Angular 2. Routing. Workshop.

Contents

Знакомство со структурой проекта

>git clone https://github.com/VZhyrytskiy/An2-3-Routing.git

Step 01. Basic Setup. <base> Tag.

Добавте тег base в файле ***index.html***

```
<title>Angular 2: Routing</title>
<base href="/">
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Если нет доступа к тегу <head> то можно добаить программно в **app.module.ts**

```
import {Component, NgModule} from '@angular/core';
import {APP_BASE_HREF} from '@angular/common';

@NgModule({
providers: [{provide: APP_BASE_HREF, useValue: '/my/app'}]
})
class AppModule {}
```

Step 02. Components

Создайте папку app/components и перейдите в нее в командной строке.

Создайте три компонента-заглушки:

1. HomeComponent (>ng g c home)
2. AboutComponent (>ng g c about)
3. PageNotFoundComponent (>ng g c page-not-found)

Step 03. Routes Config

Создайте файл **app.routing.ts**. Добавьте в него следующий фрагмент кода.

```
import { ModuleWithProviders } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { HomeComponent } from './components/home';
import { AboutComponent } from './components/about';
import { PageNotFoundComponent } from './components/page-not-found';

const appRoutes: Routes = [
  {
    path: '',
    redirectTo: 'home',
    pathMatch: 'full'
  },
  {
    path: 'home',
    component: HomeComponent
  },
  {
    path: 'about',
    component: AboutComponent
  },
  {
    // The router will match this route if the URL requested
    // doesn't match any paths for routes defined in our configuration
    path: '**',
    component: PageNotFoundComponent
  }
];

export const routing: ModuleWithProviders = RouterModule.forRoot(appRoutes);
```

Step 04. Import Routes

Добавьте в файле **app.module.ts** следующие фрагменты кода:

```
// 1
import { routing } from './app.routing';

// 2
import { HomeComponent } from './components/home';
import { AboutComponent } from './components/about';
import { PageNotFoundComponent } from './components/page-not-found';

// 3
```

Добавьте импортированные компоненты в раздел **declarations**

```
declarations: [
    TodoAppComponent,
    HomeComponent,
    AboutComponent,
    PageNotFoundComponent
]

// 4
```

Добавьте импортированный модуль routing в раздел imports

```
imports: [
    BrowserModule,
    CommonModule,
    FormsModule,
    routing
  ]
```

Step 05. <router-outlet>

В файле ***app.component.html*** добавьте директиву router-outlet

```
<div class="container">
    <router-outlet></router-outlet>
</div>
```

Step 06. routerLink

В файле ***app.component.html*** добавьте директиву routerLink

```
// 1
<a class="navbar-brand" routerLink="/">Task Manager</a>
// 2
<a routerLink="/about">About</a>
```

Step 07. routerLinkActive

В файле **app.component.html** добавьте директиву **routerLinkActive**

```
<li routerLinkActive="active">
      <a routerLink="/about">About</a>
</li>
```

Step 08. Task Feature Module

1. Создайте папку **models**
2. Создайте файл **models/task.ts** следующего содержания

```
export class Task {
      constructor(
            public id: number,
            public action: string,
            public priority: number,
            public estHours: number,
            public actHours?: number,
            public done?: boolean
      ) {
            this.actHours = 0;
            this.done = false;
      }
}
```

3. Создайте папку **tasks/task-array-service**
4. Создайте файл следующего содержания

```
import { Injectable } from '@angular/core';
import 'rxjs/add/operator/toPromise';

import { Task } from './../../models/task';

let taskList = [
  new Task(1, "Estimate", 1, 8, 8, true),
  new Task(2, "Create", 2, 8, 4, false),
  new Task(3, "Deploy", 3, 8, 0, false)
];

let taskListPromise = Promise.resolve(taskList);

@Injectable()
export class TaskArrayService {

  getTasks() {
    return taskListPromise;
  }

  getTask(id: number) {
    return this.getTasks()
      .then(tasks => tasks.find(task => task.id === id));
  }

  addTask(task: Task) {
    taskList.push(task);
  }

  updateTask(task: Task) {
    let i = -1;
```

```
    taskList.forEach((item, index) => {
      if (item.id === task.id ) {
        i = index;
        return false;
      }
    });

    if (i > -1) {
      taskList.splice(i,1,task);
    }
  }
}
```

5. В папке **tasks** создайте компонент **TaskListComponent** следующего содержания:

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

import { Task } from './../../models/task';
import { TaskArrayService } from './../task-array-service/task-array.service';

@Component({
  selector: 'task-list',
  templateUrl: 'task-list.component.html',
  styleUrls: ['task-list.component.css']
})
export class TaskListComponent implements OnInit {
  tasks: Array<Task>;

  constructor(
    private tasksService: TaskArrayService) { }

  ngOnInit() {
    console.log(this.tasks);
    this.tasksService.getTasks()
      .then(tasks => this.tasks = tasks);
  }

  completeTask(task: Task): void {
    task.done = true;
  }
}
```

6. Добавьте в файл темплейта **task-list.component.html** следующую разметку:

```
<task
  *ngFor='let task of tasks'
  [task]="task"
  (onComplete)="completeTask($event)">
</task>
```

