

Programmation Web – Avancé

JavaScript & Node.js

Partie 17 : MPA & Framework Express

Version 2020



Attribution –
Partage dans les
Mêmes Conditions
4.0 International
(CC BY-SA 4.0)

*Presentation template
by [SlidesCarnival](#)*



Création d'une Multi Page Application en utilisant Express

Accélérer le développement de son backend : le framework Express...

Express

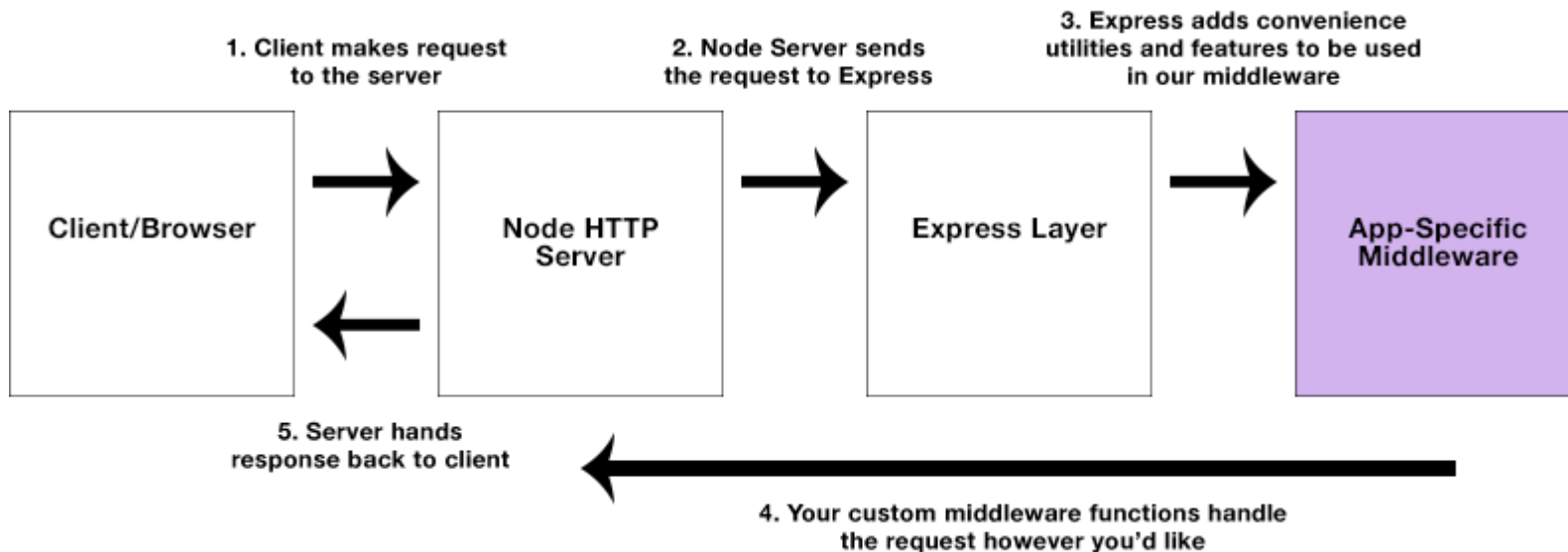
*“Fast, unopinionated, minimalist
web framework for Node.js”*

Express [\[57.\]](#)

