Razvoj Web Aplikacija

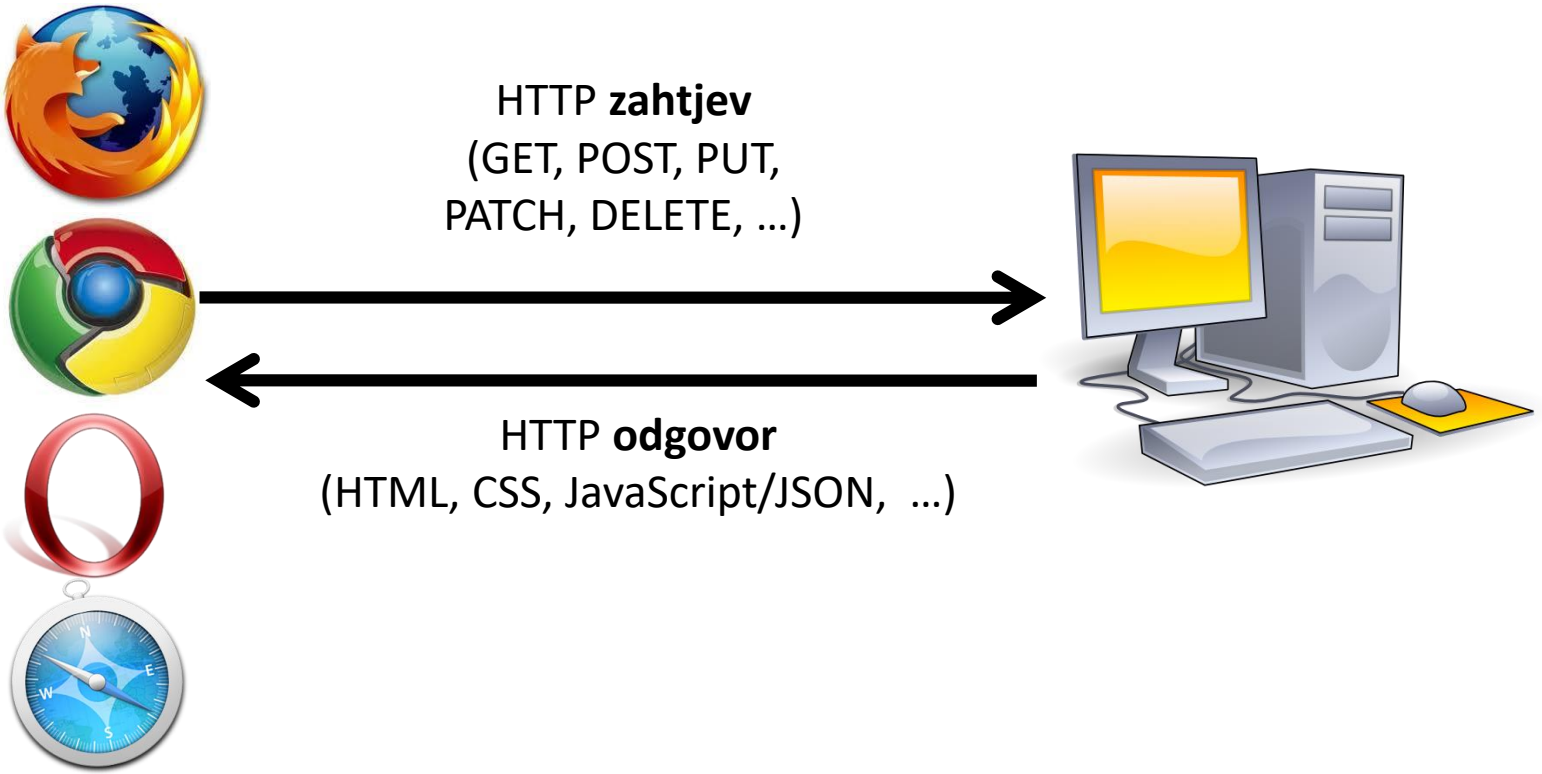
Predavanje 1

- Paradigma klijent – server
- URL
- HTTP protokol
- HTTP zahtjev / odgovor
- Uvod u Web API

