# Сервисы и dependency injection

Сервисы в Angular представляют довольно широкий спектр классов, которые выполняют некоторые специфические задачи, например, логгирование, работу с данными и т.д.

В отличие от компонентов и директив сервисы не работают с представлениями, то есть с разметкой html, не оказывают на нее прямого влияния. Они выполняют строго определенную и достаточно узкую задачу.

Стандартные задачи сервисов:

* Предоставление данных приложению. Сервис может сам хранить данные в памяти, либо для получения данных может обращаться к какому-нибудь источнику данных, например, к серверу.
* Сервис может представлять канал взаимодействия между отдельными компонентами приложения.
* Сервис может инкапсулировать бизнес-логику, различные вычислительные задачи, задачи по логгированию, которые лучше выносить из компонентов. Тем самым код компонентов будет сосредоточен непосредственно на работе с представлением. Кроме того, тем самым мы также можем решить проблему повторения кода, если нам потребуется выполнить одну и ту же задачу в разных компонентах и классах.

Добавим в папку src/app новый файл data.service.ts. Этот файл будет содержать код сервиса. Согласно условностям в названии файла после названия самого сервиса должна идти подстрока .service. Если название сервиса состоит из нескольких слов, которые начинаются с большой буквы, то в названии файла все эти слова пишутся с маленькой буквы и разделяются дефисами:

export class DataService {

    private data: string[] = ["Apple iPhone XR", "Samsung Galaxy S9", "Nokia 9"];

    getData(): string[] {

        return this.data;

    }

    addData(name: string) {

        this.data.push(name);

    }

}

Теперь определим код компонента:

import { Component } from '@angular/core';

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

@Component({

    selector: 'angular-app',

    templateUrl: './app.component.html',

    providers: [DataService]

})

export class AppComponent {

    items: string[] = [];

    constructor(private dataService: DataService) { }

    addItem(name: string) {

        this.dataService.addData(name);

    }

    deleteData(name: string) {

        console.log(`index - ${this.data.indexOf(name)}`)

        if (this.data.indexOf(name) < 0) return;

        this.data.splice(this.data.indexOf(name), 1);

    }

    ngOnInit() {

        this.items = this.dataService.getData();

    }

}

И его html-разметку:

<div class="panel">

    <div><input [(ngModel)]="name" placeholder="Модель" />

        <button (click)="addItem(name)">Добавить</button>

    </div>

    <table>

        <tr \*ngFor="let item of items">

            <td>{{item}}</td>

            <td><button (click)="deleteItem($event)">Delete</button> </td>

        </tr>

    </table>

</div>

Чтобы задействовать сервис в компоненте, его не только надо импортировать:

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

Но также необходимо его добавить в коллекцию providers компонента:

@Component({

    selector: 'angular-app',

    templateUrl: './app.component.html',

    providers: [DataService]

})

Все используемые сервисы добавляются в коллекцию providers. Тем самым определяется провайдер сервиса, который будет использоваться для его получения. И после этого мы сможем задействовать встроенный в Angular механизм dependency injection и получить объект сервиса в конструкторе компонента и затем использовать по необходимости:

constructor(private dataService: DataService) { }

Для загрузки данных реализуется метод ngOnInit() интерфейса OnInit, который вызывается после конструктора.

# Внедрение сервиса в другой сервис

Вполне вероятна ситуация, когда мы захотим использовать один сервис в другом сервисе. Что если нам необходимо логгировать все операции с данными. Для логгирования определим новый сервис. Для этого добавим в папку src/app новый файл log.service.ts со следующим содержимым:

export class LogService {

    write(logMessage: string) {

        console.log(logMessage);

    }

}

Для логгирования в сервисе определен метод write, который выводит некоторое сообщение на консоль.

Теперь используем этот сервис. Для этого изменим код в файле data.service.ts:

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

import { LogService } from './log.service';

@Injectable()

export class DataService {

    private data: string[] = ["Apple iPhone XR", "Samsung Galaxy S9", "Nokia 9"];

    constructor(private logService: LogService) { }

    getData(): string[] {

        this.logService.write("операция получения данных");

        return this.data;

    }

    addData(name: string) {

        this.data.push(name);

        this.logService.write("операция добавления данных");

    }

    deleteData(name: string) {

        console.log(`index - ${this.data.indexOf(name)}`)

        if (this.data.indexOf(name) < 0) return;

        this.data.splice(this.data.indexOf(name), 1);

this.logService.write("операция удаления данных");

    }

}

Чтобы указать, что сервис сам может использовать другие сервисы, к классу сервиса применяется декоратор Injectable. Если класс не будет иметь подобного декоратора, то встроенный механизм внедрения зависимостей не сможет создать объект этого класса и выдаст ошибку.

Существует общая рекомендации от разработчиков Angular применять Injectable к любому классу сервиса, хотя в принципе это необязательно. Дело в том, что декоратор Component, который применяется к компоненту, является подтипом Injectable.

И также в случае с DataService сервис LogService также надо зарегистрировать в списке провайдеров AppComponent:

import { Component } from '@angular/core';

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

import { LogService } from './log.service';

@Component({

    selector: 'angular-app',

    templateUrl: './app.component.html',

    providers: [DataService, LogService]

})

И несмотря на то, что напрямую LogService не используется в компоненте AppComponent, но он используется в DataService, который вызывается в AppComponent.

И при выполнении операций с данными в консоли браузера мы сможем увидеть работу сервиса LogService.

# Pipes

Pipes представляют специальные инструменты, которые позволяют форматировать отображаемые значения.

Angular есть ряд встроенных pipes. Основные из них:

* CurrencyPipe: форматирует валюту
* PercentPipe: форматирует проценты
* UpperCasePipe: переводит строку в верхний регистр
* LowerCasePipe: переводит строку в нижний регистр
* DatePipe: форматирует дату
* DecimalPipe: задает формат числа
* SlicePipe: обрезает строку

При применении классов суффикс Pipe отбрасывается (за исключением DecimalPipe - для его применения используется название "number"):

<div class="panel">

    <div><input [(ngModel)]="name" placeholder="Модель" />

        <button (click)="addItem(name)">Добавить</button>

    </div>

    <table>

        <tr \*ngFor="let item of items | slice :1:4">

            <td>{{item| uppercase}}</td>

            <td><button (click)="deleteItem($event)">Delete</button> </td>

        </tr>

    </table>

</div>