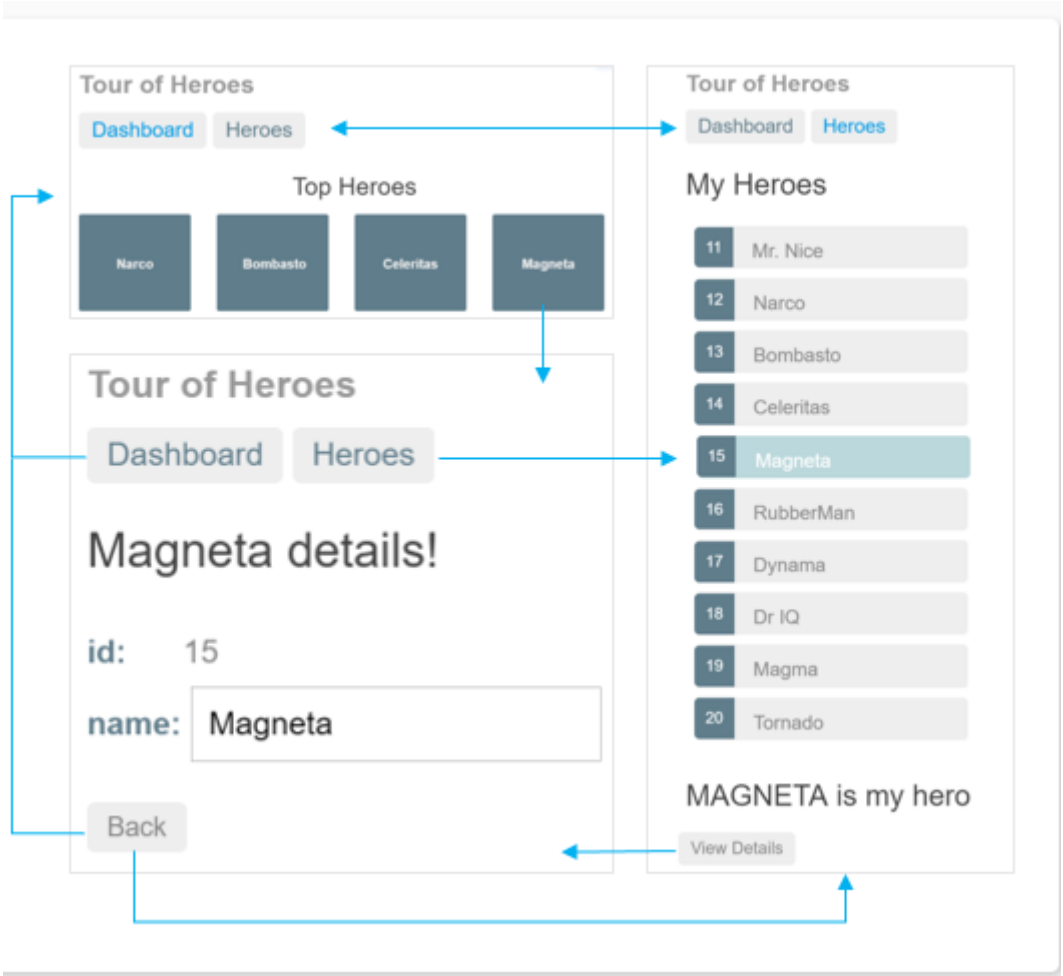
آموزش Angular6_آموزش Routing در Angular6

الزامات جدیدی برای اپلیکیشن Tour of Heroes وجود دارد:

- اضافه کردن Dashboard view
- اضافه کردن قابلیت حرکت بین Hero ها وDashboard view
- هنگامی که کاربر در هر کدام از view ها بر روی نام Hero کلیک می کند به detail view هیروی انتخاب شده دسترسی پیدا کند.
- هنگامی که کاربر بر روی یک لینک در ایمیل خود کلیک می کند detail view یک Hero خاص نمایش داده شود.

در نهایت کاربر می تواند مانند زیر به برنامه دسترسی داشته باشد:



آموزش اضافه کردن AppRoutingmodule

بهترین کاربرد Angular این است که router را در یک ماژول جداگانه و سطح بالا که به routing اختصاص دارد و توسط root AppModule , import شده است ، بارگیری و پیکربندی کند.

طبق قرارداد، نام کلاس ماژول AppRoutingModule است و در app-routing.module.ts در پوشه src / app قرار دارد.

برای ایجاد آن از CLI استفاده کنید.

```
ng generate module app-routing --flat --module=app
```

ـ ـflat فایل را به جای پوشه خود در src / app قرار می دهد.

ـ ـmodule = app به CLI می گوید که آن را درimports array مربوط به AppModule ثبت کند.

