1. **TCP/IP協議族**

連接導向的TCP/IP協議族包含了TCP、IP、UDP、ARP及HTTP等協議，由國際互連網工程任務組IETF(Internet Engineering Task Force)所制訂。TCP/IP協議族目的是為了實現網路體系的通訊，常見的通訊協議族還有OSI、IPX/SPX等，其中OSI將網路體系分為七層，而TCP/IP協議族將網路體系分為四層(差別為最底層的物理實體層，傳輸層上方連線的會議層、格式轉換的表現層)。

1. **應用層**

此層為連線成通後進行的通信應用，包含的協議有HTTP、Telnet、Ftp、Email、STMT、DNS及SNMP等。

1. **傳輸層**

此層主要的協議有TCP與UDP，TCP的工作是建立穩定傳輸，並指派Port通信端口，而UDP工作是將通信的報文進行分組。連線與斷線流程如下：(C=Client、S=Server)

TCP三次握手：C請求連線→S授予連線→C確認連線。

TCP四次分手：C請求斷開→S確認請求－(剩餘傳輸完成)→S請求斷開→C確認斷開。

1. **網路層**

路由器會建立查詢IP標頭的路由控制表，並由IP協議決定封包的來源與目的地路由端，將封包轉送。與此層的協議包含IP(IPv4、IPv6)、ICMP等。

1. **數據鏈路層 (網卡接口層)**

交換機會建立查詢MAC的地址轉發表，通過ARP與RARP協議取得兩端的MAC地址。

建立連線時，每一層都需要包裝兩端的首部字段，再送往下一層，以HTTP協議來說，訊息會包含：[MAC首部/IP首部/TCP首部/HTTP首部/主體數據]，而建立的連線是基於字節傳輸的P2P全雙工連線。

1. **Server伺服器**

網際網路存取互相連結的超文字(Hypertext)系統，就是俗稱的WWW(World Wide Web)全球資訊網，使用統一資源標識符(URL)將資訊傳給請求的使用者。

HTTP傳輸流成：URL獲取服務器→HTTP傳遞訊息→HTML解析訊息。

HTTP工作流程：Client建立TCP連線→Client請求資源→Server給予資源→Client顯示資源。

HTML的報文包含了請求行、狀態行、首部字段、其他報文首部字段、報文主體五種類別。

1. **WAMP/LAMP**

讓電腦建設WWW伺服器，通常是透過Apache伺服器連接MySQL資料庫，達成運行PHP網頁的平台，稱為**AMP系統**。WAMP/LAMP則表示在作業系統Windows與Linux上使用AMP系統。

WAMP/LAMP可以透過套件包下載並一次安裝，常見的套件包有WampServer、XAMPP、AppServ、AMPPS或是UniformServer等。

1. **Website網站**

現在我們所撰寫之網頁標準由**全球資訊網協會(W3C)**所訂定，由服務的角度分類，提供HTTP服務的稱為伺服端(Server Side)，而向伺服器提出要求、取得回應瀏覽的稱為用戶端(Client Side)。HTTP通訊協定中，並沒有設計保存連線狀態的功能，稱為**Stateless** Connection特性。

多人同時撰寫的網頁需要使用**版本控制(Revision control)**，版本控制是一種軟體工程技巧，藉此能在軟體開發的過程中，確保由不同人所編輯的同一程式檔案都得到同步。最常見的版本控制系統為SVN(Subversion)與Git，**SVN屬於集中式版本控管**(VCS)而**Git屬於分散式版本控管**(DVCS)，分散式版本控管因為每個人都有一份完整的儲存庫(Repository)所以對於資料的保存上更安全，而且分散式透過使用方法的調整，可以被當作集中式來使用，因此Git是先進於SVN的版本控制系統。

最常見的Git版本控制托管服務為**Github** ，它是一個Git的托管網站，可以讓開發者將自己的專案儲存到網路上，與任何人分享，我們可以方便的使用Git在本機進行專案管理。

