



執行期的專案監看



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

- 7-1: 使用 Nodemon
- 7-2: 設定 Nodemon
- 7-3: PM2 介紹



7-1

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

我們即將一起學習：

- 什麼是Nodemon
- Nodemon原理
- Nodemon優缺點
- 安裝 Nodemon
- Nodemon重要參數介紹
- 透過模組方式使用Nodemon
- 透過child_process方式使用Nodemon
- 模組方式與child_process方式的差異
- PM2管理監視程式運行的工具
- PM2 的 cluster、fork 模式
- PM2安裝與使用
- PM2參數設定

【Key Points】：

- 7-1: 使用 Nodemon
- 7-2: 設定 Nodemon
- 7-3: PM2 介紹

7-1：使用 Nodemon

- 什麼是Nodemon
- Nodemon原理
- Nodemon優缺點
- 安裝



designed by freepik



designed by freepik

7-2



- Nodemon透過對module做cache，cache與新的module差異做比對。
- Node.js本身就有做cache，Nodemon則是可以選擇要監視的檔案做cache，彈性較大。
- 由於是把module放到cache，只需要更新該cache部分並立即重新啟動。
- Github上有Gulp與Grunt整合Nodemon的套件，讓開發與部署流程的自動化更進步。
- Nodemon大量使用child process以及cache，這些都代表著需要更大量的記憶體。

【Key Points】：

什麼是Nodemon

Nodemon原理

Nodemon優缺點

什麼是Nodemon

- 監視程式碼。
- 热更新服務。
- 可以忽略指定檔案或目錄。
- 有支援TypeScript
- 可以撰寫config file

7-3



- Nodemon透過對module做cache，cache與新的module差異做比對。
- 有差異就自動real time地重啟服務。
- 有些檔案如node_modules，非關開發的部分，都可以直接忽略。
- TypeScript漸漸被採用，這也讓nodemon開始支援TypeScript。
- 透過config file的設定，讓工程師可以專注在開發上，無須改動任何nodemon程式碼，也無須在command line輸入各種參數。

【Key Points】：

監視程式碼: Nodemon透過對module做cache，cache與新的module差異做比對

熱更新服務: 有差異就自動real time地重啟服務

透過config file的設定，讓工程師可以專注在開發上，無須在command line輸入各種參數

Nodemon原理

流程如下：

- 對module做cache。
- 如果是原生module就呼叫NativeModule.require()。
- 如果是不存在於cache中的module就新增module並做cache。
- 重新啟動服務。
- 以上可以看出記憶體使用會增加不少。

7-4



- Nodemon可以選擇要監視的檔案做cache，彈性較大。
- 原生module如fs等，Nodemon依然會採用Node.js的原生API NativeModule.require()做加載module。
- 不存在於cache的module，代表是程式碼有更動過，於是會新增module，並做cache。
- Nodemon會需要重新啟動服務是因為Node.js並沒有更新module，Nodemon代為重啟服務以讓新module生效。
- Nodemon大量使用child process以及cache，這些都代表著需要更大量的記憶體。

【Key Points】：

對module做cache

如果是原生module就呼叫NativeModule.require()

如果是不存在於cache中的module就新增module並做cache

Nodemon優缺點

優點：

- 加快開發速度
- 選擇性監視程式碼更動
- 許多自動化工具都有整合Nodemon

缺點：

- 記憶體用量加大
- child_process使用不當會影響開發者

7-5



- 由於是把module放到cache，只需要更新該cache部分並立即重新啟動。
- 選擇性監視更新帶來更高的彈性。
- Github上有Gulp與Grunt整合Nodemon的套件，讓開發流程的自動化更進步。
- 記憶體使用量大，會讓開發者在開發過程中較難發現自己程式碼真實記憶體用量。
- child_process issue曾在網路上引起大量討論。<https://medium.com/@will.wang/nodejs-and-nodemon-blackhole-2651c7f8307e>

【Key Points】：

加快開發速度

選擇性監視程式碼更動

許多自動化工具都有整合Nodemon

安裝 Nodemon

- 全域安裝
- 安裝於專案
- command line 啟動
- package.json 設定
- 設定config file

7-6



- npm install nodemon -g
- npm install --save-dev nodemon
- nodemon app.js
- 可以在package.json新增“nodemonConfig”: {"watch": ["*.js"], "ignore": ["node_modules"]}
- 可以選擇新增nodemon.json · {"watch": ["*.js"], "ignore": ["node_modules"]}

