

(ASP.NET Core MVC)

Realizacije tipičnih odnosa unutar aplikacije

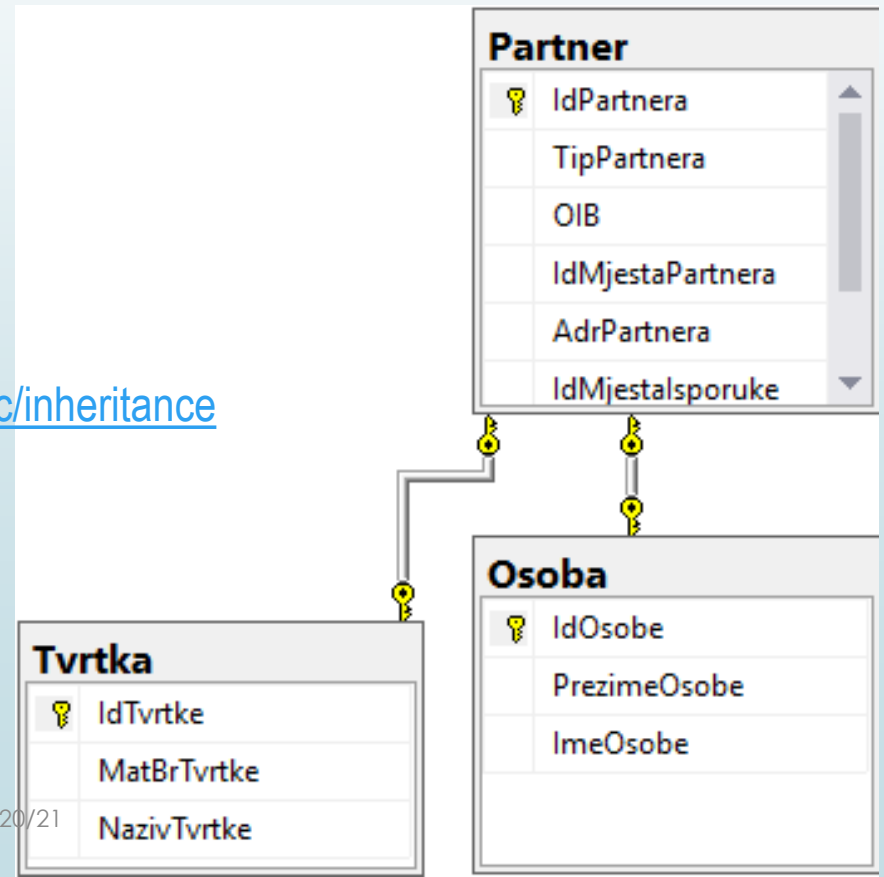
2020/21.08

Specijalizacija i generalizacija + nadopunjavanje umjesto padajuće liste
Primjer zaglavlje-stavke (engl. *master-detail*)

Problem prikaza specijalizacija nekog entiteta

2

- U oglednom modelu Osoba i Tvrтка su specijalizacije Partnera na principu TPT (Table per type)
 - Alternative
 - TPH (Table per hierarchy)
 - TPC (Table per concrete class)
 - EF Core navedene tablice preslika u 3 entiteta
 - U nekim slučajevima, moguće uspostaviti nasljeđivanje u EF modelu, ali to neće biti slučaj u ovom primjeru
 - <https://docs.microsoft.com/en-us/aspnet/core/data/ef-mvc/inheritance>



Problem prikaza specijalizacija nekog entiteta












3

- Što ako u prikazu svih partnera želimo ispisati id partnera, vrstu partnera, OIB i naziv partnera?
 - Upit korištenjem EF-a se komplicira i postaje neefikasan
 - potrebno dohvatiti sve osobe, pa tvrtke te napraviti uniju
 - Što ako treba sortirati podatke?

Popis partnera

Unos novog partnera

1.. 31 32 33 34 35 36 37 38 39 40 41 .. 58

Id partnera	Tip partnera	OIB	Naziv		
20	Osoba	1410949396506	Mustapić, Hrvoje		
750	Tvrtka	3319684	MZOPU		
751	Tvrtka	3316041	NADA DIMIC		
752	Tvrtka	3312400	NAMA		
40	Osoba	1507971335042	Nekić, Ivan		
753	Tvrtka	3308761	NEMO		

