



Clase 27. Programación Backend

Proceso principal del servidor



OBJETIVOS DE LA CLASE

- Conocer el pasaje de parámetros al servidor y sus funciones.
- Pasar argumentos en la línea de comandos en Node mediante las librerías Minimist y Yargs.
- Presentar las variables de entorno y su uso.
- Aplicar la librería Dotenv.

CRONOGRAMA DEL CURSO

Clase 26



**Estrategias de
autenticación
con redes sociales**

Clase 27



**Proceso principal del
servidor**

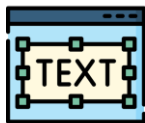
Clase 28



Global & Child process

PASAJE DE PARÁMETROS AL SERVIDOR

ARGUMENTOS DE LA LÍNEA DE COMANDOS EN NODE



¿De qué se trata?



- Los argumentos de la línea de comandos son cadenas de texto que se utilizan para pasar información adicional a un programa, cuando se ejecuta una aplicación a través del interfaz de línea de comandos (**CLI**) de un sistema operativo.
- Suelen incluir información que se utiliza para establecer la configuración o los valores de propiedad de una aplicación.
- En la mayoría de los casos, los argumentos se pasan después del nombre del programa en su indicador. Un ejemplo de la sintaxis de los argumentos de la línea de comandos es el siguiente:

```
$ [runtime] [script_name] [argument-1 argument-2 argument-3 ... argument-n]
```

- El tiempo de ejecución (runtime), en nuestro caso es **Node**.



Ventajas



- Puede pasar información a una aplicación antes de que comience. Esto es útil si deseamos realizar ajustes de configuración de gran número.
- Los argumentos de la línea de comandos se pasan como cadenas al programa. Los tipos de datos de cadena se pueden convertir fácilmente a otros tipos de datos dentro de una aplicación, lo que hace que los argumentos sean muy flexibles.
- Puede pasar un número ilimitado de argumentos a través de la línea de comando.
- Los argumentos de la línea de comandos se utilizan junto con scripts y archivos por lotes, lo que es útil para las pruebas automatizadas.



Desventajas



- La mayor desventaja de pasar información a través de la línea de comandos es que la interfaz tiene una curva de aprendizaje pronunciada, por lo que es difícil de usar para la mayoría de las personas a menos que tengan mucha experiencia en el uso de herramientas CLI.
- Las aplicaciones de línea de comandos pueden ser difíciles de usar a menos que esté usando una computadora de escritorio o portátil, por lo que normalmente no se usan en dispositivos más pequeños como teléfonos o tabletas.

PROCESS.ARGV



Uso de process.argv



- La forma más sencilla de recuperar argumentos en Node es a través del ***process.argv***. Este es un objeto global que podemos usar sin importar bibliotecas adicionales.
- Simplemente necesitamos pasar argumentos a una aplicación Node, tal como mostramos anteriormente, y se puede acceder a estos argumentos dentro de la aplicación a través del `process.argv`.



¿Cómo se usa process.argv?

1

El primer elemento del process.argv, el array, siempre será una ruta del sistema de archivos que apunta al Node ejecutable.

2

El segundo elemento es el nombre del archivo JavaScript que se está ejecutando.

3

el tercer elemento es el primer argumento que realmente pasó el usuario.

Vamos al VC 



***¿Alguna pregunta hasta
ahora?***

MINIMIST

CODER HOUSE



Usando *minimist*



- Otra forma de recuperar argumentos de línea de comando en una aplicación Node es usando el módulo ***minimist***.
- Nos permite analizar un array de strings (usualmente obtenido de los argumentos ingresados por línea de comando) y lo transformará en un objeto más fácil de usar, ya que nos permite acceder a los elementos mediante su nombre.
- Lo instalamos como:

```
$ npm install minimist
```
- Lo podemos requerir en nuestros archivos como:

```
const parseArgs = require('minimist')
```

Vamos al VC 



Usando minimist



- Ante un argumento que comienza con un guión solo, crea un atributo cuyo nombre será la primera letra dsp del guión, y cuyo valor será todo lo restante hasta el próximo argumento
- Si queremos usar palabras como nombres de atributos, debemos anteponerle un doble guión, y su valor será el argumento siguiente.
- Si luego de un argumento guionado (simple o doble), no se encuentra ningún otro argumento, o el argumento siguiente también es guionado, el primero se interpreta como un valor booleano con valor true.
- Si existen valores sueltos que no vienen precedidos por ningún argumento guionado, todos ellos se agrupan en un array bajo el nombre _ (guión bajo).