【Key Points】：

全域安裝: npm install nodemon -g

安裝於專案: npm install --save-dev nodemon

在package.json新增 nodemonConfig

7-2：設定 Nodemon

- 重要參數介紹
- 透過模組方式使用Nodemon
- 透過child_process方式使用Nodemon
- 模組方式與child_process方式的差異



designed by freepik



designed by freepik

7-7

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

- Nodemon不只可以透過command line啟動也可以導入模組啟動。
- 可以透過spawn、fork方式產生child_process，而child_process正是Nodemon的底層運作方式。
- 模組化仍然是透過Node.js一條process運行，而child_process則可以有類似multiple thread的效果。(但與多執行緒仍舊不同)
- 模組化的封閉原則讓程式更好管理，而child_process是導入原生模組，依賴原生模組，相依性增加，管理難度提高。
- 程式模組化後擴展性本就會提高許多，而child_process方式有send()可以與Nodemon互動，在擴展性方面差異不大。

【Key Points】：

重要參數介紹

透過模組方式使用Nodemon

透過child_process方式使用Nodemon

Nodemon重要參數介紹

- **ignore** 可以指定忽略檔案、資料夾的監視更新
- **execMap** 使用這個參數支援其他語言
- **watch** 可以指定要監視的檔案、資料夾
- **event** Nodemon有許多的監聽事件可以呼叫
- **env** 透過Nodemon傳遞環境變數

7-8



- ignore 可以指定忽略檔案、資料夾的監視更新。
- execMap 由於Nodemon不只支援Node.js還有其他語言，所以可以使用這個參數支援其他語言。
- watch 可以指定要監視的檔案、資料夾。
- event Nodemon有許多的監聽事件可以呼叫。
- env 由於是透過Nodemon來執行應用程式，所以也可以透過Nodemon傳遞環境變數。

【Key Points】：

ignore: 可以指定忽略檔案、資料夾的監視更新

watch: 可以指定要監視的檔案、資料夾

env: 透過Nodemon傳遞環境變數

透過模組方式使用Nodemon

```
const nodemon = require('nodemon');

nodemon({
  script: 'app.js',
  ext: 'js json'
});

nodemon.on('start', function () {
  console.log('App has started');
}).on('quit', function () {
  console.log('App has quit');
  process.exit();
}).on('restart', function (files) {
  console.log('App restarted due to: ', files);
});
```

7-9



- Nodemon不只可以透過command line啟動也可以導入模組啟動。
- 模組化可以增加程式架構的彈性。
- 可以看出Nodemon是透過外部傳參的方式執行nodemon()，外部參數決定程式行為是好設計。
- Nodemon本身也提供很多監聽事件。
- 這些監聽事件也是跟Node.js的非同步API設計一樣，透過callback function回傳，不影響原程式執行。

【Key Points】：

Nodemon不只可以透過command line啟動也可以導入模組啟動

模組化可以增加程式架構的彈性

Nodemon本身也提供很多監聽事件

透過child_process方式使用Nodemon

```
const { spawn } = require('child_process');

function spawnNodemon() {
  const cp = spawn('nodemon', ['path/to/file.js', '--watch', 'path/to/watch'], {
    stdio: ['pipe', 'pipe', 'pipe', 'ipc'],
  });

  return cp;
}

var app = spawnNodemon();

app.on('message', function (event) {
  if (event.type === 'start') {
    console.log('nodemon started');
  } else if (event.type === 'crash') {
    console.log('script crashed for some reason');
  }
});

// force a restart
app.send('restart');

// force a quit
app.send('quit');
```

7-10



- 可以透過spawn、fork方式產生child_process。
- 在這個範例是使用spawn，因為後續還需要與parent process溝通，需要IPC介面。
- Node.js有提供parent process與child process間的溝通介面(IPC)，並可透過stdio建立管道。
- ['pipe', 'pipe', 'pipe'] 是預設值，對應的是 child process 的 stdin、stdout 和 stderr。
- 再透過 send()，與 parent process互動。

【Key Points】：

透過spawn、fork方式產生child_process

['pipe', 'pipe', 'pipe'] 是預設值，對應的是 child process 的 stdin、stdout 和 stderr

透過 send()，與 parent process互動

模組方式與child_process方式的差異

- 效能差異
- 可讀性差異
- debug難度差異
- 管理差異
- 擴展性差異

7-11



- 模組化仍然是透過Node.js一條process運行，而child_process則可以有類似multiple thread的效果。(與多執行緒還是不同的)
- 模組化可讀性較child_process高，child_process方式的程式撰寫較為底層難理解。
- debug上，模組化最為簡單，child_process則需要注意更多底層細節。
- 模組化的封閉原則讓程式更好管理，而child_process是導入原生模組，依賴原生模組，相依性增加，管理難度提高。
- 程式模組化後擴展性本就會提高許多，而child_process方式有send()可以與Nodemon互動，擴展性方面差異不大。

