Programmation avancée

Node.js - MongoDB

Sylvie Trouilhet - Jean-Marie Pecatte

S4 – UE2

Utiliser Express

et

route

middleware

moteur de vues

body-parser

Express

- Facilite la conception d'un serveur http pour une API REST

- Différents services proposés à un utilisateur
- Une URI pour chaque ressource
- POST GET PUT DELETE

https://expressjs.com/fr/starter/basic-routing.html

npm **install express** --save

- Express pour la création du serveur d'application :

```
const express = require('express');
const app = express();
…
app.listen(4000, ()=>{console.log('ok');});
```

- routage :

```
app.get(route, callback);
function callback(request, response) {…}
```

- route : correspond à une URI (par exemple `'/index.html'`).
  Pour cette route, la fonction de callback est déclenchée.
  Pour toute autre route, l'application répond **`cannot GET`**

Essayer avec la fonction de callback pour la route « / »:

```
app.get('/', sendResponse);

function sendResponse(req, res) {
      res.send('Bonjour MMI!');
       }
```

- Expression régulière pour une url

- app.get(/index\w*/, callback)

  \w : caractère alphanumérique [a-zA-Z0-9_]

  * : 0 à n fois

  + : 1 à n fois

Routage de base

app.get(chemin, function (req, rep) {})

app.post(chemin, function (req, rep) {})

app.put(chemin, function (req, rep) {})

app.all(chemin, function (req, rep) {}) : pour spécifier un traitement pour une route, quelle que soit la méthode

app.use([chemin], function (req, rep) {})

→ function (req, rep) {} : « middleware function »

http://expressjs.com/en/guide/using-middleware.html

- *Middleware* functions are functions that have access to the request object (req), the response object (res), and the next middleware function in the application's request-response cycle. The next middleware function is commonly denoted by a variable named next.
- Middleware functions can perform the following tasks:
  - Execute any code.
  - Make changes to the request and the response objects.
  - End the request-response cycle.
  - Call the next middleware function in the stack.

Exemple

```
const express = require('express');
const app = express();
app.listen(4000);

app.use(function (req, res) {
  console.log('Jour : ', new Date().getDate());
});
app.use('/fr.html', function (req, res) {
  res.send('Salut MMI');
});
app.use('/gb.html', function(req, res) {
  res.send('Hello World');
});
```

Exemple avec next

```
const express = require('express');
const app=express();
app.listen(4000);
app.use(function (req, res, next) {
  console.log('Jour : ', new Date().getDate());
  next();
});
app.use('/fr.html', function (req, res) {
  res.send('Salut MMI');
});
app.use('/gb.html', function(req, res) {
  res.send('Hello World');
});
```

Fonction middleware

- Route paramétrable avec ' :param'

exemple

```
app.use('/fr.html/:region', (req, res, next) => {
        console.log('region :', req.params.region);
        next();
        });
```

Demande client : http://localhost:4000/fr.html/france

Moteur de vues : ejs|hbs|hjs|jade|pug|twig|vash

- view (modèle MVC):

  afficher un document HTML à partir du modèle de données

  modules **`pug`** ou **`ejs`** (Embedded JavaScript) comme moteur de vues

- ejs : module à installer avec npm

Ejs (www.embeddedjs.com)

- La vue se trouve dans un fichier .ejs dans le répertoire **views**
- Les éléments variables sont indiqués par des balises :

```
<%=  nom  %>
```

- Dans le fichier .js:

```
app.set('view engine', 'ejs');
```

les balises sont remplacées par les valeurs indiquées entre accolades **`{nom: valeur}`**

```
response.render(fichier, valeurs);
```

Exemple

- /views/reponse.ejs

```
<!DOCTYPE html>
<head> … </head>
<body>
…
<h1>Un exemple de vues</h1>

<p>Vous êtes le joueur <%= nom %> </p>
<p>et la proposition est <%= proposition %> </p>
…
</body>
```

- Fichier index.js

```
const express = require('express');
…
app.set('view engine', 'ejs');
…
app.get('/index.html', construireVue);
function construireVue(request, response){
 response.render('reponse.ejs',{nom:'Joueur1', proposition:10});
 }
```


localhost:4000/index.htm
localhost:4000/index.html
Un exemple de vues
Vous êtes le joueur Joueur1
et la proposition est 10

Module body-parser

- Récupérer les valeurs d'un formulaire avec la méthode POST : par exemple `'nom=toto'`
- C'est dans le body de la requête :

```
let body = '';
request.on('data', function (data) {
      body += data;
    });
```

- Analyse : **`body-parser`**

```
const bp = require('body-parser');
app.use(bp.urlencoded({extended: false}));
app.use(bp.json());
                    // request.body.nom → 'toto'
```

Application

- Reprendre le projet du jeu pierre-feuille-ciseau en utilisant les routes et les fonctions middlewares :

- créer un projet et décider des répertoires
  ./views
  ./public/javascripts
  ./public/stylesheets

- énumérer les routes

- créer les différents fichiers