Product API

อันดับแรก ไฟล์ .env กำหนด URL เพื่อเชื่อมต่อ MongoDB และ Port ที่เชื่อมต่อ

ต่อมาไฟล์

product.js ในโฟลเดอร์ model

จะ import mongoose เพื่อเรียกใช้โครงสร้าง

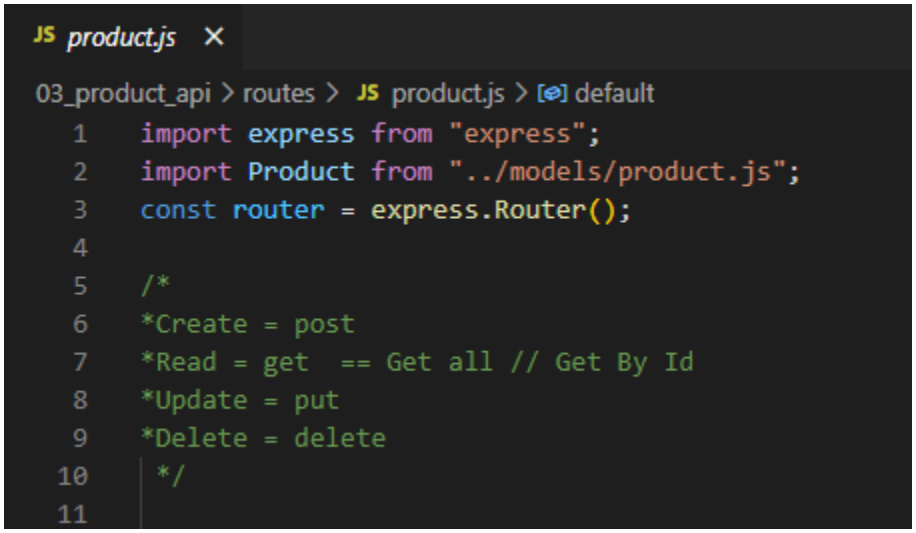
กำหนด Schema

ต่อมากำหนด ตัวแปล เพื่อกำหนด Schema หรือ โครงสร้างของข้อมูล

กำหนด ProductModel = productSchema ที่กำหนดไว้ก่อนหน้านี้

และทำการส่งออกเพื่อเรียกใช้ที่ไฟล์อื่น

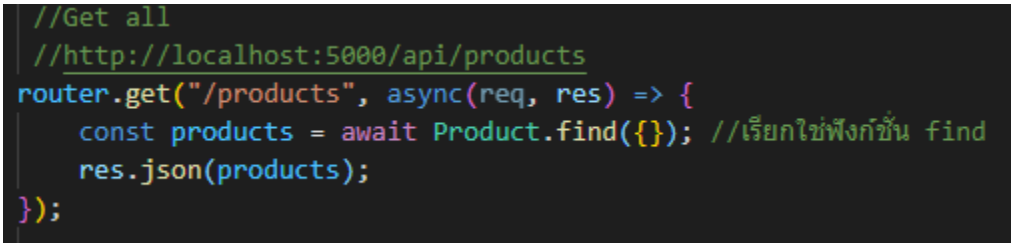
ต่อมา ไฟล์ product.js ในโฟลเดอร์ routes



มีการ import express และ Product จาก โฟลเดอร์ Models

และสร้างตัวแปล router

Get all



เป็นการกำหนด URL และการเรียกใช้ฟังก์ชั่น ในรูปนี้มีการกำหนด URL และ สร้างตัวแปล products และเรียกใช้ฟังก์ชั่น find เพื่อเรียกข้อมูลทั้งหมดบนฐานข้อมูลออกมาแสดง

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); //เรียกใช้ฟังก์ชั่น findById ส่ง id ไปด้วย
    res.json(product); //แสดงในรูปแบบ json
});
```

ในรูปนี้มีการกำหนด URL และ สร้างตัวแปล id เพื่อเก็บ พารามิเตอร์และทำการสลายโครงสร้างและเรียกใช้ฟังก์ชั่น find โดยส่ง id ไปด้วยเพื่อเรียกดุข้อมูลที่ 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

ใช้คำสัง .save เพื่อบันถึงลงฐานข้อมูล

แสดงของความ 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 และเรียกใช้ฟังก์ชั่น 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); //เรียกใช้ฟังก์ชั่น findByIdAndDelete ส่ง id ไปด้วย
    res.json({message:`Product  id ${id} is Deleted!!`})
});

export default router;
```

ในรูปนี้มีการกำหนด URL

สร้างตัวแปล id เพื่อเก็บ พารามิเตอร์และทำการสลายโครงสร้าง

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

กนกพล พวงวัดโพธิ์ 624259001