第一章 Google App Engine 介紹

- Web 應用程式
- Google App Engine 簡介
- 建置執行環境
- 使用 Eclipse 開發 Python 程式
- Hello World!
- 新增 Google App Engine 專案
- Google App Engine 專案部署
- DashBoard 效能檢視器 x

首先我們將讓讀者能夠了解什麼是 Google App Engine? 爲什麼我們要使用 Google App Engine? 以及該如何建置開發的環境、設定開發的 IDE 工具、啓動 Google App Engine 帳號及專案、撰寫並上傳執行我們的第一個 Google App Engine 專案並查詢檢視目前 Google App Engine 專案執行的效能。 有了基礎的認知之後就能進入後面的章節了解如何開發更複雜的 Web 應用程式來供網路上其他使用者來使用。

Web 應用程式

什麼是 Web 應用程式

什麼是一個 Web 應用程式 (Web Application)? 簡單地說就是一個提供特定服務能夠讓使用者在 Internet 或者 intranet 透過瀏覽器來讀取、互動的應用程式。 更明確地來說，開發人員利用瀏覽器所支援的語言 (例如. HTML、Java Script、Java、PHP 等等) 來開發的一個應用程式，且能夠透過瀏覽器來產生執行結果，即稱之爲一個 Web 應用程式。

傳統的 Web 網頁能夠提供使用者瀏覽大量的文字或者圖片資料。 然而隨著越來越多的網頁技術推陳出新，以及 Web 2.0 時代的來臨，Web 網頁所扮演的資訊提供者 (provider) 已經不足以滿足使用者的需求；使用者需要的是能夠與其『互動』的網頁而不僅僅只是靜態的網頁呈現。 因此，越來越多開發人員投入 Web 應用程式開發的領域，不管是底層基礎架構 (framework) 的設計、後端網頁程式設計甚至到了前端互動技術(例如. AJAX, Asynchronous JavaScript and XML)；各大軟體公司也都針對 Web 應用程式平台由上到下 (Top-Down) 提供完整的解決方案* ，來協助

技術人員更快速地開發 Web 應用程式。

*如 Microsoft 微軟目前最新的網頁應用程式版本 ASP.NET 3.5 ，從底層作業系統 Windows Server 2008、Web 伺服服務 Internet Information Service (IIS)及資料庫伺服器 SQL server、到上層.NET framework 提供包含伺服器端服務及客戶端介面設計等非常豐富的 Library 支援， 還有 Visual Studio .NET 幫助程設人員快速開發 Web 應用程式；另外如 SUN 昇陽公司推出的 JSP，搭配 Java EE 能夠跨平台執行，對於 MVC(Model-View-Control)也有非常多的支援(如 Apache Struts 及 Spring MVC framework)，而在開發環境方面也有 Eclipse for Java，提供程式開發人員非常便利且完整的支援。

建置、發表一個 Web 應用程式

- **硬體設備及作業系統**

該如何建置一個 Web 應用程式並發表在 Internet 或者 intranet 上供其他使用者使用呢? 我們必須要先建置一個網路伺服器來提供服務。 首先當然是得先準備伺服器的硬體及所需之網路設備，如果要提供服務給數量非常龐大的使用者可能就得設置叢集式電腦或者專業伺服器等級電腦、額外的網路儲存設備(NAS, Network Attached Storage)、高頻寬的網路設施甚至是備援系統來維持不中斷的服務。 此外電源供應也是一個大問題，如何供應如此龐大耗電的硬體設施以及提供足量的 UPS (Uninterruptible Power Supply) 電力備援等都是需要考量到的議題。

準備好硬體設備、插上插頭打開電源後下一步就是安裝作業系統了；目前的各種作業系統諸如 Microsoft 的 Windows Server 系列、SUN Solaris、各種 Linux 發行套件甚至是 Apple 的 MAC OS X 都能夠來當作網路伺服器提供服務。 此外若是想要建置分散式系統的環境則需採用分散式電腦叢集系統以及分散式檔案系統來提供服務給更多的使用者，目前也有非常多開放原碼專案可以使用；然而相對地必須花費非常多的時間與人力在分散式系統的建置與維護上，如果將來越來越多使用者來使用而增加系統的負擔，還需要考慮到擴充性以及系統配置的問題，這些問題對於系統管理員都是非常令人頭痛的。

- **Web 伺服器及資料庫系統**

接下來就是選擇合用的 Web 伺服器了。 目前最普遍的 Web 伺服器就是 Apache Web 伺服器，此外由 Microsoft 所提供支援的 IIS (Internet Information Service) 也是一個不錯的選擇；若是想要利用 Microsoft 的 ASP/ASP.NET 來開發 Web 應用程式，則使用後者能帶來最大的支援度，然而則須考量使用 Microsoft 系統可能涉及的使用權利金付費問題。 而若採用 Open source 的 Apache Web 伺服器，目前網路上也有諸多論壇可以查詢相關問題，或者直接向 Apache 官方網站提出諮詢要求。

建置完硬體設備以及作業系統及 Web 伺服器的設定後，如果您還需要提供資料庫的服務，那就得再安裝資料庫伺服器。 目前最常被使用的有 Microsoft SQL server、MySQL、SyBase、Oracle 等等，這些資料庫系統的效能都非常強大，也都提供非常友善的使用者介面以及支援 ODBC (Open Database Connectivity)，程式開發人員只要送出符合 SQL 語法的查詢即可快速地搜尋檢索資料庫內的資料。 然而，這些商業級的資料庫系統的授權使用金額不低廉，隨著授權使用節點數或者資料庫內容數量來計費，若是應用於分散式環境或者大量資料的環境，則光是在資料庫系統的支出就所費不貲了。

- **DNS 及 SMTP 服務與防火牆機制**

爲了能夠讓其他使用者搜尋到自己所提供的 Web 應用程式之服務，就得申請 IP 位置以及網域名稱對應到自己的伺服器，目前有非常多的 ISP 皆有提供 IP 轉網域名稱的服務；此外如果想自行配置副網域指向到其他主機來分散負載，就得需要再自己架設 DNS 伺服器來負責轉址的工作。 若是網站服務需要使用者 e-mail 認證，SMTP 伺服器的架設也是必要的工作。 然而沒有受到完整保護的 SMTP 伺服器常常被怪客(cracker)所入侵利用來當作寄發垃圾信件的跳板，可能造成專案的負責人涉入濫發垃圾信件的糾紛當中，因此專業且完善的防火牆系統的建立也是非常必要的。 甚至自行架設的伺服器如果沒有做好定期的系統漏洞更新與維護，很容易遭到有心人士駭入，輕則網頁被修改、重則全部資料都外洩或混損，對於企業網站來說都是非常大的傷害。 因此如何建置好伺服器的防火牆都需要非常專業的人士來負責維護，不管是自行處理或者外包專業公司，都是不小的額外負擔。

- **虛擬主機服務**

當然還有更快的方式且更安全、穩定的方式來建置 Web 伺服器的執行環境，那就是透過目前許多虛擬伺服器提供者所提供的虛擬主機環境。這些提供者能夠提供非常完整的且全日的安全機制來保護使用者的 Web 應用程式內容不會遭到惡意使用者的入侵、修改、竊取，甚至有額外的電源備援機制來支持開發者的 Web 應用程式提供 24*7 不停機的持續性服務；還有自有網域名稱轉址以及 SMTP 信件轉寄服務；完整防駭機制及 24 小時全天候監控系統狀況避免伺服器遭受駭客惡意入侵破壞；此外高速的網路頻寬、高運算效能的 CPU 和大容量的記憶體及硬碟空間，再加上作業系統及網路伺服器的合法使用權和即時更新，讓開發人員無後顧之憂地專注於自己所開發的 Web 應用程式上。

天下沒有白吃的午餐，想要享用這樣安全、快速、穩定的開發、執行、服務環境以及完善的支援，您也需要更多的小朋友來支援您! 如果只是想要測試一個小小的 Web 應用程式的功能也得付出這麼大的成本嗎? 難道沒有又便宜又穩定，且支援度高的 Web 應用程式開發環境供開發人員來開發、部署自己開發的 Web 應用程式

且又有免費的資料庫資源能夠來存取並搜尋呢?

Google App Engine 簡介

Google App Engine 是什麼?

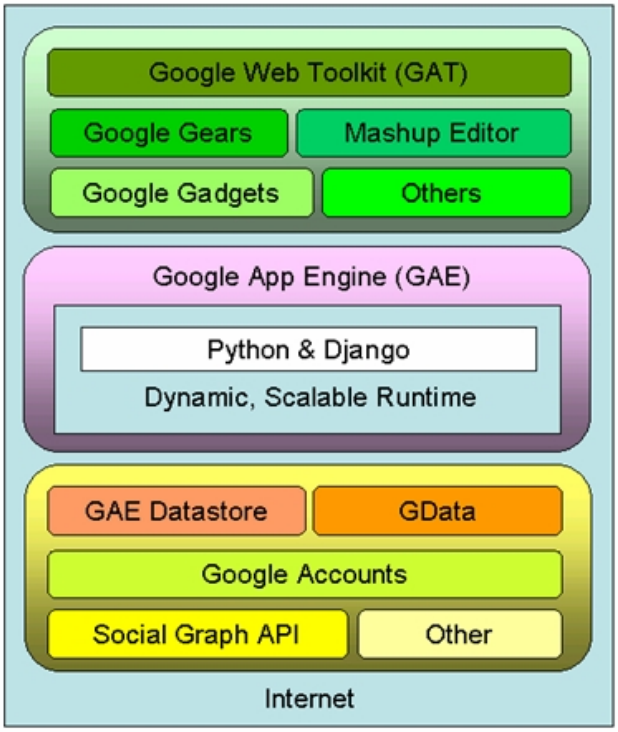
Google App Engine 簡單地說就是一個提供開發人員建置及架設 Web 應用程式網站的平台。 承續上一節，Google App Engine 想要提供使用者的，就是這樣一個這樣具備安全、快速、穩定的開發、執行、服務環境。 透過 Google App Engine 平台，開發人員能夠專注於開發自己的 Web 應用程式而無須耗費心力於平台的建置與管理，更無須花費任何金錢即可享有由 Google 提供的雲端運算環境所帶來的強大運算資源、大量儲存空間以及高速頻寬，更重要的是使用 Google 帳號以及 Google 所提供眾多服務的完整支援。 目前 Google App Engine 平台僅支援 Python 語言來開發 Web 應用程式，其他不同語言的執行環境則需等 Google 釋出支援。

Google App Engine 平台對於每一個 Web 應用程式的免付費使用限制爲：每個 Web 應用程式可使用 500MB 儲存空間、每一個月每個 Web 應用程式 5 百萬個頁面瀏覽數上限(目前國內的許多熱門網站也鮮少能夠突破這個瀏覽數量，所以可以不必擔心瀏覽數上限的問題)及每天 2 億個 CPU clock cycle 運算。 下表是 Google App Engine 與國內知名虛擬主機提供商所提供之部分服務內容比較，孰優孰劣我們並無下定論僅供讀者參考，端看使用者個人需求以及預算上限而自行選擇。

	PC home (Linux 進階型)	智邦虛擬主機 (Linux 進階型)	Google App Engine (Fixed Quotas)
作業系統	Linux	Linux	Google 雲端計算平台
程式語言	CGI/PHP/Perl	PHP/Perl	Python+Django
運算資源限制			每日 2 億個 CPU clock cycle 運算
資料庫使用	MySQL	MySQL	Google BigTable
硬碟空間	5.0 GB	5.0 GB	500 MB
每月免費流量	25GB	30GB	每日 10GB*30 = 300GB
管理介面提供	有	有	有
24 小時客服	有	有	無
收費	NTD$ 6000 / year	NTD$ 4000 / year	Free!

Google App Engine 於 2008 年四月由 Google 釋出，目前也提供了有興趣的開發人員免費申請使用，每個人最多可以在 Google App Engine 平台上建置、部署 10 個

自己的 Web 應用程式。 即使沒辦法連上網路，只要安裝好 Python 以及 Google App Engine 執行環境，開發人員也能在自己的電腦上編譯程式碼並以瀏覽器來檢視自己撰寫的 Web 應用程式執行結果(當然若是使用到需要 Google App Engine 提供的服務則可能無法使用)，非常地方便。



Google Cloud 平台架構，參考圖片自 http://news.cnet.com/8301-13953_3-9917409-80.html。

使用 Google App Engine 的其中一項好處即是開發自己的 Web 應用程式可以和其他 Google 的 Web 技術緊密結合甚至善用這些資源來加快專案的開發時間。 例如前端 Web 應用程式介面有 Google Web Toolkit 加快使用者開發專案、Google Gears 能加入離線使用功能讓使用者在沒有網路連線的環境也能在本機端上操作大部分自己的資料；而後端則有強大的 Datastore 支援存放並檢索大量資料 (這在後面章節會再詳細介紹)、Google Accounts 讓所有 Google 使用者登入後即可使用 Web 應用程式的所提供的服務等等。 下一小節我們將簡介 Google 雲端計算平台以及其背後支援的技術。

