NodeJS : lien avec MySQL

- Express Generator
- Liste des contacts avec une base de données
- **Comment faire le lien avec une application Express ?**
- Nouvelles routes et nouvelles vues
- Utilisation du modèle
- Fonctionnalités avancées avec EJS
- Amélioration de l’affichage des pages

Modification du routage

- Toutes les routes pour la gestion des messages sont regroupées dans un nouveau **Router** et les URL commencent par **/messages**
- On modifie donc le fichier **app.js**

```
var indexRouter = require('./routes/index');
//var usersRouter = require('./routes/users');  // pas besoin de ce router car pas de gestion des utilisateurs
var messagesRouter = require('./routes/messages');  // ce router gère les opérations sur les messages

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', indexRouter);
//app.use('/users', usersRouter);     // pas besoin de ce router car pas de gestion des utilisateurs
app.use('/messages', messagesRouter);   // ce router regroupe les URL pointant vers des méthodes de gestion des messages
```

Routeur pour les messages

- On définit les nouvelles routes dans un nouveau fichier **messages.js** placé dans le dossier **routes** (ici, avec un simple texte pour tester)

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

// Définition des routes pour gérer les messages
// Toutes les routes définies dans ce router commencent par l'URL : localhost:8080/messages

// Récupération de tous les messages : ReadAll
// URL => localhost:8080/messages/
router.get('/', function(req,res){
    console.log("GET Tous les messages");
    res.send('Lire tous les messages');
});

// Création d'un message : Create
// URL => localhost:8080/messages/create
router.post('/create', function(req,res){
    console.log("POST Créer un message");
});

module.exports = router;
```

Nouvelles vues

- **Vue** pour afficher la **liste des messages**
- **Sous-vues** réutilisables (entête et pied de page)
- Avec **EJS**, on a la syntaxe

  <%- include('nom_de_la_vue') -%>
- On peut créer des vues supplémentaires :
  - **header.ejs**
  - **footer.ejs**

Header et Footer

```
<!DOCTYPE html>
<html lang="fr">
    <head>
        <meta charset="utf-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title><%= titre %></title>
        <link rel='stylesheet' href='/stylesheets/style.css' />
    </head>
    <body>
```

```
    </body>
</html>
```

Envoi des données

- **Modifier** le formulaire pour envoyer les données sur la nouvelle route, servant à **créer un enregistrement** dans la base de données

```
<%- include('header') -%>
    <header>
        <h1><%=titre%></h1>
    </header>
    <section>
        <form id="contact" name="contact" method="POST" action="/messages/create">
<!--            <form id="contact" name="contact" method="POST" action="/traitement"> -->
            <label for="nom">Nom : </label>
            <input id="nom" name="nom" type="text" placeholder="Entrez votre nom" />
            <br />
            <label for="msg">Votre message : </label>
            <input id="msg" name="msg" type="text" placeholder="Entrez votre message" />
            <br />
            <input type="submit" value="Envoyer" />
        </form>
    </section>
<%- include('footer') -%>
```

Vérifier les données reçues

- **Récupérer les données** afin de vérifier que le formulaire a été rempli, sinon, retourner sur la page avec le formulaire

```
// Création d'un message : Create
// URL => localhost:8080/messages/create
router.post('/create', function(req,res){
    console.log("POST Créer un message");
    const titrePage = "Formulaire reçu";
    const lenom = req.body.nom;
    const lemessage = req.body.msg;

    // Valider le contenu de la requête (est-ce-que le formulaire a été rempli ?)
    if ((!req.body)||(lenom=="")||(lemessage=="")) {
        console.log("Le contenu ne peut pas être vide !");
        res.redirect('/contactform');   // retourner au formulaire
    } else {
```

Ajouter le modèle

- Pour établir le **lien entre l'application et la base de données**, il ne faut pas oublier d'**ajouter le modèle** qui décrit comment manipuler les données d'un message

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

// on importe l'objet Message et la connection à la DB au travers du modèle
const Message = require("../models/message.model");
```

