до колекції providers компонента:

```
providers: [DataService]
```

Усі сервіси, що використовуються, повинні бути визначені в колекції providers. І після цього ми зможемо задіяти вбудований у Angular механізм dependency injection та отримати об'єкт сервісу в конструкторі компонента і потім використовувати за потребою:

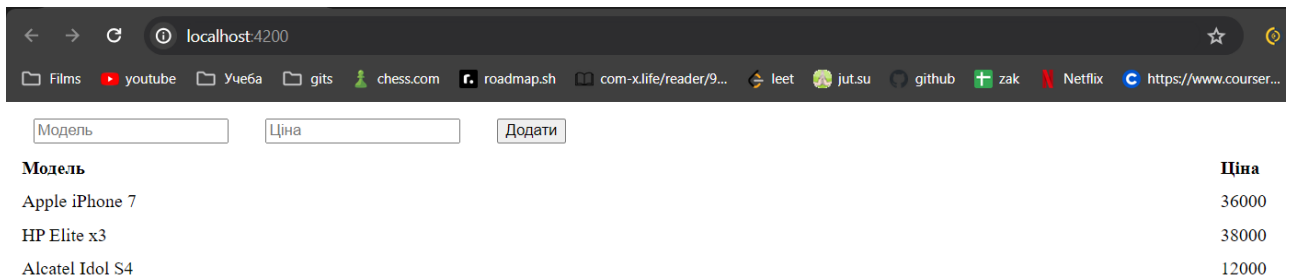
```
constructor(private dataService: DataService){ }
```

Для завантаження даних реалізується метод ngOnInit() інтерфейсу OnInit, що викликається після конструктора.

І тому що ми використовуємо директиву ngModel для роботи з елементами форми, нам потрібно підключити модуль FormsModule у файлі app.module.ts:

```
HolovniatP11Lab4 > service1 > src > app > TS app.module.ts > AppModule
1  import { NgModule } from '@angular/core'
2  import { BrowserModule } from '@angular/platform-browser'
3  import { FormsModule } from '@angular/forms'
4  import { AppComponent } from './app.component'
5  @NgModule({
6    imports: [BrowserModule, FormsModule],
7    declarations: [AppComponent],
8    bootstrap: [AppComponent],
9  })
10
11  export class AppModule { }
```

У результаті ми зможемо виводити дані та додавати нові елементи через сервіс:



Модель	Ціна
Apple iPhone 7	36000
HP Elite x3	38000
Alcatel Idol S4	12000

Файл стилів для компонента app.component.ts має наступний вміст:

```
HolovniatP11Lab4 > service1 > src > app > # app.component.css > th
1  .cardinput {
2    margin: 0.4rem 1.0rem;
3  }
4
5  td {
6    padding: 5px;
7    margin: 0.4rem 1.8rem;
8  }
9
10 table {
11   width: 100%;
12   overflow: auto;
13   color: rgb(0, 0, 0);
14   border-spacing: 1px;
15 }
16
17 th {
18   padding: 5px;
19   margin: 0.4rem 1.8rem;
20   text-align: left;
21 }
```

Вправа 2:

Впровадження сервісу в інший сервіс

Цілком ймовірна ситуація, коли ми захочемо використовувати один сервіс в іншому сервісі. Наприклад, у вправі 1 було створено сервіс для роботи з даними. Що якщо нам необхідно логувати всі операції з даними. Для логування визначимо новий сервіс. Для цього додамо до папки `src/app` новий файл `log.service.ts` з наступним вмістом:

```
HolovniaIP11Lab4 > service1 > src > app > TS log.service.ts >
1  export class LogService {
2      write(logMessage: string) {
3          console.log(logMessage)
4      }
5  }
```

Для логування у сервісі визначено метод `write`, який виводить повідомлення на консоль.

Тепер використовуємо цей сервіс. Для цього змінимо код у файлі `data.service.ts`:

```
HolovniaIP11Lab4 > service1 > src > app > TS data.service.ts > DataService
1  import { Injectable } from '@angular/core'
2  import { Phone } from './phone'
3  import { LogService } from './log.service'
4  @Injectable()
5  export class DataService {
6      private data: Phone[] = [
7          { name: 'Apple iPhone 7', price: 36000 },
8          { name: 'HP Elite x3', price: 38000 },
9          { name: 'Alcatel Idol S4', price: 12000 },
10     ]
11     constructor(private logService: LogService) { }
12     getData(): Phone[] {
13         this.logService.write('операція отримання даних')
14         return this.data
15     }
16     addData(name: string, price: number) {
17         this.data.push(new Phone(name, price))
18         this.logService.write('операція додавання даних')
19     }
20 }
```

Щоб вказати, що сервіс може використовувати інші сервіси, до класу сервісу застосовується декоратор `@Injectable`. Якщо клас не матиме такого декоратора, то вбудований механізм застосування залежностей не зможе створити об'єкт цього класу і видасть помилку.

