



FPT POLYTECHNIC



THỰC HỌC – THỰC NGHIỆP



Conceive Design Implement Operate

NODEJS & RESFUL WEB SERVICE

NODEJS VỚI NOSQL

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



- 📖 Cài đặt hệ quản trị NoSQL - mongodb
- 📖 Làm việc với mongodb bằng thư mongodb
 - ❖ Cài đặt thư viên mongodb
 - ❖ Truy xuất tới cơ sở dữ liệu mongodb
- 📖 Làm việc với mongodb bằng mongose



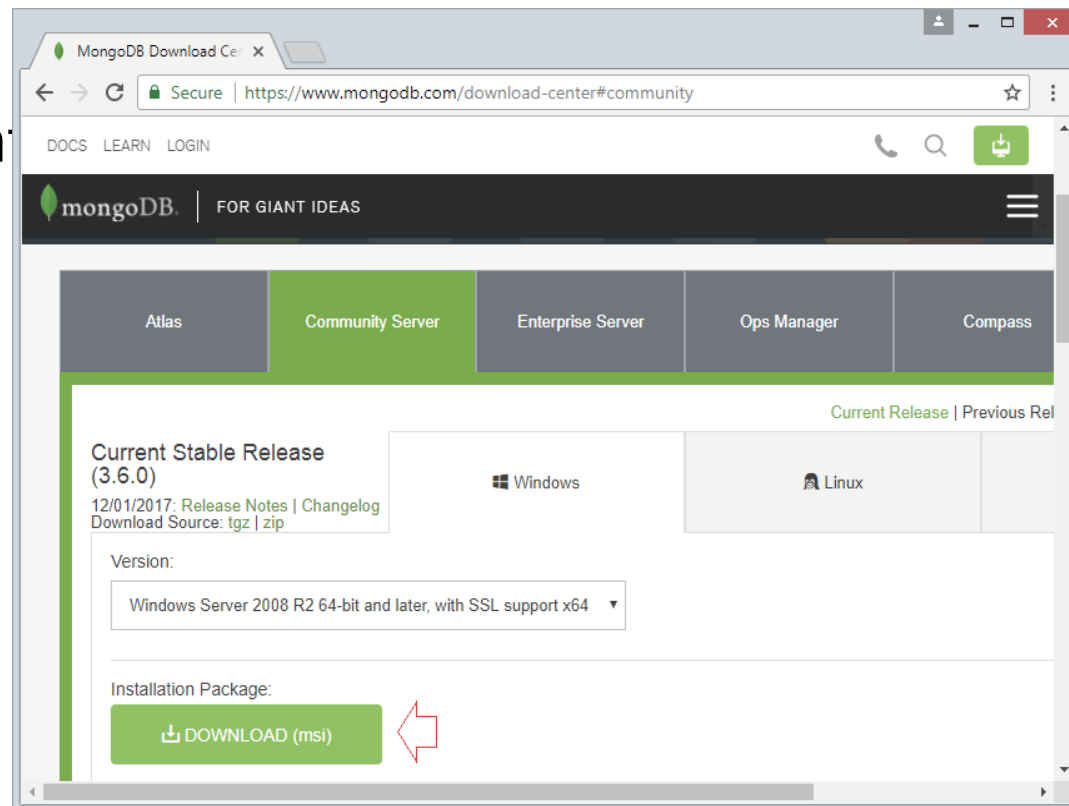


PHẦN 1: NODEJS VÀ MONGODB

❑ Là hệ quản trị cơ sở dữ liệu nosql

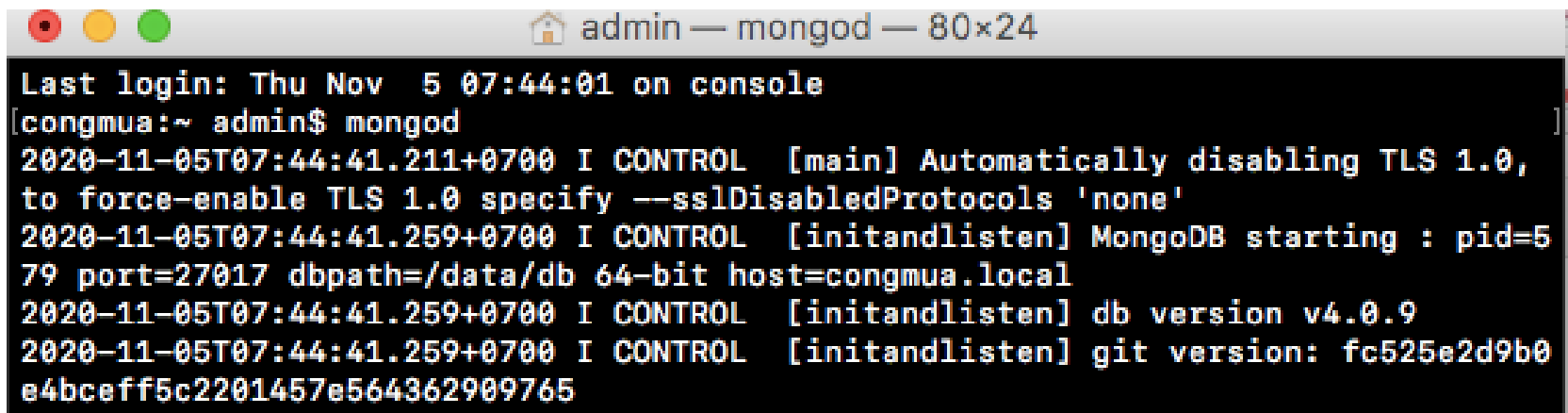
❑ Thực hiện:

❖ Download và cài đặt



❑ Chạy MongoDB:

- ❖ Mở terminal/command line và chuyển tới thư mục **bin** của **mongodb**.
- ❖ Dùng lệnh **mongod**

A screenshot of a macOS terminal window titled 'admin — mongod — 80x24'. The terminal shows the output of the 'mongod' command. It starts with 'Last login: Thu Nov 5 07:44:01 on console', followed by '[congmuai:~ admin\$ mongod]'. The output includes several log messages: '2020-11-05T07:44:41.211+0700 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'', '2020-11-05T07:44:41.259+0700 I CONTROL [initandlisten] MongoDB starting : pid=579 port=27017 dbpath=/data/db 64-bit host=congmuai.local', '2020-11-05T07:44:41.259+0700 I CONTROL [initandlisten] db version v4.0.9', and '2020-11-05T07:44:41.259+0700 I CONTROL [initandlisten] git version: fc525e2d9b0e4bceff5c2201457e564362909765'.

