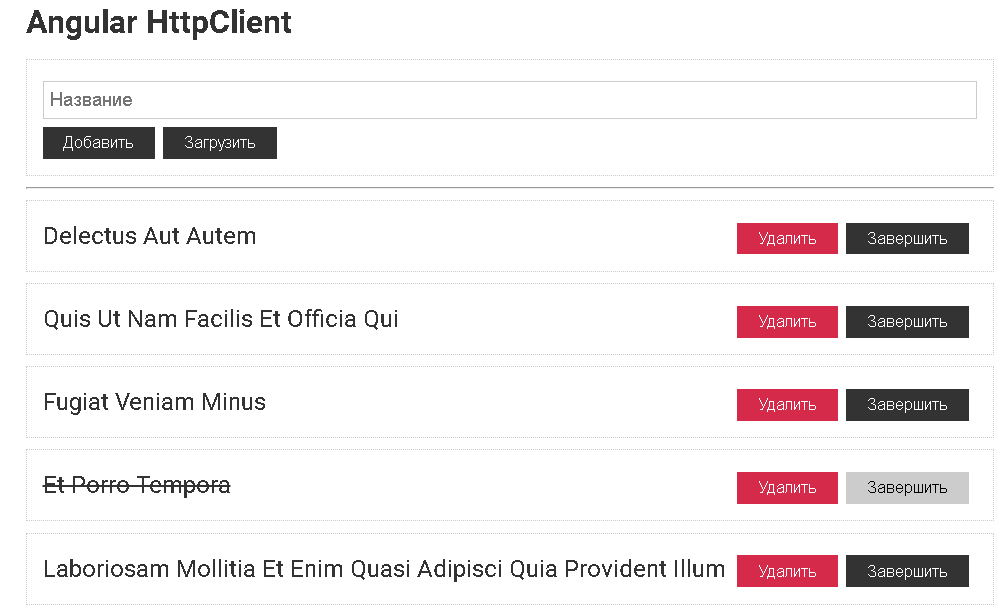
**Фреймворк Angular 9**

**Лабораторная работа № 6**

**Работа с сервером**

**Задача. Отправить запрос и получить ответ с сервера и вывести его в шаблоне компонента.**



**Ход работы**

1. **Создать новый проект Angular и в файле styles.scss прописать следующие стили:**

@import url('https://fonts.googleapis.com/css?family=Roboto');  
  
\* {  
 box-sizing: **border-box**;  
 margin: 0;  
 padding: 0;  
}  
  
body {  
 font-family: 'Roboto', **sans-serif**;  
 font-size: 1**rem**;  
 line-height: 1.6;  
 background-color: #fff;  
 color: #333;  
}  
  
.container {  
 max-width: 1000**px**;  
 margin: 0 **auto**;  
 padding: 1**rem**;  
}  
  
a {  
 text-decoration: **none**;  
}  
  
a:hover {  
 color: #666;  
}  
  
ul {  
 list-style: **none**;  
}  
  
img {  
 width: 100%;  
}  
  
.btn {  
 display: **inline-block**;  
 background: #333333;  
 color: #fff;  
 padding: 0.4**rem** 1.3**rem**;  
 font-size: 1**rem**;  
 border: **none**;  
 cursor: **pointer**;  
 margin-right: 0.5**rem**;  
 transition: **opacity** 0.2**s ease-in**;  
 outline: **none**;  
  
 &:disabled {  
 background: #ccc;  
 color: #000;  
 cursor: **not-allowed**;  
 }  
  
 &:hover {  
 opacity: 0.8;  
 }  
}  
  
.btn.btn-danger {  
 background: #dc3545;  
 color: #fff;  
}  
  
.form-control {  
 display: **block**;  
 margin-top: 0.3**rem**;  
 margin-bottom: .5**rem**;  
  
 label {  
 display: **block**;  
 font-weight: **bold**;  
 font-size: 1**rem**;  
 margin-bottom: .5**rem**;  
 }  
  
 input {  
 margin-bottom: 0;  
 }  
  
 .validation {  
 color: **red**;  
 }  
}  
  
.card {  
 padding: 1**rem**;  
 border: #ccc 1**px dotted**;  
 margin: 0.7**rem** 0;  
  
 p {  
 display: **flex**;  
 justify-content: **space-between**;  
  
