Razvoj Web Aplikacija

Predavanje 8

Danas

- Što je MVC?
- Razlika: MVC i Web API
- MVC projekt
- Kontroleri, akcije, modeli, vezanje modela (engl. Model binding)
- Pogledi (engl. Views), HTML i jezik Razor

MVC: basic concepts

MVC – što je to?

MVC (engl. Model-View-Controller) podrazumijeva dvije stvari:

- **Obrazac** (pattern) razvoja aplikacija bez obzira na tehnologiju
  - Razvijen krajem 70-ih godina prošlog stoljeća u Xeroxu
  - Ideja: logika aplikacije odvojena je od korisničkog sučelja i podataka (separation of concerns)

- **Platforma** za razvoj ASP.NET web aplikacija temeljen na MVC arhitekturi
  - Prva dostupna verzija objavljena je krajem 2007.

ASP.NET Core MVC - pregled

- ASP.NET Core MVC je moderni **web framework** za izradu web aplikacija korištenjem **Model-View-Controller** (MVC) arhitekturalnog obrasca (pattern)
- Komponente: *kontroleri*, *akcije*, *modeli, atributi, middleware*, ***pogled (view)***
- Koncepti: *usmjeravanje*, *validacija*, *sigurnost*
- Osim pogleda, trebali smo naučiti sve iz Web API
- Neki novi koncepti: *Razor sintaksa*, *HTML pomagači (helpers)*
- Ideja ovog predavanja je da dobijete razumijevanje kako ASP.NET Core MVC funkcionira općenito, a posebno nadovezujući se na web API znanje

ASP.NET MVC vs Web API

MVC i **Web API** dva su frameworka web aplikacija koja ASP.NET Core nudi

Što je slično?

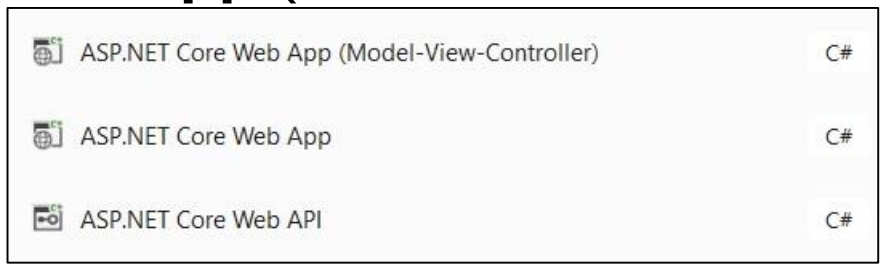
- Oba pružaju bogat skup svojstava za izgradnju web aplikacija
- Oba koriste kontrolere, akcije i modele
- Oba koriste iste mehanizme usmjeravanja i middlewarea
- Oba podržavaju ubacivanje zavisnosti (DI, dependency injection)

Što je drugačije za MVC?

- MVC koristimo za izradu **web aplikacija** koje imaju UI, dok Web API koristimo za izradu **RESTful API** (npr. web servisi) gdje UI nije u fokusu
  - MVC ➔ UI + podaci
  - Web API ➔ podaci

Postavljanje projekta

- Naš stari prijatelj Visual Studio 2022
- Koristimo projektni predložak ASP.NET Code Web **App** (Model-View-Controller)
  - NE API ovaj puta ☺



