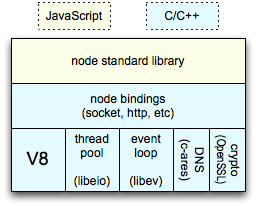
# Node JS

Es un entorno de ejecución Javascript multiplataforma, el entorno de ejecución interpreta y ejecuta el Javascript usando V8 (el motor de Javascript de Google).

Node.js está basado en una arquitectura orientada a eventos dotada con entrada y salida asíncrona; lo que busca optimizar la escalabilidad y el throughput en aplicaciones WEB con muchas operaciones de entrada y salida, así como, aplicaciones WEB en tiempo real.

## Arquitectura



## NPM

Es el gestor de paquetes por defecto instalado con Node.js. Consta de una herramienta de línea de comandos que permite a los usuarios interactuar con el registro remoto de paquetes de Node.js.

Los paquetes en el registro están en el formato CommonJS e incluyen un archivo package.json que contiene los metadatos de cada módulo.

Para crear el archivo de metadatos usamos el comando ***npm init***.

Ejemplo:

{

"name": "rest-api",

"version": "1.0.0",

"description": "CES 4 example project",

"main": "index.js",

"scripts": {

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

},

"author": "",

"license": "ISC"

}

El archivo package.json provee un elemento script que permite definir rutinas a ejecutarse a través del comando npm. La siguiente es la lista de scripts predefinidos por npm.

* prepublish: Se ejecuta antes de que el paquete sea publicado. También es ejecutado cuando al llamar ***npm install*** en paquetes locales.
* publish, postpublish: Se ejecuta después de que el paquete es publicado.
* preinstall: Se ejecuta antes de que el paquete sea instalado.
* install, postinstall: Se ejecuta después de instalar un paquete con el comando ***npm install***.
* preuninstall, uninstall: Se ejecuta después de ejecutar el comando ***npm uninstall***.
* postuninstall: Se ejecuta después que el paquete ha sido desinstalado.
* postuninstall: Se ejecuta después que el paquete ha sido desinstalado.
* pretest, test, posttest: Es ejecutado al llamar el comando ***npm test***.
* prestop, stop, poststop: Es ejecutado al llamar el comando ***npm stop***.
* prestart, start, poststart: Es ejecutado al llamar el comando ***npm start***.
* prerestart, restart, postrestart: Es ejecutado al llamar el comando ***npm restart***, en caso que no se provea un script para restart, se ejecutaran los comandos stop y start.
* run-script: Permite ejecutar comandos definidos por el usuario, también serán ejecutados los comandos definidos con el prefijo pre y el sufijo post seguidos por el nombre del comando.

Ejemplo:

{

"name": "rest-api",

"version": "1.0.0",

"description": "CES 4 example project",

"main": "index.js",

"scripts": {

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

"start": "node index.js"

},

"author": "",

"license": "ISC"

}

### Instalando dependencias

#### Globales

Son dependencias que se instalan localmente en el directorio de módulos del sistema y estarán disponibles para todos los módulos ejecutados.

Ejemplo:

npm install -g nodemon

#### Locales

Son dependencias que se instalan al interior del directorio node\_modules del proyecto y estarán disponibles solo para dicho proyecto, en caso que se hayan registrado las dependencias del proyecto en el archivo package.json, dichas dependencias se instalaran al ejecutar el comando ***npm install*** al interior del directorio del proyecto.

Ejemplos:

npm install express //Sin registrar la dependencia en el package.json

npm install express --save //Registrando la dependencia en el package.json

## Express

Es un framework para el desarrollo de aplicaciones WEB, es minimalista con muchas funcionalidades que pueden instalarse como plugins.

Para instalar express ejecutamos el siguiente comando:

**npm install express --save**

Ejemplo:

**#index.js**

var express = require('express');

var app = express();

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

res.send('hello world get');

});

app.post('/', function(req, res){

res.send('hello world post');

});

app.put('/', function(req, res){

res.send('hello world put');

});

app.delete('/', function(req, res){

res.send('hello world delete');

});

var server = app.listen(3000, function() {

console.log(`Server listening on port ${server.address().port}`);

});

### Parámetros

Por defecto Express nos provee acceso a los parámetros de la petición a través del objeto request que recibimos como primer parámetro de la función callback de cada método. Por defecto podemos acceder a los parámetros que se configuren en la ruta (req.params) o a los parámetros que se pasen como parte del query string en la petición (req.query).

Ejemplo:

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

res.send(`Hello ${req.query.name} from query param`);

});

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

res.send(`Hello ${req.params.name} from route param`);

});

### Cuerpo

Para poder acceder e interpretar el cuerpo de las peticiones usaremos el plugin body-parser.

**npm install body-parser --save**

Ejemplo:

**#index.js**

var express = require('express');

var bodyparser = require('body-parser');

var routes = require('./routes');

var app = express();

app.use(bodyparser.urlencoded({extended: true}));

app.use(bodyparser.json());

routes.configure(app);

var server = app.listen(3000, function() {

console.log(`Server listening on port ${server.address().port}`);

});

**#routes.js**

module.exports = {

configure: function (app) {

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

res.send(`Hello ${req.query.name} from query param`);

});

app.post('/', function (req, res) {

var person = req.body;

console.log(person);

res.send(`Hello ${person.name} ${person.surname} from json body`);

});

app.put('/', function (req, res) {

res.send('hello world put');

});

app.delete('/', function (req, res) {

res.send('hello world delete');

});

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

res.send(`Hello ${req.params.name} from route param`);

});

}

};

Hasta el momento nuestra aplicación está construida usando ES5, a continuación, echáremos un vistazo a cómo quedaría usando clases de ES6 (lo que mejorará la legibilidad y mantenibilidad de nuestra aplicación).

Ejemplo:

**#app.js**

var express = require('express');

var bodyparser = require('body-parser');

var {ContactRouter} = require('./routes');

let app = express();

app.use(bodyparser.urlencoded({ extended: true }));

app.use(bodyparser.json());

let contactRouter = new ContactRouter(app);

var server = app.listen(3000, function () {

console.log(`Server listening on port ${server.address().port}`);

});

**#router.js**

class Router {

get routes() {

return {};

}

constructor(app) {

this.app = app;

if (!this.app)

throw new Error('Se esperaba el parámetro app');

this.configure();

}

configure() {

var routes = this.routes;

Object.keys(routes).forEach((path) => {

var pathConfig = routes[path];

for (let action of pathConfig) {

this.app[action.verb](path, this[action.method].bind(this));

}

});

}

}

exports.Router = Router;

class Route {

constructor(verb, method) {

this.\_verb = verb;

this.\_method = method;

}

get verb() {

return this.\_verb;

}

get method() {

return this.\_method;

}

}

exports.Route = Route;

**#routes.js**

var {Router, Route} = require('./router');

class ContactRouter extends Router {

get routes() {

return {

'/': [

new Route("get", "getHello"),

new Route("post", "postHelloJSON"),

new Route("put", "putHello"),

new Route("delete", "deleteHello")

],

'/:name': [

new Route("get", "getHelloParameters")

]

};

}

getHello(req, res) {

res.send(`Hello ${req.query.name} from query param`);

}

postHelloJSON(req, res) {

var person = req.body;

console.log(person);

res.send(`Hello ${person.name} ${person.surname} from json body`);

}

putHello(req, res) {

res.send('hello world put');

}

deleteHello(req, res) {

res.send('hello world delete');

}

getHelloParameters(req, res) {

res.send(`Hello ${req.params.name} from route param`);

}

}

exports.ContactRouter = ContactRouter;