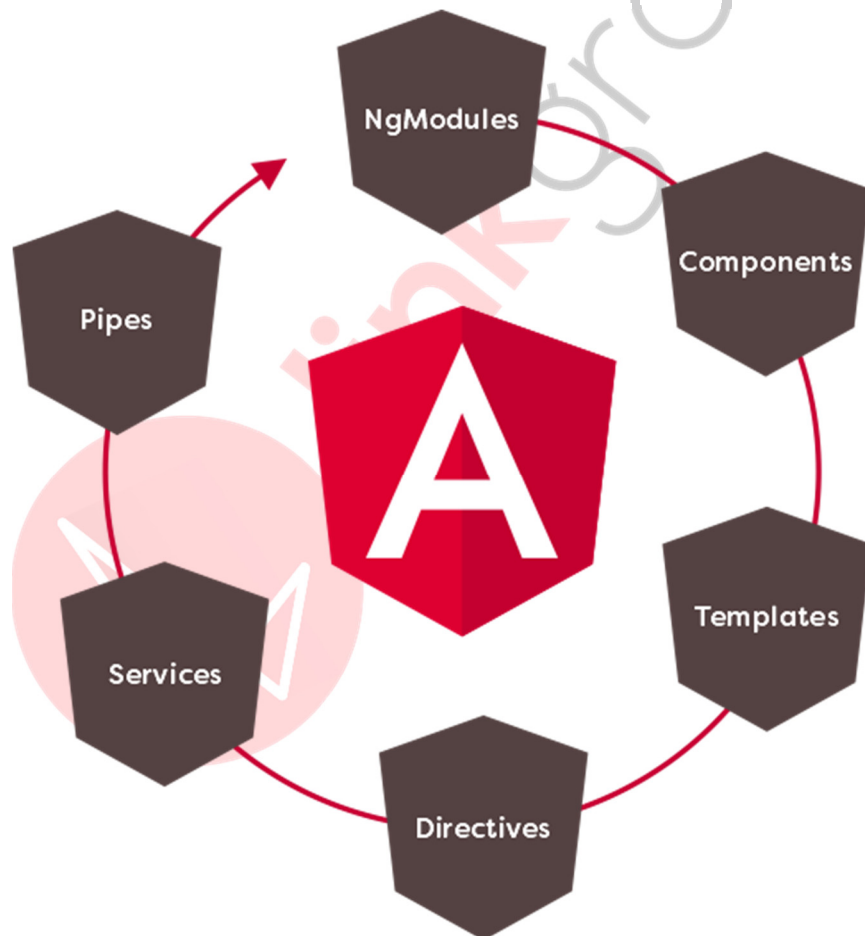


Angular moduli i komponente

Angular aplikacije sačinjene su iz različitih elemenata. Do sada smo već imali prilike da se načelno upoznamo sa komponentama i to prilikom kreiranja prve Angular aplikacije u uvodnoj lekciji o Angularu. Ipak, iako vrlo značajan, komponenta je samo jedan od pojmova koji se koristi tokom izgradnje Angular aplikacija. Stoga će sledeća lekcija biti posvećena upoznavanju svih elemenata koji čine arhitekturu jedne Angular aplikacije.

Arhitektura Angular aplikacije

Svaka Angular aplikacija zasniva se na strukturi određenih unapred utvrđenih pojmova koji unutar Angulara ekosistema imaju posebno značenje. Tako se za vreme korišćenja Angulara zapravo operiše različitim gradivnim blokovima, od kojih je sačinjen takav sistem (slika 14.1).



Slika 14.1. Angular arhitektura

Slika 14.1. ilustruje osnovne gradivne blokove Angular arhitekture. Za početak ćemo se načelno upoznati sa njima:

- **Angular moduli (NgModules)** – osnovni gradivni blokovi Angular aplikacija; koriste se za objedinjavanje koda u funkcionalne celine,
- **komponente (Components)** – elementi koji definišu delove korisničkog okruženja Angular aplikacija; unutar Angular komponenata objedinjuje se stilizacija, prezentacija i logika jednog segmenta onoga što korisnik vidi unutar web pregledača,
- **šabloni (Templates)** – definišu prezentaciju Angular komponenata korišćenjem čistog HTML koda; unutar HTML koda Angular šablona mogu se definisati i brojne tvorevine koje su validan HTML, ali za Angular imaju posebno značenje, pa tako na kraju omogućavaju jednostavnu interpolaciju, povezivanje podataka, generisanje koda, obradu događaja i slično...
- **direktive (Directives)** – direktive omogućavaju da se unutar šablona, pored podataka, priključi i neka proizvoljna logika koja može da utiče na način kreiranja prezentacije Angular komponente,
- **servisi (Services)** – elementi za rukovanje podacima koji se dele između više komponenata; servisi omogućavaju da se određene operacije, poput čitanja podataka sa servera, validacija korisničkog unosa ili beleženja unutrašnjih događaja aplikacije, izmeste izvan komponenata i da se zatim stave na raspolaganje svim onim komponentama u kojima se za određenom funkcionalnošću javi potreba,
- **Pipeovi (Pipes)** – elementi koji omogućavaju lako transformisanje podataka, neposredno pre njihovog prikazivanja unutar šablona neke Angular komponente; pri tome mehanizam Pipes omogućava da izvorni podaci ostanu nepromenjeni, a da samo korisnik dobije njihovu prilagođenu, odnosno transformisanu varijantu.

Pitanje

Na vrhu hijerarhije gradivnih blokova Angulara nalaze se:

- a) **Angular moduli**
- b) Angular komponente
- c) Angular servisi
- d) Angular šabloni

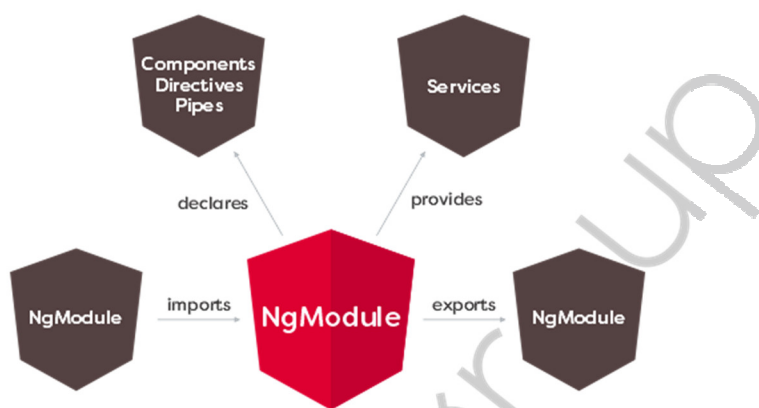
Objašnjenje:

Angular moduli (NgModules) su osnovni gradivni blokovi aplikacija Angular i nalaze se na vrhu takve hijerarhije.

Angular moduli

Upoznavanje upravo predstavljenih Angular pojmova započecemo pričom o osnovnim gradivnim blokovima takvog sistema – Angular modulima. Naime, Angular aplikacije sačinjene su iz modula, i to iz posebne vrste modula koji se slikovito nazivaju NgModules (naravno, prefiks Ng odnosi se na skraćenicu za **Angular**).

Odmah na početku je potrebno napraviti razliku, distinkciju između ES6 modula i NgModulea. Obe vrste modula se koriste tokom razvoja aplikacija Angular, ali poseduju različite namene. Sa upotrebom ES6 modula već smo se upoznali u prethodnim lekcijama i mogli ste da vidite da se oni koriste za izolovanje funkcionalnosti u zasebne fajlove i kontrolu deljenja takvih funkcionalnosti. NgModuli imaju nešto drugačiju ulogu. Oni služe kao kontejneri za elemente od kojih je sačinjena jedna Angular aplikacija ili jedan njen deo. Tako se unutar Angular modula objedinjuju komponente, direktive, servisi i ostali delovi koda koji čine jednu zaokruženu celinu (slika 14.2).



Slika 14.2. Pojam NgModulea

Sa slike 14.2. možete da vidite i to da jedan Angular modul može da uveze funkcionalnosti iz nekog drugog modula, ali i da svoje funkcionalnosti izloži nekom drugom modulu na korišćenje, baš kao što je to slučaj i kod ES6 modula.

