

# Web programiranje

Dr. Goran Artonović

goran.aritonovic@bpa.edu.rs

kabinet 211

# Softver

- Visual Studio Community 2022 (17.13)
- Visual Studio Code
- SQL Server 2019 (2017) Express
- SQL Server Management Studio Express
- Docker Desktop

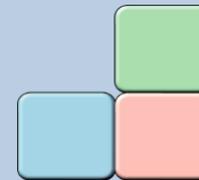
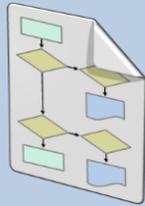
# Uvod u ASP.NET core MVC web aplikacije

# ASP.NET core

- ASP.NET core je besplatan, open-source web framework
- Modularan je i radi na Windows, Linux i macOS operativnim sistemima
- Trenutna verzija je ASP.NET Core 9.x i deo je .NET 9 platforme
- Distribuirana se kao deo .NET SDK-a (skup alata potrebnih za razvoj i pokretanje .NET aplikacija) i putem NuGet paketa
- Omogućava kreiranje web aplikacija i Web API-ja
- Lako se integriše sa različitim klijentskim tehnologijama
- Podržava MVC (Model-View-Controller) obrazac

# MVC pattern

## MVC patern uključuje sledeće komponente:



### Models

Modeli su klase koje predstavljaju podatke i poslovnu logiku aplikacije.

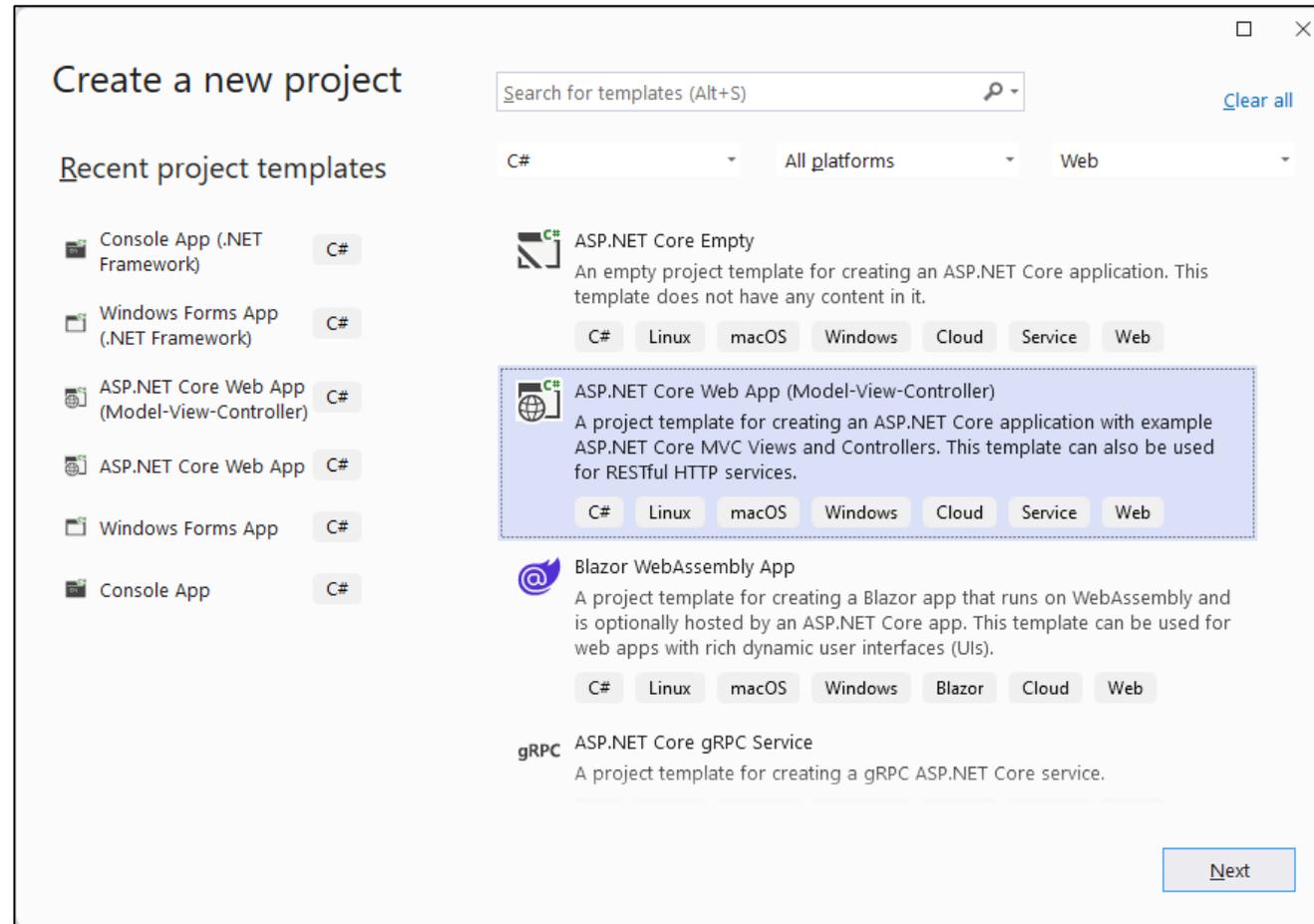
### Views

Pogledi su komponente generišu korisnički interfejs i prikazuju podatke korisniku.

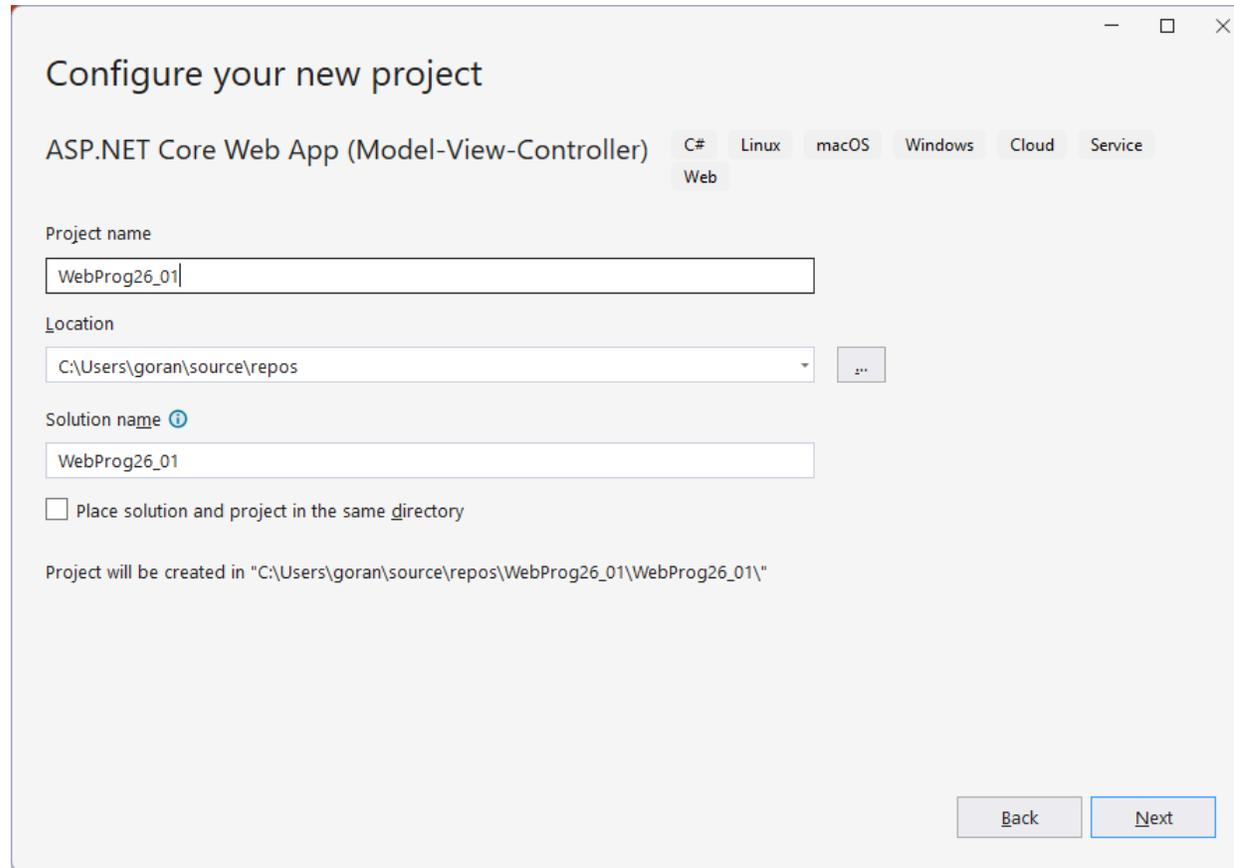
### Controllers

Kontroleri su komponente koje upravljaju korisničkim zahtevima, rade sa modelom i odabiraju pogled kojem prosleđuju podatke.

# Kreiranje ASP.NET Core web aplikacije -1



# Kreiranje ASP.NET Core web aplikacije -2



The image shows a 'Configure your new project' dialog box in Visual Studio. The title bar includes standard window controls (minimize, maximize, close). The main title is 'Configure your new project'. Below it, the project type is 'ASP.NET Core Web App (Model-View-Controller)'. There are several tabs for platform selection: 'C#' (selected), 'Linux', 'macOS', 'Windows', 'Cloud', and 'Service'. Under 'C#', there is a sub-tab 'Web'. The 'Project name' field contains 'WebProg26\_01'. The 'Location' field is a dropdown menu showing 'C:\Users\goran\source\repos' with a '...' button to the right. The 'Solution name' field, which has a help icon, also contains 'WebProg26\_01'. There is an unchecked checkbox labeled 'Place solution and project in the same directory'. At the bottom, it states 'Project will be created in "C:\Users\goran\source\repos\WebProg26\_01\WebProg26\_01\"'. At the bottom right, there are 'Back' and 'Next' buttons.

Configure your new project

ASP.NET Core Web App (Model-View-Controller) C# Linux macOS Windows Cloud Service  
Web

Project name  
WebProg26\_01

Location  
C:\Users\goran\source\repos

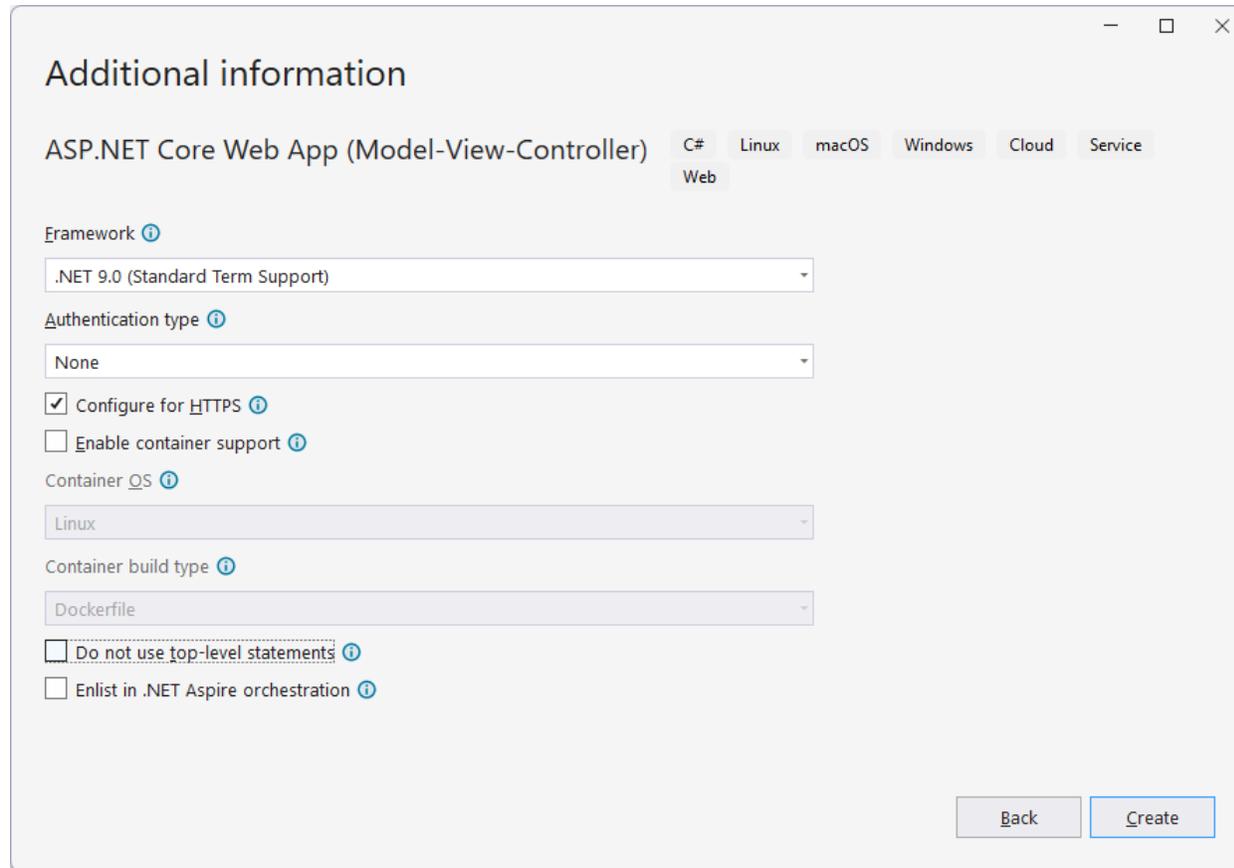
Solution name ⓘ  
WebProg26\_01

Place solution and project in the same directory

Project will be created in "C:\Users\goran\source\repos\WebProg26\_01\WebProg26\_01"

