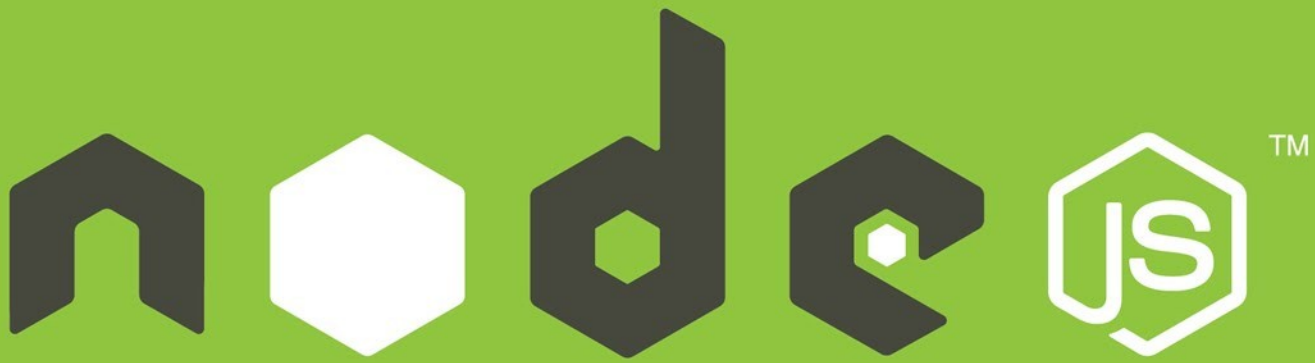
Express.js



Web-framework express.js

- Приложение express
- Маршрутизация
- Шаблонизация

1 | Что такое web-framework ?

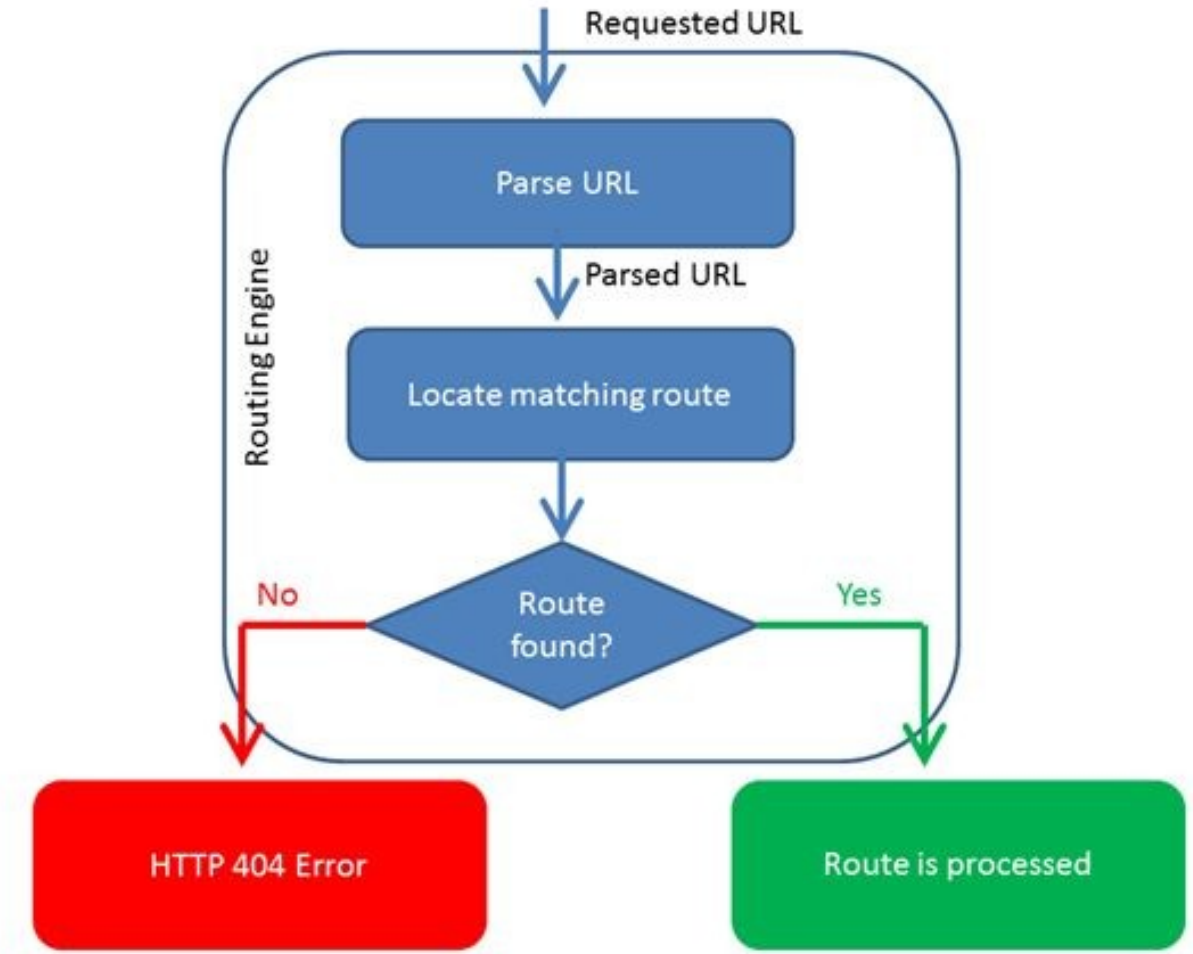
Под этим термином обычно понимают web application framework (WAF) - программный каркас веб-приложений. Обычно такой продукт содержит средства маршрутизации запросов, библиотеки для работы с источниками данных (с базами данных и/или NoSQL-хранилищами, шаблонизаторы, управление сеансами работы, генерацию форм, работу с почтой и MIME-типами).

