peatutor.com/express

Noden avulla pystyy tekemään backendiä Javascriptillä

Esimerkki Javascriptilllä

let a=5;

let b=11;

let c=a+b;

console.log(“summa on “´+c);

Terminaalissa node eka.js

VSCodessa Terminal. Voidaan ajaa debuggerilla.

Joudutaan Noden kanssa käyttämään asynkronisia funktioita. Onko säie tai thread tuttu?

Apachen veppipalvelussa jos olisi pankkiautomaatilla monta käyttäjää, se antaisi jokaiselle oman säikeen. Sen sijaan Node työskentelee yhdellä säikeellä.

Jos tehdään login synkronisella funktiolla, koko sovellus pysähtyy eikä etene ennen kuin saa vastauksen onko login onnistunut vai ei. Vaikka se kestää millisekunteja niin jos on tuhansia käyttäjiä niin se kestää. -> Ei käytetä

Käytetään asynkronista siirtoa. Kun automaatilta lähetetään http-pyyntö asynkronisella siirrolla. Vastaus kun tulee niin homma etenee -> Joudutaan käyttämään CAllbackeja.

Esimerkki osoitteessa peatutor.com/express/javascript.php

toka.js

**function doSomething(callback, delay) {**

**setTimeout(callback,delay);**

**}**

**function sayHello() {**

**console.log(”Hello”);**

**}**

**doSomething(sayHello,2000);**

**console.log(”The application is not blocked”);**

**EI ole oikea asyncroninen funktio, mutta delay ”Imitoi” sitä.**

**Callback on funktio, joka voidaan antaa toiselle funktiolle argumenttina. Qt:ssa tulee sama homma. Lähetetään asynkronisia http-pyyntöjä. Qt:ssa voisi käyttää callbackeja, mutta sinne on tehty erillinen SIgnal-SLot-systeemi.**

**On siirrytty PHP:stä Nodeen. PHP:tä ei ole opetettu muualla meillä enää. Node on tosi suosittu. PHP:tä on prosentuaalisesti hirveästi käytössä.**

**Express.js on Nodejs:ään tehty framework, joka helpottaa http-sovellusten käyttöä. Sanotaan Web-frameworkiksi. Esimerkissä on käytetty apuohjelmaa nimeltä Express-generator. Se generoio sovellukselle rungon, pääsee nopeammin eteenpäin.**

**SEURAAVAKSI GITHUB:**

**Jos haluaa niin voi ruveta tekemään samaa koodia.**

**git clone jokurepo**

**Kannattaa tehdä uusi branch, ei mainiin pusketa suoraan.**

**git checkout -b dev**

**Express-asennus**

**Jokonkin sopivaan hakemistoon Expressin asennus peatutor.com/express/Examples/express.php**

**Sieltä asennukset vanhaan malliin.**

**CORS = Cross ORigin resource sharing. REST-API:lla on oma IP-osoite. Clientti eli Qt-sovellus on mahdollisesti eri koneella, jolloin sillä on eri IP-osoite. Se ei suostu vastaamaan jos ei ole CORSIA.**

**Helmet parantaa http-juttujen turvallisuutta.**

**Express-generator kirjoittaa package.json tiedostoon. Nyt kun katsotaan em. tiedostoa.**

**npm install lukee tätä tiedostoa ja ne menee node\_modulesiin. Jos node\_modulesin hävittää, ei npm start toimi, mutta kun ajaa npm instal, se lukee package.jsonin ja asentaa tarvittavat systeemit.**

**.gitignore-tiedostoon node\_modules/**

**Seuraavaksi demo Pull Requestista. checkout -b dev, sinne joku tiedosto tee vaikka**

**ESIM: git add remote myRepo https://github.com/jjauhiai/testi.git**

git push -u myRepo dev

Pyritään pitämään main puhtaana, että sinne mitä vaan bugista koodia jonka muut kloonaa.

Kavereita kutsutaan näin: Mennään repoon, Settings, Collaborators

npm start kun katsotaan package.jsonin sisältö, se viittaa bin/www-hakemistoon. Sieltä www sisältää mm.

var port = normalizePort(process.env.PORT || '3000');

app.set('port', port);

Eli portti joko luetaan ympäristömuuttujasta TAI se on 3000…

Nyt ei siis ole ympäristömuuttujaa, joten käytetään 3000. Ympäristömuuttujat voidaan määritellä tietokannan yhteydessä tai ne saadaan pilvestä esim Heroku tai Azure.

app.js

Routes-kansiosta löytyy index.js ja users.js

app.js:ssä

var indexRouter = require('./routes/index');

vastaa

app.use('/', indexRouter);

ja

var usersRouter = require('./routes/users');

app.use('/users', usersRouter);

Pekka tekee tässä välissä uuden branchin examples

Ajetaan localhost:3000 ja locahost:3000/users

get on yksi http-metodeista. get put post delete. Lisätään users.js ennen module.exports=router;

router.get('/toka', function(req, res) {

  console.log("Olen toka :D !");

  res.send('Tokapa hyvinkin...')

});

Nextiä ei tässä tarvita. Kirjoittaa kaikki käsin.

<http://localhost:3000/users/toka>

router.get('/kolmas/:name', function(req, res) {

  console.log(req.params.name);

  res.send('Terve '+req.params.name);

});

router.get('/neljas/:fname/:lname', function(req, res) {

  //console.log(req.params.name);

  res.send('Terve '+req.params.fname +" "+req.params.lname);

});

**Parametrit:**

http-pyynto lähetetään (request tai req)

http-vastaus tulee paluupostissa (response tai res)

GET-metodissa data menee urlin perässä. http Header-osassa

POST-metodissa menee Body-osassa.