【Key Points】：

模組化仍是透過Node.js一條process運行，而child_process則可以有類似multiple thread的效果

debug上，模組化最為簡單

模組化的封閉原則讓程式更好管理

7-3: PM2 介紹

- 管理監視程式運行的工具
- cluster、fork 模式
- PM2安裝與使用
- PM2參數設定



designed by freepik



designed by freepik

7-12

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

- PM2是一套管理監視程式運行的工具，可以有效維持服務正常運作。
- Node.js本身是single process，所以PM2的cluster模式是透過child_process，而fork模式則是原本的single process。
- npm install pm2 -g
- PM2 設定config file。
- PM2讓Node.js程式服務不中斷且有完整的log機制，但docker容器化後，就不再需要透過PM2。

【Key Points】：

PM2是一套管理監視程式運行的工具

PM2的cluster模式是透過child_process，而fork模式則是原本的single process

PM2安裝與使用

PM2管理監視程式運行的工具

- 服務不中斷
- 管理介面完整
- log完整
- CI/CD

7-13



- PM2的PM是Process Manager的意思。
- 可以讓程式crash後自動重啟或server重啟後自動重啟程式。
- 管理介面有豐富的資料，CPU、記憶體、process 資訊。
- 具備完整log方便日後debug。
- 可以搭配Gitlab的CI/CD工具，做部署。

【Key Points】：

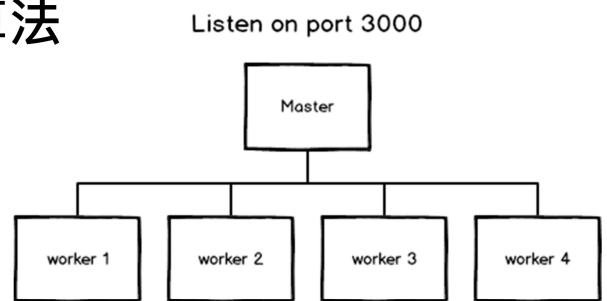
PM2的PM是Process Manager的意思

PM2可以讓程式crash後自動重啟

具備完整log方便日後debug

PM2 的 cluster、fork 模式

- fork 使用 child_process
- cluster 則是建立多個實例
- cluster 使用 Round-robin 演算法
- 兩者的應用情境



1-14

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

- fork 是單一個instance，開多個child_process。
- cluster 底層也是child_process，但開多個instance多個child_process。
- Round-robin演算法，簡而言之，分散式系統依序將請求分配給集群的機器。
- cluster，因為開多個instance，更適合用在運算密集的業務邏輯上或流量大的情境。
- fork，流量不大的服務可以選擇此模式，資源消耗較少。

【Key Points】：

fork 是單一個instance，開多個child_process

cluster 開多個instance多個child_process

兩者的應用情境與差異

PM2安裝與使用

- 全域安裝pm2:

npm install pm2 -g

- 透過 pm2 執行應用程式

pm2 start app.js

- 列出所有透過pm2管理的程式

pm2 list

- 重啟應用程式

pm2 restart <app_name|namespace|id|'all'|json_conf>

- 停止透過pm2啟動的應用程式

pm2 delete <app_name|namespace|id|'all'|json_conf>

7-15



- npm install pm2 -g 全域安裝pm2
- pm2 start 透過 pm2 執行應用程式
- pm2 list 列出所有透過pm2管理的程式
- pm2 restart 重啟應用程式
- pm2 delete 停止透過pm2啟動的應用程式

【Key Points】：

全域安裝pm2

透過 pm2 執行應用程式

停止透過pm2啟動的應用程式

PM2參數設定

- **name**
定義在pm2 list顯示的應用程式名稱
- **watch**
boolean值，監視程式碼變化real time重啟服務
- **max-memory-restart**
記憶體達到設定數值就重啟服務
- **log、output、error、log-date-format、merge-logs**
- **Env**
透過env參數物件，可以傳遞環境變數至PM2再啟動應用程式

7-16



- name: 定義在pm2 list顯示的應用程式名稱。
- watch: boolean值，監視程式碼變化real time重啟服務。
- max-memory-restart，記憶體達到設定數值就重啟服務。
- log方面，PM2有完整的log機制。
- env: 透過env參數物件，可以傳遞環境變數至PM2再啟動應用程式。