Існує загальна рекомендація від розробників Angular застосовувати `@Injectable` добудь-якого класу сервісу, хоча це необов'язково.

Хоча у вправі 1 ми могли використовувати сервіс у компоненті без застосування до компоненту декоратора `@Injectable`. Справа в тому, що

декоратор `@Component`, який застосовується до компонента, є підтипом `@Injectable`.

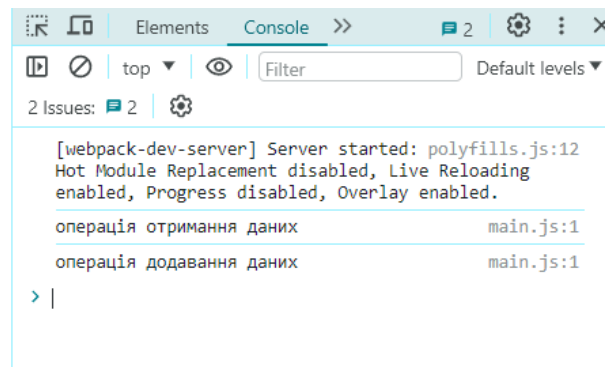
І також у випадку з `DataService` сервіс `LogService` також треба зареєструвати у списку провайдерів `AppComponent`:

```
HolovniatP11Lab4 > service1 > src > app > TS app.component.ts > ...
1  import { Component, OnInit } from '@angular/core'
2  import { DataService } from './data.service'
3  import { Phone } from './phone'
4  import { LogService } from './log.service'
5
   providers: [DataService, LogService],
  })
```

І незважаючи на те, що безпосередньо `LogService` не використовується в компоненті `AppComponent`, але він використовується в `DataService`, який викликається в `AppComponent`.

І при виконанні операцій із даними в консолі браузера ми зможемо побачити роботу сервісу `LogService`.

Модель	Ціна
Apple iPhone 7	36000
HP Elite x3	38000
Alcatel Idol S4	12000
POGO	5



Вправа 3:

Опціональні сервіси

Припустимо, з якоїсь причини сервіс `LogService` не доступний для інжектування, наприклад ми не додали в провайдери компонента `AppComponent`. Наприклад, буде зазначено так:

```
providers: [DataService],
  })
```

Якщо ми запустимо програму, то в цьому випадку ми отримаємо помилку. Оскільки для сервісу `LogService` не визначено провайдера.


```

✖ ERROR NullInjectorError: main.js:1
  R3InjectorError(e)[j0 -> j0]:
    NullInjectorError: No provider for j0!
      at rp.get (main.js:1:41908)
      at ol.get (main.js:1:44809)
      at ol.get (main.js:1:44809)
      at e0.get (main.js:1:63945)
      at Xf (main.js:1:32177)
      at eh (main.js:1:32663)
      at N (main.js:1:50688)
      at De (main.js:1:17635)
      at ko.ɵfac [as factory] (main.js:1:142713)
      at Dr (main.js:1:34319)

```

І тут можна визначити сервіс LogService як опціональний, застосовуючи декоратор Optional. Для цього змінимо код DataService:

```

HolovniaIP11Lab4 > service1 > src > app > TS data.service.ts > DataService
1  import { Injectable, Optional } from '@angular/core'
2  import { Phone } from './phone'
3  import { LogService } from './log.service'
4  @Injectable()
5  export class DataService {
6      private data: Phone[] = [
7          { name: 'Apple iPhone 7', price: 36000 },
8          { name: 'HP Elite x3', price: 38000 },
9          { name: 'Alcatel Idol S4', price: 12000 },
10     ]
11     constructor(@Optional() private logService: LogService) { }
12     getData(): Phone[] {
13         // this.logService.write('операція отримання даних')
14         if (this.logService)
15             this.logService.write('операція отримання даних')
16         return this.data
17     }
18     addData(name: string, price: number) {
19         this.data.push(new Phone(name, price))
20         this.logService.write('операція додавання даних')
21     }
22 }

```

Далі при зверненні до сервісу тепер потрібно перевірити, чи він встановлений, і якщо він встановлений, використовувати його:

```

if (this.logService)
    this.logService.write('операція отримання даних')

```

При цьому помилки вже не буде.

