

# 目錄

序	1.1
推廣	1.2
關於我 & 公司	1.2.1
相關活動	1.2.2
第一週 - 測試基礎	1.3
測試分類	1.3.1
手動測試 vs 自動化測試	1.3.2
測試金字塔	1.3.3
測試環境的種類	1.3.4
安裝測試環境	1.3.5
介紹 Selenium	1.3.6
錄製一個測試案例	1.3.7
第二週 - 指令的應用	1.4
Selenium IDE UI 功能解說	1.4.1
測試相關專有名詞	1.4.2
選取元素	1.4.3
指令的種類	1.4.4
Selenium IDE 實戰練習 - 登入和登出	1.4.5
Selenium IDE 實戰練習 - 新增帳號	1.4.6
第三週 - 網頁自動化測試基礎	1.5
安裝 VS Code	1.5.1
Javscript 基礎	1.5.2
NPM 套件管理工具	1.5.3
介紹 WebdriverIO	1.5.4
設定 Selenium Server 環境	1.5.5
用 REPL 來練習撰寫自動化程式	1.5.6
練習寫單元測試	1.5.7

---

第四週 - 網頁自動化測試進階(WebdriverIO)	1.6
初始化測試的專案	1.6.1
WebdriverIO 設定檔	1.6.2
WebdriverIO 常用指令語法	1.6.3
實戰練習 - 登入和登出	1.6.4
Page Object 模式	1.6.5
自行挑戰題 - 計算機	1.6.6

---

# 2017 德明財經科技大學 - 網站自動化測試課程

講師：劉艾霖 (alincode) - 創科資訊軟體技術顧問

## 授權

本書採用 [Attribution-NonCommercial-ShareAlike 4.0 International](#) 授權，你不需要為本書付費。你可以免費的複製、發佈、修改或展示本書。然而，本書的版權是屬於我，劉艾霖，並且請勿將本書用於商業目的。你可以透過以下連結了解授權內容的全文：<https://creativecommons.org/licenses/by-nc-sa/4.0/>

## 勘誤通知

如發現內容勘誤，歡迎利用這些管道和我聯繫：

- 使用 [GitHub Issues](#) 回報錯誤
- 發 pull request
- 寄信到 [alincode@gmail.com](mailto:alincode@gmail.com)

最新更新時間：2017/06/07

# 關於我(Alin)

**Ai-Lin Liou**  
alincode

Add a bio

📍 Taiwan  
✉ alincode@gmail.com

**Organizations**

2,080 contributions in the last year

Contribution settings ▾

May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr

Mon Wed Fri

Learn how we count contributions.

Less More

The GitHub profile page displays Alin's personal information, including her name, GitHub handle, location, email, and organization. It also shows her repositories, which include "learn geb" (HTML, 38 stars), "jsdc-tw-2016-webdriverio" (JavaScript, 8 stars), "modern-web-2016-e2etest" (JavaScript, 5 stars), "front-end-test-sandbox" (JavaScript, 2 stars), "listening-pronunciation-extension" (JavaScript, 1 star), and "sails-react-sample" (JavaScript, 1 star). A prominent feature is the "Contribution heatmap" at the bottom, which visualizes her activity over the past year by day of the week and month. The heatmap uses a color scale from light green (Less) to dark green (More).

劉艾霖 (alincode)

- 創科資訊 - 軟體技術顧問
- <https://github.com/alincode>
- 具有實務上的 Web Full Stack 開發，及協助企業導入網站自動化測試實務經驗，熟悉 Java 及 Javascript 開發，目前從事企業技術教學與顧問。

## 擅長領域

- 網頁開發 (前端 + 後端 = 全端)
- 自動化測試開發
- 電子商務網站開發
- 第三方服務整合

## 教學 / 講師經歷

- 2016 Trunk studio - 講者 WebdriverIO 起手式

- 2016 Modern Web 2016 講者
- 2016 JSDC.tw 講者
- 2017 企業內訓講師 / 大專院校業師 / 社群講者
- 2017 iThome 測試週講者

# 測試相關活動

## 2016 Modern Web

**JSDC.tw** ABOUT VENUE SPEAKER **AGENDA** SPONSOR MEMBER BLOG  中文

14:10 | 14:40  
下午茶

14:40   15:20	 淺談網站自動化測試 - 以 WebdriverIO 框架為例 by 劉艾霖 <a href="#">Slide</a> <a href="#">Youtube</a>	 fbreactions.io - 用AWS Lambda 爬數據視覺化 by 劉俊彥(Vincent) <a href="#">Slide</a> <a href="#">Youtube</a>
---------------	--	--

## 2016 JSDC.tw

Modern Web 2016 講師與講題 大會議程表 講程影片 焦點講師 線上交流 線上共筆 會後問卷 交通資訊   

NaN:NaN-NaN:NaN	<b>Universal Angular 2 in FinTech</b> 湯桂川 / 廣發證券前端技術專家 	如何透過語意網路與使用者行為優化使用者經驗 洪進吉 (食夢黑貓) / 網管工程師、技術顧問 	無痛網站自動化測試 劉艾霖 / Exma-Square資深軟體工程師 
-----------------	--	---	--

## 2017 測試週

iThome #Testing Day 講程講師 活動資訊 交通資訊 立即購票 

### 講師陣容

 <b>柯仁傑</b> 趨勢科技資深經理	 <b>曾明賢</b> StreetVoice 網路技術部副總經理	 <b>劉艾霖</b> 創科資訊軟體開發技術顧問	 <b>謝宗穎</b> 創科資訊創辦人
---	--	--	--

## 2017 網站測試開發 with WebdriverIO 實戰講堂

The screenshot shows a workshop listing on the iThome LEARNING website. The title is "網站測試開發 with WebdriverIO 實戰講堂". The details include: Date: 6/25 (日), Time: 2017/06/25 09:30 ~ 17:30, Location: BOOKSHOW 說書會 (台北市信義區忠孝東路五段293號3樓), Price: \$ 3990 元起. The listing also features a photo of the instructor, 劉艾森 (alincode), and includes tags: CI, DevOps, AutomationTesting, webdriverIO. Buttons for "立即報名" (Register Now) and "課程問答" (Course Questions) are visible, along with social sharing icons for Facebook, Twitter, and LINE.

## 社團 Test Corner

# 測試分類

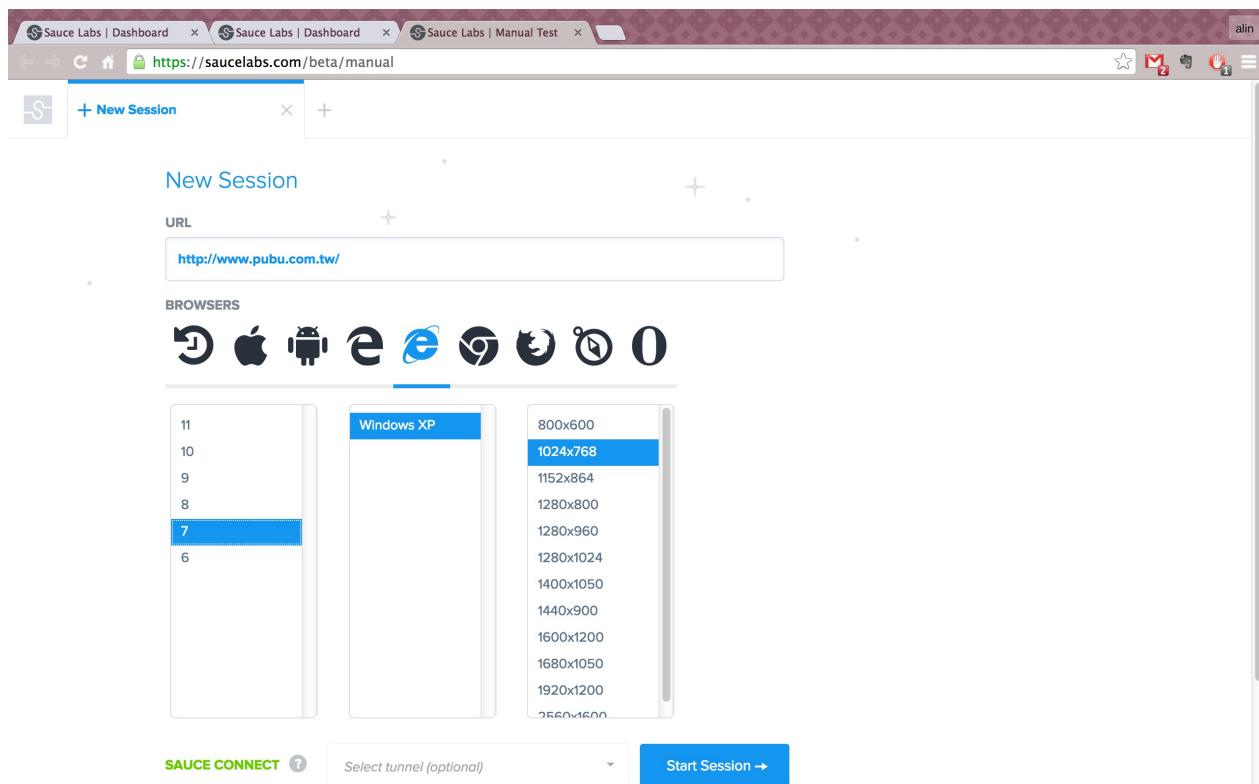
## 黑箱測試 vs 白箱測試

- 黑箱測試：假設看不到程式碼的方式去測試，主要透過使用者介面去測試功能。
- 白箱測試：假設看得到程式碼的方式去測試，會直接撰寫可與程式碼對接的測試程式。

## 功能測試 vs 效能測試

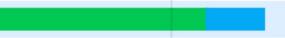
### 功能測試

- 邏輯功能測試
- 好用性測試
- 相容性測試(ie9, ie10, ie11, android 版本...)



