FORMATION FULL STACK

הרווחה, משרד העבודה
והשירותים החברתיים

Node.js

By Charles Cohen

Charles Cohen
Senior Full Stack Developer
charlesc@edandweb.co.il
050-4449764

Cours 8

MongoDB et Node.js…

Plan du module Node.js

Cours	Date	Cours
1	Lun. 20/05	• Introduction to Node.js (1)
2	Mer. 29/05	Building your Stack (2) • Command Line • File System • Arrow Functions • Modular JavaScript with Node.js
3	Mer. 05/06	• Asynchronous JS • Consuming API – HTTP requests
4	Lun. 10/06	• Consuming API – HTTP requests • Exercises – Notes & Weather
5	Lun. 17/06	• Express framework – templates with Handlebars • Building a get API entry point based on previous exercices (get coordinates and weather from mapbox.com and darksky.net) and returning data as JSON • Implement a search address box that consuming the GET API
6	Lun. 01/07	• Promise, await, async, Async module … • Avoiding common pitfalls with Async.js

Module Node.js & MongoDB

Cours	Date	Cours
7	Lun. 08/07	• Introduction to MongoDB • Installation MongoDB server locally • Installation MongoDB GUI • MongoDB - working with Node.js
8	Lun 15/07	• MongoDB Basics • CRUD Operations. • Exrcice: Integrate into our Tasks application a DB • API mongoose ½
9	Lun 22/07	• API Mongoose 2/2 • Async/Await integration with mongoDB
10	Lun 29/07	• Authentication ?

Environnement de travail

Editeur de code:

- Visual Studio Code
  - Extensions:

Liens Utiles:

- Node.js: https://nodejs.org/
- Moteur V8 Javascript: https://v8.dev/
- Express : http://expressjs.com

