BSc (Hons) in Information Technology
Specializing in Software Engineering
Year 3 - 2021
SE3040 – Application Frameworks
Lab 06 – Koa.js part 02

1. Make sure that you have MongoDB running instance.
2. If not start MongoDB on the local machine or remote (the below instructions are based on assuming the instance is running on the local machine without authentication).
3. Install mongodb driver to your application.
   npm install mongodb --save
4. Create a new directory as 'dal' to keep all DB related code.
5. Add a file named 'index.js' with the following to initiate the client.

```
const {MongoClient} = require('mongodb');

const client = new MongoClient('mongodb://localhost:27017', {
    useNewUrlParser: true,
    useUnifiedTopology: true
});

client.connect(err => {
    if (err) {
        console.error(err);
        process.exit(-1);
    }

    console.log('Successfully connected to Mongo DB');
});

module.exports = client;
```

6. Crate a file to support the db insert of a post (all other DB level operations mentioned below should go in this file).

```
const posts = require('./').db('posts').collection('posts');

const save = async ({id, name, description, postedDate}) => {
    const result = await posts.insertOne({id, name, description,
postedDate});
    return result.ops[0];
};

module.exports = {save};
```

7. Use the method created above to insert by removing the in-memory map.

```
const UUID = require('uuid');

const {getAll, getById, removeById, save, update} =
require('../dal/posts.dao');

const createPost = async ({name, description}) => {
    const post = {
        id: UUID.v4(),
```

```
            name,
            description,
            postedDate: new Date()
        }
        return await save(post);
    }
```

8. Change the route file accordingly as well.

```
router.post('/', async ctx => {
    let post = ctx.request.body;
    post = await createPost(post);
    ctx.response.status = 201;
    ctx.body = post;
});
```

9. Make sure you run your DB connection file in the app.js.

```
require('./dal');
```

10. Add a method to get all posts from the DB.

```
const getAll = async () => {
    const cursor = await posts.find();
    return cursor.toArray();
};
```

11. Update the API and route file accordingly (refer 7, 8).
12. Add a method to get post by ID to work with the DB.

```
const getById = async id => {
    return await posts.findOne({id});
};
```

13. Update the API and route file accordingly (refer 7, 8).
14. Add two DB operation to insert and replace the document.

```
const removeById = async id => {
    await posts.deleteOne({id});
};

const update = async (id, {name, description, postedDate}) => {
    const result = await posts.replaceOne({id}, {id, name,
description, postedDate});
    return result.ops[0];
};
```

15. Do the needful wire up in API and route file to get PUT and DELETE operations working.

Optional – Serving static files.

1. Install koa-static into your project.
2. Create a directory called public and add 2 html files to it.

home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Home</title>
</head>
<body>
<h1>Welcome</h1>
<h3>This is server by KOA</h3>
<h5>This is from the route /home</h5>
<p>
    <a href="about.html">About</a>
</p>
</body>
</html>
```

about.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>About</title>
</head>
<body>
<h1>This is the about page</h1>
</body>
</html>
```

3. Register koa-static to your Koa application by given your static directory ‘public’. Register this after all other routes.

```
serve = require('koa-static')
```

```
app.use(serve('public/'));
```

4. Start the application and navigate to http://localhost:3000/home.html to see your static files being served by Koa.