

嚴選唱片網站

Web程式設計 期末專題報告

S10350315 陳宏達



TABLE OF CONTENTS

01

動機

02

Story Board

03

使用到的技術

04

碰到的問題

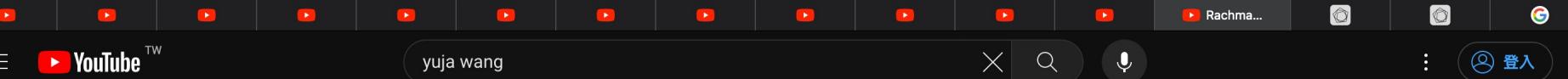
05

未來計畫

動機



TO
↑



- 分頁開太多看起來很亂
- 唱片管理網站(基本CRUD功能)
- 嚴選唱片網站(只有R功能)



Rachmaninoff: Piano Concerto No. 2 in C Minor, Op. 18 - II. Adagio sostenuto

Story Board



02

LOGO

Home News Blog About me

照 片

Welcome to Henry's Select

• Story Board

歌曲

CRUD 按扭

□ □ □ □ □ □ □

作曲家按扭

曲名 |

唱片資料

嵌入 Youtube

使用到的 技術



CO

WHAT WE ARE WORKING ON

前端語言: **ejs**

可以寫JavaScript的html

後端語言: **node.js**

可用 JavaScript 來寫伺服器端的程式碼

資料庫: **sql server**

express 模組
mssql模組
`app.set('view engine', 'ejs');`

建資料表

The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer displays a tree view of servers, databases, and objects. In the center, a query editor window titled 'SQLQuery_1 - localhost...um (sa)' is open, showing the creation of a table 'Chopin'. The table has columns for曲名 (nvarchar(100)), 指揮 (nvarchar(100)), 樂團 (nvarchar(100)), 獨奏家 (nvarchar(100)), 唱片公司 (nvarchar(100)), 嵌入網址 (nvarchar(255)), and 網址 (nvarchar(255)). The primary key constraint is defined as '網址'. Below the code, the 'Messages' pane shows the execution log.

```
create table Chopin
(
    曲名 nvarchar(100),
    指揮 nvarchar(100),
    樂團 nvarchar(100),
    獨奏家 nvarchar(100),
    唱片公司 nvarchar(100),
    嵌入網址 nvarchar(255),
    網址 nvarchar(255),
    primary key(網址)
)
```

Messages

下午1:52:20 Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.012

新增資料

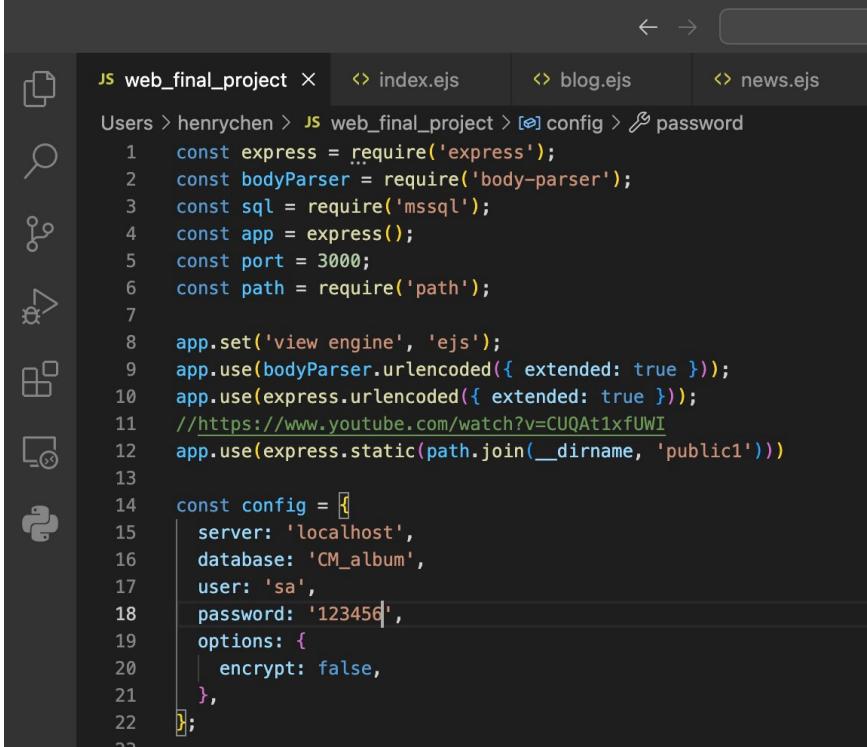
The screenshot shows the SQL Server Management Studio interface with three query windows. The central window is titled 'SQLQuery_4 - localhost...um (sa)' and contains an 'insert into' statement for the 'Chopin' table. The statement inserts seven rows of data, each representing a piece by Chopin with its title, conductor, orchestra, soloist, record company, embedded URL, and main URL. The rightmost window shows the execution messages, indicating the query started at line 1, completed successfully, and took 0 seconds.

```
insert into Chopin
values('Chopin: Piano Concerto No.1', 'Krystian Zimerman', 'Polish Festival Orchestra', 'Krystian Zimerman', 'Dejanira Noseda', 'London Symphony Orchestra', 'Seong-Jin Cho', 'Deutsche Grammophon', 'Chopin: Nocturnes, Op.9, No.1', null, null, 'Jan Lisiecki', 'Deutsche Grammophon', 'https://www.youtube.com/embed/Y7UTWY02'), ('Chopin: Nocturnes, Op.9, No.2', null, null, 'Seong-Jin Cho', 'Deutsche Grammophon', 'https://www.youtube.com/embed/Y7UTWY02'), ('Chopin: Nocturnes, Op.27, No.2', null, null, 'Yu-chien Tseng', 'Deutsche Grammophon', 'https://www.youtube.com/embed/Y7UTWY02'), ('Chopin: Nocturnes, Op.27, No.2', null, null, 'Maria João Pires', null, 'https://www.youtube.com/embed/Y7UTWY02')
```