فایل ایجاد شده به این صورت است:

```
import { NgModule } from '@angular/core'; import { CommonModule } from '@angular/common'; @NgModule({ imports: [ CommonModule ], declarations: [] }) export class AppRoutingModule { }
```

شما برای اینکه بتوانید آرایه @Ngmodule.declarations و رفرنس های CommonModule را حذف کنید، عموما نیازی ندارید که اجزای routing module را اعلام کنید.

همچنین router را با Routes در routerModule پیکربندی خواهید کرد بنابراین می توانید این دو را از کتابخانه ی @angular/router وارد کنید.

یک آرایه @NgModule.exports با RouterModule در آن ایجاد کنید.

Export کردن RouterModule باعث می شود دستورات router برای استفاده ی اجزای AppModule در دسترس قرار گیرند.

اکنون AppRoutingModule به این صورت است:

```
import { NgModule } from '@angular/core'; import { RouterModule, Routes } from '@angular/router'; @NgModule({ exports: [ RouterModule ] }) export class AppRoutingModule {}
```

آموزش اضافه کردنroute ها

هنگامی که کاربر بر روی لینکی کلیک کند یا نشانی اینترنتی را در نوار آدرس مرورگر قرار می دهد، Route ها به Router دستور میدهند که کدام view نمایش داده شود.

یک Angular Route معمولی دو ویژگی دارد:

1. مسیر(path): یک رشته که URL را در نوار آدرس مرورگر مطابقت می دهد.
2. کامپوننت: کامپوننتی است که که روتر باید هنگام رفتن به این route آن را ایجاد کند.

شما زمانی باید به HeroesComponent بروید که شکل URL چیزی مانند localhost:4200/heroes است.

HeroesComponent را import کنید تا بتوانید آن را در یک Route ارجاع دهید. سپس آرایه ای از route ها را تنها با یک route به سمت این کامپوننت تعریف کنید.

```
import { HeroesComponent } from './heroes/heroes.component'; const routes: Routes = [ { path: 'heroes', component: HeroesComponent } ];
```

پس از اتمام نصب، روتر آن آدرس URL را با مسیر: 'Heroes' مطابقت داده و HeroesComponent را نمایش می دهد.

آموزش RouterModule.forRoot()

اول باید روتر را راه اندازی (initialize) کنید و آن را آماده عکس العمل نشان دادن به شروع به تغییرات مکان مرورگر کنید.

RouterModule را به آرایه @NgModule.imports اضافه کنید و آن را با routes در یک مرحله با فراخوانی RouterModule.forRoot() در آرایه imports پیکربندی کنید. مانند این:

```
imports: [ RouterModule.forRoot(routes) ],
```

به این متد، forRoot() گفته می شود. زیرا شما روتر را در سطح ریشه نرم افزار پیکربندی می کنید. متد forRoot() ارائه دهندگان خدمات (service providers) و دستورالعمل های لازم برای Routing را فراهم می کند و گشت و گذار اولیه را بر اساس URL مرورگر فعلی اجرا می کند.

آموزش اضافه کردن RouterOutlet

قالب AppComponent را باز کنید و المان <app-heroes> را با المان <router-outlet> جایگزین کنید.

src/app/app.component.html (router-outlet)

```
<h1>{{title}}</h1><router-outlet></router-outlet><app-messages></app-messages>
```

شما <app-heroes> را حذف کردید زیرا می خواهیم هنگامی که کاربر آن را باز می کند فقط HeroesComponent نمایش داده شود.

<router-outlet> به router دستور می دهد که view های route شده را کجا نمایش دهد.

RouterOutlet یکی از دستورات روتر است که برای AppComponent در دسترس قرار گرفته است زیرا AppModule ، AppRoutingModule را وارد می کند که AppRoutingModule ، RouterModule را export کرده است.

امتحان کنید

شما همچنان باید با فرمان CLI از برنامه Run بگیرید.

ng serve

- مرورگر باید به روز شود و عنوان اپلیکیشن را نمایش دهد، نه فهرست Hero ها.
- به نوار آدرس مرورگر نگاه کنید. URL به / ختم می شود. مسیر منتهی به HeroesComponent با heroes/ نمایش داده شده است.
- در نوار آدرس مرورگر heroes/ را به URL اضافه کنید.
- حالا باید heroes master/detail view را مشاهده کنید.
- یک navigation link اضافه کنید (routerLink)

کاربران نباید route URL را در نوار آدرس بارگذاری کنند. آنها باید بتوانند برای گشت و گذار در نرم افزار روی یک لینک کلیک کنند.

