

Reaktive forme

Reaktivne forme

- U glavni modul aplikacije importuje se `ReactiveFormsModule`
- Kreira se model forme u klasi komponente korišćenjem klasa `FormGroup` i `FormControl` (kreira se programski korišćenjem typescript koda)
- Forma se sinhronizuje sa DOM

Importovanje modula **ReactiveFormsModule**

```
import { ReactiveFormsModule } from "@angular/forms";

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

Klasa komponente

```
@Component({
  selector: 'app-primer1',
  templateUrl: './primer1.component.html',
  styleUrls: ['./primer1.component.css']
})
export class Primer1Component {

  title = 'Reaktivne forme';

  form1 = new FormGroup({
    ime: new FormControl(''),
    prezime: new FormControl('')
  });
}
```

Šablon glavne komponente

```
<div class="container">
  <div class="jumbotron">
    <h5>Reaktivne forme</h5>
  </div>

  <div class="row">
    <div class="col-md-6">

      <form>

        <div class="form-group">
          <label for="ime">Ime</label>
          <input type="text" name="ime" id="ime" class="form-control">
        </div>

        <div class="form-group">
          <label for="prezime">Prezime</label>
          <input type="text" name="prezime" id="prezime" class="form-control">
        </div>

        <div class="form-group">
          <button class="btn btn-primary" type="submit">Posalji</button>
        </div>

      </form>

    </div>
  </div>
</div>
```

Povezivanje šablona sa modelom

```
<form [formGroup]="form1">
```

sinhronizacija HTML-a sa dinamički kreiranom formom

one-way binding, direktiva formGroup

Povezivanje polja forme sa modelom

```
<input type="text" name="ime" id="ime" class="form-control"  
formControlName="ime">
```

```
<input type="text" name="prezime" id="prezime" class="form-control"  
formControlName="prezime">
```

Sabmitovanje forme

```
<form [formGroup]="form1" (ngSubmit)="onSubmit()">
```

Prihvatanje podataka u klasi komponente

