# Web programiranje

Dr. Goran Aritonović

goran.aritonovic@bpa.edu.rs

kabinet 211

# Softver

- Visual Studio Community 2022 (17.5)
- Visual Studio Code
- SQL Server 2019 (2017)  Express
- SQL Server Management Studio Express
- Docker Desktop

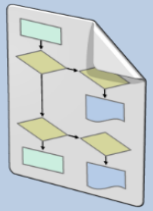
# Uvod u ASP.NET core MVC web aplikacije

# ASP.NET core

- ASP.NET core je besplatan, open-source web framework
- ASP.NET je modularni framework i  radi na Windows, Linux, ili Mac operativnom sistemu
- Verzija ASP.NET Core 7.0 (novembar 2022)
- Isporučuje se u vidu Nuget paketa
- Omogućava kreiranje web aplikacija i web API-ja
- Lako se integriše sa različitim klijentskim tehnologijama
- Koristi  Model-View-Controller (MVC) pattern

# MVC patern

**MVC patern uključuje sledeće komponente:**

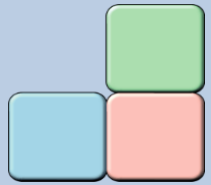


**Models**

Modeli su skup klasa koje opisuju podatke sa kojima radimo.

**Views**

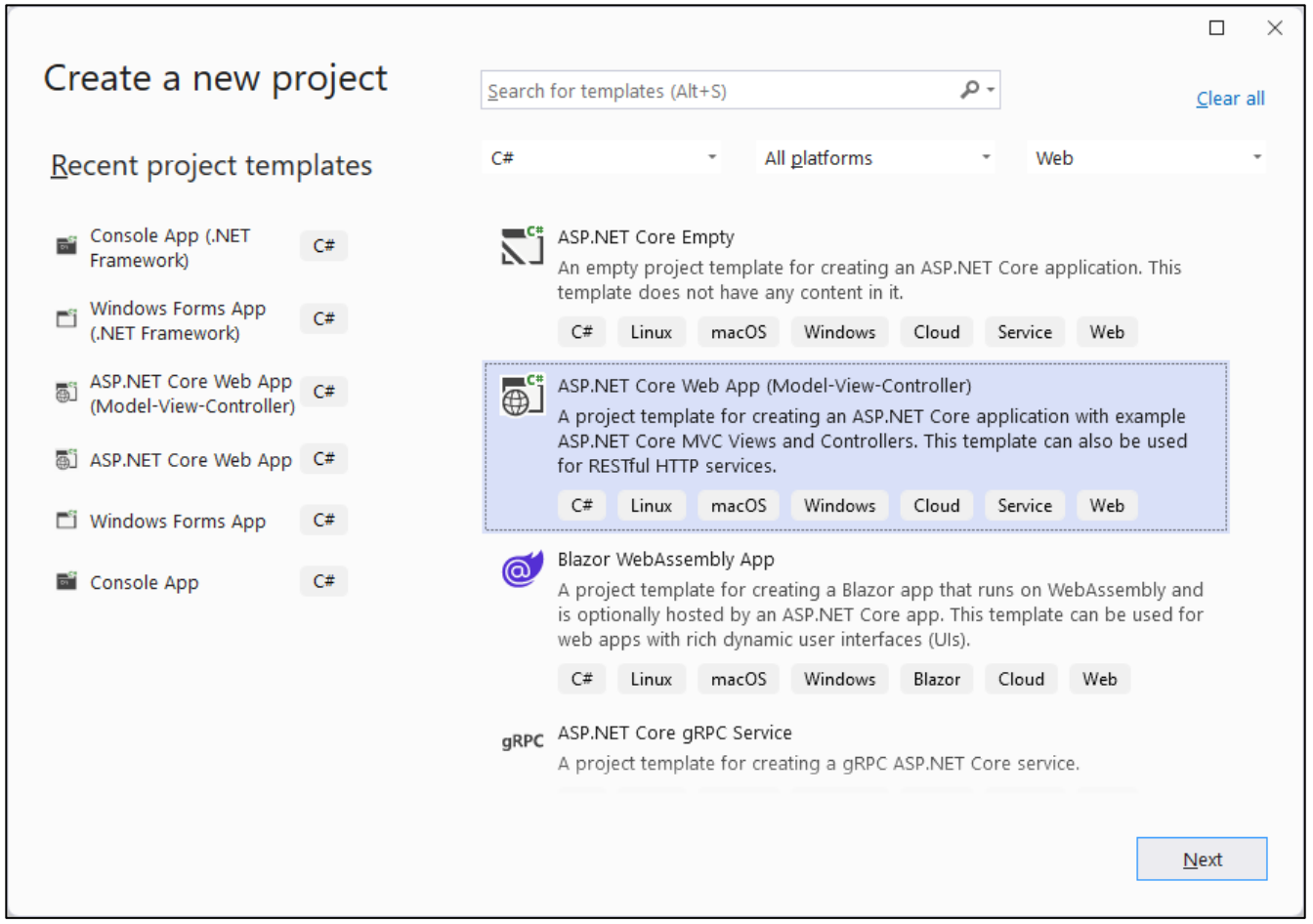
Pogledi su komponente koje generišu korisnički interfejs aplikacije



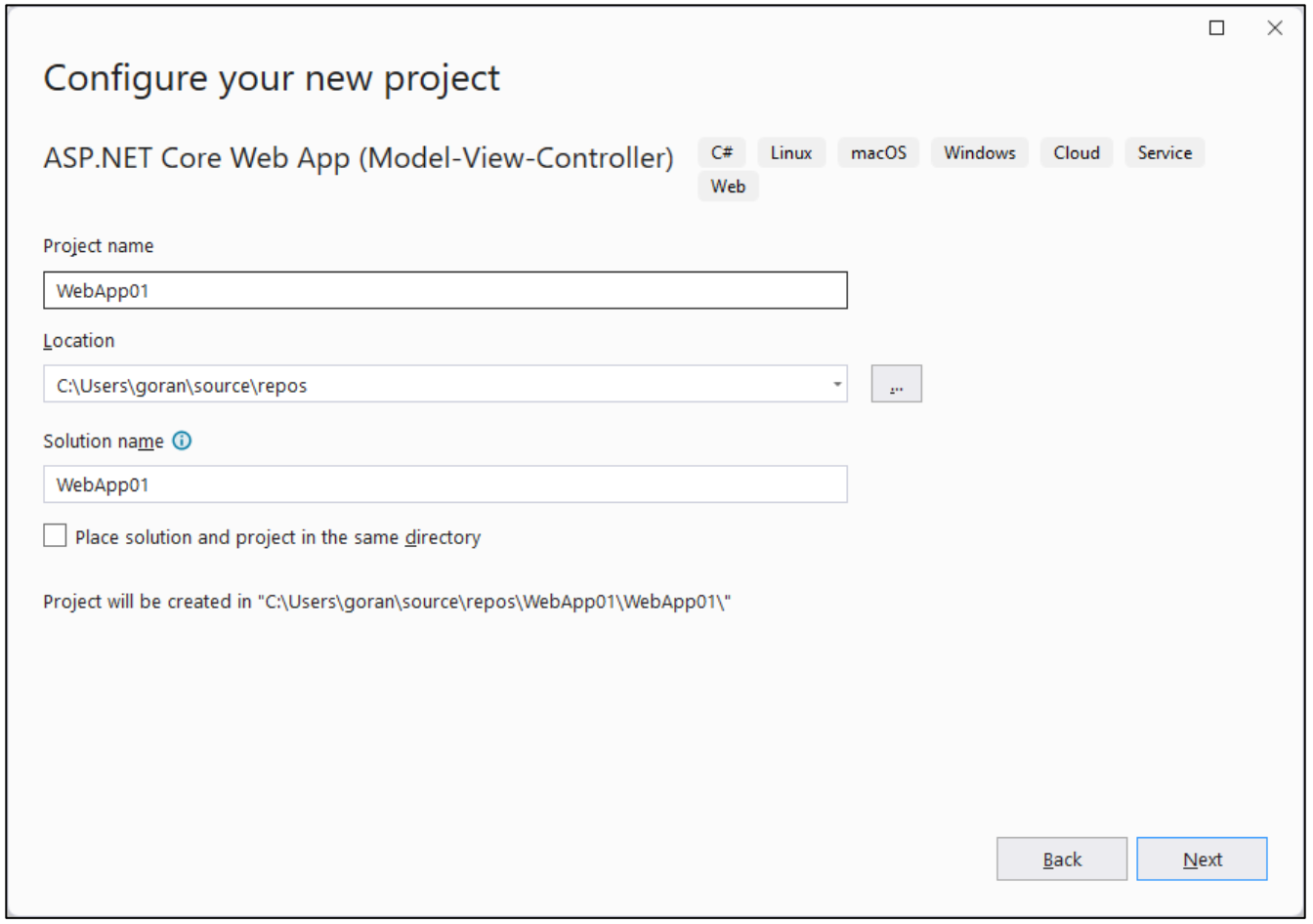
**Controllers**

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

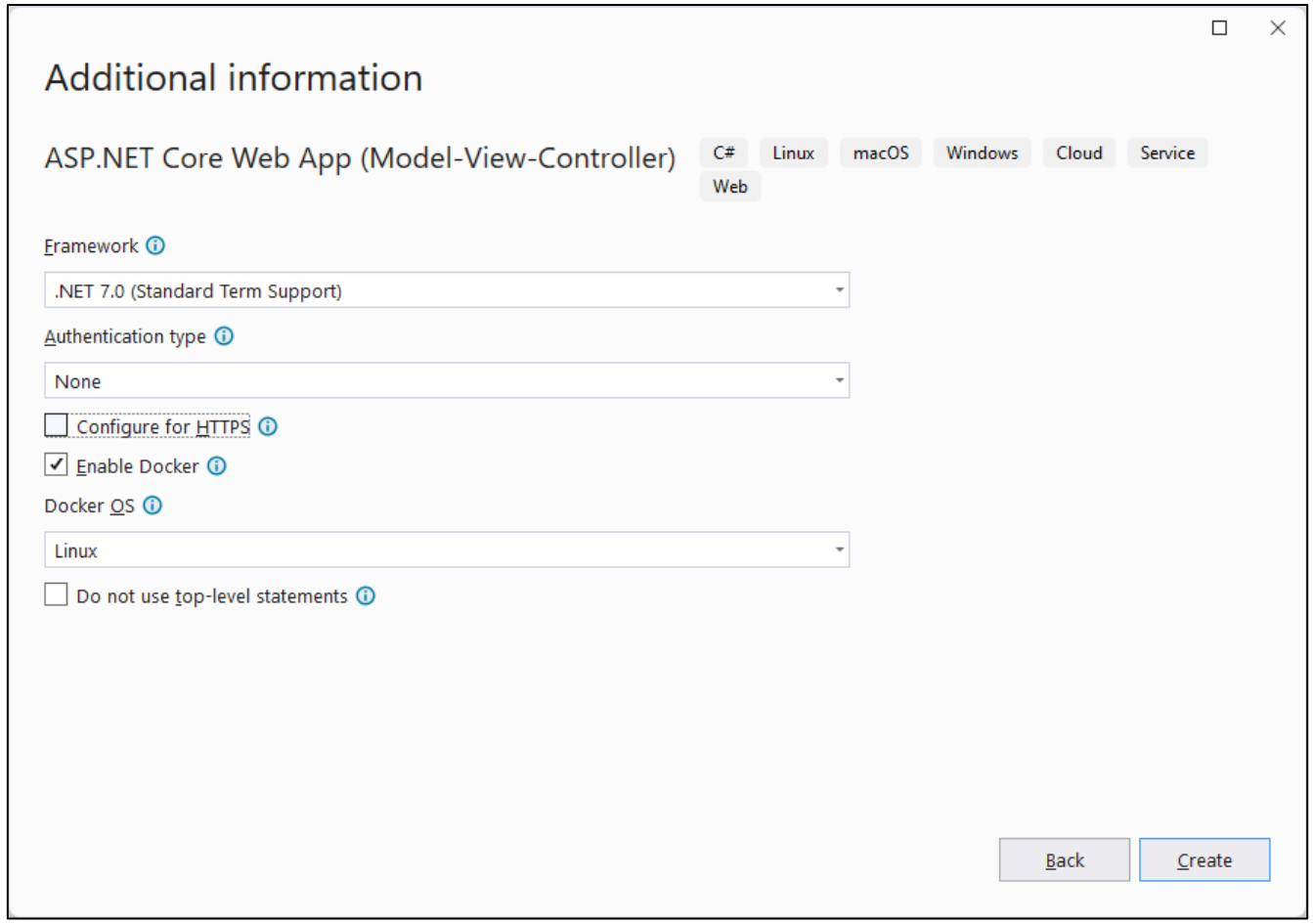
# Kreiranje ASP.NET Core web aplikacije -1



