Node - Express

Introduction générale

Node Express, c’est quoi ?

Node.js

- Node.js est un environnement d'exécution JavaScript côté serveur.
- C’est donc du JavaScript, mais exécuté sur le serveur plutôt que dans le navigateur.

- Node.js permet de faire ce qu’on fait normalement avec un langage serveur, mais avec JavaScript.
- De plus, il permet d’exécuter facilement un serveur WEB, jouant ainsi un rôle similaire à celui d'Apache, mais avec une approche différente.

- Techniquement :
  - Node.js est basé sur le moteur JavaScript V8 de Google Chrome :
    - Ca qui lui permet d'être rapide et efficace dans l'exécution de code.
  - Node.js gére des applications asynchrones et non-bloquantes :
    - C’est adapté pour des applications web en temps réel, telles que les chats en ligne, les jeux multi-joueurs ou des applications de streaming.
  - Le JavaScript de Node.js suit Ła normę EcmaScript 2015 (ES6) et les versions suivantes.
    - C’est un codage objet standard et « propre ».
    - Le vieux JavaScript côté client, tel qu'on le connaissait auparavant, est quasiment obsolète aujourd'hui !"

Node Express, c’est quoi ?

Express.js

- Express.js est un **micro-framework Nodes.js** puissant et extensible.
  - Il est **comparable à Flask** en Python.
- Il permet de développer des **applications WEB naturellement avec routage** et avec **architecture MVC si on le souhaite**.
- Il est **surtout utilisé pour développer des services d’API**.

- C'est un **excellent choix pour les projets WEB** qui nécessitent rapidité, simplicité et évolutivité.
- C’est un excellent choix pour le **développement d’API RESTful**.

- Express bénéficie d’une **large communauté**.
- Express permet la **scalabilité** : gérer facilement avec certains outils l’ajout automatique de serveurs pour gérer l’augmentation du trafic (le load balancing).
- Express utilise des « **middlewares** » (une couche logiciel intermédiaire) qui permet de mieux organiser le code qu’avec un MVC classique.

Node Express, c’est quoi ?

Autour de Node.js : React

- React est une bibliothèque JavaScript open-source développée par Meta (anciennement Facebook).
- React sert principalement côté client.
- React permet de construire des interfaces utilisateur (UI) interactives, notamment des petites applications intégrées dans une page, principalement pour des applications web et mobiles.

- React permet de créer des « composants » d’interface pour les navigateurs.
  - Ces composants peuvent être réutilisés et gèrent l'affichage et l'interactivité de l'application.
  - Ca rend la création d'interfaces dynamiques plus simple et plus efficace.

Node Express, c’est quoi ?

Autour de Node.js : Angular

- Angular est un framework côté client développé par Google.
- Il permet de développer des applications WEB mono page (Single Page Application : SPA) avec routage et avec architecture MVC ou équivalent (MVVM)
- Il peut se comparer à des frameworks full-stock comme Django ou à Laravel, mais il en est spécifiquement conçu pour être côté client.

- Il utilise le langage TypeScript qui est un sur-ensemble de JavaScript qui est fortement typé et permet une gestion plus facile du code dans des applications complexe.

Une API RESTful, c’est quoi ?

- Une API est est un moyen simple, flexible, extensible et standard qui permet à une application de donner accès à ses données à d’autres applications, en consultation ou en modification en utilisant des méthodes HTTP standard (GET, POST, PUT, DELETE, etc.).
- Le format des données échangé est le plus souvent du JSON : c’est un format léger, facile à comprendre et à implémenter.

