程式人 月刊 雜誌

Programmer

讀書做善事、寫書做公益 – 歡迎程式人認養專欄或捐出您的網誌

參考價： NT 50 元，如果您喜歡本雜誌，請將書款捐贈公益團體

羅慧夫顱顏基金會 彰化銀行 (009) 帳號： 5234-01-41778-800

0123778

愛心條碼

程式人雜誌

2015 年 5 月

本期焦點：JavaScript 語言的新進展（ES6, io.js 與 koa）

程式人雜誌

前言

編輯小語

最近聽到許多用 node.js 的網友改去用 io.js ，然後他們還開始用了 koa.js 這個框架。

筆者不看不知道，一看不得了，發現這是個重要的好東西，只好趕快來介紹給大家認識。

於是就產生了這期的『JavaScript 語言的新進展 (ES6, io.js 與 koa)』 ....

授權聲明

後面標示其授權，此時該文章將以該標示的方式授權釋出，請修改者注意這些授權標示，以避免產生侵權糾紛。

例如有些文章可能不希望被作為「商業性使用」，此時就可能會採用創作共用：[姓名標示、非商業性、相同方式分享] 的授權，此時您就不應當將該文章用於商業用途上。

最後、懇請勿移除公益捐贈的相關描述，以便讓愛心得以持續散播！

本期焦點

JavaScript 語言的新進展

JavaScript 是瀏覽器前端所能使用的唯一語言，您可以在 HTML 裡面的 <script>....</script> 標記內寫入這種程式，通常用來操控網頁的動態顯示部份，但是在 2009 年 node.js 出現之後，JavaScript 也變成了 server 端最常用的語言之一。

甚至、您可以 JavaScript 用來撰寫『視窗、手機、遊戲』等方面的程式，只要採用 QML、PhoneGap、Unity3D 等平台就可以了，如果您想進一步瞭解 JavaScript 語言，可以參考筆者正在撰寫的下列這本書籍。

- JavaScript 語言

在1995年時，Netscape 公司的布蘭登·艾克為了讓瀏覽器可以執行程式，在 Netscape 瀏覽器上加入了 LiveScript 這個語言，但是由於 Netscape和 Sun 聯盟推動 Java 語言，因此改名為 JavaScript，並且將 JavaScript 登記為商標用語。

因此、雖然我們稱副檔名為 .js 的那個語言為 javascript ，但是由於 JavaScript 是甲骨文公司的註

冊商標，所以現在這個語言的制定單位只能說自己在制定的是 ECMAScript 語言，而不能說是 JavaScript (這種不可說不可說的東西，通常是法律或政治領域造成的）。

由於我對這兩個名詞基本上不加以區分，因此以下我所說 JavaScript 的時候，請自動理解為『那個東西』。

以下是 JavaScript 各個版本制定的年份與內容大要，提供給讀者參考！

版本	完成年份	說明
1	1997	ECMA 組織成立後的第一個 JavaScript 版本
2	1998	格式修正，以使得其形式與ISO/IEC16262國際標準一致
3	1999	加入 regular expressions, try/catch exception handling 等等
4	廢棄	因為政治因素還有某些無法向上相容的語法設計問題
5	2009	加入 strict mode, getters and setters, library support for JSON, 和更完整的 reflection 功能

6	2015	加入超多新語法，較重要的有 for of, yield, import 與物件導向 class 類別等等

您可以看到 ECMAScript 在 1999 年後沈寂了10年，之後在 2009 年又開始動了起來，並且在 2015 年推出了第六版，而這個第六版也就是我們這一系列文章所要介紹的主要對像了。

ECMAScript 6 的新語法

如前文所述，ECMAScript 6 (ES6) 是 2015 年制定完成的 JavaScript 語言標準，既然筆者這篇文章撰寫的時間點也是 2015 年，那您應該會想，這個標準恐怕還得要好幾年才會實現吧！

如果您這樣想的話，那可就大錯特錯了！ 在這個變動快速的今天， 在 ES6 還在一邊制定的時候， Google Chrome 就已經將這些功能加入到瀏覽器當中了，其他瀏覽器當然也不希望因此而落後，所以 Firefox, IE, Opera, Safari 無不卯足了勁的更新他們的瀏覽器引擎。

如果您想要瞭解各家瀏覽器與 javascript 引擎對 ES6 的支援程度，可以參考下列表格：

- https://kangax.github.io/compat-table/es6/

而 server 端的 node.js 既然採用 Google 的 V8 引擎，自然也會隨之更新，於是 node.js 的新版也已經擁有這些功能，只是在使用時必須加上 --harmony 參數才會起動，而為了要不要預設啟動 ES6 功能的問題，以便進一步加入更多先進功能到 node.js 當中，更導致了 node.js 陣營的分裂 (fork)，關於 io.js 之所以要從 node.js 分裂出來的原因，您可以參考下列文章！

- 您不可不知的 io.js -- http://blog.wu-boy.com/2015/02/getting-to-know-io-js/

問題是、到底 ES6 的新語法有哪些呢？ 關於這點說來就長了，筆者整理了一份表格如下。

語法	範例	說明
arrows		
classes		
enhanced object literals		
template strings		
destructuring		

default + rest + spread		
let + const		
iterators + for..of		
generators		
unicode		
modules		
module loaders		
map + set + weakmap + weakset		
proxies		
symbols		
subclassable built-ins		

promises		
math + number + string + array + object APIs		
binary and octal literals		
reflect api		
tail calls		

如果您想更進一步瞭解這些新語法，已經有人寫好很棒的教材了，請參考下列的書籍和範例：

- ECMAScript 6入门 (作者：阮一峰) -- http://es6.ruanyifeng.com/
- github/lukehoban/es6features
- 初探ECMAScript 6 (上)
- 初探ECMAScript 6 (下)

如果您看完了上述的書籍和範例，勢必產生一個驚嘆號！

ECMAScript 第 6 版怎麼改變得這麼大阿！

先別急著驚訝，更厲害的是，這些功能所衍生出來的開放原始碼框架，已經滿坑滿谷了。

要瞭解那滿坑滿谷基於 ES6 的開放原始碼框架，請繼續看下一篇文章！

Yield 與 Generator -- 讓 callback 不再是 javascript 的痛

雖然 新版的 JavaScript (ECMAScript 6, ES6) 有很多新穎的功能，但是其中最重要的，恐怕就是基於 yield 語法所衍生出來的一連串功能了，這種語法在 ruby 當中早就有了，並不是新鮮事，但是引入了 JavaScript 這個語言之後，就讓長期依靠 callback 的 node.js 等平台，有了全新的函式庫框架。

要理解 yield 與 generator 之前，最好先瞭解一下 iterator 這個概念，

Iterator

Iterator 是用來遊走某些容器的物件，裡面通常包涵了 next() 這樣一個用來遊走的函數，可以前進到下一步，以下是一個陣列的 iterator 之實作方法。

檔案： iterator.js

```
var c = console;
```

```
function arrayIterator(array){
  var i = 0;
  var obj = {
    next: function(){
      return i < array.length ?
        {value: array[i++], done: false} :
        {value: undefined, done: true};
    }
  }
  return obj;
}

var ai = arrayIterator(['x', 'y', 'z']);

c.log(ai.next()); // { value: "x", done: false }
```

```
c.log(ai.next()); // { value: "y", done: false }
c.log(ai.next()); // { value: "z", done: false }
c.log(ai.next()); // { value: undefined, done: true }
```

執行結果

```
D:\git\generator>iojs iterator
{ value: 'x', done: false }
{ value: 'y', done: false }
{ value: 'z', done: false }
{ value: undefined, done: true }
```

有了這樣的『遊走裝置』之後，我們就可以對各種容器從頭走到尾，而不需要在意到底容器的內部結構長什麼樣了，只要該容器支援 next() 函數就行了。

例如我們可以用下列程式對陣列的 iterator 進行從頭到尾的遊走過程。

```
while (true) {
```

```
    var item = ai.next();
    if (item.done === true)
        break;
  c.log(item);
}
```

Yield 語法

當您理解了上述的 iterator 之設計方式之後，就可以開始來看看 ECMAScript 6 當中新制定的 yield 語法了，以下是一個使用 yield 語法的範例！

檔案： yield1.js

```
var c = console;

function *gen1(){
    var a=5, b=3;
```

```
    yield "x";
    c.log("a=%d", a);
    yield "y";
    b = a+b;
    c.log("b=%d", b);
    return "z";
}

var g1 = gen1();

c.log(g1.next()); // { value: "x", done: false }
c.log(g1.next()); // { value: "y", done: false }
c.log(g1.next()); // { value: "z", done: true }
c.log(g1.next()); // { value: undefined, done: true }
```

執行結果

```
D:\git\generator>iojs yield1
{ value: 'x', done: false }
a=5
{ value: 'y', done: false }
b=8
{ value: 'z', done: true }
{ value: undefined, done: true }
```

上述的程式看來有點詭異，但是如果您將函數想像成一個『擁有狀態的可執行容器』，而 next() 則是讓該函數前進一步，那麼所謂的 yield 只不過是把該函數目前的狀態傳回來而已。