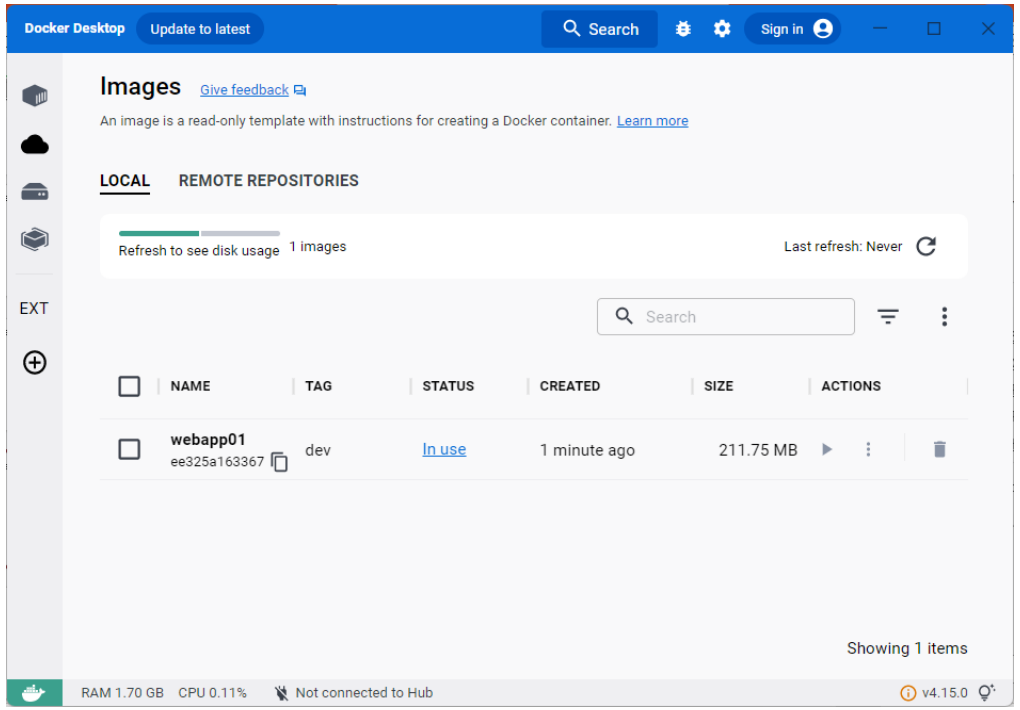
# Kreiranje ASP.NET Core web aplikacije -2