7. В папке **tasks** создайте компонент **TaskComponent** следующего содержания:

```
import { Component, EventEmitter, Input, Output } from '@angular/core';
```

```
import { Task } from './../../models/task';

@Component({
  selector: 'task',
  templateUrl: 'task.component.html',
  styleUrls: ['task.component.css']
})
export class TaskComponent {
  @Input()  task: Task;
  @Output() onComplete = new EventEmitter<Task>();

  constructor() { }

  completeTask(event: any): void {
    this.onComplete.emit(this.task);
  }

  editTask(task: Task) {

  }
}
```

8. Добавьте в файл темплейта **task.component.html** следующую разметку:

```
<div class="panel panel-default">
      <div class="panel-heading">Task</div>
      <div class="panel-body">
            <ul>
                  <li>Action: {{task.action}}</li>
                  <li>Priority: {{task.priority}}</li>
                  <li>Estimate Hours: {{task.estHours}}</li>
                  <li>Actual Hours: {{task.actHours}}</li>
                  <li>Done: {{task.done}}</li>
            </ul>
            <button class="btn btn-primary"
                  (click)="completeTask($event)">
                  Done
            </button>
            <button class="btn btn-warning btn-sm"
                  (click)="editTask(task)">
                  Edit
            </button>
      </div>
</div>
```

9. Создайте файл **tasks.module.ts** следующего содержимого:

```
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';

import { TaskListComponent } from './task-list/task-list.component';
import { TaskComponent } from './task/task.component';
```

```
import { TaskArrayService } from './task-array-service/task-array.service';

@NgModule({
  declarations: [
    TaskListComponent,
    TaskComponent
  ],
  imports: [
    CommonModule,
    FormsModule
  ],
  providers: [
    TaskArrayService
  ]
})
export class TasksModule {}
```

Step_09. Tasks Feature Route Configuration

1. Создайте файл **tasks/tasks.routing.ts** следующего содержания:

```
import { ModuleWithProviders } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { TaskListComponent } from './task-list';

const tasksRoutes: Routes = [
  {
    path: 'task-list',
    component: TaskListComponent
  }
];

export const tasksRouting: ModuleWithProviders = RouterModule.forChild(tasksRoutes);
```

Step_10. Register Task Feature Routing

1. Добавьте следующий фрагмент кода

```
import { tasksRouting } from './tasks.routing';
```

2. Внесите изменения в следующий фрагмент кода:

```
imports: [
    CommonModule,
    FormsModule,
    tasksRouting
  ]
```

Step_11. Register Tasks Feature Module

1. Добавьте в файл app.module.ts следующий фрагмент кода:

```
import { TasksModule } from './tasks/tasks.module';
```

2. Внесите изменения в следующий фрагмент кода:

```
imports: [
    BrowserModule,
    CommonModule,
    FormsModule,
    TasksModule,
    routing
  ]
```

Step_12. Tasks List on Home Page

1. Внесите изменения в файл **tasks/tasks.routing.ts**

```
const tasksRoutes: Routes = [
  {
    path: 'task-list',
    path: 'home',
    component: TaskListComponent
  }
];
```

2. Внесите изменения в файл **app.routing.ts**

```
import { HomeComponent } from './components/home';

{
    path: 'home',
    component: HomeComponent
  },
```

3. Внесите изменения в файл app.module.ts

```
import { HomeComponent } from './components/home';

declarations: [
    TodoAppComponent,
    HomeComponent,
    AboutComponent,
    PageNotFoundComponent
  ]
```

4. Удалите компонент **HomeComponent** (папка components/home)

Step_13. Navigate

1. Создайте компонент **TaskFormComponent** в папке tasks используя следующий фрагмент кода:

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

import { Task } from './../../models/task';
import { TaskArrayService } from './../task-array-service/task-array.service';

@Component({
  selector: 'task-form',
  templateUrl: 'task-form.component.html',
  styleUrls: ['task-form.component.css']
})
export class TaskFormComponent implements OnInit, OnDestroy {
  task: Task;

  constructor(
    private tasksService: TaskArrayService,
  ) { }

  ngOnInit(): void {
    this.task = new Task(null, "", null, null);
  }

  ngOnDestroy(): void {
  }

  saveTask() {
    let task = new Task(
      this.task.id,
      this.task.action,
      this.task.priority,
      this.task.estHours
    );


    if (task.id) {
      this.tasksService.updateTask(task);
    }
    else {
      this.tasksService.addTask(task);
    }
  }

  goBack(): void {
  }
}
```

2. Создайте темплейт для компонента **TaskFormComponent** использу следующий фрагмент разметки

```
<div class="panel panel-default">
    <div class="panel-heading">
        <h4 class="pannel-title">
            Task Form
```

```
        </h4>
    </div>
    <div class="panel-body">
        <form *ngIf="task">
          <div class="form-group">
            <label for="action">Action</label>
            <input type="text"
                class="form-control"
                id="action" name="action"
                placeholder="Action"
                [(ngModel)]="task.action">
          </div>
          <div class="form-group">
            <label for="priority">Priority</label>
            <input type="number"
                min="1" max="3"
                class="form-control"
                id="priority" name="priority"
                placeholder="Priority"
                [(ngModel)]="task.priority">
          </div>
          <div class="form-group">
            <label for="estHours">Est. Hours</label>
            <input type="number"
                min="0"
                step="2"
                class="form-control"
                id="estHours" name="estHours"
                placeholder="Est. Hours"
                [(ngModel)]="task.estHours">
          </div>
          <button
            type="button"
            class="btn btn-primary"
            (click)="saveTask()">Save</button>
          <button class="btn btn-primary" (click)="goBack()">Back</button>
        </form>
    </div>
</div>
```

3. Внесите изменения в файл **tasks.module.ts**

