Routage de base à l'aide d'Express Node.js

1. Objectifs

- Connaitre le routage avec express et Node.js
- Être capable de gérer les routes et les méthodes de requête HTTP dans un projet Express

2. Présentation

- Le **routage**, même s'il est possible, peut devenir assez complexe si le projet prend de l'ampleur. Il est alors plus efficace de recourir à un Framework spécialisé.
- La Fondation Node.js développe ainsi le Framework Express qui permet, par exemple de gérer des URL sur la base d'expressions rationnelles.
- **Le routage**: c'est l'interception d'une requête client (via une méthode HTTP) et la rediriger vers un composant capable de retourner une réponse
- Le routage définit la manière dont les demandes des clients sont traitées par les points de terminaison de l'application.
- **Routage** fait référence à la détermination de la façon dont une application répond à un nœud final spécifique, c’est-à-dire un URI (ou chemin) et une méthode de requête HTTP (GET, POST, etc.).
- Chaque route peut avoir une ou plusieurs fonctions de gestionnaire, qui sont exécutées lorsque la route est mise en correspondance.
- Il existe deux façons d'implémenter le routage dans node.js qui sont répertoriées ci-dessous:
    1. Sans utiliser Framework : un fichier de routing sera crée manuellement pour gérer le routage dans un projet node.js
    2. En utilisant Framework : Node dispose de nombreux Framework pour vous aider à faire fonctionner votre serveur. Le plus populaire est Express.js.

3. Routage de base avec Express.js

- **Express.js** a un objet "app" correspondant à HTTP. Nous définissons les routes en utilisant les méthodes de cet objet "app".
- Cet objet d'application spécifie une fonction de rappel, qui est appelée lorsqu'une demande est reçue. Nous avons différentes méthodes dans l'objet d'application pour un type de requête différent.

apcpedagogie.com

- **Syntaxe:**
  - La définition de la route utilise la structure suivante :
    - `app.METHOD(PATH, HANDLER)` ou de la même façon : `app.method (chemin, gestionnaire)`
    - Avec :
      - `app` est une instance d'express.
      - `METHOD` est une méthode de demande HTTP.
      - `PATH` est un chemin sur le serveur, le chemin est la route sur laquelle la requête s'exécutera.
      - `HANDLER` est la fonction exécutée lorsque la route est mise en correspondance.
- **Exmples:**
  - `app.js`

```
app.get ('/', fonction (req, res) {
    res.send ("Bonjour tout le monde!");
});
```

  - Cela produirait **Bonjour tout le monde!** lorsque les utilisateurs accèdent à l'index ou à la page d'accueil.
- `app.js`

```
app.get ('/monSite', fonction (req, res) {
    res.send ("notre site apcpedagogie");
});
```

- Cela produirait "**notre site apcpedagogie**" lorsque les utilisateurs accèdent à la route/monSite.
- Les exemples suivants illustrent la définition de routes simples.

4. **Les appels chaînés**
   - L'avantage de méthode **get** est qu'elle renvoie à son tour un objet similaire ; les appels peuvent donc être chaînés. Par exemple...

apcpedagogie.com

```
app.get('/page1', function(req, res){
  res.send('Ceci est la page 1') ;
}) ;
.get('/page2', function(req, res){
  res.send('Ceci est la page 2') ;
}) ;
.get('/page3', function(req, res){
  res.send('Ceci est la page 3') ;
}) ;
```

5. **Routage dynamique avec Express.js**
   - Express permet de gérer des routes dynamiques, c'est-à-dire des routes dont certaines parties sont variables. Pour reprendre l'exemple précédent avec page1, page2 et page3, on aurait pu écrire...

```
app.get('/page:num', function(req, res){
  res.send('Ceci est la page '+req.params.num) ;
}) ;
```

6. `app.js`

```
app.get ('/user/: nomUtilisateur', function (req, res) {
    res.send (" Le profile de : " + req.params.nomUtilisateur);
});
```

7. Lorsque nous demandons des informations sur un utilisateur particulier en utilisant son nom d'utilisateur dans l'URL, il récupère le nom d'utilisateur à l'aide de l'objet de demande et affiche le message approprié.
8. **Exemple:** si l'utilisateur demande **/user/Mohamed**, il affichera le profil de Mohamed!
9. **Gestion des erreurs 404**
   - Express permet aussi de gérer facilement les entêtes http, et notamment les réponses du serveur en cas d'erreur, la plus connue étant l'erreur 404 (lorsqu'une ressource ne peut pas être trouvée sur le serveur). Pour cela, à la toute fin du code, juste avant app.listen, il faut inclure le code suivant :

apcpedagogie.com

```
app.use(function(req, res, next){
  //la ligne suivante spécifie l'encodage de la réponse
  res.setHeader('Content-Type', 'text/plain; charset=UTF-8') ;
  //La ligne suivante spécifie le message à envoyer par le serveur quand la réponse est un
code d'erreur 404
  res.status(404).send('Page introuvable') ;
})

.listen(3000);
```

10. Activité

1. Créez un nouveau projet node à l'aide de npm init
2. Ajoutez Express comme dépendance.
3. Affichez le texte "Bonjour tout le monde" en réponse à une requête sur le port 3000 de localhost. Précisez l'encodage de la réponse.
4. Affichez le texte "Je suis la page numéro 2" en réponse à une requête sur sur la page **localhost:3000/page1**.
5. Affichez le texte "Cette page n'existe pas" en cas d'erreur 404.
6. Affichez le texte "Tout va à merveille pour cette page xxx" sur la page localhost:8888/essaixxx, où xxx désigne une chaîne de caractères quelconque.

Sommaire du cours Node.js

Source : https://apcpedagogie.com/routage-de-base-a-laide-dexpress-node-js · téléchargé le 05/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com