ДОДАТОК

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт по лабораторній роботі No 7

Сервіси. Створення локальних та глобальних сервісів.

З дисципліни: «Реактивне програмування»

Студент: Трофимов Данило Олегович

Група: ІП-02

Дата захисту роботи:

Викладач: доц. Полупан Юлія Вікторівна

Захищено з оцінкою:

Київ, 2023

ЗМІСТ

Загальний хід роботи:

1. Згідно з інструкціями ініціалізовано angular проєкти Service 1 та Service 2;
2. Було опрацьовано завдання згідно з методичними вказівками;
3. Було розгорнуто додаток Service 2 на платформі firebase.

1) Сервіси: призначення та приклади використання;

Сервіси - це ключовий елемент будь-якого фреймворку чи платформи розробки програмного забезпечення. Вони дозволяють структурувати та відокремлювати функціонал програми, роблячи код більш читабельним та підтримуваним. Розглянемо призначення та приклад використання сервісів на прикладі Angular.

В світі Angular сервіси - це класи, які надають конкретний функціонал та можуть бути використані в будь-якому компоненті чи модулі. Вони дозволяють поділити функціонал між компонентами, створити єдиний екземпляр сервісу для використання в різних частинах програми та забезпечити вищий рівень абстракції.

У наведеному коді ми вже використовуємо сервіс для отримання та додавання даних про телефони. Давайте розглянемо основні аспекти коду:

```
export class DataService {
  private data: Phone[] = [
    { name: 'Apple iPhone 7', price: 36000 },
    { name: 'HP Elite x3', price: 38000 },
    { name: 'Alcatel Idol S4', price: 12000},
  ]

  getData(): Phone[] {
    return this.data
  }

  addData(name: string, price: number) {
    this.data.push(new Phone(name, price))
```

```
  }
}
```

У цьому класі DataService ви описали методи для отримання (getData) та додавання (addData) даних. Цей сервіс може бути використаний в будь-якому компоненті вашого Angular додатку.

У компоненті AppComponent ми використовуємо сервіс через dependency injection. Ключовим моментом є те, що ми вказали DataService в масиві providers метаданих компонента. Це робить сервіс доступним для використання всередині цього компоненту.

У шаблоні ии використовуємо дані, отримані з сервісу, та передаємо нові дані для додавання:

```
html
<div class="row">
  <input class="form-control cardinput" [(ngModel)]="name" placeholder="Модель"
/>
  <input type="number" class="form-control cardinput" [(ngModel)]="price"
placeholder="Ціна" />
  <button class="btn btn-default cardinput" (click)="addItem(name,
price)">Додати</button>
</div>

<table>
  <thead>
    <tr>
      <th class="cardinput">Модель</th>
      <th class="cardinput">Ціна</th>
    </tr>
  </thead>
  <tbody>
```

```
    <tr *ngFor="let item of items">
      <td class="cardinput">{{item.name}}</td>
      <td class="cardinput">{{item.price}}</td>
    </tr>
  </tbody>
</table>
```

Цей шаблон використовує двустороннє зв'язування даних ([(ngModel)]) для отримання даних з полів вводу та відображення списку телефонів. Коли ми натискаємо кнопку "Додати", викликається метод addItem, який використовує сервіс для додавання нового телефону до списку.

2) **Впровадження сервісу в інший сервіс;**

Додаємо новий сервіс для логування подій. Новий сервіс, LogService, буде використовуватися для виведення повідомлень в консоль про операції отримання та додавання даних.

Створення сервісу LogService:

typescript

```
// log.service.ts

export class LogService {
  write(logMessage: string) {
    console.log(logMessage);
  }
}
```

Внедрення LogService в DataService:

typescript

```
// data.service.ts

import { Injectable } from '@angular/core';
```

```
import { Phone } from './phone';
import { LogService } from './log.service';

@Injectable()
export class DataService {
  private data: Phone[] = [
    { name: 'Apple iPhone 7', price: 36000 },
    { name: 'HP Elite x3', price: 38000 },
    { name: 'Alcatel Idol S4', price: 12000 },
  ];

  constructor(private logService: LogService) {}

  getData(): Phone[] {
    this.logService.write('Операція отримання даних');
    return this.data;
  }

  addData(name: string, price: number) {
    this.data.push(new Phone(name, price));
    this.logService.write('Операція додавання даних');
  }
}
```

У цьому коді ми додаємо LogService в конструктор DataService, щоб внедрити його в клас. Тепер кожен раз, коли ми викликаємо методи getData або addData, вони виводять повідомлення про відповідну операцію у консоль.

