

MỤC TIÊU:

Kết thúc bài thực hành này bạn có khả năng

- ✓ Biết cách sử dụng hệ quản trị cơ sở dữ liệu NoSQL
- ✓ Sử dụng nodejs với NoSQL
- ✓ Biết sử dụng thư viện mongoose

PHẦN I

Bài 1 (3 điểm)

Thao tác đọc và tạo cơ sở dữ liệu, collection và document cho mongodb với nodejs

Bước 1: Cài đặt mongodb và trình điều khiển mongodb

Bước 2: Tổ chức project lab08 và thực hiện kết nối với mongodb

<pre>▼ Lab08 > controllers ▼ models JS post.js > node_modules > public ▼ routes JS blog.js ▼ util JS database.js</pre>	<pre>util > JS database.js > ... const mongodb = require('mongodb'); const MongoClient = mongodb.MongoClient; let _db; MongoClient.connect('mongodb://localhost:27017/blogDb') .then(client => { console.log('Connected!'); _db = client.db(); }) .catch(err => { console.log(err); throw err; }); const getDb = () => { if (_db) { return _db; } throw 'No database found!'; }; exports.getDb = getDb;</pre>
---	---

Bước 3: Viết hàm xử lý tạo collection và thêm một document:

```
1  const mongodb = require('mongodb');
2  const getDb = require('../util/database').getDb;
3  module.exports = class Post {
4      constructor(title, content, create_date) {
5          this.title = title;
6          this.content = content;
7          this.create_date = create_date;
8      }
9      //thêm một bài viết
10     save() {
11         const db = getDb();
12         return db
13             .collection('posts')
14             .insertOne(this)
15             .then(result => {
16                 console.log(result);
17             })
18             .catch(err => {
19                 console.log(err);
20             });
21     }
22 }
```

Bước 4: Xây dựng controller và định nghĩa Route cho blog

```
35 exports.createPost = (req, res, next) => {
36     const title = req.body.title;
37     const content = req.body.content;
38     const post = new Post(title, content, new Date().toISOString());
39     post
40         .save()
41         .then(result => {
42             res.status(201).json({
43                 message: 'Thêm thành công bài viết mới!',
44                 post: result
45             });
46         })
47         .catch(err => {
48             if (!err.statusCode) {
49                 err.statusCode = 500;
50             }
51             next(err);
52         });
53 };
```

```
const express = require('express');

const blogController = require('./controllers/blog');

const router = express.Router();

// GET /blog/posts
// POST /blog/post
router.post('/posts', blogController.createPost);

module.exports = router;
```

Phần server: sử dụng hệ thống route đã xây dựng bước trên

```
const express = require('express');
const bodyParser = require('body-parser');
const blogRoutes = require('./routes/blog');

const app = express();
app.use(bodyParser.json()); // application/json
const port = 3000;
app.use('/blog', blogRoutes);

app.listen(port, () => {
  console.log(`Ứng dụng đang chạy với port: ${port}`);
})
```

Bài 2 (2 điểm)

Viết các hàm thực hiện tìm theo id, xóa document theo id, sửa document theo id

Test tất cả chức năng trên với công cụ Postman

PHẦN II

Bài 3 (2 điểm)

Sử dụng mongoose và mongodb Xây dựng RESTFUL API tương ứng cho trang quản lý bài viết (post) của 1 blog có giao diện như sau

#	Tiêu đề	Ngày cập nhật	Thao tác
1	New MV	20-08-2020	Xóa Sửa
2	Phong Tom	19-08-2020	Xóa Sửa

Thêm - sửa bài viết

Tiêu đề bài viết

Nội dung bài viết

Lưu

Bước 1: Phân tích và tạo cơ sở dữ liệu: sinh viên tự xây dựng

Bước 2: Thiết kế REST API

HTTP method	Route	Body	Access
GET	{host}/blog/posts	Không	Public
GET	{host}/blog/posts/:id	Không	Public
POST	{host}/blog/posts	Có	Admin
PUT	{host}/blog/posts/:id	Có	Admin

DELETE	{host}/blog/posts/:id	Không	Admin
--------	-----------------------	-------	-------

