Node.js

Node.js

Приветствовать пользователя

```
const http = require('http');
const server = new http.Server();

server.on('request', (req, res) => {
    res.setHeader('content-type', 'text/html');

    res.write('Hello, <b>User!</b>');

    res.end();
});

server.listen(8080);
```

Node.js

На несуществующие страницы отвечать с кодом 404

```
server.on('request', (req, res) => {
    if (req.url === '/') {
        res.setHeader('Content-Type', 'text/html');
        res.end('Hello, <b>User!</b>');
    } else {
        res.statusCode = 404;
        res.setHeader('Content-Type', 'text/plain');
        res.end('Not Found');
    }
});
```

Node.js

Приветствовать пользователя по имени (/name/sergey)

```
server.on('request', (req, res) => {
    const matches = req.url.match(/^\/name\/([a-z]+)/);
    const name = matches && matches[1];

    if (name) {
        res.setHeader('Content-Type', 'text/html');
        res.end(`Hello, <b>${name}</b>`);
    } else {
        res.statusCode = 404;
        res.setHeader('Content-Type', 'text/plain');
        res.end('Not Found');
```

Express.js

Express.js

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

app.get('/name/:name', (req, res) => {
    res.send(`Hello, <b>${req.params.name}</b>`);
});

app.all('*', (req, res) => {
    res.sendStatus(404);
});

app.listen(8080);
```

Наиболее популярное решение

Много готовых расширений

Поддержка со стороны Node.js Foundation

Сервис «Заметки»

```
 GET /                Главная страница

 GET /notes           Список заметок

POST /notes           Добавление заметки

 GET /notes/films     Просмотр заметки по имени
```

Маршрутизация запросов

Маршрутизация запросов

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

// GET /
app.get('/', () => {});

// GET /notes
app.get('/notes', () => {});

// POST /notes
app.post('/notes', () => {});
```

Маршрутизация запросов

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

// GET /notes
// POST /notes
app
    .route('/notes')
    .get(() => {})
    .post(() => {})
```

Маршрутизация запросов

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

// GET /notes/books
app.get('/notes/:name', (req, res) => {
    const name = req.params.name;

    res.send(`Note with "${name}" name`);
});
```

?

```
// GET /notes
// GET /note
app.get('/notes?', () => {});
```

+

```
// GET /notes
// GET /notesss
app.get('/notes+', () => {});
```

[]

```
// GET /notes/films
// GET /notes/books

// GET /notes/my-books
app.get('/notes/[a-z]+', () => {});
```

()

```
// GET /notes/films
// GET /notes/films/westerns
app.get('/notes/:name([a-z]+)', req => {
    console.log(req.params.name); // films
});
```

404

```
app.get('*', (req, res) => {
    res.sendStatus(404)
});

app.post('*', (req, res) => {
    res.sendStatus(404)
});
```

404

```
app.all('*', (req, res) => {
    res.sendStatus(404)
});
```

```
app/
└── index.js
└── routes.js
```

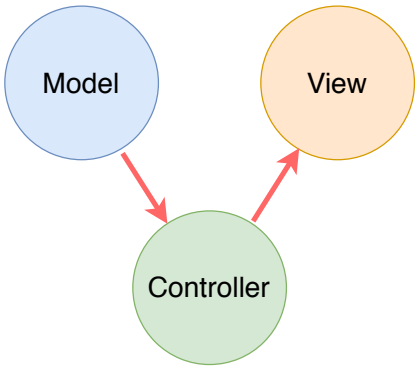
```
module.exports = app => {
    app.get('/', (req, res) => {});
};
```

```
app/
└── index.js
└── routes.js
```

```
require('./routes')(app);
```

express-route-tester

Разделение ответственности


Model
View
Controller

```
app/
└── index.js
└── routes.js
└── models
    └── note.js
```

```
class Note {
    constructor({ name }) {
        this.name = name;
    }

    save() {}

    static find(name) {}

    static findAll() {}
}
```

```
app/
└── index.js
└── routes.js
└── models

└── controllers
    └── notes.js
```

```
const Note = require('../models/note');

exports.item = (req, res) => {
    const note = Note.find(req.params.name);
}
```

```
app/
└── index.js
└── routes.js
└── models

└── controllers
    └── notes.js
```

```
const Note = require('../models/note');

// ...

exports.list = (req, res) => {
    const notes = Note.findAll();
};
```

```
app/
└── index.js
└── routes.js
└── models

└── controllers
    └── notes.js
```

```
const Note = require('../models/note');

// ...

exports.create = (req, res) => {
    const note = new Note({ name: req.body.name });

    note.save();
```

```
app/
└── index.js
└── routes.js
└── models
└── controllers
    └── notes.js
    └── errors.js
```

```
exports.error404 = (req, res) => res.sendStatus(404);
```

```
app/
└── index.js
└── routes.js
└── models
└── controllers
```

```
const { error404 } = require('./controllers/errors');
const { create, item, list } = require('./controllers/notes');

module.exports = app => {
    app.get('/', list);
    app.get('/notes', list);
    app.get('/notes/:name', item);
    app.post('/notes', create);
    app.all('*', error404)
};
```

Шаблонизация

Шаблон + Данные = HTML

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>My notes</title>
</head>
<body>
    Hello, User!
</body>
</html>
```

Данные

```
{
    title: 'My notes',

    message: 'Hello, User!'
}
```

Шаблон

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>{{ title }}</title>
</head>
<body>
    {{ message }}
</body>
</html>
```

1. Компиляция

```
function index(data) {
    return `<!DOCTYPE html>
    <html lang="en">
    <head>
    <meta charset="utf-8">
    <title>${data.title}</title>
    </head>
    <body>${data.message}</body>
    </html>`;
}
```

2. Исполнение

```
index({
    title: 'My notes',

    message: 'Hello, User!'
})
```

Handlebars

```
app/
└── index.js
└── routes.js
└── models
└── controllers
```

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

app.set('view engine', 'hbs');
require('./routes')(app);

app.listen(8080);
```

```
app/
└── index.js
└── routes.js
└── models
└── controllers
└── views
    └── index.hbs
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>{{ title }}</title>
</head>
<body>{{ message }}</body>
</html>
```

```
app/
└── index.js
└── routes.js
└── models
└── controllers
    └── index.js
└── views
    └── index.hbs
```

```
exports.index = (req, res) => {
    res.send('Hello, User!');
    res.render('index', {
        title: 'Awesome notes',
        message: '<h1>Hello, User!</h1>'
    });
}
```

{{ }}

```
<div>{{ message }}</div>

<!-- <div>&lt;h1&gt;Hello, User!&lt;/h1&gt;</div> -->
```

{{{ }}}

```
<div>{{{ message }}}</div>

<!-- <div><h1>Hello, User!</h1></div> -->
```

{{ meta.description }}

```
res.render('index', {
    title: 'Awesome notes',
    meta: {
        description: 'My awesome notes',
        charset: 'utf-8'
    }
});
```

```
<head>
    <meta charset="{{ meta.charset }}">
    <meta name="description"
          content="{{ meta.description }}">
    <title>{{ title }}</title>
</head>
```

{{#each}}

```
res.render('notes', {
    notes: [
        { name: 'Films' },
        { name: 'Books' },
        { name: 'Todo' }
    ]
});

<ul>
    {{#each notes}}
    <li>{{ name }}</li>
    {{/each}}
</ul>
```

{{#each}}

```
res.render('notes', {
    notes: [
        { name: 'Films' },
        { name: 'Books' },
        { name: 'Todo' }
    ]
});
```

```
{{#each notes}}
<p>{{ @index }}. {{ name }}</p>
{{/each}}
```

{{ ../ }}

```
res.render('notes', {
    prefix: 'My',
    notes: [
        { name: 'Films' },
        { name: 'Books' },
        { name: 'Todo' }
    ]
});
```

```
{{#each notes}}
<p>{{ ../prefix }}. {{ name }}</p>
{{/each}}
```

{{#if }}

```
{{#if notes.length}}
<ul>
    {{#each notes}}
    <li>{{ name }}</li>
    {{/each}}
</ul>
{{^}}
<p>Notes not found!</p>
{{/if}}
```

188 handlebars helpers