Google 雲端計算平台

雲端計算(Cloud Computing)是電腦運算上一個新的"概念"而非一個新的"技術"，本質上其實就是分散式運算；只是一個概念利用不同的方法去實踐。

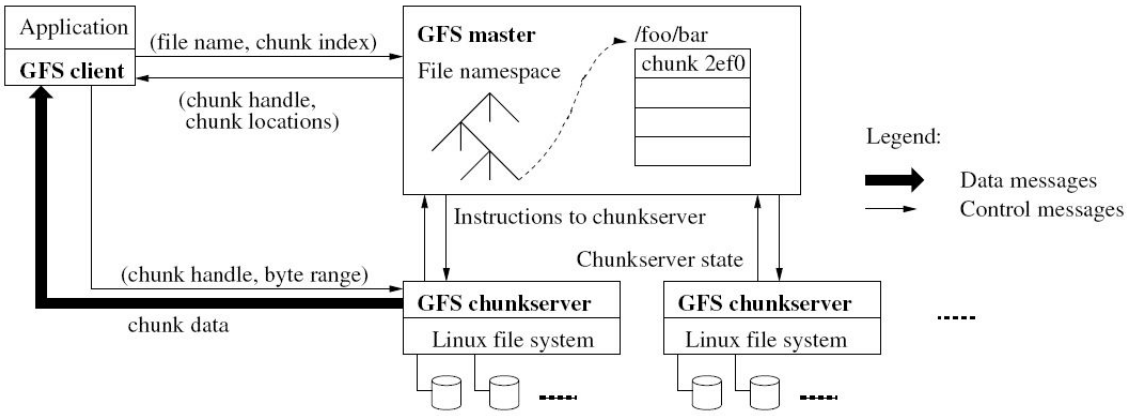
雲端計算經常會被人與格網計算(Grid Computing)或者公用運算(Utility Computing)搞混，然而雲端計算最基本的概念則是將需要計算的問題作切割，分成無數較小的子問題，派送給後端由大量伺服器所組成之系統來運算，最後再將結果作整合並回傳給客戶端；對於使用者而言，根本無需知道這樣龐大的計算與儲存資源從何而來，彷彿從天邊的雲端取得需要的資料或者計算結果，只要輸入簡單的指令即可獲得如此大量的計算結果，這也是雲端運算目前非常火熱的一大主因。

Google 的雲端計算平台其實已經進入到使用者的生活當中。 相信大部分的使用者都曾經使用過 Gmail、Google Talk、Google Calendar、Google Docs 等等服務。使用者只要在任何一台電腦上，打開瀏覽器連上網路輸入網址，即可從 Google 雲端平台取得自己的 Mail、行事曆內容、文件等等資料，完全無須知道資料來自哪台伺服器；除了 Google 所提供免費且大量的頻寬、空間、運算效能，還有非常人性化的介面，不管在哪一個平台上，只要有瀏覽器就能像在自己的電腦上操作那樣便利。

Google 雲端計算平台建置在三個重要的核心之上：Google File System、MapReduce 及 BigTable。 底下我們會簡短地介紹這些核心技術的內容；此外還會再介紹 Google App Engine 以之爲基礎而開發的 Django Web framework：

- **Google File System**

Google File System 是 Google 在 2003 年所設計、實作且已經應用於 Google 的雲端運算的一個具有高延展性之分散式檔案系統，對於存取極大量資料的大型應用程式能提供高度容錯性及高效能。 別於一般分散式檔案系統多架設於大型主機及伺服器等級硬碟之上，由 Google 所建置之 Google File System 的分散式檔案系統，是由極大量價廉的個人電腦等級的平價電腦叢集(PC Cluster)所組成；每一份儲存在 Google File System 的資料會被製作三分以上的副本(replication)存放在不同的機器上，一旦其中有一台電腦毀損，也能確保資料不至於因而與之毀損。



Google File System 架構圖。 節錄自 The Google File System, SOSP '03.

- **MapReduce**

MapReduce 是由 Google 所提出的一種適用於在大量叢集電腦集合中執行分散式運算的計算軟體架構，將需要計算的問題－特別是針對大型檔案，無論是存放在一般檔案系統或者資料庫系統－分成 map 及 reduce 兩個階段來處理。

Map 階段由主要節點將需要計算的問題切割成數個子問題，再將這些子問題派送給其他的工作節點來運算；如果這些子問題計算量仍太大，工作節點也可再繼續切割子問題成爲更小的問題群組來派送給其他工作節點運算，形成樹狀的多層次架構。而每一個被切割產生的子問題，都必須與其他子問題之間具有獨立的關係彼此不相依，如此一來每個子問題便能同時且平行地被處理完畢。 Reduce 階段最主要的目的就是回收這些已經計算完成的結果並依照其對應的工作節點劃分之群組將它們結合成爲單一輸出，也就是最原始的問題之答案。

MapReduce 在某些問題的處理上可能不會比直接利用序列式(sequential)計算要來得有效率，然而應用於 Google 雲端計算平台利用數以萬計的平價電腦組成的架構上，則能夠有非常高的經濟效益，無須大型伺服器主機來專一處理一個問題，更能同時分散地處理多個大量的計算問題，也更加符合雲端計算將欲解決的問題切割成爲大量較小的子程序來分散計算之主要宗旨。

- **Big Table**

Big Table 是由 Google 針對結構化的資料設計、開發的一套高延展性的分散式極大量資料(petabytes)儲存系統。 前面章節介紹 Google File System 時我們有提到，Google 利用成千上萬台平價電腦來組成大型分散式檔案系統儲存使用者的資料，而 Big Table 更是針對這樣的特性而設計的一套高容錯、高延展性的儲存系統。 且目前已經有多個 Google 提供的服務是基於 Big Table 系統之上：Google Analytics(網站訪客行爲分析及最佳化工具)、Google Finance、Orkut(Google 社群服務)、Personal Search、Writely(Google 文件, docs)、Google Earth 等等，都非常穩定且有效率地服務來自全球大量使用者的要求。

使用者在 Google App Engine 平台上開發的應用程式及資料皆存放在 Big Table 中，也因此使用者的 Web 應用程式以及資料儲存量能夠不受限於伺服器的規模而無限延伸；此外不管資料量有多龐大，當 Web 應用程式提出資料存取要求時，Big Table 也能提供 Google 資料搜尋同等級的高效能資料搜尋速度來支援 Web 應用程式的運作。 相信這也是使用者選擇 Google App Engine 平台來開發個人的 Web 應用程式的一大利基。

- **Django**

Django 爲一種以 Python 程式語言爲基礎的開放原碼 Web 應用程式基礎架構，發音爲//ˋdʒæŋgo://，最早是由美國堪薩斯州羅倫斯市的老牌報社 The World Company 在 2005 年基於 BSD License 釋出的新聞網站系統。

Django 最主要的目的在於能夠減低新聞網站使用資料庫的複雜性、加快網頁開發的速度，快速地將新聞內容發布在網站上爭取時效；Django 也強調系統元件的再用性(reusability)及可掛載性(pluggability)，以系統的簡潔以及快速發展爲最主要的訴求。利用 Django 本身之特點來當作 Web 基礎架構(framework)開發有以下的特點：

■ 物件關聯式對照(Object-relational mapper)

在一般 Web 應用程式(如 PHP, ASP.NET)，在儲存資料時必須先利用 SQL 指令來新增一個資料表以及其內容欄位，新增完畢之後則仍需要透過 SQL 指令來存取資料；在 Django 的 Object-relational 對應的機制下，所有的資料欄位內容，全都定義成 Python 的類別，當一個定義完整的類別被實作出來(也就是大多數物件導向語言的 new)，這樣的資料模型就已經被實際產生出來，再透過簡單的 put()與 get()便能在後端的資料庫中寫入及讀出這些定義好的資料了。

而關聯式資料庫中資料表的關係也能利用物件與物件之間的關連來表示，甚至也支援 SQL 語言，使用者仍能下 SQL 指令在後端的資料庫中搜尋相關的資料模型內容。

■ 樣板功能(template)分離視覺與控制元件

視窗或 Web 應用程式於開發時往往在外觀呈現與程式邏輯控制的程式碼之間會有太過密切牽扯，而造成往後維護時的困擾。 因此樣板是能夠將這兩部分元件給精準切割的重要功能，能夠讓設計使用者介面(UI)以及程式邏輯控制的開發人員各自開發自己撰寫的程式內容，之後再依照共同定義的標籤來將兩部分嵌合在一起。 這樣除了能夠平行發展，於日後程式維護或者除錯時，也能很清楚地分出問題所在癥結點。

■ 快取(cache)機制增進服務效能

快取機制在計算機中一直都扮演著增進系統效能的角色。 Django 也支援將常用到的資料內容加入到自己的快取當中，尤其是經常需要到後端資料庫中讀取或者可重複被存取的複雜計算結果，不但能節省資料庫 IO 的時間，更能使得複雜的運算結果只要計算一次便能被重複地利用，使系統效能能夠爲其他需要的應用程式所利用。

- 管理介面(Automatic admin interface)與多國語系(Internationalization)
  Django 本身就提供簡單的指令即可自動產生的應用程式(如自動產生簡易版本的 blog)，包括管理介面；程式開發人員在開發會員管理系統時便可直接產生介面，無須再花時間去撰寫複雜的程式碼內容。 此外 Django 也支援多國語系，利用樣板產生多國語言版本的網頁，對於走向國際化的 Web 應用程式在開發上具有非常大的助益。

由於 Google App Engine 是基於 Django Web framework 來發展，這也是爲何我們會在此著墨這麼多的原因，因此上述的這些優勢在稍後我們開始實際開發 Google App Engine 專案時都會在實際介紹到。 而且對於原本開發 Django 專案的程式開發人員來說，只要經過小部分的修改與調整，便能將自己的 Djang 專案快速地移植到 Google App Engine 平台上執行了。

下個小節我們則準備開始建置執行環境，之後便能實際動手來撰寫 Gpogle App Engine 程式了。

建置執行環境

建置開發環境

要開始動手撰寫自己的 Google App Engine Web 應用程式之前，必須先確認目前的工作平台上有 Python 的執行環境以及 Google App Engine 的 SDK (Software Development Kit)，以下就分別就 Microsoft Windows 平台 (以 Windows XP 爲例) 、Linux 平台 (以 Ubuntu 8.10 爲例) 以及 Mac OS X 如何建置 Python 執行環境以及 Google App Engine SDK 作簡單的介紹。

安裝 Python 執行環境

由於 Google App Engine 目前僅支援以 Python 撰寫之 Web 應用程式，想要開發且測試 Google App Engine Web 應用程式得先安裝 Python 的執行環境。 首先到 http://www.python.org/ 下載最新的 Python 執行環境安裝檔。 目前筆者撰寫時最新的版本爲 3.0 rc2 以及 2.6 final 兩種版本；但依照 Python 官方網站上的說明 Python 3.0 與 Python 2.x 系列爲不同的兩條分支且 3.0 並不相容於 2.x 系列，而 Google App Engine 的文件說明表示支援 **Python 2.5** 以上的版本，因此建議讀者下載 Python 2.6 final 版本的執行環境下來安裝，避免不必要的執行錯誤結果。

- **於 Windows 平台建置 Python 執行環境**

在 Microsoft Windows 上安裝就簡單得多了，只要下載適用的安裝程式 (Windows 平台有 x86 及 AMD64 兩種版本供下載) ，按照預設選項下一步、下一步就能夠完成安裝。

- **於 Linux 平台建置 Python 執行環境**

在 Linux 上(在此以 Ubuntu 8.10 LTS 版本爲例)，一般的套件版本都已經在作業系統安裝時就已預先安裝 Python 的執行環境。可開啓終端器輸入

```
user@localhost:~$ python -V          # 請注意爲大 V，其他指令可下 python -h 查詢
Python 2.5.2
```

來檢視目前系統所安裝的 python 版本，在此爲 Python 2.5.2 版。 在 Ubuntu 上可利用 apt-get 來安裝

```
user@localhost:~$ sudo apt-get install python
```

- **於 Mac OS X 建置 Python 執行環境**

Mac OS X 在預設的情況下並沒有 Python 的執行環境，首先下載 Mac OS X 上的 Python 執行環境安裝檔，安裝過程其實和在 Windows 平台上安裝差不多， 下載完 python-2.6-macosx.dmg 後，Mac OS X 作業系統會自動解開到 Universal MacPython 2.6 資料夾，接著執行 MacPython.mpkg 檔即可開始安裝，只要一直點選下一步就能完成安裝。

安裝 Google App Engine SDK

Google App Engine SDK (Standard Development Kit)是用來開發、測試、部署 Google App Engine 專案所必須的程式，包含了 Python 語言的直譯器、Python 標準函式庫、Google App Engine 的函式庫和 API 以及 Web server layer 標準介面。其中最主要也最常用到的兩支 Python 程式爲：

dev_appserver.py：可在本機端執行測試撰寫完成的專案；
appcfg.py：用來將撰寫完成的專案上傳到 Google App Engine 平台。

使用者可到 http://code.google.com/appengine/downloads.html 下載 Google App Engine SDK。 以目前筆者撰稿時最新的版本爲 1.1.9 版本，且 Google

App Engine SDK 更新版本速度頗快，也有針對不同作業系統所提供的不同版本，只要下載適合自己的作業平台的最新版本安裝即可。

- **Windows 平台建置 Google App Engine SDK**

下載 Windows 版本的 Google App Engine SDK，一樣是一直下一步即可完成安裝。 預設是安裝於 C:\Program Files\Google\google_appengine 目錄底下，且安裝程式會於桌面上建立一個安裝目錄的捷徑方便使用者來讀取；此外還會在系統的路徑參數上加入 SDK 的安裝目錄，如此一來使用者只要打開終端機模式即可直接下指令來執行程式，無須指定完整正確的目錄來執行。