於是、函數就變成了一種容器物件，而 next() 則可以讓函數向前執行，直到碰到下一個 yield 或 return 指令為止，而 yield 指令傳回的 done 會是 false，但 return 指令則會傳回 done=true。

Yield 的真正用途

大致瞭解了 yield 的原理之後，讓我們來看一下 yield 的真正用途，以下我們將用 delay 函數作為範例，示範 yield 的奇特用法。

檔案： yieldDelay.js

```
var g = gen(); // 建立函數物件
g.next(); // 開始執行到第一個 yield

var totalDelay = 0;

function delay(ms) {
  setTimeout(function() {
    console.log("delay "+ms+" ms");
    totalDelay += ms;
    g.next(totalDelay); // 這個 yield 完成後，傳回 totalDelay 並
呼叫 next() 繼續執行到下一個 yield
  }, ms);
}
```

```
function *gen(){
  var a=5, b=3, t;
  t = yield delay(800);
  console.log("a=%d t=%d", a, t);
  t = yield delay(500);
  b = a+b;
  console.log("b=%d t=%d", b, t);
  t = yield delay(300);
  console.log("totalDelay=%d t=%d", totalDelay, t);
}
```

執行結果

```
D:\git\generator>iojs yieldDelay
delay 800 ms
a=5 t=800
```

```
delay 500 ms
b=8 t=1300
delay 300 ms
totalDelay=1600 t=1600
```

您可以看到

用 **yield** 取代 **callback**

大部分的 javascript 環境並不提供多線程 (multi-thread) 機制，於是輸出入的函數都會改用 callback 的方式設計。

舉例而言， node.js 的輸出入函數就幾乎都有兩個版本，設計者通常會用採用 callback 的非同步版本以便在等待輸出入完成的時候還可以繼續執行程式，以下就是一個採用非同步輸出入的 node 程式範例。

檔案： copyFile.js

```
var fs = require('fs');
```

```
function copyFile(fromFile, toFile) {
  fs.readFile(fromFile, "utf8", function(err, data) {
    console.log('read %s complete!', fromFile);
    fs.writeFile(toFile,  data, function(err) {
      console.log('write %s complete!', toFile);
    });
  });
}

copyFile(process.argv[2], process.argv[3]);
```

執行結果

```
D:\git\generator>node copyFile copyFile.js copyFile2.js
read copyFile.js complete!
```

```
write copyFile2.js complete!
```

但是、有了 yield 指令與有狀態的函數之後，就可以利用下列的方法，讓 callback 轉變為沒有 callback 的函數，只是在尚未結束之前要改用 yield 而已，以下是一個範例。

檔案： yieldFile.js

```
var fs = require('fs');
var gen;

function run(generator) {
  gen = generator();
  gen.next();
}

function read(file) {
  fs.readFile(file, function(err, data) {
```

```
    if (!err) console.log('read %s success!', file);
    gen.next(data);
  });
}

function write(file, data) {
  fs.writeFile(file,  data, function(err) {
    if (!err) console.log('write %s success!', file);
    gen.next();
  });
}

run(function* () {
  var text = yield read('yieldFile.js');
  yield write('yieldFile.bak', text);
});
```

執行結果

```
nqu-192-168-61-142:generator mac020$ iojs yieldFile
read yieldFile.js success!
write yieldFile.bak success!
```

Co - 用 yield 讓控制流程更好用的套件

很好的講解: https://github.com/dead-horse/co-and-koa-talk

安裝

官網： https://github.com/tj/co

範例：複製檔案

程式： coReadWrite.js

```
var co = require('co');
var fs = require('mz/fs');

co(function* () {
  var file1 = yield fs.readFile('coReadWrite.js');
  yield fs.writeFile('coReadWrite.js.bak', file1);
  var file2 = yield fs.readFile('coReadWrite.js.bak');

  console.log("===coReadWrite.js===\n"+file1);
  console.log("===coReadWrite.js.bak===\n"+file2);
});
```

執行結果

```
NQU-192-168-60-101:iojs csienqu$ iojs coReadWrite
===coReadWrite.js===
```

```
var co = require('co');
var fs = require('mz/fs');

co(function* () {
  var file1 = yield fs.readFile('coReadWrite.js');
  yield fs.writeFile('coReadWrite.js.bak', file1);
  var file2 = yield fs.readFile('coReadWrite.js.bak');

  console.log("===coReadWrite.js===\n"+file1);
  console.log("===coReadWrite.js.bak===\n"+file2);
});
===coReadWrite.js.bak===
var co = require('co');
var fs = require('mz/fs');

co(function* () {
```

```
  var file1 = yield fs.readFile('coReadWrite.js');
  yield fs.writeFile('coReadWrite.js.bak', file1);
  var file2 = yield fs.readFile('coReadWrite.js.bak');

  console.log("===coReadWrite.js===\n"+file1);
  console.log("===coReadWrite.js.bak===\n"+file2);
});
NQU-192-168-60-101:iojs csienqu$ ls
coRead2.js      iterator.js     yieldFile.js
coReadWrite.js      node_modules
coReadWrite.js.bak  yieldFile.bak
```

將 **callback** 包裝成可以 **yield** 的形式

以下範例來自 -- https://github.com/dead-horse/co-and-koa-talk

範例： 自行將 callback 函數 thunkify

```
fs.stat(filename, callback);

// =>

function stat(filename) {
  return function (done) {
    fs.stat(filename, done);
  }
}

// =>

function *() {
  yield stat('./README.md');
}
```

範例： 使用 thunkify 套件

```
var thunkify = require('thunkify');
var co = require('co');
var fs = require('fs');

var stat = thunkify(fs.stat);
var readFile = thunkify(fs.readFile);

co(function *() {
  var stat = yield stat('./README.md');
  var content = yield readFile('./README.md');
})();
```

自己製作一個 **myCo** 套件

上文的說明應該已經闡述了如何用 yield 取代 callback 的方法，當然這種方法已經被實作成完

整的套件了，像是 co.js 就是一個使用在 koa.js 裏的重要套件。

為了更瞭解 co.js 的原理，我們實作了一個 myCo.js 以便更進一步體會其原理。

檔案： myCo.js

```
var fs = require('fs');
var c = console;

var co = (function() {
  var gen;

  var resume = function(value) {
    gen.next(value);
  }

  function read(file) {
```

```
  fs.readFile(file, function(err, data) {
    if (!err) c.log('read %s success!', file);
    resume(data);
  });
}

function write(file, data) {
  fs.writeFile(file,  data, function(err) {
    if (!err) c.log('write %s success!', file);
    resume();
  });
}

function run(generator) {
    gen = generator();
    gen.next();
```

```
  }

  return {
    run:run,
    write:write,
    read:read
  }
})();


function *copyFile(fromFile, toFile) {
  c.log('copyFile %s %s', fromFile, toFile);
  var text = yield co.read(fromFile);
  yield co.write(toFile, text);
}
```

```
co.run(function* () {
  c.log('run ...');
  yield *copyFile(process.argv[2], process.argv[3]); // 如果被 yield 的函數裡還有 yield 的話，就要用 yield *
});
```

執行結果

```
nqu-192-168-61-142:generator mac020$ iojs myCo myCo.js myCo.bak
run ...
copyFile myCo.js myCo.bak
read myCo.js success!
write myCo.bak success!
```

koa -- 建構 web 網站的核心框架

簡介

- 說明： express 原班人馬創造支援 ES6 新語法 yield 的套件
- 官網專案： https://github.com/koajs/koa
- 推薦文章：A Simple CRUD Demo with Koa.js Sunday, January 12, 2014
- 推薦影片： James Moore (共十集） -- http://knowthen.com/category/node-js/
- 官方範例： https://github.com/koajs/examples
- single page app : https://medium.com/@adam_bickford/creating-a-basic-site-with-koa-pt-1-f3e1711f7a9

Koa 與 Express 相關套件的對應表

	Koa 相關套件	Express 相關套件
中間件	koa	connect
路徑比對	koa-router	express

檔案靜態化	koa-static, koa-static-folder	express.static
資料夾索引	koa-serve-index	serve-index
檔案輸出入	co-fs, mz/fs, save-to	fs
post body 處理	koa-bodyparser, co-busboy	body-parser
session 管理	koa-session	express-session

官方範例

請先安裝 git, io.js 並用下列指令下載 koa.js 官方範例後，再開始執行後面給的例子。

```
$ git clone https://github.com/koajs/examples.git
$ cd examples
$ ren examples koa-ex
$ npm install
$ npm install swig
```

範例: Hello World

- 原始碼： https://github.com/koajs/examples/tree/master/hello-world
- 使用方法：執行 iojs app 之後看 http://localhost:3000/

範例:

- 原始碼： https://github.com/koajs/examples/tree/master/stream-file
- 使用方法：執行 iojs app 之後看 http://localhost:3000/README.md

範例: templates

- 原始碼： https://github.com/koajs/examples/tree/master/templates
- 使用方法： 執行 iojs index 之後看 http://localhost:4000/

程式人文集

簡單留言系統 -- 使用多頁技術

簡介

在 koa 的 github 官方專案上，提供了一個簡單的網誌留言系統（說是網誌，但比較像留言），這個系統設計得非常簡單，因此是學習 koa 很好的入門材料，以下是該專案的原始碼網址。

