

Angular

Kuvar

Više od 80 praktičnih recepata koje bi svaki programer trebalo da zna

Muhammad Ahsan Ayaz

Angular

KUVAR

Više od 80 praktičnih recepata koje bi svaki programer trebalo da zna

Muhammad Ahsan Ayaz

Izdavač:



Obalskih radnika 4a, Beograd

Tel: 011/2520272

e-mail: kombib@gmail.com

internet: www.kombib.rs

Urednik: Mihailo J. Šolajić

Za izdavača, direktor:

Mihailo J. Šolajić

Autor: Muhammad Ahsan Ayaz

Prevod: Biljana Tešić

Lektura: Miloš Jevtović

Slog: Zvonko Aleksić

Znak Kompjuter biblioteke:

Miloš Milosavljević

Štampa: „Pekograf“, Zemun

Tiraž: 500

Godina izdanja: 2021.

Broj knjige: 546

Izdanje: Prvo

ISBN: 978-86-7310-569-7

Angular Cookbook

Muhammad Ahsan Ayaz

ISBN 978-1-83898-943-9

Copyright © July 2021 Packt Publishing

All right reserved. No part of this book may be reproduced or transmitted in any form or by means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Autorizovani prevod sa engleskog jezika edicije u izdanju „Packt Publishing“, Copyright © July 2021.

Sva prava zadržana. Nije dozvoljeno da nijedan deo ove knjige bude reprodukovan ili snimljen na bilo koji način ili bilo kojim sredstvom, elektronskim ili mehaničkim, uključujući fotokopiranje, snimanje ili drugi sistem presnimavanja informacija, bez dozvole izdavača.

Zaštitni znaci

Kompjuter Biblioteka i „Packt Publishing“ su pokušali da u ovoj knjizi razgraniče sve zaštitne oznake od opisnih termina, prateći stil isticanja oznaka velikim slovima.

Autor i izdavač su učinili velike napore u pripremi ove knjige, čiji je sadržaj zasnovan na poslednjem (dostupnom) izdanju softvera. Delovi rukopisa su možda zasnovani na predizdanju softvera dobijenog od strane proizvođača. Autor i izdavač ne daju nikakve garancije u pogledu kompletnosti ili tačnosti navoda iz ove knjige, niti prihvataju ikakvu odgovornost za performanse ili gubitke, odnosno oštećenja nastala kao direktna ili indirektna posledica korišćenja informacija iz ove knjige.

Mojoj majci Zahidi Khatoon i mom ocu Muhammadu Ayazu za njihove molitve i požrtvovanost i za to što su mi poslužili kao primer odlučnosti tokom mog odgoja. Mojoi Nani (baki) Aarif-un-Nisa Begum za tone molitava za moj uspeh. I mojoj supruzi Najli Obaid, koja mi je pružila mnogo ljubavi tokom našeg zajedničkog životnog puta.

Muhammad Ahsan Ayaz

PREDGOVOR

Poznajem Ahsana i radim sa njim više od 10 godina. On je jedan od vodećih stručnjaka globalne Angular zajednice. Veoma je strastven u služenju zajednici programera i ima uticaja na svet, tako što obučava pojedince da pomognu da svet postane bolje mesto, zahvaljujući razvoju softvera. Održao je veliki broj govora i sesija širom sveta o JavaScriptu, Angularu i veb razvoju, a aktivno kreira video tutorijale koje postavlja na svoj YouTube kanal i piše članke na svojoj veb stranici. Ova knjiga je njegov najnoviji napor da ostavi dobar utisak i da obuči što više ljudi i da im pomogne da nauče Angular i da sazrevaju kao Angular programeri.

Ahsan ovom sveobuhvatnom knjigom vodi čitaoce na „putovanje“ razumevanja osnovnih koncepta Angulara i načina implementacije jediničnih i end-to-end testova u Angular aplikacijama. Recepti obuhvataju obrasce koji su vođeni šablonima i reaktivne obrasce, do istraživanja kako da kreirate svoje prilagođene kontrole obrazaca. Štaviše, nećete samo učiti o „stvarima“ kao što su Angular animacije, Angular ruter i upravljanje stanjem pomoću NgRx-a, već ćete i „zaroniti“ u neke neverovatne alatke i API-e pomoću Angular CDK-a. I, na kraju, ali ne najmanje važno, izradićete, koristeći Angular, nešto o čemu softverska industrija godinama „govori“ - progresivne veb aplikacije (PWA - Progressive Web App).

Ukratko, Ahsan je „pretočio“ godine učenja i iskustva u kreiranje ove knjige. Knjiga vam obezbeđuje doživljaj scenarija iz stvarnog života i njihovih tehničkih rešenja u obliku receptata. Sadržaj ove knjige je praktičan, precizan i dobro objašnjen.