Вправа 4:

Один сервіс для всіх компонентів

Цілком можливо, що у нашому додатку буде кілька різних компонентів, які використовують сервіси. При цьому компоненти можуть використовувати ті самі сервіси. Наприклад, у попередніх вправах було створено сервіс DataService, який керує даними.

Визначимо спеціальний компонент для роботи з даними. Для цього візьмемо проект із минулої вправи та додамо до папки src/app новий файл data.component.ts:

Визначимо у цьому файлі наступний код:

```
HolovniaIP11Lab4 > service1 > src > app > TS data.component.ts > DataComponent
1 import { Component, OnInit } from '@angular/core'
2 import { DataService } from '../data.service'
3 import { LogService } from '../log.service'
4 import { Phone } from '../phone'
5 @Component({
6   selector: 'data-comp',
7   template: `
8     <div class="row">
9       <input class="form-control cardinput"
10         [(ngModel)]="name" placeholder="Модель"/>
11       <input type="number"
12         class="form-control cardinput"
13         [(ngModel)]="price"
14         placeholder="Ціна"/>
15       <button
16         class="btn btn-default cardinput"
17         (click)="addItem(name, price)"
18       >
19         Додати
20       </button>
21     </div>
22     <table>
23     <thead>
24     <tr>
25     <th class="cardinput">Модель</th>
26     <th class="cardinput">Ціна</th>
27     </tr>
28     </thead>
29     <tbody>
30     <tr *ngFor="let item of items">
31     <td class="cardinput">{{item.name}}</td>
32     <td class="cardinput">{{item.price}}</td>
33     </tr>
34     </tbody>
35     </table>
36   `,
```

```

36   },
37   styleUrls: ['./app.component.css'],
38   providers: [DataService, LogService]
39 })
40 export class DataComponent implements OnInit {
41   name: string = '';
42   price: number;
43   items: Phone[] = []
44   constructor(private dataService: DataService) { }
45   addItem(name: string, price: number) {
46     this.dataService.addData(name, price)
47   }
48   ngOnInit() {
49     this.items = this.dataService.getData()
50   }
51 }

```

DataComponent завантажує та додає дані. Для роботи із сервісами декоратор Component визначає секцію providers:

providers: [DataService, LogService]

Використовуємо цей компонент DataComponent у головному компоненті AppComponent:


```

HolovniaLP11Lab4 > service1 > src > app > ts app.component.ts >
1  import {Component} from '@angular/core';
2  @Component({
3    selector: 'my-app',
4    template: `<data-comp></data-comp>
5    <data-comp></data-comp>`
6  })
7  export class AppComponent {}

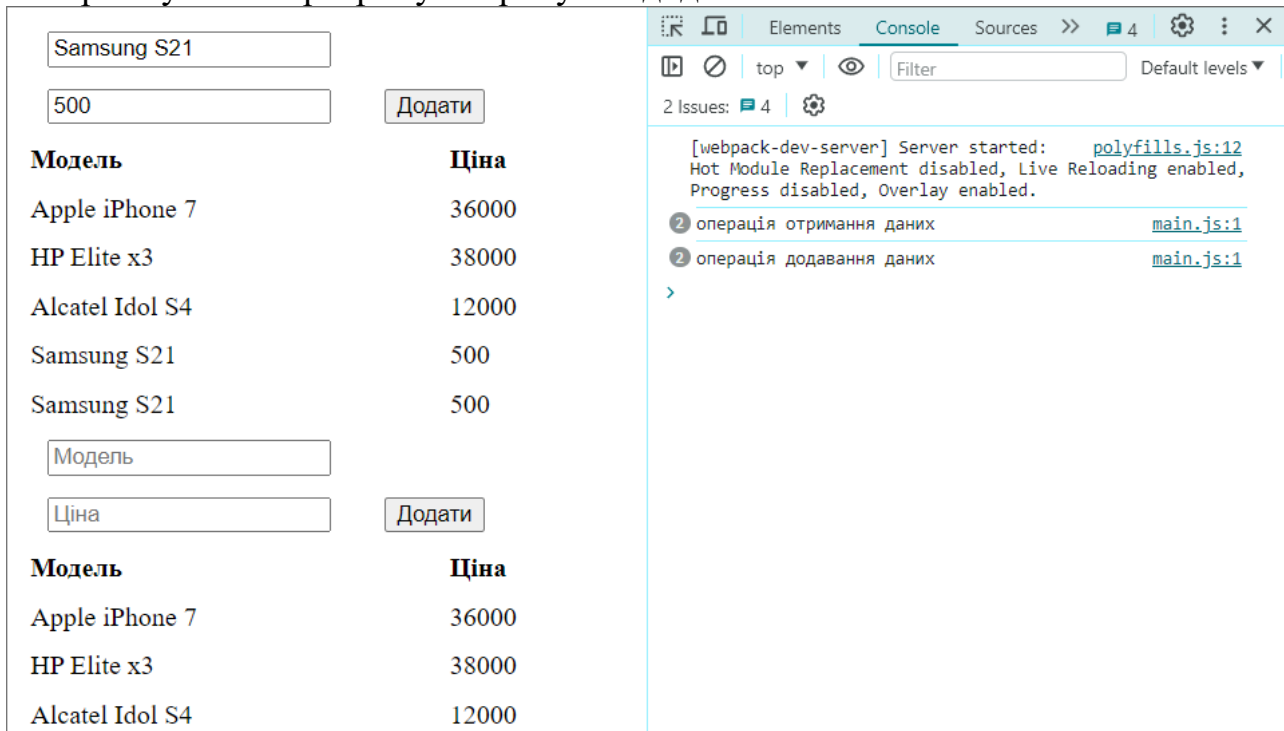
```

