ASP.NET MVC

Vjezba 6: Routing (advanced), partial view

Sadržaj

URL usmjeravanja (routing) 3
Usmjeravanje pomoću atributa 5
Djelomični (partial) view 8
Proslijeđivanje modela u partial view 8
Generiranje HTML input elemenata 10
Naredbe EditorFor i TextBoxFor 10
Tag helper <input> 10
Form helper 11

URL usmjeravanja (routing)

Dosad se koristio samo osnovni oblik routinga - /Controller/Action/{id – opcionalno}. U nastavku će se razmotriti kompleksniji i prilagođeniji scenariji za rukovanje URL usmjeravanjima kako bi aplikacija bolje odgovarala korisnicima.

Startup.cs

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    ...

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "quiz-manager/{controller=Home}/{action=Index}/{id?}");
    });
}
```

Pogledajmo gornju jedinu definiranu rutu – osnovnu rutu (otisnuto masnim slovima). Slijede objašnjenja pojedinih parametara:

- **name** – jedinstveni naziv rute, koristi se kako bi svaka definicija rute imala jedinstveno ime
- **pattern** – url shema preko koje se aktivira ruta. Primjerice, gornja ruta definira da se URL shema može sastojati od najviše 3 dijela (recimo HOST/Xxx/Yyy/123), gdje se iz prvog parametra (Xxx) iščitava naziv odgovarajućeg Controller-a – u ovom slučaju to bi bila klasa XxxController; iz drugog parametra se iščitava naziv odgovarajuće akcije u Controlleru – to bi bila akcija (ActionResult Yyy(...) { }); a iz trećeg dijela se izvlači parametar koji se prosljeđuje akciji (ActionResult Yyy(int id), gdje parametar „id“ poprima vrijednost 123). Id nije nužno cijeli broj – može biti i string, iako u većini aplikacija se iza imena id očekuje cijeli broj.
    - **Inicijalne vrijednosti** – ako URL putanja nije u potpunosti definirana, tada se koriste inicijalne (ili fallback) vrijednosti. Primjerice, za controller, *fallback* vrijednost je Home. Za parametar action, *fallback* vrijednost je Index.
        - **URL: HOST/Xxx/Yyy/123** – inicijalne vrijednosti nemaju nikakvog efekta jer su svi parametri zadani. Poziva se controller = **XxxController**, akcija = **Yyy**, paramter id = 123.
        - **URL: HOST/Xxx/Yyy** –treći parametar je opcionalan, stoga se Id jednostavno nigdje ne pridjeljuje
        - **URL**: **HOST/Xxx** – u ovom slučaju definiran je **XxxController**, a zaključuje se da ako drugi parametar nije specificiran, podrazumjeva se akcija **Index**.
        - **URL**: **HOST/** - u ovom slučaju niti jedan parametar nije definiran, te se podrazumjeva akcija **Index** i controller **Home**.
    - **defaults** – dodatni parametar funkcije MapControllerRoute je defaults. Ukoliko se iz URL-a ne može odrediti koji controller i koja akcija obrađuje web zahtjev, tada se iz *defaults* parametra preuzimaju tražene vrijednosti. Pogledati tablicu na idućoj stranici za više detalja.

Važna napomena: nomenklatura u ASP.NET MVC je veoma bitna, i treba poštivati određena pravila. Ukoliko je potrebno, moguće je ta pravila zaobići i prilagoditi potrebama, no to je preporučljivo samo u iznimnim situacijama. Evo nekoliko bitnijih pravila:

- Controller klasa za pojam „Xyz“ će se zvati XyzController.
- Controller klasu je obvezno staviti na istu razinu s ostalim controllerima
- View datoteke vezane uz taj controller, moraju biti u mapi Views/Xyz/...
- Pri definiranju rute, ne koristi se naziv XyzController, nego samo Xyz

Nekoliko primjera pravila usmjeravanja uz pojašnjenje:

Naziv rute: Korisnici_ruta