Résumé du cours 7

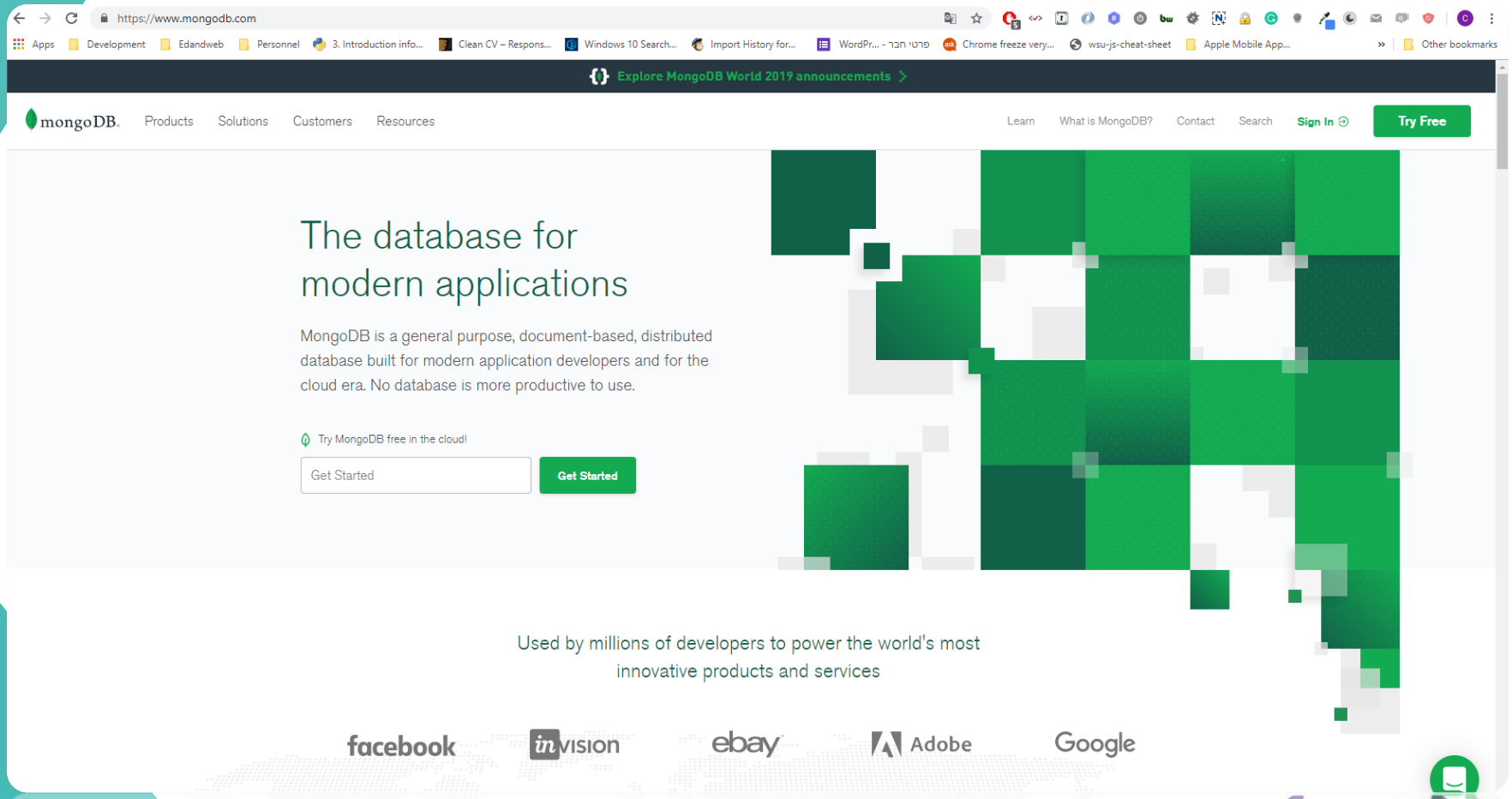
- Decouverte de MongoDB:
  - NoSQL : not only SQL
  - Cluster => moteur de base de donnée Mongo
  - Collection => equivalent à des tables dans une bdd MYSQL
  - Document => equivalent à un enregistrement dans une table
- Installation sous unix:
  - Dézipper l'archive tgz, renommer le dossier (mongodb) et placer le dans le dossier user
  - Créer un autre dossier mongodb-data au même niveau
  - Lancer dans un terminal :
    ~/path_to_your_mongo/bin/mongod –dbpath=/path_to_your_mondb-data/
  - Port by default of mongo 27017
  - Mongodb (official)drivers API: http://mongodb.github.io/node-mongodb-native/3.2/api/
- MongoDB is a cross-platform document-oriented database program.
- Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemata
- Connection au cluster Mongodb
- Insert Document
- Découverte du GUI Compass
- client.close();

Résumé cours 7 - suite

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

const connectionURL = 'mongodb://127.0.0.1:27017'
const databaseName = 'task-manager'

MongoClient.connect(connectionURL, { useNewUrlParser: true }, (error, client) =>
{
    if (error) {
        return console.log('Unable to connect to database’)
    }

    console.log('Connected correctly !');
});
```

Mongo DB

http://www.mongodb.com



Inserting document into MongoDB

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

const connectionURL = 'mongodb://127.0.0.1:27017'
const databaseName = 'task-manager'

MongoClient.connect(connectionURL, { useNewUrlParser: true }, (error, client) => {
    if (error) {
        return console.log('Unable to connect to database’)
    }

    const db = client.db(databaseName)
    db.collection('users').insertOne(
    {
        name: 'Charles’,
        age: 36
    }, (error, result) => {
         if (error) {
              return console.log('Unable to insert user’)
         }

         console.log(result.ops);
    })
});
```

Find document

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

const connectionURL = 'mongodb://127.0.0.1:27017'
const databaseName = 'task-manager'

MongoClient.connect(connectionURL, { useNewUrlParser: true }, (error, client) => {
    if (error) {
        return console.log('Unable to connect to database’)
    }

    const db = client.db(databaseName)
    db.collection('users').findOne({name: 'Charles'}, (error, user) => {
        if (error) {
            return console.log('Unable to find the user’)
        }
        console.log(user);
    })

    // Pointer - go to doc
    db.collection('tasks').find({completed: false}).toArray()
});
```

Update a document

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

const connectionURL = 'mongodb://127.0.0.1:27017'
const databaseName = 'task-manager'

MongoClient.connect(connectionURL, { useNewUrlParser: true }, (error, client) =>
{
    if (error) {
        return console.log('Unable to connect to database’)
    }

    const db = client.db(databaseName)

    db.collection('users').updateOne(
        { name: 'Charles'},
        $set: {
            name: 'Mike’
        }
    )
});
```

