คำแนะนำทุกคนดีหมด ลองทำดูได้เลย ถ้าไม่ไหวก็กลับมา
ลองรู้เยอะดูก่อน ถ้าไม่ไหว ลองรู้น้อย

มีอาชีพจะใช้ Postman ซึ่งมี Cookies ในตัว
แต่ผมจะเขียน Code เองเลย ใช้คำสั่ง fetch()

```
GET     /api/me
POST    /api/login
GET     /api/logout
GET     /api/list-products
```

จะเปิด Web Browser เข้าไปที่ /api/login ไม่ได้
เมื่อวานไปที่นี่ /test.html
Login 1, Log In 2, Check Me, Log Out, List Products

https://codestar.work/full-stack-2023-09-19.txt

```
web-20
'-- main.js
'-- public
    '-- test.html
```

File: public/test.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>iCoffee</title>
  </head>
  <body>
    <button onclick="testLogIn1()">Log In 1</button>
    <button onclick="testLogIn2()">Log In 2</button>
    <button onclick="checkMe()">Check Me</button>
    <button onclick="checkPrice()">Check Price</button>
    <button onclick="testLogOut()">Log Out</button>
    <script>
      async function testLogOut() {
        var response = await fetch("/api/logout")
        var result   = await response.json()
      }
      async function checkPrice() {
        var response = await fetch("/api/list-products")
        var result   = await response.json()
        // console.log(result)
        console.table(result)
      }
    </script>

    <script>
      async function checkMe() {
        var response = await fetch("/api/me")
        var result = await response.json()
        console.log(result)
        if (result.status == "OK") {
          alert("Hello " + result.name)
        } else {
          alert(result.status)
        }
      }
      async function testLogIn1() {
        var detail = { }
        detail.email = "jeff@amazon.com"
        detail.password = "xxxxx"
        var payload = { }
        payload.method = "POST"
        payload.headers = { }
        payload.headers["Content-Type"] = "application/json"
        payload.body = JSON.stringify(detail)
        var response = await fetch("/api/login", payload)
        var result = await response.json()
        alert(result.status)
      }
      async function testLogIn2() {
        var detail = { }
        detail.email = "jeff@amazon.com"
        detail.password = "Jeff123"
        var payload = { }
        payload.method = "POST"
        payload.headers = { }
        payload.headers["Content-Type"] = "application/json"
        payload.body = JSON.stringify(detail)
        var response = await fetch("/api/login", payload)
        var result = await response.json()
        alert(result.status)
      }
    </script>
  </body>
</html>
```

File: main.js

```
var crypto  = require("crypto")
var express = require("express")
var server  = express()
server.listen(19000)
var readJson = express.json()
var sessions = [ ]
var cookie   = require("cookie-parser")
var readCookie = cookie()  // similar to express.js

server.use (readCookie)    // work on all requests
server.get ("/api/me",            getCurrentUserDetail)
server.post("/api/login",         readJson, tryLogIn)
server.get ("/api/logout",        tryLogOut)
server.get ("/api/list-products", listProducts)
server.use (express.static("public"))

function getCurrentUserDetail(request, response) {
  // Middleware จะทำงานยังไง ขึ้นอยู่คนเขียน สำหรับ cookie-parser จะทำงานโดย
  // อ่านข้อมูลจาก HTTP Header และ เก็บไว้ที่ request.cookies.token
  var t = request.cookies.token
  if (sessions[t] == null) {
    response.send({status: "Error - Invalid current user"})
  } else {
    var result = { }
    result.status = "OK"
    result.name   = sessions[t].name
    result.email  = sessions[t].email
    response.send(result)
  }
}

function tryLogIn(request, response) {
  // Middleware จะทำงานยังไง ขึ้นอยู่กับคนเขียน คนเขียน express.json() คิดไว้แล้วว่า
  // จะอ่านข้อมูลเป็น JSON และเก็บไว้ที่นี่ request.body.email
  var email    = request.body.email    || ""
  var password = request.body.password || ""
  if (email == "" || password == "") {
    response.send({status: "Error - Invalid user detail"})
    return
  }

  password = encrypt(password)
  var found = -1
  for (var i = 0; i < users.length; i++) {
    if (users[i].email == email &&
        users[i].password == password) {
        found = i
    }
  }

  if (found == -1) {
    response.send({status: "Error - Fail to log in"})
    return
  }
  // 1. create token e.g. FD87-837E-DD75-DZ38
  // 2. store the token an our array
  // 3. send the token to client via HTTP cookie
  var t = randomToken()                          // 1
  sessions[t] = users[found]                     // 2
  response.header("Set-Cookie", "token=" + t)    // 3
  response.send({status: "OK"})
}

function tryLogOut(request, response) {
  var t = request.cookies.token
  delete sessions[t]
  response.send({status: "OK"})
}

function listProducts(request, response) {
  var t = request.cookies.token
  if (sessions[t] == null) {
    response.send(products)
  } else {
    var items = [ ]
    for (var i = 0; i < products.length; i++) {
      var item   = { }
      item.name  = products[i].name
      item.price = products[i].price * 0.90
      items.push(item)
    }
    response.send(items)
  }
}

var products = [ {name: "Latte", price: 80},
                 {name: "Mocha", price: 90},
                 {name: "Cappuccino", price: 80} ]

// https://codestar.work/full-stack-2023-09-20.txt

var users = [ ]
users.push( {email:"jeff@amazon.com",
             password: encrypt("Jeff123"),
             name:"Jeff Bezos"} )
users.push( {email:"sheryl@amazon.com",
             password: encrypt("Sheryl123"),
             name:"Sheryl Sandberg"} )

function encrypt(text) {
  return crypto.createHash("sha512")
               .update(text)
               .digest("hex")
}

function randomToken() {
  var width = 4
  var block = 8
  // var pattern = "0123456789"
  var pattern = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  var items = [ ]
  for (var i = 0; i < block; i++) {
    var buffer = ""
    for (var j = 0; j < width; j++) {
      var r = parseInt(Math.random() * pattern.length)
      buffer = buffer + pattern[r]
    }
    items.push(buffer)
  }
  // items is [ "xxxx", "yyyy", "xxxx", "yyyy" ]
  return items.join("-")
}
```

```
https://codestar.work/full-stack-2023-09-20.txt
mkdir web-20
cd web-20
npm install express cookie-parser
nano main.js
mkdir public
nano public/test.html
killall node
node main.js
```

https://codestar.work/full-stack-2023-09-20.txt
https://javapuzzle.com/mongodb.html

```
Operating System            Database Management System
OS                          DBMS
Windows, Linux, ...         MySQL, SQL Server, MongoDB
File                        Database
'                           '
'---------------------------'

MySQL:    show databases;
          show tables;

MongoDB:  show dbs
          show collections
          show tables
```

```
rm *.deb

wget https://repo.mongodb.org\
/apt/ubuntu/dists/bionic/mongodb-org/4.0\
/multiverse/binary-amd64\
/mongodb-org-server_4.0.8_amd64.deb

apt install ./mongo[Tab]
service mongod status
service mongod start
service mongod status

rm *.deb

wget https://downloads.mongodb.com\
/compass/mongodb-mongosh_1.0.0_amd64.deb

apt install ./mongo[Tab]

rm *.deb

mongosh

wget https://codestar.work/libssl.deb
apt install ./libssl.deb
```

```
test> use shop [Enter]
switched to db shop
จากจุดนี้ไป ถ้าใช้คำว่า db จะหมายถึง shop
shop> db.branch.insertOne( { name: "New York", area: 87.5 } )
         '
         '
         '
         '-------> เดิมที MongoDB เรียกว่า Collection
                   นี่คือ Collection ชื่อ branch
                   แต่ต่อมาก็เปลี่ยนใจ เรียกว่า Table ก็ได้

                   show collections
                   show tables

shop> db.branch.insertOne( { name: "Atlanta", area: 120 } )

shop> db.staff.insertOne( { name: "Harry P", salary: 48000, gender: "Male" } )

shop> db.staff.insertOne( { name: "Luna L", wage: 400 } )

shop> db.staff.find()

shop> db.staff.deleteOne( {name: "Luna L"} )

shop> db.staff.updateOne( {name: "Luna L"}, { $set: {gender: "Female"} })

shop> db.staff.updateOne( {name: "Luna L"}, { $set: {wage: 450} })
```

```
                            mkdir web-20b
web-20b                     cd web-20b
'-- show.js                 npm install mongodb express
'-- service.js
```

File: show.js

```
var mongo = require("mongodb")
var source = "mongodb://127.0.0.1"
show()

async function show() {
  var cn = await mongo.MongoClient.connect(source)
  var data = await cn.db("shop")
                     .collection("staff")
                     .find()
                     .toArray()
  await cn.close()
  console.log(data)
}
```

File: service.js

```
var express = require("express")
var server  = express()
server.listen(20000)
var mongo   = require("mongodb")
var source  = "mongodb://127.0.0.1"
server.get("/api/list-branch", listBranch)
server.get("/api/list-staff",  listStaff)

async function listBranch(request, response) {
  var cn = await mongo.MongoClient.connect(source)
  var data = await cn.db("shop")
                     .collection("branch")
                     .find()
                     .toArray()
  await cn.close()
  response.send(data)
}

async function listStaff(request, response) {
  var cn = await mongo.MongoClient.connect(source)
  var data = await cn.db("shop")
                     .collection("staff")
                     .find()
                     .toArray()
  await cn.close()
  response.send(data)
}
```

https://telehealth.bnhhospital.com/setup.html

ระดับโลกจะใช้ GET ในการ List ข้อมูล หรือ Search หาข้อมูล

CRUD: Create, Read, Update, Delete

Read - Search - List ---> GET

องค์กรชั้นนำของไทย ไม่ได้ใช้ GET
Verb ไหน ก็ไม่ถูก
การใช้ HTTP ทำ Web Service มันผิดตั้งแต่แรกแล้ว

```
RESTful Web Service
'
'---------> S ย่อมาจาก State
```

RESTful Web Service คือการมี Operation บนข้อมูลแต่ละ Entity

```
/api/add-branch        /api/add-user
/api/update-branch     /api/update-user
/api/delete-branch     /api/delete-user
/api/list-branch       .
/api/search-branch     .
```

https://javapuzzle.com/mongodb.html