

Web-aplikacije

ASP.NET Core MVC

2016/17.08

➤ Što je web aplikacija?

- Programska aplikacija kojoj se pristupa preko internetskog preglednika ili drugog programa koji implementira HTTP (Hyper Text Transfer Protocol)

➤ Da li je skup web stranica ujedno i web aplikacija?

- Aplikacija koristi programsku logiku da bi prikazala sadržaj korisniku
- Kod web aplikacija obično se izvršava neki programski kod na serveru
- Primjer: statički cjenik nije aplikacija; dinamički cjenik je aplikacija

➤ Elementi web stranice

- HTML (HyperText Markup Language): osnovni jezik za definiranje web stranica
- JavaScript: jezik za klijentske skripte koje izvodi preglednik
- Programski kôd (u nekom jeziku) koji se izvodi na poslužitelju
 - Baza podataka + programska logika za generiranje web stranica, identifikaciju i autorizaciju korisnika, ...

Karakteristike web-aplikacija

3

- Izvršava se i na poslužitelju i na klijentu
 - Klijentski dio mora biti napisan u nekom od jezika koji Web preglednik podržava (HTML, JavaScript).
 - Ograničen pristup resursima na strani klijenta (npr. JavaScript koji se izvršava unutar preglednika ne može čitati s korisnikovog diska)
- Prednosti
 - Korisnik može biti bilo tko s pristupom Internetu
 - nema instalacijske procedure na strani klijenta
 - koriste se na bilo kojem OS, koji ima internetski preglednik
 - Jednostavno održavanje i nadogradnja na novu verziju
- Nedostatci
 - Složenija izrada u odnosu na samostojne klijentske aplikacije
 - Potreba za posebno dizajniranim sučeljem (Web design)
 - Mogući problemi pri prikazu u različitim preglednicima
 - Potrebna prilagodba regionalnim posebnostima korisnika
 - Sigurnosni problemi (neovlašten pristup aplikaciji i poslužitelju, zatrpavanje prometom)

Kratka povijest razvoja web-aplikacija

...ili zašto danas neke stvari (ne) radimo na određeni način...

Interaktivnost na klijentskoj strani

5

- „Duga” povijest, ali šira upotreba tek u nekoliko zadnjih godina
- JavaScript 1995. godine
 - Nakadno se pojavljuju (s više ili manje uspjeha) Flash, Java Applets i Silverlight
- jQuery – prva verzija 2006. godine
 - Verzija 1.5: 2011.godine
 - Verzija 2.0: 2013. godine
 - Verzija 3.1: 7. mjesec 2016.
- HTML 5 2014. godine
- CSS predložen 1994., objavljen prvi put 1996.
 - Aktualna verzija CSS 3 (CSS 4?)

CGI skripte

6

- Začeci između 1993. i 1995. – protokol Common Gateway Interface
 - CGI skripte, obično u nekom od skriptnih jezika (npr. Perl)
 - Skripta na standardni izlaz ispisuje HTML dokument koji web server preusmjerava korisniku
 - Parametri se prenose *query stringom*
 - parovi oblika ključ=vrijednost iza znaka upitnik u adresi zahtjeva
 - U skripti dostupan unutar varijable okruženja QUERY_STRING

```
#!/bin/sh
echo "Content-type: text/html"
echo
echo "<html> <head> <title> CGI script </title> </head> <body>"
echo "Sadržaj query stringa: $QUERY_STRING <br>"
echo "</body>"
```

- U POST varijanti parametri sadržani u tijelu HTTP zahtjeva
- Svaki poziv CGI skripte je novi proces

- 1995. i 1996. PHP i ASP (Active Server Pages), kasnije i Java Servlets
 - Web server ima dodatak koji omogućava izvršavanje više zahtjeva unutar istog procesa
 - Odsječci koji počinju s <?php odnosno <% izvršavaju se na serveru
 - Veće mogućnosti izvršavanja na serveru i mogućnost direktnog pisanja HTML-a (bez *echo* za svaki tekst)
- Stil pisanje aplikacije i dalje nalik CGI skriptama
 - Izmiješan HTML sadržaj sa serverskom logikom
 - Teško za održavanje i čitanje koda.

```
<% Do While Not RSSes.EOF And BrRedova>0 %>
<TR><TD VALIGN=TOP><%= absPos %>.</TD>
<TD VALIGN=TOP>
<a href="bib_details.asp?idRef=<%=RSSes("IdReference") %>">
<%=RSSes("AutoriRef") %></A>,
<%=RSSes("Godina") %><BR><%=RSSes("Naslov") &RSSes("Naslov2") %>
<% If existData(RSSes("Volumen")) Then %>
, Vol. <%= RSSes("Volumen") %>
<% End If %>
```

