Node.js: how to get a list of WooCommerce products with the REST API

In this article we will see how to get the list of products in a WooCommerce store using Node.js and the WooCommerce REST API. We will use Ubuntu Server 16.04 with nginx.

The package.json file

The core modules are ExpressJS and WooCommerce. The others only serve to implement the accessory functions of our app such as authentication.

```
{
  "name": "WCNode",
  "version": "1.0.0",
  "description": "WCNode",
 "author": "Nome Cognome <account@sito.com>",
 "dependencies": {
"body-parser": "^1.17.2",
"cookie-parser": "^1.4.1",
"ejs": "^2.5.6",
"express": "^4.15.3",
"helmet": "^2.1.1",
"serve-favicon": "^2.4.3"
"woocommerce": "^2.4.0"
 },
 "license": "MIT"
 }
```

At this point we install the modules:

```
npm install
```

SSL settings

We create a new set of rules for nginx after acquiring root privileges (`sudo -i` and then `nano /etc/nginx/sites-available/wcnode`):

```
upstream wcnode {
 server 127.0.0.1:3000;
 }
server {
listen 80;

server_name wcnode.sito.com;

root /home/wcnode/www;
index index.html;

client_max_body_size 8m;

location / {
  try_files $uri @wcnode;
}

location @wcnode {
  proxy_pass https://wcnode;
  proxy_set_header X-Real-IP $remote_addr;
  proxy_set_header Host $host;
  proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
  proxy_http_version 1.1;
  proxy_set_header Upgrade $http_upgrade;
  proxy_set_header Connection "upgrade";
}
}
```

Then we enable the new set of rules:

```
ln -s /etc/nginx/sites-available/wcnode
/etc/nginx/sites-enabled/wcnode
```

If you have not already installed the Let's Encrypt tool you can do it as follows: first, add the repository:

```
add-apt-repository ppa:certbot/certbot
```

Then update the package list:

```
apt-get update
```

Finally install Certbot:

```
apt-get install python-certbot-nginx
```

Before restarting nginx, we get an SSL certificate from Let's Encrypt:

```
certbot --nginx -d wcnode.sito.com
```

We choose option 2 during the installation in order to have the SSL redirect inserted in our set of rules.

Now we can copy the certificate files into the directory of our app:

```
cp
/etc/letsencrypt/live/wcnode.sito.com/fullchain.pem >
/home/wcnode/app/fullchain.pem && chown wcnode:wcnode
/home/wcnode/app/fullchain.pem

cp /etc/letsencrypt/live/wcnode.site.com/privkey.pem
> /home/wcnode/app/privkey.pem && chown wcnode:wcnode
/home/wcnode/app/privkey.pem
```

Now we have to create a new service for `systemd` so that the application keeps running and is live at reboot. We type `nano /etc/systemd/system/wcnode.service` and insert the following contents:

```
[Service]
    WorkingDirectory=/home/wcnode/app
    ExecStart=/usr/local/bin/node app.js
    Restart=always
    StandardOutput=syslog
    StandardError=syslog
    SyslogIdentifier=wcnode
    User=wcnode
    Group=wcnode
    Environment=NODE_ENV=production

    [Install]
    WantedBy=multi-user.target
```

Now we can enable the service:

```
systemctl enable wcnode
```

And we start it:

```
systemctl start wcnode
```

Let's test the nginx configuration:

```
nginx -t
```

If everything is ok, we restart it:

```
systemctl restart nginx
```

The app code

Basically, a GET request will be performed to the WooCommerce store using the REST API endpoint `/wp-json/wc/v2/products` with our credentials.

The base structure of our app is as follows:

```
'use strict';
const express = require('express');
const fs = require('fs');
const https = require('https');
const port = process.env.PORT || 3000;
const app = express();
const routes = require('./routes');
const API = require('./lib/API');
const sslOptions = {
  key: fs.readFileSync('privkey.pem'),
  cert: fs.readFileSync('fullchain.pem')
 };

 app.disable('x-powered-by');
 app.set('view engine', 'ejs');

 app.use( (req, res, next ) => {
     req.API = API;
     next();
 });

 app.get('/products', routes.products);

 https.createServer(sslOptions, app).listen(port);
```

The management of calls to the API is delegated to the `API` class that we have made available in all the routes of our app through the middleware function defined in `.use()` . In this way, the `request` object now has a reference to this class that is visible throughout the application lifecycle.

The class is very simple. First, it gets the `WooCommerce` module and defines the basic options:

```
'use strict';
const WC = require('woocommerce');
const WooCommerce = new WC({
    url: 'https://ecommerce.com',
    ssl: true,
    consumerKey: 'ck_123456789abcd',
    secret: 'cs_abcdefg12345'
});
```

Then comes the body of the class:

```
class API {
static products() {
    return WooCommerce.get('/products');
}
}
module.exports = API;
```

The static `products()` method returns a Promise that can have two results:

1. The request is successful; an array of objects is returned.
2. The request fails; an error is returned with details.

We use this method in our route:

```
'use strict';
// routes/index.js
module.exports = {
    products: (req, res) => {
        req.API.products().then(data => {
           res.render('products', {products:
data.products});
        }).catch(err => {
           res.json(err);
        });
    }
};
```

The result is shown below.