Node y el gestor de paquetes

Interfaces de línea de comando con Yargs

{desafío} latam_

Aplicar procedimiento de puesta en ejecución de una aplicación Node.js implementando mecanismos para la detección de errores durante la ejecución.

- Unidad 1:
  Introducción a Node
- Unidad 2:
  Node y el gestor de paquetes
- Unidad 3:
  Persistencia

Te encuentras aquí

{desafío} latam_

¿Qué aprenderás en esta sesión?

- *Reconocer los parámetros básicos del paquete Yargs para crear una interfaz de línea de comando*
- *Implementar una interfaz de línea de comando con el paquete Yargs para levantar una aplicación Node*

Imagina que estás desarrollando una aplicación en Node.js y deseas permitir a los usuarios interactuar con ella a través de la línea de comando. ¿Cómo podrías lograr esto de manera eficiente?

{desafío} latam_

/* El paquete Yargs */

El paquete Yargs

Yargs, es uno de los paquetes más conocidos en NPM para el desarrollo de líneas de comando, definido en su sitio oficial como “constructor de líneas de comando interactivas analizando argumentos y generando una elegante interfaz de usuario”.

Es importante que sepas que una interfaz de línea de comando no es lo mismo que una interfaz gráfica, incluso sus siglas representativas no son las mismas. Las siglas “CLI” significan Command Line Interface, por otro lado tenemos las siglas “GUI” que significan Graphical User Interface.

El paquete Yargs

Instalación

Para instalar este paquete deberás usar el siguiente comando:

```
npm install yargs
```

Una vez instalado solo necesitarás importarlo en una constante y empezar a usar su API, la cual es bastante extensa y podrás conocerla en profundidad en su documentación oficial.

El paquete Yargs

Mi primer CLI

Dentro del objeto importado del paquete “Yargs” contamos con el método “command”, el cual, tiene el objetivo de definir los parámetros y la configuración de nuestra interfaz de línea de comando, y se usa por medio de la siguiente sintaxis

```
.command( <comando> , <descripción>, <constructor>, <callback> )
```

El paquete Yargs

Definición de parámetros

- **Comando:** Acá deberás escribir un dato tipo String que representará el comando principal de nuestra CLI.
- **Descripción:** Es la descripción de nuestro comando. Cada comando podrá ser accedido próximamente en una lista, acompañados de sus descripciones para un mejor entendimiento con el usuario.
- **Constructor:** En este parámetro definiremos en forma de objeto el único o los diferentes argumentos que queremos disponer en nuestro CLI, además de sus especificaciones personales (descripción, requerido, alias). Los argumentos en Yargs se deberán escribir con 2 guiones antes, por ejemplo --argumento, o con un solo guión utilizando su alias como si se tratara de un flag (bandera), ejemplo: -a
- **Callback:** La función se ejecutará una vez que se ejecute la línea de comando. Esta función tiene una particularidad, la cual es recibir como parámetro el objeto “argv” que contendrá los argumentos escritos en la línea de comando.

Demostración "Aplicando Yargs"

Ejercicio guiado

Aplicando Yargs

Construir una aplicación que use el paquete Yargs para la definición de una interfaz de línea de comandos. El objetivo será definir un comando “saludo” que reciba como argumento tu nombre y responda con un mensaje en consola saludando y deseando un excelente día, sigue los siguientes pasos:

- **Paso 1:** Importar en una constante el paquete Yargs.
- **Paso 2:** Inicializar el método “command” para el paso de parámetros.
- **Paso 3:** Definir del comando con el primer parámetro el cual será “saludo”.

Ejercicio guiado

Aplicando Yargs

- **Paso 4:** Definir la descripción del comando “saludo” como segundo parámetro del método “command”.
- **Paso 5:** Definir el objeto para la configuración del constructor del comando.
- **Paso 6:** Declarar que se esperará recibir un argumento llamado “nombre”.
- **Paso 7:** Definir la descripción de este argumento.
- **Paso 8:** Declarar que este argumento es requerido con un true en la propiedad “demand”.

Ejercicio guiado

Aplicando Yargs

- **Paso 9:** Declarar el alias del argumento nombre, el cual será “n”. Esto sirve para simplificar la delación de un argumento recortando su mención a solo 1 letra o siglas.

- **Paso 10:** Crear la función callback la cual recibe como parámetro el objeto args que contendrá los argumentos como propiedades. A su vez la función mandará un mensaje por consola saludando con el nombre recibido como argumento.