 span.completed {  
 text-decoration: **line-through**;  
 }  
 }  
}  
  
input,  
select,  
textarea {  
 display: **block**;  
 width: 100%;  
 padding: 0.4**rem**;  
 font-size: 1.2**rem**;  
 border: 1**px solid** #ccc;  
 margin-bottom: 1**rem**;  
}  
  
hr {  
 margin: .5**rem** 0;  
}

**В шаблоне компонента app.component.http добавьте следующую структуру документа**

<div class**="container"**>  
 <h1>**Angular HttpClient**</h1>  
  
 <div class**="card"**>  
 <div class**="form-control"**>  
 <input type**="text"** placeholder**="Название"**>  
 </div>  
 <button class**="btn"**>**Добавить**</button>  
 <button class**="btn"**>**Загрузить**</button>  
 </div>  
  
 <hr>  
  
 <div class**="card"** >  
 <p>  
 <span class**="completed"**>**Название**</span>  
 <span>  
 <button class**="btn btn-danger"**>**Удалить**</button>  
 <button class**="btn"** >**Завершить**</button>  
 </span>  
 </p>  
 </div>  
  
</div>

1. Используя сайт <https://jsonplaceholder.typicode.com/> создадим get запрос на получение постов. Для удобства разработаем интерфейс Todo. Для этого в фале app.component.ts запишем следующий код.

import {*Component*, OnInit} from '@angular/core';  
import {HttpClient} from '@angular/common/http';

export interface Todo {  
 id?: number;  
 title: string;  
 completed: boolean;  
}

@Component({  
 selector: 'app-root',  
 templateUrl: './app.component.html',  
 styleUrls: ['./app.component.scss']  
})

export class AppComponent implements OnInit {  
   
 posts: Todo[] = [];  
   
  
 constructor(private http: HttpClient) {  
 }  
  
 ngOnInit(): void {  
 this.http.get<Todo[]>('https://jsonplaceholder.typicode.com/todos?\_limit=5')  
 .subscribe(response => {  
 console.log(response)  
 this.posts = response;  
 });  
  
 }  
  
 }

И реализуем в шаблоне вывод на экран полученной с «сервера» информации. В случае если пост был завершён (поле completed стоит в позиции true) заблокировать кнопку «Завершить».

<div class**="container"**>  
 <h1>**Angular HttpClient**</h1>  
 <div class**="card"**>  
 <div class**="form-control"**>  
 <input type**="text"** placeholder**="Название"** >  
 </div>  
 <button class**="btn"** >**Добавить**</button>  
 <button class**="btn"** >**Загрузить**</button>  
 </div>  
 <hr>  
 <div class**="card"** *\*ngFor***="**let *post* of posts**"**>  
 <p>  
 <span [class.completed]**="***post*.completed**"**>{{*post*.title | titlecase}}</span>  
 <span>  
 <button class**="btn btn-danger"**>**Удалить**</button>  
 <button class**="btn"** [disabled]**="***post*.completed**"**>**Завершить**</button>  
 </span>  
 </p>  
 </div>  
 </div>

Проверьте полученный результат.

**!!!Самостоятельно.**  **Реализовать сортировку полученных с сервера постов по алфавиту.**

1. Реализуем Post запрос на сервер для добавления постов и соответственно метод под кнопку «Добавить». Для этого в шаблоне компонента добавим обработку события по нажатию на мышку:

<input type**="text"** placeholder**="Название"** [(ngModel)]**="**namePost**"**>  
  
<button class**="btn"** (click)**="**addPost()**"**>**Добавить**</button>

А в самом компоненте реализуем метод addPost() с начальной проверкой пустого поля.

namePost = ''

addPost() {  
 if (!this.namePost.trim()) {  
 return  
 }  
 const post: Todo = {  
 title: this.namePost,  
 completed: false  
 }  
 this.http.post('https://jsonplaceholder.typicode.com/todos/', post)  
 .subscribe(res => {  
 console.log(res)  
 this.posts.unshift(post)  
 });  
  
}

1. Реализуем метод под кнопку «Загрузить», который выполнят по сути ту же работу по получению данных с сервера, которую уже реализовали в методе ngOnInit(). Также для имитации работы с сервером создадим loading на странице и задержку в 1,5 секунды. Для этого воспользуемся операторами rxjs.

