

HTML Helper-i

Kreiramo MVC projekat uz pomoć gotovih template-a, scaffolding-a. Scaffolding je framework za generisanje koda u ASP.NET veb aplikacijama.

Napomena: Ovaj način izrade završnog projektnog zadatka je za minimalan broj bodova. Ovaj način izrade nije dozvoljen prilikom izrade projekta za veći broj bodova i na ostalim predispitnim obavezama.

Uvod u HTML Helper-e

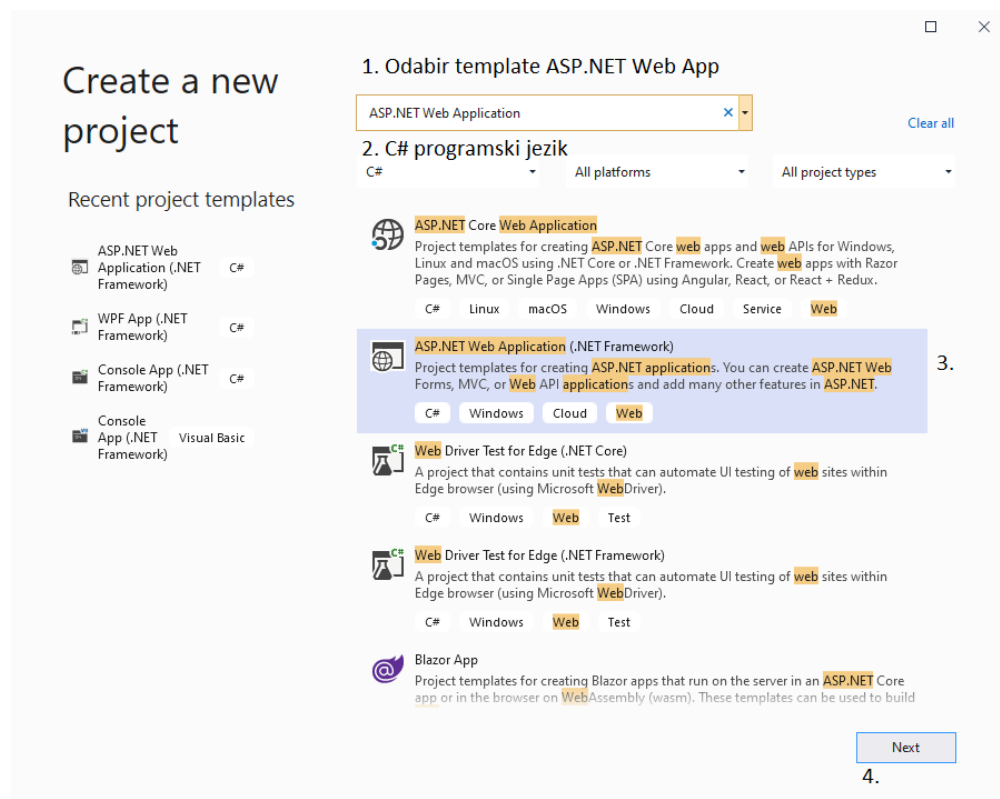
Uz pomoć HTML helper-a generišemo boiler-plate kod tj. kod koji se najčešće pojavljuje prilikom pisanja veb aplikacija. To su uglavnom CreateReadUpdateDelete funkcionalnosti. Pristup HTML helper-a unutar Razor stranica je omogućen putem @Html.NazivHelper-a. Na linije gde se nalazi HTML helper prilikom izvršavanja aplikacije, tj. pristup prikazu sa HTML helper-ima na klijentskoj strani će se prikazati html sadržaj.

Kreiranje MVC projekta uz pomoć scaffolding-a

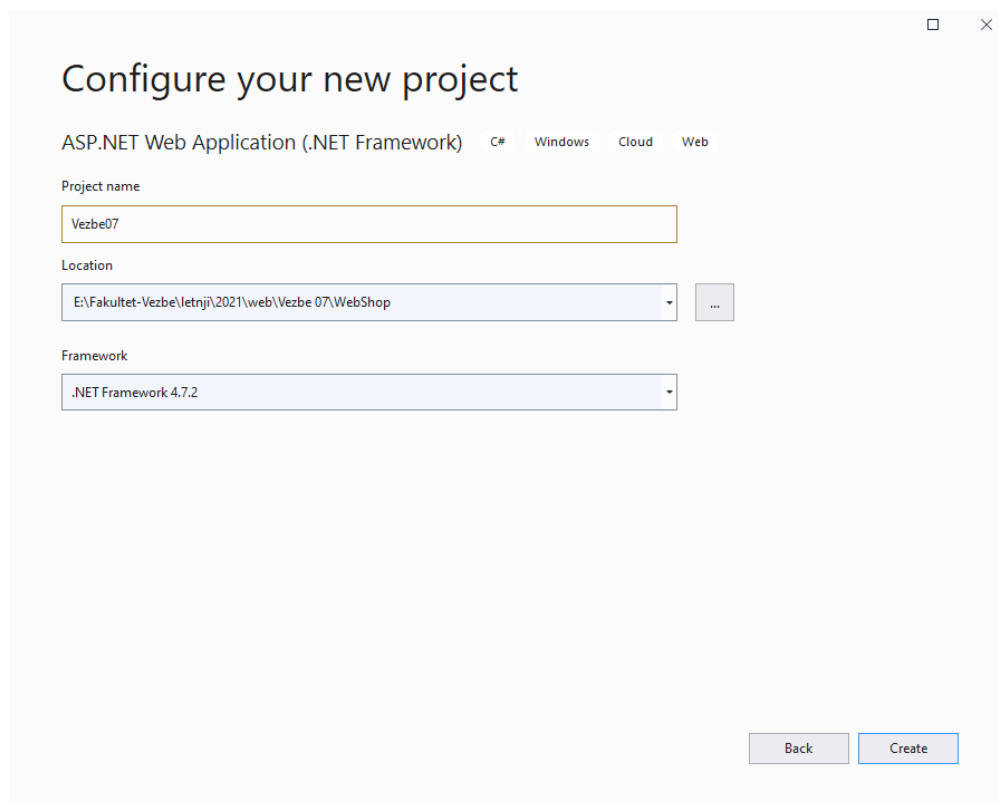
Potrebno je da kreiramo ASP.NET Web Application projekat uz pomoć MVC template-a.

1. Korak: Kreiramo novi projekat u postojećem solution ili u novom.

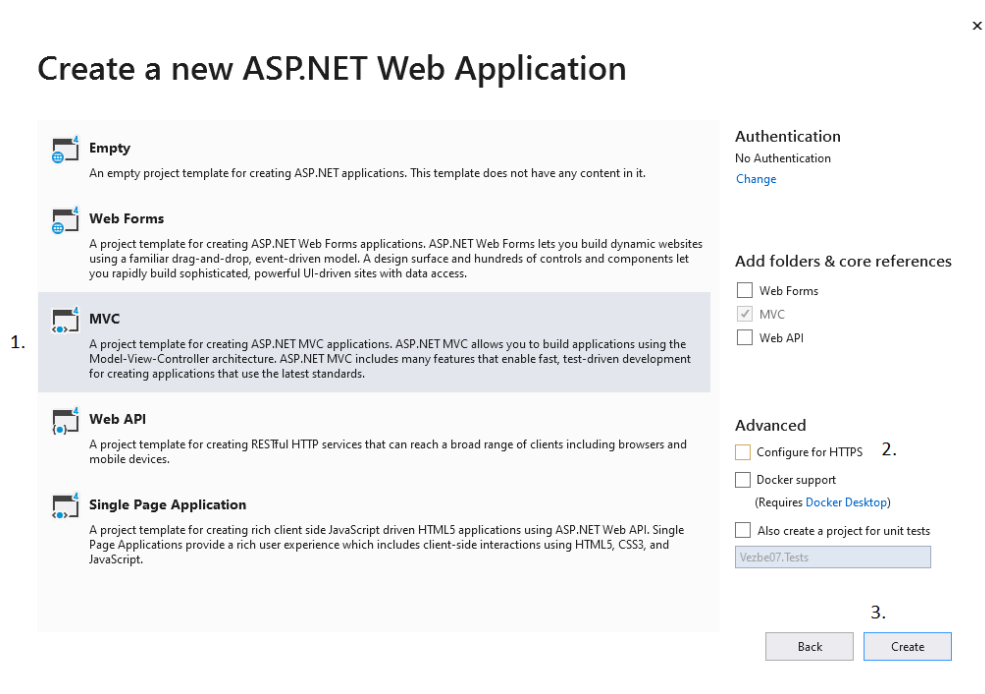
Opcija File>New>Project ili desni klik na solution (iz IDE Visual Studio) i odabir opcija Add>New Project... Nakon toga ispratite Slike 1., 2., i 3.



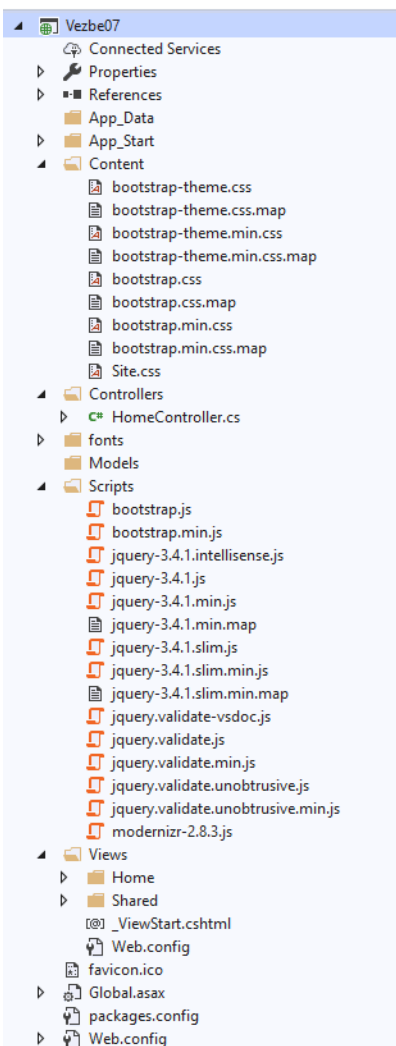
Slika 1. Prvi prozor za odabir project template-a ASP.NET Web Application



