NodeJS : Web Application

- **Express Generator**
- Arborescence de l'application
- Démarrage de l'application
- Affichage de l'application
- Outil de développement
- Explication des fichiers générés (contenu, rôle, lien,...)

Express Generator

- **Express** = framework permettant de construire des applications web
- **Express Generator** = squelette d'application
- Installation à réaliser avec **Command Prompt** en mode **administrateur** (erreurs avec PowerShell). A faire une seule fois de manière globale

  npm install express-generator -g
- Vérification de l'installation (aide et options)

  express -h

Express Generator (options)

```
PS E:\nodeProjects> cd demo_generator
PS E:\nodeProjects\demo_generator> npm install express-generator -g
npm WARN deprecated mkdirp@0.5.1: Legacy versions of mkdirp are no longer supported. Please update to mkdirp 1.x. (Note that the API surface ha
s changed to use Promises in 1.x.)
+ express-generator@4.16.1
added 10 packages from 13 contributors in 6.517s
PS E:\nodeProjects\demo_generator> express -h

  Usage: express [options] [dir]

  Options:

        --version        output the version number
    -e, --ejs            add ejs engine support
        --pug            add pug engine support
        --hbs            add handlebars engine support
    -H, --hogan          add hogan.js engine support
    -v, --view <engine>  add view <engine> support (dust|ejs|hbs|hjs|jade|pug|twig|vash) (defaults to jade)
        --no-view        use static html instead of view engine
    -c, --css <engine>   add stylesheet <engine> support (less|stylus|compass|sass) (defaults to plain css)
        --git            add .gitignore
    -f, --force          force on non-empty directory
    -h, --help           output usage information
PS E:\nodeProjects\demo_generator>
```

- Choix du **moteur de template** et du **moteur CSS** par défaut, Jade (Pug) et CSS standard

Arborescence de l'application

- **public** : contient les fichiers statiques
- **routes** : définition des routes (URL)
- **views** : définition des pages HTML (vues)
- **bin/www** : fichier www de démarrage
- **app.js** : configuration

express --view=ejs maWebApp

```
PS E:\nodeProjects\demo_generator> express --view=ejs maWebApp

   create : maWebApp\
   create : maWebApp\public\
   create : maWebApp\public\javascripts\
   create : maWebApp\public\images\
   create : maWebApp\public\stylesheets\
   create : maWebApp\public\stylesheets\style.css
   create : maWebApp\routes\
   create : maWebApp\routes\index.js
   create : maWebApp\routes\users.js
   create : maWebApp\views\
   create : maWebApp\views\error.ejs
   create : maWebApp\views\index.ejs
   create : maWebApp\app.js
   create : maWebApp\package.json
   create : maWebApp\bin\
   create : maWebApp\bin\www

   change directory:
     > cd maWebApp

   install dependencies:
     > npm install

   run the app:
     > SET DEBUG=mawebapp:* & npm start

PS E:\nodeProjects\demo_generator>
```

Démarrage de l'application

- Changer de répertoire **cd maWebApp**
- Installer les dépendances (liste dans *package.json*) **npm install**
- Mettre l'application en mode « debug » **SET DEBUG=maWebApp:***

  et démarrer avec **npm start**

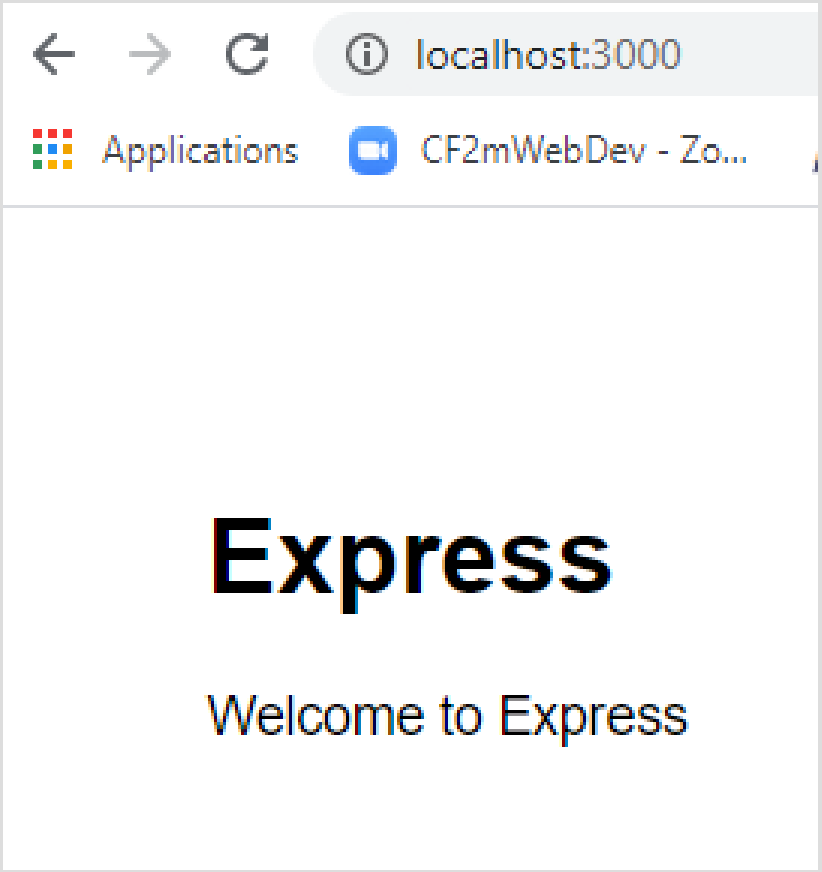
```
PS E:\nodeProjects\demo_generator> cd maWebApp
PS E:\nodeProjects\demo_generator\maWebApp> npm install
npm notice created a lockfile as package-lock.json. You should commit this file.
added 54 packages from 38 contributors and audited 55 packages in 17.231s
found 0 vulnerabilities

PS E:\nodeProjects\demo_generator\maWebApp> SET DEBUG=mawebapp:* & npm start
Au caractère Ligne:1 : 22
+ SET DEBUG=mawebapp:* & npm start
+                      ~
Le caractère perluète n'est pas autorisé. L'opérateur & est réservé à une utilisation future. Placez un caractère perluète entre guillemets
doubles ("&") pour que ce symbole soit considéré comme une chaîne.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : AmpersandNotAllowed

PS E:\nodeProjects\demo_generator\maWebApp> SET DEBUG=mawebapp:*
PS E:\nodeProjects\demo_generator\maWebApp> npm start

> mawebapp@0.0.0 start E:\nodeProjects\demo_generator\maWebApp
> node ./bin/www
```