- **Paso 11:** Concatenar el método command con el método “help” y la propiedad argv.

```
// Paso 1
const yargs = require('yargs')
// Paso 2
const argv = yargs
 .command(
   // Paso 3
   'saludo',
   // Paso 4
   'Comando para saludar',
   // Paso 5
   {
     // Paso 6
     nombre: {
       // Paso 7
       describe: 'Argumento para definir tu nombre',
       // Paso 8
       demand: true,
       // Paso 9
       alias: 'n',
     },
   },
   // Paso 10
   (args) => {
     console.log(`¡Saludos cordiales ${args.nombre}, espero que tengas un excelente día`)
   }
 )
  // Paso 11
 .help().argv
```

Ejercicio guiado

Aplicando Yargs

- **Paso 12**: Ahora probemos, abre la terminal y ejecuta el siguiente comando:

```
node index.js saludo --nombre=Jocelyn
```

Y deberás recibir por pantalla el mensaje que se muestra en la siguiente imagen:

```
➜  ejercicio lectura git:(master) ✗ node index.js saludo --nombre=Jocelyn
¡Saludos cordiales Jocelyn, espero que tengas un excelente día
```

/* Evaluando los argumentos */

Evaluando los argumentos

Ahora que sabemos que podemos usar los argumentos recibidos por línea de comandos, podríamos escribir un condicional dentro de la función callback, que evalúe si el valor recibido es igual al valor que esperamos para realizar alguna acción. Por ejemplo: “Tenemos una aplicación en un archivo aparte que se ejecutará sólo si los argumentos ingresados por la línea de comando son las credenciales de un usuario Administrador”.

¿Cómo desarrollaremos esta validación? Simple, alojamos en nuestro código un usuario y contraseña específico y evaluamos si los valores recibidos coinciden con estos, entonces de ser así la función callback podría ejecutar con el módulo “child_process” un archivo externo y diferente a la aplicación con la que estamos interactuando, es decir, que podemos ejecutar o levantar una aplicación externa a partir de la evaluación de argumentos pasados en la línea de comandos.

Demostración
" Evaluando los argumentos"

Ejercicio guiado

Evaluando los argumentos

Desarrollar una aplicación que reciba argumentos por la línea de comandos y los evalúe, estos argumentos corresponden a unas credenciales, en caso de éxito, es decir que las credenciales son las correctas, se ejecutará otra aplicación que devolverá un mensaje por consola indicando “Bienvenido al Área 51”.

Lo primero será crear un archivo nuevo llamado acceso.js con el siguiente código:

```
console.log('Bienvenido al Área 51')
```

Ejercicio guiado

Evaluando los argumentos

Este archivo entonces será llamado desde el index.js solo si la condicional lo permite. Sigue los siguientes pasos para la solución de este ejercicio:

- **Paso 1:** Importar en una constante el paquete child_process.
- **Paso 2:** Definir las credenciales de acceso.
- **Paso 3:** Definir el comando “acceso”.
- **Paso 4:** Definir la descripción del comando “acceso”.

Ejercicio guiado

Evaluando los argumentos

- **Paso 5:** Definir qué se requerirá el argumento “user” con sus características.
- **Paso 6:** Definir qué se requerirá el argumento “pass” con sus características.
- **Paso 7:** Declarar el operador ternario que condicione si los valores ingresados en los argumentos coinciden con las credenciales.
- **Paso 8:** Ejecutar el archivo acceso.js con el método exec del módulo child_process e imprimir por consola su respuesta.
- **Paso 9:** Devolver en el “else” del operador ternario un mensaje por consola que diga “Credenciales incorrectas”

```
const yargs = require('yargs')
// Paso 1
const child = require('child_process')
// Paso 2
const user = 'Ovni22'
const pass = 123457
const argv = yargs
 .command(
   // Paso 3
   'acceso',
   // Paso 4
   'Comando para acceder al Área 51',
   {
     // Paso 5
     user: {
       describe: 'Usuario',
       demand: true,
       alias: 'u',
     },
     // Paso 6
     pass: {
       describe: 'Contraseña',
       demand: true,
       alias: 'p',
     },
   },
   (args) => {
     // Paso 7
     args.user == user && args.pass == pass
       ? // Paso 8
         child.exec('node acceso.js', (err, stdout) => {
           err ? console.log(err) : console.log(stdout)
         })
       : // Paso 9
         console.log('Credenciales incorrectas')
   }
)
.help().argv
```

{desafío latam_}

Ejercicio guiado

Evaluando los argumentos

- Paso 10: Ejecutar el archivo index.js con el siguiente comando:

```
node index.js acceso -u=Ovni22 -p=123457
```

Obtendrás lo que te muestro en la siguiente imagen:

```
→  ejercicio lectura git:(master) ✗ node index.js acceso -u=Ovni22 -p=123457
Bienvenido al Área 51
```

Resumen

- Se aprendió a cómo utilizar el paquete Yargs para crear una interfaz de linea de comando en una aplicación Node.js.

- El paquete Yargs es presentado como una herramienta popular para el desarrollo de líneas de comando. Se define como un "constructor de líneas de comando interactivas" que analiza argumentos y genera una interfaz de usuario elegante. Yargs ayuda a crear comandos personalizados y a definir sus descripciones, argumentos y opciones.

¿Puedes mencionar dos ventajas de utilizar Yargs en comparación con enfoques tradicionales para manejar argumentos y comandos en aplicaciones Node.js?

{desafío} latam_

Próxima sesión...

- *Guía de ejercicios - Node y el gestor de paquetes*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales