

# Web 程式設計

## 第1章 Web 程式設計簡介

### (1) 全球網

#### - 全球網(World wide web)發展歷史

- \* 網際網路(Internet)：連結全球網路的網路，全球網是網際網路的一部份
- \* 1960 年 8 月：美國麻省理工學院(MIT)教授 J.C.R. Licklider 開始發想"Galactic Network"，為網際網路的前身
- \* 1960 年代：網際網路由美國國防部先進研究計畫局(Advanced Research Project Agency, ARPA)開始發展
  - # 將大學及研究機構的主要電腦以網路連結，稱為 ARPANET，僅允許大學、研究機構使用
- \* 1980 年代：區域網路(Local area network, LAN)及個人電腦大為風行，使美國開放 Internet 的商業應用
- \* 1990 年代：研究學者 Tim Berners-Lee 在歐洲粒子物理實驗室(European Laboratory for Particle Physics, CERN)研發全球網(World wide Web)，以便在 CERN 網路中能夠輕易地存取相互參考的文件，透過超文件連結(Hypertext link, hyperlink)存取相關的文件
- \* 在全球網上的一份文件稱為網頁(Web page)，可利用統一資源定位(Unified resource locator, URL)的位址找到(也稱為網頁位址，Web address)
- \* 網站(Web site)：網頁及其相關檔案(例如圖檔)在網際網路中所存在的地方，通常屬於一個組織、企業、或個人
- \* 瀏覽器(Browser)：顯示網頁的程式，可透過輸入網頁的 URL 位址或者點選網頁中的連結(Link)而開啟其他網頁
  - # 在瀏覽器輸入 URL 或點選連結時，瀏覽器向網站伺服器提出送出網頁的要求(Request)
- \* Web server (網站伺服器)：回應瀏覽器的要求而送出網頁的電腦，所送出的網頁稱為回覆(Response)
  - # 在電腦上安裝網站伺服器軟體後，電腦就變成網站伺服器，目前最為流行的軟體是 Apache HTTP Server

#### - 全球網通訊協定

- \* URL 由 2 個部分組成：協定(通常是 HTTP)及網站伺服器的域名(Domain name)或網際網路協定位址(Internet Protocol address, IP address)，例如：<http://www.cyut.edu.tw>
  - # 超文件傳輸協定(Hypertext transfer protocol, HTTP)：傳輸網頁的通訊協定
    - 格式：http://
  - # 域名(Domain name)：能夠識別網際網路上伺服器的唯一位址，通常由 3 個部分組成：
    - 表達組織或機構名稱的文字，例如：www.cyut
    - 領域識別名(Domain identifier)：識別組織的型態，例如，.com, .edu, .org, .gov, ...
    - 國名，例如：.tw

- \* 安全超文件傳輸協定(Hypertext transfer protocol secure, HTTPS)：提供安全性高的網際網路連結，常用於銀行系統
- 網站發佈(Publish the web site)：將網頁上載至網站伺服器，使一般大眾能夠存取
  - \* 網站託管(Web hosting)：利用網際網路服務供應商(Internet service provider, ISP)的空間或電腦當作網站伺服器，較為方便
  - \* 使用自己的電腦：電腦連接網路並安裝網站伺服器軟體，需自行負責安全、維護、網路速率等工作
  - \* 雲端平台及服務(Platform as a service, PaaS)
- 域名註冊(Domain name registration)
  - \* [InterNIC](#)：負責註冊域名之組織
  - \* 應選擇較能代表組織名稱，並且不重複的域名
  - \* 亦可由 ISP 代為註冊

## (2) 網際系統開發

- 電腦程式或系統的演進：
  - \* 文字介面程式：在 UNIX、DOS、或 Linux 之文字介面環境中執行程式，指令、輸入、及輸出均為文字
  - \* 圖形介面程式(GUI programs，桌上型或筆記電腦)：在 X Windows、Microsoft Windows、或 Mac OS X 環境中執行程式，系統的操作大多以圖形元件控制(例如滑鼠、按鈕等)，可有圖形化輸出
  - \* 網際系統(Web-based systems)：因應網際網路盛行，企業或個人開始架設網頁，並開發網際系統，達到隨時及隨地均可連上系統的目標
  - \* 行動裝置程式(Apps)：因應手持裝置盛行，圖形介面程式開始佈署在行動裝置上，達到程式隨行的目標
- 網頁設計(Web page design, Web design)：網頁之視覺效果設計(非本課程之範圍)
- 網頁製作(Web page authoring, Web authoring)：網頁之標籤、屬性、及資料製作(非本課程之範圍)
- 網際系統開發(Web development, Web programming)：網站之軟體應用程式設計(本課程之範圍)
  - \* 利用 Google App Engine 之 webapp2 為開發及執行平台，Python 程式語言設計應用軟體，能存取資料庫或檔案、能與其他系統互通、並能完成更進階的工作
  - \* 網管人員(Webmaster)：負責網頁設計、網頁製作、及網際系統開發
- 主從式架構(Client/server architecture)：2 層式系統(Two-tier system)