Back Next

# Kreiranje ASP.NET Core web aplikacije -3

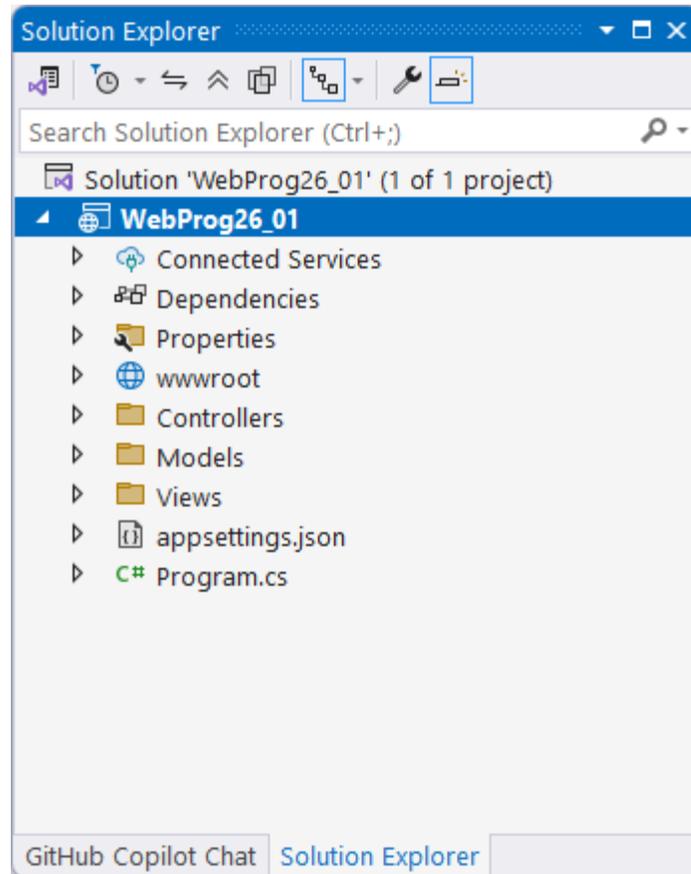


The screenshot shows a dialog box titled "Additional information" for creating an ASP.NET Core Web App. The dialog is styled with a light gray background and rounded corners. At the top, the title "Additional information" is followed by a standard window control bar (minimize, maximize, close). Below the title, the project type is identified as "ASP.NET Core Web App (Model-View-Controller)". A series of tabs are visible: "C#", "Linux", "macOS", "Windows", "Cloud", "Service", and "Web", with "Web" being the active tab. The main content area contains several configuration options:

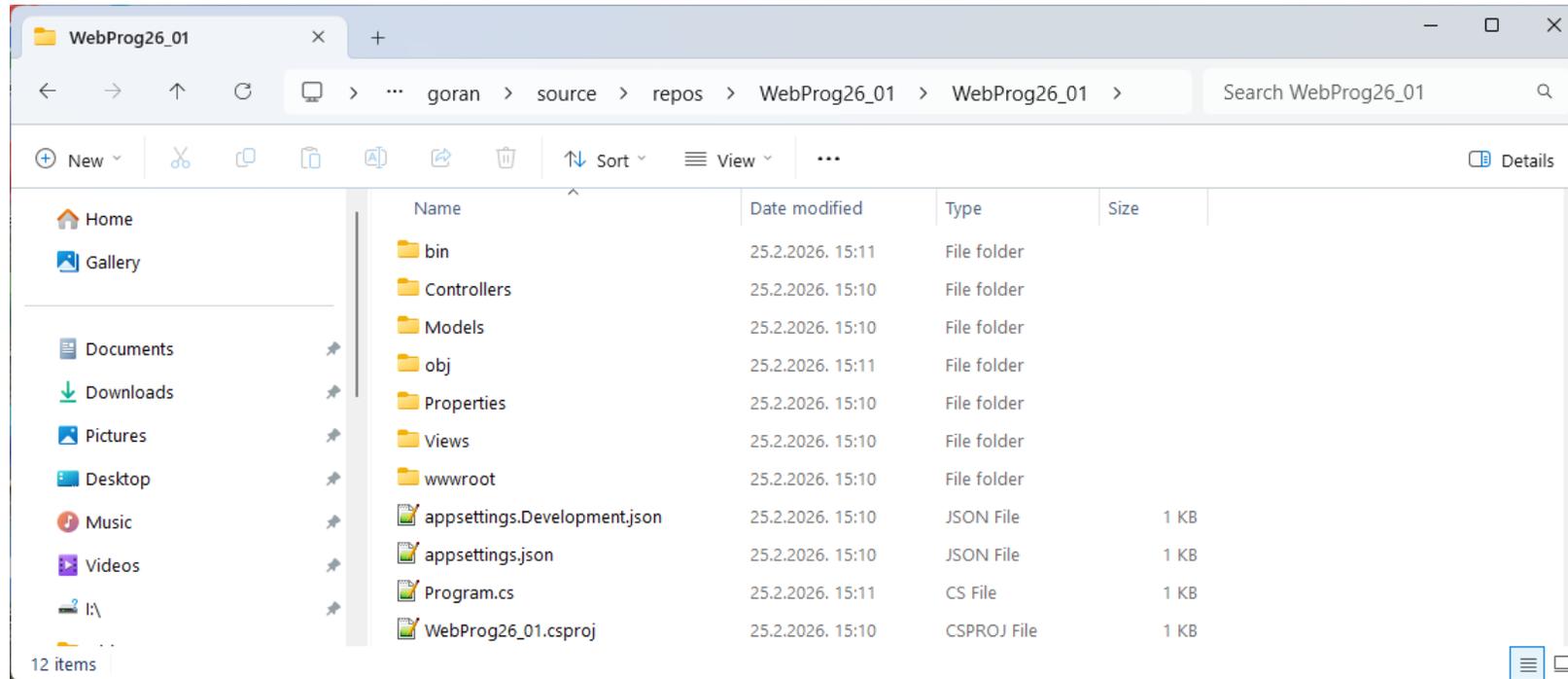
- Framework:** A dropdown menu currently set to ".NET 9.0 (Standard Term Support)".
- Authentication type:** A dropdown menu currently set to "None".
- Configuration options:** Two checkboxes are present: "Configure for HTTPS" (checked) and "Enable container support" (unchecked).
- Container OS:** A dropdown menu currently set to "Linux".
- Container build type:** A dropdown menu currently set to "Dockerfile".
- Advanced options:** Two checkboxes at the bottom: "Do not use top-level statements" (unchecked) and "Enlist in .NET Aspire orchestration" (unchecked).

At the bottom right of the dialog, there are two buttons: "Back" and "Create". The "Create" button is highlighted with a blue border, indicating it is the primary action.

# Struktura ASP.NET Core web aplikacije



# Folder aplikacije



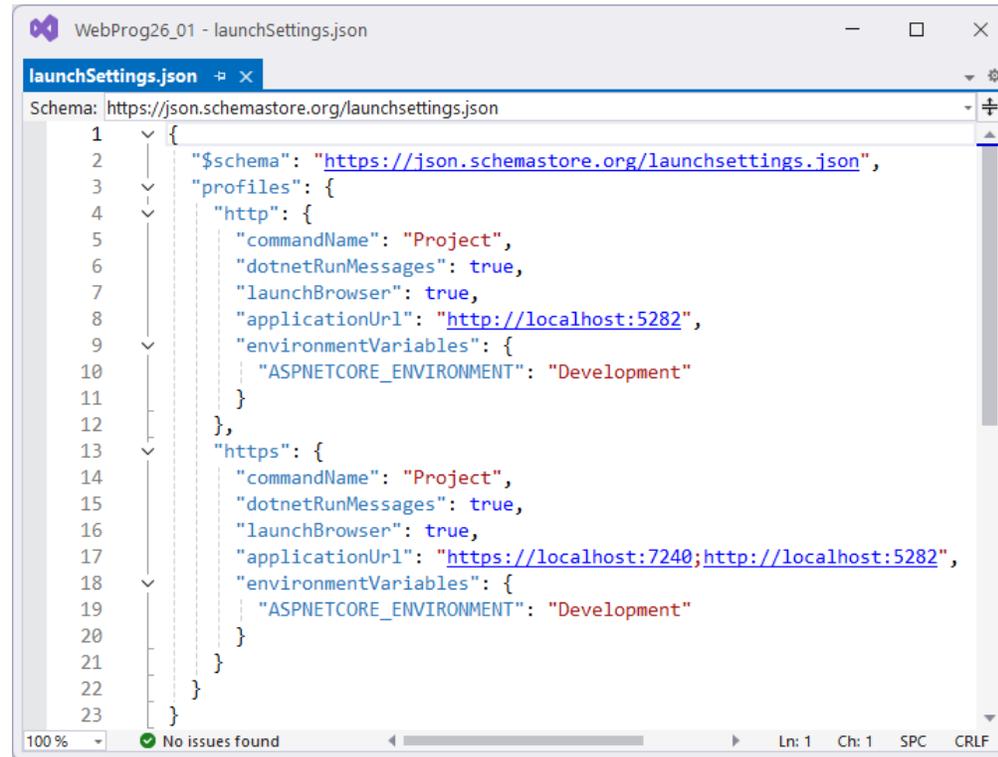
# Fajl .csproject

Desni klik na projekat → opcija „Edit Project File“

```
<Project Sdk="Microsoft.NET.Sdk.Web">  
  
  <PropertyGroup>  
    <TargetFramework>net9.0</TargetFramework>  
    <Nullable>enable</Nullable>  
    <ImplicitUsings>enable</ImplicitUsings>  
  </PropertyGroup>  
  
</Project>
```

.csproj fajl je konfiguracioni fajl koji definiše podešavanja projekta, verziju .NET platforme i zavisnosti (pakete) koje projekat koristi.

