Express.js : introduction

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en Programmation par contrainte (IA)
Ingénieur en Génie logiciel

elmouelhi.achref@gmail.com

Plan

Express.js

ExpressJS ou **Express.js** ou **Express**

- Framework **Node.js**
- Permettant de :
  - construire des applications web
  - simplifier la création et la gestion des serveurs **Node.js**
  - faciliter le routage

Express.js

Autres frameworks **Node.js**

- Ionic
- NestJS
- ...

Express.js

Exemple d'applications créées avec **ExpressJS**

- Netflix
- Pinterest
- ...

Express.js

Commençons par initier le projet

```
npm init -y
```

Express.js

Commençons par initier le projet

```
npm init -y
```

Pour installer Express.js, il faut

```
npm install express
```

Express.js

Première étape : importer le package `http`

```
const http  = require('http');
```

Express.js

Première étape : importer le package `http`

```
const http  = require('http');
```

Deuxième étape : utiliser le module `http` pour créer un serveur : la fonction `createServer` prend en paramètre une fonction qui sera exécutée à chaque fois qu'on appelle le serveur

```
const server = http.createServer((request, response) => {
    response.write('Hello World!');
    response.end();
});
```

Express.js

Première étape : importer le package `http`

```
const http  = require('http');
```

Deuxième étape : utiliser le module `http` pour créer un serveur : la fonction `createServer` prend en paramètre une fonction qui sera exécutée à chaque fois qu'on appelle le serveur

```
const server = http.createServer((request, response) => {
    response.write('Hello World!');
    response.end();
});
```

On peut aussi fusionner `write` et `end`

```
const server = http.createServer((request, response) => {
    response.end('Hello World!');
});
```

Express.js

On peut aussi importer seulement la fonction `createServer` et l'utiliser

```
const { createServer }  = require('http');

const server = createServer((request, response) => {
    response.end('Hello World!');
});
```

Express.js

On peut aussi importer seulement la fonction `createServer` et l'utiliser

```
const { createServer } = require('http');

const server = createServer((request, response) => {
    response.end('Hello World!');
});
```

On peut également indiquer le type de la réponse et le code serveur

```
const { createServer } = require('http');

const server = createServer((request, response) => {
    response.writeHead(200, { 'Content-Type': 'text/plain' });
    response.end('Hello World!');
});
```

Express.js

Troisième étape : lancer le serveur pour écouter les requêtes sur le port 3000

```
server.listen(3000);
```

Express.js

Troisième étape : lancer le serveur pour écouter les requêtes sur le port 3000

```
server.listen(3000);
```

On peut aussi ajouter une fonction à exécuter pendant que le serveur est à l'écoute

```
server.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```

Express.js

Code final (à placer dans un fichier `app.js`)

```
const { createServer } = require('http');

const server = createServer((request, response) => {
    response.writeHead(200, { 'Content-Type': 'text/plain' });
    response.end('Hello World!');
});

server.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```

Express.js

Code final (à placer dans un fichier `app.js`)

```
const { createServer } = require('http');

const server = createServer((request, response) => {
    response.writeHead(200, { 'Content-Type': 'text/plain' });
    response.end('Hello World!');
});

server.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```

pour tester, lancer la commande `node app.js` et aller à l'URL `localhost:3000`

Express.js

Équivalent en `Express.js`

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

app.get('/', (request, response) => {
    response.send('Hello World!');
});

app.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```

Express.js

Équivalent en `Express.js`

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

app.get('/', (request, response) => {
    response.send('Hello World!');
});

app.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```

Pour tester, lancer la commande `node app.js` et vérifier que l'URL `localhost:3000` est accessible.

Express.js

Pour construire la réponse ensuite l'envoyer, on utilise `write` et `end`

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

app.get('/', (request, response) => {
    response.write('First Express Message');
    response.write('Hello World!');
    response.end('Hello World again!');
});

app.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```

Express.js

Pour construire la réponse ensuite l'envoyer, on utilise `write` et `end`

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

app.get('/', (request, response) => {
    response.write('First Express Message');
    response.write('Hello World!');
    response.end('Hello World again!');
});

app.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```

Pour tester, lancer la commande `node app.js` et aller à `localhost:3000`.

Express.js

La méthode `listen` permet aussi de spécifier une adresse IP différente de `localhost`

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

app.get('/', (request, response) => {
    response.write('First Express Message');
    response.write('Hello World!');
    response.end('Hello World again!');
});

app.listen(3000, '127.0.0.10',  () =>
    console.log('Adresse du serveur : http://127.0.0.10:3000')
);
```