- \* 客戶端(Client)：存取網路資源，利用使用者介面收集使用者資訊並傳送給伺服器

- \* 伺服器(Server)：提供網路資源服務，接收使用者資訊、將送給使用者的文件格式化、並負責大多數的運算

#### - 全球網系統(Web system)

- \* 建構在 2 層式的主從架構上

- \* 客戶端為瀏覽器，透過 HTTP 協定向伺服器要求網頁，並負責格式化及呈現網頁

- \* 伺服器端為網站伺服器，負責回應客戶端之要求

- \* 若伺服器端再加上資料庫或其他應用系統，則成為 3 層式(Three-tier)或多層式(Multitier)主從架構系統：客戶層、處理層、及資料儲存層

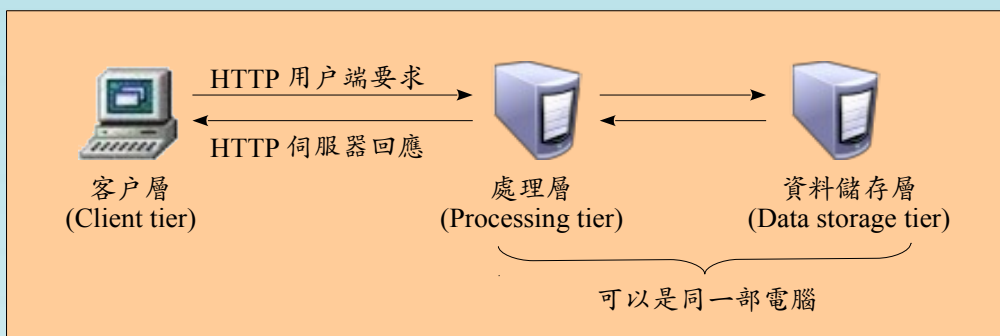
- # 客戶層(即瀏覽器)：處理使用者介面，將要求傳送至處理層

- # 處理層(或中間層)：處理瀏覽器客戶層與資料處理層之間的互動

- 客戶層向資料庫提出資料要求

- 處理層依據客戶層的要求執行相關處理或計算，從資料庫中寫入或讀出資料，然後將結果傳送給客戶層

- # 資料儲存層：於資料庫中儲存資料，並回覆處理層之要求



#### - 伺服器端命令稿設計(Server-side scripting)

- \* 伺服器端命令稿語言在伺服器上執行，主要目的在於產生互動式網站(Interactive web site)，能和資料庫溝通

- \* 伺服器端語言：PHP, JSP, ASP.net, Perl, Python, ...

#### - 客戶端命令稿設計(Client-side scripting)

- \* HTML 發展初期，網頁均為靜態(Static)，亦即瀏覽器呈現網頁後，網頁就不會再變動

- \* 之後，對於網頁內容的互動性及視覺效果的需求日增

- \* 由於 HTML 及 XHTML 僅能產生靜態文件，因此 Netscape 開發 Javascript 語言來提升網頁的視覺效果

- \* 伺服器端語言：JavaScript, Flash, ActiveX, ...

### (3) 建構網際系統開發環境

#### - 網際系統所牽涉的項目包括：

- \* 瀏覽器：Firefox, Internet Explorer (IE), Google Chrome, Opera, Safari, ...

- \* 瀏覽器內容格式顯示語言：HTML, CSS

- \* 網站伺服器：Apache, Apache Tomcat, IIS, ...

- # 伺服器軟體比較

- \* 伺服器端(後端)程式語言：PHP, JSP, ASP, Perl, Python, ...

- \* 瀏覽器端(前端)程式語言：JavaScript (jQuery), Flash, ...

- \* 資料庫：MySQL, MS SQL, Oracle, ...

\* 網際系統開發框架(Web frameworks)：

# PHP framework：CakePHP, CodeIgniter, Prado, Symfony, Yii, Zend, ...

# JSP framework：Struts 2, JSF, Spring MVC, Wicket, ...

# Python framework：Django, Grok, Pylons, TurboGears, web2py, Zope2, ...

# Ruby framework：Camping, Ruby on Rails, Ramaze, ...

# Google App Engine framework：webapp, webapp2

- 統計資料

\* 瀏覽器使用率

\* 最常使用的伺服器端語言

\* 重要網站所使用的語言



## 第2章 Google App Engine 簡介

### (1) 雲端運算

- 「雲端」一詞常用來描述大致的概念，而不牽涉細節部份
  - \* 我們有時會使用某些東西，但對於其內部細節的機制則沒有興趣
  - \* 此即為「抽象化」(Abstraction)：以簡單易懂的概念描述複雜的細節
    - # 例如駕駛汽車：
      - 我們對於駕駛汽車的概念會簡化到一把鑰匙、方向盤、換檔、油門、及煞車等，很少了解汽車成千上萬個零件是如何運作，這就是將汽車駕駛「抽象化」
      - 汽車的細節部份只有在汽車出問題時才重要，而且我們會花錢請人代為修理
  - \* 網際網路也是一種抽象化概念、一種雲端，我們上網處理許多事情，但對於電腦如何連線、資料如何流動或處理等細節亦不在意，因此就以一朵雲將其抽象化

#### - 何謂雲端運算(Cloud computing)

- \* 雲端運算並非全新的網路技術，而是一種全新的網路應用整合，主要都是透過許多的主機，整合原先各自獨立的運算資源，以加速網路服務的運作，進而提升作業效能
- \* 特性：
  - # 數據存在雲端：資料不放在單機
  - # 軟體存在雲端：使用者端不需安裝軟體，以瀏覽器連上 Internet 即可使用
  - # 基於公開的標準協定：Open source project，例如：Linux、Ajax、LAMP
  - # 任何裝置都可以連上雲端：例如 Cell phone、notebook 等
- \* 綜合主要文獻的雲端定義，必須有下列特點才能稱為雲端運算：
  - # 高通用性與高延展性
  - # 快速佈署靈活度與虛擬化
  - # 高可靠性
  - # 可衡量、可被監控與量測、計價的隨需服務
  - # 成本低廉
  - # 多人共享資源池，輕鬆實現不同設備間的資料共享
  - # 雲端運算對用戶端的硬體設備要求低且不限制使用地點



#### - 雲端運算主要的服務模式

- \* 雲端軟體服務(Software as a service, SaaS)
  - # 雲端廠商提供軟體讓使用者直接使用，例如：Google mail、Google calendar ...
  - # 主要廠商：Google (Docs, Gmail, Calendar, ...)、Salesforce.com (CRM, HR)、Zoho (Docs, CRM)、趨勢科技(Trend Micro)、Oracle (Siebel on Demand)、Microsoft (Online service)等
- \* 雲端平台服務(Platform as a service, PaaS)
  - # 雲端廠商提供開發平台，讓使用者開發自己的系統，例如：Google App Engine