Pokušaji odvajanja prezentacijskih elemenata

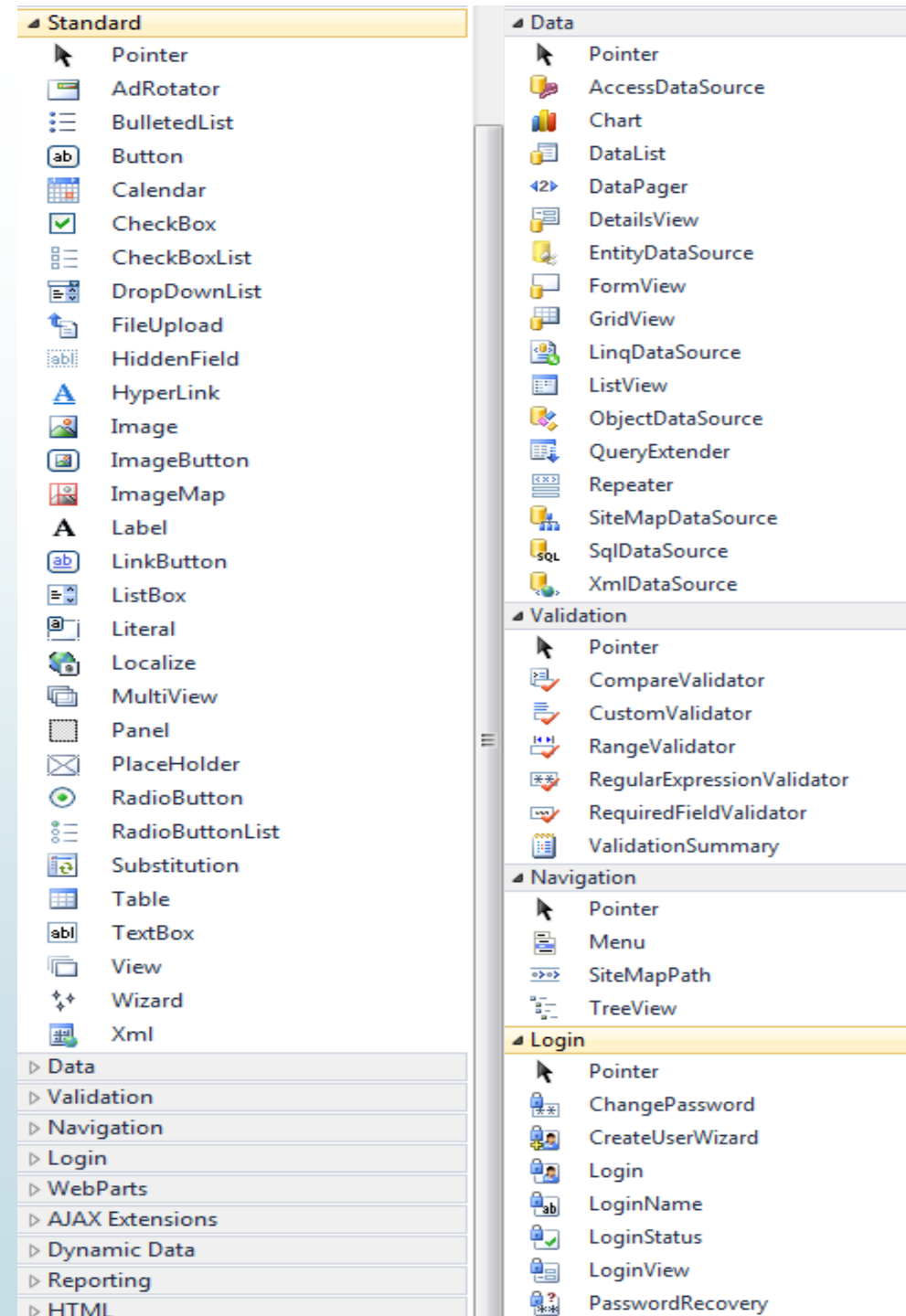
8

- Problem kako odvojiti prezentacijski dio od dohvata i pripreme podataka
- Izbjeći miješanje koda koji se izvršava na serveru i HTML-a koji se prikazuje korisniku.
- Kod PHP-a se počinju koristiti predlošci (npr. *Smarty*)
 - u posebnim datotekama kreiraju se predlošci s posebno označenim varijablama i oznaka
 - u PHP skripti se vrši dohvat podataka koji se predaje predlošku.
 - Iz današnje MVC perspektive moglo bi se reći da je PHP skripta pripremila model koji se onda prezentirao pogledom pisanom u smarty-u.
- Microsoft umjesto ASP-a 2002.g. stvara ASP.NET Web Forms
 - Uspješan u svoje vrijeme, jer je nudio jednostavan način izrade web aplikacija, omogućen i onima koji su imali oskudna znanja HTML-a

ASP.NET Web Forms

9

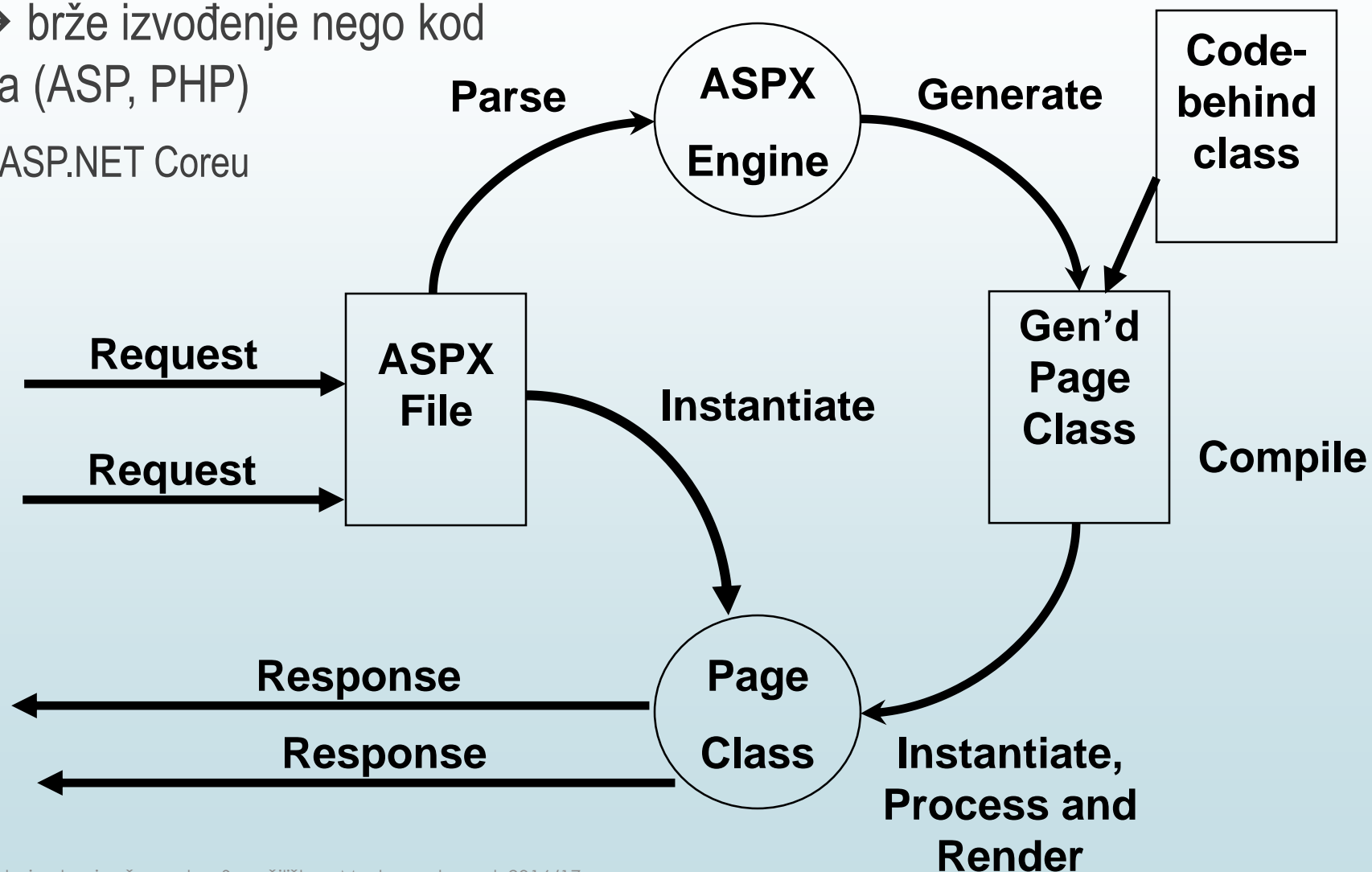
- cca 60 unaprijed definiranih serverskih kontrola
 - simulira se izrada web aplikacija nalik windows formama s ograničenjima web tehnologije
- Realizacija
 - Stranica.aspx + Stranica.aspx.cs + Stranica.aspx.designer.cs
 - .NET Framework vrši transformaciju kontrola u HTML
 - kontrole imaju mogućnosti HTML kontrola uz dodatnu funkcionalnost (npr. validacija)
 - Da bi se očuvao sadržaj stranice, podaci se prenose zajedno sa stranicom (rekonstrukcija sadržaja pomoću skrivenog polja __VIEWSTATE)



Prevođenje i prikaz stranica

10

- poslužiteljska logika se piše u nekom od .NET jezika
- prevođenje → brže izvođenje nego kod skriptnih jezika (ASP, PHP)
 - aktualno i u ASP.NET Coreu



WebForms - Odvajanje dizajna od koda

11

➤ Datoteka s kontrolama - Default.aspx

- ASPX stranica uobičajeno sadrži HTML, serverske kontrole, JavaScript i ostale prezentacijske elemente
- Source i/ili Design pogled na istu stranicu

The screenshot displays the Visual Studio IDE with the Source view of the Default.aspx file. The code is as follows:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transiti
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Pozdrav (web)</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:Button ID="Button1" runat="server"
            Text="Klikni me" OnClick="Button1_Click" />
        <hr /><asp:Label ID="Label1" runat="server"/>
    </form>
</body>
</html>
```

The Client Objects & Events window on the left shows "(No Events)". The Solution Explorer on the right shows the project structure:

- Default.aspx
- Default.aspx.cs
- Default.aspx.designer.cs

The bottom of the window shows the Design, Source, and <html> tabs, with the Source tab selected.

Odvajanje dizajna od koda

12

- Datoteka s kôdom (CodeFile, Code behind) – Default.aspx.cs i Default.aspx.designer.cs
 - parcijalni razred
 - sadrži kôd koji se izvršava na serveru