```
import { TaskFormComponent }  from './task-form/task-form.component';

declarations: [
    TaskListComponent,
    TaskComponent,
    TaskFormComponent
]
```

4. Внесите изменения в файл **tasks/tasks.routing.ts**

```
import { TaskFormComponent } from './task-form';

const tasksRoutes: Routes = [
```

```
  {
    path: 'home',
    component: TaskListComponent
  },
  {
    path: 'edit/:id',
    component: TaskFormComponent
  }
];
```

5. Внесите изменения в компонент **TaskComponent**

```
import { Router } from '@angular/router';

constructor(
    private router: Router
  ) { }


editTask(task: Task) {
    let link = ['/edit', task.id];
    this.router.navigate(link);
  }
```

Step_14. Getting the route parameter

1. Внесите изменения в компонент **TaskFormComponent** в файле **tasks/task-form.component.ts**

```
import { ActivatedRoute } from '@angular/router';
import { Subscription } from 'rxjs/Subscription';
```

2. Добавьте приватное свойство

```
private sub: Subscription;
```

3. Внесите изменения в конструктор

```
constructor(
    private tasksService: TaskArrayService,
    private route: ActivatedRoute
  ) { }
```

4. Внесите изменения в ngOnInit

```
ngOnInit(): void {
    this.task = new Task(null, "", null, null);

    this.sub = this.route.params.subscribe(params => {
      let id = +params["id"];

      // NaN - for new task, id - for edit
      if (id) {
        this.tasksService.getTask(id)
          .then(task => this.task = Object.assign({}, task));
      }
    });
  }
```

5. Внесите изменения в ngOnDestroy

```
ngOnDestroy(): void {
    this.sub.unsubscribe();
}
```

Step_15. Navigate Back

1. Внесите изменения в компонент TaskFormComponent в файле **tasks/task-form.component.ts**

```
import { ActivatedRoute, Router } from '@angular/router';
```

2. Внесите изменения в конструктор

```
constructor(
    private tasksService: TaskArrayService,
    private router: Router,
    private route: ActivatedRoute
  ) { }
```

3. Внесите изменения в метод goBack()

```
goBack(): void {
    this.router.navigate(["home"]);
    // or
    // window.history.back();
  }
```

4. Внесите изменения в метод **saveTask()**

```
if (task.id) {
      this.tasksService.updateTask(task);
}
else {
      this.tasksService.addTask(task);
}

this.router.navigate(["home"]);
```

Step_16. Users Components

1. создайте файл **models/user.ts** используя слудющий фрагмент кода:

```
export class User {
      constructor(
            public id: number,
            public firstName: string,
            public lastName: string
      ) {}
}
```

2. Создайте папку **users**
3. Создайте сервис **UserArrayService** в файле users/user-array-service/user-array-service.ts используя следующий фрагмент кода

```
import { Injectable } from '@angular/core';
import 'rxjs/add/operator/toPromise';

import { User } from './../../models/user';

let userList = [
  new User(1, 'Anna', 'Borisova'),
  new User(2, 'Boris', 'Vlasov'),
  new User(3, 'Gennadiy', 'Dmitriev')
];

let userListPromise = Promise.resolve(userList);

@Injectable()
export class UserArrayService {
  getUsers() {
    return userListPromise;
  }

  getUser(id: number) {
    return this.getUsers()
      .then(users => users.find(user => user.id === id));
  }

  addUser(user: User) {
    userList.push(user);
  }

  updateUser(user: User) {
    let i = -1;

    userList.forEach((item, index) => {
      if (item.id === user.id ) {
        i = index;
        return false;
      }
    });

    if (i > -1) {
```

```
      userList.splice(i, 1, user);
    }
  }
}
```

4. Создайте компонент **UserListComponent** используя следующий код

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

import { User } from './../../models/user';
import { UserArrayService } from './../user-array-service/user-array.service';

@Component({
  selector: 'user-list',
  templateUrl: 'user-list.component.html',
  styleUrls: ['user-list.component.css']
})
export class UserListComponent implements OnInit, OnDestroy {
  users: Array<User>;

  constructor(
    private usersService: UserArrayService,
  ) { }

  ngOnInit() {
    this.usersService.getUsers()
          .then(users => this.users = users);
  }

  ngOnDestroy() {
  }
}
```

5. Создайте разметку для **UserListComponent** в файле **user-list.component.ts** используя следующий фрагмент разметки

```
<user
  *ngFor='let user of users'
  [user]="user">
</user>
```

6. Создайте компонент **UserComponent** используя слудующий фрагмент кода

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

import { User } from './../../models/user';

@Component({
  selector: 'user',
  templateUrl: 'user.component.html',
  styleUrls: ['user.component.css']
})
export class UserComponent {
  @Input() user: User;
```

```
  constructor(
  ) { }

  editUser(user: User) {
  }
}
```

7. Создайте разметку для **UserComponent** в файле **user.component.html** используя следующий фрагмент разметки

