

Шаблонизатор Handlebars

К примеру, нам нужно создать блог. Будет удобно хранить шаблон - *представление*, и вставлять в него пост. Это логику реализуют *шаблонизаторы*.

Управлять шаблонами позволяют *движки представлений*. Реализаций движков много, одна из самых основных - это Handlebars. Установим ее:

npm i handlebars

Создадим в проекте папку views, в который положим файл contact.hbs:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>{{title}}</title>

</head>

<body>

<h1>Welcome on Web Site</h1>

</body>

</html>

Здесь {{title}} это свойство, которое нужно подставить.

const express = require('express')

const hbs = require('hbs')

const app = express()

// устанавливаем handlebars, как движок

app.set('view engine', 'hbs')

app.get ("/", (req, res) => {

// устанавливаем значения и выдаем файл

res.render ('home.hbs', {

title: 'Home'

})

})

app.listen (3000)

Запускаем и видим, что значение подставилось.

**Циклы и условия**

Для вывода значений часто нужны циклы и условия. Пример, когда они используются:

<!DOCTYPE html>

<html>

<head>

<title>{{title}}</title>

<meta charset="utf-8" />

</head>

<body>

<h1>{{title}}</h1>

{{#if visible}}

<ul>

{{#each links}}

<li>{{this}}</li>

{{/each}}

</ul>

{{else}}

<h2>Нет ссылок</h2>

{{/if}}

</body>

<html>

Рендерим ее:

const express = require('express')

const hbs = require('hbs')

const app = express()

app.set ('view engine', 'hbs')

app.get ('/contact', (req, res) => {

res.render ('contact.hbs', {

title: 'Контакты',

visible: false,

links: ['qwertyShura@gmail.com', '+7(915)748-59-71']

})

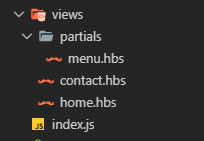
})

app.listen(3000)

**Частичные преставления**

У нас есть много повторяющихся элементов. Например, меню, footer и т.д. Их можно вынести в *частичные представления*.

Создадим в папке view папку partials, в нее будем класть все частичные представления



Сделаем простенькое меню:

<nav><a href="/">Главная</a> | <a href="/contact">Контакты</a></nav>

И в других файлах его можно использовать так:

{{> menu}}

Просто так использовать частичные представления нельзя, надо явно указать, где они находятся.

const app = express()

app.set ('view engine', 'hbs')

hbs.registerPartials (path.join (\_\_dirname, '/views', '/partials'))

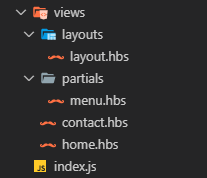
**Layout**

Механизм Layout позволяет создать один файл - макет и подставить содержимое файла. К примеру, выше у home и contact уникальные части лишь пару строк. Все что дублируется можно вынести в отдельный файл layout и содержать в home и contact эти уникальные строки и при выдаче обворачивать в layout.

Сначала нужно портировать эту возможность:

npm i express-handlebars

Потом в проекте создадим такую структуру:



В папке layouts содержится главный макет. Собственно самый главный макет:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>{{title}}</title>

</head>

<body>

{{> menu}}

{{{body}}}

</body>

</html>

Обратите внимание на то, что три фигурных скобки. Но еще нужно прописать настройки:

const express = require('express')

const expressHbs = require('express-handlebars')

const hbs = require('hbs')

const path = require('path')

const app = express()

app.engine ('hbs', expressHbs ({

layoutsDir: "views/layouts",

defaultLayout: "layout",

extname: ".hbs"

}))

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

hbs.registerPartials (path.join (\_\_dirname + 'views', 'partials'))

**Хелперы**

*Хелперы* – это js функции, вызвать, которые можно в шаблоне. Они могут возвращать просто строку или html. Если нужно вернуть html, то результат нужно обвернуть в new hbs.SafeString:

hbs.registerHelper ('makeList', (list) => {

return new hbs.SafeString (`<ul>

${list.reduce ((prev, val) => prev + `<li>${val}</li>`, '')}

</ul>`)

})

Вот так можно его вызвать:

{{makeList arr}}

**Итого**

В этой главе было рассказано про шаблоны. Зная материал этой главы можно делать сайт, особо не уходя во frontend. Но все-таки шаблонизаторы используются исключительно в MPA, которые редко пишут на Node JS. Поэтому следует посмотреть на Angular, когда будете делать frontend.