express MongoDB

สูตรโกงการใช้ Express + MongoDB

โดย นครโค้ด (NakornCode)

(ห้ามแจกฟรี! สำหรับผู้ซื้อคอร์สเรียนบน SkillLane เท่านั้น)

ติดต่อผู้สอน https://m.me/nakorncode

7 RESTful Routes	
Index	`GET: /users`
New	`GET: /users/new`
Create	`POST: /users`
Show	`GET: /users/1`
Edit	`GET: /users/1/edit`
Update	`PATCH|PUT: /users/1`
Delete	`DELETE: /users/1`

Express - req	
Parameter	`req.params`
URL Query	`req.query`
JSON, Text	`req.body`
Cookie	`req.cookies`
Headers	`req.get('Content-Type')` `req.get('X-Custom-Header')`

Express - app	
ติดตั้ง	`npm i express`
Applica-tion	`const express = require('express')` `const app = express()`

ตัวแปร Global	`// Set`
	`app.locals.title = 'MyWeb'`
	`app.locals.debug = true`
	
	`// Get`
	`res.locals.title // 'MyWeb'`
	`res.locals.debug // true`

Router	`const cb = (req, res) => {}`
	
	`app.all('*', cb)`
	`app.get('/users', cb)`
	`app.post('/users', cb)`
	`app.put('/users/:id', cb)`
	`app.patch('/users/:id', cb)`
	`app.delete('/users/:id, cb)`

Middle-ware	`app.use(express.json())`
	
	`app.use((req, res, next) => {`
	`  // interception`
	`  next()`
	`})`
	
	`app.use((err, req, res, next) => {`
	`  // when error`
	`  next()`
	`})`

การตั้งค่า	`app.set('views', '/path/to/views')`
	`app.set('view engine', 'pug')`
	`// npm install pug`
	
	`app.disable('x-powered-by')`

Listener	`app.listen(3000, () => {`
	`  // app started`
	`})`

Express - res	
ตัวแปร Global	`res.locals[key]`
เช็คส่ง Header	`res.headersSent`
ส่ง Headers	`const type = 'text/plain'` `res.set('Content-Type', type)`
ส่ง Cookies	`res.cookie('name', 'value')`
ล้าง Cookies	`res.clearCookie('name')`
เปลี่ยนส่งทาง	`res.redirect('/some/path')` `res.redirect('http://someurl')`
ส่งไฟล์	`res.download('/path/to/file')`
ส่ง Text	`res.send('foobar')`
ส่ง HTML	`res.send('<h1>foobar</h1>')`
ส่ง JSON	`res.send({ data: 'foobar' })`
ส่ง Buffer	`res.send(Buffer.from('foobar'))`

ส่งตาม Content-Type	`res.format({`
	`  'text/plain': () => {`
	`  res.send('hi')`
	`  },`
	`  'text/html': () => {`
	`  res.send('<h1>hi</h1>')`
	`  },`
	`  'application/json': () => {`
	`  res.send({ message: 'hi' })`
	`  },`
	`  default: () => {`
	`  next('error!')`
	`  }`
	`})`

ส่ง SSR	`res.render('/path/of/html', {`
	`  extraData: 'foobar'`
	`})`

ตั้งสถานะ	`res.status(404)`
ส่งสถานะ	`res.sendStatus(404)`
สิ้นสุดการส่ง	`res.end()`

Express - router	
Router	`const router = express.Router()`

Method	`const path = '/'`
	`const cb = (req, res) => {}`
	
	`router.get(path, cb)`
	`router.post(path, cb)`
	`router.put(path, cb)`
	`router.patch(path, cb)`
	`router.delete(path, cb)`

Param Trigger	`let pm = (req, res, next, id) => {`
	`  // trigger with :id only`
	`}`
	
	`router.param('id', pm)`
	
	`router.get('/:id', cb)`
	`router.post('/:id', cb)`

Middle-ware	`router.use(someMiddleware)` `router.use((req, res, next) => {})`
ใช้กับ App	`app.use(router)` `app.use('/users', router)`

