



Москва, 2020

Цель: научиться работать с несколькими серверами, передавать параметры скрипту и изучить дочерние процессы и способы взаимодействия между ними.

Задание 1

Создать сервер А. На стороне сервера хранится файл с содержимым в формате JSON. При получении запроса на `/insert/record` идёт добавление записи в файл. При получении запроса на `/select/record` идёт получение записи из файла. Каждая запись хранит информацию о машине (название и стоимость).

Создать сервер Б. На стороне сервера хранится файл с содержимым в формате JSON. Каждая запись в файле хранит информацию о складе и массиве машин, находящихся на данном складе. То есть каждая запись хранит в себе название склада (строку) и массив названий машин (массив строк). При получении запроса на `/insert/record` идёт добавление записи в файл. При получении запроса на `/select/record` идёт получение записи из файла.

Создать сервер С. Сервер выдаёт пользователю страницы с формами для ввода информации. При этом сервер взаимодействует с серверами А и Б. Реализовать для пользователя функции:

- создание нового типа машины
- получение информации о стоимости машины по её типу
- создание нового склада с находящимися в нём машинами
- получение информации о машинах на складе по названию склада

Реализовать удобный для пользователя интерфейс взаимодействия с системой (использовать поля ввода и кнопки).

index_A.js (Сервер A)

```
let file = "cars.txt";

// импорт библиотеки
const express = require("express");
const fs = require("fs");

// запускаем сервер
const app = express();
const port = 5002;
app.listen(port);
console.log("Server on port " + port);

// заголовки для ответа
app.use(function(req, res, next) {
  res.header("Cache-Control", "no-cache, no-store, must-revalidate");
  res.header("Access-Control-Allow-Headers", "Origin, X-Requested-
With, Content-Type, Accept");
  res.header("Access-Control-Allow-Origin", "*");
  next();
});

// загрузка тела
function loadBody(request, callback) {
  let body = [];
  request.on('data', (chunk) => {
    body.push(chunk);
  }).on('end', () => {
    body = Buffer.concat(body).toString();
    callback(body);
  });
}

function add_record(name, price) {
  let data = [];
  data.push(fs.readFileSync(file, "utf-8"));

  if (data !== '')
  {
    let temp = JSON.parse(data);
    for (let i = 0; i < temp.length; i++)
      if (temp[i].name == name)
        return "ERROR: car is already exist";

    temp.push({"name": name, "price": price});
    data = temp;
  }
  else
    data[0] = {"name": name, "price": price};
}
```

```

    fs.writeFileSync(file, JSON.stringify(data))
    return "Information was added";
}

function show_record(name)
{
    let data = [];
    data.push(fs.readFileSync(file, "utf-8"));

    if (data != '')
    {
        let temp = JSON.parse(data);
        for (let i = 0; i < temp.length; i++)
            if (temp[i].name == name)
                return temp[i];
    }
    return {"name": null, "price": null};
}

// приём запроса
app.post("/insert/record", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        const name = obj["name"];
        const price = obj["price"];
        let msg = add_record(name, price);
        response.end(JSON.stringify(msg));
    });
});

app.post("/select/record", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        const name = obj["name"];
        let car_info = show_record(name);
        response.end(JSON.stringify(car_info));
    });
});

```

index_B.js (Сервер B)

```

let file = "stores.txt";

// импорт библиотек
const express = require("express");
const fs = require("fs");

```

```

// запускаем сервер
const app = express();
const port = 5003;
app.listen(port);
console.log(`Server on port ${port}`);

// заголовки в ответ клиенту
app.use(function(req, res, next) {
  res.header("Cache-Control", "no-cache, no-store, must-revalidate");
  res.header("Access-Control-Allow-Headers", "Origin, X-Requested-
With, Content-Type, Accept");
  res.header("Access-Control-Allow-Origin", "*");
  next();
});

// загрузка тела
function loadBody(request, callback) {
  let body = [];
  request.on('data', (chunk) => {
    body.push(chunk);
  }).on('end', () => {
    body = Buffer.concat(body).toString();
    callback(body);
  });
}

function add_record(name, cars) {
  let data = [];
  data.push(fs.readFileSync(file, "utf-8"));

  if (data !== '')
  {
    let temp = JSON.parse(data);
    for (let i = 0; i < temp.length; i++)
      if (temp[i].name == name)
        return "ERROR: store is already exist";

    temp.push({"name": name, "cars": cars});
    data = temp;
  }
  else
    data[0] = {"name": name, "cars": cars};

  fs.writeFileSync(file, JSON.stringify(data))
  return "Information was added";
}

function show_record(name)
{
  let data = [];

```

```

    data.push(fs.readFileSync(file, "utf-8"));

    if (data != '')
    {
        let temp = JSON.parse(data);
        for (let i = 0; i < temp.length; i++)
            if (temp[i].name == name)
                return temp[i];
    }
    return {"name": null, "price": null};
}

// приём запроса
app.post("/insert/record", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        const name = obj["name"];
        const cars = obj["cars"];
        let msg = add_record(name, cars);
        response.end(JSON.stringify(msg));
    });
});

app.post("/select/record", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        const name = obj["name"];
        let store_info = show_record(name);
        response.end(JSON.stringify(store_info));
    });
});
});

```

index_C.js (Сервер C)

```

const express = require("express");
const request = require("request");

const app = express();
const port = 5000;
app.listen(port);
console.log(`Server on port ${port}`);

// отправка статических файлов
const way = __dirname + "/static";
app.use(express.static(way));

app.set("view engine", "hbs");

```

```

app.use(function(req, res, next) {
  res.header("Cache-Control", "no-cache, no-store, must-revalidate");
  res.header("Access-Control-Allow-Headers", "Origin, X-Requested-
With, Content-Type, Accept");
  res.header("Access-Control-Allow-Origin", "*");
  next();
});

// функция для отправки POST запроса на другой сервер
function sendPost(url, body, callback) {
  const headers = {};
  headers["Cache-Control"] = "no-cache, no-store, must-revalidate";
  headers["Connection"] = "close";

  request.post({
    url: url,
    body: body,
    headers: headers,
  }, function (error, response, body) {
    if(error) {
      callback(null);
    } else {
      callback(body);
    }
  });
}

app.get("/insert/car", function(request, response) {
  const name = request.query.name;
  const price = request.query.price;

  sendPost("http://localhost:5002/insert/record", JSON.stringify({
    name: name,
    price: price
  }), function(answerString) {
    response.end(answerString);
  });
});

app.get("/select/car", function(request, response) {
  const name = request.query.name;

  sendPost("http://localhost:5002/select/record", JSON.stringify({
    name: name,
  }), function(answerString) {
    response.end(answerString);
  });
});

app.get("/insert/store", function(request, response) {

```

```

    const name = request.query.name;
    const cars = request.query.cars.split(" ");
    sendPost("http://localhost:5003/insert/record", JSON.stringify({
      name: name,
      cars: cars
    })), function(answerString) {
      response.end(answerString);
    });
  });
})

app.get("/select/store", function(request, response) {
  const name = request.query.name;

  sendPost("http://localhost:5003/select/record", JSON.stringify({
    name: name,
  })), function(answerString) {
    response.end(answerString);
  });
});

app.get("/insert/record/car", function(request, response) {
  response.render("page_add_car.hbs", null);
})

app.get("/select/record/car", function(request, response) {
  response.render("page_show_car.hbs", null);
})

app.get("/insert/record/store", function(request, response) {
  response.render("page_add_store.hbs", null);
})

app.get("/select/record/store", function(request, response) {
  response.render("page_show_store.hbs", null);
})

```

Тесты

Машины и склады

Машины

Создать новый тип машины

Узнать стоимость машины

Склады

Создать новый склад с машинами

Узнать информацию о машинах на складе

| | |
|--|--|
| <div> <div>Добавить машину</div> <div> Введите название: <div>car8</div> </div> <div> Введите стоимость: <div>760000</div> </div> <div>Готово</div> </div> | <div>"Information was added"</div> |
| <div> <div>Добавить машину</div> <div> Введите название: <div>car8</div> </div> <div> Введите стоимость: <div>760000</div> </div> <div>Готово</div> </div> | <div>"ERROR: car is already exist"</div> |

