# Mise en place

Créez un répertoire db-ref

C:\Users\DD>mkdir 📁db-ref

C:\Users\DD>cd db-ref

Lancez Visual Studio.

C:\Users\DD\db-ref>code .[1]

# Installation des modules

Nous devons initialiser un fichier package.json qui servira de support aux installations des différents modules.

Lancez dans votre éditeur un terminal à l'aide de la commande crtl+ù

🪛Lancez npm init -y pour créer le fichier  package.json.

Installez mongoose avec npm i mongoose

# Choix d'une DB

Vous avez le choix de travailler en local ou avec le cloud Atlas.

Pour le cloud

1. mongoose.connect('mongodb+srv://nom:pass@cluster0.code.mongodb.net/TD?retryWrites=true&w=majority')
2. .then(() => console.log('connected to MongoDB'))
3. .catch(err => console.error(' Could not connect to MongoDB'))

[1] notez le point .

Nous allons travailler en local.

Commencez par définir le schéma d'un auteur. On se limite à deux propriétés { name: String, website: String}.

Créez un fichier  populate.js

 populate.js

```
const mongoose = require("mongoose");

mongoose
  .connect("mongodb://127.0.0.1:27017/mongo-references")
  .then(() => console.log("Connected to MongoDB..."))
  .catch((err) => console.error("Could not connect to MongoDB...", err));

const Author = mongoose.model(
  "Author",
  new mongoose.Schema({
    name: String,
    website: String,
  })
);

async function createAuthor(name, website) {
  const author = new Author({
    name,
    website,
  });

  const result = await author.save();
```

```
23.  console.log(result);
24.}
25.createAuthor("superDupont", "blogspot");
```

🪛Lancez la commande `node population.js`

```
$ node population.js
```

Vérifiez la création d'un document dans la collection authors à l'aide de Visual Studio et de l'extension mongodb.


Vous allez créer une référence du modèle de cours avec une référence sur l'auteur.

Modifiez votre fichier populate.js et repérez dans le code la syntaxe :

```
author: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Author",
  },
```

populate.js

```
1.  const mongoose = require("mongoose");
```

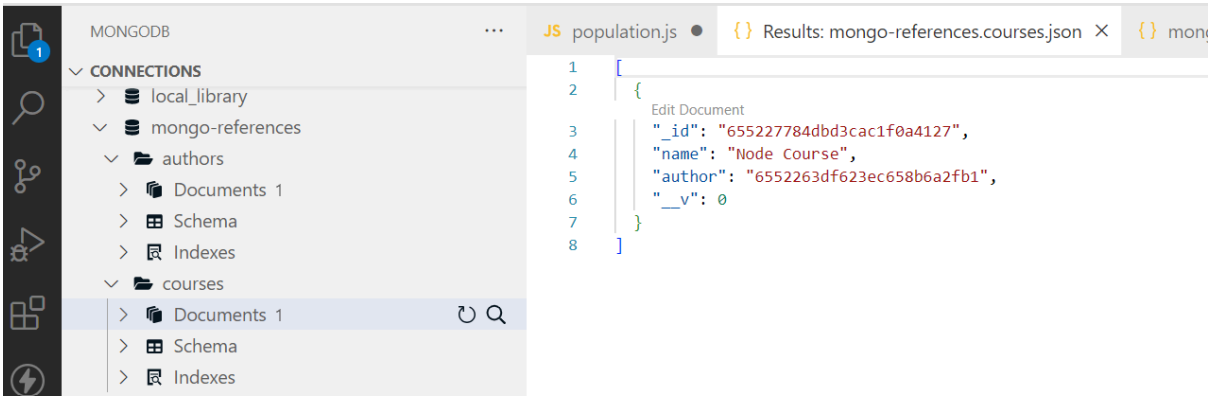
```
2.
3.  mongoose
4.    .connect("mongodb://127.0.0.1:27017/mongo-references")
5.    .then(() => console.log("Connected to MongoDB..."))
6.    .catch((err) => console.error("Could not connect to MongoDB...", err));
7.
8.  const Author = mongoose.model(
9.    "Author",
10.  new mongoose.Schema({
11.    name: String,
12.    website: String,
13.  })
14.);
15.
16.const Course = mongoose.model(
17.  "Course",
18.  new mongoose.Schema({
19.    name: String,
20.    author: {
21.      type: mongoose.Schema.Types.ObjectId,
22.      ref: "Author",
23.    },
24.  })
25.);
26.
27.async function createAuthor(name, website) {
28.  const author = new Author({
29.    name,
30.    website,
31.  });
```

```
32.
33.  const result = await author.save();
34.  console.log(result);
35.}
36.
37.async function createCourse(name, author) {
38.  const course = new Course({
39.    name,
40.    author,
41.  });
42.
43.  const result = await course.save();
44.  console.log(result);
45.}
46.
47.//createAuthor("superDupont", "My Website");
48.createCourse("Node Course", "6552263df623ec658b6a2fb1");
```

Lig.21 : On fait une référence sur un auteur.

