Desarrollo del lado servidor con NodeJS, Express y MongoDB

Express

Express[1] es la librería más popular de NodeJS y proporciona varias funcionalidades, para el desarrollo de aplicaciones web y mobile que te detallamos a continuación. Fue desarrollada al poco tiempo de que apareciera Node, a fines 2009.

Presenta una concepción minimalista y es la base de muchas otras librerías. Está basada en el framework *connect*[2] que es un modelo de servidor HTTP para Node, con la posibilidad de agregarle módulos o extensiones, conocidos también como *middlewares.*

Algunas de sus funcionalidades principales son:

- Escribir métodos de gestión de peticiones con diferentes verbos HTTP en diferentes rutas URL.
- Integrar con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en páginas html.
- Establecer ajustes de aplicaciones web como qué puerto usar para conectar y la localización de las páginas html que se utilizan para renderizar la respuesta.
- Añadir procesamiento de peticiones "middleware" adicional en cualquier punto dentro del proceso de manejo de la petición.

[1] https://expressjs.com/
[2] https://github.com/senchalabs/connect#readme

Para asistir el desarrollo web, los programadores de diferentes partes del mundo han creado decenas de *middlewares* y los han hecho públicos vía este sitio: https://expressjs.com/es/resources/middleware.html

donde el mismo equipo de desarrollo Express se encarga de mantenerlos actualizados.

Allí podrás observar que contamos con soluciones para la mayoría de los problemas habituales de todo proyecto web: manejo de sesiones, cookies, datos del POST, temas de seguridad, etc.

Trabajar con Express es similar a hacerlo con Node. En la siguiente lección veremos algunas características comunes al desarrollo.

Antes, veamos cómo crear el típico programa Hola Mundo!.

Primero instala Express haciendo, npm install express -g. Lo hacemos con el f*lag* -g para que quede en el entorno global de npm y poder utilizarlo desde cualquier lugar.

Creamos un archivo app.js con el siguiente código:

```
var express = require('express');
var app = express();

app.get('/', function(req, res) {
  res.send('Hola Mundo!');
});

app.listen(3000, function() {
  console.log('Aplicación ejemplo, escuchando el puerto 3000!');
});
```

Si ahora ejecutas *node app.js ,* verás que ya tenemos la aplicación corriendo.

Analicemos qué hicimos.

La instrucción *require* indica la incorporación de un módulo, en este caso Express.

En la variable app (se suele nombrar así) creamos la aplicación servidor que posee métodos para enrutamiento de las peticiones HTTP, configuración del 'middleware', y visualización de las vistas de HTML, uso de motores de 'templates' y gestión de las configuraciones de las aplicaciones[3] que controlan la aplicación.

El método get en app (app.get) define una ruta que se especifica en el primer parámetro, en este caso '/', usualmente denominada como ruta raíz, que puede accederse mediante el verbo GET del protocolo HTTP. El otro parámetro es un *callback,* una función que se ejecuta dentro del método, y conoce la petición en sí vía el parámetro *req* y la respuesta a resolver vía el parámetro *res.*

Luego, resuelve enviar el *string, cadena de caracteres,* 'Hola Mundo!' con el método *send* aplicado al *res.*

El bloque final de código define y crea el servidor, escuchando el puerto 3000, e imprime un comentario en la consola. Cuando se está ejecutando el servidor, es posible ir hasta la dirección localhost:3000 en un navegador y ver cómo el servidor de este ejemplo devuelve el mensaje de respuesta.

Pasemos a revisarlo en el siguiente breve tutorial.

[3] https://expressjs.com/en/4x/api.html#app.settings.table