- # 主要廠商：Salesforce.com (force.com)、Amazon (EC2, S3)、Google (Google App Engine)、Microsoft (Azure)、Yahoo (Yahoo! Application Platform, YAP)等
- \* 雲端設備服務(Infrastructure as a service, IaaS)
  - # 雲端廠商提供伺服器硬體，有些提供系統執行環境，有些可讓使用者安裝自己的軟體或開發環境(甚至作業系統)，例如：租用虛擬主機
  - # 主要廠商：Amazon (Amazon web services, AWS)、IBM (Blue cloud)、EMC (VMware)、Citrix (Xen server), Citrix System (Xen server)、中華電信 (HiCloud)、HP (Flexible computing services)等
- 雲端與非雲端環境
  - \* 以電話為例：
    - # 有線電話利用前幾個號碼來判別所在位置並將其接通
    - # 行動電話則需在行動網路中「追蹤」到該行動裝置，並將電話轉接到最接近該裝置的基地台，然後接通電話
  - \* 非雲端環境：
    - # 網際網路的運作方式就像有線電話環境一般，伺服器有固定位址，並透過 Internet Protocol (IP)位址來指定，應用程式在固定的伺服器上執行，就像一個有線電話
  - \* 雲端環境：
    - # 應用程式與資料在全世界遊走，網路請求(類似撥打行動電話)需在雲端環境中「追蹤」到該應用程式，然後進行連結
    - # Google 會動態地設定應用程式在哪個資料中心執行(可能選負載較少者)，如果在某個時間應用程式受到某個地區的大量點閱，Google 可能會將程式及部份資料再複製一份到鄰近該地區的資料中心，同時間有幾份程式一起執行(行動電話只有一支！)
    - # 一個應用程式可能在執行中、也有可能不在執行中、有可能在任何地方執行、也可能有好幾份程式正在執行，Google 將這些細節隱藏都起來，並且以最快速最有效率的方式執行
    - # 執行 Google 的雲端程式就好像搭飛機一般，你只管喝紅酒、觀賞電影、聆聽音樂、享受美食、或小憩片刻，所有空服人員與地勤人員會努力的協助你到達目的地
- 自行管理伺服器的缺點
  - \* 用哪個作業系統及哪個版本、是否該安裝某個功能、如何防止網路或實體入侵、是否需要安裝防火牆、如何監控伺服器運作狀況、半夜當機、資料備份、系統備援、用哪個資料庫管理系統及哪個版本、單一資料伺服器或多個、如何應付流量峰值、軟體及硬體升級、冷卻系統... → 管理員的夢魘！
- 雲端運算的優點
  - \* 降低設備及人力成本：不需購(建)置伺服器、控制室、冷卻系統、電力系統、備援系統、資料備份等，亦無需伺服器管理人員
  - \* 沒有升級成本：不需升級硬體或系統軟體
  - \* 優質的伺服器與資料管理：伺服器沒有停機維護時間(Google：99.9%[服務水準](#))，至少有3部備援系統，至少有3份資料備份，資料的各個更新版本均保存而不刪除
  - \* 動態的資源調整：依照實際需求動態調整系統所需資源，例如流量、CPU、及資料等，資源的使用效率更高
  - \* 真正的「雲端」：系統可以在全球資料中心複製多份，以因應大量使用者的需



求，系統就好像漂浮在雲端，不知道到底在哪裡執行，也不知道到底有多少份正在執行

## (2) Google App Engine (Google 應用引擎)

### - Google 資料中心

- \* Google 在全球擁有許多 [資料中心](#)(Data center)，每個中心都有成千上萬部伺服器，[設備齊全](#)
- \* 資料中心的伺服器(甚至整個資料中心)可能需要維修、當機、停電、網路停擺等
- \* Google 研發了一套軟體框架(Software framework)，也就是一個抽象層(Abstraction layer)，將所有細節均隱藏起來，包括資料儲存位置、哪一個軟體在哪一部伺服器上執行、在哪一個資料中心...
- \* 有了抽象層，Google 在資源的重新配置方面有很大的彈性：隨著世界的作息，全球有些地區的人在睡覺、有些地區的人在工作，資料、軟體、執行等都可以動態地「跟著太陽」移動位置
- \* 在人們休息的地區，資料中心就可以進行網路蜘蛛爬行(Web spidering)、建構索引(Index)、執行備份(Backup)、維修、或幫忙處理其他地區超載的運算負擔
- \* 雲端運算的真正價值：資源的動態調整(數量及地區)，使成本效益最大化

### - Google App Engine (GAE)

- \* 於 2008 年推出
  - # 提供網際網路應用程式的虛擬主機服務
  - # 與一般虛擬主機及私有伺服器不同：只針對應用程式所使用的資源付費，程式能夠自動擴充或縮減(Scaling)
    - 資源：CPU 使用量、儲存資料量、網路流量頻寬、資料讀取與寫入量、郵件送出量...
- \* GAE 設定「沙盒」([Sandbox](#))來隔絕不同的應用程式，以確保彼此安全，在沙盒中，應用程式的某些運算是受限的：
  - # 只能透過 URL fetch 或 email 服務連到其他電腦，其他的[網路連結方法](#)是禁止的
  - # 其他電腦只能透過 HTTP request 連到 GAE 的應用程式
  - # 不能寫入或儲存檔案，只能寫入資料儲存庫(Datastore)
  - # 可以讀取檔案，但只限於和程式碼一起上載的檔案
  - # 只能執行以下工作：回應網路請求(Web request)、排隊工作(Queued task)、預定工作(Scheduled task)
  - # 網路請求的處理程式必須在 60 秒內回覆資料，而且回覆之後程式不能再產生子程序(Sub-process)或再執行其他程式碼
- \* GAE 持續監控所有應用程式的執行狀況，確保某個程式不會使用太多資源而影響到其他程式
  - # 如果應用程式花了太久的時間回應某個請求，Google 就會放棄該請求
    - 程式設計師因而必須撰寫好的程式
  - # 如果應用程式持續使用過多不合理的資源，Google 就會將其關閉一段時間
  - # 如果應用程式因為廣受歡迎而使用較多整體資源，Google 會很高興的收取費用
  - # 資源的限制：重點不在於限制整體資源，而在於確保每一次使用者點選網頁的某個連結或元件時，應用程式都能使用合理的資源數量來回應請求，因此每個在 GAE 運行的程式都必須是個「好市民」
- \* GAE 分為三大部份：執行環境(Run time environment)，資料儲存庫(Datastore)，可