المان <nav> را اضافه کنید و در آن یک المان anchor است که وقتی کلیک میکنید، شما را به HeroesComponent هدایت می کند. قالب AppComponent اصلاح شده مانند زیر است:

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

<nav>

<a routerLink="/heroes">Heroes</a>

</nav>
```

```
<router-outlet></router-outlet>

<app-messages></app-messages>
```

مشخصه routerLink بر روی عبارت "/heroes" قرار گرفته است . این عبارت رشته ای است که router مسیر را با HeroesComponent مطابقت می دهد. routerLink انتخابگر دستورالعمل RouterLink است که کلیک کردن کاربر را به router navigation تبدیل می کند. این یکی دیگر از دستورالعمل های عمومی در RouterModule است.

مرورگر بروزرسانی می شود و عنوان اپلیکیشن و لینک Hero ها را نمایش می دهد اما لیست Hero ها را نشان نمی دهد.

بر روی لینک کلیک کنید. نوار آدرس به heroes/ آپدیت می شود و لیست hero ها نمایش داده می شود.

این لینک ها و navigation link های بعدی را با اضافه کردن سبک های خصوصی CSS به app.component.css ارتقا دهید(همانطور که در بازبینی نهایی کد در ادامه ذکر شده است.)

یک dashboard view اضافه کنید.

هنگامی که view های متعددی وجود دارد Routing منطقی تر به نظر میرسد. تاکنون فقط heroes view وجود داشته است.

یک DashboardComponent را با استفاده از CLI اضافه کنید:

```
ng generate component dashboard
```

CLI فایل ها را برای DashboardComponent ایجاد می کند و آن را در AppModule اعلان می کند.

محتوای فایل پیشفرض را در این سه فایل به صورت زیر جایگزین کنید و سپس به ادامه ی بحث می پردازیم:

src/app/dashboard/dashboard.component.html

```
<h3>Top Heroes</h3>
<div class="grid grid-pad">
<a *ngFor="let hero of heroes" class="col-1-4">
<div class="module hero">
<h4>{{hero.name}}</h4>
</div>
</a>
</div>
```

src/app/dashboard/dashboard.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Hero } from '../hero';
import { HeroService } from '../hero.service';

@Component({
selector: 'app-dashboard',
templateUrl: './dashboard.component.html',
styleUrls: [ './dashboard.component.css' ]
})
export class DashboardComponent implements OnInit {
heroes: Hero[] = [];

constructor(private heroService: HeroService) { }

ngOnInit() {
this.getHeroes();
}

getHeroes(): void {
this.heroService.getHeroes()
.subscribe(heroes => this.heroes = heroes.slice(1, 5));
}
}
```

src/app/dashboard/dashboard.component.css

```
/* DashboardComponent's private CSS styles */
[class*='col-'] {
float: left;
padding-right: 20px;
padding-bottom: 20px;
}
```

```
[class*='col-']:last-of-type {
padding-right: 0;
}
a {
text-decoration: none;
}
*, *:after, *:before {
-webkit-box-sizing: border-box;
-moz-box-sizing: border-box;
box-sizing: border-box;
}
h3 {
text-align: center; margin-bottom: 0;
}
h4 {
position: relative;
}
.grid {
margin: 0;
}
.col-1-4 {
width: 25%;
}
.module {
padding: 20px;
text-align: center;
color: #eee;
max-height: 120px;
min-width: 120px;
background-color: #607d8b;
border-radius: 2px;
}
.module:hover {
background-color: #eee;
cursor: pointer;
color: #607d8b;
```

```
}
.grid-pad {
padding: 10px 0;
}
.grid-pad > [class*='col-']:last-of-type {
padding-right: 20px;
}
@media (max-width: 600px) {
.module {
font-size: 10px;
max-height: 75px; }
}
@media (max-width: 1024px) {
.grid {
margin: 0;
}
.module {
min-width: 60px;
}
}
```

این قالب یک شبکه از لینک نام Hero ها را ارائه می دهد.

- تکرار کننده ی ngFor* به همان اندازه که لینک در آرایه heroes وجود دارد، اقدام به تولید لینک می کند.
- لینک ها به صورت بلوک های رنگی توسط dashboard.component.css طراحی شده اند.
- لینک ها به هیچ جایی متصل نیستند ، اما به زودی خواهند شد.

این کلاس مشابه کلاس HeroesComponent است.

- کلاس مذکور یک heroes array property را تعریف می کند.
- Constructor از Angular انتظار دارد که HeroService را در private heroService property تزریق کند.
- قلاب چرخه ی عمر (lifecycle hook calls) ngOnInit() ، getHeroes را فراخوانی می کند.

این getHeroes تعداد Hero های نمایش داده شده را به چهار عدد (دوم، سوم، چهارم و پنجم) کاهش می دهد.

```
getHeroes(): void {
this.heroService.getHeroes()
.subscribe(heroes => this.heroes
= heroes.slice(1, 5));
}
```

dashboard route را اضافه کنید.

