

# Product API

อันดับแรก ไฟล์ `.env` กำหนด URL เพื่อเชื่อมต่อ MongoDB และ Port ที่เชื่อมต่อ

```
.env x
03_product_api > .env
1 MONGODB_URL=mongodb+srv://adminDB:kn123456@nosql.abavc.mongodb.net/Store?retryWrites=true&w=majority
2 PORT=5000S|
```

ต่อมาไฟล์

`product.js` ในโฟลเดอร์ `model`

```
JS product.js x
03_product_api > models > JS product.js > [?] default
1 import mongoose from "mongoose";
2 const Schema = mongoose.Schema;
3 const productSchema = new Schema({ //กำหนดโครงสร้าง
4   name: String,
5   category: String,
6   price: Number,
7   tags: [String],
8 });
9 const ProductModel = mongoose.model("Product", productSchema); //กำหนด ProductModel = productSchema
10 export default ProductModel;
```

จะ `import mongoose` เพื่อเรียกใช้โครงสร้าง

กำหนด `Schema`

ต่อมากำหนด ตัวแปร เพื่อกำหนด `Schema` หรือ โครงสร้างของข้อมูล

กำหนด `ProductModel = productSchema` ที่กำหนดไว้ก่อนหน้านี้

และทำการส่งออกเพื่อเรียกใช้ที่ไฟล์อื่น

ต่อมา ไฟล์ **product.js** ในโฟลเดอร์ **routes**

```
JS product.js X
03_product_api > routes > JS product.js > [0] default
1  import express from "express";
2  import Product from "../models/product.js";
3  const router = express.Router();
4
5  /*
6  *Create = post
7  *Read = get == Get all // Get By Id
8  *Update = put
9  *Delete = delete
10 */
11
```

มีการ **import express** และ **Product** จาก โฟลเดอร์ **Models**

และสร้างตัวแปล **router**

**Get all**

```
//Get all
//http://localhost:5000/api/products
router.get("/products", async(req, res) => {
  const products = await Product.find({}); //เรียกใช้ฟังก์ชัน find
  res.json(products);
});
```

เป็นการกำหนด **URL** และการเรียกใช้ฟังก์ชัน ในรูปนี้มีการกำหนด **URL** และ สร้าง  
ตัวแปล **products** และเรียกใช้ฟังก์ชัน **find** เพื่อเรียกข้อมูลทั้งหมดบนฐานข้อมูล  
ออกมาแสดง

## Get By Id

```
//Get By Id
//http://localhost:5000/api/products/61cd68b46ace93a51fcb8fd1
router.get("/products/:id", async(req, res) => {
  const {id} = req.params; //ดึงพารามิเตอร์
  const product = await Product.findById(id); //เรียกใช้ฟังก์ชัน findById ส่ง id ไปด้วย
  res.json(product); //แสดงในรูปแบบ json
});
```

ในรูปนี้มีการกำหนด URL และ สร้างตัวแปล id เพื่อเก็บ พารามิเตอร์และทำการสลายโครงสร้างและเรียกใช้ฟังก์ชัน find โดยส่ง id ไปด้วยเพื่อเรียกดูข้อมูลที่ id ตรง กับค่าที่ส่งไป

## Create new Product

```
//Create new Product
//http://localhost:5000/api/products
router.post("/products", async(req, res) => {
  const payload = req.body;
  const product = new Product(payload); //ส่งข้อมูลไปสร้างชุดข้อมูล
  await product.save(); //save ลง MongoDB
  res.json({message:"Product added !!"})
});
```

ในรูปนี้มีการกำหนด URL

กำหนดตัวแปล payload เพื่อเข้าถึง req.body

กำหนดตัวแปล product ส่งข้อมูลไปสร้างเป็นชุดข้อมูล

ใช้คำสั่ง .save เพื่อบันทึกลงฐานข้อมูล

แสดงข้อความ Product added !!

## Update By Id

```
//Update Product By Id
//http://localhost:5000/api/products/61cd6ab06ace93a51fcb8fd3
✓ router.put("/products/:id", async(req, res) => {
  const {id} = req.params; //ดึงพารามิเตอร์
  const payload = req.body; //ดึงข้อมูลใหม่
  const product = await Product.findByIdAndUpdate(id,{$set:payload});
  res.json({message:`Product id ${id} is Updated !!`})
});
```

ในรูปนี้มีการกำหนด URL

สร้างตัวแปล id เพื่อเก็บ พารามิเตอร์และทำการสลายโครงสร้าง

กำหนดตัวแปล payload เพื่อเข้าถึง req.body

สร้างตัวแปล products และเรียกใช้ฟังก์ชัน findByIdAndUpdate ส่ง id  
และ set ส่งค่าไปด้วย payload

แสดงข้อความ

## Delete Product By Id

```
//Delete Product By Id
//http://localhost:5000/api/products/61cd68b46ace93a51fcb8fd1
router.delete("/products/:id", async(req, res) => {
  const {id} = req.params; //ดึงพารามิเตอร์
  const product = Product.findById(id);
  await Product.findByIdAndDelete(id); //เรียกใช้ฟังก์ชัน findByIdAndDelete ส่ง id ไปด้วย
  res.json({message:`Product id ${id} is Deleted!!`})
});

export default router;
```

ในรูปนี้มีการกำหนด URL

สร้างตัวแปล **id** เพื่อเก็บ พารามิเตอร์และทำการสลายโครงสร้าง

เรียกใช้ฟังก์ชัน **findByIdAndDelete** ส่ง **id**

แสดง ข้อความ

และส่งออก **router**

## Index.js

```
import express from "express";
import bodyParser from "body-parser";
import mongoose from "mongoose";
import productRouter from "../routes/product.js";
import dotenv from 'dotenv' //ให้รู้จักไฟล์
dotenv.config();

//Create server
const app = express();

//Use Middleware
app.use(bodyParser.json({ limit: "30mb", extended: true })); //แปลงให้อยู่ในรูปแบบ json
app.use(bodyParser.urlencoded({ limit: "30mb", extended: false })); //เข้ารหัส URL

// Use Router
app.use("/api", productRouter);

const CONNECTION_URL = process.env.MONGODB_URL; //กำหนดช่องทางการเชื่อมต่ออยู่ในไฟล์ ENV
const PORT = process.env.PORT || 5000; //กำหนดหมายเลข port

//Connect to MongoDB
mongoose
  .connect(CONNECTION_URL, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => //ถ้าเชื่อมต่อได้ให้โชว์
    app.listen(PORT, () => console.log(`Server running on port: ${PORT}`))
  )//ถ้าไม่ให้ให้แสดง error
  .catch((error) => console.log(error.message));
```

เรียกใช้ module ต่างๆ

```
import express from "express";
import bodyParser from "body-parser";
import mongoose from "mongoose";
import productRouter from "../routes/product.js";
import dotenv from 'dotenv' //ให้รู้จักไฟล์
dotenv.config();
```

dot.env สำคัญเพราะเก็บ URL ที่จะทำให้เชื่อมต่อกับฐานข้อมูลไว้

```
//Create server
const app = express();

//Use Middleware
app.use(bodyParser.json({ limit: "30mb", extended: true })); //แปลงให้อยู่ในรูปแบบ json
app.use(bodyParser.urlencoded({ limit: "30mb", extended: false })); //เข้ารหัส URL

// Use Router
app.use("/api", productRouter);
```

## ส่วนของ MONGODB

```
const CONNECTION_URL = process.env.MONGODB_URL; //กำหนดช่องทางการเชื่อมต่ออยู่ในไฟล์ ENV

const PORT = process.env.PORT || 5000; //กำหนดหมายเลข port

//Connect to MongoDB
mongoose
  .connect(CONNECTION_URL, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => //ถ้าเชื่อมต่อได้ให้โชว์
    app.listen(PORT, () => console.log(`Server running on port: ${PORT}`))
  )//ถ้าไม่ให้เห็นแสดง error
  .catch((error) => console.log(error.message));
```

กนกพล พวงวัดโพธิ์ 624259001