AngularJS forme i direktive

Validacija podataka i pravljenje proizvoljnih direktiva i validatora

HTML forma

- Forma predstavlja skup polja (`input`, `select`, `textarea`) putem kojih korisnik može da unosi podatke
- HTML 5 forma pruža pristojan sistem validacije ispravnosti podataka
- Validacija podataka na klijentu pozitivno utiče na iskustvo korisnika
- Validacija podataka na klijentu se lako može zaobići te je bitnije uraditi iste provere na serveru
- Angular nudi bolji sistem validacije od podrazumevanog
- Potrebno je u tag forme dodati `novalidate` atribut kako bi se isključilo podrazumevano ponašanje i izbegao konflikt
- Potrebno je izbaciti `action` atribut iz forme, jer on izaziva osvežavanje stranice pri izvršavanju `submit` operacije nad formom primer 1

Angular forma

- Angular nudi API nad `form` elementom i unutrašnjim poljima za:
  - Određivanje stanja forme ili polja
  - Validacija unesenih podataka
- Da bi se forma i njena polja pojavila u kontroleru potrebno je da popunimo `name` atribut za formu i za svako željeno polje
- Napomena:
  - Ako koristimo `$scope` objekat direktno u kontroleru naziv forme se zadaje normalno
  - Ako koristimo `controllerAs` sintaksu naziv forme treba zadati u sledećem formatu: `<ime kontrolera>.<ime forme>`
  - I kod `controllerAs` i kod `$scope` sintakse naziv polja se zadaje normalno
  - Polju pristupamo u kontroleru: `<ime forme>.<ime polja>`

Stanja forme i polja

- Moguća stanja forme i polja forme su:
  - Da li su polja u formi dirana (`$touched`)
  - Da li polja forme nisu dodirnuta (`$untouched`)
  - Da li su podaci u poljima forme menjani (`$dirty`)
  - Da li su podaci u poljima forme neizmenjeni (`$pristine`)
  - Da li ima polja sa nevalidnim podacima (`$invalid`)
  - Da li sva polja sadrže validne podatke (`$valid`)
- U zavisnosti od stanja u kom se nalazi, angular dodaje odgovarajuće CSS klase formama i poljima formi
- Klase imaju naziv `ng-<stanje>` (npr. `ng-touched`, `ng-invalid`)

primer 2

Validacija unesenih podataka

- “Never trust user input”
- Input polje ima ugrađene atribute koji se koriste za različite validacije:
  - `required/ng-required` - označava da li je polje obavezno
  - `ng-minlength=”broj”` - minimalan broj karaktera
  - `ng-maxlength=”broj”` - maksimalan broj karaktera
  - `ng-pattern=”string”` - provera da li unos odgovara regularnom izrazu (https://sr.wikipedia.org/wiki/Regularni_izraz)
- Svako imenovano polje sadrži `$error` objekat u kom su navedene sve greške u formatu `“<ime greške>”:true`

Prikaz validacionih grešaka

- `ngMessages` modul nam omogućava da iskoristimo `$error` objekat da na čist način ispišemo greške
- Struktura `ng-messages` direktive:

```
<div ng-messages="<controllerAs ime>.<ime forme>.<ime atributa>.$error">
    <p ng-message="<ime greške>">Tekst greške</p>
    …
</div>
```

- Napomena: `ngMessages` je modul koji nije deo osnovne angular biblioteke što znači da je potrebno ubaciti zavisnost u `bower.json`, instalirati paket i uključiti ga u listu zavisnosti aplikacije (`angular.module('app', ['ngMessages'])`)
- `ng-messages-include` direktiva omogućava još čistije, ali generičko rešenje, kao što ćemo videti u primeru primer 3

Prikaz validacionih grešaka

- Kad prikazati validacione greške:
  - Čim se otvori forma
  - Kad korisnik unese podatke u polje
  - Kad korisnik napusti polje
  - Kad korisnik stisne “submit”
- `ng-model-options` je atribut input polja koji omogućava proizvoljno definisanje kada će se model ažurirati (kada napustimo polje, kada nismo ništa novo ukucali 1 sekund i slično)
- `ng-model-options` prihvata objekat kao vrednost koji ima nekoliko atributa, a za nas su relevantni `updateOn`, koji definiše događaje kada će se model ažurirati i `debounce`, koji definiše vreme u milisekundama koje je potrebno da prođe da se izvrši ažuriranje

primer 4

ng-model-options

- ng-model-options=
  - `"{ updateOn: 'blur mouseover'}"`
    - model se ažurira kada se desi blur ili mouseover događaj (više događaja se odvaja sa razmakom)
  - `"{ updateOn: 'default blur', debounce: {'default': 500, 'blur':0} }"`
    - debounce može da prihvati objekat umesto broj, i kad je to slučaj objekat treba da sadrži `'<ime događaja>':<broj milisekundi>` parove
    - u kombinaciji sa updateOn možemo definisati različite intervale ažuriranja (u primeru će se model ažurirati odmah ukoliko se polje napusti ili nakon 0.5 sekundi)
- Korisno je koristiti ng-model-options kada treba da se izvrši neka "skupa" operacija prilikom promene modela

