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

**Angular & React»**

Тема: Angular (основы).

**Домашняя работа №1**

Выполнил студент гр. ВПУ-911: Мулин Николай Сергеевич

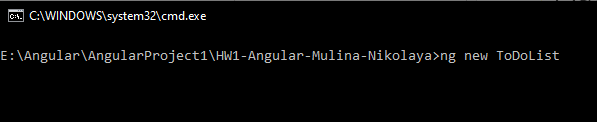
IDE: Microsoft Visual Studio Code

**Условие задания**

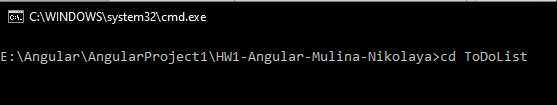
*Создать новый компонент в Angular.*

***Решение***

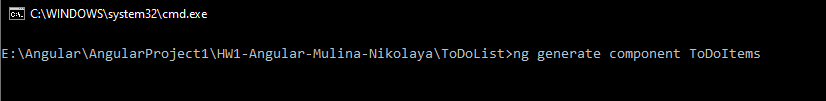
1. ***Создадим новый пространство для нового проекта ToDoList:***



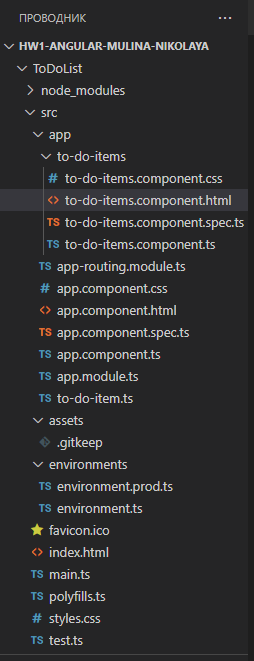
1. ***Перейдем в папку с созданным проектом ToDoList:***



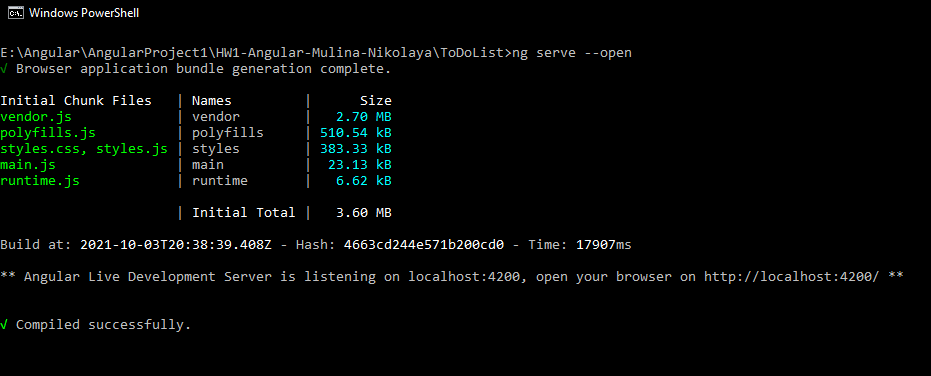
1. ***Создадим новый компонент ToDoItems:***



1. ***Структура нашего проекта:***



1. ***Запустим локальный сервер:***



***Список измененных файлов***

***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';

// В imports нужно добавить FormsModule

@NgModule({

declarations: [

AppComponent,

ToDoItemsComponent

],

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 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']

})

export class ToDoItemsComponent implements OnInit {

name: string = "";

description: string = "";

// Начальный список заданий, который будет выводиться на страницу

items: 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 добавления в список items

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

// Если поля пустые, то выходим

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

return;

// Иначе - добавляем в список

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

}

title = "Таблица заданий (дочерний компонент)";

ngOnInit() {

}

}

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

<style>

table {

border: 2px solid grey;

width: 600px;

border-collapse: collapsed;

text-align: center;

margin: 10px;

}

td {

border: 2px solid grey;

font-weight: bold;

}

td:hover {

background-color: dodgerblue;

color: white;

font-weight: bold;

}

button {

padding: 5px 10px;

background-color: dodgerblue;

opacity: 0.6;

color: white;

outline: none;

font-weight: bold;

border: 2px solid grey;

margin: 10px;

}

button:hover {

opacity: 1;

}

.form-inline > div {

display: inline-block;

}

input[type="text"] {

margin: 10px;

height: 1.4rem;

}

</style>

<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>

<!-- 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;

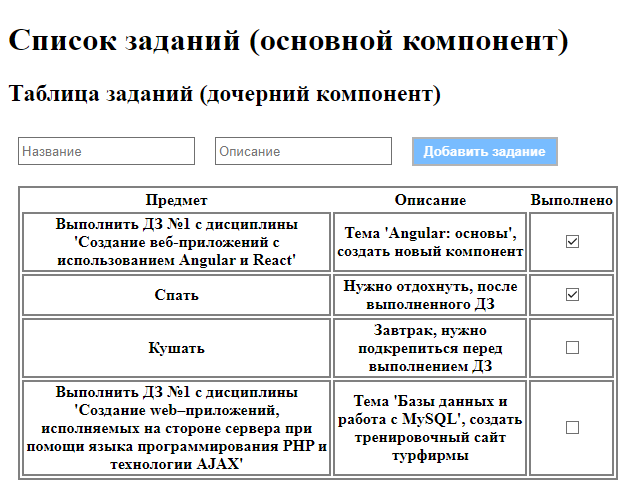
this.description = description;

this.done = false;

}

}

***Результат работы:***



***Добавим задание:***