Uvod u Web aplikacije

- Web aplikacija = web site
- Web aplikacijom smatramo svaku aplikaciju kojoj pristupamo **preglednikom** (engl. *browser*)
- Web aplikacija se sastoji od dva osnovna dijela:
  - Serverski dio
    - Izvršava se na serveru:
      - ASP.NET
  - Klijentski dio
    - Izvršava se na klijentu, u pregledniku
    - Najčešće se sastoji od HTML-a, CSS-a i JavaScript-a

Komunikacija klijent - server



URL – Uniform Resource Locator

- Predstavlja resurs na webu
- http://moja-domena.com
- Različiti resursi unutar iste web stranice
- http://moja-domena.com/artikli
- http://moja-domena.com/projekti

URL segmenti

- Karakteristični dijelovi adrese http://moja-domena.com/artikli
  - http – URL shema
    - URL se koristi za različite tipove protokola: http, ftp, mailto…
  - moja-domena.com – računalo (*host)* na kojem se nalazi resurs
    - Računalo koristi DNS (DomainNameSystem) da pronađe mrežnu IP adresu računala na koje se šalje zahtjev (http://moja-domena.com – 201.192.21.13)
  - /artikli – URL putanja
    - Predstavlja konkretan resurs koji se potražuje, često je hijerarhijske strukture

URL segmenti

- http://moja-domena.com**:80**/artikli
- Predefinirani port na kojem se traži resurs je 80 koji se radi jednostavnosti izbacuje iz adrese
- http://moja-domena.com:80/artikli**?id=21&kategorija=13**
- Nakon znaka **?** slijede parametri u obliku `ključ=vrijednost` koji se nižu pomoću znaka **&**
- http://moja-domena.com:80/artikli?id=21&kategorija=13**#opis**
- Fragment koristi samo klijent (ne i poslužitelj) koji fragment stavlja na vrh stranice

URL enkodiranje

- U adresnoj traci mogu se koristiti samo sigurni znakovi (ASCII tablica)
  - Znakovi tipa **#** ili razmak ne spadaju u skupinu sigurnih znakova
- Zabranjeni znakovi trebaju biti URL enkodirani
  - Razmak - **%20** (20 je heksadecimalna vrijednost dekadske vrijednosti (32) oznake razmaka iz ACSII tablice)
  - # - %23
  - $ - %24
  - % - %25
  - ...

Komunikacija – HTTP protokol

- HTTP 1.1 specifikacija definira jezik koji razumiju i klijenti i poslužitelji
- Http zahtjev i http odgovor se odvija unutar jedne http transakcije
- Svaki zahtjev mora imati specificiranu metodu zahtjeva (predefinirana: GET)

```
GET / slika.jpeg HTTP/1.1
host: moja-domena.com
```

- Najčešće metode:
  - **GET** - dohvaćanje
  - **POST** - ažuriranje
  - **PUT** - pohranjivanje
  - **DELETE** – brisanje

Zahtjev (request)

```
[Method] [URL] [version]

[headers]

[body]
```

```
GET maps?q=restorani HTTP/1.1

Host: maps.google.com
```

- Sva zaglavlja osim Host su opcionalna
- Poslužitelj može posluživati više web stranica

Odgovor (response)

[version] [status] [reason]

[headers]

[body]

```
HTTP/1.1 200 OK
Content-type: text/html; charset=utf-8
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Content-Length: 211310
<html>…</html>
```

Content type

- Predstavlja tip resursa koji se potražuje
- Pretraživači ih koriste da znaju koji tip podataka trebaju procesuirati (ne gledaju na ekstenziju datoteke)
  - **`application/json`** – JSON podaci
  - **`image/png`** – PNG slika
  - **`image/gif`** – GIF slika
  - **`text/html`** – HTML datoteka
  - **`text/plain`** – tekstualna datoteka
  - **`video/mp4`** – MP4 datoteka