برای حرکت به داشبورد، روتر نیاز به یک مسیر مناسب دارد.

DashboardComponent را در AppRoutingModule ، import کنید.

```
import { DashboardComponent } from
'./dashboard/dashboard.component';
```

یک مسیر را به آرایه ی AppRoutingModule.routes اضافه کنید. به گونه ای که این مسیر را با DashboardComponent مطابقت دهد.

```
{ path: 'dashboard', component: DashboardComponent },
```

یک route پیش فرض اضافه کنید.

هنگامی که برنامه شروع می شود، نوار آدرس مرورگر به روت وب سایت(web site's root) اشاره می کند. این با هیچ مسیر موجودی مطابقت ندارد بنابراین روتر به هیچ وجه حرکتی ندارد. فضای زیر <router-outlet> خالی است.

برای اینکه برنامه به طور خودکار به داشبورد حرکت کند، مسیر زیر را به آرایه AppRoutingModule.Routes اضافه کنید.

```
{ path: '', redirectTo: '/dashboard', pathMatch: 'full' },
```

این روت یک URL که با یک مسیر خالی مطابقت دارد را به روتی که مسیرش "/dashboard" است هدایت می کند.

پس از اینکه مرورگر refresh می شود، روتر DashboardComponent را بارگذاری می کند و نوار آدرس مرورگر dashboard URL/ را نشان می دهد.

dashboard link را به shell اضافه کنید.

کاربر باید بتواند با کلیک کردن روی navigation area در بالای صفحه بین DashboardComponent و HeroesComponent به عقب و جلو حرکت کند.

یک dashboard navigation link به قالب پوسته ی AppComponent دقیقا بالای Heroes link اضافه کنید.

src/app/app.component.html

```
<h1>{{title}}</h1>
<nav>
<a routerLink="/dashboard">Dashboard</a>
<a routerLink="/heroes">Heroes</a>
</nav>
<router-outlet></router-outlet>
<app-messages></app-messages>
```

پس از اینکه مرورگر refresh می شود شما می توانید به راحتی با کلیک بر روی لینک ها بین دو view حرکت کنید.

آموزش رفتن به hero details

HeroDetailsComponent جزئیات یک Hero انتخاب شده را نمایش می دهد. در این لحظه HeroDetailsComponent تنها در پایین HeroesComponent قابل مشاهده است.

کاربر به سه طریق باید بتواند به این جزئیات دسترسی پیدا کند.

1. با کلیک کردن روی یک Hero در dashboard
2. با کلیک کردن رو یک Hero در heroes list
3. با paste کردن URL " deep link " در نوار آدرس مرورگر که Hero را برای نمایش شناسایی می کند.

در این بخش، امکان رفتن به HeroDetailsComponent را فعال کنید و آن را از HeroesComponent آزاد کنید.

hero details را از HeroesComponent حذف کنید.

هنگامی که کاربر بر روی یک آیتم hero در HeroesComponent کلیک می کند، برنامه باید به HeroDetailComponent برود و heroes list view را با hero detail view جایگزین کند. heroes list view دیگر نباید hero details را همانطور که در حال حاضر است نشان دهد.

- قالب HeroesComponent را باز کنید (heroes/heroes.component.html) و المان <app-hero-detail> را از پایین پاک کنید.
- فعلا کلیک کردن بر روی آیتم Hero کاری انجام نمی دهد. بعد از اینکه روتر را در HeroDetailComponent فعال کنید، حل می شود.

یک hero detail route اضافه کنید.

یک نشانی اینترنتی مانند ~/detal/11 می تواند یک URL خوب برای رفتن به Hero Detail view از Hero ای باشد که id آن 11 است. AppRoutingModule را باز کنید و HeroDetailComponent را import کنید.

src/app/app-routing.module.ts (import HeroDetailComponent)

```
import { HeroDetailComponent } from './hero-detail/hero-detail.component';
```

سپس مسیر پارامتری را به آرایه AppRoutingModule.routes اضافه کنید به گونه ای که الگوی مسیر (path pattern) را با hero detail view مطابقت دهد.

```
{ path: 'detail/:id', component: HeroDetailComponent },
```

کولون (:) در مسیر نشان می دهد که: id یک placeholder برای یک hero id خاص است.