Pošto poznajem Ahsana veoma dugo, mogu reći da on posvećuje svu svoju snagu i sposobnost da, kada odluči da uradi nešto, uradi to najbolje što može, a ova knjiga nije izuzetak. A zahvaljujući receptima, izvornom kodu i načinu na koji vas Ahsan vodi kroz sadržaj ove knjige, sigurno ćete mnogo naučiti i bićete bolji softverski inženjer i Angular programer.

Najla Obaid

poslovni analitičar u kompaniji „IOMechs“

O AUTORU

Muhammad Ahsan Ayaz je Angular programer i Synchron softverski arhitekta. Predaje programiranje širom sveta poslednjih osam godina pomoću članaka, video sadržaja, individualnog mentorstva i tehničkih govora na različitim globalnim konferencijama. Razvio je nekoliko biblioteka i pluginova koje koriste stotine hiljada programera, uključujući `ingx-device-detector`, koji ima više od pet miliona instalacija i više od 2.000 GitHub projekata koji ga koriste. On piše na blogu <https://ahsanayaz.com> i ima YouTube kanal na adresi <https://ahsanayaz.com/youtube>, na kojem redovno postavlja video tutorijale za razvoj veb aplikacija i mobilnih aplikacija. Osim što voli programiranje, Ahsan voli da putuje i da igra video igre sa više igrača. Živi sa suprugom u Švedskoj.

Zahvaljujem se ljudima koji su mi bliski i koji su me podržavali svih ovih godina, posebno mojim roditeljima Zahidi Khatoon i Muhammadu Ayazu i mojoj supruzi Najli. Takođe se zahvaljujem čitaocima ove knjige i drugih knjiga koje ću pisati u budućnosti i ljudima koji me prate na mojim profilima na društvenim mrežama i koji su pretplaćeni na moj YouTube kanal.

O RECENZENTU

Pawel Czekaj je diplomirao računarske nauke. Ima 12 godina iskustva kao frontend programer. Trenutno radi kao vodeći razvojni programer u kompaniji „Ziflow Ltd“. Specijalizovan je za radne okvire AngularJS, Angular, Amazon Web Services, Auth0, NestJS i druge. Trenutno izrađuje rešenja za proveru na nivou preduzeća, u potpunosti zasnovana na Angularu.

„PACKT“ TRAŽI AUTORE KAO ŠTO STE VI

Ako ste zainteresovani da postanete autor za „Packt“, prijavite se na stranicu authors.packtpub.com. Saradujemo sa hiljadama programera i tehničkih profesionalaca da bismo im pomogli da podele svoje mišljenje sa globalnom tehničkom zajednicom. Možete da podnesete osnovnu prijavu, da se prijavite za specifičnu temu za koju tražimo autore ili da pošaljete neke svoje ideje.

Predgovor

Angular je jedan od najpopularnijih radnih okvira u svetu za izradu ne samo veb aplikacija, već i mobilnih i desktop aplikacija. Ovaj radni okvir koji obezbeđuje Google koriste milioni aplikacija. Iako je radni okvir dobro prilagođen za bilo koju skalu primene, preduzeća posebno vole Angular zbog njegove dogmatike i zbog doslednog eko-sistema koji uključuje sve alatke koje su vam potrebne za kreiranje aplikacija zasnovanih na veb tehnologijama.

Dok je učenje osnovnih tehnologija, poput JavaScripta, HTML-a i CSS-a, apsolutno neophodno za napredak veb programera, kada je reč o radnom okviru, učenje osnovnih koncepata samog radnog okvira je takođe važno. Kada je reč o Angularu, možemo učiniti neverovatne „stvari“ pomoću veb aplikacija, upoznavajući i koristeći prave alatke u Angular eko-sistemu. Sada „u igru“ ulazi ova knjiga.

Kome je namenjena ova knjiga

Ova knjiga je napisana za srednje i napredne Angular programere da bi usavršili svoje veštine razvoja Angulara pomoću recepata koje mogu lako pratiti, mogu eksperimentisati pomoću njih i mogu uvežbavati svoje varijacije. Nećete učiti samo iz recepata, već i iz stvarnih projekata iz realnog života povezanih sa receptima. Dakle, u ovim receptima i projektima postoji mnogo skrivenih „dragulja“ za vas.

Šta obuhvata ova knjiga

U Poglavlju 1, „Pobednička komunikacija komponenta“, objašnjene su različite tehnike koje se koriste za implementaciju komunikacije između komponenta u Angularu. Obuhvaćeni su i dekoratori, usluge i hooks „životnog ciklusa“ `@Input()` i `@Output()`. Sadrži i recept za kreiranje dinamičke Angular komponente.

Poglavlje 2, „Razumevanje i korišćenje Angular direktiva“, sadrži uvod u Angular direktive i neke recepte koji koriste Angular direktive, uključujući direktive atributa i strukturalne direktive.

Poglavlje 3, „Magija injektovanja zavisnosti u Angularu“, sadrži recepte koji obuhvataju opcione zavisnosti, konfigurisanje tokena za injektovanje i upotrebu metapodataka `providedIn: 'root'` za Angular servise, provajdere vrednosti i provajdere klasa.