```
Last login: Thu Nov 5 07:44:01 on console
[congmuai:~ admin$ mongod
2020-11-05T07:44:41.211+0700 I CONTROL [main] Automatically disabling TLS 1.0,
to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2020-11-05T07:44:41.259+0700 I CONTROL [initandlisten] MongoDB starting : pid=5
79 port=27017 dbpath=/data/db 64-bit host=congmuai.local
2020-11-05T07:44:41.259+0700 I CONTROL [initandlisten] db version v4.0.9
2020-11-05T07:44:41.259+0700 I CONTROL [initandlisten] git version: fc525e2d9b0
e4bceff5c2201457e564362909765
```

❑ Cần cài đặt và sử dụng trình điều khiển MongoDB

❖ Dùng Command Terminal và thực hiện như sau: npm

install --save mongodb

❖ Bây giờ ta có thể sử dụng mô-đun này để thao tác cơ

sở dữ liệu MongoDB: ***const mongo =***

require('mongodb')

Kết nối đến csdl MongoDB trên máy cục bộ

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

```
// kết nối đến csdl
```

*import mô đun mongodb và đối tượng
MongoClient*

```
MongoClient.connect("mongodb://localhost:27017/BlogDb",
```

```
function (err, db) { if(err) throw err;
```

```
// viết code thêm, xóa, sửa tại đây...
```

```
});
```

