Node.js - Laboratorium 7

Wbudowany moduł `HTTP`(https://nodejs.org/dist/latest-v10.x/docs/api/http.html)

Node posiada wbudowany moduł pozwalający na obsługę protokołu http. Moduł ten jest przeznaczony zarówno do wysyłania żądać po stronie klienta jak również do odbioru i przetwarzania żądań jako serwer.

Dla przykładu, aby wysłać żądanie do serwera po dane musimy wykorzystać funkcję `.get()` lub `.request()`, np.:

```
const http = require('http');

http.get('http://localhost:4500', response => {
    console.log(response);
});
```

Konstrukcja bardzo podobna do zewnętrznego modułu `request` jaki wykorzystywaliśmy we wcześniejszych zajęciach.

Uruchomienie serwera z wbudowanego modułu `HTTP` wygląda następująco:

```
const http = require('http');

// tworzymy nową instancję naszego serwera
const server = http.createServer((req, res) => {
    // ustawiamy statusCode oraz typ wysłanej odpowiedzi
    res.writeHead(200, {
        'Content-Type': 'text/plain'
    });
    // zamykamy połączenie wysyłając tekst
    res.end('...');
});

// uruchamiamy nasz serwer na porcie 4500
server.listen(4500);
```

Wbudowany moduł `URL`(https://nodejs.org/dist/latest-v10.x/docs/api/url.html)

Moduł pozwala na parsowanie adresu url na obiekt zawierający podział na protokół, adres, parametry wyszukiwania, itp..

```
const url = new URL('http://localhost:8080/abc?q=some-string');
```

```
console.log(url);
```

```
URL {
  href: 'http://localhost:8080/abc?q=some-string',
  origin: 'http://localhost:8080',
  protocol: 'http:',
  username: '',
  password: '',
  host: 'localhost:8080',
  hostname: 'localhost',
  port: '8080',
  pathname: '/abc',
  search: '?q=some-string',
  searchParams: URLSearchParams { 'q' => 'some-string' },
  hash: '' }
```

`Express` (https://expressjs.com/)

Web framework pozwalający w łatwy i szybki sposób postawić serwer HTTP w środowisku NodeJS.

Pierwsze kroki

1. `npm install express`
2. zaimportowanie modułu do aplikacji i odpowiednie użycie:

```
const express = require('express');

const app = express(); // tworzymy nową instancję serwera

// dodajemy reguły do naszego serwera
// jeżeli użytkownik wejdzie na nasz web serwer
app.get('/', (req, res) => {
    // wysyłamy do klienta dane
    res.send('hello world!');
});

app.listen(4500); // uruchamiamy nasz web serwer na porcie 4500
```

Przydatne linki

Dokumentacja `HTTP`: https://nodejs.org/dist/latest-v10.x/docs/api/http.html

Dokumentacja `Express`: https://expressjs.com/en/4x/api.html

`Express` na `mozilla.org`: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs

Tutorial z `Express`: https://www.tutorialspoint.com/expressjs/index.htm

Zadania do wykonania na laboratorium

1. Stwórzmy nasz pierwszy serwer wykorzystując wbudowany moduł `HTTP`, który wyśle do naszego klienta przywitanie. Sprawdźmy czy działa aplikacja poprzez uruchomienie przeglądarki i wysłanie żądania do naszego serwera. (port 4700)

2. Dodajmy do naszej aplikacji z zadania 1 warunek, jeżeli w adresie pojawi się parametr `name` przywitajmy naszego użytkownika po nazwie. (wykorzystajmy wbudowany moduł `URL`)

3. Jak możemy zaobserwować podczas tworzenia aplikacji na wbudowanym module `HTTP` uciążliwe jest pobieranie danych wysłanych przez klienta. Aby usprawnić tworzenie serwera web powstały różne frameworki, m.in. `Express` który pozwala na szybsze postawienie naszego serwera.

Zmieńmy nasz kod z zadania 2 tak aby był wykorzystywany framework `Express`.

4. Stwórzmy aplikację która pobierając 2 parametry `a` i `b` z adresu url wykona mnożenie w naszej aplikacji. Rezultat działania powinniśmy wysłać do użytkownika końcowego(klienta).

5. Rozszerzmy naszą aplikację z zadania 4 o dodatkowe działania matematyczne takie jak mnożenie, dzielenie i odejmowanie. Podzielmy zadania na odpowiednie ścieżki.

```
/mnozenie?a=1&b=3
/dzielenie?a=1&b=3
/dodawanie?a=1&b=3
/odejmowanie?a=1&b=3
```

6. Stwórzmy aplikację której zadaniem będzie operacja na tablicy zawierającej użytkowników

    - Stworzyć 'końcówkę' `/add` do dodawania użytkownika która przyjmuje parametry `?name=Jan&username=janko&email=jan@nowak.abc`
    - Dodać ścieżkę wyświetlania wszystkich użytkowników oraz jeżeli zostanie podany odpowiedni `id` wyświetlić jedynie jednego użytkownika
    - Rozszerzyć aplikację o kasowanie użytkownika poprzez odpowiednią ścieżkę.

7. Wzorując się na zadaniu 6 stwórzmy analogicznie obsługę tablicy zawierającej posty. Aplikacja ma rozszerzyć naszą już istniejącą aplikację z zadania 6.

Struktura `postu`:

```
{
    id: 0,
    userId: 0,
    title: '',
    body: ''
}
```

8. Podzielmy odpowiednio naszą aplikację z zadania 7 wykorzystując `express.Router`(https://expressjs.com/en/4x/api.html#router)