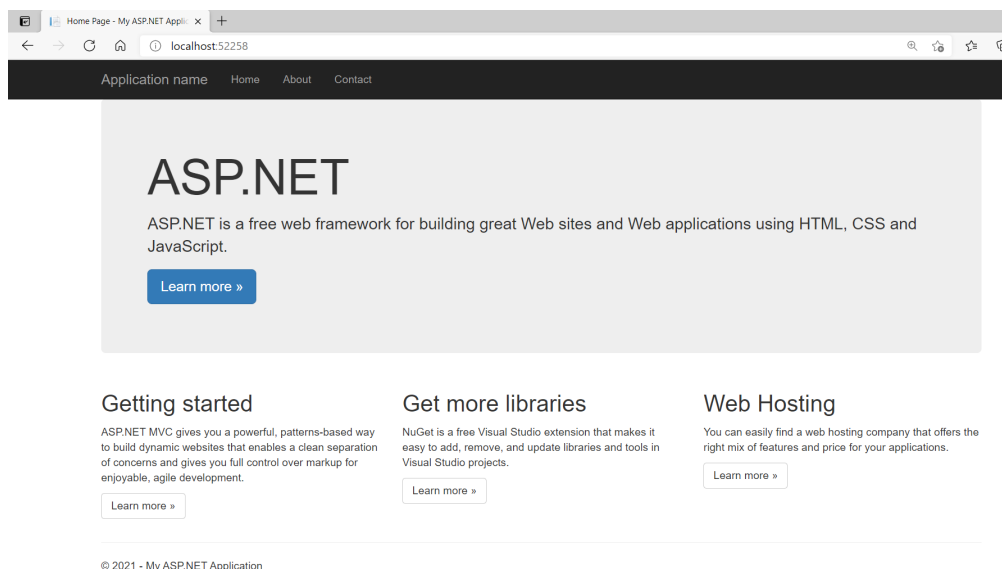
Slika 2. Drugi prozor za definisanje naziv novog projekta, odabira lokacije kreiranja projekta u novi solution ili postojeći. Na kraju ide odabir Framework-a. (Za domaće zadatke i projektni zadatak možete do verzije 4.7.0)!



Slika 3. U posljednjem prozoru odaberemo da želimo project template za kreiranje ASP.NET MVC aplikacije. Sa ovom opcijom automatski će biti odabrana opcija MVC pod Add folders & core references. Obavezno isključite Configure for HTTPS.



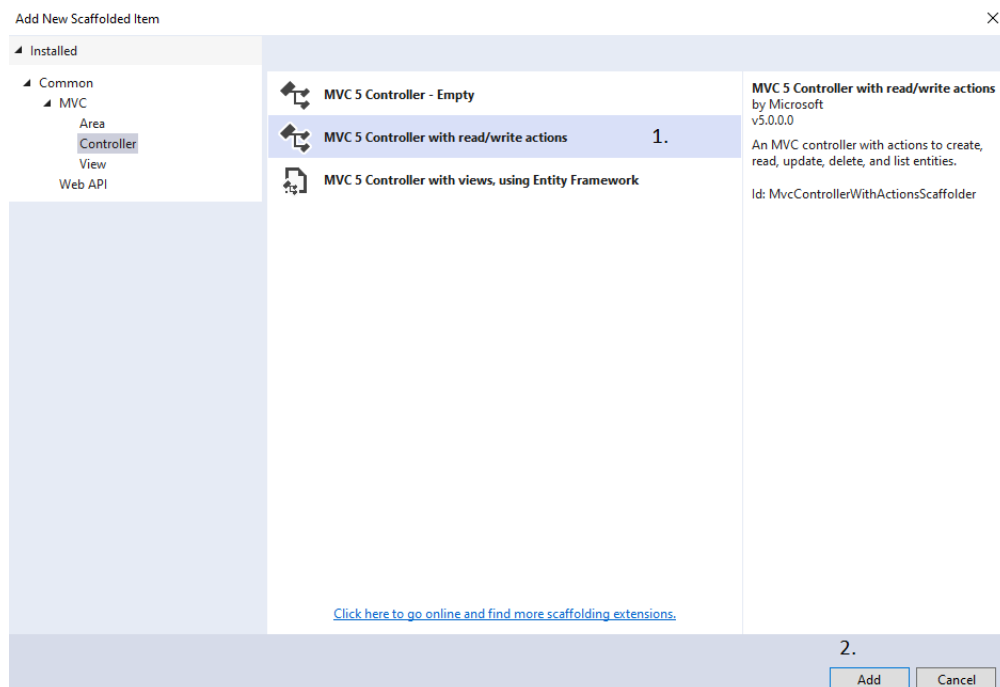
Slika 4. Za razliku od odabira Empty project template-a i samo odabir MVC kao opciju pod Add folders & core references, sad vidimo da se automatski kreirao HomeController sa svojim odgovarajućim Views/Home folder-om. Takođe, automatski su se dodali i bootstrap.css fajlovi kao i javascript, jquery biblioteke. Ove fajlove/biblioteke/frameworks možemo da koristimo u sklopu ove aplikacije.



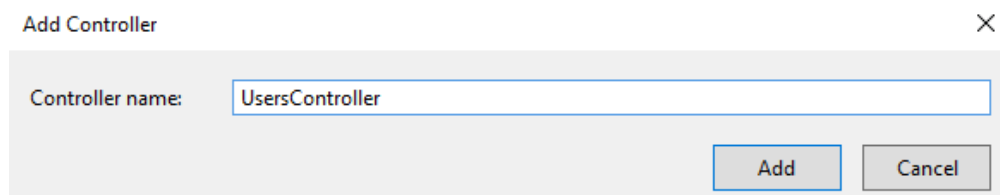
Slika 5. Ako pokrenete ovu aplikaciju prikazaće se sadržaj iz Home/Index.cshtml-a.