```
onSubmit() {  
    console.log(this.form1.value);  
}
```

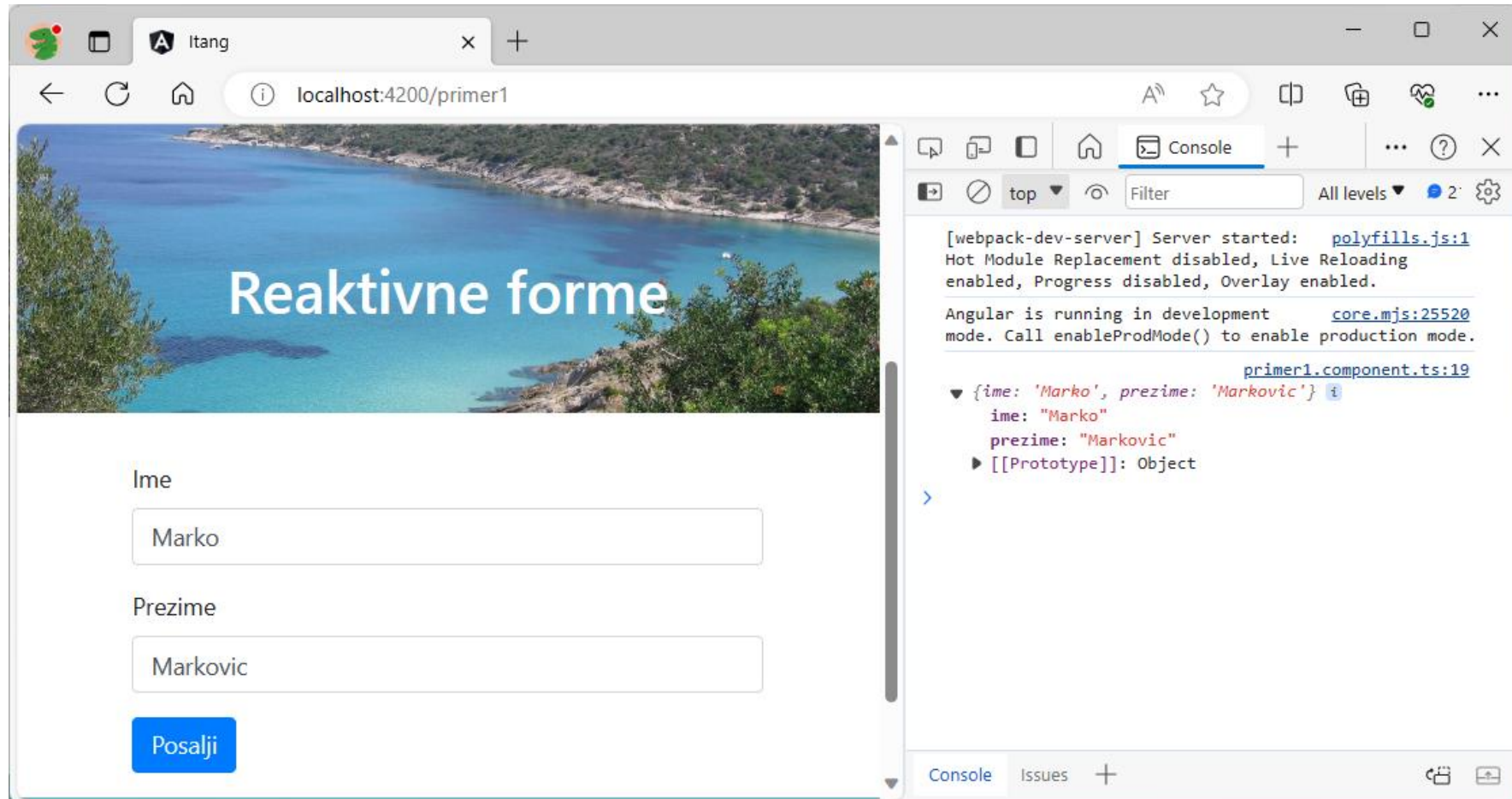

Šablon komponente

```
<div class="col-6">
  <form [formGroup]="form1" (ngSubmit)="onSubmit()">
    <div class="form-group">
      <label for="ime">Ime</label>
      <input type="text" name="ime" id="ime" class="form-control" formControlName="ime">
    </div>

    <div class="form-group">
      <label for="prezime">Prezime</label>
      <input type="text" name="prezime" id="prezime" class="form-control"
formControlName="prezime">
    </div>

    <div class="form-group">
      <button class="btn btn-primary" type="submit">Posalji</button>
    </div>
  </form>
</div>
```

Rezultat sabmitovanja forme



The screenshot shows a web browser window with the URL `localhost:4200/primer1`. The page displays a scenic background image of a lake and the title "Reaktivne forme". Below the title, there is a form with two input fields: "Ime" (Name) containing "Marko" and "Prezime" (Surname) containing "Markovic". A blue "Posalji" (Send) button is located below the form.

The browser's developer console is open, showing the following log entries:

```
[webpack-dev-server] Server started: polyfills.js:1  
Hot Module Replacement disabled, Live Reloading  
enabled, Progress disabled, Overlay enabled.  
  
Angular is running in development core.mjs:25520  
mode. Call enableProdMode() to enable production mode.  
  
primer1.component.ts:19  
▼ {ime: 'Marko', prezime: 'Markovic'}  
  ime: "Marko"  
  prezime: "Markovic"  
  ► [[Prototype]]: Object
```

Validacija reaktivne forme