1 | Приложение express

Простое приложение на express. Методе app.get() - реализует функцию прослушивания запросов.

```
var express = require('express');
var app = express();
var port = process.env.PORT || 3000;
app.get ('/', function ( req, res) {
        res.send('Hello, Express!')
})
app.listen(port);
console.log('Listening on port ' + port);
```

Разбор URL запроса



2 | Маршрутизация

Маршрутизация определяет, как приложение отвечает на клиентский запрос к конкретному адресу (конечной точке), которым является URI (или путь), и определенному методу запроса HTTP (GET, POST и т.д.).
Каждый маршрут может иметь одну или несколько функций обработки, которые выполняются при сопоставлении маршрута.

Определение маршрута имеет следующую структуру:

```
app.METHOD(PATH, HANDLER)
```

- APP - это экземпляр EXPRESS.
- METHOD – метод запроса HTTP.
- PATH - путь на сервере.
- HANDLER - функция, выполняемая при сопоставлении маршрута

2 | Пути маршрутов

Ответ на запрос GET в корневом маршруте (/), на домашней странице приложения:

```
app.get('/', function (req, res) {
  res.send('Hello World!');
});
```

Ответ на запрос POST в корневом маршруте (/), на дом. странице приложения:

```
app.post('/', function (req, res) {
  res.send('Got a POST request');
});
```

Ответ на запрос PUT, адресованный маршруту '/user'

```
app.put('/user', function (req, res) {
  res.send('Got a PUT request at /user');
});
```

Ответ на запрос DELETE, адресованный маршруту '/user'

```
app.delete('/user', function (req, res) {
  res.send('Got a DELETE request at /user');
});
```

2 | Типы маршрутов

Данный путь маршрута сопоставляет запросы с /about.

```
app.get('/about', function (req, res) {
  res.send('about');
});
```

Приведенный ниже путь маршрута сопоставляет acd **и** abcd.

```
app.get('/path/ab?cd', function(req, res) {
  res.send('ab?cd');
});
```

Этот путь маршрута сопоставляет /path/x/100

```
app.get('/path/:key/:value', function(req, res) {
  res.send('ab?cd');
});
```

Этот путь маршрута сопоставляет abcd, abbcd, abbbcd **и т.д.**

```
app.get('/ab+cd', function(req, res) {
  res.send('ab+cd');
});
```

Этот путь маршрута сопоставляет abcd, abxcd, abRABDOMcd, ab123cd **и т.д.**

```
app.get('/ab*cd', function(req, res) {
  res.send('ab*cd');
});
```

2 | Типы маршрутов

Символы ?, +, * и () представляют собой подмножества соответствующих им регулярных выражений

***** - Обозначает повторение 0 или более раз. Например, /bo*/ найдет 'boooo' в "A ghost booooed" и 'b' в "A bird warbled", но ничего не найдет в "A goat grunted".

+ - Обозначает повторение 1 или более раз. Эквивалентно {1,}. Например, /a+/ найдет 'a' в "candy" и все 'a' в "caaaaaaandy".

? - Обозначает, что элемент может как присутствовать, так и отсутствовать. Например, /e?le?/ найдет 'el' в "angel" и 'le' в "angle.

2 | Методы ответа

Перечисленные в таблице методы ответа, могут передавать ответ клиенту и завершать цикл «запрос-ответ». Если ни один из этих методов не будет вызван из обработчика маршрута, клиентский запрос зависнет.