Цей компонент через елемент data-comp викликає компонент DataComponent. Причому викликає двічі. Тобто для обробки кожного елемента створюватиметься свій об'єкт DataComponent.

І відповідно змінимо головний модуль програми AppModule:

```
HolovniaIP11Lab4 > service1 > src > app > ts app.module.ts >  AppModule
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { FormsModule } from '@angular/forms';
4  import { AppComponent } from './app.component';
5  import { DataComponent } from './data.component';
6  @NgModule({
7    imports: [BrowserModule, FormsModule],
8    declarations: [AppComponent, DataComponent],
9    bootstrap: [AppComponent]
10 })
11
12
13 export class AppModule { }
```

Тепер запусимо програму і спробуємо додати новий елемент:



The screenshot shows a web application interface with a form and a table. The form has two input fields: 'Model' (containing 'Samsung S21') and 'Price' (containing '500'), followed by a 'Додати' button. Below the form, there is a table with two columns: 'Модель' and 'Ціна'. The table contains the following data:

Модель	Ціна
Apple iPhone 7	36000
HP Elite x3	38000
Alcatel Idol S4	12000
Samsung S21	500
Samsung S21	500

Below the table, there are two more input fields: 'Модель' and 'Ціна', followed by another 'Додати' button. The right side of the screenshot shows the browser's console with the following messages:

- [webpack-dev-server] Server started: polyfills.js:12
- Hot Module Replacement disabled, Live Reloading enabled, Progress disabled, Overlay enabled.
- 2 Issues: 4
- операція отримання даних main.js:1
- операція додавання даних main.js:1

Як видно на скріншоті, при додаванні в одному компоненті новий елемент додаватиметься тільки для цього компонента. Тому що у нас два окремі компоненти, і для кожного з них буде створюватись свій набір сервісів DataService та LogService.

Така поведінка не завжди може бути кращою. Можливо, потрібно, щоб компоненти використовували той самий об'єкт сервісу, замість створення різних сервісів кожного компонента. Для цього ми можемо зареєструвати всі сервіси не в компоненті, а в головному модулі AppModule:

```
HolovniaIP11Lab4 > service1 > src > app > TS app.module.ts > AppModule
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { FormsModule } from '@angular/forms';
4  import { AppComponent } from './app.component';
5  import { DataComponent } from './data.component';
6  import { DataService } from './data.service';
7  import { LogService } from './log.service';
8  @NgModule({
9      imports: [BrowserModule, FormsModule],
10     declarations: [AppComponent, DataComponent],
11     providers: [DataService, LogService], // реєстрація
12     bootstrap: [AppComponent]
13 })
14
15 export class AppModule { }
```