### 效能測試

時間效能：回應時間長短

Name	Status	Type	Initiator	Size	Time	Waterfall	4.00 s
selenium.html	200	doc...	Other	48.7 KB	2.89 s		
style.css	200	style...	<a href="#">selenium...</a>	10.6 KB	224 ms		
plugin.css	200	style...	<a href="#">selenium...</a>	906 B	229 ms		
website.css	200	style...	<a href="#">selenium...</a>	2.3 KB	237 ms		
search.css	200	style...	<a href="#">selenium...</a>	394 B	238 ms		
website.css	200	style	selenium	1.2 KB	247 ms		

33 requests | 694 KB transferred | Finish: 9.57 s | DOMContentLoaded: 6.95 s | Load: 9.57 s

空間效能：消耗的系統資源，例如：記憶體使用率、網路頻寬。



```
[root@NGINX ~]# free -m
              total        used        free      shared  buffers   cached
Mem:       499         473         25          0        68       290
-/+ buffers/cache:    114        384
Swap:     1023          0       1023
[root@NGINX ~]#
```

## 延伸閱讀

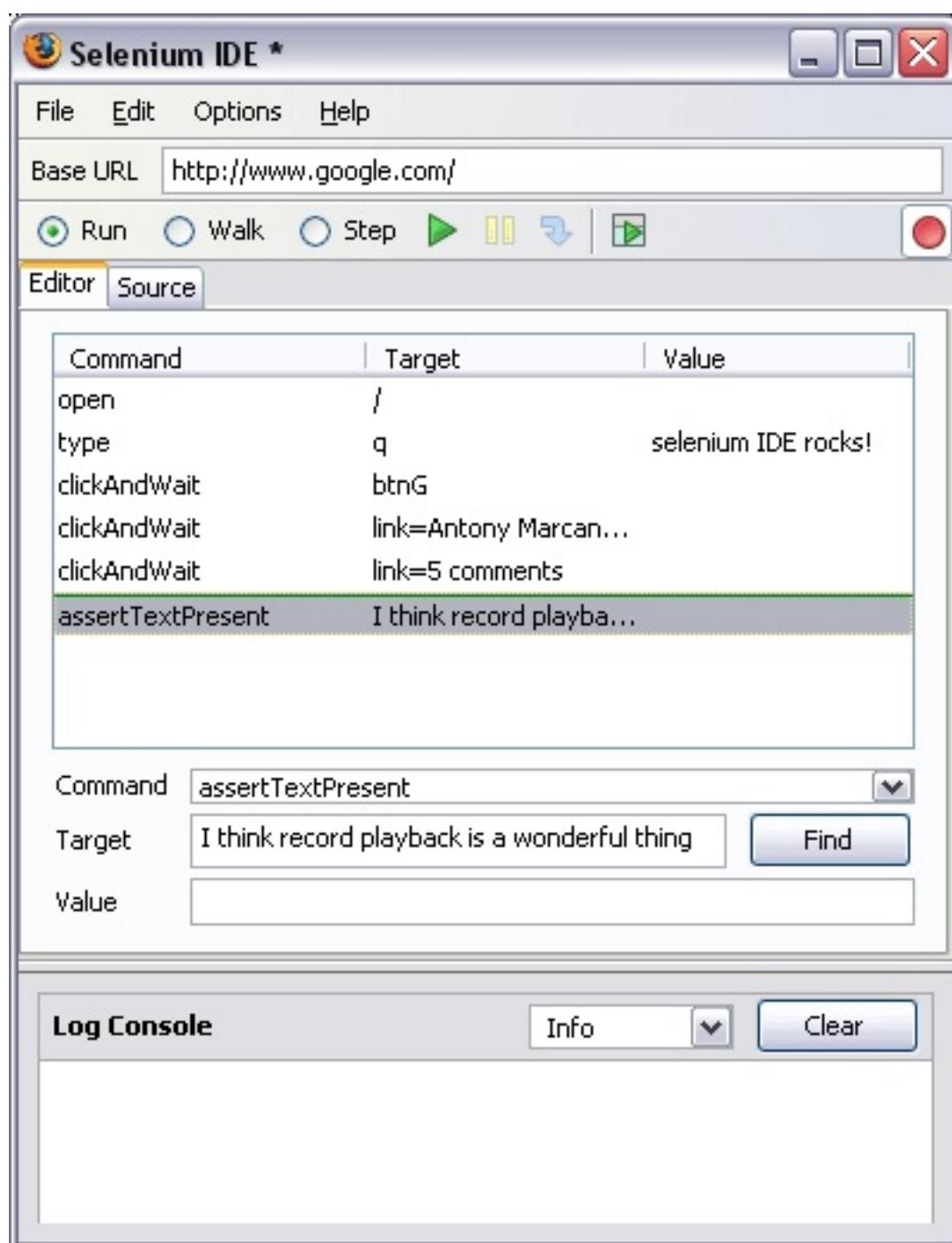
- [When to Start Automation Testing - GlobalSQA](#)
- [Google Testing Blog: Just Say No to More End-to-End Tests](#)

# 手動測試 vs 自動化測試

以自動化程度來劃分，可分成：

- 手動測試 (Manual Testing)：一個步驟一個步驟的手動去執行
- 自動化測試 (Automation Testing)

錄製測試步驟



撰寫自動化程式

```
var webdriver = require('selenium-webdriver'),
  By = webdriver.By,
  until = webdriver.until;

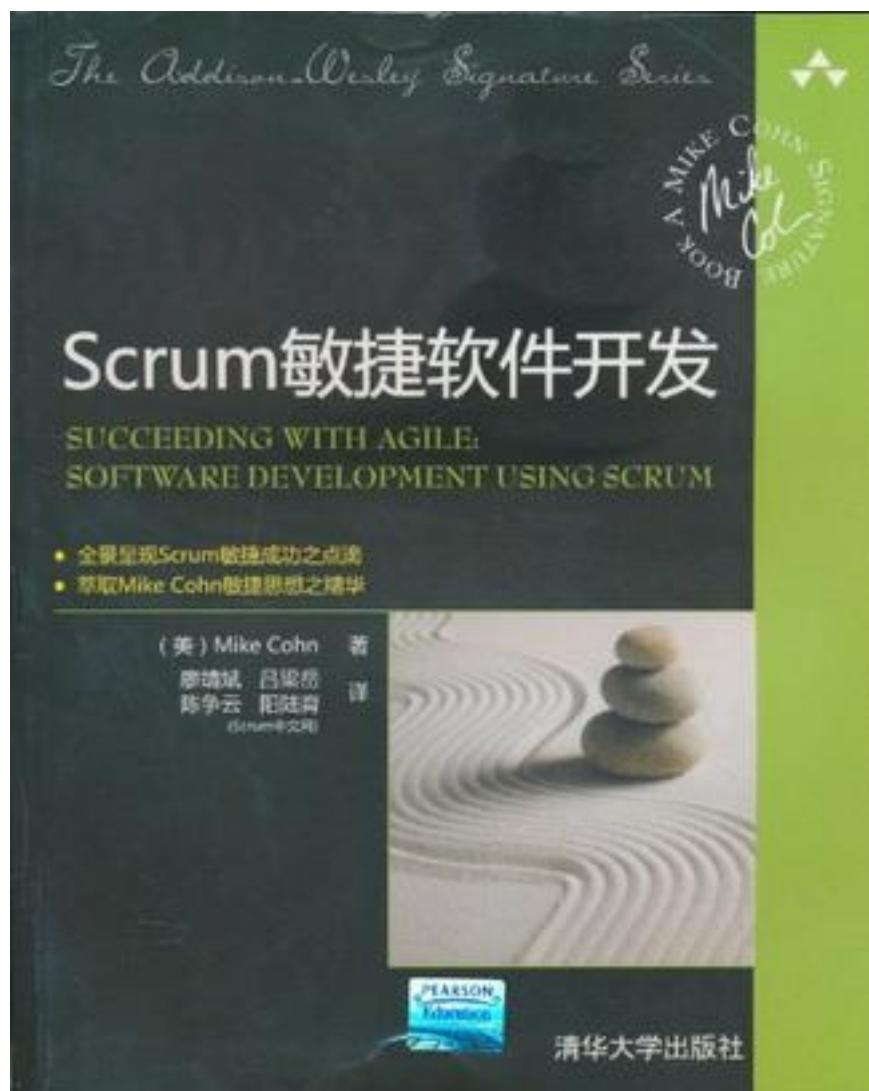
var driver = new webdriver.Builder()
  .forBrowser('firefox')
  .build();

driver.get('http://www.google.com/ncr');
driver.findElement(By.name('q')).sendKeys('webdriver');
driver.findElement(By.name('btnG')).click();
driver.wait(until.titleIs('webdriver - Google Search'), 1000);
driver.quit();
```

