МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чернівецький національний університет імені Юрія Федьковича

Сучасні технологій розробки web-додатків

Практична робота №1

Чернівці 2020

1) **Встановлення Node.js**
   https://htmlacademy.ru/blog/boost/backend/installing-nodejs

2) **Node.js + Express.js, простий web-сервер**

Node.js володіє неблокуючим введенням-виведенням, це добре для API, до якого буде звертатися безліч клієнтів. Express.js - розвинений, легкий фреймворк, що дозволяє швидко описати всі шляхи (API endpoints), які ми будемо обробляти. Так само до нього можна знайти безліч корисних модулів.

Створюємо новий проект з єдиним файлом server.js. Так як додаток буде цілком покладатися на Express.js, встановимо його. Установка сторонніх модулів відбувається через Node Package Manager виконанням команди npm install modulename в папці проекту.

CMD

```
mkdir NodeAPI
cd NodeAPI
npm i express
```

Express встановиться в папку node_modules. Підключивши його до додатка:

server.js

```
var express = require('express');
var app = express();

app.listen(1337, function() {
   console.log('Express server listening on port 1337');
});
```

Запустимо додаток через IDE або консоль (node server.js). Даний код створить веб-сервер на “http://localhost:1337/”. Якщо спробувати його відкрити - він виведе повідомлення “Can not GET /”. Це тому що ми ще не задали жодного шляху (route). Далі створимо кілька шляхів:

server.js

```
var express = require('express');
var app = express();

app.get('/api', function (req, res) {
   res.send('API is running');
});

app.listen(1337, function() {
   console.log('Express server listening on port 1337');
});
```

Тепер “http://localhost:1337/api” поверне наше повідомлення.

3) Error handling

Спершу підключимо зручний модуль для логування (збереження) помилок «Winston». Використовувати ми його будемо через свою обгортку. Встановимо в корені проекту «Winston» потім створимо папку «libs» і там файл log.js:

CMD

```
npm i winston
```

libs/ log.js

```
var winston = require('winston');

function getLogger(module) {
    //отобразим метку с именем файла, который выводит сообщение
    var path = module.filename.split('/').slice(-2).join('/');

    return winston.createLogger({
        transports : [
            new winston.transports.Console({
                colorize: true,
                level: 'debug',
                label: path})
        ]
    });
 }
 module.exports = getLogger;
```

Ми створили один сценарій для логів - в консоль. Так само ми можемо окремо сортувати і зберігати повідомлення, наприклад, в базу даних або файл. Підключимо логгер в наш server.js:

server.js

```
var express = require('express');
var app = express();
var log = require('./libs/log')(module);

app.get('/api', function (req, res) {
   res.send('API is running');
});

app.listen(1337, function() {
   log.info('Express server listening on port 1337');
});
```

Наше інформаційне повідомлення тепер красиво окремо виводиться в консоль. Добавимо обробку помилок 404 і 500:

server.js

```
var express = require('express');
var app = express();
var log = require('./libs/log')(module);

app.get('/api', function (req, res) {
    res.send('API is running');
});

app.get('/ErrorExample', function(req, res, next) {
    next(new Error('Random error!'));
});

app.use(function(req, res, next) {
    res.status(404);
    log.debug('Not found URL: ' + req.url);
    res.send({ error: 'Not found' });
    return next();
});

app.use(function(err, req, res, next){
    res.status(err.status || 500);
    log.error('Internal error(' + res.statusCode + '): ' + err.message);
    res.send({ error: err.message });
    return next();
});
```

```
app.listen(1337, function() {
    log.info('Express server listening on port 1337');
});
```

Тепер, якщо немає доступних шляхів, Express поверне наше повідомлення. При внутрішній помилці додатку спрацює так само наш обробник, це можна перевірити, звернувшись за адресою «http://localhost:1337/ErrorExample».

4) RESTful API endpoints, CRUD

Додаємо шляхи для обробки деяких «статей» (articles). Логікою їх наповнювати поки не будемо, зробимо це в наступній практичній, з підключенням бази даних.

```
app.get('/api/articles', function(req, res) {
    res.send('This is not implemented now');
});

app.post('/api/articles', function(req, res) {
    res.send('This is not implemented now');
});

app.get('/api/articles/:id', function(req, res) {
    res.send('This is not implemented now');
});

app.put('/api/articles/:id', function (req, res) {
    res.send('This is not implemented now');
});

app.delete('/api/articles/:id', function (req, res) {
    res.send('This is not implemented now');
});
```

Для тестування «post / put / delete» пораджу чудову обгортку над URL – ***Postmen***.

Обов’язковим завданням практичної написати звіт, прочитати та вивчити основні моменти з статі:

https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/