| | |
|--|---|
| <p>Узнать стоимость машины по типу</p> <p>Введите тип:</p> <input type="text" value="car1"/> <p>Готово</p> | <p>Результат поиска:</p> <p>Не найдено</p> |
| <p>Узнать стоимость машины по типу</p> <p>Введите тип:</p> <input type="text" value="car2"/> <p>Готово</p> | <p>Результат поиска:</p> <p>Название: car2</p> <p>Стоимость: 500</p> |
| <p>Добавить склад</p> <p>Введите название:</p> <input type="text" value="store3"/> <p>Введите названия машин через пробел:</p> <input type="text" value="car1 car2 car3 car4"/> <p>Готово</p> | <p>"Information was added"</p> |
| <p>Добавить склад</p> <p>Введите название:</p> <input type="text" value="store3"/> <p>Введите названия машин через пробел:</p> <input type="text" value="car1 car2 car3 car4"/> <p>Готово</p> | <p>"ERROR: store is already exist"</p> |

Получить информацию о машинах на складе

Введите название склада:

Готово

Результат поиска:

Название: store3

Машины на складе: car1,car2,car3,car4

Получить информацию о машинах на складе

Введите название склада:

Готово

Результат поиска:

Не найдено

Задание 2

Написать скрипт, который принимает на вход число и считает его факториал. Скрипт должен получать параметр через process.argv.

Написать скрипт, который принимает на вход массив чисел и выводит на экран факториал каждого числа из массива. Скрипт принимает параметры через process.argv.

При решении задачи вызывать скрипт вычисления факториала через execSync.

index.js

```
"use strict";

const execSync = require("child_process").execSync;

function find_factorial(number) {
  const options = {encoding: 'utf8'};
  const cmd = `node find_factorial.js ${number}`;
  const answer = execSync(cmd, options);
  return parseInt(answer);
}

console.log("Число      Факториал");
for (let i = 2; i < process.argv.length; i++)
  console.log(`${process.argv[i]}      ${find_factorial(process.argv[i])}`);
```

find_factorial.js

```
"use strict";

function factorial(num) {
  if (num < 0)
    return;
  let temp = 1;
  for (let i = 2; i <= num; i++)
    temp *= i;
  return temp;
}

console.log(factorial(parseInt(process.argv[2])));
```

Тесты:

| Тест | Результат | | | | | | | | | | | | | | | | | | | | | | |
|--|---|-------|-----------|----|-----------|---|---|---|---|---|---|---|----|---|-----|---|-----|---|------|---|-------|---|--------|
| <code>node index.js 0 1 2 3 4 5 6 7 8 9</code> | <table><tr><th>Число</th><th>Факториал</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td></tr><tr><td>2</td><td>2</td></tr><tr><td>3</td><td>6</td></tr><tr><td>4</td><td>24</td></tr><tr><td>5</td><td>120</td></tr><tr><td>6</td><td>720</td></tr><tr><td>7</td><td>5040</td></tr><tr><td>8</td><td>40320</td></tr><tr><td>9</td><td>362880</td></tr></table> | Число | Факториал | 0 | 1 | 1 | 1 | 2 | 2 | 3 | 6 | 4 | 24 | 5 | 120 | 6 | 720 | 7 | 5040 | 8 | 40320 | 9 | 362880 |
| Число | Факториал | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 1 | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 1 | | | | | | | | | | | | | | | | | | | | | | |
| 2 | 2 | | | | | | | | | | | | | | | | | | | | | | |
| 3 | 6 | | | | | | | | | | | | | | | | | | | | | | |
| 4 | 24 | | | | | | | | | | | | | | | | | | | | | | |
| 5 | 120 | | | | | | | | | | | | | | | | | | | | | | |
| 6 | 720 | | | | | | | | | | | | | | | | | | | | | | |
| 7 | 5040 | | | | | | | | | | | | | | | | | | | | | | |
| 8 | 40320 | | | | | | | | | | | | | | | | | | | | | | |
| 9 | 362880 | | | | | | | | | | | | | | | | | | | | | | |
| <code>node index.js 12</code> | <table><tr><th>Число</th><th>Факториал</th></tr><tr><td>12</td><td>479001600</td></tr></table> | Число | Факториал | 12 | 479001600 | | | | | | | | | | | | | | | | | | |
| Число | Факториал | | | | | | | | | | | | | | | | | | | | | | |
| 12 | 479001600 | | | | | | | | | | | | | | | | | | | | | | |

Вывод

В ходе лабораторной работы была освоена работа с несколькими серверами, скриптами и дочерними процессами. На основе полученных знаний были выполнены соответствующие практические задания и составлен отчет о проделанной работе.