* 1. **Traditional Web**

傳統的PHP網頁，同時有 HTML、CSS、JavaScript、PHP與SQL**五種語言並存**，具有複雜度高、難以傳承、不易除錯等缺點，也因此傳統PHP一直被戲稱是義大利麵程式碼。

* 1. **CMS**

CMS(Content Management System) **内容管理系统**，此系統可透過以建置好的模板進行網站架設，一個CMS系統網站會同時擁有前台(瀏覽)與後台(管理)兩個主要頁面，而使用者可以簡單的使用後台管理前台頁面上的所有資訊，或是下載網路的模板資源，幾乎不用撰寫任何程式碼就可以直接套用各種頁面特效。

常見的CMS系統有Drupal、WordPress、Joomla與XOOPS等。

* 1. **Framework**
     1. **框架概念**

函式庫與框架的差異：Library 定義很多方便好用的函式供我們使用，而專案整體架構 (目錄、檔案) 則由開發者自行決定；Framework強勢定義整個架構的寫法，我們必須遵守它所定義的架構，去將程式碼填進去。

PHP Framework (框架)是使用PHP之OOP實現MVC(Model-View-Controller)架構，將模型、視圖與控制器分開，因此所寫出來的程式碼具有容易維護、團隊合作等優點。

常見的Framework有Laravel、Phalcon、Symfony、Yii以及CodeIgniter等，使用者最多的是Laravel。

* + 1. **MVC架構**

統一資源定符URL會先通過路由器取得get請求，再導向控制器，而控制器只是路由器導向的一個類別參數，路由也可以直接顯示Html語法或是呼叫視圖。

* **模組Model**

模組透過控制器呼叫，主要功能為存取資料庫。

* **示圖View**

示圖可透過路由、控制器或是其他示圖呼叫，視圖即模板(Template)，內容主要為Html語法與少量的模板參數或是簡單邏輯，通常會通過控制器給予模板參數。

* **控制器Controller**

路由聯繫控制器，控制器呼叫服務、存取模組，再將參數交給視圖進行顯示。

* + 1. **框架技術**

優秀的框架需要具有**組件化開發(Composer)**、**IoC容器技術(Inversion of Control)**、**分布式應用架構設計**。

* **組件化開發：**讓使用者可以快速加載Open Source的現有程序模塊，並提供模塊的版本管理與更新功能，所有的模塊都有共同的規範與接口，規範為PHP-FIG群所制訂的PSR編碼規範，其中最主要的是PSR-0和PSR-4之代碼自動加載(Autoloader)標準，包含了加載虛擬路徑的映射關係。
* **IoC容器技術：**控制反轉IoC(Inversion of Control)即是依賴注入DI(Dependency Injection)。依賴關係是指類別引用類別，類別內部的依賴具有不易維護、解讀等缺點，因此需要將依賴關係往外拉，從外面注入依賴類別，而IoC容器技術則是提供了一個容器來實現依賴注入的功能。

Laravel架構的容器儲存了\*回調(閉包)函數與\*服務實例，其使用變數$bindings儲存回調函數、$Instances儲存共享的\*單例。詳細功能敘述如下：

**\*回調函數**又稱為閉包函數，表示被return function的被製造函數。

**\*服務實例**即是已具類別的物件，產生類別物件稱為實例化。

**\*單例**為不重複的共享回調函數，只能創建一次，再呼叫即是使用。

* + **bind服務綁定**：將服務名稱與服務實體綁定至容器。
  + **make服務解析**：將已綁定的服務透過服務名稱解析出來。
  + **build服務生成：**將以解析的服務生成功能實例。
* **分布式應用架構設計：**是面向接口編程技術之一，其透過數個獨立服務器組成架構，服務與服務之間通過接口規範依賴關係，遵循接口的設計具有低耦合、高可擴性等特性。

1. **Language程式語言**

網頁語言就是純文字，所以所有的純文字編輯器都能撰寫網頁，其中有一種叫做IDE(Integrated Development Environment)整合開發環境編輯器，IDE編輯器通常還提供了除錯、協助、提示、測試等功能，常見的網頁IDE編輯器有NetBeans IDE、Zend Studio、NuSphere PHPEd、PHP Coder等。

