

# Razvoj primijenjene programske potpore

---

## **8.1 Primjer zaglavlje-stavke (1.dio)**


Prikaz svih dokumenata, filtriranje dokumenata,  
pojedinačni prikaz

# Pogled za prikaz svih dokumenata

- Kao i u primjeru s partnerima za dohvat svih dokumenata koristi se pogled iz baze podataka
  - Značajno pojednostavljuje kôd za dohvat podataka u upravljaču


```
CREATE VIEW [dbo].[vw_Dokumenti] AS
SELECT  dbo.Dokument.IdDokumenta, dbo.Dokument.VrDokumenta,
        dbo.Dokument.BrDokumenta, dbo.Dokument.DatDokumenta,
        dbo.Dokument.IdPartnera, dbo.Dokument.IdPrethDokumenta,
        dbo.Dokument.PostoPorez, dbo.Dokument.IznosDokumenta,
        CASE dbo.Partner.TipPartnera WHEN 'O' THEN
            dbo.Osoba.PrezimeOsobe + ', ' + dbo.Osoba.ImeOsobe
        ELSE dbo.Tvrtka.NazivTvrtke END
        + ' (' + dbo.Partner.OIB + ')' AS NazPartnera
FROM    dbo.Dokument INNER JOIN dbo.Partner
        ON dbo.Dokument.IdPartnera = dbo.Partner.IdPartnera
LEFT OUTER JOIN dbo.Osoba
        ON dbo.Partner.IdPartnera = dbo.Osoba.IdOsobe
LEFT OUTER JOIN dbo.Tvrtka
        ON dbo.Partner.IdPartnera = dbo.Tvrtka.IdTvrtke
```

# (Podsjetnik) Dodavanje pogleda u EF model (1)

- Potrebno definirati razred koji svojoj strukturom odgovara rezultatu pogleda
  - Svojstva koja imaju set dio, a ne odgovaraju nekom stupcu iz rezultata označavaju se atributom *NotMapped*
  - Primjer:  MVC \ ModelsPartial \ ViewDokumentInfo.cs

```
public class ViewDokumentInfo {  
    public int IdDokumenta { get; set; }  
    public decimal PostoPorez { get; set; }  
    public int? IdPrethDokumenta { get; set; }  
    public DateTime DatDokumenta { get; set; }  
    public int IdPartnera { get; set; }  
    public string NazPartnera { get; set; }  
    public decimal IznosDokumenta { get; set; }  
    public string VrDokumenta { get; set; }  
    public int BrDokumenta { get; set; }  
    [NotMapped]  
    public int Position { get; set; } //Position in result  
}
```

## (Podsjetnik) Dodavanje pogleda u EF model (2)

- U definiciju konteksta dodati novi DbSet za pogled
  - Primjer:  MVC \ ModelsPartial \ FirmaContext.cs

```
namespace MVC.Models
public partial class FirmaContext {
    ...
    public virtual DbSet<ViewDokumentInfo> vw_Dokumenti { get; set; }

    partial void OnModelCreatingPartial(ModelBuilder modelBuilder) {
        ...
        modelBuilder.Entity<ViewDokumentInfo>(entity => {
            entity.HasNoKey();
            //entity.ToView("vw_Dokumenti");
            //u slučaju da se DbSet svojstvo zove drugačije
        });
    }
}
```

# Filtriranje podataka

- Popis dokumenata moguće filtrirati po partneru, iznosu ili datumu.
- Za odabir partnera nadopunjavanje, a za odabir datuma kalendar
  - Kalendar po automatizmu (zbog tipa *DateTime* i *input asp-for*)
  - Atribut `[DataType(DataType.Date)]` uklanja odabir vremena
- Gumb na formi za unos kriterija šalje popunjene podatke na akciju *Filter* koja spaja kriterije u jedan string koji se šalje kao parametar akciji Index
  - Primjerice za odabir sa slike i aktiviranje filtera vrijednost parametra filter bit će `filter=0-01.01.2015-10.05.2017-300,00-500,00`

- Gumb za uklanjanje filtra je poveznica na akciju Index bez parametara
- Kriterij pretrage parcijalni pogled uključen u pogled Index

```
<partial name="KriterijPretrage"
model="Model.Filter" />
```

IdPartnera: 0    Zadovoljni korisnik (01234567988)

Iznos: 300 - 500

Datum: 01.05.2015. - 10.05.2017.

svibanj, 2017

pon	uto	sri	čet	pet	sub	ned
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

12.01.2014.


