• Démarche de généralisation

- Lorsque nous développons une application Node.js avec Express nous avons a charger par un “**require**” le module contrôleur dans app.js, puis nous avons aussi à définir la route dans un « **app.use(‘/xxxx’, nomContrôleur)** » pour chaque action, nous chargeons donc en RAM tous les contrôleurs au lancement de l’application.

- Nous allons donc factoriser le chargement des contrôleurs et et externaliser leur définition dans un fichier JSON que l’on pourrait comparer à un annuaire d’actions (pathname).

- Pour ce faire nous devons réfléchir comment définir la structure de cet annuaire. Nous devons gérer l’action (pathname) comme clé mais également la méthode HTTP (GET, POST) afin de pouvoir gérer les 2 cas.

- **Exemples d'un chargeur de routes :**
  - Ecrire un chargeur de contrôleurs génériques permettant de définir les routes (actions) et les contrôleurs dans une description JSON :

```
{
    "GET/" : {"controler": "index"},
    "GET/users": {"controler": "users"},
    "GET/exos": {"controler": "exos"},
    "GET/countries": {"controler": "countries"},
    "GET/formUser": {"controler": "formUser"},
    "GET/modifyUser": {"controler": "modifyUser"},
    "GET/newUser": {"controler": "newUser"},
    "GET/createUser": {"controler": "createUser"}
}
```

  - Créez ce fichier "**config_actions.json**" dans le dossier "**/routes**" de notre application.

- **Exercice (suite) :**
  - Afin de garder notre précédente application, nous allons copier “**app.js**” dans “**appdyn.js**”.
  - Voici les lignes de code à supprimer dans le nouveau fichier appdyn.js :

```
// Lignes 9 à 20
var index = require('./routes/index');
var users = require('./routes/users');
var exos = require('./routes/exos');
var countries = require('./routes/countries');

var users = require('./routes/users');

var formUser = require('./routes/formUser');
var modifyUser = require('./routes/modifyUser');

var newUser = require('./routes/newUser');
var createUser = require('./routes/createUser');
```

```
// ligne 72 à 83
app.use('/', index);'
app.use('/users', users);
app.use('/exos', exos);
app.use('/countries', countries);
app.use('/users', users);
app.use('/formUser', formUser);
app.use('/formUser/:_id', formUser);
app.use('/modifyUser', modifyUser);
app.use('/modifyUser/:_id', modifyUser);
app.use('/newUser', newUser);
app.use('/createUser', createUser);
```

  - Nous allons ajouter la lecture de notre configuration JSON des routes :

- **Workshop router dynamique :**
  - Nous allons ajouter la lecture de notre configuration JSON des routes à la place de la liste des requires des contrôleurs :

```
/* chargement configuration JSON des actions --> controleurs */
global.actions_json = JSON.parse(fs.readFileSync("./routes/config_actions.json", 'utf8'));
```

  - "**actions_json**" est un tableau associatif des actions comme clé et des contrôleurs comme valeurs.
  - Nous allons devoir créer un **routeur Dynamique** qui en fonction de la configuration dans le fichier JSON associe le contrôleur à l'action (le pathname). Appelons le "**dynamicRouter.js**" et placez le dans le même dossier que **app.js** et **appdyn.js**

```
// Gestion des routes dynamiques via configuration json
require('./dynamicRouter')(app);
```

- **Routeur dynamique (suite) :**
  - Comment écrire ce **“dynamicRouter.js”** ?

```
var express = require("express");
var router = express.Router();
var appContext;
var url = require("url");

function dynamicRouter(app) {
    //-- Context applicatif
    appContext = app;
    // -- Perform Automate action
    router.use(manageAction);
    // -- routeur global
    appContext.use(router);
}
(…) // → suite colonne de droite
```

A T T E N T I O N ! ! !
NB : Pensez à changer dans le fichier /bin/www le nom du fichier app en appdyn

var app = require('../appdyn');

```
/* Fonction qui permet d'aguiller les requêtes HTTP
vers le bon contrôleur en fonction de l'action du pathname  */
function manageAction(req, res, next) {
    var path;  // Le pathname après le port 3000 dans l'URL.
    var type;  //(GET ou POST, ... http méthode)
    var controler;  // nom du contrôleur à charger
    path = url.parse(req.url).pathname;
    // Il faut supprimer pour le routage le param après l'action
    if (path.split('/').length > 0) path = '/'+path.split('/')[1]
    // On récupère la méthode HTTP : GET ou POST
    type = req.method;
    // [type + path] permet de retrouver le bon contrôleur
    if (typeof GLOBAL.actions_json[type + path] == 'undefined') {
        console.log("Erreur pas d'action : " + path);
        next();
    }
    else {
       instanceModule = require('./routes/'
                + GLOBAL.actions_json[type + path].controler);
       router.use(path, instanceModule);
       next();
    }
}

module.exports = dynamicRouter;
```

Node, Express, MongoDB

- **Routeur dynamique (suite) :**
  - Comment gérer le paramètre “:_id” dans un contrôleur

```
var express = require('express');
var router = express.Router();
var ObjectID = require('mongodb').ObjectID;

/* Delete one user into database. */
router.route('/:_id').get(function (req, res) {
    // ici on a un élément en plus _id dans l'URL "/deleteUser/5b0d5e95488eaf5161489a1e"
    // on découpe l'url et on ne récupère que le premier élément du tableau "deleteUser"
    // on ajoute devant un "/"
    var path = '/' + req.originalUrl.split('/')[1];  // retourne ‘/deleteUser’ dans path
    var type = req.method;
    console.log('req.originalUrl : ', req.originalUrl);
    GLOBAL.db.collection('users').remove({_id: new ObjectID(req.params._id)},
        function (err, result) {
            if (err) {
                throw err;
            }
            console.log('createUser: ', result);
            GLOBAL.db.collection("users").find().toArray(function(err, listusers) {
                res.render(global.actions_json[type+path].view, {
                    title: 'List of users :',
                    users: listusers
                });
            });
        } // fin callback du Delete
    ); // fin de l'insert()
}); // fin de la gestion de la route
module.exports = router;
```

- **Conclusion Routeur Dynamique**
  - Nous n’avons volontairement géré que le nom du contrôleur dans notre fichier de configuration JSON. Pour simplifier l’exemple.
  - Mais nous pourrions ajouter des informations, comme la vue, nous pourrions ajouter le type MIME de retour (html, json, ...), bref, des paramètres supplémentaires qui peuvent être nécessaires.
  - Maintenant dans notre application Express, MongoDB, Handlebars il n’est plus nécessaire d’ajouter du code dans appdyn.js pour ajouter une action !! youpi ;-)
  - Il suffit de compléter et d’ajouter dans le fichier **config_actions.json** une action et un contôleur ...