*Phương thức connect () trả về tham chiếu cơ
sở dữ liệu nếu cơ sở dữ liệu được chỉ định đã
tồn tại, nếu không nó sẽ tạo ra một cơ sở dữ
liệu mới*

Node.js MongoDB Create Collection

Một **collection** trong MongoDB giống như một **bảng** trong MySQL

Tạo collection: Để tạo một collection trong MongoDB, sử dụng phương thức `createCollection()`

```
const MongoClient = require('mongodb').MongoClient;
const url = "mongodb://localhost:27017/blogDB";
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  db.createCollection("posts", function(err, res) {
    if (err) throw err; console.log("Collection created!");
    db.close();
  }); });
```

Tổ chức code Node.js MongoDB tạo Collection

```
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;
```

```
models > JS post.js > ...
const getDb = require('../util/database').getDb;
module.exports = class Post {
  constructor(title, content, create_date) {
    this.title = title;
    this.content = content;
    this.create_date = create_date;
  }
  //thêm một bài viết
  save() {
    const db = getDb();
    return db
      .collection('posts')
      .insertOne(this)
      .then(result => {
        console.log(result);
      })
      .catch(err => {
        console.log(err);
      });
  }
}
```

- ▼ Lab08
 - ▼ controllers
 - JS blog.js
 - > models
 - > node_modules
 - > public
 - ▼ routes
 - JS blog.js
 - ▼ util
 - JS database.js

Node.js MongoDB Insert

Một **document** trong MongoDB giống như một **bản ghi** trong MySQL

Để chèn một bản ghi, hoặc *document* như nó được gọi trong MongoDB, vào một collection, ta sử dụng phương thức `insertOne()`

```
models > JS post.js > ...
const getDb = require('../util/database').getDb;
module.exports = class Post {
  constructor(title, content, create_date) {
    this.title = title;
    this.content = content;
    this.create_date = create_date;
  }
  //thêm một bài viết
  save() {
    const db = getDb();
    return db
      .collection('posts')
      .insertOne(this)
      .then(result => {
        console.log(result);
      })
      .catch(err => {
        console.log(err);
      });
  }
}
```

Node.js MongoDB với phương thức find

Tìm tất cả

Để chọn dữ liệu từ một bảng trong MongoDB, ta dùng find(). Phương thức này trả về tất cả những gì xuất hiện.

```
db
.collection('posts')
.find()
.toArray()
.then(posts => {
  console.log(posts);
  return posts;
})
.catch(err => {
  console.log(err);
});
```

Node.js MongoDB với phương thức find

Lọc kết quả

Khi tìm document trong collection, ta có thể lọc kết quả bằng cách sử dụng đối tượng truy vấn. Đối số đầu tiên của **find()** là đối tượng truy vấn và được sử dụng để giới hạn tìm kiếm.

```
//tìm bài viết theo id
static findById(postId) {
  const db = getDb();
  return db
    .collection('posts')
    .find({ _id: new mongodb.ObjectId(postId) })
    .next()
    .then(post => {
      console.log(post);
      return post;
    })
    .catch(err => {
      console.log(err);
    });
}
```

Node.js MongoDB với phương thức find

Lọc kết quả

Khi tìm document trong collection, ta có thể lọc kết quả bằng cách sử dụng đối tượng truy vấn. Đối số đầu tiên của **find()** là đối tượng truy vấn và được sử dụng để giới hạn tìm kiếm.

```
//tìm bài viết theo id
static findById(postId) {
  const db = getDb();
  return db
    .collection('posts')
    .find({ _id: new mongodb.ObjectId(postId) })
    .next()
    .then(post => {
      console.log(post);
      return post;
    })
    .catch(err => {
      console.log(err);
    });
}
```

Node.js MongoDB với update

```
if (this._id) {  
  // Update the post  
  db  
    .collection('posts')  
    .updateOne({ _id: new mongodb.ObjectId(this._id) }, { $set: this });  
} else {  
  db.collection('posts').insertOne(this);  
}
```

Node.js MongoDB với delete

```
db
.collection('posts')
.deleteOne({ _id: new mongodb.ObjectId(prodId) })
.then(result => {
  console.log('Deleted');
})
.catch(err => {
  console.log(err);
});
```