Status code - kategorije

- Govori klijentu kakav je rezultat zahtjeva
- 100-199 – informacijski
- 200-299 – uspješni
- 300-309 – preusmjeravanje
  - 302 – moved temporary
- 400-499 – klijentska greška
  - 400 – bad request
  - 401 - unauthorized
- 500-599 – Serverska greška
  - 500 – internal server error

Primjer komunikacije

- HTML stranica sa CSS-om i 2 slike.

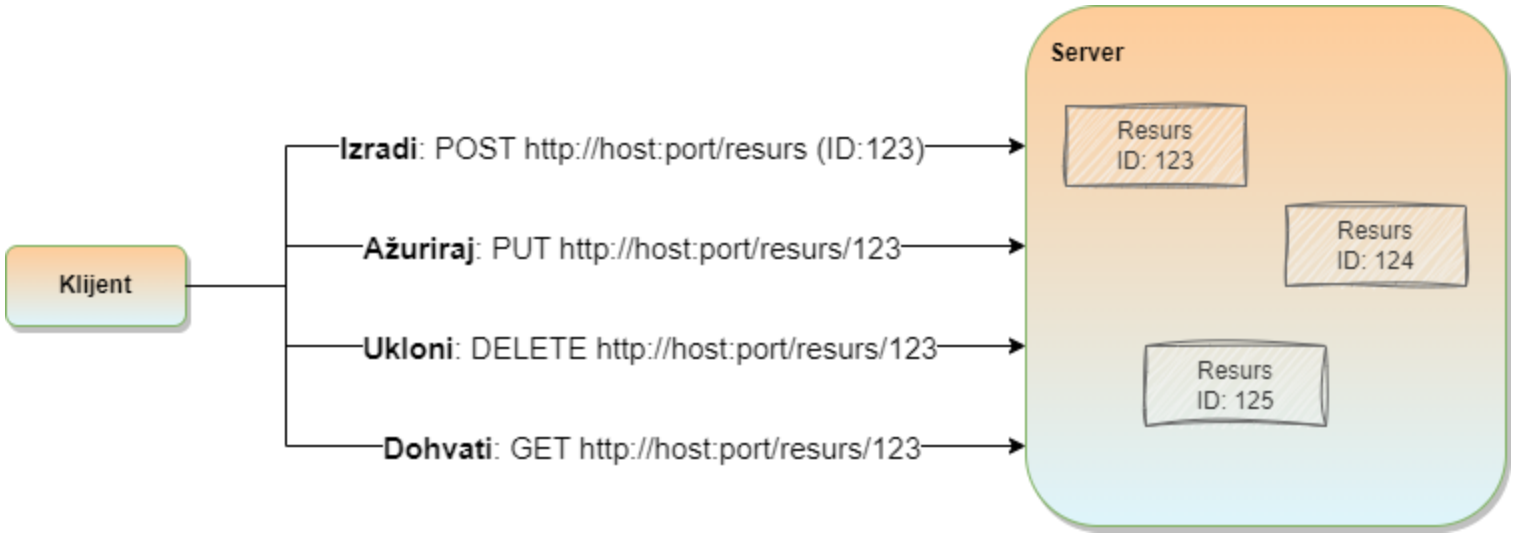
1. zahtjev: GET **Stranica1.htm** Odgovor: **200** OK Content-Type: **text/html** Content-Length: 747 <HTML> ... <HTML>	3. zahtjev: GET **Slika1.png** Odgovor: **200** OK Content-Type: **image/x-png** Content-Length: 49731 ...
2. zahtjev: GET **MyWebSite.css** Odgovor: **200** OK Content-Type: **text/css** Content-Length: 100 p { ... }	4. zahtjev: GET **Slika2.jpg** Odgovor: **200** OK Content-Type: **image/jpeg** Content-Length: 41514 ...

