ExpressJS : 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

Plan

ExpressJS

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

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

ExpressJS

Autres frameworks **NodeJS**

- Ionic
- NestJS
- ...

ExpressJS

Exemple d'applications créées avec **ExpressJS**

- Netflix
- Pinterest
- ...

ExpressJS

Commençons par initier le projet

```
npm init
```

ExpressJS

Commençons par initier le projet

```
npm init
```

Pour installer Express.js, il faut

```
npm install --save express
```

ExpressJS

Première étape : importer le package `http`

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

ExpressJS

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();
});
```

ExpressJS

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!');
});
```

ExpressJS

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

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

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

ExpressJS

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!');
});
```

ExpressJS

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

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

ExpressJS

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')
);
```

ExpressJS

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')
);
```

ExpressJS

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`

ExpressJS

Équivalent en `ExpressJS`

```
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')
);
```

ExpressJS

Équivalent en `ExpressJS`

```
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.

ExpressJS

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')
);
```

ExpressJS

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')
);
```

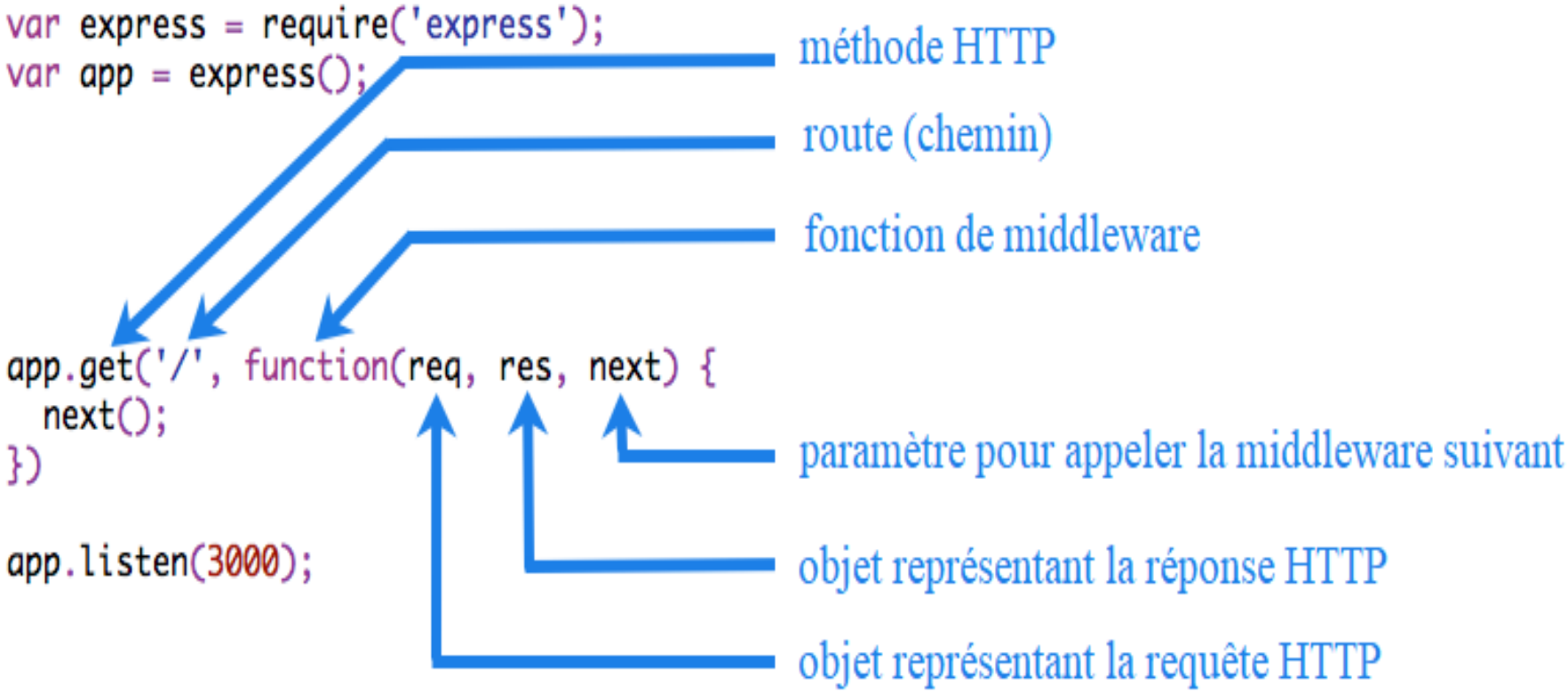
Pour tester, lancer la commande `node app.js` et aller à `localhost:3000`.