- 原始碼： https://github.com/koajs/examples/tree/master/blog

安裝執行過程如下：

```
D:\git\koa-ex>npm install swig
swig@1.4.2 node_modules\swig
 ├─── optimist@0.6.1 (wordwrap@0.0.2, minimist@0.0.10)
 └─── uglify-js@2.4.19 (uglify-to-browserify@1.0.2, async@0.2.10
```

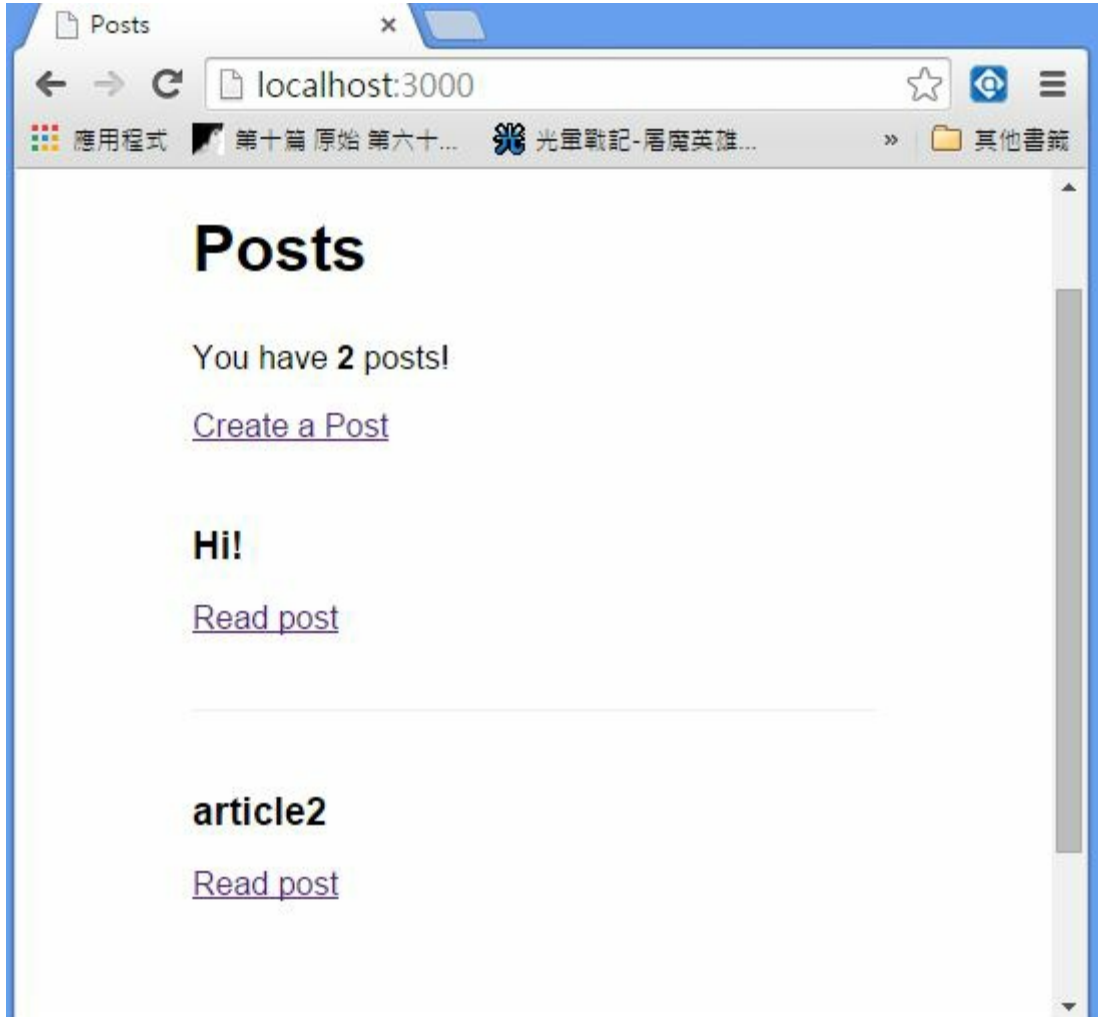
```
, yargs@3.5.4,
source-map@0.1.34)

D:\git\koa-ex>cd blog

D:\git\koa-ex\blog>iojs index
listening on port 3000
  <-- GET /post
  --> GET /post 404 23ms -
  <-- GET /
  --> GET / 200 116ms -
  <-- GET /post/new
  --> GET /post/new 200 18ms -
  <-- POST /post
  --> POST /post 302 47ms -
  <-- GET /
```

```
  --> GET / 200 10ms -
  <-- GET /post/new
  --> GET /post/new 200 40ms -
  <-- POST /post
  --> POST /post 302 32ms -
  <-- GET /
  --> GET / 200 25ms -
  <-- GET /post/0
  --> GET /post/0 200 8ms -
  <-- GET /post/1
  --> GET /post/1 200 7ms -
  <-- GET /post/new
  --> GET /post/new 200 5ms -
```

執行結果如下


Posts
localhost:3000
應用程式
第十篇 原始 第六十...
光里戰記-屠魔英雄...
其他書籤
Posts
You have 2 posts!
Create a Post
Hi!
Read post
article2
Read post

這個專案除了採用 koa 框架撰寫伺服端之外，還採用了 swig 這個樣板引擎來呈現網頁，首先讓我們來看伺服端的寫法。

伺服端主程式

檔案： index.js

```
/**
 * Module dependencies.
 */

var render = require('./lib/render');
var logger = require('koa-logger');
var route = require('koa-route');
var parse = require('co-body');
```

```
var koa = require('koa');
var app = koa();

// "database"

var posts = [];

// middleware

app.use(logger());

// route middleware

app.use(route.get('/', list));
app.use(route.get('/post/new', add));
app.use(route.get('/post/:id', show));
```

```
app.use(route.post('/post', create));

// route definitions

/**
 * Post listing.
 */

function *list() {
  this.body = yield render('list', { posts: posts });
}

/**
 * Show creation form.
 */
```

```
function *add() {
  this.body = yield render('new');
}

/**
 * Show post :id.
 */

function *show(id) {
  var post = posts[id];
  if (!post) this.throw(404, 'invalid post id');
  this.body = yield render('show', { post: post });
}

/**
 * Create a post.
```

```
 */

function *create() {
  var post = yield parse(this);
  var id = posts.push(post) - 1;
  post.created_at = new Date;
  post.id = id;
  this.redirect('/');
}

// listen

app.listen(3000);
console.log('listening on port 3000');
```

您可以看到伺服端的 list 功能之處理，大致如下所示。

```
app.use(route.get('/', list));
...
function *list() {
  this.body = yield render('list', { posts: posts });
}
```

這代表當您訪問網址為根目錄時，會呼叫樣板引擎去呈現 list.html 這個樣板，並將貼文 posts 傳入到樣板中。

但是、主程式裏並沒有指定使用哪個樣板引擎，指定的工作是在 lib/render.js 這個檔案裏設定的。(https://github.com/koajs/examples/blob/master/blog/lib/render.js)

```
var views = require('co-views');

// setup views mapping .html
// to the swig template engine
```

```
module.exports = views(__dirname + '/../views', {
  map: { html: 'swig' }
});
```

您可以看到其中設定使用 swig 樣板引擎，還有其樣板的置放目錄為 views 。

接著讓我們看看樣板到底長甚麼樣？ 以下是 list.html 的內容。

檔案： list.html

```
{% extends 'layout.html' %}

{% block title %}Posts{% endblock %}

{% block content %}
  <h1>Posts</h1>
  <p>You have <strong>{{ posts.length }}</strong> posts!</p>
  <p><a href="/post/new">Create a Post</a></p>
```

```
  <ul id="posts">
    {% for post in posts %}
      <li>
        <h2>{{ post.title }}</h2>
        <p><a href="/post/{{ post.id }}">Read post</a></p>
      </li>
    {% endfor %}
  </ul>
{% endblock %}
```

其中第一行又引入了一個 layout.html 的樣板，其內容如下：

```
<html>
<head>
  <title>{% block title %}Blog{% endblock %}</title>
  <style>
```

```
body {
  padding: 80px;
  font: 16px Helvetica, Arial;
}
h1 {
  font-size: 2em;
}
h2 {
  font-size: 1.2em;
}
#posts {
  margin: 0;
  padding: 0;
}
#posts li {
  margin: 40px 0;
```

```
  padding: 0;
  padding-bottom: 20px;
  border-bottom: 1px solid #eee;
  list-style: none;
}
#posts li:last-child {
  border-bottom: none;
}
textarea {
  width: 500px;
  height: 300px;
}
input[type=text],
textarea {
  border: 1px solid #eee;
  border-top-color: #ddd;
```

```
      border-left-color: #ddd;
      border-radius: 2px;
      padding: 15px;
      font-size: .8em;
    }
    input[type=text] {
      width: 500px;
    }
  </style>
</head>
<body>
  <section id="content">
    {% block content %}
      <p>Missing content!</p>
    {% endblock %}
  </section>
```

```
</body>
</html>
```