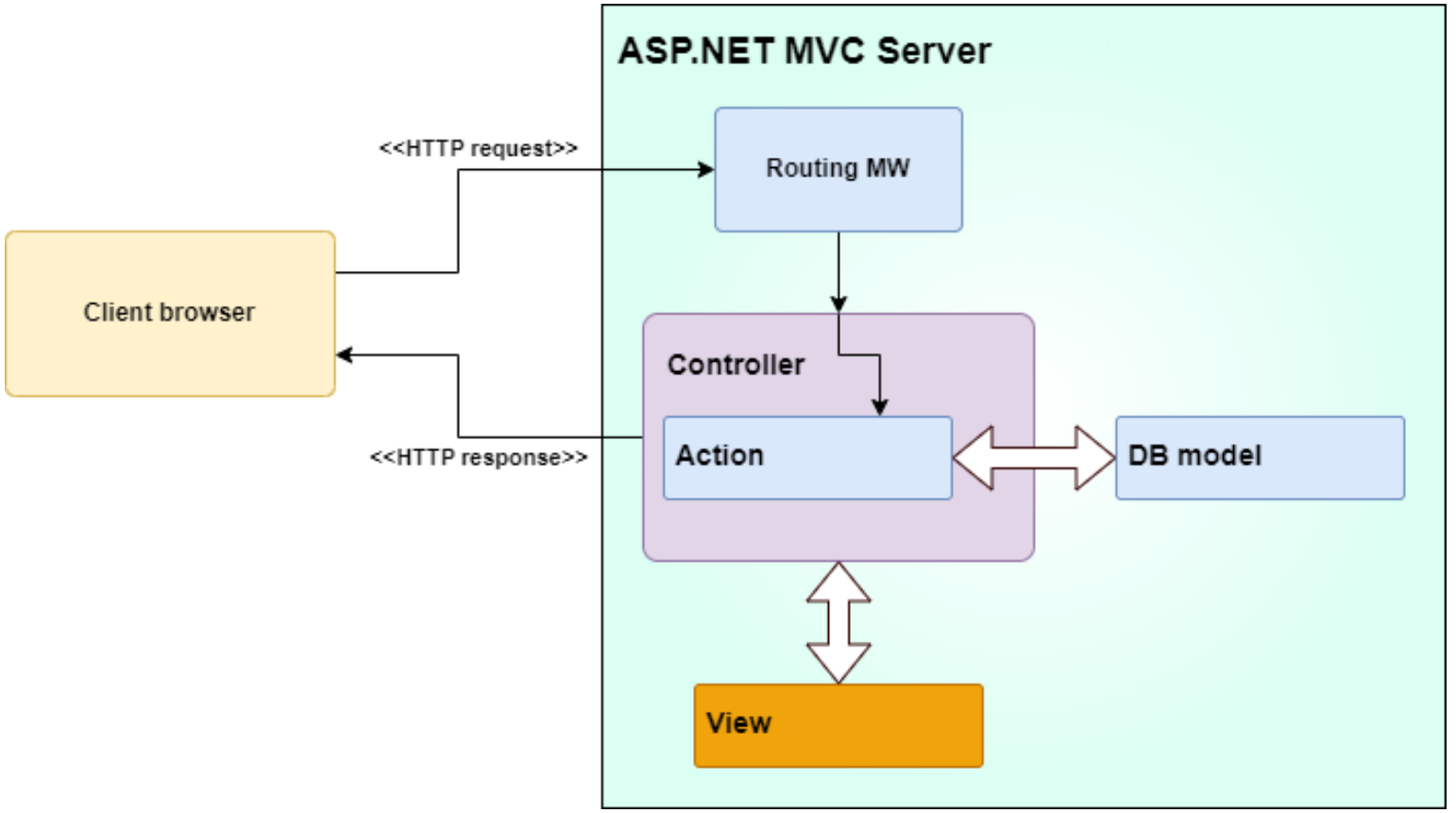
- Dodajemo kontrolere, akcije, modele i **poglede (views)**
  - Korištenje vlastite zasebne strukture mapa za poglede, s mapom Views kao korijenskom
- Upravljanje konfiguracijom projekta u **launchSettings.json**
- Dodavanje servisa (DI podešavanje) i middleware-a u **Program.cs**
- Koristimo konvencionalno usmjeravanje ili usmjeravanje zasnovano na atributima

Postavljanje projekta

- DEMO

Šira slika...