### (3) 執行環境

#### - HTTP 的請求/回應週期(Request/response cycle)

- \* 週期開始：使用者在瀏覽器裡點了某個網頁或執行某個動作
- \* 週期結束：新網頁在使用者的瀏覽器中呈現
- \* 基本概念：



#### \* 過程範例：

1. 使用者瀏覽器連上某個 URL (例如：<http://www.google.com/search>)
  - # 瀏覽器開啓網際網路連線到 URL 的伺服器端，然後請求 /search 頁面，此即爲一個 HTTP request
  - # HTTP request 透過網際網路到達適當的伺服器，該伺服器是 Google 在全球的資料中心(Data center)裡幾百萬台伺服器中的一台
3. 伺服器收到該請求，將空白的 /search 頁面傳給使用者瀏覽器顯示，此即爲一個 HTTP response
  - # 至此完成一個 Request 及 Response 週期，通常不會超過 1 秒鐘(當然，這是理想狀況)
  - # 完成一個週期的時間取決於幾個因素：
    - 網頁元素的多寡：頁面數量、影像、音訊、視訊、程式、CSS、JavaScript、...
    - 網路、伺服器、及瀏覽器的速度
4. 使用者接著在搜尋欄位中填入字串，按下「搜尋」或者是 Enter 按鈕，產生另一個 HTTP request
5. 伺服器收到請求後，擷取出該字串並到資料庫裡搜尋，將搜尋結果產生一份新的 HTML 檔案，回傳給使用者的瀏覽器
6. 瀏覽器解析該檔案(HTML, CSS)，將資料格式化然後呈現給使用者
  - # 此即完成另一個 Request 及 Response 週期

#### - HTML 文件類型

- \* 靜態文件(Static document)：內容固定的文件
  - # 當瀏覽器提出請求，伺服器將某份固定內容的檔案回傳給瀏覽器
- \* 動態文件(Dynamic document)：程式執行後產生文件
  - # 當瀏覽器提出請求，伺服器會執行程式然後將執行結果回傳給瀏覽器
  - # 程式：CGI, PHP, ASP.NET, JSP, Python, Ruby on Rails, ...
  - # 程式可儲存在文件裡(程式與 HTML 混在一起)，或與網頁文件分離(程式檔與網頁檔分開)
  - # 伺服器端執行程式後回傳文件，傳回的文件不含程式
- \* 主動文件(Active document)：文件內含程式



- # 當瀏覽器提出請求，伺服器將含有程式的文件回傳，使用者的瀏覽器執行儲存在文件裡的程式然後呈現執行結果
- # 程式：Java applet, JavaScript, Flash, ...
- 系統開發者所需了解的知識
  - \* 整個 Request 與 Response 的細部過程
  - \* 瀏覽器如何利用 HTML 與 CSS 呈現格式化的資料
  - \* 如何利用 Javascript 與 AJAX (Asynchronous JavaScript and XML)來增加網頁互動性
  - \* 瀏覽器如何發出利用 HTTP 的協定來發出 Request：GET 或 POST 兩種型態
  - \* 如何處理使用者在網頁表單中所輸入的資料或欲上載的檔案，這些也都是 Request 的一部分
  - \* 伺服器端：Python 程式設計，Google 的資料儲存(Datastore)，利用 Django 的範本(Template)來產生動態 HTML
- GAE 的執行環境
  - \* 當一個請求(Request)出現時，應用程式開始執行，並由請求處理器(Request handler)處理該請求，處理完畢後應用程式可能會消失
  - \* 應用程式的出現與消失表示網際系統是無狀態的(Stateless)，亦即先前處理的狀態不會保留
  - \* 狀態資料的保存僅能靠在執行環境之外的資料儲存庫(Datastore)
  - \* 因為網際系統無狀態，因此 GAE 可以自由的將各請求配置在不同的伺服器上執行
    - # 兩份請求可能會由不同的伺服器硬體來處理，即使來自同一個使用者
  - \* 每個應用程式待在自己的沙盒裡
    - # 可以讀取自己檔案系統裡的檔案，但不能寫入檔案
    - # 不能讀取其他應用程式的檔案
    - # 僅能透過 GAE 的服務進行網路連結，不能使用其他方式連結網路
    - # 不同的應用程式絕對不會互相干擾
  - \* 存取伺服器的資料受到限制
  - \* 每個請求所使用的資源受到限制，以防止失控(Runaway)的程式
- GAE 提供四種程式語言執行環境
  - \* Python
    - # 在 Python interpreter 上執行，支援 Python 2.x 版本與 CPython
    - # 可使用 Django、web2py、Pylon 等框架，GAE 亦提供 webapp 及 webapp2 框架
  - \* Java
    - # 在 Java virtual machine (JVM)上執行，只要能編譯後能在 JVM 上執行的程式語言都可使用，例如 Java、PHP (使用 Quercus)、Ruby (使用 JRuby)、JavaScript (使用 Rhino interpreter)、Scala、以及 Groovy
    - # 支援 Java servlet 與 Java Persistence API (JPA)
    - # 使用 Google Web Toolkit (GWT)框架
  - \* Go：實驗中
  - \* PHP：實驗中
- GAE 應用程式伺服器模型(Application server model)
  - \* 一個客戶請求被傳送到伺服器，伺服器啟動應用程式，應用程式處理請求並產生回應，然後將回應傳送給客戶
  - \* 每個伺服器的執行環境均事先載入，因此只有應用程式會動態的載入

- \* 越忙碌的應用程式會停留在執行環境越久，依據狀況有些冷門應用程式可能會被剔除

## (4) 檔案與資料

### - 靜態檔案(Static file)：內容不變的檔案

- \* 例如：影像、CSS、JavaScript、沒有動態資料的 HTML
- \* 傳送靜態檔案不需要執行程式，因此不應由應用程式伺服器(Application server)來處理
- \* GAE 提供專門處理靜態檔案的伺服器，這些伺服器的內部架構及網路拓樸都經過精心設計，以提供最佳的靜態檔案處理

### - 關聯式資料庫(Relational database, RDB)

- \* 資料表包括列(Row)與欄(Column)，一列稱為一筆記錄(Tuple, record)，欄位稱為屬性(Attribute)

Table: Teacher

Tid (PK)	name	
101	張三	← Tuple
102	李四	← Tuple
103	王五	← Tuple

↑ Attribute                      ↑ Attribute

- \* 可利用 Join 指令將不同表單的資料集合起來：

Table: Course

Cid (PK)	name	Tid
201	網際系統開發	101
202	自由軟體	101
203	Java 程式設計	102
204	PHP	101

Join:

Cid (PK)	Cname	Tname
201	網際系統開發	張三
202	自由軟體	張三
203	Java 程式設計	李四
204	PHP	張三

- \* 其他資料庫：階級式資料庫(Hierarchical database)，物件資料庫(Object database)

### - Google 資料儲存庫(Datastore)

- \* 類似物件資料庫
- \* 實體與特性(Entity and property)
  - # 一群相同類型的資料稱為一個類型(Kind)，類似關聯式資料庫裡的 Table
  - # Kind 名稱一旦設定即無法更改
  - # 一個類型裡的一筆資料稱為一個實體(Entity)，類似關聯式資料庫裡的 Tuple，實

體內含有一或多個特性(Property)

- 例如：學生資料包括姓名、學號、電話、地址等，這些資料稱為 Student (Kind 的名稱)，一筆學生資料稱為一個 Entity，而姓名、學號等即為 Entity 的 Properties

Kind: Student

姓名	學號	電話	地址	
張三	1010001	1234-5632	台中市霧峰區吉峰東路 168 號 3 樓	← Entity
李四	1010002	3216-5420	台中市霧峰區吉峰東路 168 號 4 樓	← Entity
王五	1010003	2541-2238	台中市霧峰區吉峰東路 168 號 5 樓	← Entity

↑            ↑            ↑            ↑  
Property   Property   Property   Property

# Entity 和 Tuple 類似，但 Entity 的某個 Property 可以有多個值，Tuple 則不行

# 每個 Entity 有一個唯一鍵值(Key)，可以由系統指定或者使程式設計師指定

# Entity key 和 RDB 裡的 Primary key (主要鍵)不同，Primary key 是個欄位，但 Entity key 並非欄位，而是附屬於 Entity 的一項資料

# Entity key 一旦設定即無法更改，Primary key 可以修改

#### \* 查詢及索引(Query and index)

# RDB 可利用查詢來取出多筆資料，查詢是事先規劃好並且是在資料庫中即時執行，但程式設計師可選擇儲存查詢結果，或設計使用索引(Index)來加快查詢速度

# GAE 則非常不同：Datastore 對於每一個查詢都自動產生一份索引，儲存在 index.yaml 檔案中，以加快下次查詢的速度，程式設計師亦可以設定索引

# 註：yaml (Yet another markup language, YAML ain't markup language)為資料或組態設定之格式，利用冒號(:)與縮排來表示關鍵詞與其值

#### \* 交易(Transaction)

# 當許多使用者同時讀或寫同一份資料，資料一致性有可能發生問題

# 例如提款流程如下：

1. 驗證帳戶資料
2. 接受提款請求
3. 檢查餘額
4. 修改餘額
5. 吐鈔

- 假設存款餘額為\$1,500 元，輸入提款金額\$1,000 元，系統執行到第 5 步驟時突然停電而未吐鈔

- 恢復電源後，使用者再次嘗試提款，發現存款餘額只剩 500 元！

# 解決方案：利用 Transaction 技術

- 當一系列的資料庫讀寫步驟遭到中斷，資料應回復到未執行前的狀態

\* 亦即：在一次交易中，要不然所有步驟均成功，要不然所有步驟均失敗，不能部份成功

- 若有許多筆資料需要修改(可能不同 Kind)，則需要設定「跨組」(Cross group)機制，如此 GAE 才知道如何正確分散實體到不同的伺服器上

- 當某個使用者嘗試修改一筆資料，而同時間另一個使用者正在修改該筆資料，Datastore 會立即回覆並行失敗(Concurrency failure)的例外訊息，此時系統會再次嘗試執行該筆交易(程式設計師可以設定執行次數)，此稱為「最佳並行控



## (5) 工作佇列與排程工作

- 工作佇列(Task queue)
  - \* 網際系統的回應一般而言應在 1 秒鐘以內，使用者才會有「順暢」的感覺
  - \* 若有某項工作需花較長時間，即可使用工作佇列，讓該工作稍候再執行，若該工作稍候執行失敗，可以重複執行直到成功
- 排程工作(Cron job)
  - \* 程式設計師可以設定系統在某個特定時間執行某項工作，稱為排程工作
  - \* 可設定某些天的某些時間來定期執行工作

## (6) 開發工具

- GAE 之軟體開發工具箱(Software development kit, SDK)
  - \* 開放源碼
  - \* 分為 Python、Java、Go 三種語言版本
  - \* 可在 Windows、Mac OS X、及 Linux 等平台上執行
  - \* Windows 與 Mac OS X 版本擁有圖形界面開發環境
  - \* 可嵌入至 Eclipse 環境
  - \* 內建有網站伺服器軟體
  - \* 開發及執行過程中會自動產生搜尋索引(檔案 index.yaml)
  - \* 內建有資料儲存庫之資料查詢功能
  - \* 有上載並發佈網際系統的程序

## (7) 管理控制台

- GAE 之管理控制台(Administration console)
  - \* 產生應用程式帳號
  - \* 設定應用程式各種屬性
  - \* 查閱應用程式執行狀況及各項統計資料
  - \* 資料儲存庫查詢及操作
  - \* 設定付費方式

## (8) 其他服務

- 寄送及接收電子郵件
- 發送及接收簡訊：XMPP 協定
- 簡單的影像處理

## 第3章 撰寫應用程式

### (1) 安裝 Python, Eclipse, 及 GAE SDK

- Eclipse + PyDev + Google App Engine SDK 之整合開發環境

- \* 安裝 Python 2.x (GAE 目前尚未支援 Python3)

- # Ubuntu : Ubuntu 軟體中心 → 搜尋 IDLE Python → 選擇 2.x 版本 → 安裝

- # Windows : 下載並安裝 Python 2.x (<http://www.python.org/getit/>, Python 2.x.x Windows Installer, 假設安裝路徑為 C:\Program Files\Python2x)

- \* 安裝 Eclipse

- # Ubuntu : Ubuntu 軟體中心 → 搜尋 Eclipse → 選擇 Eclipse 整合式開發環境 → 安裝

- # Windows

- 下載並安裝 Java JRE (網址 : <http://www.oracle.com/us/downloads>)

- 下載並解壓縮 Eclipse Standard x.x.x

- \* 網址 : <http://www.eclipse.org/downloads/>

- \* Eclipse 為 Portable 軟體，不需要安裝，只要將解壓縮的目錄放到適當地方，再執行 eclipse.exe 即可，亦可建立該執行檔的連結放到桌面，以方便執行

- \* 在 Eclipse 上安裝 Pydev plugins

- # 執行 Eclipse → Help → Install New Software ... → Add → Name: PyDev and PyDev extensions, Location: <http://pydev.org/updates> → OK → 勾選 PyDev → Next → Finish (之後勾選 Brainy)

- \* 下載 Google App Engine SDK for Python

- # 網址 : <https://developers.google.com/appengine/downloads>

- # 解壓縮產生 google\_appengine 目錄，將目錄移至適當地方

- \* 安裝 HTML 與 JavaScript 編輯器：

- # Help → Install New Software ... → Work with:

- <http://download.eclipse.org/releases/kepler> (or Indigo, Juno, ...) → 展開 Web, XML, Java EE, ... 並勾選 JavaScript Development Tools 與 JSF Tools - Web Page Editor → Next → Next → 同意條款 → Finish

- \* 設定檔案與編輯器的關聯：

- # Window → Preferences → 展開 General 及 Editor，選 File Association

- File types: \*.htm, \*.html, Associated editors: Add ... : HTML Editor 或 Web Page Editor

- File types: \*.yaml, Associated editors: Text Editor

- File types: \*.js, Associated editors: Text Editor

- \* 設定編輯器的等寬字體：

- # Window → Preference → General → Editors → Text Editors → Color and Fonts (右頁上方) → Edit → Fonts: (DejaVu Sans Mono), Size: 12 → OK

- \* 設定編輯器顯示行號：

- # Window → Preference → General → Editors → Text Editors → 右邊勾選 Show line numbers → OK

- \* 設定預設編碼為 UTF-8

- # Window → Preference → General → Editors → Text Editors → Spelling →

Encoding: UTF-8

#### - 建立新專案(範例：AppEngineLearn)

\* File → New → Project ... → PyDev Google App Engine Project → Next → Project Name: **AppEngineLearn** → 設定 Interpreter (如下) → Next → Google App Engine Directory: **xxx/xxx/xxx** → Finish

\* 設定 Interpreter：

1. 點擊 Please configure an interpreter ... 之連結

2. 點右上方 New

Interpreter Name: **Python2**

Interpreter Executable: 點右方 Browse → 選執行檔(如下) → OK


\* Ubuntu：選 /usr/bin/python2

\* Windows：選 C:\Program Files\Python2x\python.exe

\* 設定 Run Configuration：

# 右鍵點選專案名稱 → Run As → PyDev: Google App Run

# 右鍵點選專案名稱 → Run As → Run Configurations ... → 將出現兩次的專案名稱改為一個名稱 → Apply → Close

\* 執行專案：點 ，選單中選取專案名稱

#### - Eclipse 的疑難雜症：

\* Console 不見了：Window → Show View → Other ... → General → Console → OK

\*  不見了：Window → Customize Perspective → Command Groups Availability → 勾 Launch → OK

\* Perspective 弄亂了：Window → Reset Perspective

\* PyDev 在 `import webapp2` 顯示 Unresolved import: webapp2：Ctrl-1 → UnresolvedImport

## (2) 最簡單的程式

#### - 最簡單的程式

\* 一個目錄裡包含 2 個檔案：

# 組態檔：app.yaml

# Requester handler 的 Python 程式檔(例如：main.py)

\* Hello world 程式範例：

# Eclipse 步驟: File → New → Project ... → PyDev, PyDev Google App Engine Project → Next → Project name: **helloworld** (名稱有許多限制：只接受英文字母、數字、短橫線等等) → Next → Google App Engine directory: **xxx/xxx/xxx** → Finish

# 設定 Run configuration

# 在 helloworld 目錄中建立以下 2 個檔案：

app.yaml:

```
application: helloworld
version: 1
runtime: python27
api_version: 1
threadsafe: true

libraries:
- name: django
  version: "1.5"
```



```
handlers:
- url: /*
  script: main.app
```

- `application` : 應用程式名稱
- `version` : 版本(可同時存在許多版本)
- `runtime` : 執行環境之程式語言
- `api_version` : Google App Engine 的版本
- `threadsafe: true` : GAE 可並行送出 Request
- `libraries` : 使用到 Django 程式庫, 版本 1.2
- `handlers` : Request handler 的設定

→ `/*` 為常規表示式: 點號表示任何字元, 星號表示其左方的字元可以發生許多次, 亦即在主網址後無論接任何字元或字串之網址, 均以 `main.py` 程式裡的 `app` 實例來處理

- `main.app` : `main.py` 模組裡的 `app` 物件

`main.py`:

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python2.7

import webapp2

class Main(webapp2.RequestHandler):
    def get(self):
        self.response.out.write('<p>Hello world!</p>')