Delete a document

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

const connectionURL = 'mongodb://127.0.0.1:27017'
const databaseName = 'task-manager'

MongoClient.connect(connectionURL, { useNewUrlParser: true }, (error, client) =>
{
    if (error) {
        return console.log('Unable to connect to database’)
    }

    const db = client.db(databaseName)

    db.collection('users’).deleteOne(
        { name: 'Charles’}
    ).then((result) => {}).catch((error) => {})
});
```

EXERCICES & TP

Intégrer une base de donnees dans notre application de taches.

TP

Utiliser l’application des taches que nous avons créés et remplacer le système de sauvegarde avec le fichier JSON par une base de données MongoDB.

MongoDB - L’identifiant _id

L’identifiant _id utilise est généré par un algorithme: https://docs.mongodb.com/manual/reference/method/ObjectId/

Nous pouvons le générer nous même et le transmettre à la bdd, pour cela nous utiliserons le constructor/ méthode ObjectID:

```
const ObjectID = mongo.ObjectID
let new_id = new ObjectID()
```

Returns a new ObjectId value. The 12-byte ObjectId value consists of:

- a 4-byte value representing the seconds since the Unix epoch,
- a 5-byte random value, and
- a 3-byte counter, starting with a random value.

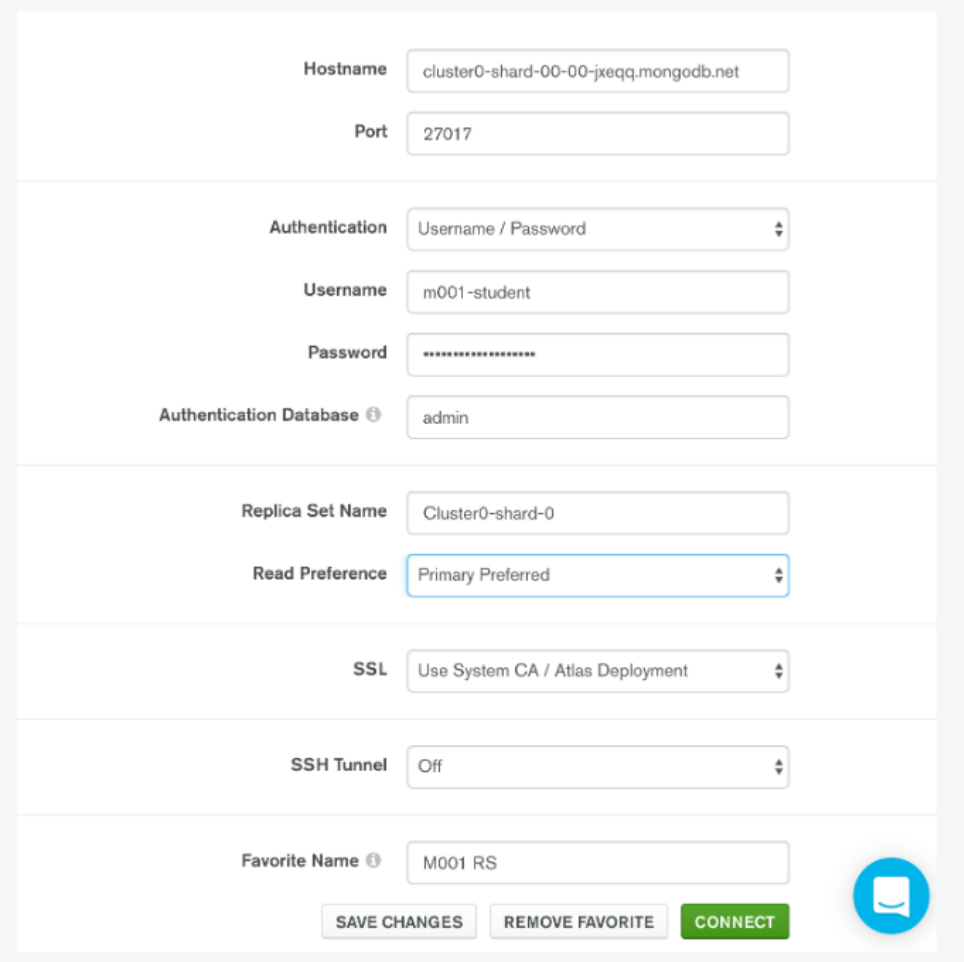
`ObjectId()` can accept the following parameter:

Pour rechercher un id en bdd : ObjectID(‘’ 507f1f77bcf86cd799439011’’)

Attribute/Method	Description
`str`	Returns the hexadecimal string representation of the object.
`ObjectId.getTimestamp()`	Returns the timestamp portion of the object as a Date.
`ObjectId.toString()`	Returns the JavaScript representation in the form of a string literal “`ObjectId(...)`”.
`ObjectId.valueOf()`	Returns the representation of the object as a hexadecimal string. The returned string is the `str` attribute.

Connect to MongoDB learning DB

from : mongodb learning center



- Hostname:
  cluster0-shard-00-00-jxeqq.mongodb.net
- Username
  m001-student
- Password
  m001-mongodb-basics

EXERCICES & TP

TP

What is the value type of the **ts** field for documents in the 100YWeatherSmall.data collection?

- array
- coordinates
- date
- document
- double
- int32
- mixed string and int32
- mixed string and double
- String

EXERCICES & TP

TP

What is the value type of the ***airTemperature*** field for documents in the ***100YWeatherSmall.data*** collection ?

- array