Використання LogService в AppComponent:

typescript

```
// app.component.ts
import { Component, OnInit } from '@angular/core';
import { DataService } from './data.service';
import { Phone } from './phone';
import { LogService } from './log.service';

@Component({
  selector: 'my-app',
  template: `
    <!-- HTML-код -->
  `,
  styleUrls: ['./app.component.css'],
  providers: [DataService, LogService],
})
export class AppComponent implements OnInit {
  name: string = '';
  price: number;
  items: Phone[] = [];
```

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

  addItem(name: string, price: number) {
    this.dataService.addData(name, price);
  }

  ngOnInit() {
    this.items = this.dataService.getData();
  }
}
```

Тепер ми внедрили LogService в компонент AppComponent так само, як і DataService. Таким чином, сервіс для логування доступний в усьому компоненті, і його можна використовувати для докладного виведення подій.

3) Опціональні сервіси

Використання Опціонального Сервісу в DataService:

Додаємо опціональний параметр logService в конструктор
constructor(@Optional() private logService: LogService) {}

Ми використали декоратор @Optional() для позначення параметра logService як опціональний у конструкторі. Це дозволяє Angular інжектувати null, якщо сервіс не був зареєстрований в додатку.

Перевірка та Використання Опціонального Сервісу в DataService

typescript

```
getData(): Phone[] {
  if (this.logService) {
    this.logService.write('Операція отримання даних');
```

```
  }
  return this.data;
}
```

Ми перевіряємо, чи logService не є null перед викликом методу write. Це запобігає помилкам, якщо сервіс не був наданий (не був зареєстрований) у вашому додатку.

4) Один сервіс для всіх компонентів

Для використання сервісів для всіх компонентів ми впровадили реєстрацію провійдеру в AppModule

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';
import { DataComponent } from './data.component';
import { DataService } from './data.service';
import { LogService } from './log.service';

@NgModule({
  imports: [BrowserModule, FormsModule],
  declarations: [AppComponent, DataComponent],
  providers: [DataService, LogService], // реєстрація сервісів
  bootstrap: [AppComponent],
})
export class AppModule {}
```

У нашому AppModule ми залишаємо реєстрацію сервісів (DataService та LogService) на рівні модуля. Це забезпечить їх доступність для всіх компонентів, які будуть включені в цей модуль.

Тепер, оскільки ми зареєстрували сервіси на рівні модуля, вам більше не потрібно реєструвати їх в компоненті DataComponent. Angular автоматично надасть доступ до зареєстрованих сервісів усім компонентам, які входять до складу цього модуля.

5) Ієрархія сервісів

Кореневий рівень: DataService:

Для реалізації сервісу кореневого рівня скористаємося параметром для декоратора injectable.

```
@Injectable({
  providedIn: 'root',
})
```

Рівень модуля: DataModule

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { DataComponent } from './data.component';
import { DataService } from './data.service';

@NgModule({
  imports: [BrowserModule, FormsModule],
  declarations: [DataComponent],
  exports: [DataComponent],
```

```
  providers: [DataService], // реєстрація сервісу
})
export class DataModule {}
```

На рівні модуля ми використовуємо DataModule для організації компонентів та сервісів. Реєстрація DataService в провайдерах DataModule робить його доступним для всіх компонентів, що входять до складу цього модуля.

Рівень компоненту: DataComponent

```
import { Component } from '@angular/core';
import { DataService } from './data.service';