```
<div class="panel panel-default">
      <div class="panel-heading">User</div>
      <div class="panel-body">
            <ul>
                  <li>FirtsName: {{user.firstName}}</li>
                  <li>LastName: {{user.lastName}}</li>
            </ul>
            <button class="btn btn-warning btn-sm"
                  (click)="editUser(user)">
                  Edit
            </button>
      </div>
</div>
```

8. Создайте компонент **UserFormComponent** использу следующий фрагмент кода

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { ActivatedRoute } from '@angular/router';

import { Subscription } from 'rxjs/Subscription';

import { User } from './../../models/user';
import { UserArrayService } from './../user-array-service/user-array.service';

@Component({
  selector: 'user-form',
  templateUrl: 'user-form.component.html',
  styleUrls: ['user-form.component.css'],
})
export class UserFormComponent implements OnInit, OnDestroy {
  user: User;
  oldUser: User;
  private sub: Subscription;

  constructor(
    private usersService: UserArrayService,
    private route: ActivatedRoute,
  ) { }

  ngOnInit(): void {
    this.user = new User(null, '', '');

    this.sub = this.route.params.subscribe(params => {
```

```
      let id = +params["id"];

      // NaN - for new user, id - for edit
      if (id) {
        this.usersService.getUser(id)
          .then(user => {
            this.user = Object.assign({}, user);
            this.oldUser = user;
          });
      }
    });
  }

  ngOnDestroy(): void {
    this.sub.unsubscribe();
  }

  saveUser() {
    let user = new User(
      this.user.id,
      this.user.firstName,
      this.user.lastName
    );

    if (user.id) {
      this.usersService.updateUser(user);
      // if success
      this.oldUser = this.user;
    }
    else {
      this.usersService.addUser(user);
      // if success
      this.oldUser = this.user;
    }
  }

  goBack() {
  }
}

}
```

9. Создайте разметку для **UserFormComponent** в файле **user-form.component.html** используя следующий фрагмент разметки

```
<div class="panel panel-default">
    <div class="panel-heading">
        <h4 class="pannel-title">
            User Form
        </h4>
    </div>
    <div class="panel-body">
        <form *ngIf="user" (ngSubmit)="saveUser()" id="user-form">
          <div class="form-group">
            <label for="action">First Name</label>
```

```
            <input type="text"
                class="form-control"
                id="firstName" name="firstName"
                placeholder="First Name"
                [(ngModel)]="user.firstName">
          </div>
          <div class="form-group">
            <label for="priority">Last Name</label>
            <input type="text"
                class="form-control"
                id="lastName" name="lastName"
                placeholder="Last Name"
                [(ngModel)]="user.lastName">
          </div>
        </form>
        <button
            type="submit"
            class="btn btn-primary"
            form="user-form">Save
        </button>
        <button class="btn btn-primary"
            (click)="goBack()">Back
        </button>
    </div>
</div>
```

Step_17. Users Feature Area

1. Создайте компонент **UsersComponent** в папке **users** используя следующий фрагмент кода

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

@Component({
  selector: 'users',
  templateUrl: 'users.component.html',
  styleUrls: ['users.component.css']
})
export class UsersComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

2. Создайте разметку для **UsersComponent** в файле **users/users.component.html** используя следующий фрагмент разметки

```
<h2>Users</h2>
```

3. Создайте файл **users/users.module.ts** используя следующую разметку

```
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';

import { UserListComponent } from './user-list/user-list.component';
import { UserFormComponent } from './user-form/user-form.component';
import { UserComponent } from './user/user.component';

import { UserArrayService } from './user-array-service/user-array.service';

@NgModule({
  imports: [
    CommonModule,
    FormsModule,
  ],
  declarations: [
    UserListComponent,
    UserFormComponent,
    UserComponent,
  ],
  providers: [
    UserArrayService,
  ]
})
export class UsersModule {}
```

4. Внесите изменения в файл **app.module.ts**

```
import { TasksModule } from './tasks/tasks.module';
import { UsersModule } from './users/users.module';

imports: [
    BrowserModule,
    CommonModule,
    FormsModule,
    TasksModule,
    UsersModule,
    // Step 04
    routing
  ]
```

Step_18. Users Nested Routing

1. Внесите изменения в файл **app.component.html**

```
<div>
   <ul class="nav navbar-nav">
      <li routerLinkActive="active">
         <a [routerLink]="['/users']">Users</a>
      </li>

      <li routerLinkActive="active">
         <a [routerLink]="['/about']">About</a>
      </li>
   </ul>
</div>
```

2. Внесите изменения в файл **users/users.component.html**

```
<h2>Users</h2>
<router-outlet></router-outlet>
```

3. Создайте файл роутинга **users/users.routing.ts** использу следующий фрагмент кода

```
import { ModuleWithProviders }  from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { UsersComponent } from './users.component';
import { UserListComponent } from './user-list';
import { UserFormComponent } from './user-form';

const usersRoutes: Routes = [
  {
    path: 'users',
    component: UsersComponent,
    children: [
      {
        path: '',
        component: UserListComponent
      },
      {
        path: 'add',
        component: UserFormComponent
      },
      {
        path: 'edit/:id',
        component: UserFormComponent,
      }
    ]
  }
];

export const usersRouting: ModuleWithProviders = RouterModule.forChild(usersRoutes);
```

4. Внесите изменения в файл **users/users.module.ts**

```
import { usersRouting } from './users.routing';


imports: [
    CommonModule,
    FormsModule,
    usersRouting
  ]
```

Step_19. Relative Navigation

1. Внесите изменения в **UserComponent**

```
import { Router, ActivatedRoute } from '@angular/router';
```

2. Внесите изменения в конструктор

```
constructor(
    private router: Router,
    private route: ActivatedRoute
  ) { }
```