- **Linux 平台建置 Google App Engine SDK**

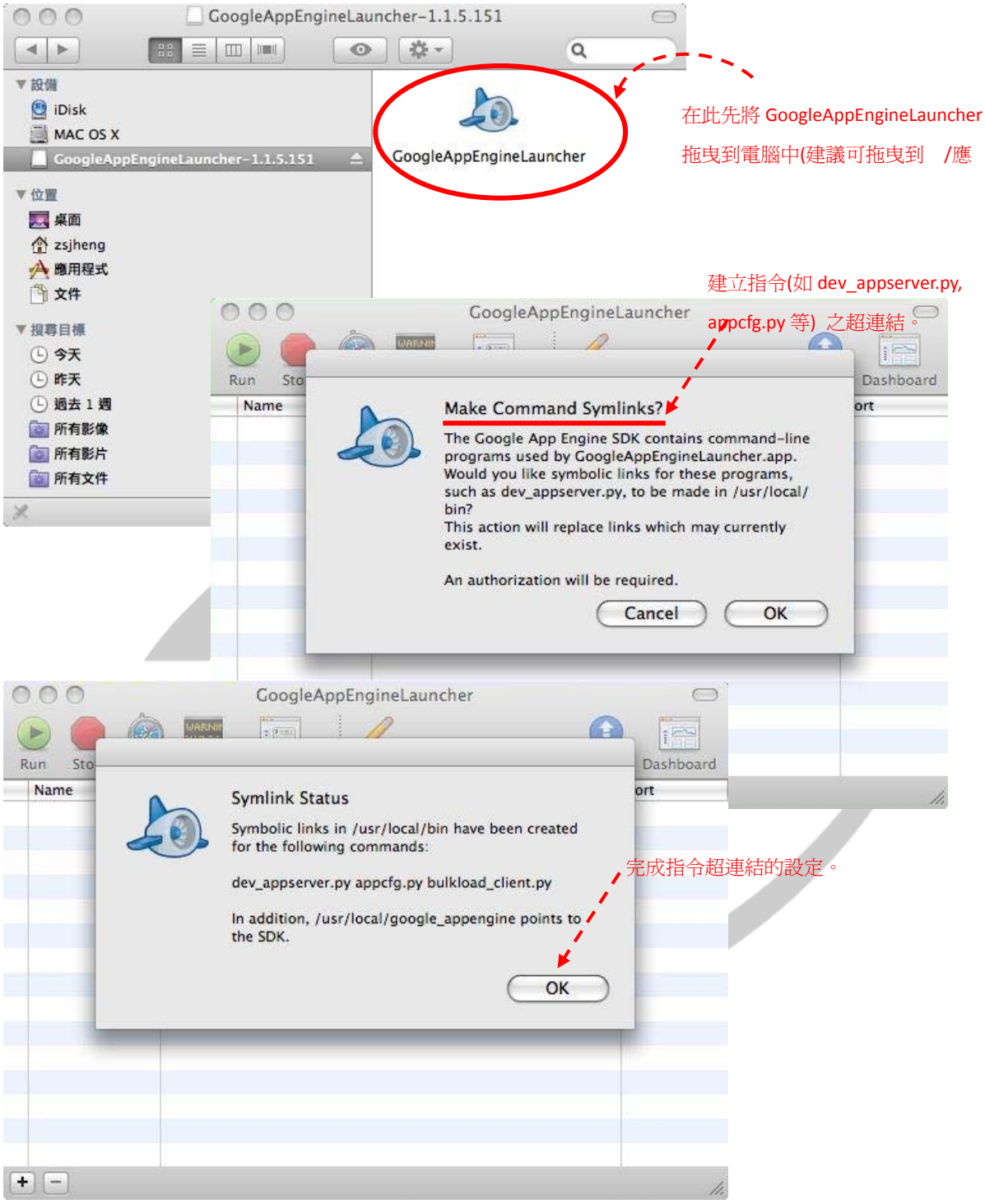
在 Linux 上建置 Google App Engine SDK 就簡單地多，只要下載 Google App Engine SDK zip 檔，下解壓縮指令解壓縮，之後會在原目錄下產生一個 google_appengine 目錄，其中就包含了開發 Google App Engine Web 應用程式所需的執行程式。

```
user@localhost:~$ unzip google_appengine_1.1.9.zip
```

讀者可以將 Google App Engine SDK 解壓縮到任何的目錄，只要在稍後開始開發 Google App Engine 專案時記得將 SDK 的目錄路徑引入即可。

- **Mac OS X 建置 Google App Engine SDK**

Google App Engine SDK 針對 MAC 平台提供了一套非常便利的上傳系統－Google App Engine Launcher。 在下載完畢之後，MAC OS X 便會自動解開封包套件，使用者可以點選 GoogleAppEngineLauncher 的圖示來啓動 Google App Engine SDK 的安裝。 GoogleAppEngineLauncher 的安裝程式會先爲 SDK 中比較最常使用到的三個 python 程式：dev_appserver.py、appcfg.py 以及 bulkload_client.py 建立連結到 /usr/local/bin 當中，這樣使用者也可以直接在終端機模式下直接下這些指令來執行工作。



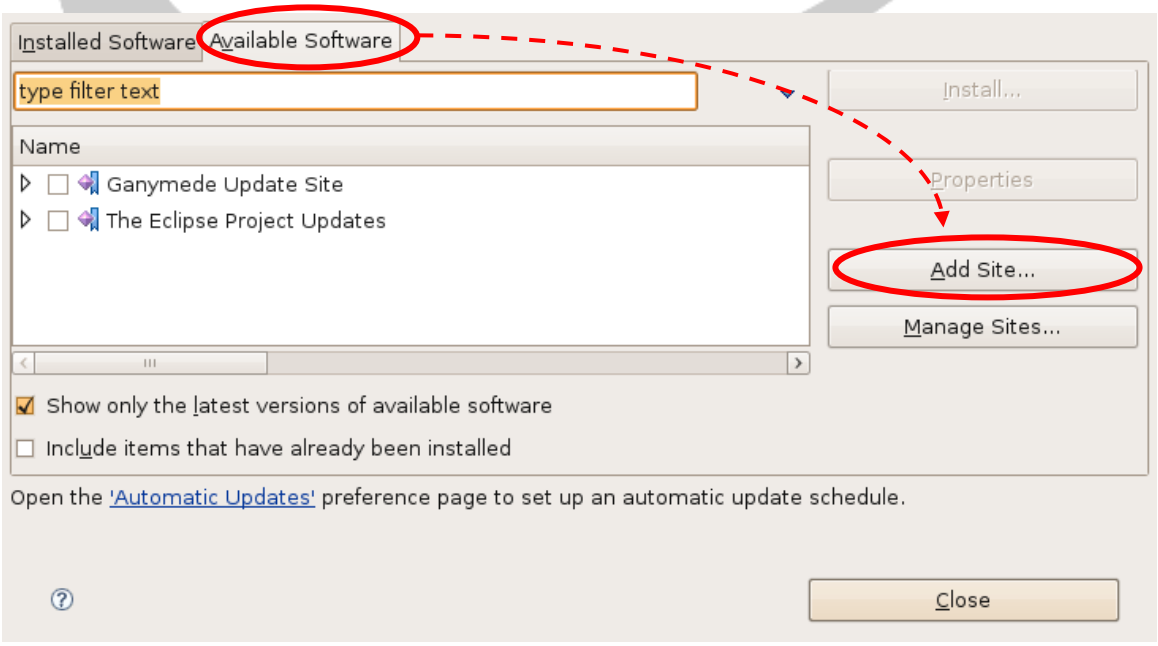
GoogleAppEngineLauncher 提供了其他平台版本的 SDK 所未提供的便利功能，即是圖形化的專案上傳及測試程式，只要將自己的專案加入到 GoogleAppEngineLauncher 的管理視窗當中，當專案內容有更新需要測試或者上傳到 Google App Engine 平台時，只要按個按鍵就可以輕鬆地完成這些工作，這些功能我們將留在後面章節再來介紹。

使用 Eclipse 開發 Python 程式

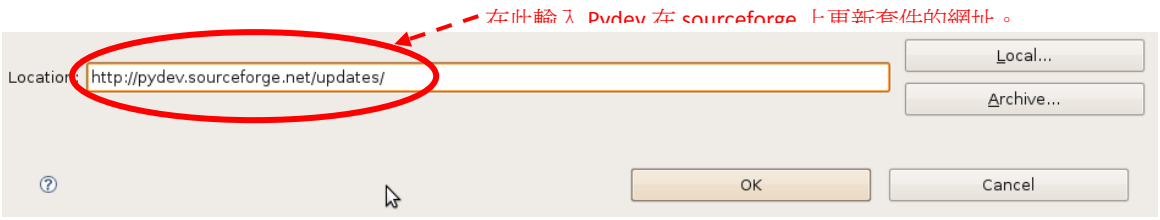
在安裝完 Python 的執行環境以及 Google App Engine 的 SDK 之後，已經可以開始發開自己的 Google App Engine Web 應用程式了。 然而，Google 並未提供類似微軟 Visual Studio .NET 這樣的整合工具來協助開發人員來開發，因此我們只能利用文字編輯器加上 Google App Engine SDK 所提供的應用程式來在本機端測試及上載到 Google 平台執行。 不過使用者仍能可以利用許多現有自由軟體的 IDE(Integrated Development Environment)來開發自己的 Google App Engine Web 應用程式，Eclipse 就是一個非常好的選擇。

Eclipse 爲一個開放原始碼社群，主要目標爲開發及建置一個開放的程式開發平台；Eclipse 的程式開發平台能夠擴充新基礎架購(framework) 、開發工具和編譯、部署及管理之執行環境(runtime)，幫助程式人員快速開發專案。 我們希望能夠藉助 Eclipse 的優點來協助我們來開發 Google App Engine 的程式。 然而 Eclipse 本身並不支援 Python 專案的開發，爲了使 Eclipse 能夠開啓並讀取 Python 專案，我們必須加入額外的外掛程式(plug-in)。 支援 Python 專案的外掛不少，在此我們則會介紹如何安裝 **Pydev** 這個有不少 Python 愛好者使用的外掛程式。

首先，先開啓 Eclipse(在此我們以 ubunt Linux 平台爲例，Windows 與 MAC 平台的設定步驟則大致相同)。 選擇功能表上 **Help** 選項中的 **Software Updates...** 的選項打開 **Software Updates and Add-ons** 視窗準備新增 Pydev 的外掛。

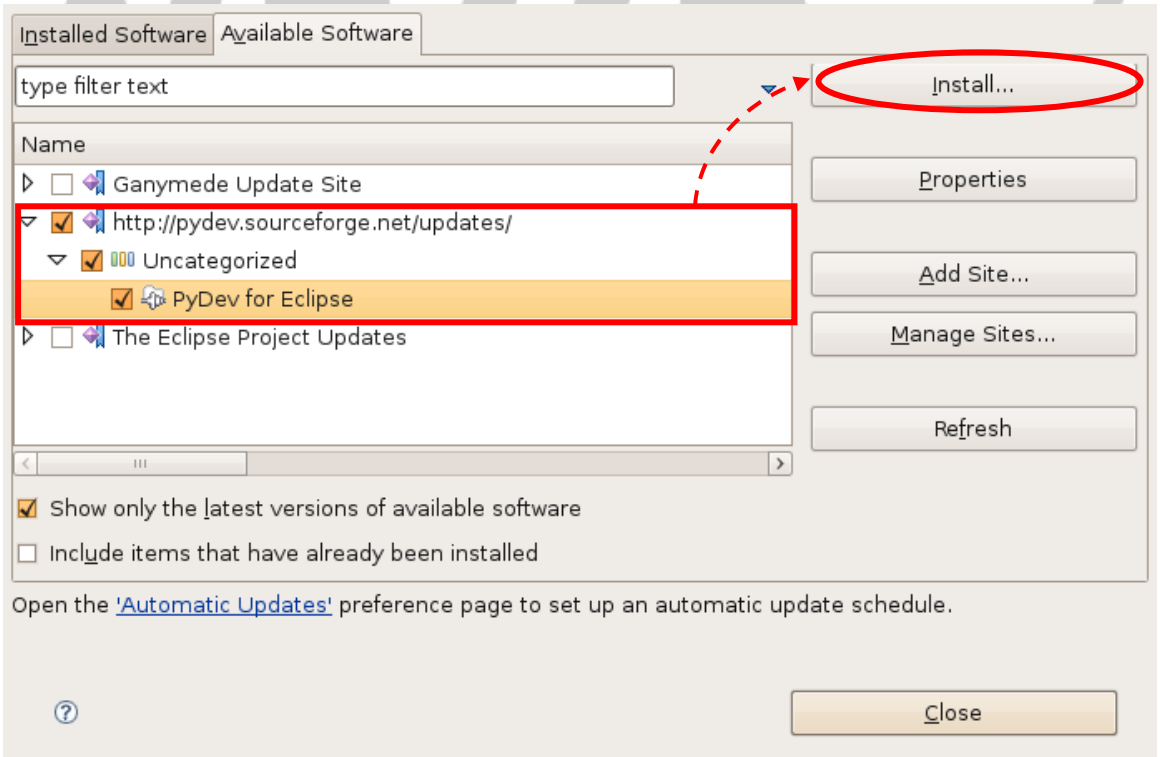


選擇 **Available Software** 的分頁，按下 **Add Site…** 按鈕來新增 Pydev 的下載網址： **http://pydev.sourceforge.net/updates/** ，按 ok 後 Eclipse 則會將這個更新網址加到自己的軟體更新名單中。



新增軟體更新名單後可以在軟體列表中看到新增的項目 **http://pydev.sourceforge.net/updates/** 。 打開 **pydev** 選項後可以看到 **Pydev** 選項*，最基本的開發 Python 程式環境必須要勾選 **Pydev** 的安裝選項。

*在某些版本的 Eclispe 可能會出現 **Pydev Optional Extension (requires Mylyn)** 的這個選項，這個並不影響功能因此可選可不選；但是 Pydev 是必須選取的才能正常運作。



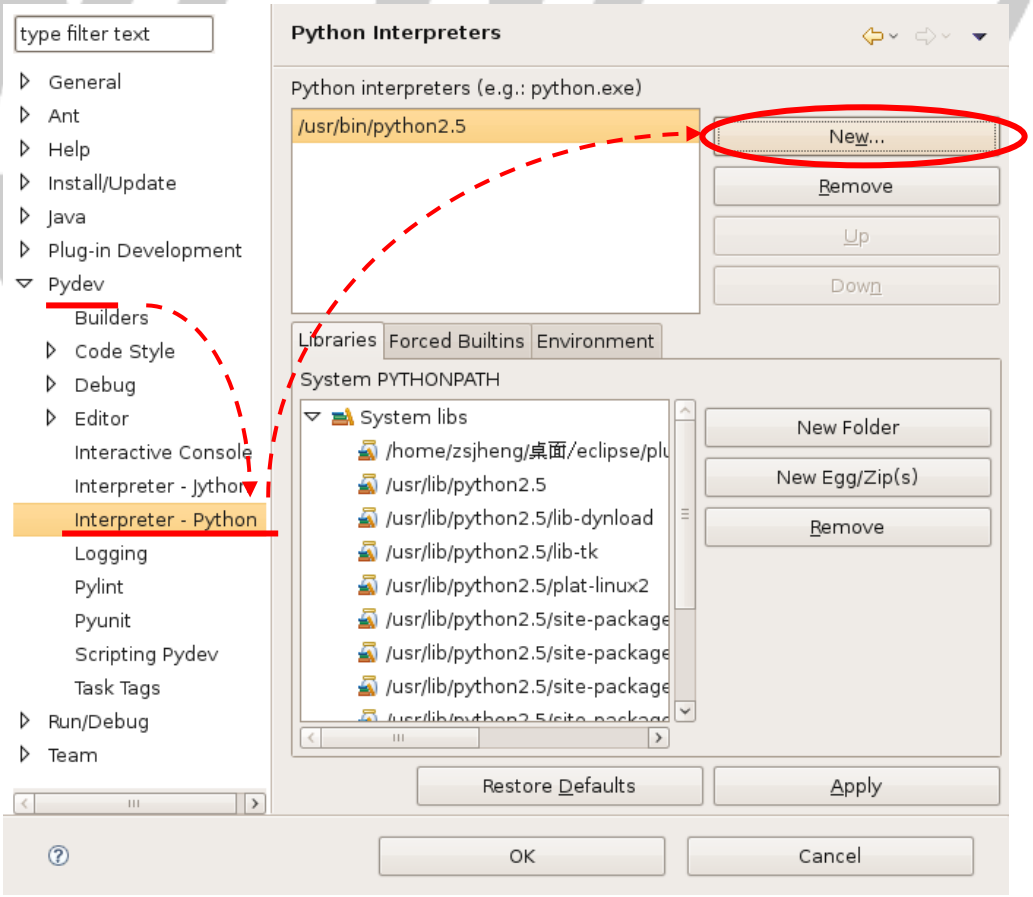
勾選完之後再按下 install 按鈕來安裝我們所選取的 Pydev 外掛程式。 之後還會有確認安裝以及使用版權宣告(同意 EPL, Eclipse Public License)兩個頁面，都同意以後就可以進行安裝。 安裝完成後則需要重新啓動 Eclipse 來完成 Pydev 的安

裝，在正常無錯誤訊息發生的情況下則是成功地安裝了 Pydev 的外掛程式。

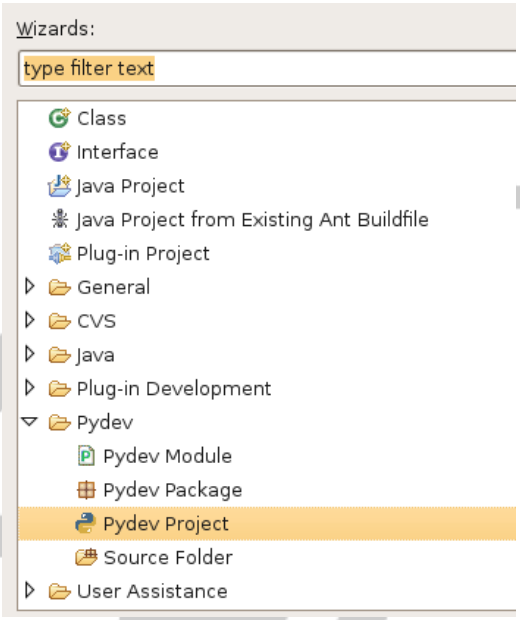
在安裝完 Pydev 之後，還要做一些設定才能讓 Eclipse 來驅使 Pydev 執行我們所撰寫的 Python 程式。 接著回到 Eclipse 主程式畫面，選取功能表上的 **Window** 選項中的 **Preferences** 來設定 Python 的執行路徑。

首先打開 **Pydev** 選項選取 **Interpreter – Python** 的子選項。 右方則會出現設定 **Python interpreter** 的選項，按下 **New…** 按鈕輸入 Python 的系統路徑*。 設定完之後 Eclipse 會自動幫使用者設定 Python 的一些執行程式以及函式庫的路徑，基本上除非使用者自己本身有特別的需求，否則只要套用 Eclipse 預設的設定內容即可。 設定結束後按 **ok** 完成即可。 正確地完成上面的步驟沒有錯誤訊息表示已經成功地將 Pydev 外掛程式加到 Eclipse 中了。

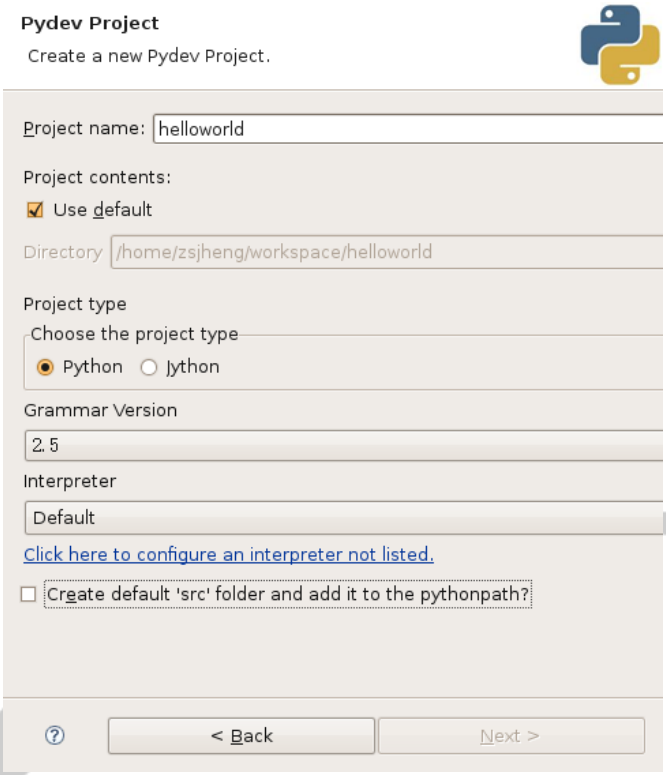
*在 ubuntu Linux 平台上我們選取 Python 安裝的路徑 /usr/bin/Python25 (使用者在 /usr/bin 底下會看到 Python 及 Python25 兩個目錄，我們擇選取 Python25 這個目錄下的 interpreter 作爲預設的直譯器)；而在 Windows 平台，預設的路徑則是 Python 安裝路徑的根目錄 (在此爲 C:\Python26\ 資料夾)，並選取 pythonw.exe ； MAC 平台上的目錄也是選取 pythonw 所在的位置即可 (預設應爲 /System/Library/Frameworks/Python.framework/Versions/2.5/…..)。



接下來就準備來嘗試開發一個自己的 Google App Engine 專案，這還需要引入 Google App Engine 的 SDK 路徑設定。 第一次執行時，要先選取功能表中 **File > New > Other...** 來設定 project 精靈。 接著選取 **Pydev** 中的 **Pydev Project** 選項後按 **Next** 繼續下一步。

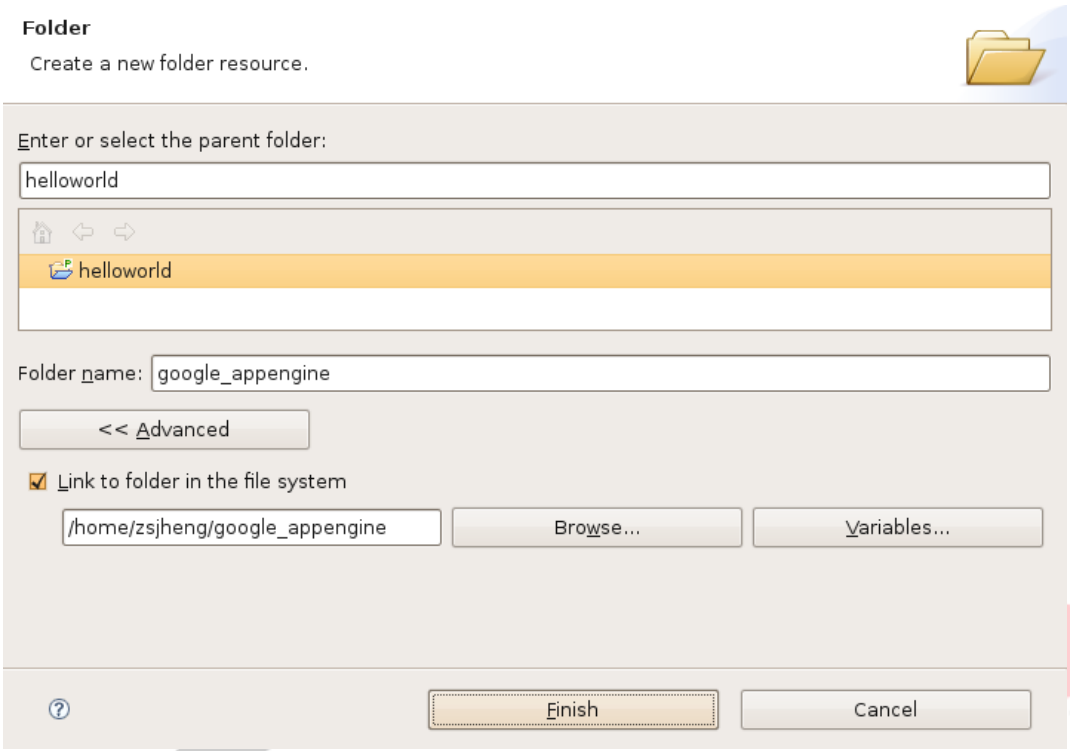


接著則是設定新增的 Pydev 專案的相關屬性如： **Project Name** (所要開發之專案的名稱，在此我們以 hellowolrd 程式爲例)、**Project contents** (設定專案資料夾所在的路徑，在此我們則勾選預設值，即是使用者開啓 Eclipse 時所指定的 workspace 底下的專案同名目錄) 、**Project Type** (我們開發的程式爲 Python 專案，因此選取 Python 選項)以及所使用的 Python 版本 (在此爲 2.5)。

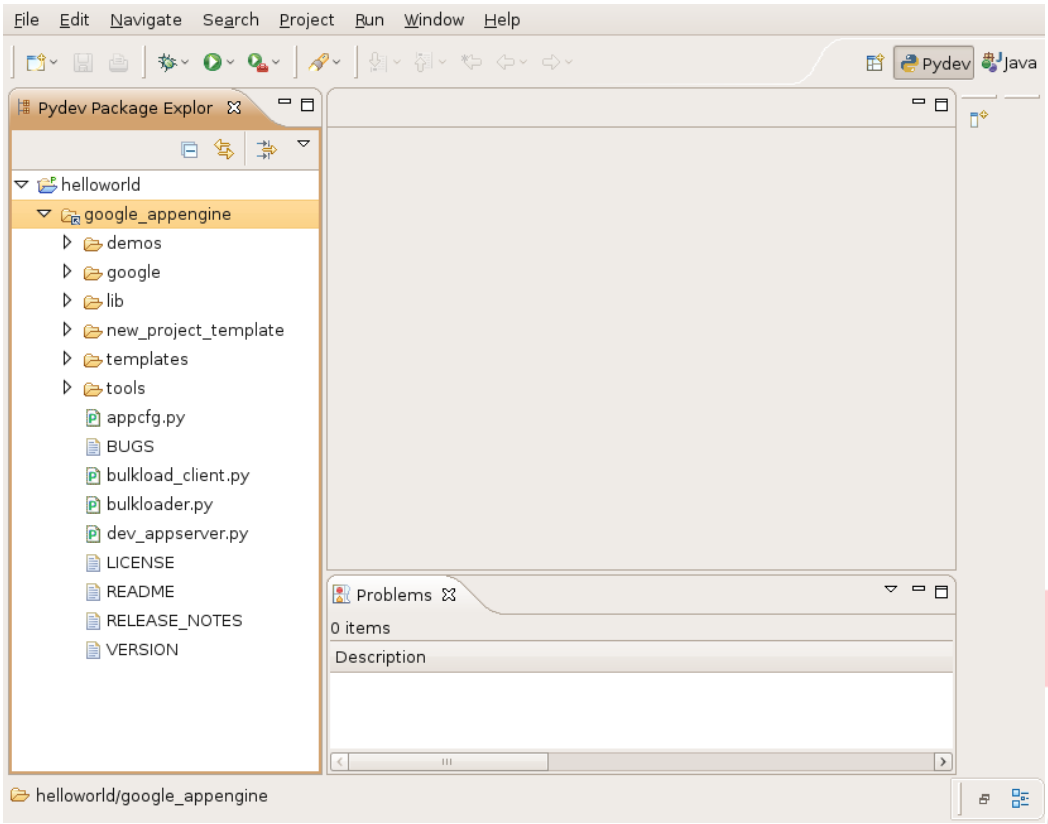


另外還有一個小選項是選擇性的，如果勾選的話 Eclipse 會在專案根目錄下新增一個 src 的目錄成爲開發程式原始碼的根目錄；然而我們在此則不勾選這個選項，使用者可以依照自己的需求與習慣決定是否要勾選。 按下 **Finish** 後則可完成新增專案的設定。 完成後就會進入到 Ecplise 中開啓的一個 PyDev 專案的工作畫面，這時候已經可以開始在左邊 Pydev Package Explorer 欄位的 helloworld 專案加入 Python 程式了。