Messages

下午2:05:48 Started executing query at Line 1
(6 rows affected)
Total execution time: 00:00:00.035

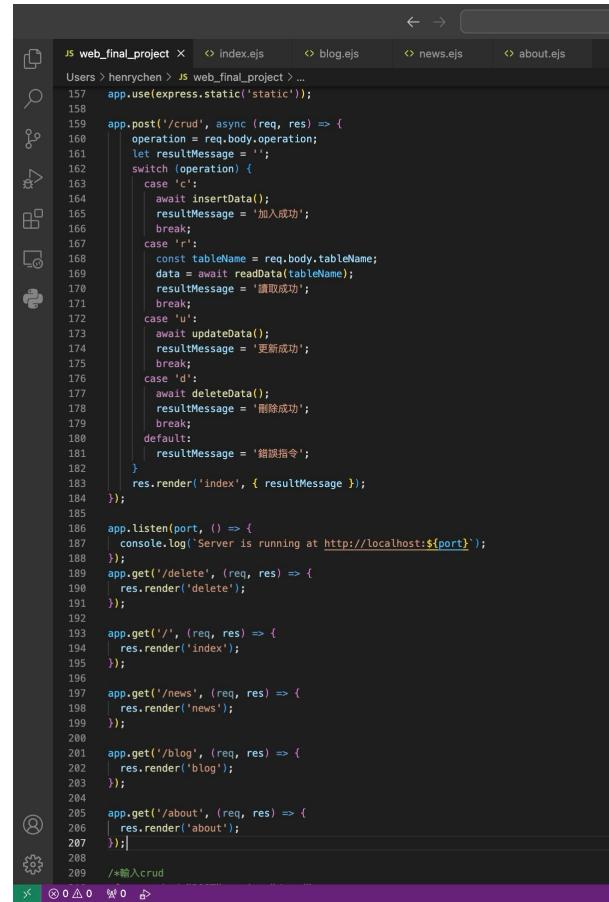
使用到的工具



```
JS web_final_project X < index.ejs > blog.ejs > news.ejs
Users > henrychen > JS web_final_project > [o] config > password
1 const express = require('express');
2 const bodyParser = require('body-parser');
3 const sql = require('mssql');
4 const app = express();
5 const port = 3000;
6 const path = require('path');

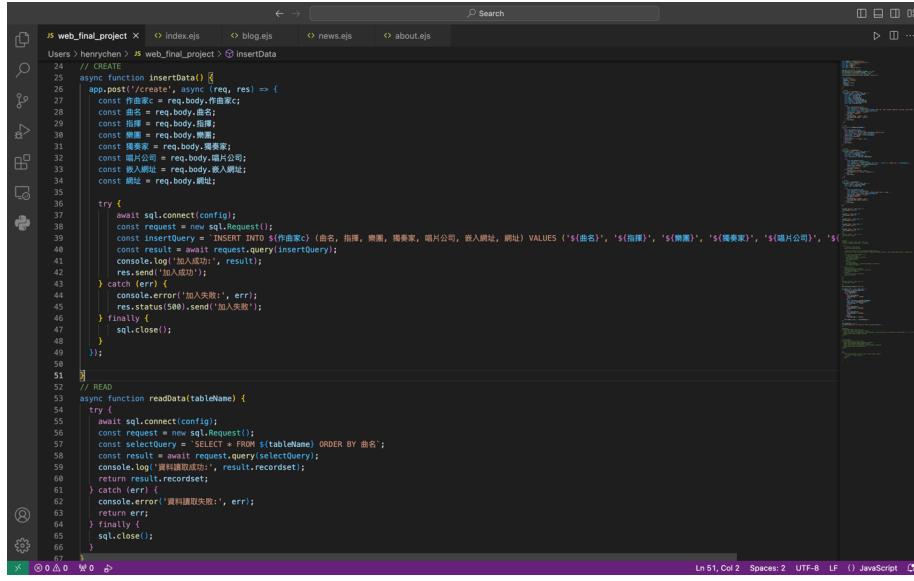
7 app.set('view engine', 'ejs');
8 app.use(bodyParser.urlencoded({ extended: true }));
9 app.use(express.urlencoded({ extended: true }));
10 //https://www.youtube.com/watch?v=CUQAt1xfUWI
11 app.use(express.static(path.join(__dirname, 'public1')));
12
13
14 const config = [
15   server: 'localhost',
16   database: 'CM_album',
17   user: 'sa',
18   password: '123456',
19   options: {
20     encrypt: false,
21   },
22 ];
23
```