@Component({
  selector: 'data-comp',
  template: `
    <!-- HTML-код -->
  `,
  providers: [DataService], // реєстрація сервісу
})
export class DataComponent {
// логіка компоненту
}
```

У компоненті DataComponent ми реєструємо DataService в провайдерах цього компонента. Це гарантує, що кожен екземпляр компонента отримає свій власний екземпляр сервісу, ізольований від інших компонентів.

6) Детальний огляд та призначення всіх структурних блоків Angular-додатку Service2.

1. LocalCounterService:

Це Angular сервіс, який має за мету зберігати та обробляти локальний лічильник (counter). Його функціональність включає методи increase() та decrease(), які відповідають за збільшення та зменшення значення counter відповідно.

Призначення: Цей сервіс визначає локальний лічильник і дозволяє збільшувати та зменшувати його значення.

Методи:

increase(): void: Збільшує значення лічильника на одиницю.

decrease(): void: Зменшує значення лічильника на одиницю.

2. AppCounterService:

Цей Angular сервіс є схожим на LocalCounterService. Однак він визначений як сервіс, надаючи його на рівні кореня додатку (providedIn: 'root'). Це робить його глобальним сервісом, який може бути використаний в будь-якому місці додатку.

Призначення: Цей сервіс також визначає лічильник, але на рівні додатку. Його значення може бути спільно використано всіма компонентами та сервісами на рівні додатку.

Методи:

increase(): void: Збільшує значення лічильника на одиницю.

decrease(): void: Зменшує значення лічильника на одиницю.

3. AppModule:

AppModule - це головний модуль нашого Angular-додатку. У цьому модулі ми реєструємо наші компоненти та сервіси. Зокрема, ми підключаємо

AppCounterService як провайдера на рівні модуля, щоб забезпечити його доступність в усій області цього модуля.

Призначення: Головний модуль додатку, де визначаються всі компоненти та сервіси, які будуть використовуватися на рівні всього додатку.

Блоки:

declarations: Список всіх компонентів, які використовуються в додатку (AppComponent, CounterComponent).

imports: Підключені модулі, у даному випадку лише BrowserModule.

providers: Реєстрація сервісу AppCounterService як провайдера на рівні додатку.

bootstrap: Вказує головний компонент для завантаження, у даному випадку - AppComponent.

4. AppComponent:

AppComponent - головний компонент нашого додатку. У цьому компоненті ми використовуємо як глобальний (AppCounterService), так і локальний (LocalCounterService) сервіси для відображення та керування лічильниками. Компонент також вказує, що LocalCounterService має бути доступний тільки в області цього компонента.

Призначення: Головний компонент додатку, який відповідає за відображення головної сторінки та управління лічильниками.

Блоки:

providers: Реєстрація сервісу LocalCounterService як провайдера для цього компоненту.

5. **CounterComponent:**

CounterComponent - це ще один компонент нашого додатку, який також використовує як глобальний, так і локальний сервіси для відображення та керування лічильниками. Компонент також вказує, що LocalCounterService повинен бути доступний тільки в області цього компонента.

Призначення: Компонент для відображення лічильників на окремій сторінці або в інших частинах додатку.

Блоки:

providers: Реєстрація сервісу LocalCounterService як провайдера для цього компоненту.

HTML-шаблон:

Інтерфейс для взаємодії з лічильниками App та Local.

Кнопки для збільшення та зменшення значення обох лічильників.

6. HTML-шаблони

В обох компонентах (AppComponent та CounterComponent) використовуються HTML-шаблони для відображення поточного значення лічильників та кнопок для їх збільшення та зменшення.

Ці структурні блоки утворюють архітектуру Angular-додатку "Service2", де глобальний сервіс доступний в усій додатковій області, а локальний - тільки в межах відповідних компонентів.

7) Посилання на розгорнутий додаток

Додаток розміщений за адресою: https://trofymovip02laba7.web.app/

Висновки

Використання сервісів в Angular є ключовим елементом для створення модульних та ефективних додатків. Локальні сервіси, призначені для конкретних компонентів, забезпечують ізольований обсяг функціональності, що полегшує управління станом та реакцією на події в межах компоненту. Глобальні сервіси, визначені на рівні кореня додатку, стають доступними в усій його області, що дозволяє зберігати та обмінюватися даними між різними частинами додатку. Використання обох типів сервісів дозволяє створювати добре структуровані та легко розширювані Angular-додатки.

Список літератури

1. Introduction to the Angular docs [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/docs.
2. Angular. Angular tutorials [Електронний ресурс] / Angular – Режим доступу до ресурсу: https://angular.io/tutorial.
3. The Angular Book [Електронний ресурс] – Режим доступу до ресурсу: https://angular-book.dev/.
4. Daniel Schmitz. Angular 14 from Scratch / Daniel Schmitz., 2022.
5. Denys Vuika. Developing with Angular / Denys Vuika., 2018.