【Key Points】：

name: 定義在pm2 list顯示的應用程式名稱。

watch: boolean值，監視程式碼變化real time重啟服務。

env: 透過env參數物件，可以傳遞環境變數至PM2再啟動應用程式。

Summary〈精華回顧〉

- Nodemon可以加快開發速度。
- Nodemon 可以監視程式碼變化。
- Nodemon 的優缺點。
- PM2可以監控應用程式。
- PM2 可以有效管理達到服務不中斷。



7-17

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

- Nodemon透過對module做cache，cache與新的module差異做比對。
- Nodemon大量使用child process以及cache，這些都代表著需要更大量的記憶體。
- Nodemon不只可以透過command line啟動也可以導入模組啟動。
- PM2是一套管理監視程式運行的工具，可以有效維持服務正常運作。
- 可以讓程式crash後自動重啟或server重啟後自動重啟程式。

【Key Points】：

Nodemon 可以監視程式碼變化

Nodemon 的優缺點

PM2是一套管理監視程式運行的工具，可以有效維持服務正常運作

線上程式題

- **7-1 以 Nodemon 監看程式**

新增一個專案，並在專案根目錄新增app.js，透過Nodemon監看。

- **7-2 忽略監看特定程式**

如何設定Nodemon監看專案時，忽略特定程式？

- **7-3 改寫 ecosystem.config.js**

假設有一程式app.js 且該程式是透過以下command line執行：

`export NODE_ENV=dev && pm2 start app.js –name app`

請將command line改寫成ecosystem.config.js，

並且使用command line執行以下命令：

`pm2 start ecosystem.config.js`

7-18



7-1 以 Nodemon 監看程式

新增一個專案，並在專案根目錄新增app.js，透過Nodemon監看。並填入下方程式碼。

```
console.log('test01')
```

接著不關閉app.js程式的執行，直接修改程式碼如下：

```
console.log('test02')
```

請觀察command line顯示的console log。

7-2 忽略監看特定程式

如何設定Nodemon監看專案時，忽略特定程式？

7-3 改寫 ecosystem.config.js

假設有一程式app.js 且該程式是透過以下command line執行：

```
export NODE_ENV=dev && pm2 start app.js –name app
```

請將command line改寫成ecosystem.config.js，

並且使用command line執行以下命令：

```
pm2 start ecosystem.config.js
```

【Key Points】：

7-1 以 Nodemon 監看程式

7-2 忽略監看特定程式

7-3 改寫 ecosystem.config.js

課後練習題(Lab)

- **情節描述:**

開發過程常會需要修改程式碼啟動程式看執行結果，如果有用Nodemon就可以所見即所得，不需要手動重啟程式，加快開發速度。開發完，程式也需要使用PM2，持續對Production程式做監控，程式crash能自動重啟且有完善的log。

- **預設目標:**

- 理解Nodemon安裝方式
- 理解Nodemon配置
- 理解PM2安裝及配置

Estimated time:

20 minutes

7-19



【情節描述】

開發過程常會需要修改程式碼啟動程式看執行結果，如果有用Nodemon就可以所見即所得，不需要手動重啟程式，加快開發速度。開發完，程式也需要使用PM2，持續對Production程式做監控，程式crash能自動重啟且有完善的log。

【預設目標】

- 理解Nodemon安裝方式
- 理解Nodemon配置
- 理解PM2安裝及配置

Lab01:全域安裝Nodemon

Lab02:理解Nodemon配置

Lab03:理解PM2安裝及配置

完成後的程式與檔案，請參考 Example 資料夾的內容。

【Key Points】：

Lab01:全域安裝Nodemon

Lab02:理解Nodemon配置

Lab03:理解PM2安裝及配置

範例程式使用說明

- 範例程式資料夾: Module_07_example
- 使用步驟:
 1. 安裝 Node.js
 2. 安裝 Visual Studio Code
 3. 以 Visual Studio Code 開啟本模組的範例資料夾
 4. 在 Visual Studio Code 按下「Ctrl + 滑鼠右鍵」開啟 terminal 終端機視窗。
 5. 在終端機視窗，輸入下列指令:
`npm install -g pm2`
 6. 在終端機視窗，輸入 `pm2 start lab03\lab03.js --name lab03`

7-20



使用步驟:

1. 安裝 Node.js
<https://nodejs.org/en/>
2. 安裝 Visual Studio Code
<https://code.visualstudio.com/>
3. 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」
4. 在 Visual Studio Code 按下「Ctrl + 滑鼠右鍵」開啟 terminal 終端機視窗。
5. 在終端機視窗，輸入下列指令:
`npm install -g pm2`
6. 在終端機視窗，輸入 `pm2 start lab03\lab03.js --name lab03`

【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 + 滑鼠右鍵」開啟 terminal 終端機視窗，輸入指令