У цьому випадку ми вже можемо прибрати реєстрацію сервісів з DataComponent:

```
HolovniaIP11Lab4 > service1 > src > app > TS data.component.ts > DataComponent
1  import { Component, OnInit } from '@angular/core'
2  import { DataService } from './data.service'
3  import { LogService } from './log.service'
4  import { Phone } from './phone'
5  @Component({
6      selector: 'data-comp',
7      template: `
8      <div class="row">
9      <input
10      class="form-control cardinput"
11      [(ngModel)]="name"
12      placeholder="Модель"
13      />
14      <input
15      type="number"
16      class="form-control cardinput"
17      [(ngModel)]="price"
18      placeholder="Ціна"
19      />
20      <button
21      class="btn btn-default cardinput"
22      (click)="addItem(name, price)"
23      >
24      Додати
25      </button>
26      </div>
27      <table>
28      <thead>
29      <tr>
30      <th class="cardinput">Модель</th>
31      <th class="cardinput">Ціна</th>
32      </tr>
```

```

33 </thead>
34 <tbody>
35 <tr *ngFor="let item of items">
36 <td class="cardinput">{{item.name}}</td>
37 <td class="cardinput">{{item.price}}</td>
38 </tr>
39 </tbody>
40 </table>
41 `
42 styleUrls: ['./app.component.css'],
43
44 }]]
45 export class DataComponent implements OnInit {
46     name: string = '';
47     price: number;
48     items: Phone[] = []
49     constructor(private dataService: DataService) { }
50     addItem(name: string, price: number) {
51         this.dataService.addData(name, price)
52     }
53     ngOnInit() {
54         this.items = this.dataService.getData()
55     }
56 }

```

Після цього обидва об'єкти DataComponent будуть використовувати той самий сервіс DataService. Тому додавання об'єкта в одному компоненті автоматично позначиться на іншому:

Модель	Ціна
Apple iPhone 7	36000
HP Elite x3	38000
Alcatel Idol S4	12000
Samsung NEW	5

Модель	Ціна
Apple iPhone 7	36000
HP Elite x3	38000
Alcatel Idol S4	12000
Samsung NEW	5

Elements
Console
Sources
4

top
Filter
Default levels

2 Issues: 4

[webpack-dev-server] Server started: polyfills.js:12
Hot Module Replacement disabled, Live Reloading enabled,
Progress disabled, Overlay enabled.

операция отримання даних main.js:1

[webpack-dev-server] App updated. polyfills.js:12
Recompiling...

[webpack-dev-server] Nothing changed. polyfills.js:12

операция додавання даних main.js:1

Ієрархія сервісів

Як було показано раніше, ми можемо впроваджувати сервіси у компонентах та модулях. Залежно від цього, де впроваджуються послуги, бувають різні рівні провайдерів сервісів. Рівень по суті визначає сферу дії та

життєвий цикл сервісу. Є три рівні провайдерів сервісів:

- 1) Глобальний чи кореневий рівень
- 2) Рівень модуля
- 3) Рівень компоненту

Для встановлення відповідного рівня сервісу є два способи: додавання сервісу до колекції `providers` у модулі або компоненті та встановлення рівня за допомогою параметра `providedIn` у декораторі `@Injectable`.

З одного боку, може здатися, що немає сенсу в такому розподілі - чому б для всіх сервісів не зробити один кореневий рівень, щоб один об'єкт сервісу був доступний для всього додатку на кшталт синглтона. Однак нерідко виникає необхідність розмежувати функціонал між окремими функціональними частинами додатка. Наприклад, коли два компоненти працюють із різним набором даних – в цьому випадку вони можуть використовувати один клас сервісу, але різні його об'єкти.

Кореневий рівень

Кореневий рівень (root level) передбачає дію сервісу для всієї програми. Створюється та використовується один об'єкт сервісу для всіх частин програми. Подібний рівень встановлюється, якщо сервіс додається до колекції `providers` головного модуля, який зазвичай називається `AppModule` і з якого починається робота програми. Наприклад:

```
HolovniaIP11Lab4 > service1 > src > app > TS app.module.ts > AppModule
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { FormsModule } from '@angular/forms';
4  import { AppComponent } from './app.component';
5  import { DataComponent } from './data.component';
6  import { DataService } from './data.service';
7  import { LogService } from './log.service';
8  @NgModule({
9      imports: [BrowserModule, FormsModule],
10     declarations: [AppComponent, DataComponent],
11     providers: [DataService, LogService], // реєстрація
12     bootstrap: [AppComponent]
13 })
14
15 export class AppModule { }
```

Якщо `AppModule` є головним модулем, то для сервісу `DataService` визначено кореневий рівень. Тобто під час роботи програми буде створюватись один об'єкт даного сервісу для всієї програми.