您可以看到 swig 的樣板有類似繼承的架構，可以透過子樣板中的標記將母樣板的內容修改掉，而這也正是使用樣板引擎的好處之一。

結語

如果還有不清楚的地方，請各位讀者直接參考該專案的原始碼，網址如下：

- https://github.com/koajs/examples/tree/master/blog/views

簡單留言系統 -- 使用單頁技術

簡介

最近在 HTML5 的加持與前端框架越來越強大的影響下，單頁技術開始快速興起，但是我們卻很少看到單頁技術的完整範例。

為了在上課時示範給學生看，筆者寫了一個非常簡單的單頁留言系統。

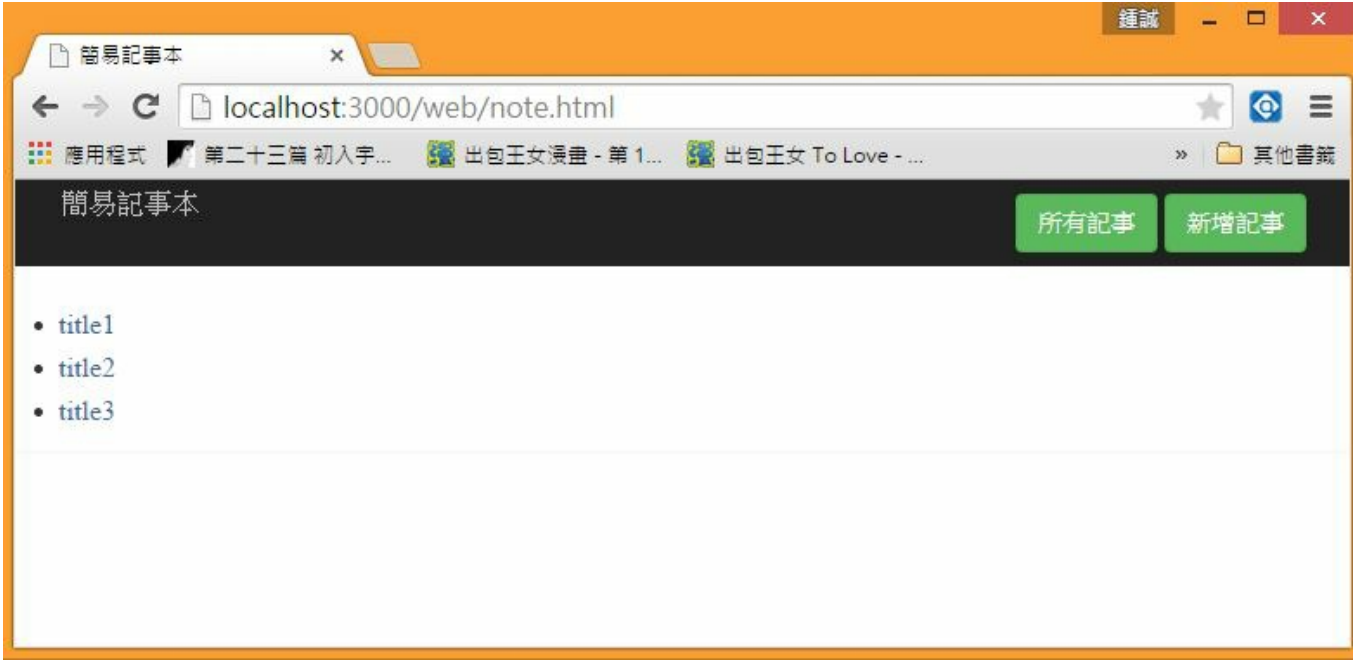
為了簡單起見，該系統沒有用到資料庫，暫時將留言都存在記憶體的 JSON 物件裡面。

這個專案已經上傳到 github 上，以下是專案的網址。

- 專案: https://github.com/ccckmit/KoaSpaNote/

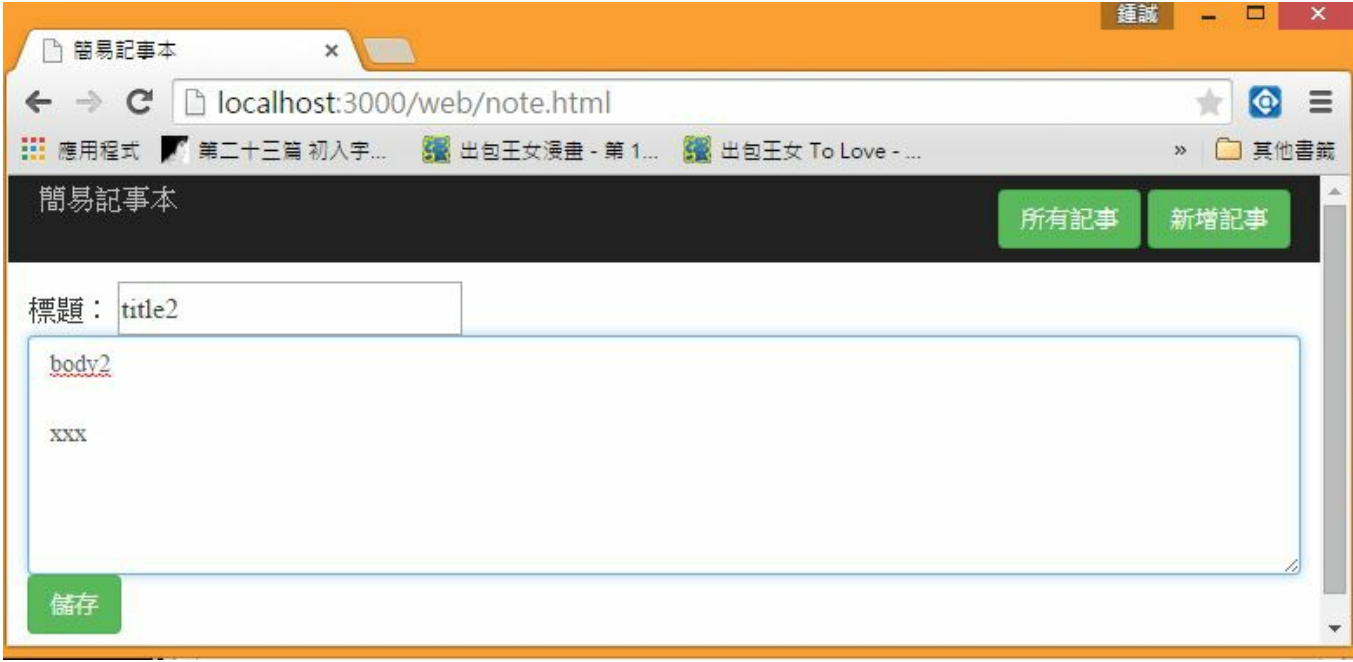
我們在後端使用了 Koa 這個採用 yield 語法的框架，然後在前端採用 bootstrap 和 jQuery 這兩個框架，並且用 AJAX 作為前後端的溝通方法。

這個留言系統可以新增、修改、或列出留言，但我們沒有實做刪除功能，以下是列出留言的畫面。



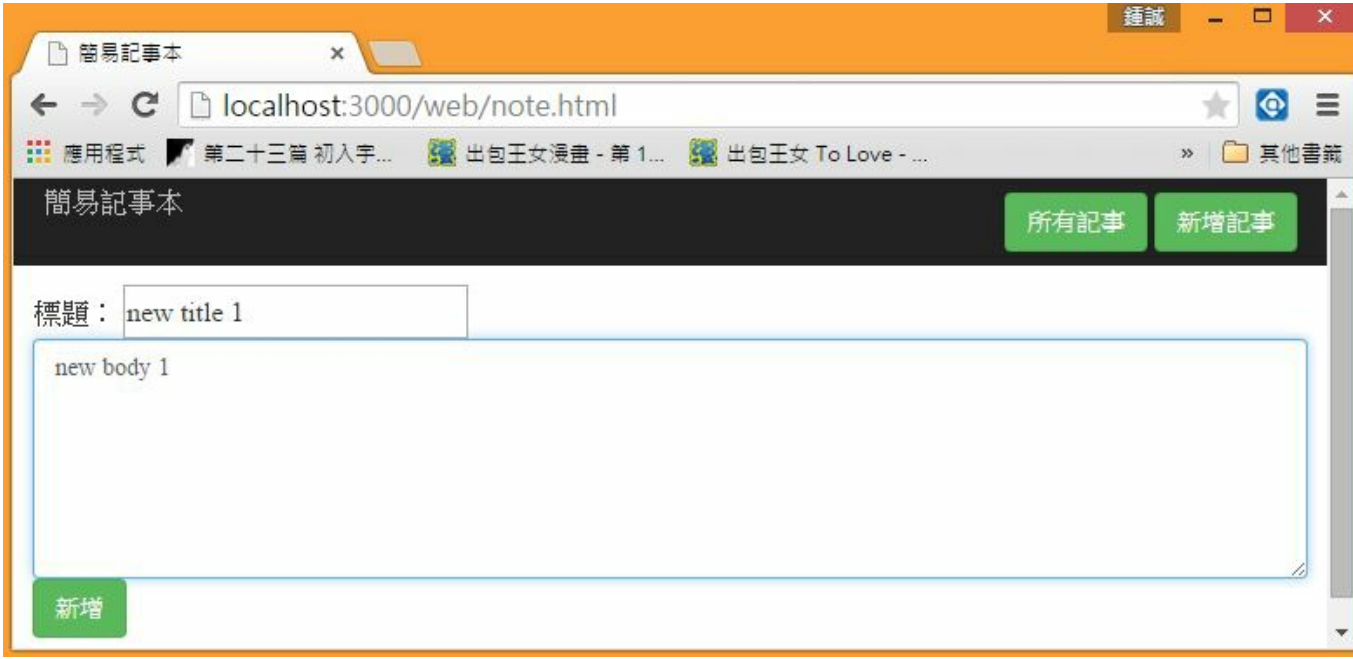
圖、留言列表

然後我們可以按某一筆留言進去編輯，以下是編輯時的畫面。



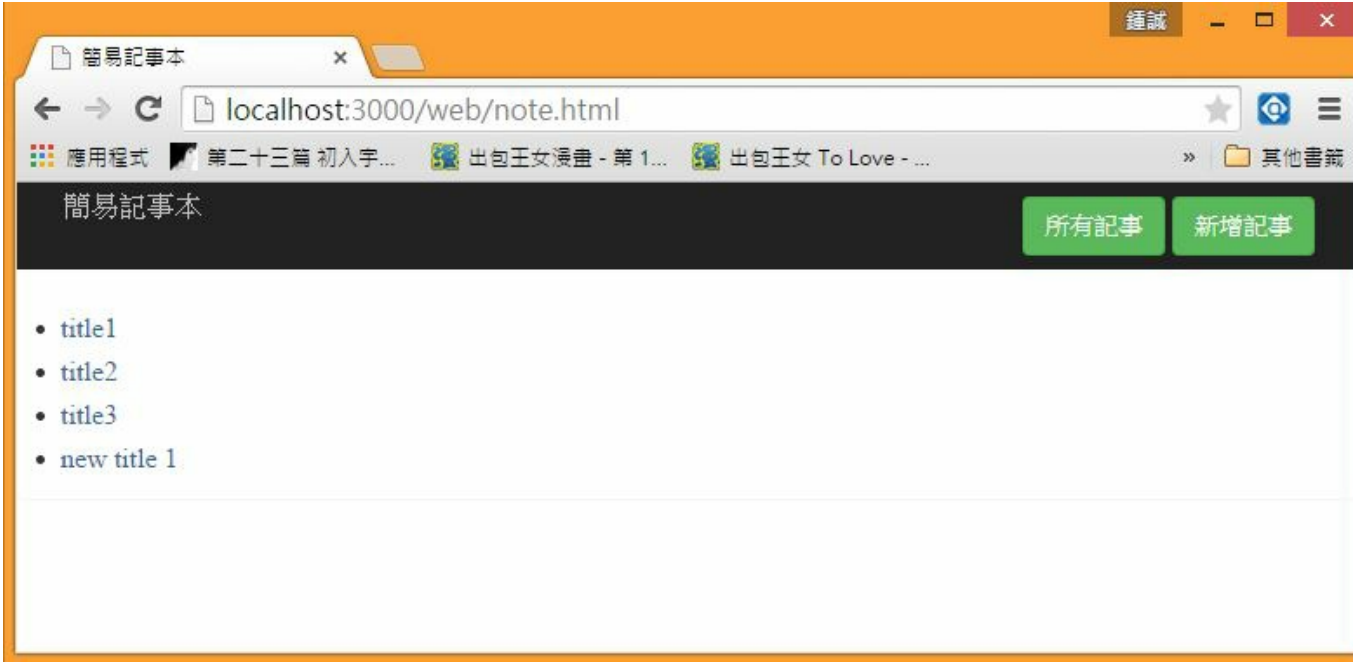
圖、編輯第二筆留言

我們也可以按『新增記事』按鈕去增加新的留言，如下所示。



圖、新增留言

新增完之後又會自動跳回列出留言的畫面如下。



圖、新增後留言列表

在以上的互動過程中，伺服器會印出下列訊息。

```
D:\git\KoaSpaNote>iojs KoaNoteServer.js
```

```
noteServer started at port : 3000
get /web/note.html
get /web/main.css
/list path=/list
/list req=[object Object]
/list notes=[{"title":"title1","body":"body1"},{"title":"title2",
"body":"body2"}
,{"title":"title3","body":"body3"}]
response: code=200
[{"title":"title1","body":"body1"},{"title":"title2","body":"body
2"},{"title":"t
itle3","body":"body3"}]

get /web/favicon.ico

  Error: ENOENT: no such file or directory, open 'D:\git\KoaSpaNo
```

```
te\web\favicon.
ico'
      at Error (native)

response: code=200
{"title":"title1","body":"body1"}

response: code=200
{"title":"title2","body":"body2"}

response: code=200
{"title":"title2","body":"body2\n\nxxx"}

post new title 1:new body 1
response: code=200
write success!
```

```
/list path=/list
/list req=[object Object]
/list notes=[{"title":"title1","body":"body1"},{"title":"title2",
"body":"body2\n
\nxxx"},{"title":"title3","body":"body3"},{"title":"new title 1",
"body":"new bod
y 1"}]
response: code=200
[{"title":"title1","body":"body1"},{"title":"title2","body":"body
2\n\nxxx"},{"ti
tle":"title3","body":"body3"},{"title":"new title 1","body":"new
body 1"}]
```

這就是整個留言系統的功能了，接著讓我們來看看前後端的程式碼。

前端網頁

檔案： note.html

```
<html>
<head>
  <meta charset="utf-8" />
  <link href="favicon.ico" rel="icon">
  <link href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/
bootstrap.min.css" rel="stylesheet">
  <link href="main.css" rel="stylesheet">
  <title>簡易記事本</title>
</head>
<body onload="init()">
  <nav class="navbar navbar-inverse navbar-fixed-top">
    <div class="container">
      <button type="button" class="navbar-toggle collapsed" data
-toggle="collapse" data-target="#navbar" aria-expanded="false" ar
```

```
ia-controls="navbar">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <div class="navbar-header" style="color:#C0C0C0">
        簡易記事本
      </div>
      <div id="navbar" class="navbar-collapse collapse">
        <form class="navbar-form navbar-right">
          <div class="form-group">
            <input id="filepath" type="hidden" class="form-contro
l" placeholder="filepath" aria-describedby="basic-addon1">
            <button class="btn btn-success" type="button" onclic
k="list()">所有記事</button>
```

```
            <button class="btn btn-success" type="button" onclic
k="newNote()">新增記事</button>
          </div>
        </form>
      </div>
    </div>
  </nav>

  <div id="panelList" class="tab-pane panel">
  </div>
  <div id="panelEdit" class="tab-pane panel">
    標題: <input id="editTitle" type="text" value="">
    <br/>
    <textarea id="editBody" class="form-control" style="width:10
0%; height:60%"></textarea>
    <button class="btn btn-success" onclick="save()">儲存</butto
```

```
n>
  </div>
  <div id="panelNew" class="tab-pane panel">
    標題： <input id="newTitle" type="text" value="">
    <br/>
    <textarea id="newBody" class="form-control" style="width:100
%; height:60%"></textarea>
    <button class="btn btn-success" onclick="add()">新增</button>

  </div>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11
.2/jquery.min.js"></script>
  <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js
/bootstrap.min.js"></script>
<script>
$('.panel').css( "display", "none");
```

```
Server = {
  timeout : 4000
};
Server.save=function(file, text) {
  $.ajax({
    type: "POST",
    url: "/note/"+file,
    timeout: this.timeout,
    data: { text: text }
  })
  .done(function(data) {
    alert( "存檔完成!");
  })
  .fail(function() {
    alert( "存檔失敗! " );
  });
```

```
}
Server.load=function(file) {
  return $.ajax({
    type: "GET",
    url: "/note/"+file,
    timeout: this.timeout,
    data: {}
  });
}
function init() {
  list();
}
function switchPanel(name) {
  $('.panel').css( "display", "none");
  $('#'+name).css( "display", "block");
}
```

```
function list() {
  switchPanel('panelList');
  $.ajax({
    type: "GET",
    url: "/list",
    timeout: this.timeout,
    data: {}
  })
  .done(function(data) {
    var notes = JSON.parse(data);
        $('#panelList').empty();
        $('#panelList').append("<ol>");
    for (var i in notes) {
      var title = notes[i].title;
      var body  = notes[i].body;
      $('#panelList').append('<li><a href="javascript:edit('+i+')
```

```
">'+title+"</a></li>")
    }
        $('#panelList').append("</ol>");
  });
}
var noteID;
function edit(id) {
  switchPanel('panelEdit');
    noteID = id;
  $.ajax({
    type: "GET",
    url: "/note/"+id,
    timeout: this.timeout,
    data: {}
  })
  .done(function(data) {
```

```
    var note = JSON.parse(data);
    $('#editTitle').val(note.title);
    $('#editBody').val(note.body);
  });
}
function save() {
  var title = $('#editTitle').val();
  var body  = $('#editBody').val();
  $.ajax({
    type: "POST",
    url: "/note/"+noteID,
    timeout: this.timeout,
    data: {
      title:title,
            body:body
    }
```

```
  })
  .done(function(data) {
      alert('存檔完成!');
    });
}
function newNote() {
  switchPanel('panelNew');
  $('#editTitle').val('');
  $('#editBody').val('');
}
function add() {
  var title = $('#newTitle').val();
  var body  = $('#newBody').val();
  $.ajax({
    type: "POST",
    url: "/new",
```

```
    timeout: this.timeout,
    data: {
      title:title,
            body:body
    }
  })
  .done(function(data) {
      alert('新增完成!');
        list();
    });
}
</script>
</body>
</html>
```

後端伺服器

檔案： KoaNoteServer.js

```
var http    = require('http');
var url     = require('url');
var koa     = require('koa');
var fs      = require('mz/fs');
var path    = require('path');
var bodyParser = require("koa-bodyparser"); // 參考: http://codef
orgeek.com/2014/09/handle-get-post-request-express-4/
var route   = require('koa-route');
var c       = console;

var app = koa();
app.use(bodyParser());

var notes = [ {title:'title1', body:'body1'},
```

```
                {title:'title2', body:'body2'},
                {title:'title3', body:'body3'} ];

function response(res, code, msg) {
  res.status = code;
  res.set({'Content-Length':''+msg.length,'Content-Type':'text/pl
ain'});
  res.body = msg;
  c.log("response: code="+code+"\n"+msg+"\n");
}

app.use(route.get('/', function*() {
  this.redirect('/web/note.html');
}));

var mime = { ".css":"text/css", ".html": "text/html", ".htm":"tex
```

```
t/html", ".jpg":"image/jpg", ".png":"image/png", ".gif":"image/gi
f", ".pdf":"application/pdf"};

function fileMimeType(path) {
  for (var tail in mime) {
    if (path.endsWith(tail))
      return mime[tail];
  }
}

app.use(route.get(/\/web\/.*/, function *toStatic() {
  var req = this.request, res = this.response;
  c.log('get %s', this.path);
  var mimetype = fileMimeType(this.path)
  if (mimetype) this.type = mimetype+";";
  this.body = fs.createReadStream(__dirname+this.path);
```

```
}));

app.use(route.get("/list", function *list() {
  c.log("/list path=%s", this.path);
  c.log("/list req=%s", this.request);
  c.log("/list notes=%j", notes);
  response(this.response, 200, JSON.stringify(notes));
}));

app.use(route.get("/note/:id", function *edit(id) {
  var i = parseInt(id);
  response(this.response, 200, JSON.stringify(notes[i]));
}));

app.use(route.post("/note/:id", function *save(id) {
  var i = parseInt(id);
```

```
  var title = this.request.body.title;
  var body = this.request.body.body;
  notes[i] = { title:title, body:body };
  response(this.response, 200, JSON.stringify(notes[i]));
}));

app.use(route.post('/new', function *newNote() {
  var req = this.request, res = this.response;
  var title = this.request.body.title;
  var body = this.request.body.body;
  c.log('post %s:%s', title, body);
  notes.push({title:title, body:body});
  response(res, 200, 'write success!');
}));

http.createServer(app.callback()).listen(3000);
```

```
c.log("noteServer started at port : 3000");
```

讓 wikidown 從 express 進化為 koa 版本

在 2015 年 3 月的時候，我們介紹了 wikidown.js 這個維基網誌系統，當時我們是採用 express 作為後端伺服器的。

但是後來筆者一邊學習 koa 套件與 yield 語法，決定一邊將 wikidown.js 改為 koa 版本，於是就將原本的 wikiServer.js 程式改用 KoaServer.js 替代。

以下是筆者的個人網站，該網站就是用 wikidown.js 架設的，如果您從以下的 http 版本進入，您將可以瀏覽，但不能編輯。

- http 版入口 -- http://ccc.nqu.edu.tw/

但是筆者個人要編輯的時候，則必須從有安全加密的 https (ssl) 版本進入，然後輸入帳號密碼之後，就可以進行編輯與瀏覽了。

- https 版入口 -- https://ccc.nqu.edu.tw/

本系統的前後端程式碼，分別列出如下：

後端程式

檔案： KoaServer.js

```
var http    = require('http');
var https   = require('https');
var c       = console;
var url     = require('url');
var fs      = require('mz/fs');
var koa     = require('koa');
var send    = require('koa-send');
var path    = require('path');
var bodyParser = require("koa-bodyparser"); // 參考: http://codef
orgeek.com/2014/09/handle-get-post-request-express-4/
var session = require('koa-session');
```

```
var serveIndex = require('koa-serve-index');
var route   = require('koa-route');
var parse   = require('co-busboy');
var saveTo  = require('save-to');
var jslib   = require('./web/jslib');
var wdlib   = require('./wdlib');
var setting = require('./setting');
var passwords = setting.passwords;

var app = koa();
var webDir = path.join(__dirname, 'web');
var dbRoot = path.join(__dirname, 'db');

c.log("dbRoot=%s", dbRoot);

app.keys = ['xxxxxxxxxxxxxxxxx']; // 使用時請改掉
```

```
app.use(session(app));
app.use(bodyParser({formLimit:5*1000*1000, jsonLimit:5*1000*1000}
));

function response(res, code, msg) {
  res.status = code;
  res.set({'Content-Length':''+msg.length,'Content-Type':'text/pl
ain'});
  res.body = msg;
  c.log("response: code="+code+"\n"+msg+"\n");
}

function isPass(req) {
  c.log("loginToSave="+setting.loginToSave);
  if (setting.loginToSave === false)
    return true;
```

```
  c.log('req.session.user='+req.session.user);
  return typeof(req.session.user)!=='undefined';
}

// 處理檔案上傳 : multipart upload
app.use(function *(next){
  var domain = this.request.url.split("/").pop();
  if ('POST' !== this.request.method) return yield next;
  if (!this.request.url.startsWith('/upload/')) return yield ne
xt;
  if (!this.request.header["content-type"].startsWith("multipart/
form-data;")) return yield next;
  var part, parts = parse(this);
  var files = [], file;
  while (part = yield parts) {
    console.log('part=%j', part);
```

```
    if (typeof part.filename !== 'undefined') {
      files.push(file = path.join(dbRoot, domain, part.filename))
;
      console.log('uploading %s -> %s', part.filename, file);
      yield saveTo(part, file);
    }
  }
  this.body = files;
});

app.use(route.get('/', function*() {
  this.redirect('/web/wikidown.html');
}));

var mime = { ".html": "text/html", ".htm":"text/html", ".jpg":"im
age/jpg", ".png":"image/png", ".gif":"image/gif", ".pdf":"applica
```

```
tion/pdf"};

function fileMimeType(path) {
  for (var tail in mime) {
    if (path.endsWith(tail))
      return mime[tail];
  }
}

app.use(route.get(/.*/, function *toStatic() {
    if (!this.path.startsWith("/setting.js")) {
    var mimetype = fileMimeType(this.path)
    if (mimetype) this.type = mimetype+";";
    c.log('get %s', this.path);
    this.body = fs.createReadStream(__dirname+this.path);
    }
```

```
}));

app.use(route.post('/wd/:domain/:file', function*(domain, file)
{
  var req = this.request, res = this.response;
  var obj = this.request.body.obj;
  if (!isPass(this)) {
    response(res, 401, 'Please login to save!')
    return;
  }
  c.log('post %s:%s', domain, file)
  // 寫入檔案 （通常是 *.wd 檔案）
  yield fs.mkdir(dbRoot+"/"+domain+"/").catch(function(){});
  yield fs.writeFile(dbRoot+"/"+domain+"/"+file, obj).then(funct
ion() {
    response(res, 200, 'write success!');
```

```
  }).catch(function() {
    response(res, 403, 'write fail!'); // 403: Forbidden
  });  // 將 *.wd 轉為 *.html 後寫入
  if (jslib.endsWith(file, '.wd')) {
    var html = wdlib.toHtml(obj, domain);
    var htmFile = file.replace(/\.wd$/, '.html');
    yield fs.writeFile(dbRoot+"/"+domain+"/"+htmFile, html);

  }
}));

// 以下為 https 版本，必較不容易被竊取密碼，但是有 self signed 的認證問題
// 目前設定只有 https 版本可以登入並修改資料，http 版不行。
// var secureApp = app;
```

```
var secureApp = app;

secureApp.use(route.post("/login", function*() {
  var req = this.request, res = this.response;
  var p = this.request.body;
  if (req.protocol !== 'https') {
    response(res, 401, p.user+":login fail!");
    return;
  }
  if (passwords[p.user] === p.password) {
    this.session.user = p.user;
    response(res, 200, p.user+":login success!");
  } else {
    response(res, 401, p.user+":login fail!");
  }
}));
```

```
secureApp.use(route.post("/logout", function*() {
  var req = this.request, res = this.response;
  this.session = null;
  response(res, 200, "logout success!");
}));

var port = process.env.PORT || 80; // process.env.PORT for Heroku

console.log('Server started: http://localhost:'+port);

http.createServer(app.callback()).listen(port);

var sslPort = 443;
https.createServer({
      key: fs.readFileSync('key.pem'),
```

```
        cert: fs.readFileSync('cert.pem'),
        // 以下只有 self signed 認證的方式需要
        requestCert: true,
        ca: [ fs.readFileSync('csr.pem') ]
}, app.callback()).listen(sslPort);

console.log('Http Server started: http://localhost:'+sslPort);
```

前端程式

檔案： wikidown.js

```
<html>
<head>
  <meta charset="utf-8" />
  <link href="favicon.ico" rel="icon">
  <link href="css/bootstrap.min.css" rel="stylesheet">
```

```
  <link href="css/highlight.default.min.css" rel="stylesheet">
  <link href="css/fileinput.css" media="all" rel="stylesheet"/>
  <link href="wikidown.css" rel="stylesheet">
  <title>Wikidown</title>
</head>
<body onload="init()">
  <nav class="navbar navbar-inverse navbar-fixed-top">
    <div class="container">
      <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#navbar" aria-expanded="false" ari
a-controls="navbar">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
```

```
      <div class="navbar-header" style="color:#C0C0C0">
<!--        <a class="navbar-brand" href="#main:home">
          <span class="glyphicon glyphicon-home"></span>  
        </a> -->
                <span id="titleHome" class="navbar-nav titleLink"
><p><a href="#main:home" class="innerLink">首頁</a> /</p></span>
        <span id="titleHead" class="navbar-nav titleLink"></span>
      </div>
      <div id="navbar" class="navbar-collapse collapse">
        <ul class="nav navbar-nav navbar-right">
          <li class="dropdown">
            <a class="dropdown-toggle" data-toggle="dropdown"><sp
an class="glyphicon glyphicon-globe"></span><!--<span class="care
t">--></span></a>
            <ul class="dropdown-menu" role="menu">
              <li><a onclick="Mt.switchLang('us')">English</a></l
```

```
i>
            <li><a onclick="Mt.switchLang('tw')">中文</a></li>
          </ul>
        </li>
      </ul>
      <ul class="nav navbar-nav navbar-right hide" id="menuLogi
n">
        <li class="dropdown">
          <a class="dropdown-toggle" data-toggle="dropdown">登
入</a>
          <ul class="dropdown-menu" role="menu">
            <li><a onclick="login()">登入</a></li>
            <li><a onclick="logout()">登出</a></li>
          </ul>
        </li>
      </ul>
```

```
        <ul class="nav navbar-nav navbar-right">
          <li class="dropdown">
            <a class="dropdown-toggle" data-toggle="dropdown">分
享</a>
            <ul class="dropdown-menu" role="menu">
              <li><a onclick="facebookShare()">Facebook</a></li>
              <li><a onclick="staticShare()">靜態連結</a></li>
            </ul>
          </li>
        </ul>
        <form class="navbar-form navbar-right hide" id="menuEdit"
>
          <div class="form-group">
            <input id="filepath" type="hidden" class="form-contro
l" placeholder="filepath" aria-describedby="basic-addon1">
            <button class="btn btn-success" type="button" onclick=
```

```
"show()">預覽</button>
            <button class="btn btn-success" type="button" onclick
="edit()">編輯</button>
            <button class="btn btn-success" type="button" onclick=
"save()">儲存</button>
            <button class="btn btn-success" type="button" onclick
="upload()">上傳</button>
<!--        <button class="btn btn-success" type="button" onclick=
"share()">分享</button>-->
          </div>
        </form>
      </div>
    </div>
  </nav>

  <div id="panelShow" class="tab-pane panel"> <!-- 預覽 -->
```

```
    <div id="htmlBox" style="overflow-y:scroll;height:90%;padding
:10px;"></div>
  </div>
  <div id="panelEdit" class="tab-pane panel"> <!-- 編輯 -->
    <br/>
    <textarea id="wdBox" class="form-control" style="width:100%;
height:80%"></textarea>
  </div>
  <div id="panelUpload" class="tab-pane panel" style="height:75%"
>
    <center>
    <div class="kv-main">
      <form enctype="multipart/form-data">
        <div class="form-group">
      <input id="imageUpload" name="imageUpload" type="file" mult
iple=true class="file-loading">
```

```
        </div>
      </form>
    </div>
    </center>
  </div>
  <div id="panelLogin" class="tab-pane panel"> <!-- 登入 -->
    <form class="form-signin" role="form">
      <input type="text" id="loginUser" class="form-control" requ
ired autofocus data-mt="User" placeholder="帳號"/>
      <input type="password" id="loginPassword" class="form-contr
ol" required data-mt="Password" placeholder="密碼"/>
      <button class="btn btn-lg btn-primary btn-block" type="butt
on" data-mt="Login" onclick="Server.login()">登入</button>
    </form>
  </div>
  <script src="js/jquery.min.js"></script>
```

```
  <script src="js/bootstrap.min.js"></script>
<script>
$('.panel').css( "display", "none");

var domain, file, converter; // filepath,
var wdNewFile = '# 標題：文件不存在\n\n您可以編輯後存檔！\n## 語法
\n* [內部連結](innerLink.html)\n* [外部連結](link)';

function isLogin() {
  if (localStorage.wd_login !== "true") { // 注意：sessionStorage
 不能跨頁面持續，所以得用 localStorage
    alert('未登入無法存檔上傳，請先登入!');
    login();
    return false;
  }
  return true;
```

```
}

Server = {
  timeout : 4000
};

Server.save=function(domain, file, doc) {
  $.ajax({
    type: "POST",
    url: "/wd/"+domain+"/"+file+".wd",
    timeout: this.timeout,
    data: { obj: doc },
        statusCode: {
      401: function() { // 401:Unauthorized
        localStorage.wd_login = "false";
        isLogin();
```

```
      }
    }
  })
  .done(function(data) {
    alert( "存檔完成!");
  })
  .fail(function() {
    alert( "存檔失敗! " );
  });
}

Server.load=function(domain, file) {
  return $.ajax({
    type: "GET",
    url: "../db/"+domain+"/"+file+".wd",
    timeout: this.timeout,
```

```
    data: {}
  });
}

Server.login=function() {
  $.ajax({
    type: "POST",
    url: "/login",
    timeout: this.timeout,
    data: { user:$('#loginUser').val(), password:$('#loginPasswor
d').val() },
  })
  .done(function(data) {
    localStorage.wd_login = "true";
    alert( "登入成功!");
    $('#loginPassword').val('');
```

```
    edit();
  })
  .fail(function() {
    localStorage.wd_login = "false";
    alert( "登入失敗！" );
    login();
  });
}

Server.logout=function() {
  $.ajax({
    type: "POST",
    url: "/logout",
    timeout: this.timeout,
    data: {},
  })
```

```
  .done(function(data) {
    localStorage.wd_login = "false";
    alert( "登出成功!");
    show();
  })
  .fail(function() {
    alert( "登出失敗! " );
  });
}

window.onhashchange = function () {
  filepath = window.location.hash.substring(1);
  var parts  = filepath.split(':');
  domain = parts[0], file=parts[1];
  loadFile(domain, file);
  show();
```

```
}

window.onbeforeunload = function(){}

function md2html(md) {
  return converter.makeHtml(md);
}

function init() {
  converter = new Showdown.converter({ extensions: ['mathjax', 't
able'] });
  if (window.location.hash === '')
    window.location.hash = '#main:home';
  window.onhashchange();
  if (window.location.protocol === 'https:') {
    // $('#navbar').css('display', 'none');
```

```
    $('#menuLogin').removeClass('hide');
    $('#menuEdit').removeClass('hide');
  }
  MathJax.Hub.Config({
      extensions: ["tex2jax.js"],
      jax: ["input/TeX", "output/HTML-CSS"],
            displayAlign: "left",
      tex2jax: {
        inlineMath: [ ['$','$'], ["\\(","\\)"] ],
        displayMath: [ ['$$','$$'], ["\\[","\\]"] ],
        processEscapes: true
      },
      "HTML-CSS": { availableFonts: ["TeX"], scale: 130 }
    });
}
```

```
function switchPanel(name) {
  $('.panel').css( "display", "none");
  $('#'+name).css( "display", "block");
}

function templateApply(wd) {
  var templateNow = config.template[domain];
  if (templateNow===undefined)
    templateNow = config.template['default'];
  return templateNow.split('<%=wd%>').join(wd); // 不能用 replace(reg, str), 因為 str 中可能有參數 $...$ 符號, 所以改用 split + join
}

function wdToHtml(wd, domain) {
  wd  = '\n'+wd+' ';
```

```
  wd  = wd.replace(/(\s)@\[\[([^\]]*?)\]\]\((.*?)\)/gi, '$1<a href="../db/'+domain+'/$3" class="innerLink">$2</a>'); // 內部檔案 [text](pathToFile.html)
  wd  = wd.replace(/(\s)@\[\[([^\]]*?)\]\]/gi, '$1<a href="../db/'+domain+'/$2" class="innerLink">$2</a>'); // 內部檔案 [pathToFile](pathToFile.html)
  wd  = wd.replace(/(\s)\!\[\[([^\]]*?)\]\]\((.*?)\)/gi, '$1<div class="figure"><img src="../db/'+domain+'/$3"/><p class="caption">$2</p></div>'); // 內部圖片 ![text](file)
  wd  = wd.replace(/(\s)\[\[([^\]]+?)\]\]\(([^:\)]+):([^:\)]+)\)/gi, '$1<a href="#$3:$4" class="innerLink">$2</a>'); // 內部連結 [text](../domain/file.html)
  wd  = wd.replace(/(\s)\[\[([^\]]+?)\]\]\((.*?)\)/gi, '$1<a href="#'+domain+':$3" class="innerLink">$2</a>'); // 內部連結 [text](file.html)
  wd  = wd.replace(/(\s)\[\[([^\]:]+):([^\]:]+)\]\]/gi, '$1<a hre
```

```
f="#$2:$3" class="innerLink">$2:$3</a>'); // 內部連結 [[domain:file]]
  wd  = wd.replace(/(\s)\[\[([^\]:]+?)\]\]/gi, '$1<a href="#'+domain+':$2" class="innerLink">$2</a>'); // 內部連結 [file](file.html)
//  wd  = wd.replace(/(\s)\$([^$\n]+?)\$/gi, '$1\n<script type="math/tex">$2</'+'script>\n');// 數學式 $$[latex]$$,  刻意把 '</'+'script>' 分開, 避免瀏覽器認為是真的 script 區塊
  return md2html(wd);
}

function login() {
  switchPanel('panelLogin');
}

function logout() {
```

```
  Server.logout();
}

function edit() {
  switchPanel('panelEdit');
}

function upload() {
  if (!isLogin()) return;
  switchPanel('panelUpload');
  $("#imageUpload").fileinput({
    uploadUrl: "/upload/"+domain,
    maxFileCount: 10,
    uploadAsync: false,
    uploadExtraData: {
        domain: domain,
```

```
        file: file
    }
  });
}

function absolute(base, relative) {
    var stack = base.split("/"),
        parts = relative.split("/");
    stack.pop(); // remove current file name (or empty string)
                 // (omit if "base" is the current folder without
 trailing slash)
    for (var i=0; i<parts.length; i++) {
        if (parts[i] == ".")
            continue;
        if (parts[i] == "..")
            stack.pop();
```

```
        else
            stack.push(parts[i]);
    }
    return stack.join("/");
}

function httpRef() {
  return window.location.href.replace('https:', 'http:');
}

function facebookShare() {
  window.open("https://www.facebook.com/sharer/sharer.php?u="+esc
ape(absolute(httpRef(), '../db/'+domain+'/'+file+'.html'))+'&t='+
document.title, '', 'menubar=no,toolbar=no,resizable=yes,scrollba
rs=yes,height=300,width=600')
}
```

```
function staticShare() {
//  window.open('../db/'+domain+'/'+file+'.html', '', 'menubar=no
,toolbar=no,resizable=yes,scrollbars=yes,height=300,width=600')
  window.open(absolute(httpRef(), '../db/'+domain+'/'+file+'.html'
), '', 'menubar=no,toolbar=no,resizable=yes,scrollbars=yes,height
=300,width=600')
}

function show() {
  var wd = $('#wdBox').val();
  wd  = templateApply(wd);
  var html = wdToHtml(wd, domain);
  $('#htmlBox').html(html);
  $('pre code').each(function(i, block) {
    hljs.highlightBlock(block);
```

```
  });
  switchPanel('panelShow');
  if (typeof(MathJax) !== 'undefined') {
    MathJax.Hub.Queue(["Typeset",MathJax.Hub, "htmlBox"]);
    }
  showTitle();
}

function showTitleHead(domain) {
  var titleHead = config.title['default'];
  if (config.title[domain] !== undefined)
    titleHead = config.title[domain];
  var titleHtml = wdToHtml(titleHead, domain);
  $('#titleHead').html(titleHtml);
}
```

```
function showTitle() {
  if ($('h1').length > 0)
    $('title').html($('#titleHead').text()+' -- '+$('h1')[0].inne
rText);
  else if ($('h2').length>0)
    $('title').html($('#titleHead').text()+' -- '+$('h2')[0].inne
rText);
}

function loadFile(domain, file) {
  showTitleHead(domain);
  window.location.hash = '#'+domain+':'+file;
  Server.load(domain, file)
  .done(function(wd) {
    $('#wdBox').val(wd);
    show();
```

```
  })
  .fail(function() {
    $('#wdBox').val(wdNewFile);
    show();
  });
}

function save() {
  if (!isLogin()) return;
  var wd = $('#wdBox').val();
  Server.save(domain, file, wd);
}
/*
// 以下這種作法是為了加快速度，避免一開始載入 MathJax 花太久時間
...
// 刻意把 '</'+'script>' 分開，避免瀏覽器認為是真的 script 區塊
```

```
$( document ).ready(function() {
    $('body').append('<'+'script src="http://cdn.mathjax.org/mathjax/latest/MathJax.js?config=TeX-AMS-MML_SVG"></'+'script>');
});
*/
</script>

<script src="js/Showdown/Showdown.js"></script>
<script src="js/Showdown/extensions/table.min.js"></script>
<script src="js/Showdown/extensions/mathjax.js"></script>
<script src="js/highlight.min.js"></script>
<script src="js/fileinput.js" type="text/javascript"></script>
<script src="config.js"></script>
<script src="MathJax/MathJax.js?config=TeX-AMS-MML_SVG"></script>
</body>
</html>
```