```
app/
└── index.js
└── routes.js
└── models
└── controllers
└── views
    └── index.hbs
    └── partials
        └── notes.hbs
```

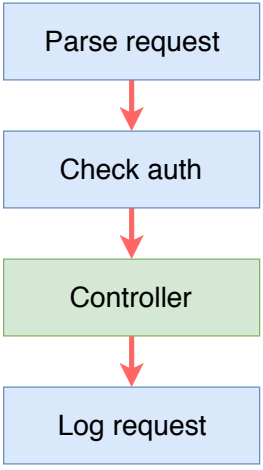
```
app.set('view engine', 'hbs');
hbs.registerPartials(__dirname + '/views/partials', () => {
    app.listen(8000);
});
```

{{> }}

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>{{ title }}</title>
</head>
<body>
    {{> notes}}
</body>
</html>
```

handlebarsjs.com

Цепочки ответственностей


Parse request
Check auth
Controller
Log request

Запрос последовательно проходит через обработчики в цепочке

Обработчик сам принимает решение, обрабатывать запрос или пропустить дальше

Sync middleware

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


// Логирует все запросы
app.use((req, res, next) => {
    console.log(`→ ${req.method} ${req.originalUrl}`);

    next();
});
```

Async middleware

```
const express = require('express');
const app = express();
const fetch = require('node-fetch');

// Получаем какие-либо общие данные из api
app.use((req, res, next) => {
    fetch('https://github.com/api/')
        .then(data => {
            // locals - рекомендованное поля для таких данных
            res.locals.data = data;
            next();
        })
        .catch(err => next(err));
});
```

morgan

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

app.use(morgan('dev'));
```

```
GET /notes 200 18.079 ms - 301
|       |    |      |      |
method  url  code   time   bytes
```

body-parser

```
POST /notes
Accept: application/json
Host: localhost

{
    "name": "films",
    "text": "Films to watch"
}
```

Без body-parser

```
let body = '';

req.on('data', chunk => {
    body += chunk;
});

JSON.parse(body);
```

C body-parser

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();

app.use(bodyParser.json());

app.use(() => console.log(req.body));
```

express.static

```
app/
├── index.js
├── routes.js
├── models
├── controllers
├── views
└── public
    ├── favicon.ico
    └── styles.css
```

```
// GET /styles.css
app.use(express.static(__dirname + '/public'));
```

Обработка ошибок

Обработка ошибок

```
app.use((err, req, res, next) => {
    console.error(err);

    res.sendStatus(500);
});
```

Обработчик ошибок отличается количеством аргументов – 4

Обработчиков ошибок может быть несколько в цепочки в разных местах

Если в обычном обработчике произошла ошибка, все последующие игнорируются вплоть до первого обработчика ошибок

Обработчик ошибок может не прерывать процесс, а передать управление следующей middleware (обычному обработчику)

Обработка ошибок

```
app.use(bodyParser.json());



app.use((err, req, res, next) => {
    // Выводим ошибку
    console.error(err);

    // Но продолжаем обрабатывать запрос
    next();
});

app.use((req, res, next) => {
    // ...
});
```

Конфигурация

Секретные настройки спускаем в приложение через переменные окружения

```
$ DBPASS=p@ssw0rd node index.js
```

Остальные настройки храним в файлах, разделённых по окружениям – локальное, тестовое, боевое

Окружение определяем переменной
NODE_ENV=production

```
app/
└── index.js
└── routes.js
└── models
└── controllers
└── views
└── public
└── config
    └── default.js
    └── productions.js
```

```
module.exports = {
    debug: false
}
```

```
app/
└── index.js
└── routes.js
└── models
└── controllers
└── views
└── public
└── config
    └── default.js
    └── productions.js
```

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

if (config.get('debug')) {
    console.log('Some debug notes');
}
```

```
$ NODE_ENV=production node index.js
```

Express generator

```
$ npm install --global express-generator

$ express --view=hbs awesome-notes
```

```
create : awesome-notes
create : awesome-notes/package.json
create : awesome-notes/app.js
...

install dependencies:
    $ cd awesome-notes && npm install

run the app:
    $ DEBUG=awesome-notes:* npm start
```

Исходный код «Заметок»