Web aplikacije (WA)

Nositelj: doc. dr. sc. Nikola Tanković
Asistent: Luka Blašković, mag. inf.

Ustanova: Sveučilište Jurja Dobrile u Puli, Fakultet informatike u Puli

(2) Usmjeravanje na Express poslužitelju

Usmjeravanje odnosno *routing* se odnosi na određivanje kako će krajnje rute koje definiramo na našoj poslužiteljskoj strani odgovarati na dolazne zahtjeve klijenata. U prošloj skripti smo već definirali osnovni primjer usmjeravanja za nekoliko GET ruta i posluživali smo statične datoteke i jednostavne JSON objekte. Danas ćete naučiti kako definirati složenije usmjeravanje kroz sve HTTP metode, koja su pravila usmjeravanja i dodatni parametri koje koristimo.

🆙 **Posljednje ažurirano: 28.10.2024.**

- skripta nije dovršena

Sadržaj

1. Ponavljanje

Nastavljamo s radom na Express poslužitelju, na ovim ćemo vježbama detaljnije proučiti **usmjeravanje** i **obradu zahtjeva** u Express aplikacijama.

Usmjeravanje (*eng. routing*) se odnosi na određivanje kako će krajnje rute koje definiramo na našoj poslužiteljskoj strani odgovarati na dolazne zahtjeve klijenata. U prošloj skripti smo već definirali osnovni primjer usmjeravanja za nekoliko ruta i posluživali smo statične datoteke i JSON objekte.

Osnovna sintaksa za definiranje ruta u Express aplikacijama je sljedeća:

```
app.METHOD(PATH, HANDLER);
```

gdje je:

- `app` je instanca Express aplikacije
- `METHOD` je HTTP metoda (npr. GET, POST, PUT, DELETE, itd.) koju želimo posluživati
- `PATH` je putanja na koju želimo reagirati (npr. `/`, `/about`, `/contact`, itd.)
- `HANDLER` je callback funkcija koja se izvršava kada se zahtjev podudara s definiranom rutom

Tako smo definirali rutu za početnu stranicu:

```
app.get('/', function (req, res) {
  res.send('Hello, world!');
});

// odnosno

app.get('/', (req, res) => {
  res.send('Hello, world!');
});
```

`PATH` koji smo ovdje koristili je `/`, što znači da će se ova ruta pokrenuti kada korisnik posjeti početnu stranicu našeg web sjedišta.

U ovom primjeru koristili smo `GET` metodu, za koju smo općenito rekli da se koristi kada korisnik želi dohvatiti neki resurs s poslužitelja, bio on HTML dokument, slika, CSS datoteka, JavaScript datoteka, JSON objekt, itd.

2. Osnovno usmjeravanje

2.1 `GET` metoda i parametri

U prošloj smo skripti već naučili kako koristiti `GET` metodu za dohvat resursa s poslužitelja. U ovom ćemo primjeru proširiti našu aplikaciju tako da možemo dohvatiti resurs s poslužitelja na temelju **parametara** koje korisnik prenosi u URL-u.

Osnovna sintaksa za definiranje GET rute je sljedeća:

```
app.get(PATH, (req, res) => {
  // Ovdje pišemo kod koji će se izvršiti kada korisnik posjeti PATH
});
```

Primjerice, zamislimo da radimo **aplikaciju za naručivanje pizze** 🍕. Recimo da korisnik odluči pogledati koje su pizze dostupne, želimo da dohvati sve dostupne pizze definirane na našem poslužitelju. U tom slučaju, korisnik bi mogao posjetiti URL `/pizze`.

```
app.get('/pizze', (req, res) => {
  res.send('Ovdje su sve dostupne pizze!');
});
```

Rekli smo da možemo koristiti metodu `res.json` kako bismo poslali JSON objekt korisniku. U ovom slučaju, možemo poslati listu dostupnih pizza kao JSON objekt:

No prvo moramo definirati listu dostupnih pizza:

```
const pizze = [
  { id: 1, naziv: 'Margherita', cijena: 6.5 },
  { id: 2, naziv: 'Capricciosa', cijena: 8.0 },
  { id: 3, naziv: 'Quattro formaggi', cijena: 10.0 },
  { id: 4, naziv: 'Šunka sir', cijena: 7.0 },
  { id: 5, naziv: 'Vegetariana', cijena: 9.0 }
];

app.get('/pizze', (req, res) => {
  res.json(pizze);
});
```

Kada korisnik posjeti URL `/pizze`, dobit će JSON objekt s listom dostupnih pizza. Ako nemate instaliranu jednu od ekstenzija za web preglednik koje omogućuju pregled JSON objekata u pregledniku, JSON će vam se prikazivati kao običan tekst (*eng. raw*) bez formatiranja, što može biti nepregledno. Preporuka je preuzeti jednu od JSON Formatter ekstenzija za preglednik, npr. JSON Formatter za Chromium preglednike.