Test với postman

GET Send

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

Query Params

| KEY | VALUE | DESCRIPTION |
|-----|-------|-------------|
| Key | Value | Description |

Body Cookies Headers (9) Test Results ⊕ Status: 200 OK Time: 30 ms Size: 539 B Save

Pretty Raw Preview Visualize JSON ↺

```

1 {
2   "message": "Fetched posts successfully.",
3   "posts": [
4     {
5       "_id": "5fa2e36250dee027f0ee9985",
6       "title": "C#",
7       "content": "Lập trình C#",
8       "create_date": "2020-11-04T17:22:42.872Z"
9     }
10  ]
11 }
```

DELETE Send

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

Query Params

| KEY | VALUE | DESCRIPTION |
|-----|-------|-------------|
| Key | Value | Description |

Body Cookies Headers (9) Test Results ⊕ Status: 200 OK Time: 113 ms Size: 402 B Save

Pretty Raw Preview Visualize JSON ↺

```

1 {
2   "message": "Đã xóa post."
3 }
```

demo

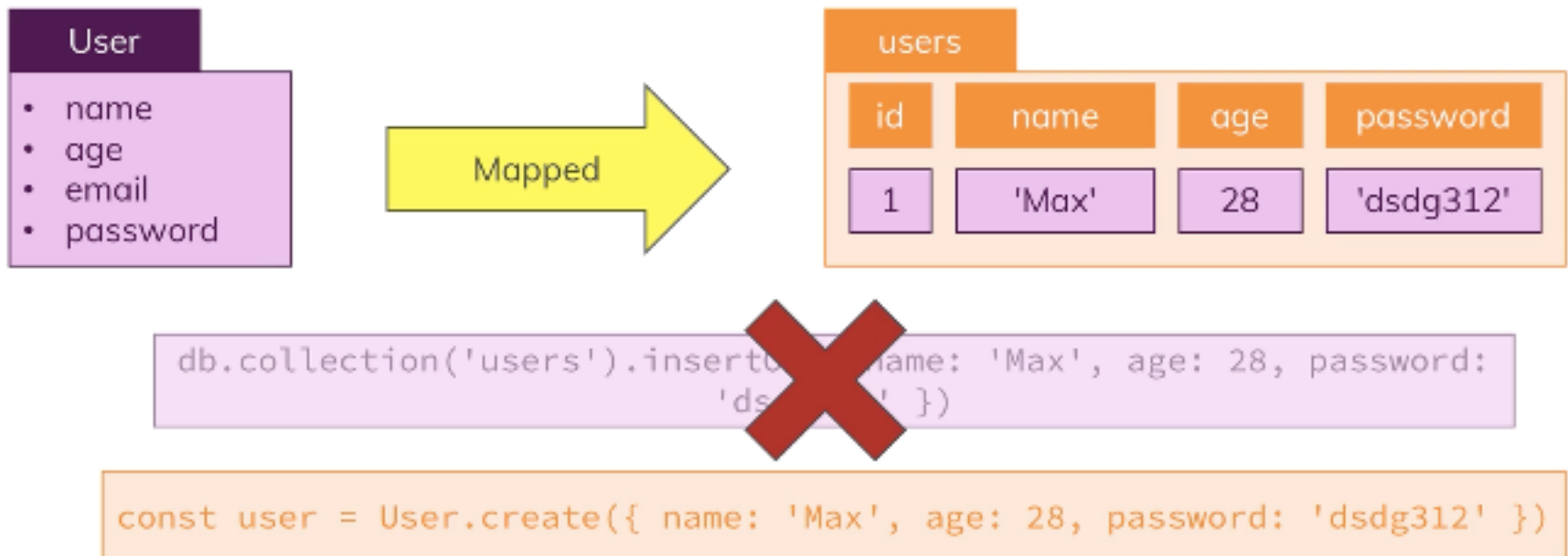


PHẦN 2: LÀM VIỆC VỚI MONGOOSE

Mongoose là gì?