1. Primljen HTTP zahtjev od klijenta
2. Zahtjev je preusmjeren kontroleru
3. Zahtjev je usmjeren na akciju
4. Model binding (*viewmodel*)
5. Model baze podataka je dohvaćen ili promijenjen
6. Pogled (view) je odabran
7. HTTP odgovor poslan klijentu



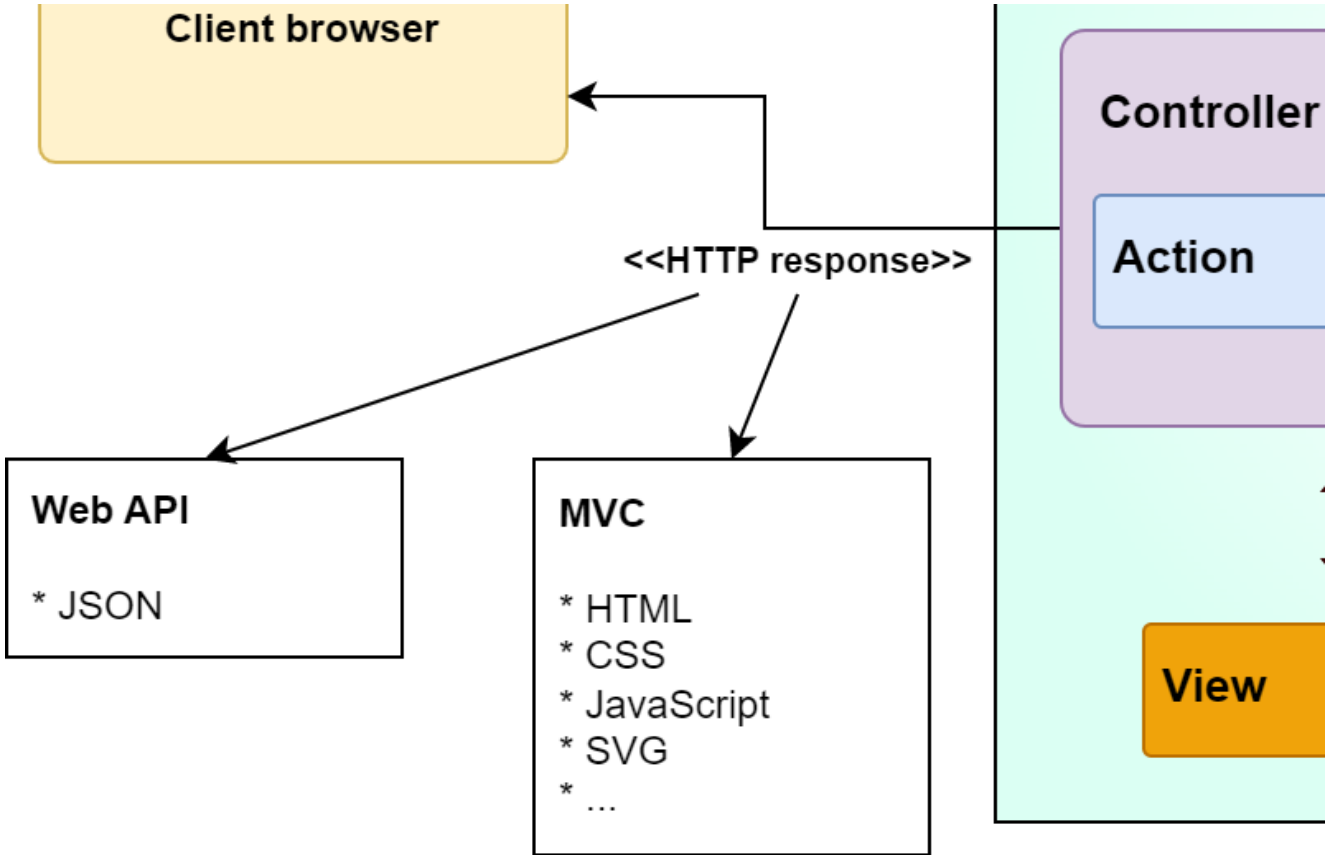
I velika razlika...

Web API i MVC odgovori općenito se razlikuju prema formatu odgovora.

Web API vraća JSON.