در این مرحله، تمام مسیرهای برنامه در جای مناسب خود قرار دارند.

src/app/app-routing.module.ts (all routes)

```
const routes: Routes = [
{ path: '', redirectTo: '/dashboard', pathMatch: 'full' },
 { path: 'dashboard', component: DashboardComponent },
 { path: 'detail/:id', component: HeroDetailComponent },
 { path: 'heroes', component: HeroesComponent }
 ];
```

آموزش لینک های DashboardComponent hero

لینک های DashboardComponent hero در حال حاضر هیچ کاری انجام نمی دهند. حالا که روتر مسیری به HeroDetailComponent دارد، لینک های dashboard hero را برای حرکت از طریق parameterized dashboard route تنظیم کنید.

src/app/dashboard/dashboard.component.html (hero links)

```
<a *ngFor="let hero of heroes" class="col-1-4" routerLink="/detail/{{hero.id}}">
```

برای insert کردن hero.id حلقه ی تکرار فعلی داخل هرکدام از routerLink ها از Angular interpolation binding در تکرار کننده ی *ngFor استفاده کنید.

آموزش لینک های HeroesComponent hero

آیتم های Hero در HeroesComponent المان های <li> هستند که click event ها در آن به متد onSelect() کامپوننت محدود هستند.

src/app/heroes/heroes.component.html (list with onSelect)

```
<ul class="heroes">
 <li *ngFor="let hero of heroes"
 [class.selected]="hero === selectedHero"
 (click)="onSelect(hero)">
 <span class="badge">{{hero.id}}</span> {{hero.name}}
 </li>
 </ul>
```

 را به گونه ای خلاصه کنید که فقط ngFor باقی بماند، badge و نام را در المان anchor (<a>) قرار دهید و یک routerLink attribute به anchor اضافه کنید که دقیقا مشابه آن در dashboard template است.

src/app/heroes/heroes.component.html (list with links)

```
<ul class="heroes">
 <li *ngFor="let hero of heroes">
 <a routerLink="/detail/{{hero.id}}">
 <span class="badge">{{hero.id}}</span> {{hero.name}}
 </a>
 </li>
 </ul>
```

برای اینکه ظاهر لیست را به حالت قبلی خود برگردانید لازم است که private stylesheet (heroes.component.css) را بر روی آن تنظیم کنید. استایل های بازبینی شده در بررسی نهایی کد در پایین این راهنما قرار دارند.

آموزش حذف کدهای مرده (dead code) (اختیاری)

در حالی که کلاس HeroesComponent هنوز کار می کند، متد onSelect() و ویژگی selectedHero دیگر استفاده نمی شود. بهتر است که مرتب شود، بعدها به این خاطر از خودتان تشکر خواهید کرد. در زیر کلاس بعد از برداشتن کدهای مرده آمده است:

src/app/heroes/heroes.component.ts (cleaned up)

```
export class HeroesComponent implements OnInit {

heroes: Hero[];

constructor(private heroService: HeroService) { }

ngOnInit() {

this.getHeroes();

}

getHeroes(): void {

this.heroService.getHeroes()

.subscribe(heroes => this.heroes = heroes);

}

}
```

آموزش HeroDetailComponent با قابلیت مسیریابی

پیش از این،HeroesComponent مادر ،ویژگی HeroDetailComponent.hero رامشخص می کرد و HeroDetailComponent ، Hero را نمایش می داد.

HeroesComponent دیگر این کار را انجام نمیدهد. اکنون روتر HeroDetailComponent را در پاسخ به یک URL مانند detail/11 /~ ایجاد می کند.

HeroDetailComponent نیاز به یک متد جدید برای به دست آوردن hero-to-display دارد.

- روتی را دریافت کنید که آن را ایجاد کرده است.
- id را از route استخراج کنید
- با استفاده از HeroService ، Hero را با آن id از سرور دریافت کنید

Import های زیر را اضافه کنید:

src/app/hero-detail/hero-detail.component.ts

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

import { Location } from '@angular/common';
```

```
import { HeroService } from '../hero.service';
```

خدمات ActivatedRoute، HeroService، و Location را به constructor، تزریق کنید و مقادیر آنها را در فیلدهای خصوصی ذخیره کنید:

```
constructor(

private route: ActivatedRoute,

 private heroService: HeroService,

private location: Location

) {}
```

ActivatedRoute اطلاعاتی در مورد route این نمونه از HeroDetailComponent را نگه می دارد. این کامپوننت در route's bag بر روی پارامترهای استخراج شده از URL متمرکز است. پارامتر "id" شناسه ی hero برای نمایش است.

HeroService ، hero data را از سرور از راه دور دریافت می کند و این کامپوننت، از آن برای دریافت hero-to-display استفاده می کند.

Location یک سرویس Angular برای تعامل با مرورگر است. بعدا از آن استفاده میکنید تا به view که در اینجا به آن رفته اید برگردید.

پارامتر مسیر id را استخراج کنید.

getHero() را در ngOnInit() lifecycle hook ، فراخوانی کنید و آن را به صورت زیر تعریف کنید:

```
ngOnInit(): void {

 this.getHero();

 }

 getHero(): void {

 const id =

 +this.route.snapshot.paramMap.get('id');

this.heroService.getHero(id)

 .subscribe(hero => this.hero =

 hero);

 }
```

route.snapshot یک تصویر استاتیک از اطلاعات مسیر در مدت کوتاهی پس از ایجاد کامپوننت است.

paramMap یک دیکشنری از مقادیر پارامتری مسیر استخراج شده از URL است. کلید "id" شناسه ی Hero را برای دریافت کردن بازگشت می دهد.

پارامترهای مسیر همیشه به صورت رشته هستند. اپراتور جاوا اسکریپت (+) رشته را به یک عدد تبدیل می کند، این همان چیزی است که یک hero id باید باشد.

مرورگر refresh می شود و برنامه بعد از مواجه شدن با یک خطای کامپایلر بسته می شود. HeroService متد getHero() ندارد. الان آن را اضافه کنید.

آموزش اضافه کردن HeroService.getHero()

HeroService را باز کنید و متد getHero() را اضافه کنید.

src/app/hero.service.ts (getHero)

```
getHero(id: number):
 Observable<Hero> {
 // TODO: send the message _after_ fetching the hero
this.messageService.add(`HeroService: fetched hero id=${id}`);
return of(HEROES.find(hero => hero.id === id));
}
```

به این علامت توجه کنید (‘)backticks کار این علامت تعریف کردن یک JavaScript template literal برای جاگذاری id است.

مانند getHeroes()، getHero() هم یک امضای ناهمزمان دارد که این متد با استفاده از تابع RxJS of() یک هیروی ساختگی (mock hero) را به عنوان یک Observable برگشت می دهد.

شما قادر خواهید بود تا getHero() را به عنوان یک درخواست واقعی Http بدون نیاز به تغییر HeroDetailComponent که آن را فراخوانی می کند، دوباره اجرا کنید.

نرم افزار را امتحان کنید

مرورگر refresh می شود و برنامه دوباره کار می کند. شما می توانید با کلیک بر روی یک hero در dashboard و یا در heroes list به hero's detail view دسترسی پیدا کنید.

اگر آدرس localhost را 4200/detail/11 در نوار آدرس مرورگر وارد کنید، router به detail view مربوط به hero با id شماره 11 که در اینجا نام آن Mr. Nice است، می رود.

راه برگشت را پیدا کنید

با کلیک کردن بر روی دکمه برگشت مرورگر، می توانید به hero list یا dashboard view بروید، بسته به اینکه کدام یک شما را به detail view فرستاده است.

بهتر است دکمه ای در HeroDetail view وجود داشته باشد تا این کار را انجام دهد.

دکمه بازگشت را به پایین قالب کامپوننت اضافه کنید و آن را در متد goBack () کاپوننت بچسبانید.

src/app/hero-detail/hero-detail.component.html (back button)

```
<button (click)="goBack()">go

 back</button>
```

با استفاده از سرویس location که قبلا وارد کرده اید، متد goBack () را به کلاس component اضافه کنید که کار این متد برگشتن به یک قدم قبل در history مرورگر است.

src/app/hero-detail/hero-detail.component.ts (goBack)

```
goBack(): void {

this.location.back();

}
```

مرورگر را Refresh کرده و کلیک کنید. کاربران می توانند در برنامه حرکت کنند، از dashboard به hero details و بازگشت، از heroes list تا mini detail به hero details و دوباره بازگشت به heroes.

شما تمامی پیش نیازهای حرکتی که باعث بالا و پایین شدن این صفحه می شود را آموخته اید.

بازبینی نهایی کد

در ادامه می توانید فایلهای کد بحث شده در این صفحه را مشاهده کنید. برنامه ی شما باید مانند کد زیر باشد.

آموزش استفاده از AppRoutingModule، AppModule و HeroService

src/app/app-routing.module.ts

```
1. import { NgModule } from '@angular/core';
2. import { RouterModule, Routes } from '@angular/router';
3.
4. import { DashboardComponent } from './dashboard/dashboard.component';
```

```
import { HeroesComponent } from './heroes/heroes.component';
import { HeroDetailComponent } from './hero-detail/hero-detail.component';

const routes: Routes = [
{ path: '', redirectTo: '/dashboard', pathMatch: 'full' },
{ path: 'dashboard', component: DashboardComponent },
{ path: 'detail/:id', component: HeroDetailComponent },
{ path: 'heroes', component: HeroesComponent }
];

