



修改及刪除資料



designed by freepik

Estimated time:

50 min.

III 資訊工業策進會 Institute for Information Industry

【Key Points】：

學習目標

- 15-1: 使用 AJAX 發送修改表單
- 15-2: 更新資料表
- 15-3: 刪除單筆資料



15-1

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

我們即將一起學習：

- 沿用並修訂Module 13、14的專案，修改：
 - 修改分頁路由及程式
 - 修訂新增路由及前端AJAX設定
- 製作 index.ejs 首頁
- 修訂 app.js 主程式
- 設計編輯表單
- 編輯按鈕與程式
- 更新按鈕的事件處理函式
- 編寫detail路由的程式
- POST /update 路由
- 情境範例 – 更新資料表
- 刪除按鈕的事件處理函式
- POST /delete 路由
- 完成情境範例 – 刪除單筆資料

【Key Points】：

15-1: 使用 AJAX 發送修改表單

15-2: 更新資料表

15-3: 刪除單筆資料

15-1：使用 AJAX 發送修改表單

- 修改分頁路由及程式
- 修訂新增路由及前端AJAX設定
- 製作 index.ejs 首頁與修訂 app.js 主程式
- 設計編輯表單



designed by freepik



designed by freepik

15-2



我們會先整理之前寫過的程式，整合成一個完整的服務。然後再繼續創作編輯和刪除程式。

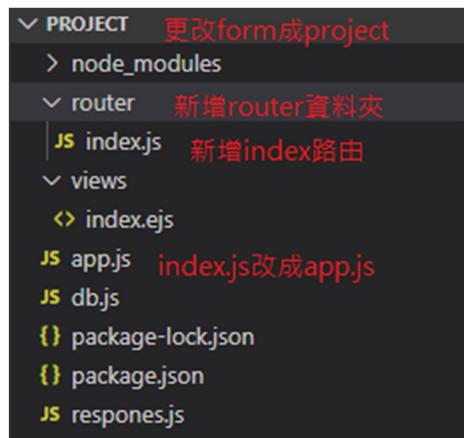
- 沿用並修訂Module 13、14的專案
- 修改分頁路由及程式
- 修訂新增路由及前端AJAX設定
- 製作 index.ejs 首頁
- 修訂 app.js 主程式
- 撰寫編輯表單

【Key Points】：

先整理之前寫過的程式，整合成一個完整的服務
修訂 app.js 主程式
設計編輯表單

沿用並修訂Module 13、14的專案

- 複製form專案，更名成project
 - index.js更名成app.js
 - 新增router資料夾
 - 新增index.js
- router/index.js
 - 引入 mysql, db, responses, express
 - 將index註冊成路由



```
var express = require("express");
var db = require('../db')
var { Success, Error } = require('../response')
var index = express.Router();
```

15-3



1. 複製 form 資料夾，並將其更改成 project
2. 將 index.js 改成 app.js
3. 新增 router 資料夾，在裏頭新增 index.js，此為 index 路由
4. 開啟 router/index.js
5. 引入 mysql, db, response, express 模組，再設定 index 為 express 的路由

【Key Points】：

複製form專案，更名成project
index.js更名成app.js
編寫 router/index.js 的程式

修改分頁路由及程式

• router/index.js

— 新增Module 13的page路由

- 新增根路由無值轉向 /1
- 把原本/page改成/，參數不變
- 轉向路由改成/1
- connection.query改成db.exec
- /page/改成/
- 渲染模板改成index

```
//跳轉到有真數的路由
index.get('/', function(req, res){
  res.redirect('/1') //如果訪問根路由沒有傳遞參數自動轉向/1
})
//獲取分頁資料 /page/:page([0-9]+) 改成 /:page([0-9]+)
index.get('/:page([0-9]+)', function(req, res){
  var page = req.params.page
  //把<=0的id強制改成1
  if(page <= 0) {
    res.redirect('/1') //page/1 改成 /1
    return
  }
  //每頁資料數
  var nums_per_page = 10
  //定義資料偏移量
  var offset = (page - 1) * nums_per_page //把原本connection.query改成我們封裝的db.exec
  db.exec(`SELECT * FROM inventory LIMIT ${offset}, ${nums_per_page};`, [], function(data, fields) {
    db.exec(`SELECT COUNT(*) AS COUNT FROM inventory`, [], function(nums, fields) {
      var last_page = Math.ceil(nums[0].COUNT / nums_per_page)

      //避免請求超過最大頁數
      if(page > last_page) {
        res.redirect('/' + last_page) //page/ 改成 /
        return
      }

      res.render('index',{ page改成 index
        data: data,
        curr_page: page,
        //本頁資料數量
        total_nums: nums[0].COUNT,
        //總數除以每頁資料數，再無條件取整數
        last_page: last_page
      })
    })
  })
})
```