Poglavlje 4, „Razumevanje Angular animacija“, sadrži recepte za implementaciju animacija sa više stanja, neverovatnih animacija, animacija ključnih kadrova i animacija za prebacivanje između ruta u vašim Angular aplikacijama.

Poglavlje 5, „Angular i RxJS - kombinovane fenomenalnosti“, sadrži recepte za RxJS instance i statičke metode. Takođe sadrži neke recepte za upotrebu operatora `combineLatest`, `flatMap` i `switchMap` i neke savete i trikove za korišćenje RxJS strimova.

Poglavlje 6, „Upravljanje reaktivnim stanjem pomoću NgRxa“, sadrži recepte koji se odnose na čuvenu NgRx biblioteku i njene osnovne koncepte. Obuhvata osnovne koncepte, kao što su NgRx akcije, reduktori, selektori i efekti, i bavi se korišćenjem paketa, kao što su `@ngrx/store-devtools` i `@component/store`.

U Poglavlju 7, „Razumevanje Angular navigacije i usmeravanja“, istražujemo recepte za sporo učitanje rute, čuvare ruta, strategije za učitanje ruta unapred i neke zanimljive tehnike koje će se koristiti u Angular ruteru.

Poglavlje 8, „Savladavanje Angular obrazaca“, sadrži recepte za obrasce zasnovane na šablonima, za reaktivne obrasce, za obrasce za validaciju, za obrasce za testiranje i za kreiranje kontrole obrazaca.

Poglavlje 9, „Angular i Angular CDK“, ima mnogo odličnih Angular CDK recepata, uključujući recepte za virtuelno pomeranje, za navigaciju tastaturom, za overlay API, za clipboard API, za CDK drag and drop, za CDK stepper API i za CDK textfield API.

Poglavlje 10, „Pisanje jediničnih testova u Angularu pomoću Jesta“, sadrži recepte za jedinično testiranje pomoću Jesta, istraživanje globalnih simulatora u Jestu, simuliranje servisa/podređenih komponenta/kanala, korišćenje Angular CDK komponentnih testiranja i objekte Observable za jedinično testiranje.

Poglavlje 11, „E2E testiranja u Angularu pomoću Cypressa“, sadrži recepte za E2E testiranje pomoću Cypressa u Angular aplikacijama. Obuhvata obrasce za validaciju, čekanje na XHR pozive, simuliranje HTTP odgovora na pozive, korišćenje pratećih paketa pomoću Cypressa i korišćenje uređaja za testiranje u Cypressu.

Poglavlje 12, „Optimizacije performansi u Angularu“, sadrži neke odlične tehnike za poboljšanje performansi aplikacije Angular korišćenjem strategije otkrivanja promena OnPush, odloženo učitavanje ruta funkcija, odvajanje detektora promena od komponente, korišćenje veb workera u Angularu, korišćenje čistih kanala, dodavanje budžetiranja zasnovanog na performansama u Angular aplikaciju i korišćenje analizatora `webpack-bundle`.

Poglavlje 13, „Izrada PWA-ova pomoću Angulara“, sadrži recepte za kreiranje PWA-ova pomoću Angulara. Obuhvata navođenje boje teme za PWA, korišćenje tamnog režima uređaja, obezbeđivanje prilagođenog upita za instalaciju PWA, keširanje zahteva unapred pomoću Angularovog workera servisa i korišćenje App Shella.

Izvueite maksimum iz ove knjige

Recepti u ovoj knjizi izrađeni su pomoću radnog okvira Angular v12, a Angular prati semantičku verziju za svoja izdanja. Pošto se Angular stalno poboljšava radi stabilnosti, Angular tim je obezbedio predvidljiv ciklus izdanja za ažuriranja. Učestalost izdanja je sledeća:

- glavno izdanje na svakih šest meseci
- jedno do tri manja izdanja za svako glavno izdanje
- izdanje „zakrpa“ i predizdanje (sledeće ili rc) skoro svake nedelje

Izvor:<https://angular.io/guide/releases#release-frequency>

SOFTVER KOJI JE OBUHVAĆEN U KNJIZI	ZAHTEVI OPERATIVNOG SISTEMA
Angular	Windows, macOS ili Linux
TypeScript 4.2.4+	
ECMAScript 11	

Ako koristite digitalnu verziju ove knjige, savetujemo vam da sami otkucate kod ili pristupite kodu iz GitHub spremišta knjige (link je dostupan u sledećem odeljku). Na taj način ćete izbeći moguće greške koje se odnose na kopiranje i „lepljenje“ koda.