- coordinates
- date
- document
- double
- int32
- mixed string and int32
- mixed string and double
- string

EXERCICES & TP

TP

What is the value type of the ***year*** field for documents in the ***video.movies*** collection?

- array
- coordinates
- date
- document
- double
- int32
- mixed string and int32
- mixed string and double
- string

EXERCICES & TP

TP

How many movies in the video collection were directed by Patty Jenkins. Stated more precisely, how many documents in the video.movies collection have a value of "Patty Jenkins" for the director field?

- 6
- 13
- 47
- 98
- 143

EXERCICES & TP

TP

How many documents in the citibike.trips collection have a tripduration that is greater than or equal to 60 and less than 65?

- 0
- 94
- 216
- 355
- 754

Db.collection.find()

Reference > mongo Shell Methods > Collection Methods > db.collection.find()

db.collection.find()

On this page

- Definition
- Behavior
- Examples

Definition

`db.collection.find(query, projection)`

Selects documents in a collection or view and returns a cursor to the selected documents.

Parameter	Type	Description
`query`	document	Optional. Specifies selection filter using query operators. To return all documents in a collection, omit this parameter or pass an empty document (`{}`).
`projection`	document	Optional. Specifies the fields to return in the documents that match the query filter. To return all fields in the matching documents, omit this parameter. For details, see Projection.

Returns: A cursor to the documents that match the `query` criteria. When the `find()` method "returns documents," the method is actually returning a cursor to the documents.

Console mongo

Pour se connecter à la console nous utiliserons le script /bin/mongo (pour Winddows: mongo.exe).

Commandes:

show dbs
db.getCollectionNames()
show collections

db.createCollection(‘COLLECTION_NAME’)
db.COLLECTION_NAME.insert()

db.COLLECTION_NAME.drop()
db.COLLECTION_NAME.find()
db.COLLECTION_NAME.update()

Opérateurs de comparaisons

https://docs.mongodb.com/manual/reference/operator/query-comparison/

- $gt : greater than
- $lte: less than or equal
- Between runtime10 -20 : {‘’runtime’’: { $gt: 20, $lte: 30 } }
- $ne : not equal
- $in: [‘’a’’, ‘’b’’]
- $exists: boolean (check if specific field exist)
- $type: ‘’int’’(check for a specific type)
- $or : [{QUERY1}, {QUERY2}]
- $and

Opérateurs de comparaisons exemple de requêtes

