УДК 004

Пример развёртывания веб-сервера на основе языка программирования Node.js

Якимов Антон Сергеевич
Приамурский государственный университет им. Шолом-Алейхема
Магистрант

Пасюков Александр Андреевич
Приамурский государственный университет им. Шолом-Алейхема
Магистрант

Николаев Сергей Валерьевич
Приамурский государственный университет им. Шолом-Алейхема
Магистрант

Баженов Руслан Иванович
Приамурский государственный университет им. Шолом-Алейхема
к. п. н., доцент, зав. кафедрой информационных систем, математики и методик обучения

Аннотация
Во время новых технологий все чаще используют веб-ресурсы для хранения каких-либо сайтов или баз данных. В связи с этим, встает вопрос в организации функционального сервера, который сможет выдерживать большие нагрузки и большой объем информации.
Ключевые слова: Веб-сервер, Linux, Node.js, JavaScript.

An example of deploying a web server based on the programming language Node.js

Yakimov Anton Sergeevich
Sholom-Aleichem Priamursky State University
Master student

Pasyukov Alexandr Andreevich
Sholom-Aleichem Priamursky State University
Master student

Nikolaev Sergey Valerievich
Sholom-Aleichem Priamursky State University
Master student

Bazhenov Ruslan Ivanovich
Sholom-Aleichem Priamursky State University
Candidate of pedagogical sciences, associate professor, Head of the Department of Information Systems, Mathematics and teaching methods

Abstract
During new technologies, more and more web resources are used to store any sites or databases. In this connection, the question arises in the organization of a functional server that can withstand heavy loads and a large amount of information.
Keywords: Web server, Linux, Node.js, JavaScript.

Веб-сервер – это своего рода программное обеспечение, которое позволяет принимать HTTP-запросы от клиентов, затем он обрабатывает эти запросы и выдает им обратно, в виде HTTP-ответов с нужными ресурсами. В качестве ресурса могут быть html-страницы, видео-потоки, аудио-потоки, изображения или другие ресурсы, которые нужны клиенту. Существует множество систем для организации веб-сервером. Таким образом, в данной статье было решено использовать Node.js для создания стабильного и функционального веб-сервера.

Многие русские и зарубежные ученные занимались данной проблемой. А.С.Якимов и Р.И. Баженов [1] описали установку и настройку веб-сервера с MySQL и PHP7 на Ubuntu 16.04. М. Ц.Эрдынеев и А. В. Гвоздев [2] описали разработку веб-сервера на Nginx. Произвели разработку системы анализа и блокировки запросов к веб-серверу на языке программирования PHP и описали ее работу Е. В.Пальчевский, А. Р. Халиков [3]. И. В. Ананченко и Е. К. Гришечко [4] описали разработку клиент-серверных приложений, работающих в облачных средах. A. Martínez-Álvarez [5] описал настройку компиляции веб-сервера на платформе apache. Р. А. Рубан [6] произвел исследование веб-сервера на устойчивость к повышениям нагрузок и описал результаты.

Node.js – это серверная JavaScript-платформа, предназначенная для создания масштабируемых распределенных сетевых приложений, использующая событийно-ориентированную архитектуру и неблокирующее асинхронное взаимодействие. Папка Node.js хранится на операционной системе Linux по пути /var/www/html/, где и хранится веб-сервер. Далее создаём в ней папку с именем «routing», где хранится index.js и все папки для работы сервера.

Первым делом создаём в папке сервера файл server.js, с помощью которого код будет запускать сервер. На рисунке 1 изображен код файла server.js.

```
const http = require('http');
const routing = require('./routing');


let server = new http.Server(function(req, res) {
  var jsonString = '';
  res.setHeader('Content-Type', 'application/json');
  req.on('data', (data) => {
      jsonString += data;
  });

  req.on('end', () => {
      routing.define(req, res, jsonString);
  });
});
server.listen(8000, 'localhost');
```

Рисунок 1. Код файла server.js

Для того чтобы принять запросы, нам нужно добавить код в файл routing/index.js. На рисунке 2 отображен код файла index.js.

```
const define = function(req, res, postData) {
  res.end('Hello, world!');
}
exports.define = define;
```