ExpressJS

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`)

ExpressJS


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

ExpressJS

Déclaration de middleware

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

ExpressJS

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.

ExpressJS

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')
);
```

ExpressJS

nodemon

- Package **NodeJS**
- Il surveille les modifications de fichiers sources
- Et il redémarre automatiquement le serveur de l'application
- Repository **GitHub** : `https://github.com/remy/nodemon`
- Documentation : `https://www.npmjs.com/package/nodemon`

ExpressJS

Installation

```
npm install -g nodemon
```

ExpressJS

Installation

```
npm install -g nodemon
```

Utilisation

```
nodemon [répertoire ou fichier]
```

ExpressJS

Routage

- Intercepter une requête client
- Vérifier la méthode **HTTP** utilisée
- Rediriger vers le composant associé pour retourner une réponse

ExpressJS

Deux routes sont accessibles : `/` et `/home`

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

app.get('/',  (req, res) => {
    res.send('GET: /');
});

app.get('/home',  (req, res) => {
    res.send('GET:  /home');
});

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

ExpressJS

Explication

- `GET` : méthode **HTTP**
- `/` et `/home` : les routes
- `res.send()` : méthode permettant de retourner une réponse (Autres méthodes disponibles : `redirect`, `render`...)

ExpressJS

Pour intercepter toutes les routes, on utilise `*`

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

app.get('/*',  (req, res) => {
    res.send('GET');
});

app.get('/home',  (req, res) => {
    res.send('GET: /home');
});

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

ExpressJS

Pour intercepter toutes les routes, on utilise `*`

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

app.get('/*',  (req, res) => {
    res.send('GET');
});

app.get('/home',  (req, res) => {
    res.send('GET: /home');
});

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

La route `/home` n'est plus accessible.

ExpressJS

L'ordre de définition de route est important

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

app.get('/home',  (req, res) => {
    res.send('GET: /home');
});

app.get('/*',  (req, res) => {
    res.send('GET');
});

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

ExpressJS

L'ordre de définition de route est important

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

app.get('/home',  (req, res) => {
    res.send('GET: /home');
});

app.get('/*',  (req, res) => {
    res.send('GET');
});

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

La route `/home` est accessible.

ExpressJS

Pour intercepter les requêtes HTTP de type `POST`

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

app.get('/',  (req, res) => {
    res.send('GET: /');
});

app.post('/home',  (req, res) => {
    res.send('POST:  /home');
});

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

ExpressJS

Pour tester les requêtes **HTTP** de type **POST**, il faut utiliser **Postman**

- Préciser la méthode `POST`
- Saisissez l'URL `http://localhost:3000/home`
- Cliquez sur `Send`

ExpressJS

Pour chaque verbe **HTTP**, il existe une méthode **ExpressJS**

- `put` pour **PUT**
- `delete` pour **DELETE**
- ...

ExpressJS

Pour retourner la même réponse quelle que soit le verbe HTTP

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

app.all('/all',  (req, res) => {
    res.send('GET: /');
});

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

ExpressJS

On peut utiliser les expressions régulières

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

app.get('/ab?cd',  (req, res) => {
    res.send(req.url);
});

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

ExpressJS

On peut utiliser les expressions régulières

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

app.get('/ab?cd',  (req, res) => {
    res.send(req.url);
});

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

Seules les routes `/acd` et `/abcd` sont accessibles.

ExpressJS

Pour récupérer la route demandée, on utilise l'objet `request`

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

app.get('/*',  (req, res) => {
    res.send('GET: ' + req.url);
});

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

ExpressJS

Pour récupérer la route demandée, on utilise l'objet `request`

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