“



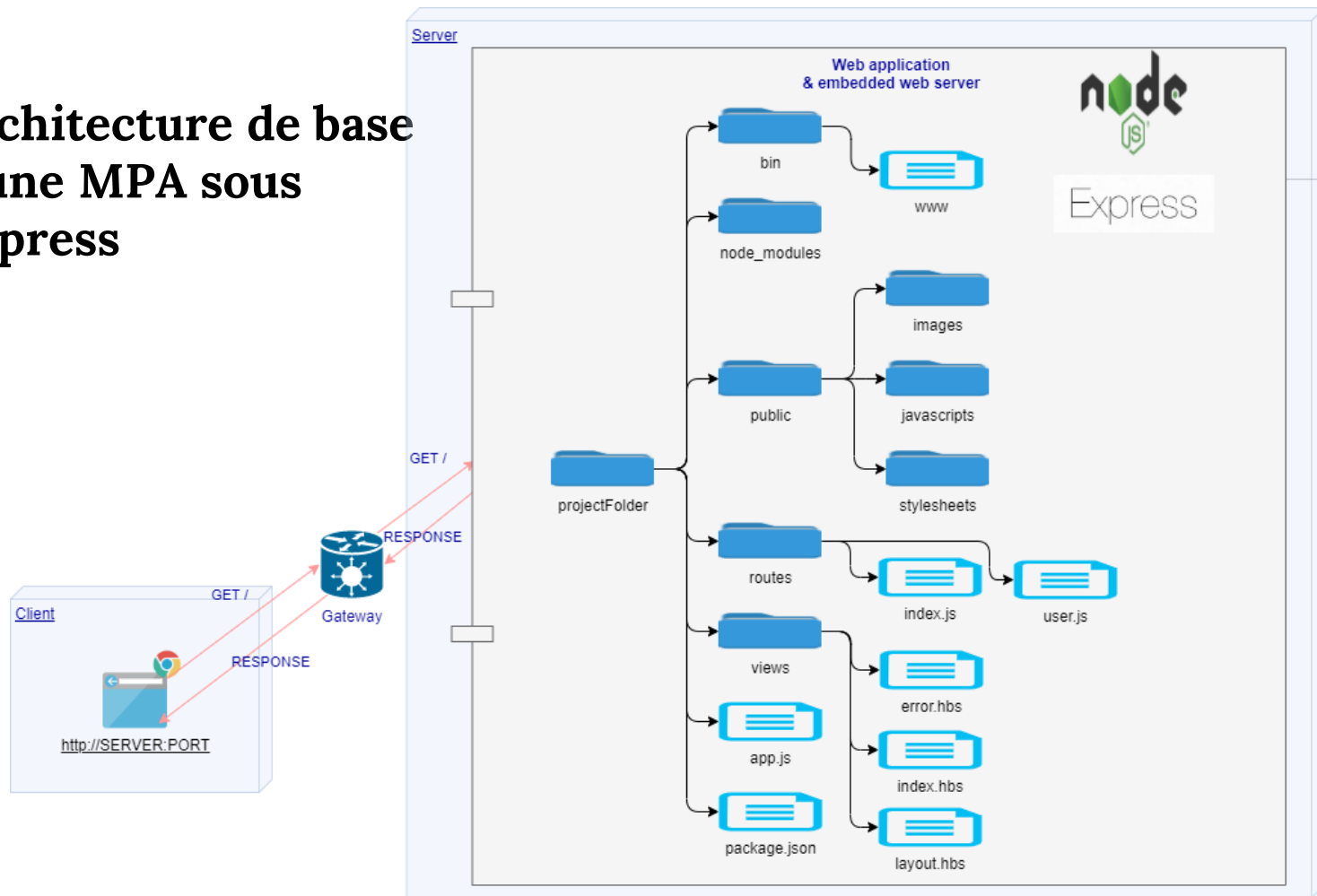
Introduction au framework Express



Introduction à Express [58.]



Architecture de base d'une MPA sous Express





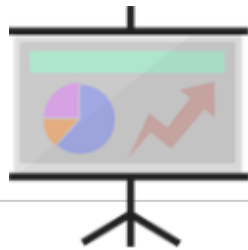
Créer une application Express

- Installation globale du générateur d'applications : `npm i express-generator -g`
- Créer une app : `express --view=hbs hello-world`
`express --view=pug hello-world`
- Installation des dépendances (données dans le fichier package.json) :
 - `cd hello-word`
 - `npm install`



Lancer une application (Express)

- Lancer l'application : **npm start**



Express

- **DEMO : Serveur web sous Express (hbs)**
Génération de l'application Hello World et intégration des fichiers partagés par le serveur web minimaliste fait sous Node.js
- NB: **DEMO : Serveur web sous Express (pug)**



Concepts principaux associés à une application Express

- Configuration et démarrage d'une application
- Serveur dynamique
- Middlewares
- Routing
- Vues et moteur de templating
- Serveur de fichiers statiques



Configuration et démarrage d'une application

```
{
  "name": "more-than-hello-world-hbs",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "dev": "nodemon ./bin/www",
    "start": "node ./bin/www"
  },
  "dependencies": {
    "express": "~4.16.1",
    "hbs": "~4.0.4",
    ...
  }
}
```