```
db.movieDetails.find({runtime: {$gt: 90}})

db.movieDetails.find({runtime: {$gt: 90}}, {_id: 0, title: 1, runtime: 1})

db.movieDetails.find({runtime: {$gt: 90, $lt: 120}}, {_id: 0, title: 1, runtime: 1})

db.movieDetails.find({runtime: {$gte: 90, $lte: 120}}, {_id: 0, title: 1, runtime: 1})

db.movieDetails.find({runtime: {$gte: 180}, "tomato.meter": 100}, {_id: 0, title: 1, runtime: 1})

db.movieDetails.find({rated: {$ne: "UNRATED"}}, {_id: 0, title: 1, rated: 1})

db.movieDetails.find({rated: {$in: ["G", "PG"]}}, {_id: 0, title: 1, rated: 1})

db.movieDetails.find({rated: {$in: ["G", "PG", "PG-13"]}}, {_id: 0, title: 1, rated: 1}).pretty()

db.movieDetails.find({rated: {$in: ["R", "PG-13"]}}, {_id: 0, title: 1, rated: 1}).pretty()
```

Opérateurs d’éléments

https://docs.mongodb.com/manual/reference/operator/query-element/

```
db.moviesDetails.find({mpaaRating: {$exists: true}})

db.moviesDetails.find({mpaaRating: {$exists: false}})

db.movieDetails.find({mpaaRating: null})

db.movieDetails.find({})

db.movies.find({viewerRating: {$type: "int"}}).pretty()

db.movies.find({viewerRating: {$type: "double"}}).pretty()
```

Opérateurs Logiques

https://docs.mongodb.com/manual/reference/operator/query-logical/

```
db.movieDetails.find({$or: [{"tomato.meter": {$gt: 95}},
                            {"metacritic": {$gt: 88}}]},
                     {_id: 0, title: 1, "tomato.meter": 1, "metacritic": 1})

db.movieDetails.find({$and: [{"tomato.meter": {$gt: 95}},
                     {"metacritic": {$gt: 88}}]},
                     {_id: 0, title: 1, "tomato.meter": 1, "metacritic": 1})

db.movieDetails.find({"tomato.meter": {$gt: 95},
                      "metacritic": {$gt: 88}},
                     {_id: 0, title: 1, "tomato.meter": 1, "metacritic": 1})

db.movieDetails.find({$and: [{"metacritic": {$ne: null}},
                             {"metacritic": {$exists: true}}]},
                     {_id: 0, title: 1, "metacritic": 1})

db.movieDetails.find({$and: [{"metacritic": null},
                             {"metacritic": {$exists: true}}]},
                     {_id: 0, title: 1, "metacritic": 1})
```

Opérateurs d'Array - $all

https://docs.mongodb.com/manual/reference/operator/query-array/

Reference > Operators > Query and Projection Operators > Array Query Operators

Array Query Operators

NOTE:
For details on specific operator, including syntax and examples, click on the specific operator to go to its reference page.

Name	Description
`$all`	Matches arrays that contain all elements specified in the query.
`$elemMatch`	Selects documents if element in the array field matches all the specified `$elemMatch` conditions.
`$size`	Selects documents if the array field is a specified size.

```
db.movieDetails.find({genres: {$all: ["Comedy", "Crime", "Drama"]}},
                     {_id: 0, title: 1, genres: 1}).pretty()
```

Opérateurs d'Array - $size

https://docs.mongodb.com/manual/reference/operator/query-array/

Recherche sur un champ de type Array qui contient seulement 1 élément :

```
db.movieDetails.find({countries: {$size: 1}}).pretty()
```

Opérateurs d'Array - $elemMatch

https://docs.mongodb.com/manual/reference/operator/query-array/

Convient pour un Array qui contient des documents

```
boxOffice: [
    {"country": "USA", "revenue": 228.4},
    {"country": "Australia", "revenue": 19.6},
    {"country": "UK", "revenue": 33.9},
    {"country": "Germany", "revenue": 16.2},
    {"country": "France", "revenue": 19.8}
]
```

```
db.movieDetails.find({"boxOffice.country": "Germany",
                      "boxOffice.revenue": {$gt: 17}})
```

```
db.movieDetails.find({boxOffice: {$elemMatch: {"country": "Germany",
                                                "revenue": {$gt: 17}}}})
```

Fin