آموزش Angulr 6 ـ آموزش کامپوننت های Master/Detail در Angulr 6

در حال حاضر، HeroesComponent هم لیست و هم جزئیات Hero های انتخاب شده را نمایش می دهد. هر چه نرم افزار شما بیشتر توسعه یابد، قرار دادن تمامی این ویژگی ها در یک کامپوننت کار سختی است. بهتر است که کامپوننت های بزرگ را به زیر کامپوننت های کوچک تر تبدیل کنید و هر کدام از آن ها را بر روی یک وظیفه یا گردش کار مشخص متمرکز کنید.

شما در این بخش اولین قدم را در راستای این کار خواهید برداشت و جزئیات Hero ها را به کامپوننت مجزا و با قابلیت استفاده مجدد HeroDetailComponent انتقال خواهید داد.

HeroesComponent تنها لیست Hero ها و HeroDetailComponent جزئیات hero های انتخابی را ارائه می کنند.

آموزش ایجاد HeroDetailComponent

با استفاده از Angular CLI کامپوننت جدیدی به نام hero-detail ایجاد کنید.

ng generate component hero-detail

این فرمان فایل های HeroDetailComponent را به یکدیگر متصل ساخته و در AppModule کامپوننت را اعلان می کند.

آموزش نوشتن قالب (template)

Hero detail را از پایین قالب HeroesComponent کات کنید و آن را در قالب HeroDetailComponent داخل boilerplate تولید شده paste کنید.

HTML پیست شده به یک selectedHero اشاره دارد. HeroDetailComponent جدید می تواند نه تنها بیانگر hero انتخابی بلکه می تواند بیانگر هر hero ای باشد. در نتیجه selectedHero را در کل قالب جایگزین hero بکنید.

وقتی کارتان تمام شد، قالب HeroDetailComponent باید مانند کد زیر باشد :

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>

</div>
```

آموزش افزودن @Input() hero property

قالب HeroDetailComponent به hero property کامپوننت متصل می شود که همنوع Hero است.

فایل کلاس HeroDetailComponent را باز کنید و سمبل hero را import کنید.

src/app/hero-detail/hero-detail.component.ts (import Hero)

```
import { Hero } from '../hero';
```

hero property باید یک input property باشد و با عبارت @input() نمایش داده شود، زیرا HeroesComponent خارجی مانند زیر به آن چسبانده خواهد شد:

```
<app-hero-detail [hero]="selectedHero"></app-hero-detail>
```

عبارت import کردن @angular/core را اصلاح کنید و سمبل input (symbol) را به آن اضافه کنید.

src/app/hero-detail/hero-detail.component.ts (import Input)

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

پس از @input() decorator ، hero property را اضافه کنید .

```
@Input() hero: Hero;
```

تنها تغییری که نیاز است در کلاس HeroDetailComponent ایجاد کنید همین است. نه دیگر خبری از property دیگری هست و نه منطق ارائه (presentation logic) دیگری. تنها کاری که این کامپوننت انجام می دهد این است که از طریق hero property خود، hero object را دریافت کرده و آن را نمایش می دهد.

آموزش نمایش دادن HeroDetailComponent

herosComponent هنوز در حالت master/detail view قرار دارد. قبل از اینکه این بخش از قالب را کات کنید، کاربرد آن به خودی خود نمایش دادن hero details است. حالا این کار به HeroDetailComponent محول می شود. این دو کامپوننت رابطه parent/child دارند به این صورت که هر زمان که کاربر بر روی hero ای از لیست کلیک می کند، herosComponent که نقش parent را دارد، hero ی جدیدی به HeroDetailComponent (child) ارسال می کند تا کامپوننت child آن را نمایش دهد.

نیازی به تغییر کلاس HeroesComponent نیست و تنها تغییر قالب آن کفایت می کند.

آموزش به روز کردن قالب Heroes Component

'app-hero-detail' نقش انتخابگر (selector) HeroDetailComponent را ایفا می کند. پایین قالب heroes component در جایی که قبلا hero detail view قرار داشت، <app-hero-detail> را اضافه کنید. مانند زیر، HeroesComponent.selectedHero را به hero property المان ها بچسبانید:

heroes.component.html (HeroDetail binding)

```
<app-hero-detail [hero]="selectedHero"></app-hero-detail>
```

"[hero]="selectedHero ، یک angular property binding محسوب می شود. این یکی از راه های data binding از selectedHero property مربوط به HeroesComponent در Hero property مربوط به المان مقصد است که به hero property مربوط به HeroDetailComponent متصل می شود. حالا وقتی که کاربر بر روی hero ای کلیک می کند، selectedhero تغییر می کند. زمانی که selectedHero تغییر می کند، property binding ، hero را آپدیت کرده و HeroDetailComponent ، hero جدید را نمایش می دهد.

قالب بازبینی شده herosComponent مانند کد زیر می شود:

heroes.component.html

```
<h2>My Heroes</h2>


<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>

    <app-hero-detail [hero]="selectedHero"></app-hero-detail>
```

مرورگر refresh شده و نرم افزار مانند قبل مجدد شروع به کار می کند.

چه چیزی تغییر کرده است؟

مانند گذشته هر زمان که کاربر بر روی اسم hero کلیک می کند، جزئیات آن hero در زیر لیست hero ها نمایش داده می شود. حالا به جای HeroesComponent ، HeroDetailComponent وظیفه ارائه این جزییات را بر عهده دارد. ریفاکتور کردن HeroesComponent اصلی به دو کامپوننت مجزا بازدهی نرم افزار را چه الان چه در آینده افزایش می دهد:

1. با این کار با کاهش مسئولیت های HeroesComponent ، آن را ساده سازی کرده اید.
2. بدون اینکه نیازی به تغییر HeroesComponent مادر (parent) داشته باشید، HeroDetailComponent را به یک ویرایشگر hero حرفه ای و غنی تبدیل کرده اید.
3. بدون اعمال کردن کوچک ترین تغییری در hero detail view ، می توانید HeroesComponent را ایجاد کنید.
4. در قالب کامپوننت های آینده هم می توانید از HeroDetailComponent مجددا استفاده کنید.

بازبینی نهایی کد

کدهایی که در این بخش به آن پرداختیم در پایین ارائه شده اند.

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

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

@Component({
```

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

constructor() { }

ngOnInit() {
}

}
```

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>
</div>
```

src/app/heroes/heroes.component.html

```
<h2>My Heroes</h2>
 <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>

<app-hero-detail [hero]="selectedHero"></app-hero-detail>
```

src/app/app.module.ts

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

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

@NgModule({
declarations: [
AppComponent,
HeroesComponent,
HeroDetailComponent
],
imports: [
BrowserModule,
FormsModule
],
providers: [],
bootstrap: [AppComponent]
})
export class AppModule { }
```

خلاصه

- شما توانستید یک HeroDetailComponent مجزا و با قابلیت استفاده مجدد ایجاد کنید.
- برای آنکه کنترل HeroDetailComponent کودک را در اختیار HeroesComponent مادر قرار دهید از property binding استفاده کردید.
- برای ایجاد hero property موجود برای مقید سازی توسط HeroesComponent خارجی از @Input decorator استفاده کردید.