Значення `root` для параметра `providedIn` у декораторі `Injectable` також дозволяє встановити кореневий рівень дії:


```
HolovniaIP11Lab4 > service1 > src > app > TS data.service.ts > DataService
1  import { Injectable, Optional } from '@angular/core'
2  import { Phone } from './phone'
3  import { LogService } from './log.service'
4  @Injectable({ providedIn: 'root' })
5  export class DataService {
6      private data: Phone[] = [
7          { name: 'Apple iPhone 7', price: 36000 },
8          { name: 'HP Elite x3', price: 38000 },
9          { name: 'Alcatel Idol S4', price: 12000 },
10     ]
11 }
```

У цьому випадку ми можемо не додавати цей сервіс до колекції providers у головному модулі.

Рівень модуля

Сервіси рівня модуля діють лише для поточного модуля і його класів - компонентів, директив, сервісів. Це все ті сервіси, які додаються до колекції providers у всіх інших модулях, крім головного модуля. Наприклад:

```
HolovniaIP11Lab4 > service1 > src > app > TS data.module.ts > DataModule
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { FormsModule } from '@angular/forms';
4  import { DataComponent } from './data.component';
5  import { DataService } from './data.service';
6  @NgModule({
7      imports: [BrowserModule, FormsModule],
8      declarations: [DataComponent],
9      exports: [DataComponent],
10     providers: [DataService],
11 })
12 export class DataModule { }
```

Або ми можемо задати рівень модуля безпосередньо у сервісі:

```
HolovniaIP11Lab4 > service1 > src > app > TS data.service.ts > DataService
1  import { Injectable, Optional } from '@angular/core'
2  import { Phone } from './phone'
3  import { LogService } from './log.service'
4  import { DataModule } from './data.module'
5
6  @Injectable({ providedIn: DataModule }) // тепер сервіс доступний
7  export class DataService {
8      private data: Phone[] = [
9          { name: 'Apple iPhone 7', price: 36000 },
10         { name: 'HP Elite x3', price: 38000 },
11     ]
12 }
```


Рівень компоненту

І тут дія сервісу обмежена поточним компонентом. Для кожного об'єкта компонента створюється об'єкт сервісу. Сам сервіс додається також до колекції providers компонента:

```
1  ,
2  styleUrls: ['./app.component.css'],
3  providers: [DataService, LogService], // добавление модуля Da
4  ])
```

Вправа 5: Виконати самостійно:

Створимо новий проект Service2. Створимо папку app/services. У ній два сервіси з різними областями видимості. Перший сервіс app-counter.service.ts глобального рівня. Інший сервіс local-counter.service.ts локального рівня. Використайте глобальний сервіс app-counter.service.ts та локальний сервіс local-counter.service.ts для двох компонентів app.component.ts та counter.component.ts як показано нижче (компонент counter.component.ts створить самостійно):

Виконання

1) Перший сервіс app-counter.services з класом AppComponentService кореневого рівня, який зареєстрований глобально та інжектований у компоненті app.component.ts має наступний вміст:

```
HolovniyP11Lab4 > service2 > src > app > services > TS app-counter.service.ts
1  import { Injectable } from '@angular/core';
2
3  @Injectable({
4    providedIn: 'root',
5  })
6  export class AppComponentService {
7    counter: number = 0;
8
9    increment() {
10     this.counter++;
11   }
12
13   decrement() {
14     this.counter--;
15   }
16
17   getCounter() {
18     return this.counter;
19   }
20 }
```

Шаблон app.component.html при цьому має наступний вміст:

```

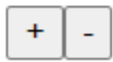
template: `
  <h2>
    Сервіс верхнього рівня App Counter {{ appCounterService.counter }}
  </h2>
  <button class="btn" (click)="appCounterService.increment()">+</button>
  <button class="btn" (click)="appCounterService.decrement()">-</button>

```

Результат:

Компонент app.component.ts

Сервіс верхнього рівня App Counter 7



При цьому кнопки "+" та "-" працюють зменшуючи або збільшуючи значення counter.

2) Інший сервіс local-counter.services з класом LocalCounterService зареєстрований локально на рівні компонента app.component.ts та інжектований в нього. Сервіс має наступний вміст:

```

HolovniaIP11Lab4 > service2 > src > app > services > TS local-counter.service.ts >
1  import { Injectable } from '@angular/core';
2
3  @Injectable()
4  export class LocalCounterService {
5      counter: number = 0;
6
7      increment() {
8          this.counter++;
9      }
10
11     decrement() {
12         this.counter--;
13     }
14
15     getCounter() {
16         return this.counter;
17     }
18 }

```

Результат:

Компонент `app.component.ts`

Сервіс верхнього рівня App Counter 2

Сервіс рівня компоненту Local Counter 6

3) Створимо окремий компонент у папці `app/counter` з ім'ям `counter.component.ts` та імпортуйте в нього сервіс глобального рівня `app-counter.services`. У шаблоні цього компоненту повторіть логіку `app.component.html`.

