Порядок действий:

1 видео Настройка окружения и установка пакетов <https://www.youtube.com/watch?v=fpgaVF9iLh8&list=PL_RVw8KXnKnP_8vWPmXq3Vrnj25tkf-5p>

1. Создать папку и перейти в нее
2. Инициализировать git git init
3. Инициализировать пакетный менеджер npm npm init
4. Настроим eslint (служит для поиска ошибок в коде до запуска интерпретатора)

Ссылка на описание пакета <https://www.npmjs.com/package/eslint-plugin-node>

Устанавливаем командой npm install --save-dev eslint eslint-plugin-node

1. Создаем файл .eslintrc и прописываем в файл (?)

{

*"plugins"*: ["node"],

*"extends"*: ["eslint:recommended", "plugin:node/recommended"],

*"rules"*: {

*"node/exports-style"*: ["error", "module.exports"] ,

*"no-console"*: 0

    }

}

1. В package.json прописать в конце файла после запятой

*"engines"*: {

*"node"*: ">=6.0.0"

   }

1. Чтобы eslint работал в VSCode необходимо установить в VSCode плагин ESLint автор Dirk Baeumer

После этого пр наведении на, например, переменную в hint будет подсвечиваться что не так (например, чир переменная нигде не используется)

1. Чтобы устанавливаемые модули не попадали в git создаем gitignore

Создаем файл .gitignore и прописываем в нем node\_modules

1. Устанавливаем express (<https://www.npmjs.com/package/express>) npm i express
2. Прописываем для проверки работы в файл index.js пример для express:

*const* express = require('express')

*const* app = express()

app.get('/', *function* (*req*, *res*) {

    res.send('Hello World')

})

app.listen(3000, () *=>* *console*.log("Example app listening on port 3000"));

Запускаем файл index.js командой node index

В браузере в строке адреса набираем localhost:3000

Выведется Hello Wourld

1. Для того, чтобы при изменении в файлах не перезапускать сервер для отображения изменений используется пакет nodemon

Устанавливаем nodemon командой npm i --save-dev nodemon

1. Для работы nodemon необходимо создать секцию scripts:

*"scripts"*: {

*"start"*: "node index.js",

*"dev"*: "nodemon inde.js"

  }

1. Для использования nodemon в строке вместо node index пишется npm run dev

Теперь при сохранении изменений в файле изменения применятся на сервере, необходимо только перезагрузить страницу браузера

Видео 2 Работа с Post и Get запросами, шаблонизатор ejs

<https://www.youtube.com/watch?v=wsj3YZVohMw&list=PL_RVw8KXnKnP_8vWPmXq3Vrnj25tkf-5p&index=2>

1. Установим шаблонизатор EJS (<https://www.npmjs.com/package/ejs>) командой

npm I --save-dev ejs

Шаблонизатор позволяет создавать HTML код из записей, заключенных в спец. символы

1. Указываем в index.js, что будет использоваться шаблонизатор

*const* express = require('express')

*const* app = express()

app.set('view engine', 'ejs');

app.get('/', *function* (*req*, *res*) {

    res.send('Hello World!')

})

app.listen(3000, () *=>* *console*.log("Example app listening on port 3000"));

1. По-умолчанию все шаблоны будут искаться в папке views. Создадим папку views.
2. В папке views создадим файл index.ejs. VSCode работает с этим файлом как с обычным HTML файлом. Для проверки что-нибудь написать в файл.
3. В index.js изменим res.send на res.render

*const* express = require('express')

*const* app = express()

app.set('view engine', 'ejs');

app.get('/', *function* (*req*, *res*) {

    res.render('index');

})

app.listen(3000, () *=>* *console*.log("Example app listening on port 3000"));

1. Можно проверить работу запустив сервер командой npm run dev и перейдя в браузере на страницу localhost:3000
2. Передаваемые данные указываются в рендере вторым аргументом. Изменяем файл index.js

*const* express = require('express')

*const* app = express()

app.set('view engine', 'ejs');

*const* data = 'Hi!';

app.get('/', *function* (*req*, *res*) {

    res.render('index', {data: data});

})

app.listen(3000, () *=>* *console*.log("Example app listening on port 3000"));

1. Изменим файл index.ejs:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

    <%= data %>

</body>

</html>

1. Проверим работу, запустив сервер – должно вывестись Hi!
2. Get запрос передается в адресной строке, Post запрос передает невидимый обьект

Пример работы с POST запросами

req - обьект запроса (здесь будут содержаться отправленные данные из текстового поля)

res – обьект ответа

1. Создадим файл create.ejs и создадим простую форму для отправки Post запроса

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

    <form method="POST">

        <input type="text" name="text">

        <button type="submit">Submit</button>

    </form>

</body>

</html>

1. Изменим файл index.ejs. Вместо вывода данных поставим ссылку на страницу с формой

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

    <a href="/create">Add</a>

</body>

</html>

1. Для указанного пути (страницы с формой) прописываем обработчик в index.js

Добавляем в файл

app.get('/create', *function* (*req*, *res*) {

    res.render('create');

})

Где

/create – путь к странице, который вбивается в браузере

create – это шаблон create.ejs в папке views

1. Для того, чтобы взять требуемую информацию из полей используется пакет body-parser (<https://www.npmjs.com/package/body-parser>)

