

# Razvoj primijenjene programske potpore

---




## 8.2 Primjer zaglavlje-stavke (2.dio)

Ažuriranje podataka

# Forma za ažuriranje dokumenta





- Forma za ažuriranje dokumenta sastoji se od
  - zaglavlja za ažuriranje podataka koji pripadaju tablici Dokument (*master*)
    - odabir partnera i prethodnog dokumenta izveden nadopunjavanjem
  - detalja sa stavkama dokumenta (*detail*)
    - moguće promijeniti vrijednost stavke (količina i rabat), obrisati je ili dodati novu (poseban redak nakon svih stavki)
- Zajednička akcija za spremanje podataka (zaglavlje + stavke)

Dokument br: 5555


Vrsta dokumenta  Porez (u %)  Datum    

Broj  Partner

Prethodni dokument  Iznos **2.414,38 kn**



Artikl	Količina	Rabat [0-1]	Jedinična cijena	Iznos	
AC Adapter ASIIN Mitac - 442687900004	<input type="text" value="3,00000"/>	<input type="text" value="0,00"/>	106,00 kn	318,00 kn	
ArtiklProba F	<input type="text" value="3,00000"/>	<input type="text" value="0,00"/>	35,00 kn	105,00 kn	
AC/DC adapter za 3Pod KINGSINGTON, 70W (33335EU)	<input type="text" value="2,00000"/>	<input type="text" value="0,00"/>	778,00 kn	1.556,00 kn	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	

# Priprema dokumenta za ažuriranje (1)

- Za početak ažuriranja dokumenta i njegovih stavki potrebno iz BP dohvatiti osnovne podatke o dokumentu i popuniti model iz prethodnih slajdova
- Slično kao kod prikaza pojedinačnog složenog podatka
  - Dohvat podatka je isti, samo se koristi drugačiji pogled za prikaz
  - Primjer:  MVC \ Controllers \ DokumentController.cs


```
[HttpGet]
public Task<IActionResult> Edit(int id,
                                int? position, string filter, int page = 1,
                                int sort = 1, bool ascending = true) {
    return Show(id, position, filter, page, sort, ascending,
                nameof(Edit));
}
```

## Priprema dokumenta za ažuriranje (2)

- Odabir partnera i prethodnog dokumenta vrši se nadopunjavanjem, pa ne treba pripremati podatke za padajuće liste
  - Primjer:  MVC \ Controllers \ AutoCompleteController.cs : postupci Dokument i Partner
- Potrebno dohvatiti nazive partnera i dokumenta kako bi se prikazali korisniku
  - Za partnera provjeriti tip i dohvatiti ime i prezime, odnosno naziv tvrtke
  - Slično za dokument, pri čemu je potrebno koristiti isti format koji se koristi u upravljaču koji služi kao izvor za nadopune
  - Programski kod prevelik za slajd, ali relativno jednostavan...
  - Primjer:  MVC \ Controllers \ DokumentController.cs

```
[HttpGet]
public async Task<IActionResult> Show(int id, ...
...
dokument.NazPartnera = ...
if (dokument.IdPrethDokumenta.HasValue) {
    dokument.NazPrethodnogDokumenta = ...
```

# Pogled za ažuriranje dokumenta

- Dio sa zaglavljem nalik ostalim formama za ažuriranje pojedinačnog podatka
  - Input kontrole + kontrole za nadopunjavanje
  - Gumb za povratak na popis dokumenata, osvježavanje trenutnog dokumenta i spremanje promjena
- Na dnu forme uključen parcijalni pogled za stavke dokumenta
  - Primjer:  MVC \ Views \ Dokument \ Edit.cshtml

```
@model DokumentViewModel
```

```
...
```

```
<form id="form" method="post" asp-action="Edit" ... >
```

```
  <input type="hidden" asp-for="IdDokumenta"/>
```

```
  ...
```

```
  <input asp-for="StopaPoreza" class="form-control" />
```

```
  ...
```

```
  <button id="save" type="submit" title="Spremi">...</button>
```

```
  ...
```

```
  <partial name="Stavke" model="Model.Stavke" />
```