提供使用者介面表現層(Presentation)的稱為前端(Front-End)，其包括Html、CSS、JavaScript等，另外在伺服器、資料庫做存取運算的稱為後端(Back-End)，其包括PHP、SQL等。

* 1. **PHP**

PHP(Personal HomePage)是目前最常見的動態網頁使用之語言，屬於直譯式Script，在網頁送出之前由伺服器先執行，具有跨平台、開放原始碼、功能強大且容易學習等優點。

* 1. **HTML/CSS**

超文本記錄語言HTML(HyperText Markup Language)與層疊樣式表CSS(Cascading Style Sheets)為網頁顯示之最基本要件，兩者結構都必須符合W3C之HTML/XHTML標準。

動態超文字標記語言DHTML(Dynamic HTML)，指的是網頁內容隨著用戶或電腦程式提供的參數而變化的HTML，通常是結合javaScript與css透過DOM文件物件模型創造的動態效果。

常見的CSS Framework有BootStrap、Pure css、less、sass等，使用框架可以簡單設計出RWD適應性網頁設計。

* 1. **JavaScript/API**

JavaScript是一種透過Java所開發衍生出的一種全新語言，程式碼是伺服器送出之後才由瀏覽器執行，其標準由國際標準組織ECMA International所制定，其規格為**ES(ECMAScript)**，ES5是目前支援與使用最廣泛之JavaScript版本。常見的JavaScript Framework有React、AngularJS、Enyo.js、ExtJS與Meteo等。

許多網站會使用HTML或JavaScript語言開發API(Application Programming Interface)應用程序接口，例如FaceBook或是Google，使用者可以套用AIP之開放程式碼，達成特定服務或是網頁效果之功能。

用於網頁程式的稱做用戶端JavaScript，另外還有一種設計成伺服器端和網路應用的執行環境，例如Node.js，稱做伺服器端JavaScript。

* 1. **Structural Query Language(SQL)**

結構化查詢語言SQL為存取資料庫使用之語言，常見的SQL資料庫有MySQLi、MS-SQL、MariaDB以及PostgreSQL，目前最普及的MySQLi屬於關連式資料庫(Relational Database)，由數個**欄位資料值連結**之二維資料表所組成。

SQL最常見的攻擊為**SQL隱碼攻擊**(SQL injection)，是一種在傳輸字串中夾帶SQL指令，讓SQL忽略檢查的一中方法。其解決方法大致上有兩種，第一種是將不安全字元替換，或是使用PHP addslashes 或SQL real\_escape\_string之類的語法將不安全字元前加入跳脫字元(反斜線)；第二種方式是現在的主流，透過呼叫Statement物件使用Bind Column(綁定列)的功能，能讓不安全字元無法產生效用。

另外不是使用SQL語言的資料庫，稱為NoSQL，在目前許多環境下它是跟SQL共存的。

1. **Technology技術**
   1. **Session/Cookie**

由於HTTP協定具有Stateless的性質，每次的要求都不會互相影響，因為server並不會紀錄之前的狀態，而要讓server記住client的行為與資料時，就需要cookies與Session的協助。

**Cookie**是一種**存在client端**瀏覽器資料夾下的純文字記錄檔，網頁讀取時會載入header欄位。因為Cookie存在客戶端，因此資料很容易被取出或是竄改，故Cookie不適合存取跟安全性有關的資料。在儲存Cookie狀態時必須給予時間標記，也就是Cookie的生命期限，若沒有設置時間標記則創造的Cookie會隨著瀏覽器關閉而刪除。每個網域都有自己的Cookie檔，通常瀏覽器不會允許Cookie超過4K，不同網域的Cookie無法相互存取。

