MongoDB et Mongoose

MONGODB OBJECT MODELING

FERCOQ FLORENT

Sommaire

MongoDB et Node.js

MongoDB et Node.js

Télécharger MongoDB depuis le site official :

- https://www.mongodb.org/downloads
- On trouve des installateurs pour toutes les plateformes les plus courantes

Une fois installé, pour utiliser MongoDB avec une plateforme de développement particulière, un pilote (driver) est nécessaire

- Pilote pour node.js
- Pilote pour PHP

Installation du pilote MongoDB pour Node.js :

```
npm install mongodb --save
```

MongoDB et Node.js

Une fois installé, MongoDB doit être démarré pour être utilisé avec Node.js

Deux possibilités :

- Aller dans le répertoire d'installation et lancer le programme mongod
- Ajouter le répertoire d'installation dans le PATH

```
cd chemin/vers/mongodb

mondgod
```

Se connecter à la base

Le module mongodb doit être chargé

```
var MongoClient = require('mongodb').MongoClient
```

Se connecter à la base

```
// Connection URL
var url = 'mongodb://localhost:27017/myproject';

// Use connect method to connect to the Server
MongoClient.connect(url, function(err, db) {
  if(err)
        throw err;
  console.log("Connected correctly to server");
  db.close();
});
```

Requêtes MongoDB avec Node.js

Une fois connecté, il devient possible d'effectuer différentes requêtes :

- Création de documents et ajouts de champs
- Modification de documents et de leurs champs
- Suppression de documents
- Ajout de collections
- Recherche dans les collections

Présentation de mongoose

Mongoose Overview

Mongoose est un module node.js pour MongoDB

- Il encapsule les fonctionnalités natives du pilote MongoDB
- Il permet l'utilisation de modèles pour contrôler les enregistrements
- Il permet la validation lors des enregistrements
- Il étend les possibilités des requêtes natives

Installer Mongoose

Installer le module mongoose

```
npm install mongoose
```

Dans Node.js

- Charger le module

```
var mongoose = require('mongoose');
```

- Se connecter à la base

```
mongoose.connect(mongoDbPath);
```

- Créer les modèles et utiliser la persistance

```
var Client = mongoose.model('Client', { prenom: String} );

new Client({prenom: 'Patrick'}).save(callback); // création

Client.find({prenom: 'Patrick'}).exec(callback); // requête
```

Les modèles mongoose

Les modèles mongoose

Mongoose utilise des modèles

- Ils permettent de typer les documents et s'utilisent comme les constructeurs JavaScript
- Ils sont créés avec `mongoose.Schema`

```
var modelSchema = new mongoose.Schema({
  propString: String,
  propNumber: Number,
  propObject: {},
  propArray: [],
  propBool: Boolean

});

var Model = mongoose.model('Model', modelSchema);
```

Les modèles mongoose

Chaque propriété a un type

- Les types peuvent être `Number`, `String`, `Boolean`, tableau ou objet
- Il peut aussi s'agir d'objets imbriqués

```
var modelSchema = new mongoose.Schema({
  propNested: {
    propNestedNumber: Number,
    propDoubleNested: {
      propArr: []
    }
});

var Model = mongoose.model('Model', modelSchema);
```

Les modèles mongoose et les méthodes

Comme les modèles mongoose sont simplements des constructeurs, ils peuvent avoir des méthodes

- Elles sont ajoutées au schéma

```
var unitSchema = new mongoose.Schema({…});

unitSchema.methods.deplacer = function(deplacement){
        ...
};
```

- Et peuvent ensuite être appelées

```
var unit = new Unit({ … } );

unit.deplacer({x: 5, y: 6});
```

Les modèles mongoose et les propriétés virtuelles

Toutes les propriétés n'ont pas forcément besoin d'être persistées dans la base de données

- Avec mongoose, il est possible de créer des propriétés qui seront accessibles mais non persistantes dans la base de données
- Ces propriétés peuvent avoir un accesseur et/ou un mutateur

```
var personSchema = new Schema({
        name: {
                first: String,
                last: String
        }
});

personSchema.virtual('fullName')
        .get(function() { return this.name.first + ' ' + this.name.last; })
        .set(function(v) {
                this.name.first = v.substr(0, v.indexOf(' '));
                this.name.last = v.substr(v.indexOf(' ') + 1);
        });
```

La validation

La validation

Pour chaque propriété, il est possible de définir des règles de validation personnalisées

- Ces règles permettent de valider les données lors de la sauvegarde

```
var unitSchema = new mongoose.Schema({…});

unitSchema.path('position.x').validate(function(value){

  return value>=0 && value <= maxX;

});

unitSchema.path('position.y').validate(function(value){

  return value>=0 && value <= maxY;

});
```

Les opérations CRUD avec mongoose

CRUD with Mongoose

Mongoose permet toutes les opérations CRUD :

- Création
  - `objet.save(callback)`
- Recherche
  - `Modele.find(query, fields, options, callback)`
    - Le callback est constitué des deux paramètres `(err, docs)`
  - `Modele.findOne(query, fields, options, callback)`
    - Le callback est constitué des deux paramètres `(err, doc)`
  - `Modele.findById(_id, callback)`
    - Le callback est constitué des deux paramètres `(err, doc)`
- Mise à jour
  - `objet.update(props, callback)`
  - `Model.update(condition, props, multi, callback)`
- Suppression
  - `objet.remove(callback)`
  - `Model.remove(condition, callback)`

Les requêtes

Les requêtes

Mongoose permet de faire des requêtes plus faciles à lire et à concevoir

Au lieu de :

```
{$or: [{conditionOne: true},
       {conditionTwo: true}]
}
```

On écrit :

```
.where({conditionOne: true}).or({conditionTwo: true})
```

Les requêtes

Mongoose gère de nombreux types de requêtes :

- Egalité / Inégalité
- Sélection de plusieurs propriétés
- Tri
- Limite et Décalage

Toutes ces requêtes sont ajoutées via l'objet retourné par `Model.find*()`

- Appeler `.exec()` à la fin pour exécuter la requête