12.01.2014.

1.803.667,82 kn

# Parametar ili sjednica?


- Prednosti korištenja paramet(a)ra za informaciju o filtriranju
  - Moguće imati nekoliko aktivnih filtara unutar više kartica istog preglednika
  - Moguće pohraniti poveznicu za buduće posjete
    - Informacija se ne gubi istekom sjednice
  - Može se koristiti ako server koristi *load balancing*
- Mane:
  - Potreba za korištenjem više parametara, odnosno pronalaskom efikasnog načina za spremanje više kriterija unutar istog stringa
  - Potrebno prenositi parametre u različite akcije
    - vidi primjer s državama i parametrima page, sort ascending
  - Treba obratiti pažnju na znakove koji mogu utjecati na rekonstrukciju vrijednosti pojedinog filtra
  - Prevelik broj kriterija i dugi tekstovi kriterija mogli bi uzrokovati adresu izvan raspona dopuštene duljine
    - Alternativa: kriterij pohraniti negdje (sjednica, BP) i pridružiti mu neku vrijednost koja bi se koristila unutar adrese

# Razred za informacije o filtru (1)

- Za prihvatanje podataka o filteru koristi se razred *DokumentFilter*
  - Osim svojstava za prihvatanje unesenih vrijednosti sadrži i nekoliko pomoćnih metoda kojima se olakšava rad s filtriranjem
    - *IPageFilter* - vlastito sučelje kao „zajednički nazivnik” za pager
  - Primjer:  MVC \ ViewModels \ DokumentFilter.cs

```
public class DokumentFilter : IPageFilter {  
    public int? IdPartnera { get; set; }  
    public string NazPartnera { get; set; }  
    public DateTime? DatumOd { get; set; }  
    public DateTime? DatumDo { get; set; }  
    public decimal? IznosOd { get; set; }  
    public decimal? IznosDo { get; set; }  
    public bool IsEmpty() {  
        bool active = IdPartnera.HasValue  
                    || DatumOd.HasValue || DatumDo.HasValue  
                    || IznosOd.HasValue || IznosDo.HasValue;  
        return !active;  
    }  
}
```


## Razred za informacije o filtru (2)

- Uneseni podaci mogu se spojiti u string te rekonstruirati iz stringa
  - Primjer:  MVC \ ViewModels \ DokumentFilter.cs

```
public class DokumentFilter {
    public override string ToString() {
        return string.Format("{0}-{1}-{2}-{3}-{4}",
            IdPartnera, DatumOd?.ToString("dd.MM.yyyy"),
            DatumDo?.ToString("dd.MM.yyyy"), IznosOd, IznosDo);
    }
    public static DokumentFilter FromString(string s) {
        var filter = new DokumentFilter();
        string[] arr = s.Split('-', StringSplitOptions.None);
        filter.IdPartnera = string.IsNullOrEmpty(arr[0]) ?
            new int?() : int.Parse(arr[0]);
        filter.DatumOd = string.IsNullOrEmpty(arr[1]) ?
            new DateTime?() : DateTime.ParseExact(arr[1],
                "dd.MM.yyyy", CultureInfo.InvariantCulture);
        ...
        return filter;
    }
}
```



## Razred za informacije o filtru (3)

- Upit za dohvat svih dokumenata filtrira se po odabranim kriterijima
  - Primjer:  MVC \ ViewModels \ DokumentFilter.cs

```
public class DokumentFilter {  
    ...  
    public IQueryable<ViewDokumentInfo> Apply(  
        IQueryable<ViewDokumentInfo> query) {  
  
        if (IdPartnera.HasValue)  
            query = query  
                .Where(d => d.IdPartnera == IdPartnera.Value);  
  
        if (DatumOd.HasValue)  
            query = query  
                .Where(d => d.DatDokumenta >= DatumOd.Value);  
  
        ...  
    }  
}
```

# Dohvat svih partnera (za nadopunjavanje)

- Izvedeno korištenjem pogleda korištenog za pregled svih partnera
  - Primjer:  MVC \ Controllers \ AutoComplete.cs


```
public async Task<IEnumerable<IdLabel>> Partner(string term) {  
    var query = ctx.vw_Partner  
        .Select(p => new IdLabel  
        {  
            Id = p.IdPartnera,  
            Label = p.Naziv + " (" + p.OIB + ")")  
        })  
        .Where(l => l.Label.Contains(term));  
  
    var list = await query.OrderBy(l => l.Label)  
        .ThenBy(l => l.Id)  
        .Take(appData.AutoCompleteCount)  
        .ToListAsync();  
}
```

