

# Web-aplikacije

## ASP.NET Core MVC

2016/17.10

Padajuće liste za odabir povezanih vrijednosti

Slanje datoteke na server. Prikaz slike artikla.

Specijalizacija i generalizacija

Nadopunjavanje umjesto padajuće liste

# Primjer unosa objekta s ograničenim skupom vrijednosti za neko svojstvo

2

- Mjesto ima strani ključ na tablicu Država
- Umjesto unosa šifre države omogućiti korisniku da odabere državu iz popisa država.
- Popis država nije dio modela, već se prenosi koristeći *ViewBag* ili *ViewData*

Države

**Mjesta**

Artikli

Partneri

Dokumenti

## Unos novog mjesta

Poštanski broj mjesta

Naziv mjesta

Poštanski naziv mjesta


Država

Odaberite državu

- Bahamas
- Bahrain
- Bangladesh
- Barbados
- Belarus
- Belgium
- Belize
- Benin
- Bermuda
- Bhutan
- Bolivia
- Bosnia and Herzegovina**
- Botswana

# Priprema podataka za padajuću listu

3

- Na osnovu liste država stvara se objekt tipa *SelectList*
  - Navodi se izvor podataka, svojstvo koje predstavlja vrijednost odabranog elementa i svojstvo koje se prikazuje kao tekst u padajućoj listi
  - Stvoreni objekt se pogledu prenosi koristeći ViewBag (ili ViewData)
    - Može se upotrijebiti proizvoljno ime za novu vrijednost u ViewBagu
- Primjer:  Web \ Firma.Mvc \ Controllers \ MjestoController.cs


```
[HttpGet]
public IActionResult Create() {
    PrepareDropDownLists();
    return View();
}

private void PrepareDropDownLists() {
    var drzave = ctx.Drzava.AsNoTracking().OrderBy(d => d.NazDrzave)
        .Select(d => new { d.NazDrzave, d.OznDrzave })
        .ToList();

    ViewBag.Drzave = new SelectList(drzave,
        nameof(Drzava.OznDrzave), nameof(Drzava.NazDrzave)) ;
}
```

# Prikaz padajuće liste u pogledu

4

- Padajuća lista se koristi unutar HTML oznake *select*, pri čemu se izvor navodi atributom *asp-items*
  - Potrebno i navesti *asp-for* za određivanje svojstva kojem će se odabir pridružiti
- Moguće umetnuti i dodatne elemente u padajuću listu (osim onih koji su već pripremljeni).
  - Dodaju se na početak
  - Npr. poruka da se odabere neki element iz liste koja je inicijalno odabrana
    - Nakon što se jednom promijeni vrijednost (zbog atributa *disabled*) više se neće moći ponovo odabrati element s porukom
- Primjer:  Web \ Firma.Mvc \ Views \ Mjesto \ Create.cshtml

```
<select class="form-control" asp-for="OznDrzave"
        asp-items="ViewBag.Drzave">
    <option disabled selected value="">
        Odaberite državu
    </option>
</select>
```

# Ponovna priprema podataka za padajuću listu

5

- U slučaju neispravnih ili nepotpunih podataka potrebno je ponovno prikazati pogled za unos, ali i pripremiti podatke za padajuću listu

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

```
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Create(Mjesto mjesto) {
    if (ModelState.IsValid) {
        try {
            ...
        }
        catch (Exception exc) {
            ...
            PrepareDropDownLists();
            return View(mjesto);
        }
    }
    else {
        PrepareDropDownLists();
        return View(mjesto);
    }
}
```

# Model za pregled svih mjesta

6

- Umjesto entiteta Mjesto za pojedinačno mjesto koristi se novi prezentacijski pogled

- Umjesto oznake države sadrži naziv države

- Primjer:  Web \ Firma.Mvc \ ViewModels \ MjestaViewModel.cs

```
public class MjestaViewModel {  
    public IEnumerable<MjestoViewModel> Mjesta { get; set; }  
    public PagingInfo PagingInfo { get; set; }  
}
```

