**Курс «Создание веб-приложений с использованием**

**Angular & React»**

Тема: Angular: директивы, data binding, сервисы,

dependency injection, навигация.

**Домашняя работа №2**

Выполнил студент гр. ВПУ-911: Мулин Николай Сергеевич

IDE: Microsoft Visual Studio Code

Ссылка на проект в GitHub: <https://github.com/Nickomu92/hw2_Angular>

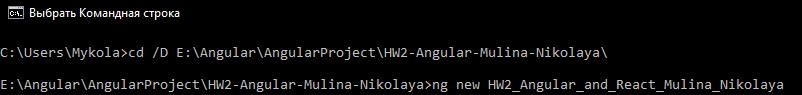
**Условие задания**

1. *Создание и использование сервисов* – *сервис “DataService”.*

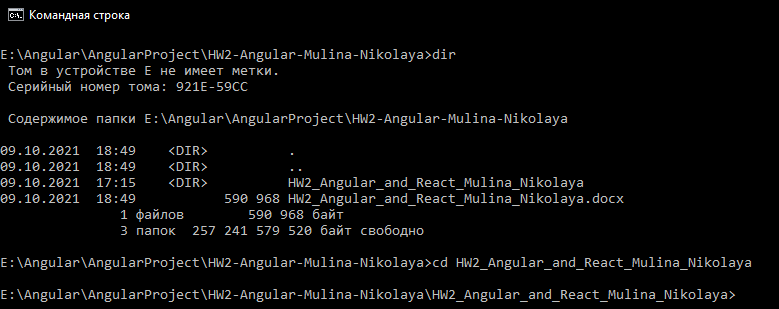
*+ Реализовано:*

* *Использование Pipes (каналов, фильтров) – компонент “PipesTableHeaderComponent”;*
* *Использование* директив (\*ngFor, ngModel) и двунаправленных привязок *– компонент “ToDoItemsComponent”.*

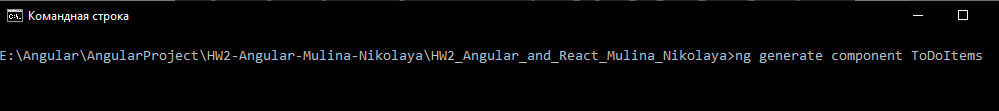
1. ***Создадим пространство для нового проекта HW2\_Angular\_and\_React\_Mulina\_Nikolaya:***



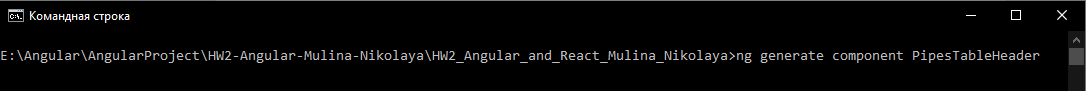
1. ***Перейдем в папку с созданным проектом HW2\_Angular\_and\_React\_Mulina\_Nikolaya:***



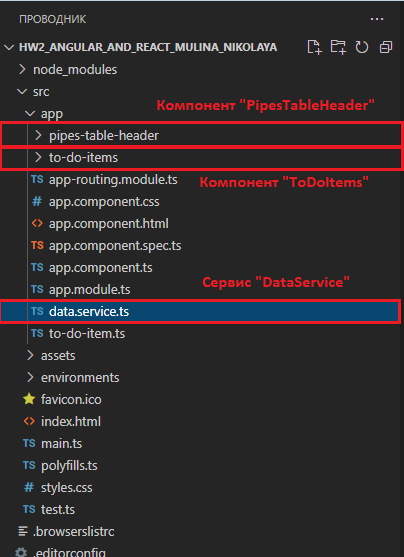
1. ***Создадим новый компонент ToDoItems:***



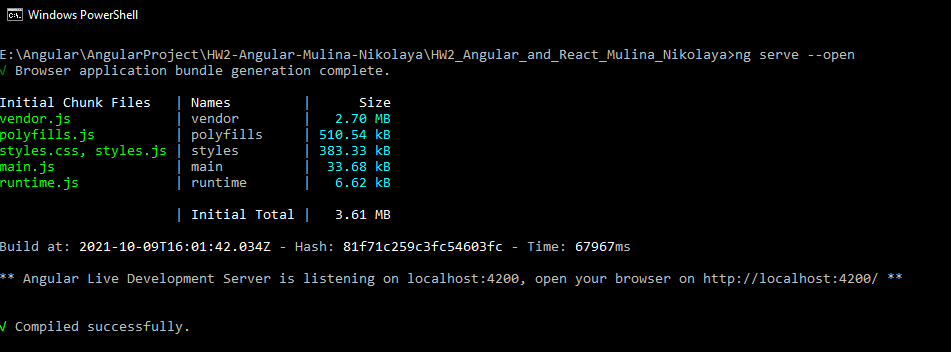
1. ***Создадим новый компонент PipesTableHeader:***



1. ***Сервис “DataService” создадим вручную (код в приложении см. ниже);***
2. ***Структура нашего проекта:***



1. ***Запустим локальный сервер:***



**Проект состоит из следующих файлов**

*(указаны только основные используемые скриптовые файлы проекта, а остальные -* [*тут*](https://github.com/Nickomu92/hw2_Angular)*):*

* ***app.module.ts:***

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

import { BrowserModule } from '@angular/platform-browser';

// Импортируем FormsModule

import { FormsModule } from '@angular/forms';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { ToDoItemsComponent } from './to-do-items/to-do-items.component';

import { PipesTableHeaderComponent } from './pipes-table-header/pipes-table-header.component';

// В imports нужно добавить FormsModule

@NgModule({

declarations: [

AppComponent,

ToDoItemsComponent,

PipesTableHeaderComponent

],

imports: [

BrowserModule,

FormsModule,

AppRoutingModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

* ***app.component.ts***

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

@Component({

selector: 'app-root',

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

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'Список заданий (основной компонент)';

}

* ***app.component.html***

<h1>{{title}}</h1>

<!-- добавляем дочерний компонент -->

<app-to-do-items></app-to-do-items>

<router-outlet></router-outlet>

* ***to-do-items.component.ts***

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

// Импортируем созданный сервис "DataService"

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

// Импортируем класс "ToDoItem"

import { ToDoItem } from '../to-do-item';

@Component({

selector: 'app-to-do-items',

templateUrl: './to-do-items.component.html',

styleUrls: ['./to-do-items.component.css'],

// Добавляем в коллекцию providers созданный сервис "DataService"

providers: [DataService]

})

export class ToDoItemsComponent implements OnInit {

items: ToDoItem[] = [];

name: string = "";

description: string = "";

/\* Задействуем встроенный в Angular механизм dependency injection и получим объект

сервиса "dataService" в конструкторе компонента \*/

constructor(private dataService: DataService){}

// Метод добавления в список "data" сервиса "dataService"

addItem(name: string, description: string): void {

// Если поля ввода пустые, то выходим

if(name == null || name.trim() == "" || description == null || description.trim() == "")

return;

// Иначе - добавляем в список "data", который находится в объекте сервиса "dataService"

this.dataService.addData(name, description);

}

/\* Для получения данных в массив "items" из массива "data" объекта сервиса "dataService"

реализуем метод ngOnInit() интерфейса OnInit \*/

ngOnInit() {

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

}

title = "Таблица заданий (дочерний компонент)";

}

* ***to-do-items.component.html***

<div class="page-header">

<h2>{{title}}</h2>

</div>

<div class="panel">

<!-- Форма для добавления заданий список -->

<div class="form-inline">

<div class="form-group">

<div class="col-md-8">

<!-- Директива NgModel с помощью переданной модели создает объект FormControl и

привязывает эту модель к созданному элементу формы. Объект FormControl отслеживает

значение модели, а также отвечает за валидацию этого значения и взаимодействие с пользователем.

Данная директива принимает переданную ей модель в качестве входного свойства, используем двунаправленную привязку. -->

<input type="text" class="form-control" [(ngModel)]="name" placeholder = "Название" />

</div>

</div>

<div class="form-group">

<div class="col-md-6">

<!-- Директива NgModel, используем двунаправленную привязку -->

<input type="text" class="form-control" [(ngModel)]="description" placeholder="Описание" />

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-8">

<button class="btn btn-default" (click)="addItem(name, description)">Добавить задание</button>

</div>

</div>

</div>

<!-- Добавим дочерний компонент PipesTableHeader c инструментами Pipes, которые позволяют форматировать отображаемые значения -->

<app-pipes-table-header></app-pipes-table-header>

<!-- Таблица для вывода списка заданий -->

<table class="table table-striped">

<thead>

<tr>

<th>Предмет</th>

<th>Описание</th>

<th>Выполнено</th>

</tr>

</thead>

<tbody>

<!-- Директива \*ngFor для вывода данных из массива items в таблицу -->

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

<td>{{item.name}}</td>

<td>{{item.description}}</td>

<!-- Директива NgModel, используем двунаправленную привязку -->

<td><input type="checkbox" [(ngModel)]="item.done" /></td>

</tr>

</tbody>

</table>

</div>

* ***to-do-item.ts***

// Класс с конструктором ToDoItem

export class ToDoItem {

name: string;

description: string;

done: boolean;

// Конструктор

constructor(name: string, description: string) {

this.name = name;

this.description = description;

this.done = false;

}

}

* ***pipes-table-header.component.ts***

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

@Component({

selector: 'app-pipes-table-header',

templateUrl: './pipes-table-header.component.html',

styleUrls: ['./pipes-table-header.component.css']

})

export class PipesTableHeaderComponent implements OnInit {

constructor() { }

ngOnInit(): void {}

text = "Состояние выполненных дел";

date = new Date();

}

* ***pipes-table-header.component.html***

<p>Таблица 1.1 - <span class="bold">{{text | uppercase}}</span> на <span class="tomato bold">{{date | date : "dd / MM / yyyy"}}</span> (<span class="bold">компонент с использованием каналов Pipes</span>)</p>

* ***data.service.ts (Сервис)***

// Импортируем декоратор Injectable

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

// Импортируем класс "ToDoItem"

import { ToDoItem } from './to-do-item';

// Указываем, что сервис может использоваться в других сервисах,

// для этого к классу сервиса применяем декоратор Injectable

@Injectable()

export class DataService {

// Начальный список заданий, который будет выводиться на страницу

private data: ToDoItem[] = [

{ name: "Выполнить ДЗ №1 с дисциплины 'Создание веб-приложений с использованием Angular и React'", description:"Тема 'Angular: основы', создать новый компонент", done: true },

{ name: "Спать", description:"Нужно отдохнуть, после выполненного ДЗ", done: true },

{ name: "Кушать", description:"Завтрак, нужно подкрепиться перед выполнением ДЗ", done: false },

{ name: "Выполнить ДЗ №1 с дисциплины 'Создание web–приложений, исполняемых на стороне сервера при помощи языка программирования PHP и технологии AJAX'", description:"Тема 'Базы данных и работа с MySQL', создать тренировочный сайт турфирмы", done: false }

];

// Метод для получения данных

getData(): ToDoItem[] {

return this.data;

}

// Метод для дабавления данных

addData(name: string, description: string) {

this.data.push(new ToDoItem(name, description));

}

}

***Результаты работы:***