Što ako korisnik želi dohvatiti **samo jednu pizzu**, a ne sve? Kako ćemo definirati rutu za dohvat jedne pizze?

Možemo definirati posebnu rutu za svaku pizzu, npr. `/margherita`, `/capricciosa`, `/quattro-formaggi`, itd. Međutim, koliko je to rješenje pametno?

Možda bi mogli proći s ovim ako restoran ima 4-5 pizza, ili 15. Što ako restoran ima 50 pizza? ili 100?

Navedeno je primjer lošeg dizajna i nepotrebno ponavljanje koda. Umjesto toga, možemo koristiti **parametre** u URL-u kako bismo dohvatili jednu pizzu.

URL parametar je dio URL-a koji se koristi za prenošenje informacija između klijenta i poslužitelja. URL parametri se definiraju u URL-u s prefiksom `:`.

Primjerice, ako možemo definirati rutu `/pizze/:id` koja će dohvatiti pizzu s određenim `id` parametrom:

```
app.get('/pizze/:id', (req, res) => {
  res.json(pizze);
});
```

Kako bi sad dohvatili određenu pizzu, moramo poslati zahtjev u obliku `/pizze/1`, `/pizze/2`, `/pizze/3`, itd. Nećemo navoditi eksplicitno `"id"` u URL-U, već nam služi kao svojevrsni **placeholder**.

Pošaljite zahtjev na `/pizze/1` i provjerite rezultat.

Zašto nismo dobili dobili podatke samo za jednu pizzu iako smo poslali `id` parametar?

▶ Spoiler alert! Odgovor na pitanje

Idemo sada definirati logiku koja će dohvatiti samo jednu pizzu na temelju `id` parametra. Za početak, stvari možemo odraditi na "ručni" način, tj. prolaskom kroz sve dostupne pizze i pronalaskom one koja ima traženi `id`.

> `id` iz URL-a je tipa string i možemo ga jednostavno dohvatiti pomoću `req.params` objekta.

```
app.get('/pizze/:id', (req, res) => {
  const id_pizza = req.params.id; // dohvaćamo id parametar iz URL-a

  for (pizza of pizze) {
    if (pizza.id == id_pizza) {
      // ako smo pronašli podudaranje u id-u
      res.json(pizza); // vrati objekt pizze kao rezultat
    }
  }
});
```

Sada kada pošaljemo zahtjev na `/pizze/1`, dobit ćemo JSON objekt s podacima o pizzi s `id`-om 1, odnosno Margheriti.

```
curl -X GET http://localhost:3000/pizze/1
```

Rezultat:

```
{
  "id": 1,
  "naziv": "Margherita",
  "cijena": 6.5
}
```

Naš endpoint `/pizze` funkcionira i dalje i možemo ga pozvati bez parametara:

```
curl -X GET http://localhost:3000/pizze
```

Rezultat:

```
[
  {
    "id": 1,
    "naziv": "Margherita",
    "cijena": 6.5
  },
  {
    "id": 2,
    "naziv": "Capricciosa",
    "cijena": 8
  },
  {
    "id": 3,
    "naziv": "Quattro formaggi",
    "cijena": 10
  },
  {
    "id": 4,
    "naziv": "Šunka sir",
    "cijena": 7
  },
  {
    "id": 5,
    "naziv": "Vegetariana",
    "cijena": 9
  }
]
```

Kod možemo pojednostaviti korištenjem metode `find` koja će nam vratiti prvi element koji zadovoljava uvjet:

```
app.get('/pizze/:id', (req, res) => {
  const id_pizza = req.params.id; // dohvaćamo id parametar iz URL-a

  const pizza = pizze.find(pizza => pizza.id == id_pizza); // pronalazimo pizzu s traženim
id-em

  res.json(pizza);
});
```

Što ako korisnik pošalje zahtjev za pizzu koja ne postoji? Kako ćemo riješiti taj slučaj? 🤔

▶ Spoiler alert! Odgovor na pitanje

```
app.get('/pizze/:id', (req, res) => {
  const id_pizza = req.params.id; // dohvaćamo id parametar iz URL-a

  const pizza = pizze.find(pizza => pizza.id == id_pizza);

  if (pizza) {
    // ako je pronađeno podudaranje, vratimo pizza objekt
    res.json(pizza);
  } else {
    // ako je rezultat undefined, vratimo poruku da pizza ne postoji
    res.json({ message: 'Pizza s traženim ID-em ne postoji.' });
  }
});
```

Sada kada pošaljemo zahtjev na `/pizze/6`, dobit ćemo poruku da pizza s traženim ID-em ne postoji.

```
curl -X GET http://localhost:3000/pizze/6
```

Rezultat:

```
{
  "message": "Pizza s traženim ID-em ne postoji."
}
```

Što ako korisnik pošalje zahtjev na `/pizze/vegetariana`? Kako ćemo riješiti taj slučaj? 🤔

▶ Spoiler alert! Odgovor na pitanje

Možemo koristiti metodu `isNaN` (is Not a Number) kako bismo provjerili je li `id` parametar broj:

```
app.get('/pizze/:id', (req, res) => {
  const id_pizza = req.params.id;

  if (isNaN(id_pizza)) {
    // provjeravamo je li id_pizza "Not a Number"
    res.json({ message: 'Proslijedili ste parametar id koji nije broj!' });
```

```
    return;
  }

  const pizza = pizze.find(pizza => pizza.id == id_pizza);

  if (pizza) {
    res.json(pizza);
  } else {
    res.json({ message: 'Pizza s traženim ID-em ne postoji.' });
  }
});
```

2.2 `POST` metoda i slanje podataka

Do sada smo koristili `GET` metodu za dohvat resursa s poslužitelja. Sada ćemo naučiti kako koristiti `POST` metodu za slanje podataka na poslužitelj.

`POST` metoda se koristi kada korisnik želi poslati podatke na poslužitelj, npr. kada korisnik želi **izraditi novi resurs na poslužitelju**, a podaci se šalju u **tijelu zahtjeva** (*eng. request body*).

Osnovna sintaksa za definiranje POST rute je sljedeća:

```
app.post(PATH, (req, res) => {
  // Ovdje pišemo kod koji će se izvršiti kada korisnik pošalje POST zahtjev na PATH
});
```

Vratimo se na primjer aplikacije za naručivanje pizze. Zamislimo da korisnik želi **naručiti pizzu**. Kako bismo omogućili korisniku da naruči pizzu, moramo definirati POST rutu koja će omogućiti korisniku da nekako pošalje podatke o narudžbi na poslužitelj.

Idemo napisati kostur POST rute za naručivanje pizze:

```
app.post('/naruci', (req, res) => {
  // Ovdje ćemo napisati logiku za naručivanje pizze
});
```

Ako otvorite ovu rutu u pregledniku, dobit ćete poruku `"Cannot GET /naruci"`. To je zato što smo definirali POST rutu, a pokušavamo je otvoriti u pregledniku, što će automatski poslati GET zahtjev!

Možemo dodati jednostavnu poruku koja će korisniku reći da je narudžba uspješno zaprimljena:

```
app.post('/naruci', (req, res) => {
  res.send('Vaša narudžba je uspješno zaprimljena!');
});
```

Zahtjev možemo poslati kroz terminal aplikaciju `curl` koju smo koristili u prethodnim primjerima:

```
curl -X POST http://localhost:3000/naruci
```

Kako možemo poslati podatke o narudžbi kroz POST HTTP zahtjev? 🤔

Hoćemo li to raditi kroz parametre u URL-u?

```
//?
app.post('/naruci/:id', (req, res) => {
  res.send(`Zaprimio sam narudžbu za pizzu ${req.params.id}`);
});
```

▶ Spoiler alert! Odgovor na pitanje

Kako bismo poslali veličinu pizze koju želimo naručiti?

```
// ?
app.post('/naruci/:id/:velicina', (req, res) => {
  res.send(`Zaprimio sam narudžbu za ${req.params.velicina} pizza ${req.params.id}`);
});
```

Dva isječka koda iznad primjeri su jako loše prakse. Zašto?

- **URL parametri su javno vidljivi** i mogu sadržavati osjetljive informacije (kako ćemo poslati podatke o plaćanju?)
- **Kod postaje nečitljiv** i teško održiv
- **Nije skalabilno** (što ako želimo poslati još više podataka? Ili više pizza?! 🍕🍕🍕)
- **Nije standardizirano** (kako ćemo znati koji parametar odgovara kojem podatku?)

Dakle, rekli smo da podatke šaljemo u **tijelu zahtjeva** (*eng. request body*). Kako ćemo to napraviti?

U prvoj skripti smo već naučili da podaci koji se šalju u tijelu zahtjeva mogu biti u različitim formatima, npr. JSON, XML, HTML, itd. Mi ćemo u pravilu slati podatke u **JSON** formatu.

Međutim, u našem web pregledniku nemamo mogućnost slanja POST zahtjeva s tijelom zahtjeva kada direktno pristupamo URL-u neke rute poslužitelja. Možemo poslati kroz naš `curl` alat s opcijom `-d`:

```
curl -X POST http://localhost:3000/naruci -d '{"pizza": "Margherita", "velicina":
"srednja"}'
```

Kako ćemo sada u našoj Express aplikaciji dohvatiti podatke koje je korisnik poslao u tijelu zahtjeva?

> Podaci koje korisnik šalje u tijelu zahtjeva se nalaze u `req.body` objektu.

Primjer:

```
app.post('/naruci', (req, res) => {
  const narudzba = req.body;
  console.log('Primljeni podaci:', narudzba);
  res.send('Vaša narudžba je uspješno zaprimljena!');
});
```

Primijetit ćete da će se u konzoli ispisati poruka `"Primljeni podaci: undefined"`. Razlog zašto se ne ispisuju podaci je taj što Express ne zna kako parsirati podatke u tijelu zahtjeva. Da bismo to omogućili, moramo koristiti **middleware** koji će parsirati podatke u tijelu zahtjeva. O middleware funkcijama više u sljedećim lekcijama, međutim za sada ćemo koristiti ugrađeni middleware `express.json()` koji će parsirati podatke u JSON formatu.

Jednostavno dodajemo na početku naše aplikacije, nakon definiranja instance aplikacije:

```
app.use(express.json());
```

Pokušajte ponovo. Vidjet ćete da podaci i dalje ne dolaze kada šaljemo kroz `curl`. Razlog je taj što `curl` ne šalje podatke u JSON formatu po *defaultu*, već to moramo specificirati u **zaglavlju** našeg HTTP zahtjeva.

Zaglavlja možemo specificirati pomoću opcije `-H`, a dodat ćemo zaglavlje `Content-Type: application/json`:

```
curl -X POST http://localhost:3000/naruci -H "Content-Type: application/json" -d
'{"pizza": "Margherita", "velicina": "srednja"}'
```

Ako ste upisali točno naredbu, trebali biste vidjeti ispis u konzoli:

```
Primljeni podaci: { pizza: 'Margherita', velicina: 'srednja' }
```

Sada kada imamo podatke o narudžbi, možemo ih koristiti u našoj aplikaciji. Na primjer, možemo poslati korisniku poruku s informacijama o narudžbi:

```
app.post('/naruci', (req, res) => {
  const narudzba = req.body;
  console.log('Primljeni podaci:', narudzba);
  res.send(`Vaša narudžba za ${narudzba.pizza} (${narudzba.velicina}) je uspješno
zaprimljena!`);
});
```

Što ako korisnik ne pošalje podatke o pizzi ili veličini pizze? Kako ćemo riješiti taj slučaj? 🤔

Možemo izvući ključeve JavaScript objekta kroz metodu `Object.keys` i provjeriti jesu li svi ključevi prisutni:

```
app.post('/naruci', (req, res) => {
  const narudzba = req.body;
  const kljucevi = Object.keys(narudzba);

  if (!(kljucevi.includes('pizza') && kljucevi.includes('velicina'))) {
    res.send('Niste poslali sve potrebne podatke za narudžbu!');
    return;
  }

  res.send(`Vaša narudžba za ${narudzba.pizza} (${narudzba.velicina}) je uspješno
zaprimljena!`);
});
```

Sada kada pošaljemo zahtjev bez podataka:

```
curl -X POST http://localhost:3000/naruci -H "Content-Type: application/json" -d '{}'
```

Ili s pogrešnim podacima:

```
curl -X POST http://localhost:3000/naruci -H "Content-Type: application/json" -d
'{"pizza": "Margherita", "cijena": 6.5}'
```

2.2.1 Kako slati `POST` zahtjeve jednostavnije?

Kako ne bismo morali svaki put pisati `curl` naredbe za slanje POST zahtjeva, možemo koristiti alate koji nam omogućuje puno jednostavnije slanje HTTP zahtjeva s tijelom zahtjeva, zaglavljima i drugim opcijama.

Jedan od takvih alata je Postman, koji je dostupan za sve platforme i omogućuje nam jednostavno slanje HTTP zahtjeva, testiranje API-ja, automatsko generiranje dokumentacije, itd.

Preuzmite Postman s ovog linka. Potrebno je izraditi račun, ali je besplatan za korištenje.

Jednom kada se prijavite, morate napraviti novi radni prostor (*workspace*). Kliknite na `New Workspace` i unesite naziv radnog prostora. Možete ga nazvati `Web aplikacije - Vježbe`.

Odaberite '+' i dodajte novu kolekciju koju možete nazvati `WA2` te dodajte novi zahtjev u kolekciju odabirom `"Add a request"`. Nazovite zahtjev `Jelovnik` i odaberite GET zahtjev (po defaultu je GET).

Vidjet ćete razno-razne opcije koje možete koristiti za slanje zahtjeva, kao što su **URL**, **HTTP metoda**, **zaglavlja**, **tijelo zahtjeva**, **autorizacija** itd.
Uočite da se unutar zaglavlja već nalazi postavljeno čak 7 različitih zaglavlja, dakle Postman automatski postavlja neka zaglavlja za nas.

Pošaljite zahtjev na endpoint `/pizze` i vidjet ćete rezultat u obliku JSON objekta s dostupnim pizzama.
Morate unijeti puni URL u formatu:

```
http://localhost:3000/pizze
```

Ako je sve OK, ispod će vam se prikazati JSON objekt unutar **Body** taba, ali možete vidjeti i **zaglavlja koja su došla s odgovorom**.

Postoji puno alternative Postmanu, npr. Insomnia, Paw, Thunder Client, HTTPie, od kojih se neki izvode na webu, a neki lokalno na računalu.

Zgodno je preuzeti i **Thunder Client** koji je dostupan kao ekstenzija za Visual Studio Code.

Otvorite Thunder Client ekstenziju i odaberite `New Request`. Unesite URL `http://localhost:3000/pizze` i odaberite metodu `GET`. Kliknite na `Send Request` i vidjet ćete isti rezultat kao i u Postmanu.

`POST` zahtjev možete poslati na isti način, samo odaberite metodu `POST` i unesite URL `http://localhost:3000/naruci`. U tijelo zahtjeva unesite JSON objekt s podacima o narudžbi:

```
{
  "pizza": "Capricciosa",
  "velicina": "jumbo"
}
```

Trebali biste dobiti poruku: `Vaša narudžba za Capricciosa (jumbo) je uspješno zaprimljena!`.

Vježba 1 - Naručivanje više pizze 🍕🍕🍕

Nadogradite Express poslužitelj na način da pohranjujete podatke o narudžbama "in-memory", odnosno u varijablu koja će se resetirati svaki put kada se poslužitelj ponovno pokrene.

Nadogradite POST rutu `/naruci` tako da očekuje od korisnika **polje objekata** s podacima o narudžbi. Svaki objekt mora sadržavati ključeve `pizza`, `velicina` i `kolicina`.

```
[
  {
    "pizza": "Capricciosa",
    "velicina": "jumbo",
    "kolicina": 1
  },
  {
    "pizza": "Vegetariana",
    "velicina": "srednja",
    "kolicina": 2
  }
]
```

Ako neki od ključeva nedostaje, vratite korisniku poruku da nije poslao sve potrebne podatke.

Provjerite je li korisnik naručio pizzu koja postoji u vašem jelovniku. Ako korisnik naruči pizzu koja ne postoji, vratite korisniku poruku da jedna ili više pizza koju je naručio ne postoji

Ako korisnik pošalje podatke u ispravnom formatu, dodajte narudžbu u listu narudžbi i vratite korisniku poruku da je narudžba za pizze (izlistajte naručene nazive pizza) uspješno zaprimljena.

Vježba 2 - Zanima nas i adresa dostave 🚙🏠

Nadogradite POST rutu `/naruci` tako da očekuje od korisnika dodatne podatke o narudžbi, kao što su `prezime`, `adresa` i `broj_telefona`.

Na jednak način kao u vježbi 1, provjerite jesu li svi potrebni podaci poslani i jesu li sve pizze koje je korisnik naručio prisutne u vašem jelovniku.

Primjer JSON objekta koji se šalje:

```
{
  "narudzba": [
    {
      "pizza": "Capricciosa",
      "velicina": "jumbo",
      "kolicina": 1
    },
    {
      "pizza": "Vegetariana",
      "velicina": "srednja",
      "kolicina": 2
    }
  ],
  "prezime": "Perić",
  "adresa": "Alda Negrija 6",
  "broj_telefona": "0912345678"
}
```

Ako korisnik pošalje podatke u ispravnom formatu, dodajte narudžbu u listu narudžbi i vratite korisniku `JSON` poruku sa sljedećim podacima:

```
message: "Vaša narudžba za pizza_1_naziv (pizza_1_velicina) i pizza_2_naziv
(pizza_2_naziv) je uspješno zaprimljena!",
prezime: "Perić",
adresa: "Alda Negrija 6",
ukupna_cijena: izračunajte ukupnu cijenu narudžbe
```

> to be continued...