@NgModule({
imports: [ RouterModule.forRoot(routes) ],
exports: [ RouterModule ]
})
export class AppRoutingModule {}
```

src/app/app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { DashboardComponent } from './dashboard/dashboard.component';
import { HeroDetailComponent } from './hero-detail/hero-detail.component';
import { HeroesComponent } from './heroes/heroes.component';
import { MessagesComponent } from './messages/messages.component';

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

@NgModule({
imports: [
BrowserModule,
FormsModule,
AppRoutingModule
```

```
],
declarations: [
AppComponent,
DashboardComponent,
HeroesComponent,
HeroDetailComponent,
MessagesComponent
],
bootstrap: [ AppComponent ]
})
export class AppModule { }
```

src/app/hero.service.ts

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

import { Observable, of } from 'rxjs';

import { Hero } from './hero';
import { HEROES } from './mock-heroes';
import { MessageService } from './message.service';

@Injectable({ providedIn: 'root' })
export class HeroService {

constructor(private messageService: MessageService) { }

getHeroes(): Observable<Hero[]> {
// TODO: send the message _after_ fetching the heroes
this.messageService.add('HeroService: fetched heroes');
return of(HEROES);
}

getHero(id: number): Observable<Hero> {
// TODO: send the message _after_ fetching the hero
```

```
22. this.messageService.add(`HeroService: fetched hero id=${id}`);
23. return of(HEROES.find(hero => hero.id === id));
24. }
25. }
```

AppComponent

src/app/app.component.html

```
<h1>{{title}}</h1>
 <nav>
<a
 routerLink="/dashboard">Dashboard</a>
 <a
 routerLink="/heroes">Heroes</a>
 </nav>
 <router-outlet></router-outlet>
 <app-messages></app-messages>
```

src/app/app.component.css

```
1. /* AppComponent's private CSS styles */
2. h1 {
3. font-size: 1.2em;
4. color: #999;
5. margin-bottom: 0;
6. }
7. h2 {
8. font-size: 2em;
```

```
  margin-top: 0;
  padding-top: 0;
}
nav a {
  padding: 5px 10px;
  text-decoration: none;
  margin-top: 10px;
  display: inline-block;
  background-color: #eee;
  border-radius: 4px;
}
nav a:visited, a:link {
  color: #607d8b;
}
nav a:hover {
  color: #039be5;
  background-color: #cfd8dc;
}
nav a.active {
  color: #039be5;
}
```

DashboardComponent

src/app/dashboard/dashboard.component.html

```
<h3>Top Heroes</h3>
<div class="grid grid-pad">
<a
*ngFor="let hero of heroes"
class="col-1-4"
routerLink="/detail/{{hero.id}}">
<div class="module hero">
<h4>{{hero.name}}</h4>
```

```
</div>
</a>
</div>
```

src/app/dashboard/dashboard.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Hero } from '../hero';
import { HeroService } from '../hero.service';

@Component({
selector: 'app-dashboard',
templateUrl: './dashboard.component.html',
styleUrls: [ './dashboard.component.css' ]
})
export class DashboardComponent implements OnInit {
heroes: Hero[] = [];

constructor(private heroService: HeroService) { }

ngOnInit() {
this.getHeroes();
}

getHeroes(): void {
this.heroService.getHeroes()
.subscribe(heroes => this.heroes = heroes.slice(1, 5));
}
}
```

src/app/dashboard/dashboard.component.css

```
/* DashboardComponent's private CSS styles */
[class*='col-'] {
```

```
3.  float: left;
4.  padding-right: 20px;
5.  padding-bottom: 20px;
6.  }
7.  [class*='col-']:last-of-type {
8.  padding-right: 0;
9.  }
10. a {
11. text-decoration: none;
12. }
13. *, *:after, *:before {
14. -webkit-box-sizing: border-box;
15. -moz-box-sizing: border-box;
16. box-sizing: border-box;
17. }
18. h3 {
19. text-align: center; margin-bottom: 0;
20. }
21. h4 {
22. position: relative;
23. }
24. .grid {
25. margin: 0;
26. }
27. .col-1-4 {
28. width: 25%;
29. }
30. .module {
31. padding: 20px;
32. text-align: center;
33. color: #eee;
34. max-height: 120px;
35. min-width: 120px;
36. background-color: #607d8b;
37. border-radius: 2px;
38. }
```

```
39. .module:hover {
40. background-color: #eee;
41. cursor: pointer;
42. color: #607d8b;
43. }
44. .grid-pad {
45. padding: 10px 0;
46. }
47. .grid-pad > [class*='col-']:last-of-type {
48. padding-right: 20px;
49. }
50. @media (max-width: 600px) {
51. .module {
52. font-size: 10px;
53. max-height: 75px; }
54. }
55. @media (max-width: 1024px) {
56. .grid {
57. margin: 0;
58. }
59. .module {
60. min-width: 60px;
61. }
62. }
```

