基本設置

1. 建立資料夾proshop，進入後執行npx create-react-app fronted安裝react
2. 將fronted裡的.git刪除(使用指令)，並將.gitignore移到最上層的資料夾
3. .gitignore內，將/node\_modules改成node\_module並增加node\_modules/ 及 .env
4. 到最上層資料夾啟用git init安裝版本控制，之後git add . 及git commit –m

製作資料

製作component資料都要記得導出

export default products

安裝樣式

在fronted內npm install react-bootstrap

按需引入樣式

import { Container } from 'react-bootstrap'

建立路由

執行npm install react-router-dom@5 react-router-bootstrap

在App.js引入路由器及路由

import { BrowserRouter as Router, Route } from 'react-router-dom'

<Router>

      <Header />

      <main className="py-3">

        <Container>

          <Route path="/" component={HomeScreen} exact />

          <Route path="/product/:id" component={ProductScreen} />

        </Container>

      </main>

      <Footer />

    </Router>

(版本六需要引入Routes)

import { BrowserRouter as Router, Routes, Route } from 'react-router-dom'

路由上改以element標示導入頁面並需要插入完整標籤

    <Router>...

          <Routes>

            <Route path="/" element={<HomeScreen />} exact />

          </Routes>

    </Router>

到Header引入react-bootstrap專用的路由

import { LinkContainer } from 'react-router-bootstrap'

將組件庫的標籤用LinkContainer包裹

          <LinkContainer to="/">

            <Navbar.Brand>Proshop</Navbar.Brand>

          </LinkContainer>

              <LinkContainer to="/cart">

                <Nav.Link>

                  <i className="fas fa-shopping-cart"></i>Cart

                </Nav.Link>

              </LinkContainer>

後端設置

在底層執行npm init並安裝npm install express

建立服務器位置

const express = require('express')

const app = express()

app.listen(5000, () => {

  console.log('Server is running on port 5000.')

})

運行服務器 到運行腳本把增加start

這樣運行只需輸入npm start

  "scripts": {

    "start": "node backend/server"

  },

新增backend資料夾並建立server.js及新增data資料夾存放資料

並將導出改成module.exports

module.exports = products

設定api讓服務器找到指定的json資料

app.get('/api/products', (req, res) => {

  res.json(products)

})

app.get('/api/products/:id', (req, res) => {

  const product = products.find((p) => p.\_id === req.params.id)

  res.json(product)

})

到客戶端安裝axios

以後端設置的api獲取資料

const ProductScreen = ({ match }) => {

  const [product, setProduct] = useState({})

  useEffect(() => {

    const fetchData = () => {

      axios.get(`/api/products/${match.params.id}`).then((res) => {

        let product = res.data

        setProduct(product)

      })

    }

    fetchData()

  }, [])

解決跨域問題

到package.json加入代理服務器

  "proxy": "http://localhost:5000/",

配置便捷的執行環境

執行npm install -D nodemon concurrently

增加運行配置

    "server": "nodemon backend/server",

    "client": "npm start --prefix frontend", //可以到前端資料夾執行npm run start

    "dev": "concurrently \"npm run server \" \"npm run client\"" //可以同時運行

建立隱藏環境

執行npm install dotenv並獲取

const dotenv = require('dotenv')

新增檔案.env到最上層，並加入隱藏內容

NODE\_ENV = development

PORT = 5000

const PORT = process.env.PORT || 5000

app.listen(PORT, () => {

  console.log(

    `Server is running in ${process.env.NODE\_ENV} mode on port ${PORT}.`,

  )

})

統一引入技術

在package.json加入

  "type":"module",

並將引入方式改成import from

引入檔案需要加上後綴檔名

import express from 'express'

import dotenv from 'dotenv'

import products from './data/products.js'

資料也要以export default方式導出

export default products

連接資料庫

得到資料庫URI

在後端資料夾新增config並在內建立db.js做為資料庫的連接檔

import mongoose from 'mongoose'

const connectDB = async () => {

  try {

    const connect = await mongoose.connect(process.env.MONGO\_URI, {

      useUnifiedTopology: true,

      useNewUrlParser: true,

    })

    console.log(`MongoDB connected: ${connect.connection.host}`)

  } catch (error) {

    console.error(`Error: ${error.message}`)

    process.exit(1)

  }

}

export default connectDB

到服務器引入

import connectDB from './config/db.js'

使用元件

connectDB() //需要放在dot.config()後面

強調提示訊息

執行npm install colors不需引入即可使用

只要在字串後連接樣式

  console.log(`Server is on ${process.env.NODE\_ENV} mode on port ${PORT}.`.bgGreen.italic)

  console.error(`Error: ${error.message}`.red.bold)