● Configuration

- **package.json**
- Détails [\[59.\]](#)

● Démarrage

- **npm start**
- **npm run dev**



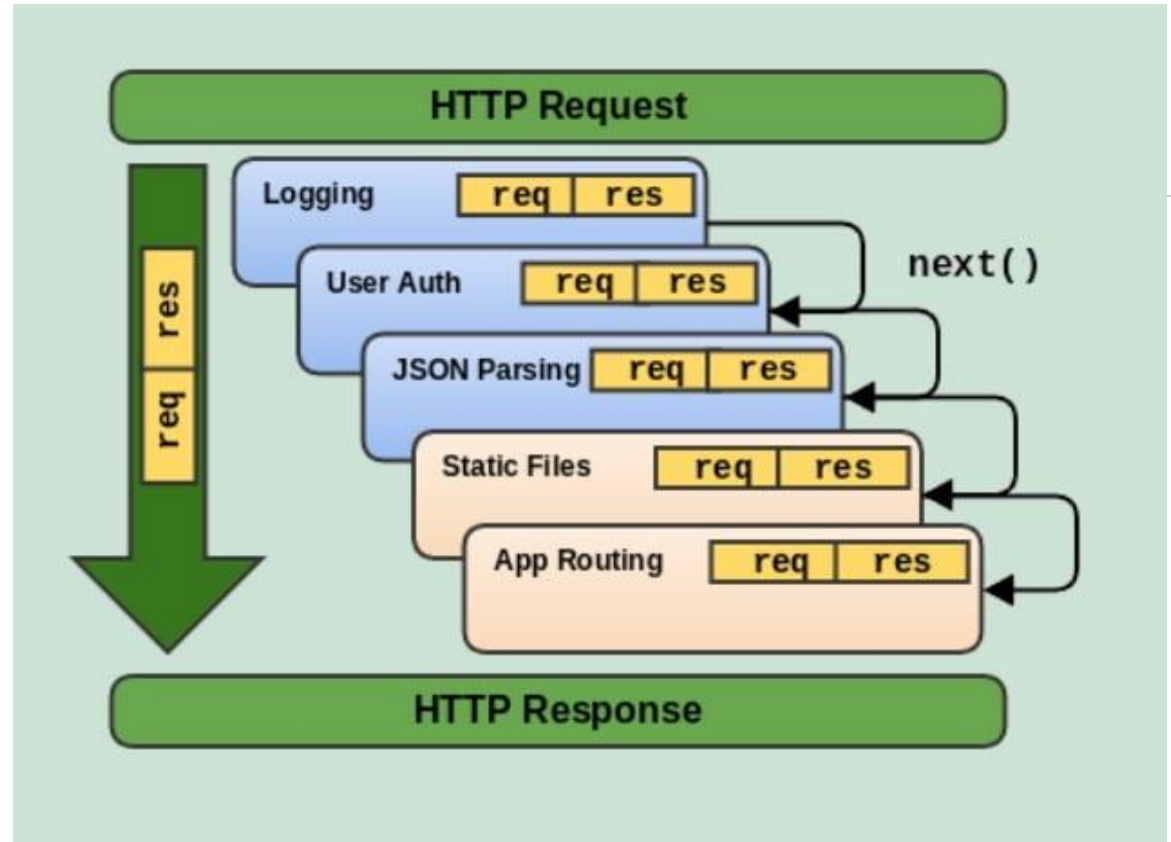
Serveur dynamique

◎ bin/www

```
var app = require('../app');  
var http = require('http');  
var port = normalizePort(process.env.PORT || '80');  
app.set('port', port);  
var server = http.createServer(app);  
server.listen(port);
```



Middlewares



Comprendre les Middlewares sous Express [60.]



Middlewares

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

HTTP method for which the middleware function applies.

```
app.get('/', function(req, res, next) {  
  next();  
})
```

Path (route) for which the middleware function applies.

The middleware function.

Callback argument to the middleware function, called "next" by convention.

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

HTTP **response** argument to the middleware function, called "res" by convention.

HTTP **request** argument to the middleware function, called "req" by convention.

Ecrire un middleware [61.]