Napomena: Prethodne vežbe smo detaljno prošli kroz najbitnije direktorijume i datoteke i objasnili njihove uloge.

2. Korak: Kreiranje kontrolera, uz pomoć scaffolding-a.



Slika 6. Odabraćemo opciju MVC 5 Controller with read/write actions. Ova opcija će automatski generisati metode/akcije za CRUD tj. za kreiranje entiteta preko FormCollection, za prikaz i brisanje entiteta preko id parametra i modifikaciju entiteta preko id parametra i FormCollection. Id parametar predstavlja jedinstveni identifikator nekog entiteta.



Slika 7. Prilikom davanja naziva kontrolerima uvek moramo da završimo naziv sa Controller po konvenciji

3. Korak: Kreiranje modela (entiteta) User sa poljima username i password unutar direktorijuma Models.

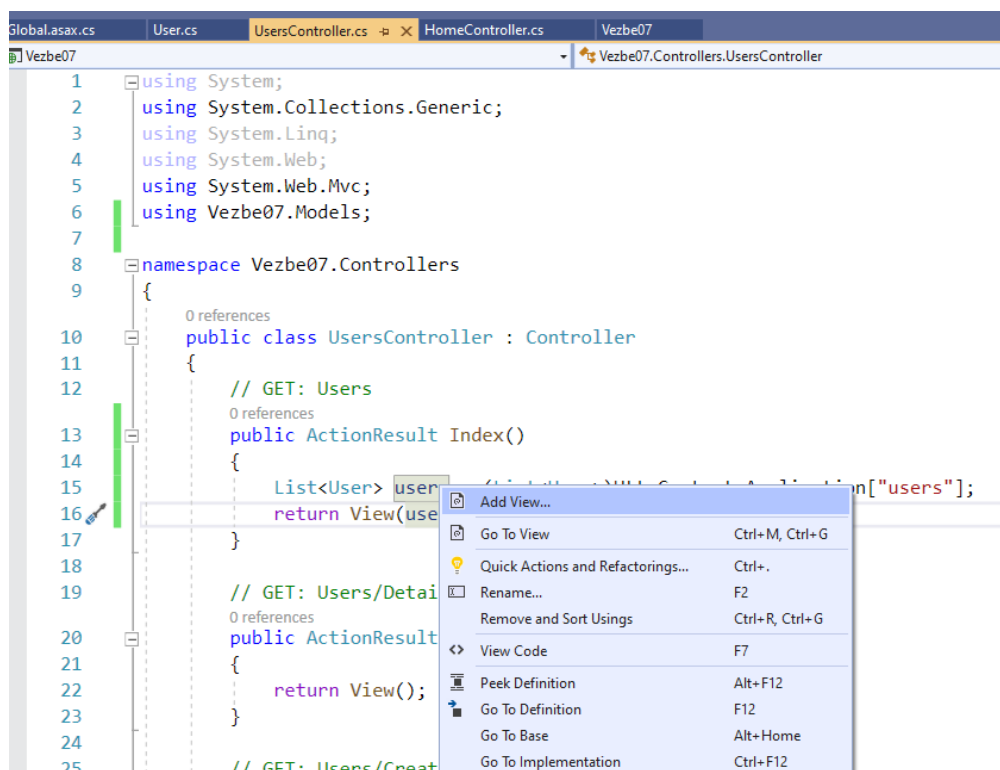
Radi kompletnosti zadatka unutar Global.asax.cs fajla na kraju Application_Start() metode dodati listu korisnika u

```
In [ ]: HttpContext.Current.Application["users"] = new List<User>
{
    new User {
        Username = "pera",
        Password = "pera"
    },
    new User {
        Username = "mika",
        Password = "mika"
    }
};
```

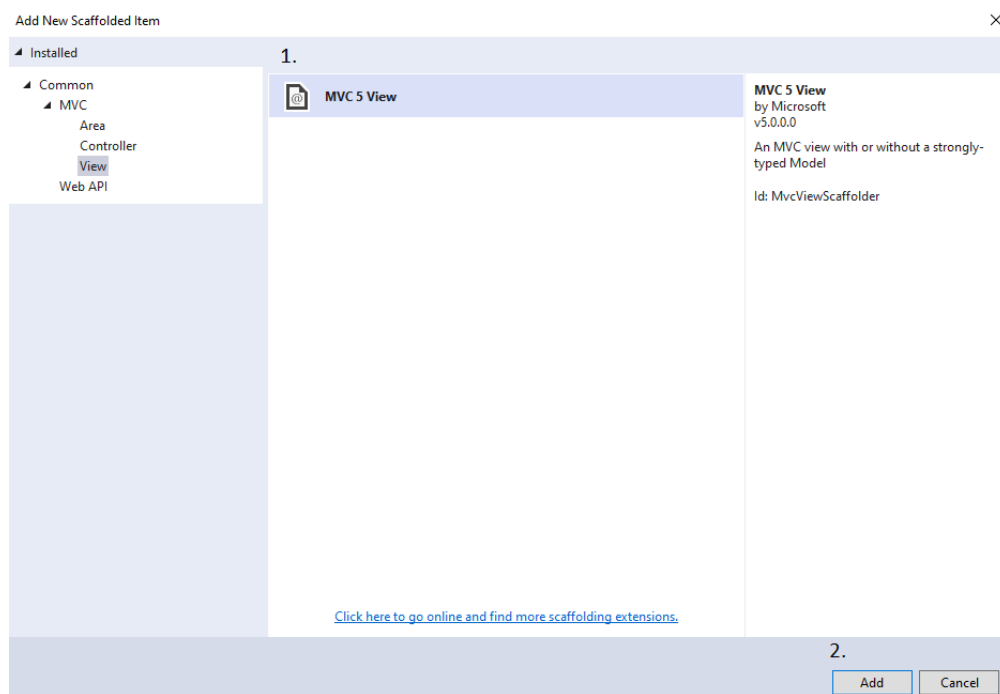
4. Korak: Dopuniti UsersController Index akciju da prosleđuje kao model listu korisnika iz Application objekta

```
In [ ]: // GET: Users
public ActionResult Index()
{
    List<User> users = (List<User>)HttpContext.Application["users"];
    return View(users);
}
```

5. Korak: Dodajemo prikaz svih korisnika uz pomoć scaffolding-a (Slika 8, Slika 9, Slika 10)



Slika 8. Desni klik na return View(users); i odabir opcije Add View



Slika 9. Ovaj postupak je isti kao i prošle nedelje

Add View

View name:

Template:

Model class:

Options:

☐ Create as a partial view

☒ Reference script libraries

☒ Use a layout page:

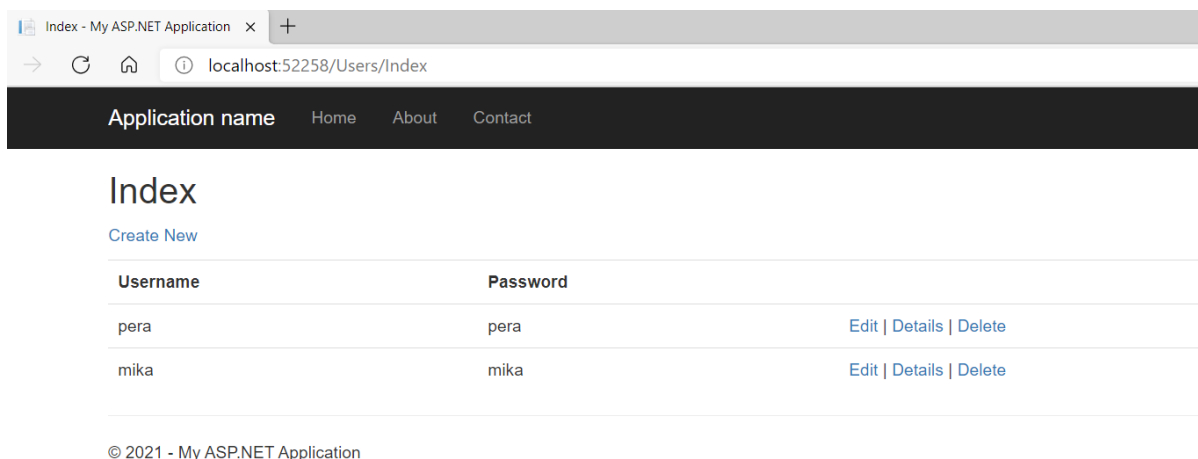
...

(Leave empty if it is set in a Razor _viewstart file)

3.

Slika 10. U sledećem prozoru treba da odaberemo pod Template:List, Model class:User(Vezbe07.Models) - gde je Vezbe07 naziv projekta. i odaberemo na kraju Add

Na kraju se izgeneriše u Views/Users Index.cshtml fajl. Pokrenite ponovo aplikaciju i prosledite u URL-u localhost:port/Users/.



Slika 11. Vidimo tabelaran prikaz svih korisnika u aplikaciji sa dodatnim opcijama Edit|Details|Delete koji još uvek nisu implementirane

Analiziranje Index.cshtml stranice.

Na Index.cshtml stranici se pojavilo nekoliko @Html.NazivHelper-a

1. `@Html.ActionLink("Create New", "Create")`: izgenerisaće a link tag. Prvi parametar Create New je sadržaj link taga, a drugi parametar je naziv akcije. Postoji overload metoda gde je treći parametar naziv kontrolera, ako ne navede naziv kontrolera podrazumevana vrednost je naziv kontrolera unutar kojeg se nalazi akcija koja vraća ovaj prikaz.

```
In [ ]: <a href="/Users/Create">Create New</a>
```

1. `@DisplayNameFor(model => model.Username)` vraća naziv svojstva `Username`. U našem slučaju to se poklapa sa nazivom svojstva, ali ova vrednost može da se redefiniše unutar modela sa anotacijom `[Display(Name=string_value)]` iznad svojstva npr. `Username`. Dopunite `User.cs` sa sledećim kodom.

```
In [ ]: public class User
{
    [Display(Name = "Korisnicko Ime")]
    public string Username { get; set; }
    public string Password { get; set; }
}
```

1. `@Html.DisplayFor(modelItem => item.Username)` kao rezultat vraća vrednost korisničkog imena jednog korisnika. `item` je `User`.
2. Svaki korisnik će imati `Edit`, `Details` i `Delete` opciju. `@Html.ActionLink("Edit", "Edit", new { id=item.PrimaryKey / })` Ovde je u komentaru ostavljeno da sami definišemo jedinstven identifikator po kojem ćemo razlikovati korisnike.