Url: /Korisnici/Index aktivira UserController.Index() akciju
/Korisnici/Edit aktivira UserController.Edit() akciju
Defaults: ovdje razni URL-ovi oblika /Korisnici/* aktiviraju akcije iz UserController-a
	`endpoints.MapControllerRoute(`
	`  name: "Korisnici_ruta",`
	`  pattern: "Korisnici/{action}",`
	`  defaults: new { controller = "User" }`
	`);`

Naziv rute: Profile_default
Url: /moj-profil aktivira AccountController.Profile() akciju
Defaults: samo jedan jedini URL moze aktivirati ovu akciju
	`endpoints. MapControllerRoute(`
	`  "Profile_default",`
	`  "moj-profil",`
	`  new { controller = "Account", action = "Profile" }`
	`);`

Naziv rute: BlogDetails
Url: /blog/icesar/uvod-u-dot-net aktivira akciju BlogController.Details(string blog, string post), gdje se u varijabli blog nalazi string „icesar“, a u varijabli post string „uvod-u-dot-net“.
Defaults: ovdje iz samog URL-a ne definiramo controller i akcije (naime, blogovi se mogu dinamički dodavati, ne možemo napraviti posebnu akciju za svaki novi blog), već se fiksno usmjerava ovakav URL na Articles controller i akciju DetailsBlog
	`endpoints.MapControllerRoute(`
	`  "BlogDetails",`
	`  "blog/{blog}/{post}",`
	`  new { controller = "Blog",`
	`  action = "Details" }`
	`);`

URL: /icesar/uvod-u-dot-net ili /icesar, obje rute aktiviraju akciju BlogController.Details(string blog, string post)
Pojašnjenje: U ovom slučaju, scenarij je sličan kao i gore – imamo dva parametra, no ovaj put se definiraju ograničenja na izgled tih parametara. Parametar **blog** se sastoji od barem jednog (može i više – znak '+' iza uglate zagrade) malih i velikih slova eng. abecede, znamenke i znaka '-'. Parametar **post** je opcionalan, i samim time je ograničenje definirano na način da može biti nula ili više (znak '*' iza uglate zagrade) malih i velikih slova eng. abevede, znamenki i znakova '-'.
	`endpoints.MapControllerRoute(`
	`  name: "BlogDetails2",`
	`  url: "{blog}/{post}",`
	`  defaults: new {`
	`  controller = "Blog",`
	`  action = "Details",`
	`  post = UrlParameter.Optional },`
	`  constraints: new {`
	`  blog = @"[a-zA-Z0-9-]+",`
	`  post = @"[a-zA-Z0-9-]*"});`

Zadatak 6.1

Definirati specifične rute za akcije **Contact** i **About** u **HomeController** klasi.

- Akcija Home -> Contact se mora pozivati ukoliko se za URL unese http://localhost:PORT/kontakt-forma
- Akcija Home -> Privacy se mora pozivati ukoliko se za URL unese http://localhost:PORT/o-aplikaciji/LANG, gdje je LANG parametar od točno 2 slova engleske abecede (dodati ograničenje u definiciju usmjeravanja).

- Kao rezultat, akcija Privacy treba u ViewBag spremiti poruku na jeziku koji je poslan kao parametar, te tu poruku prikazati u **Privacy.cshtml** View-u. Napraviti za en, hr, de, zh.
- Link u izborniku možda neće raditi ispravno nakon ove promjene – zasad to ignorirati i upisati URL ručno.

Korisni linkovi:

- https://docs.microsoft.com/en-us/aspnet/core/fundamentals/routing?view=aspnetcore-5.0#route-constraint-reference

Usmjeravanje pomoću atributa

Od ASP.NET MVC verzije 5 dostupan je intuitivniji i lakši način definiranja ruta za controller-e i akcije – koristeći posebne anotacije. Točnije, koristi se atribut **[Route]** koji se dodaje na akciju controller-a ili na sami controller. U njemu se definira URL koji je potrebno unijeti za pristup određenoj akciji.

Primjer definiranja ruta za CityController koja se aktivira na URL **/gradovi/po-drzavi/CRO** ili **/gradovi/po-drzavi/SLO**:

```
namespace Vjezba4.Web.Controllers
{
    [Route("gradovi")]
    public class CityController : Controller
    {
        [Route("po-drzavi/{country:length(3)}")]
        public ActionResult List(string country)
        {
            //Obrada
            //...

            return View();
        }
    }
}
```

Napomena: U prijašnjoj verziji ASP.NET MVC radnog okivra, bilo je potrebeno eksplicitno uključiti tzv. *Attribute routing*. Od ove verzije to je uključeno automatski.

Slijedi nekoliko primjera korištenja [Route] atributa:

URL: /Home/Index

```
[Route("[controller]/[action]")]
public class HomeController : Controller
{
    public IActionResult Index(string lang = null)
```

/Home/Index ili **/Home/Index/e** ili **/Home/Index/en** ili **/**

```
[Route("[controller]/[action]")]
public class HomeController : Controller
{
    [Route("/")]
    [Route("{lang:minlength(1):maxlength(2)?}")]
    public IActionResult Index(string lang = null)
```

/Home/Index ili **/Home/Index?lang=en**

```
[Route("[controller]")]
public class HomeController : Controller
{
    [Route("[action]")]
    public IActionResult Index(string lang = null)
```

/dom/indeks

```
[Route("dom")]
public class HomeController : Controller
{
    [Route("indeks")]
    public IActionResult Index()
```

/dom

```
[Route("dom")]
public class HomeController : Controller
{
    [Route("")]
    public IActionResult Index()
```

/dom ili /dom/indeks

```
[Route("dom")]
public class HomeController : Controller
{
    [Route("")]
    [Route("indeks")]
    public IActionResult Index()
```

Zadatak 6.2

1. Definirati **Route** anotaciju na akciju Faq u HomeController klasi
    a. Ograničiti da ID može biti samo cjelobrojna vrijednost.
    b. Ograničiti da ID može biti minimalno 1 a najviše 2 znamenke
    c. ID je opcionalan parametar

2. Pristupiti se mora moći sa URL-a localhost:PORT/cesto-postavljana-pitanja

Korisni linkovi:

- https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/routing?view=aspnetcore-5.0#attribute-routing

Djelomični (partial) view

Djelomični view možemo na neki način poistovjetiti s korisničkim kontrolama u ASP.NET WebForms tehnologiji – služi za iscrtavanje jednog specifičnog dijela stranice, kojeg eventualno koristimo na više mjesta u aplikaciji. Djelomični view se može koristiti na dva načina:

- **Pozivom tijekom iscrtavanja „običnog“ view-a, uz proslijeđivanje odgovarajućeg modela, u samoj razor sintaksi**
- Pozivom akcije controller-a koja može vratiti PartialViewResult

Za razliku od običnog view-a, djelomični view se **ne** iscrtava pomoću _Layout stranice – kad se iscrtava „obični“ view, sadržaj se ubacuje na odgovarajuće mjesto unutar _Layout stranice. Djelomični view ne uzima u obzir nikakav kontekst, već samo iscrtava sadržaj koji se tamo nalazi. Djelomični view se koristi u slučajevima kad postoji HTML/js/C# kod koji se može nalaziti na više mjesta, kako se kod ne bi kopirao i ponavljao; i samim time povećavala mogućnost pogreške.

Primjer za iscrtavanje jednog smislenog djelomičnog view-a se može naći u _Layout datoteci, kod iscrtavanja kontrole za prijavu:

_Layout.cshtml

```
<div class="float-right">
    <section id="login">
        @await Html.PartialAsync("_LoginPartial")
    </section>
    <nav>
```

U gornjem slučaju, _LoginPartial ne prima nikakav objekt kao model, i nalazi se u Shared folderu, kako bi bio dostupan u view-u. Generalno, ne postoji obvezna nomenklatura za djelomične poglede, no preporuča se korištenje gornje nomenklature – sa znakom '_' (underscore) prije imena.

Od .NET Core verzije postoji i posebni asp tag kojim se iscrtava partial view, primjer kojega se može naći u _Layout stranici:

_Layout.cshtml

