CSCI446 – Connecting MongoDB to Express

MongoDB Node Connector

- MongoDB publishes a basic driver for Node runtimes[1]
  - This is what we'll be using
- API very similar to what we've seen in the Mongo shell (`mongosh`)
- Installation:

```
$ npm add mongodb
```

1. MongoDB package on NPM ↩️

Connecting to MongoDB

- We use the `MongoClient` class to instantiate a connection
- Takes a connection URI string, format:
  - Specifically: `mongodb://localhost:27017`

```
1  import { MongoClient } from 'mongodb';
2
3  async function connect() {
4    const client = new MongoClient('mongodb://localhost:27017');
5    await client.connect();
6
7    return client;
8  }
```

Aside: Breaking down the connection URI

- The URI has six parts:
  - scheme
  - username
  - password
  - host
  - database
  - options
- Example: `{scheme}://{username}:{password}@{host}/{database}?{options}`

Reusing the connection

- By default, `MongoClient` pools connections
  - This means we have multiple connections open to the database that we can borrow and use
  - Having a pool of connections allows us to not have to open a new connection on every request
- We can configure the pool size using a second options argument[1]:
  - `maxPoolSize` sets the maximum number of connections that can be opened, defaults to 100
  - `minPoolSize` sets the minimum number, defaults to 0
- As we use the database, it'll open new connections up to the max

---

1. MongoClientOptions docs ↩️

Extending our connection function

- We can extend the function to take a database name or automatically set the collection

```
// Setting the database by default
async function connect(databaseName) {
  const client = new MongoClient('mongodb://localhost:27017');
  await client.connect();

  return client.db(databaseName);
}
```

```
// Setting the database and collection by default
async function connect(databaseName, collectionName) {
  const client = new MongoClient('mongodb://localhost:27017');
  await client.connect();

  return client.db(databaseName).collection(collectionName);
}
```

Using `Application.{set, get}` in Express

- Express has a way of setting properties globally on the application
  - `Application.set()` and `Application.get()`
  - `set(name, value)` takes a key and value ` get(name)` returns the value with given key
- We can store objects of all kinds using this scheme
- To retrieve, we get the `app` reference through the `request` argument

```
async function connect(databaseName) {
  const client = new MongoClient("mongodb://localhost:27017");
  await client.connect();
  return client.db(databaseName);
}

app.set("db", connect('test'));
```

```
app.get('/', async (req, res) => {
  const db = await req.app.get('mongoDatabase');
  const allWidgets = await db.collection('widgets').find().toArray();
  return res.json(allWidgets);
});
```