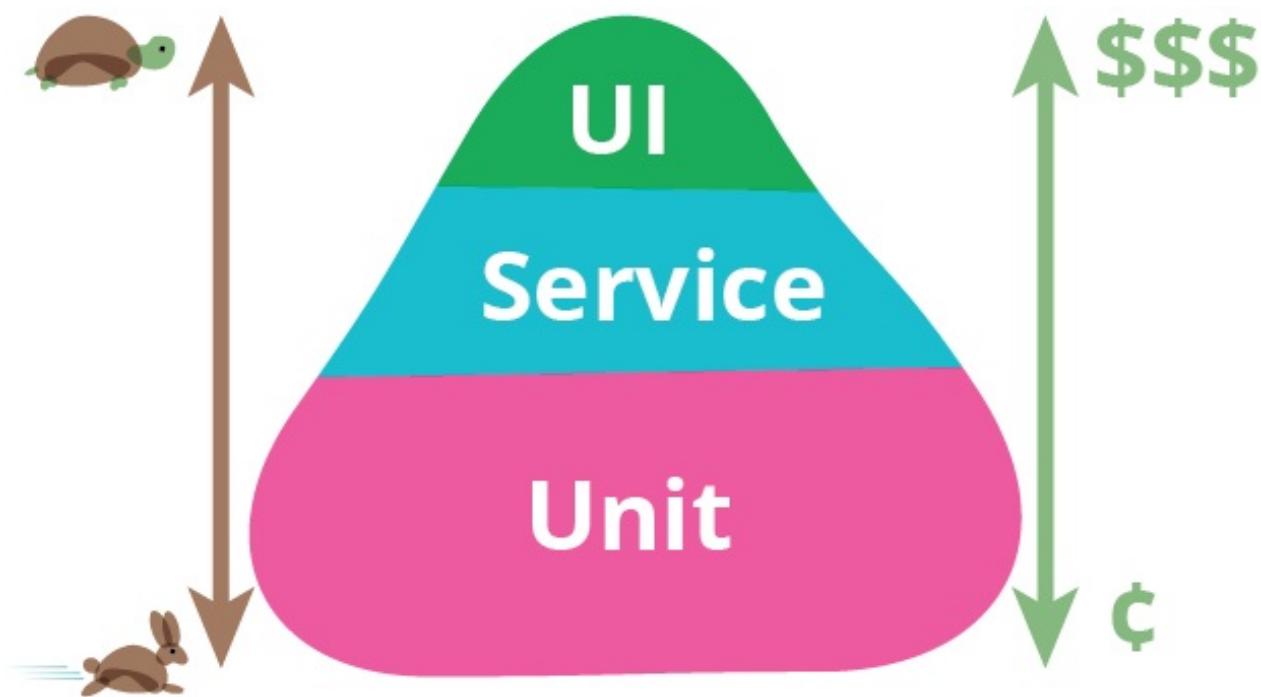
### 比較表

項目	手動測試	自動化測試
花費的時間	長	短
可測出的問題分佈	新功能的 bug	已知的測試流程的 bug
同時測試數量	一次只能測一個	一次可以測多個

# 測試金字塔 (Test Pyramid)



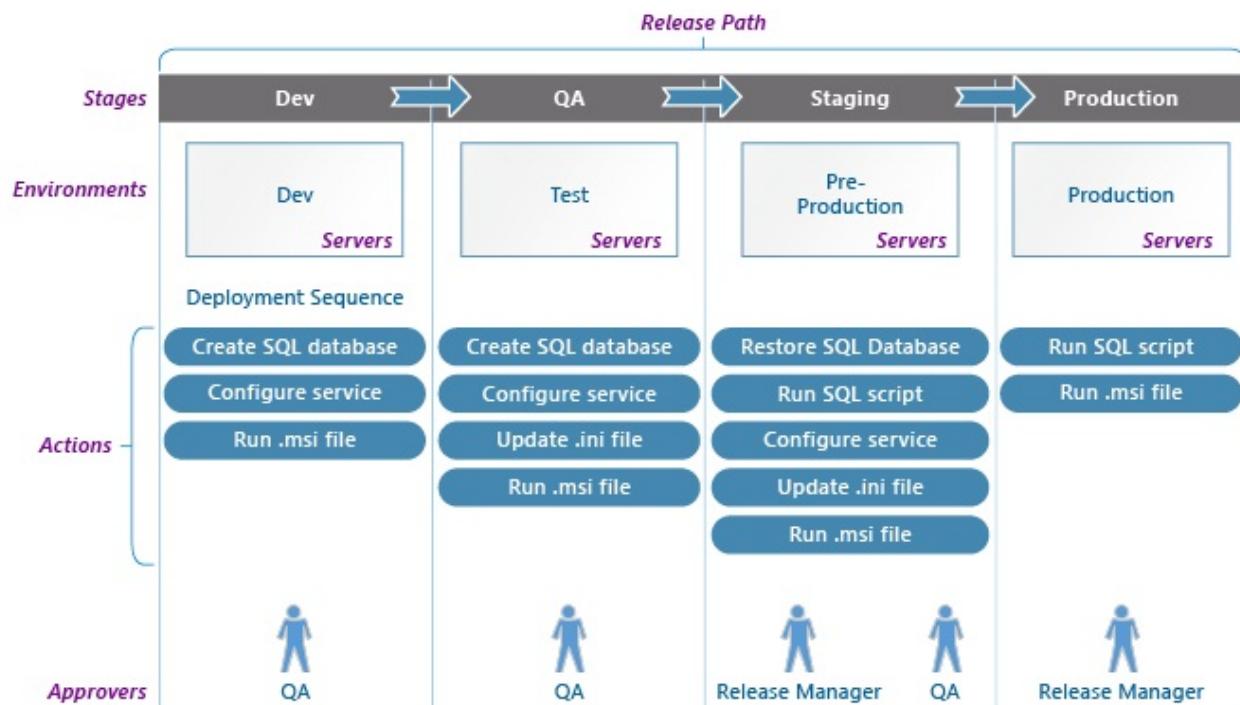
測試金字塔的概念是由 Mike Cohn 所提出的一個測試開發概念，被 Martin Fowler 大師轉述而聲名大噪，相關細節描述在 [Succeeding with Agile](#) 這本書中，其中很重要的一點是強調你應該擁有更多低階的單元測試。



層級	測試程式的數量	描述
第三層	10%	UI 測試 / 前端 (end-to-end) 測試
第二層	20%	商業邏輯測試 / 整合測試
第一層	70%	單元測試：針對程式中的最小測試單元進行驗證。

相關資料來源：<https://martinfowler.com/bliki/TestPyramid.html>

# 測試環境



## DEV

- 開發環境
- 資料庫：假資料

## QA

- 有時又稱使用者驗收測試(UAT - User Acceptance Testing)
- 使用情境：驗收功能

## Staging

- 跟 Production 一致的環境
- 資料庫：與正式環境一致
- 使用情境：上版前的最後測試、預先測試 SQL 腳本

## Production

- 正式環境



## 安裝測試環境

- 安裝 JDK
- 安裝 NPM
- 安裝 Selenium IDE

## 安裝 NPM

### 下載軟體

<https://nodejs.org/en/>



Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#). Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, [npm](#), is the largest ecosystem of open source libraries in the world.

[Download for macOS \(x64\)](#)

**v6.10.3 LTS**

Recommended For Most Users

**v7.10.0 Current**

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)      [Other Downloads](#) | [Changelog](#) | [API Docs](#)

Or have a look at the [LTS schedule](#).

## 測試安裝結果

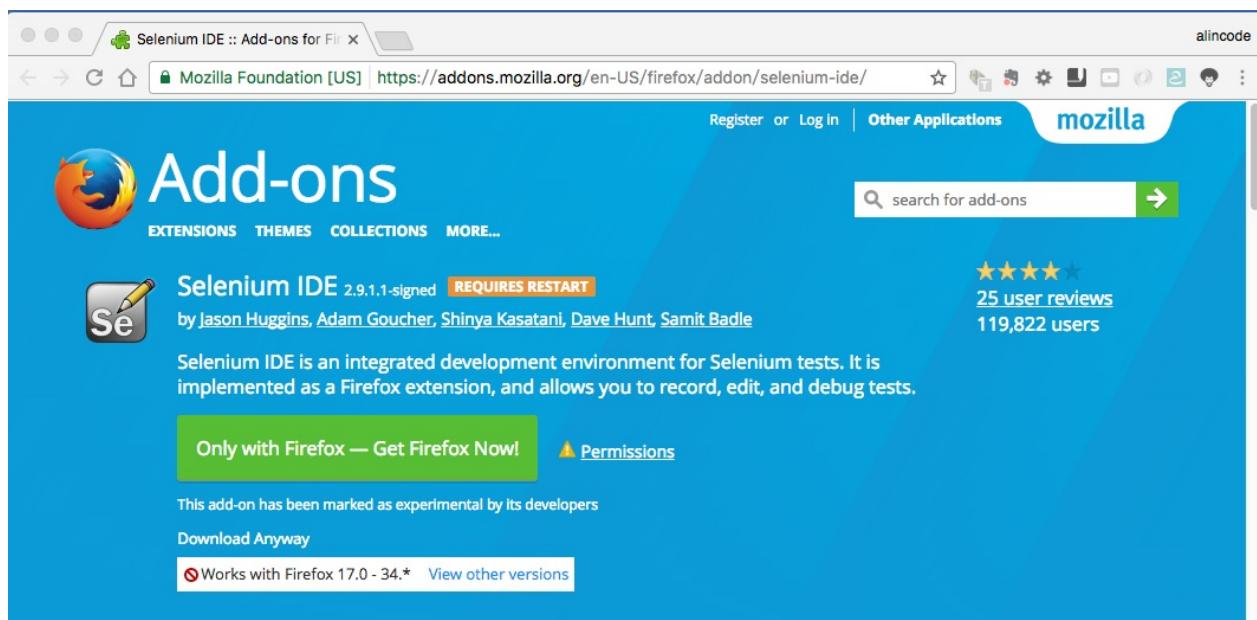
```
node -v  
npm -v
```

## 安裝 Selenium IDE

- 第一步：首先你必須要安裝 Firefox



- 第二步：然後你必須要安裝 Selenium IDE，你可以至 <https://addons.mozilla.org/en-US/firefox/addon/selenium-ide/> 安裝它。



# 介紹 Selenium

- 免費使用 (Open source project)
- 可透過自動化的方式來控制使用者介面
- 支援多種瀏覽器：Firefox、Chrome、IE、Edga...
- 支援多平台：MAC、Window、Linux
- 支援多語言：Java、Python、Ruby、C#、Javascript、C++
- <https://github.com/christian-bromann/awesome-selenium>