```
import { FormGroup, FormControl, Validators } from '@angular/forms';
```

```
@Component({
  selector: 'app-primer2',
  templateUrl: './primer2.component.html',
  styleUrls: ['./primer2.component.css']
})
export class Primer2Component {

  title = 'Reaktivne forme';

  form1 = new FormGroup({
    ime: new FormControl('', Validators.required),
    prezime: new FormControl('', Validators.required)
  });

  onSubmit(){
    console.log(this.form1.value);
  }
}
```

Prikaz validacionih grešaka – polje ime

```
<div class="form-group">
  <label for="ime">Ime</label>
  <input type="text" id="ime" class="form-control" formControlName="ime">
  <div *ngIf="form1.get('ime')?.invalid && (form1.get('ime')?.dirty ||
form1.get('ime')?.touched)" class="alert alert-danger">
    Unesi ime
  </div>
</div>
```

Prikaz validacionih grešaka – polje prezime

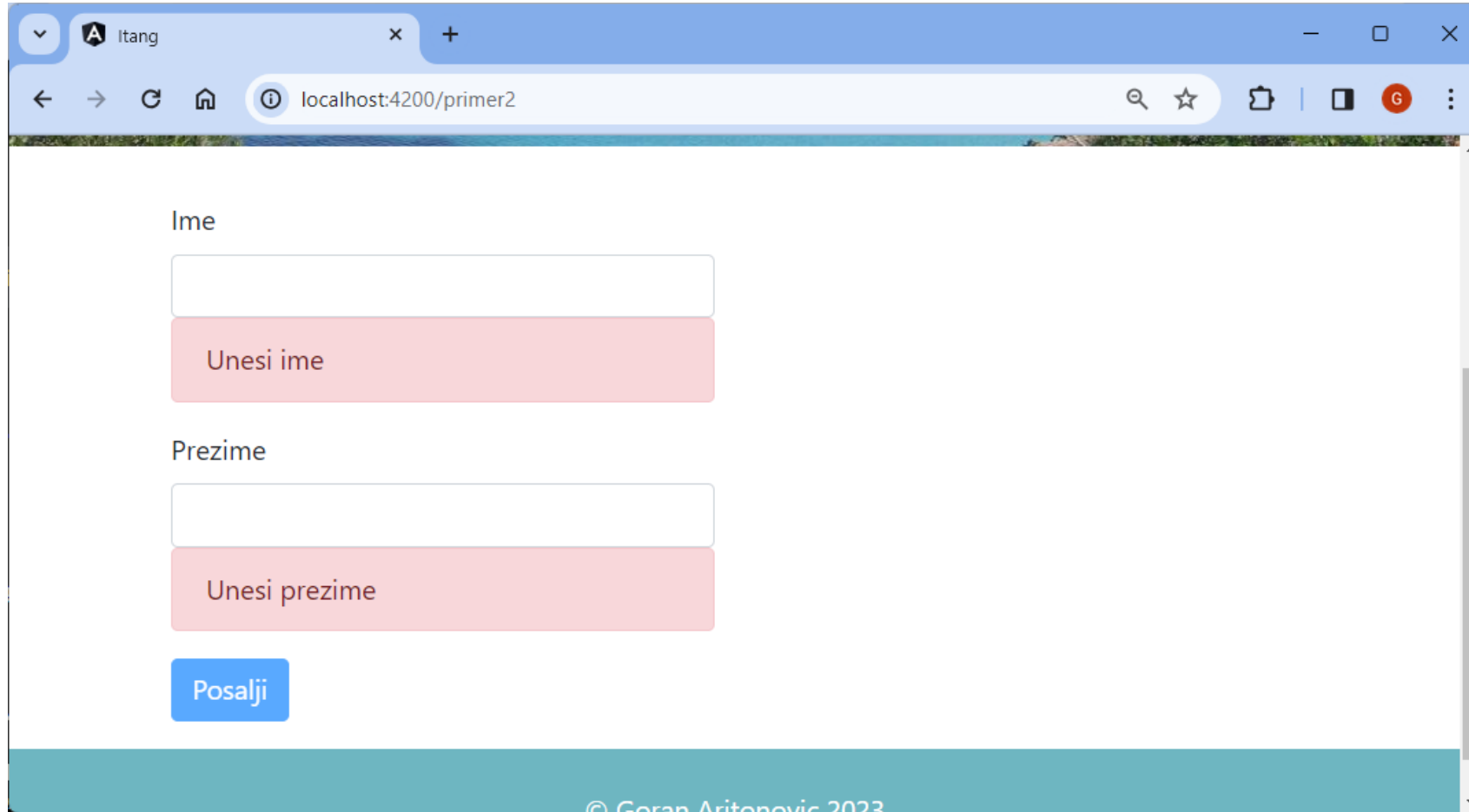
```
<div class="form-group">
  <label for="prezime">Prezime</label>
  <input type="text" id="prezime" class="form-control" formControlName="prezime">

