

# NODE SERVER

劉治廷

# 前言

# 課程目標

- 安裝Node.js
- 安裝Sublime
- 基本Node.js語法
- 使用Node.js + Express架設自己的API Server
- 建立Java應用程式呼叫自建API

# 範例下載處

- <https://goo.gl/VKnHgg>
- 可以使用Clone or Download中Download Zip下載
- 建議：使用git抓取  
git clone  
[https://github.com/silencecork/shuworkshop\\_late2017.git](https://github.com/silencecork/shuworkshop_late2017.git)

Node.js

安裝NODEJS與SUBLIME

# 什麼是NODE.JS

- Node.js是2009年推出
- 使用JavaScript撰寫Server後端
- 基於Google推出的v8引擎
- 為單一執行緒，環境配置容易
- 事件驅動，且使用**非同步方式**撰寫程式
- 只需要安裝Node.js，使用筆記本寫程式 (<http://nodejs.org/>)

簡單建立一個WebServer的範例

```
var express = require('express');
var app = express();
app.get('/', function(req, res){
  res.end('<html><body><H1>Hello World</H1></body></html>');
});
app.listen(5000);
```

# 安裝NODE.JS

- 官方網站 <https://nodejs.org/en/>
- 本次課程使用Node版本為6.x版
  - 下載點 <https://nodejs.org/download/release/v6.11.2/>
- 安裝後請記得安裝的位置，與接下來要安裝的Sublime整合即可邊開發邊看執行結果

# 安裝SUBLIME

- Sublime是一款文字編輯軟體，可以整合多樣編譯系統，加上易用的操作方式，是非常適合用來寫直譯型程式的工具
- 官方網站
  - <https://www.sublimetext.com/>
- 安裝完後，依照以下連結說明結合Node.js編譯系統
  - <https://ithelp.ithome.com.tw/articles/10184828>



Node.js

# NODE.JS基本教學

# NODE.JS基本教學

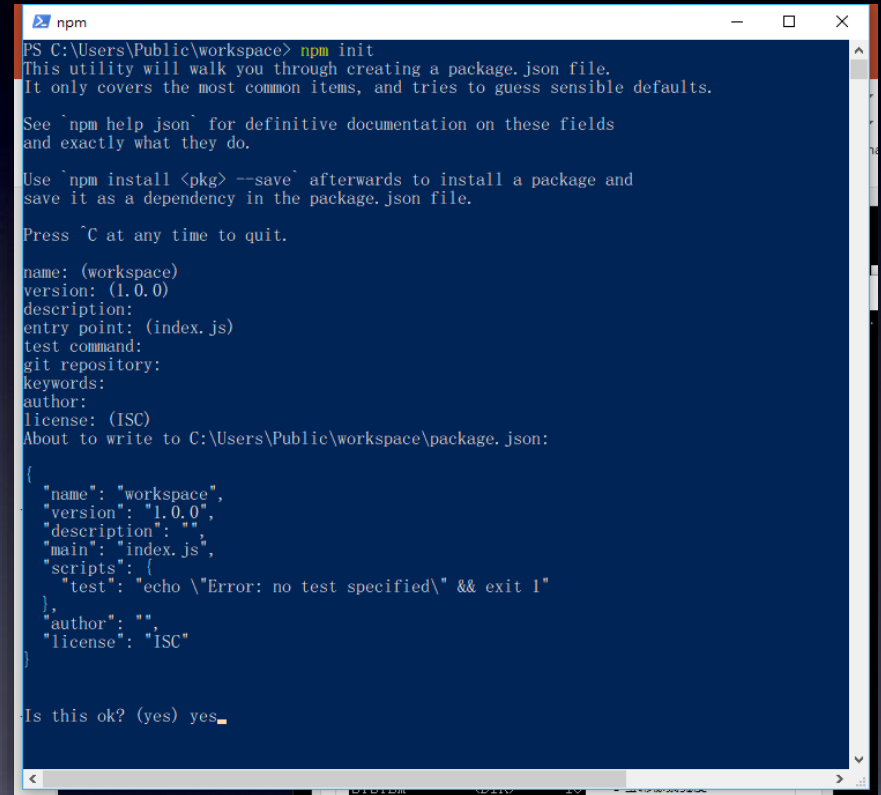
- <http://www.codedata.com.tw/javascript/using-nodejs-to-learn-javascript>
- <http://www.codedata.com.tw/javascript/using-nodejs-to-learn-javascript-2-control-flow/>
- <http://www.codedata.com.tw/javascript/using-nodejs-to-learn-javascript-3-function-parameter-closure/>