app = webapp2.WSGIApplication([
    ('/*.*', Main)],
    debug=True)
```

- 1、2 行: 指定檔案編碼及使用的程式(對 Python 而言是註解)
  - 3 行: 匯入 `webapp2` 模組
  - `class Main` : 定義一個請求處理(Request handler)類別, 並且是 `webapp2.RequestHandler` 的子類別
    - \* 當應用程式在處理請求時, 首先產生一個 `Main` 實例(Instance), 然後呼叫適當的函式(即 `get()`)
    - \* `get(self)` 函式: 處理 HTTP 之 GET request
    - \* `self.response.out.write()` : 輸出資料
  - `app = webapp2.WSGIApplication()` :
    - \* Web server gateway interface : Python 程式與網站伺服器的溝通介面, 類似 CGI
    - \* 應用程式利用 `webapp2.WSGIApplication()` 類別產生一個實例 `app`
    - \* `[( ... )]` : 利用 Python 的 Tuple 資料結構來對應哪個 URL 要由哪個 Request handler 來處理, 一個 `WSGIApplication` 實例可以處理許多 request
    - \* `debug=True` : 錯誤訊息會顯示在網頁中(真正上線時, GAE 不會顯示錯誤訊息)
- # 在瀏覽器的 URL 欄位中輸入 `http://localhost:8080` 網址, 產生一個 GET request
- 由於在主網址之後沒有任何字元, 因此由 `app.yaml` 檔中設定的 `main.py` 模組中的 `app` 負責處理
  - 由於在主網址之後沒有任何字元, 因此由 `main.app` 實例中的 `Main` 類別負責處理

- 任何在瀏覽器 URL 欄位所輸入的網址，都是 HTTP GET request，因此由 `get(self)` 函式處理

- 應用程式回覆(Response) '`<p>Hello world!</p>`' 字串顯示在網頁中

# 可在 Eclipse 的 Console 區檢查執行情況：

INFO 2013-04-09 15:15:34,441 server.py:528] "GET / HTTP/1.1" 200 58

- GET / : URL request

- 200 : 成功取得網頁([HTTP status codes](#))

# 練習：加上時間顯示

- `import datetime`

- `self.response.out.write('<p>The time is: ' + str(datetime.datetime.now()) + '</p>')`

# 練習：加上完整 HTML 結構

- `self.response.headers['Content-Type'] = 'text/html'`

- `self.response.out.write('<html><head></head><body>')`

- `self.response.out.write('</body></html>')`

- Google App Engine 的疑難雜症

\* 錯誤訊息：

`google.appengine.tools.devappserver2.wsgi_server.BindError: Unable to bind localhost:8080`

→ 表示 Socket 正在使用中，可能是之前的執行無法順利結束，可利用 `fuser -k 8080/tcp` 指令刪除

### (3) 處理器回叫函數

- 處理器回叫模式(Handler callback pattern)

\* 主要概念：將主要的工作交給框架(Framework)來處理，然後讓 Framework 回過來呼叫我們所寫的程式

\* 此模式在物件導向程式語言很常見，例如：圖形介面程式設計

\* HTTP GET request 到達 Framework 的主程式(`webapp2.WSGIApplication`)

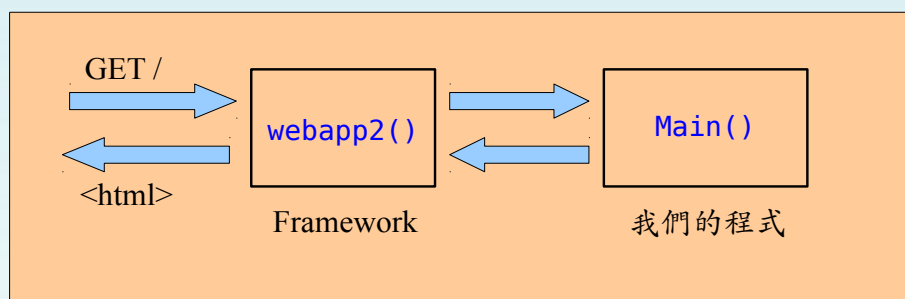
1. Framework 先處理該 Request

2. 然後透過所設定的請求及處理器配對(例：`(('/', *), Main)`)，來呼叫我們的程式

\* 如果是 GET request，呼叫我們程式裡的 `get()` 函式

\* 如果是 POST request，呼叫我們程式裡的 `post()` 函式

3. 最後透過 Framework 將處理結果回傳給使用者



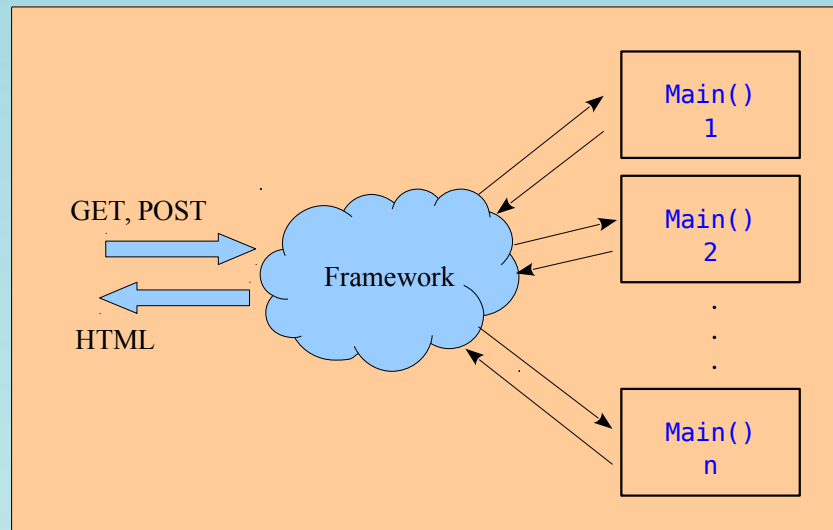
- 如果網路流量高，`Main()` 類別會產生許多實例(Instance)，以分散處理眾多的 Request

\* 每一份 `Main()` 實例各自獨立，有自己的資料(`self`)

\* 我們不知道總共產生多少實例，也不知道哪個實例在哪一部伺服器上執行

\* 我們利用物件導向方式撰寫程式，給予 Framework 許多彈性依照實際需求來重組

- 我們的應用程式，此即稱為抽象化(Abstraction)，因為太多細節是我們所不知道的
- \* 這些細節我們也不在乎，只要 Google framework 與 Google 工程師們確保我們的程式能正確與快速地執行即可，這也就是雲端基礎建設所提供的舒適性




## (4) HTTP Request

### - HTTP request

- \* 在瀏覽器的 URL 欄位中輸入網址時並按下 Enter 後，即是要求瀏覽器從網站中取回一份檔案
- \* 例如：<http://www.cyut.edu.tw/~yltang/index.html>
  - # 網址分為三部份：
    - http://：網路通訊協定
    - [www.cyut.edu.tw/](http://www.cyut.edu.tw/)：主機網址
    - ~yltang/index.html：檔案位置(含目錄)
- \* 瀏覽器遵循 HTTP 協定取回 index.html 檔案：
  - # 透過預設 Port 80 與 www.cyut.edu.tw 主機連結
  - # 連結成功後，瀏覽器送出以下指令請求檔案  
`GET http://www.cyut.edu.tw/~yltang/index.html HTTP/1.1`
  - # 主機接到 Request 後，找到 index.html 檔案，將檔案內容 Response 給瀏覽器，然後結束連結
  - # 最後瀏覽器接收到 Response，將其內容呈現，至此完成 request/response 週期

### - 觀察 HTTP 的動作

- \* 利用 Firefox 的 [Firebug](#) 插件(Joe Hewitt 撰寫)來觀察 HTTP 細部資料，並協助除錯
- \* 點選 Firefox 右上方  圖示，或檢視 → Firebug
- \* 點選「網路」，「清除」後，再點選網頁之超連結，即可看到 HTTP request

## (5) HTML：建立表單

### - 猜數字遊戲 GET 版：

- 建立一個網頁表單讓使用者可以輸入數字，然後撰寫 GAE 程式來處理輸入
- \* 程序：
  1. 使用者在瀏覽器的 URL 欄位輸入網址 → 產生 GET request
  2. 伺服器接到 Request，將表單網頁回傳



3. 瀏覽器顯示表單網頁

4. 使用者輸入資料，按下「送出」按鈕 → 表單透過 GET 協定送出資料

5. 伺服器接收資料，處理資料，並產生新頁面傳送給瀏覽器

\* HTML 表單：