- Une API RESTful (ou simplement API REST) est API qui respecte les principes de l'architecture REST (Representational State Transfer). Voici les 3 principes clés d'une API RESTful :
- 1. Les requêtes utilisent des méthodes HTTP :
  - GET : Pour récupérer des données (ex : obtenir des informations sur un utilisateur).
  - POST : Pour envoyer des données (ex : créer une nouvelle ressource).
  - PUT : Pour mettre à jour une ressource existante (ex : modifier les informations d'un utilisateur).
  - DELETE : Pour supprimer une ressource (ex : supprimer un utilisateur).
- 2. Les requêtes sont «stateless (sans état) :
  - Chaque requête envoyée à une API RESTful doit contenir toutes les informations nécessaires à son traitement :
    - Le serveur ne conserve aucune donnée entre deux requêtes, garantissant ainsi que chaque requête est indépendante.
- 3. Les ressources identifiées par des URL :
  - Les ressources (objets, données, services) sont identifiées par des URL. Par exemple :
    - /users pour accéder à la liste des utilisateurs.
    - /users/1 pour accéder aux informations du premier utilisateur.

Express et base de données

- Express permet d'accéder à des bases de données.

- A partir d'une URL (une route), Express permet de gérer facilement les requêtes d'une API :
    - GET : Pour récupérer des données (ex : obtenir des informations sur un utilisateur).
    - POST : Pour envoyer des données (ex : créer une nouvelle ressource).
    - PUT : Pour mettre à jour une ressource existante (ex : modifier les informations d'un utilisateur).
    - DELETE : Pour supprimer une ressource (ex : supprimer un utilisateur).

- La base de données pourra être SQL (par exemple MySQL avec la bibliothèque mysql2) ou NoSQL (par exemple MongoDB avec la bibliothèque mongoose).

Bases du JavaScript de Node.js

JavaScript d'origine :

- 1995 - EcmaScript-1997 - ES-1

JavaScript moderne :

- EcmaScript-2015 - ES-6
  - let, const,
  - fonction fléchée,
  - déstructuration,
  - classe,
  - mode strict

➡ http://bliaudet.free.fr/IMG/pdf/JS-ES6-ES2015.pdf

Exemple de code Node.js

```
// Importation du module http de Node.js
const http = require('http');

// Création du serveur
const server = http.createServer((req, res) => {
  // Définit l'en-tête HTTP (code de statut et type de contenu)
  res.writeHead(200, { 'Content-Type': 'text/plain' });

  // Envoi de la réponse "Hello World" au navigateur
  res.end('Hello World Node.js');
});

// Le serveur écoute sur le port 3000
server.listen(3000, () => {
  console.log('Serveur en cours d\'exécution sur http://
localhost:3000');
});
```

➡Tester ce code : bliaudet.free.fr/IMG/txt/app_node_de_base.js

➡On pourrait avoir du HTML plus subtil avec ‘text/html’

Exemple de code Express

```
// Importation du module Express
const express = require('express');

// Création de l'application Express
const app = express();

// Route qui renvoie "Hello World" lorsque l'utilisateur accède à la
racine
app.get('/', (req, res) => {
  res.send('Hello World Express');
});

// Le serveur écoute sur le port 3000
app.listen(3000, () => {
  console.log('Serveur en cours d\'exécution sur http://
localhost:3000');
});
```

➡ Tester ce code : bliaudet.free.fr/IMG/txt/app_express_de_base.js

Exemple de code Express - 2

```
// Importation du module Express
const express = require('express');

// Création de l'application Express
const app = express();

// Route pour la première page (page 1)
app.get('/', (req, res) => {
  res.send(`
    <html>
      <body>
        <h1>Page 1</h1>
        <ul>
          <li><a href="/">Page 1</a></li>
          <li><a href="/page2">Page 2</a></li>
        </ul>
        <p>Bienvenue sur la première page !</p>
      </body>
    </html>
  `);
});
```

Exemple de code Express - 2, suite

```
// Route pour la deuxième page (page 2)
app.get('/page2', (req, res) => {
  res.send(`
    <html>
      <body>
        <h1>Page 2</h1>
        <ul>
          <li><a href="/">Page 1</a></li>
          <li><a href="/page2">Page 2</a></li>
        </ul>
        <p>Bienvenue sur la deuxième page !</p>
      </body>
    </html>
  `);
});

// Le serveur écoute sur le port 3000
app.listen(3000, () => {
  console.log('Serveur en cours d\'exécution sur http://localhost:3000');
});
```

➡ Tester ce code : bliaudet.free.fr/IMG/txt/app_express_de_base_2.js