# Povezivanje kolekcije podataka

- U slučaju više kontrola s istim atributom *name*, unesene vrijednosti se na upravljaču prihvaćaju u istoimenom argumentu koji je polje određenog tipa
- Što ako treba povezati više složenih podataka (npr. više stavki)?
  - Za atribut *name* se koristi oblik NazivKolekcije[indeks].NazivSvojstva
    - npr. name="Stavke[0].IdStavke", name="Stavke[1].IdStavke", name="Stavke[2].IdStavke" ...
  - Indeksi moraju biti bez prekida počevši od 0
- Što ako nije moguće osigurati neprekinuti niz indeksa?
  - Mogu se koristiti bilo koje oznake (čak ni ne moraju biti brojevi), ali dodatno treba stvoriti (skriveno) elemente naziva *Stavka.Index* čija je vrijednost jednaka korištenoj vrijednosti u uglatim zagradama


```
<input type="hidden" name="Stavke.Index" value="knjiga" />
```

```
<input type="text" name="Stavke[knjiga].IdStavke" ... />
```

```
<input type="hidden" name="Stavke.Index" value="5" />
```


```
<input type="text" name="Stavke[5].IdStavke" ... />
```

# Parcijalni pogled za ažuriranje stavki

- *DokumentViewModel* sadrži svojstvo *Stavke*
  - Povezivanje kolekcije stavki i njenih svojstva ostvareno varijantom sa *Stavke.Index* pri čemu se kao indeks koristi šifra artikla
  - Primjer:  MVC \ Views \ Dokument \ Stavke.cshtml

```
@model IEnumerable<StavkaViewModel>
...
@foreach (var stavka in Model) {
    <input type="hidden" name="Stavke.Index"
        value="@stavka.SifArtikla"/>
    <input type="hidden"
        name="Stavke[@stavka.SifArtikla].IdStavke"
        value="@stavka.IdStavke" />
    ...
    <input name="Stavke[@stavka.SifArtikla].KolArtikla"
        value="@stavka.KolArtikla"/>
    ...
}
```

# Izmjena dokumenta i njegovih stavki (1)


- Prilikom dohvata dokumenta potrebno dohvatiti njegove stavke
- Kao prvi korak vrši se kopiranje podataka dokumenta iz povezanog modela u objekt koji je dohvaćen iz BP
  - Primjer:  MVC \ Controllers \ DokumentController.cs

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(DokumentViewModel model, ...
    var dokument = await ctx.Dokument
        .Include(d => d.Stavka)
        .Where(d => d.IdDokumenta == model.IdDokumenta)
        .FirstOrDefaultAsync();

... Kopiraj podatke o dokumentu iz modela u objekt dokument
... dokument.PostoPorez = model.StopaPoreza / 100m;
    dokument.VrDokumenta = model.VrDokumenta;
```




## Izmjena dokumenta i njegovih stavki (2)

- U listu cijelih brojeva spremaju se identifikatori povezanih stavki
  - Predstavlja stavke koje nisu uklonjene iz HTML-a prije snimanja
  - Iz konteksta se izbacuju sve stavke dokumenta koje se ne nalaze u toj listi
  - Primjer:  MVC \ Controllers \ DokumentController.cs


```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(DokumentViewModel model, ...
...
    List<int> idStavki = model.Stavke
                                .Where(s => s.IdStavke > 0)
                                .Select(s => s.IdStavke)
                                .ToList();
    //izbaci sve koje su nisu više u modelu
    ctx.RemoveRange(
        dokument.Stavka.Where(s => !idStavki.Contains(s.IdStavke)));
```

# Izmjena dokumenta i njegovih stavki (3)

- Stavke koje se nalaze u povezanom modelu ili nove stavke ili neke od postojećih koje treba ažurirati
  - Postojeće treba dohvatiti iz objekta prethodno dohvaćenog iz baze podataka
  - Nove treba stvoriti i dodati u dokument
  - U oba slučaja u novi objekt kopirati svojstva iz povezanog modela
- Primjer:  MVC \ Controllers \ DokumentController.cs