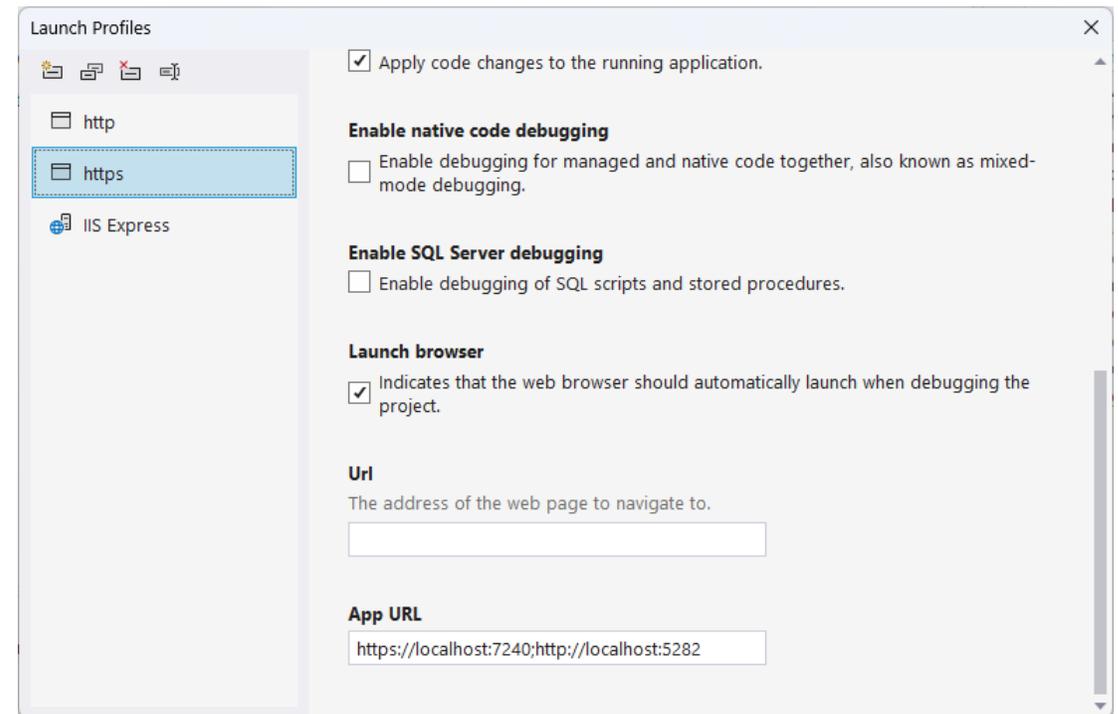
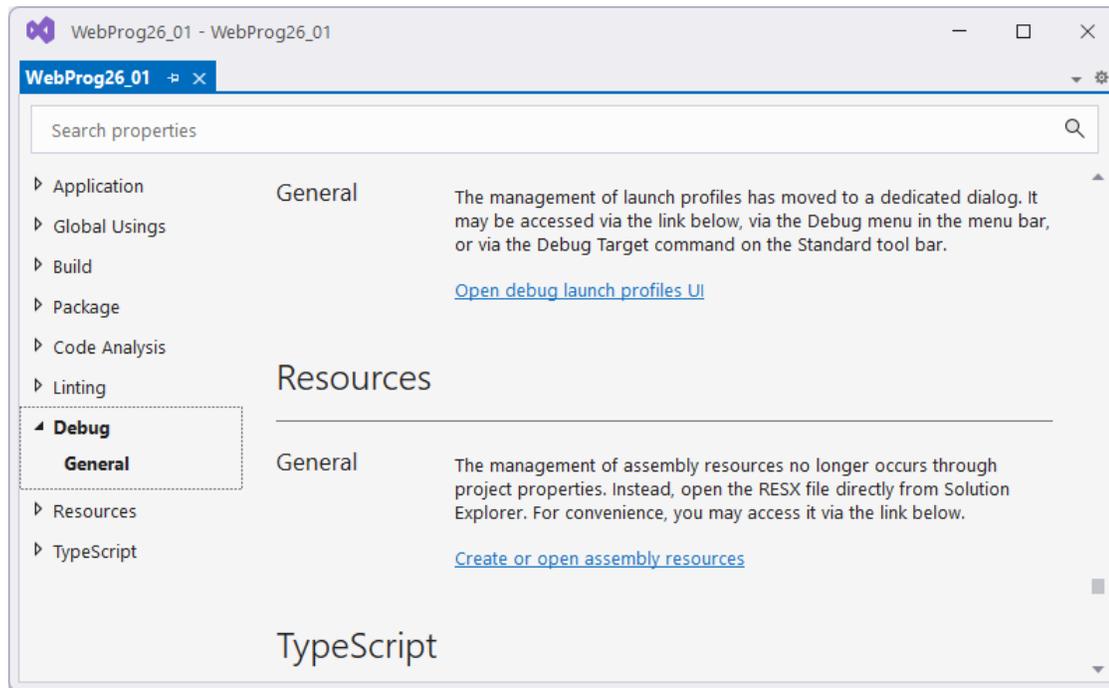
# Folder properties fajl launchSettings.json



```
1 {
2   "$schema": "https://json.schemastore.org/launchsettings.json",
3   "profiles": {
4     "http": {
5       "commandName": "Project",
6       "dotnetRunMessages": true,
7       "launchBrowser": true,
8       "applicationUrl": "http://localhost:5282",
9       "environmentVariables": {
10        "ASPNETCORE_ENVIRONMENT": "Development"
11      }
12    },
13    "https": {
14      "commandName": "Project",
15      "dotnetRunMessages": true,
16      "launchBrowser": true,
17      "applicationUrl": "https://localhost:7240;http://localhost:5282",
18      "environmentVariables": {
19        "ASPNETCORE_ENVIRONMENT": "Development"
20      }
21    }
22  }
23 }
```

launchSettings.json je konfiguracioni fajl koji definiše kako se ASP.NET Core aplikacija pokreće u razvojnom okruženju.

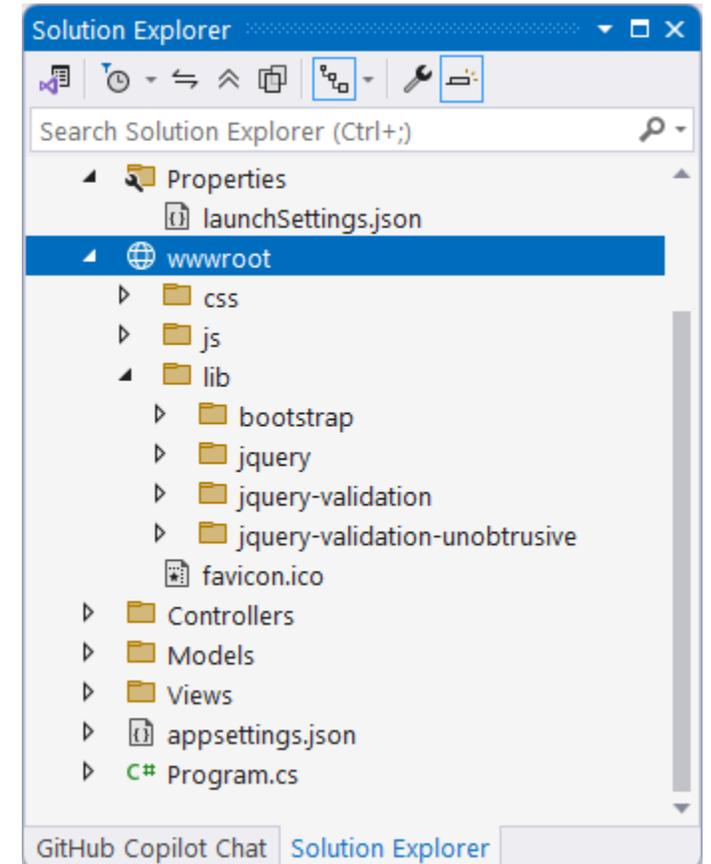
# Prikaz profila za pokretanje aplikacije



- Profilima za pokretanje se pristupa preko opcije "Properties" → "Debug" (Launch Profiles)
- Profil za pokretanje aplikacije definiše način na koji se aplikacija pokreće tokom razvoja (protokol, port, okruženje i dodatna podešavanja)

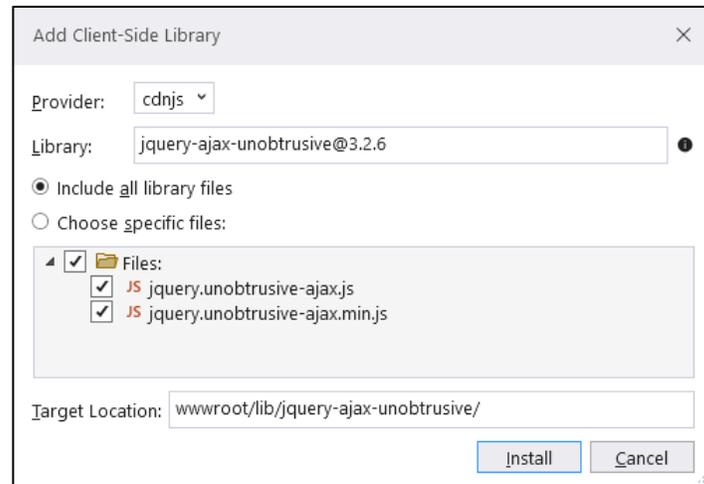
# Folder wwwroot

- U folderu wwwroot čuvaju se statički fajlovi (CSS, JavaScript, slike itd.)
- Svi fajlovi u folderu wwwroot dostupni su klijentu putem HTTP zahteva
- Novi projekat sadrži biblioteke jQuery i Bootstrap

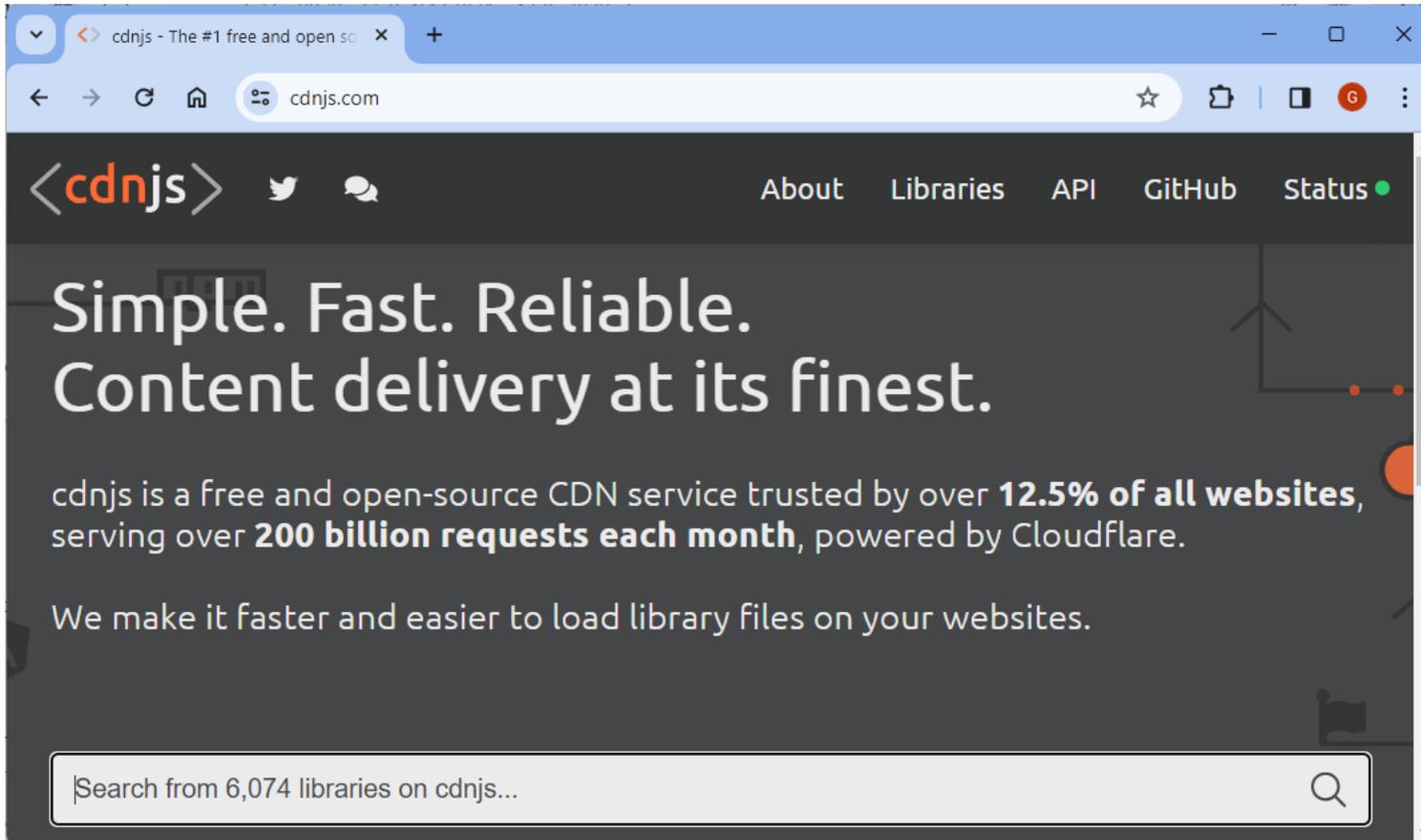


# Dodavanje klijentskih biblioteka

- Desni klik na folder **lib** → opcija "Add Client-Side Library" ili
- U meniju **Project** → opcija "Add Client-Side Library"