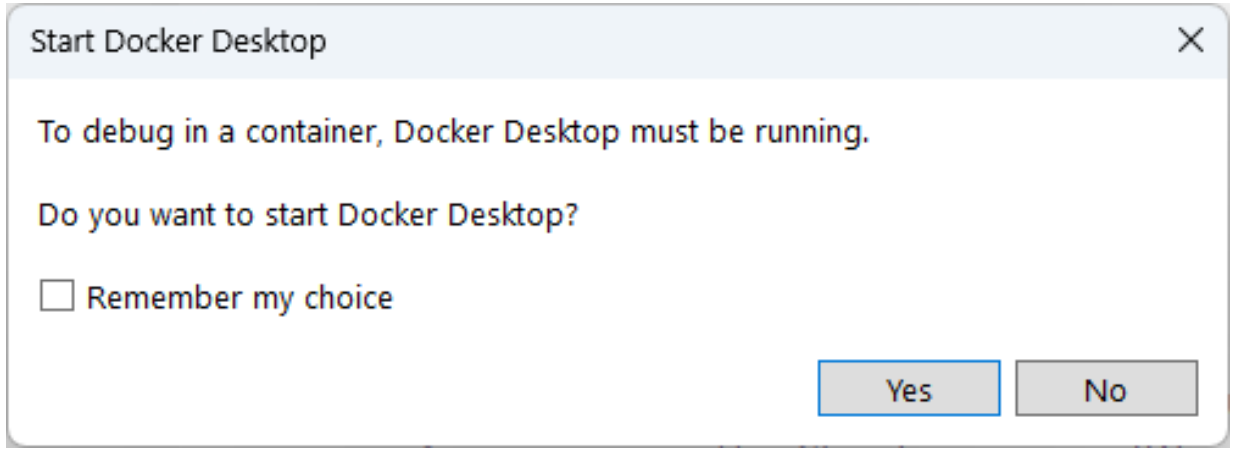


# Kreiranje ASP.NET Core web aplikacije -3

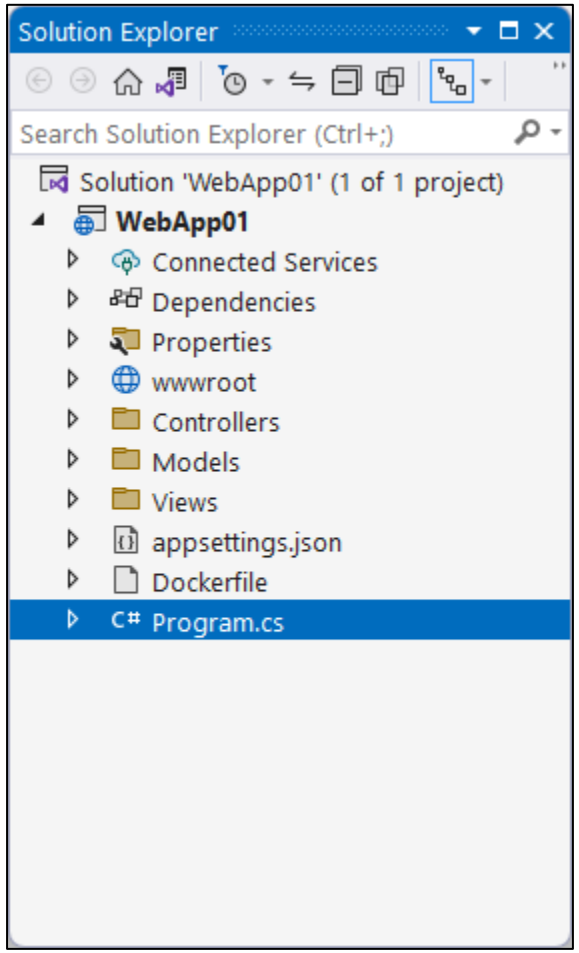


# Docker Desktop

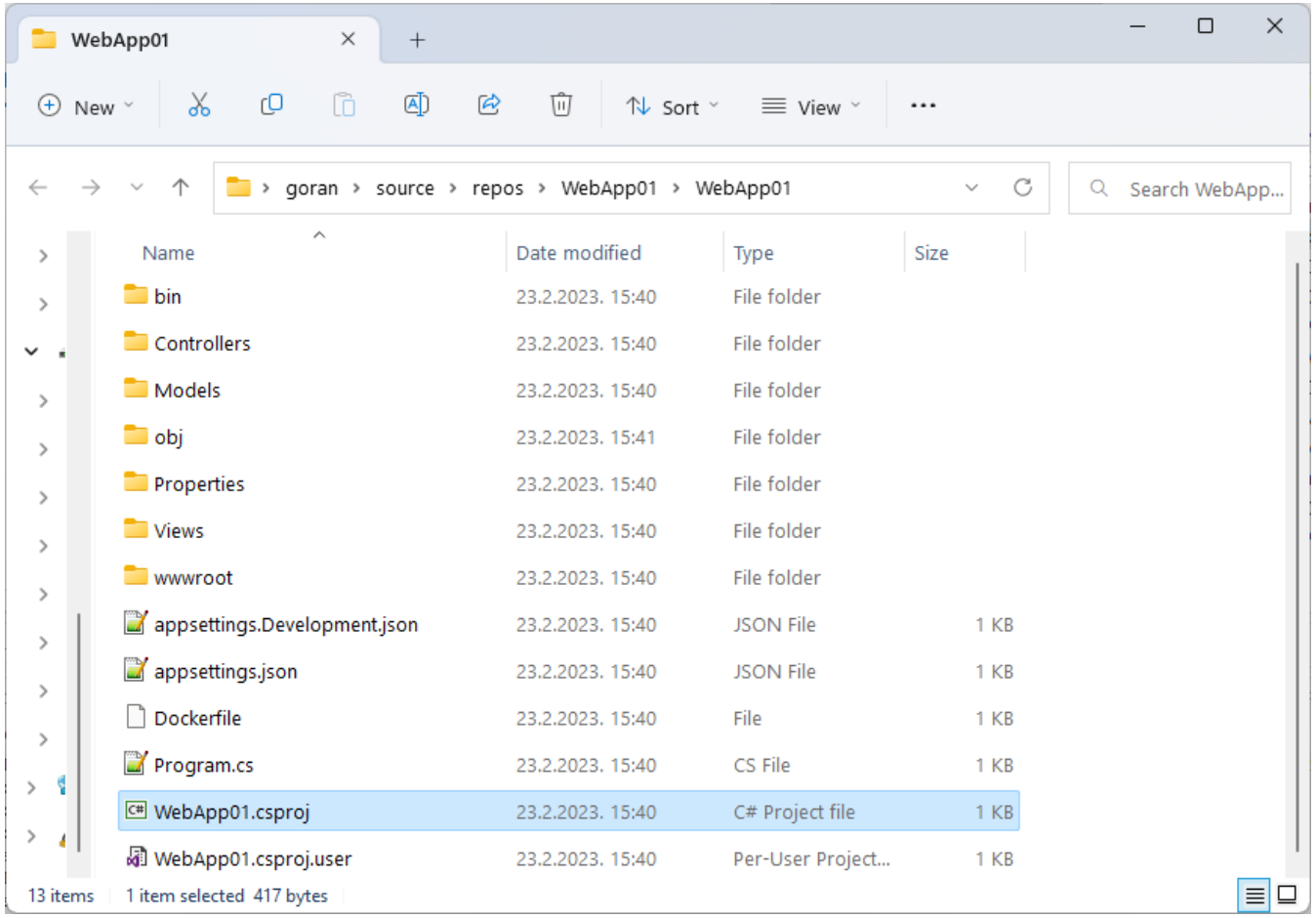


# Struktura ASP.NET Core web aplikacije



# Folder aplikacije

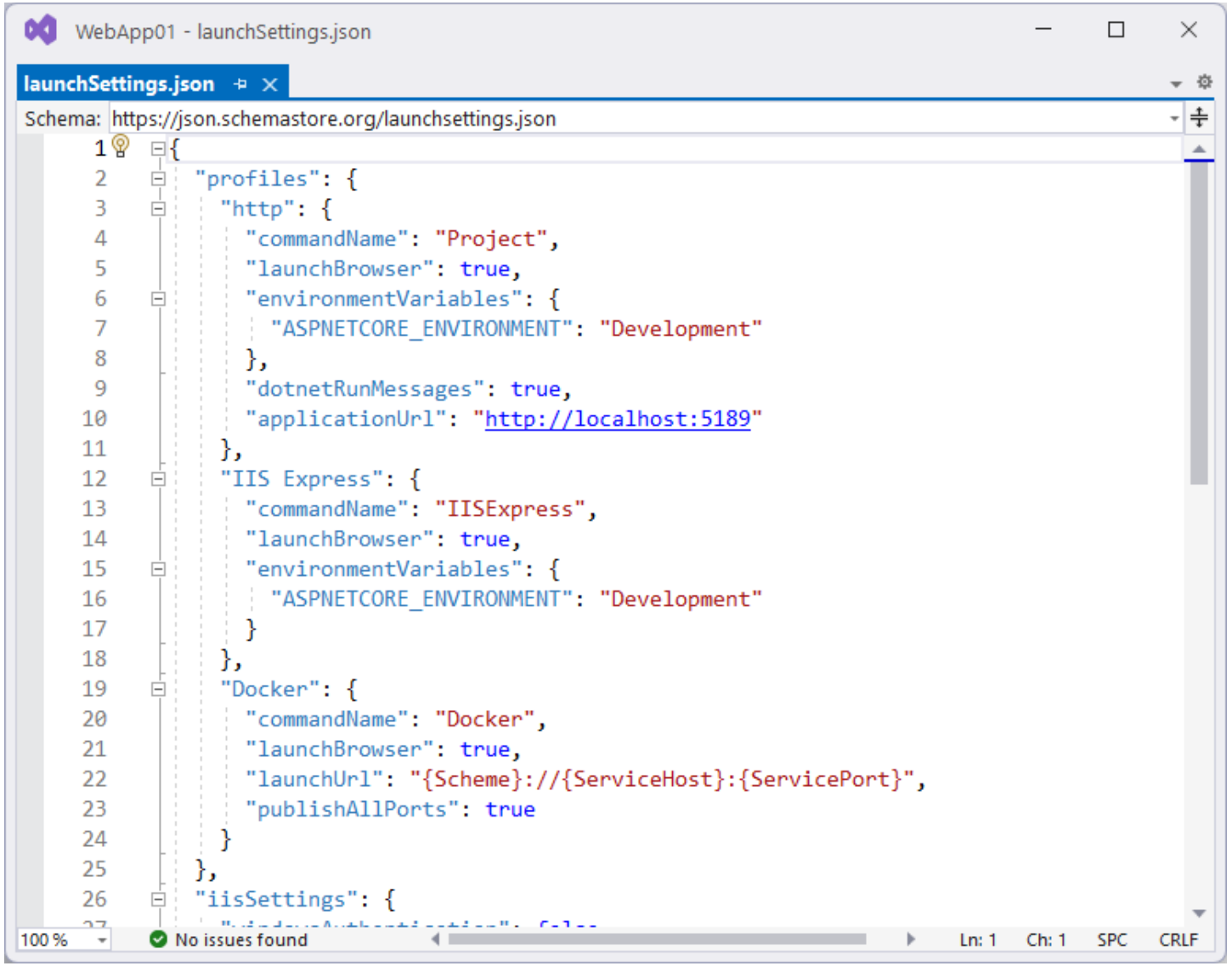


# Fajl .csproject

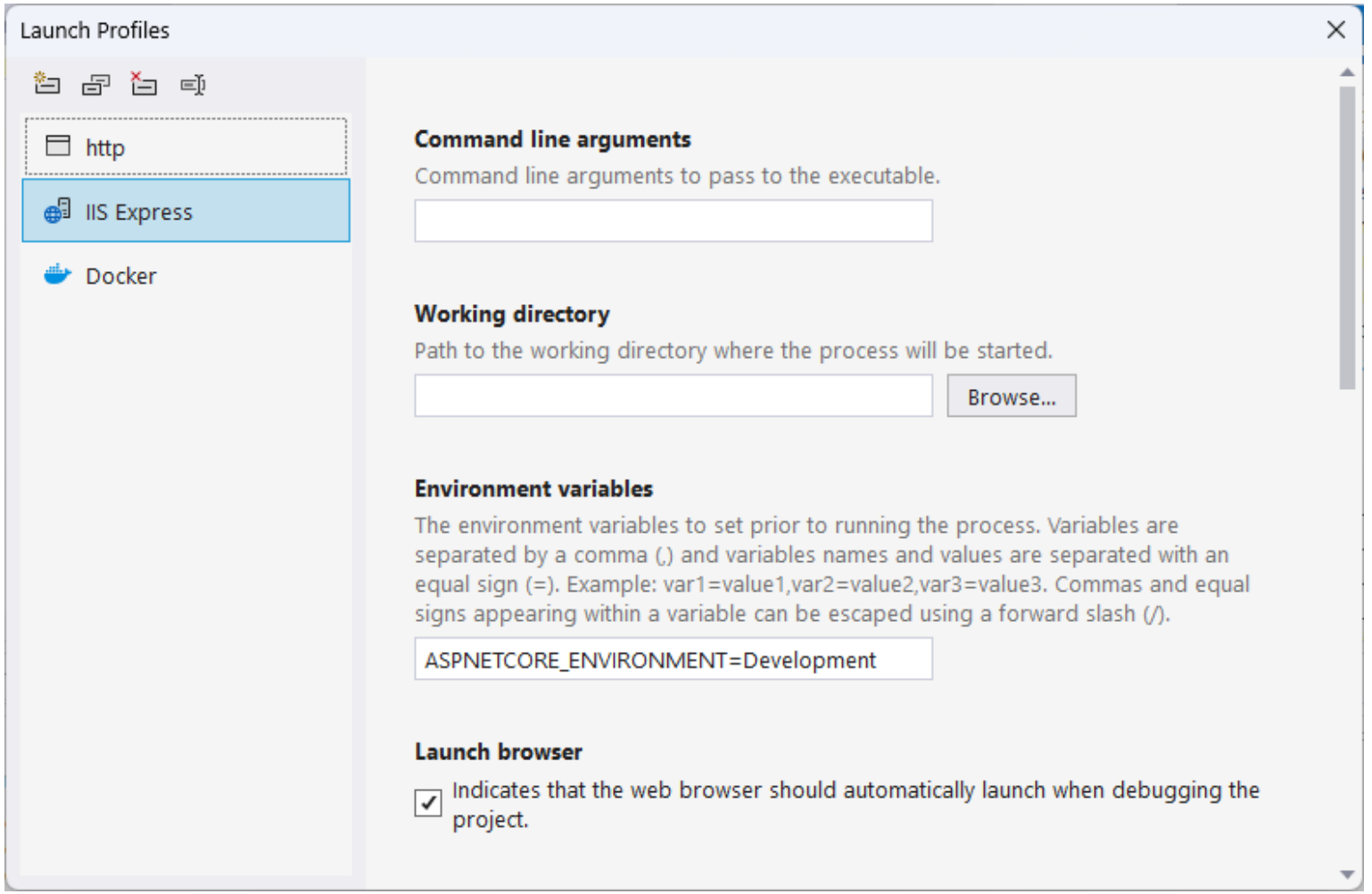
Desni klik na projekat pa opcije Edit Project File

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

  <PropertyGroup>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <DockerDefaultTargetOS>Linux</DockerDefaultTargetOS>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.VisualStudio.Azure.Containers.Tools.Targets"
Version="1.17.2" />
  </ItemGroup>