- Primjer:  Web \ Firma.Mvc \ ViewModels \ MjestoViewModel.cs

```
public class MjestoViewModel {  
    public int IdMjesta { get; set; }  
    public int PostBrojMjesta { get; set; }  
    public string NazivMjesta { get; set; }  
    public string PostNazivMjesta { get; set; }  
    public string NazivDrzave { get; set; }  
}
```

# Pregled svih mjesta

7

➡ Kao i u primjeru s državama zaglavlje tablice sadrži poveznice za sortiranje po određenom stupcu u prikazu


➡ Izvedeno petljom i poljima koja sadrže nazive stupaca i parametre sortiranja

➡ Primjer:  Web \ Firma.Mvc \ Views \ Mjesto \ Index.cshtml

```
<table class="table table-striped"> <thead>
  <tr>
    @{
      string[] nazivi = { "Poštanski broj", "Naziv mjesta",
                          "Poštanski naziv mjesta", "Država" };
      for (int i = 1; i <= nazivi.Length; i++) {
        <th>
          <a asp-route-sort="@i" asp-route-page="@Model.PagingInfo.CurrentPage"
            asp-route-ascending="@ (Model.PagingInfo.Sort == i ?
              Model.PagingInfo.Ascending? false : true : true)">
            @nazivi[i - 1]
          </a>
        </th>
      }
    }
```

# Parcijalni pogled za prikaz mjesta (1)

8

- Prikaz pojedinog retka izveden parcijalnim pogledom
  - Parcijalni pogled je pogled koji ne koristi glavnu stranicu te se umeće u neke druge poglede
  - Može imati svoj model (u prikazanom primjeru je to pojedinačno mjesto)
- U pogledu *Index* poziva se parcijalni pogled *Row* te se njegov sadržaj (tj. rezultat izvršavanja) umetne u konačni rezultat
  - Poziv se obavlja pozivom postupka `Html.Partial`
- Primjer:  `Web \ Firma.Mvc \ Views \ Mjesto \ Index.cshtml`

```
<table class="table table-striped">
  ...
  <tbody>
    @foreach (var mjesto in Model.Mjesta)
    {
      @Html.Partial("Row", mjesto)
    }
  </tbody>
</table>
```



# Parcijalni pogled za prikaz mjesta (2)

9


► Primjer:  Web \ Firma.Mvc \ Views \ Mjesto \ Row.cshtml

► Model je tipa MjestoViewModel (sadrži naziv umjesto oznake države)

```
@model MjestoViewModel
<tr>
  <td class="text-left">@Model.PostBrojMjesta</td>
  <td class="text-left">@Model.NazivMjesta</td>
  <td class="text-left">@Model.PostNazivMjesta</td>
  <td class="text-left">@Model.NazivDrzave</td>
  <td>
    ...
  </td>
</tr>
```

# Forma za slanje datoteke


10

- Prilikom unosa novog artikla može se poslati datoteka sa slikom artikla
- Za unos se koristi HTML *input* kontrola tipa *file*
  - Naziv može biti proizvoljan, ali mora odgovarati argumentu u akciji upravljača
- Forma mora imati atribut *enctype* postavljen na *multipart/form-data*
- Primjer:  Web \ Firma.Mvc \ Views \ Artikl \ Create.cshtml

```
<form asp-action="Create" method="post" enctype="multipart/form-data">
...
<label asp-for="SlikaArtikla" class="col-sm-2 col-form-label"></label>
  <div class="col-sm-5">
    <input type="file" name="slika" />
  </div>
...
<button class="btn btn-primary" type="submit">Dodaj</button>
```

