

# Web-aplikacije

## ASP.NET Core MVC

2019/20.12

Specijalizacija i generalizacija

Nadopunjavanje umjesto padajuće liste

# Rad sa specijalizacijama i generalizacijama

# Problem prikaza specijalizacija nekog entiteta

3

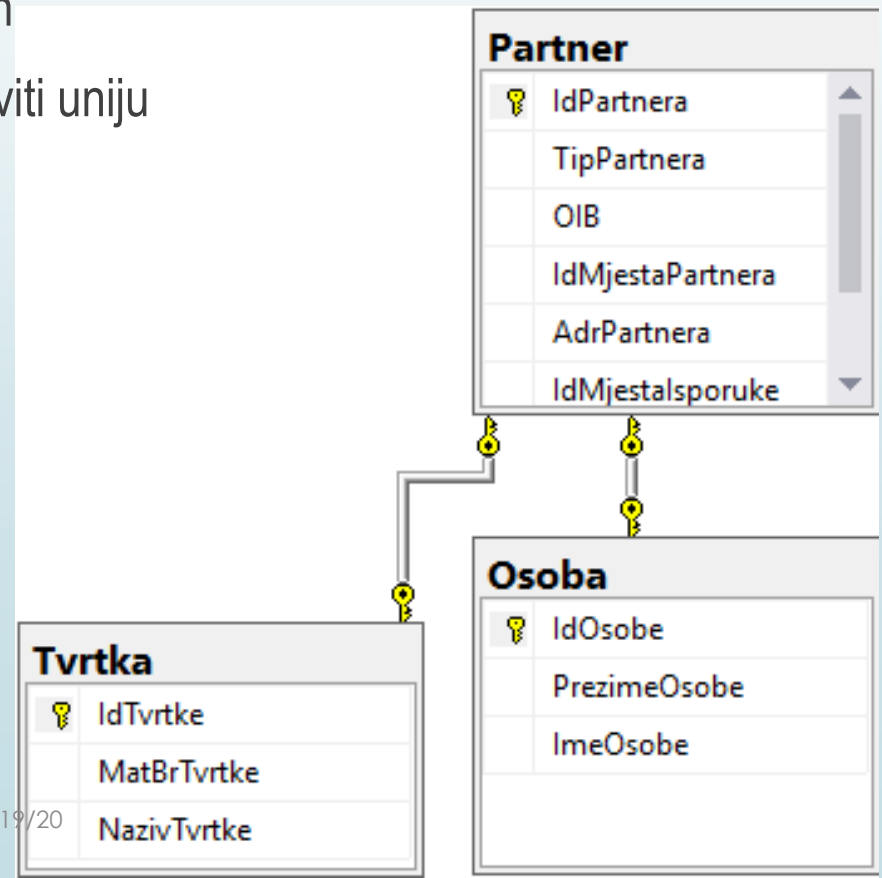
- U oglednom modelu Osoba i Tvrtka su specijalizacije Partnera te ih EF Core preslikava u 3 tablice
- Što ako u prikazu svih partnera želimo ispisati id partnera, vrstu partnera, OIB i naziv partnera?
  - Upit korištenjem EF-a se komplicira i postaje neefikasan
  - Potrebno dohvatiti sve osobe, zatim sve tvrtke te napraviti uniju
  - Što ako treba sortirati podatke?

## Popis partnera

Unos novog partnera

1.. 20 21 22 23 24 25 26 27 28 29 30

Id partnera	Tip partnera	OIB	Naziv		
677	Tvrtka	3822025	ISOT		
678	Tvrtka	3818116	ISVU		
679	Tvrtka	3814209	ITI		
969	Osoba	01234567890	Ivić, Ive		
680	Tvrtka	3810304	JMBG		



# 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:  Firma.Mvc \ Models \ 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


- U kontekst dodati *DbSet* razreda koji odgovara pogledu
  - naziv svojstva obično odgovara nazivu pogleda, ali nije nužno
    - Može se navesti u `CollectionView("naziv pogleda")`
  - također, može se i navesti SQL upit koji vraća rezultat traženog tipa, npr.  
`ctx.vw_Partner.FromSqlRaw("SELECT * FROM vw_Partner");`
- Primjer:  `Firma.Mvc \ Models \ FirmaContext.cs`