Node.js

建立NODE.JS專案

# 建立NODE.JS專案

- 假設專案建立在  
C:\Users\Public\work  
space下
- 1. Windows，打開  
CMD；Mac，打開  
Console
- 2. 使用CD指令切換到  
該目錄下
- 3. 使用npm init啟動專  
案建立器



```
PS C:\Users\Public\workspace> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.

name: (workspace)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\Public\workspace\package.json:

{
  "name": "workspace",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this ok? (yes) yes_
```

Node.js + NPM + Express

使用NPM安裝EXPRESS

# 什麼是NPM

- Node Package Manager
- Node.js預設的套件管理器，可以使用簡單的指令即可安裝他人的套件在自己的專案中
  - `npm install <套件名稱> --save`
- 更詳細內容可以參考
  - [https://chenyiqiao.gitbooks.io/documentation\\_for\\_npm/content/what\\_is\\_npm.html](https://chenyiqiao.gitbooks.io/documentation_for_npm/content/what_is_npm.html)

# 什麼是EXPRESS

- Node.js上可以幫助快速建立Server的套件
- 指令 `npm install express --save`
- Express介紹
  - <http://expressjs.com/zh-tw/>
- 使用Node.js，是直接與JSON相容，使得API回傳JSON非常容易

對應專案

02\_NodeJSServer/node\_server


Node.js

EXPRESS SERVER



# EXPRESS SERVER

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



使用模組express

```
app.get('/', function(request, response) {  
    response.status(200).send(' Hello World');  
    response.end();  
});
```

```
app.get('/api/test', function(request, response) {  
    var ret = {  
        msg : 'Hello World',  
        status : 0  
    }  
    response.status(200).send(JSON.stringify(ret));  
    response.end();  
});
```

```
app.listen(5000);
```

# NODE.JS伺服器

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

建立express實體

```
app.get('/', function(request, response) {  
    response.status(200).send(' Hello World');  
    response.end();  
});
```

```
app.get('/api/test', function(request, response) {  
    var ret = {  
        msg : 'Hello World',  
        status : 0  
    }  
    response.status(200).send(JSON.stringify(ret));  
    response.end();  
});
```

```
app.listen(5000);
```

# NODE.JS伺服器

```
var express = require('express');  
var app = express();  
  
app.get('/', function(request, response) {  
    response.status(200).send(' Hello World');  
    response.end();  
});  
  
app.get('/api/test', function(request, response) {  
    var ret = {  
        msg : 'Hello World',  
        status : 0  
    }  
    response.status(200).send(JSON.stringify(ret));  
    response.end();  
});  
  
app.listen(5000);
```



啟動伺服器，聆聽port 5000


# NODE.JS伺服器

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

```
app.get('/', function(request, response) {  
    response.status(200).send(' Hello World');  
    response.end();  
});
```

```
app.get('/api/test', function(request, response) {  
    var ret = {  
        msg : 'Hello World',  
        status : 0  
    }  
    response.status(200).send(JSON.stringify(ret));  
    response.end();  
});
```

```
app.listen(5000);
```



若request的網址是主機  
要處理的邏輯

# NODE.JS伺服器

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

app.get('/', function(request, response) {
    response.status(200).send(' Hello World');
    response.end();
});

app.get('/api/test', function(request, response) {
    var ret = {
        msg : 'Hello World',
        status : 0
    }
    response.status(200).send(JSON.stringify(ret));
    response.end();
});

app.listen(5000);
```

若request的網址是主機/api/test  
要處理的邏輯

# NODE.JS伺服器

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

app.get('/', function(request, response) {
    response.status(200).send(' Hello World');
    response.end();
});

app.get('/api/test', function(request, response) {
    var ret = {
        msg : 'Hello World',
        status : 0
    }
    response.status(200).send(JSON.stringify(ret));
    response.end();
});

app.listen(5000);
```

回覆request字串  
內容

# NODE.JS伺服器

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

app.get('/', function(request, response) {
    response.status(200).send(' Hello World');
    response.end();
});

app.get('/api/test', function(request, response) {
    var ret = {
        msg : 'Hello World',
        status : 0
    }
    response.status(200).send(JSON.stringify(ret));
    response.end();
});

app.listen(5000);
```



回覆request json內容  
為一個JSONObject

Java + Node.js

# JAVA呼叫自建的API



Homework

作業

# 作業說明

- 自己使用Express建立API Server
- 提供三種API
  - /api/single -> 回傳JSON Object
  - /api/list -> 回傳JSON Array
  - /api/text -> 回傳字串
- 自己撰寫Java應用程式呼叫自建API
  - 進入應用程式讓用戶選擇要呼叫哪種API?
  - 記得物件化結果
  - 記得/api/list的結果要用ArrayList來儲存
  - 最後列印出呼叫API的結果

Q & A