```
<partial name="_CookieConsentPartial" />
```

Česti primjer gdje se koristi djelomično iscrtavanje je prilikom kreirana standardnog mehanizma za kreiranje i osvježavanje podataka. Konkretno, kada dodajemo novi podatak ili kada uređujemo taj podatak, možemo primjetiti da je forma za prikaz gotovo identična – postoji eventualno razlika u tome koja akcija controller-a će obraditi zahtjev kad se zatraži prazna forma, ili kad se forma s podacima pošalje na server.

Proslijeđivanje modela u partial view

Prilikom poziva partial metode, kao i kod svakog drugog view-a, ASP.NET MVC očekuje da se proslijedi model. Primjerice, pretpostavimo da je definiran partial view:

_ClientFilter.cshtml

```
@model Vjezba.Web.Models.ClientFilterModel

<div>
    Ime: @Html.EditorFor(p => p.yName)
</div>
```

Također, pretpostavimo da imamo pregled klijenata definiran ovako:

Index.cshtml

```
@model List<Vjezba.Web.Models.Mock.Client>

@{
    ViewBag.Title = "Index";
}

<h2>Pregled klijenata</h2>
```

Postoji nekoliko načina kako bi mogli pozvati iscrtavanje _ClientFilter djelomičnog view-a:

- Pozivom `@await Html.PartialAsync("_ClientFilter")` – prouzročit će **pogrešku**, jer ukoliko ne navedemo model koji se prosljeđuje u Partial metodu, tada se automatski prosljeđuje model iz view-a koji poziva Partial metodu – u ovom slučaju proslijedio bi se kao model List<Client>, a nas view _ClientFilter očekuje kao model ClientFilterModel.
- Pozivom `@await Html.PartialAsync ("_ClientFilter", null)` – prouzročit će **pogrešku**, model koji šaljemo neće biti ispravnog tipa ali imati vrijednost „null“
- Pozivom `@await Html.PartialAsync ("_ClientFilter", new ClientFilterModel())`– šaljemo novi objekt, ovaj puta ispravnog tipa
- Pozivom `<partial name="_ClientFilter" model="new ClientFilterModel()" />`

Također, postoji opcija koristeći **RenderPartial** metodu, ali to se ostavlja čitatelju na vlastito istraživanje.

Generiranje HTML input elemenata

Od .NET Core verzije MVC-a, postoji nekoliko načina na koje možemo generirati HTML input elemente.

Naredbe EditorFor i TextBoxFor

Naredba **EditorFor** prima kao parametar **Func<>** objekt kojim se definira na koje polje se odnosi te prema tome generira HTML „name“ atribut. Slično funkcionira i naredba **TextBoxFor**, ali ona uvijek generira **input type=text**, dok **EditorFor** generira HTML input element u ovisnosti o tipu podatka. Varijabla **p** je istog tipa kojeg je i model koji se koristi unutar view-a – u gornjem primjeru to je **ContactModel** klasa. Uz EditorFor, dostupna je i funkcija NameFor koja na sličan način generira odgovarajući „name“ parametar:

Contact.cshtml

```
@model QuizManager.Web.Models.ContactModel

...

<section class="contact-form">
    <header>
        <h3>Pošaljite nam upit!</h3>
        <form action="/Home/Contact" method="post">
            <div>
                Ime: @Html.TextBoxFor(p => p.Ime)
                Prezime: <input type="text" name="@Html.NameFor(p => p.Prezime)" />
            </div>
            <div>
```

Tag helper <input>

Sličnog koncepta kao EditorFor, međutim prilagođenije HTML sintaksi je tzv. Tag helper za generiranje input elemenata.

Contact.cshtml

```
@model QuizManager.Web.Models.ContactModel

...