Usando minimist



Además también podemos agregar algunas opciones adicionales. Entre ellas, podemos definir algunos valores por defecto para argumentos que no estén presentes:

```
const options = { default: { nombre: 'pepe', apellido: 'copado' } }

console.log(parseArgs(['-a', '1', '-b', '2', 'un-valor-suelto', '--nombre', 'juanita'], options));
// { _: [ 'un-valor-suelto' ], a: 1, b: 2, nombre: 'juanita', apellido: 'copado' }
```

Y podemos agregarle algunos alias para renombrar los valores ingresados como argumentos, dándoles nombres más descriptivos

```
const options = { alias: { a: 'campoA', b: 'campoB', } }

console.log(parseArgs(['-a', '1', '-b', '2'], options));
// { _: [], a: 1, campoA: 1, b: 2, campoB: 2 }
```



ARGUMENTOS POR LÍNEA DE COMANDOS

Tiempo: 5 minutos

ARGUMENTOS POR LÍNEA DE COMANDOS

Desafío
generico



Tiempo: 5 minutos

Realizar una aplicación en Javascript ejecutada a través de Node.JS que al ejecutarse de la siguiente manera:

node main.js 1 2 3 -m dev -p 8080 -d

Construya y muestre por pantalla el siguiente objeto:

{ modo: 'dev', puerto: 8080, debug: true, otros: [1, 2, 3] }

Y con el siguiente llamado:

node main.js 1 2 3

Construya y muestre por pantalla el siguiente objeto:

{ modo: 'prod', puerto: 0, debug: false, otros: [1, 2, 3] }



***¿Alguna pregunta hasta
ahora?***

YARGS



Usando yargs



- Otro módulo que nos ayudará a analizar los argumentos de la línea de comandos pasados a los programas de Node es el módulo **yargs**. Inicialmente funciona de igual manera que Minimist, con algunas modificaciones y en su sintaxis. Sin embargo, esta librería posee muchísimas más funcionalidades.
- Lo instalamos con npm (`npm i yargs`) y lo utilizamos de la siguiente manera:

```
const yargs = require('yargs/yargs')(process.argv.slice(2))  
const args = yargs.argv
```

Vamos al VC 



USANDO YARGS

Tiempo: 5 minutos

USANDO YARGS

Tiempo: 5 minutos

Desafío
generico



Realizar la operación del desafío anterior, esta vez usando la dependencia YARGS en lugar de minimist.



***¿Alguna pregunta hasta
ahora?***



BREAK

¡5/10 MINUTOS Y VOLVEMOS!

VARIABLES DE ENTORNO



¿De qué se trata?



- Las variables de entorno son variables externas a nuestra aplicación que residen en el sistema operativo o en el contenedor de la aplicación que se está ejecutando. Una variable de entorno es simplemente un nombre asignado a un valor.
- Nos permiten administrar la configuración de nuestras aplicaciones por separado de nuestro código base.
- Las configuraciones separadas facilitan la implementación de nuestra aplicación en diferentes entornos (desarrollo, test, producción, etc).
- Por convención, el nombre se escribe con mayúscula y los valores son cadenas de texto, por ejemplo: PORT=8080.



¿De qué se trata?



- Normalmente, nuestras aplicaciones requieren que se establezcan muchas variables de entorno para que funcionen. Al confiar en configuraciones externas, nuestra aplicación se puede implementar fácilmente en diferentes entornos. Estos cambios son independientes de los cambios en el código, por lo que no requieren que nuestra aplicación sea reconstruida.
- Los datos que cambian según el entorno en el que se ejecuta su aplicación deben configurarse como variables de entorno. Algunos ejemplos comunes son:
 - Dirección y Puerto HTTP.
 - Credenciales de Base de Datos.
 - Ubicación de archivos y carpetas estáticos.
 - Credenciales de API's externas.

Vamos al VC 



AGREGANDO VARIABLES DE ENTORNO

Tiempo: 5 minutos



AGREGANDO VARIABLES DE ENTORNO

Tiempo: 5 minutos

Realizar una aplicación en nodeJS que al pasar las siguientes variables de entorno:

MODO=dev PUERTO=8080 DEBUG=true

Construya y muestre por pantalla el siguiente objeto:

`{ modo: 'dev', puerto: 8080, debug: true }`

Y que al ejecutarlo sin pasar ninguna variable de entorno construya y muestre por pantalla el siguiente objeto:

`{ modo: 'prod', puerto: 0, debug: false }`



***¿Alguna pregunta hasta
ahora?***

DOTENV

¿De qué se trata?



- **Dotenv** es un módulo de Node que carga variables de entorno desde un archivo **.env** a *process.env*.
- El desarrollo con múltiples variables de entorno se vuelve rápidamente difícil de mantener. Entonces, utilizamos dotenv para que las variables queden almacenadas todas juntas en el archivo *.env*.



Usando dotenv



- En primer lugar, instalamos el módulo con el comando: `npm i dotenv`
- Luego, creamos el archivo `.env`, con las variables de entorno que queremos definir. Por ejemplo:

```
FONDO=blanco  
FRENTE=negro
```

Vamos al VC 



IMPLEMENTANDO DOTENV

Tiempo: 5 minutos

IMPLEMENTANDO DOTENV

Tiempo: 5 minutos

Realizar el desafío anterior utilizando la librería dotenv.

Desafío
generico



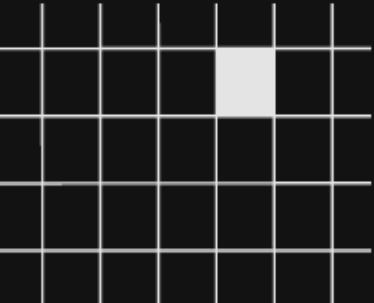
¿PREGUNTAS?





¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Argumentos en la línea de comandos de Node
 - Objeto process.
 - Process.argv
 - Minimist y Yargs
 - Variables de entorno y Dotenv
- 



OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDOLAEDUCACIÓN