



Web 應用程式技術概要

綱 要

- 一、Web Application
- 二、TCP/IP 網路模型
- 三、HTTP 協定

綱 要

- 一、Web Application
- 二、TCP/IP 網路模型
- 三、HTTP 協定

Web Application

- *Web Application*

- 直接由 *Web* 來進行應用程式服務。
 - *ex : web email service , shopping cart ...*
- *Client* 端
 - *Web 瀏覽器 (IE、Netscape 或 Mozilla 等 ...) 或任何支援 HTTP 協定的應用程式接可直接進行 web application 服務。*
- 優點：
 - 使用者不需要安裝該應用程式於本身的系統中。
 - 應用程式的版本更新與 *Bug* 修正可立即同步進行並反應給所有 *End user*。

Web Application

- *passive* 與 *active* :

- *Web application* 對於資源服務的處理可分為 *passive* 與 *active*

- *passive* : 靜態的資源服務，例如：*html*、*txt* 或 *css* 等 ...
 - *active* : 動態的資源服務，例如：*Servlet*、*JSP* 等 ...

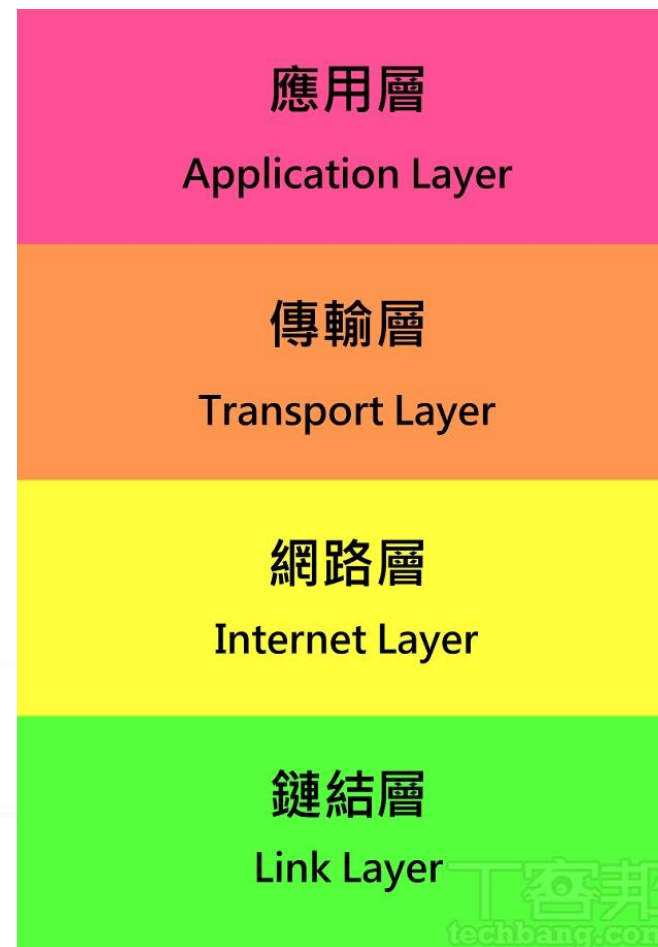
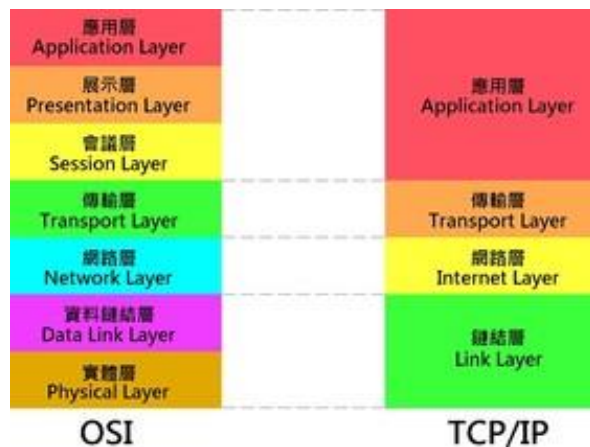
Web Application