  <div *ngIf="form1.get('prezime')?.invalid && (form1.get('prezime')?.dirty ||
form1.get('prezime')?.touched)" class="alert alert-danger">
    Unesi prezime
  </div>
</div>
```

Zabrana submitovanja ne validnih podataka

```
<button class="btn btn-primary" [disabled]="form1.invalid" type="submit">  
Posalji  
</button>
```

Validacione greške



A screenshot of a web browser window showing a form with two input fields, 'Ime' and 'Prezime', both of which are empty. Below each input field is a red error message: 'Unesi ime' and 'Unesi prezime' respectively. A blue 'Posalji' button is located below the 'Prezime' field. The browser's address bar shows 'localhost:4200/primer2'. The browser's title bar shows 'Itang'. The footer of the page contains the text '© Goran Artonovic 2023'.

Ime

Unesi ime

Prezime

Unesi prezime

Posalji

© Goran Artonovic 2023

Getteri za pristup kontrolama forme

```
get ime() { return this.form1.controls.ime; }  
// get ime() { return this.form1.get('ime'); }  
  
get prezime() { return this.form1.controls.prezime; }  
// get prezime() { return this.form1.get('prezime'); }
```


Validacija upotrebom getera

```
<div class="form-group">
  <label for="ime">Ime</label>
  <input type="text" id="ime" class="form-control" formControlName="ime">

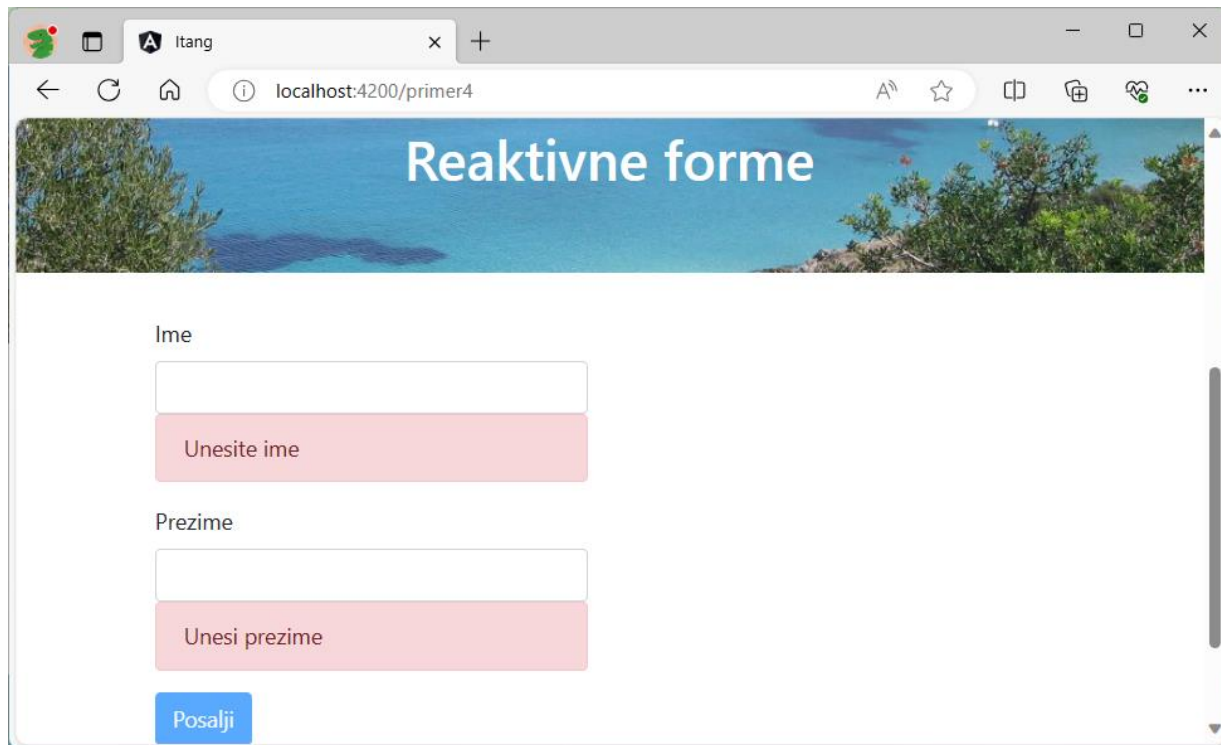
  <div *ngIf="ime.invalid && (ime.dirty || ime.touched)" class="alert alert-danger">
    Unesi ime
  </div>
</div>
```

Definisanje više validatora za polje forme