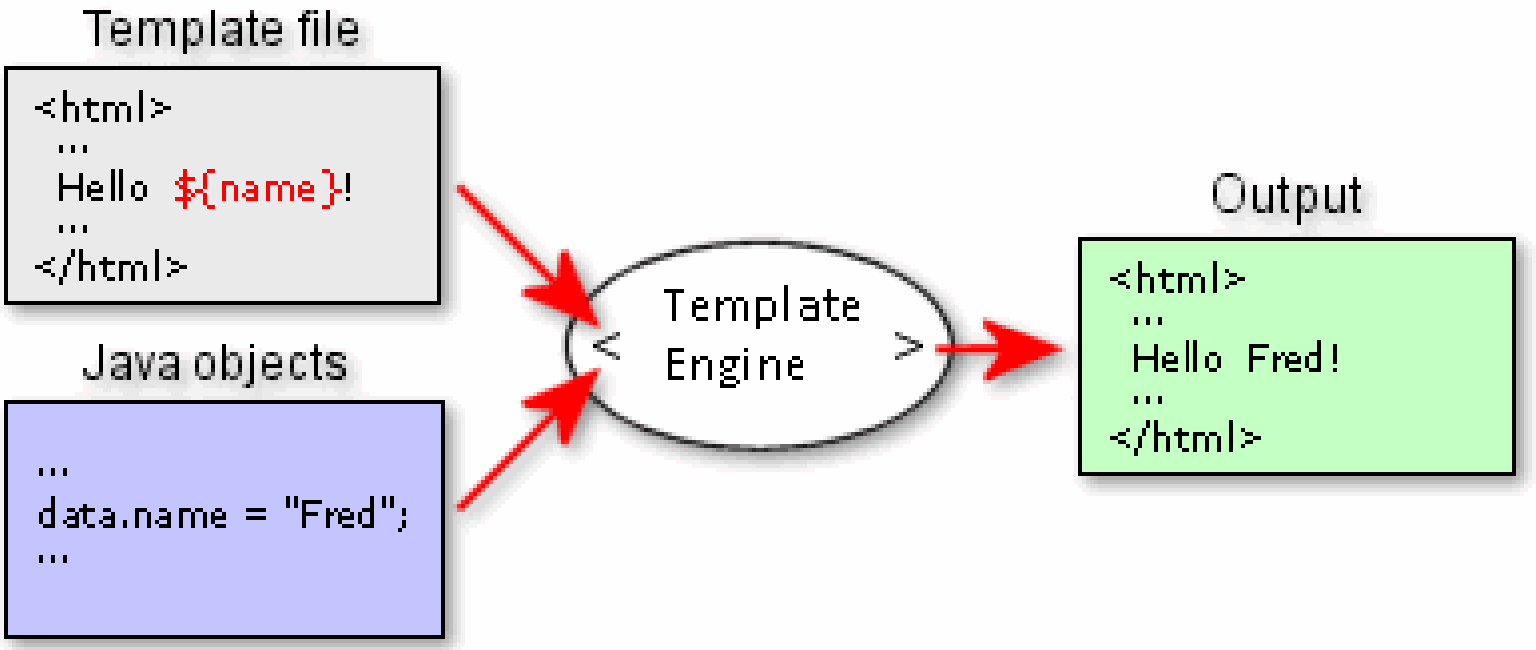
Метод	Описание
res.download()	Приглашение загрузки файла.
res.end()	Завершение процесса ответа.
res.json()	Отправка ответа JSON.
res.jsonp()	Отправка ответа JSON с поддержкой JSONP.
res.redirect()	Перенаправление ответа.
res.render()	Вывод шаблона представления.
res.send()	Отправка ответа различных типов.
res.sendFile	Отправка файла в виде потока октетов.
res.sendStatus()	Установка кода состояния ответа и отправка представления в виде строки

2 | Методы ответа

В теле ответа HTTP сервера может быть буфер объекта , строка , объект или массив.

```
res.send(new Buffer('whoop'));
res.send({ some: 'json' });
res.send('<p>some html</p>');
res.status(404).send('Sorry, we cannot find that!');
res.status(500).send({ error: 'something blew up' });
```

Шаблонизация



3 | Шаблонизатор mustache

Шаблонизация — подстановка конкретных значений в заданный шаблон. Основные теги для замены в шаблоне выглядят так {{ }}. В них содержится название поля json объекта значение которого будет подставляться в место фигурных скобок.

3 | Шаблонизатор mustache | подключение

```
var sys = require('sys');
var mustache = require('mustache');

var view = {
  title: "Joe",
  calc: function() {
    return 2 + 4;
  }
};

var template = "{{title}} spends {{calc}}";
var html = mustache.to_html(template, view);
console.log(html);
});
```

Полезный ссылки

http://expressjs.com/ru/ - express минималистичный веб фреймворк

http://websketches.ru/plugins/mustache - статья про шаблонизатор