Lig.48 : Il vous faudra mettre le numéro de votre auteur.

Exécutez le code avec `node populate.js`




Ajoutez la fonction de lecture des cours au fichier populate.js

🔧Modifiez votre fichier populate.js et repérez le code suivant

```
async function listCourses() {
  const courses = await Course.find().select("name");
  console.log(courses);
}
```

populate.js

```
1. const mongoose = require("mongoose");
2.
3. mongoose
4.   .connect("mongodb://127.0.0.1:27017/mongo-references")
5.   .then(() => console.log("Connected to MongoDB..."))
6.   .catch((err) => console.error("Could not connect to MongoDB...", err));
7.
8. const Author = mongoose.model(
9.   "Author",
```

```
10.  new mongoose.Schema({
11.    name: String,
12.    website: String,
13.  })
14.);
15.
16.const Course = mongoose.model(
17.  "Course",
18.  new mongoose.Schema({
19.    name: String,
20.    author: {
21.      type: mongoose.Schema.Types.ObjectId,
22.      ref: "Author",
23.    },
24.  })
25.);
26.
27.async function createAuthor(name, website) {
28.  const author = new Author({
29.    name,
30.    website,
31.  });
32.
33.  const result = await author.save();
34.  console.log(result);
35.}
36.
37.async function createCourse(name, author) {
38.  const course = new Course({
39.    name,
```

```
    author,
  });

  const result = await course.save();
  console.log(result);
}

async function listCourses() {
  const courses = await Course.find().select("name");
  console.log(courses);
}

// createAuthor("superDupont", "My Website");
// createCourse("Node Course", "6552263df623ec658b6a2fb1");
listCourses();
```

🪛Lancez la commande `node population.js`

```
$ node population.js
Connected to MongoDB...
[
  {
    _id: new ObjectId('655227784dbd3cac1f0a4127'),
    name: 'Node Course'
  }
]
```

🪛Modifiez le select de la fonction select

48. const courses = await Course.find().select("name **author**");

🔧Lancez la commande `node population.js`

```
$ node population.js
Connected to MongoDB...
[
 {
   _id: new ObjectId('655227784dbd3cac1f0a4127'),
   name: 'Node Course',
   author: new ObjectId('6552263df623ec658b6a2fb1')
 }
]
```

L'affichage de author: new ObjectId('6552263df623ec658b6a2fb1')

n'est pas lisible. Nous allons ajouter les valeurs de l'auteur.

🪛Ajoutez à la lig.48 la méthode populate

```
const courses = await Course.find().populate("author").select("name author");
```

🔧Lancez la commande `node population.js`

```
$ node population.js
[
 {
   _id: new ObjectId('655227784dbd3cac1f0a4127'),
   name: 'Node Course',
```

```
    author: {
      _id: new ObjectId('6552263df623ec658b6a2fb1'),
      name: 'superDupont',
      __v: 0
    }
  }
]
```

Pour n'afficher que le nom de l'auteur mettez un second argument qui inclut les propriétés ciblées.

🪛Ajoutez à la lig.48 la méthode populate

```
const courses = await Course.find().populate("author","name").select("name author");
```

🔧Lancez la commande `node population.js`

```
$ node population.js
[
  {
    _id: new ObjectId('655227784dbd3cac1f0a4127'),
    name: 'Node Course',
    author: {
```

```
    _id: new ObjectId('6552263df623ec658b6a2fb1'),
    name: 'superDupont'
  }
 }
]
```

Pour exclure une propriété on utilise le signe - devant la propriété en second paramètre.

🪛Ajoutez à la lig.48 la méthode populate

```
const courses = await Course.find().populate("author","-_id").select("name author");
```

🔧Lancez la commande `node population.js`

```
$ node population.js
Connected to MongoDB...
[
 {
   _id: new ObjectId('655227784dbd3cac1f0a4127'),
   name: 'Node Course',
   author: { name: 'superDupont', __v: 0 }
 }
]
```

Pour inclure et exclure une propriété, on écrira simplement

🪛Ajoutez à la lig.48 la méthode populate

const courses = await Course.find().populate("author",**"name -_id"**).select("name author");

On exclut _id ( avec le signe -)et inclut name.

🔧Lancez la commande node population.js

$ node population.js

Connected to MongoDB...

```
[
  {
    _id: new ObjectId('655227784dbd3cac1f0a4127'),
    name: 'Node Course',
    author: { name: 'superDupont' }
  }
]
```