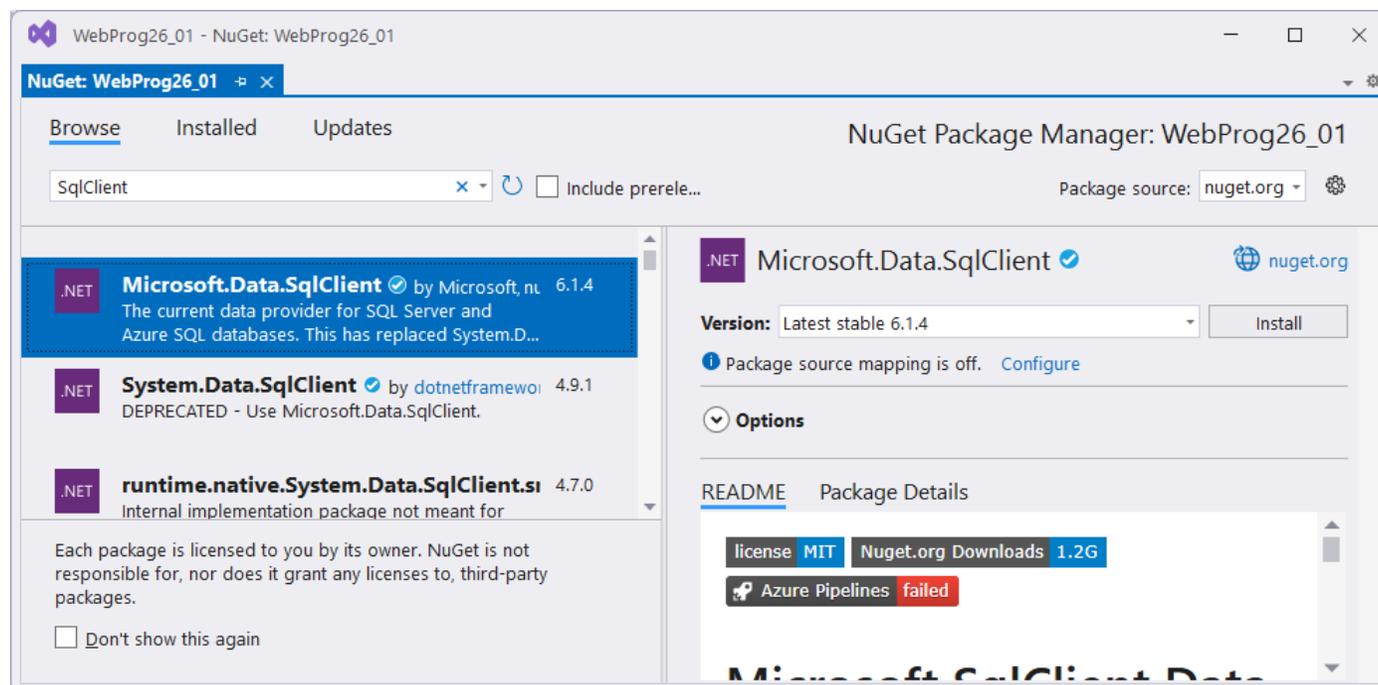
# cdnjs.com



cdnjs je Content Delivery Network (CDN) koji hostuje popularne klijentske biblioteke i omogućava njihovo učitavanje putem linka bez lokalne instalacije.

# Dodavanje serverskih biblioteka

U meniju **Project** → **Manage NuGet Packages**



Serverske biblioteke se dodaju putem NuGet Package Manager-a

# Fajl .csproj

```
<Project Sdk="Microsoft.NET.Sdk.Web">

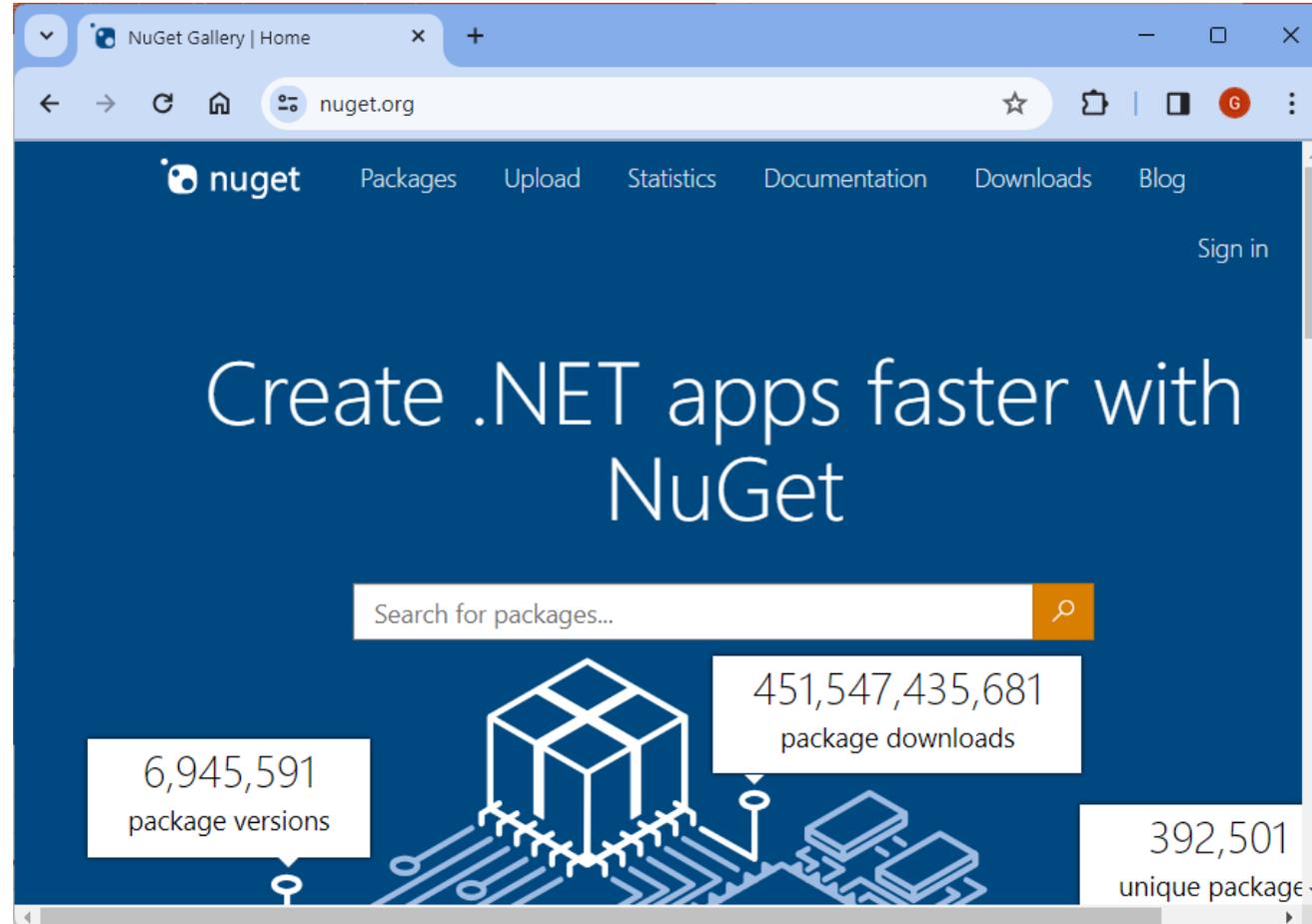
  <PropertyGroup>
    <TargetFramework>net9.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.Data.SqlClient" Version="6.1.4" />
  </ItemGroup>

</Project>
```

U .csproj fajlu je vidljivo da je instalirana serverska biblioteka Microsoft.Data.SqlClient putem NuGet-a.

# www.nuget.org



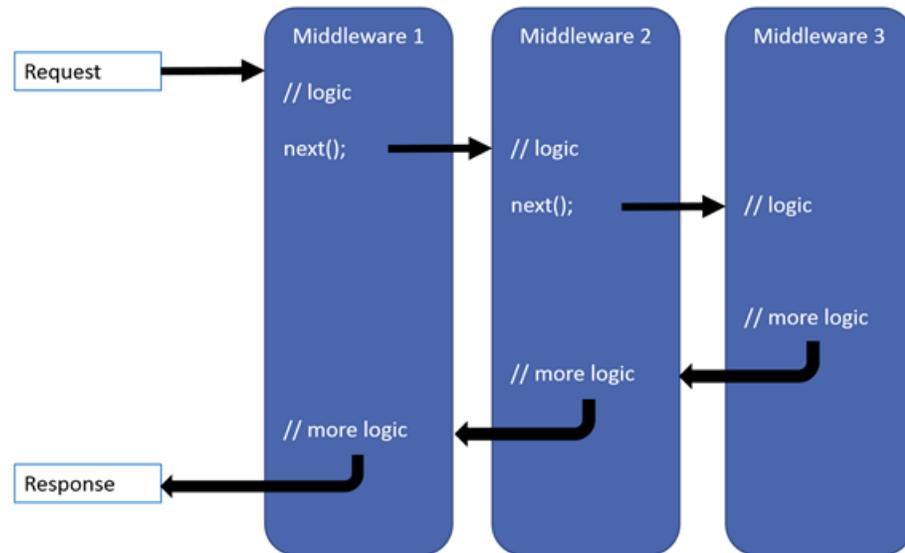
# Dependency Injection container

- Servis je klasa koja pruža određenu funkcionalnost i koristi se u drugim klasama
- Dependency Injection (DI) kontejner je komponenta koja upravlja kreiranjem i ubacivanjem zavisnosti u aplikaciji
- ASP.NET Core ima ugrađen Dependency Injection kontejner
- Servis se mora registrovati u DI kontejneru pre korišćenja
- DI kontejner automatski kreira instancu servisa i ubacuje je u konstruktor zavisne klase

# Middleware

- ASP.NET Core koristi middleware za obradu HTTP zahteva
- Middleware pipeline je cevovod middleware komponenti kroz koji prolazi HTTP zahtev i odgovor
- Svaka middleware komponenta može obraditi zahtev i proslediti ga sledećoj komponenti
- Middleware se konfigurise u fajlu **Program.cs**
- Redosled dodavanja middleware komponenti određuje tok obrade zahteva i odgovora
- Redosled je važan za sigurnost, performanse i funkcionalnost aplikacije

# Cevovod(pipeline) obrade zahteva



# Cevovod(pipeline) obrade zahteva

- ASP.NET Core koristi cevovod (pipeline) za obradu HTTP zahteva
- HTTP zahtev ulazi u cevovod middleware komponenti
- Svaka middleware komponenta može obraditi zahtev i proslediti ga sledećoj komponenti
- Nakon obrade, odgovor se vraća nazad kroz isti cevovod obrnutim redosledom
- Ako middleware ne prosledi zahtev dalje (ne pozove next()), prekida se dalja obrada (short-circuiting)
- Redosled middleware komponenti određuje tok obrade zahteva i odgovora

