ANGULAR 5

Peter Gurský, peter.gursky@upjs.sk

Validátory

- Angular má niekoľko vlastných validátorov na kontrolu formulárov, alebo môžeme dorobiť vlastné
- Neparametrické validátory sú funkcie (ValidatorFn), ktoré majú
  - na vstupe kontrolovaný formulárový element, skupinu formulárových elementov alebo celý formulár
  - na výstupe
    - null ak je validátor úspešný
    - objekt s nájdenými chybami, napr. {'length': 'small string', 'format': 'wrong character', }
- Parametrické validátory sú funkcie druhého rádu
  - na vstupe majú jeden alebo viac hodnôt na ich nakonfigurovanie (napr. Validator.minLength(5))
  - na výstupe má ValidatorFn

Vstavané validátory

- https://angular.io/api/forms/Validators
  - parametrické
    - min, max, minLength, maxLength, pattern, compose, composeAsync
  - neparametrické
    - required, requiredTrue, email, nullValidator

Template-driven formuláre

- založené na
  - [(ngModel)] na mapovanie hodnôt
  - povinný parameter name
- model formulára, ktorý vyhodnocuje validitu, je dostupný **len v šablóne** cez premennú elementu
  - ngForm v elemente form
  - ngModel vo vstupných formulárových komponentoch
- model formulára nie je dostupný z kódu komponentu
- ak chceme použiť vo formulári validátory, vieme ich dodať formulárovým elementom iba cez direktívy, t.j. atribúty elementov
  - https://angular.io/guide/form-validation#adding-to-template-driven-forms

Vloženie validátora pre TD-formuláre

- direktívy pre zabudované validátory:
  - https://angular.io/api/forms#directives
    - CheckboxRequiredValidator, PatternValidator,,...
- Napr. EmailValidator je direktíva definovaná pravidlami:
  - [email][formControlName] ← pre Reakt. formuláre
  - [email][formControl] ← pre Reakt. formuláre
  - [email][ngModel] ← pre TD-formuláre
- takže ho aktivujeme tak, že napíšeme v šablóne
  - <input type="email" name="email" **ngModel email**>
  - <input type="email" name="email" [(**ngModel**)]="user.email" **email**>

Použitie validácie

- validita sa dá zobraziť používateľovi pomocou css, ak je prítomný validátor nastaví sa pre daný formulárový element jedna z tried
  - ng-touched / ng-untouched – element bol navštívený / nebol
  - ng-dirty / **ng-pristine** – hodnota je zmenená / nie je
  - **ng-valid** / **ng-invalid** - hodnota je správna / nie je
    - ak element má atribút required, hodnota musí byť vyplnená
    - okrem required máme aj: minlength, maxlength, pattern
    - …zložitejšie kontroly viď. Validator v ngModel-i
- Pozrime si cez inšpektora

Validita cez Angular Material

- nevalidný input je označený červenou farbou
- mat-form-field môže mať v sebe okrem input elementu aj
  - <mat-hint> - zobrazenie textu pod input elementom
  - <mat-label> - pomenovanie vstupu
    - ak je input prázdny zobrazuje sa namiesto hodnoty
    - ak je input vyplnený zobrazuje sa nad hodnotou
  - <mat-error> - zobrazenie červenej chybovaj hlášky pod input elementom
    - použijeme @if, aby sa chyba zobrazila, iba ak je input nevalidný
    - ak sa zobrazuje mat-error, nezobrazuje sa mat-hint

Signálne formuláre

- Dve základné zložky v triede komponentu
  - model = signal<Entita>({niečo: '', ...}) - signál s dátami
  - myForm = form(model) - strom políčok FieldTree
- V šablóne mapujeme formulárové elementy
  - <input [formField]="myForm.niečo“>
- Čítanie hodnoty cez
  - myForm.niečo().value() alebo
  - model().niečo

Komponent na registráciu používateľa

- Používateľ zadá používateľské meno, e-mail a 2x heslo
- Budeme overovať:
  - Používateľské meno musí mať aspoň 3 znaky
  - E-mail musí byť v správnom formáte
  - Heslo musí byť dostatočne silné
  - Zadané 2 heslá sa musia zhodovať
  - Nedovolíme pridať používateľa, ktorý už v systéme existuje (kontrolujeme meno aj e-mail)

FieldTree signály

- myForm.niečo().
  - value() – hodnota v modeli
  - valid() – boolean, či nie sú žiadne validačné chyby
  - invalid() – boolean, či sú nejaké validačné chyby
  - errors() – pole validačných chýb v tvare
    - { kind: názov chyby,
      message: text chyby,
      field: FieldTree, ktorý má chybu}
  - pending() – boolean, či sa ešte vyhodnocujú chyby
  - touched() – boolean, či už používateľ bol na políčku a odišiel
  - dirty() – boolean, či používateľ zmenil pôvodnú hodnotu
  - disabled() – boolean, políčko je dočasne needitovateľné
  - readonly() – boolean, políčko je trvalo needitovateľné
  - hidden() – boolean, políčko by malo byť schované (treba schovať cez @if)

Validátory pre FieldTree

- Napríklad
  - registerForm = form(this.model, schemaPath => {

    required(schemaPath.login, { message: 'Login name is required' }),

    minLength(schemaPath.login, 3, { message: 'Login must have at least 3 characters' }),

    email(schemaPath.email, { message: 'Please enter a valid email address' }),

    …

    })

- Ďalšie vstavané funkcie
  - max, min, maxLength, pattern

Výsledok validity vo formulári

```
<input matInput [formField]="registerForm.login">
<mat-error>
   @if(registerForm.login().invalid() && !registerForm.login().pending()) {
         {{registerForm.login().errors()[0].message}}
    }
</mat-error>
```

Vlastné validátory – synchrónne

- Vlastné validátory – validate
  - myForm = form(this.model, schemaPath => {

    validate(schemaPath.niečo), ({value}) => {

    if (je to validné) return null;

    return { kind: ‘myError’, message: ‘You cannot do that’};

    }})
    - Okrem value, môžeme zapýtať
      - state, field – na získanie stavu môjho fieldTree a referencie na môj FieldTree
      - valueOf, stateOf, fieldTreeOf – na získanie hodnôt pre iné FieldTree

Vlastný validátor pre formulárový komponent

- vytvoríme si validátor na kvalitu hesla cez knižnizu zxcvbn
  - npm install @zxcvbn-ts/core @zxcvbn-ts/language-common @zxcvbn-ts/language-en

Vlastné validátory – asynchrónne

- validateHttp(schemaPath.niečo, {
  request: input => {url: '…'},
  onSuccess: (result, context) => null alebo {kind, message},
  onError: (error) => null alebo {kind, message}
  }
- validateAsync(schemaPath.niečo, {
  params: input => dáta pre factory napr. na základe input.value(),
  factory: (params) => rxResource<návratový typ, typ vstupu>({
  params: () => vracia vstup pre stream na základe params()
  stream:({params}) => vracia observable<návratový typ>
  }),
  onSuccess: (result, context) => null alebo {kind, message},
  onError: (error) => null alebo {kind, message}
  }