TP 3 : Routage avec Express

Méthodes de requête HTTP

HTTP définit un ensemble de méthodes de requête qui indiquent l'action que l'on souhaite réaliser sur la ressource indiquée. Bien qu'on rencontre également des noms (en anglais), ces méthodes sont souvent appelées verbes HTTP. Chacun d'eux implémente une sémantique différente mais certaines fonctionnalités courantes peuvent être partagées par différentes méthodes (e.g. une méthode de requête peut être sûre (safe), idempotente ou être mise en cache (cacheable)).

GET

La méthode GET demande une représentation de la ressource spécifiée. Les requêtes GET doivent uniquement être utilisées afin de récupérer des données.

POST

La méthode POST est utilisée pour envoyer une entité vers la ressource indiquée. Cela entraîne généralement un changement d'état ou des effets de bord sur le serveur.

PUT

La méthode PUT remplace toutes les représentations actuelles de la ressource visée par le contenu de la requête.

DELETE

La méthode DELETE supprime la ressource indiquée.

Exercice 1:

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 la requête localhost:3000/accueil.

4) Affichez le texte "Je suis la page numéro 2" en réponse à une requête sur la page localhost:3000/page1.

4) Affichez le texte "Cette page n'existe pas" en cas d'erreur.

5) Affichez le texte "Tout va à merveille pour cette page xxx" sur la page localhost:3000/essaixxx, où xxx désigne une chaîne de caractères quelconque.

6) Créer des vues pour votre projet.

Solution :

```
var express = require("express"); //appel au module express
var app = express(); //création d'une instance de type express
//app.use(express.static("public")); /* Cette ligne de code indique
Application Server que vous voulez utiliser le dossier public pour stocker
des données statiques, l'utilisateur peut accéder aux fichiers dans ce
dossier.*/

app.set("view engine", "ejs"); /*Cette ligne de code indique  Application
Server que vous voulez utiliser la bibliothèque EJS. C'est une machine pour
gérer votre page. Le EJS créera du HTML pour retourner au navigateur de
l'utilisateur.*/
app.set("views", "./views"); /*Cette ligne de code indique  Application
Server le chemind'accès au dossier contenant vos pages*/

app.listen(3000); /*Votre application écoutera sur le port 3000 quand elle
est déployée.*/

app.get("/home", function(request, response)  { /* définition de chemin en
réponse à une requête de type GET */
  // response.send("je suis la page d'accueil");
   response.render("acc"); /* vue retourné */
});

app.get("/page1", function(request, response)  {
   // response.send("je suis la page 1");
    response.render("page1"); /* vue retourné */
}
);
app.get("/essai:xxx", function(request, response)  {
   response.send('tout va à merveille dans cette page
essai'+request.params.xxx);
    //response.render("testpage"); /* vue retourné */
}
);
app.use(function(request, response)  {
    //response.send('Cette page n'existe pas');
    response.render("error");
});
```

Exercice 2:

1) En utilisant le routage dynamique du module express de nodeJS écrire un programme qui permet consulter l'ensemble des pages web d'un site composé de :

Page d'accueil

Page contenant une liste d'utilisateur

Page accessible à partir d'un pattern noxxx

2) Ajouter des vues à votre application en utilisant le template EJS

Solution :

```
var express = require('express');
var app = express();

// This responds with "Hello World" on the homepage
app.get('/', function (req, res) {
   console.log("Got a GET request for the homepage");
   res.send('Hello GET');
})

// This responds a POST request for the homepage
app.post('/', function (req, res) {
   console.log("Got a POST request for the homepage");
   res.send('Hello POST');
})

// This responds a DELETE request for the /del_user page.
app.delete('/del_user', function (req, res) {
   console.log("Got a DELETE request for /del_user");
   res.send('Hello DELETE');
})

// This responds a GET request for the /list_user page.
app.get('/list_user', function (req, res) {
   console.log("Got a GET request for /list_user");
   res.send('Page Listing');
})

// This responds a GET request for abcd, abxcd, ab123cd, and so on
app.get('/no:vvv', function(req, res) {
   console.log("Got a GET request for /ab*cd");
   res.send('Page Pattern Match');
})
app.get('/test', function (req, res) {
    console.log("Got a GET request for /test");
    res.send("<html> <head> <title>Home</title></head><body><form name='f1'
action='/' method='post'><input type='text'></input><input
type='submit'></input> </form><form name='f2' action='/'
method='delete'><input type='submit'></input> </form></body></html>");
 })
```

```
var server = app.listen(8081, function () {
   var host = server.address().address
   var port = server.address().port

   console.log("Example app listening at http://%s:%s", host, port)
})
```