```
form1 = new FormGroup({
  ime: new FormControl('', [Validators.required, Validators.minLength(3)]),
  prezime: new FormControl('', Validators.required)
});
```

```
<div *ngIf="ime.invalid && (ime.dirty || ime.touched)" class="alert alert-danger">
  <div *ngIf="ime.hasError('required')">
    Unesite ime
  </div>
  <div *ngIf="ime.hasError('minlength')">
    Najmanje 3 karaktera
  </div>
</div>
```

Validacija forme



Itang localhost:4200/primer4

Reaktivne forme

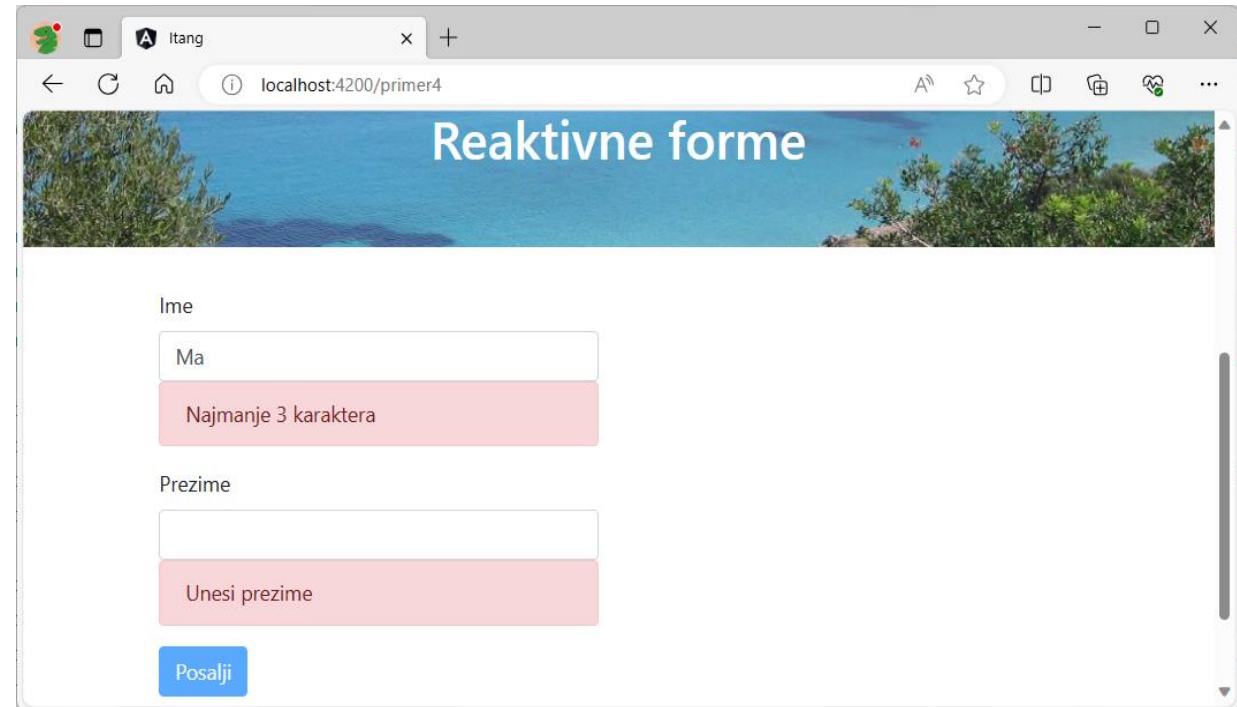
Ime

Unesite ime

Prezime

Unesi prezime

Posalji



Itang localhost:4200/primer4

Reaktivne forme

Ime

Najmanje 3 karaktera

Prezime

Unesi prezime

Posalji

FormBuilder

- Koristi se za kreiranje reaktivnih formi uz minimalnu količinu koda
- Metoda `group()` kreira instancu `FormGroup`
- Metoda `control()` kreira instancu `FormControl`
- Ubacuje se u konstruktor komponente