3. Внесите изменения в метод

```
editUser(user: User) {
    let link = ['users/edit', user.id];
    this.router.navigate(link);
    // or
    // let link = ['edit', user.id];
    // this.router.navigate(link, {relativeTo: this.route});
  }
```

1. Внесите изменения в компонент **UserFormComponent**

```
import { ActivatedRoute, Router } from '@angular/router';
```

2. Внесите изменения в конструктор

```
constructor(
    private usersService: UserArrayService,
    private route: ActivatedRoute,
    private router: Router
  ) { }
```

3. Внесите изменения в метод **saveUser**

```
if (user.id) {
      this.usersService.updateUser(user);
      // if success
      this.oldUser = this.user;
}
else {
      this.usersService.addUser(user);
      // if success
      this.oldUser = this.user;
}

this.router.navigate(['./../../'], { relativeTo: this.route});
```

4. Внесите изменения в метод goBack()

```
goBack() {
     this.router.navigate(['./../../'], { relativeTo: this.route});
  }
```

Step_20. Optional Parameters

1. Внесите изменения в метод saveUser компонента UserFormComponent

```
if (user.id) {
      this.usersService.updateUser(user);
      this.oldUser = this.user;
      this.router.navigate(['/users', {id: user.id}]);
    }
    else {
      this.usersService.addUser(user);
      this.oldUser = this.user;
      this.router.navigate(['/users']);
    }
this.router.navigate(['./../../'], { relativeTo: this.route});
```

2. Внесите изменения в компонент **UserListComponent**

```
import { ActivatedRoute } from '@angular/router';
import { Subscription } from 'rxjs/Subscription';
```

3. Добавьте приватные свойства

```
private selectedUserId: number;
private sub: Subscription;
```

4. Внесите изменения в конструктор

```
constructor(
    private usersService: UserArrayService,
    private route: ActivatedRoute
  ) { }
```

5. Внесите изменения в метод ngOnInit

```
ngOnInit() {
    this.usersService.getUsers()
          .then(users => this.users = users);

    // listen id from UserFormComponent
    this.sub = this.route.params
      .subscribe(params => {
        let id = +params['id'];
        if (id) {
          this.selectedUserId = +params['id'];
          console.log(`Last time you edit user with id ${this.selectedUserId}`);
        }
      });
  }
```

6. Внесите изменения в метод **ngOnDestroy**

```
ngOnDestroy() {
    this.sub.unsubscribe();
  }
```

Step_21. Admin Feature Area

1. создайте папку **admin**
2. сгенерируйте три компонента выполнив команду **ng g c component-name** --nospec в папке **admin:**
    a. AdminDashboardComponent
    b. ManageTasksComponent
    c. ManageUsersComponent
3. сгенерируйте компонент **AdminComponent** выполнив команду **ng g c component-name** --flat --nospec в папке **admin**
4. Удалите из селекторов этих компонентов префикс **app-**
5. Если компоненты добавились в файл **app.module.ts**, то удалите их оттуда: команды import и секцию declarations
6. Создайте файл **admin/admin.module.ts** используя следующий фрагмент кода

```
import { NgModule }       from '@angular/core';
import { CommonModule }   from '@angular/common';

import { AdminComponent }          from './admin.component';
import { AdminDashboardComponent } from './admin-dashboard/admin-dashboard.component';
import { ManageTasksComponent }    from './manage-tasks/manage-tasks.component';
import { ManageUsersComponent }    from './manage-users/manage-users.component';

@NgModule({
  imports: [
    CommonModule
  ],
  declarations: [
    AdminComponent,
    AdminDashboardComponent,
    ManageTasksComponent,
    ManageUsersComponent
  ]
})
export class AdminModule {}
```

6. Создайте темплейт для компонента AdminComponent используя следующий фрагмент разметки

```
<h3>Admin</h3>
<nav>
  <a routerLink="./" routerLinkActive="active"
    [routerLinkActiveOptions]="{ exact: true }">Dashboard</a>
  <a routerLink="./tasks" routerLinkActive="active">Manage Tasks</a>
  <a routerLink="./users" routerLinkActive="active">Manage Users</a>
</nav>
<router-outlet></router-outlet>
```

7. Создайте файл **admin.routing.ts** используя следующий фрагмент кода

```
import { ModuleWithProviders }  from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { AdminComponent }          from './admin.component';
import { AdminDashboardComponent } from './admin-dashboard/admin-dashboard.component';
```

```
import { ManageTasksComponent }    from './manage-tasks/manage-tasks.component';
import { ManageUsersComponent }    from './manage-users/manage-users.component';

const adminRoutes: Routes = [
  {
    path: 'admin',
    component: AdminComponent,
    children: [
      {
        path: '',
        children: [
          { path: 'users', component: ManageTasksComponent },
          { path: 'tasks', component: ManageUsersComponent },
          { path: '', component: AdminDashboardComponent }
        ]
      }
    ]
  }
];

export const adminRouting: ModuleWithProviders = RouterModule.forChild(adminRoutes);
```

8. Внесите измениния в файл **admin/admin.module.ts**

```
import { adminRouting } from './admin.routing';


imports: [
    CommonModule,
    adminRouting
  ]
```