# Digresija: Dohvat svih partnera upitom bez pogleda

- Bez pogleda, kôd za upit korištenjem EF-a bi bio puno složeniji

```
var queryOsobe = ctx.Osoba.Select(o => new IdLabel {
    Id = o.IdOsobe,
    Label = o.PrezimeOsobe + ", " +
o.ImeOsobe + " (" + o.IdOsobeNavigation.Oib + ")")
    })
    .Where(l => l.Label.Contains(term));
var queryTvrtnke = ctx.Tvrtnka.Select(t => new IdLabel {
    Id = t.IdTvrtnke,
    Label = t.NazivTvrtnke + ", " +
" (" + t.IdTvrtnkeNavigation.Oib + ")")
    })
    .Where(l => l.Label.Contains(term));
var list = queryOsobe.Union(queryPartneri)
    .OrderBy(l => l.Label)
    .ThenBy(l => l.Id)
    .ToList();
```


# Aktiviranje nadopunjavanja za partnere

- Izvedeno slično kao kod padajuće liste za mjesta prilikom dodavanja novog partnera
  - Vlastita skripta aktivira nadopunjavanje za sve kontrole koje imaju definiran atribut *data-autocomplete* pri čemu vrijednost tog atributa predstavlja relativnu adresu izvora podataka
  - Razlika je u tome što se id odabranog partnera vidi na formi
    - Nije skriveno polje, ali se ne može mijenjati (atribut *readonly*)
    - Primijetiti da se veže za svojstvo *NazPartnera* iz razreda *DokumentFilter*
  - Primjer:  MVC \ Views \ Dokument \ KriterijPretrage.cshtml

```
<input asp-for="IdPartnera" readonly="readonly" class="form-control" data-autocomplete-placeholder="partner" />
```


```
<input class="form-control" data-autocomplete="partner" asp-for="NazPartnera" />
```

# Identifikator i naziv partnera u filtru

- Naziv odabranog partnera dio razreda DokumentFilter, ali nije dio teksta koji se prenosi u adresi unutar parametra *filter*
  - Potrebno ga je obnoviti upitom temeljem *IdPartnera*
- Primjer:  MVC \ Controllers \ DokumentController.cs


```
public async Task<IActionResult> Index(string filter, ...
...
DokumentFilter df = DokumentFilter.FromString(filter);
if (!df.IsEmpty()) {
    if (df.IdPartnera.HasValue) {
        df.NazPartnera = await ctx.vw_Partner
            .Where(p => p.IdPartnera == df.IdPartnera)
            .Select(vp => vp.Naziv)
            .FirstOrDefaultAsync();
    }
    query = df.Apply(query);
} ...
```

# Model za rad s dokumentima (1)

- Prezentacijski model s validacijskim atributima (bit će korišteni kod ažuriranja)
  - Primjer:  MVC \ ViewModels \ DokumentViewModel.cs

```
public class DokumentViewModel {  
    public int IdDokumenta { get; set; }  
    ...  
    [DataType(DataType.Date)]  
    [Display(Name = "Datum")]  
    [Required(ErrorMessage = "Potrebno je ... dokumenta")]  
    public DateTime DatDokumenta { get; set; }  
    [Display(Name = "Porez (u %)")]  
    [Required(ErrorMessage = "Potrebno je postotak poreza")]  
    [Range(0, 100, ErrorMessage = "Porez mora biti 0-100")]  
    public int StopaPoreza { get; set; }  
    public decimal PostoPorez {  
        get { return StopaPoreza / 100m; }  
        set { StopaPoreza = (int) (100m * value); }  
    }  
    ...  
}
```


## Model za rad s dokumentima (2)

- Osim atributa koji će u konačnici završiti u tablici Dokument, model sadrži i kolekciju stavki
  - Primjer:  MVC \ ViewModels \ DokumentViewModel.cs

```
public class DokumentViewModel {  
    ...  
    public IEnumerable<StavkaViewModel> Stavke { get; set; }  
    public DokumentViewModel() {  
        this.Stavke = new List<StavkaViewModel>();  
    }  
}
```