MongoDB - CLI

CLI	`mongo`
ช่วยเหลือ	`help`
แสดง DB	`show databases`

เรียกใช้ DB และจัดการ	`use <db>`
	`show collections`
	`db.help()`

สร้าง Col	`db.createCollection('users')` `db.users.help()`

จัดการ Col	`db.users.find()`
	`db.users.insert()`
	`db.users.deleteOne()`

MongoDB - Node

ติดตั้ง	`npm i mongodb`

Client & Connect	`const { MongoClient } = require('mongodb')`
	
	`let url = 'mongodb://localhost'`
	`let client = new MongoClient(url)`
	`client.connect() // Promise`

DB	`const db = client.db('mydb')`
Collection	`const col = db.collection('users')`
Operation	`col.find({ name: 'john' })` `// Promise`

Projection Operators

ตัวอย่าง	`db.users.find({}, { posts: { $slice: 5 } })`
ตรงกัน	`$elemMatch: { <query> }`
ตัดจำนวน	`$slice: <end>` `$slice: [<start>, <end>]`
คะแนนตรงกัน	`$meta: "textScore"`

Query Operators

หมายเหตุ	`<exp> Expression`
ตัวอย่าง	`db.users.find({ posts: { $exists: 1 }, created: { $gte: "2022-01-01" } })`
เท่ากัน	`$eq: <value>`
ไม่เท่ากัน	`$ne: <value>`
น้อยกว่า	`$lt: <value>` `$lte: <value> // น้อยกว่าหรือเท่ากับ`
มากกว่า	`$gt: <value>` `$gte: <value> // มากกว่าหรือเท่ากับ`
มีฟิลด์อยู่	`$exists: <boolean>`
มีข้อมูลตรง	`$type: <type>`
มีใน Arr	`$in: [<value>]`
ไม่มีใน Arr	`$nin: [<value>]`
ตรงกัน Arr	`$all: [<value>]` `$elemMatch: { <query> }`
ขนาด Arr	`$size: <number>`
และ	`$and: [<exp>]`
หรือ	`$or: [<exp>]`
ไม่	`$not: <exp>`
ไม่ทั้งหมด	`$nor: [<exp>]`
Expression	`$expr: <exp>`
JavaScript	`$where: <jsCallbackFunction>`
JSON	`$jsonSchema: <json>`
Regex	`$regex: /pattern/`
ค้นหาข้อความ	`$text: { $search: <string> }`

Constructor	
วันเวลา	`new Date() // now` `new Date('2022-01-30T12:30:55')`
ObjectID	`ObjectID() // generate id` `ObjectID("objid here")`
UUID	`UUID() // generate uuid` `UUID("uuid here")`

Aggregation Pipeline Stages (1)	

ตัวอย่าง	`const stage1 = {`
	`  $project: { name: 1, pass: 0 },`
	`}`
	`const stage2 = {`
	`  $match: { name: "John" }`
	`}`
	`db.users.aggregate([`
	`  stage1,`
	`  stage2`
	`])`

หมายเหตุ	<field> ชื่อฟิลด์ที่มี <newFields> ตั้งฟิลด์ใหม่หรือฟิลด์เดิม <acc> Accumulator <order> 1 หรือ -1
แสดงข้อมูล	`$project: { <field>: <bool|exp> }`
ตรงกัน	`$match: { <field>: <exp> }`
เพิ่มข้อมูล	`$addFields: { <newField>: <exp> }`
นับข้อมูล	`$count: <newField>`
จำกัดชุด	`$limit: <number>`
ข้ามชุด	`$skip: <number>`
สุ่มเลือก	`$sample: <number>`
จัดเรียง	`$sort: { <field>: <order> }`
แก้ไขข้อมูล	`$set: { <newField>: <exp> }`
แทนข้อมูล	`$replaceRoot: { newRoot: <exp> }`
ลบข้อมูล	`$unset: <field>` `$unset: [<field>, <field>]`