```
<form method=get action=/>
```

```
  請猜一個數字:<input type=text name=guess>
```

```
  <input type=submit value=送出>
```

```
</form>
```

# **method**：告訴瀏覽器利用 HTTP GET 協定將資料送到伺服器

# **action**：告訴瀏覽器，在使用者按下「送出」按鈕後，應該送到哪個 URL

# **input**：傳送資料的欄位

# **type**：資料型態

# **name**：變數名稱

# **value**：值

\* 程式：

# 建立一個新專案：guessnumber-get

# 建立 app.yaml 檔案：

```
application: guessnumber-get
version: 1
runtime: python27
api_version: 1
threadsafe: true
```

```
libraries:
```

```
- name: django
  version: "1.5"
```

```
handlers:
```

```
- url: /.*
  script: main.app
```

# 建立 main.py 檔案：

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python2.7

import webapp2

class Main(webapp2.RequestHandler):
    def get(self):
        # Read the form input which is a single line
        answer = 4
        guess = -1
        try:
            guess = int(self.request.get('guess'))
        except:
            guess = -1
        self.response.headers['Content-Type'] = 'text/html'
        self.response.out.write('''
<!DOCTYPE html>
<html>
  <head>
    <title>猜數字</title>
  </head>
  <body>''')
```

```

self.response.out.write('<p>猜數字：1 到 10 之間</p>')
self.response.out.write('<p>你所猜數字是 ' + str(guess) + '</p>')
if guess < answer:
    self.response.out.write('<p> 你的數字太小</p>')
elif guess == answer:
    self.response.out.write('<p> 恭喜! </p>')
else:
    self.response.out.write('<p> 你的數字太大</p>')
self.response.out.write('''
<form method=get action=/>
    請猜一個數字:<input type=text name=guess>
    <input type=submit value=送出>
</form>
</body>
</html>''')

app = webapp2.WSGIApplication([
    ('/*.*', Main)],
    debug=True)

```

- `self.request.get()`：取出 GET request 裡所傳送的變數

\* 觀察 Eclipse 之 Console 資料(或透過 Firebug)：GET 所傳的參數直接顯示在 URL 資料中(GET /?guess=2 HTTP/1.1)

- 猜數字遊戲 GET-POST 版：

\* 最常見的表單運作流程：

1. 進入頁面：利用 GET 產生空白表單網頁
2. 使用者輸入資料
3. 使用者按下「送出」按鈕後，利用 POST 將資料傳到伺服器
4. 再次呈現表單網頁

\* main.py 程式：

# 將 GET 和 POST 都會使用到的資料設為共用字串變數(part1String, formString, part2String)

# 定義兩個函式

- `get(self)`：顯示空白表單
- `post(self)`：接收資料、處理資料、將結果回傳

```

# -*- coding: utf-8 -*-
#!/usr/bin/env python2.7

import webapp2

class Main(webapp2.RequestHandler):
    part1String = '''
    <!DOCTYPE html>
    <html>
    <head>
        <title>猜數字</title>
    </head>
    <body>'''
    formString = '''<form method=post action=/>
        請猜一個數字:<input type=text name=guess>
        <input type=submit value=送出>
    </form>'''
    part2String = '''</body></html>'''

```

```

def get(self):
    self.response.headers['Content-Type'] = 'text/html'
    self.response.out.write(self.part1String)
    self.response.out.write('<p>祝好運! </p>')
    self.response.out.write(self.formString)
    self.response.out.write(self.part2String)

def post(self):
    try:
        guess = int(self.request.get('guess'))
    except:
        guess = -1
    answer = 4
    if guess < 0:
        msg = '<p>請輸入數字</p>'
    elif guess < answer:
        msg = '<p>你的數字太小</p>'
    elif guess == answer:
        msg = '<p>恭喜! </p>'
    else:
        msg = '<p>你的數字太大</p>'
    self.response.headers['Content-Type'] = 'text/html'
    self.response.out.write(self.part1String)
    self.response.out.write('<p>數字: ' + str(guess) + '</p>')
    self.response.out.write('<p>' + msg + '</p>')
    self.response.out.write(self.formString)
    self.response.out.write(self.part2String)

app = webapp2.WSGIApplication([
    ('/*', Main)],
    debug=True)

```

\* 觀察 Eclipse 之 Console 資料(或透過 Firebug)： POST 不會顯示參數(**POST / HTTP/1.1**)

- 何時使用 GET 或 POST ?

\* GET

- 可直接在瀏覽器的 URL 欄位中輸入指令而執行
- # 傳送的資料沒有安全疑慮(資料會顯示在 URL 中)
- # 傳送的資料不會改變伺服器資料的狀態(不寫入資料)

\* POST

- 無法直接在瀏覽器的 URL 欄位中輸入指令而執行
- # 傳送的資料有安全考慮(資料不會顯示在 URL 指令中，但資料攔截者還是可以取出)
- # 傳送的資料會改變伺服器資料的狀態(會寫入資料)

\* 最安全的傳輸方式：<https://> (所有資料均加密)

- 練習

1. 建立一網頁，有兩個資料輸入欄，使用者輸入 2 份資料，系統顯示何者較大
2. 建立一網頁，有兩個資料輸入欄，使用者選擇運算方式(+, -, \*, /)，再輸入 2 份資料，系統顯示計算結果