Affichage de l'application



- Ouvrir dans un navigateur l'URL : http://localhost:3000/

  (le port 3000 est configuré dans le fichier ./bin/www)
- **La page d'accueil** est affichée

Développer avec nodemon

- Aucun changement dans le code ne sera visible tant que le serveur n'est pas redémarré
- Outil **nodemon** redémarre automatiquement

  npm install --save-dev nodemon

- Dans *package.json*, ajouter dans "scripts"

  "dev" : "nodemon ./bin/www"

- Redémarrer l'application avec

  npm run dev

NB : Dans des versions récentes, nodemon est déjà installé

package.json

```
{
  "name": "test",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www",
    "dev": "nodemon ./bin/www"
  },
  "dependencies": {
    "cookie-parser": "~1.4.3",
    "debug": "~2.6.9",
    "ejs": "^3.1.9",
    "express": "^4.18.2",
    "http-errors": "~1.6.2",
    "morgan": "~1.9.0"
  },
  "devDependencies": {
    "nodemon": "^3.0.1"
  }
}
```

NB : les numéros de version peuvent varier

- On trouve le 2ème script de démarrage ajouté (**devstart**)
- On a les **dépendances** (définies par défaut par express-generator)
- On a **nodemon** (ajouté pour développer)
- On a le fichier **./bin/www** au démarrage

./bin/www

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

var app = require('../app');
var debug = require('debug')('mawebapp:server');
var http = require('http');

/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);

/**
 * Listen on provided port, on all network interfaces.
 */

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);
```

- C'est le **point d'entrée** de l'application (app.js)

  var app=require('../app');
- Configuration du **port d'écoute (3000)**
- Création du **serveur** (createServer) et **démarrage** (listen)
- Voir le reste du code...

Configuration : app.js (1)

demo_generator > maWebApp > JS app.js > ...

```
1   var createError = require('http-errors');
2   var express = require('express');
3   var path = require('path');
4   var cookieParser = require('cookie-parser');
5   var logger = require('morgan');
6
7   var indexRouter = require('./routes/index');
8   var usersRouter = require('./routes/users');
9
10  var app = express();
11
12  // view engine setup
13  app.set('views', path.join(__dirname, 'views'));
14  app.set('view engine', 'ejs');
15
16  app.use(logger('dev'));
17  app.use(express.json());
18  app.use(express.urlencoded({ extended: false }));
19  app.use(cookieParser());
20  app.use(express.static(path.join(__dirname, 'public')));
21
```

- Ajout des **dépendances** (lignes 1-5)
- Gestion des **routes** (lignes 7-8)
- **Création** de l'application (ligne 10 + ligne 2)
- **Moteur de template** (lignes 12-14)
- **Middlewares** (lignes 16-20)

Configuration : app.js (2)

```
21
22  app.use('/', indexRouter);
23  app.use('/users', usersRouter);
24
25  // catch 404 and forward to error handler
26  app.use(function(req, res, next) {
27    next(createError(404));
28  });
29
30  // error handler
31  app.use(function(err, req, res, next) {
32    // set locals, only providing error in development
33    res.locals.message = err.message;
34    res.locals.error = req.app.get('env') === 'development' ? err : {};
35
36    // render the error page
37    res.status(err.status || 500);
38    res.render('error');
39  });
40
41  module.exports = app;
42
```

- Définition des **routes** :

  - URL définies (lignes 22-23)
  - URL inconnue (lignes 25-28)

- Gestion des **erreurs** (lignes 30-39)
- **Export** du module pour utilisation dans bin/www (ligne 41)

/routes/index.js

index.js

demo_generator > maWebApp > routes > index.js > ...