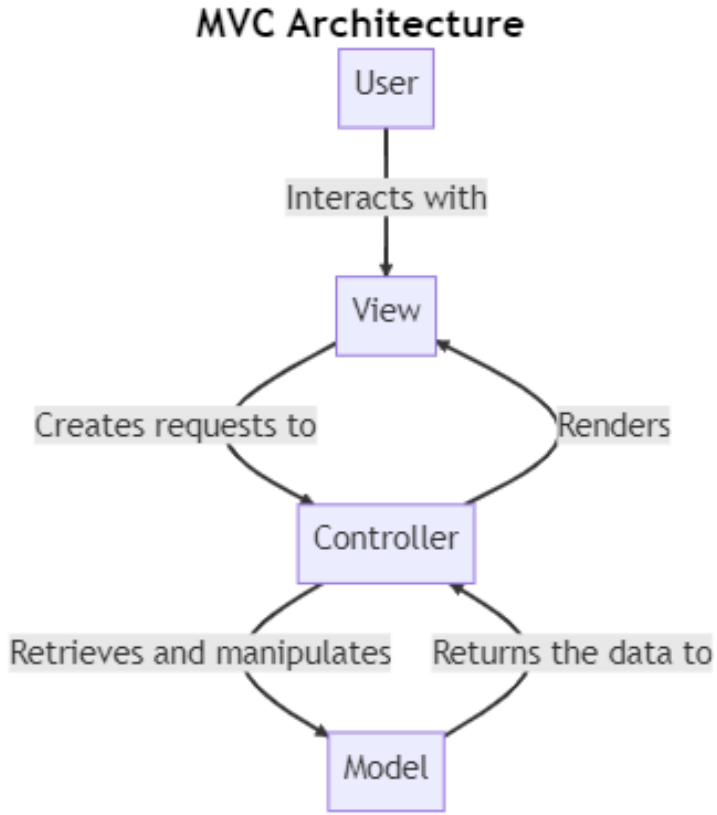
MVC vraća HTML.

Ovo su smjernice ➔ *MVC kontroler može vratiti svakakve vrste podataka, uključujući JSON!*