15-4

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

1. 針對 router/index.js · 加入之前 Module 13 的 page 路由
2. 新增根路由 · 將所有訪問不帶值的轉向 /1
3. 把原本 /page/:page([0-9]+) 改成 /:page([0-9]+)
4. Page <= 0 時 · 轉向路由改成 /1
5. 把原本 connection.query() 改成使用我們封裝好的 db.exec
6. 裡頭超過最大頁數的轉向路由 · 從 /page 改成 /
7. 把 render() 的 page 改成 index

【Key Points】：

新增根路由 · 將所有訪問不帶值的轉向 /1
把原本 /page/:page([0-9]+) 改成 /:page([0-9]+)
把 render() 的 page 改成 index

修訂新增路由及前端AJAX設定

- router/index.js

- 新增Module 14的路由

- 把/路由改成/add
 - 模板改成add
 - /add路由改成/insert

- 將index模組化輸出

- 更改views/index.ejs 成 views/add.ejs

- 更改請求路由成/insert

```
insert.get('/add', function(req, res){ //改成/add
    res.render('add') index改成add
})
insert.post('/insert', function(req, res){
    var body = req.body
    var sql = `INSERT INTO inventory(name, phone, address, adult_mask, child_mask) VALUES(?, ?, ?, ?, ?)`
    var data = [body.name, body.phone, body.address, parseInt(body.adult_mask), parseInt(body.child_mask)]
    db.exec(sql, data, function(results, fields) {
        if(results.insertId){
            res.end(
                JSON.stringify(new Success('insert success'))
            )
        } else {
            res.end(
                JSON.stringify(new Error('insert failed'))
            )
        }
    })
})
module.exports = index; 輸出此index路由

//ajax請求
$.ajax({
    url: "/insert", /add 改成 /insert
    type: "POST",
    contentType: "application/json; charset=utf-8",
    data: JSONData,
    success: function(res) {
        var res = JSON.parse(res)
        //後端會封裝一個response class，預先寫好執行No.
        if(res errno === 1) {
            alert("新增成功!")
        } else if(res errno === 0) {
            alert("新增失敗!")
        }
    },
    error: function() {
        alert("系統錯誤!")
    }
})
```

15-5



1. 針對 router/index.js，把原本 insert.get("/", ...) 路由函式，改成 insert.get("/add", ...)
2. 針對 insert.get("/add", ...)，把 render("index") 的 render("add")
3. 把原本 insert.post("/add", ...)，改成 insert.post("/insert", ...)
4. 最後，將 module.exports 輸出改成 index 路由
(上述四點，均是針對 router/index.js 的修訂)
5. 將原本 views/index.ejs(表單) 改名為 add.ejs
6. 針對 add.ejs 檔案，將AJAX請求路由/add改成/insert

【Key Points】：

針對 router/index.js，把原本 insert.get("/", ...) 路由函式，改成 insert.get("/add", ...)

insert.post("/add", ...)，改成 insert.post("/insert", ...)