```
public partial class Default : System.Web.UI.Page {  
    protected void Page_Load(object sender, EventArgs e){  
    }  
    protected void Button1_Click(object sender, EventArgs e)  
    {  
        Label1.Text += DateTime.Now.ToString();  
    }  
}
```

- Klikom na gumb izvodi se postupak Page_Load (ako postoji), a zatim slijede obrade događaja
 - Rekonstrukcija sadržaja kontrola pri svakom zahtjevu
 - Npr. U gornjem primjeru klikom na gumb se na postojeći tekst (sa svim prethodnim nadopisivanjima) nadopiše trenutno vrijeme

Ključne karakteristike Web Formi

13

- Za svaku kontrolu automatski se generira odgovarajuća HTML kontrola
- Automatski se rekonstruira stanje/sadržaj kontrola između dvaju zahtjeva na server (koristi se skriveno polje __ViewState)
- Glavna stranica (Master)
 - Predstavlja zajednički izgled (okvir, dizajn) za više stranica
 - Uobičajeno sadrži zajedničke elemente (izbornike, zaglavlje, podnožje, ...)
 - Web aplikacija može imati više master stranica
 - Može sadržavati što i obična stranica uz jedan ili više okvira
- Povezivanje podataka
 - Kontrola se veže na neki ObjectDataSource (ili neki drugi DataSource) kojem su definirani CRUD postupci + stranicenja, sortiranja i slično
 - Automatsko kreiranje tablica s podacima + kontrole za Edit/Update/Cancel i prelaske među njima
 - Korisnik piše samo postupke za manipulaciju podacima, ne i postupke za prihvatanje podataka s prezentacijskog sučelja
- Izrada stranice u dizajnu ili direktnim pisanje koda

Nedostatci ASP.NET web formi

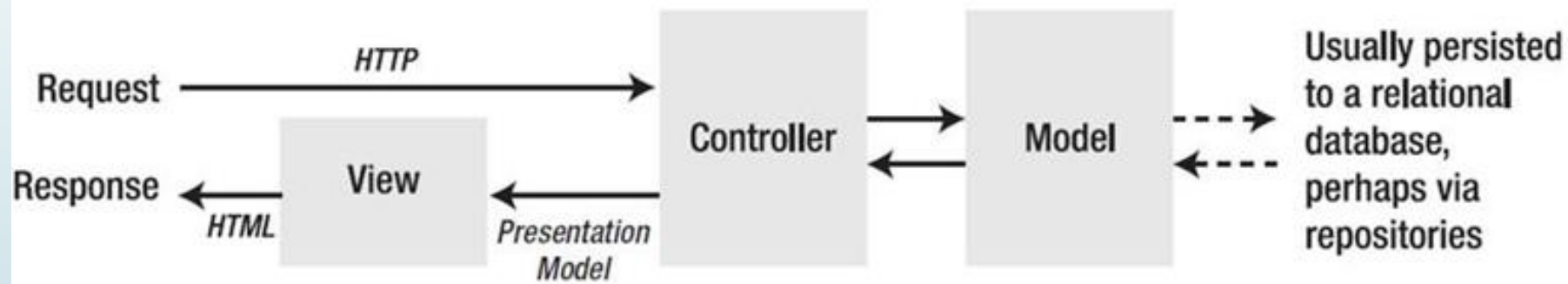
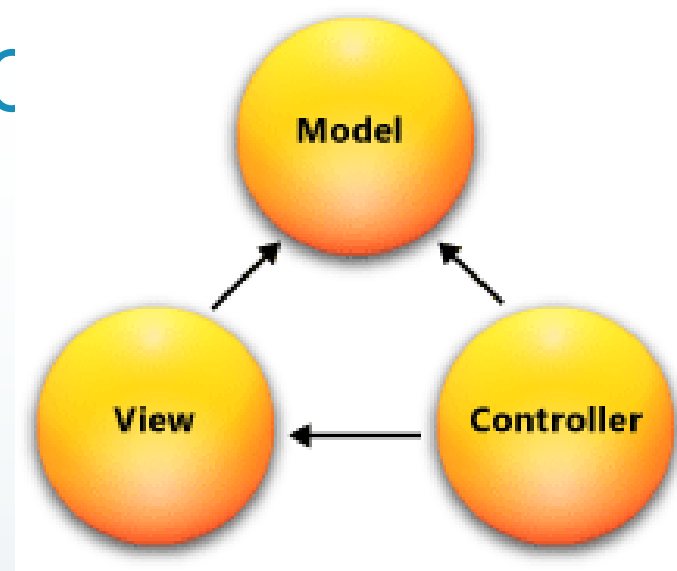
14

- ViewState proizvodi veliku količinu podataka pri svakom zahtjevu
 - Komplikiran životni ciklus stranice
 - Ograničena kontrola nad proizvedenim HTML-om
 - Prevelike *.cs datoteke uz često miješanje prezentacijske i aplikacijske logike
 - Slaba mogućnost testiranja
-
- ASP.NET MVC kao rješenje navedenih problema
 - Objedinjena iskustva MVC implementacija u drugim jezicima
 - MVC kao koncept nastao u kasnim sedamdesetim godinama prošlog stoljeća (Smalltalk projekt unutar Xeroxa)

MVC

ASP.NET Core MVC – osnovne postavke

- Model - Pogled - Upravljač
- Detaljnija razrada prezentacijskog sloja
 - Pogled definira izgled korisničkog sučelja
 - Obično vezan za objekt iz modela
- Upravljač predstavlja prezentacijsku logiku
 - Prima ulaz iz pogleda, obrađuje ga, puni i dohvaća model poziva niže slojeve i određuje redoslijed prikaza pogleda



- U jednostavnim aplikacijama model objedinjava i poslovnu logiku i sloj pristupa podacima
- U složenijim se kao model koristi “pravi” poslovni model, a model unutar projekta služi za definiranje pomoćnih modela za lakši prikaz (“prezentacijski model”, a ne model u smislu poslovnog objekta)

Prednost MVC-a nad web formama

17

- Smanjena složenost podjelom aplikacije u model, pogled i upravljač
- Ne koristi se ViewState ni serverske kontrole čime se omogućava potpuna kontrola ponašanja aplikacije.
- Zahtjevi centralizirani na jedan upravljač - Front Controller pattern
 - bogata podrška za interpretiranje/usmjeravanje zahtjeva
 - WebForms koriste Page Controller pattern
- Podrška za test-driven development (TDD).
- Dobar okvir i za veće razvojne timove i za dizajnere

- Podrжан na različitim platformama (Windows, Linux, MacOS, Docker)
- Ključne promjene u odnosu na „stari” ASP.NET MVC:
 - Tag-helperi za formiranje poveznica i kontrola
 - proširuju postojeće HTML kontrole dodatnim atributima koji se koriste prilikom generiranja stranica, npr. za formiranje poveznica, popunjavanje polja za unos podatka i slično (više naknadno)
 - Veće mogućnosti konfiguracijskih datoteka
 - Intenzivno korištenje tehnike *Dependency Injection*
 - posljedično lakše testiranje i veća modularnost

Stvaranje nove web-aplikacije

19

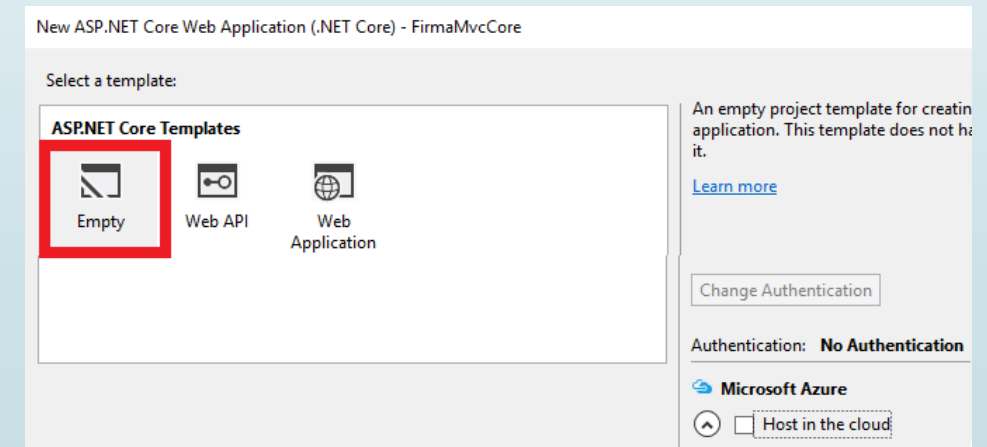
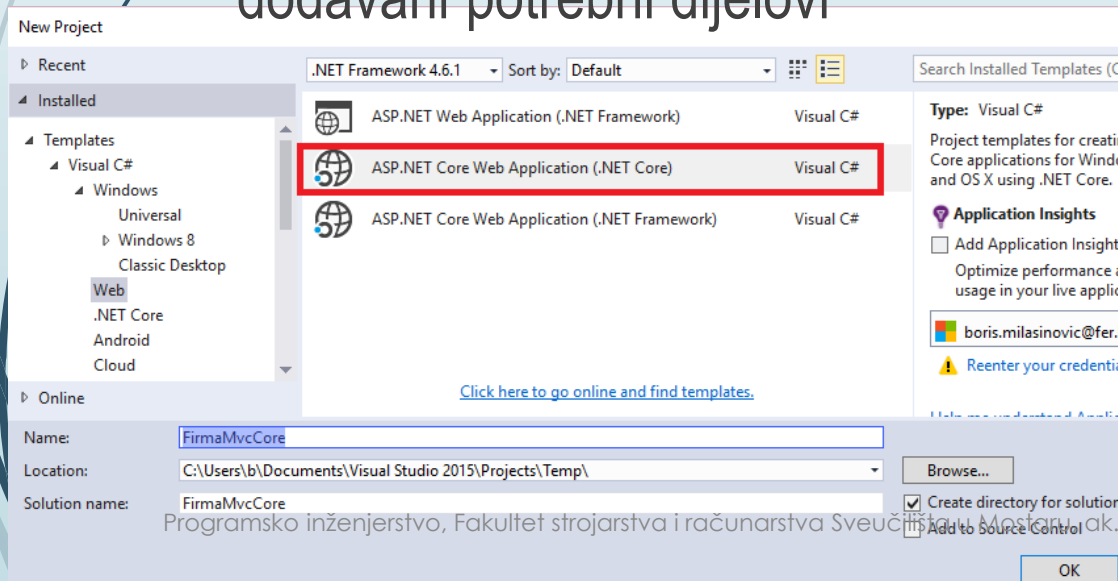
➤ Iz naredbenog retka

- `dotnet new mvc --auth None -n Naziv` (obrisati višak)
- `dotnet new web -n Naziv` (dodati potrebne pakete i konfigurirati za MVC)

➤ Koristeći razvojno okruženje

- File → New Project → Web → ASP.NET Core Web Application (.NET Core)
 - Empty (ili Web Application) uz opciju No Authentication


➤ U primjeru koji slijedi bit će stvorena prazna web aplikacija, pa će postupno biti dodavani potrebni dijelovi



Sadržaj „prazne” web-aplikacije

20

➤ Samo Program.cs i Startup.cs.

- Web-aplikacija se izvršava na privremenom web-serveru (localhost:port)
- Ispisuje *HelloWorld* - definirano u postupku *Configure*
- Primjer:  BiloKojaPraznaAplikacija \ Startup.cs


