2. *Property Binding* i *Event Binding*

Sadržaj

- Postavljanje vrednosti obeležja komponente
- Emitovanje događaja u komponenti
- Klase modela domena
- Referentne promenljive u HTML-u

Skrivanje elementa

- Skrivanje HTML elementa moguće je navođenjem **atributa** `hidden`
  `<div **hidden**>...</div>`
- Angular omogućava skrivanje elemenata navođenjem **obeležja** unutar `[]` i dodelom vrednosti `true`
  `<div **[hidden]="true"**>...</div>`
- Ovo u Angularu nazivamo **Input Property Binding**

HTML atribut naspram DOM obeležja

- HTML je skup instrukcija koje određuju kako će se prikazati web stranica
- *Browser* čita HTML i kreira DOM (*Document Object Model*) u memoriji
- Postoje skoro 1 na 1 mapiranja naziva i vrednosti HTML atributa (engl. *attribute*) i njima ekvivalentnih DOM obeležja (engl. *property*)
- Da bi se sakrio element, u HTML-u je dovoljno da postoji atribut `hidden`, tako da će i `hidden="false"` sakriti element
- U DOM-u će element biti sakriven samo kada je `hidden="true"`
- **Angular ne manipuliše HTML atributima već DOM obeležjima**
- Zato se prikazani postupak zove *Input* ***Property*** *Binding*

Input Property Binding

- Unutar `[]` zagrada navodi se naziv DOM obeležja
- Tekst desno od operatora dodele je JavaScript kod koji se izvršava
- Rezultujuća vrednost se dodeljuje navedenom DOM obeležju
- *Input Property Binding* možemo koristiti samo da bismo promenili vrednost obeležja
- *Input Property Binding* **ne možemo** koristi da bismo bili obavešteni da se promenila vrednost obeležja

Skrivanje rešenja zagonetki

- Dodaćemo novo obeležje `hide` u `PuzzleComponent`
- Vrednost tog obeležja iskoristićemo za postavljanje vrednosti DOM obeležja `hidden`
- Potrebno je
  - u `PuzzleComponent` definisati obeležje `hide: boolean`
  - u `PuzzleListComponent` u nizu `puzzles` dodati svakom objektu obeležje `hide` i postaviti na `true`
  - u `template`-u iskoristiti vrednost obeležja `hide` za postavljanje DOM obeležja `hidden`

puzzlelist.component.ts (1)

```
...
@Component({
   selector: 'app-puzzle-list',
   template:`
       <div *ngFor="let p of puzzles">
           <mat-card>
               <mat-card-header>
                   <mat-card-title><h1>{{ p.question }}</h1></mat-card-title>
               </mat-card-header>
               <mat-card-content>
                   <p [hidden]="p.hide">{{ p.solution }}</p>
               </mat-card-content>
           </mat-card>
           <br>
       </div>`
}) ...
```

puzzlelist.component.ts (2)

```
...
export class PuzzleListComponent {
   puzzles: PuzzleComponent[];

   constructor() {
       this.puzzles = [
           {question: "Dva lokvanja oko panja.", solution: "Glava i uši.", hide: true},
           {question: "Bele koke ispod strehe vire.", solution: "Zubi.", hide: true},
           {question: "Vodu pije a živo nije.", solution: "Sunđer.", hide: true}
       ];
   }
}
```

Prikaz odgovora na klik

- Ukoliko želimo da nakon klika na dugme menjamo vidljivost rešenja zagonetke potrebno je da za “click” događaj vežemo odgovarajuću metodu
- Za korišćenje Material Design dugmeta potrebno je dodati odgovarajuće import izraze u komponentu i modul
- Dodavanje dugmeta u *template* za svaku zagonetku

```
<button mat-raised-button color="primary" (click)="p.hide=!p.hide">Prikaži</button>
```

- Alternativni način je da se doda metoda

```
toggle(puzzle) {
  puzzle.hide = !puzzle.hide;
}
```

u `PuzzleListComponent` i pozove u *template*-u

```
<button mat-raised-button color="primary" (click)="toggle(p)">Prikaži</button>
```

