Raport z wykonania ćwiczenia REST, Mikroserwisy

Jakub Płotnikowski

Styczeń 2020

Spis treści

1 Kod z zadań 1, 2, 3

Jako, że zadania 1, 2 oraz 3 wykonałem oraz oddałem na zajęciach, wklejam tylko ich kod bez dodatkowego omówienia.

1.1 Zadanie 1

```
// File 01_HttpServer/app.js
// Import Express web framework
const express = require("express");
// Create main app
const app = express();

// function printReqSummary(request) {
//   // Display handled HTTP method and link (path + queries)
//   console.log(`Handling ${request.method} ${request.originalUrl}`);
//}

// Helper function -- print request summary
function printReqSummary(request) {
  // Display handled HTTP method and link (path + queries)
  console.log(`Handling ${request.method} ${request.originalUrl} ${Date.now()
      }`);
}

// GET / -- Show main page
app.get("/", function(request, response) {
  printReqSummary(request);
  // Send response to the request (here as a HTML markup)
  response.send("<h1>HTTP Server</h1><p>Go to /hello subpage!</p>");
});

// GET /hello -- Show message
app.get("/hello", function(request, response) {
  printReqSummary(request);
  response.send("<p>Anonymous message: Oh, Hi Mark!</p>");
});

app.get("/time", function(request, response) {
  printReqSummary(request);
  response.send(`<p>Printing current time: ${new Date()}</p>`);
});

// Start HTTP server at port 3000
//  (type in the browser or use cURL: http://localhost:3000/)
app.listen(3000);
```

1.2 Zadanie 2

```javascript
// File 02_UrlParameters/app.js
const express = require("express");
const app = express();

function printReqSummary(request) {
  console.log(`Handling ${request.method} ${request.originalUrl}`);
}

function getRandomInt(min, max) {
  return Math.floor(Math.random() * (max - min + 1)) + min;
}

function getRandomParameter(parameters) {
  return parameters["p" + getRandomInt(1, 3)];
}

// GET / -- Show main page
app.get("/", function(request, response) {
  printReqSummary(request);
  response.send(
    `<h1>URL Parameters (and Queries)</h1><ul>
      <li>Show normal message (GET /hello/segment1)</li>
      <li>Show special message (GET /hello/segment1/segment2?age=NUMBER&height=
          NUMBER)</li>
    <li> where segment1, segment2 - any valid URL part</li>
    </ul>`
  );
});

// GET /hello/:name -- Show normal message for a named person
app.get("/hello/:name", function(request, response) {
  printReqSummary(request);
  // Grab URL parameters from `request.params` object
  response.send(`<p>Normal message for: ${request.params.name}</p>`);
});

// GET /hello/:name/:surname -- Show special message with plenty of parameters
app.get("/hello/:name/:surname", function(request, response) {
  printReqSummary(request);
  // Grab (optional) URL queries from `request.query` object
  const age = request.query.age !== null ? request.query.age : "unknown";
  const height =
    request.query.height !== null ? request.query.height : "unknown";
  response.send(
    `<p>Special message for: ${request.params.name} ${request.params.surname}
      (age: ${age} years, height: ${height} cm)</p>`
  );
});

app.get("/random/:p1/:p2/:p3", function(request, response) {
  printReqSummary(request);
  response.send(`<p>Random parameter: ${getRandomParameter(request.params)}</p
      >`);
});


app.listen(3000);
```

1.3 Zadanie 3

```
// File 03_HttpMethods/app.js
//
const express = require("express");
const app = express();

function printReqSummary(request) {
  console.log(`Handling ${request.method} ${request.originalUrl}`);
}

/* Store items collection in this array */
let items = [];

/* GET / -- Show main page */
app.get("/", function(request, response) {
  printReqSummary(request);
  response.send(
    `<h1>HTTP Methods</h1><ul>
      <li>Show items (GET /item)</li>
      <li>Add an item (PUT /item/:name)</li>
      <li>Remove an item (DELETE /item/:name)</li></ul>`
  );
});

/* GET /item -- Show all items from the collection */
app.get("/item", function(request, response) {
  printReqSummary(request);
  response.send(`<p>Available items: ${items.toString()}</p>`);
});

// adding new item
app.post("/item/", function(request, response) {
  printReqSummary(request);
  const name = request.query.name !== null ? request.query.name : "undefined";
  /* Is the item in collection? */
  if (items.includes(name)) {
    response.send(`<p>Item "${name}" already in collection</p>`);
  } else {
    items.push(name);
    response.send(`<p>Item "${name}" added successfully</p>`);
  }
});

// modification of an item
app.put("/item/:name", function(request, response) {
  printReqSummary(request);
  const itemName = request.params.name;
  const newItemName = request.query.newItemName !== null ? request.query.
      newItemName : "undefined";
  /* Is the item in collection? */
  if (items.includes(itemName)) {
    items[items.indexOf(itemName)] = newItemName;
    response.send(`<p>Item "${itemName}" changed to "${newItemName}"
        successfully</p>`);
  } else {
    response.send(`<p>Item "${itemName}" not found</p>`);
  }
});

```

```
/* DELETE /item/:name -- remove a given item from the collection */
app.delete("/item/:name", function(request, response) {
  printReqSummary(request);
  const itemName = request.params.name;
  /* Is the item in collection? */
  if (items.includes(itemName)) {
    items = items.filter(item => item !== itemName);
    response.send(`<p>Item "${itemName}" removed successfully</p>`);
  } else {
    response.send(`<p>Item "${itemName}" doesn't exists</p>`);
  }
});

app.listen(3000);
```

2 Zadanie 4

2.1 Treść zadania

Wyprowadzić i przeanalizować wynik metody GET dla bazowego URLa. Zaobserwować rezultaty dla URLa zawierającego numer (id) pacjenta w zależności od użytej metody HTTP. Dla testowania wygodnie jest użyć wywołania curl -X Przeanalizować różnice w logice poszczególnych implementacji kodu dla obsługi tych metod. Uwaga: Zwrócić uwagę na fragment kodu pomiędzy db i średnikiem, który odwołuje się do kodu pakietu lowdb, zainicjowanego na początku. Zamieścić w raporcie przykładowe wyniki (i komentarz do nich).

2.2 Pobranie listy pacjentów

```
  curl -X GET localhost:3000/patient
  {"error":"No patients are registered"}
```

2.3 Dodanie przykładowego pacjenta

```
curl -X POST "localhost:3000/patient?name=Jakub&surname=Plotnikowski"
{"id":1,"name":"Jakub","surname":"Plotnikowski"}
```

2.4 Zmiana imienia i nazwiska pacjenta

2.5 Ponowne pobranie listy pacjentów

```
  curl -X GET localhost:3000/patient
  {"error":"No patients are registered"}
```

2.6 Usunięcie pacjenta o id równym 1

```
curl -X DELETE "localhost:3000/patient/1"
{"message":"Patient removed successfully"}
```

2.7 Ponowne pobranie listy pacjentów

```
  curl -X GET localhost:3000/patient
  {"error":"No patients are registered"}
```