```
public void Configure(IApplicationBuilder app, IHostingEnvironment env,
                      ILoggerFactory loggerFactory) {
    loggerFactory.AddConsole();

    if (env.IsDevelopment()) {
        app.UseDeveloperExceptionPage();
    }

    app.Run(async (context) =>
    {
        await context.Response.WriteAsync("Hello World!");
    });
}
```

Postavke pokretanja programa

21

- Program.cs je „glavni program” web-aplikacije
 - ASP.NET Core koristi Kestrel kao svoj web server
 - Može se spojiti s klasičnim web serverom (IIS, Apache i slično)
 - Postavljaju se ostale postavke pokretanja i određuje razred koji konfigurira postavke web-aplikacije
 - razred Startup ako nije ništa eksplicitno mijenjano
 - Primjer:  Web \ Firma.Mvc \ Program.cs

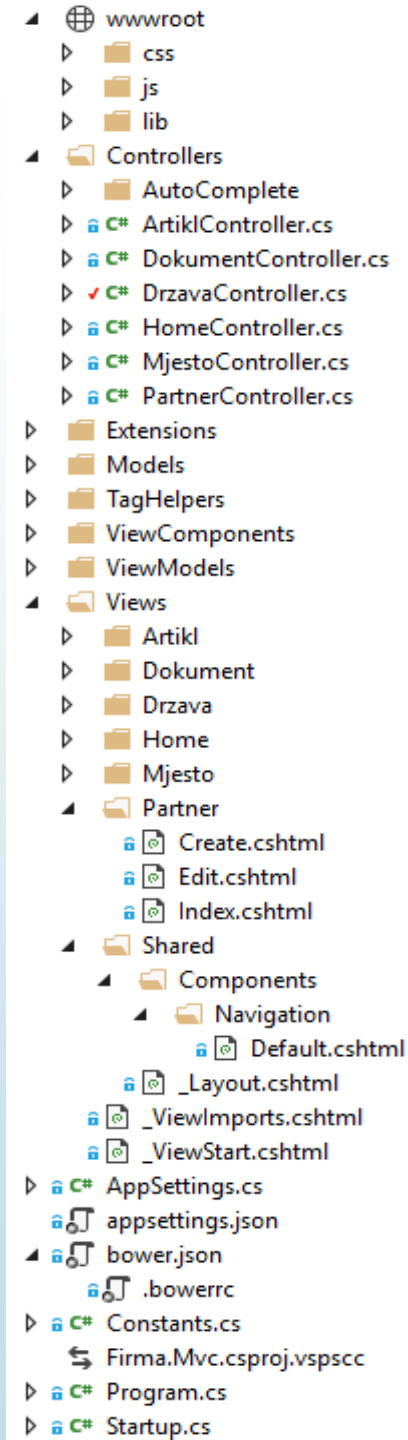
```
public class Program{  
    public static void Main(string[] args) {  
        var host = new WebHostBuilder()  
            .UseKestrel()  
            .UseContentRoot(Directory.GetCurrentDirectory())  
            .UseIISIntegration()  
            .UseStartup<Startup>()  
            .Build();  
  