Pogled za dohvat podataka o partnerima


4

- ➡ Rješenje prethodnog problema je napisati pogled u bazi podataka te ga uključiti u model
- ➡ Pogled ima sljedeću definiciju

```
CREATE VIEW [dbo].[vw_Partner]
AS
SELECT IdPartnera, TipPartnera, OIB,
       ISNULL(NazivTvrtke, NazivOsobe) AS Naziv
FROM
(
    SELECT IdPartnera, TipPartnera, OIB,
           PrezimeOsobe + ', ' + ImeOsobe AS NazivOsobe,
           NazivTvrtke
    FROM Partner
    LEFT OUTER JOIN Osoba ON Osoba.IdOsobe = Partner.IdPartnera
    LEFT OUTER JOIN Tvrtka ON Tvrtka.IdTvrtke = Partner.IdPartnera
) T
```

Uključivanje pogleda u EF-model (1)


5

- ➡ Kreirati razred koji bi odgovarao podacima u pogledu
 - ➡ Primjer:  MVC \ ModelsPartial \ ViewPartner.cs
- ➡ Dodatno napisano svojstvo koje opisno prikazuje tip partnera

```
public class ViewPartner {  
    public int IdPartnera { get; set; }  
    public string TipPartnera { get; set; }  
    public string OIB { get; set; }  
    public string Naziv { get; set; }  
    public string TipPartneraText {  
        get {  
            if (TipPartnera == "O") {  
                return "Osoba";  
            }  
            else {  
                return "Tvrtka";  
            }  
        }  
    }  
}
```

Uključivanje pogleda u EF-model (2)


6

- Kontekst proširen novim *DbSetom* koji odgovara pogledu
 - naziv svojstva obično odgovara nazivu pogleda, ali nije nužno
 - Može se navesti u `CollectionView("naziv pogleda")` ili SQL upit koji vraća rezultat traženog tipa, npr. `ctx.vw_Partner.FromSqlRaw("SELECT * FROM vw_Partner");`
 - Koristi se kao i drugi *DbSetovi*, ali samo za čitanje
- Primjer:  MVC \ ModelsPartial \ FirmaContext.cs

```
public partial class FirmaContext {  
    public virtual DbSet<ViewPartner> vw_Partner { get; set; }  
    partial void OnModelCreatingPartial(ModelBuilder modelBuilder)  
    {  
        modelBuilder.Entity<ViewPartner>(entity => {  
            entity.HasNoKey();  
            entity.ToView("vw_Partner");  
        });  
    }  
    ...  
}
```

Model za prikaz svih partnera

7

- ➡ (Kao i u prethodnim primjerima) model se sastoji od enumeracije razreda kojim se opisuje pojedinačni podatak i informacija o straničenju i sortiranju
- ➡ Primjer:  MVC \ ViewModels \ PartneriViewModel.cs

```
namespace MVC.ViewModels{  
    public class PartneriViewModel  
    {  
        public IEnumerable<ViewPartner> Partneri { get; set; }  
        public PagingInfo PagingInfo { get; set; }  
    }  
}
```

Priprema modela za prikaz svih partnera

8

➡ Podaci se pripremaju kreiranjem upita za pogled dodan u EF model

➡ Primjer:  MVC \ Controllers \ PartnerController.cs

```
public IActionResult Index(string filter, int page = 1,
                           int sort = 1, bool ascending = true) {
    int pagesize = appData.PageSize;
    var query = ctx.vw_Partner.AsQueryable();
    ...proširenje upita (sortiranje)
    var partneri = query
        .Skip((page - 1) * pagesize)
        .Take(pagesize)
        .ToList();
    var model = new PartneriViewModel {
        Partneri = partneri,
        PagingInfo = pagingInfo
    };
    return View(model);
}
```


Proširenje upita redoslijedom sortiranja

9

► Primjer:  MVC \ Extensions \ Selectors \ PartnerSort.cs

```
public static IQueryable<ViewPartner> ApplySort(
    this IQueryable<ViewPartner> query, int sort, bool ascending) {

    Expression<Func<ViewPartner, object>> orderSelector = null;
    switch (sort) {
        case 1:
            orderSelector = p => p.IdPartnera;
            break;
        case 2:
            orderSelector = p => p.TipPartnera;
            break;
        ...
    }
    if (orderSelector != null)
        query = ascending ?
            query.OrderBy(orderSelector) :
            query.OrderByDescending(orderSelector);
    return query;
}
```

Prikaz svih partnera


10

➡ Primjer (po uzoru na prethodne)  MVC \ Views \ Partner \ Index.cshml

```
@model PartneriViewModel
...
@foreach (var partner in Model.Partneri) {
    <tr>
        <td class="text-left">@partner.IdPartnera</td>
        <td class="text-left">@partner.TipPartneraText</td>
        <td class="text-left">@partner.OIB</td>
        <td class="text-left">@partner.Naziv</td>
        <td>
            <a asp-action="Edit"
                asp-route-id="@partner.IdPartnera"
                asp-route-page="@Model.PagingInfo.CurrentPage"
                ... class="btn btn-warning btn-sm" title="Ažuriraj">
                <i class="fas fa-edit"></i></a>
        </td>
        <td>
            <form asp-action="Delete" method="post"
                asp-route-id="@oartner.IdPartnera">
```

Stvaranje objekta kao jedne od specijalizacija

11

- Za prijenos podataka između pogleda i upravljača za stvaranje novog partnera definiran je novi prezentacijski model koji sadrži sve attribute osobe, ali i tvrtke
 - Alternativa: razviti dvije odvojene akcije i dva različita pogleda
- Primjer:  MVC \ ViewModels \ PartnerViewModel.cs

```
public class PartnerViewModel {  
    public int IdPartnera { get; set; }  
    [RegularExpression("[OT]")]  
    public string TipPartnera { get; set; }  
    public string PrezimeOsobe { get; set; }  
    public string ImeOsobe { get; set; }  
    public string MatBrTvrtke { get; set; }  
    public string NazivTvrtke { get; set; }  
    [Required]  
    [RegularExpression("[0-9]{11}")]  
    public string Oib { get; set; }  
    public string AdrPartnera { get; set; }  
    public int? IdMjestaPartnera { get; set; }  
    public string NazMjestaPartnera { get; set; }  
}
```

Priprema za unos novog partnera

12

➡ Inicijalno postavljeno da se radi o osobi, ali moguće promijeniti prije samog unosa

➡ Primjer:  MVC \ Controllers \ PartnerController.cs

```
[HttpGet]
public IActionResult Create()
{
    PartnerViewModel model = new PartnerViewModel
    {
        TipPartnera = "O"
    };
    return View(model);
}
```

Odabir tipa partnera (1)

13

➤ Odabir se vrši korištenjem *radiobuttona*


- *radiobutton* ima naziv generiran na osnovi *tag-helpera asp-for* i naziva odgovarajućeg svojstva iz modela te pridruženu vrijednost
- tekst se neovisno navodi ispred ili iza

➤ Primjer:  MVC \ Views \ Partner \ Create.cshtml

```
@model PartnerViewModel
...
<form asp-action="Create" method="post">
  <label class="radio-inline">
    <input type="radio" asp-for="TipPartnera" value="O">Osoba</label>
  <label class="radio-inline">
    <input type="radio" asp-for="TipPartnera" value="T">
    Tvrtka</label>
```

Odabir tipa partnera (2)

14

- Dio kontrola treba prikazati samo ako se unosi nova osoba, odnosno ako se unosi nova tvrtka.
 - Bit će skriveno/otkriveno korištenjem JavaScripta
 - Kontrole pronalazimo po odgovarajućem stilu (nema svoje vizualne osobine, već služi za pronalazak takvih kontrola)
 - Primjer:  MVC \ Views \ Partner \ Create.cshtml

```
@model PartnerViewModel
...
<form asp-action="Create" method="post">
  ...
  <div class="form-group samotvrtka">
    <label asp-for="MatBrTvrtke"></label>
    <input asp-for="MatBrTvrtke" class="form-control" />
  </div>
  <div class="form-group samoosoba">
    <label asp-for="ImeOsobe"></label>
    <input asp-for="ImeOsobe" class="form-control" />
  </div>
```

Odabir tipa partnera (3)

15

- Nakon učitavanja stranice te pri svakoj promjeni označenog radiobuttona skrivaju se odnosno prikazuje odgovarajuće kontrole

➤ Primjer:  MVC \ Views \ Partner \ Create.cshtml

```
@section scripts{
    <script type="text/javascript">
        $(function () {
            $('input:radio').change(function () {
                OsobaIliTvrтка($(this).val());
            });
            OsobaIliTvrтка($('input:checked').val());
        });
        function OsobaIliTvrтка(tip) {
            if (tip == 'O') {
                $(".samotvrтка").hide(); $(".samoosoba").show();
            }
            else {
                $(".samoosoba").hide(); $(".samotvrтка").show();
            }
        }
    </script>
}
```


Validacija prilikom unosa novog partnera (1)

16

- Model *PartnerViewPartner* sadrži podatke i za osobu i za tvrtku
- Atribut *Required* na imenu osobe (ili *RuleFor(p => p.ImeOsobe).NotEmpty()* koristeći *FluentValidation*) nema smisla ako je partner tvrtka
 - Štoviše morao bi se popuniti nekim besmislenim podatkom da bi se forma mogla poslati na server
- Potrebno napraviti dodatnu vlastitu provjeru
 - Može se napisati vlastiti atribut
 - Moguće koristiti *FluentValidation* i kombinirati metode *DependentRules*, *Must* i/ili *Custom*
 - U oba slučaja moguće (iako ne baš jednostavno) moguće napraviti i javascript za klijentsku validaciju
 - Implementirati sučelje *IValidatableObject*
 - Validacija samo na serveru, ali prihvatljivo za primjer

Validacija prilikom unosa novog partnera (2)


17

- Implementirati sučelje *IValidatableObject*
 - Vraća enumeraciju s opisom pogrešaka (objekti tipa *ValidationResult*)
 - Validacija se odvija samo na serveru
- Primjer:  MVC \ ViewModels \ PartnerViewModel.cs

```
public class PartnerViewModel : IValidatableObject {  
    public IEnumerable<ValidationResult>  
        Validate(ValidationContext validationContext) {  
        ...  
  
        ValidationResult("Potrebno je upisati ime osobe", new [] {nameof(ImeOsobe) }  
        );  
        ...  
    }  
}
```

Validacija prilikom unosa novog partnera (3)

18

- Primjer:  MVC \ ViewModels \ PartnerViewModel.cs
- yield return vraća jedan po jedan rezultat
 - Može se koristiti ako je povratni tip IEnumerable ili IEnumerator

```
public IEnumerable<ValidationResult>
    Validate(ValidationContext validationContext) {
    if (TipPartnera == "O") {
        if (string.IsNullOrEmpty(ImeOsobe))
            yield return new ValidationResult(
                "Potrebno je upisati ime osobe",
                new [] { nameof(ImeOsobe) } );
        if (string.IsNullOrEmpty(PrezimeOsobe))
            yield return new ValidationResult(
                "Potrebno je upisati prezime osobe",
                new [] { nameof(PrezimeOsobe) } );
    }
    ...
}
```

Unos novog partnera (1)

19

➡ Potrebno stvoriti novi objekt tipa Partner i inicijalizirati mu svojstvo Osoba ili Tvrtka

➡ Primjer:  MVC \ Controllers \ PartnerController.cs

```
public async Task<IActionResult> Create(PartnerViewModel model) {  
    if (ModelState.IsValid) {  
        Partner p = new Partner();  
        p.TipPartnera = model.TipPartnera;  
        CopyValues(p, model); //kopiraj podatke iz modela u p  
        try  
        {  
            ctx.Add(p);  
            await ctx.SaveChangesAsync();  
            ...  
        }  
    }  
}
```

Unos novog partnera (2)

20

➡ Kopiraju se potrebna svojstva te stvara nova instanca Osobe ili Tvrtke

➡ Primjer:  MVC \ Controllers \ MjestoController.cs

```
void CopyValues(Partner partner, PartnerViewModel model) {  
    partner.AdrIsporuke = model.AdrIsporuke;  
    partner.AdrPartnera = model.AdrPartnera;  
    partner.IdMjestaIsporuke = model.IdMjestaIsporuke;  
    partner.IdMjestaPartnera = model.IdMjestaPartnera;  
    partner.Oib = model.Oib;  
    if (partner.TipPartnera == "O") {  
        partner.Osoba = new Osoba();  
        partner.Osoba.ImeOsobe = model.ImeOsobe;  
        partner.Osoba.PrezimeOsobe = model.PrezimeOsobe;  
    }  
    else {  
        partner.Tvrtka = new Tvrtka();  
        partner.Tvrtka.MatBrTvrtke = model.MatBrTvrtke;  
        partner.Tvrtka.NazivTvrtke = model.NazivTvrtke;  
    }  
}
```

Strani ključ i *identity* tip podatka

21

- U postupku CopyValues stvoren novi objekt tipa Osoba ili Tvrtka

➤ Primjer:  MVC \ Controllers \ PartnerController.cs

```
public IActionResult Create(PartnerViewModel model) {  
    ...  
    Partner p = new Partner();  
    ...  
    //CopyValues: p.Osoba = new Osoba();  
    ...  
    ctx.Add(p);  
    ctx.SaveChanges();  
}
```

- EF će automatski stvoriti odgovarajuće dvije *Insert* naredbe
- Uz prvu Insert naredbu EF izvršava i upit za dohvat *identity* vrijednosti primarnog ključa koja se koristi kao primarni i strani ključ za tablicu Osoba odnosno Tvrtka.

Nadopunjavanje umjesto padajuće liste (1)

22

- Izbor mjesta partnera
 - Velik broj mogućih mjesta – nije prikladno za padajuću listu
- Koristi se nadopunjavanje (engl. *autocomplete*), odnosno dinamička padajuća lista
 - U pogledu se definira obično polje za unos kojem se pridružuje klijentski kod koji poziva određenu stranicu na serveru koja vraća tražene podatke osnovi trenutno upisanog teksta
 - Npr. za tekst `breg` stranica će vratiti sva mjesta koja u nazivu sadrže riječ `breg` (npr. Bregana, Lugarski Breg, Bregi, ...)
 - Rezultat ovisi o postupku na serveru
 - Podaci koje stranica vraća bit će parovi oblika (identifikator, oznaka)
 - npr. `5489, "21000 Split"`
 - Upisani tekst predstavljat će naziv mjesta, a dohvaćeni identifikator mjesta će se pohraniti u skriveno polje: podaci su parovi oblika (identifikator, tekst)

Razred za pohranu rezultata servisa (1)

23

➡ Podaci za dinamičku padajuću listu (nadopunjavanje) imaju identifikatora i tekst

➡ Primjer:  MVC \ Controllers \ AutoComplete \ IdLabel.cs

```
public class IdLabel {  
    public string Label { get; set; }  
    public int Id { get; set; }  
    public IdLabel() { }  
    public IdLabel(int id, string label) {  
        Id = id;  
        Label = label;  
    }  
}
```

➡ Pretvorbom u JSON nastat će rezultat nalik sljedećem tekstu:

```
[{"label":"42000 Varaždin","id":6245}, {"label":"42204 Varaždin  
Breg","id":6246}, {"label":"42223 Varaždinske Toplice","id":6247}]
```

pri čemu nije sigurno hoće li nazivi svojstava biti zapisani velikim ili malim slovom

Razred za pohranu rezultata servisa (2)

24

- Za izbjegavanje nedoumica i potencijalnih problema kod pretvorbe u/iz JSON-a, koristi se atribut *JsonPropertyName*

➤ Primjer:  MVC \ ViewModels \ IdLabel.cs


```
public class IdLabel {  
    [JsonPropertyName("label")]  
    public string Label { get; set; }  
    [JsonPropertyName("id")]  
    public int Id { get; set; }  
    ...  
}
```

- Alternativno u Startup.cs se može postaviti PropertyNamingPolicy na JsonNamingPolicy.CamelCase ili null (ako se ne želi camel-case)

```
services.AddControllersWithViews()  
    .AddJsonOptions(configure =>  
        configure.JsonSerializerOptions.PropertyNamingPolicy = ...)
```


Upravljač za dohvat podataka za nadopunjavanje

25

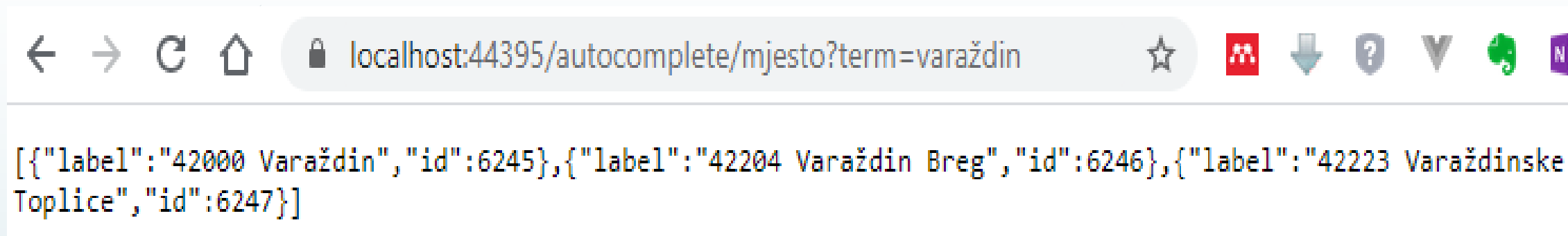
- Postupak ne vraća pogled, već enumeraciju parova (id, oznaka)
 - ASP.NET Core automatski pretvara u JSON (ili xml ovisno o postavkama)
 - traži se podniz – ulazni argument se mora zvati *term*
 - u čemu tražimo podniz? uključujemo li i pretragu po poštanskog broja
 - projekcija iz skupa entiteta *Mjesto* u listu objekata tipa *IdLabel*
 - Primjer:  MVC \ Controllers \ AutoCompleteController.cs (*Mjesto*)

```
public async Task<IEnumerable<IdLabel>> Mjesto(string term) {  
    var query = ctx.Mjesto  
        .Select(m => new IdLabel {  
            Id = m.IdMjesta,  
            Label = m.PostBrMjesta + " " + m.NazMjesta  
        })  
        .Where(l => l.Label.Contains(term));  
    var list = await query.OrderBy(l => l.Label)  
        .Take(appData.AutoCompleteCount)  
        .ToListAsync();  
    return list;  
}
```