Là một ODM dành cho **Node. Js**

An Object – Document Mapping Library



Kết nối Mongoose với mongodb

Cài đặt mongoose

Npm install –save mongoose

Kết nối mongoose với mongodb

```
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 ánh xạ Mongoose với mongodb

Lược đồ được xây dựng tại models

Ví dụ: lược đồ cho collection posts

```
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);
```

Lưu dữ liệu với Mongoose

Sử dụng hàm save của đối tượng mongoose

Ví dụ: Thêm một bài post

```
exports.createPost = (req, res, next) => {  
  const title = req.body.title;  
  const content = req.body.content;  
  const post = new Post({ title: title, content: content, create_date: new Date().toISOString() });  
  post.save()  
    .then(result => {  
      res.status(201).json({  
        message: 'Thêm thành công bài viết mới!',  
        post: result  
      });  
    })  
    .catch(err => {  
      if (!err.statusCode) {  
        err.statusCode = 500;  
      }  
      next(err);  
    });  
};
```

Các tác vụ Xoá với Mongoose

Thực hiện tương tự với hàm **findByIdAndRemove**

Ví dụ: xoá document của collection posts

```
exports.deletePost = (req, res, next) => {  
  const postId = req.params.postId;  
  Post.findById(postId)  
    .then(post => {  
      if (!post) {  
        const error = new Error('Không tìm thấy bài viết - post.');        error.statusCode = 404;  
        throw error;  
      }  
      return Post.findByIdAndRemove(postId);  
    })  
    .then(result => {  
      console.log(result);  
      res.status(200).json({ message: 'Đã xoá post.' });  
    })  
    .catch(err => {  
      if (!err.statusCode) {  
        err.statusCode = 500;  
      }  
      next(err);  
    });  
};
```


Các tác vụ tìm document với Mongoose

Thực hiện tìm document với hàm **findById**

Ví dụ: tìm document của collection posts

```
exports.getPostById = (req, res, next) => {  
  const postId = req.params.postId;  
  Post.findById(postId)  
    .then(post => {  
      if (!post) {  
        const error = new Error('Không tìm thấy bài viết- post.');        error.statusCode = 404;  
        throw error;  
      }  
      res.status(200).json({ message: 'Post được tìm thấy.', post: post });  
    })  
    .catch(err => {  
      if (!err.statusCode) {  
        err.statusCode = 500;  
      }  
      next(err);  
    });  
};
```

Quan hệ trong mongoose

Ví dụ: Tạo mối quan hệ – liên kết giữa user và bài viết (posts)

```
userId: {  
  type: Schema.Types.ObjectId,  
  ref: 'User',  
  required: true  
}
```

Test với postman

GET Send

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

Query Params

| KEY | VALUE | DESCRIPTION |
|-----|-------|-------------|
| Key | Value | Description |

Body Cookies Headers (9) Test Results ⊕ Status: 200 OK Time: 30 ms Size: 539 B Save

Pretty Raw Preview Visualize JSON ↺

```

1 {
2   "message": "Fetched posts successfully.",
3   "posts": [
4     {
5       "_id": "5fa2e36250dee027f0ee9985",
6       "title": "C#",
7       "content": "Lập trình C#",
8       "create_date": "2020-11-04T17:22:42.872Z"
9     }
10  ]
11 }
```

DELETE Send

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

Query Params

| KEY | VALUE | DESCRIPTION |
|-----|-------|-------------|
| Key | Value | Description |

Body Cookies Headers (9) Test Results ⊕ Status: 200 OK Time: 113 ms Size: 402 B Sa

Pretty Raw Preview Visualize JSON ↺

```

1 {
2   "message": "Đã xoá post."
3 }
```

demo

- ☑ Cài đặt hệ quản trị NoSQL - mongodb
- ☑ Làm việc với mongodb bằng thư mongodb
 - ❖ Cài đặt thư viên mongodb
 - ❖ Truy xuất tới cơ sở dữ liệu mongodb
- ☑ Làm việc với mongodb bằng mongose



thank
you!