TME 6 – Node et Express

TME 6 – Node et Express

Ce TME a pour objectif d'aborder le fonctionnement de base d'Express :
- installation
- routage
- réponse à des requêtes basiques
- gestion du JSON

5.1 Installation

Nous n'allons pas faire de React dans le cadre de ce TME, car nous abordons la partie serveur. Pour cela, il faut mettre en place le code qui permettra de répondre aux requêtes HTTP.

1. créez un répertoire nommé `TME6`
2. **placez-vous dans ce répertoire** (`cd TME6`)
3. tapez `npm init` pour initialiser un nouveau projet dans ce répertoire
4. tapez `npm install express`. Si l'installation ne se passe pas correctement, supprimez d'autres répertoires `node_modules` créés par ailleurs (par exemple lors du TME précédent). Ne vous inquiétez pas, l'essentiel est de ne jamais supprimer le fichier `package.json` qui contient tout le nécessaire pour installer un projet et re-télécharger les modules node nécessaires.

Vous pouvez regarder rapidement ce qui a été installé dans le répertoire `node_modules`.
Pour plus de facilité, nous allons aussi installer le module `nodemon`. Ce module permet de ne pas avoir à *tuer* node à chaque modification d'un fichier du projet en redémarrant automatiquement le serveur.

1. tapez `npm install nodemon`
2. dans le fichier package.json, ajoutez la ligne `"start": "nodemon index.js",` (sans oublier la virgule, c'est au format JSON !) juste avant la ligne commençant par `"test"`
3. créez un fichier index.js, contenant la ligne `require('./app/index1.js');`
4. créez un fichier index1.js dans le répertoire app. C'est dans ce fichier que vous allez placer le code du TME

Cela donne la possibilité de conserver ce fichier pour garder une trace des exercices successifs, et de reprendre le TME dans un autre fichier, par exemple app/index2.js, en écrivant dans index.js la ligne `require('./app/index2.js');`.

5.2 Site de base

Dans ce premier exercice, nous n'utiliserons que des méthodes `GET`, ce qui permet de visualiser le résultant dans une fenêtre de navigateur.

1. Affichez le texte "Tout va à merveille" en réponse à une requête sur le port 8000 de localhost. Précisez l'encodage de la réponse.
2. Affichez le texte "Cette page n'existe pas" en cas d'erreur 404.
3. Affichez le texte "Tout va à merveille pour cette page 1" sur la page localhost :8000/test1
4. Affichez le texte "Tout va à merveille pour cette page xxx" sur la page localhost :8000/essaixxx, où xxx désigne une chaîne de caractères quelconque.

Importez maintenant le module `path` (en ajoutant `const path = require('path');`). Créez dans le répertoire `app` un répertoire `public` dans lequel vous placerez le fichier test.html suivant :

```
<!DOCTYPE html>
<html lang="fr">
<head>
        <meta charset="utf-8" />
        <title>Fichier statique</title>
</head>
<body>
        <p>Page test<p>
</body>
</html>
```

En vous servant notamment de la documentation de la méthode sendFile et de cette ressource, définissez une route permettant d'envoyer ce fichier index.html en réponse à un GET de l'URL `http://localhost:8000/test.html`.

5.3 Utilisation de Router et de static

La situation précédente a comme inconvénient d'exposer les chemins sur le serveur pour accéder aux fichiers statiques. Il vaut mieux utiliser le *middleware* `static`.

- utilisez `Router` pour réécrire les routes précédentes
- en consultant la documentation d'Express, utilisez le *middleware* `static` pour éviter d'avoir à expliciter le chemin lors du routage.

5.4 Réception et décodage d'une requête

En vous inspirant des exemples donnés en cours, mettre en place un serveur écoutant le port 8000, et répondant à une requête POST :

- postée à l'URL `http://localhost:8000/api`
- possédant un body en JSON valant `{"name": "test"}`

Le résultat est l'affichage dans la console du serveur du contenu de la propriété `name`
Tester la requête avec Postman. Que faut-il ajouter aux instructions précédentes ? Pourquoi ?
Complétez la requête en envoyant cette fois-ci dans le body

```
{
        "name":"test",
        "nb1": 3,
        "nb2": 5
}
```

À l'aide de la méthode `json` de l'objet réponse `res`, renvoyer au client le JSON

```
{
    "addition": 8,
    "prop": "hello"
}
```

où `addition` est calculé à partir de `nb1` et `nb2`.

5.5 Gestion des erreurs de requête

Complétez le code précédent pour :

- renvoyer le code d'erreur 400 et le message "Requête invalide : name nécessaire" si la propriété `name` n'est pas indiquée dans le body de la requête
- renvoyer le code d'erreur 400 et le message "Requête invalide : nb1 et nb2 doivent être des nombres" si `nb1` ou `nb2` n'est pas un nombre

5.6 Utilisation d'une session

Nous allons voir comment utiliser le *middleware* express-session.

1. installez le module express-session
2. en vous inspirant de cet exemple de base, testez si une variable de session `user` est définie.

— Si ce n'est pas le cas, l'initialiser avec par exemple votre prénom, et afficher dans le navigateur un message de bienvenue.
— Si c'est le cas, afficher "Déjà de retour, " puis la variable de session dans le navigateur.