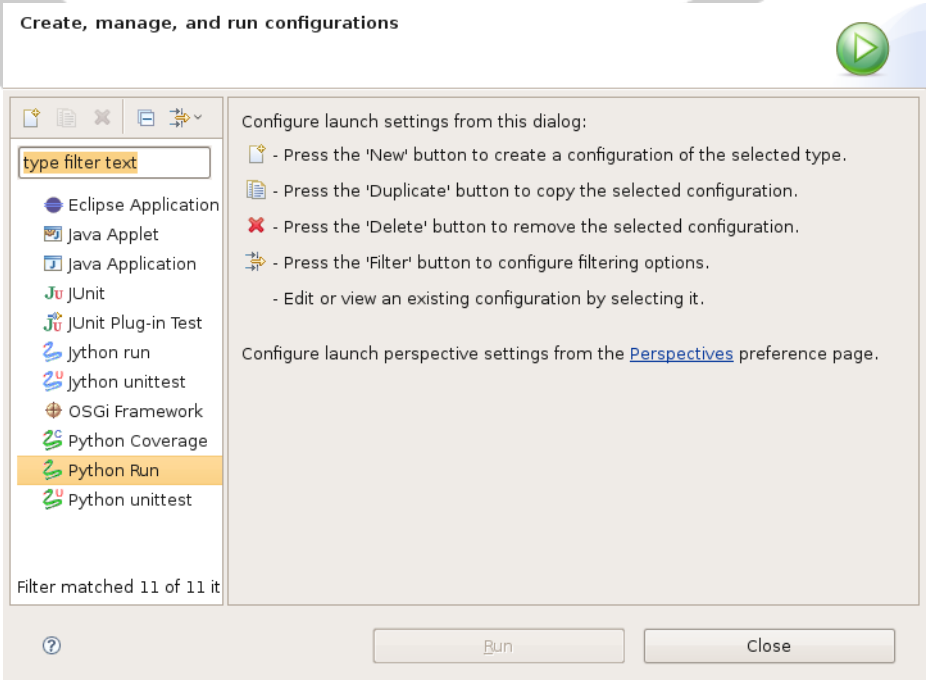
然而，一個完整的 Google App Engine 專案還需要引入 Google App Engine 的 SDK 才可呼叫、使用 Google App Engine 所提供的支援。 接著選取功能表上 **File > New > Folder** 來新增資料夾。 在 **Folder name** 中填入所要新增的 Google App Engine 資料夾的名稱，在此我們設定爲 **google_appengine**，不過我們要將這個目錄作爲一個捷徑並指向我們最初在系統上安裝 Google App Engine SDK 的路徑(由於在前一節在 Linux 上安裝 SDK 時指出可安裝在任何路徑下，因此讀者只要指定到正確的 SDK 安裝路徑即可，在此爲**~/google_appengine**)，我們可以從下方的 **Advanced** 選項中來設定路徑的指向。



設定結束後即可按 **Finish** 完成設定。 接著在工作頁面左邊的 Pydev 專案瀏覽區塊即可看到我們所設定的 helloworld 專案以及新增的 google_appengine 目錄(注意：在此這個目錄的圖示有捷徑的符號表示鏈結到 Google App Engine SDK 實體目錄)。

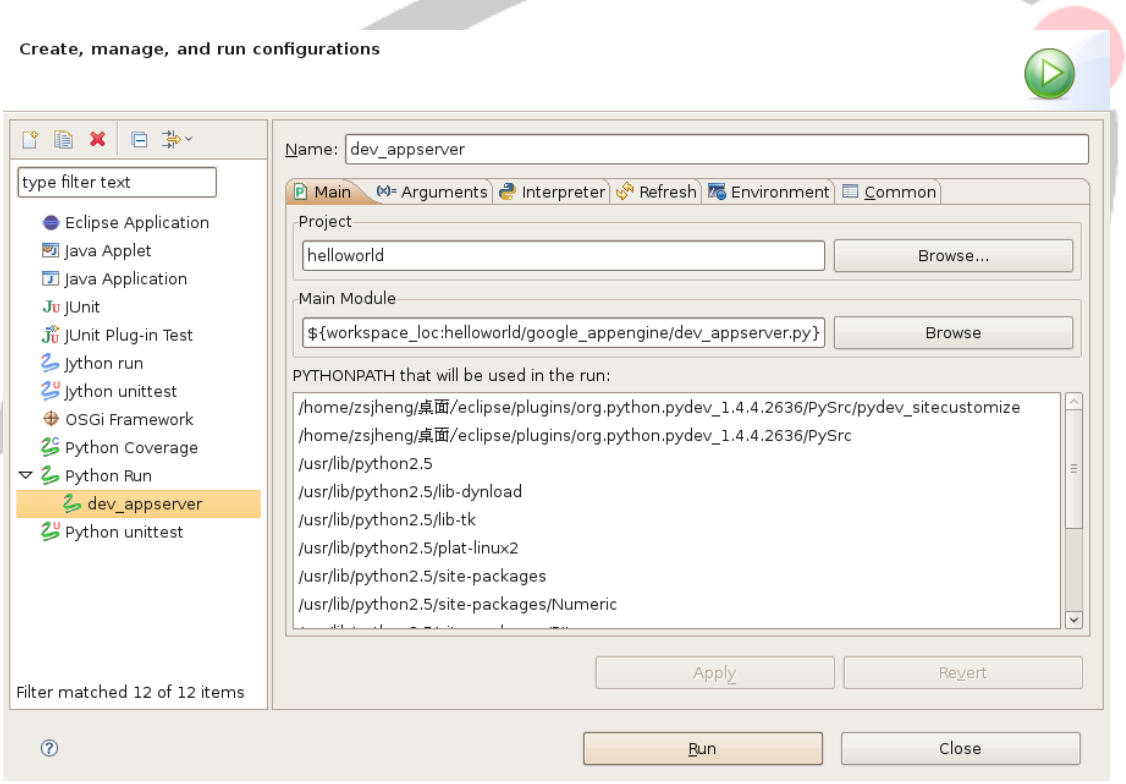


到了這個步驟已經完成了：1. 將 Pydev 外掛程式安裝到 Eclipse 中、2. 將 Google App Engine SDK 目錄鏈結掛載到專案目錄中。 我們最終希望達成的目標是當我們完成專案的撰寫，只要像在其他 IDE 上撰寫完程式後按個按鈕後就可以來編譯執行程式，在此我們需要做最後一點設定來完成這個功能。



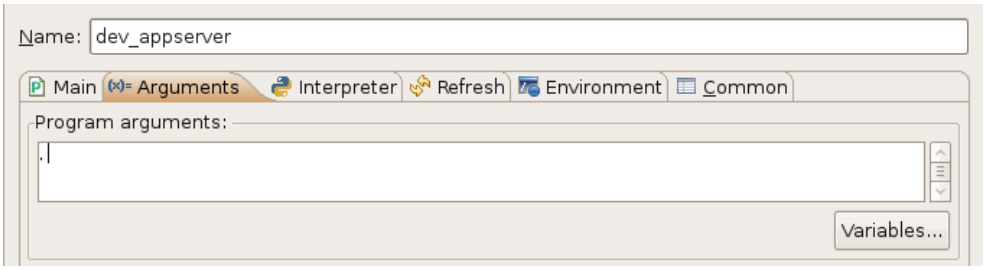
接著我們點選左邊 Pydev Package Explorer 欄位的 hellowold 專案按右鍵，選取 **Run As** 底下的 **Run Configurations...** 的選項來開啓 Run Configurations 的視窗。 在 Run Configurations 視窗下的有多個預設的專案型態，我們則選取 **Python Run** 這個選項按右鍵，選取 **New** 選項來新增新的 Python 專案執行的設定。

新增一個 Python 專案執行設定之後，右邊視窗會出現許多的設定選項。 首先將 **Name** 改成 **dev_appserver** (這並非預設值，使用者可以依照自己的喜好來修改名稱)；誠如前面章節介紹，Google App Engine SDK 下的 dev_appserver.py 這個程式可以讓我們在本機端測試我們所撰寫的專案，我們現在加入這個選項的目的即是可以透過 Eclipse 的環境直接呼叫 dev_appserver.py 來幫忙我們開啓本機端的測試環境，接下來有幾個重要的設定必須完成才能正確地呼叫 dev_appserver.py 執行程式。

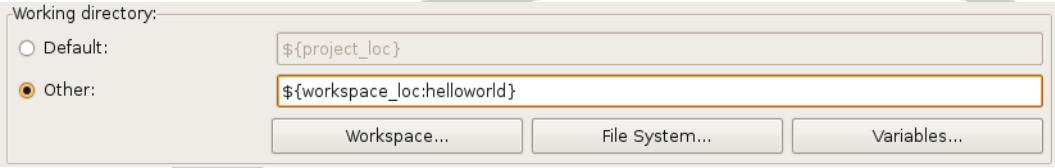


首先在 **Main** 的這個分頁，使用者必須要先指定 **Project** 以及 **Main Module** 這兩個項目。 **Project** 選項即是指定目前所要編輯的專案名稱，使用者可以利用 browse 按鈕來選取自己的專案；而 **Main Module** 則是指定當專案執行 Run 的指令時所觸發的程式，在此當然就是 dev_appserver.py 這個程式，使用者一樣也能利用 browse 按鈕來選取。 接著在切換到 **Arguments** 這個頁面，在 **Program arguments** 這個輸入格輸入 **.** 參數(即是一個點，代表的意義則是當下目錄的意思)。 設定到了這邊已經完成了，加入這個參數後表示，當使用者的專案執行 Run 指令時，Eclipse 會自動送出 **dev_appserver.py .** 的指令，最後只要按右下方的 Apply 鍵即可完成

設定。

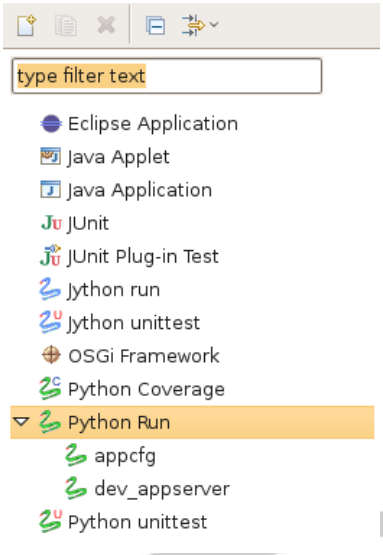


還有一個地方就是底下的 **working directory** 的部分，選擇 **Other**，並在 **Workspace** 下選擇我們所設定的 helloworld 目錄。



完成之後，該如何在 Eclipse 下利用 Google App Engine SDK 所提供的本機端測試程式 dev_appserver.py 來測試我們所撰寫好的專案呢？ 只要在左邊 Pydev Package Explorer 欄位中的 hellowold 專案按右鍵，選取 **Run As** 底下的 **Run Configurations...** 的選項來開啓 **Run Configurations** 的視窗，在 **Run Configurations** 視窗下的 **Python Run** 這個選項底下，雙擊 **appserver** 這個剛剛設定完成的選項來在本機上測試自己的專案。

除了這個在本機端上測試的程式之外，Google App Engine SDK 還有另外一個更重要的程式：appcfg.py，即是將專案內容上傳到 Google App Engine 的 appspot.com 平台上公布給其他使用者來使用。 步驟也跟上面一樣：1. 在 **Run Configurations** 視窗下新增一個選項命名爲 **appcfg** (或者其他使用者想要的命名)、2. 在 **Main** 頁面中指定 **Project** 名稱及 **Main Module** 內容(即 Google App Engine SDK 中的 **appcfg.py** 程式)、3. 切換到 **Arguments** 頁面，在 **Program arguments** 輸入格中輸入 **update .** 參數(update 後一個空格再加一個點)、4.修改 **Working directory** 爲專案內容所在的目錄(在此即爲 helloworld)、5. 點選 **Apply** 完成。



大致上的目的和設定 **dev_appserver.py** 相同，而當使用者完成專案的撰寫時，只要重複和呼叫 dev_appserver.py 程式相同的動作 (在左邊 Pydev Package Explorer 欄位中的 hellowold 專案按右鍵，選取 **Run As** 底下的 **Run Configurations…** 的選項來開啓 **Run Configurations** 的視窗，在 **Run Configurations** 視窗下的 **Python Run** 這個選項底下，雙擊 **appcfg** 這個剛剛設定完成的選項來在本機上測試自己的專案) 則可命令 Eclipse 執行了 **appcfg.py update .** 指令，即是將已撰寫完成的 Google App Engine 專案上傳到 Google 的平台上。 完成這些設定之後，後面的章節我們則將實際撰寫一個簡單的 helloworld 專案，在本機端以及 Google App Engine 平台上執行；除此之外，還會在終端機模式下實際去下指令來了解 Eclipse 實際上幫忙我們做了哪些工作。

Hello World!

安裝完 Python 以及 Google App Engine 的執行環境之後，我們先來寫一個簡單的 Hello World Web 應用程式，在本機上執行體驗一下 Google App Engine 的執行結果，並了解如何編譯、執行 Google App Engine 的程式碼。 基本上，在安裝完 Google App Engine SDK 後，在其安裝目錄內會有一個 demo 的資料夾，底下還有一個 guestbook 資料夾，裡面則是一個簡單且基本的範例程式。 不過在這邊還是先介紹更簡單的 Hello World 程式來當作第一個體驗 Google App Engine 的程式。

於 Eclipse IDE 執行程式

前一節我們已經設定好 Eclipse 及 PyDev，並開了一個 helloworld 專案，我們現在就承續前章節內容來利用 Eclipse 環境來開發 Hello World 的程式。

首先，點選左邊 Pydev Package Explorer 欄位的 hellowold 專案按右鍵，選取 **New** 選項下的 **File** 來新增一個檔案命名爲 hello.py。 按 **Finish** 完成以後則會回到編輯工作畫面，這時 Eclipse 中間的文字編輯區就會出現一個 hello.py 的空白檔案，在這個檔案中加入以下的兩行程式碼內容：

```
print 'Content-Type: text/plain'
print 'Hello, world!'
```

這個 hello.py 範例程式節錄自 Google App Engine 官方文件。 基本上只要 print 'Hello, world!' 這段程式碼即可正常執行，第一行的 print 'Content-Type: text/plain' 用意僅在 metadata 的宣告，不加入亦可正常執行。

接下來我們以同樣方式再新增一個檔案命名爲 app.yaml，這是一個設定檔且檔案名稱是固定的，請不要隨意更換名稱避免造成執行時的錯誤。 當我們下指令編譯 hello.py 或者任何一個我們要開發的 Google App Engine Web 應用程式時，Google App Engine 都會先讀取這個檔案內容來取得相關的資訊：應用程式名稱、應用程式版本編號、執行環境、API 版本編號以及 handler 程式檔案目錄等等。 接下來我們先在 app.yaml 這個檔加入以下的內容。

```
application: helloworld
version: 1
runtime: python
api_version: 1

handlers:
- url: /.*
  script: hello.py
```