**Katsotaan esimerkkinä POST:**

router.post('/',function(request,response){

        response.send(request.body);

        console.log(request.body);

    }

);

Tätähän ei voi selaimen kautta kutsua. Tarvitaan joku työkalu.

Postman ! Komentoriviltä CURL

Collections -> New Collection -> vaikka nimeksi TVT21SPL -> Add Request -> POST eka

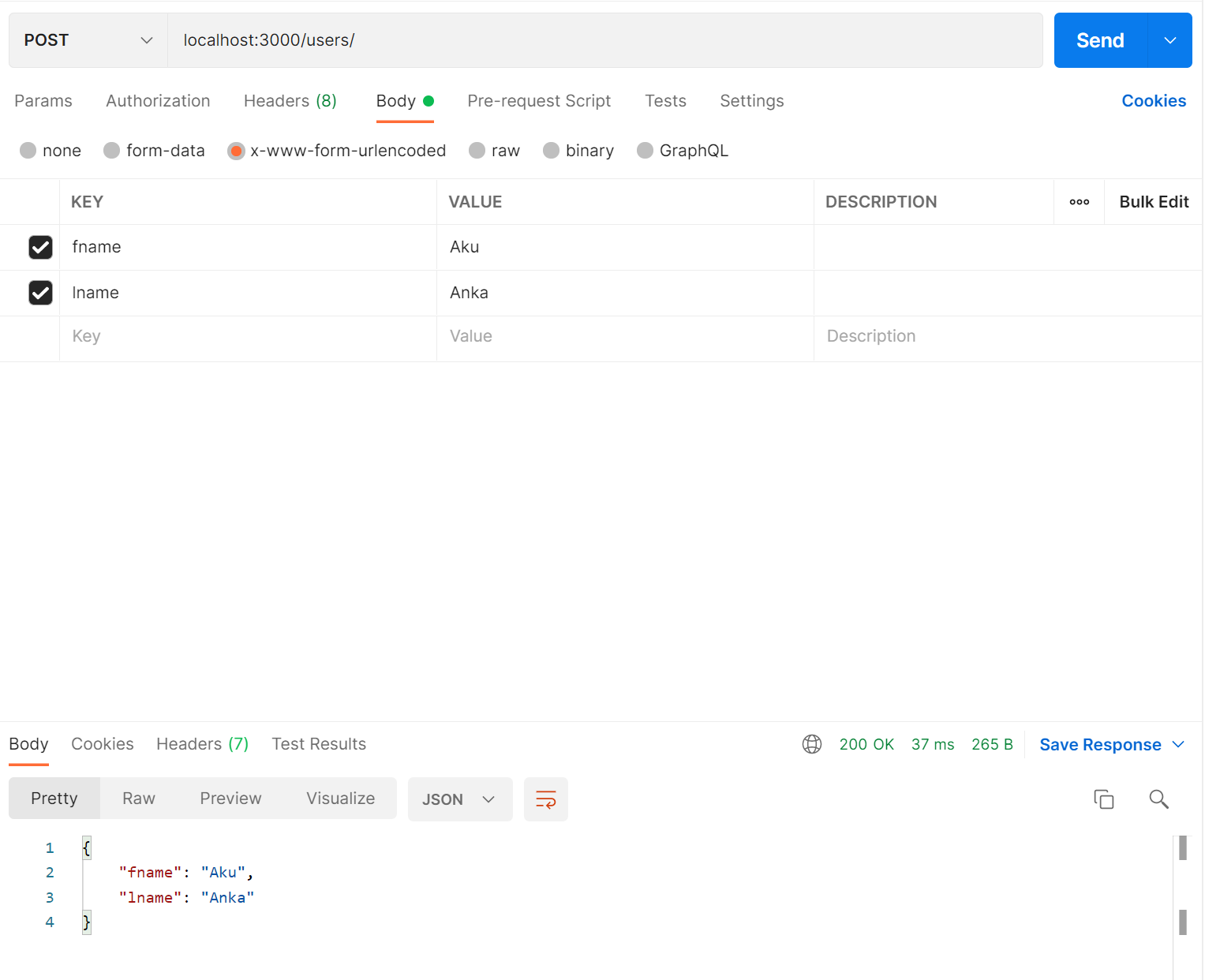
Body -> x-www-form-urlencoded

**MIDDLEWARE**

Funktio on sellainen, että sitä ei kutsuta requestilla, mutta ne suoritetaan aina kun kutsutaan requestia, joka on sen jälkeen.

.GET sijaan .USE

Javascriptissä puolipiste ei ole pakollinen



Paluupostissa saadaan JSON-objekti.

app.js sisältää:

app.use(express.json());

app.use(express.urlencoded({ extended: false }));

Voidaan lähettää muodossa urlencoded (vrt. Postman).

router.post('/',function(request,response){

  response.send(request.body.lname+" "+request.body.fname);

  console.log(request.body.lname+" "+request.body.fname);

}

);

Nyt kenttien nimien on oltava lname ja fname !

**PUT**

Putissa on parametreja sekä headerissa että bodyssa.

router.put ('/:id',function(request,response){

  response.send("Henkilölle "+request.params.id+" muutetaan nimeksi "+request.body.fname);

  console.log("Henkilölle "+request.params.id+" muutetaan nimeksi "+request.body.fname);

});