# Fajl Program.cs

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseRouting();

app.UseAuthorization();

app.MapStaticAssets();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}")
    .WithStaticAssets();
app.Run();
```

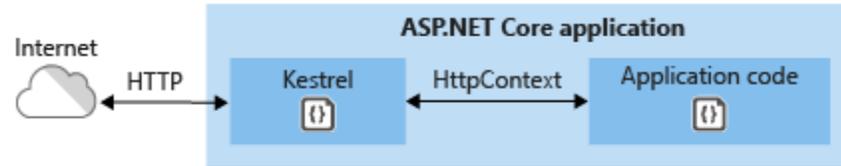
# Fajl Program.cs

- Prvo se kreira instanca klase **WebApplicationBuilder** korišćenjem statičke metode **CreateBuilder** klase **WebApplication**
- Zatim se u kolekciju servisa aplikacije dodaju MVC usluge korišćenjem metode **AddControllersWithViews**
- Metoda **Build()** objekta **WebApplicationBuilder** kreira instancu **WebApplication**
- Nakon toga se konfigurirše middleware cevovod obrade HTTP zahteva
- Zatim se konfiguriršu endpointi za rutiranje
- Pozivom metode **Run** startuje se web aplikacija

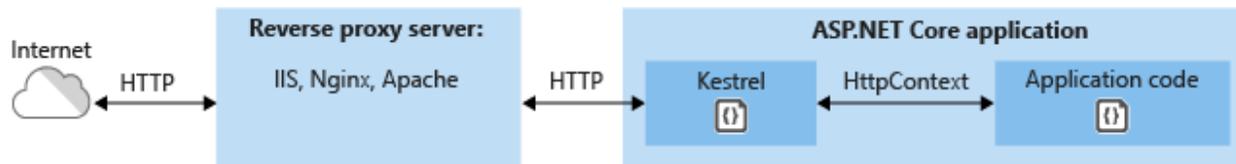
# Novi hosting model

- ASP.NET Core (od .NET 6) koristi minimalni hosting model
- U starom modelu konfiguracija je bila podeljena između fajlova Startup.cs i Program.cs
- Minimalni hosting model objedinjuje konfiguraciju servisa i middleware-a u jednom fajlu (Program.cs)
- Podrazumevano se koristi ugrađeni **Kestrel** web server

# Kestrel web server

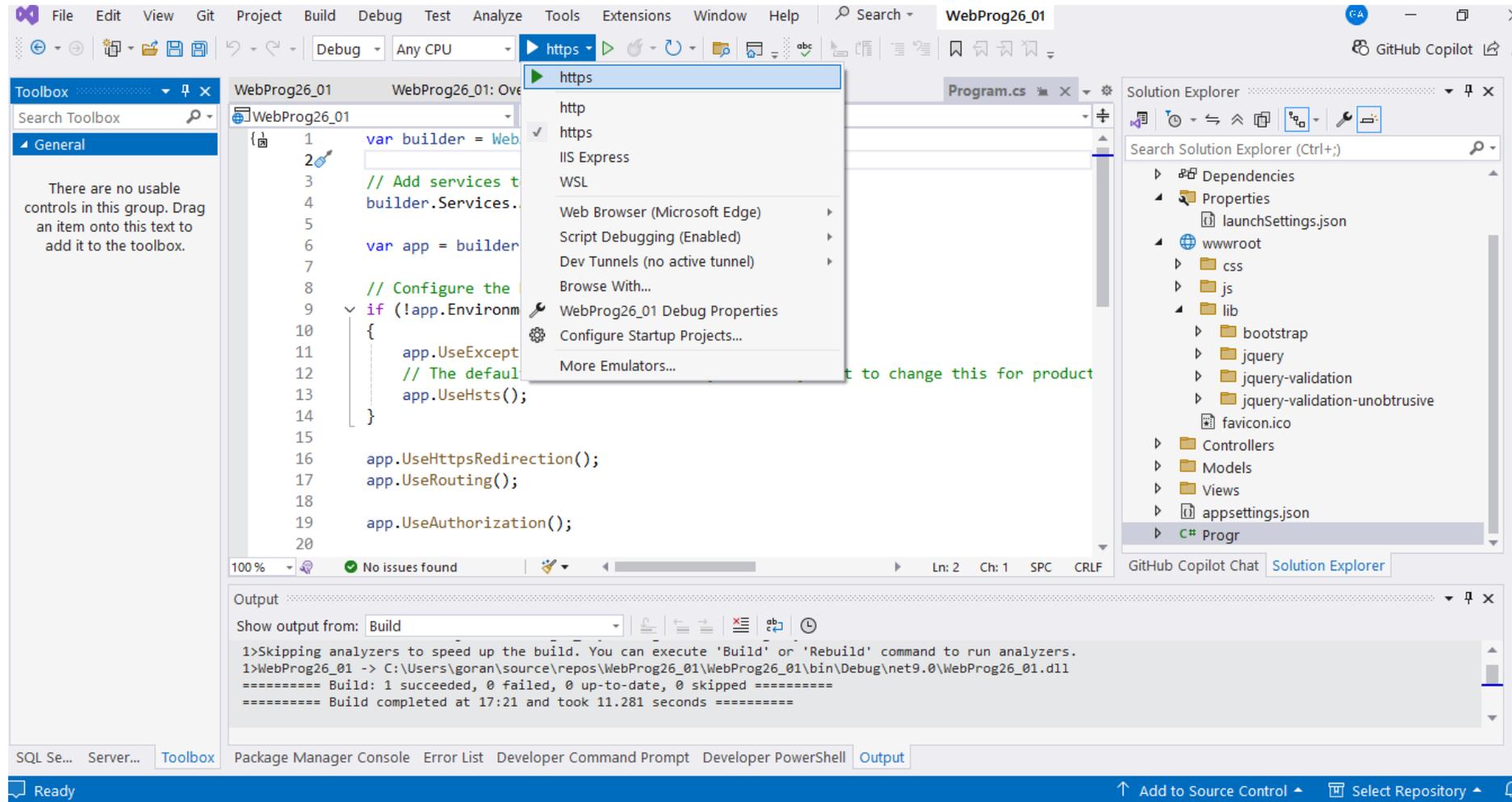


- Više aplikacija ne može istovremeno koristiti istu IP adresu i port
- Zbog toga se za hostovanje više aplikacija koristi reverse proxy server



- **Kestrel** se često koristi iza reverse proxy servera (IIS, Nginx, Apache)
- **Reverse proxy** prima HTTP zahteve sa interneta i prosleđuje ih Kestrelu
- Reverse proxy omogućava hostovanje više aplikacija na istoj IP adresi i portu
- Reverse proxy omogućava da više aplikacija bude dostupno preko iste IP adrese i porta tako što zahteve prosleđuje različitim internim portovima

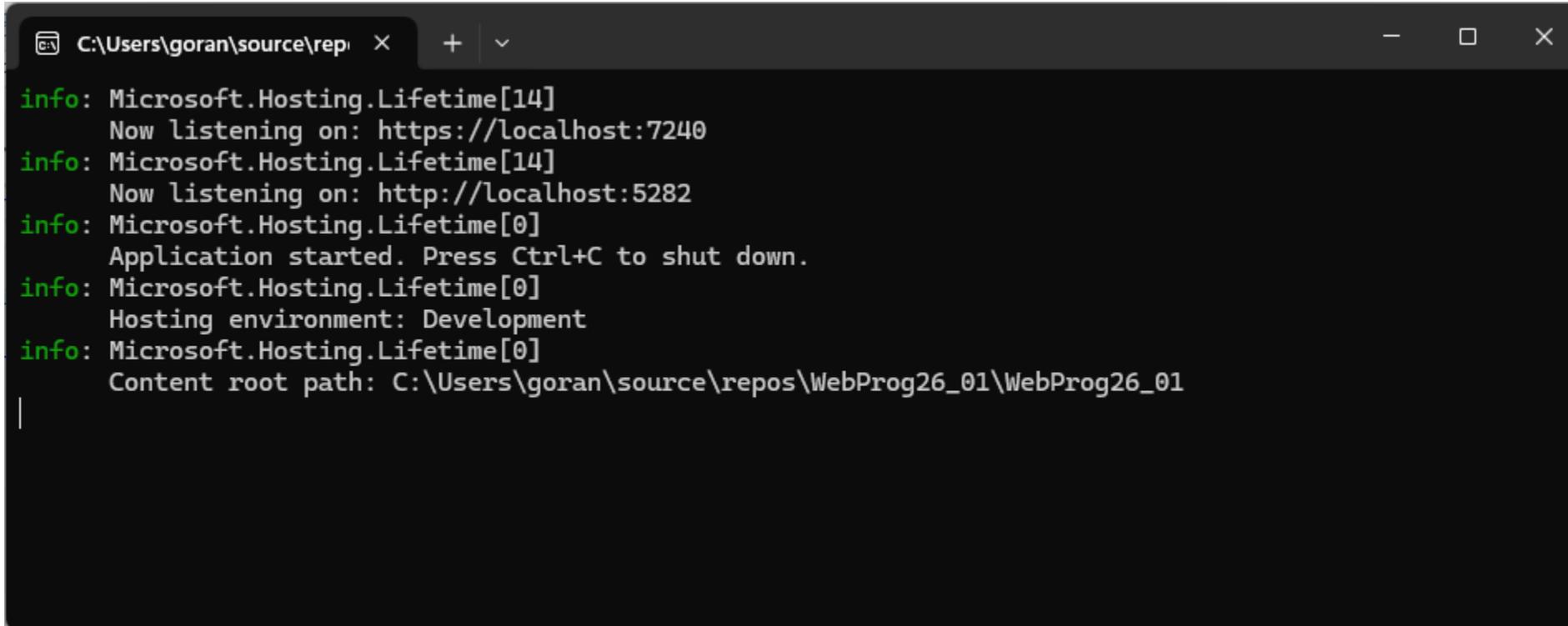
# Prvo pokretanje ASP.NET Core web aplikacije



# Prvo pokretanje ASP.NET Core web aplikacije

- Aplikacija se pokreće izborom launch profila (npr. http ili IIS Express)
- Klikom na Run (▶) Visual Studio pokreće aplikaciju
- Pokreće se web server (podrazumevano Kestrel)
- Aplikacija počinje da osluškuje HTTP zahteve na definisanom portu
- Automatski se otvara web pregledač sa URL adresom aplikacije
- Project → (http / https)
  - Pokreće se direktno Kestrel
- Project → IIS Express pokreće se:
  - IIS Express
  - Kestrel
  - IIS Express prosleđuje zahteve Kestrelu

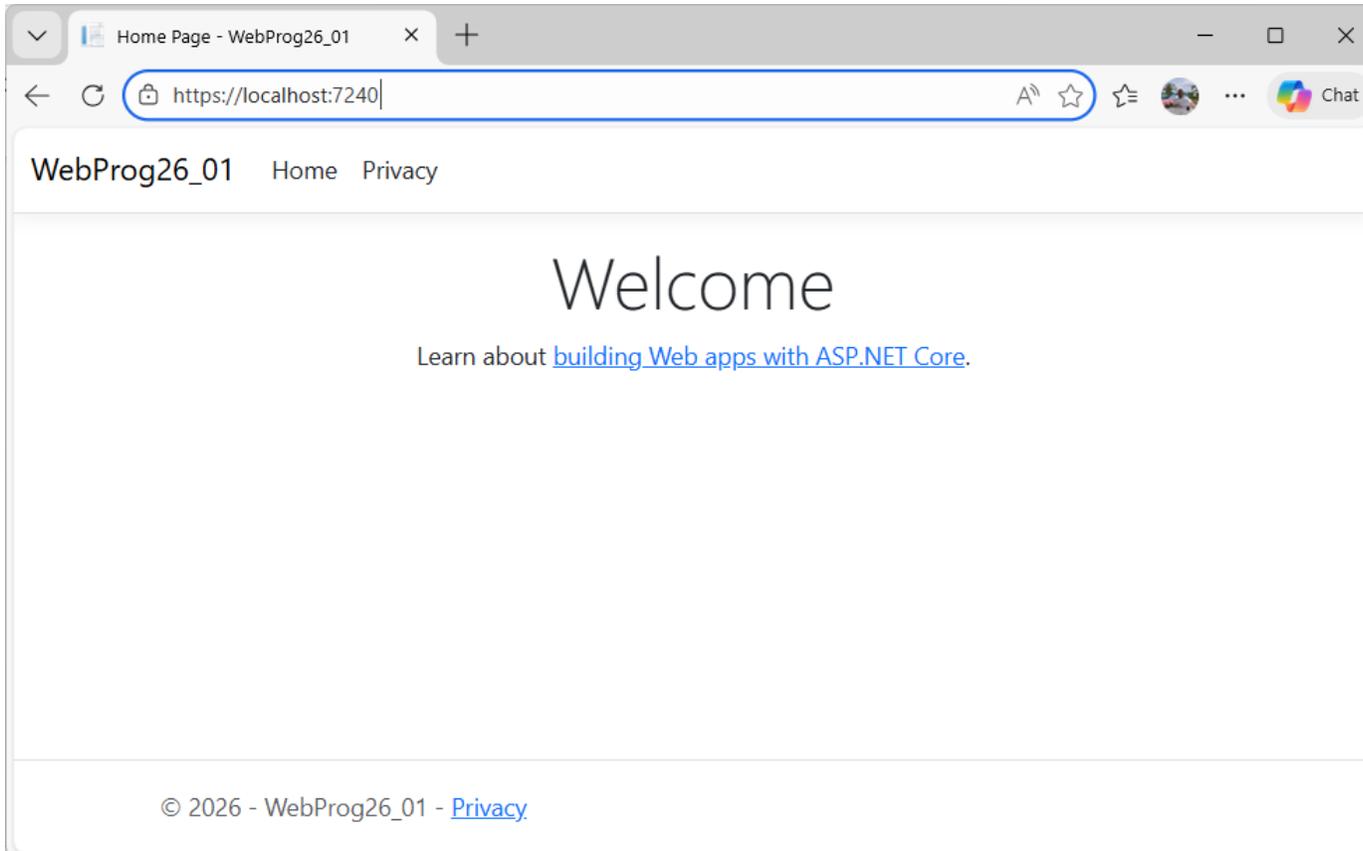
# Kestrel server – log pri pokretanju aplikacije

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Users\goran\source\rep' and standard window controls. The terminal output consists of several lines of log messages from Microsoft.Hosting.Lifetime, indicating that the application is listening on https://localhost:7240 and http://localhost:5282, and that it has started in the Development environment with a content root path of C:\Users\goran\source\repos\WebProg26\_01\WebProg26\_01.

```
C:\Users\goran\source\rep x + v
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7240
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5282
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\goran\source\repos\WebProg26_01\WebProg26_01
|
```

- Kestrel server je pokrenut
- Aplikacija sluša na definisanim portovima
- Prikazan je režim rada (Development)

# Prikaz aplikacije u browseru

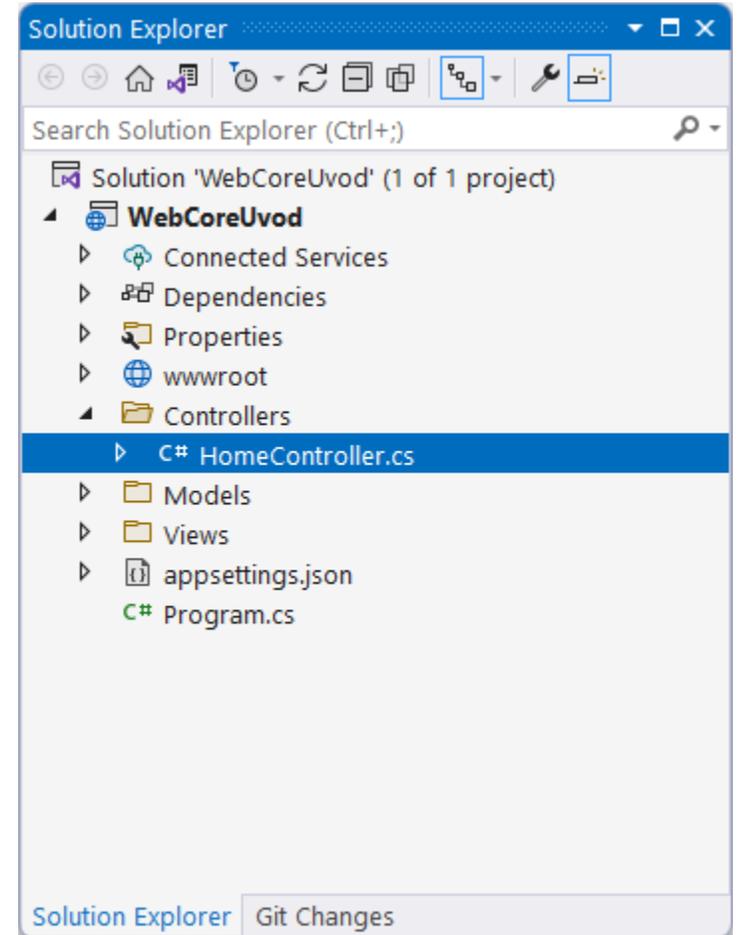


- Aplikacija je uspešno pokrenuta na adresi [https://localhost:7240](https://localhost:7240/)
- Zahtev iz browsera obrađuje Kestrel server

ViewData i ViewBag

# Klasa HomeController.cs

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



# Index pogled klase HomeController

```
@{  
    ViewData["Title"] = "Home Page";  
}  
  
<div class="text-center">  
    <h1 class="display-4">Welcome</h1>  
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps with ASP.NET  
Core</a>.</p>  
</div>
```

# ViewData

- **ViewData** je mehanizam u ASP.NET Core za prenos podataka iz kontrolera u pogled bez korišćenja posebnog modela
- ViewData je svojstvo kontrolera tipa **ViewDataDictionary**
- Koristi se za jednosmernu komunikaciju (kontroler → pogled)
- Namijenjen je prenosu jednostavnih podataka potrebnih samo za prikaz
- Implementiran je kao rečnik tipa **Dictionary<string, object>**
- Ključevi su stringovi
- Pri čitanju podataka u pogledu potrebno je kastovanje ako vrednost nije string

# Prosleđivanje podataka Index pogledu HomeController klase

```
public IActionResult Index()
{
    ViewData["brojPonavljanja"] = 5;
    ViewData["poruka"] = "Uvod u ASP.NET Core web aplikacije";

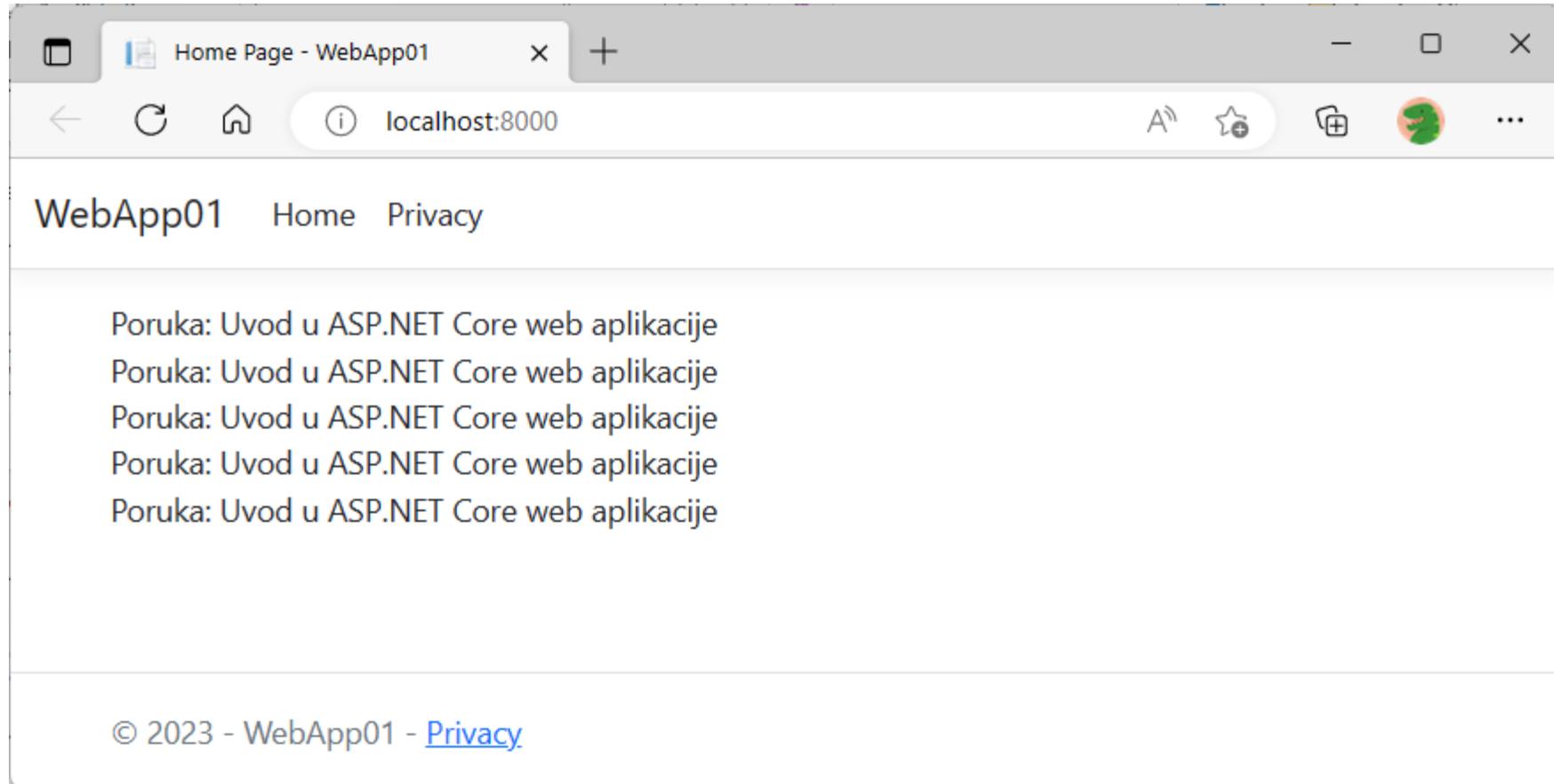
    return View();
}
```

Ako se u akciji pozove samo `return View();` bez navođenja imena pogleda, ASP.NET Core automatski vraća pogled koji ima isto ime kao akcija (u ovom slučaju `Index`).

# Pogled Index.cshtml

```
@{  
    ViewData["Title"] = "Home Page";  
}  
  
<div class="row">  
    <div class="col-6">  
        @for (int i = 0; i < (int)ViewData["brojPonavljanja"]; i++)  
        {  
            <span>Poruka: @ViewData["poruka"]</span><br />  
        }  
    </div>  
  