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

- Ovako kreirani *DbSet* koristi se kao i drugi *DbSet*ovi, ali samo za čitanje

# 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:  Web \ Firma.Mvc \ ViewModels \ PartneriViewModel.cs

```
namespace Firma.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:  Web \ Firma.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:  Web \ Firma.Mvc \ Controllers \ PartnerController.cs

```
public IActionResult Index(string filter, int page = 1,
                           int sort = 1, bool ascending = true) {
    ...
    System.Linq.Expressions.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);
    }
    ...
}
```

# Prikaz svih partnera


10

➡ Primjer (po uzoru na prethodne)  Firma.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-page="@Model.PagingInfo.CurrentPage" ...>
                <input type="hidden" name="IdPartnera" value="@partner.IdPartnera" />
                <button type="submit" title="Obriši">...
```

# 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:  Firma.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:  Firma.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:  Firma.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 izvršeno korištenjem *jQuerya*, ali je potrebno takve kontrole označiti odgovarajućim imenima ili stilovima
    - U primjeru se koristi stil koji nema svoje vizualne osobine već služi samo za pronalazak takvih kontrola

➤ Primjer:  Firma.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:  Firma.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 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 provjeru vlastitim programskim kodom
  - Moguće napisati vlastiti validacijski atribut ili implementirati sučelje *IValidatableObject*
  - Vraća enumeraciju s opisom pogrešaka (objekti tipa *ValidationResult*)
  - Validacija se odvija samo na serveru!
- Primjer:  Web \ Firma.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 (2)

17

- Primjer:  Web \ Firma.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)

18

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

➡ Primjer:  Web \ Firma.Mvc \ Controllers \ PartnerController.cs

```
public IActionResult Create(PartnerViewModel model) {  
    ValidateModel(model);  
    if (ModelState.IsValid) {  
        Partner p = new Partner();  
        p.TipPartnera = model.TipPartnera;  
        CopyValues(p, model); //kopiraj podatke iz model u p  
    }  
}
```

## Unos novog partnera (2)

19

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

➡ Primjer:  Firma.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

20

- U postupku CopyValues stvoren novi objekt tipa Osoba ili Tvrtka

➤ Primjer:  Web \ Firma.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

21

- 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 (id, tekst)
- Kako prepoznati kontrole kojima treba pridružiti dinamičke padajuće liste i gdje pohraniti identifikator?
  - Te informacije bit će zapisane u *data* attribute oblika *data-naziv*

# Razred za pohranu rezultata servisa (1)

22

- Podaci za dinamičku padajuću listu (nadopunjavanje) sastoje se od identifikatora i oznake

➤ Primjer:  Firma.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}]
```

# Razred za pohranu rezultata servisa


23

- Za izbjegavanje nedoumica i potencijalnih problema kod serijalizacije i deserijalizacije u JSON, eksplicitno se navode nazivi korištenjem atributa `JsonPropertyName`
  - Primjer:  Firma.Mvc \ Areas \ AutoComplete \ Models \ IdLabel.cs

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

# Upravljač za dohvat podataka za nadopunjavanje

24


- Upravljač ima postupak koji ne vraća pogled, već enumeraciju parova id, oznaka
  - traži se podniz – ulazni argument se mora zvati *term*
  - projekcija iz skupa entiteta *Mjesto* u listu objekata tipa *Label*
- Primjer:  Firma.Mvc \ Controllers \ AutoComplete \ MjestoController.cs

```
public IEnumerable<IdLabel> Get(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 = query.OrderBy(l => l.Label)  
        .Take(appData.AutoCompleteCount)  
        .ToList();  
  
    return list;  
}
```



# Usmjeravanje do upravljača

25

- Upravljač s prethodnog slajda ne slijedi uobičajenu putanju već definira vlastitu atributom *Route*
  - Atribut *HttpGet* označava postupak koji će se izvršiti pozivom oblika *autocomplete/Mjesto*
- Primjer:  Firma.Mvc \ Controllers \ AutoComplete \ MjestoController.cs