Aggregation Pipeline Stages (2)	
แยกกลุ่ม	`$unwind: <exp>`

รวมกลุ่ม	`$group: {`
	`  _id: <exp>,`
	`  <field>: { <acc>: <exp> }`
	`}`

กลุ่มตามเงื่อนไข	`$bucket: {`
	`  groupBy: <exp>,`
	`  boundaries: [<num>, <num>],`
	`  default: <string>,`
	`  output: { <newField>: <acc|exp> }`
	`}`

ความสัมพันธ์	`$lookup: {`
	`  from: <collection>,`
	`  localField: <field>,`
	`  foreignField: <field on from>,`
	`  as: <newField>`
	`}`

ความสัมพันธ์หลายขั้น	`$graphLookup: {`
	`  from: <collection>,`
	`  startWith: <exp>,`
	`  connectFromField: <string>,`
	`  connectToField: <string>,`
	`  as: <string>`
	`}`

Union	`$unionWith: {`
	`  coll: <collection>,`
	`  pipeline: [<stage>, <stage>]`
	`}`

รวมไปยัง Collection	`$merge: {`
	`  into: <collection>,`
	`  on: <newField>`
	`}`

Sub-query	`$facet: {`
	`  <newField>: [<stage>, <stage>]`
	`}`

Accumulator Operators	

ตัวอย่าง	db.users.aggregate([
	  {
	  $group: {
	  _id: "$role",
	  count: { $count: {} }
	  }
	  }
	])

นับจำนวน	$count: {}
ค่าน้อยสุด	$min: <exp>
ค่ามากสุด	$max: <exp>
ค่าเฉลี่ย	$avg: <exp>
รวมผล	$sum: <exp>
ตัวแรก	$first: <exp>
ตัวท้าย	$last: <exp>
เพิ่ม Array	$push: <exp> $addToSet: <exp> // เพิ่มเฉพาะไม่ซ้ำกัน
รวม Object	$mergeObject: <document>

Expression Operators (1)	

ตัวอย่าง	db.users.aggregate([{
	  $project: {
	  balanceRounded: {
	  $round: ["$balance", 2]
	  }
	  }
	}])

Reference	$field $$variable
คำนวณคณิต	$add: ["$field"] $subtract: ["$field"] $multiply: ["$field"] $divide: ["$field"]

Expression Operators (2)	
ตัดทศนิยม	$round: ["$field", <place>] $ceil: "$field" $floor: "$field" $trunc: ["$field", <place>]
Absolute	$abs: "$field"
รวม Str	$concat: [<exp>, <exp>]
แบ่ง Str	$split: ["$field", <delimiter>]
ตัด Str	$substr: ["$field", <start>, <length>]

Trim	$trim: {
	  input: "$field"
	}

แทนที่ Str	$replaceOne: {
	  input: "$field",
	  find: <exp>,
	  replacement: <exp>
	} // ใช้เหมือนกับตัวอื่น
	$replaceAll: {}

Regex	$regexFind: {
	  input: "$field",
	  regex: /pattern/
	} // ใช้เหมือนกันกับตัวอื่น
	$regexFindAll: {}
	$regexMatch: {}

ขนาด Arr	$size: "$field"
ชิ้นแรก Arr	$first: <exp>
ชิ้นท้าย Arr	$last: <exp>
ตัด Arr	$slice: ["$field", <start>, <end>]
เป็น Arr	$isArray: ["$field"]
ตำแหน่ง Arr	$arrayElemAt: ["$field", <index>]
ค้นหา Arr	$indexOfArray: ["$field", <exp>]
รวม Arr	$concatArrays: ["$field"]

Expression Operators (3)	

กรองและแปลง Arr	`$filter: {`
	`  input: "$field",`
	`  as: <string>,`
	`  cond: <exp>`
	`} // ใช้เหมือนกันกับ $map`
	`$map: {}`

ลด Arr	`$reduce: {`
	`  input: "$field",`
	`  initialValue: <exp>,`
	`  in: <exp>`
	`}`

ย้อน Arr	`$reverseArray: "$field"`

Unit	`year  quarter  month`
	`week  day  hour`
	`minute  second  millisecond`

เพิ่มและลดวันเวลา	`$dateAdd: {`
	`  startDate: <exp>,`
	`  unit: <unit>,`
	`  amount: <exp>`
	`} // ใช้เหมือนกันกับ $dateSubtract`
	`$dateSubtract: {}`