```
import { FormGroup, FormBuilder, Validators } from '@angular/forms';
```

FormBuilder

```
export class Primer5Component {  
  
  title = 'Form builder';  
  form1: FormGroup;  
  
  constructor(private fb: FormBuilder) {  
    this.form1 = fb.group({  
      ime: ['', [Validators.required, Validators.maxLength(10)]],  
      prezime: ['', Validators.required]  
    });  
  }  
  
  get ime() { return this.form1.controls['ime']; }  
  get prezime() { return this.form1.controls['prezime']; }  
  
  onSubmit() {  
    console.log(this.form1.value);  
  }  
  
}
```

Šablon komponente

```
<div class="form-group">
  <label for="ime">Ime</label>
  <input type="text" id="ime" class="form-control" formControlName="ime">
  <div *ngIf="ime.invalid && (ime.dirty || ime.touched)" class="alert alert-danger">
    <div *ngIf="ime.hasError('required')">
      Unesite ime
    </div>
    <div *ngIf="ime.hasError('maxlength')">
      Najviše karaktera
    </div>
  </div>
</div>
```

```
<div class="form-group">
  <button class="btn btn-primary" [disabled]="form1.invalid" type="submit">Posalji</button>
</div>
```

Reaktivna forma primer -1

```
export class Student {  
  constructor(public ime: string, public prezime: string, public pol: string,  
  public smer: string ) {  
  }  
}
```

Reaktivna forma primer -2

```
export class StudentComponent {  
  
  title = 'Rektivne Forme Vezba';  
  
  student: Student = new Student('', '', 'muski', 'Informatika');  
  smerovi = ['Informatika', 'Marketing', 'Menadzment'];  
  
  studentForm = new FormGroup({  
    ime: new FormControl('', [Validators.required]),  
    prezime: new FormControl('', [Validators.required]),  
    pol: new FormControl('muski', [Validators.required]),  
    smer: new FormControl('Informatika', [Validators.required])  
  });
```


Geteri za pristup kontrolama

```
get ime() { return this.studentForm.controls.ime; }  
  get prezime() { return this.studentForm.controls.prezime; }  
  get pol() { return this.studentForm.controls.pol; }  
  get smer() { return this.studentForm.controls.smer; }
```

Metode resetuj() i setDefault()

