JS

Роутинг.

1. При создании приложения положительно ответим на вопрос о создании роутинга.

Это приведет к тому, что в проекте появится файл app-routing.module.ts в котором находится класс роутера и массив маршрутов. Массив маршрутов задает соответствие адресной строки и компонентов, которые будут показаны пользователю в теге

<router-outlet></router-outlet>

Подключим стили bootstrap к приложению.

Создадим несколько компонентов:

header – компонент, содержащий меню

main – компонент главной страницы

catalog – компонент каталога

item-edit – компонент редактора элемента списка каталога.

* 1. Header

В шаблоне компонента header создадим горизонтальное меню

header.component.html

<nav *class*="navbar navbar-expand-lg navbar-light bg-light">

  <a *class*="navbar-brand">Routing - lesson</a>

  <div *class*="collapse navbar-collapse" *id*="navbarNavAltMarkup">

    <div *class*="navbar-nav">

      <a *class*="nav-item nav-link" *[routerLink]*="'/'" *routerLinkActive*="active" *[routerLinkActiveOptions]*="{exact: true}">Главная</a>

      <!-- routerLinkActive используется для применения css класса к выбранному пункту меню, в данном случае active - сss класс -->

      <!-- [routerLinkActiveOptions]="{exact: true}" - используется, для того, чтобы указать полное и строгое соответствие адресу в строке и роуту -->

      <a *class*="nav-item nav-link" *[routerLink]*="'/catalog'" *routerLinkActive*="active">Каталог</a>

    </div>

  </div>

</nav>

Обратите внимание, что адрес ссылки следует записывать в свойство [routerLink], а не атрибут href (чтобы избежать перезагрузки страницы).

* 1. App-component

Внутри шаблона основного компонента разместим header, а также router-outlet, внутри которого будут отображаться компоненты, в соответствии с адресной строкой.

app.component.html

<!-- Компонент основного меню -->

<app-header></app-header>

<!-- В этом тего роутер отображает необходимые компоненты -->

<router-outlet></router-outlet>

* 1. Роутер

Сформируем массив маршрутов

app-routing.module.ts

const routes: *Routes* = [

  {path: '', component: MainComponent},

  {path: 'catalog', component: CatalogComponent},

  {path: 'edit/:id', component: ItemEditComponent}

];

path – адрес, component – компонент. Обратите внимание на адрес для компонента ItemEdit, в нем содержится параметр (под названием id), он нужен для того, чтобы было понятно, какой из элементов списка редактировать.

Обратите внимание на то, что класс роутера импортирован в модуль app.module.ts.

Запустим приложение в режиме разработки и посмотрим на его работу: при переходе по ссылкам под горизонтальным меню будут показаны компоненты main и catalog.

* 1. Создадим сервис, в котором будут содержаться данные для каталога.

ng g s services/item

items.service.ts

@Injectable({

  providedIn: 'root'

})

export class ItemsService {

  goods = [

    { id: 0, name: 'яблоко' },

    { id: 1, name: 'груша' },

    { id: 2, name: 'томат' }

  ];

  constructor() { }

  putItemById(*id*: *number*, *name*: *string*) {

*this*.goods.map(*good* => {

      if (good.id === id) {

        good.name = name;

      }

    });

  }

  getItemById(*id*: *number*) {

    return *this*.goods.find(*good* => good.id === id);

  }

}

* 1. Заполним каталог

catalog.component.ts

export class CatalogComponent implements *OnInit* {

  goods = [];

  constructor(

    private *itemsService*: *ItemsService*

  ) { }

  ngOnInit() {

*this*.goods = *this*.itemsService.goods;

  }

}

catalog.component.html

<div *class*="container">

  <p *\*ngFor*="let good of goods">

    <a *[routerLink]*="['/edit', good.id]">{{good.name}}</a>

  </p>

</div>

Обратите внимание на то, что в свойство [routerLink] передается параметр id, который был указан в маршрутизаторе.

Теперь при клике по элементу списка в адресной строке будет id элемента, осталось научить компонент ItemEdit понимать этот параметр и, в соответствии с ним, получать нужный элемент списка.

* 1. Сделаем форму для редактирования элемента списка.

Не забудьте импортировать в модуле

*ReactiveFormsModule*

item-edit.component.ts

@Component({

  selector: 'app-item-edit',

  templateUrl: './item-edit.component.html',

  styleUrls: ['./item-edit.component.css']

})

export class ItemEditComponent implements *OnInit* {

  id: *number*;

  itemForm: *FormGroup*;

  selectedItem;

  constructor(

    private *activatedRouter*: *ActivatedRoute*, // хранит url адрес и параметры

    private *router*: *Router*, // этот сервис используется для навигации

    private *itemsService*: *ItemsService*

  ) {

*this*.activatedRouter.params.subscribe(*param* => {

*this*.id = param.id;

    });

  }

  ngOnInit() {

*this*.selectedItem = *this*.itemsService.getItemById(+*this*.id);

*this*.itemForm = **new** *FormGroup*({

      'name': **new** *FormControl*(*this*.selectedItem.name)

    });

  }

  onSaveForm() {

*this*.itemsService.putItemById(+*this*.id, *this*.itemForm.value.name);

*this*.router.navigate([`/catalog`]); // метод navigate используется для программной навигации

  }

}

item-edit.component.html

<div *class*="container">

  <form *[formGroup]*="itemForm" *(ngSubmit)*="onSaveForm()">

    <div *class*="col-12">

      <div *class*="form-group">

        <input *type*="text" *formControlName*="name">

      </div>

    </div>

    <div *class*="col-12">

      <button *type*="submit" *class*="btn btn-primary">Сохранить</button>

    </div>

  </form>

</div>

Теперь наше приложение умеет редактировать элементы списка.

<https://github.com/KhDmitrii/project16>