Компонент `app.component.ts`

Сервіс верхнього рівня App Counter 11

Сервіс рівня компоненту Local Counter 6

Компонент `counter.component.ts`

Сервіс верхнього рівня App Counter 11

4) Окремо у властивості providers компонента counter.component.ts зареєструємо локальний сервіс LocalCounterService наступним чином:

```
providers: [LocalCounterService],
```

та інжектуємо його наступним чином:

```
export class CounterComponent {  
  constructor(  
    public appCounterService: AppCounterService,  
    public localCounterService: LocalCounterService  
  ) {}  
}
```

Змінимо шаблон counter.component.html наступним чином:

```
template: `  
  <h2>  
    Сервіс верхнього рівня App Counter {{ appCounterService.counter }}  
  </h2>  
  <button class="btn" (click)="appCounterService.increment()">+</button>  
  <button class="btn" (click)="appCounterService.decrement()">-</button>  
  <h2>  
    Сервіс рівня компоненту Local Counter {{ localCounterService.counter }}  
  </h2>  
  <button class="btn" (click)="localCounterService.increment()">+</button>  
  <button class="btn" (click)="localCounterService.decrement()">-</button>  
  <hr/>  
`
```

Компонент app.component.ts

Сервіс верхнього рівня App Counter 19

Сервіс рівня компоненту Local Counter 15

Компонент counter.component.ts

Сервіс верхнього рівня App Counter 19

Сервіс рівня компоненту Local Counter 3

Вправа 6: Виконати самостійно:

Створіть новий проект Service3, в якому створіть сервіс для отримання користувачів (users) з сервера <https://jsonplaceholder.typicode.com/users>.

Отримавши відповідь від сервера, вивести дані користувачів на сторінку в наступному виді:

Id	Name	Username	Email	Phone	Website
1	Leanne Graham	Bret	Sincere@april.biz	1-770-736-8031 x56442	hildegard.org
2	Ervin Howell	Antonette	Shanna@melissa.tv	010-692-6593 x09125	anastasia.net
3	Clementine Bauch	Samantha	Nathan@yesenia.net	1-463-123-4447	ramiro.info
4	Patricia Lebsack	Karianne	Julianne.OConner@kory.org	493-170-9623 x156	kale.biz
5	Chelsey Dietrich	Kamren	Lucio_Hettinger@annie.ca	(254)954-1289	demarco.info
6	Mrs. Dennis Schulist	Leopoldo_Corkery	Karley_Dach@jasper.info	1-477-935-8478 x6430	ola.org
7	Kurtis Weissnat	Elwyn.Skiles	Telly.Hoeger@billy.biz	210.067.6132	elvis.io
8	Nicholas Runolfsdottir V	Maxime_Nienow	Sherwood@rosamond.me	586.493.6943 x140	jacynthe.com
9	Glenna Reichert	Delphine	Chaim_McDermott@dana.io	(775)976-6794 x41206	conrad.com
10	Clementina DuBuque	Moriah.Stanton	Rey.Padberg@karina.biz	024-648-3804	ambrose.net

(Для візуальної частини можна використати bootstrap або інші фреймворки).

При цьому завдання має два варіанти! 1 Варіант – парні номери в загальному списку; 2 Варіант – непарні номери в загальному списку.

76	ІП-11	Очна (денна)	Головня Олександр Ростиславович
----	-------	--------------	---------------------------------

1 Варіант: При виведенні даних, у шаблоні використовувати async пайп;

2 Варіант: При виведенні даних, у шаблоні не використовувати async пайп.