        host.Run();  
    }  
}
```

Konačna struktura ogleadne aplikacije

22


➤ Uobičajene mape i datoteke

- *wwwroot*: statički sadržaj (css, skripte, klijentske biblioteke)
- *Controllers*: upravljači unutar aplikacije
- *Views*: pogledi podijeljeni u podmape za svaki upravljač i Shared za zajedničke poglede (npr. glavna stranica)
- *Models*: razredi koji predstavljaju domenske modele (u slučaju manje aplikacije)
- *ViewModels*: razredi koji služe za prijenos podataka pogledu i prihvrat podataka iz pogleda
 - prezentacijski modeli
- *TagHelpers*: vlastiti razredi s atributima kojim se proširuju uobičajene HTML oznake
- *ViewComponents*: komponente (dijelovi aplikacije) koje se koriste na više mjesta, dizajniraju se zasebno te se uključuju u pojedinim pogledima
- Konfiguracijske i projektne datoteke
- Ostali razredi po potrebi



Paketi klijentskih biblioteka

23

- NuGet za pakete koji se izvršavaju na serveru
- Bower za klijentske biblioteke
- Datoteka .bowerrc sadrži postavke alata Bower
 - Npr. lokaciju gdje će se pohraniti preuzeti sadržaji paketa
 - Primjer:  Web \ Firma.Mvc \ .bowerrc

```
{  
  "directory": "wwwroot/lib"  
}
```

Uključivanje klijentskih biblioteka

24

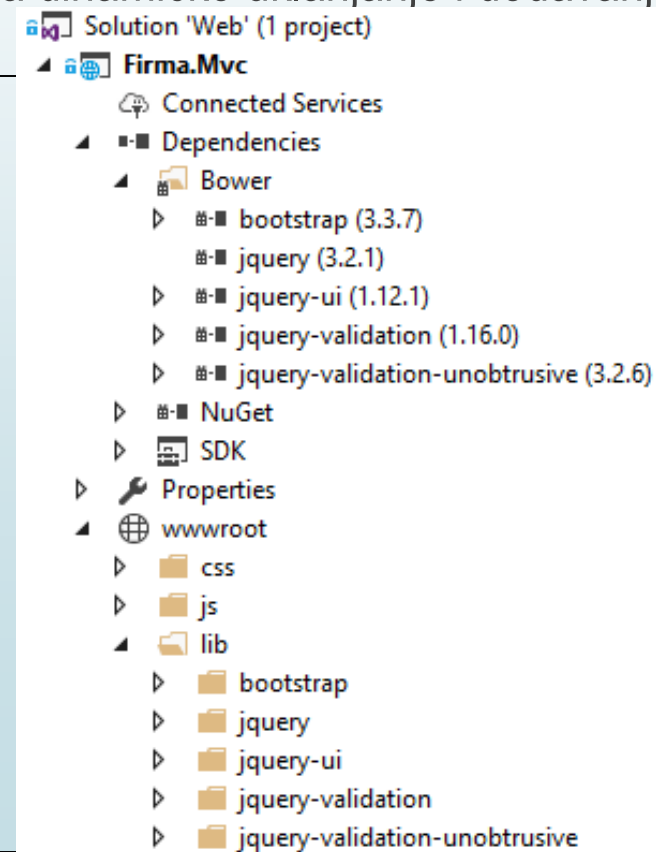
➤ Desni klik na projekt → Manage Bower Packages

➤ ili ažuriranjem datoteke bower.json

➤ Primjer:  Web \ Firma.Mvc \ bower.json

➤ Primjer koji slijedi koristi bootstrap za stil i ikone, te jQuery biblioteke za validaciju na klijentskoj strani, prikaz kalendara umjesto direktnog unosa datuma te za dinamičko uklanjanje i dodavanje redaka prilikom ažuriranja

```
{  
  "name": "asp.net",  
  "private": true,  
  "dependencies": {  
    "bootstrap": "v3.3.7",  
    "jquery": "3.2.1",  
    "jquery-validation": "1.16.0",  
    "jquery-validation-unobtrusive": "3.2.6",  
    "jquery-ui": "1.12.1"  
  }  
}
```



NuGet paketi za MVC

25

➤ Potrebno dodati sljedeće pakete

- Microsoft.AspNetCore.Mvc
 - Podrška za MVC u ASP.NET Coreu
- Microsoft.AspNetCore.Session
 - Omogućuje sjednicu unutar web-aplikacije (koristi se za TempData)
- Microsoft.AspNetCore.StaticFiles
 - Uključuje podršku za posluživanje statičkih datoteka (npr. za stilove i vlastite skripte)
 - pretpostavljena vršna lokacija je wwwroot



Microsoft.AspNetCore.Mvc by Microsoft

✓ v1.1.2

ASP.NET Core MVC is a web framework that gives you a powerful, patterns-based way to build dynamic websites and w...



Microsoft.AspNetCore.Session by Microsoft

✓ v1.1.1

ASP.NET Core session state middleware.



Microsoft.AspNetCore.StaticFiles by Microsoft

✓ v1.1.1

ASP.NET Core static files middleware. Includes middleware for serving static files, directory browsing, and default files.

Uključivanje podrške za MVC

26

- U skup definiranih servisa dodati i aktivirati podršku za MVC i sjednicu
- Ukloniti odsječak za ispis *Hello World* u svakom zahtjevu
 - Dodati postavke usmjerenja
- Primjer:  Web \ Firma.Mvc \ Startup.cs

```
public void ConfigureServices(IServiceCollection services) {  
    services.AddMvc();  
    services.AddSession();  
    ...  
  
public void Configure(...) {  
    ...  
    app.Run(async (context) => {  
        await context.Response.WriteAsync("Hello World!");  
    });  
    app.UseSession();  
    app.UseStaticFiles();  
    app.UseMvc(routes => ...vidi sljedeći slajd ...);
```

Način rada MVC aplikacije


27

- Korisnik od MVC aplikacije traži akciju upravljača, nakon čega aplikacija (tj. pozvani upravljač) popunjava model i vraća ažurirani pogled korisniku
 - Kako prepoznati koja akcija i koji upravljač se zahtijeva?
 - Na osnovu URL-a i definiranih usmjeravanja
- Primjer:  Web \ Firma.Mvc \ Startup.cs