Nadopunjavanje umjesto padajuće liste (2)

26


- ➡ Upravljač i akciju možemo isprobati direktnim pozivom u pregledniku



- ➡ Na klijentskoj strani koristit će se autocomplete iz *jQuery UI*
 - ➡ To je razlog zašto se argument morao zvati *term*
- ➡ Kako prepoznati kontrole kojima treba pridružiti dinamičke padajuće liste i gdje pohraniti identifikator?
 - ➡ Te informacije bit će zapisane u *data* attribute oblika *data-naziv*
 - ➡ Tekstualno polje za unos teksta i (uobičajeno skrivena) kontrola za pohranu vrijednosti odabira (*id*)

Priprema i označavanje kontrola za unos

27

- *autocomplete* ćemo aktivirati na svim kontrolama koje imaju atribut *data-autocomplete*, a vrijednost atributa će biti naziv akcije u upravljaču *AutoComplete*
 - Koristiti ćemo i atribut *data-autocomplete-placeholder-name* za prepoznati (skrivenu) kontrolu u koju ćemo pohraniti odabir
 - Ta kontrola mora definirati atribut *data-autocomplete-placeholder* čija vrijednost odgovara onoj iz *data-autocomplete-placeholder-name*
 - Primjer:  MVC \ Views \ Partner \ Create.cshml

```
<label asp-for="IdMjestaPartnera"></label>
<input class="form-control" asp-for="NazMjestaPartnera"
      data-autocomplete="mjesto"
      data-autocomplete-placeholder-name="mjestopartnera"
      value="@Model.NazMjestaPartnera" />
<input type="hidden" asp-for="IdMjestaPartnera"
      data-autocomplete-placeholder="mjestopartnera" />
```

Javascript biblioteke za nadopunjavanje

28

➡ jQuery UI (dodati u libman.json) + vlastita skripta

➡ Ne koristi se svugdje, uključuje se po potrebi

➡ Primjer:  MVC \ Views \ Partner \ Create.cshml

```
@section styles{
    <link rel="stylesheet" href="~/lib/jqueryui/themes/base/jquery-
ui.css" /> }
@section scripts{
    <script src="~/lib/jqueryui/jquery-ui.js"></script>
    <script src="~/js/autocomplete.js"></script>
```


➡ Potrebno znati putanju do naše aplikacije (može biti npr. oblika (<https://server/apps/my-app> pa trebamo znati korijen, npr. apps)

➡ Primjer:  MVC \ Views \ Shared \ _Layout.cshml

```
<script>
    window.applicationBaseUrl = '@Url.Content("~/")';
</script>
```

Aktiviranje nadopunjavanja (1)

29

- Primjer:  MVC \ wwwroot \ js \ autocomplete.js
 - Za svaki element koji ima definiran vlastiti atribut data-autocomplete:
 - dohvati relativnu adresu izvora podataka (iz data-autocomplete)
 - dohvati naziv elementa kojim se prepoznaje kontrola za pohranu dohvaćene vrijednosti
 - Ako se ne navede (bit će praktično kasnije) koristi se naziv akcije
 - ...

```
$("#[data-autocomplete]").each(function (index, element) {  
    var action = $(element).data('autocomplete');  
    var resultplaceholder =  
        $(element).data('autocomplete-placeholder-name');  
    if (resultplaceholder === undefined)  
        resultplaceholder = action;  
    ...  
});
```

Aktiviranje nadopunjavanja (2)

30

➤ Primjer:  MVC \ wwwroot \ js \ autocomplete.js

➤ Za svaki element koji ima definiran vlastiti atribut data-autocomplete:

➤ ...

➤ Definiraj sljedeće ponašanje: *Ako se promijeni tekst, obriši pohranjeni identifikator*

➤ ...

```
$("#[data-autocomplete]").each(function (index, element) {  
    ...  
    $(element).change(function () {  
        var dest = $('[data-autocomplete-placeholder  
                                = '${resultplaceholder}']`);  
        var text = $(element).val();  
        if (text.length === 0 || text !== $(dest).data('selected-label')) {  
            $(dest).val('');  
        }  
    }); ...  
});
```

Aktiviranje nadopunjavanja (3)

31

► Primjer:  MVC \ wwwroot \ js \ autocomplete.js

► Za svaki element koji ima definiran vlastiti atribut data-autocomplete:

► ... Aktiviraj autocomplete (postavlja se adresa izvora podataka, minimalna potrebna duljina teksta za nadopunjavanje i slično)