Kada završite čitanje knjige, obavezno pošaljite post na adresu <https://ahsanayaz.com/twitter> da biste mi predočili vaše povratne informacije o knjizi. Osim toga, kod koji ste dobili uz ovu knjigu možete da izmenite po svojoj želji, da ga postavite u svoje GitHub spremište i da ga podelite. Pobrinuću se da ga ponovo objavim :)

Preuzimanje datoteka primera koda

Možete da preuzmete datoteke sa primerima koda za ovu knjigu na adresi <https://bit.ly/2XMANKH>

Kod, ako postoji njegovo ažuriranje, biće ažuriran u GitHub spremištu: <https://bit.ly/3Ejt4dg>

Na raspolaganju su vam i drugi paketi kodova iz našeg bogatog kataloga knjiga i video zapisa dostupnih na adresi <https://github.com/PacktPublishing/>.

Preuzimanje kolornih slika

Takođe smo obezbedili PDF datoteku koja sadrži kolorne snimke ekrana i dijagrama koji su upotrebljeni u knjizi. Možete da preuzmete ovu datoteku sa adrese: <https://bit.ly/3bdgfoA>

Upotrebljene konvencije

U ovoj knjizi upotrebljen je veliki broj konvencija za tekst.

CodeInText ukazuje na reči koda u tekstu, u nazivima tabele baze podataka, u nazivima direktorijuma, u nazivima fajlova, u ekstenzijama fajlova, putanjama, lažnim URL-ovima, korisničkim unosima i Twitter postovima. Primer je sledeći: „Sada ćemo premestiti kod iz datoteke `the-amazing-listcomponent.html` u datoteku `the-amazing-list-item.component.html` radi označavanja stavke.“

Blok koda je postavljen na sledeći način:

```
openMenu($event, itemTrigger) {  
  if ($event) {  
    $event.stopImmediatePropagation();  
  }  
  this.popoverMenuTrigger = itemTrigger;  
  this.menuShown = true;  
}
```

Kada želimo da privučemo vašu pažnju na određeni deo bloka koda, relevantne linije ili stavke će biti ispisane zadebljanim slovima:

```
.menu-popover {  
  ...  
  &::before {...}  
  
  &--up {  
    transform: translateY(-20px);  
    &::before {  
      top: unset !important;  
      transform: rotate(180deg);  
      bottom: -10px;  
    }  
  }  
  
  &__list {...}  
}
```

Zadebljana slova Novi termini, važne reči ili reči koje vidite na ekranu - na primer, u menijima ili okvirima za dijalog, biće prikazani u tekstu. **zadebljanim slovima**. Na primer: „Primitićete da ne možete videti ceo sadržaj unosa - ovo je pomalo dosadno i u najboljim okolnostima, jer sadržaj ne možete zaista pregledati pre nego što pritisnete dugme **Action**.”

Saveti ili trikovi se prikazuju ovako.



Kontaktirajte sa nama

Povratne informacije naših čitalaca su uvek dobrodošle.

Opšte povratne informacije - Ako imate pitanja o bilo kojem aspektu ove knjige, pošaljite nam e-poruku na adresu kombib@gmail.com.

Štamparske greške - Iako smo preduzeli sve mere da bismo obezbedili tačnost sadržaja, greške su moguće. Ako pronađete grešku u nekoj od naših knjiga – u tekstu ili u kodu, bili bismo zahvalni ako biste nam to javili. Na taj način možete da pomognete drugim čitaocima da izbegnu frustracije, a nama da poboljšamo sledeće verzije ove knjige. Ako pronađete grešku, molimo vas da nas o tome obavestite na email kombib@gmail.com.

Piraterija - Ako pronađete ilegalnu kopiju naših knjiga u bilo kojoj formi na Internetu, molimo vas da nas o tome obavestite i da nam pošaljete adresu lokacije ili naziv veb sajta. Pošaljite nam poruku na adresu kombib@gmail.com i pošaljite nam link ka sumnjivom materijalu.

Ako ste zainteresovani da postanete autor - Ako postoji tema za koju ste specijalizovani i zainteresovani ste da pišete ili sarađujete na nekoj od knjiga, pogledajte vodič za autore na adresi authors.packtpub.com.

Podelite svoje mišljenje

Voleli bismo da saznamo vaše mišljenje nakon što pročitate „Angular kuvar“! Kliknite ovde da biste direktno pristupili stranici Amazon pregleda <https://packt.link/r/1838989439> za ovu knjigu i napišite vaše povratne informacije.

Vaša recenzija je važna za nas i tehničku zajednicu i pomoći će nam da se uverimo da isporučujemo sadržaj odličnog kvaliteta.



Postanite član Kompjuter biblioteke

Kupovinom jedne naše knjige stekli ste pravo da postanete član Kompjuter biblioteke. Kao član možete da kupujete knjige u pretplati sa 40% popustai učestvujete u akcijama kada ostvarujete popuste na sva naša izdanja. Potrebno je samo da se prijavite preko formulara na našem sajtu. Link za prijavu: <http://bit.ly/2TxeK5a>

Skenirajte QR kod
registrujte knjigu
i osvojite nagradu



1

Pobednička komunikacija komponentata

U ovom poglavlju upoznaćete komunikaciju komponentata u Angularu. Naučićete različite tehnike za uspostavljanje komunikacije između komponentata i koja je tehnika pogodna u određenoj situaciji. Takođe ćete naučiti kako da kreirate dinamičku Angular komponentu.