```
resetuj() {  
    this.student = new Student('', '', 'muski', 'Informatika');  
    this.studentForm.reset(this.student);  
}  
  
setDefault() {  
    this.studentForm.patchValue({ ime: 'Marko', prezime: 'Markovic', pol: 'muski',  
        smer: 'Informatika' });  
}
```

```
setDefault() {  
    this.studentForm.setValue({ ime: 'Marko', prezime: 'Markovic', pol: 'muski',  
        smer: 'Informatika' });  
}
```

setValue() zahteva dodeljivanje vrednosti svim kontrolama, patchValue() dozvoljava izbor kontrola

Metoda onSubmit()

```
onSubmit() {  
  this.student.ime = this.ime?.value || '';  
  this.student.prezime = this.prezime?.value || '';  
  this.student.pol = this.pol?.value || '';  
  this.student.smer = this.smer?.value || '';  
  
  console.log('Ime : ' + this.student.ime);  
  console.log('Prezime : ' + this.student.prezime);  
  console.log('Pol: ' + this.student.pol);  
  console.log('Smer: ' + this.student.smer);  
  
  this.resetuj();  
}
```

```
<form [formGroup]="studentForm" (ngSubmit)="onSubmit()">

  <div class="form-grup">
    <label for="ime">Ime:</label>
    <input type="text" id="ime" name="ime" formControlName="ime"
class="form-control">
    <div *ngIf="ime.invalid && (ime.dirty || ime.touched)">
      Unesite ime.
    </div>
  </div>

  <div class="form-grup">
    <label for="ime">Prezime:</label>
    <input type="text" id="prezime" name="prezime" formControlName="prezime"
class="form-control">
    <div *ngIf="prezime.invalid && (prezime.dirty || prezime.touched)">
      Unesite prezime.
    </div>
  </div>
</form>
```

```
<div class="form-group">

  <label>Odaberi pol:</label>
  <div class="form-check">
    <input type="radio" name="pol" value="muski" id="radio1"
      class="form-check-input" formControlName="pol">
    <label for="radio1" class="form-check-label">Muski</label>
  </div>

  <div class="form-check">
    <input type="radio" name="pol" value="zenski" id="radio2"
      class="form-check-input" formControlName="pol">
    <label for="radio2" class="form-check-label">Zenski</label>
  </div>
</div>

<div class="form-group">
  <label for="smer">Odaberi smer</label>
  <select name="smer" id="smer" formControlName="smer" class="form-control">
    <option *ngFor="let smer of smerovi" [value]="smer">
      {{smer}}
    </option>
  </select>
</div>
```

```
<div class="form-group">
  <button [disabled]="studentForm.invalid" class="btn btn-primary">Posalji</button>

  <button type="button" (click)="setDefault()"
    class="btn btn-primary">Default</button>

  <button type="button" (click)="resetuj()"
    class="btn btn-primary">Reset</button>
</div>
```

Pitanje 1

Povezivanje postojeće instance klase FormGroup sa elementom form kod reaktivnih formi vrši se korišćenjem direktive:

- a. `formGroup`
- b. `formControlName`
- c. `bindForm`

Odgovor: a

Pitanje 2

Sinhronizacija instance klase FormControl sa DOM elementom kod reaktivnih formi vrši se posredstvom direktive:

- a. control
- b. controlBind
- c. FormControlName

Odgovor: c

Pitanje 3

Kod reaktivnih formi :

- a. Instance klase FormGroup i FormControl se kreiraju na osnovu šablona komponente gde se nalazi forma
- b. Kreira se model forme u kome se instanciraju klase FormGroup i FormControl pa se vrši sinhronizacija modela sa formom
- c. Forma se kreira instanciranjem klasi ReactiveForm

Odgovor: b

Pitanje 4

Kod reaktivnih formi klasa koja omogućava brzo kreiranje modela forme naziva se :

- a. FormGenerator
- b. CreateForm
- c. FormBuilder

Odgovor: c