Uvod u Web API

- **Web API ➔** to je API putem weba ☺
- **API** = Application Programming Interface predstavlja sredstvo programatske komunikacije s entitetom
  - Programatski – podrazumijeva dobro definiranu strukturu komunikacije (kao što jezik ima dobro definiranu strukturu), zajedničko sučelje za sudionike u komunikaciji
- **REST ➔ arhitekturalni stil koji se koristi za izgradnju Web API rješenja**
  - Izraz representational state transfer (REST) je uveo i definirao Roy Fielding 2000. godine u svojoj doktorskoj dizertaciji
  - Osim zajedničkog sučelja, pretpostavlja model klijent-poslužitelj i odsustvo stanja (statelessness), što su dobro poznate osobine HTTP-a
  - Web API može komunicirati npr. putem SOAP-a, ali ako koristi REST, onda impliciramo da koristi HTTP protokol
  - REST na Web API nameće korištenje ograničenog skupa standardnih operacija: GET, POST, PUT, PATCH, DELETE
  - Kada Web API koristi REST, može se nazvati se naziva RESTful servisom, i upravo s tim ćemo raditi

Web API i resursi

- **Resource** ➔ podatak koji ima identifikator (ID)
- Primjeri: JSON podaci (npr. trenutna temperatura u Zagrebu), slika, video…
- Tip resursa je moguće prepoznati po MIME tipu (MIME type): text, image, audio, video, font…
- REST: metode CREATE, RETRIEVE, UPDATE, DELETE ➔ "CRUD" opisuje potpunu RESTful implementaciju za resurs



ASP.Net Web API

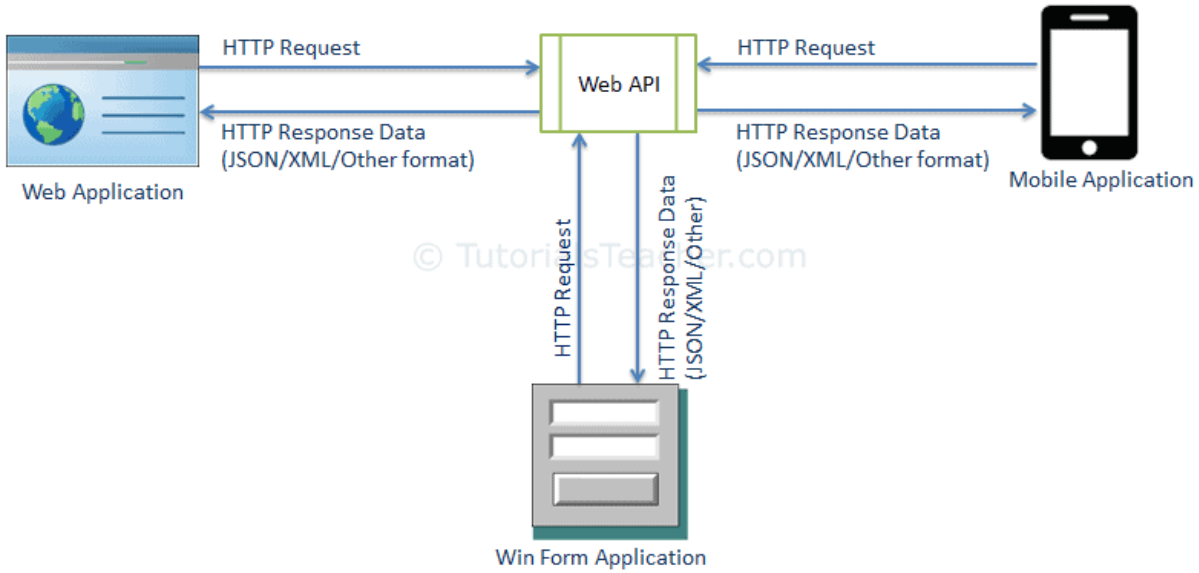
- ASP.NET tehnologije za izradu web aplikacija:
  - Web Forms
  - MVC
  - **Web API** (također poznat i kao REST API)
  - **…**
- Za Web API razvoj ćemo koristiti **Visual Studio**
- Za određene vježbe trebat će nam i **SQL Server**
- Ishodišni element pri radu s Visual Studiom je **Solution**
- *Solution* se sastoji od jednog ili više **projekata**
- Napomena: postoje **controller-based API** i **minimal API**; mi ćemo koristiti controller-based API