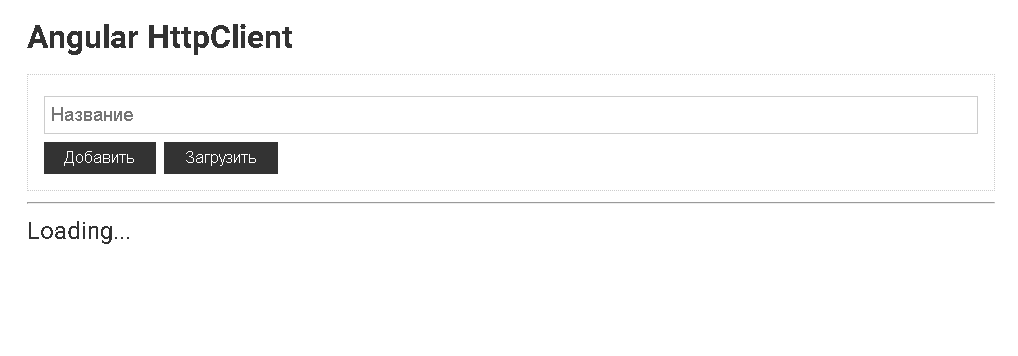
import {delay} from "rxjs/operators";

export class AppComponent implements OnInit {  
 namePost = ''  
 posts: Todo[] = [];  
 flagLoad = false;  
  
 constructor(private http: HttpClient) {  
 }  
  
 ngOnInit(): void {  
 this.loadPost()  
 }  
  
 addPost() {  
 if (!this.namePost.trim()) {  
 return  
 }  
 const post: Todo = {  
 title: this.namePost,  
 completed: false  
 }  
 this.http.post('https://jsonplaceholder.typicode.com/todos/', post)  
 .pipe(delay(1000))  
 .subscribe(res => {  
 console.log(res)  
 this.posts.unshift(post)  
 });  
  
 }  
  
 loadPost() {  
 this.flagLoad = true;  
 this.http.get<Todo[]>('https://jsonplaceholder.typicode.com/todos?\_limit=5')  
 .pipe(delay(1500))  
 .subscribe(response => {  
 console.log(response)  
 this.posts = response;  
 this.flagLoad = false;  
 });  
  
 }  
}

**В шаблоне компонента добавим имитацию loading**

<div class**="container"**>  
 <h1>**Angular HttpClient**</h1>  
 <div class**="card"**>  
 <div class**="form-control"**>  
 <input type**="text"** placeholder**="Название"** [(ngModel)]**="**namePost**"**>  
 </div>  
 <button class**="btn"** (click)**="**addPost()**"**>**Добавить**</button>  
 <button class**="btn"** (click)**="**loadPost()**"**>**Загрузить**</button>  
 </div>  
 <hr>  
 <div *\*ngIf***="**!flagLoad; else *loading***"**>  
 <div class**="card"** *\*ngFor***="**let *post* of posts**"**>  
 <p>  
 <span [class.completed]**="***post*.completed**"**>{{*post*.title | titlecase}}</span>  
 <span>  
 <button class**="btn btn-danger"**>**Удалить**</button>  
 <button class**="btn"** [disabled]**="***post*.completed**"**>**Завершить**</button>  
 </span>  
 </p>  
 </div>  
  
 </div>  
 <ng-template #*loading*>  
 <p>**Loading...**</p>  
 </ng-template>  
</div>

Теперь при загрузке страницы будет виден loading а через 1,5 сек. загрузятся посты.



1. Следующим элементом реализации формы является обработка кнопки «Удалить» для удаления отдельного поста. Для этого в компоненте реализуем следующий код

removePost(id: number) {  
 this.http.delete(`https://jsonplaceholder.typicode.com/todos/${id}`)  
.subscribe(() => {  
 console.log(id)  
 this.posts = this.posts.filter(item => item.id != id)  
 console.log(this.posts)  
});

}

А в шаблоне компонента поставим прослушку на нажатие кнопки

<button class**="btn btn-danger"** (click)**="**removePost(*post*.id)**"**>**Удалить**</button>

1. И последним элементов формы является кнопка «Завершить» для внесения изменений в пост. Для этого в компоненте запишем метод который посылает на сервер put запрос на внесения изменений.

completedPost(id: number) {  
 return this.http.put(`https://jsonplaceholder.typicode.com/todos/${id}`,{completed:true})

.subscribe((res) => {  
 this.posts.find(item=> item.id===res.id).completed=true  
  
})

}

А в шаблоне компонента поставим прослушку на нажатие кнопки

<button class**="btn"** [disabled]**="***post*.completed**"** (click)**="**copmletedPost(*post*.id)**"**>**Завершить**</button>