Svaka Angular aplikacija poseduje makar jedan NgModule i on se naziva koreni Angular modul. U primeru naše prve Angular aplikacije iz jedne od prethodnih lekcija takav modul je za nas automatski kreiran prilikom generisanja projekta i smešten unutar fajla sa nazivom `app.module.ts`:

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

import { AppComponent } from './app.component';

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

Na osnovu sadržaja fajla `app.module.ts` može se zaključiti dosta toga o osobinama `NgModule`ea. Za početak možete da vidite da se Angular moduli definišu korišćenjem običnih klasa, koje su dekorisane jednim posebnim dekoratorom `@NgModule()`. Korišćenjem takvog dekoratora obavlja se konfigurisanje modula. Funkciji koja predstavlja dekorator prosleđuje se objekat sa metapodacima o modulu. Najvažnija svojstva takvog objekta za konfigurisanje modula su:

- `declarations` – komponente, direktive i pipeovi koji pripadaju modulu,
- `exports` – elementi modula koji će biti na raspolaganju drugim Angular modulima,
- `imports` – moduli čije funkcionalnosti koriste elementi modula koji se koristi,
- `providers` – servisi koje modul izlaže na korišćenje,
- `bootstrap` – korena komponenta modula koja se koristi za izgradnju i objedinjavanje svih ostalih komponentata; ukoliko aplikacija poseduje više Angular modula, samo koreni modul može definisati ovo svojstvo, kako bi se osiguralo postojanje jedinstvene ulazne tačke kompletne aplikacije.

Sada, kada znamo za šta služe različita svojstva za konfigurisanje jednog modula Angulara, nije teško da razumemo osobine Angular modula koji je kreiran prilikom izgradnje prvog projekta. Takav Angular modul:

- deklariše jednu Angular komponentu sa nazivom `AppComponent`,
- uvozi jedan `NgModule` sa nazivom `BrowserModule`,
- ne izvozi ništa, s obzirom na to da je reč o korenom modulu, pa iznad njega ne postoji nijedan modul koji bi takve funkcionalnosti uvezao,
- kao korenu komponentu čitave aplikacije definiše komponentu sa nazivom `AppComponent`.

Ugrađeni Angular moduli (Angular biblioteke)

Angular je softverski okvir koje je sačinjen iz mnoštva modula Angulara koji obezbeđuju skup ugrađenih funkcionalnosti. Tako se tokom kreiranja Angular aplikacija, sem `NgModule`ima koje samostalno kreiramo, rukuje i modulima koji su sastavni deo sistema. Takvi ugrađeni moduli se drugačije nazivaju Angular biblioteke.

Angular biblioteke su zapravo npm paketi, čiji naziv započinje prefiksom `@angular` i koje je moguće preuzeti korišćenjem npm menadžera paketa. Ipak, mi to ne moramo da obavljamo ručno, zato što unutar projekta koji se kreira korišćenjem Angular CLI-ja automatski postoje sve neophodne zavisnosti. Ipak, kako bi se koristile, neophodno ih je importovati.

Takva situacija postoji i unutar korenog modula naše Angular aplikacije. Koreni modul naše aplikacije uvozi Angular biblioteku pod nazivom `BrowserModule`. Procedura koja se koristi za importovanje ilustruje svojevrsni primer međusobnog dopunjavanja ES6 i Angular modula. Prvo se Angular biblioteka importuje kao ES6 modul:

```
import { BrowserModule } from '@angular/platform-browser';
```

Ipak, kako bi se funkcionalnostima `BrowserModule` moglo pristupiti i iz korenog Angular modula, naziv ovakvog modula se postavlja kao jedna od vrednosti svojstva `imports`, objekta za konfigurisanje korenog Angular modula:

```
imports: [  
  BrowserModule  
]
```

Angular komponente

Angular komponentama definiše se jedan deo korisničkog okruženja Angular aplikacije, odnosno jedan deo prikaza koji korisnik vidi unutar web pregledača. Unutar jedne Angular komponente objedinjuje se prezentacija i programska logika, odnosno svi oni elementi koji su potrebni za formiranje prikaza unutar web pregledača.