Posebne mape i datoteke

- **`Connected services`** – skup alata za spajanje projekta na neke od podržanih vanjskih servisa
  - npr. Azure servise, OpenAPI krajnje točke, gRPC krajnje točke, WCF krajnje točke, razne baze podataka
- **`Dependencies`** – skup biblioteka o kojima projekt zavisi
  - biblioteke su grupirane prema vrsti (*Frameworks, Packages, Analyzers*)
  - kažemo da su navedene biblioteke *instalirane* u projekt
- **`Properties`**
  - Sadrži **launchSettings.json** datoteku – **postavke** (profile) pokretanja projekta
  - Naziv profila je vidljiv na gumbu za startanje aplikacije u debug načinu rada, npr. *http*
- **`Controllers`**
  - Sadrži datoteke koje u sebi imaju programsku logiku koja se pokreće na zahtjev npr. korisničkog sučelja
- **`appsettings.json`** – datoteka koja sadrži konfiguraciju projekta (prije: web.config)
- **`Program.cs`** – ulazna točka projekta

Web API struktura aplikacije

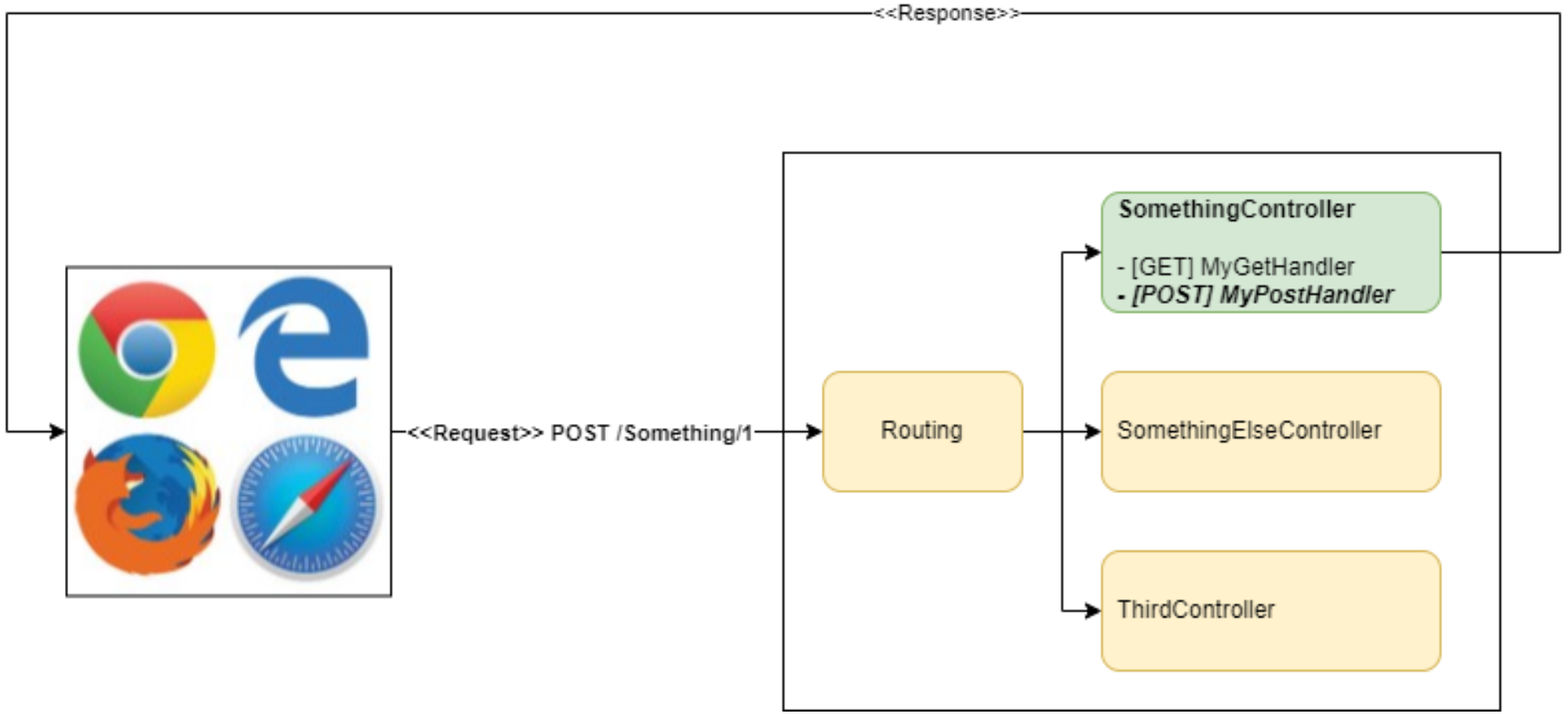
- Web API aplikacija se sastoji od:
  - Web API krajnjih točaka (endpoint) iza kojih se nalazi programska logika
  - Korisničkog sučelja (HTML stranice, Windows forme, mobilne i druge aplikacije)