Slede recepti koje ćemo obuhvatiti u ovom poglavlju:

- komunikacija komponentata pomoću komponentata `@Input(s)` i `@Output(s)`
- komunikacija komponentata pomoću servisa
- korišćenje settera za presretanje izmena svojstava ulaza
- korišćenje `ngOnChanges`a za presretanje izmena svojstava unosa
- pristup podređenoj komponenti u nadređenom šablonu pomoću promenljivih šablona
- pristup podređenoj komponenti u klasi nadređenih komponentata pomoću `ViewChild`a
- kreiranje vaše prve dinamičke komponente u Angularu

Tehnički zahtevi

Zbog recepata u ovom poglavlju proverite da li su na vašem računaru instalirani Git i Node.js. Takođe morate imati instaliran paket `@angular/cli`, koji možete da instalirate pomoću komande `install -g @angular/cli` iz vašeg terminala. Kod za ovo poglavlje možete pronaći na adresi <https://github.com/PacktPublishing/Angular-Cookbook/tree/master/chapter01>.

Komunikacija komponentata pomoću komponentata `@Input` i `@Output`

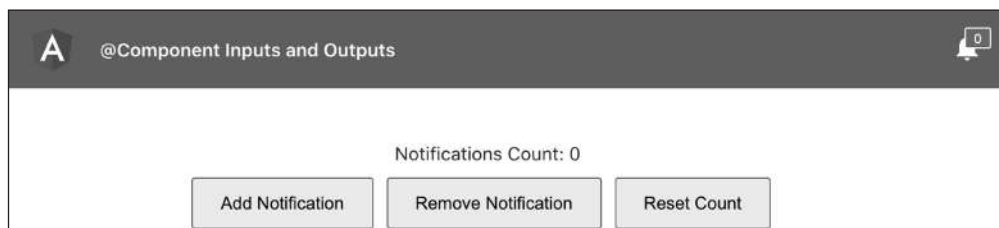
Prvo ćete koristiti aplikaciju sa nadređenom komponentom i dve podređene komponente. Zatim ćete koristiti Angular dekoratore `@Input` i `@Output` da biste uspostavili komunikaciju između njih, koristeći atribute i `EventEmitters`.

Priprema

Projekat koji ćemo koristiti nalazi se u `chapter01/start_here/cc-inputs-outputs` unutar kloniranog spremišta:

1. Otvorite projekat u Visual Studio Codeu.
2. Otvorite terminal i pokrenite `npm install` da biste instalirali zavisnosti projekta. Nakon toga, pokrenite `ng serve -o`.

Sada bi trebalo da se otvori aplikacija u novoj kartici pregledača i trebalo bi da vidite sledeće:



Slika 1.1 Aplikacija `cc-inputs-outputs` koja je pokrenuta na adresi `http://localhost:4200`

Kako to uraditi...

Sada imamo aplikaciju koja sadrži komponente `AppComponent`, `NotificationsButtonComponent` i `NotificationsManagerComponent`. Iako je `AppComponent` nadređena komponenta druge dve pomenute komponente, ne postoji apsolutno nikakva komunikacija između njih za sinhronizaciju vrednosti broja obaveštenja. Uspostavićete odgovarajuću komunikaciju između njih pomoću sledećih koraka:

1. Premestite promenljivu `notificationsCount` iz komponente `NotificationsManagerComponent` i smestite je u komponentu `AppComponent`. Da biste to uradili, samo kreirajte svojstvo `notificationsCount` u datoteci `app.component.ts`:

```
export class AppComponent {  
  notificationsCount = 0;  
}
```

2. Zatim, konvertujte svojstvo `notificationsCount` u datoteci `notifications-manager.component.ts` u komponentu `@Input()`. Tu komponentu preimenujte u `count` i zamenite njenu upotrebu na sledeći način:

```
import { Component, OnInit, Input } from '@angular/core';  
@Component({  
  selector: 'app-notifications-manager',  
  templateUrl: './notifications-manager.component.html',  
  styleUrls: ['./notifications-manager.component.scss']  
})  
export class NotificationsManagerComponent implements  
  OnInit {  
  @Input() count = 0  
  constructor() { }  
  ngOnInit(): void {  
  }  
  addNotification() {  
    this.count++;  
  }  
  removeNotification() {  
    if (this.count == 0) {  
      return;  
    }  
    this.count--;
```

```
}  
resetCount() {  
  this.count = 0;  
}  
}
```

3. Ažurirajte datoteku `notifications-manager.component.html` da biste koristili `count`, umesto `notificationsCount`:

```
<div class="notif-manager">  
  <div class="notif-manager__count">  
    Notifications Count: {{count}}  
  </div>  
  ...  
</div>
```