9. Внесите изменения в файл **app.module.ts**

```
import { TasksModule } from './tasks/tasks.module';
import { UsersModule } from './users/users.module';
import { AdminModule } from './admin/admin.module';

imports: [
    BrowserModule,
    CommonModule,
    FormsModule,
    TasksModule,
    UsersModule,
    AdminModule,
    // Step 04
    routing
  ]
```

10. Внесите изменения в файл **app.component.html**

```
<li routerLinkActive="active">
    <a [routerLink]="['/users']">Users</a>
</li>
```

```
<li routerLinkActive="active">
    <a [routerLink]="['/admin']">Admin</a>
</li>
```

Step_22. canActivate Guard

1. Создайте защитника AuthGuard в папке guards используя следующий фрагмент кода

```
import { Injectable }  from '@angular/core';
import { CanActivate } from '@angular/router';

@Injectable()
export class AuthGuard implements CanActivate {
  canActivate() {
    console.log('CanActivateGuard is called');
    return true;
  }
}
```

2. Внесите изменения в файл **admin/admin.routing.ts**

```
import { AuthGuard } from './../guards/auth.guard';

const adminRoutes: Routes = [
  {
    path: 'admin',
    component: AdminComponent,
    canActivate: [AuthGuard]
    children: [
      {
        path: '',
        children: [
          { path: 'users', component: ManageTasksComponent },
          { path: 'tasks', component: ManageUsersComponent },
          { path: '', component: AdminDashboardComponent }
        ]
      }
    ]
  }
];
```

3. Внесите изменения в файл **app.module.ts**

```
import { AuthGuard } from './../guards/auth.guard';

providers: [
    AuthGuard
  ]
```

Step_23. Auth Service

1. Создайте сервис **AuthService** в файле **services/auth.service.ts** используя следующий фрагмент кода

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

import { Observable } from 'rxjs/Observable';
import 'rxjs/add/observable/of';
import 'rxjs/add/operator/do';
import 'rxjs/add/operator/delay';

@Injectable()
export class AuthService {
  isLoggedIn: boolean = false;

  // store the URL so we can redirect after logging in
  redirectUrl: string;

  login(): Observable<boolean> {
    return Observable.of(true).delay(1000).do(val => this.isLoggedIn = true);
  }

  logout(): void {
    this.isLoggedIn = false;
  }
}
```

2. Внесите изменения в файл **guards/auth.guard.ts**

```
import {
  CanActivate, Router,
  ActivatedRouteSnapshot,
  RouterStateSnapshot
}                     from '@angular/router';
import { AuthService } from './../services/auth.service';
```

3. Добавьте конструктор

```
constructor(
    private authService: AuthService,
    private router: Router
  ) {}
```

4. Добавьте метод checkLogin()

```
checkLogin(url: string): boolean {
    if (this.authService.isLoggedIn) { return true; }

    // Store the attempted URL for redirecting
    this.authService.redirectUrl = url;

    // Navigate to the login page with extras
    this.router.navigate(['/login']);
    return false;
  }
```

5. Внесите изменения в метод canActivate

```
canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean {
    let url: string = state.url;
    return this.checkLogin(url);
    console.log('CanActivateGuard is called');
    return true;
  }
```

6. Внесите изменения в файл **app.module.ts**

```
import { AuthGuard } from './guards/auth.guard';
import { AuthService } from './services/auth.service';

providers: [
    AuthGuard,
    AuthService
]
```

Step_24. Login Component

1. Сгенерируйте новый компонент **LoginComponent** в папке components, выполнив команду ng g c login --nospec
2. Удалите из селектора компонента префикс app-
3. Проверьте, чтобы компонент был добавлен в файл app.module.ts, елси его там нет, внесите слудующие изменения:

```
import { AboutComponent } from './components/about';
import { PageNotFoundComponent } from './components/page-not-found';
import { LoginComponent } from './components/login/login.component';

declarations: [
    TodoAppComponent,
    // Step 04
    AboutComponent,
    PageNotFoundComponent,
    LoginComponent
  ]
```

4. Внесите изменения в файл **app.routing.ts**

```
import { AboutComponent } from './components/about';
import { PageNotFoundComponent } from './components/page-not-found';
import { LoginComponent } from './components/login';

{
    path: 'about',
    component: AboutComponent
},
{
    path: 'login',
    component: LoginComponent
},
```

5. Внесите измнения в **LoginComponent**

```
import { Router }      from '@angular/router';
import { AuthService } from './../../services/auth.service';
```

6. Добавьте свойство

```
message: string;
```

7. Добавьте методы

```
setMessage() {
    this.message = 'Logged ' + (this.authService.isLoggedIn ? 'in' : 'out');
  }

  login() {
    this.message = 'Trying to log in ...';
    this.authService.login().subscribe(() => {
      this.setMessage();
      if (this.authService.isLoggedIn) {
        // Get the redirect URL from our auth service
```

```
        // If no redirect has been set, use the default
        let redirect = this.authService.redirectUrl ? this.authService.redirectUrl :
'/admin';
        // Redirect the user
        this.router.navigate([redirect]);
      }
    });
  }

  logout() {
    this.authService.logout();
    this.setMessage();
  }
```

8. Внесите изменения в конструктор **LoginComponent**

```
constructor(
    public authService: AuthService,
    public router: Router
  ) {
    this.setMessage();
  }
```

9. Добавьте разметку для компонента **LoginComponent** импользуя следующий фрагмент разметки