## 優點

- 支援重複執行測試程式
- 支援並行執行
- 支援背景執行
- 提高測試準確定，減少人為產生的錯誤。
- 省下測試的時間

## 應用領域

- 自動化測試
- 網站擷取 / 網路爬蟲

## 目前有四個主要的專案

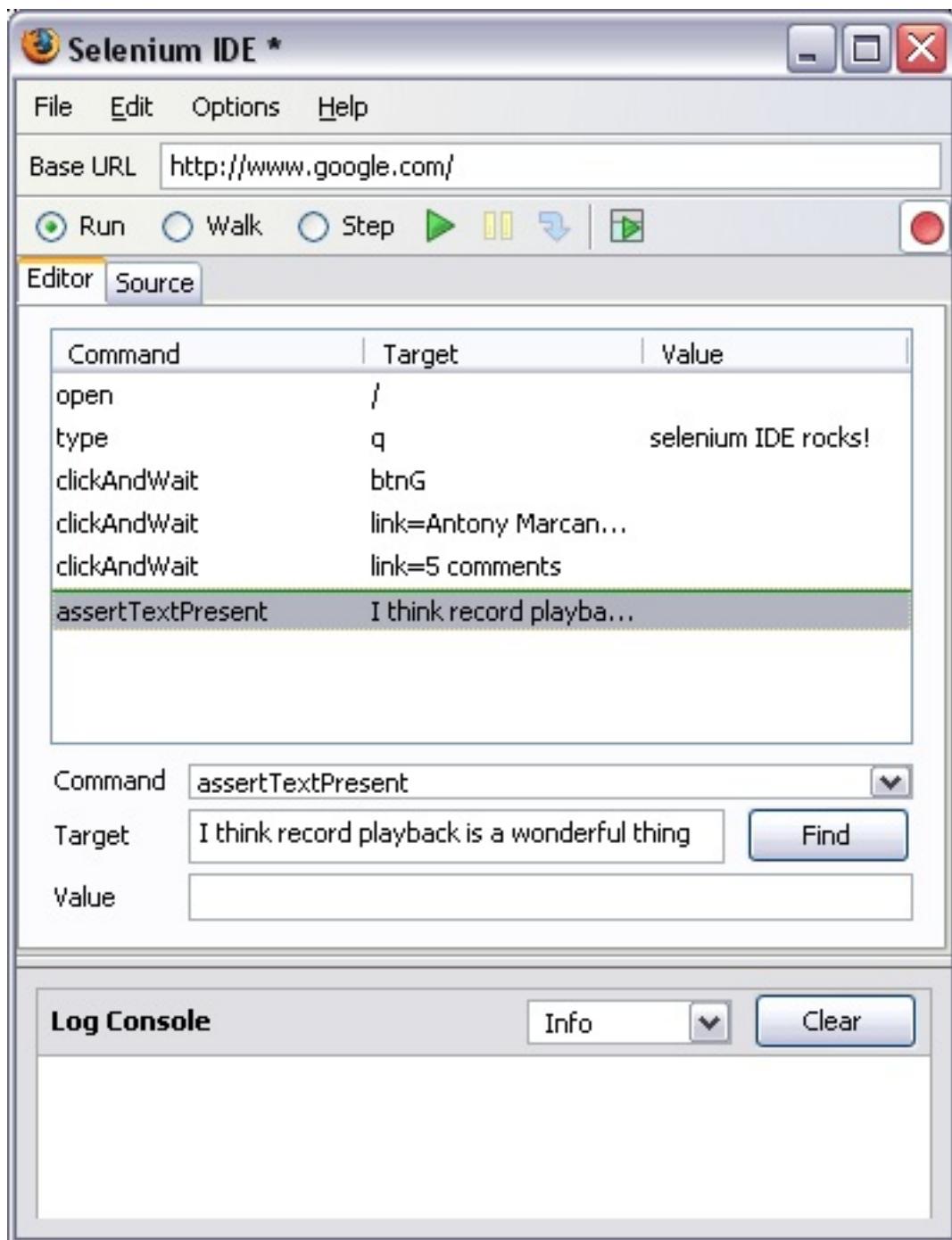
- Selenium IDE
- Selenium Remote Control (RC)
- Selenium Grid
- Selenium WebDriver

## Selenium IDE

Selenium IDE 是 Firefox 附加元件 (extension)，需要搭配 Firefox 瀏覽器才能使用。

### 開啟 Selenium IDE

在 Firefox 瀏覽器的工具選單，打開 **Selenium IDE** 會出現下面這個視窗畫面：



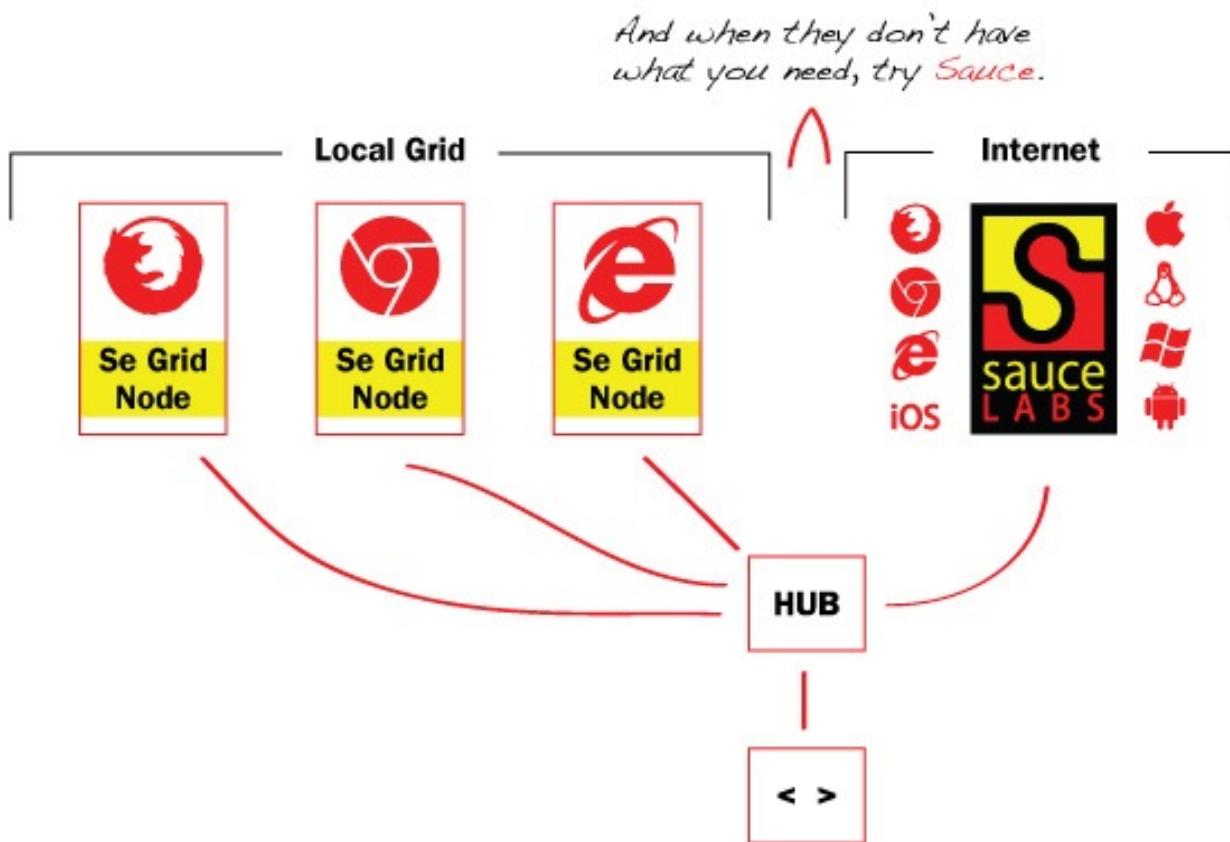
## Selenium Remote Control

簡稱 **Selenium RC**，它提供可以遠端執行 Selenium 的 Client / Server 架構。

Selenium Server 是複雜控制瀏覽器行為，Selenium Client 則是用於撰寫測試腳本來跟 Selenium Server 溝通。

## Selenium Grid

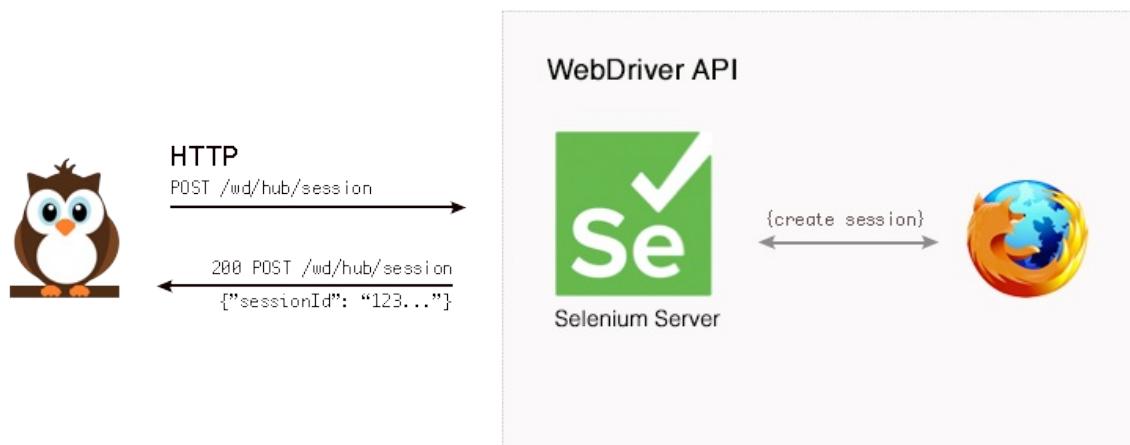
圖片來源：[Introducing the Sauce Plugin for Selenium Grid | Sauce Labs](#)