</div>
```

# Generisani html



# ViewBag

- **ViewBag** omogućava prenos podataka iz kontrolera u pogled bez korišćenja modela
- Koristi se za jednosmernu komunikaciju (kontroler → pogled)
- ViewBag je dinamički objekat (dynamic)
- Svojstva se definišu dinamički u toku izvršavanja
- Ne zahteva eksplicitno kastovanje pri čitanju vrednosti
- Interno koristi ViewData (predstavlja dinamički omotač oko ViewDataDictionary)

# Definisanje dinamičkih svojstava

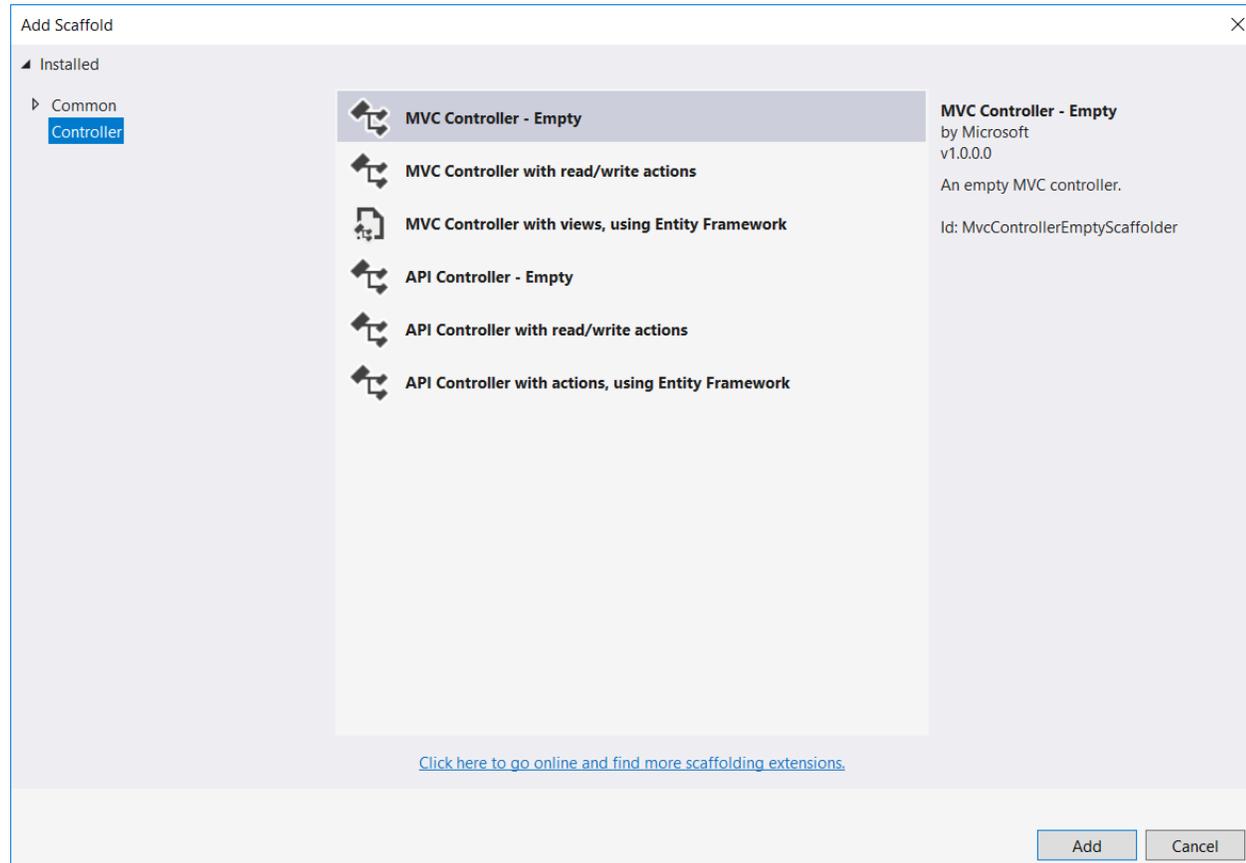
```
public IActionResult Index()
{
    ViewBag.brojPonavljanja = 5;
    ViewBag.poruka = "Uvod u ASP.NET Core web aplikacije";
    return View();
}
```

# Pogled Index.cshtml

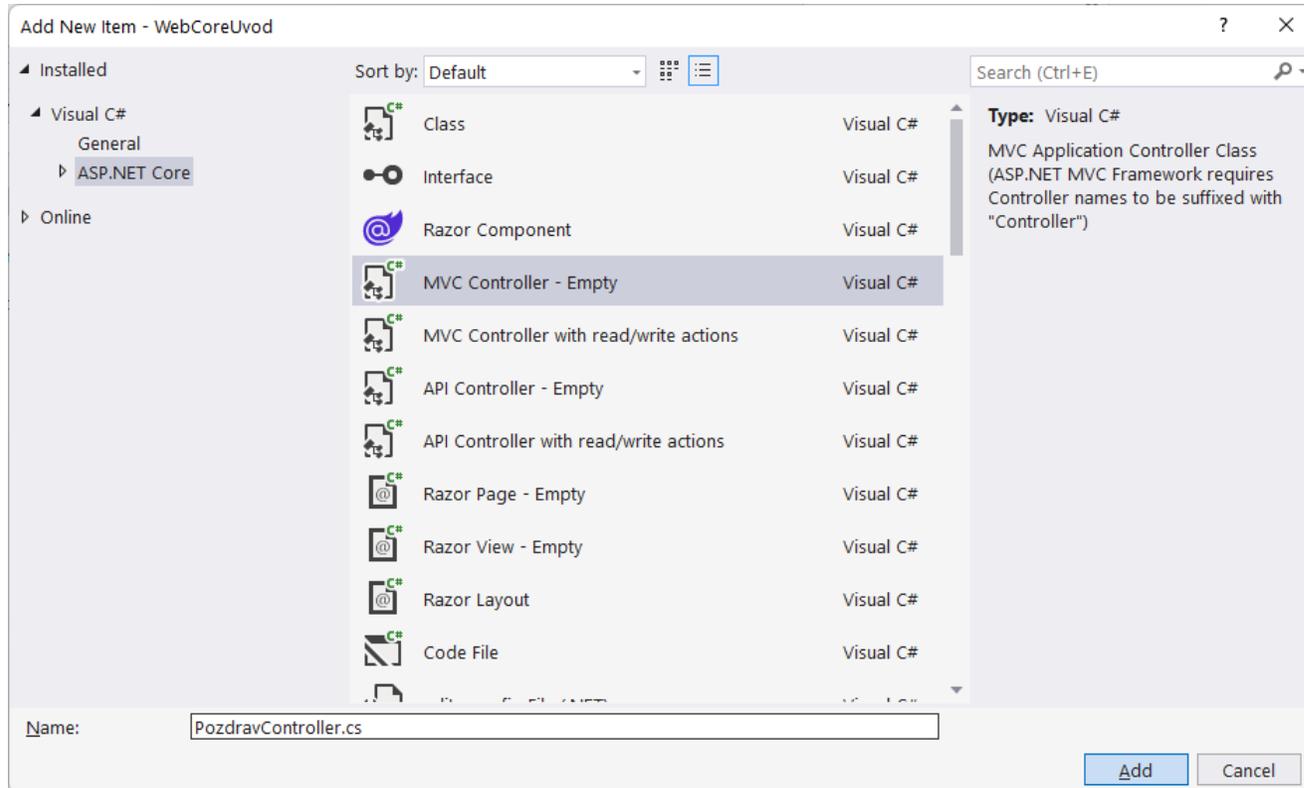
```
<div class="row">
  <div class="col-md-6">
    @for (int i = 0; i < ViewBag.brojPonavljjanja; i++)
    {
      <span>Poruka: @ViewBag.poruka</span><br />
    }
  </div>
</div>
```

Prosleđivanje podataka metodi  
kontrolera

# Kreiranje praznog MVC kontrolera



# Imenovanje kontrolera



# Klasa PozdravController

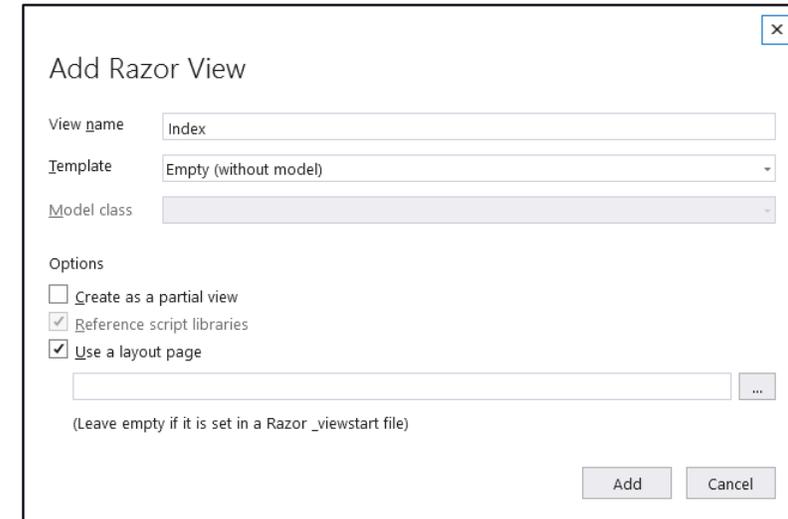
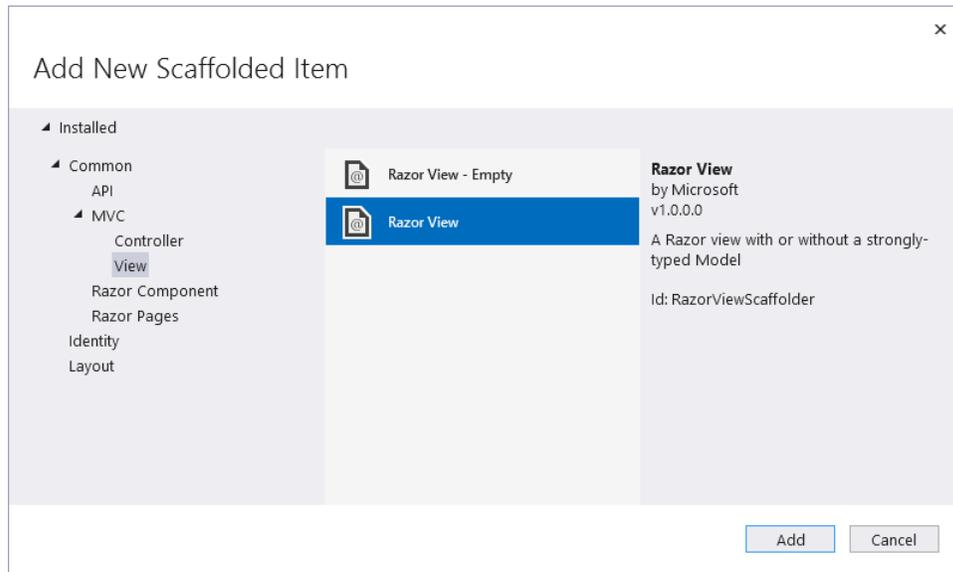
```
public class PozdravController : Controller
{
    public IActionResult Index()
    {
        return View();
    }
}
```

# Index metoda klase PozdravController

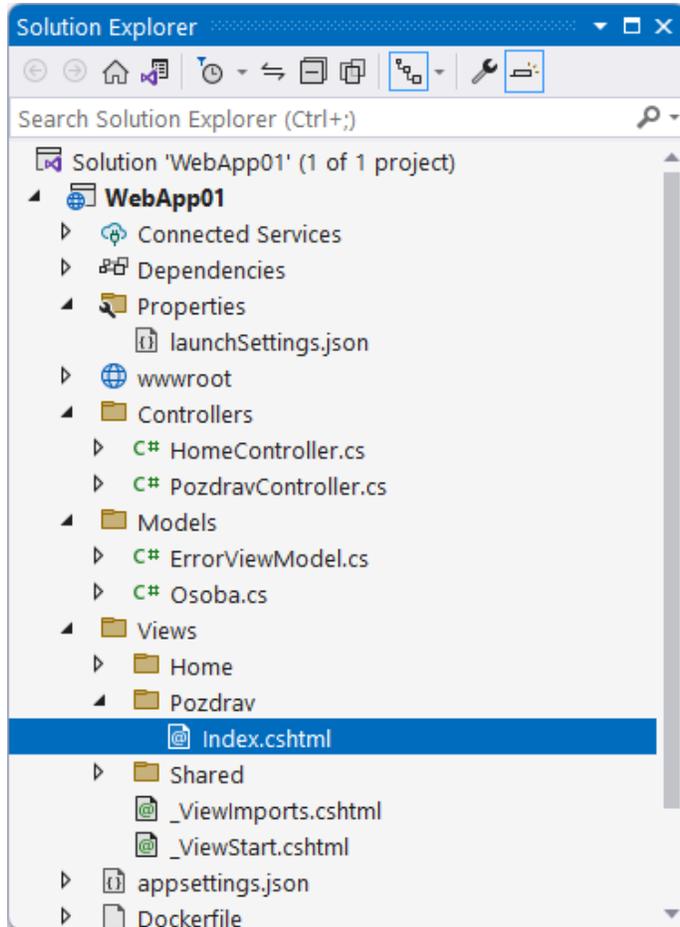
```
public IActionResult Index()
{
    ViewBag.poruka = "Zdravo";
    return View();
}
```

# Index pogled PozdravController klase

Desni klik unutar metode Index pa opcija Add-> View



# Folder Views/Pozdrav



# Index.cshtml pogled klase PozdravController

```
@{
    ViewData["Title"] = "Pozdrav";
}

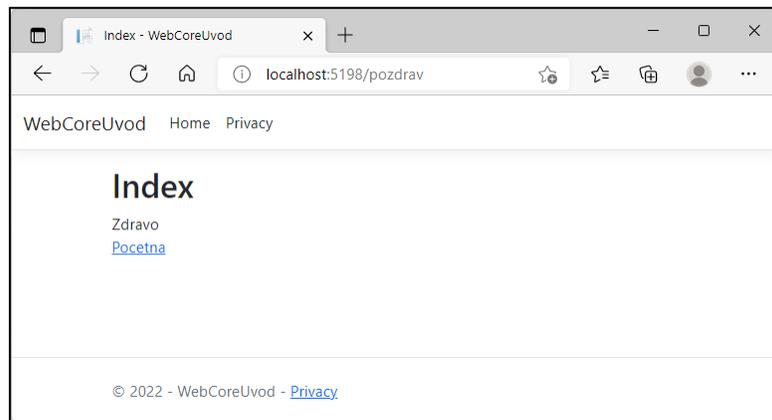
<h1>Pozdrav-Index</h1>

<div class="row">
    <div class="col-6">
        @ViewBag.Poruka
        <br />
        <a href="/Home/Index">Pocetna</a>
    </div>
</div>
```

# Poziv Index akcione metode klase PozdravController posredstvom url adrese

<http://localhost:5024/pozdrav/index>

<http://localhost:5024/pozdrav>



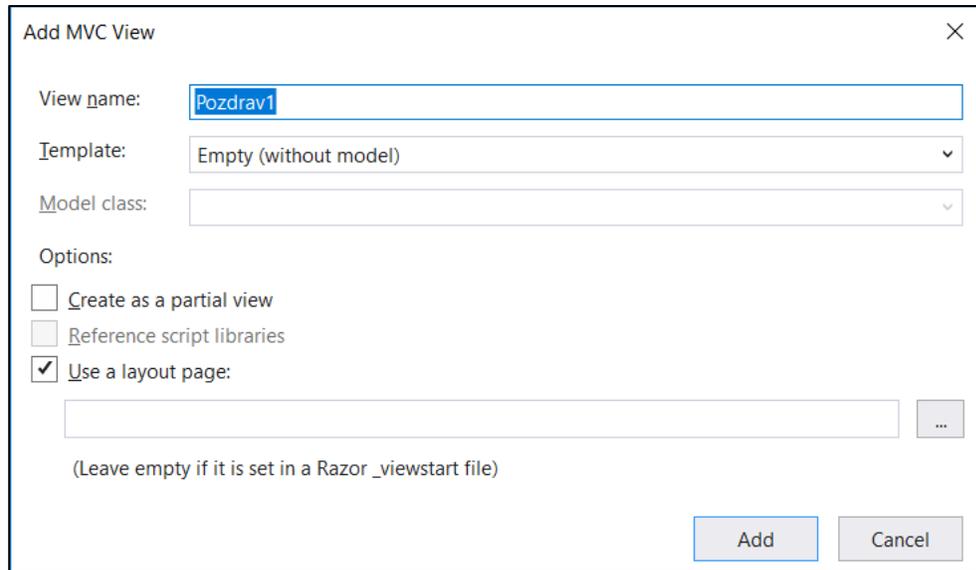
- Ruta je definisana kao {controller}/{action}
- Ako se akcija ne navede, podrazumevano se poziva Index
- Zato oba URL-a pozivaju istu akciju

# Akciona metoda Pozdrav1 PozdravController klase

```
public IActionResult Pozdrav1(int id = 1)
{
    ViewBag.BrojPonavljanja = id;
    ViewBag.Poruka = "Dobar dan";
    return View();
}
```

- Akciona metoda Pozdrav1 prima parametar id sa podrazumevanom vrednošću 1
- Ako se vrednost ne navede u URL-u, koristi se ta podrazumevana vrednost

# Kreiranje pogleda Pozdrav1



Add MVC View

View name:

Template:

Model class:

Options:

- Create as a partial view
- Reference script libraries
- Use a layout page:

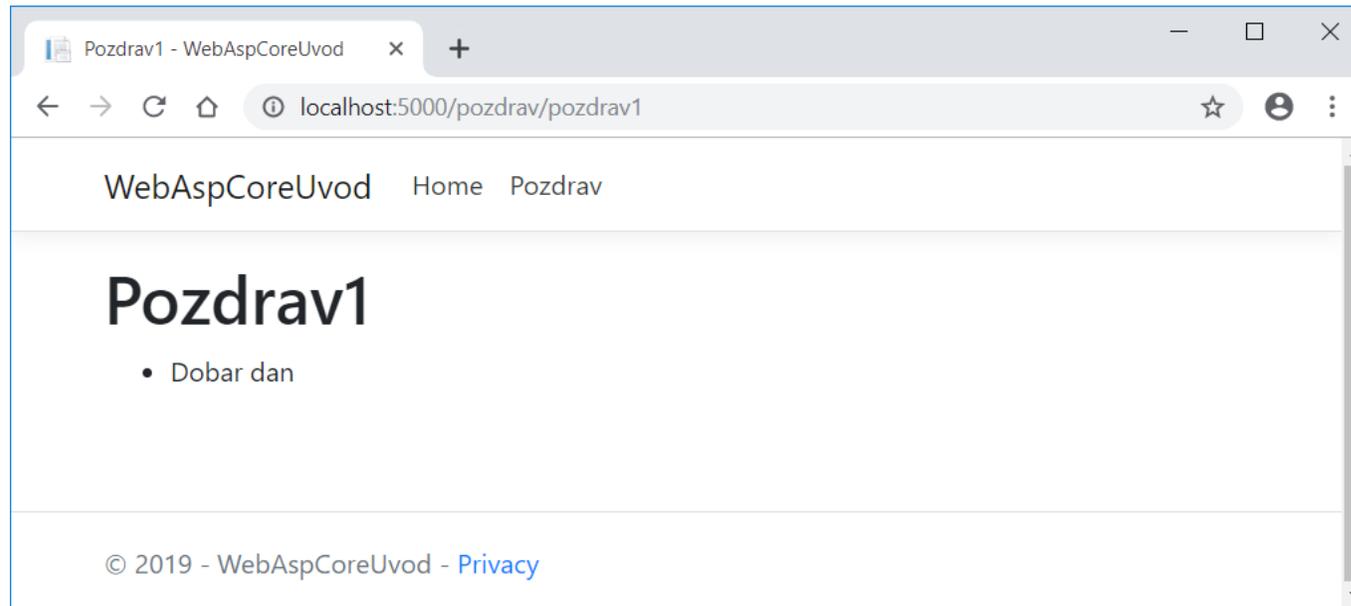
...

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

# Pogled Pozdrav1.cshtml

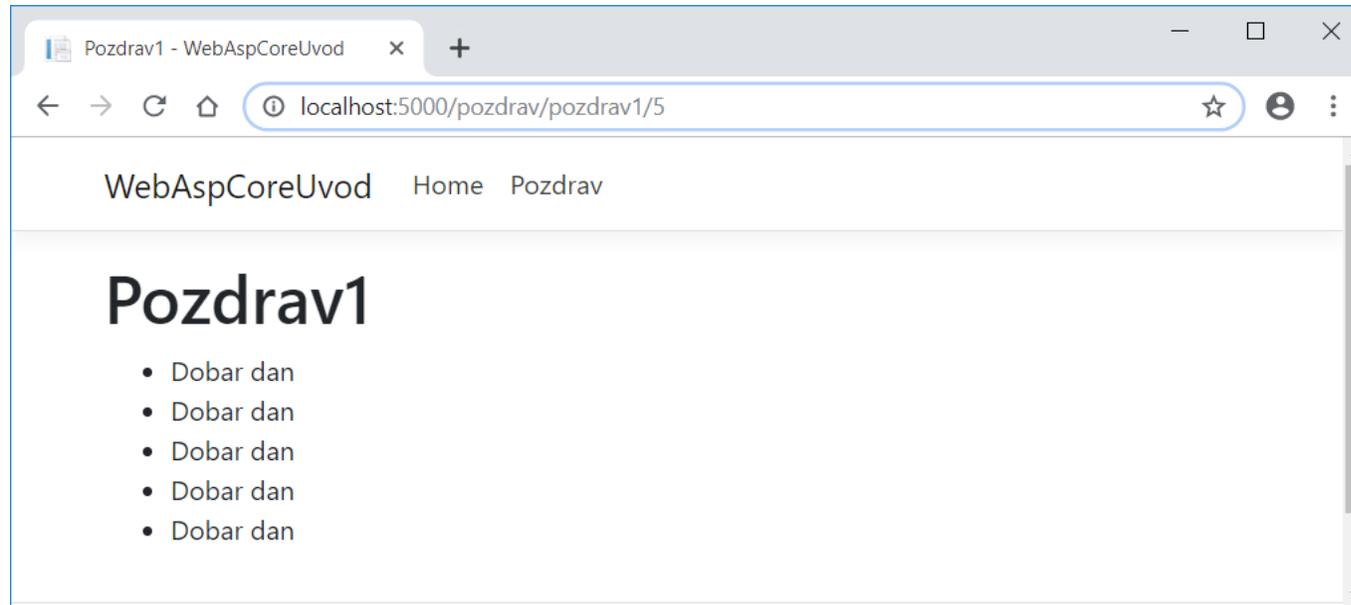
```
@{  
    ViewData["Title"] = "Pozdrav1";  
}  
  
<h1>Pozdrav1</h1>  
  
<div>  
    <ul>  
        @for (int i = 0; i < ViewBag.BrojPonavljanja; i++)  
        {  
            <li>@ViewBag.Poruka</li>  
        }  
    </ul>  
</div>
```

# Poziv Pozdrav1 akcione metode



# Prosledjivanje parametara akcionoj metodi

<http://localhost:5000/pozdrav/pozdrav1/5>



- U adresi/pozdrav/pozdrav1/5 vrednost 5 se mapira na parametar id u metodi Pozdrav1(int id)
- Mapiranje se vrši na osnovu rute:{controller}/{action}/{id?}

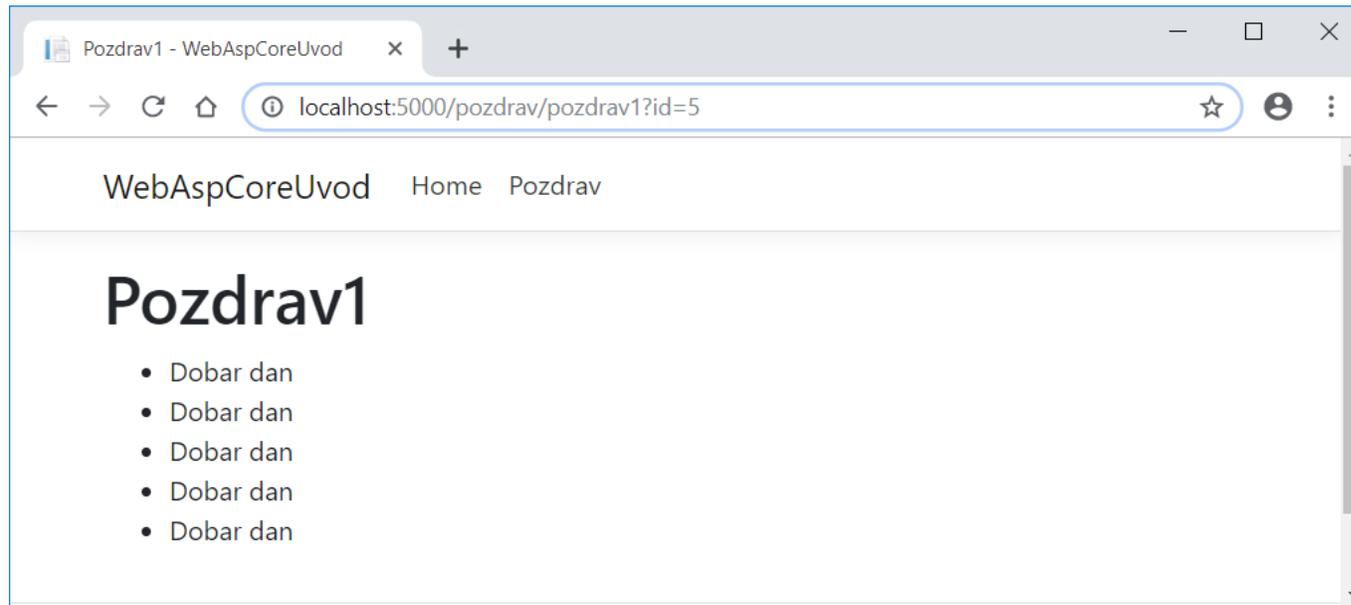
# Query string

- Query string je deo URL adrese koji dolazi nakon znaka ? i koristi se za prosleđivanje podataka serveru u obliku key=value parova
- Više parametara se razdvaja znakom &

```
/pozdrav/pozdrav1?id=5&ime=Marko
```

# Korišćenje query stringa

<http://localhost:5000/pozdrav/pozdrav1?id=5>



- Parametar se može proslediti i putem query stringa
- U adresi/pozdrav/pozdrav1?id=5 vrednost 5 se mapira na parametar id akcione metode
- ASP.NET Core automatski povezuje naziv parametra u URL-u (id) sa parametrom metode (int id)

# Pitanje 1

Servisi u terminologiji ASP.NET Core web aplikacija su:

- a. Klase ASP.NET Core web aplikacije koje zavise od drugih klasa
- b. Klase izvedene iz klase Controller
- c. Klase od kojih zavise druge klase ASP.NET Core web aplikacije

Odgovor: c

# Pitanje 2

Registracija servisa - dodavanje servisa u kolekciju servisa aplikacije kod ASP.NET Core 9 aplikacija, vrši se unutar

- a. Fajla Program.cs
- b. Fajla appsettings.json
- c. Fajla launchSettings.cs

Odgovor: a

# Pitanje 3

Cevovod (pipeline) za protočnu obradu HTTP zahteva u ASP.NET Core 9 web aplikacijama konfigurira se unutar:

- a. Fajla Program.cs
- b. Fajla appsettings.json
- c. Fajla launchsettings.cs

Odgovor: a

# Pitanje 4

ASP.NET Core web aplikacija ima ugrađen Dependency Injection kontejner koji:

- a. Ubacuje objekat klase servisa u zavisnu klasu
- b. Ubacuje pogled u ASP.NET Core web aplikaciju
- c. Ubacuje kontroler u ASP.NET Core web aplikaciju

Odgovor: a

# Pitanje 5

Statički fajlovi kod ASP.NET Core web aplikacija čuvaju se unutar sledećeg foldera:

- a. Models
- b. Data
- c. wwwroot

Odgovor: c