เปรียบเทียบวันเวลา	`$dateDiff: {`
	`  startDate: <exp>,`
	`  endDate: <exp>,`
	`  unit: <unit>`
	`}`

แบ่งสัดส่วนวันเวลา	`$dateToParts: {`
	`  date: <exp>`
	`}`

เรียกตามสัดส่วนวันเวลา	`$dayOfYear: <exp>`
	`$dayOfMonth: <exp>`
	`$dayOfWeek: <exp>`
	`$year: <exp>`
	`$month: <exp>`
	`$week: <exp>`
	`$hour: <exp>`
	`$minute: <exp>`
	`$second: <exp>`
	`$millisecond: <exp>`

Schema Validation	

ตัวอย่างโครงสร้าง	`$jsonSchema: {`
	`  type: "object",`
	`  required: [`
	`  "name",`
	`  "age"`
	`  ],`
	`  properties: {`
	`  name: {`
	`  type: "string",`
	`  description: "A name",`
	`  minLength: 3`
	`  },`
	`  age: {`
	`  bsonType: "int",`
	`  minimum: 18`
	`  },`
	`  role: {`
	`  type: "string",`
	`  enum: ["admin", "user"]`
	`  },`
	`  profileInfo: {`
	`  type: "array",`
	`  items: {`
	`  anyOf: [`
	`  { type: "string" },`
	`  {`
	`  type: "object",`
	`  required: ["title"]`
	`  properties: {`
	`  title: {`
	`  type: "string",`
	`  maxLength: 100`
	`  },`
	`  body: {`
	`  type: "string"`
	`  },`
	`  tags: {`
	`  type: "array",`
	`  uniqueItems: true`
	`  }`
	`  }`
	`  }`
	`  ]`
	`  }`
	`  }`
	`  }`
	`}`

Mongoose - Connect

ติดตั้ง

```
npm i mongoose
```

เชื่อมต่อฐานข้อมูล

```
let mongoose = require('mongoose')
let url = 'mongodb://localhost'
mongoose.connect(url) // Promise
```

Mongoose - Schema (1)

ตัวอย่าง Schema

```
let schema = new mongoose.Schema({
  name: {
    type: String,
    required: true
  },
  age: {
    type: Number,
    min: 18
  },
  birthDate: Date,
  roles: [String],
  settings: {
    getNotification: Boolean,
    timezone: {
      type: String,
      default: 'Asia/Bangkok'
    }
  }
})

module.exports = mongoose.model(
  'Users', schema
)
```

API

```
mongoose.Schema(schema, options)
```

Options

```
{
  id: <boolean>,
  _id: <boolean>,
  timestamps: <boolean>,
  strict: <boolean>,
  versionKey: <boolean>,
  collection: <string>
}
```

Mongoose - Schema (2)

Type

```
String    Number    Date
Boolean   Buffer    Array
Array     ObjectId  Schema
```

Schema

```
<name>: {
  type: <type>,
  required: <boolean>,
  default: <value>,
  select: <boolean>,
  index: <boolean>,
  unique: <boolean>,
  validate: {
    validator: <callback>,
    message: <string|callback>
  },
  transform: <callback>,
  // string
  trim: <boolean>,
  match: /pattern/,
  enum: [<value1>, <value2>],
  minLength: <number>,
  maxLength: <number>,
  // number, date
  min: <number>,
  max: <number>,
}
```

ข้อความ Error

```
<name>: {
  required: [true, 'Need this']
}
```

Instance Methods

```
schema.methods.fn = function () {
  // .fn ตั้งชื่อตรงนี้
  // ต้องใช้ function () {} เพื่อยังคง
  // bind this ให้ถูกต้อง
}
```

Static Methods

```
schema.statics.fn = function () {
  // .fn ตั้งชื่อตรงนี้
}
```

