آموزش Hero Editor در Angular

این برنامه اکنون یک عنوان خوب و اساسی دارد. در این جا شما یک کامپوننت (component) برای نمایش اطلاعات hero ایجاد می کنید و سپس آن کامپوننت (component) را در پوسته برنامه قرار دهید.

آموزش ساخت کامپوننت heroes در Angular

با استفاده از انگولار (Angular CLI) یک کامپوننت جدید به نامه heroes ایجاد می کنیم.

ng generate component heroes

CLI یک پوشه جدید ایجاد می کند، src / app / heroes / و سه فایل HeroesComponent را تولید می کند.

فایل کلاس HeroesComponent به شرح زیر است:

```
import { Component, OnInit } from '@angular/core';
@Component({
selector: 'app-heroes',
templateUrl: './heroes.component.html',
styleUrls: ['./heroes.component.css']
})
export class HeroesComponent implements OnInit {
constructor() { }
ngOnInit() {
}
}
```

شما همیشه برای وارد کردن نماد Component از کتابخانه (Angular core) استفاده کنید و برای نوشتن کلاس Component از Component@ استفاده کنید.

Component@ یک تابع دکوراتور (decorator)است که متادیتای "Angular " را برای "component " مشخص می کند.

CLI سه ویژگی متادیتای (metadata properties)ایجاد کرد:

- Selector : انتخاب کامپوننت های عنصر CSS
- templateUrl : محل کامپوننت های فایل قالب
- styleUrls : محل کامپوننت های استایل CSS خصوصی

انتخاب عنصر CSS، app-heroes ، مطابق با نام عنصر HTML است که این component را در قالب یک component اصلی شناسایی می کند.

ngOnInit یک حلقه چرخه عمر طولانی است که پس از ایجاد یک کامپوننت ، ngOnInit انگولار(Angular) را فرا خوانی میکند.

همیشه کلاس کامپوننت را exportکنید تا بتوانید آن را در جای دیگر وارد import کنید ... مانند در AppModule.

```
hero = 'Windstorm';
```

آموزش نمایش hero در Angular

فایل قالب heroes.component.html را باز کنید.متن پیش فرض تولید شده توسط Angular CLI را حذف کرده و آن را با ویژگی(property) جدید hero اتصال(data binding) بدهید.

```
{{hero}}
```

آموزش نمایش HeroesComponent در Angular

برای به نمایش در آوردن HeroesComponent، شما باید آن را به قالب پوسته AppComponent اضافه کنید.

به یاد داشته باشید که app-heroes انتخاب کننده تگ برای HeroesComponent است.

بنابراین در زیر Title تگ <app-heroes> را به پرونده قالب AppComponent اضافه کنید.

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

<app-heroes></app-heroes>
```

فرض بر این است که دستور CLI ng serve هنوز در حال اجرا است، مرورگر باید رفرش شود و نمایش عنوان برنامه و نام hero را نشان دهد.

آموزش ساخت کلاس Hero در Angular

Heroواقعی بیش تر از یک نام است.

یک کلاس hero را در فایل خود در پوشه src / app ایجاد کنید. به آن ویژگی id و name بدهید.

```
export class Hero {
 id: number;
 name: string;
}
```

به کلاس HeroesComponent برگردید و کلاس Hero را وارد کنید.

در کامپوننت ها ویژگی hero به Hero تبدیل می شود. ابتدا آن را با یک آی دی 1 و نام Windstorm آغاز کنید.

فایل کلاس HeroesComponent باید به شکل زیر شده باشد :

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

@Component({
 selector: 'app-heroes',
 templateUrl: './heroes.component.html',
 styleUrls: ['./heroes.component.css']
})
export class HeroesComponent implements OnInit {
 hero: Hero = {
  id: 1,
  name: 'Windstorm'
 };

 constructor() { }

 ngOnInit() {
 }

}
```

صفحه به درستی نمایش داده نمی شود زیرا شما hero را از یک string به یک object تغییر دادید.

آموزش نمایش آبجکت hero در Angular

در فایل heroes.component.html برای نام hero را اعلام کرده و id و name را مانند شکل زیر درست کنید :

```
<h2>{{ hero.name }} Details</h2>
<div><span>id: </span>{{hero.id}}</div>
<div><span>name: </span>{{hero.name}}</div>
```

مرورگر را رفرش کرده تا اطلاعات hero را نمایش دهد.

آموزش فرمت با UppercasePipe در Angular

نام hero.name مانند زیر تغییر دهید :

```
<h2>{{ hero.name | uppercase }} Details</h2>
```

مرورگر را رفرش کنید و حالا نام hero با حروف بزرگ نمایش داده می شود.

کلمه uppercase در اتصال بینابینی ، درست بعد از کاراکتر pipe (|)، UppercasePipe ساخته شده را فعال می کند.

Pipes ها یک راه خوب برای فرمت رشته ها، مقدار ارز، تاریخ و دیگر داده های صفحه نمایش است. Angular ships با چند Pipe داخلی ساخته شده و شما می توانید Pipe خود را ایجاد کنید.

آموزش ویرایش hero در Angular

کاربران باید بتوانند نام hero را در کادر ورود <input> ویرایش کنند.

جعبه متن باید هر دو name ویژگی hero را نمایش دهد و آن ویژگی را به عنوان نوع کاربر به روز کند. این بدان معنی است که جریان داده ها از کلاس component به صفحه و از روی صفحه به کلاس باز می شود.

برای اینکه به طور خودکار جریان داده اتفاق بیفتد، یک اتصال دو طرفه را بین تگ فرم <input> و ویژگی hero.name تنظیم کنید.

