

# PUG en Express

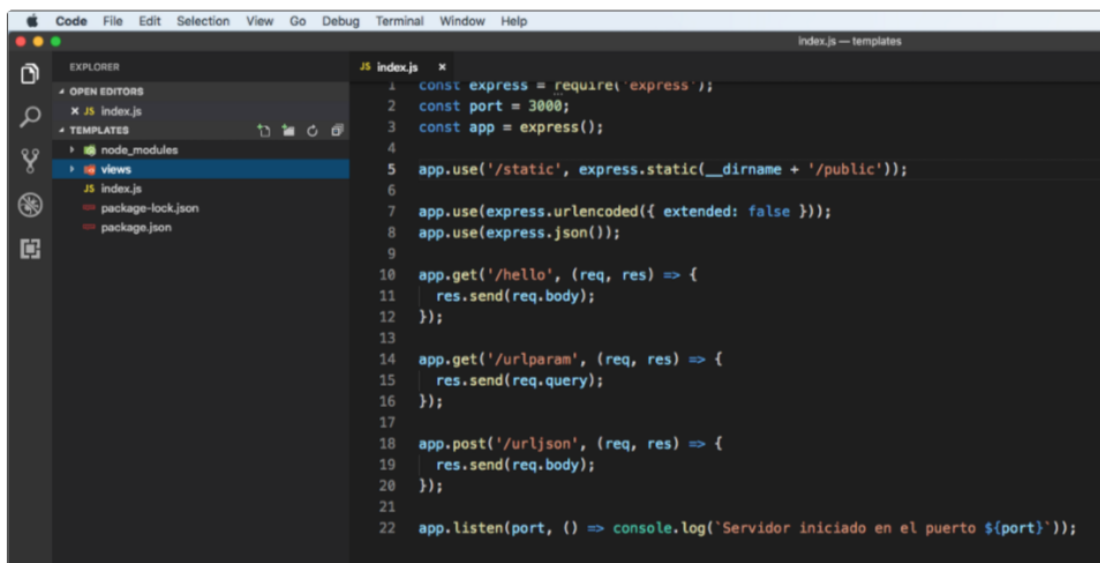


En esta ocasión les traemos un tutorial para poder usar Pug en el framework Express, pug nos permite utilizar archivos estáticos como plantillas, nos permite enviar valores para reemplazar variables dentro de las mismas y puede transformar estos archivos en páginas HTML las cuales se envían al cliente.

Hablando un poco de Express podemos decir que Express nos permite trabajar con muchos motores de plantilla entre los que se encuentran ubicados en Pug, el cual tiene una manera fácil de implementar, solo bastará un par de líneas de código para indicarle a Express que use Pug como su motor de plantillas.

## Configuración

Para nuestra configuración seguiremos una serie de pasos, como primer paso tenemos que crear un directorio para guardar las plantillas que se utilizarán en nuestra aplicación, para eso entonces lo que tenemos que hacer es crear esa carpeta en la raíz de nuestro proyecto o donde está ubicado nuestro proyecto, a continuación se presentará una imagen en una carpeta llamada views donde pondremos nuestros archivos.



Luego de esta parte, ahora vamos a decir que nuestra carpeta sera de plantillas, también vamos a indicar cual sera el motor de plantillas, en este caso PUG

```
1  const express = require('express');
2  const port = 3000;
3  const app = express();
4
5  app.use('/static', express.static(__dirname + '/public'));
6
7  app.use(express.urlencoded({ extended: false }));
8  app.use(express.json());
9
10 // Se indica el directorio donde se almacenarán las plantillas
11 app.set('views', './views');
12
13 // Se indica el motor del plantillas a utilizar
14 app.set('view engine', 'pug');
15
16 app.get('/hello', (req, res) => {
17   res.send('Hola mundo');
18 });
19
20 app.get('/urlparam', (req, res) => {
21   res.send(req.query);
22 });
23
24 app.post('/urljson', (req, res) => {
25   res.send(req.body);
26 });
27
28 app.listen(port, () => console.log(`Servidor iniciado en el puerto ${port}`));
```

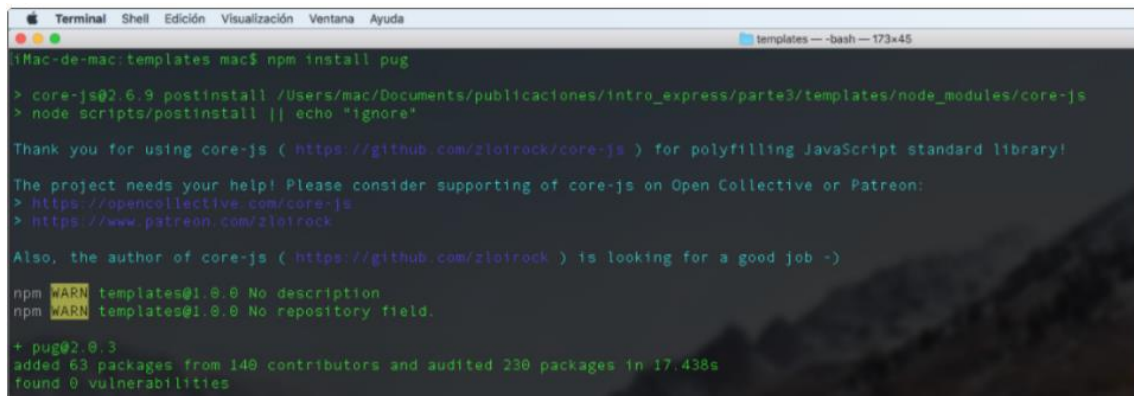
Como vemos, tenemos que en las líneas 11 y 14 configuramos correctamente el Pug.

