

Osnove izrade PHP aplikacija

Provjera obrazaca u JavaScriptu

Izradio: Marijan Šuflaj
Uredio: Davor Cihlar

Plan

- ▶ Što ćemo naučiti
 - ▶ Ručna provjera obrasca
 - ▶ Automatizirana provjera obrasca
 - ▶ Rana provjera obrasca pomoću poslužitelja

Zašto provjera obrazaca u JS?

- ▶ Bolji korisnički doživljaj
 - ▶ Nije potrebno čekati da se ponovno učita stranica nakon slanja obrasca
 - ▶ Korisnik može vidjeti problem i prije nego pošalje obrazac
- ▶ **Upozorenje:** Nije zamjena za provjeru na poslužitelju
 - ▶ JS je moguće isključiti
 - ▶ Bez provjere na poslužitelju skripta je ranjiva

Kada provjeravati

- ▶ Četiri korisna događaja:
 - ▶ `submit()` – poziva se tik prije slanja obrasca
 - ▶ Vrijedi samo za element `<form>`
 - ▶ `blur()` – poziva se kad element izgubi fokus
 - ▶ Npr. kad korisnik prijeđe iz jednog polja za unos teksta u drugo
 - ▶ Vrijedi za bilo koji element
 - ▶ `change()` – poziva se nakon izmjene vrijednosti
 - ▶ Vrijedi samo za elemente `<input>`, `<textarea>` i `<select>`
 - ▶ `click()` – poziva se nakon što korisnik klikne na element
 - ▶ Vrijedi za bilo koji element

Načini provjere

- ▶ Provjeriti obrazac možemo na dva načina:
 - ▶ Ručnom provjerom
 - ▶ Automatskom provjerom pomoću dodatka za jQuery

Primjer (1.1)



► Obrazac:

► `<form id="obrazac">`

Ime:

```
<input type="text" id="ime" name="ime" />
```

```
<span id="greska0"></span><br />
```

Prezime:

```
<input type="text" id="prez" name="prez" />
```

```
<span id="greska1"></span><br />
```

Dob:

```
<input type="text" id="dob" name="dob" />
```

```
<span id="greska2"></span><br />
```

```
<input type="submit" value="Pošalji" />
```

```
</form>
```

Primjer (1.2)



▶ Zabrana slanja obrasca

```
▶ $(document).ready(function() {  
    // nakon što se stranica ucitala
```

```
    greske = [false, false, true];
```

```
    $('#obrazac').submit(function(e) {  
        // tik prije slanja obrasca  
        for (i in greske) {  
            if (greske[i]) {  
                alert('Ispravi podatke!');  
                e.preventDefault();  
                return;  
            }  
        }  
    });  
});
```

Primjer (1.3)



► Provjera dobi:

```
► $( '#dob' ).blur( function() {  
    // nakon što "dob" izgubi fokus
```

```
var dob = $(this).val();  
dob = parseInt(dob, 10);
```

```
if (isNaN(dob) || dob < 1 || dob > 150) {  
    $( '#greska2' ).text( 'Dob ne valja!' );  
    greske[2] = true;  
} else {  
    $( '#greska2' ).text( '' );  
    greske[2] = false;  
}  
});
```

Zadatak (1)



- ▶ Dovršite prvi primjer tako da se provjerava da li je ime i prezime ispunjeno.
 - ▶ Svojstvom `length` moguće je provjeriti duljinu stringa

jQuery validate plugin

- ▶ Oko ručne provjere ima dosta posla
 - ▶ Puno bolje bi bilo automatizirano kao u Smartyju
- ▶ Na svaki element se "zakači" metoda provjere s određenim parametrima
 - ▶ Preko atributa `class`
 - ▶ Preko pravila zadanih prilikom inicijalizacije
- ▶ jQuery sam pokreće metode provjere kada su potrebne
- ▶ Dodatno potrebno priključiti:
 - ▶

```
<script type="text/javascript" src="http://ajax.aspnetcdn.com/ajax/jquery.validate/1.8/jquery.validate.min.js"></script>
```
- ▶ Više na:
 - ▶ <http://docs.jquery.com/Plugins/Validation>

Način provjere zadan atributom class

- ▶ Najjednostavniji i najkraći način
 - ▶ Samo jedna linija u JS, ostalo je sve HTML
- ▶ Bitnije provjere:
 - ▶ `required` - element ne smije biti prazan
 - ▶ `email` - vrijednost mora biti valjana adresa e-pošte
 - ▶ `number` - vrijednost mora biti broj
 - ▶ `url` - vrijednost mora biti ispravan URL
- ▶ Provjere je moguće kombinirati tako da se odvajaju razmacima
 - ▶ Od navedenih jedino ima smisla kombinirati `required` s ostalima

Primjer (2)



- ▶ Provjera da li je valjana adresa e-pošte

- ▶ Obrazac:

- ▶ `<form id="obrazac">`

- e-mail:

- `<input type="text" name="email" class="required email"/>`

- `
`

- `<input type="submit" value="Pošalji"/>`

- `</form>`

- ▶ Inicijalizacija:

- ▶ `$(document).ready(function() {`
 `// nakon što se stranica ucitala`
 `$('#obrazac').validate();`
`});`