Рисунок 2. Код файла index.js

После этого стоит проверить, имеется ли данная страница. Для этого переходим в `/routing/nopage` и создаем файл `index.html` для отображения неизвестной страницы и снова переходим в файл index.js для написания блока catch. Данный фрагмент работает таким образом, что проверяет, если имеется введенная страница, то направляет на нее, а если данная страница отсутствует, то выводит ошибку 404. На рисунке 3 изображен блок catch.

```
catch (err)
{
    let filePath = prePath+'/static'+path+'/index.html';

    fs.readFile(filePath, 'utf-8', (err, html) => {
        if(err) {
            let nopath = '/var/www/html/nodejs/routing/nopage/index.html';
            fs.readFile(nopath, (err , html) => {
            if(!err) {
                res.writeHead(404, {'Content-Type': 'text/html'});
                res.end(html);
            }
            else{
                let text = "Not found";
                res.writeHead(404, {'Content-Type': 'text/plain'});
                res.end(text);
            }
            });
        }
        else{
            res.writeHead(200, {'Content-Type': 'text/html'});
            res.end(html);
        }
    });
}
```

Рисунок 3. Блок catch

После данных действий можно уже использовать сервер для наших целей, и он будет по запросу возвращать страницы. Проблема в том, что если поместить код стиля в папку со страницей, то стиль не применится с помощью стандартного кода «href=«style.css». Поэтому потребуется писать код иначе, а именно – «href=«/routing/nopage/style.css». Даже если правильно подключить стиль, то все равно ничего не будет и отобразится голая страница. Для того что бы это исправить, требуется написать скрипт, где сможет получать и обрабатывать запросы, читая разметку страницы. На рисунке 4 отображен скрипт подключения стилей.

```
if(/\./.test(path)) {
      if(path == 'favicon.ico') {
        let readStream = fs.createReadStream(prePath+path);
        readStream.pipe(res);
        return;
      }
      else{
            if(/\.mp3$/gi.test(path)) {
                res.writeHead(200, {
                    'Content-Type': 'audio/mpeg'
                });
                }
                else if(/\.css$/gi.test(path)) {
                res.writeHead(200, {
                    'Content-Type': 'text/css'
                });
                }
                else if(/\.js$/gi.test(path)) {
                res.writeHead(200, {
                    'Content-Type': 'application/javascript'
                });
                }
                 let readStream = fs.createReadStream(prePath+path);
                readStream.pipe(res);
                return;
      }
    }
```

Рисунок 4. Скрипт подключения стилей

Таким образом, в ходе работы был развернут веб-сервер на серверном языке программирования Node.js. Сервер тестировался при сильной нагрузке и проявил себя с положительной стороны. На данном сервере был развернут веб-сайт и показал стабильную бесперебойную работу. Так же планируется в дальнейшем развернуть сервер базы данных.

Библиографический список

1. Якимов А.С., Баженов Р.И. Настройка веб-сервера в связке с MySQL и PHP7 на Ubuntu 16.04 // Постулат. 2017. №1. URL: http://epostulat.ru/index.php/Postulat/article/view/368/387
2. Эрдынеев М. Ц., Гвоздев А. В. Nginx-веб-сервер для высоконагруженных проектов //Аллея науки. 2017. Т. 1. №. 10. С. 763-769.
3. Пальчевский Е. В., Халиков А. Р. Разработка системы анализа и блокировки запросов к веб-серверу на языке программирования PHP //Составители: Научно-издательский центр «Мир науки». 2017. С. 80-85.
4. Ананченко И. В., Гришечко Е. К. Разработка клиент-серверных приложений, работающих в облачных средах //Успехи современной науки и образования. 2016. Т. 2. №. 4. С. 124-126.
5. Martínez-Álvarez A. et al. Tuning compilations by multi-objective

optimization: Application to apache web server //Applied Soft Computing. 2015. Т. 29. С. 461-470.
6. Рубан Р. А. Исследование веб-сервера, устойчивость к повышениям нагрузок //Научное сообщество студентов. 2016. С. 130-136.