```
public void Configure(...) {  
    ...  
    app.UseMvc(routes =>  
    {  
        routes.MapRoute(null, "Artikl/Page{page}",  
            new { Controller = "Artikl", action = "Index" });  
        routes.MapRoute(null, "Mjesto/Page{page}",  
            new { Controller = "Mjesto", action = "Index" });  
        routes.MapRoute(  
            name: "default",  
            template: "{controller=Home}/{action=Index}/{id?}");  
    });  
}
```

Primjeri usmjeravanja

28

- `http://.../NazivUpravljača/Akcija/OpcionalniParametri...`
 - Za traženi naziv upravljača mora postojati razred `NazivController.cs` u mapi `Controllers`
 - Za traženu akciju u nekom upravljaču mora postojati postupak (s potrebnim parametrima) naziva jednakog traženoj akciji
 - Moguće definirati više istoimenih akcije za različite varijante zahtjeva (npr. GET i POST)
- Primjer:  `Web \ Firma.Mvc`
 - `http://.../Artikl/Edit/3`
 - Upravljač `Artikl(Controller)` ima postupak `Edit` koji za parametar `id` ima 3
 - `http://.../Artikl/Page10`
 - Upravljač `Artikl` izvršava pretpostavljeni postupak (`Index`), vrijednost parametra `page` je 10
 - `http://.../Dokument/Details?page=7`
 - Upravljač `Dokument(Controller)` izvršava `Details`, a za vrijednost parametra `page` uzima vrijednost 7
 - `http://.../` izvršava akciju `Index` na upravljaču `Home`

Početna stranica (1)

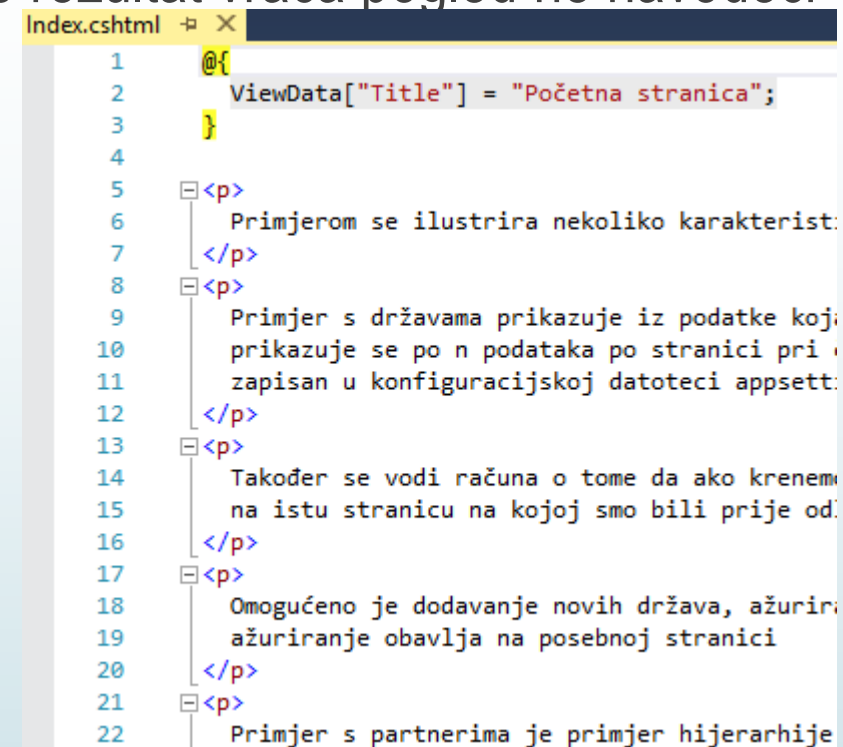
29

- Programski kod akcije Index na upravljaču Home kao rezultat vraća pogled ne navodeći ime pogleda

- Tip rezultata je *ActionResult*
 - Može se specificirati i konkretniji
- Podrazumijeva se ime pogleda jednako nazivu postupka
 - Pogled se očekuje u datoteci Views \ Home \ Index.cshtml ili istog imena negdje u mapi Shared
- O sintaksi pogleda na kasnijim slajdovima

- Primjer:  Web \ Firma.Mvc \ Controllers \ HomeController.cs

```
public class HomeController : Controller {  
    public ActionResult Index() {  
        return View();  
    }  
}
```



```
Index.cshtml  + X  
1  @{  
2      ViewData["Title"] = "Početna stranica";  
3  }  
4  
5  <p>  
6      Primjerom se ilustrira nekoliko karakterist  
7  </p>  
8  <p>  
9      Primjer s državama prikazuje iz podatke koji  
10     prikazuje se po n podataka po stranici pri  
11     zapisan u konfiguracijskoj datoteci appsett  
12 </p>  
13 <p>  
14     Također se vodi računa o tome da ako krenem  
15     na istu stranicu na kojoj smo bili prije od  
16 </p>  
17 <p>  
18     Omogućeno je dodavanje novih država, ažurir  
19     ažuriranje obavlja na posebnoj stranici  
20 </p>  
21 <p>  
22     Primjer s partnerima je primjer hijerarhije
```

Uobičajeni rezultati akcije upravljača (izvedeni iz *ActionResult*)

30

Tip	Opis rezultata/povratne vrijednosti	Postupak u upravljaču
ViewResult	Prikazuje pogled	View
PartialViewResult	Prikazuje parcijalni pogled	PartialView
RedirectToRouteResult	Privremeno ili trajno preusmjerava zahtjev (HTTP kod 301 ili 302), stvarajući URL na osnovu postavki usmjeravanja	RedirectToAction RedirectToActionPermanent RedirectToRoute RedirectToRoutePermanent
RedirectResult	Privremeno ili trajno preusmjerava rezultat na određeni URL	Redirect RedirectPermanent
ContentResult	Vraća tekstualni sadržaj	Content
FileResult	Vraća binarni sadržaj	File
JsonResult	Vraća objekt serijaliziran u JSON format	Json
JavaScriptResult	Vraća JavaScript odsječak	JavaScript
HttpUnauthorizedResult	Vraća HTTP kod 401	-
HttpNotFoundResult	Vraća HTTP kod 404	HttpNotFound
HttpStatusCodeResult	Vraća određeni HTTP kod	-
EmptyResult	Bez povratne vrijednosti	-

Sintaksa pogleda

31

- Ako drugačije nije navedeno koristi se pogled ekstenzijom cshtml čije ime odgovara pozvanoj akciji, a nalazi se u mapi Views \ Pozvani upravljač
 - Pogled predstavlja mješavinu html i mvc koda
 - U pogledu se obično koristi jednostavni kod (npr. petlja, grananje i sl) vezan uz prikaz
 - MVC kod počinje s @ iza kojeg slijedi naredba ili blok naredbi unutar vitičastih zagrada i html oznake
 - Komentari oblika @* *@
 - Početni redak pogleda (opcionalno) sadrži podatak koji se model koristi
 - *@model naziv razreda* koji se koristi za model
 - Konkretni vrijednosti predanog modela dobije se s *@Model*
 - Tekst izvan html oznake prefiksira se s @: ili se stavlja oznaka <text>
 - Prostori imena uključuju se s @using
 - Često korišteni mogu se dodati u datoteku Shared_ViewImports.cshtml
- Detaljnije o sintaksi pogleda: <https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor>
- Pogled može biti parcijalni
 - nema html zaglavlje niti koristi glavnu stranicu

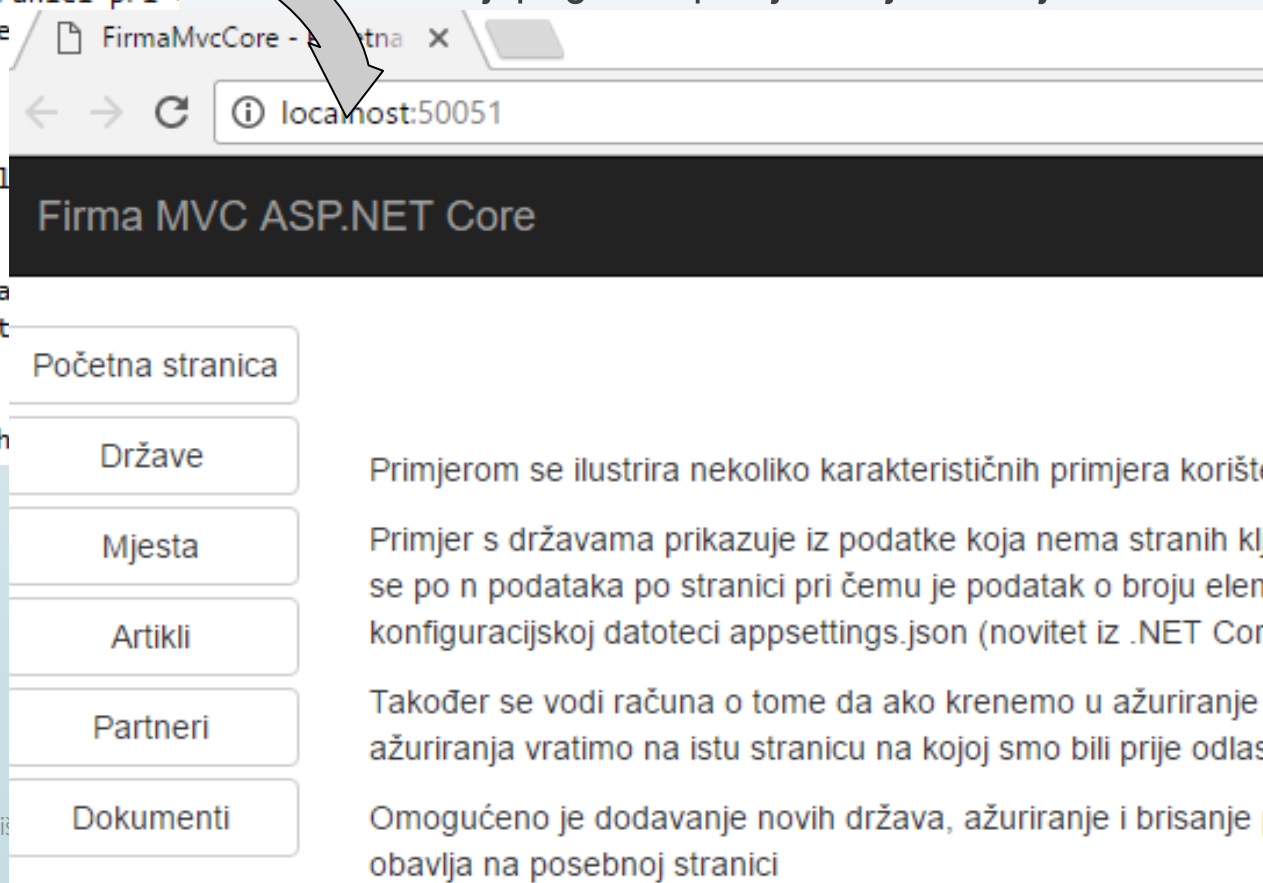
Početna stranica (2)

32