```
1  var express = require('express');
2  var router = express.Router();
3
4  /* GET home page. */
5  router.get('/', function(req, res, next) {
6    res.render('index', { title: 'Express' });
7  });
8
9  module.exports = router;
```

- **Import** du module **Express** et de l'objet **Router** (lignes 1-2)
- **Analyse de l'URL** à récupérer (requête GET) (ligne 5)
- **Envoi de la réponse** pour affichage (ligne 6)

/views/index.ejs

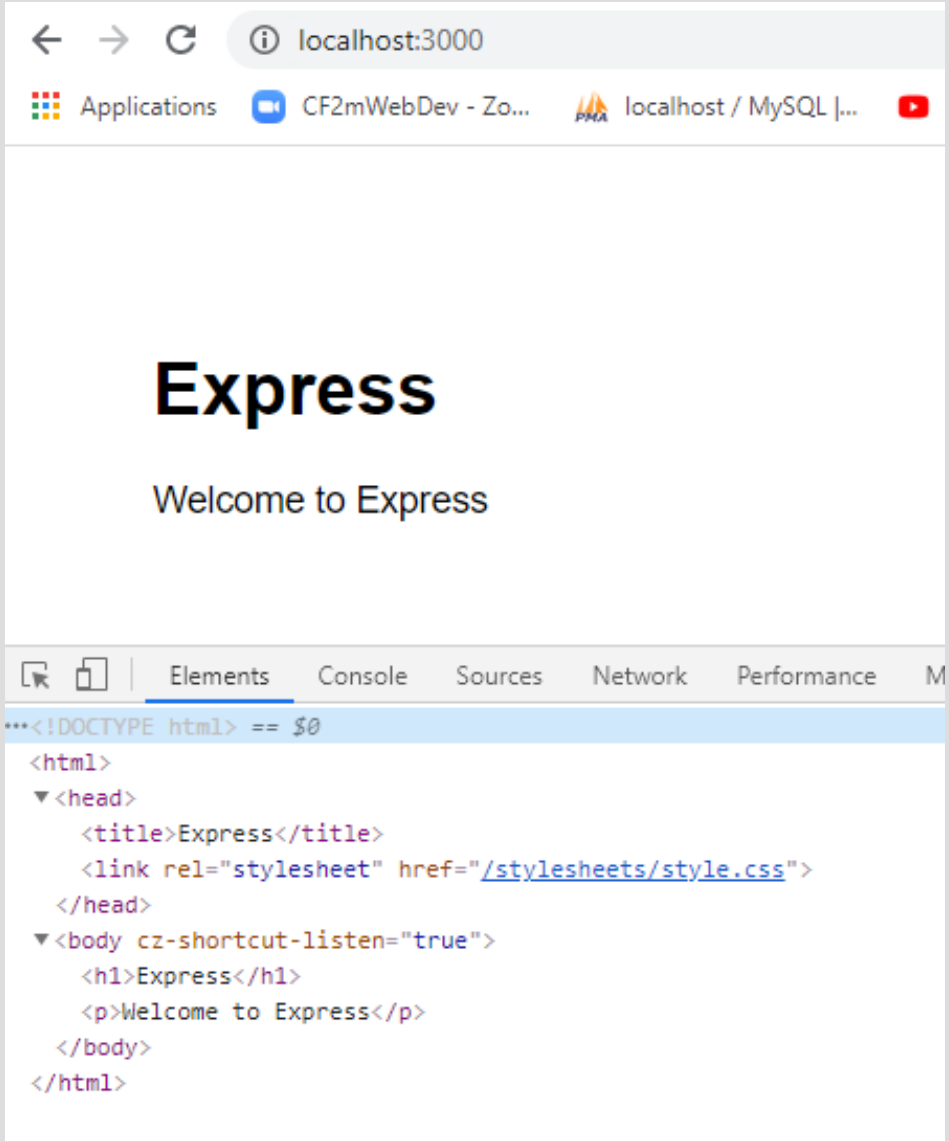
index.ejs

demo_generator > maWebApp > views > index.ejs > ...

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <title><%= title %></title>
5       <link rel='stylesheet' href='/stylesheets/style.css' />
6     </head>
7     <body>
8       <h1><%= title %></h1>
9       <p>Welcome to <%= title %></p>
10    </body>
11  </html>
```

- Utilisation du template **index.ejs** appelé par *res.render('index',...);*
- **title** est remplacé par sa valeur définie dans *res.render(...,{title:'...'});*

Page d'accueil affichée



Utilisation avancée

- Si on veut modifier le code généré par le module express-generator, notamment le package.json, il faut le modifier manuellement.
- Comme on l'a installé globalement, il est dans un répertoire du disque C:/ où est installé node.js
  Par exemple :
  ..\AppData\Roaming\npm\node_modules\express-generator\bin
- On corrige le fichier **express-cli.js** avec les nouvelles versions de modules à indiquer dans package.json