Créer un serveur Web en Node.js

1. Objectifs

- Apprendre à créer un serveur Web à l'aide du module HTTP
- Être capable de construire des serveurs web en utilisant le module http qui est inclus dans Node.js.

2. Qu'est-ce qu'un serveur Web

- **Node.js** permet aux développeurs d'utiliser JavaScript pour écrire du code back-end, même si, traditionnellement, il était utilisé dans le navigateur pour écrire du code front-end. Le fait d'avoir le front-end et le back-end ensemble de cette manière réduit l'effort nécessaire pour créer un serveur web, ce qui est une des raisons majeures pour lesquelles Node.js est un choix populaire pour écrire du code back-end.
- Un serveur Web est un serveur HTTP dont le travail consiste à accepter les requêtes HTTP et à renvoyer une réponse. C'est un morceau de code backend exécuté sur une machine quelque part qui répond à notre demande. Chaque fois que nous tapons une URL dans la barre d'adresse et demandons une page Web, il y a un serveur Web qui prend notre demande et renvoie la page Web appropriée. Les fichiers dont nous avons besoin pour visiter la page Web sont hébergés ou stockés sur un serveur Web et il comprend très bien les URL. Il lit l'URL demandée, vérifie quel fichier doit être consulté et renvoie ce fichier en réponse.

3. Comment créer un serveur Web

1. **Étape 1: Configurer un répertoire de projet**

   - Ceci est une nouvelle application, alors configurons-la dans un nouveau répertoire. Nous nommerons le répertoire **web_server**.
   - Une fois que vous avez terminé, entrez dans le dossier et créez un fichier **server.js**.
     Nous ajouterons tout notre code dans ce fichier.

2. **Étape 2: Créer un serveur et écouter un port**

apcpedagogie.com

- Importons d'abord le module http. Le module http a une méthode `createServer()` qui est utilisée pour instancier le serveur.
- Après cela, nous écouterons les demandes entrantes sur un message particulier.

```
const http = require('http');
const port = 1337;
const host = 'localhost';

const server = http.createServer(function (req, res) {
    // add code to handle requests
});

server.listen(port, host, function () {
    console.log('Web server is running on port 1337');
});
```

3. **Le port**

- **port** est un numéro que les serveurs utilisent comme point d'accès ou "port" à notre adresse IP. Dans notre exemple, nous utiliserons le port 1337 pour notre serveur web.
- Lorsque nous lierons notre serveur à cet hôte et à cet port, nous pourrons atteindre notre serveur en visitant http://localhost:1337 dans un navigateur local.

4. **La valeur localhost**

- **localhost** est une adresse privée spéciale que les ordinateurs utilisent pour se désigner eux-mêmes. Elle est généralement l'équivalent de l'adresse IP interne **127.0.0.1** et n'est disponible que pour l'ordinateur local, et non pour les réseaux locaux que nous avons rejoints ou pour l'internet.

5. **La méthode createServer**

- Le premier argument **req** est l'objet de requête. Il contient des informations sur la demande. Il répondra à des questions telles que quel type de demande s'agit-il? Y a-t-il des données transmises avec lui? Que sont les paramètres de requête? etc.
- Le deuxième argument **res** est l'objet de réponse. Nous l'utiliserons pour renvoyer une réponse lorsque la demande est réussie ou renvoyer une erreur en cas d'échec. Nous définirons également l'état de la réponse HTTP sur cet objet pour mieux expliquer le type de réponse. 404 est une erreur "fichier non trouvé" , 500

apcpedagogie.com

est une "erreur de serveur" et 200 le code d'état est pour "les demandes réussies". Bien sûr, il y en a beaucoup plus.

6. **`server.listen`**

   - **`server.listen ()`** commence à écouter les requêtes entrantes sur le numéro de port et l'hôte passés dans les premier et deuxième arguments.
   - Le troisième argument est un rappel qui est appelé une fois que le serveur commence à écouter. Nous enregistrons un message afin de savoir que la configuration du serveur est terminée avec succès.

4. Gérer les demandes entrantes

- Il existe peu de méthodes attachées aux objets `req` et `res`. Nous pouvons les utiliser pour effectuer différentes opérations. Voyons comment:

- Ici, nous vérifions l'**url** de l'objet de requête et si sa valeur est "text" , nous retournerons une réponse sous forme de chaîne en utilisant `res.end()`. Voyons la sortie que nous obtenons sur le navigateur pour cette réponse.

- Mais le texte brut n'est pas le seul format que nous pouvons utiliser pour envoyer une réponse. Les formats **HTML**, **JSON** et **CSV** peuvent également être envoyés de la même manière.

Sommaire du cours Node.js

Source : https://apcpedagogie.com/creer-un-serveur-web-en-node-js · téléchargé le 05/11/2026 · © APC Pédagogie · apcpedagogie.com