Veebiarendus

Back-End arendus

Martti Raavel

martti.raavel@tlu.ee

Tänased teemad

- Meenutame eelmist loengut
- Kordamine
  - Struktureerimine
    - Vaatame üle teenused ja kontrollerid
  - Andmete saatmine API-sse
    - `req.params`
    - `req.body`
    - `req.query`
- Middleware

Millest rääkisime eelmises loengus?

Kordamine

Struktureerimine

- **Teenused** - loogika, mis tegeleb andmete töötlemisega
- **Kontrollerid** - vastutavad päringute vastuvõtmise ja vastuste saatmise eest

Andmete saatmine API-sse

- `req.params`
- `req.body`
- `req.query`

URI parameetrid (*req.params*)

URL-i kaudu saab andmeid saata URL-i parameetrite abil. Näiteks kui soovite saata toote ID, võite kasutada järgmist URL-i: `/tooted/:id`. Rakenduses Express pääsete URL-i parameetrile juurde, kasutades `req.params` objekti.

`req.params` näide

```
...
app.get('/users/:id', (req, res) => {
  const userId = req.params.id;
  res.send(`User ID: ${userId}`);
});
...
```

> Kui kasutate URI `/users/123`, siis `req.params` objekt näeb välja selline `{ id: '123' }`.

Päringu keha (*req.body*)

Päringu kehas saab andmeid saata selliste meetoditega nagu POST, PUT ja DELETE. Andmete saatmiseks kasutatakse `JSON` või `form` andmeid. Selleks, et `Express` rakenduses saada päringu keha, tuleb kasutada vahevara:

```
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
```

`req.body` näide

```
...
app.post('/students/add', (req, res) => {
  const student = req.body;
  console.log(student);
});
...
```

Kui saadate päringu `/students/add` URI-le objektiga:

```
{
  "firstName": "Angus",
  "lastName": "Ingram",
  "curriculum": "RIF22"
}
```

siis `req.body` objekt näeb välja selline:

`{ firstName: 'Angus', lastName: 'Ingram', curriculum: 'RIF22' }`.

Päringu parameetrid (*req.query*)

Päringuparameetrite abil saate andmeid saata URL-i kaudu. Näiteks kui soovite saata otsingupäringu, võite kasutada järgmist URL-i: `/search?q=example`. Rakenduses Express pääsete päringu parameetrile juurde, kasutades `req.query` objekti.

`req.query` näide

Saates päringu aadressile `/search?q=example`, saame sealt päringu prameetrite kättesaamiseks kasutada järgmist koodi:

```
...
app.get('/search', (req, res) => {
  const query = req.query.q; // 'example'
  res.send(`Search query: ${query}`);
});
...
```

> `req.query` objekt sisaldab selle näite puhul järgmist: `{ q: 'example' }`.

Middleware

`Middleware` funktsioonid on funktsioonid, millel on juurdepääs päringuobjektile (`req`), vastuseobjektile (`res`) ja järgmisele funktsioonile rakenduse päringu-vastuse tsüklis.

Next funktsioon

`Next` funktsioon on Express-ruuteri funktsioon, mis käivitamisel käivitab `middleware` praeguse `middleware`'i järel.

Põhimõtteliselt on tegemist funktsiooniga, mille käivitamine annab järjekorra üle järgmisele etapile rakenduse päringu-vastuse tsüklis.

Middleware näide

```
// Http päringute konsooli logimise middleware

const logger = (req, res, next) => {
  // Väljastatakse päringu sihtaadress, meetod ja aeg
  console.log(req.url, req.method, new Date().toISOString());
  // Next funktsiooni käivitamine annab järjekorra üle järgmisele middleware'le
  return next();
}
```

Middleware registreerimine

`Middleware` -t saab rakendada erinevalt.

Üks variantidest on registreerida `middleware` kõikidele päringutele:

```
...
// Middleware importimine (teekond sõltub faili asukohast)
const logger = require('./middlewares/logger');
...
// Middleware registreerimine
app.use(logger);
...
```

Middleware registreerimine teatud marsuutidele

Teine variant on registreerida `middleware` ainult teatud marsuutidele:

```
...
// Middleware importimine
const logger = require('./middlewares/logger');
...
// Middleware registreerimine
app.get('/api', logger, (req, res) => {
  res.send('Hello World!');
});
...
```

Kodutöö

- Paigalda oma arvutisse Docker Desktop rakendus