Query Helpers

```
schema.query.q = function () {
  // .q ตั้งชื่อตรงนี้
}
```

Virtuals

```
schema
  .virtual('name')
  .get(function () { // หรือ .set
    // 'name' ตั้งชื่อตรงนี้
  })
```

Mongoose - Model

เพิ่ม	`let User = require('./User.js')`
	`User.create(object)`
	`User.insertMany([obj1, obj2])`
	`// ส่วนมาก Statics return Promise`

เรียกและจัดการ	`User.find(query[, project, opts])`
	`User.findOne()`
	`User.findById(objectId)`
	`User.fineOneAndDelete()`
	`User.fineOneAndUpdate()`
	`User.findByIdAndDelete()`
	`User.findByIdAndUpdate()`

นับจำนวน	`User.countDocuments(query)`
แก้ไข	`User.updateOne(query, toNewData)` `User.updateMany()`
ลบ	`User.deleteOne(query)` `User.deleteMany()`
Aggregate	`User.aggregate([stage])`
Query	`User.find({ age: { $gte: 18 })` `User.where('age').gte(18)`

Mongoose - Document

ตัวอย่าง	`let q = { name: 'John' }` `let user = await User.findOne(q)`
ID	`q._id`
แก้ไข	`q.name = 'Joe'` `await q.save()`
ลบ	`q.age = undefined` `await q.save()`
แปลง JSON	`q.toJSON()`

Mongoose - Populate

ตัวอย่าง	`let userSchema = new Schema({`
	`  name: String,`
	`  posts: [{`
	`  type: Schema.Types.ObjectId,`
	`  ref: 'Post'`
	`  }]`
	`})`
	
	`let postSchema = new Schema({`
	`  title: String,`
	`  body: String,`
	`  user: {`
	`  type: Schema.Types.ObjectId,`
	`  ref: 'User'`
	`  }`
	`})`
	
	`let u = model('User', userSchema)`
	`let p = model('Post', postSchema)`
	
	`u.findOne({ name: 'John' })`
	`  .populate('posts')`
	`  .then(user => {`
	`  user.posts // [post1, post2]`
	`  })`

Projection	`User.findOne(query)`
	`  .populate({`
	`  path: 'posts',`
	`  select: '-_id' // นำ _id ออก`
	`  })`

Multiple	`User.fineOne(query)` `  .populate(['posts', 'comments'])`

Dynamic Reference	`new Schema({`
	`  on: {`
	`  type: Schema.Types.ObjectId,`
	`  refPath: 'model'`
	`  },`
	`  model: String`
	`})`

Virtual	`userSchema.virtual('posts', {`
	`  ref: 'Post',`
	`  localField: '_id',`
	`  foreignField: 'user'`
	`})`

Mongoose - Middleware

ตัวอย่าง
```
schema.pre('save', function (cb) {
  cb() // ต้องเรียกทุกครั้ง
})
```

ทำก่อน
```
.pre(event, cb)
```

ทำหลัง
```
.post(event, cb)
```

จัดการ Error
```
.post('save', function(e, d, cb) {
  // e = error
  // d = document
})
```

pug (1)

ติดตั้ง
```
npm i pug
```

Render
```
let pug = require('pug')
let path = 'index.pug'
let opt = { extra: 'foobar' }
pug.renderFile(path, opt)
```

Express
```
app.set('view engine', 'pug')
```

ตัวอย่าง
```
.content
  h1.title Text
  input(type="text" value="abc")
```

Interpola-tion
```
- let msg = 'Hi!'
p.alert #{msg}
p.alert= msg
input(value=msg)
```

Plain Text
```
p
  | Hello
  | world!
script.
  if (ok) {
    alert('OK!')
  }
```

if-else
```
if user.role == 'admin'
  p.alert-success Ok!
else
  p.alert-error Error!
```

Iteration
```
ul
  each val in [1, 2, 3]
    li= val
```

pug (2)

Includes
```
include path/to/html.pug
```

Layouts
```
//- layout.pug
html
  head
    link(href="bootstrap.css")
  body
    .container
      block content
```