結語

這個程式改成 koa 之後，所有的 callback 幾乎都不見了，而這也正是 koa 框架從 yield 語法上得到的優點阿！

雜誌訊息

讀者訂閱

程式人雜誌是一個結合「開放原始碼與公益捐款活動」的雜誌，簡稱「開放公益雜誌」。開放公益雜誌本著「讀書做善事、寫書做公益」的精神，我們非常歡迎程式人認養專欄、或者捐出您的網誌，如果您願意成為本雜誌的專欄作家，請加入 程式人雜誌社團 一同共襄盛舉。

我們透過發行這本雜誌，希望讓大家可以讀到想讀的書，學到想學的技術，同時也讓寫作的朋友的作品能產生良好價值 － 那就是讓讀者根據雜誌的價值捐款給慈善團體。 讀雜誌做公益也不需要有壓力，您不需要每讀一本就急著去捐款，您可以讀了十本再捐，或者使用固定的月捐款方式，當成是雜誌訂閱費，或者是季捐款、一年捐一次等都 OK！甚至是單純當個讀者我們也都很歡迎！

本雜誌每期參考價：NT 50 元，如果您喜歡本雜誌，請將書款捐贈公益團體。例如可捐贈給「羅慧夫顱顏基金會 彰化銀行(009) 帳號：5234-01-41778-800」。(若匯款要加註可用「程式人雜誌」五個字)

投稿須知

給專欄寫作者： 做公益不需要有壓力。如果您願意撰寫專欄，您可以輕鬆的寫，如果當月的稿件出不來，我們會安排其他稿件上場。

給網誌捐贈者： 如果您沒時間寫專欄或投稿，沒關係，只要將您的網誌以 [創作共用的「姓名標示、非商業性、相同方式分享」授權] 並通知我們，我們會自動從中選取需要的文章進行編輯，放入適當的雜誌當中出刊。

給文章投稿者： 程式人雜誌非常歡迎您加入作者的行列，如果您想撰寫任何文章或投稿，請用 markdown 或 LibreOffice 編輯好您的稿件，並於每個月 25 日前投稿到程式人雜誌社團 的檔案區，我們會盡可能將稿件編入隔月1號出版程式人雜誌當中，也歡迎您到社團中與我們一同討論。

如果您要投稿給程式人雜誌，我們最希望的格式是採用 markdown 的格式撰寫，然後將所有檔按壓縮為 zip 上傳到社團檔案區給我們， 如您想學習 markdown 的撰寫出版方式，可以參考 [看影片學 markdown 編輯出版流程] 一文。

如果您無法採用 markdown 的方式撰寫，也可以直接給我們您的稿件，像是 MS. Word 的 doc 檔或 LibreOffice 的 odt 檔都可以，我們 會將這些稿件改寫為 markdown 之後編入雜誌當中。

參與編輯

您也可以擔任程式人雜誌的編輯，甚至創造一個全新的公益雜誌，我們誠摯的邀請您加入「開放公益出版」的行列，如果您想擔任編輯或創造新雜誌，也歡迎到 程式人雜誌社團 來與我們討論相關事宜。

公益資訊

公益團體	聯絡資訊	服務對象	捐款帳號
財團法人羅慧夫顱顏基金會	http://www.nncf.org/ lynn@nncf.org 02-27190408分機232	顱顏患者 (如唇顎裂、小耳症或其他罕見顱顏缺陷）	銀行：009 彰化銀行民生分行 帳號：5234-01-41778-800
社團法人台灣省兒	http://www.cyga.org/ cyga99@gmail.com	單親、隔代教養.弱勢及一	銀行：新光銀行 戶名：台灣省兒童少年成長協會

童少年成長協會	04-23058005	般家庭之兒童青少年	帳號：103-0912-10-000212-0