```
<h2>LOGIN</h2>
<p>State: {{message}}</p>
<p>
 <button (click)="login()"  *ngIf="!authService.isLoggedIn">Login</button>
 <button (click)="logout()" *ngIf="authService.isLoggedIn">Logout</button>
</p>
```

10. Внесите изменения в темплейт TodoAppComponent

```
<li routerLinkActive="active">
     <a [routerLink]="['/admin']">Admin</a>
 </li>
<li routerLinkActive="active">
     <a [routerLink]="['/login']">Login</a>
</li>
```

Step_25. canActivateChild Guard

1. Внесите изменения в guards/auth.guard.ts

```
import {
  CanActivate, CanActivateChild, Router,
  ActivatedRouteSnapshot,
  RouterStateSnapshot
}                       from '@angular/router';

export class AuthGuard implements CanActivate, CanActivateChild {
…
}
```

2. Добавьте метод canActivateChild()

```
canActivateChild(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean {
      console.log("canActivateChild Guard is called");
      return this.canActivate(route, state);
  }
```

3. Внесите изменения в файл admin/admin.routing.ts

```
{
        path: '',
        canActivateChild: [AuthGuard],
        children: [
          { path: 'users', component: ManageTasksComponent },
          { path: 'tasks', component: ManageUsersComponent },
          { path: '', component: AdminDashboardComponent }
        ]
}
```

Step_26. canDeactivate Guard

1. Создайте новый сервис в файле services/dialog.service.ts используя следующий фрагмент кода

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

@Injectable()
export class DialogService {

  confirm(message?: string) {
    return new Promise<boolean>(resolve => {
      return resolve(window.confirm(message || 'Is it OK?'));
    });
  };
}
```

2. Внесите изменения в файл app.module.ts

```
import { DialogService } from './services/dialog.service';

providers: [
    { provide: NgModuleFactoryLoader, useClass: AsyncNgModuleLoader },
    DialogService
    // Step 01
    // { provide: APP_BASE_HREF, useValue: '/' }
  ],
```

3. Создайте CanDeactivateGuard в файле guards/can-deactivate.guard.ts используя следующий фрагмент кода

```
import { Injectable }    from '@angular/core';
import { CanDeactivate } from '@angular/router';
import { Observable }    from 'rxjs/Observable';

export interface CanComponentDeactivate {
 canDeactivate: () => Observable<boolean> | Promise<boolean> | boolean;
}

@Injectable()
export class CanDeactivateGuard implements CanDeactivate<CanComponentDeactivate> {
  canDeactivate(component: CanComponentDeactivate) {
    return component.canDeactivate ? component.canDeactivate() : true;
  }
}
```

4. Внесите изменения в компонент UserFormComponent

```
import { DialogService }  from './../../services/dialog.service';

constructor(
    private usersService: UserArrayService,
    private route: ActivatedRoute,
    private router: Router,
    public dialogService: DialogService
  ) { }
```

```
canDeactivate(): Promise<boolean> | boolean {
    if (!this.oldUser || this.oldUser.firstName === this.user.firstName) {
      return true;
    }

    return this.dialogService.confirm('Discard changes?');
}
```

5. Внесите изменения в файл users/users.module.ts

```
import { CanDeactivateGuard }    from './../guards/can-deactivate.guard';

providers: [
    UserArrayService,
    CanDeactivateGuard
  ]
```

6. Внесите изменения в файл users.routing.ts

```
import { CanDeactivateGuard }    from './../guards/can-deactivate.guard';



{
        path: 'edit/:id',
        component: UserFormComponent,
        canDeactivate: [CanDeactivateGuard]
}
```

Step_27. resolve Guard

1. Создайте **UserResolveGuard** в файле **guards/user-resove.guard.ts** используя следующий фрагмент кода

```
import { Injectable } from '@angular/core';
import { Router, Resolve, ActivatedRouteSnapshot } from '@angular/router';

import { User } from './../models/user';
import { UserArrayService } from './../users/user-array-service/user-array.service';

@Injectable()
export class UserResolveGuard implements Resolve<User> {

  constructor(
    private userArrayService: UserArrayService,
    private router: Router
  ) {}

  resolve(route: ActivatedRouteSnapshot): Promise<User> | boolean {
    let id = +route.params['id'];

    return this.userArrayService.getUser(id).then(user => {
      // todo: check maybe -1 if id not found
      if (user) {
        return user;
      }
      else { // id not found
        this.router.navigate(['/users']);
        return false;
      }
    });
  }
}
```

2. Внесите изменения в файл **users/users.routing.ts**

```
import { CanDeactivateGuard } from './../guards/can-deactivate.guard';
import { UserResolveGuard }   from './../guards/user-resolve.guard';

{
        path: 'edit/:id',
        component: UserFormComponent,
        canDeactivate: [CanDeactivateGuard],
        resolve: {
          user: UserResolveGuard
        }
}
```

3. Внесите изменения в файл **users/users.module.ts**

```
import { UserArrayService } from './user-array-service/user-array.service';
import { CanDeactivateGuard }    from './../guards/can-deactivate.guard';
import { UserResolveGuard }    from './../guards/user-resolve.guard';

providers: [
```

```
    UserArrayService,
    CanDeactivateGuard,
    UserResolveGuard
]
```