</Project>
```

# Folder properties fajl launchSettings.json

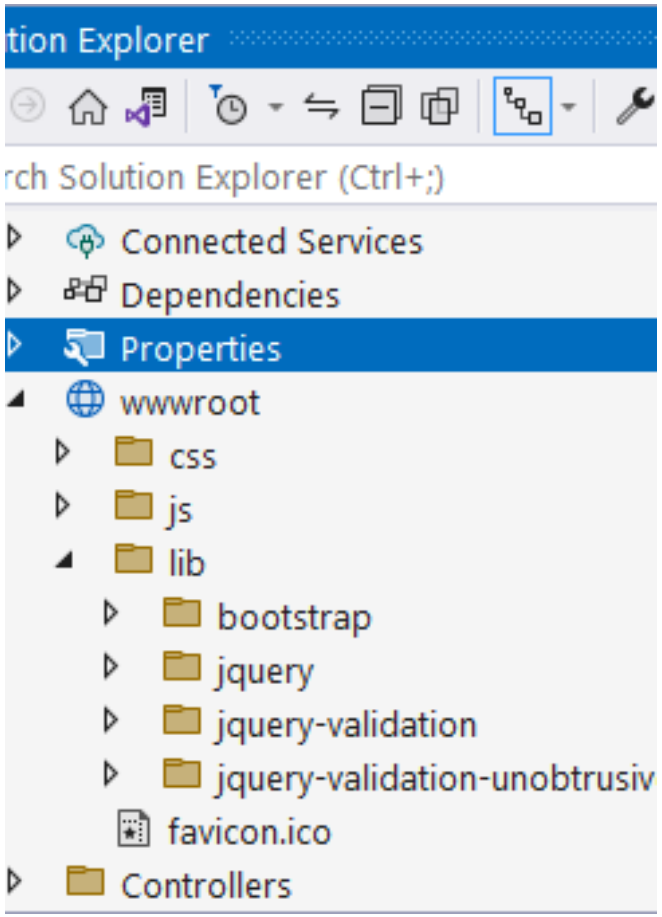


# Prikaz profila za pokretanje aplikacije

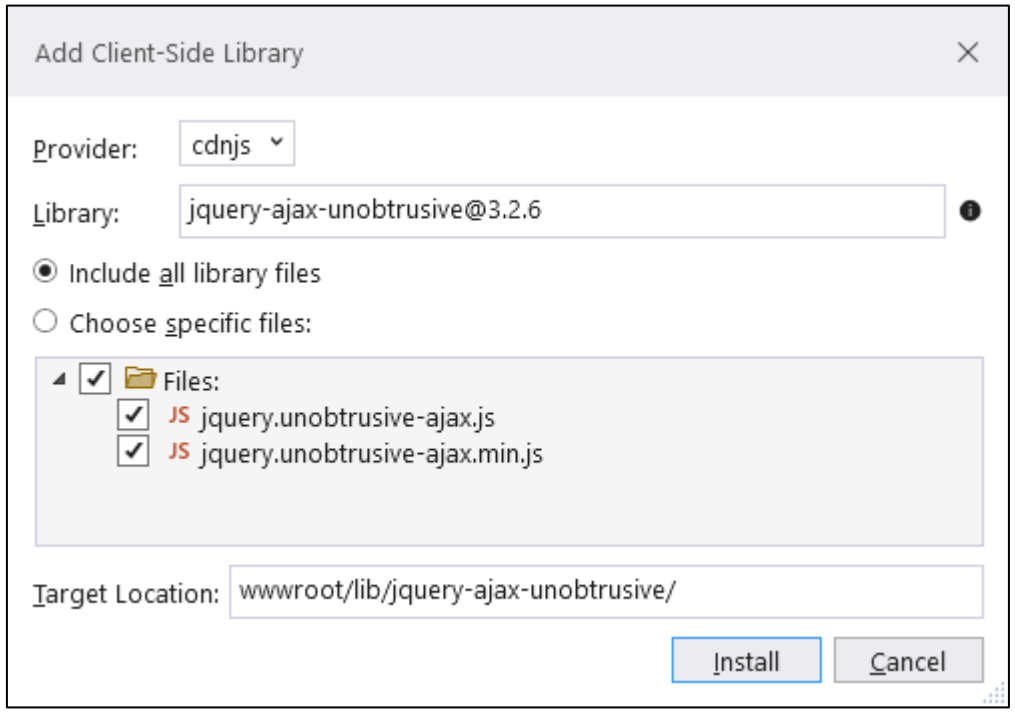


# Folder wwwroot

- Unutar podfoldera foldera wwwroot čuvaju se statički fajlovi
- Svi fajlovi unutar foldera wwwroot su javno dostupni
- Novi projekat sadrži biblioteke jquery i bootstrap

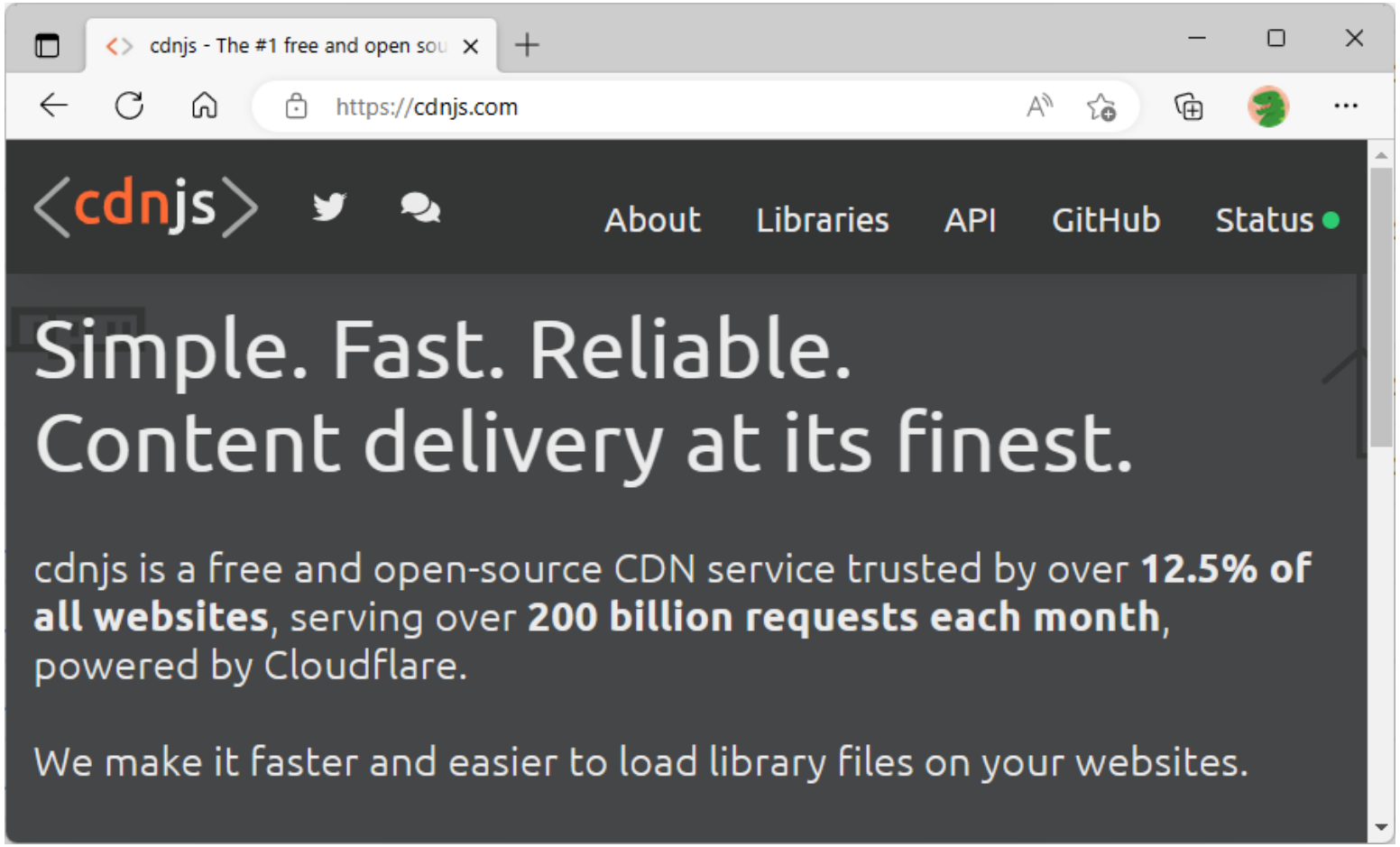


# Dodavanje klijentskih biblioteka



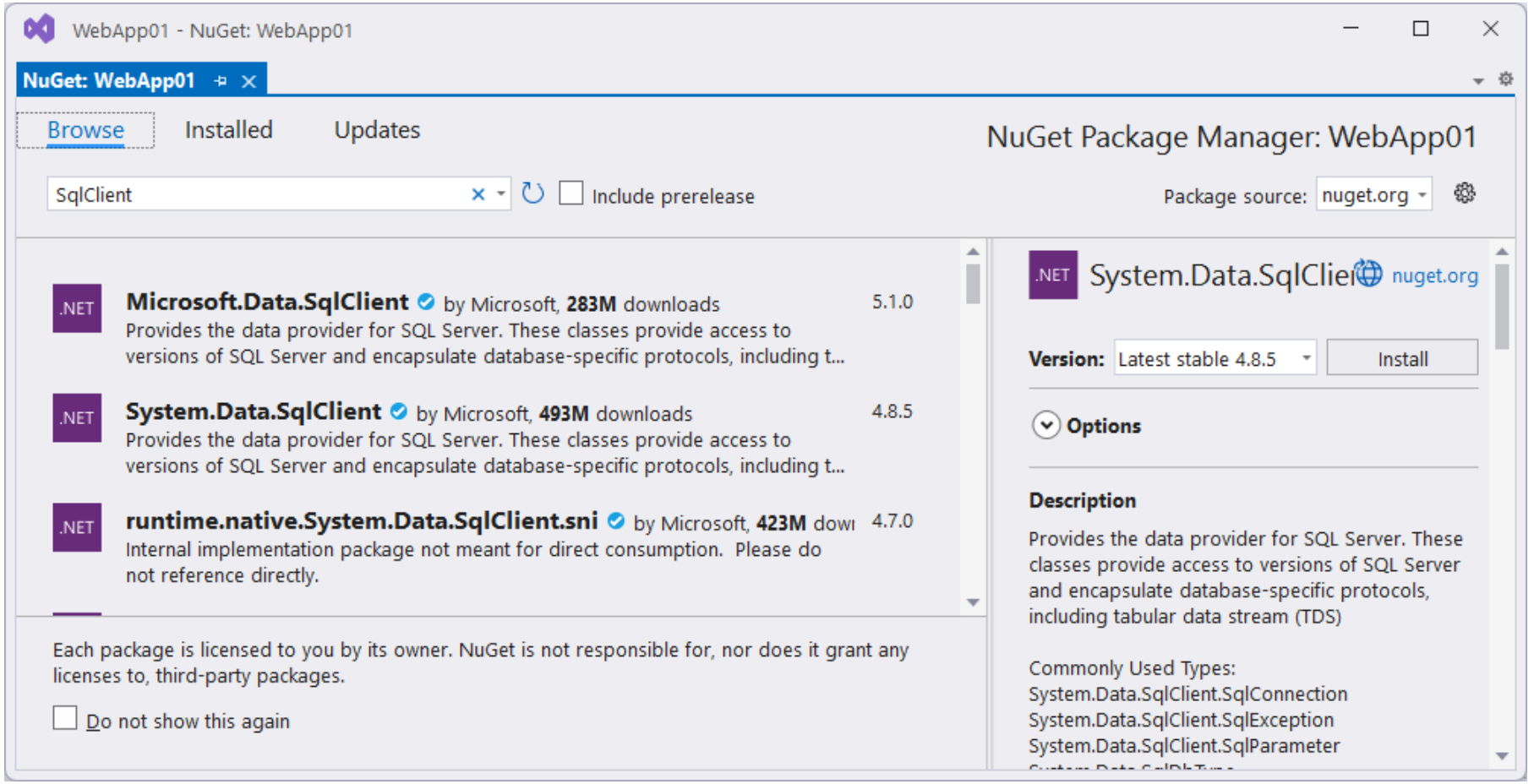

Desni klik na folder lib Add Client-Side library
ili
Project meni opcija Add Client-Side library

# cdnjs.com


cdnjs - The #1 free and open sou
https://cdnjs.com
cdnjs
About
Libraries
API
GitHub
Status
Simple. Fast. Reliable.
Content delivery at its finest.
cdnjs is a free and open-source CDN service trusted by over **12.5% of all websites**, serving over **200 billion requests each month**, powered by Cloudflare.
We make it faster and easier to load library files on your websites.

# Dodavanje serverskih biblioteka

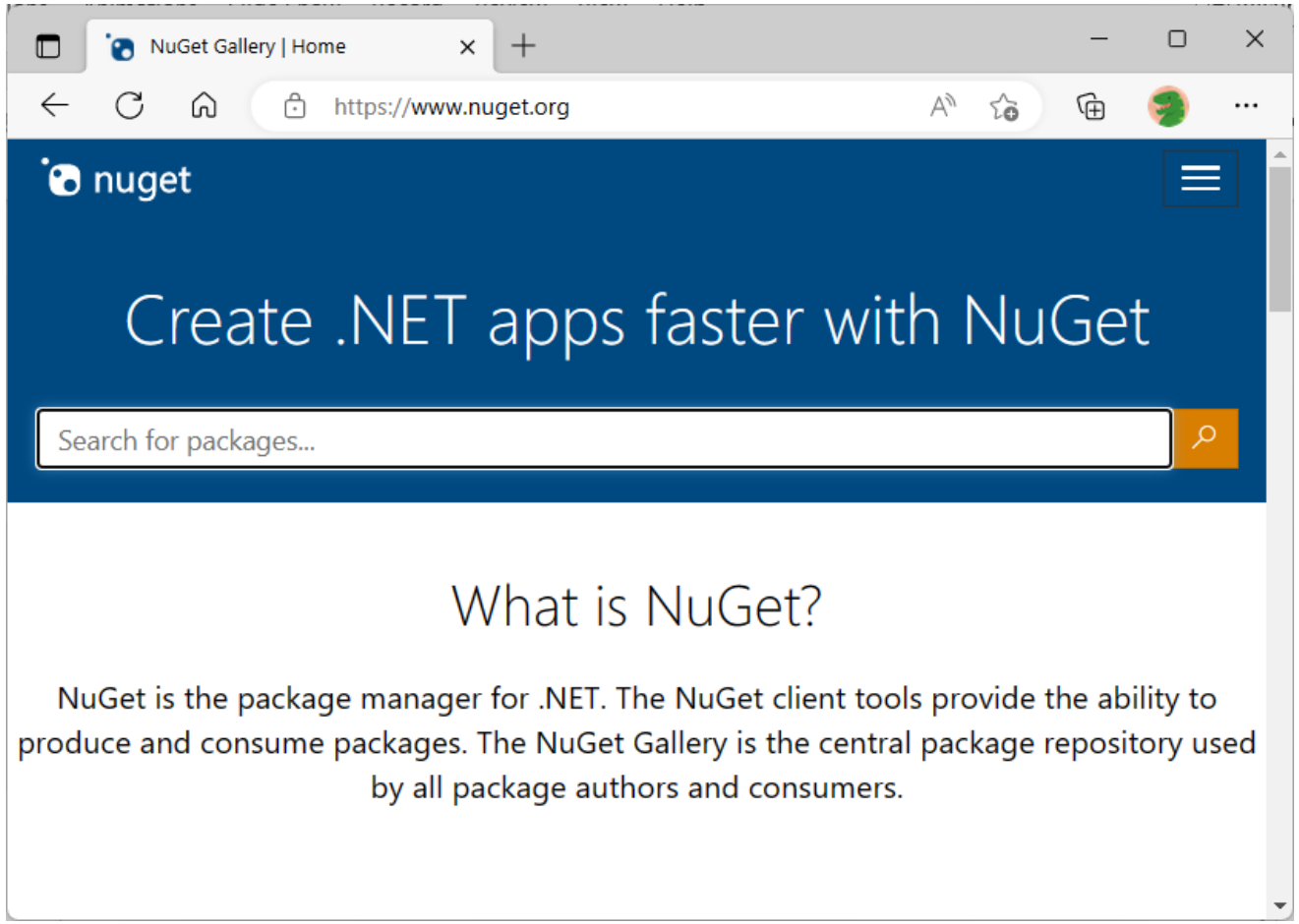
Project -> Manage Nuget Packages



# Fajl .csproj

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

  <PropertyGroup>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <DockerDefaultTargetOS>Linux</DockerDefaultTargetOS>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.VisualStudio.Azure.Containers.Tools.Targets"
Version="1.17.2" />
    <PackageReference Include="System.Data.SqlClient" Version="4.8.5" />
  </ItemGroup>

</Project>
```

# www.nuget.org



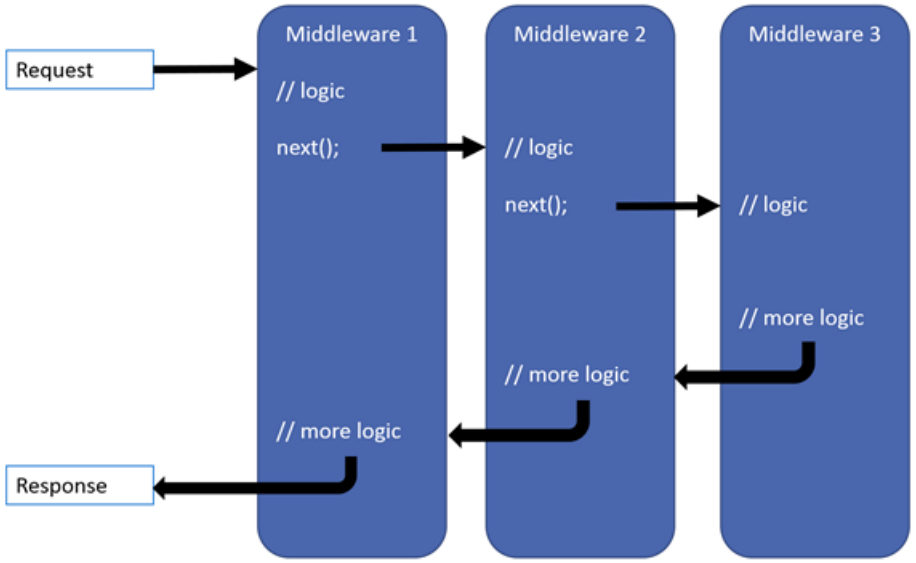
# Dependency Injection container

- Servis je klasa u ASP.NET Core web aplikaciji koja će biti korišćena u drugoj klasi
- ASP.Core ima ugrađen Dependency Injection container koji vrši automatsko ubacivanje klase servisa u zavisnu klasu
- Servis se mora registrovati pre korišćenja
- Nakon registracije servis se može koristiti bilo gde u aplikaciji
- U aplikaciji je samo potrebno da u konstruktoru zavisne klase kao parametar definišemo referencu tipa servisa i DI kontejner će je automatski ubaciti

# Middleware

- ASP.NET Core uvodi koncept koji se zove middleware
- Middleware je komponenta koja se izvršva pri svakom zahtevu
- Middleware se sastoji od komponenti koje obrađuju zahtev putem protočne obrade
- Middleware se konfiguriše unutar fajla Program.cs
- Redosled kojim su komponente dodate  određuju redosled kojim se pozivaju pri obradi zahteva i obrnutom redosledu pri odgovoru
- Redosled middleware komponenti je bitan za sigurnost, perfomanse aplikacije i njenu funkcionalnost.

# Protočna obrada zahteva




- Zahtev se prosleđuje od jedne do druge komponente
- Svaka komponenta može odlučiti da ne prosledi zahtev sledećoj komponenti što se zove kratko spojanje protočne obrade zahteva
- Kratko spajanje je često poželjno jer se izbegava nepotreban rad.
- Npr. komponenta za obradu statičkih fajlova može vratiti zahtev za statičkim fajlom i prespojiti ostatak protočne obrade.