Angular komponente se definišu kao obične klase, koje se dekorišu korišćenjem specijalnog dekoratora `@Component()`, kome se prosleđuje objekat sa metapodacima za konfigurisanje komponente. Angular komponenta koju je tokom procesa generisanja projekta kreirao alat Angular CLI izgleda ovako:

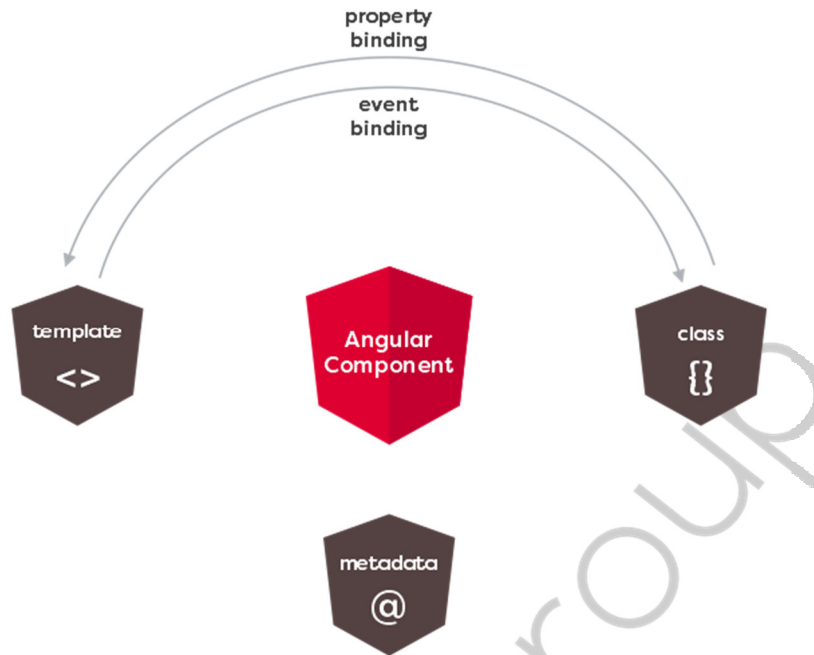
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'my-first-app';
}
```

Analizom prikazanog koda možemo uvideti ono što je rečeno – na klasu `AppComponent` je postavljen dekorator `@Component()` i tom prilikom je njemu prosleđen objekat za konfigurisanje komponente. Objekat za konfigurisanje poseduje sledeća svojstva:

- `selector` – selektor koji definiše element, unutar koga će da bude umetnuta kompletna prezentacija komponente; u primeru je definisano `app-root`, što znači da će kompletna prezentacija komponente da bude smeštena unutar elementa `<app-root></app-root>`,
- `templateUrl` – putanja na kojoj se nalazi fajl sa šablonom; u jednoj od prethodnih lekcija ste mogli da vidite da se šablon može definisati i direktno unutar objekta za dekorisanje jedne komponente; tada se, umesto svojstva `templateUrl`, koristi svojstvo `template`,
- `styleUrls` – putanja na kojoj se nalazi fajl sa kodom za stilizovanje komponente; i stilizaciju je moguće definisati direktno unutar fajla u kome se nalazi i klasa komponente i tada se koristi svojstvo `styles`.

Nakon ovog kratkog upoznavanja sa osnovnim osobinama komponenata Angulara, može se reći da njihova struktura izgleda kao na slici 14.3.



Slika 14.3. Struktura Angular komponente

Na slici 14.3. su prikazana i tri osnovna elementa koja učestvuju u kreiranju Angular komponente: šablon, programska logika unutar klase i metapodaci za konfigurisanje takve komponente. Ipak, na slici 14.3. možete videti još jednu veoma važnu osobinu komponente, koja se odnosi na povezivanje podataka. To praktično znači da Angular omogućava da se podaci direktno povežu se šablonom komponente i da se promene nad podacima automatski propagiraju do sloja prezentacije. Pored toga, Angular omogućava i da sloj prezentacije logici komponente šalje dojave o DOM događajima do kojih dolazi tokom korišćenja aplikacije. Takve dve osnovne vrste komunikacije između logike komponente i njenog šablona ilustrovane se strelicama na vrhu slike 14.3.

Šablon Angular komponente

Ono što korisnik vidi kada se unutar web pregledača prikaže jedna komponenta Angulara jeste proizvod koda koji se nalazi unutar šablona komponente. Šabloni Angular komponente grade se korišćenjem HTML koda, unutar koga se upotrebljavaju specifični elementi koji za Angular imaju posebno značenje. Takvi posebni elementi Angular šablona omogućavaju da se postigne:

- interpolacija,
- definisanje izraza,
- jednosmerno i dvosmerno povezivanje,
- pretplata na događaje,
- izmena DOM strukture korišćenjem direktiva,
- transformisanje vrednosti korišćenjem Pipeova.

Pre nego što se upustimo u detaljnu analizu svih pobrojanih pojmova koji su specifični za izgradnju Angular šablona, pogledaćemo kako izgleda šablon koji je za nas automatski generisan prilikom izgradnje projekta korišćenjem alata Angular CLI.

U primeru prve Angular aplikacije koja je generisana korišćenjem Angular CLI-ja šablon je smešten unutar zasebnog fajla sa nazivom `app.component.html`. Ukoliko pogledate sadržaj ovoga fajla, moći ćete da vidite da je reč o HTML kodu, koji poseduje i neke elemente sa kojima se do sada nismo susretali. Reč je o specijalnoj Angular sintaksi za izgradnju šablona, kojima se utiče na prikaz podataka i generisanje prezentacije. Na primer, unutar fajla koji predstavlja šablon možete pronaći jednu ovakvu liniju:

```
<span>{{ title }} app is running!</span>
```

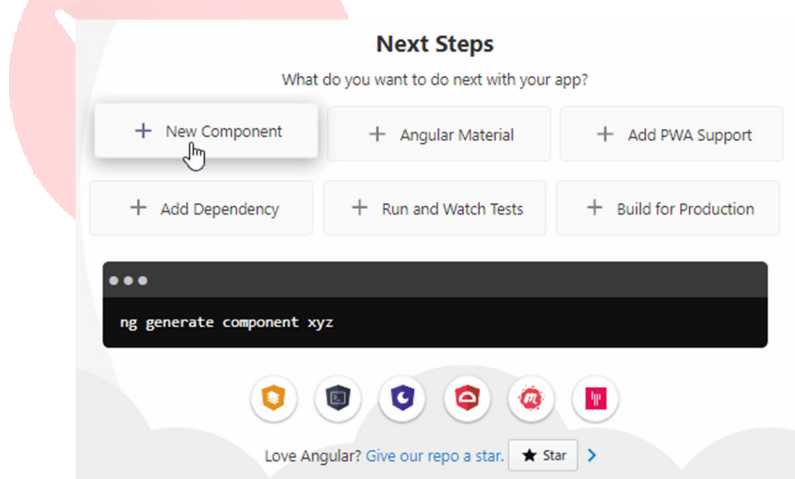
Reč je o liniji HTML koda kojom se ispisuje pozdravna poruka unutar stranice koja se dobija unutar web pregledača. Ukoliko pažljivo pogledate sadržaj ovoga `span` elementa, moći ćete da primetite i upotrebu specifične sintakse koja podrazumeva dva para vitičastih zagrada unutar kojih je definisan tekst `title`. Tekst `title` je zapravo naziv promenljive koja se nalazi unutar klase koja predstavlja komponentu. Dva para vitičastih zagrada predstavljaju specifičnu Angular sintaksu za postizanje interpolacije.

Ovakvih specifičnih Angular tvorevina ima još unutar šablona komponente. Na primer, unutar sekcije *Next Steps* možete pronaći nekoliko `div` elemenata, čiji kod za markiranje izgleda ovako:

```
<div class="card card-small" (click)="selection.value = 'component'">  
  <span>New Component</span>  
</div>
```

Na prikazanom `div` elementu možete videti način na koji se u Angularu obavlja pretplata na događaje – unutar običnih zagrada postavljen je naziv događaja (`click`) i na taj način je registrovana logika koja će se aktivirati kada korisnik klikne na ovakav `div` element.

Ovakav `div` element koristi se za izgradnju dela stranice koja se dobija generisanjem projekta korišćenjem Angular CLI-ja, a koji možete videti na sledećoj animaciji.



Animacija 14.1. Jedan deo stranice koji je izgrađen upravo prikazanim elementima

Klikom na neku od stavki sekcije *Next Steps* automatski se menja sadržaj unutar terminala koji se nalazi ispod. Ovakvo ponašanje kreirano je u potpunosti korišćenjem specifičnih elemenata Angular šablona. Prvi takav element ste videli u prethodnim redovima, a odnosi se na pretplatu na `click` događaj korišćenjem HTML atributa u specifičnom obliku – (`click`). Kao vrednost atributa (`click`) postavljen je kod kojim se definiše `value` svojstvo jedne promenljive sa nazivom `selection`:

```
(click)="selection.value = 'component'"
```

Ukoliko pažljivije proučite šablon naše komponente, moći ćete da vidite da unutar njega postoji i jedan ovakav element:

```
<input type="hidden" #selection>
```

Korišćenjem karaktera `#` kreirana je jedna šablonska referentna promenljiva (engl. *template reference variable*). Reč je o specifičnoj funkcionalnosti Angular šablona, koja omogućava da se direktno unutar šablona dobije referenca na neki DOM element.

Sada je jasnije šta se nešto ranije postizalo postavljanjem vrednosti `selection.value` – postavljala se vrednost `input` elementa na kome je definisana referentna promenljiva `#selection`.

Na kraju, vrednost koja se klikom na neku od stavki upisuje unutar skrivenog `input` elementa koristi se od sledeće šablonske logike, za generisanje teksta unutar terminala:

```
<!-- Terminal -->
<div class="terminal" [ngSwitch]="selection.value">
  <pre *ngSwitchDefault>ng generate component xyz</pre>
  <pre *ngSwitchCase="'material'">ng add @angular/material</pre>
  <pre *ngSwitchCase="'pwa'">ng add @angular/pwa</pre>
  <pre *ngSwitchCase="'dependency'">ng add _____</pre>
  <pre *ngSwitchCase="'test'">ng test</pre>
  <pre *ngSwitchCase="'build'">ng build --prod</pre>
</div>
```

Unutar ovakvog koda Angular šablona upotrebljen je još jedan specifičan Angular pojam – **direktiva**. Direktiva koja je upotrebljena u prikazanom primeru ima naziv `NgSwitch`, a kao što možete da naslutite, koristi se za dinamičko renderovanje koda u zavisnosti od nekog uslova. Definisani uslov jeste vrednost promenljive `selection.value` i, u zavisnosti od vrednosti takve promenljive, izvršava se jedna od nekoliko definisanih `NgSwitchCase` direktiva. Na ovaj način se, u zavisnosti od vrednosti promenljive `selection.value`, unutar konzole na web stranici prikazuje odgovarajući kod, baš kao što je to prikazano animacijom 14.1.

Generisanje nove komponente i priprema projekta za narednu lekciju

U prethodnim redovima ste mogli da vidite samo malu demonstraciju mogućnosti Angular šablona. Ovde ćemo se zaustaviti, a priča o šablonima i različitim elementima koji se unutar njih mogu naći biće nastavljena u narednoj lekciji.

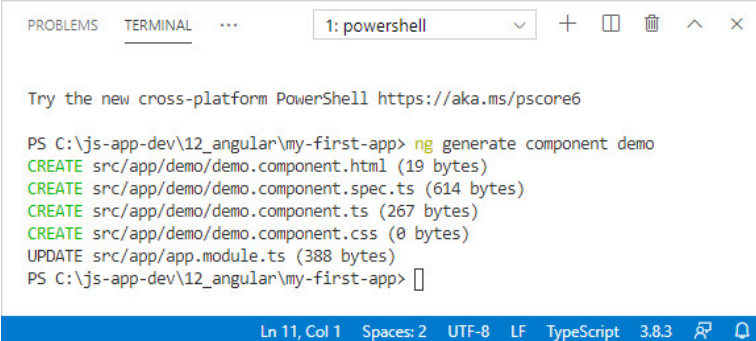
Kako bismo pripremili radno okruženje za narednu lekciju, u narednim redovima ćemo prvi put objasniti samostalno kreiranje jedne komponente.

Nova komponenta sa korišćenjem Angular CLI-ja može se kreirati upućivanjem sledeće komande:

```
ng generate component demo
```

Prilikom upućivanja ovakve komande potrebno je da budete pozicionirani unutar korenog foldera projekta ili radnog prostora ukoliko je reč o radnom prostoru sa jednim projektom (kao što je to kod nas slučaj).

Uspešno generisanje nove komponente rezultuje prikazom kao na slici 14.4.