Middleware

- Application-level middleware
- Router-level middleware
- Error-handling middleware
- Built-in middleware
- Third-party middleware
- Details [\[62.\]](#)



Application-level middleware

● Utilisation de middleware [\[62.\]](#)

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

app.use(function (req, res, next) {
  console.log("Time:", Date.now());
  next();
});
```



Router-level middleware

● Utilisation de middleware [\[62.\]](#)

```
var router = express.Router();  
// a middleware function with no mount path. This code is executed for every r  
equest to the router  
router.use(function (req, res, next) {  
  console.log("Time:", Date.now());  
  next();  
});  
// handler for the /user/:id path, which renders a special page  
router.get("/user/:id", function (req, res, next) {  
  console.log(req.params.id);  
  res.render("special");  
});
```




Error-handling middleware

● Utilisation de middleware [\[62.\]](#)

```
app.use(function (err, req, res, next) {  
  console.error(err.stack);  
  res.status(500).send("Something broke!");  
});
```

- 4 arguments au lieu de trois
- A définir après tous les middlewares pouvant générer une erreur via **next(err)**



Built-in middleware & third-party middleware

```
var createError = require("http-errors");
var express = require("express");
var path = require("path");
var cookieParser = require("cookie-parser");
var logger = require("morgan");
app.use(logger("dev")); // HTTP request logger
app.use(express.json()); // Parse requests with JSON payloads
app.use(express.urlencoded({ extended: false })); // Parse requests with URL-
                                                    encoded payload
app.use(cookieParser()); // Parse cookie header (req.cookies)
app.use(express.static(path.join(__dirname, "public"))); // Serve static assets
```



Routing

- Contrôle de la réponse à une requête client pour un endpoint/URI/PATH et une méthode HTTP
- Définition d'une route [\[63.\]](#) :
app.METHOD(PATH, HANDLER)

```
var express = require("express");
var app = express();
// respond with "hello world" when a GET request is made to the homepage
app.get("/", function (req, res) {
  res.send("hello world");
});
```



Routing

🕒 Définition d'un router (mini-app) :

```
var indexRouter = require('./routes/index');  
var usersRouter = require('./routes/users');  
app.use('/', indexRouter);  
app.use('/users', usersRouter);
```

/app.js

```
var express = require('express');  
var router = express.Router();  
/* GET /users/ */  
router.get('/', function(req, res, next) {  
  res.send('respond with a resource');  
});  
module.exports = router;
```

/routes/users.js

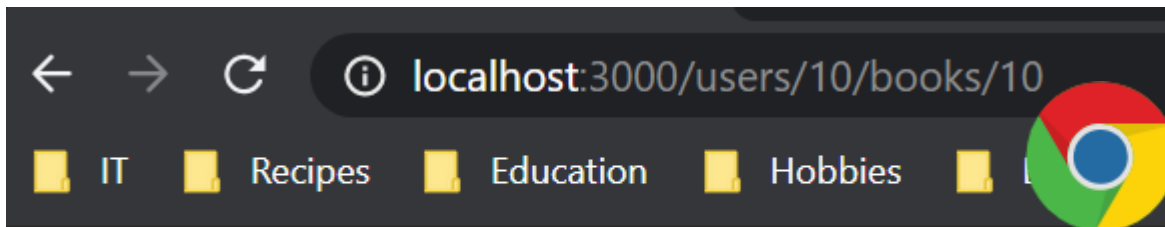


Routing

- Chemin et paramètres d'une route [\[63.\]](#) :
 - Paramètres : **req.params**

```
router.get("/:userId/books/:bookId", function (req, res) {  
  res.send(`Parameters in path: userID: ${req.params.userId}  
    bookId:${req.params.bookId}`);  
});
```

JS



Parameters in path: userID: 10 bookId:10



Routing

- Parser le body d'une requête :
 - Paramètres : **req.body** grâce à **express.urlencoded()**

```
<input class="form-control" id="email" type="text" name="email"
  {{!-- name is used in req.body as key --}}
  placeholder="Enter your email" required
  pattern="^\w+([.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,4})+$" />
```