Selenium Grid 主要控制多台機器(RC Node)，每次測試任務都先呼叫 Hub，然後再由路由(Hub) 分配給節點(Node)。

- Selenium Grid Hub
- Selenium Grid Node

## Selenium WebDriver



許多網頁自動化測試框架，都是以 Selenium WebDriver API 作為基礎，功能強大且穩固已經讓 Selenium 成為瀏覽器自動化的基石。Selenium 2.0 帶來 WebDriver 的實作，跨越不同瀏覽器的自動化操作，有更清楚定義的標準可循，目前 [WebDriver API](#) 規範已提交 W3C，若能夠被標準化且在各大瀏覽器實作，執行跨瀏覽器的自動化測試工作將會被簡化許多。

```
var webdriver = require('selenium-webdriver'),
  By = webdriver.By,
  until = webdriver.until;

var driver = new webdriver.Builder()
  .forBrowser('firefox')
  .build();

driver.get('http://www.google.com/ncr');
driver.findElement(By.name('q')).sendKeys('webdriver');
driver.findElement(By.name('btnG')).click();
driver.wait(until.titleIs('webdriver - Google Search'), 1000);
driver.quit();
```

## 延伸閱讀

- [GitHub - SeleniumHQ/selenium: A browser automation framework and ecosystem.](#)
- [Selenium Documentation — Selenium Documentation](#)
- <https://www.w3.org/TR/webdriver/>
- <http://seleniumhq.github.io/selenium/docs/api/javascript/module/selenium-webdriver/>
- <https://github.com/SeleniumHQ/selenium/tree/master/javascript/node/selenium-webdriver>

# 錄製一個測試案例

操作 Selenium IDE 就像錄影機，在開始「錄製」後，在瀏覽器操作網站的動作就會被捕捉，產生測試案例（Test Case）的內容。錄製完成後，可以用「播放」重新把網站操作過程重播一次。

這是使用 Selenium 進行網站自動化的基礎，以 Google 搜尋為例，建立一組測試案例包含：

1. 前往 <https://www.google.com.tw> 網址
2. 在文字輸入框輸入 `selenium ide`
3. 然後按下**Google 搜尋按鈕**
4. 取得搜尋結果，檢查結果是否包含預期的內容

## HTML 原始碼

```
<input class="gsfi" id="lst-ib" maxlength="2048" name="q" autocomplete="off" title="搜尋" value="" aria-label="搜尋" aria-haspopup="false" role="combobox" aria-autocomplete="both" dir="ltr" spellcheck="false" type="text">

<input value="Google 搜尋" aria-label="Google 搜尋" name="btnK" jsaction="sf.chk" type="submit">
```

## 錄製一個自動填寫表單的程式

可供測試的網站：<http://bit.ly/watir-example>

## 錄製一個測試案例

Untitled (untitled suite) - Selenium IDE 2.9.1 \*

Base URL <https://docs.google.com/>

Test Case Untitled \*

Command	Target	Value
open	/forms/d/e/1FAipQLSefsV3eQqD...	
type	id=entry_1000000	alincode
type	id=entry_1000001	test123
click	id=group_1000002_1	
click	id=group_1000003_1	
click	id=group_1000003_2	
select	id=entry_1000004	label=Firefox
click	id=group_1000005_1	
click	id=group_1000006_1	
click	id=group_1000007_2	
clickAndWait	id=ss-submit	
waitForAllWindowTit...	感謝您！	
verifyText	xpath=/html/body/div/div/div[2]/... Your response has been recorded.	

Runs: 1  
Failures: 0

Log Reference UI-Element Rollup

**waitForAllWindowTitles(pattern)**  
Generated from `getAllWindows()`

Returns:  
Array of titles of all windows that the browser knows about.  
Returns the titles of all windows that the browser knows about in an array.

# Selenium IDE UI 功能解說

## 功能列 (Toolbar)



快速 / 慢速執行



執行 Test Suit



執行 Test Case



練習題一

- 建立新的測試案例 (Test Case)
  - 一個 Test Case 名稱叫做 signin
  - 一個 Test Case 名稱叫做 signout

練習題二

- 分別儲存 Test Case
- 儲存測試集合 (Test Suit)

練習題三

- 刪除 Test Case
- 添加 Test Case
- 切換 Test Case

錄影 (Record)



逐步執行



暫停 / 恢復



## 練習題四

- 錄影 <http://demoqa.com/>
- 設定中斷點 / 移除中斷點
- 逐步執行
- 恢復

## 面板

### Log

Log	Reference	UI-Element	Rollup	Info	Clear
[info] Executing:  waitForPageToLoad					
[info] Executing:  clickAndWait   xpath=id('menu_download')/a					
[info] Executing:  assertTitle   Downloads					
[info] Executing:  verifyText   xpath=id('mainContent')/h2   Downloads					

### 使用文件

Log	Reference	UI-Element	Rollup
<b>clickAndWait(locator)</b> Generated from <code>click(locator)</code>			

**Arguments:**

- locator - an element locator

Clicks on a link, button, checkbox or radio button. If the click action causes a new page to load (like a link usually does), call `waitForPageToLoad`.

### 測試案例面板 (Test Case Pane)

Command	Target	Value
open	/	
waitForPageToLoad		
clickAndWait	xpath=id('menu_download')/a	
assertTitle	Downloads	
verifyText	xpath=id('mainContent')/h2	Downloads

## 延伸閱讀

- [Selenium IDE :: Firefox 附加元件](#)
- [Selenium-IDE — Selenium Documentation](#)

# 測試相關的專有名詞

## 測試案例 (Test Case) vs 測試集合 (Test Suit)

Test Case
signin
signout

## 測試步驟 (Test Step)

## 測試程序 (Test Run)

# 元素選取器

常見的英文稱呼叫：Target / Element Locators / Selector

## identifier

identifier=id：選擇帶有特定 @id 屬性的元素。如果 @id 沒有找到元素，則會改以選擇 @name 中符合 id 的值。指令需要元素定位器為參數時，大多都是以此作為預設值。

```
id=myId
```

```
<div id="myId">hello</div>
```

## Name

name=name：選擇帶有特定 @name 屬性的元素。

```
<div name="myName">hello</div>
```

### 練習題：使用 identifier 與 Name 選擇方式

## DOM

dom=javascriptExpression：以 JavaScript 的方式選擇 DOM，開頭必定是 document.。例如 dom=document.images[2]。

```
dom=document.getElementById('loginForm')
dom=document.forms['loginForm']
dom=document.forms[0]
document.forms[0].username
document.forms[0].elements['username']
document.forms[0].elements[0]
document.forms[0].elements[3]
```

```
<html>
  <body>
    <form id="loginForm">
      <input name="username" type="text" />
      <input name="password" type="password" />
      <input name="continue" type="submit" value="Login" />
      <input name="continue" type="button" value="Clear" />
    </form>
  </body>
<html>
```

## xpath

xpath=xpathExpression：以 XPath 表示式來定位元素。

```
xpath=id('myId')
```

- [XPath Checker](#)
- [W3Schools XPath Tutorial](#)

## link

link=textPattern：選擇包含指定文字比對模式（text patterns）的連結（link）或錨點（anchor）元素，也就是 [。](#)

```
link=Continue
link=Cancel
```

```
<html>
  <body>
    <p>Are you sure you want to do this?</p>
    <a href="continue.html">Continue</a>
    <a href="cancel.html">Cancel</a>
  </body>
<html>
```

練習題：使用 xpath 與 link 選擇方式

## CSS selector

css=cssSelectorSyntax：以 CSS 選擇器來選擇元素，請參考 CSS2 選擇器、CSS3 選擇器的說明。這應該是網頁工程師最容易使用的定位器吧。

```
css=form#loginForm  
css=input[name="username"]  
css=input.required[type="text"]  
css=input.passfield  
css=#loginForm input[type="button"]  
css=#loginForm input:nth-child(2)
```

```
<html>  
  <body>  
    <form id="loginForm">  
      <input class="required" name="username" type="text" />  
      <input class="required passfield" name="password" type="password" />  
      <input name="continue" type="submit" value="Login" />  
      <input name="continue" type="button" value="Clear" />  
    </form>  
  </body>  
<html>
```

## 練習題：使用 CSS 選擇方式

<http://flukeout.github.io/>

選擇器	例子	例子描述	CSS
<u>.class</u>	.intro	選擇 class="intro" 的所有元素。	1
<u>#id</u>	#firstname	選擇 id="firstname" 的所有元素。	1
*	*	選擇所有元素。	2
<u>element</u>	p	選擇所有 <p> 元素。	1
<u>element,element</u>	div,p	選擇所有 <div> 元素和所有 <p> 元素。	1
<u>element element</u>	div p	選擇 <div> 元素內部的所有 <p> 元素。	1
<u>element&gt;element</u>	div>p	選擇父元素為 <div> 元素的所有 <p> 元素。	2
<u>element+element</u>	div+p	選擇緊接在 <div> 元素之後的所有 <p> 元素。	2
<u>[attribute]</u>	[target]	選擇帶有 target 屬性所有元素。	2
<u>[attribute=value]</u>	[target=_blank]	選擇 target="_blank" 的所有元素。	2
<u>[attribute~=value]</u>	[title~=flower]	選擇 title 屬性包含單詞 "flower" 的所有元素。	2
<u>[attribute =value]</u>	[lang =en]	選擇 lang 屬性值以 "en" 開頭的所有元素。	2

## 延伸閱讀

- [CSS 選擇器參考手冊](#)
- [w3schools - css selectors](#)