Extends
```
extends path/to/layout.pug

block content
  h1 Hello world
```

Mixin
```
minix check(text, status=false)
  label= text
  input(
    type="checkbox"
    checked=status)

+check('Accept')
+check('Remember me', true)
```

morgan

ติดตั้ง
```
npm i morgan
```

Express
```
app.use(morgan('combined'))
```

redis, connect-redis

ติดตั้ง
```
npm i redis connect-redis
```

Express
```
let RedisStore =
require('connect-redis')(session)

let redis = require('redis')
let client = redis.createClient()

app.use(session({
  store: new RedisStore({ client })
}))
```

multer

ติดตั้ง

```
npm i multer
npm i bytes # ช่วยกำหนดขนาดไฟล์
```

Form

```
<form enctype="multipart/form-data">
```

Express

```
let multer = require('multer')
let bytes = require('bytes')
let fileFilter = (req, f, cb) => {
  // f = file api
  cb(null, false) // reject
  cb(null, true)  // resolve
  cb(new Error()) // error
}
let limits = {
  fileSize: bytes('1MB')
}
let opt = {
  dest: 'path/to/dir',
  fileFilter
  limits
}
let upload = multer(opt)

let upPic
// 1 ไฟล์
upPic = upload.single('pic')
// 8 ไฟล์
upPic = upload.array('pic', 8)
// แยกไฟล์
upPic = upload.fields([
  { name: 'pic', maxCount: 8 },
  { name: 'avatar', maxCount: 1 }
])
app.post('/', upPic, requestCb)
```

File API

```
// req.file หรือ req.files
{
  fieldname,
  originalname,
  encoding,
  mimetype,
  size,
  destination,
  filename,
  path
}
```

cookie-parser

ติดตั้ง

```
npm i cookie-parser
```

Express

```
let cookieParser =
require('cookie-parser')
app.use(cookieParser())
```

express-session

ติดตั้ง

```
npm i express-session
npm i ms # ช่วยกำหนดเวลา
```

Express

```
let ms = require('ms')
let session =
require('express-session')

let opt = {
  secret: 'someRandomString',
  cookie: {
    maxAge: ms('7d'),
    secure: 'auto'
  },
  resave: false,
  saveUninitialized: false,
  rolling: true
}
app.use(session(opt))
```

Set

```
req.session.user = 'John'
```

Get

```
req.session.user // 'John'
```

Delete

```
delete req.session.user
req.session.destroy() // ลบหมด
```

method-override

ติดตั้ง

```
npm i method-override
```

Form

```
<form
action="path/to/url?_method=PUT"
method="POST">
```

Express

```
const methodOverride =
require('method-override')

app.use(methodOverride('_method'))
```

connect-flash

ติดตั้ง

```
npm i connect-flash
```

Express

```
let flash = require('connect-flash')
app.use(flash())

app.get('/flash', (req, res) => {
  req.flash('info', 'Message') //set
  res.redirect('/')
})

app.get('/', (req, res) => {
  res.render('index', {
    info: req.flash('info') // get
  })
})
```

faker

ติดตั้ง

```
npm i @faker-js/faker
# ระวัง Malware จาก npm i faker
```

Random

```
const faker =
require('@faker-js/faker')

faker.internet.userName()
```

joi

ติดตั้ง

```
npm i joi
```

Schema

```
const Joi = require('joi')
const schema = Joi.object({
  username: Joi.string()
    .alphanum()
    .min(3)
    .max(20)
    .required(),
  email: Joi.string().email(),
  age: Joi.number()
    .integer()
    .greater(18),
  birthDate: Joi.date()
    .less('now')
})

await schema.validateAsync(data)
```

bcrypt

ติดตั้ง

```
npm i bcrypt
```

เข้ารหัส

```
const bcrypt = require('bcrypt')

const pass = '1234'
const hash = await bcrypt
.hash(pass, 10) // gen hash
```

ถอดรหัส

```
const result = await bcrypt
.compare('abc', hash) // true|false
```

passport (1)