# 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.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

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

app.Run();
```

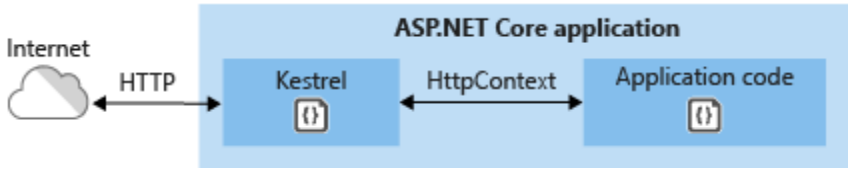
# Fajl Program.cs

- Prvo se kreira instanca klase **WebApplicationBuilder** korišćenjem metode CreateBuilder klase WebApplication
- Zatim se u kolekciju servisa aplikacije registruju MVC specifični servisi korišćenjem metode **AddControllersWithViews**
- Web aplikacije se kreira korišćenjem **Build** metode
- Nakon toga se konfiguriše middleware komponenta tj. protočna obrada http zahteva
- Zatim se konfigurišu endpointi za rutiranje
- Pozivom metode **run** startuje se web aplikacije

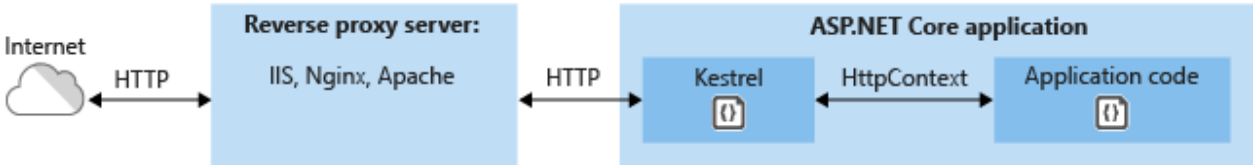
# Novi hosting model

- ASP.NET Core 7.0 aplikacije koriste novi minimalni hosting model
- Registracija i konfigurisanje servisa i protočna obrada zahteva se konfiguriše unutar fajla Program.cs
- Podrazumevano se za pokretanje aplikacije koristi Kestrel web server

# Kestrel web server

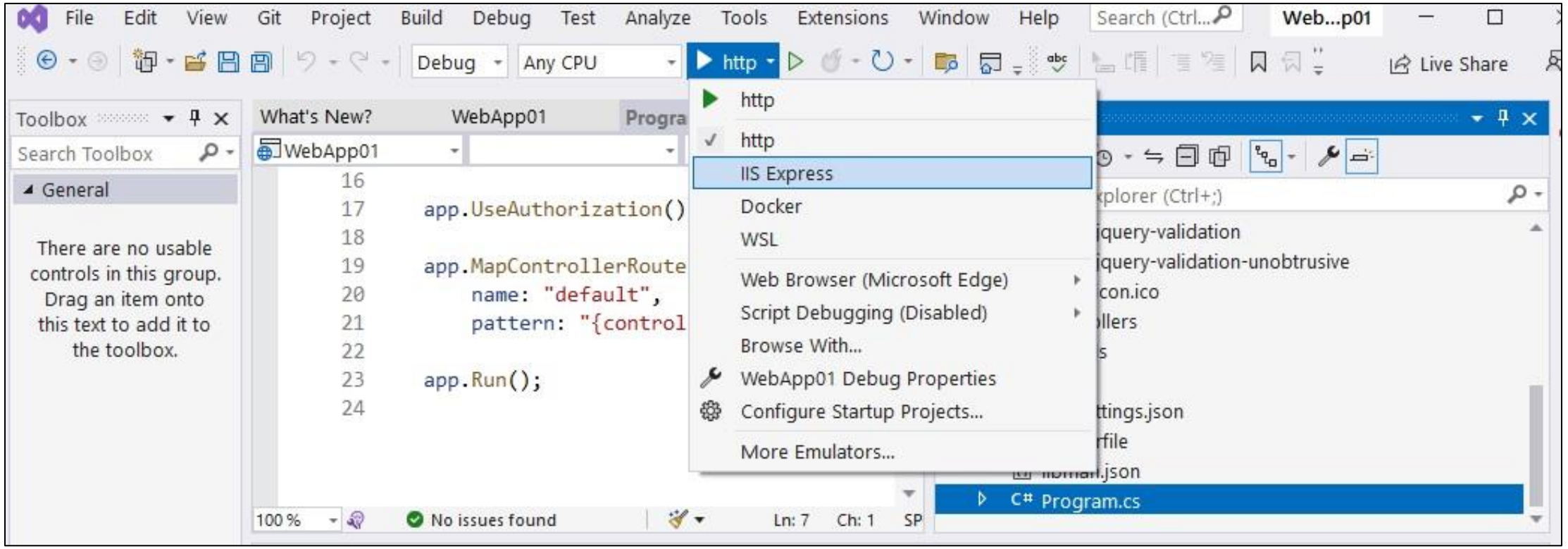



- Kestrel ne podržava deljenje iste IP adrese i porta između više aplikacija
- Ne može da hostuje više web sajtova na istoj IP adresi i portu

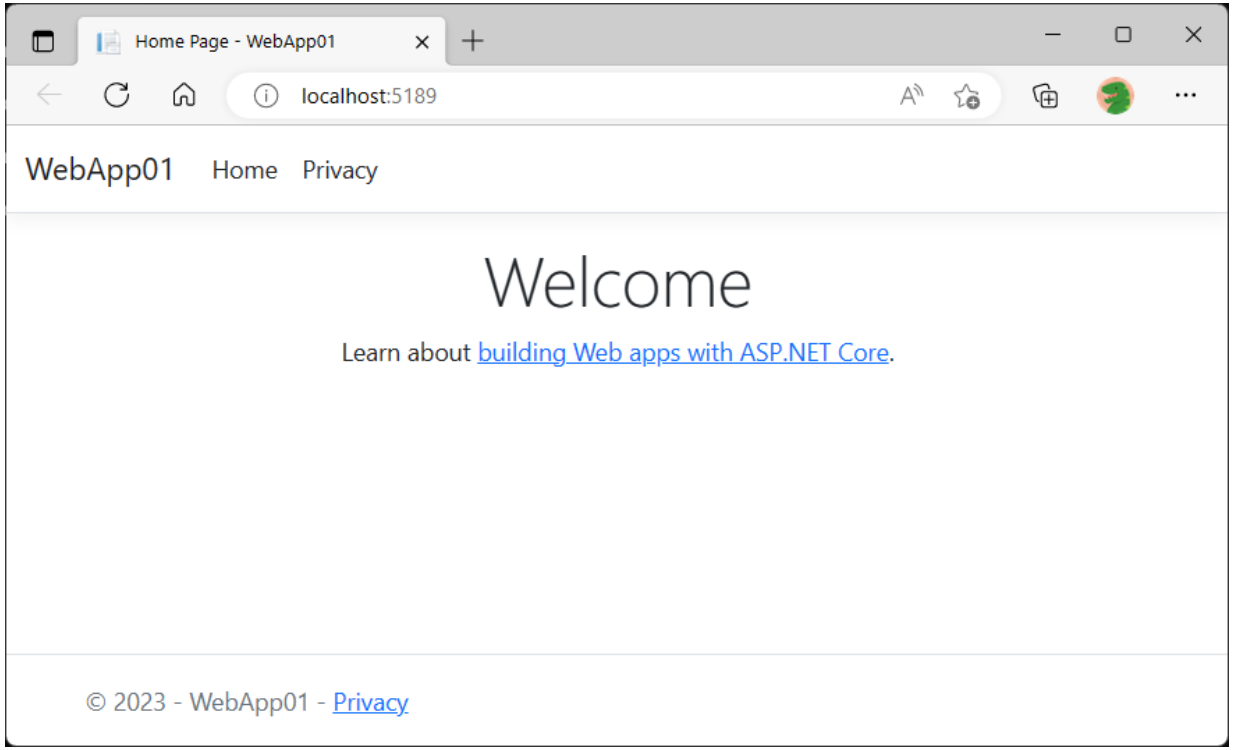


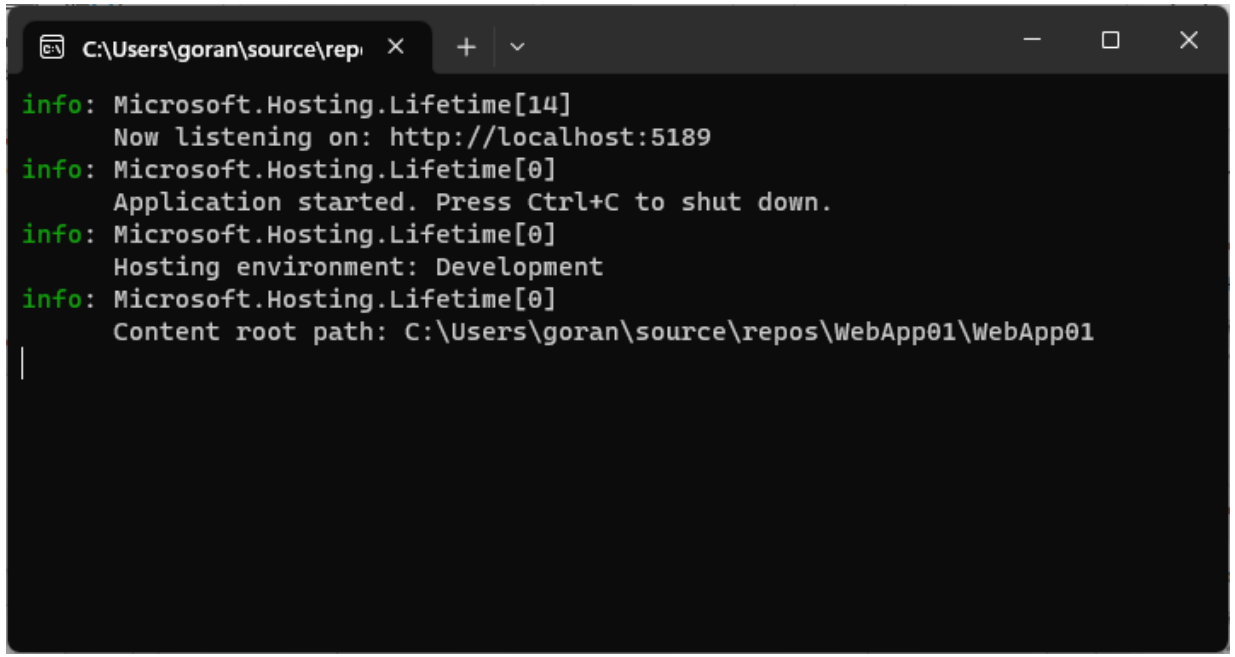

- Integracija sa IIS web serverom
- Reversni proxy server (IIS) prima HTTP zahteve sa interneta i prosleđuje ih kestrelu vršeći prethodno neku obradu
- Scenario koji zahteva reverzni proxy je kada imamo više aplikacija koje dele istu IP adresu  i port i rade na istom serveru

# Prvo pokretanje ASP.NET Core web aplikacije

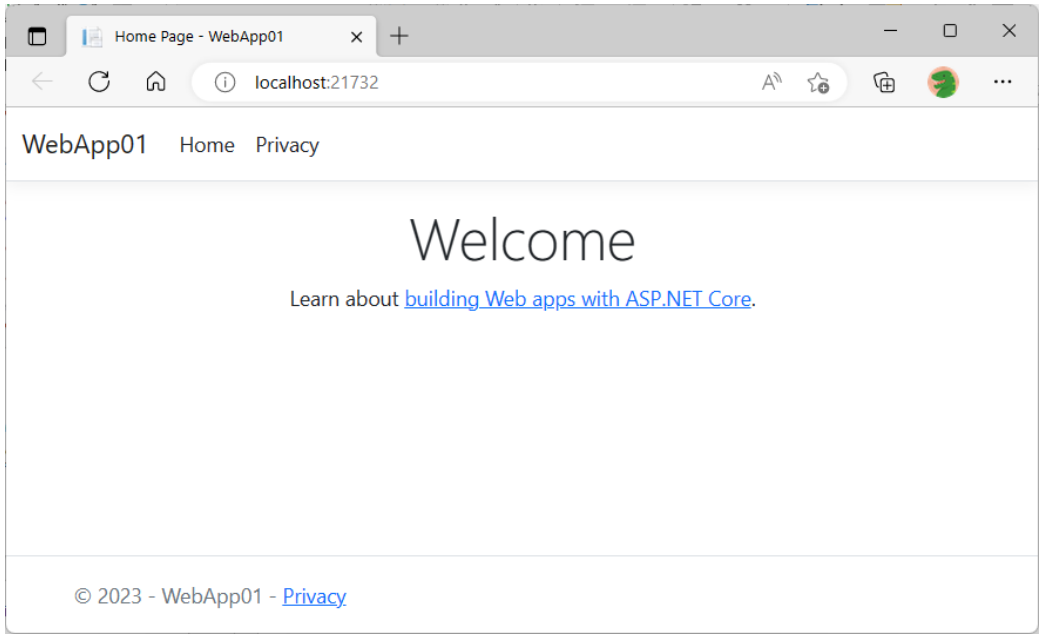


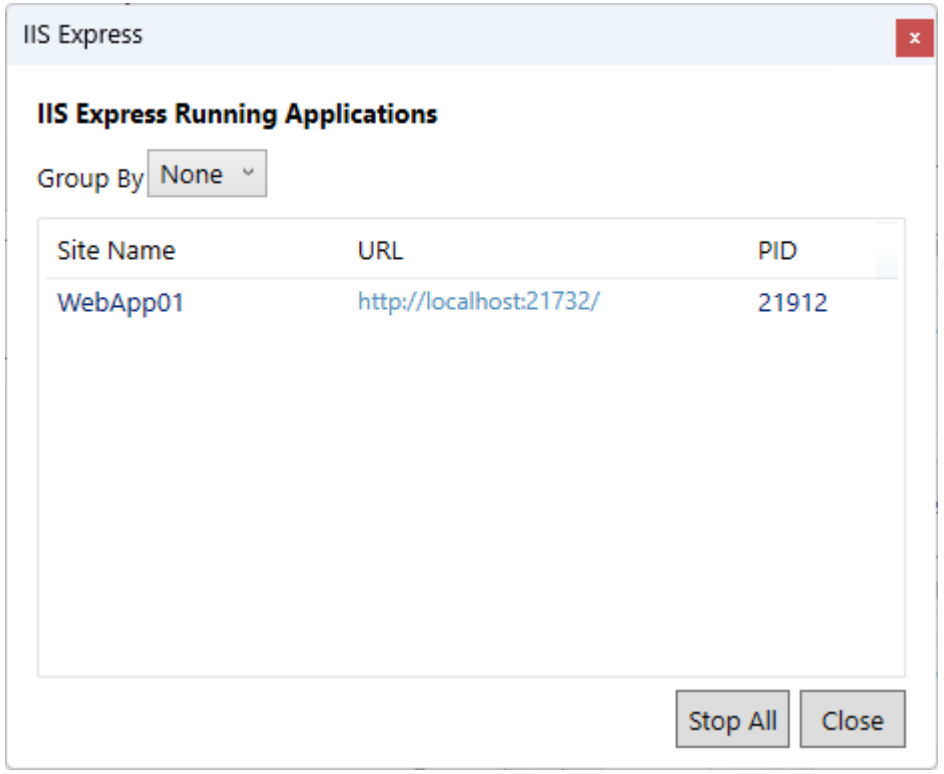
# Prvo pokretanje ASP.NET Core web aplikacije korišćenjem Kestrel servera (http profil)



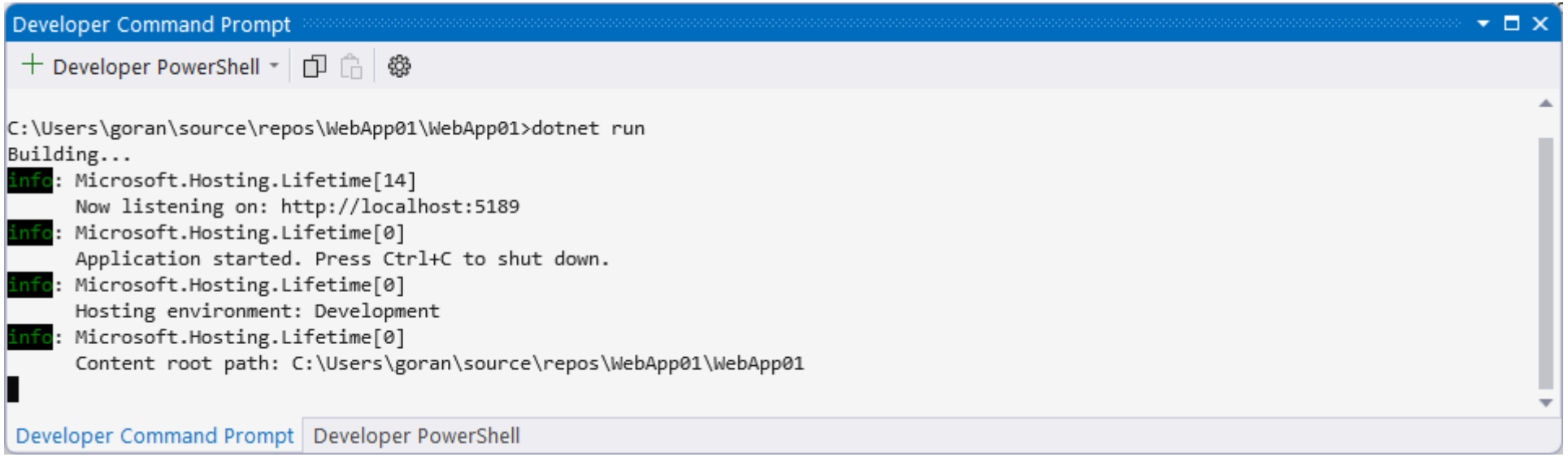



# Prvo pokretanje ASP.NET Core web aplikacije – profil iis



# Pokretanje aplikacije iz terminala



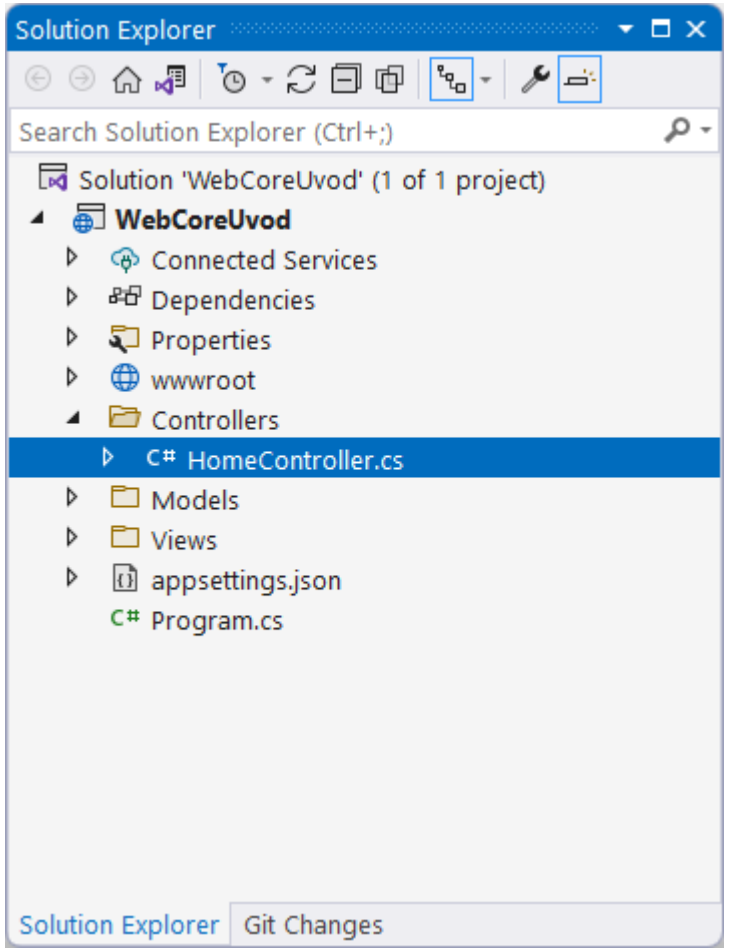

View->Terminal za prikaz terminala

dotnet run – pokretanje web aplikacije

# 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 rečnik koji sadrži key-value parove
- Ključevi su stringovi
- Koristi se za prosleđivanje podataka iz kontrolera ka pogledu
- Kada se pristupa podacima iz pogleda moramo izvršiti neophodno kastovanje ukoliko ne čuvamo stringove

# Prosleđivanje podataka Index pogledu HomeController klase

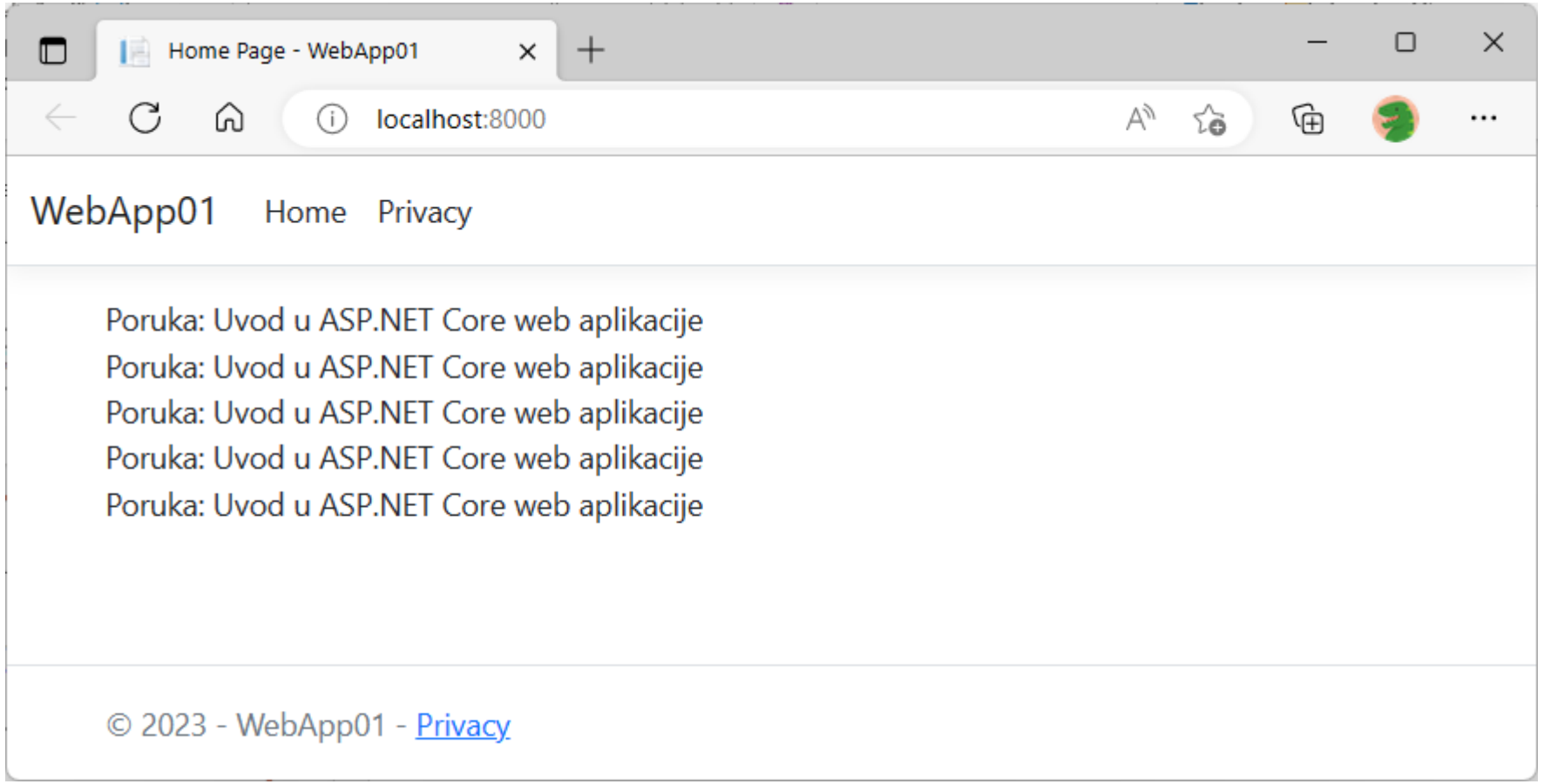
```
public IActionResult Index()
{
    ViewData["brojPonavljanja"]= 5;
    ViewData["poruka"] = "Uvod u ASP.NET Core web aplikacije";

