



登入功能



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

- 16-1: 建立管理者資料表
- 16-2: 登入判斷
- 16-3: 登入後才能修改及刪除資料



16-1

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

本篇將介紹如何利用登入來設定權限讓使用者操作資料：

- User資料表主鍵
- User 資料表欄位說明
- 加入管理者資料
- 安裝express-session
- 前端登入頁面
- 後端程式處理登入
- 後端登入邏輯
- 權限判斷與畫面渲染
- 路由權限判斷
- 情境範例 – 登入

【Key Points】：

16-1: 建立管理者資料表

16-2: 登入判斷

16-3: 登入後才能修改及刪除資料

16-1：建立管理者資料表

- **User**資料表主鍵
- **User** 資料表欄位說明
- 加入管理者資料



designed by freepik



designed by freepik

16-2

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

我們將建立**user**表，並詳細說明每個欄位屬性，再手動加入管理者的資料。學習重點如下：

- 主鍵的值，按規定要「不重複、不為空值」
- 主鍵的屬性是「**primary key**」
- 方便使用 **id** 來定位，以便於精準進行更新、刪除、查詢等的動作
- 帳號、密碼我們使用字串型態儲存
- 如果儲存資料有中文，盡量將編碼設置為**utf8**
- **rights**是用來判斷使用者的權限大小
- 加入管理者資料

【Key Points】：

建立**user**表

說明每個欄位屬性

手動加入管理者的資料

User資料表主鍵

- 資料表設計

- **id (int 主鍵 AUTO_INCREMENT UNSIGNED 0~65535)**

- 內容必須唯一，不與其他記錄的 id 值重複
 - 不能為 NULL 值 (NULL 代表「未知」的意思)
 - 每個資料表只能有一個主鍵
 - UNSIGNED 代表不包含負數

#	名稱	主鍵	類型	編導與排序	屬性	空值(Null)	預設值	備註	額外資訊	動作
1	id		int(11)	0~65535	UNSIGNED	否	無	AUTO_INCREMENT	修改 刪除 ▾ 更多	
2	account		varchar(128)	utf8_unicode_ci		否	無	自增id，自動新增資料id，	修改 刪除 ▾ 更多	
3	password		varchar(1024)	utf8_unicode_ci		否	無	該欄位要為主鍵才可使用	修改 刪除 ▾ 更多	
4	rights		tinyint(3)	0~255	UNSIGNED	否	無	新增資料會自動新增時間	修改 刪除 ▾ 更多	
5	created_at		datetime			否	current_timestamp()		修改 刪除 ▾ 更多	
6	updated_at		datetime			否	current_timestamp()		修改 刪除 ▾ 更多	

16-3



- id 欄位的值是唯一的，整張資料表的 id 值各不重複。
- 方便使用 id 來定位，以便於精準進行更新、刪除、查詢等的動作，不會誤植。
- id 欄位的資料型態是整數(int)，屬性是「primary key」，同時具有AUTO_INCREMENT自動編號功能（該欄位是主鍵才能使用）。
- 主鍵的值，按規定要「不重複、不為空值」。主鍵也未必要用 int 資料型態，只要符合「唯一、NOT NULL」即可。
- UNSIGNED是讓 $\text{int} \geq 0$ 的設定，可以增大 int 的範圍，不然原本 int 是正負皆有(-32768~32767)

【Key Points】：

主鍵的值，按規定要「不重複、不為空值」

主鍵的屬性是「primary key」

方便使用 id 來定位，以便於精準進行更新、刪除、查詢等的動作

User 資料表欄位說明

- 資料表設計

- account, password (varchar)

- 儲存字元
 - 編碼為 utf8_unicode_ci

- rights (tinyint UNSIGNED 0~255)

- 利用數字來判斷權限大小

#	名稱	主鍵	類型	編碼與排序	屬性	空值(Null)	預設值	備註	額外資訊	動作
1	id	是	int(11)	0~65535	UNSIGNED	否	無	AUTO_INCREMENT	修改 刪除 更多	修改 刪除 更多
2	account		varchar(128)	utf8_unicode_ci		否	無	自增id，自動新增資料id，	修改 刪除 更多	修改 刪除 更多
3	password		varchar(1024)	utf8_unicode_ci		否	無	該欄位要為主鍵才可使用	修改 刪除 更多	修改 刪除 更多
4	rights		tinyint(3)	0~255	UNSIGNED	否	無	新增資料會自動新增時間	修改 刪除 更多	修改 刪除 更多
5	created_at		datetime			否	current_timestamp()		修改 刪除 更多	修改 刪除 更多
6	updated_at		datetime			否	current_timestamp()		修改 刪除 更多	修改 刪除 更多

16-4



- 帳號、密碼我們使用字串型態儲存
- 編碼為 utf8_Unicode_ci
- 如果儲存資料有中文，盡量將編碼設置為utf8，除了可存繁體中文、也能接受日文、韓文等多國文字。
- tinyint 是小型的整數，使用 UNSIGNED，範圍在 0~255
- rights是用來判斷使用者的權限大小

<Note> 關於密碼欄，基於資安考量，儲存密碼的雜湊值（例如：SHA256）是比較好的作法。使用者登入時，也將密碼雜湊，然後與資料庫儲存的雜湊值進行比對。

【Key Points】：

帳號、密碼我們使用字串型態儲存

如果儲存資料有中文，盡量將編碼設置為utf8

rights是用來判斷使用者的權限大小

User 資料表欄位說明

- 資料表設計

- **created_at, updated_at (datetime current_timestamp)**

- 記錄創建和登入時間
 - 格式為 YYYY-MM-DD HH:ii:ss
 - Current_timestamp依照系統時間賦值

#	名稱	主鍵	類型	編碼與排序	屬性	空值(Null)	預設值	備註	額外資訊	動作
1	id		int(11)	0~65535	UNSIGNED	否	無	AUTO_INCREMENT	修改 刪除 更多	
2	account		varchar(128)	utf8_unicode_ci		否	無	自增id，自動新增資料id，	修改 刪除 更多	
3	password		varchar(1024)	utf8_unicode_ci		否	無	該欄位要為主鍵才可使用	修改 刪除 更多	
4	rights		tinyint(3)	0~255	UNSIGNED	否	無	新增資料會自動新增時間	修改 刪除 更多	
5	created_at		datetime			否	current_timestamp()		修改 刪除 更多	
6	updated_at		datetime			否	current_timestamp()		修改 刪除 更多	

16-5



- created_at 是用來記錄該筆「使用者資料」何時加入資料表
- updated_at 是用來記錄最近一次登入的時間，每次登入皆要更新內容。
- 上述兩個欄位的記錄格式為: YYYY-MM-DD HH:ii:ss
- 預設值設為 current_timestamp，在新增資料時，MySQL 會依照系統時間自動賦予內容。
- updated_at 的資料更新，仍要程式設計師寫程式。

【Key Points】：

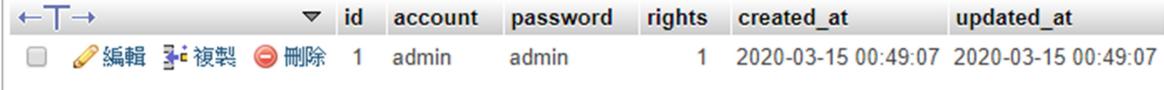
created_at 是用來記錄該筆「使用者資料」何時加入資料表

updated_at 是用來記錄最近一次登入的時間

上述兩個欄位的記錄格式為: YYYY-MM-DD HH:ii:ss

加入管理者資料

- 使用SQL Insert

- `INSERT INTO `user` ('account', 'password', 'rights')
VALUES ('admin', 'admin', 1)`
 - 查看MySQL系統是否有自動設定我們沒有賦值的欄位：
`id, created_at, updated_at`
- 
- `id` 因為 `auto_increment` · 不用設欄位值 · 系統自動會處理
 - `current_timestamp` 只在新增時自動賦值 · 後來還是要我們來update

16-6



1. 開啟 phpMyAdmin
2. 輸入並且執行Insert 敘述：`INSERT INTO `user` ('account', 'password', 'rights') VALUES ('admin', 'admin', 1)`
3. 查看系統是否有自動設定我們沒有賦值的欄位
 - `id`在新增時完全不用處理 · MySQL系統會自動編號 ·
 - `updated_at`在登入時 · 我們的程式要處理更新時間 ·

【Key Points】：

`INSERT INTO `user` ('account', 'password', 'rights') VALUES ('admin', 'admin', 1)`
查看系統是否有自動設定我們沒有賦值的欄位
`updated_at`在登入時 · 我們的程式要處理更新時間

16-2：登入判斷

- 安裝express-session
- 前端登入頁面
- 後端程式處理登入
- 後端登入邏輯



designed by freepik



designed by freepik

16-7

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

接下來，將說明如何製作登入表單並完成登入邏輯程式設計。學習的重點：

- 安裝express-session
- 設定好 session 工作階段的各項參數與程式
- 製作登入表單，內含帳號和密碼文字輸入方塊
- 後端程式處理登入
- `app.get("/login", ...)` 用來傳出登入表單給瀏覽器
- `app.post("/login", ...)` 則是處理登入功能

【Key Points】：

`app.get("/login", ...)` 用來傳出登入表單給瀏覽器
`app.post("/login", ...)` 則是處理登入功能
到資料庫比對帳號密碼是否相符

安裝express-session

- 沿用之前的口罩庫存專案專案

- npm install express-session
 - 打開app.js
 - require() 引用 express-session
 - 設定 session 工作階段

```
C:\Users\Administrator\Desktop\16_1ng\project>npm install express-session
npm WARN form@1.0.0 No description
npm WARN form@1.0.0 No repository field.

+ express-session@1.17.0
added 6 packages from 5 contributors and audited 183 packages in 2.449s
found 0 vulnerabilities
```

```
var express = require('express');
var bodyParser = require('body-parser');
//引入index路由
var index = require('./router/index')
var session = require('express-session');
var app = express();
//session設定
app.use(session({
    secret: 'iamagooddeveloperofjavascript,iwilllearnaboutallofthisapplication',
    resave: false,
    saveUninitialized: false,
    cookie: {
        path: '/',
        httpOnly: true,
        secure: false,
        maxAge: 10 * 60*1000
    }
}))
//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);
```

16-8



1. 用 VS Code 編輯之前的口罩庫存專案
2. 在 VS Code 按下「Ctrl + 撇號」開啟 terminal 終端機視窗
3. 輸入「npm install express-session」安裝模組套件
4. 打開 app.js
5. 以 require() 引用 express-session
6. 參考「Module 11. 使用 session」的說明，設定 session 工作階段的各項參數與程式。

【Key Points】：

npm install express-session
require("express-session")
設定好 session 工作階段的各項參數與程式

前端登入頁面

- Views資料夾內，新增 login.ejs

- 製作登入表單
- 撰寫 AJAX 登入的請求
- 若登入成功，轉向「/」根路由

```
<form id="form">
    帳號: <input type="text" name="account">
    密碼: <input type="password" name="password">
    <button type="button" id="submit">送出</button>
</form>

<script
src="https://code.jquery.com/jquery-3.4.1.min.js"
integrity="sha256-CSXorXvZcTkaix6Yvo6IppcZGetbYMGWSFlBw8HfCJo="
crossorigin="anonymous"></script>
<script>
$('#submit').on('click', function() {
    var data = $('#form').serializeArray();
    JSONData = serializeToJson(data);

    $.ajax({
        url: "/login",
        type: "POST",
        contentType: "application/json; charset=utf-8",
        data: JSONData,
        success: function(res) {
            var res = JSON.parse(res);
            if(res errno === 1) {
                alert("登入成功!")
                location.href = "/"
            } else if(res errno === 0) {
                alert("登入失敗!")
            }
        },
        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)
}
</script>
```

16-9



1. 新增 views/login.ejs 檔案
2. 製作登入表單，內含帳號和密碼文字輸入方塊
3. 複製之前用過的AJAX程式碼
4. 更改 url 為 /login, tpye 設為 POST
5. 登入成功後，轉向根路由

【Key Points】：

新增 views/login.ejs 檔案

製作登入表單，內含帳號和密碼文字輸入方塊

更改AJAX程式，url 為 /login, tpye 設為 POST

後端程式處理登入

- 在db.js中，修改 `createConnection`
 - 新增 `multipleStatements: true` 屬性
 - 因為稍後我們的程式會一次送出多道 SQL 敘述
 - 沒有設定，會出現錯誤訊息

```
const connection = mysql.createConnection({  
    host: 'localhost',  
    user: 'root',  
    password: 'root',  
    database: 'mask', 多語句執行設定  
    multipleStatements: true,  
});
```

16-10



1. 在db.js中，修改 `createConnection`
2. 在 `createConnection()` 中，新增 `multipleStatements: true`
3. 目的是為了可以在單次查詢中，執行多道 SQL 敘述
4. 如果沒有設定 `multipleStatements` 的話，程式執行會出現錯誤訊息
5. 之後請繼續沿用此種配置，以免出現錯誤訊息。

【Key Points】：

`multipleStatements: true`

一次送出多道 SQL 敘述

沒有設定 `multipleStatements` 的話，程式執行會出現錯誤訊息

後端登入邏輯

- **GET /login**

- 登入頁面
 - 如果有登入了，跳轉根路由

- **POST /login**

- 先更新
 - 後查詢有無這樣的帳號密碼資料：
 - 有，設session，返回 Success
 - 沒有，返回 Error

```
index.get('/login', function(req, res){  
  if(req.session.user) {  
    res.redirect('/')  
  } else {  
    res.render('login')  
  }  
})  
index.post('/login', function(req, res){  
  var sql = `UPDATE user SET updated_at=NOW() WHERE account=? and password=?;  
           SELECT * FROM user WHERE account=? and password=?;`  
  var data = [req.body.account, req.body.password, req.body.account, req.body.password]  
  db.exec(sql, data, function(results, fields) {  
    if(results[1]) {  
      req.session.user = {  
        id: results[1].id,  
        account: results[1].account,  
        rights: results[1].rights,  
        updated_at: results[1].updated_at  
      }  
      res.end(  
        JSON.stringify(new Success('login success'))  
      )  
    } else {  
      res.end(  
        JSON.stringify(new Error('login failed'))  
      )  
    }  
  })  
})
```

16-11



1. 編輯 router/index.js 的程式
2. app.get("/login", ...) 用來傳出登入表單給瀏覽器，有登入則跳轉根路由。
3. app.post("/login", ...) 則是處理登入功能，先UPDATE updated_at的值，再查詢是否有該帳號密碼的資料
4. **如果有：設定好 session.user，返回 Success**
5. 若是沒有，返回 Error

【Key Points】：

編輯 router/index.js 的程式

app.get("/login", ...) 用來傳出登入表單給瀏覽器

app.post("/login", ...) 則是處理登入功能，先UPDATE updated_at的值，再查詢是否有該筆資料

16-3：登入後才能修改及刪除資料

- 權限判斷與畫面渲染
- 路由權限判斷
- 情境範例 – 登入



designed by freepik



designed by freepik

16-12

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

本知識點，將說明如何控制好權限，限制只有登入過的使用者才能操作特定功能。學習重點如下：

- 根據登入狀態，顯示「新增」或「登入」超連結
- 根據登入狀態，決定是否顯示「編輯」、「刪除」按鈕
- 除了畫面根據登入狀態，決定是否顯示操作點給使用者，路由本身的權限檢查，也很重要。
- 試用寫好的情境範例，未登入情況下，瀏覽單前往 <http://localhost:3000>
- 登入後，試試看新增、刪除資料功能

【Key Points】：

根據登入狀態，顯示「新增」或「登入」超連結

路由的權限檢查

試用寫好的情境範例

權限判斷與畫面渲染

- 在app.js路由前新增 locals

- ejs渲染時，避免undefined出錯

```
app.use(function(req, res, next){  
    res.locals.session = req.session;  
    next();  
});
```

- 編輯 views/index.ejs

- <table>上方新增兩個超連結

- 有 session.user，顯示「新增」超連結
 - 沒有 session.user，則顯示「登入」超連結

```
<% if(!session.user) { %>  
<a href="/login">登入</a>  
<% }else { %>  
<a href="/add">新增表單</a>  
<% } %>
```

- 兩個「編輯」、「刪除」按鈕：

- 有 session.user，顯示
 - 無 session.user，不顯示

```
<% if(session.user) { %>  
<td>  
    <a href="#ex1" rel="modal:open"><button onclick="Edit(<%= item.id%>)">編輯</button></a>  
    <button onclick="Delete(<%= item.id%>)">刪除</button>  
</td>  
<% } %>
```

16-13



- 編輯 views/index.ejs
- <table>上方新增「新增」、「登入」兩個超連結，但是...
- 如果有session.user，表示已登入過了，顯示「新增」超連結
- 沒有 session.user，表示尚未登入，顯示「登入」超連結
- 有 session.user，顯示各筆資料右側的「編輯」、「刪除」按鈕
- 無 session.user，不顯示「編輯」、「刪除」按鈕

【Key Points】：

編輯 views/index.ejs

根據登入狀態，顯示「新增」或「登入」超連結

根據登入狀態，決定是否顯示「編輯」、「刪除」按鈕

路由權限判斷

- 新增 middleware/login.js
 - 路由的預處理
 - 判斷 session.user 是否存在
 - 不存在，則返回錯誤

```
var { Error } = require('../response')
var login_render = function (req, res, next) {
  if(req.session.user) {
    next();
  } else {
    res.redirect('/');
  }
}
var login_api = function (req, res, next) {
  if(req.session.user) {
    next();
  } else {
    res.end(
      JSON.stringify(new Error('permission deined'))
    )
  }
}

module.exports = [
  login_render,
  login_api
]
```

16-14



- 除了畫面根據登入狀態，決定是否顯示操作點給使用者，路由本身的權限檢查，也很重要。很難說某些特殊使用者會逕行寫程式對我們的路由提出 HTTP 請求。
- 在路由處理主邏輯之前，預先執行我們所需要的檢查指令
- 這次我們先判斷有沒有 req.session.user 這個值存在
- 有，next()，跳到下一個 middleware 或是主邏輯去處理
- 沒有，結束程式並傳出錯誤訊息給瀏覽器

【Key Points】：

除了畫面根據登入狀態，決定是否顯示操作點給使用者，路由本身的權限檢查，也很重要。
在路由處理主邏輯之前，預先執行我們所需要的檢查指令
判斷的依據是 req.session.user

路由權限判斷

- 打開views/index.js

- 引入middleware/login.js
- 在表單新增login_render

```
var {login_render, login_api} = require('../middleware/login')
```

- 分別在AJAX請求新增login_api

- 效果如下

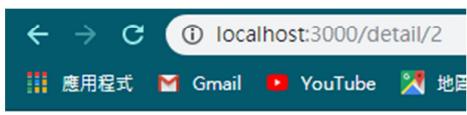
```
index.get('/add', login_render, function(req, res){
```

```
index.get('/detail/:id([0-9]+)', login_api, function(req, res){
```

```
index.post('/insert', login_api, function(req, res){
```

```
index.post('/update', login_api, function(req, res){
```

```
index.post('/delete', login_api, function(req, res){
```



16-15

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

- 編輯 views/index.js
- 引入 login_render 和 login_api
- 在路由的字串後，新增此function即可
- api跟視圖的要不一樣，否則使用者會看到訊息方面的資訊
- 完成後效果如 detail 那個頁面，沒有登入就會被擋住

【Key Points】：

編輯 views/index.js

引入 login_render 和 login_api

在路由的字串後，新增此function即可

情境範例 – 登入

- 執行node app.js
 - 前往<http://localhost:3000>
 - 前往<http://localhost:3000/add>
 - 前往<http://localhost:3000/detail/3>
 - 前往<http://localhost:3000/login>
 - 登入後，再次訪問上面路由
 - 並執行新增，編輯，刪除功能

16-16



1. 在 VS Code 按下「Ctrl + 滑鼠」開啟 terminal 終端機視窗，輸入: node app.js
2. 未登入情況下，瀏覽單前往 <http://localhost:3000>
3. 前往 <http://localhost:3000/add>
4. 前往 <http://localhost:3000/detail/3>
5. 前往 <http://localhost:3000/login>
6. 登入後，試試看新增、刪除資料功能

【Key Points】：

node app.js 啟動伺服器

未登入情況下，瀏覽單前往 <http://localhost:3000>

登入後，試試看新增、刪除資料功能

Summary〈精華回顧〉

- 介紹user表欄位資訊
- 利用 phpMyAdmin 手動新增管理者資料
- 建立登入表單路由和登入路由
- 更改 mysql 連線配置，允許多行語句執行
- 將 req.session 設定為 res.locals 方便EJS使用
- 利用 if else 來斷定登入前後要顯示什麼內容
- 建立 middleware 預先判斷是否存在 session



16-17

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

回顧一下前面的內容

- 介紹user表欄位資訊
- 介紹完之後，利用 phpMyAdmin 手動新增管理者資料，預設帳號密碼為admin/admin
- 製作登入表單
- 製作登入表單後，並建立兩路由：GET /login為視圖，POST /login為處理登入的路由
- 因為登入處理會一次用到兩個語句，所以在連線配置的時候設定可以多語句執行
- 因為EJS渲染時，如果傳遞的變數沒有值會直接出現錯誤訊息，所以直接在express設置公共變數，避免出現錯誤訊息
- 在視圖中，判斷未登入時應該顯示的超連結，登入後應該顯示的按鈕
- 使用middleware預先處理session

【Key Points】：

介紹user表欄位詳細資訊

製作登入表單後，並建立兩路由：GET /login為視圖，POST /login為處理登入的路由

在視圖中，判斷未登入時應該顯示的超連結，登入後應該顯示的按鈕

線上程式題

- **16-1 參數化查詢如何設定參數值**
參數化查詢該如何設定參數值呢？
- **16-2 查詢資料庫進行身份查核**
登入作業時，想要先用目前時刻更新updated_at的欄位值，再查詢使用者資料的帳號密碼資料。程式該怎麼寫呢？
- **16-3 批次查詢如何讀出結果**
一次送出兩個 SELECT 查詢，請問如何如何獲取第一個 SELECT 的資料？

16-18



題目名稱: 16-1 參數化查詢如何設定參數值

內容說明:

參數化查詢該如何設定參數值呢？

題目名稱: 16-2 查詢資料庫進行身份查核

內容說明:

登入作業時，想要先用目前時刻更新updated_at的欄位值，再查詢使用者資料的帳號密碼資料。
程式該怎麼寫呢？

題目名稱: 16-3 批次查詢如何讀出結果

內容說明:

一次送出兩個 SELECT 查詢，請問如何如何獲取第一個 SELECT 的資料？

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】：

16-1 參數化查詢如何設定參數值

16-2 查詢資料庫進行身份查核

16-3 批次查詢如何讀出結果

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，也假設您知道如何操作phpMyAdmin，以及知道如何使用session。
 - **預設目標:**
學會使用匯入方式創建User表，並且製作簡易的登入判斷。
-
- **Lab01: 創建User表**
 - **Lab02: 視圖登入判斷**
 - **完成後的程式與檔案，請參考 Example 資料夾的內容。**
- Estimated time:
20 minutes

16-19



【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，也假設您知道如何操作phpMyAdmin，以及知道如何使用session。

【預設目標】

學會使用匯入方式創建User表，並且製作簡易的登入判斷。

Lab01: 創建User表

Lab02: 視圖登入判斷

關鍵程式:

```
<% if(!session.user) { %>
<a href="/login">登入</a>
<% }else { %>
<a href="/add">新增表單</a>
<% } %>
```

【Key Points】:

Lab01: 創建User表

Lab02: 視圖登入判斷

```
<% if(!session.user) { %>
```

範例程式使用說明

- 範例程式資料夾: Module_16_example
- 使用步驟:
 1. 安裝 Node.js
 2. 安裝 Visual Studio Code
 3. 安裝 XAMPP, 匯入 mask 口罩庫存資料庫並且執行 user.sql 指令檔
 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/

16-20



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. 相同的方式，也執行 user.sql
9. 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」
10. 在 Visual Studio Code 按下「Ctrl + 滑鼠」開啟 terminal 終端機視窗。
11. 在終端機視窗，輸入下列指令，安裝必要的模組套件: npm install
12. 在終端機視窗，輸入 node index.js
13. 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」