- Stavke prikazane vlastitim prezentacijskim modelom


# Model za rad sa stavkama

- Novi razred kao model za rad sa stavkama da se izbjegnu problemi s vezom prema tablici Dokument
  - Primjer:  MVC \ ViewModels \ StavkaViewModel.cs

```
public class StavkaViewModel {  
    public int IdStavke { get; set; }  
    public int SifArtikla { get; set; }  
    public string NazArtikla { get; set; }  
    public decimal KolArtikla { get; set; }  
    public decimal JedCijArtikla { get; set; }  
    public decimal PostoRabat { get; set; }  
    public decimal IznosArtikla {  
        get {  
            return KolArtikla * JedCijArtikla  
                * (1 - PostoRabat);  
        }  
    }  
}
```




# Prikaz dokumenta (1)

- Prvo dohvat dokumenta, a potom i njegovih stavki
  - Podaci o prethodnom dokumentu i partneru se dohvaćaju naknadnu preko odgovarajućih pogleda (relativno jednostavan kod, ali prevelik za slajdove)
  - Primjer:  MVC \ Controllers \ DokumentController.cs
    - Naziv pogleda je argument ove metode, jer se isti kod koristi i za pripremu dokumenta za ažuriranje

```
public async Task<IActionResult> Show(int id, ...,
                                     string viewName = nameof(Show)) {
    var dokument = await ctx.Dokument
        .Where(d => d.IdDokumenta == id)
        .Select(d => new DokumentViewModel {
            BrDokumenta = d.BrDokumenta,
            DatDokumenta = d.DatDokumenta,
            ...
            VrDokumenta = d.VrDokumenta
        })
        .FirstOrDefaultAsync();
```


## Prikaz dokumenta (2)

- Nakon dohvata dokumenta, dohvatimo sve njegove stavke
  - Primjer:  MVC \ Controllers \ DokumentController.cs


```
public async Task<IActionResult> Show(int id, ...,
                                     string viewName = nameof(Show)) {
    ...
    var stavke = await ctx.Stavka
                        .Where(s => s.IdDokumenta ==
                                   dokument.IdDokumenta)
                        .OrderBy(s => s.IdStavke)
                        .Select(s => new StavkaViewModel {
                            IdStavke = s.IdStavke,
                            NazArtikla = s.SifArtiklaNavigation
                                    .NazArtikla,
                            ...
                        })
                        .ToListAsync();

    dokument.Stavke = stavke;
```

## Prikaz dokumenta (2)


- U zaglavlju prikazani podaci dokumenta, a nakon toga tablično prikazane pojedinačne stavke
- Primjer:  MVC \ Views \ Dokument \ Show.cshtml

# Određivanje sljedbenika i prethodnika (1)

- Na stranici za prikaz dokumenta poveznica za povratak na listu svih dokumenata
  - Pamti se prethodna stranica i način sortiranja
- Dodatno, poveznica na prethodni i sljedeći dokument
  - Svaki dokument ima svoju poziciju unutar baze podataka u ovisnosti o trenutnom sortu i filtru
  - Inicijalno pridijeljeno prilikom dohvata dokumenata
  - Primjer:  MVC \ Controllers \ DokumentController.cs

```
public async Task<IActionResult> Index(string filter,...  
  
... Dohvati dokumente u ovisnosti o filtru i sortu ...  
  
for(int i=0; i<dokumenti.Count; i++)  
    dokumenti[i].Position = (page - 1) * pagesize + i;
```

## Određivanje sljedbenika i prethodnika (2)

- Za prikaz dokumenta potrebno imati njegov id, a ne samo poziciju
- Formira se upit s istim filterom i sortom i dohvate identifikatori prethodnika i sljedbenika
- Primjer:  MVC \ Controllers \ DokumentController.cs

```
private async Task SetPreviousAndNext(int position, string filter,
                                     int sort, bool ascending) {
    var query = ctx.vw_Dokumenti.AsQueryable();
    ... primijeni filter i sort nad upitom
    if (position > 0)
        ViewBag.Previous = await query.Skip(position - 1)
                                     .Select(d => d.IdDokumenta)
                                     .FirstAsync();
    if (position < await query.CountAsync() - 1)
        ViewBag.Next = await query.Skip(position + 1)
                                   .Select(d => d.IdDokumenta)
                                   .FirstAsync();
}
```