6. Korak: Implementiranje opcija Edit, Details i Delete

Details

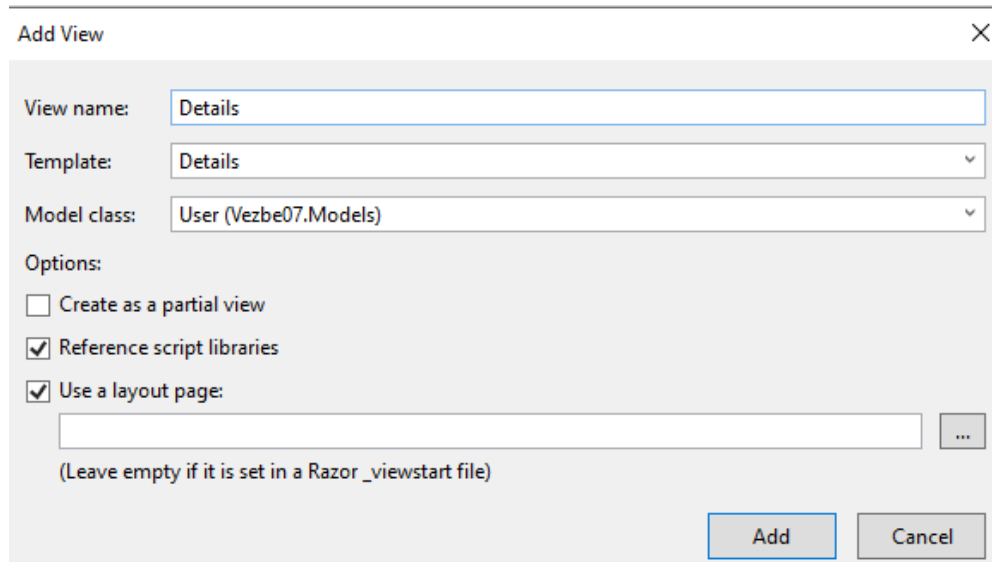
Na kraj `Index.cshtml` (`Views/Users`) u poslednji umesto prethodnih `ActionLink`-ova za `Edit`, `Details` i `Delete` dodati sledeće:

```
In [ ]: @Html.ActionLink("Edit", "Edit", new { id=item.Username }) |
        @Html.ActionLink("Details", "Details", new { id = item.Username }) |
        @Html.ActionLink("Delete", "Delete", new { id = item.Username })
```

`UserController` redefinisati `Details` akciju:

```
In [ ]: // GET: Users/Details/5
public ActionResult Details(string id)
{
    List<User> users = (List<User>)HttpContext.Application["users"];
    var user = users.Find(u => u.Username.Equals(id));
    return View(user);
}
```


Dodati View kod return View(user). Prilikom Add View (drugi prozor) opcije selektovati sledeće (Slika 12.)

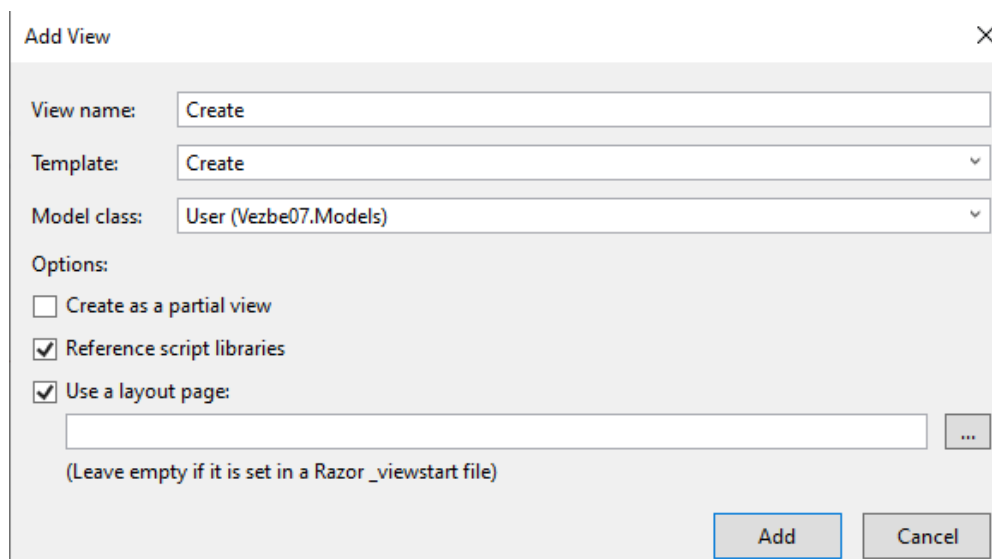


Slika 12. Kreiranje detaljnijeg pogleda preko template-a Details

Unutar Views/Users direktorijuma kreirao se novi pogled Details.cshtml. Ako pokrenete opet aplikaciju, prosledite localhost:port/Users i odaberete na Details opciju šta će se prikazati?

Create

Za kreiranje novog korisnika prvo moramo da prikazemo formu sa odgovarajućim poljima za korisnika (polja koja trebaju da se unesu). Ovo je prva akcija. Potom ide klik na dugme Ok (Create), ovo je druga akcija dobija podatke o novom korisniku od strane klijenta kojeg treba da kreira i ubaci u listu postojećih korisnika. Prepraviti sledeći kod u UsersController.cs: Unutar // GET: Users/Create kreirati pogled (desni klik na return View(); i u drugom prozoru odabrati opcije sa Slike 13.

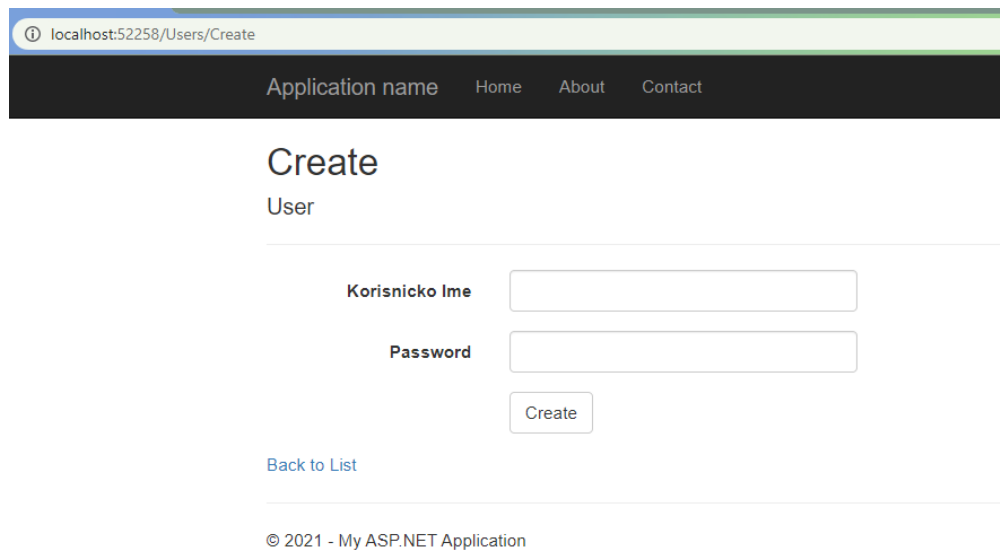


Slika 13. Kreiranje prozora za unos novog korisnika

Kreira se unutar Views/Users Create.cshtml. Ovde imamo nove @Html.NazivHelpera.

- @Html.BeginForm() kreiraće formu koja je tipa method post i po defaultu vrednost action atributa je isti naziv akcije iz koje je ovaj pogled vraćen kao rezultat (u našem slučaju to je Create).
- @Html.AntiForgeryToken() - koristi anti-fordery tokene za zaštitu veb sajta od CSRF napada
- @Html.ValidationSummary(true, "", new { @class = "text-danger" }) - prikazuje sve validacione greške kao neuređenu listu na veb stranici.
- @Html.LabelFor(model => model.Username, htmlAttributes: new { @class = "control-label col-md-2" }) - generiše labelu koja ima vrednost za atribut for i sadržaj labele kao naziv svojstva Username.
- @Html.EditorFor(model => model.Username, new { htmlAttributes = new { @class = "form-control" } }) - generiše input polje sa nazivom Username (name="Username").
- @Html.ValidationMessageFor(model => model.Username, "", new { @class = "text-danger" }) - umesto ovog polja prikazaće se ako postoji poruka definisana u ErrorMessage za dato polje Username u klasi User.cs. Ovo je poruka prilikom neispravnog unosa polja Username. Dodatna polja koja su definisana su class koja definiše izgled slova (npr. boja) i način na koji će se poruka prikazati na klijentu.

Sad ako pokrenete aplikaciju i prosledite localhost:port/Users/Create ili iz localhost:port/Users odaberete link Create new prikazaće vam se Slika 14.

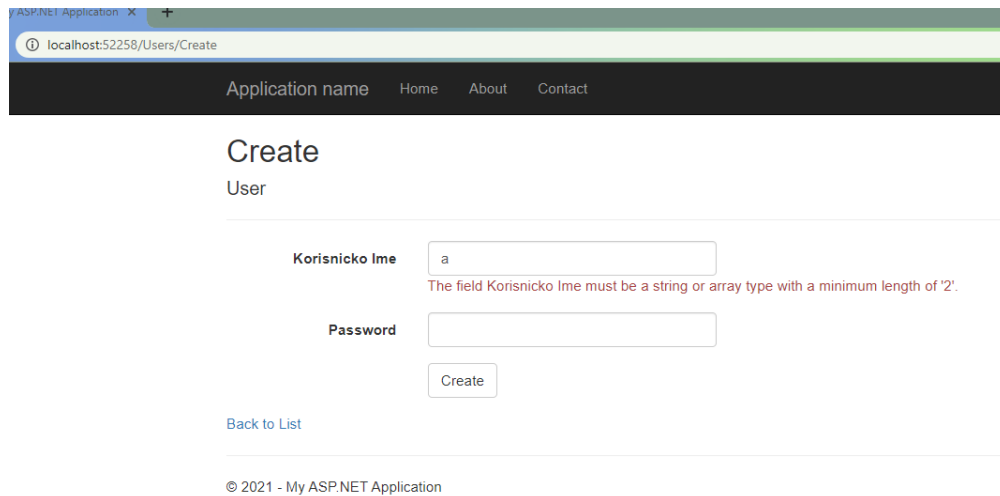


Slika 14. Forma za unos novog korisnika

Dopunićemo model User.cs sa validacijama uz pomoć anotacije [Required(ErrorMessage = "Username is required!"), MinLength(2), MaxLength(10)]. Ovde smo definisali da je polje Username obavezno i da mora da bude između 2 i 10 karaktera dužine.