Інсталуємо Bootstrap (опціонально для стилізації):

```
npm install bootstrap
```

Додамо Bootstrap у файл angular.json:

```
"styles": [ "node_modules/bootstrap/dist/css/bootstrap.min.css", "src/styles.css" ],
```

Створимо сервіс для отримання користувачів: Виконаємо команду:

```
ng generate service user
```

У файлі src/app/user.service.ts:

```
HolovniaIP11Lab4 > service3 > src > app > TS user.service.ts > ...
1  import { Injectable } from '@angular/core';
2  import { HttpClient } from '@angular/common/http';
3  import { Observable } from 'rxjs';
4
5  @Injectable({
6    providedIn: 'root'
7  })
8  export class UserService {
9    private apiUrl = 'https://jsonplaceholder.typicode.com/users';
10
11    constructor(private http: HttpClient) {}
12
13    getUsers(): Observable<any> {
14      return this.http.get<any>(this.apiUrl);
15    }
16  }
17
```

Імпортуємо HttpClientModule в src/app/app.module.ts:

```
HolovniaIP11Lab4 > service3 > src > app > TS app.module.ts > AppModule
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { FormsModule } from '@angular/forms';
4  import { AppComponent } from './app.component';
5  import { HttpClientModule } from '@angular/common/http';
6  import { UsersComponent } from './users/users.component';
7
8  @NgModule({
9    imports: [BrowserModule, HttpClientModule],
10   providers: [],
11   declarations: [AppComponent, UsersComponent],
12   bootstrap: [AppComponent],
13 })
14 export class AppModule {}
```

Створимо компонент для відображення користувачів: Виконаємо команду:

ng generate component users

У компоненті src/app/users/users.component.ts:

```
HolovniaIP11Lab4 > service3 > src > app > users > TS users.component.ts > ...
1  import { Component, OnInit } from '@angular/core';
2  import { UserService } from '../user.service';
3  import { Observable } from 'rxjs';
4
5  @Component({
6    selector: 'app-users',
7    templateUrl: './users.component.html',
8    styleUrls: ['./users.component.css']
9  })
10 export class UsersComponent implements OnInit {
11   users$: Observable<any>;
12
13   constructor(private userService: UserService) {}
14
15   ngOnInit(): void {
16     this.users$ = this.userService.getUsers();
17   }
18 }
```

У шаблоні src/app/users/users.component.html використовуємо async пайп для виведення користувачів:

```

HolovniaIP11Lab4 > service3 > src > app > users > <> users.component.html >
1  <div class="container mt-4">
2    <h2 class="mb-4">Users List</h2>
3    <table class="table table-bordered">
4      <thead>
5        <tr>
6          <th>ID</th>
7          <th>Name</th>
8          <th>Username</th>
9          <th>Email</th>
10         <th>Phone</th>
11         <th>Website</th>
12       </tr>
13     </thead>
14     <tbody>
15       <tr *ngFor="let user of users$ | async">
16         <td>{{ user.id }}</td>
17         <td>{{ user.name }}</td>
18         <td>{{ user.username }}</td>
19         <td>{{ user.email }}</td>
20         <td>{{ user.phone }}</td>
21         <td>{{ user.website }}</td>
22       </tr>
23     </tbody>
24   </table>
25 </div>
26
27

```

Результат бачимо на рисунку:

Users List

ID	Name	Username	Email	Phone	Website
1	Leanne Graham	Bret	Sincere@april.biz	1-770-736-8031 x56442	hildegard.org
2	Ervin Howell	Antonette	Shanna@melissa.tv	010-692-6593 x09125	anastasia.net
3	Clementine Bauch	Samantha	Nathan@yesenia.net	1-463-123-4447	ramiro.info
4	Patricia Lebsack	Karianne	Julianne.OConner@kory.org	493-170-9623 x156	kale.biz
5	Chelsey Dietrich	Kamren	Lucio_Hettinger@annie.ca	(254)954-1289	demarco.info
6	Mrs. Dennis Schulist	Leopoldo_Corkery	Karley_Dach@jasper.info	1-477-935-8478 x6430	ola.org
7	Kurtis Weissnat	Elwyn.Skiles	Telly.Hoeger@billy.biz	210.067.6132	elvis.io
8	Nicholas Runolfsdottir V	Maxime_Nienow	Sherwood@rosamond.me	586.493.6943 x140	jacynth.com
9	Glenna Reichert	Delphine	Chaim_McDermott@dana.io	(775)976-6794 x41206	conrad.com
10	Clementina DuBuque	Moriah.Stanton	Rey.Padberg@karina.biz	024-648-3804	ambrose.net

Посилання на додатки:

<https://holovniaip11s2.web.app/>

<https://holovniaip11s3.web.app/>

Висновок:

Під час виконання комп'ютерного практикуму навчився створювати та використовувати сервіси у Angular. Дізнався про їх призначення та використання,

Мною було створено 3 додатка. Згідно завдання, розгорнув Angular-додатки «Service2» та «Service3» на платформі FireBase.

Список використаних джерел:

1. Understanding Services: <https://v17.angular.io/guide/creating-injectable-service>