Express.js

La méthode `listen` permet aussi de spécifier une adresse IP différente de `localhost`

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

app.get('/', (request, response) => {
    response.write('First Express Message');
    response.write('Hello World!');
    response.end('Hello World again!');
});

app.listen(3000, '127.0.0.10',  () =>
    console.log('Adresse du serveur : http://127.0.0.10:3000')
);
```

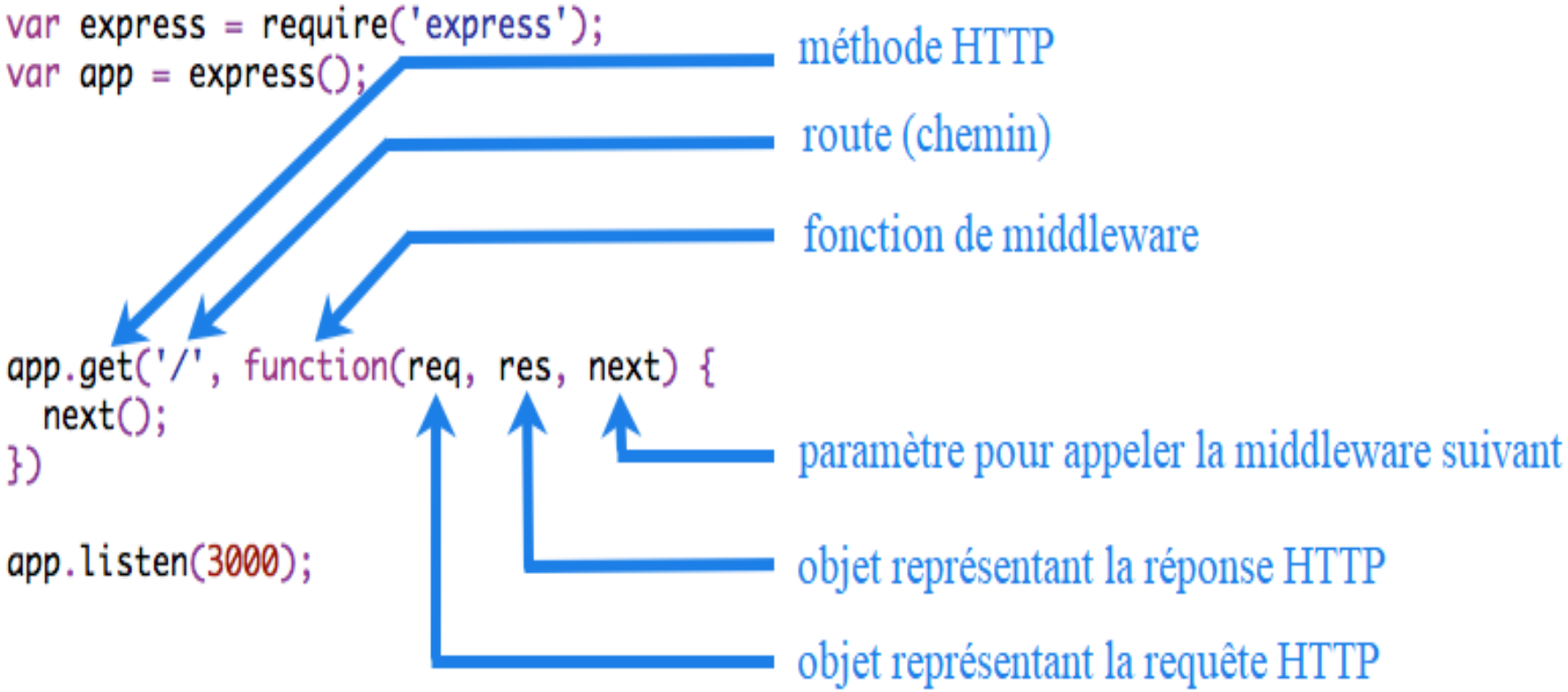
Pour tester, lancer la commande `node app.js` et aller à `127.0.0.10:3000`.

Express.js

Middleware

- Application **ExpressJS** = { fonctions } appelées **middlewares**
- **Middleware** :
  - Fonction ayant accès à l'objet `request` et `response`
  - Pouvant appelé le middleware suivant dans le cycle (désigné souvent par une variable nommée `next`)

Express.js


var express = require('express');
var app = express();
app.get('/', function(req, res, next) {
next();
})
app.listen(3000);
méthode HTTP
route (chemin)
fonction de middleware
paramètre pour appeler la middleware suivant
objet représentant la réponse HTTP
objet représentant la requête HTTP

Express.js

Déclaration de middleware