    return View();
}
```

Kada se unutar akcione metode ne napiše ime pogleda kome se prosleđuju podaci, već samo return View(); onda se podaci prosleđuju pogledu koji ima isto ime kao i akciona metoda.

# 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

- Koristi se za prenos podataka iz kontrolera ka pogledu
- ViewBag je dinamički objekat unutar koga se dninamički definišu svojstva
- ViewBag ne zahteva kastovanje pri iščitavanju vrednosti iz njega
- Svojstvo unutar ViewBaga ne sme da se poklopi sa ključem ViewData rečnika

# Definisanje dinamičkih svojstava

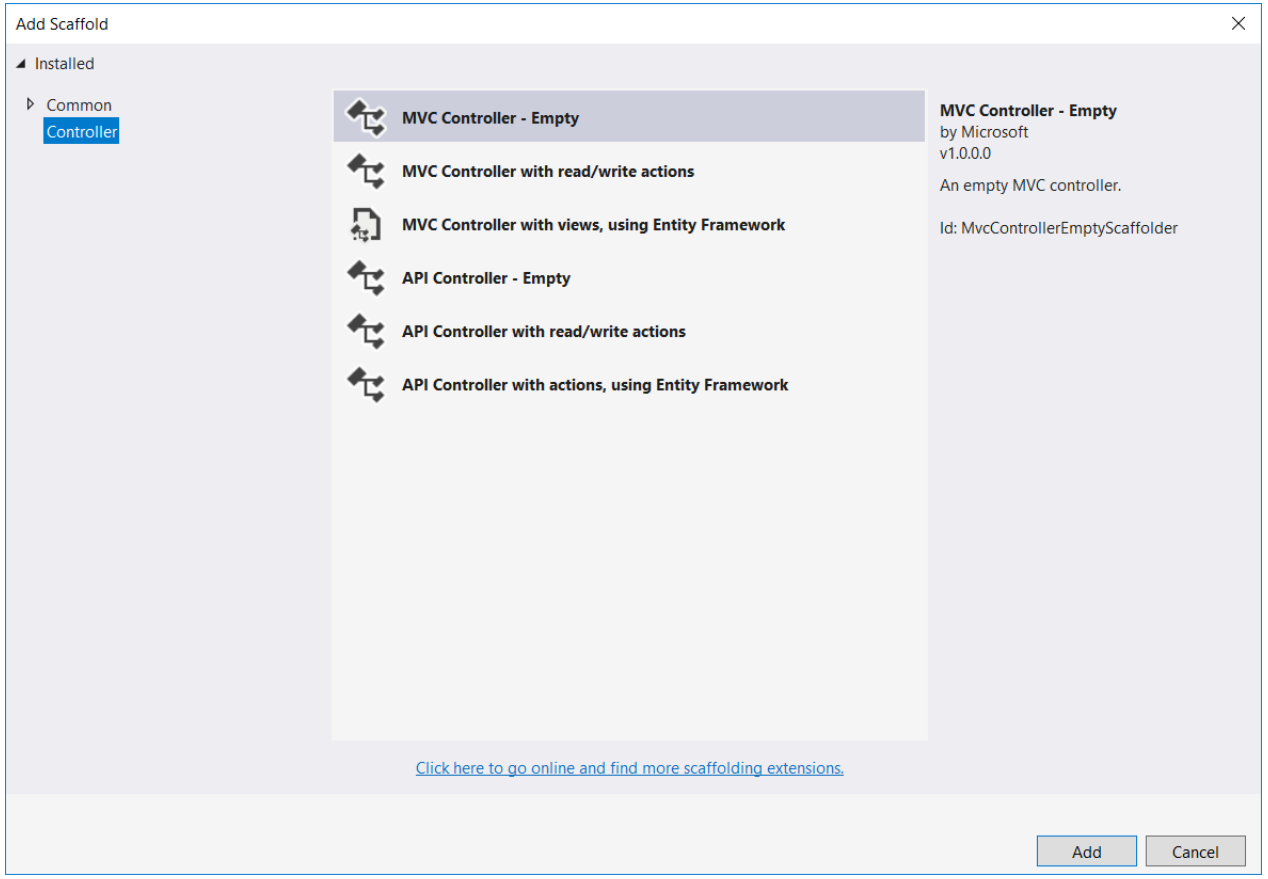
```
public IActionResult Index()
{
    ViewBag.brojPonavljanja = 5;
    ViewBag.poruka = "Uvod u ASP.NET Core web aplikacije";
    return View();
}
```

# Pogled Index.cshtml

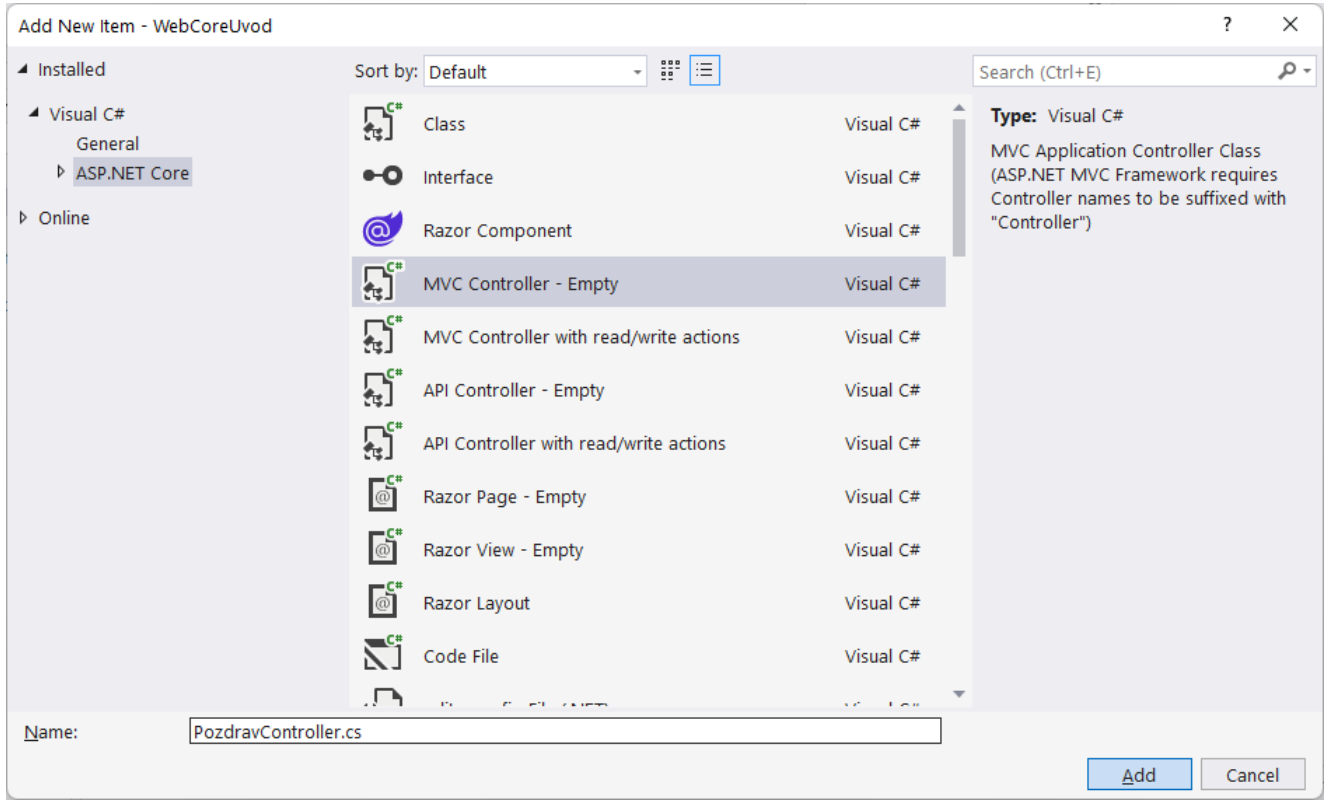
```
<div class="row">
    <div class="col-md-6">
        @for (int i = 0; i < ViewBag.brojPonavljanja; i++)
        {
            <span>Poruka: @ViewBag.poruka</span><br />
        }
    </div>