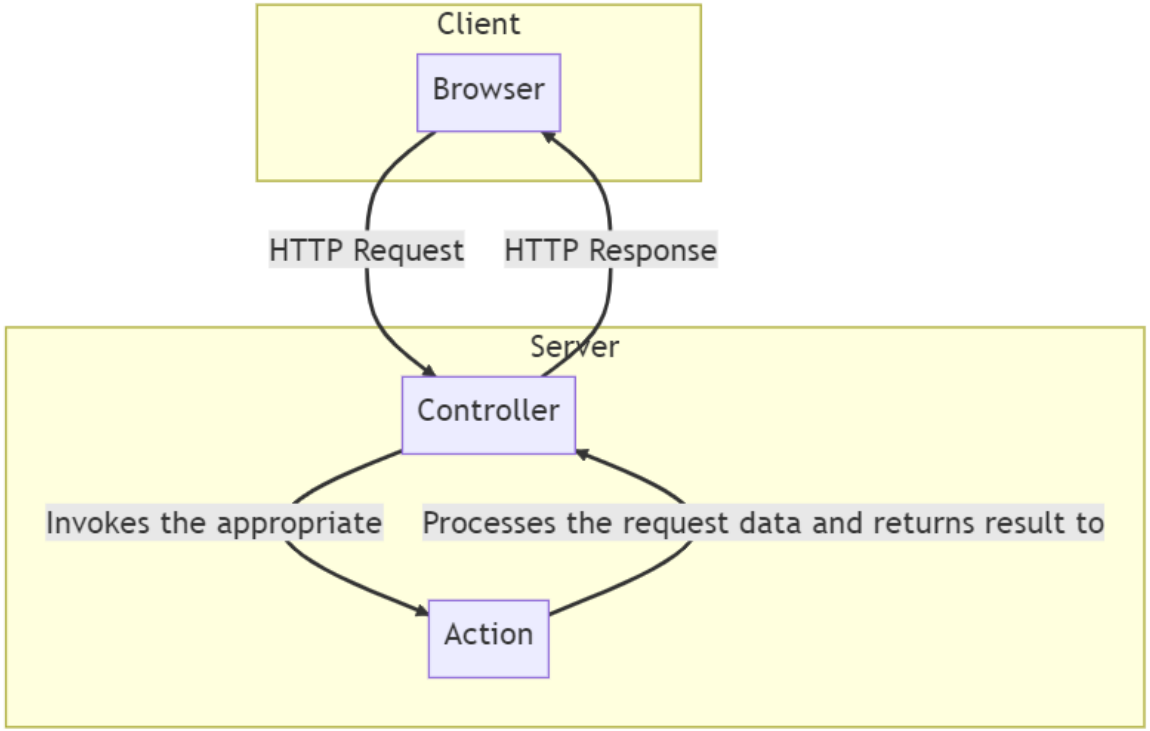
Kontroleri

- Rukuju dolaznim HTTP zahtjevima i stvaraju HTTP odgovor
- Zaprimaju HTTP zahtjev
  - Kontrolira obradu HTTP zahtjeva pozivanjem odgovarajuće akcije
  - Akcija vraća HTTP odgovor
  - *Rukovanje greškama*
- Posrednici između modela (podataka) i pogleda/viewova (prikazi podataka)
- Implementiran kao klasa koja nasljeđuje **Controller** klasu (specializacija **ControllerBase**, one koju koristi Web API)
- **Akcije** sadrže logiku specifičnu za rutu
- Ciljamo **tanki kontroler** (koristite usluge za složene logike... ili općenito, bilo kakve logike ☺)



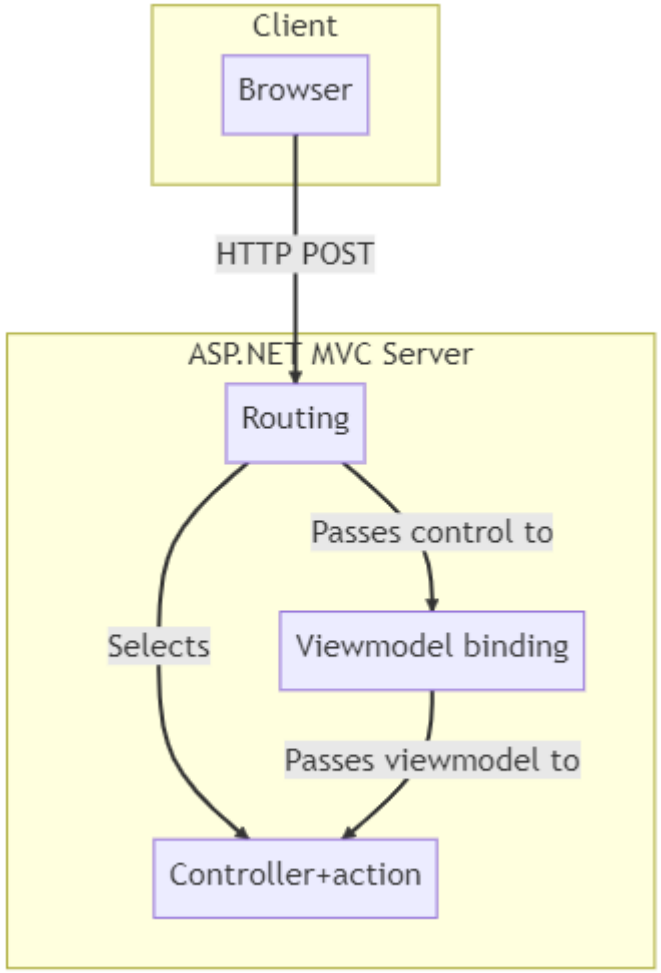
Akcije

- Metode kontrolera, koje se koriste za rukovanje dolaznim HTTP zahtjevima i vraćanje HTTP odgovora
- Mogu se nazivati po HTTP metodi (npr., **Get**, **Post**, **Put**, **Delete**)
- Akcije mogu prihvatiti ulazne parametre koji predstavljaju podatke povezane s HTTP zahtjevom, kao što su **query string**, **podaci obrasca (form)**, **podaci u tijelu zahtjeva, unutar URL-a** ili **unutar zaglavlja**
- Akcije automatski poziva ASP.NET Core MVC framework na temelju naziva metode akcije (*koncept "najboljeg kandidata" i dalje radi*) i podudaranja između HTTP glagola (verb) dolaznog zahtjeva i atributa metode
- Tip rezultata kojeg akcija vraća određen je tipom povratne vrijednosti akcije. Na primjer, akcija koja vraća **pogled (view)** obično će vratiti implementaciju sučelja **IActionResult**



