NodeJS : Express

- Introduction à Express
- MVC
- Installation et utilisation d’Express
- Routes simples
- Routes dynamiques
- Erreur 404
- Templates
- **Middlewares**

Middlewares

- Express permet à d'autres modules (les **middlewares**) de communiquer entre eux pour offrir plus de fonctionnalités
- Ils communiquent en se renvoyant **quatre paramètres** : les erreurs, la requête de l'internaute et la réponse à renvoyer, un callback vers le prochain middleware
- Il existe des **middlewares standards** pour Express et on peut en créer soi-même

Middlewares Express

- body-parser
- compression
- connect-rid
- cookie-parser
- cookie-session
- cors
- csurf
- error-handler
- method-override
- morgan
- multer
- response-time
- serve-favicon
- serve-index
- serve-static
- session
- timeout
- vhost

Fichiers statiques

- Pour **servir des fichiers statiques** tels que les images, les fichiers CSS et JavaScript, il faut utiliser le **middleware** intégré **express.static**
- On passe le nom du répertoire qui contient ces ressources statiques. Par exemple : **public**

```
const path = require('path');
app.use(express.static(path.join(__dirname,'public')));
```

où *__dirname est le dossier racine du projet*

- Ces instructions sont placées avant de définir les différentes routes

Middleware personnalisé

- On doit écrire une **fonction qui accepte 4 paramètres : err, req, resp, next**
  - **err** est un objet Error
  - **req** et **resp** sont la requête et la réponse
  - **next** est un callback pour communiquer avec le middleware suivant
- Exemple de base d'un middleware :

```
app.use(function(err,req,resp,next) {
  // ici votre code
  return next(); //ne pas oublier pour le suivant !
  });
```

Middleware (exemple - 1)

- Nouveau projet : npm init (création package.json)
- Ajouter Express : npm install express --save
- Ajouter Nodemon (mode développement) :

  npm install nodemon - -save-dev
- Ajouter dans package.json (section "scripts")

  "start": "nodemon index.js"
- Démarrer ce script (mode développement)

  npm run start

Middleware (exemple - 2)

- Hello World avec console.log

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

app.get('/', function(requete,reponse){
    console.log("Requete reçue...");
    reponse.send("Hello World : middlewares");
});

app.listen(8080, function(){
    console.log("Express : serveur en attente...");
});
```

Middleware (exemple - 3)

- Utilisation d'un middleware avec la méthode **app.use()**

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

// Définition du middleware
const middleware = function(req, resp, next) {
    console.log("Middleware : ", req.url);
    next();
};

// Utilisation du middleware
app.use(middleware);

app.get('/', function(requete,reponse){
    console.log("Requete reçue...");
    reponse.send("Hello World : middlewares");
});

app.listen(8080, function(){
    console.log("Express : serveur en attente...");
});
```

Middleware (exemple - 4)

- Attention à **l'ordre dans l'appel des méthodes**

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

// Définition du middleware
const middleware = function(req, resp, next) {
    console.log("Middleware : ", req.url);
    next();
};

app.get('/', function(requete,reponse){
    console.log("Requete reçue...");
    reponse.send("Hello World : middlewares");
});

// Utilisation du middleware
app.use(middleware);

app.listen(8080, function(){
    console.log("Express : serveur en attente...");
});
```

Middleware (exemple - 5)

- Attention de **ne pas oublier** le **next();**

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

// Définition du middleware
const middleware = function(req, resp, next) {
    console.log("Middleware : ", req.url);

};

// Utilisation du middleware
app.use(middleware);

app.get('/', function(requete,reponse){
    console.log("Requete reçue...");
    reponse.send("Hello World : middlewares");
});

app.listen(8080, function(){
    console.log("Express : serveur en attente...");
});
```

Middleware (exemple - 6)

- Avec plusieurs middlewares

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

// Définition du premier middleware
const middleware1 = function(req, resp, next) {
    console.log("Middleware 1 : ", req.url);
    next();
};

// Définition du second middleware
const middleware2 = function(req, resp, next) {
    console.log("Middleware 2 : ", req.url);
    next();
};

// Utilisation de middleware 1
app.use(middleware1);

app.get('/', function(requete,reponse){
    console.log("Requete reçue...");
    reponse.send("Hello World : middlewares");
});

// Utilisation de middleware 2
app.use(middleware2);

app.listen(8080, function(){
    console.log("Express : serveur en attente...");
});
```

Middleware (exemple - 7)

- Pour les appeler avant le callback...

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

const middleware1 = function(req, resp, next) {
    console.log("Middleware 1 : ", req.url);
    next();
};

const middleware2 = function(req, resp, next) {
    console.log("Middleware 2 : ", req.url);
    next();
};

// Utilisation des middlewares avant
app.use(middleware1,middleware2);

app.get('/', function(requete,reponse){
    console.log("Requete reçue...");
    reponse.send("Hello World : middlewares");
});

app.listen(8080, function(){
    console.log("Express : serveur en attente...");
});
```

Middleware (exemple - 8)

- Pour exécuter les middlewares quelque soit la requête (get, post,...), on utilise **app.use()**

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

const middleware1 = function(req, resp, next) {
    console.log("Middleware 1 : ", req.url);
    next();
};

const middleware2 = function(req, resp, next) {
    console.log("Middleware 2 : ", req.url);
    next();
};

app.use('/', function(requete,reponse,next){
    console.log("Requete reçue...");
    reponse.send("Hello World : middlewares");
    next();
}, middleware1, middleware2);

app.listen(8080, function(){
    console.log("Express : serveur en attente...");
});
```

Template et middleware

- On peut utiliser un middleware avec un template
- On ajoute le middleware standard **morgan** qui permet de **sauvegarder dans un journal (log)** les opérations sur le serveur
- On doit donc l'installer dans le projet
  npm install morgan --save
- Et on l'exécute en premier
- On ajoute aussi un **middleware personnalisé** qui **intercepte les messages d'erreur**

Middleware (autre exemple)

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

// Middleware standard : une ligne de log à chaque requête
const morgan = require('morgan');

app.use(morgan('[:date[clf] "UserAgent=:user-agent" ":method :url" Status=:status TempsReponse= :response-time ms'));

// Définition des routes
app.get('/', function(requete,reponse,next){
    reponse.render('accueil.ejs');
})
// Middleware personnalisé : uniquement pour ce chemin "/about"
.get('/about', function(requete,reponse,next){
    console.log("Utilisation du middleware sur la requête : " + requete.url);
    next();
    }, function(requete,reponse,next){
    reponse.render('apropos.ejs');
})
.get('/page/numero=:pagenum', function(requete,reponse,next){
    const monParametre = requete.params.pagenum;
    // exemple : on accepte uniquement des nombres de 1 à 5
    if (Number.isInteger(Number(monParametre)) && ((monParametre>0)&&(monParametre<=5))){
        reponse.render('mapage.ejs', {num: monParametre});
    } else {
        // parametre invalide : lever une erreur
        throw new Error('ERREUR : valeur passée dans page/numero non valide => ' + monParametre);
    }
})
.use(function(requete, reponse, next){
    // URL invalide : lever une erreur
    throw new Error('ERREUR : URL non prévue => ' + requete.url);
    next();
})
// Middleware personnalisé pour la gestion des erreurs
.use(function(erreur,requete,reponse,next){
    console.log("CF2M - " + erreur.message);
    reponse.status(404).render('erreur404.ejs', {msg: erreur.message});
});

app.listen(8080);   // port 8080
console.log("Express démarré");
```