```
foreach (var stavka in model.Stavke) {  
    Stavka novaStavka; //potpuno nova ili ona koju treba izmijeniti  
    if (stavka.IdStavke > 0)  
        novaStavka = dokument.Stavka  
            .First(s => s.IdStavke == stavka.IdStavke);  
    else {  
        novaStavka = new Stavka();  
        dokument.Stavka.Add(novaStavka);  
    }  
    novaStavka.SifArtikla = stavka.SifArtikla;  
    novaStavka.KolArtikla = stavka.KolArtikla;  
    ... kopiranje ostalih vrijednosti
```

## Izmjena dokumenta i njegovih stavki (4)




- Potrebno izračunati iznos dokumenta i spremiti promjene
  - Primjer:  MVC \ Controllers \ DokumentController.cs

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(DokumentViewModel model, ...
...
dokument.IznosDokumenta = (1 + dokument.PostoPorez)
                        *
                        dokument.Stavka
                            .Sum(s =>
                                s.KolArtikla
                                * (1 - s.PostoRabat)
                                * s.JedCijArtikla
                            );
await ctx.SaveChangesAsync();
```



# Odabir artikla za novu stavku

- Redak na dnu postojećih stavki u kojem se nadopunjavanjem bira artikl
  - Povratni podaci sa servera sadrže šifru, naziv i cijenu artikla
  - Cijena artikla se kopira u odgovarajuće polje
  - Primjer  MVC \ Views \ Dokument \ Stavke.cshtml

```
<tr>
  <td>
    <input id="artikl-sifra" type="hidden" data-autocomplete-
placeholder="artikl" readonly="readonly" />
    <input id="artikl-naziv" type="text" data-autocomplete="artikl" />
  </td>
  <td class="text-center col-sm-1">
    <input id="artikl-kolicina" type="text" />
  </td> ...
```

- Pogledati programski kôd u sljedećim datotekama:
  -  MVC \ ViewModels \ AutoCompleteArtikl.cs
  -  MVC \ Controllers \ AutoCompleteController.cs : Artikl
  -  MVC \ wwwroot \ js \ autocomplete.js

# Dodavanje nove stavke

- Skrivena tablica s identifikatorom *template* koja služi za dodavanje novog retka na temelju vrijednosti iz posebnog retka na dnu stavki (opisano na prethodnom slajdu)
  - Provjera ispravnosti vrijednosti, provjera postoji li duplikat artikla i slično
  - Dodavanje novog retka
  - Ponašanje tipke ENTER
- Redak s predloškom izvan forme kako ne bi sudjelovao u povezivanju
- Pogledati programski kôd u sljedećim datotekama:
  -  MVC \ Views \ Dokument \ NovaStavkaTemplate.cshtml
  -  MVC \ wwwroot \ js \ dokumenti.js
    - Postupak *dodajArtikl* i kod koji poziva postupak

# Brisanje stavke


- Gumb za brisanje stavke (prepoznaje se po stilu *deleterow*) uklanja stavku iz strukture HTML dokumenta.

- Primjer:  MVC \ wwwroot \ js \ dokument.js

```
$(document).on('click', '.deleterow', function () {  
    event.preventDefault();  
    var tr = $(this).parents("tr");  
    tr.remove();  
    ... očisti staru poruku da je dokument uspješno snimljen ...  
});
```

- Posljedično bit će izbrisana iz dokumenta nakon klika na gumb za snimanje


# Kreiranje novog dokumenta

- Kao početni model za unos novog dokumenta stvara se novi objekt kojem se postavlja trenutni datum i sljedeći broj dokumenta
  - Samo kao primjer početne inicijalizacije, jer u slučaju istovremenog dodavanja novih dokumenata može doći do ponavljanja istog broja
- Primjer:  MVC \ Controllers \ DokumentController.cs

```
[HttpGet]
public async Task<IActionResult> Create() {
    int maxbr = await ctx.Dokument.Max(d => d.BrDokumenta) + 1;
    var dokument = new DokumentViewModel {
        DatDokumenta = DateTime.Now,
        BrDokumenta = maxbr
    };
    return View(dokument);
}
```

- U slučaju uspješnog kreiranja dokumenta dolazi do preusmjerenja na akciju *Edit*

# Brisanje dokumenta

- U skladu s prethodnim primjerima.
- U BP definirano kaskadno brisanje, pa je dovoljno dohvatiti dokument i obrisati ga
  - Ne treba brisati pojedinačno svaku stavku.
  - Primjer:  MVC \ Controllers \ DokumentController.cs

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Delete(int IdDokumenta, ...) {
    var dokument = await ctx.Dokument
                            .Where(d => d.IdDokumenta == IdDokumenta)
                            .SingleOrDefaultAsync();

    if (dokument != null) {
        ...
        ctx.Remove(dokument);
        await ctx.SaveChangesAsync();
    }
}
```