4. Zatim, prosledite svojstvo `notificationsCount` iz datoteke `app.component.html` elementu `<pp-notifications-manager>` kao unos:

```
<div class="content" role="main">  
  <app-notifications-manager  
    [count]="notificationsCount">  
  </app-notifications-manager>  
</div>
```

Sada možete testirati da li se vrednost ispravno prosleđuje iz datoteke `app.component.html` u `app-notifications-manager` dodeljivanjem vrednosti `notificationsCount` u datoteci `app.component.ts` kao 10. Videćete da će u komponenti `NotificationsManagerComponent` početna prikazana vrednost biti 10:

```
export class AppComponent {  
  notificationsCount = 10;  
}
```

5. Sada kreirajte `@Input()` u datoteci `notifications-button.component.ts` pod nazivom `count`:

```
import { Component, OnInit, Input } from '@angular/core';  
...  
export class NotificationsButtonComponent implements  
  OnInit {
```

```
@Input() count = 0;
```

```
...
```

```
}
```

6. Prosledite `notificationsCount` u `<pp-notifications-button>` iz datoteke `app.component.html`:

```
<!-- Toolbar -->
```

```
<div class="toolbar" role="banner">
```

```
...
```

```
<span>@Component Inputs and Outputs</span>
```

```
<div class="spacer"></div>
```

```
<div class="notif-bell">
```

```
  <app-notifications-button  
    [count]="notificationsCount">
```

```
  </app-notifications-button>
```

```
</div>
```

```
</div>
```

```
...
```

7. Koristite unos `count` u datoteci `notifications-button.component.html`, zajedno sa ikonom zvona za obaveštenje:

```
<div class="bell">
```

```
  <i class="material-icons">notifications</i>
```

```
  <div class="bell__count">
```

```
    <div class="bell__count__digits">
```

```
      {{count}}
```

```
    </div>
```

```
  </div>
```

```
</div>
```

Sada bi trebalo da vidite i vrednost 10 za broj ikona zvona za obaveštenje.

Ako promenite broj dodavanjem/uklanjanjem obaveštenja iz komponente

NotificationsManagerComponent, broj na ikoni zvona za obaveštenja se neće promeniti.

8. Da bismo preneli promenu iz `NotificationsManagerComponent` na `NotificationsButtonComponent`, sada ćemo koristiti Angular komponente `@Output`. Koristite `@Output` i `@EventEmitter` iz `'@angular/core'` unutar datoteke `notifications-manager.component.ts`:

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

...

export class NotificationsManagerComponent implements
OnInit {
  @Input() count = 0
  @Output() countChanged = new EventEmitter<number>();
  ...
  addNotification() {
    this.count++;
    this.countChanged.emit(this.count);
  }
  removeNotification() {
    ...
    this.count--;
    this.countChanged.emit(this.count);
  }
  resetCount() {
    this.count = 0;
    this.countChanged.emit(this.count);
  }
}
```

9. Zatim ćemo u datoteci `app.component.html` preslušati prethodno emitovani događaj iz komponente `NotificationsManagerComponent` i u skladu s njim ažurirati svojstvo `notificationsCount`:

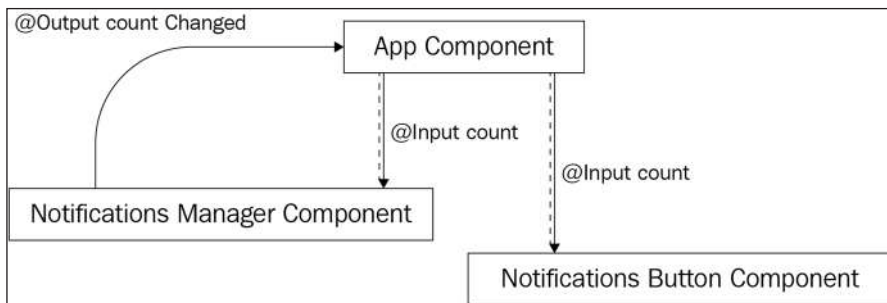
```
<div class="content" role="main">
  <app-notifications-manager
    (countChanged)="updateNotificationsCount($event)"
    [count]="notificationsCount"></app-notifications-
  manager>
</div>
```

10. Pošto smo prethodno preslušali događaj `countChanged` i pozvali metod `updateNotificationsCount`, moramo da kreiramo ovaj metod u datoteci `app.component.ts` i u skladu s njim da ažuriramo vrednost svojstva `notificationsCount`:

```
export class AppComponent {  
  notificationsCount = 10;  
  updateNotificationsCount(count: number) {  
    this.notificationsCount = count;  
  }  
}
```

Kako to funkcioniše...

Da bi komponente komunicirale pomoću komponenata `@Input` i `@Output`, tok podataka će se uvek kretati **od podređenih komponenata do nadređene komponente**, pri čemu se može obezbediti nova (ažurirana) vrednost kao *ulaz* traženim podređenim komponentama. Dakle, `NotificationsManagerComponent` emituje događaj `countChanged`. `AppComponent` (kao nadređena komponenta) osluškuje događaj i ažurira vrednost `notificationsCount`, koja automatski ažurira svojstvo `count` u komponenti `NotificationsButtonComponent`, jer se `notificationsCount` prosleđuje kao broj `@Input()` komponenti `NotificationsButtonComponent`. Na sledećoj slici prikazan je ceo proces.



Slika 1.2 Kako komunikacija komponenata funkcioniše pomoću ulaza i izlaza

Takođe pogledajte

- Kako komuniciraju Angular komponente?
<https://www.thirdrocktechkno.com/blog/how-angular-components-communicate>
- „ComponentCommunication in Angular“, Dhananjay Kumar:
<https://www.youtube.com/watch?v=I8Z8g9APaDY>

Komunikacija komponentata pomoću servisa

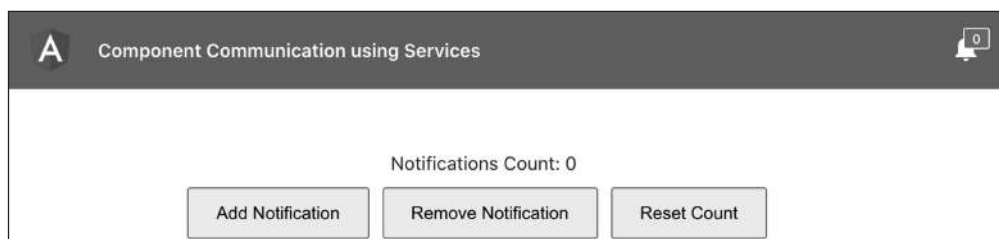
U ovom receptu ćete prvo koristiti aplikaciju sa nadređenom i podređenom komponentom. Zatim ćete koristiti Angular servis za uspostavljanje komunikacije između njih. Koristićemo strimove `BehaviorSubject` i `Observable` za komunikaciju između komponentata i servisa.

Priprema

Projekat za ovaj recept nalazi se u `chapter01/start_here/cc-services`:

1. Otvorite projekat u Visual Studio Codeu.
2. Otvorite terminal i pokrenite `npm install` da biste instalirali zavisnosti projekta.
3. Nakon toga, pokrenite `ng serve -o`.

Sada bi trebalo da se otvori aplikacija u novoj kartici pregledača i aplikaciju bi trebalo da vidite kao na sledećoj slici.



Slika 1.3 Aplikacija `cc-services` koja je pokrenuta na adresi `http://localhost:4200`

Kako to uraditi...

Slično prethodnom receptu, imamo aplikaciju sa komponentama `AppComponent`, `NotificationsButtonComponent` i `NotificationsManagerComponent`.

`AppComponent` je nadređena komponenta drugim dvema komponentama koje smo prethodno pomenuli i moramo da uspostavimo odgovarajuću komunikaciju između njih pomoću sledećih koraka:

1. Kreirajte novu fasciklu u projektu `chapter10/start_here/cc-services/src/app`, koji je nazvan `services`. Ovde ćemo smestiti naš novi servis.
2. Iz terminala uđite u projekat, tj. u `chapter01/start_here/cc-services`, i kreirajte novi servis, koji se zove `NotificationService`, na sledeći način:

```
ng g service services/Notifications
```

3. Kreirajte strim `BehaviorSubject` sa nazivom `count` u datoteci `notifications.service.ts` i inicijalizujte ga pomoću vrednosti `0`, jer `BehaviorSubject` zahteva početnu vrednost:

```
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';
@Injectable({
  providedIn: 'root'
})
export class NotificationsService {
  private count: BehaviorSubject<number> = new
  BehaviorSubject<number>(0);
  constructor() { }
}
```

Imajte na umu da je strim `BehaviorSubject` svojstvo `private` i da ćemo ga kasnije ažurirati samo unutar servisa, koristeći metod `public`.

4. Sada kreirajte `Observable` pod nazivom `count$`, koristeći metod `.asObservable()` u `countBehaviorSubjectu`:

```
import { Injectable } from '@angular/core';
import { BehaviorSubject, Observable } from 'rxjs';
...
export class NotificationsService {
  private count: BehaviorSubject<number> = new
  BehaviorSubject<number>(0);
  count$: Observable<number> = this.count.asObservable();
}
```

```
...
```

```
}
```

5. Konvertujte svojstvo `notificationsCount` u datoteci `notifications-manager.component.ts` u `strim Observable`, koji se zove `notificationsCount$`. Injektujte `NotificationsService` u komponentu i dodelite broj (`count$`) servisa za `Observable` promenljivoj komponente `notificationsCount$`:

```
import { Component, OnInit } from '@angular/core';
import { Observable } from 'rxjs';
import { NotificationsService } from '../services/
notifications.service';
...
export class NotificationsManagerComponent implements
OnInit {
  notificationsCount$: Observable<number>;
  constructor(private notificationsService:
NotificationsService) { }
...
ngOnInit(): void {
  this.notificationsCount$ = this.notificationsService.
count$;
}
...
}
```

6. Komentarišite kod koji za sada ažurira broj obaveštenja (na to ćemo se vratiti kasnije):