```
/* POST new user */
router.post("/register", function (req, res, next) {
  req.app.locals.userList.push({email:req.body.email,password:req.body.password});
  console.log("POST /register:",req.app.locals.userList) ;
  res.redirect("/list");
});
```






Routing

- Méthodes associées aux réponses [\[63.\]](#) :
 - **res.render()** : render d'un template de view
 - **res.redirect()** : redirection d'une requête
 - **res.json()** : renvoi d'une réponse au format JSON
 - **res.send()** : renvoi d'une réponse (types variés)
 - **res.end()** : fin du processus de réponse
 - ...



res.redirect()



```
/* POST new user */
router.post("/register", function (req, res, next) {
  req.app.locals.userList.push({email:req.body.email,password:req.body.password});
  console.log("POST /register:",req.app.locals.userList)  ;
  res.redirect("/list");
});
```

- Redirection du client : nouvelle requête GET du client vers URL



Vues et moteur de templates

- Utilisation d'un template engine pour créer des view dynamiques à partir de fichiers templates
- Template engines : Handlebars, Mustache, Pug, Jade...
- Pour ce cours : Handlebars (Pug pour info)
- Créer une app avec le template engine voulu :
`express --view=hbs hello-world`



Renvoi d'une view à un client : res.render()

```
/* GET home page. */
router.get("/", function (req, res, next) {
  res.render("index", {
    headerTitle: "JavaScript & Node.js full course",
    pageTitle: "Demo : MPA with Express",
    footerText: "Happy learning : )",
  });
});
```

- Passage de paramètres à un view template



Handlebars comme moteur de templates

- **hbs** [\[64.\]](#) (ou **express-handlebars** [\[65.\]](#))
- **/views/layout.hbs** :
 - Master layout
 - **{{{body}}}** : lieu utilisé pour chaque view template
- **/views/index.hbs** : contenu de la view « index » rendue dans le **{{{body}}}** du master layout



Handlebars comme moteur de templates

● Built-in helper [\[66.\]](#)

- **#if or #unless** : Rendu conditionnel d'un bloc

```

{{#if isAuthenticated}}
<a class="nav-item nav-link" href="/">Home</a>
<a class="nav-item nav-link" href="/list">List users</a>
<a class="nav-item nav-link" href="/logout">Logout</a>
<a class="nav-item nav-link" href="/">{{user}}</a>
{{else}}
<a class="nav-item nav-link" href="/">Home</a>
<a class="nav-item nav-link" href="/register">Register</a>
<a class="nav-item nav-link" href="/login">Login</a>
{{/if}}

```



Handlebars comme moteur de templates

● Built-in helper [\[66.\]](#)

- **#each** : Boucle pour rendre des éléments à partir d'un object ou array

```
<ul class="list-group list-group-horizontal-lg">
  {{#each userList}}
    <li class="list-group-item">{{this.email}}</li>
  {{/each}}
</ul>
```



Handlebars comme moteur de templates

- Utilisation de handlebars côté serveur : Server Side Rendering
- Utilisation de handlebars côté client :
 - Pas adapté pour gestion d'événements, pour les communications frontend-backend...
- **handlebars** [\[67.\]](#)



Serveur de fichiers statiques

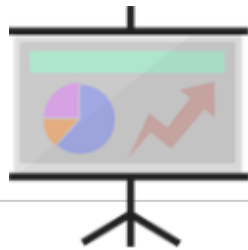
- Middleware **express.static** avant les routes

```
app.use(express.static(path.join(__dirname, "public"))); // Serve static assets
```

- Répertoire **public** servant les fichiers statiques
 - nécessaires à vos views : .js, .css, .jpg, .png...
 - appelés directement par le browser

```
<link rel="stylesheet" href="/stylesheets/style.css"> /views/layout.hbs
```

```
<script src="/javascripts/submit.js" type="module"></script> /views/user-forms.hbs
```



Express

🕒 DEMO : MPA avec un formulaire d'enregistrement d'utilisateurs



« Persistance temporaire » des données via un array

- Propriétés locales au sein de l'app :

```
app.locals.userList = [];
```

```
req.app.locals.userList.push({email:req.body.email,password:req.body.password});
```




Express

🕒 DEMO : MPA avec un formulaire d'enregistrement d'utilisateurs



Redémarrage automatique du serveur lors d'une modification de l'application : **nodemon**

■ **npm install -g nodemon**

```
"scripts": {  
  "dev": "nodemon ./bin/www",  
  "start": "node ./bin/www"  
},
```

/package.json

■ **npm run dev**