# Selenium 指令

## 命令組成

Command	type
Target	id=lst-ib
Value	selenium ide

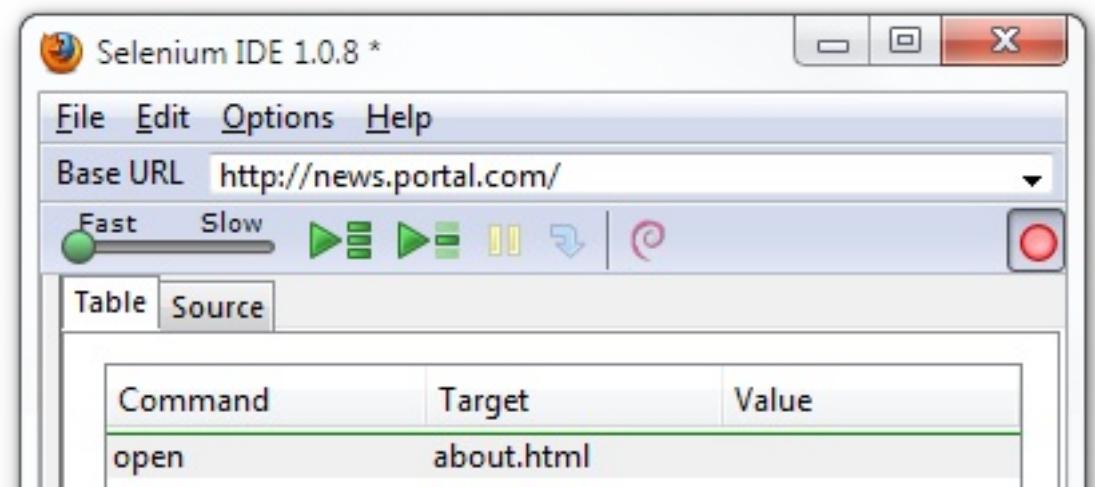
- 指令 (Command) : 行為 / 事件
- 目標 (Target) : 選取元素
- 值 (Value)

## 指令的種類

- 操作 (Actions)
- 存取 (Accessors)
- 驗證 (Assertions)

# 操作 (Actions)

**open** : 前往某個頁面



命令 (Command)	目標 (Target)	值 (Value)
open	/test	
open	<a href="http://demoqa.com">http://demoqa.com</a>	

### click : 模擬點擊一下

命令 (Command)	目標 (Target)	值 (Value)
click	myCheckbox	
clickAndWait	mySubmitButton	
clickAndWait	myLike	

### type : 模擬鍵盤輸入

命令 (Command)	目標 (Target)	值 (Value)
type	myField	Hi
typeAndWait	myField	Hi

### select : 模擬選取

命令 (Command)	目標 (Target)	值 (Value)
select	id=entry_1000004	label=Firefox
select	dropDown	index=0
select	dropDown	value=AUD
selectAndWait	dropDown	index=0
selectAndWait	dropDown	value=AUD

### pause

- 暫停
- 單位是毫秒

命令 (Command)	目標 (Target)	值 (Value)
pause	5000	

# 存取 (Accessors)

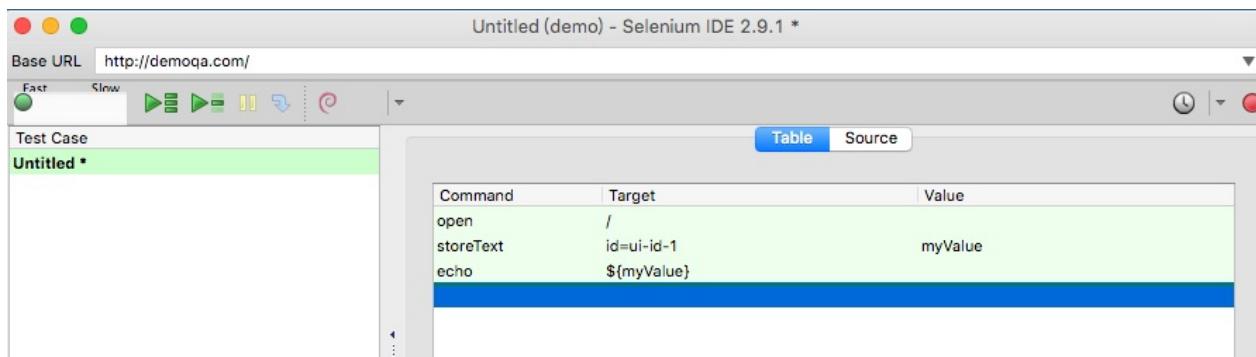
**storeTitle store**

命令 (Command)	目標 (Target)	值 (Value)
store	alincode@gmail.com	myEmail

命令 (Command)	目標 (Target)	值 (Value)
type	id=email	\${myEmail}

**storeText**

練習題



# 驗證 (Assertions)

- 驗證 (assert)
  - assertText
  - assertTitle
  - assertAlert
- 辨識 (verify)
  - verifyText
  - verifyTitle
  - verifyTextPresent
  - verifyElementPresent
  - verifyTable
- 等待 (waitFor)
  - waitForText
  - waitForPageToLoad
  - waitForElementPresent

## 驗證 (assert) vs 辨識 (verify)

			Table	Source
Command	Target	Value		
open	/			
assertText	id=ui-id-1	Tab 2		
click	id=ui-id-2			

差別在於處理錯誤的方式

- 驗證 (assert) 發生錯誤時，測試將會終止。
- 辨識 (verify)：發生錯誤時，只是將錯誤訊息留下記錄，測試將會繼續執行不會中斷。
  - 使用情境：不影響整體測試流程的小細節

### verifyText

- 驗證文字

命令 (Command)	目標 (Target)	值 (Value)
verifyText	xpath=/html/body/div[2]/div/p	驗證的文字
verifyText	//html/body/div[2]/div/p	驗證的文字

### verifyAllWindowTitles

- 驗證視窗標題

命令 (Command)	目標 (Target)	值 (Value)
verifyAllWindowTitles	Google	

## 等待 (waitFor)

等待某些情況發生時才生效，常用於非同步 (AJAX)。

### waitForAllWindowTitle

- 等待視窗指定標題出現

命令 (Command)	目標 (Target)	值 (Value)
waitForAllWindowTitle	您的標題	

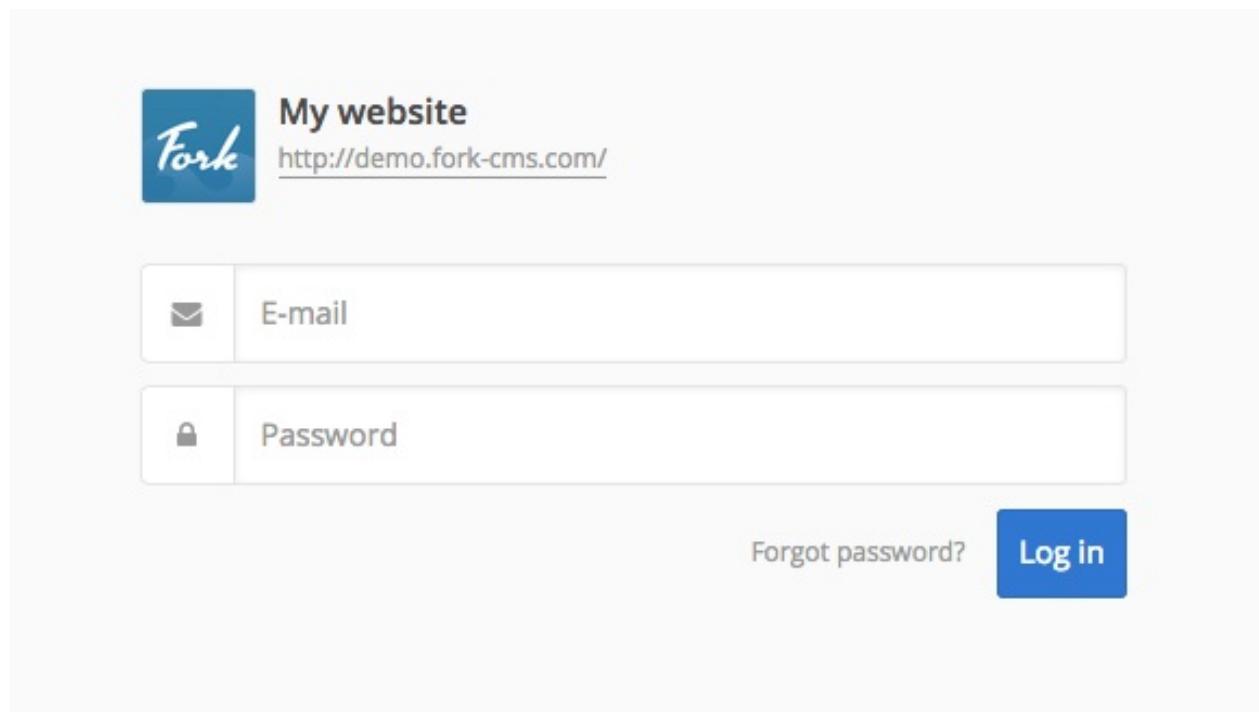
## 延伸閱讀

- [Selenium Assert vs Verify Commands and how to use in automation - YouTube](#)

# Selenium IDE 實戰練習 - 登入和登出

- <http://www.fork-cms.com/demo>
- <http://demo.fork-cms.com/private/>

帳號 : demo@fork-cms.com 密碼 : demo



## 答案

ex01		
open	/private/en/authentication/index?token=fj9pkr4c2k	
type	id=backendEmail	demo@fork-cms.com
type	id=backendPassword	demo
clickAndWait	name=login	
assertText	css=span.nav-item-text	Dashboard
click	css=img.img-circle	
clickAndWait	link=Sign out	
assertText	name=login	Log in

# Selenium IDE 實戰練習 - 新增帳號

- <http://www.fork-cms.com/demo>
- <http://demo.fork-cms.com/private/>

帳號 : demo@fork-cms.com 密碼 : demo

This screenshot shows the Fork CMS dashboard. The left sidebar has a dark blue background with white icons and text: Dashboard, Pages, Modules (selected), Marketing, Settings, Profiles (selected), Overview, and Groups. The main content area has a light gray background. At the top, it says 'My website Visit website' and 'Now editing English'. A yellow banner at the top right says '⚠ This demo resets every 2 hours.' Below that is a user profile icon. The main section title is 'Modules > Profiles > Overview'. It contains a search bar with 'E-mail' and 'Status' fields, and a large message box that says 'There are no items yet.' with a 'Update filter' button.