Model binding

- I ASP.NET Core MVC i ASP.NET Core Web API koriste istu infrastrukturu za binding/vezanje modela
  - Ista validacija podataka
  - Ista konverzija tipa
  - Isto rukovanje greškama
  - Konvencija imenovanja
- Model koji dolazi od klijenta obično se naziva **viewmodel**
  - Model koji je stigao iz prikaza koji je prikazan klijentu

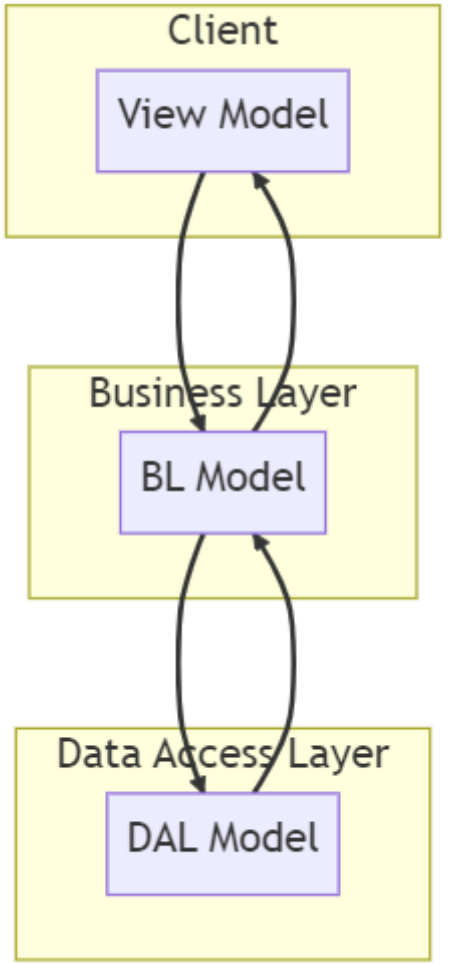


Kontroleri / akcije / model binding

- DEMO

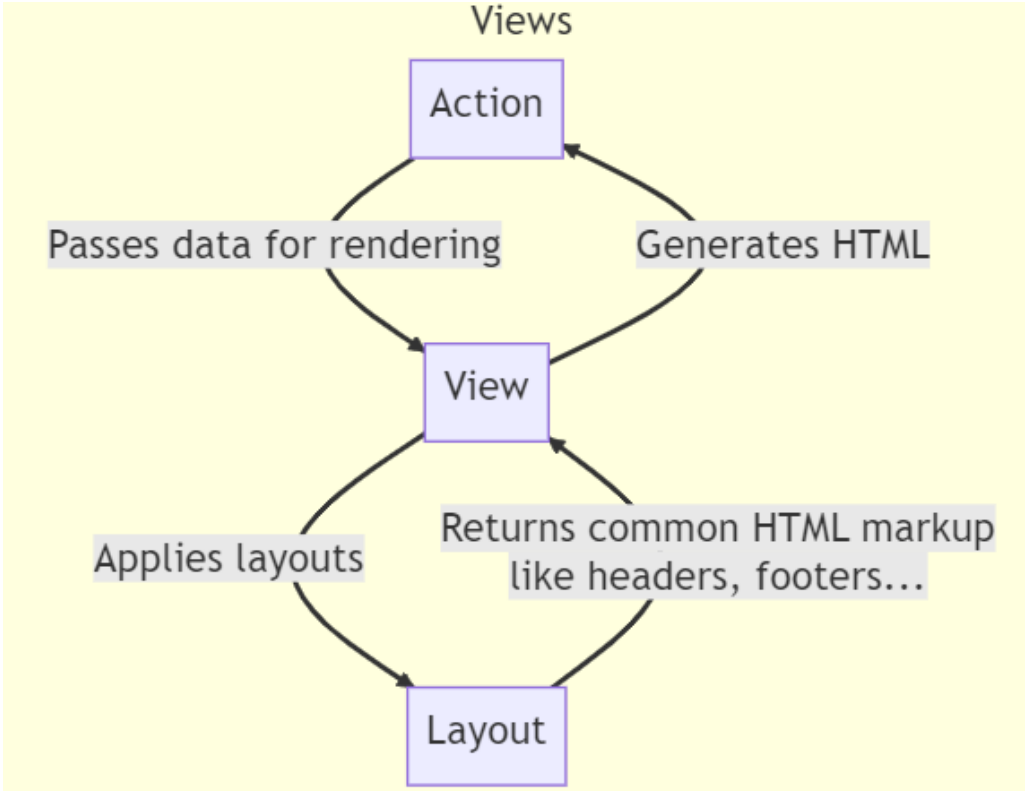
Modeli

- U troslojnoj arhitekturi koristimo nekoliko različitih vrsta modela
- U Web API pokrili smo modele za **Poslovni sloj** i **Sloj pristupa podacima**
- U MVC dodajemo Sloj prikaza i odgovarajući **View Model** (ili viewmodel)
  - Klijent: **View Model**
  - Poslovni sloj: **BL Model**
  - Sloj pristupa podacima: **DAL Model**
- Konverzije između modela nisu neuobičajene, ali nisu svi podaci potrebni u svakom modelu
  - Neki podaci nisu pohranjeni u bazi podataka
  - Neki se podaci ne šalju u korisničko sučelje



Pogledi (views)

- Pogledi (views) renderiraju **HTML**, i HTTP odgovor zatim sadrži taj HTML
- Stvoreni pomoću **Razor** markup jezika
  - Razor markup *stvara* HTML, ali Razor markup *se nikada ne vraća* klijentu
- Usko povezan s akcijama
  - konvencija: jedan pogled za jednu akciju
- Pogledi mogu pristupiti podacima koje pružaju kontroleri kroz mehanizme kao što je **model binding**
- Također mogu pristupiti podacima na "loosely typed" način, koristeći **ViewData** riječnik i/ili **ViewBag** wrapper za ViewData