```
PS C:\js-app-dev\12_angular\my-first-app> ng generate component demo
CREATE src/app/demo/demo.component.html (19 bytes)
CREATE src/app/demo/demo.component.spec.ts (614 bytes)
CREATE src/app/demo/demo.component.ts (267 bytes)
CREATE src/app/demo/demo.component.css (0 bytes)
UPDATE src/app/app.module.ts (388 bytes)
PS C:\js-app-dev\12_angular\my-first-app>
```

Slika 14.4. Generisanje nove komponente

Generisanje nove komponente korišćenjem Angular CLI-ja podrazumeva kreiranje četiri nova fajla unutar foldera sa nazivom komponente, unutar projektnog **src/app** foldera. Pored kreiranja fajlova komponente, Angular CLI za nas obavlja još nešto – automatski deklarise kreiranu komponentu unutar glavnog modula aplikacije:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { DemoComponent } from './demo/demo.component';
@NgModule({
  declarations: [
    AppComponent,
    DemoComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Unutar odeljka `declaration` možete videti da je, pored već prisutne komponente `AppComponent`, sada dodata i naša komponenta `DemoComponent`.

Ono što u nastavku želimo da postignemo jeste da u potpunosti uklonimo kod koji je za nas automatski generisao CLI prilikom kreiranja projekta i da se prezentacija naše Angular aplikacije u potpunosti sastoji iz one koja je definisana upravo kreiranom komponentom. Za obavljanje takvog posla dovoljno je da iz šablona glavne komponente uklonimo kompletan sadržaj i umesto njega postavimo sledeće:

```
<app-demo></app-demo>
```

Na ovaj način smo korišćenjem Angulara prvi put postigli kompoziciju komponenta, odnosno smestili smo jednu komponentu unutar neke druge komponente. Ali kako smo znali da je potrebno da upotrebimo baš ovakav element da bi se naša `Demo` komponenta prikazala unutar glavne komponente aplikacije?

Naziv selektora na osnovu koga se utvrđuje element unutar koga će biti smeštena prezentacija neke komponente definiše se svojstvom `selector`, dok dekorator konfiguriše komponentu:

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

Ovde možete videti da je vrednost svojstva `selector` upravo `app-demo`, što je tip elementa koji smo mi iskoristili za prikaz naše komponente unutar glavne komponente aplikacije.

Rezime

- Svaka Angular aplikacija zasniva se na strukturi određenih, unapred utvrđenih pojmova, koji unutar ekosistema Angulara imaju posebno značenje.
- Angular aplikacije sačinjene su iz modula, i to iz posebne vrste modula koji se slikovito nazivaju `NgModules`.
- `NgModuli` služe kao kontejneri za elemente od kojih je sačinjena jedna Angular aplikacija ili jedan njen deo.
- Angular moduli nisu isto što i ES6 moduli; obe vrste modula se koriste tokom razvoja Angular aplikacija i međusobno se dopunjuju.
- Svaka Angular aplikacija poseduje makar jedan `NgModule` i on se naziva koreni Angular modul.
- Angular moduli se definišu korišćenjem običnih klasa, koje su dekorisane jednim posebnim dekoratorom `@NgModule()`.

- Angular je softverski okvir koji je sačinjen iz mnoštva modula Angulara koji obezbeđuju skup ugrađenih funkcionalnosti; oni se nazivaju Angular biblioteke.
- Angular komponentama definiše se jedan deo korisničkog okruženja Angular aplikacije, odnosno jedan deo prikaza koji korisnik vidi unutar web pregledača.
- Angular komponente definišu se kao obične klase, koje se dekorišu korišćenjem specijalnog dekoratora `@Component()`, kome se prosleđuje objekat sa metapodacima za konfigurisanje komponente.
- Prezentacija jedne komponente se definiše korišćenjem šablona.
- Nova komponenta sa korišćenjem Angular CLI-ja može se kreirati upućivanjem komande `ng generate component name`, gde se *name* odnosi na naziv komponente.