آموزش اتصال دوطرفه در Angular

قسمت جزئیات در قالب HeroesComponent را بازنویسی کنید، به طوری که مانند شکل زیر بشود :

```
<div>
  <label>name:
    <input [(ngModel)]="hero.name" placeholder="name">
  </label>
</div>
```

[(ngModel)] یک سینتکس انگولار برای اتصال دوطرفه است.

در اینجا این خاصیت hero.name را به جعبه متن HTML متصل می کند. به طوری که داده ها می توانند در هر دو جهت جریان داشته باشند: از ویژگی hern.name به جعبه متن، و از جعبه متن به hero.name.

از بین رفتن FormsModule در Angular

توجه داشت باشید که زمانی [(ngModel)] را اضافه می کنیدبرنامه متوقف می شود.

برای دیدن خطا، ابزارهای توسعه مرورگر را باز کنید و در کنسول یک پیام مانند زیر مشاهده می کنید :

Template parse errors:
Can't bind to 'ngModel' since it isn't a known property of 'input'.

اگر چه ngModel یک دستورالعمل Angular معتبر است، اما به طور پیش فرض در دسترس نیست.

این سینتکس متعلق به FormsModule اختیاری است و شما باید از آن استفاده کنید.

AppModule

Angular نیاز دارد تا چگونگی تقسیم برنامه های خود با یکدیگر و سایر پرونده ها و کتابخانه ها مورد نیاز برنامه بداند.این اطلاعات مورد نیاز انگولار را metadata می نامند.

برخی از فراداده ها یا metadata ها در دکوراتور Component@ هستند که شما به کلاس های component خود اضافه کرده اید. دیگر متادیتای مهم در NgModule @است.

مهمترین طراح داخلی NgModule است که کلاس AppModule سطح بالا را اعلان می کند.

وقتی که شما پروژه را ایجاد می کنید Angular CLI یک کلاس AppModule را در src/app/app.module.ts ایجاد می کند این جایی است که شما FormsModule انتخاب می کنید.

آموزش وارد کردن FormsModule در Angular

AppModule (app.module.ts) را باز کرده و نماد FormsModule را از کتابخانه @angular/forms وارد کنید.

```
import { FormsModule } from '@angular/forms'; // <-- NgModel lives here
```

سپس FormsModule را به آرایه NgModule@ وارد (imports) کنید که حاوی لیستی از ماژول های خارجی است که برنامه نیاز دارد.

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

هنگامی که مرورگر رفرش می شود، برنامه باید دوباره کار کند. شما می توانید نام hero را ویرایش کنید و تغییراتی که بلافاصله در <h2> در بالای جعبه متن نشان داده می شود را مشاهده کنید.

تعریف HeroesComponent

هر component باید دقیقا در یک NgModule اعلام و تعریف شود.

شما HeroesComponent را اعلام نکرده اید. پس چرا برنامه کار کرد؟

این برنامه به این دلیل کار کرد که Angular CLI (HeroesComponent) را در AppModule وقتی component تولید شد اعلام کرد.

src/app/app.module.ts باز کرده و HeroesComponent را در نزدیکی بالای صفحه وارد شده است.

```
import { HeroesComponent } from './heroes/heroes.component';
```

HeroesComponent نیز در آرایه NgModule.declarations @ اعلام شده است.

```
declarations: [ AppComponent, HeroesComponent ],
```

توجه داشته باشید که AppModule هر دو کامپوننت های برنامه، AppComponent و HeroesComponent را اعلام می کند.

بررسی کد نهایی

src/app/heroes/heroes.component.ts

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

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

```
 hero: Hero = {
  id: 1,
  name: 'Windstorm'
 };

 constructor() { }

 ngOnInit() {
 }

}
```

src/app/heroes/heroes.component.html

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

src/app/app.module.ts

```
1. import { BrowserModule } from '@angular/platform-browser';
2. import { NgModule } from '@angular/core';
3. import { FormsModule } from '@angular/forms'; // <-- NgModel lives here
4.
5. import { AppComponent } from './app.component';
6. import { HeroesComponent } from './heroes/heroes.component';
7.
8. @NgModule({
9. declarations: [
10.AppComponent,
11.HeroesComponent
12.],
13.imports: [
14.BrowserModule,
15.FormsModule
16.],
17.providers: [],
18.bootstrap: [AppComponent]
19.})
20.export class AppModule { }
```

src/app/app.component.ts

```
import { Component } from '@angular/core';@Component({ selector: 'app-root', templateUrl: './app.component.html', styleUrls: ['./app.component.css']}) export class AppComponent { title = 'Tour of Heroes';}
```

src/app/app.component.html

```
<h1>{{title}}</h1><app-heroes></app-heroes>
```

src/app/hero.ts

```
export class Hero { id: number; name: string; }
```

خلاصه

شما برای ایجاد یک HeroesComponent دوم از CLI استفاده کردید.

شما HeroesComponent را با اضافه کردن آن به پوسته AppComponent نمایش دادید.

شما UppercasePipe را برای format نام استفاده کردید.

شما با استفاده از دستورالعمل ngModel به طور مستقیم از اتصال دوطرفه استفاده کردید.

شما در مورد AppModule یاد گرفتید.

شما FormsModule را در AppModule وارد کرده اید، بنابراین Angular می تواند دستورالعمل ngModel را شناسایی و اعمال کند.

شما اهمیت اعلام components در AppModule را آموختید و دیدم که CLI آن را برای شما اعلام کرده است.