Ecrire les données validées

- Si les **données sont valides**, alors on peut créer un objet Message, le stocker et ensuite afficher la page qui confirme le traitement

```
} else {
    console.log(req.body);
    // Créer mon message avec le modèle
    const unMsg = new Message({
        nom: req.body.nom,
        msg: req.body.msg
    });

    Message.create(unMsg, function(err, data){
        if (err) {
            res.status(500).send({
                message: "Erreur pendant la création du message"
            });
        } else {
            console.log('Data = ', data);
            res.render('traitementform', {titre: titrePage, nom: unMsg.nom, msg: unMsg.msg });
        }
    });
}
```

Lire les données

- **Récupérer tous les enregistrements** dans la base de données pour les envoyer sur la nouvelle vue affichant la liste des messages

```
// Récupération de tous les messages : ReadAll
// URL => localhost:8080/messages/
router.get('/', function(req,res){
    console.log("GET Tous les messages");
    Message.readAll(function(err,data){
        if (err) {
            res.status(500).send({
                message: "Erreur pendant la lecture de tous les messages"
            });
        } else {
            console.log('Data = ', data);
            const titrePage = "Liste des messages";
            res.render('listeMessages', {titre: titrePage, donnees: data });
        }
    });
});
```

Vue : liste des messages

- On crée une vue **listeMessages.ejs** et on utilise la syntaxe EJS pour tester une **condition,** c-à-d s'il y a des données à afficher ou non :

<% if (condition) { %> … <% } %>

```
<%- include('header') -%>
    <header>
        <h1><%=titre%></h1>
    </header>
    <section id="messages">
        <!-- Si je reçois des données, j'affiche le tableau -->
        <% if (donnees.length) { %>
        <table>

        </table>
        <% } %>
        <!-- S'il n'y a pas de données, j'affiche le message suivant -->
        <% if (!donnees.length) { %>
        <h3 class="au-centre">Pas de données actuellement !</h3>
        <% } %>
    </section>
<%- include('footer') -%>
```

Construire un tableau avec EJS

- On utilise la syntaxe EJS pour créer une **boucle** qui affichera tous les enregistrements qu'il y a dans la base de données :

<% for(var i=0; i<donnees.length; i++) { %> … <% } %>

```
<table>
    <thead>
        <tr>
            <th width="5%">ID</th>
            <th width="20%">Nom</th>
            <th width="50%">Message</th>
            <th width="25%">Date</th>
        </tr>
    </thead>
    <tbody>
    <% for(var i=0; i<donnees.length; i++) { %>
        <tr>
            <td><%= donnees[i].id %></td>
            <td><%= donnees[i].nom %></td>
            <td><%= donnees[i].message %></td>
            <td><%= donnees[i].datemessage %></td>
        </tr>
    <% } %>
    </tbody>
</table>
```

Améliorer l'affichage des dates

- Utilisation du module **moment.js** (à installer avec npm) pour **formater les dates/heures**

```
// import du module "moment.js" pour l'affichage des dates/heures
const moment = require('moment');

// Définition des routes pour gérer les messages
// Toutes les routes définies dans ce router commencent par l'URL : localhost:8080/messages

// Récupération de tous les messages : ReadAll
// URL => localhost:8080/messages/
router.get('/', function(req,res){
    console.log("GET Tous les messages");
    Message.readAll(function(err,data){
        if (err) {
            res.status(500).send({
                message: "Erreur pendant la lecture de tous les messages"
            });
        } else {
            console.log('Data = ', data);
            const titrePage = "Liste des messages";
            moment.locale('fr');
            res.render('listeMessages', {titre: titrePage, donnees: data, moment: moment });
        }
    });
});
```

Formatage des dates

- **Définir le formatage** des dates/heures avec moment.js pour l'affichage dans la vue

```
<% for(var i=0; i<donnees.length; i++) { %>
    <tr>
        <td><%= donnees[i].id %></td>
        <td><%= donnees[i].nom %></td>
        <td><%= donnees[i].message %></td>
        <td><%= moment(donnees[i].datemessage).format('DD-MMM-YYYY HH:mm:ss') %></td>
    </tr>
<% } %>
```

Accès à la liste

- Il reste à **ajouter un lien** dans une vue (par exemple, traiter_form) pour se diriger vers la liste des messages avec la route ***/messages***