```
const middleware = (request, response, next) => {
    console.log("middleware:", request.url);
    next();
};
```

Express.js

Déclaration de middleware

```
const middleware = (request, response, next) => {
    console.log("middleware:", request.url);
    next();
};
```


Utilisation de middleware

```
app.use(middleware);
```

Intégrons le middleware dans `app.js`

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

const middleware = (request, response, next) => {
    console.log("middleware:", request.url);
    next();
};

app.use(middleware);

app.get('/', (request, response, next) => {
    console.log("requête reçue");
    response.send('Hello World!');
    next();
});

app.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```

Intégrons le middleware dans `app.js`

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

const middleware = (request, response, next) => {
    console.log("middleware:", request.url);
    next();
};

app.use(middleware);

app.get('/', (request, response, next) => {
    console.log("requête reçue");
    response.send('Hello World!');
    next();
});

app.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```

Lancer la commande `node app.js`, aller à `localhost:3000` et vérifier dans la console que `middleware: /` a été affiché avant `requête reçue`.

Dans `app.js`**, déplacer** `app.use` **après** `app.get`

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

const middleware = (request, response, next) => {
    console.log("middleware:", request.url);
    next();
};

app.get('/', (request, response, next) => {
    console.log("requête reçue");
    response.send('Hello World!');
    next();
});

app.use(middleware);

app.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```

Dans `app.js`**, déplacer** `app.use` **après** `app.get`

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

const middleware = (request, response, next) => {
    console.log("middleware:", request.url);
    next();
};

app.get('/', (request, response, next) => {
    console.log("requête reçue");
    response.send('Hello World!');
    next();
});

app.use(middleware);

app.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```

Lancer la commande `node app.js`, aller à `localhost:3000` et vérifier dans la console que `middleware: /` a été affiché après `requête reçue`.

Ou aussi

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

const middleware = (request, response, next) => {
    console.log("middleware:", request.url);
    next();
};

app.get('/', (request, response, next) => {
    console.log("requête reçue");
    response.send('Hello World!');
    next();
}, middleware);

app.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```

Ou aussi

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

const middleware = (request, response, next) => {
    console.log("middleware:", request.url);
    next();
};

app.get('/', (request, response, next) => {
    console.log("requête reçue");
    response.send('Hello World!');
    next();
}, middleware);

app.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```

Lancer la commande `node app.js`, aller à `localhost:3000` et vérifier dans la console que `middleware: /` a été affiché après `requête reçue`.

Dans `app.js`**, supprimer** `next()` **dans** `app.get`

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

const middleware = (request, response, next) => {
    console.log("middleware:", request.url);
    next();
};

app.get('/', (request, response, next) => {
    console.log("requête reçue");
    response.send('Hello World!');
});

app.use(middleware);

app.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```

Dans `app.js`**, supprimer** `next()` **dans** `app.get`

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

const middleware = (request, response, next) => {
    console.log("middleware:", request.url);
    next();
};

app.get('/', (request, response, next) => {
    console.log("requête reçue");
    response.send('Hello World!');
});

app.use(middleware);

app.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```

Lancer la commande `node app.js`, aller à `localhost:3000` et vérifier dans la console que seul le message `requête reçue` est affiché.

Express.js

Qu'affiche le programme suivant (dans la console) ?

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

const middleware1 = (request, response, next) => {
    console.log("middleware 1");
    next();
};
const middleware2 = (request, response, next) => {
    console.log("middleware 2");
    next();
};

app.use([middleware2, middleware1]);

app.get('/', (request, response, next) => {
    response.send('Hello World!');
});

app.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```

Qu'affiche le programme suivant (dans la console) ?

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

app.get('/', (request, response, next) => {
    response.send('Hello World!');
    next();
});
const middleware1 = (request, response, next) => {
    console.log("middleware 1");
    next();
};
const middleware2 = (request, response, next) => {
    console.log("middleware 2");
};
const middleware3 = (request, response, next) => {
    console.log("middleware 3");
    next();
};
app.use([middleware3, middleware2, middleware1]);

app.listen(3000, () =>
    console.log('Adresse du serveur : http://localhost:3000')
);
```