**Node.js Workshop : Express framework**

After completing this workshop the student is knows how to:

* how to install Express
* request and read parameters
* send a response to the client
* how to work with HTTP headers
* how to serve static assets
* how to validate input in Express

**What is Express**

Official [express Web page](https://expressjs.com/).

Additional [Express learning](https://expressjs.com/en/resources/learning.html)

**Exercise 1. How to install express and make a simple express server**

Make an empty folder, and create a new Node.js project with this command:

*npm init -y*

then run this:

*npm install express*

to install Express into the project.

Makean *index.js* file in your project root folder

// haetaan express-moduuli ja luodaan sovelluksen instanssi

const express = require('express')

const app = express()

// määritellään portti, jota sovellus kuuntelee

const PORT = process.env.PORT || 3000

// määritellään juurireitti ja vastauksena lähetetään Hello World!

app.get('/', (req, res) => res.send('Hello World!'))

// avataan portti ja tulostetaan konsoliin viesti, kun sovellus on käynnistynyt

app.listen(PORT, () => console.log(`Server is running on port ${PORT}`))

and start the server using *node* command.

**Exercise 2. Get familiar with the request parameters**

Request object holds all the HTTP request information. Main properties are listed below

| **PROPERTY** | **DESCRIPTION** |
| --- | --- |
| .app | holds a reference to the Express app object |
| .baseUrl | the base path on which the app responds |
| .body | contains the data submitted in the request body (must be parsed and populated manually before you can access it) |
| .cookies | contains the cookies sent by the request (needs the cookie-parser middleware) |
| .hostname | the hostname as defined in the [Host HTTP header](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Host) value |
| .ip | the client IP |
| .method | the HTTP method used |
| .params | the route named parameters |
| .path | the URL path |
| .protocol | the request protocol |
| .query | an object containing all the query strings used in the request |
| .secure | true if the request is secure (uses HTTPS) |
| .signedCookies | contains the signed cookies sent by the request (needs the cookie-parser middleware) |
| .xhr | true if the request is an [XMLHttpRequest](https://www.freecodecamp.org/news/xhr/) |

Modify your index.js file so that you will log different req properties to console (e.g. *console.log(req.ip)*).

**Exercise 3. How set the HTTP response status and how to Work with HTTP Headers**

When your server listens for connections on a route in Express, the callback function will be invoked on every network call with a Request object instance and a Response object instance.

res.send('Hello World!')

If you pass in a string in Response object, it sets the Content-Type header to text/html.

If you pass in an object or an array in Response object, it sets the application/json Content-Type header and parses that parameter into JSON.

Use the status() method on the response object

app.get('/', (req, res) => {

    // set res.status(200).send('Hello World!')

    res.status(200).send('Hello World!')

});

Modify your index.js file

You can access all the HTTP headers using the Request.headers property.

// määritellään juurireitti ja vastauksena lähetetään Hello World!

app.get('/', (req, res) => {

    console.log(req.headers)

    // set res.status(200).send('Hello World!')

    res.status(200).send('Hello World')

});

Modify your index.js file and investigate req.headers in your terminal

You can change any HTTP header value using *Response.set():*

app.get('/', (req, res) => {

    console.log(req.headers)

    res.set('Content-Type', 'text/html')

    // set res.status(200).send('Hello World!')

    res.status(200).send(`<h1>Hello World</h1>`)

});

Modify your index.js file and see the impact on your browser window

## Exercise 4. How to Serve Static Assets with Express

It's common to have images, CSS, and more in a public subfolder, and expose them to the root level

Make a subfolder named *public* to your project root.

Create a *index.html* file in your *public* subfolder

Add the following to your index.js file and

// määritellään staattisten tiedostojen hakemisto

app.use(express.static('public'))

Now your newly created .html content should be visible on your browser.

## Exercise 5. How to validate input in Express

You have a POST endpoint in your server that accepts the name, email, and age parameters. How do you perform server-side validation on those results to make sure that:

* name is a string of at least 3 characters?
* email is a real email?
* age is a number, between 0 and 110?

Use express-validator package – run the following command in your terminal

npm install express-validator

Seuraavaksi määritellään tarvittavat muutujat, jotka tuodaan express-validator modulista

// määritellään check ja validationResult -muuttujat

const { check, validationResult } = require('express-validator');

and finally we can add new POST route to our server

// määritellään reitti /form

app.post('/form', [

    // määritellään tarkistukset

    check('name').isLength({ min: 3 }),

    check('email').isEmail(),

    check('age').isNumeric()

], (req, res) => {

    // tarkistetaan oliko req objektin muuttujien oikeellisuus ja palautetaan virheilmoitus, jos tarkistus ei mene läpi

    const errors = validationResult(req)

    if (!errors.isEmpty()) {

        return res.status(422).json({ errors: errors.array() })

    }

    // tulostetaan konsoliin lomakkeen tiedot

    const name = req.body.name

    const email = req.body.email

    const age = req.body.age

    console.log(`Name: ${name}, Email: ${email}, Age: ${age}`)

})

The form data will be sent in the POST request body. To extract it, you will need to use the express.urlencoded() middleware.

// luetaan lomakkeen tiedot ja tallennetaan ne req-objektiin

app.use(express.urlencoded({

    extended: true

}))

If you want to test your POST route you can do it using [postman](https://www.postman.com/)