</div>
```

# Prosleđivanje podataka metodi kontrolera

# Kreiranje praznog MVC kontrolera



# Imenovanje kontrolera



# Klasa PozdravController

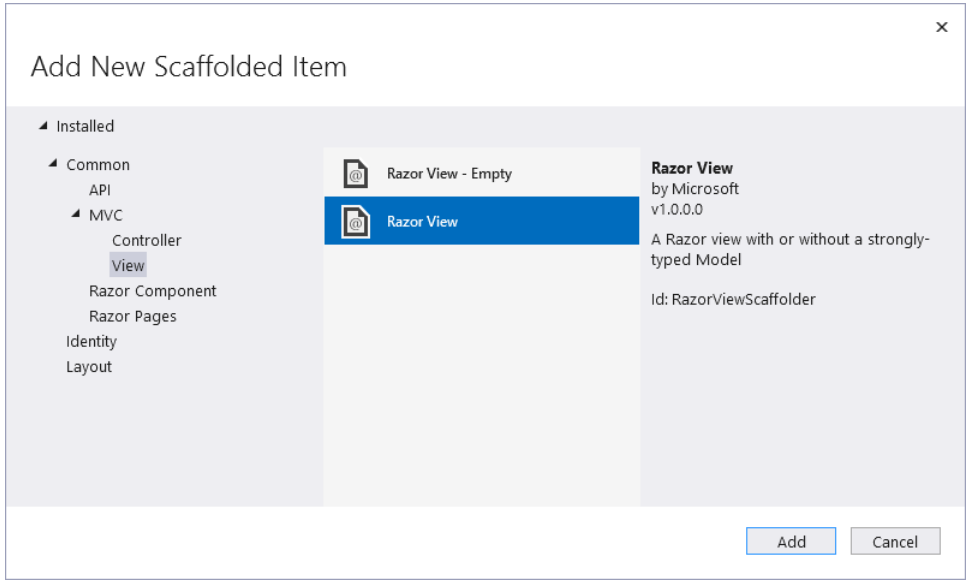
```
public class PozdravController : Controller
{
    public IActionResult Index()
    {
        return View();
    }
}
```

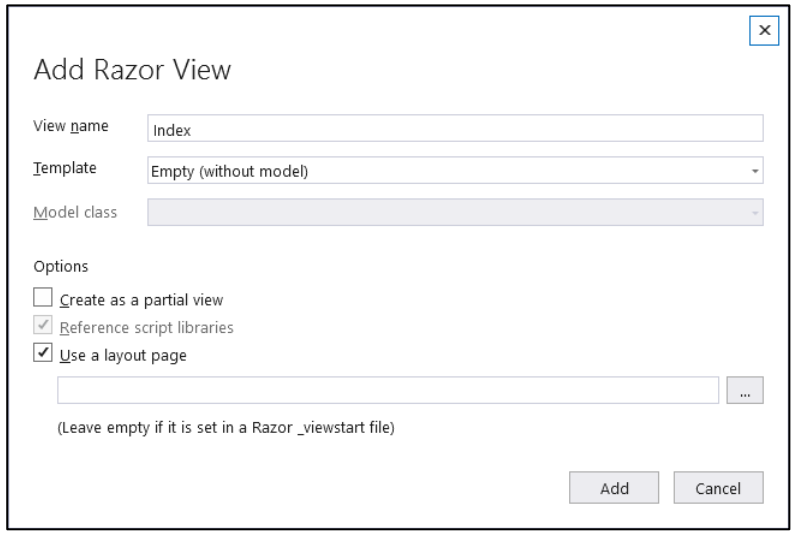
# Index metoda klase PozdravController

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

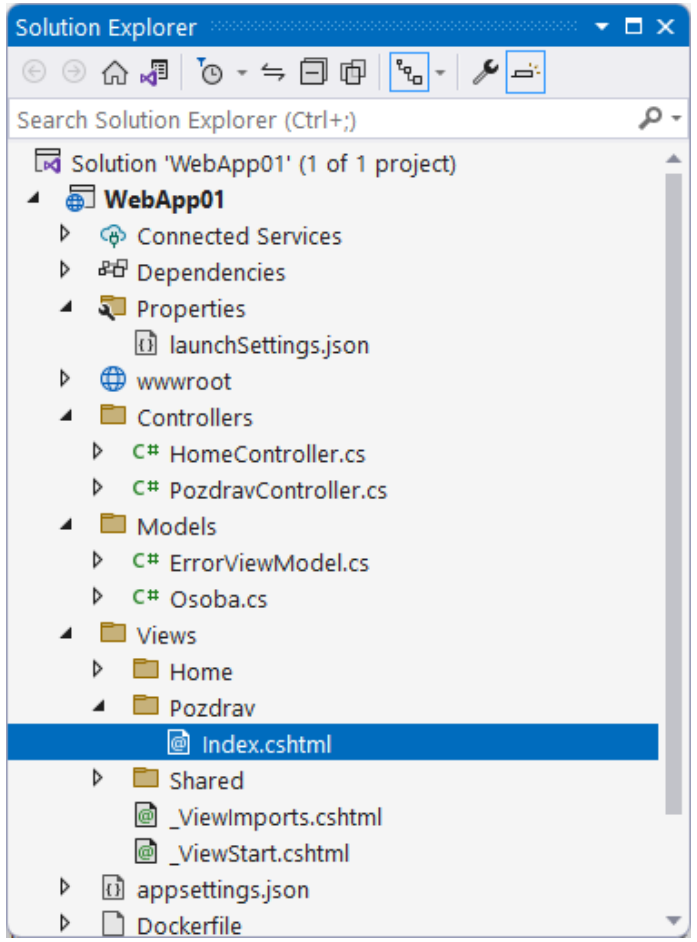
# Index pogled PozdravController klase

Desni klik unutar metode Index pa opcija Add-> View



# Folder Views/Pozdrav



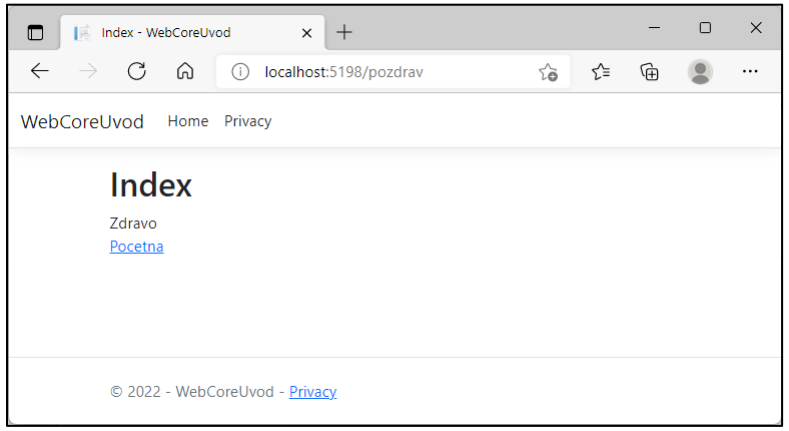
# Index.cshtml klase PozdravController