Metoda validate

- ▶ Način provjere prikazan u prethodnom primjeru ne podržava parametrisiranje
 - ▶ Npr. nije moguće zadati da je broj unutar nekog raspona
 - ▶ Nije moguće odrediti ispis greške
- ▶ Puno je bolje (moćnije) rješenje metodi `validate` proslijediti pravila provjere
- ▶ Više na:
 - ▶ <http://docs.jquery.com/Plugins/Validation/validate>

Parametri metode validate

- ▶ Parametri se zadaju kao asocijativno polje
- ▶ Bitniji parametri su:
 - ▶ `rules` – pravila provjere svakog pojedinog polja
 - ▶ `messages` – poruke o grešci za svako pojedino polje
 - ▶ `submitHandler` – poziva zadanu funkciju umjesto da pošalje obrazac
- ▶ Primjer:
 - ▶

```
jQobrazac.validate({
  rules: {...},
  submitHandler: function() {...}
});
```

Pravila provjere (1)

- ▶ Pravila se zadaju kao asocijativno polje
 - ▶ Ključ označava ime elementa
 - ▶ Vrijednost je opet asocijativno polje s pravilima za taj element
- ▶ Zapis:
 - ▶

```
jqobrazac.validate({
  rules: {
    imeElementa: {
      pravilo1: vrijednost1,
      pravilo2: vrijednost2
    }
  }
});
```
- ▶ Više o mogućim pravilima na:
 - ▶ http://docs.jquery.com/Plugins/Validation#List_of_built-in_Validation_methods

Pravila provjere (2)

▶ Bitnija pravila:

- ▶ Vrijede ista ona koja su vrijedila u `class` atributu
 - ▶ Vrijednost mora biti `true`
- ▶ `minlength(len)`, `maxlength(len)` – minimalna i maksimalna duljina vrijednosti
- ▶ `min(val)`, `max(val)` – minimalan i maksimalan iznos vrijednosti
- ▶ `equalTo("#elementId")` – provjera da li je vrijednost elementa jednaka elementu s `ID=elementId`

▶ Primjer:

```
rules: {  
  elemBroj: {required: true, max: 42}  
}
```

Primjer (3)



▶ Primjer (2) izveden pomoću pravila

▶ Obrazac:

▶ `<form id="obrazac">`

 e-mail:

`<input type="text" name="email" />
`

`<input type="submit" value="Pošalji" />`

`</form>`

▶ Inicijalizacija:

▶ `$('#obrazac').validate({`

 rules: {

 email: { required: true, email: true }

 }

});

Poruke o grešci

- ▶ Struktura slična pravilima
- ▶ Razlikuje se samo u vrijednosti koja je poruka o grešci
- ▶ Primjer:

```
▶ messages: {  
    imeElementa1: {  
        required: "Potrebna vrijednost",  
        email: "Nije valjana adresa"  
    },  
    imeElementa2: {  
        required: "Potrebna vrijednost"  
    }  
}
```

Pravilo remote

- ▶ Pravilo `remote` provjerava ispravnost elementa pomoću poslužitelja
 - ▶ Ime i vrijednost koju je korisnik unio šalje se poslužitelju metodom GET
 - ▶ Poslužitelj vraća **JSON**:
 - ▶ `true` ako je ispravna vrijednost
 - ▶ `false` ako nije i potrebno je ispisati pretpostavljenu poruku o grešci
 - ▶ String s opisom greške
- ▶ Vrijednost pravila `remote` je ime PHP skripte za provjeru
- ▶ Više na:
 - ▶ <http://docs.jquery.com/Plugins/Validation/Methods/remote>

Primjer (4.1)



► Obrazac:

► `<form id="obrazac">`

 Paran broj:

`<input type="text" name="broj" />
`

 Broj 4:

`<input type="text" name="broj4" />
`

`<input type="submit" value="Pošalji" />`

`</form>`

Primjer (4.2)



► Pravila:

```
► $( '#obrazac' ).validate({
  rules: {
    broj: {
      required: true,
      remote: 'primjer4-provjera.php'
    },
    broj4: {
      required: true,
      remote: 'primjer4-provjera.php'
    }
  },
  messages: {
    broj4: { remote: 'Unesen je krivi broj' }
  }
});
```

Primjer (4.3)



► PHP skripta za provjeru:

► `$msg = true;`

```
if (isset($_GET['broj'])) {  
    $broj = (int)$_GET['broj'];  
    if ($broj % 2) $msg = 'Broj je neparan';  
} elseif (isset($_GET['broj4'])) {  
    $broj = (int)$_GET['broj4'];  
    if ($broj != 4) $msg = false;  
}
```

```
header('Content-Type: application/json');  
echo json_encode($msg);
```

Zadatak (2)



- ▶ Napravite obrazac za registraciju koji od korisnika traži korisničko ime, zaporku, ponovljenu zaporku, e-mail i URL svoje stranice
 - ▶ Korisničko ime mora biti dulje od 6 znakova, kraće od 16 i ne smije biti zauzeto
 - ▶ Provjera pomoću `remote`
 - ▶ Lozinka mora biti identična ponovljenoj i dulja od 6 znakova
 - ▶ URL jedini nije obavezan
- ▶ Nije potrebno implementirati registraciju, ali napravite neko polje koje "glumi" postojeća korisnička imena

Zadatak (3)



- ▶ Proširite prethodni zadatak tako da korisnik ne može naštetiti ukoliko zaobiđe provjeru pomoću JavaScripta
 - ▶ Provjera na poslužitelju