```
[Route("autocomplete/[controller]")]
public class MjestoController : Controller


    [HttpGet]
    public IEnumerable<IdLabel> Get(string term) {
        ...
    }
}
```

← → ↻ ⓘ localhost:50051/autocomplete/Mjesto?term=Varaždin

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

# Priprema i označavanje kontrola za unos

26

- U pogledu definirano obično polje za unos kojem se naknadno pridružuje klijentski kod
  - Upisani tekst predstavljat će naziv mjesta, a dohvaćeni identifikator mjesta će se pohraniti u skriveno polje: servis vraća parove oblika (id, tekst)
  - *data-autocomplete* sadrži relativnu adresu servisa
  - s *data-autocomplete-placeholder-name* označena vrijednost koja će se tražiti u atributu *data-autocomplete-placeholder* skrivenog unosa
- Primjer:  Web \ Firma.Mvc \ Views \ Create.cshml

```
<label asp-for="IdMjestaPartnera"></label>
<input class="form-control"
  data-autocomplete="mjesto"
  data-autocomplete-placeholder-name="mjestopartnera"
  value="@Model.NazMjestaPartnera" />
<input type="hidden" asp-for="IdMjestaPartnera"
  data-autocomplete-placeholder="mjestopartnera" />
...
@section scripts{
  <script src="~/lib/jquery-ui/jquery-ui.js"></script>
  <script src="~/js/autocomplete.js"></script>
```

# Aktiviranje nadopunjavanja (1)

27

➡ Primjer:  Web \ Firma.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 u koji se posprema dohvaćena vrijednost
- ➡ brisanjem teksta i promjenom fokusa izbriši staru vrijednosti

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

# Aktiviranje nadopunjavanja (2)

28

## ➤ Koristi se *jQuery autocomplete*

- postavlja se adresa izvora podataka i minimalna potrebna duljina teksta za nadopunjavanje
- 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

## ➤ Primjer: Web \ Firma.Mvc \ wwwroot \ js \ autocomplete.js

```
... aktiviraj autocomplete na elementu ...
$(element).autocomplete({
  source: "/autocomplete/" + url,
  autoFocus: true,
  minLength: 1,
  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

29

- Model može biti neispravan ili se može dogoditi pogreška
  - Prethodno povezani podaci su dio modela koji se vraćaju pogledu
  - Dodatno se vrši dohvat naziva mjesta na osnovu identifikatora mjesta
    - povezani u *model.IdMjestalsporuke* i *model.IdMjestaPartnera*
    - potrebno u slučaju da je korisnik promijenio naziv mjesta, a nije odabrao neko mjesto iz padajuće liste
- Primjer:  Firma.Mvc \ Controllers \ MjestoController.cs

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

# Dohvat podataka o partneru


30

➤ 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 mogu se koristiti i varijante s Where

➤ Primjer:  Firma.Mvc \ Controllers \ MjestoController.cs

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

# Ažuriranje podataka o partneru (1)

31


- Podaci povezani kroz model se provjeravaju na validacijske pogreške (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:  Firma.Mvc \ Controllers \ MjestoController.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)

32

- 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:  Firma.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


33

- ➡ Budući da je povezan 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:  Firma.Mvc \ Controllers \ MjestoController.cs

```
public IActionResult Edit(PartnerViewModel model, ... {  
    ...  
    var partner = ctx.Partner.Find(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;  
    }  
    ctx.SaveChanges();  
}
```

# Brisanje partnera

34

- 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:  Firma.Mvc \ Controllers \ MjestoController.cs

```
public IActionResult Delete(int IdPartnera, ...) {  
    var partner = ctx.Partner.Find(IdPartnera);  
    if (partner != null) {  
        try {  
            ctx.Remove(partner);  
            ctx.SaveChanges();  
        }  
    }  
    ...  
}
```

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