**Session**儲**存在Server端**，Session也需要Cookie的輔助才能產生運作，因為Session id的資是以Cookie的形式儲存於用戶端，用戶端再透過Session id連線至Server請求資料，而儲存在伺服器的Session相對於Cookie就安全許多，但也因為裡面儲存的都是重要資訊時常成為駭客攻擊的目標。Session的生命周期通常為24小時，或是瀏覽器關必為止，不過使用者也可以藉由修改php.ini的session.gc\_maxlifetime參數改變Session的生命週期。

* 1. **Responsive Web Design(RWD)**

適應性網頁設計(Responsive Web Design, RWD)，是一種在不同螢幕解析度(手機、平板、電腦)中都能自動調整頁面內容的網頁，RWD本身是透過JavaScript控制CSS樣式表設計而成，常見的CSS 框架BootStrap就可以簡單設計RWD網頁。

* 1. **Document Object Model(DOM)**

文件物件模型(Document Object Model, DOM)是HTML、XML和SVG文件的程式介面。它提供了一個文件樹的結構化表示法，並定義讓程式可以存取並改變文件架構、風格和內容的方法。雖然常常使用 JavaScript 來存取 DOM，但它本身並不是JavaScript語言的一部分，而且它也可以被其他語言存取。

* 1. **jQuery**

jQuery是全世界使用最多的**Javascript Library**(函式庫)，jQuery的主要功能為簡化Javascript語法，它能取代DOM的物件模型存取Html元素，並提供使用AJAX的技術函式。還有另一套專門用於行動裝置的函式庫叫做jQuery Mobile。

* 1. **jQuery UI**

jQuery UI是jQuery的擴充套件，提供了多種**使用者介面UI**(User Interface)元件，能夠呈現出Html CSS JavaScript的搭配效果，主要功能為美化網頁。

* 1. **Asynchronous JavaScript and XML(AJAX)**

AJAX全名為 Asynchronous JavaScript and XML，是一種創造互動式網頁應用的網頁開發技術，也是目前JavaScript的主要應用技術之一。透過**非同步的JavaScript與XML**，AJAX可以讓網頁在不重新整理的狀態下，動態的從網路下載XML與Json資料並顯示，POST資料或是立刻存取資料並對其做出回應。

**同源政策(Same Origin Policy)**是JavaScript的一種安全限制，所謂同源是指兩份網頁具備相同協定、埠號(如果有指定)以及主機位置，這就讓AJAX存取網路資源時存在著**跨域(Cross Domain)**的問題，常見的解決方法有以下幾種：

* + - * **允許跨資源共享：**跨資源共享CORS(Cross-Origin Resource Sharing)是W3C新定的規範，網頁可以從header設定中允許CORS的網域，藉此解除JavaScript的區域限制。但是此方法會存在兩個問題，一個是允許CORS的動作必須從主機端設置，因此許多空間並不允許這個設定；另一個是瀏覽器本身也有安全性的限制，允許了CORS後還要解除瀏覽器的跨域限制。
      * **內置框架通信：**此為相當普及的跨域方式，透過iframe的置入框架載入資源，並利用location.hash進行父與子框架的傳値，完成跨域。
      * **外部Script載入：**外部的Script並不受同源政策的限制，因此先使用外部的Script載入資料後再處理便能解決跨域問題，最常見的技術稱為JSONP(JSON with Padding)。
      * **PROXY代理機制：**由代理伺服器的代理程式存取資源，由於代理程式是置於自己的網域之下，因此不存在跨域的問題，牙虎提供的YQL服務就是一種代理機制。

常見的AJAX框架有Prototype、DOJO，函式庫有xajax、AJASON、flxAJAX、AjaxAC。

另外，**單頁應用架構SPA**(Single Page Application)就是使用AJAX為主要通訊方法的一種架構，其實現方法為：Client使用Javascript的服務，透過databus將資料透過AJAX送出request，送至Server的CGI介面後，經過php服務與模組處理後，再將資料response至Client的databus，透過Javascript服務進行處理。

|  |  |  |
| --- | --- | --- |
| SPA(Single Page Application) | | |
| Client | Server | Client |
| jsService > databus > request(AJAX) > | CGI > phpService&model > CGI > response > | databus > jsService |