<section class="contact-form">
    <header>
        <h3>Pošaljite nam upit!</h3>
        <form action="/Home/Contact" method="post">
            <div>
                Ime: <input asp-for="Ime" class="form-control" />
                Prezime: <input asp-for="Prezime" class="form-control" />
            </div>
           <div>
```

Može se uočiti kako asp-for atribut zapravo zamjenjuje lambda izraz (p => p.Ime), tj., možemo zamisliti kako interno asp-for izvodi identičan lambda izraz kao gore navedeno u EditorFor/TextBoxFor.

Zadatak 6.3

Dodati četvrtu formu na istu stranicu, te dopustiti unos dodatnih parametara pretrage kao u zadacima u vježbi 5. Iskoristiti već kreiranu klasu **ClientFilterModel** iz prošle vježbe u koju će se povezivati vrijednosti s forme. Kreirati PartialView **_ClientFilter** i integrirati ga u Index stranicu. Model tog partial view-a treba biti **ClientFilterModel**.

1. Za poziv PartialView koristiti oba načina pa pri demonstraciji zakomentirati/odkomentirati pojedini način:
    a. Html.PartialAsync()
    b. <partial> tag helper
2. Kreirati još jednu formu na pregledu klijenata, koja dopušta unos istih parametara kao prošla forma
    a. Iskoristiti **EditorFor** i **NameFor** ekstenzije za kreiranje polja na formi
        i. Pri radu s EditorFor ekstenzijom, predati i **htmlAttributes** kako je definirano ranije
    b. Iskoristiti **<input asp-for..>** tag helper i postaviti mu CSS klasu form-control
    c. Iskoristiti TextBoxFor i predati klasu **'form-control'**, kako je definirano ranije
    d. Podesiti da je tab sa novom formom otvoren pri učitavanju stranice
3. Iskoristiti već kreiranu akciju **public IActionResult AdvancedSearch(ClientFilterModel model)** za obradu podataka s forme
4. Koristeći osnovne bootstrap principe, podesiti da forma izgleda kao za zadatke iz vježbe 5.

Form helper

Zadnji problem koji je još potrebno riješiti je problem ručno unešene adrese (form action atribut) na koju se šalju podaci u formi. Za to također postoji riješenje:

Contact.cshtml

```
<section class="contact-form">
    <header>
        <h3>Pošaljite nam upit!</h3>
        @using(Html.BeginForm())
        {
            ...
        }
    </header>
</section>
```

Funkcija **BeginForm** također prima niz parametara koji definiraju:

- Akciju i controller na koju se treba poslati podaci
- Method – GET ili POST
- Dodatne route parametre ili html atribute

Druga opcija koju možemo koristiti je form tag helper:

Contact.cshtml

```
<section class="contact-form">
    <header>
        <h3>Pošaljite nam upit!</h3>
        <form asp-action="SubmitQuery" method="post">
            ...
        </form>
    </header>
</section>
```

Zadatak 6.4

Iskoristivši do sada obrađeno, potrebno je napraviti novu formu koja će služiti za unos podataka o novom klijentu.

1. Modificirati **ClientController** (ranije napravljen – s index akcijom) na način da se dodaju nove akcije **Create** i **[HttpPost]Create** koje služe za prikaz i obradu podataka s forme
    a. Akcija **Create** (bez HttpPost ograničenja) služi za prikaz prazne forme
    b. Akcija **[HttpPost] Create** služi za obradu podataka koji su unešeni na formi
2. Kreirati novi view **Create.cshtml** unutar foldera Views/Client
    a. Koristiti <form> tag helper ili Html.BeginForm()
3. Pozvati u odgovarajućem trenutku metodu **MockClientRepository.Instance.InsertOrUpdate()** s objektom koji je popunjen u formi
    a. S obzirom da se koriste Mock objekti, svi podaci se resetiraju nakon restarta aplikacije
4. Vezane klase (City) i Id polje potrebno je popuniti u controlleru prije poziva **InsertOrUpdate**() funkcije.
5. Dodati link na **Index** stranicu za odlazak na formu za dodavanje novog objekta (koristeći Html.ActionLink ili Url.Action ili <a> tag helper)
6. Koristeći osnovne bootstrap principe, podesiti da forma izgleda kao na slikama dolje
    a. Dodati i „krušne mrvice“ za povratak na listu

Napomena: **ne** koristiti automatsko kreiranje view-a nego dodati elemente ručno za vježbu.