針對 add.ejs 檔案，將AJAX請求路由/add改成/insert

製作 index.ejs 首頁

- 新增index.ejs
- 將Module 13的views/page.ejs複製進來
- 更改title
- 更改頁碼超連結位置

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>首頁</title>
</head>      EJS 分頁 改成 首頁
<body>
  <table>
    <thead>
      <th>#</th>
      <th>名稱</th>
      <th>電話</th>
      <th>地址</th>
      <th>成功口置庫存</th>
      <th>孩童口置庫存</th>
    </thead>
    <tbody>
      <% data.forEach((item ,index) => { %>
        <tr>
          <td><%= index+1%></td>
          <td><%= item.name%></td>
          <td><%= item.phone%></td>
          <td><%= item.address%></td>
          <td><%= item.adult_mask%></td>
          <td><%= item.child_mask%></td>
        </tr>
      <% }) %>
    </tbody>
    <tfoot>
      <tr>
        <td colspan="4" style="text-align: center;">
          <% for(var i = 1; i <= last_page; i++) { %>
            <span><a href="/<%= i%>"><%= i%></a></span>
          <% } %>
          /page/<%= i%> 改成 /<%= i%>
        </td>
        <td colspan="2">
          <td><%= total_nums%> 筆, 共 <%= curr_page%>/<%= last_page%> 頁</p></td>
        </td>
      </tr>
    </tfoot>
  </table>
</body>
</html>
```

15-6



1. 新增 index.ejs
2. 將Module 13 的 page.ejs 的內容複製進來
3. 更改 title 成首頁
4. 把頁碼的超連結位址從 /page/<%= i%> 改成 /<%= i%>
5. 這個頁面將會變成我們專案的首頁

【Key Points】：

新增 index.ejs

將Module 13 的 page.ejs 的內容複製進來

把頁碼的超連結位址從 /page/<%= i%> 改成 /<%= i%>

修訂 app.js 主程式

- **更改app.js**

- 刪除db, response, 原始路由
 - 引入router/index.js路由
 - 使用express註冊路由

- **測試功能**

- node app.js
 - 前往<http://localhost:3000>
 - 前往<http://localhost:3000/add>
 - 送出表單測試

```
var express = require('express');
var bodyParser = require('body-parser');
//引入index路由
var index = require('./router/index')
var app = express();

//解析json資料
app.use(bodyParser.json());
app.set('view engine', 'ejs');

//註冊index路由
app.use('/', index)

app.listen(3000);
```

15-7



1. 打開 app.js
2. 刪除原本引用的 db, response, 還有 / 和 /add 路由
3. 引入router/index.js
4. 註冊 index 路由
5. 在 VS Code 按下「Ctrl + 撇號」開啟 terminal 終端機視窗，輸入: node index.js
6. 瀏覽器連接首頁: http://localhost:3000
7. 前往新增表單 http://localhost:3000/add
8. 送出表單資料，測試新增功能是否正常

【Key Points】：

引入router/index.js,註冊 index 路由
node index.js
瀏覽器連接首頁: http://localhost:3000，測試功能

設計編輯表單

- 編輯views/index.ejs

- 新增兩按鈕：「編輯」和「刪除」
- 引用jQuery函式館
- 使用jQuery Modal當作編輯表單
 - <https://jqurymodal.com/>

- 設計編輯表單

```
<tbody>
  <% data.forEach((item ,index) => { %>
    <tr>
      <td><%= index+1%></td>
      <td><%= item.name%></td>
      <td><%= item.phone%></td>
      <td><%= item.address%></td>
      <td><%= item.adult_mask%></td>
      <td><%= item.child_mask%></td>
      <td>
        <a href="#ext1" rel="modal:open"><button onclick="Edit(<% item.id%>)">編輯</button></a>
        <button onclick="Delete(<% item.id%>)">刪除</button>
      </td>
    </tr>
  <% } %>
</tbody>
<tfoot>
  <tr>
    <td colspan="4" style="text-align: center;">
      <% for(var i = 1; i <= last_page; i++) { %>
        <span ><a href="/<% i %>"/><% i %></a></span>
      <% } %>
    </td>
    <td colspan="2">
      <td><% total_nums%> 筆，共 <% curr_page%>/<% last_page%> 頁</td>
    </td>
  </tr>
</tfoot>
</table> modal新增編輯表單，hidden是不會顯示值，但是還是可以獲取到
<div id="ext1" class="modal">
  <form id="form">
    <input type="hidden" name="id">
    落戶名稱: <input type="text" name="name"><br>
    電話: <input type="text" name="phone"><br>
    地址: <input type="text" name="address"><br>
    成人口罩遮: <input type="text" name="adult_mask"><br>
    孩童口罩遮: <input type="text" name="child_mask"><br>
    <button type="button" id="submit">更新</button>
  </form>
</div>
<script
src="https://code.jquery.com/jquery-3.4.1.min.js"
integrity="sha256-CSXorXvZcTakIx6Yw6lppzGetbyMqASf1Bw8HFcJo=" crossorigin="anonymous"></script> 引入jQuery Modal函式庫，未啟用Modal
<!-- jQuery modal -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-modal/0.9.1/jquery.modal.min.js"></script>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/jquery-modal/0.9.1/jquery.modal.min.css" />
```

15-8



- 開啟views/index.ejs，在條列資料的表格，每道橫列加入新的<td>標籤，在新的<td>之內，塞入兩個按鈕，分別處理「編輯」和「刪除」。兩個按鈕均要寫作onclick事件處理函式，函式的參數值為各筆資料的id值。
- 編輯按鈕，要再用超連結標籤包住，這是jQuery Modal的使用手法，href要為modal的id，rel為打開modal的參數(model:open)。
- 在</table>之後，新增modal(其實就是一個form表單)，只是這次有多一個type="hidden"的隱藏型欄位，用來記住id，埋個伏筆，方便後來得以知道要更新的是哪一筆資料。
- 從jQuery官網的CDN引用jQuery函式館。
- jQueryModal官網上有引入jQuery Modal的標籤連結，詳情請參考：
<https://jqurymodal.com/>。

【Key Points】：

新增兩個按鈕「編輯」和「刪除」
使用jQuery Modal當作編輯表單
設計編輯表單

15-2：更新資料表

- 編輯按鈕及其程式
- 更新按鈕的事件處理函式
- 編寫detail路由的程式
- POST /update路由



designed by freepik



designed by freepik

15-9

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

接下來，將說明如何處理前端傳遞過來的detail和update請求。

- 編輯按鈕及其程式
- 更新按鈕的事件處理函式
- 編寫detail路由的程式
- POST /update路由
- 情境範例 – 更新資料表

【Key Points】：

編輯按鈕及其程式
編寫detail路由的程式
POST /update路由

編輯按鈕及其程式

- 使用者操作「編輯」的處理流程：

- 使用者點擊某筆記錄的編輯按鈕
- 藉由 id 向Server請求詳細資料
- 將詳細資料印在表單中 (modal)
- Client端更改資料
- AJAX向Server請求更新
- 編輯views/index.ejs
- 撰寫Edit(id) function
 - 返回資料後，將值印在form裡
 - 將 id 寫在URL中

```
function Edit(id) {  
    $.ajax({  
        url: "/detail/" + id, // GET /detail/:id 來獲取該筆資料  
        type: "GET",  
        success: function(res) {  
            var res = JSON.parse(res)  
            //將值傳遞進input裏頭 將值傳入input內  
            $("input[name='id']").val(res.data.id)  
            $("input[name='name']").val(res.data.name)  
            $("input[name='phone']").val(res.data.phone)  
            $("input[name='address']").val(res.data.address)  
            $("input[name='adult_mask']").val(res.data.adult_mask)  
            $("input[name='child_mask']").val(res.data.child_mask)  
        },  
        error: function() {  
            alert("系統錯誤!")  
        },  
    })  
}
```

15-10



- 點擊編輯按鈕
- 觸發編輯事件，觸發時會傳遞 id 到事件處理函式，再由事件處理函式向 Server 請求該筆資料
- 獲取資料後，將資料顯示在表單中，我們使用 modal 可隱藏的懸浮視窗當操作介面
- 使用者更改表單資料，提交更新
- AJAX向Server請求更新

<Note>

- 點擊「編輯」按鈕時會觸發執行Edit(id)，Edit() 函式使用AJAX發出請求到「/detail/id編號」撈資料
- Client端收到資料後，以選擇器語法「input[name='...']」定位，對名入值到各控制項。

【Key Points】：

點擊「編輯」按鈕時會觸發執行Edit(id)
Edit() 函式使用AJAX發出請求到「/detail/id編號」撈資料
獲取資料後，將資料顯示在表單中

更新按鈕的事件處理函式

- 編輯views/index.ejs

- 仿照 add.ejs 表單

- 監聽#submit是否被點擊
 - 表單資料序列化
 - AJAX請求POST /update

```
$('#submit').on('click', function() {
    //整理表單資料到變數
    var data = $('#form').serializeArray()
    JSONData = serializeToJSON(data)

    $.ajax({
        url: "/update",
        type: "POST",
        contentType: "application/json; charset=utf-8",
        data: JSONData,
        success: function(res) {
            var res = JSON.parse(res)
            if(res errno === 1) {
                alert("更新成功!")
                location.reload()
            } else if(res errno === 0) {
                alert("更新失敗!")
            }
        },
        error: function() {
            alert("系統錯誤!")
        }
    })
}

function serializeToJSON(data) {
    var values = {};
    for(index in data){
        values[data[index].name] = data[index].value;
    }
    return JSON.stringify(values)
}
```

15-11



Submit 按鈕的事件處理函式：

- 這個則是 modal 點擊更新後的程式碼
- 仿照 add.ejs 的方式
- 監聽 id=submit 是否有被點擊
- 點擊後將 form 表單序列化整理成 JSON
- 使用 AJAX 請求伺服器更新資料庫的資料

【Key Points】：

監聽 id=submit 是否有被點擊

點擊後將 form 表單序列化整理成 JSON

使用 AJAX 請求伺服器更新資料庫的資料

編寫detail路由的程式

- GET /detail路由

- 獲取id時，只能是整數值
- 用 WHERE 查出符合id的資料
- SELECT 執行結果有/無資料：
 - 有，返回Success和data
 - 沒有，返回Error

```
index.get('/detail/:id([0-9]+)', function(req, res){  
    var sql = `SELECT * FROM inventory WHERE id = ?;`  
    var data = [req.params.id]  
    db.exec(sql, data, function(results, fields) {  
        if(results[0]){  
            res.end(`一筆  
            ${JSON.stringify(new Success(results[0]))}`)  
        } else {  
            res.end(`  
            ${JSON.stringify(new Error('no result'))}`)  
        }  
    })  
})
```

15-12



1. 新增 GET 路由 /detail，index.get("/detail/:id([0-9]+", ...)
2. 並且使用正則表示式來過濾資料
3. 使用 WHERE id = ?，來返回符合該條件的資料
4. 執行後，結果會是二維陣列，我們取陣列編號零的元素
5. 陣列編號零的元素如果有資料，返回成功和 data 到用戶端
6. 沒有，返回錯誤

【Key Points】：

index.get("/detail/:id([0-9]+", ...)

使用正則表示式來過濾資料

如果有資料，返回成功和 data 到用戶端

POST /update 路由

- POST /update 路由
 - SET 設定欄位要更新的值
 - WHERE 獲取 id 符合的值
 - affectRows 影響行數：1，返回 Success；0，返回 Error

```
index.post('/update', function(req, res){  
    var body = req.body  
    var sql = `UPDATE inventory SET name = ?, phone = ?, address = ?, adult_mask = ?, child_mask = ? WHERE id = ?`;  
    var data = [body.name, body.phone, body.address, parseInt(body.adult_mask), parseInt(body.child_mask), parseInt(body.id)]  
    db.exec(sql, data, function(results, fields) {  
        if(results.affectedRows){  
            res.end(  
                JSON.stringify(new Success('update success'))  
            )  
        } else {  
            res.end(  
                JSON.stringify(new Error('update failed'))  
            )  
        }  
    })  
})
```

15-13



1. 新增 POST 路由 /update
2. 使用 SQL 的 UPDATE 語法，異動資料表的資料：
 UPDATE 資料表名稱
 SET 欄位1=新值1
 WHERE id = ?
3. 執行後，透過 affectedRows (受影響筆數) 來判斷是否修改成功。
4. 1，表示成功改到一筆。
5. 0，表示一筆都沒改成。

【Key Points】：

新增 POST 路由 /update

使用 SQL 的 UPDATE 語法，異動資料表的資料

透過 affectedRows (受影響筆數) 來判斷是否修改成功

情境範例 – 更新資料表

- 執行node app.js

- 前往<http://localhost:3000>
- 編輯穎川藥局
- 任意更改
- 提交更新

The modal dialog box contains the following fields:

- 藥局名稱: 穎川藥局
- 電話: (02)296606
- 地址: 新北市板橋區民權路202巷4弄1號
- 成人口罩庫存: 0
- 孩童口罩庫存: 458
-

#	名稱	電話	地址	成功口罩庫存	孩童口罩庫存	編輯	刪除
1	新穎川藥局	(02)123123	新北市板橋區民權路202巷4弄1號	60	600		

15-14

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

1. 在 VS Code 按下「Ctrl + 滑鼠」開啟 terminal 終端機視窗，輸入: node app.js
2. 用瀏覽器，連結到 <http://localhost:3000>
3. 點按第一筆資料的「編輯」按鈕。
4. 鍵入一些資料
5. 點按「更新」按鈕
6. 看看是否有成功更新

【Key Points】：

在 VS Code 按下「Ctrl + 滑鼠」開啟 terminal 終端機視窗，輸入: node app.js
用瀏覽器，連結到 <http://localhost:3000>
測試編輯功能

15-3：刪除單筆資料

- 刪除按鈕的事件處理函式
- POST /delete 路由
- 情境範例 – 刪除單筆資料



designed by freepik



designed by freepik

15-15

III 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

在這個章節，將說明如何處理前端傳遞過來的delete請求：

- 設計「刪除」按鈕的事件處理函式
- 使用AJAX請求伺服器執行 /Delete 路由函式
- 新增 POST 路由 /delete
- 使用 SQL 的 DELETE語法，請求資料庫伺服器刪除資料
- 我們的程式依據資料庫伺服器傳回的 affectedRows 值，判斷是否成功刪除資料
- 測試情境範例 – 刪除單筆資料

【Key Points】：

刪除按鈕的事件處理函式

POST /delete 路由

情境範例 – 刪除單筆資料

刪除按鈕的事件處理函式

- 「刪除資料」的處理流程：
 - 使用者點擊「刪除」按鈕
 - 傳送 id 向Server請求刪除
- 編輯views/index.ejs
撰寫Delete(id) function
 - 將data轉成JSON格式
 - AJAX請求POST /delete

```
function Delete(id) {  
    var JSONData = {"id": id}  
    JSONData = JSON.stringify(JSONData)  
    $.ajax({  
        url: "/delete",  
        type: "POST",  
        contentType: "application/json; charset=utf-8",  
        data: JSONData,  
        success: function(res) {  
            var res = JSON.parse(res)  
            if(res errno === 1) {  
                alert("刪除成功!")  
                location.reload() 成功後重整頁面  
            } else if(res errno === 0) {  
                alert("刪除失敗!")  
            }  
        },  
        error: function() {  
            alert("系統錯誤!")  
        }  
    })  
}
```

15-16



1. 使用者點擊「刪除」按鈕
2. 觸發「刪除」按鈕的事件處理函式: Delete(id)
3. Delete(id) 的id參數，是預備向 Server 請求刪除的資料id編號
4. 使用AJAX請求伺服器執行 /Delete 路由函式
5. location.reload() 的作用是重整頁面

【Key Points】：

- 使用者點擊「刪除」按鈕
- 觸發「刪除」按鈕的事件處理函式
- 使用AJAX請求伺服器執行 /Delete 路由函式

POST /delete 路由

- POST /delete 路由

- 使用 Delete 敘述請 MySQL 刪除資料
- WHERE 子句指定要刪除的是哪一個 id 值
- affectRows 影響行數
 - 1 · 返回 Success
 - 0 · 返回 Error

```
index.post('/delete', function(req, res){  
    var body = req.body  
    var sql = `DELETE FROM inventory WHERE id = ?;`  
    var data = [parseInt(body.id)]  
    db.exec(sql, data, function(results, fields) {  
        //使用affectedRows，判斷是否有被刪除  
        if(results.affectedRows){  
            res.end(  
                JSON.stringify(new Success('delete success'))  
            )  
        } else {  
            res.end(  
                JSON.stringify(new Error('delete failed'))  
            )  
        }  
    })  
})
```

15-17



1. 新增 POST 路由 /delete
2. 使用 SQL 的 DELETE 語法，請求資料庫伺服器刪除資料：
DELETE FROM 資料表名稱 WHERE id = ?
3. 資料庫伺服器執行之後，我們的程式依據 affectedRows 判斷是否成功刪除資料。
4. affectedRows 若是 1，表示成功刪除一筆，程式傳出成功結果給瀏覽器。
5. affectedRows 若是 0，表示一筆都沒刪成。

【Key Points】：

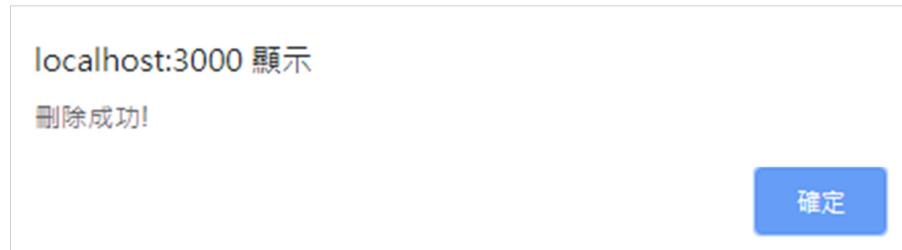
新增 POST 路由 /delete

使用 SQL 的 DELETE 語法，請求資料庫伺服器刪除資料

我們的程式依據資料庫伺服器傳回的 affectedRows 值，判斷是否成功刪除資料

情境範例 – 刪除單筆資料

- 在 VS Code 按下「Ctrl + 滑鼠」開啟 terminal 終端機視窗
在 terminal 終端機視窗: 輸入: node app.js
 - 用瀏覽器連結 <http://localhost:3000>
 - 刪除第一筆資料 (新穎川藥局)



15-18

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

1. 在 VS Code 按下「Ctrl + 滑鼠」開啟 terminal 終端機視窗
2. 在 terminal 終端機視窗: 輸入: node app.js
3. 用瀏覽器連結 <http://localhost:3000>
4. 刪除第一筆資料 (新穎川藥局)
5. 看看畫面的資料是否消失不見

【Key Points】:

在 VS Code 按下「Ctrl + 滑鼠」開啟 terminal 終端機視窗，輸入: node app.js
用瀏覽器連結 <http://localhost:3000>
刪除第一筆資料

情境範例 – 刪除單筆資料

- 啟動 XAMPP Control Panel
- 啟動 phpMyAdmin 管理程式
- 檢視看看資料是否已刪除

#	名稱	電話	地址	成功口罩庫存	孩童口罩庫存	編輯	刪除
Inventory							
1	順儻民權藥局	(02)296095	新北市板橋區民權路234號1樓	0	2065	編輯	刪除
2	順儻民權藥局	(02)296095	新北市板橋區民權路234號1樓	0	2065	編輯	刪除
3	大昌藥局	(02)896825	新北市板橋區北門街28號	0	925	編輯	刪除

15-19



1. 目前，第一筆資料已刪除
2. 在檔案總管點兩下 c:\xampp\xampp-control.exe，啟動 XAMPP Control Panel
3. 確定 Apache 與 MySQL 都正常執行中
4. 點按 MySQL 那列的「Admin」按鈕，啟動 phpMyAdmin 管理程式
5. 展開 Mask 資料庫，再點兩下 inventory 資料表
6. 檢視看看資料是否已刪除

【Key Points】：

- 啟動 XAMPP Control Panel
- 啟動 phpMyAdmin 管理程式
- 檢視看看資料是否已刪除

Summary 〈 精華回顧 〉

- 我們將Module13、14的程式，整理到router/index.js之下
- 使用jQuery Modal來製作編輯資料的表單
- 向Server請求資料，並顯示於表單中
- 點擊「更新」按鈕，向Server請求更新
- 點擊「刪除」按鈕，向Server請求刪除



15-20

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

回顧一下稍早之前的內容

我們將Module13、14的程式，整理到router/index.js之下

並且，介紹了編輯和刪除事件的詳細流程。

另外，也使用jQuery Modal來製作編輯資料的表單。

再來複習一下編輯的流程：使用者點擊「編輯」按鈕，觸發Edit(id)函式。函式向伺服器撈到資料後，顯示資料於表單。接下來，使用者點擊表單上頭的「更新」按鈕，向伺服器請求更新資料。

課程的最後，我們完成了資料刪除功能。

【Key Points】：

前端的編輯和刪除事件

完成了後端的編輯路由函式

完成了後端的刪除路由函式

線上程式題

- **15-1 使用 jQuery 採 Modal 方式顯示表單**
如何使用 jQuery 採 Modal 方式顯示表單？
- **15-2 建立 JSON 格式的字串**
已知 `contentType` 設定為 "application/json"，資料要如何譯出 JSON 格式的字串？
- **15-3 刪除單筆資料**
請問我們該怎麼撰寫按鈕 `click` 事件的程式碼以送出刪除資料的 HTTP 請求？

15-21



題目名稱: 15-1 使用 jQuery 採 Modal 方式顯示表單

內容說明:

如何使用 jQuery 採 Modal 方式顯示表單？

題目名稱: 15-2 建立 JSON 格式的字串

內容說明:

已知 `contentType` 設定為 "application/json"，資料要如何譯出 JSON 格式的字串？

題目名稱: 15-3 刪除單筆資料

內容說明:

請問我們該怎麼撰寫按鈕 `click` 事件的程式碼以送出刪除資料的 HTTP 請求？

請閱讀教學系統的程式與說明，然後，到「// 作答區」填入答案。

答案有區分大寫小寫。

【Key Points】：

15-1 使用 jQuery 採 Modal 方式顯示表單

15-2 建立 JSON 格式的字串

15-3 刪除單筆資料

課後練習題(Lab)

- **情節描述:**
本練習假設Node.js已經安裝於Windows作業系統並且已安裝好XAMPP，也假設您已學過SQL DML的UPDATE和DELETE語法。
- **預設目標:**
實作UPDATE和DELETE，並且了解JS的基礎語法規範。
- **Lab01: 修改資料**
- **Lab02: 刪除資料**
- **完成後的程式與檔案，請參考 Example 資料夾的內容**

Estimated time:
20 minutes

15-22



【情節描述】

本練習假設Node.js已經安裝於Windows作業系統並且已安裝好XAMPP，也假設您已學過SQL DML的UPDATE和DELETE語法。

【預設目標】

實作UPDATE和DELETE，並且了解JS的基礎語法規範。

Lab01: 修改資料

Lab02: 刪除資料

完成後的程式與檔案，請參考 Example 資料夾的內容。

關鍵程式：

```
db.exec(sql, function(results, fields) {  
  // ...  
})
```

【Key Points】：

Lab01: 修改資料

Lab02: 刪除資料

db.exec()

範例程式使用說明

- 範例程式資料夾: Module_15_example
- 使用步驟:
 1. 安裝 Node.js
 2. 安裝 Visual Studio Code
 3. 安裝 XAMPP, 匯入 mask 口罩庫存資料庫
 4. 以 Visual Studio Code 開啟本模組的範例資料夾
 5. 在 Visual Studio Code 按下「Ctrl + 滑鼠」開啟 terminal 終端機視窗。
 6. 在終端機視窗，輸入下列指令，安裝必要的模組套件:
npm install
 7. 在終端機視窗，輸入 node app.js
 8. 啟動瀏覽器，連接 http://localhost:3000/

15-23



使用步驟:

1. 安裝 Node.js (<https://nodejs.org/en/>)
2. 安裝 Visual Studio Code (<https://code.visualstudio.com/>)
3. 安裝 XAMPP (https://www.apachefriends.org/zh_tw/index.html)
4. 在檔案總管點兩下c:\xampp\xampp-control.exe，啟動 XAMPP Control Panel
5. 點按 Apache 與MySQL的「Start」按鈕，啟動兩套伺服器
6. 點按MySQL那列的「Admin」按鈕，啟動phpMyAdmin 管理程式，切換到 SQL 頁籤
7. 複製貼入 mask.sql 內容到SQL 頁籤，然後按下「執行」按鈕。
8. 以 Visual Studio Code 開啟範例資料夾
在檔案總管，滑鼠右鍵點按「本範例資料夾」，從快捷功能表點選「以Code開啟」
或者，啟動 Visual Studio Code 之後，功能表 File | Open Folder，選擇「本範例資料夾」
9. 在 Visual Studio Code 按下「Ctrl + 滑鼠」開啟 terminal 終端機視窗。
10. 在終端機視窗，輸入下列指令，安裝必要的模組套件:
npm install
11. 在終端機視窗，輸入 node app.js
12. 啟動瀏覽器，連接 http://localhost:3000/

【Key Points】：

程式執行環境 Node.js

程式開發編輯環境: Visual Studio Code

在 Visual Studio Code 按下「Ctrl + 滑鼠」開啟 terminal 終端機視窗，輸入: 「node 主程式.js」