Izgled stranice

Output Event Binding

- Unutar （）zagrada navodi se događaj koji se sluša
- Tekst desno od operatora dodele je JavaScript kod koji se izvršava kada se desi događaj koji je naveden
- JavaScript kod koji se izvršava može biti
  - neki izraz ili
  - poziv funkcije

Klase modela domena

- Angular komponente koriste klase modela domena za upravljanje podacima
- Klase modela domena su perzistentne klase sistema
- Analogija sa klasama koje se koriste za objektno-relaciono mapiranje
- U JPA ili Spring Data aplikacijama to su *Entity* klase

Klasa `Puzzle` u `puzzle.model.ts`

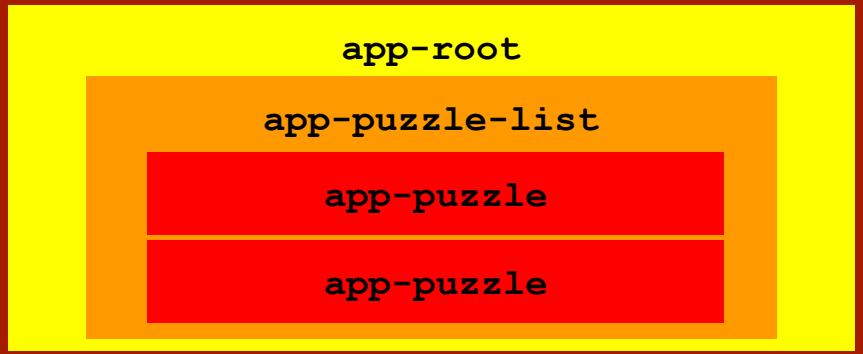
```
export class Puzzle {
   question: string;
   solution: string;
   hide: boolean;

   constructor(question, solution) {
       this.question = question;
       this.solution = solution;
       this.hide = true;
   }

   toggle() {
       this.hide = !this.hide;
   }
}
```

Modifikovana klasa `PuzzleListComponent`

```
@Component({
   selector: 'app-puzzle-list',
   template:`
       <div *ngFor="let p of puzzles">
           ... <button mat-raised-button color="primary" (click)="p.toggle()">Prikaži</button> ...
       </div>`
})
export class PuzzleListComponent {
   puzzles: Puzzle[];
   constructor() {
       this.puzzles = [
           new Puzzle("Dva lokvanja oko panja.", "Glava i uši."),
           new Puzzle("Bele koke ispod strehe vire.", "Zubi."),
           new Puzzle("Vodu pije a živo nije.", "Sunđer.")
       ];
   }
}
```

Komunikacija Angular komponenti

- Web stranice Angular aplikacije po pravilu sadrže više od jedne komponente
- Komponente treba da međusobno komuniciraju i prosleđuju podatke
- Da bi se komponenti prosledio neki podatak, potrebno je za tu komponentu definisati *Input Property Binding*
- U jednoj web stranici moguće je ugnezditi komponente



Kompozicija `PuzzleComponent` u listu

- *Template* liste zagonetki treba samo da prođe kroz sve elemente niza

```
<app-puzzle *ngFor="let p of puzzles" [puzzle]="p"></app-puzzle>
```

- *Template* `PuzzleComponent` treba da prikaže sadržaj objekta `Puzzle`

```
<mat-card>
  <mat-card-header>
    <mat-card-title><h1>{{ puzzle.question }}</h1></mat-card-title>
  </mat-card-header>
  <mat-card-content>
    <p [hidden]="puzzle.hide">{{ puzzle.solution }}</p>
  </mat-card-content>
  <button mat-raised-button color="primary" (click)="puzzle.toggle()">Prikaži</button>
</mat-card>
<br>
```

- Problem predstavlja nemogućnost postavljanja vrednosti obeležju `puzzle` komponente `PuzzleComponent` iz komponente `PuzzleListComponent`

Input Property Binding za komponentu

- Obeležje komponente koje želimo da izložimo kao deo javnog interfejsa označavamo anotacijom `@Input`

```
import { Component, Input } from "@angular/core";
...

@Component({
...

export class PuzzleComponent {
   @Input() puzzle: Puzzle;
}
```

Dodavanje nove zagonetke u listu

- Nova komponenta treba da obezbedi
  - unos pitanja
  - unos rešenja
  - dugme za kreiranje nove zagonetke
- Klik na dugme za dodavanje treba da generiše događaj
- Komponenta u koju treba da se doda nova zagonetka treba da na pojavu događaja odgovori dodavanjem nove zagonete u listu

Dodavanje forme u komponentu liste

- Dobro mesto za smeštanje forme za dodavanje nove zagonete je komponenta koja prikazuje listu zagonetki
- Selektor za novu komponentu `PuzzleFormComponent` je `<app-puzzle-form>`
- *Template* `PuzzleListComponent` bi izgledao ovako:

```
<app-puzzle-form (puzzleCreated)="addPuzzle($event)"></app-puzzle-form>
<app-puzzle *ngFor="let p of puzzles" [puzzle]="p"></app-puzzle>
```

- `PuzzleFormComponent` će emitovati događaj pod nazivom `puzzleCreated` kad god korisnik klikne na dugme za dodavanje zagonetke u listu
- Za spoljnu komponentu ugneždena komponenta je “crna kutija”

Fajl `puzzleform.component.ts` (1)

```
import ...
@Component({
   selector: 'app-puzzle-form',
   template: `<mat-card>
               <mat-card-title>Nova zagonetka</mat-card-title>
               <mat-card-content>
                 <mat-form-field class="example-full-width">
                   <input matInput placeholder="Upišite pitanje">
                 </mat-form-field>
                 <mat-form-field class="example-full-width">
                   <input matInput placeholder="Upišite rešenje">
                 </mat-form-field>
               </mat-card-content>
               <button mat-raised-button color="primary">Dodaj u listu</button>
              </mat-card>`
})
export class PuzzleFormComponent {}
```

```
import { BrowserAnimationsModule } from
@angular/platform-browser/animations';
...
@NgModule({
...
 imports: [
   BrowserModule,
   BrowserAnimationsModule,
...
```

Emitovanje događaja komponente

- Forma treba da emituje događaj `puzzleCreated` i pri tome generiše novi objekat tipa `Puzzle`
- Događaj će se emitovati nakon što se klikne na dugme
- Zasad ćemo emitovati *hardcoded* `Puzzle` objekat

```
import {Output, EventEmmiter} from '@angular/core';
…
<button mat-raised-button color="primary" (click)="createPuzzle()">Dodaj u listu</button>
…
export class PuzzleFormComponent {
  @Output() puzzleCreated = new EventEmitter<Puzzle>();

  createPuzzle() {
    this.puzzleCreated.emit(new Puzzle("pitanje", "odgovor")); // hardcoded
  }
}
```

Modifikacija `PuzzleListComponent`

- Saglasno kodu u *template*-u, kada se desi događaj `puzzleCreated` komponenta treba da izvrši metodu `addPuzzle`
- Parametar `$event` će u našem slučaju biti `Puzzle` objekat

```
…
export class PuzzleListComponent {
  puzzles: Puzzle[];

  constructor() {
    …
  }

  addPuzzle(puzzle) {
    this.puzzles.unshift(puzzle);
  }
}
```

```
…
<app-puzzle-form (puzzleCreated)="addPuzzle($event)"></app-puzzle-form>
…
```

Referentne promenljive u *template*-u

- Jedan od načina da se pristupi vrednostima tekst polja je da se koriste referentne promenljive

```
…
<input matInput placeholder="Upišite pitanje" #question>
…
<input matInput placeholder="Upišite rešenje" #solution>
…
<button … (click)="createPuzzle(question.value, solution.value)">Dodaj u listu</button>
```

- Sada metoda `createPuzzle` može da prihvati parametre i da se umesto *hardcoded* kreira objekat `Puzzle` korišćenjem odgovarajućih parametara

```
createPuzzle(q: string, s: string) {
  this.puzzleCreated.emit(new Puzzle(q, s));
}
```

Rezime

- *Input Property Binding* omogućava postavljanje vrednosti obeležja komponenti
- *Output Event Binding* omogućava generisanje događaja koje mogu da slušaju druge komponente
- Klase modela domena sadrže podatke kojima se operiše u aplikaciji
- Komunikacija između ugneždenih komponenti može da se vrši korišćenjem prethodno opisanih mehanizama
- Referentne promenljive mogu da se iskoriste za pristup vrednostima atributa HTML elementa