- Pogledi (views) mogu biti prilagodeni primjenom **izgleda (layout)**, koji pruža dosljedan "look and feel" u raznim prikazima



Razor

- Mehanizam za predloške (templating engine)
- Koristi HTML i C# kod u istoj datoteci
- @username - postavlja sadržaj varijable u HTML
  - Ako imate oznaku @ npr. za email, morate koristiti "escape" koristeći @@
- @{ … } – blok koda

```
@{ string message = "some message"; }
```

- @(…) – eksplicitni izraz

```
<p>Last week this time: @(DateTime.Now - TimeSpan.FromDays(7))</p>
```

- @Html.Raw(…) – sirovi HTML

```
@Html.Raw("<span>Hello World</span>")
```

Razor (nastavak)

Višestruki blokovi koda

```
@{
    var quote = "The future depends on what you do today. - Mahatma Gandhi";
}

<p>@quote</p>

@{
    quote = "Hate cannot drive out hate, only love can do that. - Martin Luther King, Jr.";
}

<p>@quote</p>
```

Razor (nastavak)

Zadani jezik (C#) i prijelaz jezika (u HTML)

```
@{
    var inCSharp = true;
    <p>Now in HTML, was in C# @inCSharp</p>
}
```

Eksplicitni prijelaz na HTML pomoću <text>

```
@for (var i = 0; i < people.Length; i++)
{
    var person = people[i];
    <text>Name: @person.Name</text>
}
```

Razor (nastavak)

Kontrolne strukture

```
@if (value % 2 == 0)
{
    <p>The value was even.</p>
}


@foreach (var person in people)
{
    <p>Name: @person.Name</p>
    <p>Age: @person.Age</p>
}
```

Razor (nastavak)

Using blok

```
@using (Html.BeginForm())
{
    <div>
        Email: <input type="email" id="Email" value="">
        <button>Register</button>
    </div>
}
```

Razor

- DEMO

Hvala na pažnji!