|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  |
|  | Summanus |  | Olimijada  Datum: 10. 03. 2018. |
| Uputstvo za Angular 5 | | | |

# Šta je Angular?

Angular je platforma (framework) za kreiranje jednostranih front-end aplikacija na Web-u.  
Naime, aplikacije napravljene u Angular-u se sastoje od samo jedne html strane na kojoj JavaScript kod generiše odgovarajuće delove direktno u Web browser-u.

### Zašto Angular?

Ovakva organizacija Web aplikacija omogućava reaktivnost i brzinu. Generisanje stranica pomoću JavaScripta je brže od dobijanja gotovih stranica od servera.

# Instalacija

Da bi se instalirao Angular, neophodno je da imate instaliran NodeJS, koji se može skinuti na sledećem linku:

<https://nodejs.org/en/>

Nakon uspešne instalacije NodeJS-a, možete instalirati Angular tako što ćete u Command Prompt ukucati komandu:

npm install -g @angular/cli

# Kreiranje projekta (aplikacije)

Angular projekat se kreira na sledeći način:

ng new my-app

Da biste pokrenuli aplikaciju, morate preći u folder aplikacije (u ovom slučaju my-app) i ukucati sledeću komandu:

ng serve

Ova komanda pokreće server, posmatra fajlove, i rebuild-uje aplikaciju svaki put kada sačuvate promene u nekom fajlu.

Ukoliko uz ovu komandu iskoristite opciju –open ili -o, automatski će se otvoriti podrazumevani browser na adresi <http://localhost:4200/>.

# Arhitektura

Aplikacije u Angularu se pišu (prave) pisanjem HTML *šablona*, *komponenti* (klasa) koje upravljaju datim šablonima, dodavanjem logike aplikacije u *servise* i pakovanjem komponenti i servisa u *module*.

### Moduli

Angular aplikacije su modularne, i Angular ima sopstveni sistem modularnosti koji se zove *NgModules*.

Svaka aplikacija ima najmanje jednu NgModule klasu, *root module*, nazvanu AppModule. NgModule je svaka klasa sa dekoratorom @NgModule (dekorator – funkcija koja kači metapodatke za neku klasu, kako bi Angular znao šta te klase znače i kako treba da rade).

NgModule uzima jedan objekat metapodataka čije osobine opisuju module. Najbitnije osobine su:

* declarations *– view klase* koje pripadaju modulu (*component,directive,pipe*)
* exports – deklaracije koje su vidljive i mogu se koristiti u šablonima drugih modula
* imports – drugi moduli čije su export-ovane klase potrebne u šablonima u ovom modulu
* providers *–*  kreatori servisa koje ovaj modul prilaže globalnoj kolekciji; vidljivi su u svim delovima aplikacije
* bootstrap *–*  glavni pogled aplikacije, *root component*, koji hostuje ostale poglede

Primer root modula:

|  |
| --- |
| **src/app/app.module.ts** |
| 1. import { NgModule } from ‘@angular/core’; 2. import { BrowserModule } from ‘@angular/platform-browser’; 4. @NgModule( 5. { 6. imports: [ BrowserModule ], 7. providers: [ Logger ], 8. declarations: [ AppComponent ], 9. exports: [ AppComponent ], 10. bootstrap: [ AppComponent ] 11. }) 12. export class AppModule { } |

### 

### Komponente

Komponenta kontroliše deo ekrana koji se zove *view.*

Vi definišete logiku komponente – šta radi kako bi podržača view – u okviru klase. Klasa interaguje sa view-om preko API-ja sa osobinama i metodama.

Za svaku komponentu kreiraju se 4 fajla:

* *naziv.component.ts –* logika komponente
* *naziv.component.spec.ts* – test fajl komponente
* *naziv.component.html –* šablon komponente
* *naziv.component.css* – css kod koji određuje izgled komponente

Primer komponente:

|  |
| --- |
| **src/app/hero-list.component.ts (class)** |
| 1. export class HeroListComponent implements OnInit 2. { 3. heroes: Hero[]; 5. selectedHero: Hero; 7. constructor(){} 9. ngOnInit(){} 10. selectHero(hero: Hero) 11. { 12. this.selectedHero = hero; 13. } |

### Šabloni

Šablon (template) je HTML kod koji određuje kako će Angular da prikaže komponentu.

Primer šablona:

|  |
| --- |
| **src/app/hero-list.component.html** |
| 1. <h2> Hero List </h2> 2. <p>Pick a hero from the list</p> 3. <ul> 4. <li \*ngFor=”let hero of heroes” (click)=”selectedHero(hero)”> 5. {{ hero.name }} 6. </li> 7. </ul> |

Delovi koda, kao što je \*ngFor, {{ hero.name }}, i (click) koriste Angular sintaksu za šablone.

### Metapodaci

Kada se vratimo na klasu HeroListComponent, vidimo da je to samo klasa (a ne komponenta) i da u njoj nema ničeg što je karakteristično za Angular.

Da bi Angular znao da je HeroListComponent u stvari komponenta, morate dodati metapodatke klasi. To se radi pomoću *dekoratora*.

Primer metapodataka:

|  |
| --- |
| **src/app/hero-list.component.ts (metadata)** |
| 1. @Component( 2. { 3. selector: ‘app-hero-list’, 4. templateUrl: ‘./hero-list.component.html’ 5. } 6. export class HeroListComponent implements OnInit 7. { 8. … 9. } |

Neke od opcija dekoratora @Component:

* selector – CSS selektor koji kaže Angularu da kreira i ubaci instance komponente tamo gde nađe tag <app-hero-list> u HTML šablonu
* templateUrl – relativna adresa šablona komponente
* styleUrls – relativna adresa(e) css fajla komponente

### Vezivanje podataka

Vezivanje podataka je mehanizam za koordinisanje delova šablona i delova komponente.

U Angularu postoje 4 tipa sintakse za vezivanje podataka:

* {{property}} – prikazuje vrednost *property-ja* na mestu gde je postavljen (interpolacija)
  + <li>{{hero.name}}</li>
* [property]=”value” – prosleđuje vrednost *value* od taga u kome se nalazi ka *property-ju* (vezivanje svojstva)
  + <app-hero-detail [hero]=”selectedHero”></app-hero-detail>
* (event)=”handler” – zove *handler* kada se desi događaj *event* (vezivanje događaja)
  + <li (click)="selectHero(hero)"></li>
* [(ngModel)]=”property” – dvosmerno vezivanje
  + <input [([ngModel](https://angular.io/api/forms/NgModel))]="hero.name">
  + Vrednost svojstva hero.name se prosleđuje u polje za unos iz komponente, ali se i promene korisnika u polju za unos prosleđuju komponenti, postavljajući vrednost na ono što je korisnik uneo

### Direktive

Direktiva je klasa sa dekoratorom @[Directive](https://angular.io/api/core/Directive). Komponente su direktive sa šablonom.

Druge dve vrste direktiva su *sktrukturne* i *atributske*. Pojavljuju se u okviru taga elemenata slično kao atributi.

Strukturne direktive menjaju raspores dodavajući, uklanjajući i menjajući elemente u DOM-u.

Primer:

|  |
| --- |
| **src/app/hero-list.component.html** |
| 1. <li \*[ngFor](https://angular.io/api/common/NgForOf)="let hero of heroes"></li> 2. <app-hero-detail \*[ngIf](https://angular.io/api/common/NgIf)="selectedHero"></app-hero-detail> |

* \*ngFor kaže Angularu da štampa jedan <li> element po heroju u listi heroes
* \*ngIf uključuje HeroDetail komponentu samo ukoliko selektovan heroj postoji

Atributske direktive menjaju izgled ili ponašanje elementa. U šablonu one izlgedaju skoro isto kao regularni atributi.

Primer atributske direktive je ngModel.