# Prihvat datoteke na upravljaču

11

- Postupak prima artikl stvoren na osnovu podataka iz forme i argument tipa *IFormFile*
  - Naziv argument odgovara vrijednosti atributa *name* iz kontrole za unos u pogledu
  - Ako je korisnik odabrao datoteku, njen sadržaj se može kopirati u *MemoryStream* nakon čega se može dobiti polje bajtova i pospremiti u entitet, odnosno u bazu podataka
- Primjer:  Web \ Firma.Mvc \ Controllers \ ArtiklController.cs

```
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Create(Artikl artikl, IFormFile slika) {
    ...
    if (ModelState.IsValid) {
        if (slika != null && slika.Length > 0) {
            using (MemoryStream stream = new MemoryStream()) {
                slika.CopyTo(stream);
                artikl.SlikaArtikla = stream.ToArray();
            }
        }
        ctx.Add(artikl);
        ctx.SaveChanges();
    }
}
```

# Prikaz svih artikala (1)

12

- ➡ Prikaz pojedinačnog artikla izveden u parcijalnom pogledu Row.cshtml koji se uključuje u Index.cshtml za svaki artikl

➡ Primjer:  Web \ Firma.Mvc \ Views \ Artikl \ Index.cshtml

```
@foreach (var artikl in Model.Artikli) {  
    @Html.Partial("Row", artikl)  
}
```


- ➡ Model za pojedinačni artikl je razred ArtikelViewModel

➡ Primjer:  Web \ Firma.Mvc \ ViewModels \ ArtikelViewModel.cs

```
public class ArtikelViewModel {  
    public int SifraArtikla { get; set; }  
    public string NazivArtikla { get; set; }  
    public string JedinicaMjere { get; set; }  
    public decimal CijenaArtikla { get; set; }  
    public bool Usluga { get; set; }  
    public string TekstArtikla { get; set; }  
    public bool ImaSliku { get; set; }  
    public int ImageHash { get; set; }  
}
```

## Prikaz svih artikala (2)


13

- Koristi se posebni razred umjesto razreda Artikal iz EF modela kako se ne bi prilikom dohvata svih artikala istovremeno preuzimale i sve njihove slike
  - Umjesto sadržaja slike, evidentira se postoji li slika i hashcode slike
    - Potrebno kako bi preglednik znao osvježiti sliku ako se u međuvremenu promijeni
    - U ovom primjeru umjesto izračunavanja hashcode koristi se veličina slike
- Primjer:  Web \ Firma.Mvc \ Controllers \ ArtikalController.cshtml

```
var artikli = query.Select(a => new ArtikalViewModel {  
    SifraArtikla = a.SifArtikla,  
    NazivArtikla = a.NazArtikla,  
    JedinicaMjere = a.JedMjere,  
    CijenaArtikla = a.CijArtikla,  
    Usluga = a.ZastUsluga,  
    TekstArtikla = a.TekstArtikla,  
    ImaSliku = a.SlikaArtikla != null,  
    ImageHash =  
        a.SlikaArtikla != null ? a.SlikaArtikla.Length : 0  
})
```

# Poveznica za prikaz slike


14

- Ako artikl koji se prikazuje u pojedinom retku ima sliku, stvara se HTML img kontrola
- Adresa slike je akcije *GetImage* na upravljaču *Artikl*
  - Adresi slike se dodaj parameter hash kako bi preglednik mogao prepoznati novu sliku artikla u odnosu na onu koju ima spremljenu u svojoj memoriji
- Primjer:  Web \ Firma.Mvc \ Views \ Artikl \ Row.cshtml

```
@if (Model.ImaSliku) {  
      
}
```

## Prikaz svih artikala (2)

15

- ➡ Polje bajtova koje predstavlja sliku artikla dohvati se EF upitom
- ➡ Rezultat postupka je *FileContentResult* koji nastane pozivom nasljeđenog postupka *File* iz upravljača.
- ➡ Primjer:  Web \ Firma.Mvc \ Controllers \ ArtiklController.cs

```
public FileContentResult GetImage(int id) {  
    byte[] image = ctx.Artikl  
        .Where(a => a.SifArtikla == id)  
        .Select(a => a.SlikaArtikla)  
        .SingleOrDefault();  
  
    if (image != null)  
        return File(image, "image/jpeg");  
    else  
        return null;  
}
```

- ➡ Napomena: Povratna vrijednost je mogla biti i *ActionResult*, pa je umjesto `return null` moglo pisati `return NotFound()`