ติดตั้ง

```
npm i passport passport-local
```

Strategy

```
let passport = require('passport')
let LocalStrategy =
require('passport-local')

let options = {
  usernameField: 'email',
  passwordField: 'pass',
  passReqToCallback: false,
  session: true
}

let cb = (user, pass, next) => {
  next(null, {}) // resolve
  next(null, false, msg) // reject
  next(new Error()) // exception
}
passport.use(new LocalStrategy(
  options,
  cb
))
```

Serialize

```
passport.serializeUser((user, next)
=> {
  next(null, { id: 1234 })
})

passport.deserializeUser((id, next)
=> {
  User.findById(id).then(user => {
    next(null, user)
  }).catch(next)
})
```

passport (2)

Session

```
app.use(passport.session())
```

Authen-ticate

```
app.post('/login',
passport.authenticate('local', {
  successRedirect: '/',
  failureRedirect: '/login'
}))
```

User

```
req.user
```

Login

```
req.login(user)
```

Logout

```
req.logout()
```

Custom Callback

```
app.post('/login', (req, res, next)
=> {
  let cb = (err, user, info) => {
    // มาจาก next(err, user, info)
    // ที่ passport.use(...)
    req.login(user) // ควรเรียกใช้
  }
  passport.authenticate(
    'local',
     cb
  )(req, res, next)
})
```

nodemailer

ติดตั้ง

```
npm i nodemailer
```

Transport

```
const nodemailer =
require('nodemailer')

const auth = { user, pass }
const opt = {
  host, port, secure, auth
}

const transporter =
nodemailer.createtransport(opt)
```

ส่งเมล

```
await transporter.sendMail({
  from, to, subject, html
})
```

bull

ติดตั้ง

```
npm i bull
```

เข้าคิว

```
const Queue = require('bull')
const queue = new Queue()
queue.process((job, done) => {
  job.data // ที่ส่ง Args
  done()
})
queue.add({ video: 'a.mp4' })
queue.add({ video: 'b.mp4' })
```

Cron

```
queue.add({ url: 'path/fo/url' },
{ repeat: { cron: '0 0 * * *' } })
```

socket.io

ติดตั้ง

```
npm i socket.io
```

Server

```
const express = require('express')
const http = require('http')
const { Server } =
require('socket.io')

const app = express()
const httpServer = createServer(app)
const io = new Server(httpServer)

io.on('connection', (socket) => {
  socket.emit('msg', 'Hi client!')
  socket.on('msg', (m) => {
    console.log(m) // Hi server!
  })
})

httpServer.listen(3000)
```

Client

```
<script src="socket.io.min.js">

<script>
let socket = io('http://localhost')
socket.on('msg', (m) => {
  alert(m) // Hi client!
})
socket.emit('msg', 'Hi server!')
</script>
```

jsonwebtoken	
ติดตั้ง	`npm i jsonwebtoken`

Sign	`const jwt = require('jsonwebtoken')`
	`let opt = {`
	`  expiresIn: '1h' // 1 hour`
	`}`
	`let token = jwt.sign(userData, secretKey, opt)`

Verify	`jwt.verify(token, secretKey)` `// throw ถ้าไม่ถูกต้อง`
Decode	`jwt.decode(token) // ได้ userData`

compression	
ติดตั้ง	`npm i compression`
Express	`app.use(require('compression)())`

helmet	
ติดตั้ง	`npm i helmet`
Express	`app.use(require('helmet')())`

cors	
ติดตั้ง	`npm i cors`
Express	`app.use(require('cors')())`

xss	
ติดตั้ง	`npm i xss`

ป้องกัน XSS	`const xss = require('xss')`
	`const script = '<script>...'`
	`const html = xss(script) // secure`

csurf	
ติดตั้ง	`npm i csurf`

Get & Post CSRF	`const csrf = require('csurf')()`
	
	`const getLogin = (req, res) => {`
	`  res.render('login', {`
	`  csrf: req.csrfToken()`
	`  })`
	`}`
	`app.get('/login', csrf, getLogin)`
	
	`const postLogin = (req, res) => {`
	`  res.send(req.body)`
	`}`
	`app.post('/login', csrf, postLogin)`

jest (1)	
ติดตั้ง	`npm i -D jest`
Initial	`npx jest init`

Matcher	`// *.test.js, *.spec.js`
	`test('one plus one is two', () => {`
	`  expect(1 + 1).toBe(2)`
	`})`