main程式



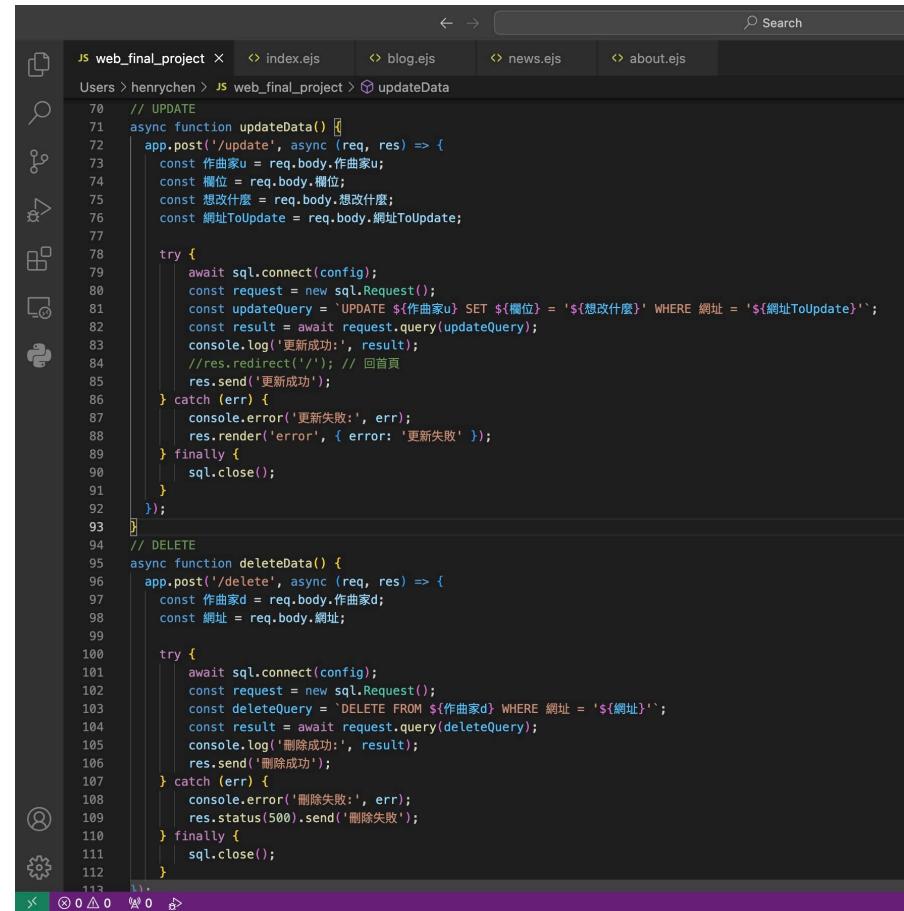
```
Users > henrychen > JS web_final_project > ...
157 app.use(express.static('static'));
158
159 app.post('/crud', async (req, res) => {
160   operation = req.body.operation;
161   let resultMessage = '';
162   switch (operation) {
163     case 'c':
164       await insertData();
165       resultMessage = '加入成功';
166       break;
167     case 'r':
168       const tableName = req.body.tableName;
169       data = await readData(tableName);
170       resultMessage = '讀取成功';
171       break;
172     case 'u':
173       await updateData();
174       resultMessage = '更新成功';
175       break;
176     case 'd':
177       await deleteData();
178       resultMessage = '刪除成功';
179       break;
180     default:
181       resultMessage = '錯誤指令';
182   }
183   res.render('index', { resultMessage });
184 }
185
186 app.listen(port, () => {
187   console.log(`Server is running at http://localhost:${port}`);
188 });
189 app.get('/delete', (req, res) => {
190   res.render('delete');
191 });
192
193 app.get('/', (req, res) => {
194   res.render('index');
195 });
196
197 app.get('/news', (req, res) => {
198   res.render('news');
199 });
200
201 app.get('/blog', (req, res) => {
202   res.render('blog');
203 });
204
205 app.get('/about', (req, res) => {
206   res.render('about');
207 });
208
209 /*輸入crud
```

新增、讀取的函式



```
24 // CREATE
25 async function insertData() {
26   app.post('/create', async (req, res) => {
27     const 作曲家c = req.body.作曲家;
28     const 曲名 = req.body.曲名;
29     const 指揮 = req.body.指揮;
30     const 標題 = req.body.標題;
31     const 插片公司 = req.body.插片公司;
32     const 唱入網址 = req.body.唱入網址;
33     const 網址 = req.body.網址;
34
35     try {
36       await sql.connect(config);
37       const request = new sql.Request();
38       const insertQuery = `INSERT INTO $作曲家 (曲名, 指揮, 標題, 插片公司, 唱入網址, 網址) VALUES ('${曲名}', '${指揮}', '${標題}', '${插片公司}', '${唱入網址}', '${網址}')`;
39       const result = await request.query(insertQuery);
40       console.log('插入成功:', result);
41       res.send('插入成功');
42     } catch (err) {
43       console.error('插入失敗:', err);
44       res.status(500).send('插入失敗');
45     } finally {
46       sql.close();
47     }
48   });
49 }
50
51 // READ
52 async function readData(tableName) {
53   try {
54     await sql.connect(config);
55     const request = new sql.Request();
56     const selectQuery = `SELECT * FROM ${tableName} ORDER BY 曲名`;
57     const result = await request.query(selectQuery);
58     console.log('資料讀取成功:', result.recordset);
59     return result.recordset;
60   } catch (err) {
61     console.error('資料讀取失敗:', err);
62   } finally {
63     sql.close();
64   }
65 }
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
```

更新、刪除的函式



```
70 // UPDATE
71 async function updateData() {
72   app.post('/update', async (req, res) => {
73     const 作曲家u = req.body.作曲家u;
74     const 標位 = req.body.標位;
75     const 想改什麼 = req.body.想改什麼;
76     const 網址ToUpdate = req.body.網址ToUpdate;
77
78     try {
79       await sql.connect(config);
80       const request = new sql.Request();
81       const updateQuery = `UPDATE ${作曲家u} SET ${標位} = '${想改什麼}' WHERE 網址 = '${網址ToUpdate}'`;
82       const result = await request.query(updateQuery);
83       console.log('更新成功:', result);
84       //res.redirect('/');
85       res.send('更新成功');
86     } catch (err) {
87       console.error('更新失敗:', err);
88       res.render('error', { error: '更新失敗' });
89     } finally {
90       sql.close();
91     }
92   });
93
94 // DELETE
95 async function deleteData() {
96   app.post('/delete', async (req, res) => {
97     const 作曲家d = req.body.作曲家d;
98     const 網址 = req.body.網址;
99
100     try {
101       await sql.connect(config);
102       const request = new sql.Request();
103       const deleteQuery = `DELETE FROM ${作曲家d} WHERE 網址 = '${網址}'`;
104       const result = await request.query(deleteQuery);
105       console.log('刪除成功:', result);
106       res.send('刪除成功');
107     } catch (err) {
108       console.error('刪除失敗:', err);
109       res.status(500).send('刪除失敗');
110     } finally {
111       sql.close();
112     }
113   });
114 }
```

04

碰到的問題



問題 問題 問題 問題



ejs讀照片

在Users建Public檔案夾



js跟ejs連接

在Users建views檔案夾



python sql server連接

改用node.js跟ejs



YouTube 嵌入網址

專屬的嵌入網址
重建資料庫

未來計畫



05

- 1.刪除CUD功能
- 2.更新唱片、新聞、部落格





Thank You Very Much

Mercury is the closest planet to the Sun and the smallest one in the Solar System—it's only a bit larger than our Moon. The planet's name has nothing to do with the liquid metal since it was named after the Roman messenger god, Mercury