primer 5

$scope.$watch

- Koristimo kada želimo da odgovorimo na promenu vrednosti nekog objekta, atributa ili promenljive
- `$scope.$watch(watchExpression, listener[, objectEquality])`
  - `watchExpression` - koji objekat, atribut ili promenljivu posmatramo
  - `listener` - funkcija koju izvršavamo kao odgovor na promenu
  - `objectEquality` - boolean polje za označavanje da li želimo dubinsku posmatranje promene
- Kad je `objectEquality == true` posmatra se promena objekta i svih njegovih atributa (koristi angular.equals, a ne samo poređenje referenci)
- Svaka dodatna `$scope.$watch` funkcija usporava (malo) aplikaciju, pogotovo ako je `objectEquality=true`

$scope.$watch

- Primer:

```
$scope.$watch('rad.pretraga', function(newVal, oldVal) {
    if(newVal !== oldVal) {
        console.log(newVal);
    }
}
```

- Posmatramo `rad.pretraga` atribut i svaki put kada se promeni vrednost ispisujemo na konzolu novu vrednost
- `listener` funkcija prihvata dva parametra, novu vrednost `watchExpression`-a koji se posmatra i staru vrednost
- `listener` se poziva svaki `$digest` ciklus ako se vrednost promenila i jednom na početku (zbog toga nam treba `if(newVal !== oldVal)` kako bi izbegli prvo pozivanje)

Pravljenje proizvoljnih validatora

- Ponekad nam ugrađeni validatori nisu dovoljni:
  - Želimo da prilikom registracije na forum odmah dobijemo podatak da li postoji korisničko ime koje smo odabrali
  - Želimo da imamo email validator koji prihvata samo emailove sa gmail.com domena
  - U polje koje prihvata numerička polja prihvatamo samo brojeve između 20 i 80
- Potrebno je napraviti novu direktivu koju ćemo dodati kao atribut input elementu, tako da nam radi cela infrastruktura za greške i stanje forme

Direktive

- Javaskript objekti koji su definisani putem `directive` funkcije `module` API-a
- Proširuju postojeće HTML elemente (atributi) ili dodaju nove
- Metoda kojom se pravi direktiva:

```
angular.module(<ime modula>).directive(<ime direktive>, <funkcija direktive>)
```

- Ime direktive se u javaskript kodu piše kamiljom notacijom (‘imeDirektive’), da bi se u HTML-u postavljala crta između reči (‘ime-direktive’)
- Preporučeno je dati svoj prefiks svakoj direktivi da bi se izbegla kolizija imena
- Funkcija direktive je fabrička funkcija koja vraća objekat koji sadrži konfiguracije direktive, odnosno koji definiše direktivu
- Angular aplikacije treba da jedino putem direktiva radi sa DOM-om

Direktive

```
angular.module('app').directive('myDirective', function(<injectables>) {
    var directiveDefinitionObject = {
        priority: 0,
        template: '<div>My directive</div>',
        templateUrl: 'directive.html', //koristiti ili template ili templateUrl
        restrict: 'E',
        scope: true,
        compile: function compile(templateElement, templateAttrs, transclude) {...}
        link: function link(scope, element, attrs) {...} //koristiti ili link ili compile
        controller: MyDirectiveController, //podržava injekciju
        controllerAs: 'md',
        bindToController: true,
        require: ['ngModel'],
        transclude: false,
        replace: false
    };
    return directiveDefinitionObject;
});
```

Direktive - template, templateUrl, restrict

- Atributi konfiguracionog objekta za definisanje direktive:
  - template
    - definiše HTML kod koji će zameniti direktivu
  - templateUrl
    - link ka .html dokumentu sa HTML kodom za zamenu direktive
    - u opštem slučaju bolje od template atributa (osim ako je HTML kod baš kratak)
  - restrict
    - definiše da li je direktiva HTML element (E) ili atribut elementa (A)
    - ‘E’ - ako pravimo komponentu koja upravlja svojim template-om
    - ‘A’ - kada dodajemo funkcionalnost postojećem elementu

primer 6

Direktive - transclude

- Atributi konfiguracionog objekta za definisanje direktive:
  - transclude
    - obično ide u paru sa ng-transclude direktivom
    - koristi se kad želimo da ubacimo HTML u našu direktivu
    - postavimo u objektu koji definiše direktivu transclude: true
    - definišemo template na sledeći način:

```
<div> <!-- šablon mojaDirektiva direktive -->
     <p>Directive template</p>
     <ng-transclude></ng-transclude>
</div>
<!-- zatim koristimo direktivu i stavimo joj unutrašnji sadržaj-->
<moja-direktiva><p>Some text</p></moja-direktiva>
<!-- krajnji rezultat je ovo -->
<div><p>Directive template</p><p>Some text</p></div>
```