Princip rada Web API rješenja

- Kod se prevodi i aplikacija starta
- Program može biti pokrenut kao samostalan HTTP poslužitelj ili pokrenut na postojećem HTTP poslužitelju (*IIS*, *IIS express*)
- HTTP poslužitelj prilikom starta detektira posebne klase, tzv. kontrolere, koji služe za obradu HTTP zahtjeva i vraćanje HTTP odgovora
- Kada na server na dođe zahtjev, poslužitelj koristi tzv. "routing" kao metodu prepoznavanja kojoj klasi i kojoj metodi proslijediti taj zahtjev
- Kreira se instanca klase i na tom objektu se poziva odgovarajuća metoda
- Nakon slanja odgovora klijentu objekt se uništava
- Svaki novi zahtjev izaziva kreiranje novog objekta

Princip rada Web API rješenja



Struktura kontrolera

- *Kontroler* datoteku čini klasa koja nasljeđuje `Microsoft.AspNetCore.Mvc`
- Mi programiramo kod unutar *metoda* kontroler klase
- Na kraju, kontroler vraća podatke koji će biti serijalizirani u JSON oblik i poslani klijentu

```
[HttpGet]
public IEnumerable<WeatherForecast> Get()
{
    var range = Enumerable.Range(1, 5);
    var forecasts = range.Select(index => new WeatherForecast
    {
        Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
        TemperatureC = Random.Shared.Next(-20, 55),
        Summary = Summaries[Random.Shared.Next(Summaries.Length)]
    });
    var forecastsArr = forecasts.ToArray();

    return forecastsArr;
}
```

Pokretanje web aplikacije

- Postoji više načina kako možemo pokrenuti web aplikaciju:
  - Otvaranjem preglednika i odlaskom na neku adresu aplikacije
  - **`Debug -> Start Without Debugging (Ctrl + F5)`** će pokrenuti preglednik
    - Najprije se radi prevođenje
  - **`Debug -> Start Debugging (F5)`** će pokrenuti preglednik i Visual Studio prikačiti na web server
    - Najprije se radi prevođenje
    - Visual Studio se nalazi u *debug* načinu rada

Hvala na pažnji!