```
Index.cshtml  X
1  @{
2      ViewData["Title"] = "Pocetna stranica";
3  }
4
5  <p>
6      Primjerom se ilustrira nekoliko karakteristika
7  </p>
8  <p>
9      Primjer s državama prikazuje iz tablice koja
10     prikazuje se po n podataka po stranici pri čemu je podatak o broju elemenata
11     zapisan u konfiguracijskoj datoteci appsettings.json (novitet iz .NET Core)
12  </p>
13  <p>
14     Također se vodi računa o tome da ako krenemo u ažuriranje novih država,
15     ažuriranja vratimo na istu stranicu na kojoj smo bili prije odlaska
16  </p>
17  <p>
18     Omogućeno je dodavanje novih država, ažuriranje i brisanje podataka
19     obavlja na posebnoj stranici
20  </p>
21  <p>
22     Primjer s partnerima je primjer h
```

➤ Nakon pokretanja aplikacije početna stranica sadrži naslov, ali i dijelove koji nisu navedeni u Index.cshtml

➤ Koristi se tzv. glavna stranica koja predviđa okvir u koji pogledi upisuju svoj sadržaj



- Pretpostavljena glavna stranica za svaki pogled definirana u datoteci Views_ViewStart.cshtml

- Primjer:  Web \ Firma.Mvc \ Views \ _ViewStart.cshtml

```
@{  
    Layout = "_Layout";  
}
```

- Očekuje se postojanje Views \ Shared \ _Layout.cshtml

- Svaki pogled po potrebi može definirati svoju glavnu stranicu ili je uopće ne koristiti promjenom vrijednosti svojstva *Layout*

```
@{  
    Layout = null;  
}
```

- Ako se unutar pojedinog pogleda ne postavi vrijednost za Layout koristi se pretpostavljena glavna stranica
- Glavna stranica uključuje stil i javascript datoteke, definira navigaciju, prostor za javascript pojedinog pogleda...
 - Konkretni sadržaj za neki zahtjev dobit će se pomoću RenderBody

Primjer glavne stranice


34

► Primjer:  Web \ Firma.Mvc \ Views \ Shared \ _Layout.cshtml

```
<!DOCTYPE html>
<html lang="hr_HR" xml:lang="hr_HR"...>
<head>
  <title>FirmaMvcCore - @ViewData["Title"]</title>
  <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />
  <link rel="stylesheet" href="~/css/site.css" />
  @RenderSection("styles", required: false)
</head><body>
  ... @await Component.InvokeAsync("Navigation") ...
  ...
  @RenderBody()
  ...
  <script src="~/lib/jquery/dist/jquery.js"></script>
  ...
  <script src="~/js/site.js" asp-append-version="true"></script>
  ...
  @RenderSection("scripts", required: false)
</body></html>
```

Vlastite komponente (1)

35

- Za sadržaje koji se pojavljuju na više mjesta, primjerice izbornik
- Komponente se izvode iz apstraktnog razreda *ViewComponent*, a rezultat se priprema u postupku *Invoke* (vraća rezultat tipa *IViewComponentResult*)
 - Odgovarajući „pogled” po sintaksi identičan ostalima
- Primjer:  ...\ ViewComponents \ NavigationViewComponent.cs
 - Poveznica na trenutni upravljač će biti drugačije označena, pa je u postupku očitana aktualna vrijednost ako postoji
 - `referenca?.član` vraća član objekta na kojeg `referenca` pokazuje ili `null` ako je `referenca` `null`
 - `referenca.clan` bi mogao izazvati `NullReferenceException`
 - O `ViewBagu`-u naknadno

```
public class NavigationViewComponent : ViewComponent {  
    public IViewComponentResult Invoke() {  
        ViewBag.Controller = RouteData?.Values["controller"];  
        return View();  
    }  
}
```

► Primjer:  Web\Firma.Mvc\Views\Shared\Components\Navigation\Default.cshtml


```
<a class="btn btn-block btn-default"
    asp-action="Index" asp-controller="Home">
    Početna stranica
</a>
<a class="btn btn-block
    @(ViewBag.Controller == "Drzava" ? "btn-primary": "btn-default") "
    asp-action="Index" asp-controller="Drzava">
    Države
</a>
...
```

► Poveznice se formiraju tzv. tag-helperima.

- Na standardnu HTML oznaku dodaju se atributi `asp-action`, `asp-controller`, `asp-route-nazivparametra` i slično, na osnovi kojih nastane konkretna poveznica

Konfiguracijske datoteke

37

- Kao i kod konzolne aplikacije može biti proizvoljna datoteka
 - Može se koristiti varijanta s umetnutim nazivom okruženja
- Primjer:  Web \ Firma.Mvc \ Startup.cs

```
public class Startup {  
    public IConfigurationRoot Configuration { get; }  
  
    public Startup(IHostingEnvironment env) {  
        var builder = new ConfigurationBuilder()  
            .AddUserSecrets("Firma")  
            .SetBasePath(env.ContentRootPath)  
            .AddJsonFile("appsettings.json",  
                optional: true, reloadOnChange: true)  
            .AddJsonFile($"appsettings.{env.EnvironmentName}.json",  
                optional: true)  
            .AddEnvironmentVariables();  
        Configuration = builder.Build();  
    }  
}
```

Preslikavanje konfiguracijske datoteke u vlastiti razred

38

- ➡ Definira se vlastiti razred koji svojom strukturom prati JSON sadržaj (ili neki njegov dio) iz konfiguracijske datoteke

- ➡ Primjer:  Web \ Firma.Mvc \ AppSettings.cs