Direktive - scope

- Atributi konfiguracionog objekta za definisanje direktive:
  - scope
    - Ako ne postoji u konfiguracionom objektu (ili mu je vrednost false) onda direktiva deli scope sa scope-om komponente u kojoj se nalazi (i ukoliko dodaje objekat na svoj scope tom objektu se može pristupiti i van direktive)
    - Ako ima vrednost true onda se vezuje novi scope za direktivu koji nasleđuje scope od komponente u kojoj se direktiva nalazi ($parent)
    - Ako ima vrednost javaskript objekta ({ ... }) stvara se novi izolovan scope koji ne deli podatke sa kontrolerom u kom se direktiva nalazi, već mu se podaci prosleđuju putem tog javaskript objekta
    - Kad koristimo controllerAs bitno nam je samo da li je izolovan scope ili ne

primer 7

Direktive - izolovan scope

- Koristiti izolovan scope, pogotovo kod direktiva koje će biti ponovo iskoristive
- Nad jednim HTML elementom ne mogu dva izolovana scope-a (više direktiva)
- Moguće je proslediti podatke izolovanom scope-u putem atributa koji idu uz direktivu, na primer ako imamo u HTMLu atribut inputName u direktivu stavljamo:
  - ```
    scope: {
        name: "=inputName" //U scope-u direktive će postojati name polje kome
    }                      //će se proslediti vrednost inputName atributa
    ```
  - ```
    scope: {
        inputName: "=" //Ako je ime isto ne mora se navoditi
    }
    ```
- Pored '=', koji je dvosmeran (ako promenimo u izolovanom scope-u, menjamo i originalnu vrednost), koristimo '@' ako želimo da prosledimo vrednost kao string, i '&' ako želimo spoljnu funkciju da izvršimo u kontekstu (scope) direktive primer 8

Direktive - compile, link

- Atributi konfiguracionog objekta za definisanje direktive:
  - compile
    - funkcija za manipulaciju DOM-a HTML **šablona**
    - kada želimo da izvršimo promene koje važe za sve instance direktive
    - ako želimo i link funkciju moramo je definisati i vratiti iz compile funkcije (ako postoji compile atribut, link atribut se ignoriše)
  - link
    - funkcija za manipulaciju DOM-a pojedinačne instance direktive
    - za razliku od compile funkcije link ima pristup scope-u
    - koristi se i za prijavu listenera za događaje (npr. `$scope.$watch`)
    - function(scope, element, attrs, ctrls)
    - moguće je koristiti injektovane servise funkcije direktive

primer 9

Direktive - controller, controllerAs

- Atributi konfiguracionog objekta za definisanje direktive:
  - controller
    - kontroler HTML šablona direktive
    - prihvata injektovanje (npr. `function DirController($scope, $timeout)`)
    - mora se koristiti kada druga direktiva treba da interaguje sa ovom direktivom (kontroler izlaže metode koje će druge direktive pozivati)
    - ako želimo da izložimo metode koristimo controller, u suprotnom dovoljna nam je link funkcija za sve što nam treba
  - controllerAs
    - ime koje će kontroler imati po poznatoj controllerAs sintaksi

Direktive - bindToController, require

- Atributi konfiguracionog objekta za definisanje direktive:
  - bindToController
    - kod controllerAs sintakse atribute definisane u scope objektu prebacujemo u ovaj objekat kako bi se vezali za kontroler, a ne scope
    - ako nema takvih atributa, a koristimo controllerAs sintaksu treba staviti vrednost polja na true
  - require
    - označavamo direktive čijim kontrolerima želimo da pristupimo
    - kontroleri direktiva koje navedemo se nalaze u 4. argumentu link funkcije
    - traži kontroler na datom elementu (atributi elementa)
    - ako ima prefiks ^ onda traži i na roditeljskim čvorovima
    - ako ima prefiks ? onda stavlja null ako ne nađe (inače greška) primer 10

Pravljenje proizvoljnih direktiva

- Uzećemo tabelu radnika koju smo videli i dodati formu za dodavanje radnika
- Želimo da prilikom upisa podataka o JMBG-u sistem odmah proveri da li postoji radnik sa tim JMBG-om
- Potrebno je napraviti novu direktivu koju ćemo dodati kao atribut input elementu, tako da nam radi cela infrastruktura za greške i stanje polja forme
- Za pristup date infrastrukture moramo uključiti kontroler ng-model direktive (`require`)
- Koristićemo link funkciju da postavimo $watch na atribut koji posmatramo (JMBG) i da reagujemo na njegovu promenu (tako što ćemo poslati HTTP zahtev MongoLab-u da proverimo da li postoji radnik sa datim JMBG-om
- Treba dodati indikator da korisnik zna da se provera izvršava