Async	`test('callback', (done) => {`
	`  withCallbackFn(done)`
	`})`
	`test('promise', () => {`
	`  return promiseFn()`
	`})`
	`test('async', async () => {`
	`  await asyncFn()`
	`})`

Setup, Teardown	`beforeEach(() => {})`
	`beforeAll(() => {})`
	`afterEach(() => {})`
	`afterAll(() => {})`

Mock	`jest.mock('./file.js', () => {`
	`  return 'change behavior'`
	`})`

jest (2)

Expect

```
expect(x).toBe(y) // ==
expect(x).not.toBe(y) // !=
expect(x).toEqual(y) // เทียบ obj ได้
expect(x).toBeLessThan(y) // <
expect(x).toBeLessThanOrEqual(y) // <=
expect(x).toBeGreaterThan(y) // >
expect(x).toBeGreaterThanOrEqual(y) // >=
expect(x).toBeTruthy() // ขอแค่มี
expect(x).toBeFalsy() // ขอแค่ไม่มี
expect(x).toContain(y) // arr มี y
expect(x).toHaveLength(y) // .length
expect(x).toHaveProperty(key, val?)// obj
expect(x).toMatch(y) // str ตรงกับ y
expect(x).toThrow(y?)
expect(x).toBeInstanceOf(y)
expect(x).toBeNull()
expect(x).toBeDefine()
expect(x).toBeUndefined()
expect(x).toBeNaN()
// function return and args
expect(x).toHaveReturned()
expect(x).toHaveReturnedWith(y)
expect(x).toHaveBeenCalled()
expect(x).toHaveBeenCalledWith(...y)
// any
expect(x)
.toHaveBeenCalledWith(expect.any(string))
// array contains
expect(x)
.toEqual(expect.arrayContaining([y, z]))
```

supertest

ติดตั้ง

```
npm i -D supertest
```

Request

```
let app = express()
let request = require('supertest')
const res = await request(app)
  .post('/login')
  .send({ user: 'j', pass: '12' })
  .expect(200)
expect(res.body.message)
  .toBe('Logged in!')
```

แนบ File

```
test('should be upload', () => {
  request(app)
    .post('/avatar')
    .attach('img', 'avatar.jpg')
    .expect(201, done)
})
```

cypress

ติดตั้ง

```
npm i -D cypress
```

Initial

```
npx cypress
```

Test

```
test('Try login', () => {
  cy.visit('/login')
  cy.get('#user').type('john')
  cy.get('#pass').type('1234')
  cy.contains('Login').click()

  cy.url()
    .should('include', '/profile')
  cy.get('h1')
    .should('contain', 'john')
})
```

API

```
.click()          .dblclick()
.rightclick()     .type()
.clear()          .check()
.uncheck()        .select()
.trigger()        .selectFile()
```

pm2	
ติดตั้ง	`npm i -g pm2`
สร้าง Config	`pm2 init`
สำหรับ Linux	`pm2 startup`

เริ่มรัน	`pm2 start # ต้องมี pm2 init`
	`pm2 start dev.js --watch`
	`pm2 start p.js --name prod`
	`pm2 start max.js -i max`

รันแบบ Cronjob ทุกๆเที่ยงคืน และห้ามรันใหม่ทันที	`pm2 start file.js`
	`--cron "0 0 * * *"`
	`--no-autorestart`

เริ่มใหม่	`pm2 restart app_name` `pm2 restart all # เริ่มทั้งหมด`
ปิดการทำงาน	`pm2 stop app_name` `pm2 stop all # หยุดทั้งหมด`
ลบ	`pm2 delete app_name` `pm2 delete all # ลบทั้งหมด`
ดูรายการรัน	`pm2 list` `pm2 monit # ละเอียดกว่า`
ดูสถานะโดยละเอียด	`pm2 show app_name`
ดู Log จากการรัน	`pm2 logs` `pm2 logs app_name`
บันทึกเพื่อ Startup	`pm2 save`