► akcija koja će se izvršiti odabirom nekog elementa iz liste – u primjeru tekst će se kopirati u polje za unos, a identifikator u skriveno polje (vidi sljedeći slajd)

```
$("#[data-autocomplete]").each(function (index, element) {  
    ...  
    $(element).autocomplete({  
        source: window.applicationBaseUrl + "autocomplete/" + action,  
        autoFocus: true,  
        minLength: 1,  
        select: function (event, ui) {  
            ... Nakon što je odabrano nešto iz liste...  
        }  
    });  
});
```

Aktiviranje nadopunjavanja (4)

32


► Primjer:  MVC \ wwwroot \ js \ autocomplete.js

► Kad se nešto odabere iz padajuće liste, identifikator (vrijednost) se kopira na odredište (skriveno polje)

```
$( "[data-autocomplete]" ).each( function ( index, element ) {  
    ...  
    $( element ).autocomplete( {  
        ...  
        select: function ( event, ui ) {  
            $( element ).val( ui.item.label );  
            var dest = $( `[data-autocomplete-  
                placeholder='${resultplaceholder}']` );  
            $( dest ).val( ui.item.id );  
            $( dest ).data( 'selected-label', ui.item.label );  
        }  
    } );  
};
```


Pogreške prilikom dodavanja novog partnera


33

- Model može biti neispravan ili se može dogoditi pogreška prilikom snimanja
 - Prethodno povezani podaci su dio modela koji se vraćaju pogledu
 - Naziv mjesta je dio modela i koristi se asp-for na odgovarajućem polju, pa ga ne treba ponovo rekonstruirati
 - Primjer:  MVC \ Controllers \ PartnerController.cs

```
try {  
    ctx.Add(p);  
    await ctx.SaveChangesAsync();  
    ...  
}  
catch (Exception exc) {  
    ModelState.AddModelError(string.Empty, exc.CompleteExceptionMessage());  
    return View(model);  
    ...  
}
```

Dohvat podataka o partneru


34

- Kao model za pogled koristi se isti model kao kod dodavanja
 - Prvo se vrši dohvat zajedničkih podataka iz tablice Partner
 - Ostatak podataka puni se upitom na tablicu Osoba ili Tvrtka
 - Umjesto Find[Async] mogu se koristiti i varijante s Where
- Primjer:  Mvc \ Controllers \ PartnerController.cs

```
public async Task<IActionResult> Edit(int id, ...) {  
    var partner = ctx.Partner.FindAsync(id);  
    ...  
    PartnerViewModel model = new PartnerViewModel {  
        IdPartnera = partner.IdPartnera,  
        IdMjestaIsporuke = partner.IdMjestaIsporuke,  
        ...  
    };  
    if (model.TipPartnera == "O") {  
        Osoba osoba = ctx.Osoba.Find(model.IdPartnera);  
        model.ImeOsobe = osoba.ImeOsobe;  
        ...  
    }
```

Ažuriranje podataka o partneru (1)


35

- Podaci povezani kroz model se provjeravaju na validacijske pogreške (slično kao kod *Create*)
 - Ako je model ispravan, vrši se dohvat partnera iz BP te se (vlastitim postupkom) kopiraju vrijednosti iz primljenog modela u entitet iz EF
 - Primijetiti da se ne radi Include na Osoba ili Tvrtka
 - više na slajdovima koji slijede
 - Primjer:  Mvc \ Controllers \ PartnerController.cs

```
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Edit(PartnerViewModel model...) {
    var partner = ctx.Partner.Find(model.IdPartnera);
    ...
    ValidateModel(model);
    if (ModelState.IsValid)
    {
        try
        {
            CopyValues(partner, model);
        }
    }
}
```

Ažuriranje podataka o partneru (2)

36

- Mijenjaju se sva svojstva osobe ili tvrtke
 - prilikom dohvata partnera nije uključen i dohvat podataka o osobi ili tvrtki
 - stoga je svojstvo Osoba (Tvrtka) jednako null te se instancira novi objekt
- Primjer:  MVC \ Controllers \ MjestoController.cs

```
void CopyValues(Partner partner, PartnerViewModel model) {  
    partner.AdrIsporuke = model.AdrIsporuke;  
    partner.AdrPartnera = model.AdrPartnera;  
    partner.IdMjestaIsporuke = model.IdMjestaIsporuke;  
    partner.IdMjestaPartnera = model.IdMjestaPartnera;  
    partner.Oib = model.Oib;  
    if (partner.TipPartnera == "O") {  
        partner.Osoba = new Osoba();  
        partner.Osoba.ImeOsobe = model.ImeOsobe;  
        partner.Osoba.PrezimeOsobe = model.PrezimeOsobe;  
    }  
    else {  
        partner.Tvrtka = new Tvrtka();  
        partner.Tvrtka.MatBrTvrtke = model.MatBrTvrtke;  
        partner.Tvrtka.NazivTvrtke = model.NazivTvrtke; ...  
    }  
}
```