app.get('/*',  (req, res) => {
    res.send('GET: ' + req.url);
});

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

En allant à `/personne`, le texte `GET: /personne` est affiché.

ExpressJS

Pour récupérer le verbe HTTP utilisé dans la requête

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

app.all('/all',  (req, res) => {
    res.send("http verb : " + req.method);
});

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

ExpressJS

Deux formes de paramètres de requête

- `/route/param1/param2` : à définir dans une route de la forme `/route/:param1/:param2`
- `/route?var1=value1&var2=value2` : à ne pas définir dans la route

ExpressJS

Deux formes de paramètres de requête

- `/route/param1/param2` : à définir dans une route de la forme `/route/:param1/:param2`
- `/route?var1=value1&var2=value2` : à ne pas définir dans la route

Deux objets différents permettant de récupérer les valeurs

- `params` pour `/route/param1/param2`
- `query` pour `/route?var1=value1&var2=value2`

ExpressJS

Première utilisation de l'objet `query`

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

app.get('/hello', (req, res) => {
    res.send(`Bonjour ${req.query['prenom']} ${req.query['nom'
       ]}`);
});

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

ExpressJS

Première utilisation de l'objet `query`

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

app.get('/hello', (req, res) => {
    res.send(`Bonjour ${req.query['prenom']} ${req.query['nom']}`);
});

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

Tester la route `http://localhost:3000/hello?nom=wick&prenom=john` et vérifier que `Bonjour john wick` s'affiche dans le navigateur.

ExpressJS

Deuxième utilisation de l'objet `query`

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

app.get('/hello', (req, res) => {
    res.send(`Bonjour ${req.query.prenom} ${req.query.nom}`);
});

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

ExpressJS

Deuxième utilisation de l'objet `query`

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

app.get('/hello', (req, res) => {
    res.send(`Bonjour ${req.query.prenom} ${req.query.nom}`);
});

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

Tester la route `http://localhost:3000/hello?nom=wick&prenom=john` et vérifier que `Bonjour john wick` s'affiche dans le navigateur.

ExpressJS

Remarque

La valeur par défaut pour les paramètres non présents dans la barre d'adresse est `undefined`.

ExpressJS

Troisième utilisation de l'objet `query`

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

app.get('/hello', (req, res) => {
    res.send(`Bonjour ${ req.query.couleurs }`);
});

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

ExpressJS

Troisième utilisation de l'objet `query`

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

app.get('/hello', (req, res) => {
    res.send(`Bonjour ${ req.query.couleurs }`);
});

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

Tester la route `http://localhost:3000/hello?couleurs[]=bleu&couleurs[]=blanc` et vérifier que `Bonjour bleu,blanc` s'affiche dans le navigateur.

ExpressJS

Quatrième utilisation de l'objet `query`

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

app.get('/hello', (req, res) => {
    res.send(`Bonjour ${req.query.personne.prenom} ${req.query.personne
       .nom}`);
});

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

ExpressJS

Quatrième utilisation de l'objet `query`

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

app.get('/hello', (req, res) => {
    res.send(`Bonjour ${req.query.personne.prenom} ${req.query.personne
       .nom}`);
});

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

Tester la route `http://localhost:3000/hello?personne[nom]=wick&personne[prenom]=john` et vérifier que `Bonjour bleu,blanc,rouge` s'affiche dans le navigateur.

ExpressJS

Première utilisation de l'objet `params`

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

app.get('/hello/:nom/:prenom', (req, res) => {
    res.send(`Bonjour ${req.params.prenom} ${req.params.nom}`);
});

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

ExpressJS

Première utilisation de l'objet `params`

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

app.get('/hello/:nom/:prenom', (req, res) => {
    res.send(`Bonjour ${req.params.prenom} ${req.params.nom}`);
});

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

