Geeks México

BLOG DE PROGRAMACIÓN EN ESPAÑOL SOBRE JAVA, FRAMEWORKS, BASES DE DATOS, CÓMPUTO EN LA NUBE, ETC. EN ESPAÑOL Y EN INGLÉS.

Crea un API REST con Node.js y Express

HACE 1 MIN DEJA UN COMENTARIO

Probablemente has escuchado de Node.js y te has preguntado que es y como usarlo, en este post vamos a abordar ambos temas.

Básicamente, Node.js es un entorno en tiempo de ejecución multi plataforma de código abierto, para la capa del servidor que se auxilia del El motor V8 JavaScript que google usa en su navegador Chrome para interpretar y ejecutar código Javascript, pero ahora del lado del servidor. Sin embargo, Node definitivamente *No* es como Apache Tomcat. Esos servidores básicamente son productos para servidor listos para instalar y que están listos para implementar aplicaciones instantáneamente, sin embargo esto no implica que sea mas complejo en uso o que tenga menos capacidades.

Entre las bondades que podemos encontrar usando Node estan:

- Está basado en eventos, y en un modelo asíncrono, que no bloquea la ejecución del hilo mientras se realizan operaciones de lectura/ escritura, lo que le permite servir una gran cantidad de peticiones al mismo tiempo.
- Permite utilizar el mismo lenguaje (Javascript) tanto en el cliente como en el servidor.
- Tiene una gran gestión de paquetes gracias a NPM (si quieres hacer algo, probablemente exista una librería/paquete que ya lo hacen).
- Nos permite hacer en el servidor todo lo que necesitamos (acceso a ficheros, a bases de datos, conexiones de clientes..)

Ahora si bien Node ya incluye un set de instrucciones para poder crear un API REST, ocuparemos Express.js el cual es un framework ligero, flexible y robusto que se acopla muy bien a Node y que facilitará el desarrollo de nuestras apis.

Para instalar Node en nuestra PC es muy sencillo , solo debemos ir a la pagina oficial de Node.js (https://nodejs.org/en/) descargar el instalador de la ultima version recomendada.

Una vez se allá instalado Node, abriremos una terminal y escribiremos el siguiente comando para verificar la instalación de node.

```
node -v
```

Esto nos dará come resultado la version de Node.js que hallamos instalado.

Ahora procederemos a instalar Express.js de modo global, así como su herramienta para generar el esqueleto de nuestro proyecto REST ,para lo cual ejecutaremos los siguientes comandos en la consola.

```
npm install express -g
```

```
npm install express-generator -g
```

Una vez se hallan ejecutados ambos comando tendremos Express.js y express generator de manera global y lo único que necesitamos para crear nuestro proyecto es colocarnos en alguna carpeta que destinada para nuestros proyectos de node y ejecutar en la consola.

```
express --view=ejs geeks-mexico
```

Una vez se ejecute este comando , crearemos el esqueleto de nuestro proyecto, usando como nuestro manejador de vistas a EJS (aunque hay soporte para otros incluiremos este por ser mas sencillo y familiar).

El siguiente paso sera ubicarnos dentro de la carpteta raíz de nuestro proyecto que en este caso es */geeks-mexico*, y sera necesario descargar la dependencias de nuestro proyecto ejecutando.

```
npm install
```

Las dependencias de nuestro proyecto se encuentran declaradas en nuestra archivo package.json (*si has usado Maven se puede decir que este es el POM de Node*), el cual si lo vemos en detalle aparte de declarar las dependencias del proyecto, también declara el nombre del proyecto, al versión y un script que le indica a en este caso a node, donde se encuentra el archivo que arranca todo el proyecto.

```
{
  "name": "geeks-mexico",
```

```
    "version": "0.0.0",
    "private": true,
    "scripts": {
      "start": "node ./bin/www"
    },
    "dependencies": {
      "body-parser": "~1.18.2",
      "cookie-parser": "~1.4.3",
      "debug": "~2.6.9",
      "ejs": "~2.5.7",
      "express": "~4.15.5",
      "morgan": "~1.9.0",
      "serve-favicon": "~2.4.5"
    }
  }
```

Antes de arrancar la aplicación vamos a explorar con mas detalle la estructura del proyecto.

```
.
├── app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
└── views
    ├── error.ejs
    └── index.ejs
```

Y como puedes ver el proyecto esta segmentado en 4 carpetas base

- **/bin** . Aquí se encuentra el archivo que arranca nuestro proyecto.
- **/public** . Aquí ubicaremos todos los recursos estáticos para un proyecto web, es decir CSS, Javascript en el front e imágenes.
- **/routes**. Aquí se encuentran declarados los endpoints de nuestro API Rest.
- **/views**. Como su nombre lo indica las vistas usadas en nuestro proyecto.

Pues bien entonces procederemos a modificar el archivo index.js y users.js, y como se muestra a continuación.

```
//index.js
var express = require('express');
var router = express.Router();
/* GET home page. */
router.get('/', function(req, res, nex
res.render('index', { title:'Hello wor
});

module.exports = router;
```

Aquí por medio de Express estamos haciendo 2 cosas, la primera es renderizar la vista ***index.ejs*** sobre la raíz de nuestro servidor y pasando como parámetro a la vista el valor de ***title***.

```
var express = require('express');
var router = express.Router();
//simulation of our resources
var users = [];
```

```
/* GET users listing. */
router.get('/', function(req, res, nex
res.json(users);
});

//Create a new user {"user":#,"name":"
router.post('/', function(req, res, ne
users.push(req.body);
res.json(req.body);
});

//Update an existing user {"user":#,"n
router.put('/', function(req, res, nex
var user=users.filter(function(user){
if(user!=null&&user!=undefined){
   user.name=req.body.name;//update na
   res.json(user);
}else{
   res.status(404);
   res.json("Userpdate not found");
}
});

module.exports = router;
```

En este archivo estamos declarando 3 endpoints, el primero solo devuelve la lista de los usuarios que tenemos, el segundo nos permite agregar un usuario a la lista y el tercero nos permite actualizar algún usuario existente, buscándolo por medio de campo *user.*
Finalmente para arrancar nuestro proyecto, solo nos colocamos en la carpeta raíz de nuestro proyecto y desde la consola ejecutamos.

```
npm start
```

Comando que por detrás ejecuta, el o los scripts declarados en el package.json, que en este caso seria equivalente a ejecutar.

```
node ./bin/www
```

Con esto habremos iniciado nuestro proyecto, y podemos acceder a este sobre *localhost:3000.*
Podemos agregar usuarios si hacemos una llamada POST a *localhost:3000/users*

```
{ "user":1, "name":"user1"}
```

Y actualizarlo por medio de la misma ruta pero haciendo un PUT.
El repositorio de git con el código de ejemplo lo puedes clonar o descargar aquí.
Por el momento es todo amigos, pero muy pronto seguiremos con este ejemplo y mostraremos como acoplar este proyecto para una aplicación WEB con angular 5.

Autor : Hugo Alberto Avelino Hernández
Contacto : hugo.avelinoh@gmail.com