INF4375 - Paradigmes des échanges Internet

Express.js

Jacques Berger

Objectifs

Introduire un framework Javascript pour le traitement des requêtes HTTP

Prérequis

Node.js

Express.js

Express.js est un framework pour Node.js facilitant le traitement des requêtes HTTP

Il simplifie le routing d'URL, la manipulation des requêtes et des réponses HTTP et fournit une structure de projet

Express.js

Version courante : 4.16.1

Installation

Express.js s'installe avec npm

```
npm install express
```

Installation

Un logiciel supplémentaire peut être installé pour générer un squelette de projet Express.js

```
npm install -g express-generator
```

Il est idéal d'installer ce module globalement

Programme express

Le programme express permet de générer un squelette de projet fonctionnel

Par exemple, la commande suivante génère un projet Express.js avec Pug :

```
express --pug
```

Programme express

Express.js utilise Jade par défaut pour le templating du HTML

Il est possible de le changer au besoin

Notamment, Jade a changé de nom et l'outil se nomme maintenant Pug

Programme express

Express.js utilise le CSS directement pour le styling de l'application web

On peut le changer pour Stylus (code CSS simplifié)

Structure du projet

Lorsque le projet est généré avec le programme express, Express.js nous propose une structure de projet

Cette structure n'est pas obligatoire

Structure du projet

/package.json : Contient toutes les dépendances du projet, essentiel pour le premier npm install

/app.js : Ce fichier contient toutes les configurations du serveur Express.js

Structure du projet

/bin/www : L'exécutable du projet

/public/ : Contient ce qui devrait être consommé par le fureteur; tous les fichiers statiques

/public/images/ : Contient les images du site web

Structure du projet

/public/javascripts/ : Contient les scripts Javascript destinés à être interprété et exécuté par le fureteur

/public/stylesheets/ : Contient les feuilles de style de l'application

Structure du projet

/routes/ : Contient le code pour traitement les routes; c'est-à-dire le code du back-end

/views/ : Contient les templates pour la génération du HTML

app.js

Dans app.js, on configure le serveur

Tout d'abord, le require :

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

app.js

Ensuite, on crée le serveur :

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

Une fois créé, on peut le configurer avec des app.set(), app.use() et les définitions des routes

bin/www

Le fichier bin/www est le point d'entrée du projet

On y lance l'écoute du serveur

```
app.listen(3000);
```

Configuration du serveur

Le programme express s'occupe de configurer le serveur pour nous lors de la création du app.js

Néanmoins, si on veut ajouter une fonctionnalité par la suite, nous devrons modifier le fichier app.js nous-même

Configuration du serveur

Avec app.set(), on spécifie les différentes propriétés du serveur

Avec app.use(), on spécifie les différentes fonctionnalités que le serveur offrira

Routes

Une route est une méthode HTTP et une URL que l'application va reconnaître

Par exemple :
GET + '/' retourne la page d'accueil
GET + '/INF4375' retourne la page d'INF4375
POST + '/INF4375/article' ajoute du contenu

Routes

Au lieu de devoir vérifier la méthode HTTP et le contenu de l'URL nous-même à chaque requête, Express.js nous fournit une fonctionnalité pour simplifier ce traitement

Routes

Dans le répertoire routes, on définit les routes qu'on veut supporter selon le format suivant :

```
router.methodeHttp(url, callback);
```

Par exemple :

```
router.get('/inf4375', function(req, res) {
    ...
});
```

Routes

Le callback passé en paramètre à une route doit recevoir deux paramètres : un objet requête et un objet réponse

```
function(req, res){
…
}
```

Réponse

La génération de la réponse se fait dans les routes, avec l'objet res

On peut spécifier un en-tête HTTP sur la réponse, par exemple pour préciser le MIME-type des données

```
res.header("Content-Type", "application/xml");
```

Réponse

On peut envoyer le résultat d'un template Pug

```
res.render('index');
```

On peut également spécifier des données au template

```
res.render('index', data);
```

Réponse

Il existe une fonctionnalité spécialement conçu pour envoyer des données en format JSON, nous évitant ainsi un appel à JSON.stringify

```
res.json(data);
```

Réponse

Sinon, on peut envoyer n'importe quel type de données avec la méthode générique send()

```
res.send("Cette page n'est pas disponible.");

res.status(500).send('Internal error');
```

Fonctionnalités

Les fonctionnalités offertes par Express.js vont plus loin que la portée de cette présentation

Notamment, tous les concepts habituels touchant HTTP sont couverts par Express.js

- Cookies, paramètres à l'URL, etc.

Consultez la documentation

Conclusion

L'utilisation d'un framework comme Express.js permet de simplifier grandement le code du serveur

Également, son intégration d'un langage de templating comme Pug permet également d'avoir du code plus propre dans le projet

Plus loin...

Express.js

http://expressjs.com/