### Servisi

Iako skoro sve može biti servis, to je obično klasa sa uskom, dobro definisanom namenom. Treba da radi nešto specifično i na dobar način.

U vezi servisa ne postoji ništa karakteristično za Angular, ne postoji definicija niti bazna klasa.

Za šta se koriste servisi?  
Servisi se koriste za preuzimanje podataka sa servera, validiraju korisnički unos i sl., kako bi klase komponenti bile kraće i omogućavale dobro korisničko iskustvo.

### Dependency injection

Dependency injection je način da se klasa snabde zavisnostima koje su joj potrebne. Zavisnosti su većinskim delom servisi.

Angular može da odredi koji servisi su potrebni komponentama na osnovu parametara u konstruktoru komponente.

Kada Angular kreira komponentu, prvo pita *injector* za servise koji su potrebni komponenti.   
Injector održava kontejner instanci servisa koje je prethodno kreirao. Ako traženi servis nije u kontejneru, injector ga pravi i dodaje u kontejner pre nego što vrati servis Angularu. Kada su svi traženi servisi kreirani i vraćeni, Angular poziva konstruktor komponente sa tim servisima kao parametrima. Ovaj proces se zove *dependency injection*.

Kako injector zna kako da napravi servis?  
Prvo morate da registrujete provajdera kod injector-a. Provajder je nešto što kreira ili vraća servis, uglavnom sama servisna klasa. Provajderi se registruju u modulima ili komponentama, ali najčešće u root modulu kako bi ista instance servisa bila dostupna u celoj aplikaciji.

# Kako funkcioniše Angular?

Kao što je rečeno, Angular aplikacije se sastoje od samo jedne strane – *index.html*. Ukoliko se aplikacija pravi pomoću Angular CLI, *index.html* izgleda ovako:

|  |
| --- |
| **src/index.html** |
| 1. <!doctype html> 2. <html lang=”en”> 3. <head> 4. <meta charset=”utf-8”> 5. <title> Angular app </title> 6. <base href=”/”> 7. <meta name=”viewport” content=”width=device-width, initial-scale=1”> 8. <link rel=”icon” type=”image/x-icon” href=”favicon.ico”> 9. </head> 10. <body> 11. <app-root></app-root> 12. </body> 13. </html> |

Kao što vidite, samo *telo* ne sadrži nikakav tekst (samo app-root tag). Međutim, kada se aplikacija pokrene, u browser-u se pojavljuje neki tekst (u primeru – „Tour of Heroes“). Kako?

Pri kreiranju aplikacije komandom ng new my-app kreira se i komponenta pod nazivom AppComponent.

Neka *app.component.ts* izgleda ovako:

|  |
| --- |
| **src/app.component.ts** |
| 1. import { Component } from '@angular/core'; 2. @Component({ 3. selector: 'app-root', 4. templateUrl: './app.component.html', 5. styleUrls: ['./app.component.css'] 6. }) 7. export class AppComponent { 8. title = 'Tour of Heroes'; 9. } |

i neka u *app.component.html* stoji samo <h1>{{title}}</h1>.

Kao što vidimo, tekst koji se prikazuje u browser-u je definisan u okviru klase AppComponent, i prikazuje se na način definisan u *app.component.html i app.component.css*.

Kako Angular zna šta treba da prikaže?

Klasa AppComponent ima dekorator @Component u kome je definisan selektor app-root.Taj selektor je upotrebljen u *index.html* fajlu i zbog toga se prikazuje komponenta *AppComponent*.

# Korisni linkovi

Više informacija i detaljnije objašnjenje možete naći na:

* <https://angular.io/guide/quickstart>
* <https://scotch.io/tutorials/creating-a-single-page-todo-app-with-node-and-angular>
* <https://www.youtube.com/watch?v=oa9cnWTpqP8>
* <https://www.youtube.com/watch?v=0eWrpsCLMJQ&list=PLC3y8-rFHvwhBRAgFinJR8KHIrCdTkZcZ>