```
public class AppSettings {  
    public int PageSize { get; set; } = 10;  
    public int PageOffset { get; set; } = 10;  
    public string ConnectionString { get; set; }  
}
```

- ➡ Primjer:  Web \ Firma.Mvc \ appsettings.json

```
{  
  "AppSettings": {  
    "PageSize": 10,  
    "PageOffset": 5,  
    "ConnectionString":  
    "Server=rppp.fer.hr,3000;Database=Firma;user  
id=rppp;password=sifra"  
  }  
}
```

Dohvat vrijednosti iz konfiguracijske datoteke

39

- Umjesto dohvata korištenjem stringa koji predstavlja putanju u JSON konfiguracijskoj datoteci koristi se vlastiti razred
- Primjer:  Web \ Firma.Mvc \ Startup.cs

```
public void ConfigureServices(IServiceCollection services) {  
    ...  
    //application settings  
    services.AddOptions();  
    var appSection = Configuration.GetSection("AppSettings");  
    services.Configure<AppSettings>(appSection);  
}
```

Podsjetnik: Stvaranje EF modela na osnovu postojeće BP

40

1. U mapi ciljanog projekta izvršiti sljedeće naredbe

```
dotnet add package Microsoft.EntityFrameworkCore.Design  
dotnet add package Microsoft.EntityFrameworkCore.SqlServer.Design  
dotnet add package Microsoft.EntityFrameworkCore.Tools
```

➡ ili dodati koristeći opciju Manage NuGet Packages

2. U csproj datoteku pod ItemGroup dodati:

```
<DotNetCliToolReference  
  Include="Microsoft.EntityFrameworkCore.Tools.DotNet" Version="1.0.0"  
/>
```

3. U naredbenom retku izvršiti

```
dotnet restore  
dotnet ef dbcontext scaffold "Server=rppp.fer.hr,3000;Database=Firma;User  
Id=rppp;Password=*" Microsoft.EntityFrameworkCore.SqlServer -o Models
```


Postavke za spajanje na bazu podataka

41

- Automatski generirani razred za EF kontekst sadrži tvrdo kodirane postavke za spajanje na bazu podataka u postupku *OnConfiguring*
 - Navedeni postupak zamijeniti konstruktorom koji primljeni parametar tipa *DbContextOptions* proslijedi baznom razredu (konkretni objekt bit će umetnut tehnikom *Dependency Injection*)

➤ Primjer:  Web \ Firma.Mvc \ Models \ Firma.Context.cs


```
protected override void OnConfiguring  
===== (DbContextOptions<FirmaContext> options) { ... }  
public FirmaContext(DbContextOptions<FirmaContext> options) :  
base(options) { }
```

➤ Primjer:  Web \ Firma.Mvc \ Startup.cs

```
public void ConfigureServices(IServiceCollection services) {  
    var appSection = Configuration.GetSection("AppSettings");  
    string connectionString =  
        appSection.GetValue<string>("ConnectionString");  
    connectionString = connectionString.Replace("sifra",  
                                                config["FirmaSqlPassword"]);  
  
    services.AddDbContext<Models.FirmaContext>(  
        options => options.UseSqlServer(connectionString)) ;
```

Akcija dohvata svih država (1)

42

- Primjer:  Web \ Firma.MVC \ Controllers \ DrzavaController.cs
 - Upravljač iz primjera vrši dohvat podataka koristeći Entity Framework Core
 - Kontekst primljen u konstruktoru
 - ASP.NET Core instancira pojedini upravljač te na osnovu postavki iz *Startup.ConfigureServices*, koristeći tehniku *Dependency Injection*, stvara potrebne argumente
 - Argumenti spremljeni kao varijable dostupne u akciji koja se poziva nakon konstruktora

```
public class DrzavaController : Controller
{
    private readonly FirmaContext ctx;
    private readonly AppSettings appData;

    public DrzavaController(FirmaContext ctx,
                           IOptions<AppSettings> options)
    {
        this.ctx = ctx;
        appData = options.Value;
    }
}
```

Akcija dohvata svih država (2)

43


➡ Primjer:  Web \ Firma.MVC \ Controllers \ DrzavaController.cs

➡ Upravljač dohvati podatke, napuni model (lista država) i izazove prikaz pogleda s imenom *IndexSimple*.

```
public class DrzavaController : Controller {  
    ...  
    public IActionResult Index() {  
        var drzave = ctx.Drzava  
            .AsNoTracking()  
            .OrderBy(d => d.NazDrzave)  
            .ToList();  
        return View("IndexSimple", drzave);  
    }  
}
```

Pregled svih država

44

- Primjer:  Web \ Firma.Mvc \ Views \ Drzava \ IndexSimple.cshtml
 - Tip modela je *IEnumerable<Drzava>* (može i *List<Drzava>*, ali nije potrebno)
 - Za svako mjesto definiran redak tablice s podacima o mjestu
 - Izgled retka definiran stilovima iz Bootstrapa
 - Vrijednost modela može se dohvatiti preko svojstva Model

```
@using Firma.Mvc.Models;
@model IEnumerable<Drzava>
...
<table class="table table-striped" id="tabledrzave">
...
@foreach (var item in Model)
{
    <tr>
        <td class="text-center">@drzava.OznDrzave</td>
        <td class="text-left">@drzava.NazDrzave</td>
        <td class="text-center">@drzava.Iso3drzave</td>
        <td class="text-center">@drzava.SifDrzave</td>
```

Straničenje na klijentskoj strani (1)

45

➡ Javascript biblioteka <https://datatables.net>

➡ Dostupna i kao plugin za Bower

➡ Aktivira se kodom u jQueryu, nakon što se dokument učitava

➡ Primjer:  Web \ Firma.Mvc \ Views \ Drzava \ IndexSimple.cshtml

```
...  
<table class="table table-striped" id="tabledrzave">  
... </table>  
@section styles{  
    <link rel="stylesheet" href="https://.../css/jquery.dataTables.min.css" />  
}  
@section scripts{  
    <script src=".../js/jquery.dataTables.min.js"></script>  
    <script>  
        $(document).ready(function() {  
            $('#tabledrzave').DataTable();  
        });  
    </script>  
}
```

Straničenje na klijentskoj strani (2)

46

➡ Brzo, jednostavno i vizualno privlačno rješenje...

➡ ali nije prikladno za velike količine podataka (svi podaci se uvijek dohvaćaju)

Firma MVC ASP.NET Core

Početna stranica

Države

Mjesta

Artikli

Partneri

Dokumenti

Popis država

Show 10 entries

Search:

Oznaka države	Naziv države	Iso3 oznaka	Šifra države		
HR	Croatia	HRV	191		
CU	Cuba	CUB	192		
CY	Cyprus	CYP	196		
CZ	Czech Republic	CZE	203		
DK	Denmark	DNK	208		
DJ	Djibouti	DJI	262		
DM	Dominica	DMA	212		
DO	Dominican Republic	DOM	214		
DD	Drzava				
EC	Ecuador	ECU	218		

Showing 51 to 60 of 236 entries

Previous 1 ... 5 6 7 ... 24 Next