This screenshot shows the same Fork CMS dashboard as the previous one, but it has been modified to show a single profile entry. The main content area now displays a table with one row. The table has two columns: 'E-mail' and 'Status'. The 'E-mail' column contains the value 'demo@fork-cms.com'. The 'Status' column contains a dropdown menu with the value 'Active'. There is also a small 'Update filter' button at the bottom right of the table.

## 答案

create-new-account		
open	/private/en/authentication/index?offset=0&order=email&sort=asc&token=xj6o8c1744	
type	id=backendEmail	demo@fork-cms.com
type	id=backendPassword	demo
clickAndWait	name=login	
clickAndWait	css=li.nav-item.nav-item-modules > a > span.nav-item-text	
clickAndWait	link=Profiles	
clickAndWait	link=Add	
store	javascript{"joe+" + Math.floor(Math.random()*11111) ;}	myName
echo	\${myName}	
type	id=email	\${myName}@eee.com
type	id=displayName	\${myName}
type	id=password	11111
type	id=firstName	ee
type	id=lastName	ee
select	id=gender	label=Male
select	id=day	label=4
select	id=month	label=May
select	id=year	label=2012
type	id=city	eee
select	id=country	label=Dominican Republic
clickAndWait	id=addButton	
assertText	css=div.form-group	E-mail: \${myName}@eee.com
click	css=img.img-circle	
clickAndWait	link=Sign out	
assertTitle	Authentication – My website – Fork CMS	

# 安裝 VS Code

<https://code.visualstudio.com/>

# Javascript 基礎

## 宣告

```
var message;
```

## 字串

```
var message = '哈囉';
var name = "alincode";
```

## 數字

```
var num1 = 1;
var num2 = 1.5;
```

## 等於

- === : Strict equality
- == : Loose equality

```
var x = 5;
console.log(x == 5);
console.log(x == "5");
console.log(x === "5");
```

## 陣列

```
var people = [{  
    firstname: 'ailin',  
    lastname: 'liou'  
, {  
    firstname: 'Jane',  
    lastname: 'Doe'  
}  
];  
  
var lang = ['中文', '英文'];  
  
// 陣列長度  
lang.length;  
  
lang.forEach(function(item) {  
    console.log(item);  
});
```

## 函式

```
function greet() {  
    console.log('哈囉');  
}  
  
greet();
```

## callback 函式

```
var lang = ['中文', '英文'];  
  
function callback(item) {  
    console.log(item);  
}  
  
lang.forEach(callback);
```

# NPM 套件管理工具



## 初始化專案

### 語法

```
npm init [-f|--force|-y|--yes]
```

- 如果你加了 `-y` 或 `-f` 參數，代表你將認同使用預設的設定值來產生 `package.json` 檔。
- [init | nam Documentation](#)

### 範例

```
npm init  
npm init -y  
npm init -f
```

### package.json

- [package.json | nam Documentation](#)

```
{  
  "name": "demo",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\"$Error: no test specified\\" && exit 1"  
  },  
  "keywords": [],  
  "license": "ISC"  
}
```

## 練習題

使用 `npm init` 指令新增一個專案

```
mkdir demo  
cd demo  
npm init -y
```

## 安裝套件

- [install | npm Documentation](#)
- 別名 i

### 參數

- -g : 表示全域安裝
- --save : production
- --save-dev : development (預設)
- 什麼都沒加的情況

安裝到專案，並將依賴寫入 `package.json` 的 `devDependencies`

```
npm install --save-dev webdriverio
```

## 執行 script

### package.json

```
{  
  "scripts":{  
    "test":"node test.js"  
  }  
}
```

### test.js

```
console.log('Hello World');
```

# 介紹 Webdriver.IO

- [awesome-selenium](#)
- <http://slides.com/alincode/deck-3#/>

## 有兩種模式 (Mode)

### Standalone Mode (獨立執行模式)

```
var webdriverio = require('webdriverio');
var options = {
  desiredCapabilities: {
    browserName: 'firefox'
  }
};
webdriverio
  .remote(options)
  .init()
  .url('http://www.google.com')
  .getTitle().then(function(title) {
    console.log('Title was: ' + title);
})
  .end();
```

### The WDIO Testrunner

```
describe('測試', function() {
  it('測試一', function() {
    browser.url('http://demo.keystonejs.com/keystone/signin');
  });
});
```

## Test Runner

The test runner is an abstraction of popular test frameworks like Mocha, Jasmine or Cucumber.

### 支援的測試框架

- Mocha
- Jasmine

- Cucumber

# 設定 Selenium Server 環境

- 下載並設定，各種版本的瀏覽器與 selenium 對應的驅動程式。
- 下載並執行，Selenium Server。

## 安裝瀏覽器驅動程式 (driver)

至 <http://www.seleniumhq.org/download/> 下載

Browser					
<a href="#">Mozilla GeckoDriver</a>	<a href="#">0.16.1</a>	<a href="#">change log</a>	<a href="#">issue tracker</a>	<a href="#">Implementation Status</a>	Released 2017-04-26
<a href="#">Google Chrome Driver</a>	<a href="#">2.29</a>	<a href="#">change log</a>	<a href="#">issue tracker</a>	<a href="#">selenium wiki page</a>	Released 2017-04-04
<a href="#">Opera</a>	<a href="#">2.27</a>		<a href="#">issue tracker</a>	<a href="#">selenium wiki page</a>	Released 2017-04-04
<a href="#">Microsoft Edge Driver</a>			<a href="#">issue tracker</a>	<a href="#">Implementation Status</a>	
<a href="#">GhostDriver</a>	(PhantomJS)		<a href="#">issue tracker</a>	<a href="#">SeConf talk</a>	
<a href="#">HtmlUnitDriver</a>	<a href="#">2.26</a>		<a href="#">issue tracker</a>		Released 2017-04-04
<a href="#">SafariDriver</a>			<a href="#">issue tracker</a>		

## 安裝 Selenium Server

### 方法一

至 <http://www.seleniumhq.org/download/> 下載 selenium-server-standalone-3.4.0.jar 檔

#### Selenium Standalone Server

The Selenium Server is needed in order to run Remote Selenium WebDriver. Selenium 3.X is no longer capable of running Selenium RC directly, rather it does it through emulation and the WebDriverBackedSelenium interface.

Download version [3.4.0](#)

To run Selenium tests exported from IDE, use the [Selenium Html Runner](#).

To use the Selenium Server in a Grid configuration [see the wiki page](#).

### 方法二

```
// 安裝 webdriver-manager 模組到全域環境  
npm install webdriver-manager -g  
  
// 更新 driver  
webdriver-manager update
```

## 執行 Selenium Server

### 方法一

```
java -jar selenium-server-standalone-3.4.0.jar
```

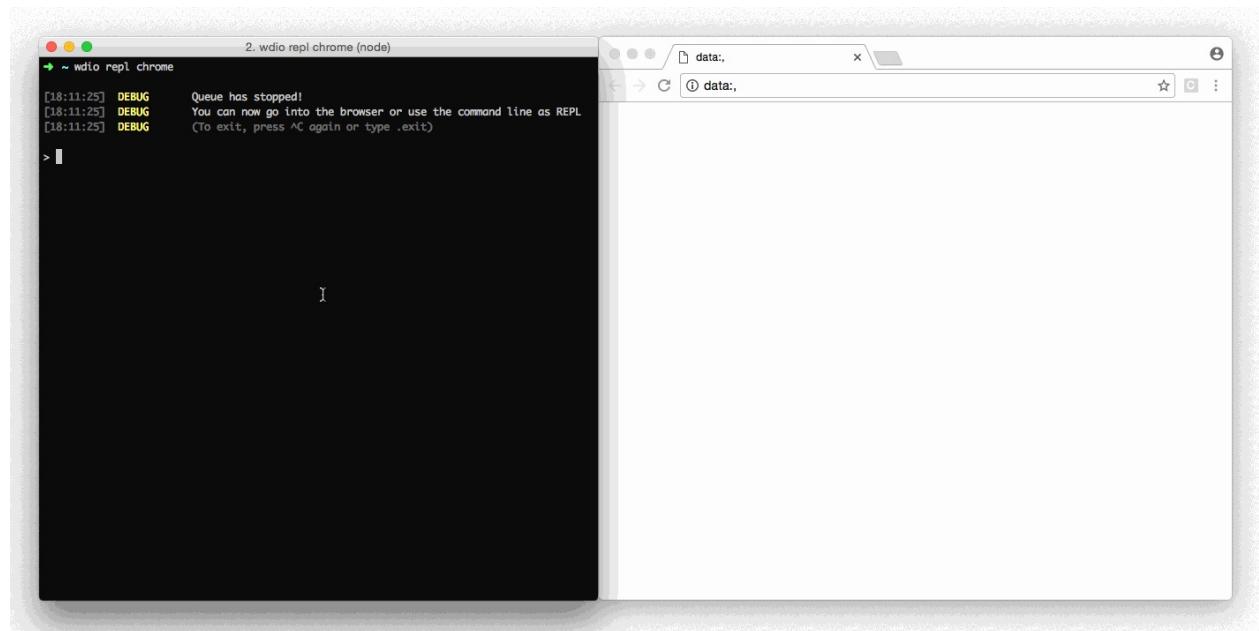
<http://127.0.0.1:4444/wd/hub>

### 方法二

```
// 啟動 selenium server  
webdriver-manager start  
  
// 查看 driver 狀態  
webdriver-manager status  
  
// 清除 driver  
webdriver-manager clean
```