Tester la route `http://localhost:3000/hello/wick/john` et vérifier que `Bonjour john wick` s'affiche dans le navigateur.

ExpressJS

Deuxième utilisation de l'objet `params`

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

app.get('/hello/:nom/:prenom', (req, res) => {
    res.send(`Bonjour ${req.params['prenom']} ${req.params['nom']}`);
});

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

ExpressJS

Deuxième utilisation de l'objet `params`

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

app.get('/hello/:nom/:prenom', (req, res) => {
    res.send(`Bonjour ${req.params['prenom']} ${req.params['nom']}`);
});

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


Tester la route `http://localhost:3000/hello/wick/john` et vérifier que `Bonjour john wick` s'affiche dans le navigateur.

ExpressJS

Remarque

Par défaut, les paramètres sont obligatoires, sinon la route ne sera pas matchée.

ExpressJS

Pour rendre les paramètres optionnels, on utilise `?`

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

app.get('/hello/:nom?/:prenom?', (req, res) => {
    res.send(`Bonjour ${req.params['prenom']} ${req.params['nom']}`);
});

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

ExpressJS

Pour rendre les paramètres optionnels, on utilise `?`

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

app.get('/hello/:nom?/:prenom?', (req, res) => {
    res.send(`Bonjour ${req.params['prenom']} ${req.params['nom']}`);
});

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


Tester la route `http://localhost:3000/hello/wick` et vérifier que `Bonjour undefined wick` s'affiche dans le navigateur.

ExpressJS

Pour indiquer le code de la réponse du serveur

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

app.get('/hello/:nom?/:prenom?', (req, res) => {
    res.send(`Bonjour ${req.params.prenom} ${req.params.nom}`);
});

app.all('/*', (req, res) => {
    res
        .status(404)
        .send('Not Found');
});

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

ExpressJS

Pour indiquer le code de la réponse du serveur

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

app.get('/hello/:nom?/:prenom?', (req, res) => {
    res.send(`Bonjour ${req.params.prenom} ${req.params.nom}`);
});

app.all('/*', (req, res) => {
    res
        .status(404)
        .send('Not Found');
});

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

En allant à `/personne`, le texte `Not Found` est affiché.

ExpressJS

Pour modifier l'entête de la réponse

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

app.get('/hello/:nom?/:prenom?', (req, res) => {
    res
        .set('Content-Type', 'text/plain')
        .send(`Bonjour ${req.params.prenom} ${req.params.nom}`);
});

app.all('/*', (req, res) => {
    res
        .status(404)
        .send('Not Found');
});

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

ExpressJS

Pour indiquer le `content-type`, on peut aussi utiliser `type`

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

app.get('/hello/:nom?/:prenom?', (req, res) => {
    res
        .type('html')
        .send(`Bonjour ${req.params.prenom} ${req.params.nom}`);
});

app.all('/*', (req, res) => {
    res
        .status(404)
        .send('Not Found');
});

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

ExpressJS

Pour retourner un objet JSON

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

app.get('/hello/:nom?/:prenom?', (req, res) => {
    res
        .type('json')
        .json({ nom: req.params.prenom, prenom: req.params.nom });
});

app.all('/*', (req, res) => {
    res
        .status(404)
        .send('Not Found');
});

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

Pour retourner une page HTML

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

app.get('/hello', (req, res) => {
    res
        .type('html')
        .sendFile(__dirname + '/index.html');
});

app.all('/*', (req, res) => {
    res
        .status(404)
        .send('Not Found');
});

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

Pour retourner une page HTML

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

app.get('/hello', (req, res) => {
    res
        .type('html')
        .sendFile(__dirname + '/index.html');
});

app.all('/*', (req, res) => {
    res
        .status(404)
        .send('Not Found');
});

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

En cas d'utilisation de moteur de template, utilisez `render` pour retourner la vue (voir chapitre suivant).