Snimanje promjena


37

- Budući da je povezani dio za osobu ili tvrtku nastao stvaranjem novog objekta, a ne ažuriranjem onog dohvaćenog iz BP, EF ga smatra novim objektom (insert upit)
 - Eksplicitno mijenjamo stanje tog objekta iz *Added* u *Modified* i postavljamo vrijednost *PK* => uzrokuje *update* upit, a ne *insert*
 - Primjer:  MVC \ Controllers \ MjestoController.cs

```
public async Task<IActionResult> Edit(PartnerViewModel model, ... {  
    ...  
    var partner = await ctx.Partner.FindAsync(model.IdPartnera);  
    ...  
    CopyValues(partner, model);  
    if (partner.Osoba != null) {  
        partner.Osoba.IdOsobe = partner.IdPartnera;  
        ctx.Entry(partner.Osoba).State = EntityState.Modified;  
    }  
    if (partner.Tvrtka != null) {  
        partner.Tvrtka.IdTvrtke = partner.IdPartnera;  
        ctx.Entry(partner.Tvrtka).State = EntityState.Modified;  
    }  
    await ctx.SaveChangesAsync();  
}
```

Brisanje partnera

38

- Definirano kaskadno brisanje u BP.
 - Prilikom generiranja EF modela ta je činjenica uzeta u obzir
- Dovoljno obrisati se entitet iz skupa Partner.
 - Odgovarajući zapis iz tablice Osoba ili Tvrtka se automatski briše
- Dohvat se može izvršiti s *Where* ili postupkom *Find* navođenjem vrijednosti primarnog ključa
- Primjer:  MVC \ Controllers \ MjestoController.cs

```
public async Task<IActionResult> Delete(int id,...) {  
    var partner = await ctx.Partner.FindAsync(id);  
    if (partner != null) {  
        try {  
            ctx.Remove(partner);  
            await ctx.SaveChangesAsync();  
        }  
    }  
    ...  
}
```

- Umjesto `ctx.Remove` moglo se napisati i
`ctx.Entry(partner).State = EntityState.Deleted;`

Primjer zaglavlje-stavke (engl. master-detail)

1. dio: Prikaz svih dokumenata, filtriranje dokumenata, pojedinačni prikaz

Pogled za prikaz svih dokumenata


40

- ➡ 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)

41

- ➡ 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)

42

➡ U definiciju konteksta dodati novi *DbSet* za pogled

➡ Primjer:  MVC \ ModelsPartial \ FirmaContext.cs \ 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

43

- Popis dokumenata moguće filtrirati po partneru, iznosu ili datumu.
- Za odabir partnera koristi se nadopunjavanje, a za odabir datuma prikazuje se kalendar
 - Ovisno o pregledniku, 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 Indeks bez parametara
- Kriterij pretrage parcijalni pogled uključen u pogled Indeks

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

IdPartnera 0 Zadovoljni korisnik (01234567988)

Iznos 300 - 500

Datum 01.05.2015. - 10.05.2017. x

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

Datum dokumenta 12.01.2014. 1.803.667,82 kn

Parametar ili sjednica?

44

- 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 filteru (1)

45

➡ 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 {  
    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)

46

➡ 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();  
        var arr = s.Split(new char[] { '-' }, 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)

47

➡ 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 dinamičke padajuće liste

48

➡ 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

49


➡ 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 queryPartneri = ctx.Tvrtka.Select(t => new IdLabel {
    Id = t.IdTvrtke,
    Label = t.NazivTvrtke + ", " +
        " (" + t.IdTvrtkeNavigation.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

50


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

Identifikator i naziv partnera u filtru

51

- 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)

52

➤ Prezentacijski model s validacijskim atributima

➤ 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)

53

- 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

54

➡ 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)

55

- Prvo dohvat dokumenta, a potom i njegovih stavki
 - Podaci o prethodnom dokumentu i partneru se dohvaćaju naknadno 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)

56


➡ 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 (3)

57

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

Određivanje sljedbenika i prethodnika (1)


58

- 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)

59

- Za prikaz dokumenta potrebno imati njegov id, a ne samo poziciju
- Formira se upit s istim filterom (sort je nebitan) 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 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();
}
```