# Rad sa specijalizacijama i generalizacijama



# Problem prikaza specijalizacija nekog entiteta











17

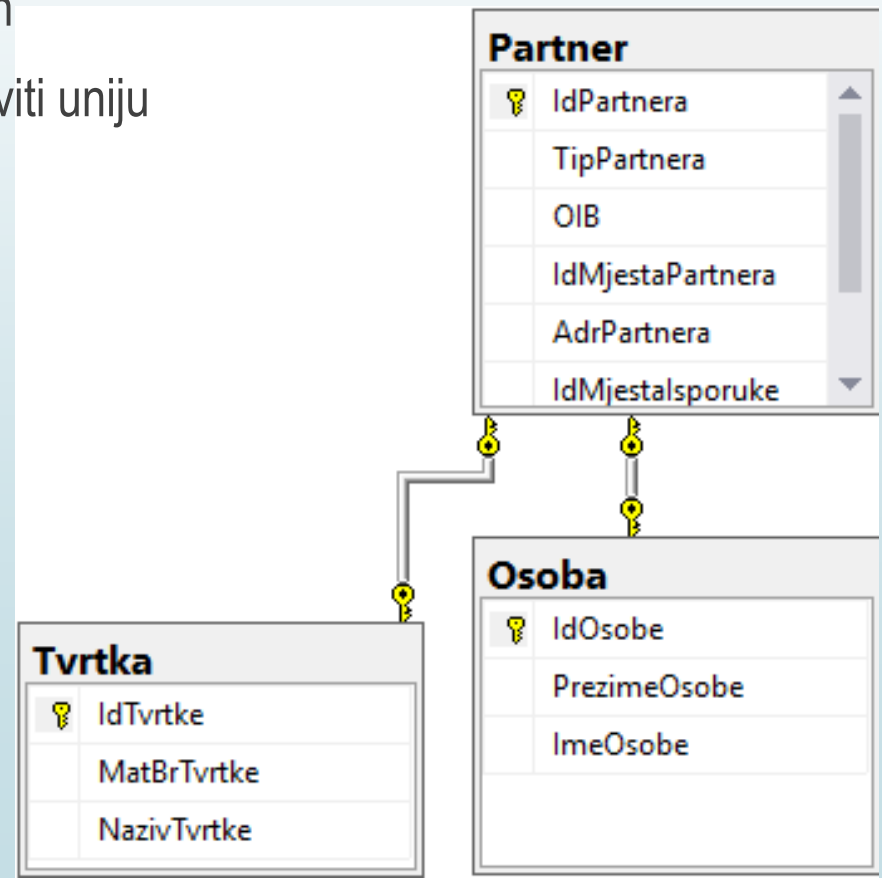
- U oglednom modelu Osoba i Tvrtka su specijalizacije Partnera te ih EF Core preslikava u 3 tablice
- Što ako u prikazu svih partnera želimo ispisati id partnera, vrstu partnera, OIB i naziv partnera?
  - Upit korištenjem EF-a se komplicira i postaje neefikasan
  - Potrebno dohvatiti sve osobe, zatim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

18

- ➡ 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)

19

➡ 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)

20

- U kontekst dodati *DbSet* koji je vezan uz pogled te definiciju dodanog entiteta pridružujući neko svojstvo koje jednoznačno određuje podatka iz pogleda
- Ako naziv skupa nije jednak nazivu pogleda tada je prilikom postavljanja upita potrebno pisati SQL upit koji bi vratio traženi rezultat

- npr. `ctx.vw_Partner.AsNoTracking()  
          .FromSql("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.HasKey(e => e.IdPartnera);  
    });  
    ...  
}
```

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

# Model za prikaz svih partnera

21

- (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

22

➡ 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.AsNoTracking();
    ...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

23

► 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

24


► Primjer (po uzoru na prethodne)  Firma.Mvc\Views\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-xs" title="Ažuriraj">
                <span class="glyphicon glyphicon-pencil"></span></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

25

- 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

26

➡ 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)

27

➤ 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)

28

- 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)

29

- 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

30

➡ Model *PartnerViewModel* sadrži podatke i za osobu i za tvrtku

➡ Atribut *Required* na imenu osobe nema smisla ako je partner tvrtka te je potrebno napraviti dodatnu provjeru vlastitim programskim kodom

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