4. Внесите изменения в UserFormComponent

```
import { Subscription } from 'rxjs/Subscription';

private sub: Subscription;

ngOnInit(): void {
    this.user = new User(null, '', '');

    this.route.data.forEach((data: { user: User }) => {
      this.user = Object.assign({}, data.user);
      this.oldUser = data.user;
    });

    this.sub = this.route.params.subscribe(params => {
      let id = +params["id"];

      // NaN - for new user, id - for edit
      if (id) {
        this.usersService.getUser(id)
          .then(user => {
            this.user = Object.assign({}, user);
            this.oldUser = user;
          });
      }
    });
}

ngOnDestroy(): void {
    this.sub.unsubscribe();
}
```

Step_28. Query Parameters and Fragment

1. Внесите изменения в файл guards/auth.guard.ts

```
import { CanActivate, CanActivateChild, Router,
  ActivatedRouteSnapshot, RouterStateSnapshot, NavigationExtras
} from '@angular/router';
```

2. Внесите изменения в метод **checkLogin()** в файле **guards/auth.guard.ts**

```
checkLogin(url: string): boolean {
    if (this.authService.isLoggedIn) { return true; }

    // Store the attempted URL for redirecting
    this.authService.redirectUrl = url;

    // Create a dummy session id
    let sessionId = 123456789;

    let navigationExtras: NavigationExtras = {
      queryParams: { 'session_id': sessionId },
      fragment: 'anchor'
    };

    // Navigate to the login page with extras
    this.router.navigate(['/login'], navigationExtras);
    return false;
  }
```

3. Внесите изменения в LoginComponent

```
import { Router, NavigationExtras } from '@angular/router';


if (this.authService.isLoggedIn) {
        let redirect = this.authService.redirectUrl
            ? this.authService.redirectUrl : '/admin';

        let navigationExtras: NavigationExtras = {
          preserveQueryParams: true,
          preserveFragment: true
        };

        // Redirect the user
        this.router.navigate([redirect], navigationExtras);
}
```

4. Добавьте в темплейт AdminDashboardComponent следующий фрагмент разметки

```
<p>Session ID: {{ sessionId | async }}</p>
<a id="anchor"></a>
<p>Token: {{ token | async }}</p>
```

5. Внесите изменения в AdminDashboardComponent

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute }     from '@angular/router';
import { Observable }         from 'rxjs/Observable';
import 'rxjs/add/operator/map';
```

6. Добавьте свойства

```
sessionId: Observable<string>;
token: Observable<string>;
```

7. Внесите изменения в контроллер

```
constructor(
    private route: ActivatedRoute
  ) { }
```

8. Внесите изменения в метод ngOnInit()

```
ngOnInit() {
    this.sessionId = this.route
      .queryParams
      .map(params => params['session_id'] || 'None');

    this.token = this.route
      .fragment
      .map(fragment => fragment || 'None');
  }
```

9. Внесите изменения в темплейт компонента AdminComponent

```
<a routerLink="./" routerLinkActive="active"
   [routerLinkActiveOptions]="{ exact: true }"
   preserveQueryParams preserveFragment>Dashboard</a>
<a routerLink="./tasks" routerLinkActive="active"
   preserveQueryParams preserveFragment>Manage Tasks</a>
<a routerLink="./users" routerLinkActive="active"
   preserveQueryParams preserveFragment>Manage Users</a>
```

Step_29. Lazy-Loading Route Configuration

1. Внесите изменения в файл app.routing.ts

```
import { load } from './async-ng-module-loader';

  // systemjs case
  // {
  //   path: 'admin',
  //   loadChildren: 'app/admin/admin.module#AdminModule'
  // },

  // webpack case
  {
    path: 'admin',
    loadChildren: load( () =>
      new Promise(resolve => {
        (require as any).ensure(
            [],
            require => {
              resolve(require('./admin/admin.module').AdminModule);
            }
        );
      })
    )
  },
```

2. Внесите изменения в файл admin/admin.routing.ts

```
const adminRoutes: Routes = [
  {
    ~~path: 'admin',~~
    path: '',
    component: AdminComponent,
    canActivate: [AuthGuard],
    children: [
      {
        path: '',
        canActivateChild: [AuthGuard],
        children: [
          { path: 'users', component: ManageTasksComponent },
          { path: 'tasks', component: ManageUsersComponent },
          { path: '', component: AdminDashboardComponent }
        ]
      }
    ]
  }
];
```

3. Внесите изменения в файл app.module.ts

```
~~import { AdminModule } from './admin/admin.module';~~

imports: [
    BrowserModule,
    CommonModule,
```

```
  FormsModule,
  TasksModule,
  UsersModule,
  AdminModule,
  // Step 04
  routing
],
```

Step_30. canLoad Guard

1. Внесите изменения в AuthGuard в файле guards/auth.guard.ts

```
import {
  CanActivate, CanActivateChild, CanLoad, Router, Route,
  ActivatedRouteSnapshot, RouterStateSnapshot, NavigationExtras
} from '@angular/router';
```

2. Внесите изменения в описание класса

```
export class AuthGuard implements CanActivate, CanActivateChild, CanLoad {
```

3. Добавьте метод

```
canLoad(route: Route): boolean {
    let url = `/${route.path}`;
    return this.checkLogin(url);
}
```

4. Внесите изменения в файл app.routing.ts

```
import { AuthGuard } from './guards/auth.guard';

// webpack case
  {
    path: 'admin',
    canLoad: [AuthGuard],
    loadChildren: load( () =>
      new Promise(resolve => {
        (require as any).ensure(
            [],
            require => {
              resolve(require('./admin/admin.module').AdminModule);
            }
        );
      })
    )
  },
```