SPA架構的優勢為前後端分離、架構清晰，具有良好的互動體驗。

* 1. **Procedural / Object-Oriented Programming(OOP)**

資料庫存取技術可分為**程序導向(Procedural)**、**物件導向OOP**(Object-Oriented Programming)以及**物件關聯對映ORM**(Object Relational Mapping)。

MySQL僅具程序導向的存取技術，使用一般程序語法對資料庫進行存取，而MySQLi是具有物件導向存取功能的進階版本，每個資料庫物件都有各自的屬性、方法、類別、繼承等，MySQLi詞尾的i表示進階(Improvement)。

PHP5內建的PDO(PHP Data Object)則是具有抽象層的物件導向存取技術，操作上還是必須自己撰寫SQL語法，與面相對向的ORM算是沾上一點邊。

* 1. **Object-Relational Mapping(ORM)**

所謂的物件關聯對映ORM，就是將**關聯式資料庫映射至物件導向**的資料**抽象層**(Abstraction Layer)技術。其理念是將資料庫的內容映射為物件，讓程式開發人員可以用操作物件的方式對資料庫進行操作，而不直接使用SQL語法對資料庫進行操作，程式設計師不用管底層的資料庫系統是哪種廠牌或哪個版本的資料庫(如：SQL Server、Oracle、DB2、MySQL、Sybase、DBMaker等)，僅須用同一套語法撰寫存取資料庫的邏輯。常見使用物件關聯對映技術的資料庫有Metabase、PEAR:DB及ADODB等。

1. **Format格式**
   1. **XML**

XML(Extensible Markup Language)可延伸標記式語言，是一種標記式語言。標記指電腦所能理解的資訊符號，通過此種標記，電腦之間可以處理包含各種資訊的文章等。

XML也廣泛使用於網站的設計上，資料庫配合匯出的XML可以有效簡少資料庫的存取次數，匯出的XML檔可以做為各種資料交換使用，開放資源平台也常將XML與JSON做為交換的資源格式。使用PHP生成XML的方式，除了直接生成XML字串格式之外，也可以使用DomDocument、XMLWriter、SimpleXML等物件產生。

XML是以<標籤>設計的一種樹狀格式，例如：

<?xml version="1.0"?>

<headers>

<Accept> text/html,application/xhtml+xml,application/ </Accept>

<Accept-Encoding> gzip, deflate, sdch </Accept-Encoding>

</headers>

* 1. **JSON**

JSON(JavaScript Object Notation) JavaScrip物件符號是表示物件的一種格式。由於本身就是JavaScript語言，因此比XML更適合使用於網頁的資源交換。

JSON以{}表示物件、[]表是陣列，內容中不管是property或是value都需要用引號框起，property與value用:設定，複數的property用,格開。例如：

{

"headers": {

"Accept": "text/html,application/xhtml+xml,application/ ",

"Accept-Encoding": "gzip, deflate, sdch"

}

}

1. **Else其它**
   1. LARAVEL特區
      1. Laravel底層是使用Symfony框架模塊進行開發，整個系統功能調度與配置的核心為IoC容器，它是所有服務運行時所需要的載體，實現上使用工廠(Factory)模式，採用Head first設計，將變化值與固定值分離，達成面向對象的原則。IoC控制反轉具有依賴注入的特性，將依賴提至容器外進行注入，注入時通過接口(Interface)編成進行限制。

請求處理階段時，會透過分層的中間件(Middleware)來處理不同種類的功能，中間件每一層都有獨立的功能，使用裝飾者模式將每一層如洋蔥般包起，每一層通過管道(Pipeline)聯繫功能，其功能依序為驗證維護模式、cookie解密、session開啟、驗證crsf令牌。

使用者發出請求後，Laravel程序啟動，會先自動加載並實例化IoC容器，再透過容器註冊基礎服務提供者、實例化Kernel核心。Kernel會配置請求環境並加載與啟動服務，將請求實例化。請求處理時會先經過Middleware，再透過Router與Controller通知View發送響應，於響應發送後終止Middleware。