首先 **application** 就是指我們所開發的 Google App Engine Web 應用程式名稱，這裡先暫定爲 helloworld；因爲 application 名稱必須和上載到 Google appspot 網站上的 Web 應用程式帳號名稱一致才行，否則會發生錯誤，不過目前我們僅先在本機測試，因此將會把這個問題留到後面章節來討論。 **version** 則是指這個應用程式的版本；**runtime** 則是指我們用來開發這個 Google App Engine Web 應用程式的所使用的執行環境，預設當然就是 Google 目前唯一使用的 Python 語言，如果將來 Google App Engine 支援更多的程式語言來開發應用程式的話則再更改爲相對的程式語言名稱；**api_version** 則是 Google App Engine 所提供的 API 版本，目前版本爲 1.x，我們只要填入 1 即可。 最後就是 **handlers** 這個參數，這是在指定當 Google

App Engine 接收到來自使用者的 request 時，該由哪一個 script 程式來回復；這部分我們會在第三章有更詳細的說明，在這邊讀者只要知道 Google App Engine 會執行 hello.py 這個 script 檔來處理來自使用者端的 request。

以上兩個程式存檔後，就要來在本機上作測試了。

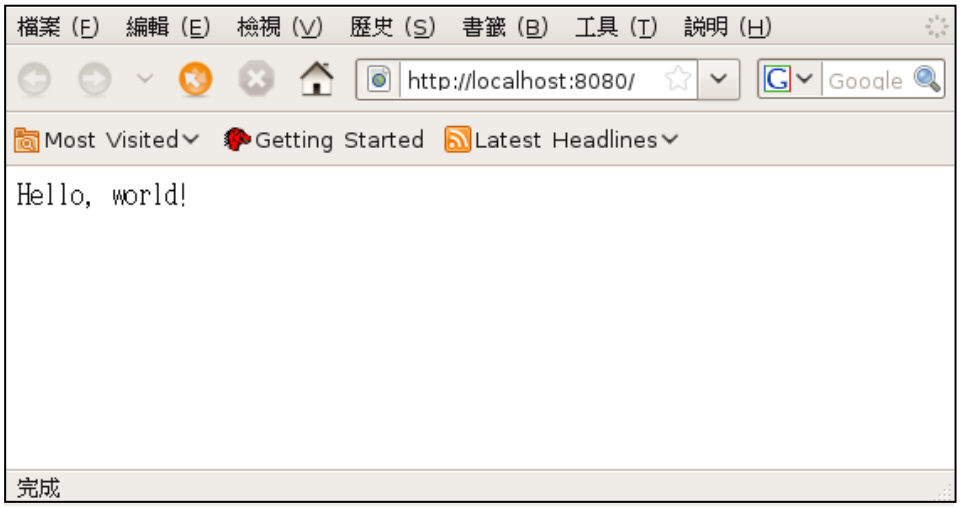
就如同上一節所說明的方式，在左邊 Pydev Package Explorer 欄位中的 hellowold 專案按右鍵，選取 **Run As** 底下的 **Run Configurations...** 的選項來開啓 **Run Configurations** 的視窗，在 **Run Configurations** 視窗下的 **Python Run** 這個選項底下，雙擊 **dev_appserver** 選項或者選取 **dev_appserver** 選項後按右下角 **Run** 的按鈕皆可。 接著在編輯視窗下方的執行結果視窗會出現以下的訊息：

```
INFO    2009-03-12 06:42:02,012 appengine_rpc.py] Server: appengine.google.com
Allow dev_appserver to check for updates on startup? (Y/n): y
dev_appserver will check for updates on startup.  To change this setting, edit
/home/zsjheng/.appcfg_nag

INFO    2009-03-12 06:42:14,771 appcfg.py] Checking for updates to the SDK.
INFO    2009-03-12 06:42:16,654 appcfg.py] The SDK is up to date.
-------------------------------------------------------------- 檢查 SDK 更新
WARNING  2009-03-12 06:42:20,187 datastore_file_stub.py] Could not read datastore data from
/tmp/dev_appserver.datastore
WARNING  2009-03-12 06:42:20,187 datastore_file_stub.py] Could not read datastore data from
/tmp/dev_appserver.datastore.history                          檢查 datastore 資料
-----------------------------------------------------------

INFO    2009-03-12 06:42:20,415 dev_appserver_main.py] Running application helloworld on port
8080: http://localhost:8080                                   成功於本機端執行
-------------------------------------------------------------
```

這個時候表示已經成功地在本機端的 8080 port 啓動 helloworld 的專案測試。 而以上的執行訊息內容意義則是一開始檢查 SDK 的版本是否符合最新版，如果這時候本機並沒有連上網路的話則會出現連現錯誤的訊息，但仍能使用現有版本的 dev_appserver.py 來作本機端的測試。 接著會檢查 datastore 的資料，這個部分我們將會在第三章在作比較詳細的介紹。 直到了最後出現 **Running application helloworld on port 8080: http://localhost:8080/** 表示已經正常在本機上執行成功了。這時候打開瀏覽器，網址列輸入 http://localhost:8080 即可看到執行結果：



恭喜！ 我們已經正確地利用 Eclipse 在本機端開發並測試第一個 Google App Engine Web 專案了(在 Windows 與 MAC OS X 平台的測試方式亦同)，接下來下一小節則會來介紹如何透過終端機模式來自行下指令在本機端作測試。

非 Eclipse 環境執行程式

首先在 Google App Engine 的安裝目錄(Windows 平台預設為 C:\Program Files\Google\google_appengine；Linux 平台則為使用者自行解縮後的目徑；Mac OS X 平台則毋需擔心此問題，可用 GoogleAppEngineLauncher 直接測試)下先新增和上一小節相同的兩個檔案：hello.py 以及 app.yaml。 接著打開終端機介面**，切換工作目錄到 Google App Engine 目錄下，直接執行 dev_appserver.py 指令，而後面接的參數為所要測試的專案所在路徑，由於目前的 helloworld 範例和 dev_appserver.py 程式所在目錄相同，因此只要設定為代表目前目錄的 . 即可，這和上一小節設定 Eclipse 時所輸入的 . 意義是一樣的。 而在 Windows 平台及 UNIX-like 的 Linux 平台下的指令有些許的不同，如下：

- Windows 平台

```
C:\Program Files\Google\google_appengine>  dev_appserver.py .
```

- Linux 平台

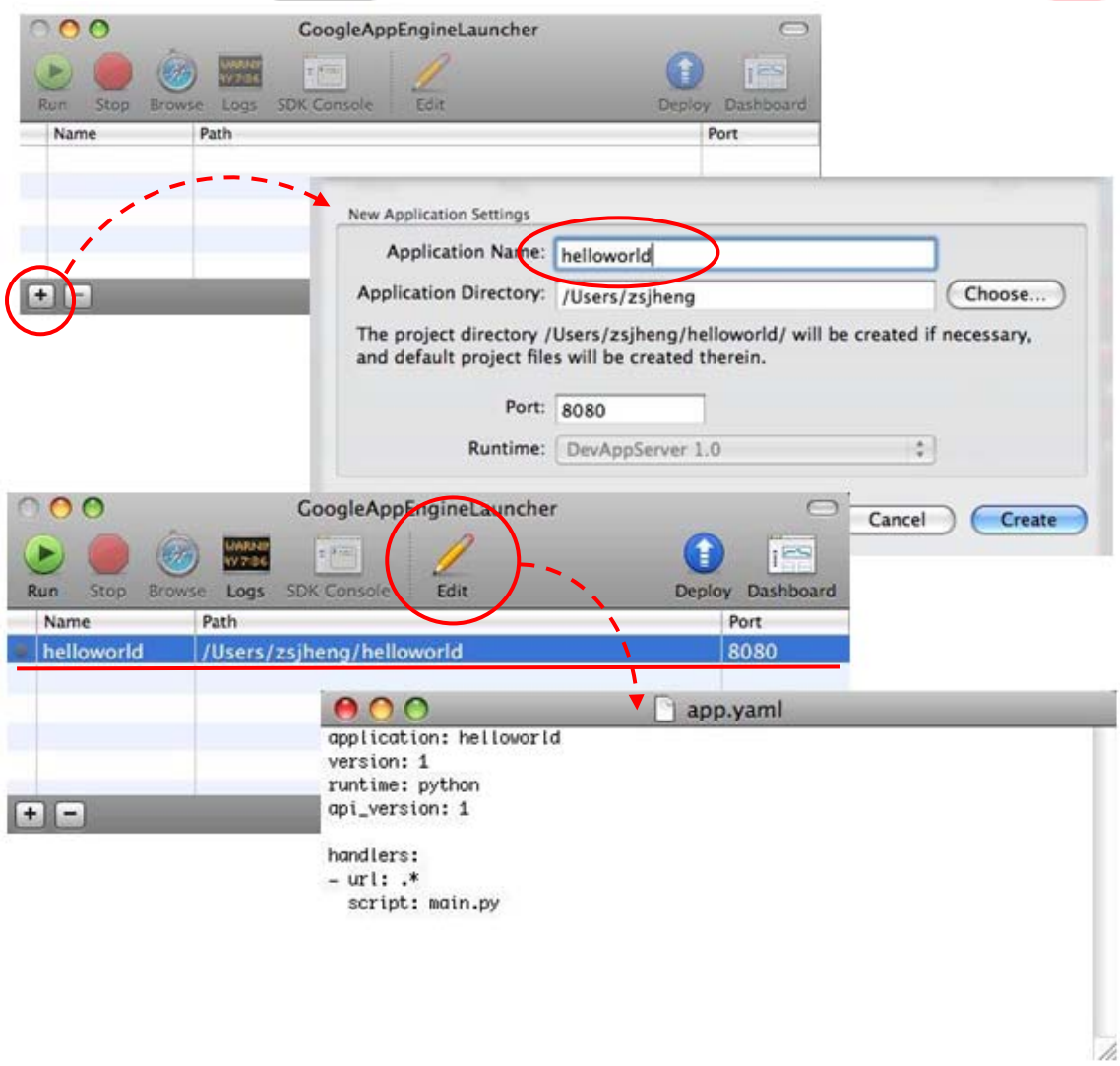
```
user@localhost:~/google_appengine$ ./dev_appserver.py .
```

這是作業系統本身的差異，等到出現了上一小節執行的顯示結果(**Running application helloworld on port 8080: http://localhost:8080**)，表示已經成功執行了，這時即可打開瀏覽器輸入 http://localhost:8080/來檢視執行的結果。

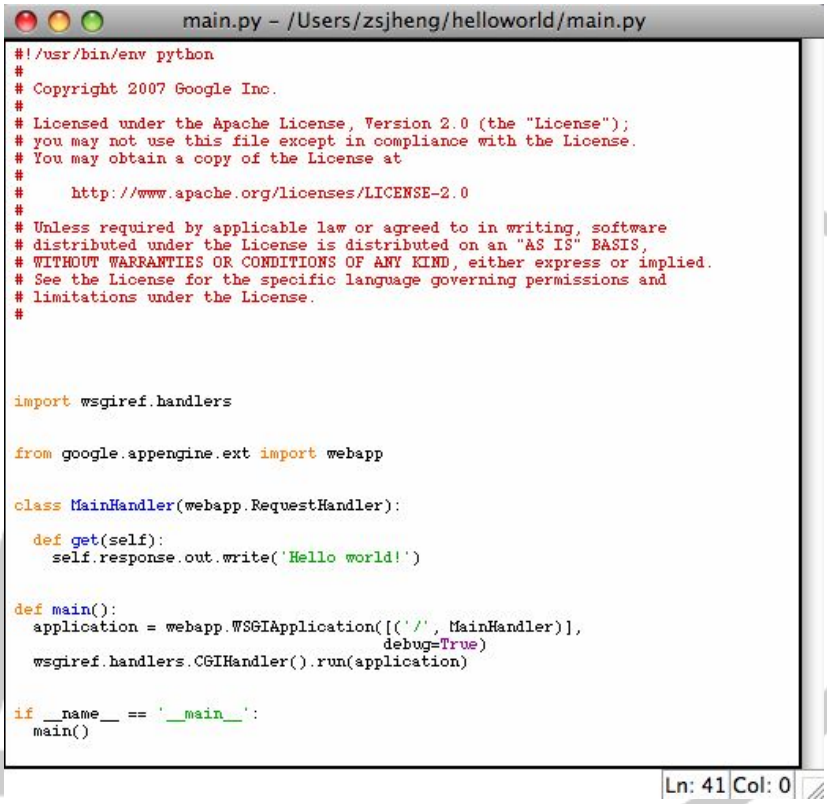
**Windows 平台請到 開始工作列 > 執行 > 下 cmd 指令 即可；Linux(在此以 Ubuntu 8.10 爲例)的終端模式可以在 X-Window 下在應用程式下 > 附屬應用程式 > 終端機選項啓動或者 ctrl + alt + F1~F6 直接進入終端機模式；MAC OS X 也有終端機模式，在應用程式 > 工具程式 > 終端機，但因 Google App Engine 在 MAC OS X 平台提供了非常方便的工具，因此不建議在 MAC OS X 平台利用終端機模式來開發。

- Mac OS X 平台

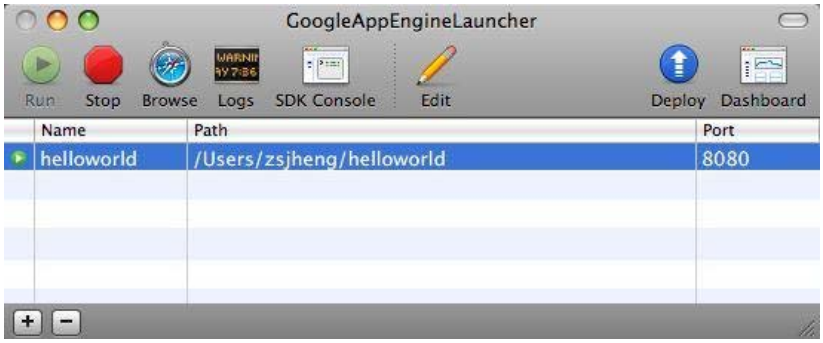
在 MAC OS X 平台上使用 GoogleAppEngineLauncher 來在本機測試程式是非常方便的。首先開啓先前下載解壓縮完並拖曳到 MAC 主機上的 GoogleAppEngineLauncher 程式，便能看到如下圖中簡單的檔案管理介面。 接著就是點選左下的加號來新增一個 Google App Engine 專案(當然反之可用減號來刪除不要的專案內容)，只要填入專案名稱、設定專案所在目錄、指定測試時本機端使用 port 號就能輕鬆新增一個專案了。



新增完畢後可以點選工具列上的 **Edit** 功能項即可開啓專案中預設的 app.yaml 檔，而預設還會產生一個 main.py 檔，就在稍早設定的專案目錄當中。 接著可以到專案目錄下找到主要的 python 程式(在此即爲 main.py 檔)，點選之後以安裝 Python 執行環境時就預設安裝的簡易編輯器 IDLE (Python GUI)來開啓，內容則是一段 Python 程式，且所達到的目的就跟上一節我們所撰寫的程式碼內容一樣－在網頁上秀出 Hello World 的字樣。



接著回到 GoogleAppEngineLauncher 主程式點選 Run 的選項。 直到 **Run** 按鍵不能被點選(disable)、**Stop** 按鍵被啓用(enable)且專案名稱左邊出現綠色的 play 圖示則表示專案已經成功地在本機端執行了。



接著只要再點選 Browse 按鍵即可自動開啓瀏覽器並連線到 Google App Engine 本機

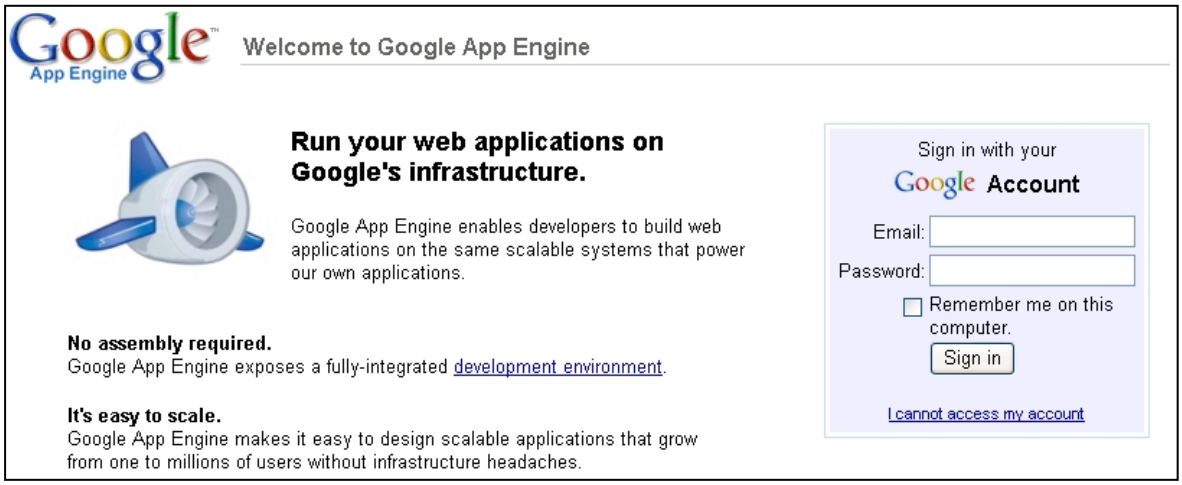
端執行的測試程式 (即 http://localhost:8080)，就如下圖開啓 safari 瀏覽器，出現”Hello World”字樣，即表示已經成功地在 MAC OS X 平台上作本機端測試成功了。

其他的功能如 **Logs** 按鍵可以檢視在終端機模式下指令時會出現的執行結果；**SDk**…… ；**Deploy** 按鍵能將撰寫完成的專案傳送上 Google App Engine 平台，這我們會在後面的小節再詳述；**Dashboard** 按鍵則會直接連線到 Google App Engine 的 DashBoard 頁面更使用者來管理自己的專案，這部份則請參考 DashBoard 效能檢視器小節來了解更多的內容。

以上讀者已經可以在本機端測試自己的 Google App Engine 專案了。 可是如果撰寫了一個非常實用的 Web 應用程式只能在自己本機端來執行就非常沒有意思了，一定要放到網路上去公開給三五好友或者廣大網眾來使用才有意思，下個章節我們將說明如何申請一個自己的 Google App Engine 專案，接著再來介紹如何將專案部署到 Google App Engine 平台來供其他使用者使用。

新增 Google App Engine 專案

在前一節我們已經試過在本機上測試並執行最簡單的 Hello World Google App Engine 程式，接下來就準備開始將我們所寫好且在本機測試無誤的 Web 應用程式上傳到 Google 的平台上提供服務。 Google App Engine 開發人員當然必須要有 Google 帳號才能新增自己的 Google App Engine 專案，而申請 Google 帳號也是完全免費的！ 接下來就先打開瀏覽器連到 http://appengine.google.com 來登入 Google App Engine 首頁來新增自己的 Web 應用程式專案。



Google App Engine 登入頁面。

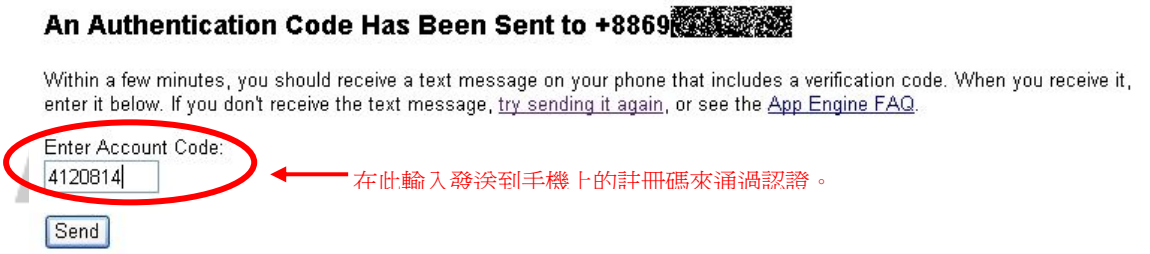
登入之後，若是第一次使用的開發人員會看到如下圖的歡迎頁面。　在此就先點選 **Create an Application** 來新增自己的 Web 應用程式專案。

第一次登入 Google App Engine 之歡迎頁面。

接下來則是透過手機來認證。　在 **Country and Carrier** 的選項是要選取使用者的所在國家以及提供手機服務的公司，不過目前僅只有美國、加拿大及日本地區的手機服務提供廠商有列出，因此這個選項就使用預設的 **Other (Not Listed)**的選項，接下來則是要填入自己的手機號碼。　在此欄位所填寫的是完整的國際電話碼，台灣的國碼爲 886，若使用者的手機號碼爲 0912345678 ，則須填入 +886912345678 ，填寫完畢後只按下 Send 送出即可，接下來大約過數分鐘的時間即可在自己的手機上收到認證碼。

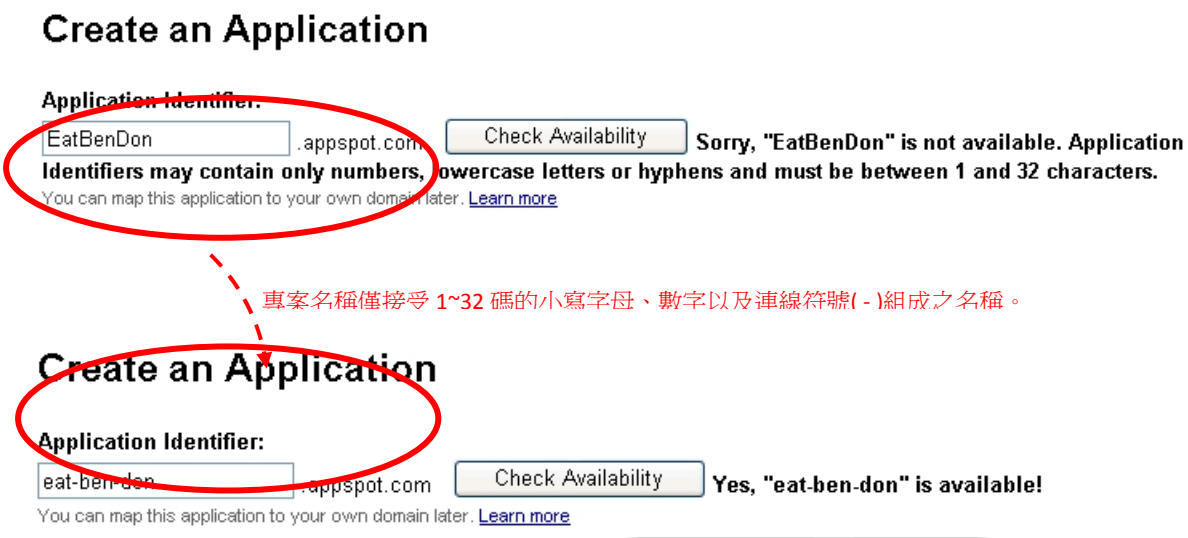
手機認證畫面(1/2)。

接下來的表單則是需要使用者填入手機所收到的認證碼來作手機認證，填寫完之後送出即可。



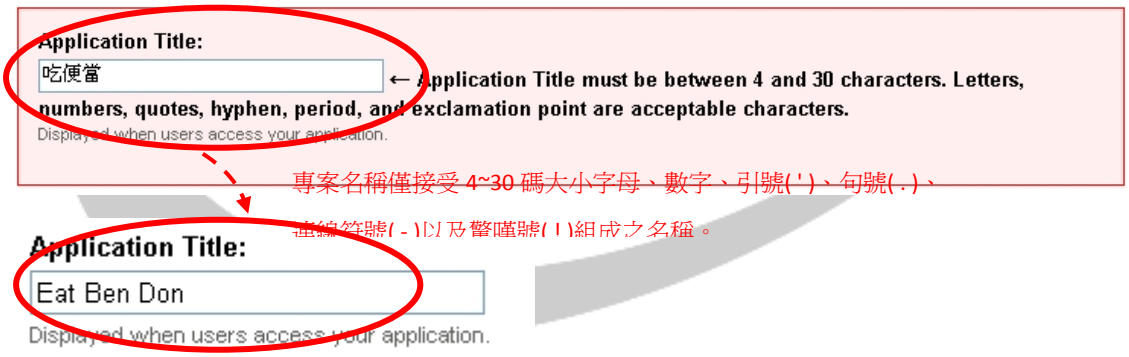
手機認證畫面(2/2)

通過認證之後即出現 Create Application 的頁面可新增自己的 Google App Engine Web 應用程式。 首先是先填入 Application Identifier ，這個 Application Identifier 結合 .appspot.com 則成爲一個完整的、可供存取的 Web 應用程式 URL；appspot.com 這個網域名稱則是 Google App Engine 所註冊使用的 (使用者應該對於同由 Google 所支援的部落格空間 blogspot 空間有印象吧!)。 在此應該要注意的是，這個 ID 只接受 1~32 碼的小寫字母、數字以及連線符號(-)所組成，輸入之後可以利用 Check Availability 來檢查是否已有人使用。 下圖圖一筆者輸入包含大小的 ID 則出現了錯誤的情形，下圖圖二所輸入的 ID 則符合 Google App Engine 的要求。



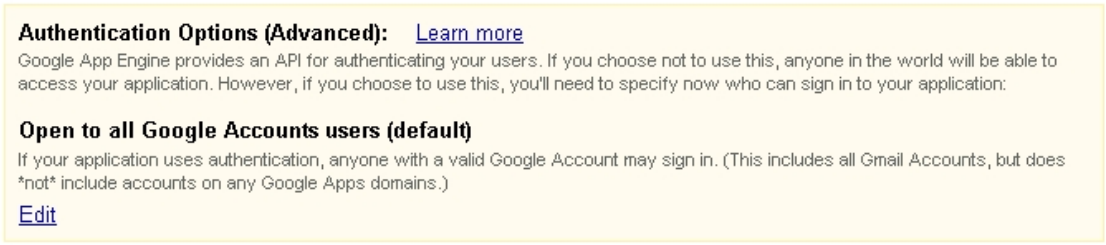
自訂專案名稱(Application Identifier)。

接著是給這個 Web 應用程式一個名稱描述。 目前只支援由 4~30 碼大小字母、數字、引號(')、句號(.)、連線符號(-)以及驚嘆號(!)所組成的名稱；當筆者輸入中文名稱實則會發生下圖一的錯誤！ 爲了能夠順利新增自己的 Web 應用程式專案，還是乖乖地填入按照 Google 所限定的名成規格 (如下圖二) 。

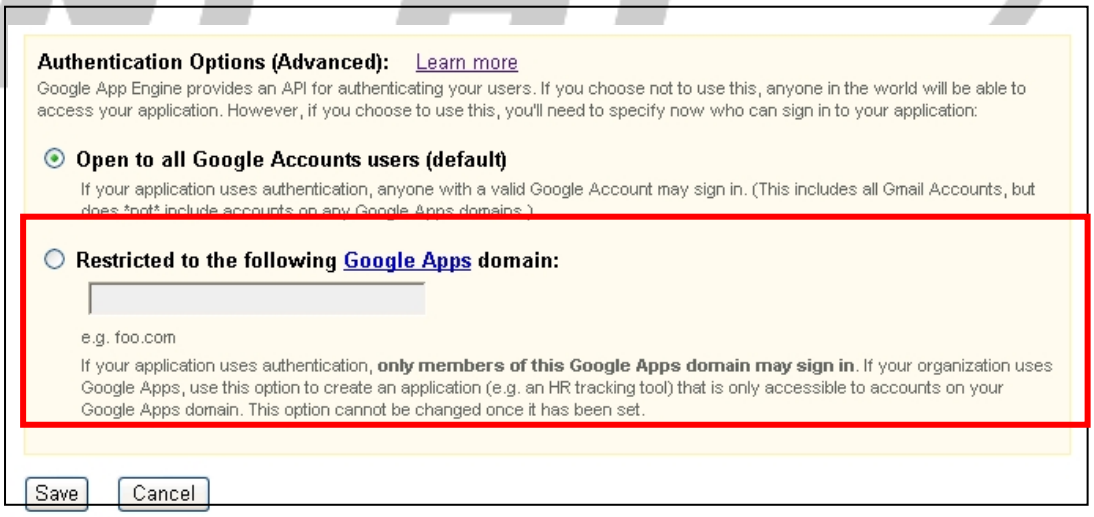


自訂專案標頭(Application Title)。

再來會有一個關於認證相關的選項設定。 Google App Engine 提供了一套 User API 供 Web 應用程式開發員來使用：當使用者要瀏覽或存取使用 Google App Engine User API 認證之 Web 應用程式時則會被導入到 Google 帳號登入頁面，使用者必須輸入自己的 Google 帳號及密碼來認證，認證過後則會回到原頁面，或者 Web 應用程式本身希望使用者所見到的頁面 (如客製化的登入歡迎頁面)。 在預設的情況下，Google App Engine 應用程式對所有具有 Google 帳號的使用者是不予設限的(**Open to all Google Accounts user**)。