```
@{
    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:5000/pozdrav/index

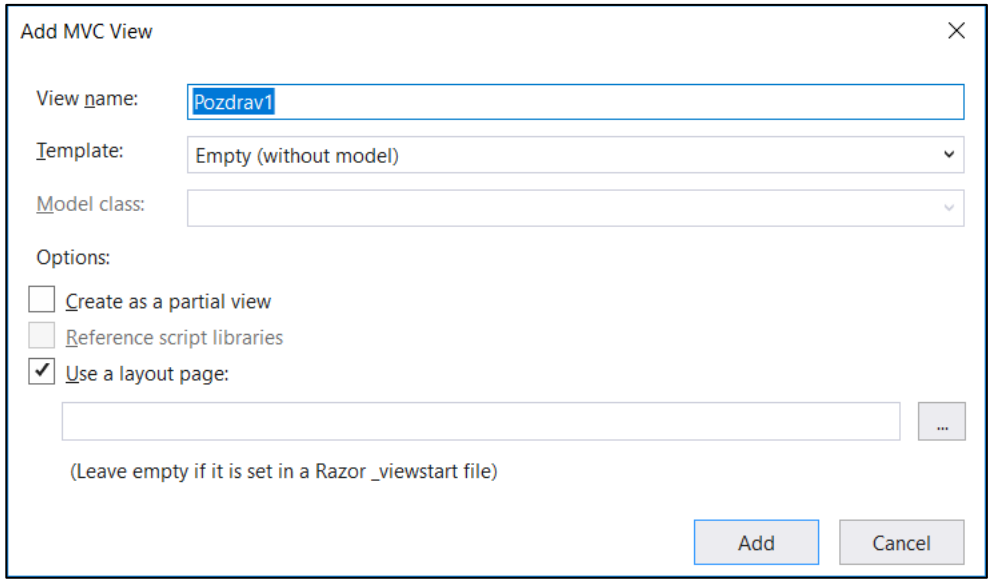
http://localhost:5000/pozdrav



# Akciona metoda Pozdrav1 PozdravController klase

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

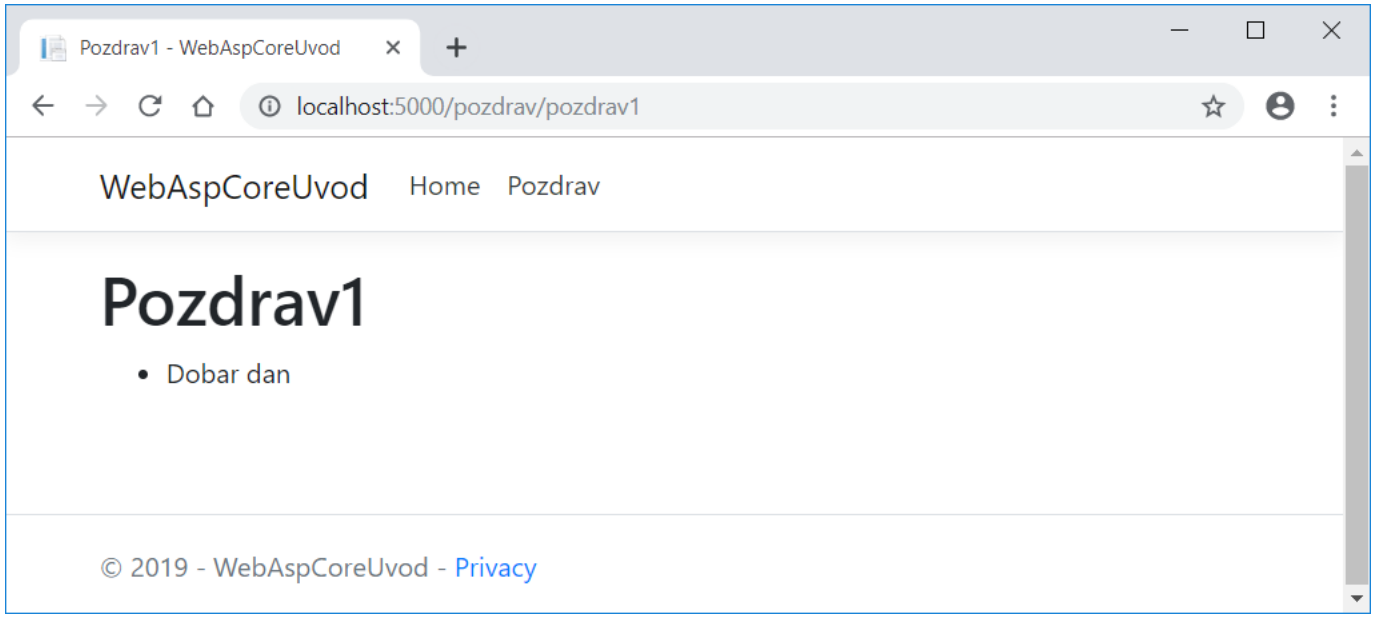
# Kreiranje pogleda Pozdrav1



# Pogled Pozdrav1.cshtml

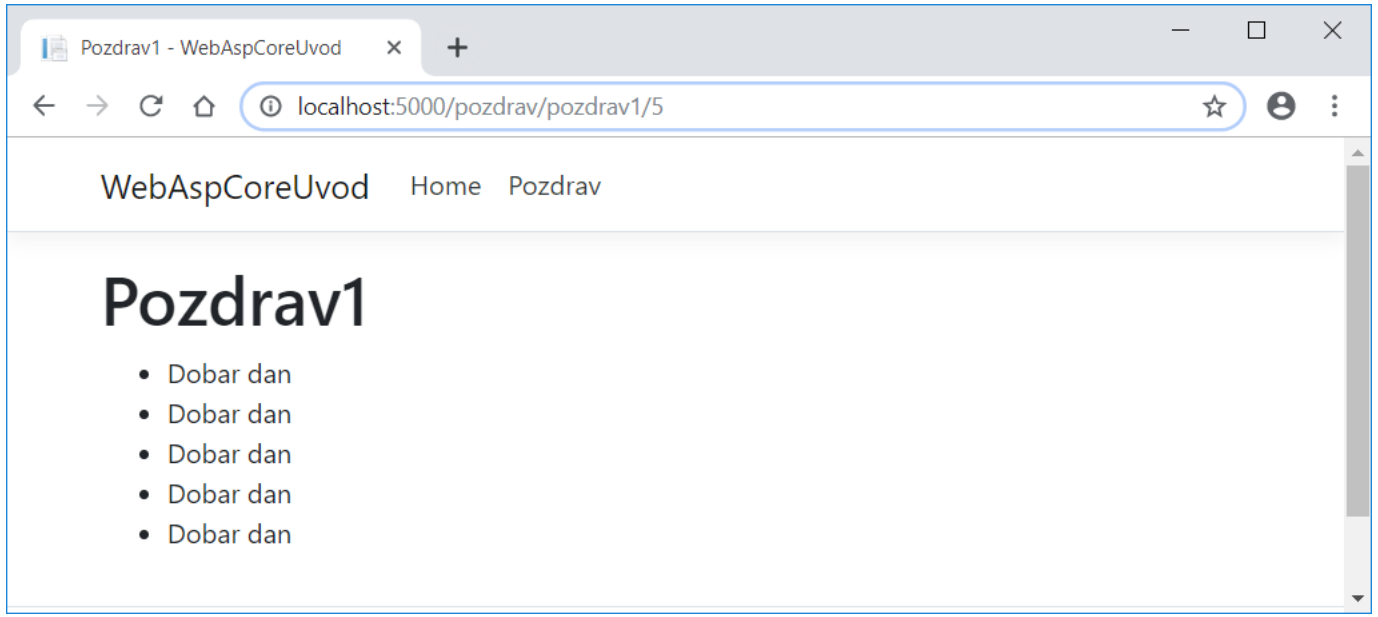
```
@{
    ViewData["Title"] = "Pozdrav1";
}

<h1>Pozdrav1</h1>

<div>
    <ul>
        @for (int i = 0; i < ViewBag.BrojPonavljanja; i++)
        {
            <li>@ViewBag.Poruka</li>
        }
    </ul>
</div>
```

# Poziv Pozdrav1 akcione metode

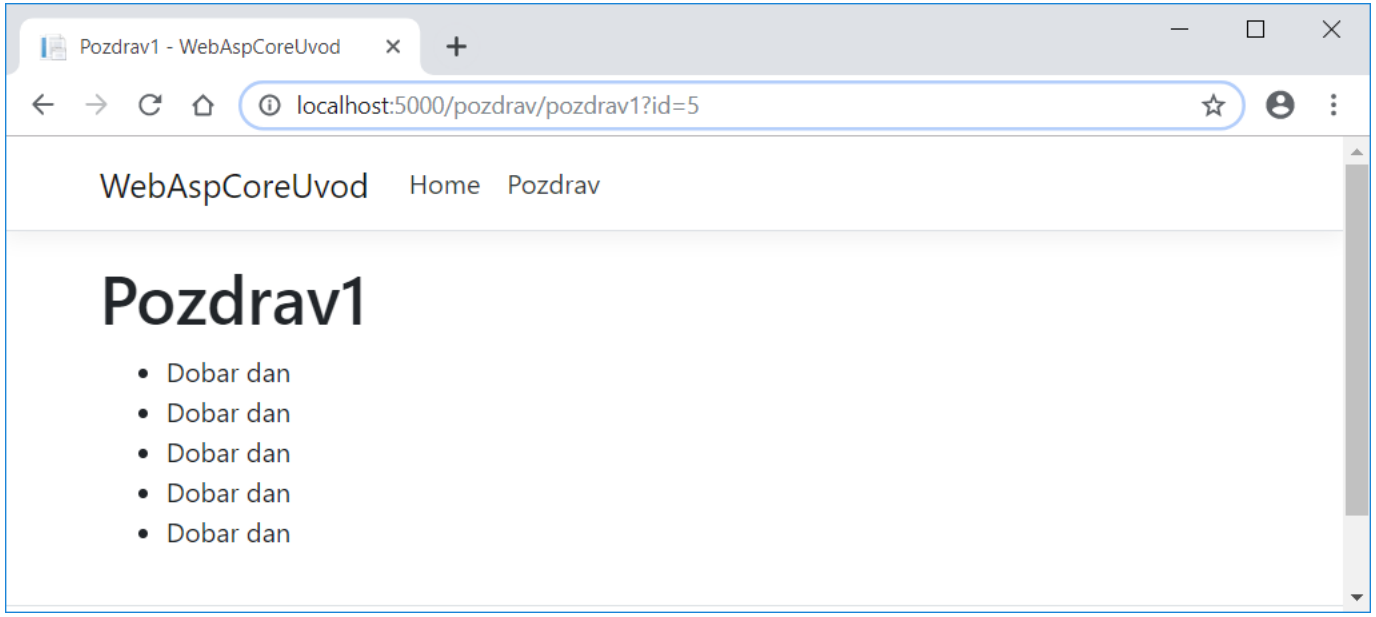


# Prosledjivanje parametara akcionoj metodi-1

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



# Korišćenje query stringa




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

# 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 6 aplikacija, vrši se unutar

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

Odgovor: a

# Pitanje 3

Protočna obrada HTTP zahteva se kod ASP.NET Core 7  web aplikacija se konfiguriše 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 podfoldera sledećeg foldera:

a. Models
b. Data
c. wwwroot

Odgovor: c