* + 1. 請求到響應執行過程

|  |  |  |  |
| --- | --- | --- | --- |
| 程序啟動準備階段 | 請求實例化階段(透過Kernel) | 請求處理階段 | 響應發送與程序終止 |
| 1. Autoload 2. IoC容器實例化 3. 基礎服務提供者註冊 4. Kernel實例化 | 1. 環境檢測 2. 配置加載 3. 日記配置 4. 異常處理 5. 外觀註冊 6. 服務提供者註冊 7. 啟動服務 | 1. Middleware 2. Router 3. Controller | 1. View 2. 終止Middleware |

* 1. GitHub透過TortoiseGit操作
     1. Git環境、TortoiseGit套件與GitHub帳號
        1. 安裝Git (windows)

至官網<https://git-scm.com/>下載安裝包，透過下一步安裝，除了Adjusting your PATH需要選Use Git from the Windows Command Prompt、Configuring the terminal emulator需要選Use Windows’default console window，安裝完成後檢查環境變數是否有C:\Program Files\Git\cmd;，最後至cmd輸入指令git version檢查安裝。

* + - 1. 安裝TortoiseGit (windows)

至官網<https://tortoisegit.org/>下載安裝包，透過下一步安裝，除了Choose SSH Client需要選TortoiseGitPLink,coming fromPutty，可以連語言包一併下載安裝，安裝完成後隨意資料夾點右鍵>TortoiseGit>Settings，General菜單內有語言，可更改至繁體中文。

* + - 1. 申請GitHub帳號

至官網<https://github.com/>註冊，註冊完成後選擇Create a new repository創造儲存庫，創造完成後記下儲存庫的網址，例如https://github.com/sdfsds5271463/test，此即為推送拉取之任意URL(Arbitrary URL)。

* + 1. 版本庫資料夾建置與基本動作

選擇一個空的資料夾，右鍵>Git在此建立版本庫，資料夾內會增加一個.git的隱藏版本資料夾，此資料夾即是版本庫。操作時，右鍵>Git同步(sync)>顯示日誌(左下，show log)，從日誌的版本圖按右鍵，可以進行基本版本操作動作：

* + - 1. 提交(commit)：本機版本庫更新。
      2. 切換/**取出**(switch/checkout)：本機取出並還原成舊版本的資料。
      3. 推送(push)：將本機資料推送至遠端。
      4. 拉取(pull)：將遠端資料拉取至本機。

[P.S 與遠端連繫時，選擇任意URL(Arbitrary URL)並輸入，再勾選自動載入Putty金鑰(Autoload Putty Key)即可。]

* + 1. 衝突(Conflict)處理

當兩台以上電腦推送(Push)相同版本時，第一台以後的電腦便會發生衝突(Conflict)而導致推送失敗，處理方法為：

* + - 1. 在Git同步視窗中，點選拉取(Pull)，先取回線上版本的檔案。
      2. Git同步視窗下方切換至衝突頁籤，雙擊衝突檔案查看，並進行變基。
      3. 衝突視窗進行變基時，可以看到兩個版本衝突的程式碼，右鍵衝突碼，可以選擇改用(留下中之一)、相加(兩者共存合併)，處理完成所有衝突後，點選標記為已解決，即變基完成。
      4. 變基完成後，再次提交(commit)>推送(push)，即完成衝突處理。
    1. 分支(Branch)處理

版本主幹叫做Master，提交(commit)時亦可創建任何版本分支。

* + - 1. 新建分支：提交時勾選新建分支。
      2. 選擇分支：右鍵TortoiseGit，選擇**切換**/取出(switch/checkout)，選擇分支名稱即可。
      3. 合併分支：右鍵TortoiseGit，選擇合併(Merge)，會以你目前的分支為主，將你所選擇的其他分支較新的檔案併入。
      4. 合併分支時發生衝突：右鍵TortoiseGit，選擇編輯衝突，完成標記解決，之後再次提交>推送。
    1. .

測試同步