HeroesComponent

src/app/heroes/heroes.component.html

```
<h2>My Heroes</h2>
 <ul class="heroes">
<li *ngFor="let hero of heroes">
 <a
routerLink="/detail/{{hero.id}}">
```

```
<span class="badge">{{hero.id}}
</span> {{hero.name}}
</a>
</li>
</ul>
```

src/app/heroes/heroes.component.ts

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

import { Hero } from '../hero';
import { HeroService } from '../hero.service';

@Component({
selector: 'app-heroes',
templateUrl: './heroes.component.html',
styleUrls: ['./heroes.component.css']
})
export class HeroesComponent implements OnInit {
heroes: Hero[];

constructor(private heroService: HeroService) { }

ngOnInit() {
this.getHeroes();
}

getHeroes(): void {
this.heroService.getHeroes()
.subscribe(heroes => this.heroes = heroes);
}
}
```

src/app/heroes/heroes.component.css

```
/* HeroesComponent's private CSS styles */
.heroes {
margin: 0 0 2em 0;
list-style-type: none;
padding: 0;
width: 15em;
}
.heroes li {
position: relative;
cursor: pointer;
background-color: #EEE;
margin: .5em;
padding: .3em 0;
height: 1.6em;
border-radius: 4px;
}

.heroes li:hover {
color: #607D8B;
background-color: #DDD;
left: .1em;
}

.heroes a {
color: #888;
text-decoration: none;
position: relative;
display: block;
width: 250px;
}

.heroes a:hover {
color:#607D8B;
}
```

```
35.
36. .heroes .badge {
37. display: inline-block;
38. font-size: small;
39. color: white;
40. padding: 0.8em 0.7em 0 0.7em;
41. background-color: #607D8B;
42. line-height: 1em;
43. position: relative;
44. left: -1px;
45. top: -4px;
46. height: 1.8em;
47. min-width: 16px;
48. text-align: right;
49. margin-right: .8em;
50. border-radius: 4px 0 0 4px;
51. }
```

HeroDetailComponent

src/app/hero-detail/hero-detail.component.html

```
<div *ngIf="hero">
<h2>{{hero.name | uppercase}} Details</h2>
<div><span>id: </span>{{hero.id}}
</div>
 <div>
 <label>name:
 <input [(ngModel)]="hero.name"
 placeholder="name"/>
```

```
</label>
 </div>
<button (click)="goBack()"
>go back</button>
 </div>
```

src/app/hero-detail/hero-detail.component.ts

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

import { Hero } from '../hero';
import { HeroService } from '../hero.service';

@Component({
selector: 'app-hero-detail',
templateUrl: './hero-detail.component.html',
styleUrls: [ './hero-detail.component.css' ]
})
export class HeroDetailComponent implements OnInit {
@Input() hero: Hero;

constructor(
private route: ActivatedRoute,
private heroService: HeroService,
private location: Location
) {}

ngOnInit(): void {
this.getHero();
}

getHero(): void {
const id = +this.route.snapshot.paramMap.get('id');
this.heroService.getHero(id)
```

```
.subscribe(hero => this.hero = hero);
}

goBack(): void {
this.location.back();
}
}
```

src/app/hero-detail/hero-detail.component.css

```
/* HeroDetailComponent's private CSS styles */
label {
display: inline-block;
width: 3em;
margin: .5em 0;
color: #607D8B;
font-weight: bold;
}
input {
height: 2em;
font-size: 1em;
padding-left: .4em;
}
button {
margin-top: 20px;
font-family: Arial;
background-color: #eee;
border: none;
padding: 5px 10px;
border-radius: 4px;
cursor: pointer; cursor: hand;
}
button:hover {
background-color: #cfd8dc;
}
button:disabled {
background-color: #eee;
```

```
28. color: #ccc;
29. cursor: auto;
30. }
```

خلاصه

- روتر Angular را برای حرکت در میان کامپوننت های مختلف اضافه کرده اید.
- با لینک های <a> و <router-outlet> ، AppComponent را به یک navigation shell تبدیل کردید.
- روتر را در AppRoutingModule پیکربندی کرده اید
- مسیرهای ساده، یک مسیر هدایت شده و مسیر پارامتری را تعریف کردید.
- از دستورالعمل routerLink در المان anchor استفاده کردید.
- یک master/detail view که تا حد زیادی به هم وابسته اند را در یک detail view دارای مسیر (Routed) ریفاکتور کردید.
- با استفاده از router link parameters از یک user-selected hero به detail view ، رفتید.
- HeroService را در بین کامپوننت های متعدد به اشتراک گذاشتید.