Tương ứng với routing như sau:

```
Lab05_2 > routes > JS blog.js > ...
1   const express = require('express');
2
3   const blogController = require('../controllers/blog');
4
5   const router = express.Router();
6
7   // GET /blog/posts
8   router.get('/posts', blogController.getPosts);
9   router.get('/posts/:postId', blogController.getPostById);
10  // POST /blog/post
11  router.post('/posts', blogController.createPost);
12  //update
13  router.put('/posts/:postId', blogController.deletePost);
14  //delete
15  router.delete('/posts/:postId', blogController.updatePost);
16
17  module.exports = router;
```

Bước 3: Xây dựng REST API

- Tạo model:

Cài đặt mongoose để thực hiện mô hình ODM

```
mongoose
  .connect(
    'mongodb://localhost:27017/blogDb'
  )
  .then(result => {
    app.listen(port, () => {
      console.log(' ứng dụng đang chạy với port: ${port}');
    });
  })
  .catch(err => {
    console.log(err);
  });
```

Tạo lược đồ

```
1  const mongoose = require('mongoose');
2
3  const Schema = mongoose.Schema;
4
5  const postSchema = new Schema({
6    title: {
7      type: String,
8      required: true
9    },
10   content: {
11     type: String,
12     required: true
13   },
14   create_date: {
15     type: Date,
16     required: true
17   }
18 });
19 module.exports = mongoose.model('posts', postSchema);
```

Bài 4 (2 điểm)

Viết các REST API để thay đổi dữ liệu trong database

- POST: Thực hiện thêm một bài viết mới

```
Lab05_2 > controllers > JS blog.js > ...
36 exports.createPost = (req, res, next) => {
37   const title = req.body.title;
38   const content = req.body.content;
39   const post = new Post({ title: title, content: content, create_date: new Date().toISOString() });
40   post
41     .save()
42     .then(result => {
43       res.status(201).json({
44         message: 'Thêm thành công bài viết mới!',
45         post: result
46       });
47     })
48     .catch(err => {
49       if (!err.statusCode) {
50         err.statusCode = 500;
51       }
52       next(err);
53     });
54 };
```

- PUT: Thực hiện cập nhật một bài viết theo mã

```
Lab05_2 > controllers > JS blog.js > ...
56 exports.updatePost = (req, res, next) => {
57   const postId = req.params.postId;
58   const title = req.body.title;
59   const content = req.body.content;
60
61   Post.findById(postId)
62     .then(post => {
63       if (!post) {
64         const error = new Error('Không tìm thấy bài viết - post. ');
65         error.statusCode = 404;
66         throw error;
67       }
68       post.title = title;
69       post.content = content;
70       return post.save();
71     })
72     .then(result => {
73       res.status(200).json({ message: 'Post update thành công!', post: result });
74     })
75     .catch(err => {
76       if (!err.statusCode) {
77         err.statusCode = 500;
78       }
79       next(err);
80     });
81 };
```

- DELETE: Xóa 1 bài post theo mã bài

```
Lab05_2 > controllers > JS blog.js > ...
83   exports.deletePost = (req, res, next) => {
84     const postId = req.params.postId;
85     Post.findByPk(postId)
86       .then(post => {
87         if (!post) {
88           const error = new Error('Không tìm thấy bài viết - post. ');
89           error.statusCode = 404;
90           throw error;
91         }
92         return post.destroy(postId);
93       })
94       .then(result => {
95         console.log(result);
96         res.status(200).json({ message: 'Đã xóa post.' });
97       })
98       .catch(err => {
99         if (!err.statusCode) {
100           err.statusCode = 500;
101         }
102         next(err);
103       });
104   };
```

Bài 5 (1 điểm)

Thực hiện test tất cả các API đã thực hiện ở bài 4 với công cụ postman.

*** Yêu cầu nộp bài:

SV nén file (hoặc share thư mục google drive) bao gồm các yêu cầu đã thực hiện trên, nộp LMS đúng thời gian quy định của giảng viên. KHÔNG NỘP BÀI COI NHƯ KHÔNG CÓ ĐIỂM.

---Hết---