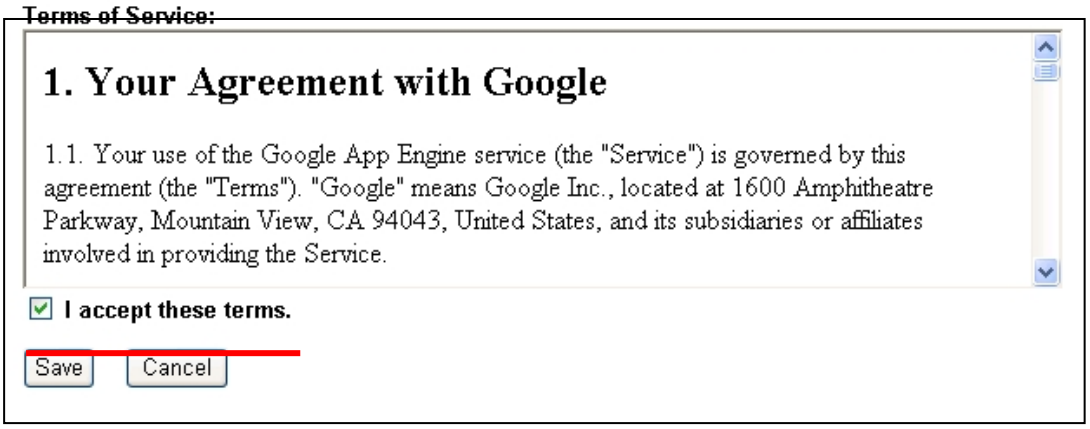


按下 **edit** 選項後會出現兩個選項：前述之 **Open to all Google Account Users (default)** 以及 **Restricted to the following Google Apps domain**。 後者所提到的 Google Apps (中文名稱爲 Google 應用服務) 與 Google App Engine 並不相同；Google 應用服務主要是針對特定的團體、公司、學校等單位內部成員提供整合性或者特定之服務 (詳細說明可到 http://www.google.com/a/help/intl/zh-TW/index.html 查詢)。 這樣認證方式的好處在於，當 Web 應用程式開發人員針對某個團體開發一套系統時 (如針對某個中小企業公司開發特定的工時系統) ，預設的情況下只有該團體的使用者才能夠登入使用，採用 Google 應用服務的 domain 名稱就能夠很輕易地將該團體的成員包含到這套系統的合法使用者名單下了。 這樣可以說是由 Google 來爲該團體及 Web 應用程式作使用者群組管理，大大地減輕 Web 應用程式開發人員的負擔。



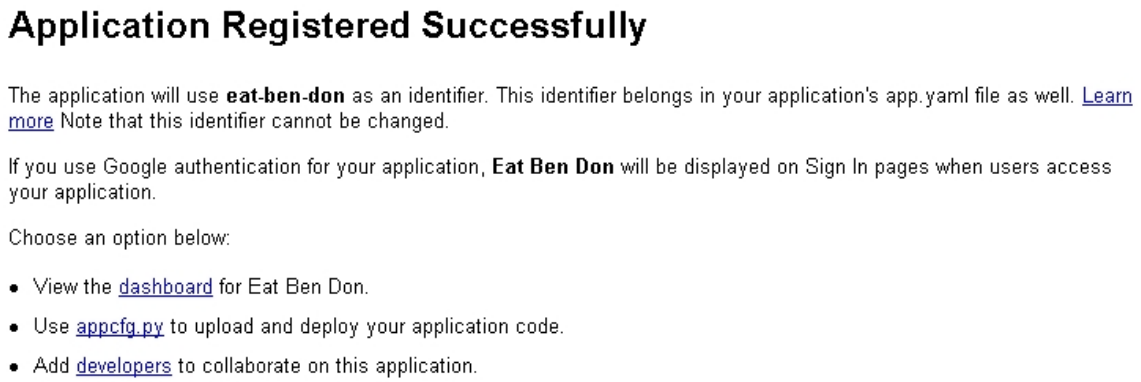
自訂專案名稱(Application Identifier)。

最後則是同意使用條款的選項，當使用者同意並勾選 Google App Engine 所條列之使用限制後，即可點選 Save 來新增第一個 Web 應用程式。



同意使用條款。

在正常無誤的情況下，按下確定後則會出現以下的新增應用程式成功畫面。 此時使用者可以點選 dashboard 連結來檢視我們剛剛建立的 Web 應用程式 eat-ben-don 的執行效能及資源使用情況 (dashboard 相關介紹請參照下一節)，或者點選 Add developers to collaborate on this application 的連結來新增此 Web 應用程式的共同管理開發員。 再繼續將介紹如何利用 appcfg.py 來上傳我們所撰寫好的 Google App Engine Web 應用程式。



使用者已成功新增一個 Google App Engine Web 應用程式。

在新增完第一個 Web 應用程式之後，使用者仍可再登入到 http://appengine.google.com 中新增其他的 Google App Engine Web 應用程式，Google App Engine 預設每個帳號可提供最多 10 個 Web 應用程式 domain 的申請 (Web 應用程式 domain 即其對應的 application-id.appspot.com 這樣的 URL)。 管理頁面中會顯示每一個 Google App Engine 目前最新的版本 (下圖中因爲剛增加新的 Web 應用程式，所以就未顯示目前的版本編號) ，Web 應用程式開發員則可點選 Web 應用程式的 ID 進入到該應用程式之 dashboard 介面作管理以及效能檢視 (dashboard 在後面

章節 **Bashboard 效能檢視器** 部分會有更詳細的說明)。

使用者已經增加之 Google App Engine 專案。

Google App Engine 專案部署

新增完 Google App Engine 專案之後，就等於已經有一個 you-project.appspot.com 的虛擬主機可以開始上傳自己的專案程式內容了。 使用者能夠利用 Google App Engine SDK 所提供的程式來上傳專案內容，或者透過設定過的 Eclipse 圖形介面來上傳，當然在 MAC OS X 還有多一個選擇就是 GoogleAppEngineLauncher，下面都會一一介紹說明。

- 使用 Eclipse 來部署 Google App Engine 專案

利用 Eclipse 來部署上傳 Google App Engine 專案在前面小節也有提到，與在本機測試差不多:在左邊 Pydev Package Explorer 欄位中的 hellowold 專案按右鍵，選取 **Run As** 底下的 **Run Configurations...** 的選項來開啓 **Run Configurations** 的視窗，在 **Run Configurations** 視窗下的 **Python Run** 這個選項底下，雙擊 **appcfg** 選項

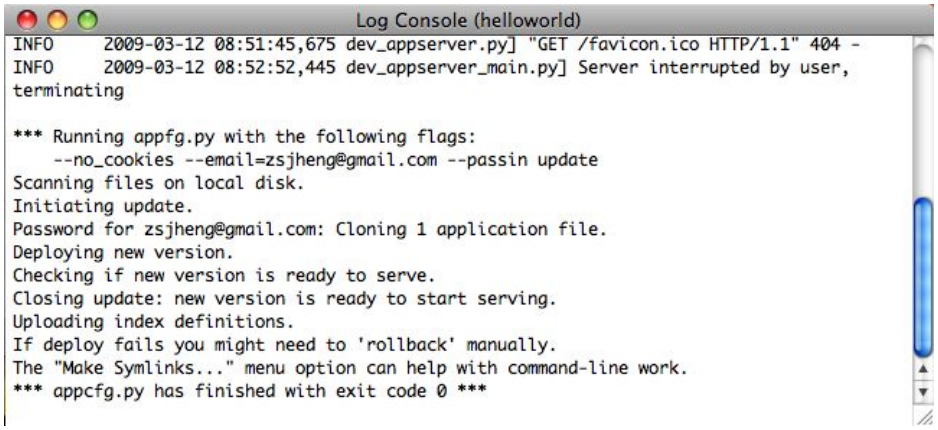
則可命令 Eclipse 執行了 **appcfg.py update .** 指令，即是將已撰寫完成的 Google App Engine 專案上傳到 Google 的平台上。 完成這些設定之後，後面的章節我們則將實際撰寫一個簡單的 helloworld 專案，在本機端以及 Google App Engine 平台上執行；除此之外，還會在終端機模式下實際去下指令來了解 Eclipse 實際上幫忙我們做了哪些工作。

- 終端機佈署 Google App Engine 專案

- Mac OS X 平台使用 GoogleAppEngineLauncher

在前面小節介紹到 GoogleAppEngineLauncher 程式的基本使用時，有提到當使用者撰寫完自己的專案之後，可以點選 **Deploy** 按鍵來上傳專案內容到 Google App Engine 平台。 點選了 Deploy 按鍵之後，會先出現一個 Google 帳號認證的視窗，即填入申請 Google App Engine 專案所使用的 Google 信箱及密碼。

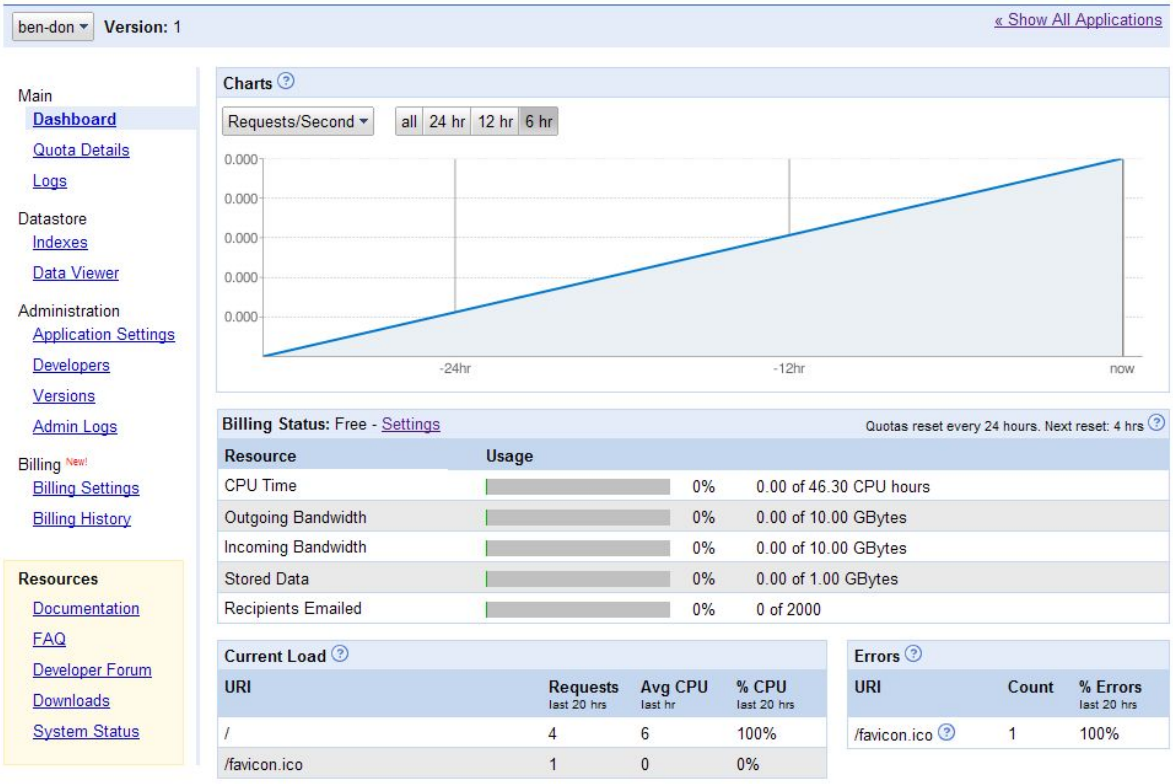
輸入完畢並點選登入之後，GoogleAppEngineLauncher 會自動將專案上傳，並出現一個終端機模式顯示出一些執行的結果，如果沒有任何錯誤的訊息則表示已經上傳成功了，使用者可以打開瀏覽器輸入 your-project-name.appspot.com 的網址來檢視，只要沒有其他錯誤訊息(常見如 404 Page Not Found 或 500 Server Error)則可以看到執行的結果－一個簡單的 Hello World。



到了這裡，已經算是把最基本的 Google App Engine 作一個介紹了，下一節則繼續將 Google App Engine 的 DashBoard 作介紹，讓使用者能夠了解自己的專案上傳到 Google App Engine 平台上執行的結果如何以及使用了多少系統資源，再來依照自己的需求作專案程式內容的調整。

DashBoard 效能檢視器

DashBoard 效能檢視器是 Google App Engine 平台免費提供給 Web 應用程式專案開發人員的一個效能查詢的利器。 一方面能夠檢視每一個 Web 應用程式專案目前所被使用的狀況，包含被 request 的次數、CPU 使用數、頻寬使用量空間使用量等等(當然這也關係到超過 Google App Engine 所限定的使用量上限收費的問題)，並提供圖表顯示讓專案開發人員能夠很清楚明瞭地了解目前自己的 Web 應用程式被使用的狀況如何。



效能檢視器主頁面