Université d'Aix-Marseille - Polytech Marseille (Informatique)

JIN81H – TD et TP Applications Web et Mobiles

2018/2019

Réalisez les tâches suivantes sur la console Terminal

1. Créez un répertoire nommé `aFaire`
2. Se déplacer dans le répertoire `aFaire`
3. Taper ces commandes pour voir si *node.js*, *Node Package Manager (npm)*, et *mongodb* sont installés :
   a. `node -v`
   b. `npm -v`
   c. `mongo --version`

   Sinon, allez sur les sites officiels pour les installer.
4. Créez un fichier `server.js`,
5. Créez un répertoire `public` et deux fichiers (`index.html` et `index.js`) dans le répertoire `public`.
6. Tapez `npm init` et remplissez les questions.
7. Installez les paquets suivants et expliquez ses fonctions :
   a. `npm install express --save`
   b. `npm install mongoose --save`
   c. `npm install morgan --save`
8. Tapez ces codes dans `server.js` et donnez une commentaire pour chaque ligne :

```
var express  = require('express');
var app      = express();
var morgan = require('morgan');
var bodyParser = require('body-parser');

app.use(express.static(__dirname + '/public'));
app.use(morgan('dev'));
app.use(bodyParser.urlencoded({'extended':'true'}));
app.use(bodyParser.json());
app.use(bodyParser.json({ type: 'application/vnd.api+json' }));


app.listen(8080);
console.log("on utilise le port: 8080");
```

9. Tapez `node server.js` . Quel est le résultat?

10. Nous allons maintenant ajouter de nouvelles lignes pour se connecter à MongoDB. Tapez les nouveaux codes (voir police en gras) et donnez quelques commentaires.

```
...
var app      = express();
var mongoose = require('mongoose');
var morgan = require('morgan');
...
app.use(bodyParser.json({ type: 'application/vnd.api+json' }));

mongoose.connect('mongodb://localhost/ListeaFaire', {
useNewUrlParser: true });

var Liste = mongoose.model('Liste', {
    text : String
});

app.listen(8080);
...
```

11. On va gerer la route de serveur. Tapez les nouveaux codes (voir police en gras) et donnez quelques commentaires.

```
...
var Liste = mongoose.model('Liste', {
    text : String
});

app.get('/', function(req, res) {
    res.sendFile('./public/index.html');
});


app.get('/api/laliste', function(req, res) {
    Liste.find(function(err, laliste) {
        if (err)
            res.send(err)
        res.json(laliste);
    });
});

app.post('/api/laliste', function(req, res) {
    Liste.create({
        text : req.body.text,
        done : false
    }, function(err, liste) {
        if (err)
            res.send(err);
        Liste.find(function(err, laliste) {
            if (err)
                res.send(err)
            res.json(laliste);
        });
    });
});
```

```
app.delete('/api/laliste/:liste_id', function(req, res) {
    Liste.deleteOne({
        _id : req.params.liste_id
    }, function(err, liste) {
        if (err)
            res.send(err);
        Liste.find(function(err, laliste) {
            if (err)
                res.send(err)
            res.json(laliste);
        });
    });
});

app.listen(8080);
...
```

12. Tapez ces codes dans `index.js` et donnez une brève description pour chaque ligne :

```
var ListeaFaire = angular.module('ListeaFaire', []);

function mainController($scope, $http) {
    $scope.formData = {};
    $http.get('/api/laliste')
        .success(function(data) {
            $scope.laliste = data;
            console.log(data);
        })
        .error(function(data) {
            console.log('Error: ' + data);
        });
    $scope.createTodo = function() {
        $http.post('/api/laliste', $scope.formData)
            .success(function(data) {
                $scope.formData = {};
                $scope.laliste = data;
                console.log(data);
            })
            .error(function(data) {
                console.log('Error: ' + data);
            });
    };
    $scope.deleteTodo = function(id) {
        $http.delete('/api/laliste/' + id)
            .success(function(data) {
                $scope.laliste = data;
                console.log(data);
            })
            .error(function(data) {
                console.log('Error: ' + data);
            });
    };
}
```

13. Tapez ces codes dans `index.html` et donnez une brève description pour chaque ligne :

```
<!doctype html>
<html ng-app="ListeaFaire">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <title>Liste Choses à Faire</title>
    <script
src="//ajax.googleapis.com/ajax/libs/angularjs/1.0.8/angular.min.js
"></script>
    <script src="index.js"></script>
</head>

<body ng-controller="mainController">
    <h1>La liste des choses à faire</h1>
    <br/> Nombre des choses: {{ laliste.length }}
    <div id="todo-list">
        <div ng-repeat="x in laliste">
            <h5><button ng-click="deleteTodo(x._id)"> x </button>
{{ x.text }}</h5>
        </div>
    </div>

    <div id="todo-form" >
        <form>
            <input type="text" placeholder="écrire quelque chose"
ng-model="formData.text">
            <br/>
            <button type="submit" ng-
click="createTodo()">Ajouter</button>
        </form>
    </div>
</body>
</html>
```

14. Voilà! Vous avez fini d'écrire un programme avec MongoDB, Express, Angularjs et Nodejs (MEAN). Avant d'exécuter le programme, installez le package suivant:

    `npm installer -g nodemon`

15. Testez le programme en tapant cette commande dans le terminal: `nodemon server.js` Quelle est la difference entre `node` et `nodemon`?

Exercice

0. Appliquez *Bootstrap* et modifiez l'apparence de `index.html` avec CSS.
1. Séparez le modèle et la route de `server.js` en créant deux nouveaux fichiers appelés `model.js` et `api-route.js`.
2. Ajouter les colonnes (*fields*) de la date de création et le nom créateur à la table (*collection*) Liste. N'oubliez pas de modifier l'interface
3. Ajouter une fonction pour mettre à jour le contenu de la liste.