# 用 REPL 來練習撰寫自動化程式

<http://webdriver.io/guide/usage/repl.html#description>



## 安裝 webdriverio 到全域環境

```
npm install webdriverio
```

## 啟動 REPL

### Mac

```
node_modules/.bin/wdio repl firefox  
node_modules/.bin/wdio repl chrome
```

### window

```
node_modules\.bin\wdio repl firefox  
node_modules\.bin\wdio repl chrome
```

## 實戰練習：取得 Dcard 的頭條

```
browser.url('https://www.dcard.tw/signup');
browser.getTitle();
browser.click('.SignupPage_forumBtn_2fGMq');
browser.getAttribute('#search input', 'placeholder');
browser$('.PostEntry_container_245XM strong').getText();
```

# 練習寫單元測試

```
npm install mocha -g
```

或

```
npm install mocha --save-dev  
node_modules/.bin/mocha
```

## 範例

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

describe('第一個單元測試', function() {

  it('加', function() {
    assert.equal(1 + 2, 3);
  });
});
```

## 執行方式

```
mocha first-unit-test.js
```

## 練習題

- 實作一個加減乘除的測試程式

# WebdriverIO 初始化專案

```
// 新增空的專案資料夾  
mkdir demo  
  
cd demo  
  
// 初始 化 npm 專案  
npm init -y  
  
// 安裝 webdriverio 模組  
npm install webdriverio --save-dev
```

## 產生 Webdriverio 設定檔

```
=====  
WDIO Configuration Helper  
=====  
  
? Where do you want to execute your tests? On my local machine  
? Which framework do you want to use? mocha  
? Shall I install the framework adapter for you? Yes  
? Where are your test specs located? ./test/specs/**/*.js  
? Which reporter do you want to use? spec - https://github.com/webdriverio/wdio-spec-reporter  
? Shall I install the reporter library for you? Yes  
? Do you want to add a service to your test setup? testingbot - https://github.com/testingbot/wdio-testingbot-service  
? Shall I install the services for you? Yes  
? Level of logging verbosity error  
? In which directory should screenshots gets saved if a command fails? ./errorShots/  
? What is the base url? http://localhost  
  
Installing wdio packages:  
  
Packages installed successfully, creating configuration file...  
  
Configuration file was created successfully!
```

```
// 執行 CLI (Mac)
$ node_modules/.bin/wdio

// 執行 CLI (Window)
$ node_modules\.\bin\wdio

=====
WDIO Configuration Helper
=====

選擇你的執行環境
? Where do you want to execute your tests? On my local machine

選擇你要使用的測試框架
? Which framework do you want to use? mocha

你要安裝測試框架的 adapter 嗎？
? Shall I install the framework adapter for you? Yes

設定你預計測試程式放置的資料夾位置
? Where are your test specs located?

選擇你想要的報表格式
? Which reporter do you want to use?

你要安裝報表函式庫嗎？
? Shall I install the reporter library for you?

你想要新增服務到你的測試專案嗎？
? Do you want to add a service to your test setup?

你要安裝服務嗎？
? Shall I install the services for you?

設定 log 的層級
? Level of logging verbosity silent

設定測試失敗後，截圖放置位置
? In which directory should screenshots gets saved if a command fails?

設定將要測試的網站的 domain url
? What is the base url?
```

## 執行方式

1. 編輯 `package.json`
2. 把 `script test` 指令設定為 `wdio wdio.conf.js`
3. 執行 `npm test`



# WebdriverIO 設定檔

## wdio.conf.js

```
exports.config = {  
  
    //  
    // =====  
    // 測試程式放置的位置  
    // =====  
    //  
    specs: [  
        './test/specs/**/*.js'  
    ],  
    //  
    // =====  
    // 設定要測試的瀏覽器屬性值  
    // =====  
    // <https://github.com/SeleniumHQ/selenium/wiki/DesiredCapabilities>  
    capabilities: [{  
        browserName: 'firefox'  
    }],  
  
    // 使用同步模式  
    sync: true,  
  
    // 設定 log 的層級 : silent | verbose | command | data | result | error  
    logLevel: 'error',  
  
    // 測試錯誤截圖放置的位置  
    screenshotPath: './errorShots/',  
  
    // 要測試的網站 domain 網址  
    baseUrl: 'http://localhost',  
  
    // 預設 waitFor* 檔案 timeout 的時間  
    waitforTimeout: 10000,  
  
    // 當 Selenium Grid 對於我們送的 request 沒有反應的時候，預設多久算 timeout。  
    connectionRetryTimeout: 90000,  
  
    // 預設重連線幾次  
    connectionRetryCount: 3,  
  
    // 使用的外掛  
    // plugins: {  
        //     webdrivercss: {
```

```
//           screenshotRoot: 'my-shots',
//           failedComparisonsRoot: 'diffs',
//           misMatchTolerance: 0.05,
//           screenWidth: [320, 480, 640, 1024]
//         },
//         webdriverrtc: {},
//         browserevent: {}
//   },
//
// 使用的服務
services: ['testingbot'],

// 使用哪一種測試框架來跑我們的前端測試程式
// 支援: Mocha, Jasmine, and Cucumber
framework: 'mocha',

// 報表格式
reporters: ['spec'],

// Mocha 的 Option 設定
mochaOpts: {
  ui: 'bdd',
  timeout: 600000
},
}
```

# WebdriverIO 常用指令 (API) 語法

<http://webdriver.io/api.html>

## 前往某網址

```
browser.url('http://www.google.com');
```

## 設定欄位的值

```
browser.element('.email').setValue('aaa@bbb.com');
// 縮寫
$('.email').setValue('aaa@bbb.com');
```

## 點選欄位的值

```
browser.click('.some-button');

// 縮寫
$('.some-button').click();

 $('[title="Sign Out"]').click();
```

## 檢查某個元素是否存在

```
browser.waitForExist('.alert-text');

// 縮寫
$('.alert-text').waitForExist();
```

## 取得某個元素的值

```
browser.getText('.alert-text');

// 縮寫
$('.alert-text').getText();
```

## 驗證

```
assert.equal('實際文字內容', '預期文字內容');
```

# WebdriverIO 實戰練習 - 登入登出

## 簡易測試規格

測試項目	測試描述	預期結果
登入失敗	...	顯示登入失敗訊息
登入成功	...	進入首頁
登出	...	顯示登出成功訊息

## 建立第一個前端測試程式

```
// 建立空的 test/specs 資料夾  
mkdir -p ./test/specs  
  
// 先把 Selenium Server 執行起來 (視窗一)  
webdriver-manage start  
  
// 然後再跑測試程式 (視窗二)  
npm test
```

## 帳號密碼

```
<http://demo.keystonejs.com/keystone/signin>  
帳號 : demo@keystonejs.com  
密碼 : demo
```

完成下面的前端測試程式

```
// 斷言函式庫
var assert = require('assert');

describe('第一個前端測試程式', function() {

  /*
  beforeEach(function() {
    browser.pause(8000);
  });

  */

  it('登入失敗', function() {
    browser.url('http://demo.keystonejs.com/keystone/signin');
    // 輸入帳號
    // 輸入錯誤密碼
    // 按送出按鈕
    // 檢查是否出現警告訊息
    // 警告訊息的文字內容，是否如預期
  });

  it('登入成功', function() {
    // 輸入帳號
    // 輸入正確密碼
    // 按送出按鈕
    // 檢查是否存在登出連結
  });

  it('登出', function() {
    // 點選登出
    // 檢查是否出現登出成功的訊息
  });
});
```

# Page Object 模式

## 目錄結構

```
.  
└── errorShots  
└── test  
    ├── pageobjects  
    └── specs
```

```
// test/pageobjects/page.js  
function Page () {  
    this.title = 'My Page';  
}  
Page.prototype.open = function (path) {  
    browser.url(path)  
}  
module.exports = new Page()
```

```
// test/pageobjects/login.page.js  
var Page = require('./page')  
var LoginPage = Object.create(Page, {  
    /**  
     * 定義元素  
     */  
    username: { get: function () { return browser.element('#username'); } },  
    password: { get: function () { return browser.element('#password'); } },  
    form: { get: function () { return browser.element('#login'); } },  
    flash: { get: function () { return browser.element('#flash'); } },  
  
    open: { value: function() {  
        Page.open.call(this, 'login');  
    } },  
  
    submit: { value: function() {  
        this.form.submitForm();  
    } }  
});  
module.exports = LoginPage;
```

```
// test/specs/login.spec.js
var expect = require('chai').expect;
var LoginPage = require('../pageobjects/login.page');
describe('登入流程', function () {
  it('登入成功', function () {
    LoginPage.open();
    LoginPage.username.setValue('alincode');
    LoginPage.password.setValue('12345678');
    LoginPage.submit();
    expect(LoginPage.flash.getText()).to.contain('恭喜你登入成功了');
  });
});
```

# 自行挑戰題 - 計算機

<http://juliemr.github.io/protractor-demo/>

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

describe('ex02', () => {

  /*
  beforeEach(function() {
    browser.pause(3000);
  });

  */

  it('加', () => {
    browser.url('http://juliemr.github.io/protractor-demo/');
    // 輸入第一個值
    // 輸入第二個值
    // 選擇計算方式
    // 按下按鈕
    // 等待結果回應
    // 比較結果
    assert.equal(0, 3);
  });

  it('減', () => {
    // 輸入第一個值
    // 輸入第二個值
    // 選擇計算方式
    // 按下按鈕
    // 等待結果回應
    // 比較結果
    assert.equal(0, 1);
  });

  it('乘', () => {
    // 輸入第一個值
    // 輸入第二個值
    // 選擇計算方式
    // 按下按鈕
    // 等待結果回應
    // 比較結果
    assert.equal(0, 6);
  });

  it('除', () => {
    // 輸入第一個值
  });
});
```

```
// 輸入第二個值  
// 選擇計算方式  
// 按下按鈕  
// 等待結果回應  
// 比較結果  
assert.equal(0, 4);  
});  
});
```