```
...
export class NotificationsManagerComponent implements
OnInit {
  ...
  addNotification() {
    // this.notificationsCount++;
  }
  removeNotification() {
    // if (this.notificationsCount == 0) {
    //   return;

```

```

    // }
    // this.notificationsCount--;
  }
  resetCount() {
    // this.notificationsCount = 0;
  }
}

```

7. Koristite Observable `notificationsCount$` u datoteci `notifications-manager.component.html` sa kanalom `async` da biste prikazali njegovu vrednost:

```

<div class="notif-manager">
  <div class="notif-manager__count">
    Notifications Count: {{notificationsCount$ | async}}
  </div>
  ...
</div>

```

8. Sada na sličan način injektujte `NotificationsService` u datoteku `notifications-button.component.ts`, unutar komponente `NotificationsButtonComponent` kreirajte strim `ObservablenotificationsCount$` i dodelite mu Observable broj servisa (`count$`):

```

import { Component, OnInit } from '@angular/core';
import { NotificationsService } from '../services/
notifications.service';
import { Observable } from 'rxjs';
...
export class NotificationsButtonComponent implements
OnInit {
  notificationsCount$: Observable<number>;
  constructor(private notificationsService:
NotificationsService) { }
...
  ngOnInit(): void {
    this.notificationsCount$ = this.notificationsService.
count$;
  }
}

```

9. Koristite `Observable notificationsCount$` u datoteci `notifications-button.component.html` sa kanalom `async`:

```
<div class="bell">
  <i class="material-icons">notifications</i>
  <div class="bell__count">
    <div class="bell__count__digits">
      {{notificationsCount$ | async}}
    </div>
  </div>
</div>
```

Ako sada „osvežite“ aplikaciju, trebalo bi da vidite vrednost 0 i za komponentu menadžera obaveštenja i za komponentu dugmeta za obaveštenja.

10. Promenite početnu vrednost za `countBehaviorSubject` u 10 i pogledajte da li se ta promena odražava na obe komponente:

```
...
export class NotificationsService {
  private count: BehaviorSubject<number> = new
    BehaviorSubject<number>(10);
  ...
}
```

11. Kreirajte metod `setCount` u datoteci `notifications.service.ts` da biste mogli da ažurirate vrednost `countBehaviorSubject`:

```
...
export class NotificationsService {
  ...
  constructor() {}
  setCount(countVal) {
    this.count.next(countVal);
  }
}
```

12. Sada, kada imamo postavljen metod `setCount`, upotrebićemo ga u datoteci `notifications-manager.component.ts` da bismo ažurirali njegovu vrednost na osnovu klikova na dugme. Da bismo to učinili, moramo da pribavimo najnoviju vrednost za `Observable notificationsCount$`, a zatim da izvršimo neku radnju. Prvo ćemo da kreiramo metod `getCountValue` unutar komponente `NotificationsManagerComponent` na način koji je prikazan u nastavku i koristićemo `subscribe` zajedno sa operatorom `new` u `Observable`u `notificationsCount$` da bismo dobili najnoviju vrednost:

```
...
import { first } from 'rxjs/operators';
...
export class NotificationsManagerComponent implements
OnInit {
  ngOnInit(): void {
    this.notificationsCount$ = this.notificationsService.
count$;
  }
  ...
  getCountValue(callback) {
    this.notificationsCount$
      .pipe(
        first()
      ).subscribe(callback)
  }
  ...
}
```

13. Sada ćemo koristiti metod `getCountValue` u okviru metoda `addNotification`, `removeNotification` i `resetCount`. Moraćemo da prosledimo funkciju povratnog poziva iz ovih metoda u metod `getCountValue`. Prvo ćemo upotrebiti metod `addNotification`:

```
import { Component, OnInit } from '@angular/core';
import { Observable } from 'rxjs';
import { NotificationsService } from '../services/
notifications.service';
import { first } from 'rxjs/operators';
```

```
...
```

```
export class NotificationsManagerComponent implements
OnInit {
    ...
    addNotification() {
        this.getCountValue((countVal) => {
            this.notificationsService.setCount(++countVal)
        });
    }
    ...
}
```

Zahvaljujući prethodnom kodu, već bi trebalo da vidite obe komponente koje pravilno odražavaju ažurirane vrednosti uvek kada kliknete na dugme **AddNotification**.

14. Sada ćemo implementirati istu logiku za `removeNotification` i `resetCount`:

```
...
export class NotificationsManagerComponent implements
OnInit {
    ...
    removeNotification() {
        this.getCountValue((countVal) => {
            if (countVal === 0) {
                return;
            }
            this.notificationsService.setCount(--countVal);
        })
    }
    resetCount() {
        this.notificationsService.setCount(0);
    }
}
```