```
private void ValidateModel(PartnerViewModel model) {  
    if (model.TipPartnera == "O") {  
        if (string.IsNullOrEmpty(model.ImeOsobe))  
            ModelState.AddModelError(nameof(model.ImeOsobe),  
                "Ime osoba ne smije biti prazno");  
        if (string.IsNullOrEmpty(model.PrezimeOsobe))  
            ModelState.AddModelError(nameof(model.PrezimeOsobe),  
                "Prezime osoba ne smije biti prazno");  
    }  
    else {  
        if (string.IsNullOrEmpty(model.MatBrTvrtke))  
            ModelState.AddModelError(nameof(model.MatBrTvrtke),  
                "Matični broj tvrtke mora biti popunjen");  
        if (string.IsNullOrEmpty(model.NazivTvrtke))  
            ModelState.AddModelError(nameof(model.PrezimeOsobe),  
                "Naziv tvrtke mora biti popunjen");  
    }  
}
```

# Unos novog partnera (1)

31

➡ 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  
        if (model.TipPartnera == "O") {  
            p.Osoba = new Osoba();  
            p.Osoba.ImeOsobe = model.ImeOsobe;  
            p.Osoba.PrezimeOsobe = model.PrezimeOsobe;  
        }  
        else {  
            p.Tvrtka = new Tvrtka();  
            p.Tvrtka.NazivTvrtke = model.NazivTvrtke;  
            ...  
        }  
    }  
}
```

## Unos novog partnera (2)

32

➡ 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

33

➡ 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

34

- 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

35

- 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}]
```

# Upravljač za dohvat podataka za nadopunjavanje


36

- 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)  
        .ThenBy(l => l.Id)  
        .ToList();  
  
    return list;  
}
```

# Usmjeravanje do upravljača

37

- 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 Varaždín", "id": 6245}, {"label": "42204 Varaždín  
Breg", "id": 6246}, {"label": "42223 Varaždinske Toplice", "id": 6247}]
```

# Priprema i označavanje kontrola za unos

38

- 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
  - *data-autocomplete-result* sadrži vrijednost koja će se tražiti u istoimenom atributu skrivenog unosa
- Primjer:  Web \ Firma.Mvc \ Views \ Create.cshml

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

...
@section scripts{
    <script src="~/lib/jquery-ui/jquery-ui.js"></script>
    <script src="~/js/autocomplete.js"></script>
```

# Aktiviranje nadopunjavanja (1)

39

➡ 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-result');  
    if (resultplaceholder === undefined)  
        resultplaceholder = url;  
    $(element).blur(function () {  
        if ($(element).val().length === 0) {  
            $("#[data-autocomplete-result='" + resultplaceholder  
                + "']").val('');  
        }  
    });  
    ... aktiviraj autocomplete na elementu ...  
});
```

# Aktiviranje nadopunjavanja (2)

40

## ➤ 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);  
        $("[data-autocomplete-result='" + resultplaceholder + "']")  
            .val(ui.item.id);  
    }  
});
```



# Pogreške prilikom dodavanja novog partnera

41

- 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


42

➤ 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)


43

- 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)

44

- 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

45

➡ Budući da je povezani dio za osobu ili tvrtku nastao stvaranjem novog objekta, ne ažuriranjem onog dohvaćenog iz BP, EF ga smatra novim objektom te bi za njega definirao insert upit

➡ EksPLICITNO mijenjamo stanje tog objekta iz *Added* u *Modified*


➡ 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)  
        ctx.Entry(partner.Osoba).State = EntityState.Modified;  
    if (partner.Tvrtka != null)  
        ctx.Entry(partner.Tvrtka).State = EntityState.Modified;  
  
    ctx.SaveChanges();  
}
```

# Brisanje partnera

46

- 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;`