Установим пакет командой npm i --save-dev body-parser

1. Изменим файл index.ejs

Из примеров по ejs допишем итерацию из массива

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

    <a href="/create">Add</a>

    <ul>

        <% arr.forEach((item)=>{ %>

        <li>

            <%= item %>

        </li>

        <% }); %>

    </ul>

</body>

</html>

1. Подключаем пакет в index.js, указываем приложению, что будем его использовать и прописываем обработчик Post запроса

*const* express = require('express');

*const* bodyParser = require('body-parser');

*const* app = express()

app.set('view engine', 'ejs');

app.use(bodyParser.urlencoded({

    extended: true

}));

*const* arr = ['Hi!', 'Hello'];

app.get('/', *function* (*req*, *res*) {

    res.render('index', {

        arr: arr

    });

})

app.get('/create', *function* (*req*, *res*) {

    res.render('create');

})

app.post('/create', (*req*, *res*) *=>* {

    arr.push(req.body.text);    //добавление в массив arr элемента с текстом из запроса res

    res.redirect('/');          //переход на главную страницу

})

app.listen(3000, () *=>* *console*.log("Example app listening on port 3000"));

1. Можно для VSCode посавить плагин Prettier для одинакового форматирования кода (у меня уже есть другой плагин)

Видео 3 Создание конфигурационного файла и подключение MongoDB

<https://www.youtube.com/watch?v=LNw0c0oOf9U&list=PL_RVw8KXnKnP_8vWPmXq3Vrnj25tkf-5p&index=3>

1. Установим пакет cross-env (<https://www.npmjs.com/package/cross-env> )для работы с переменным окружением (чтобы иметь возможность запускать сервер с параметрами, указанными до старта) npm i --save-dev
2. Меняем файл package.json

*"scripts"*: {

*"test"*: "echo \"Error: no test specified\" && exit 1",

*"start"*: "cross-env PORT=3001 node index.js",

*"dev"*: "nodemon inde.js"

  },

Теперь если запустить сервер командой npm run start, то он запустится на порту 3001

1. Создаем конфигурационный файл для хранения настроек (номера порта и др) config.js

*module*.*exports* = {

    PORT: process.env.PORT || 3000

}

1. Изменяем файл index.js – задействуем данные из конфигурационного файла

*const* express = require('express');

*const* bodyParser = require('body-parser');

*const* config = require('./config');

*const* app = express()

app.set('view engine', 'ejs');

app.use(bodyParser.urlencoded({

    extended: true

}));

*const* arr = ['Hi!', 'Hello'];

app.get('/', *function* (*req*, *res*) {

    res.render('index', {

        arr: arr

    });

})

app.get('/create', *function* (*req*, *res*) {

    res.render('create');

})

app.post('/create', (*req*, *res*) *=>* {

    arr.push(req.body.text); //добавление в массив arr элемента с текстом из запроса res

    res.redirect('/'); //переход на главную страницу

})

app.listen(config.PORT, () *=>* *console*.log(`Example app listening on port ${config.PORT}`));

## Установка MongoDB

1. Порядок установки mongoDB <https://metanit.com/nosql/mongodb/1.2.php>
2. Скачать архив с сайта <https://www.mongodb.com/try/download/community>
3. Распаковать файлы архива в папку C:\mongodb
4. Установка приложения для визуальной работы с БД – сайт <https://robomongo.org/>
5. Создаем файл настроек mongoose

Создать файл database.js и подключить к нему конфигурационный файл config.js

*const* config = require('./config');

1. Устанавливаем пакет ОРМ - mongoose командой npm i --save-dev mongoose
2. Подключаем в database.js mongoose

*const* config = require('./config');

*const* mongoose = require('mongoose');

*module*.*exports* = () *=>* {

    return new Promise((*resolve*, *reject*) *=>* {

        mongoose.Promise = global.Promise;

        mongoose.set('debug', true);

        mongoose.connection

            .on('error', *error* *=>* reject(error))

            .on('close', () *=>* *console*.log("Database connection closed"))

            .on('open', () *=>* resolve(mongoose.connections[0]));

        mongoose.connect(config.MONGO\_URL, {

            useMongoClient: true

        })

    })

}

1. Изменяем конфигурационный файл config.js

*module*.*exports* = {

    PORT: process.env.PORT || 3000

    MONGO\_URL: 'mongodb://localhost/blog'

}

## Создаем общий файл app.js

1. Создаем файл app.js
2. Переносим в него из index.js все кроме прослушивания порта и прописываем экспорт:

*const* express = require('express');

*const* bodyParser = require('body-parser');

*const* app = express()

app.set('view engine', 'ejs');

app.use(bodyParser.urlencoded({

    extended: true

}));

*const* arr = ['Hi!', 'Hello'];

app.get('/', *function* (*req*, *res*) {

    res.render('index', {

        arr: arr

    });

})

app.get('/create', *function* (*req*, *res*) {

    res.render('create');

})

app.post('/create', (*req*, *res*) *=>* {

    arr.push(req.body.text); //добавление в массив arr элемента с текстом из запроса res

    res.redirect('/'); //переход на главную страницу

})

*module*.*exports* = app;

1. Изменяем файл index.js

*const* app = require('./app');

*const* database = require('./database');

*const* config = require('./config');

database().then((*info*) *=>* {

*console*.log(`connect to ${info.host}:${info.port}/${info.name}`);

    app.listen(config.PORT, () *=>* *console*.log(`Example app listening on port ${config.PORT}`));

}).catch(() *=>* {

*console*.error('Unable connect to database');

    process.exit(1); //Выйдет из node

});