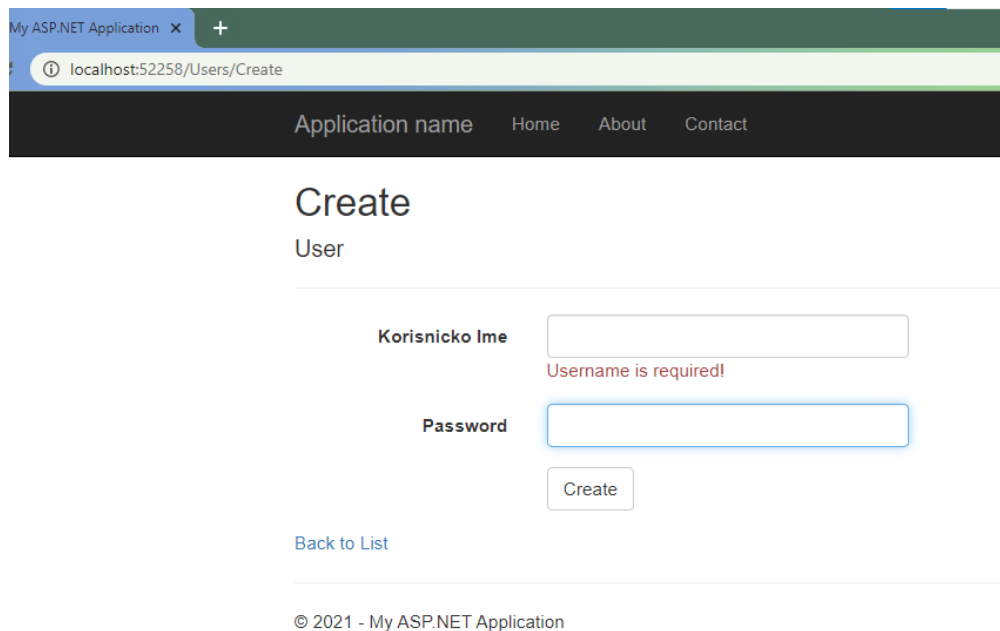
```
In [ ]: public class User
        {
            [DisplayName = "Korisnicko Ime"]
            [Required(ErrorMessage = "Username is required!"), MinLength(2), MaxLength
            (10)]
            public string Username { get; set; }
            public string Password { get; set; }
        }
```

Ako sad pokušate da unesete nevalidan unos za korisničko ime prikazaće vam se sledeći ispis (Slika 15., Slika 16.)



The screenshot shows a web browser window with the address bar displaying 'localhost:52258/Users/Create'. The page has a dark header with 'Application name', 'Home', 'About', and 'Contact' links. The main content area is titled 'Create User'. It contains two input fields: 'Korisnicko Ime' (Username) and 'Password'. The 'Korisnicko Ime' field contains the letter 'a'. Below this field, a red error message reads: 'The field Korisnicko Ime must be a string or array type with a minimum length of '2''. The 'Password' field is empty. A 'Create' button is located below the password field. At the bottom left, there is a 'Back to List' link. At the bottom center, the footer text reads '© 2021 - My ASP.NET Application'.

Slika 15. Neispravan unos novog korisnika



This screenshot is identical to the one above, showing the 'Create User' form with the same validation error for the username. The browser window title is 'My ASP.NET Application'. The error message 'Username is required!' is visible below the 'Korisnicko Ime' field. The 'Create' button and 'Back to List' link are also present.

Slika 16. Neispravan unos novog korisnika

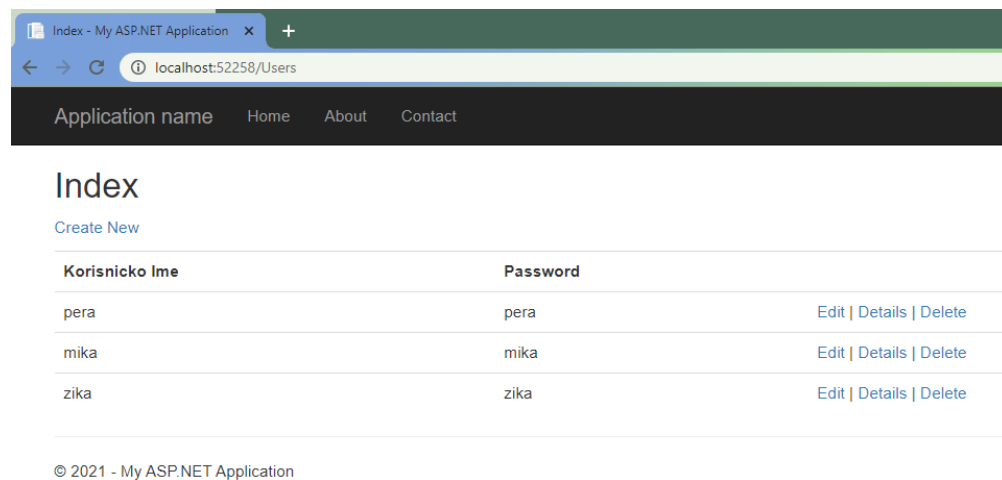
Na kraju dopunićemo UsersController.cs akciju // POST: Users/Create sa sledećim kodom:

```
In [ ]: // POST: Users/Create
[HttpPost]
public ActionResult Create(User user)
{
    try
    {
        // TODO: Add insert logic here

        List<User> users = (List<User>)HttpContext.Application["users"];
        users.Add(user);

        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}
```

Nakon ispravnog unosa novog korisnika treba da nam se prikaže na Index.cshtml stranici novi korisnik (Slika 16.)



Slika 16. Nov dodat korisnik zika