## Instalar Pug

Ahora veremos como instalar el Pug, para esto haremos uso de la línea de comandos en mi caso usaremos una MAC, ustedes pueden ver en internet que comandos son requeridos para su Sistema Operativo, entonces hacemos lo siguiente:

Escribimos el siguiente comando:

```
npm install pug
```



```
Mac-de-mac:templates mac$ npm install pug
> core-js@2.6.9 postinstall /Users/mac/Documents/publicaciones/intro_express/parte3/templates/node_modules/core-js
> node scripts/postinstall || echo "ignore"

Thank you for using core-js ( https://github.com/zloirock/core-js ) for polyfilling JavaScript standard library!

The project needs your help! Please consider supporting of core-js on Open Collective or Patreon:
> https://opencollective.com/core-js
> https://www.patreon.com/zloirock

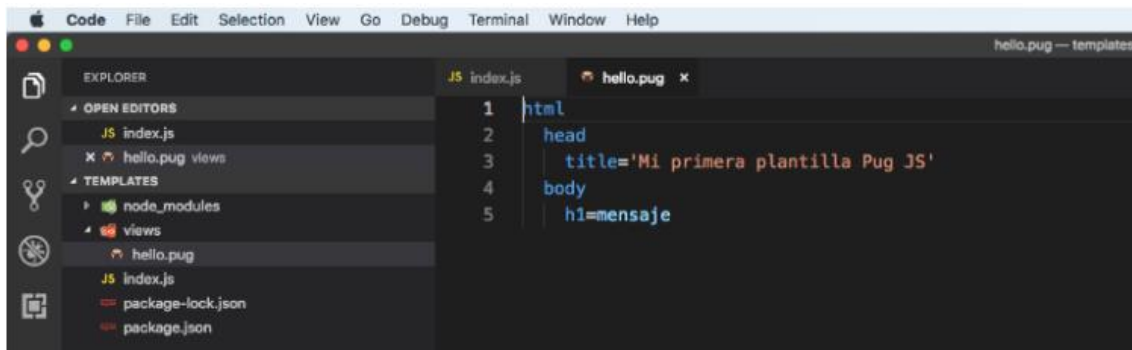
Also, the author of core-js ( https://github.com/zloirock ) is looking for a good job -)

npm WARN templates@1.0.0 No description
npm WARN templates@1.0.0 No repository field.

+ pug@2.0.3
added 63 packages from 140 contributors and audited 230 packages in 17.438s
found 0 vulnerabilities
```

## Crear nuestra primera plantilla

En este paso, una vez que hemos podido configurar e instalar correctamente nuestro pug, solo nos quedaría crear nuestra primera plantilla y mostrarla al cliente, ahora lo que debemos hacer es crear un archivo con la extensión “.pug” por ejemplo, “archivo.pug” esta extensión es para las plantillas y las mostraremos al ingresar en la url: localhost



Algo que podemos decir de Pug es que utiliza su propia sintaxis para declarar atributos html, sin necesidad de abrir y cerrar etiquetas, en cambio se usa la tabulación para indicar que una etiqueta pertenece o esta dentro de otra.

Como se puede apreciar en la línea 5 se esta declarando una variable con nombre “mensaje”, esta sera reemplazada con el valor que se enviara al momento que transforma de formato .pug a html.

Seguidamente nuestro siguiente paso sera modificar la ruta “/hello”, actualmente esta mostrando al usuario el mensaje “Hola mundo”, lo reemplazaremos con la plantilla que creamos. Entonces para ello poremos usar la función “render” la cual esta disponible en el objeto res, con esta función que recibe dos parámetros, el primero el nombre de la plantilla a mostrar y el segundo un objeto con los valores a reemplazar.

| *res.render(view: string, options?: Object)*

Ahora procederemos a mostrar el código final:

```
1  const express = require('express');
2  const port = 3000;
3  const app = express();
4
5  app.use('/static', express.static(__dirname + '/public'));
6
7  app.use(express.urlencoded({ extended: false }));
8  app.use(express.json());
9
10 // Se indica el directorio donde se almacenarán las plantillas
11 app.set('views', './views');
12
13 // Se indica el motor del plantillas a utilizar
14 app.set('view engine', 'pug');
15
16 app.get('/hello', (req, res) => {
17   res.render('hello.pug', { mensaje: 'Usando Pug JS en Express' }); // Se muestra la plantilla
18 });
19
20 app.get('/urlparam', (req, res) => {
21   res.send(req.query);
22 });
23
24 app.post('/urljson', (req, res) => {
25   res.send(req.body);
26 });
27
28 app.listen(port, () => console.log(`Servidor iniciado en el puerto ${port}`));
```

El resultado final es el mensaje “Usando Pug en Express”, al cliente a través de una pagina HTML



← → ↻ 🏠 ⓘ localhost:3000/hello

## Usando Pug JS en Express