Servidor HTTP con NodeJS

Gabriel Rodríguez Flores

November 4, 2021

ADA ITS
INSTITUTO TECNOLÓGICO SUPERIOR

- Teoría sobre servicios web
- Montar un servicio web nativo con NodeJS

Contents

1 Teoría

- Oficial
- Nodejs HTTP
- Tutorial completo y ordenado

1.1 Servidor Web

1.2 URL

La estructura de una URL es la siguiente: `protocolo://dominio:puerto/path?query`

1.2.1 Query String

```
// Parse a querystring into an object with query parameter key/value pairs
const str = 'prop1=value1&prop2=value2';
querystring.parse(str); // Returns { prop1: value1, prop2: value2}

// Build a querystring from an object of query parameters
const props = { "prop1": value1, "prop2": value2 };
querystring.stringify(props); // Returns 'prop1=value1&prop2=value2'

// Percent-encode a querystring
const str = 'prop1=foo[a]&prop2=baz[b]';
querystring.escape(str); // Returns 'prop1%3Dfoo%5Ba%5D%26prop2%3Dbaz%5Bb%5D'

// Decode a percent-encoded querystring
const str = 'prop1%3Dfoo%5Ba%5D%26prop2%3Dbaz%5Bb%5D';
querystring.unescape(str); // Returns 'prop1=foo[a]&prop2=baz[b]'
```

1.3 Protocolo HTTP

1.3.1 Códigos

- Lista de códigos

1.4 Crear servidor básico en NodeJS

- Referencia W3School

1. Necesitaremos importar el módulo nativo `http`

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

2. Crear la instancia del servidor

```
const server = http.createServer((req, res) => {
  res.end();
});
```

3. Arrancar el servidor definiendo el puerto de escucha

```
server.listen(3000, '127.0.0.1'); // Segundo parámetro (host) opcional
    si es localhost
```

1.4.1 Objetos REQ y RES

- Estructura de un objeto REQUEST en Nodejs

2 Ejemplos

- Ejemplo
- Proyecto complejo en Nodejs nativo
- Proyecto completo con dockerización
- Rest API Vanilla

2.1 Editar la respuesta de la petición

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

const server = http.createServer((req, res) => {
  res.write('Hello World!\n');
  res.end();
});

server.listen(3000);
```

Podemos escribir la respuesta directamente en el cierre:

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

const server = http.createServer((req, res) => {
```

```
  res.end('Hello World!\n');
});

server.listen(3000);
```

En lugar de crear el objeto servidor (que es recomendable hacerlo), podemos ponerlo a escuchar en el momento de la creación:

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

http.createServer((req, res) => {
  res.end('Hello World!\n');
}).listen(3000);
```

2.2 Editar la cabecera de la petición

En este ejemplo se crea un servidor en el puerto 3000 que responde con un Hello World! siempre que se le realice una petición. Se establece el código 200 para definir que el resultado de la petición es OK.

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

http.createServer((req, res) => {
    res.writeHead(200, {'Content-Type': 'text/plain'});

    res.end('Hello World!\n');
});listen(3000);
```

Otra forma de escribir el ejemplo anterior. Solo cambia la forma de construir la cabecera.

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

http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');

  res.end('Hello World!\n');
}).listen(3000);
```

2.3 Devolver un HTML como contenido (body)

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

http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/html');
```

```
6
7   res.end('<html><body><h1>Hello World!</h1></body></html>');
8 }).listen(3000);
```

2.4 Establecer rutas

```
1  const http = require('http');
2
3  http.createServer((req, res) => {
4    res.setHeader("Content-Type", "application/json");
5    switch (req.url) {
6        case "/ping":
7            res.writeHead(200);
8            res.end('pong');
9            break;
10   }
11 }).listen(3000);
```

Escribirlo sin `switch`

```
1  const http = require('http');
2
3  const routes = {
4    ping: () => {
5      res.writeHead(200);
6      res.end('pong');
7    }
8  }
9
10 http.createServer((req, res) => {
11   res.setHeader("Content-Type", "application/json");
12   routes[`/${req.url}`]();
13 }).listen(3000);
```

3 Ejercicios

1. Crear un servidor en NodeJS que devuelva *Hello World!* cuando se acceda al puerto 4000
2. Crear un servidor en NodeJS que devuelva una página web (puerto 3000)
3. Crear un servidor en NodeJS que tenga distintas páginas según la URL accedida.
4. Crear 2 rutas:
   - `/page`: Devuelve una pagina web
   - `/error`: Devuelve una pagina de error con el código 404
5. Devolver las páginas cargadas de un fichero `.html` en lugar de escritas en código.

6. Enviar el parámetro `name` por *querystring* y que devuelva `Hello ${name}!`
7. Ruta fizzbuzz con query string de un numero, y devuelve toda la secuencia de números hasta dicho numero.

4 Entregables

4.1 En clase

Ejercicios 1, 2 y 3

4.2 Tarea

Ejercicios 4, 5, 6 y 7