- *Web Application Server* :
 - 所有的 *web application* 都應該依附在該 *web application server* 中。
 - 並統一由 *server* 負責管理與統籌系統資源的分派
- 誰來執行 *servlet*、*jsp* ?
 - *Servlet Container* (*servlet container* 本身就是隸屬於 *web application server* 的一部分)。
- *JavaEE Application Server (AP Server)* ?
 - ✓ 要符合 *JavaEE* 規格的 *application server* 至少須包含下列幾項服務：
 - *Servlet container*、*EJB Container*、*JNDI Server* 與 *JMS Server*。
 - 常見的 *JavaEE application Server* : *TomEE*、*JBoss*、*WebLogic*、*WebSphere* 等 ...

綱 要

- 一、Web Application
- 二、TCP/IP 網路模型
- 三、HTTP 協定

TCP/IP 網路模型



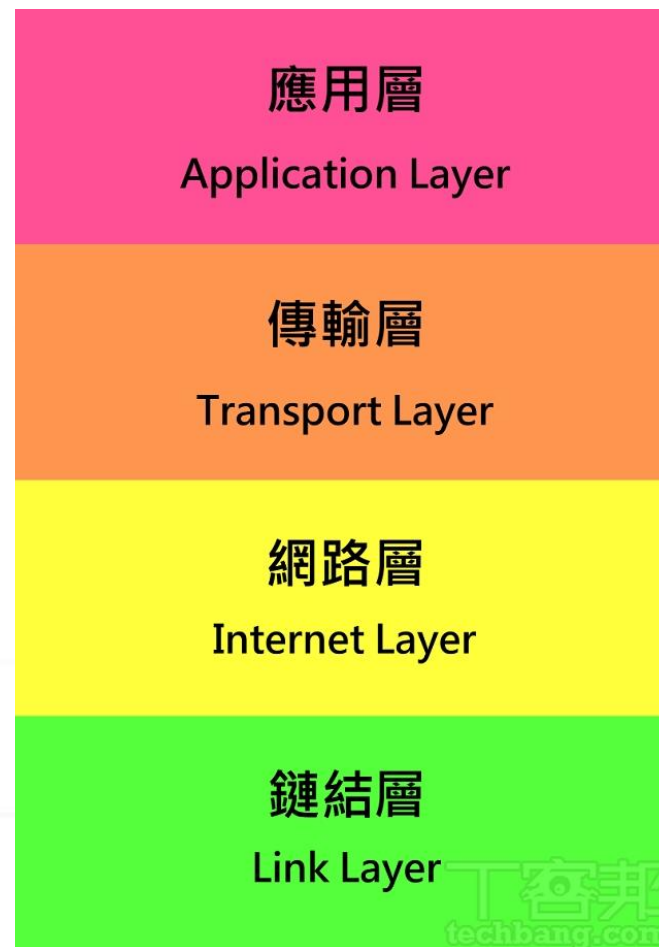
TCP/IP 網路模型

HTTP、FTP 如何包裝數據

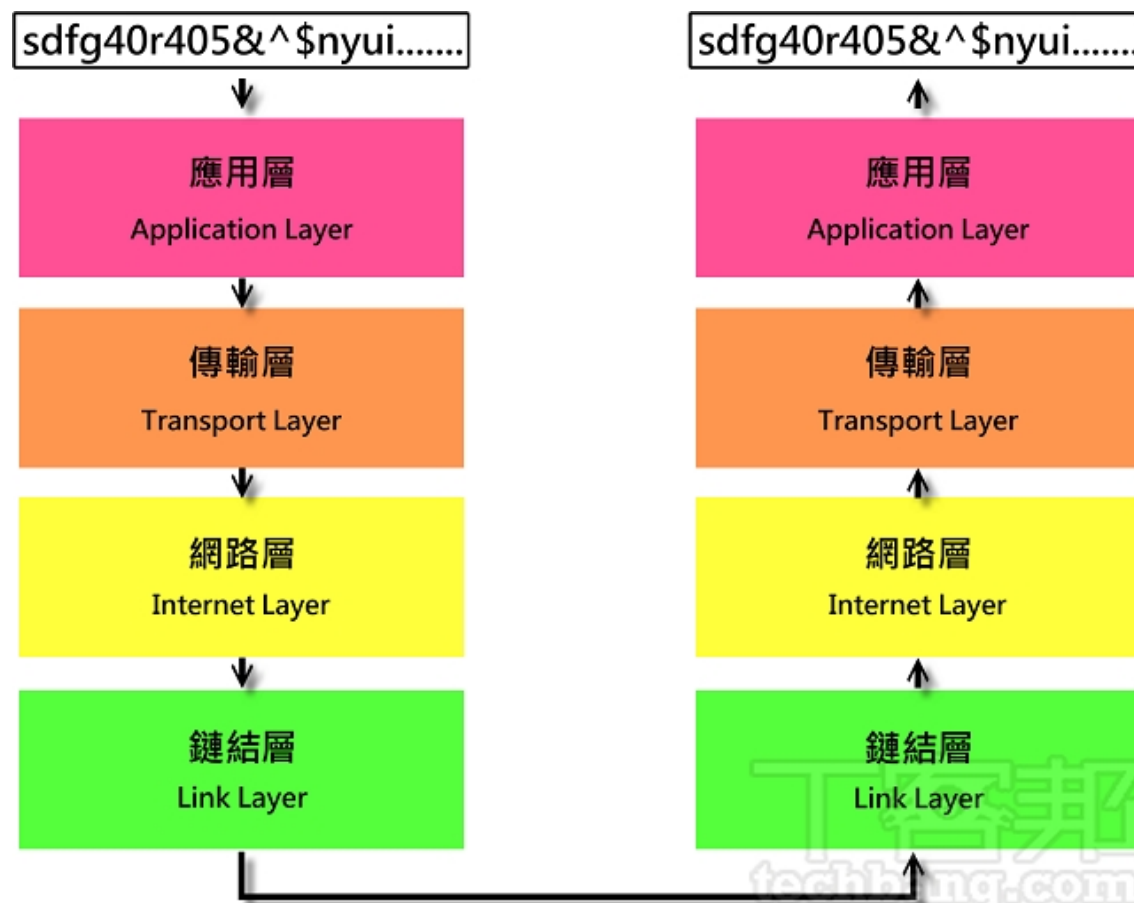
TCP、UDP、RTP、SCTP 如何有效傳輸

網際網路協議(IP)

有線網路(乙太網路)、無線網路(Wi-Fi)

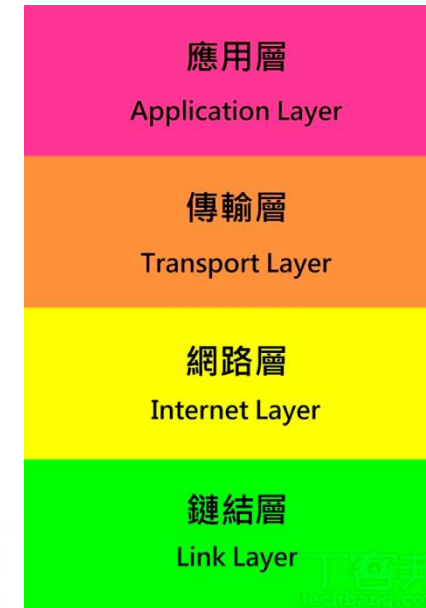


資料經過網路模型的處理方式，先將傳送端的資料一層層打包完畢之後，送到網路上傳送。接收端接受到東西之後，再依相反順序拆開。



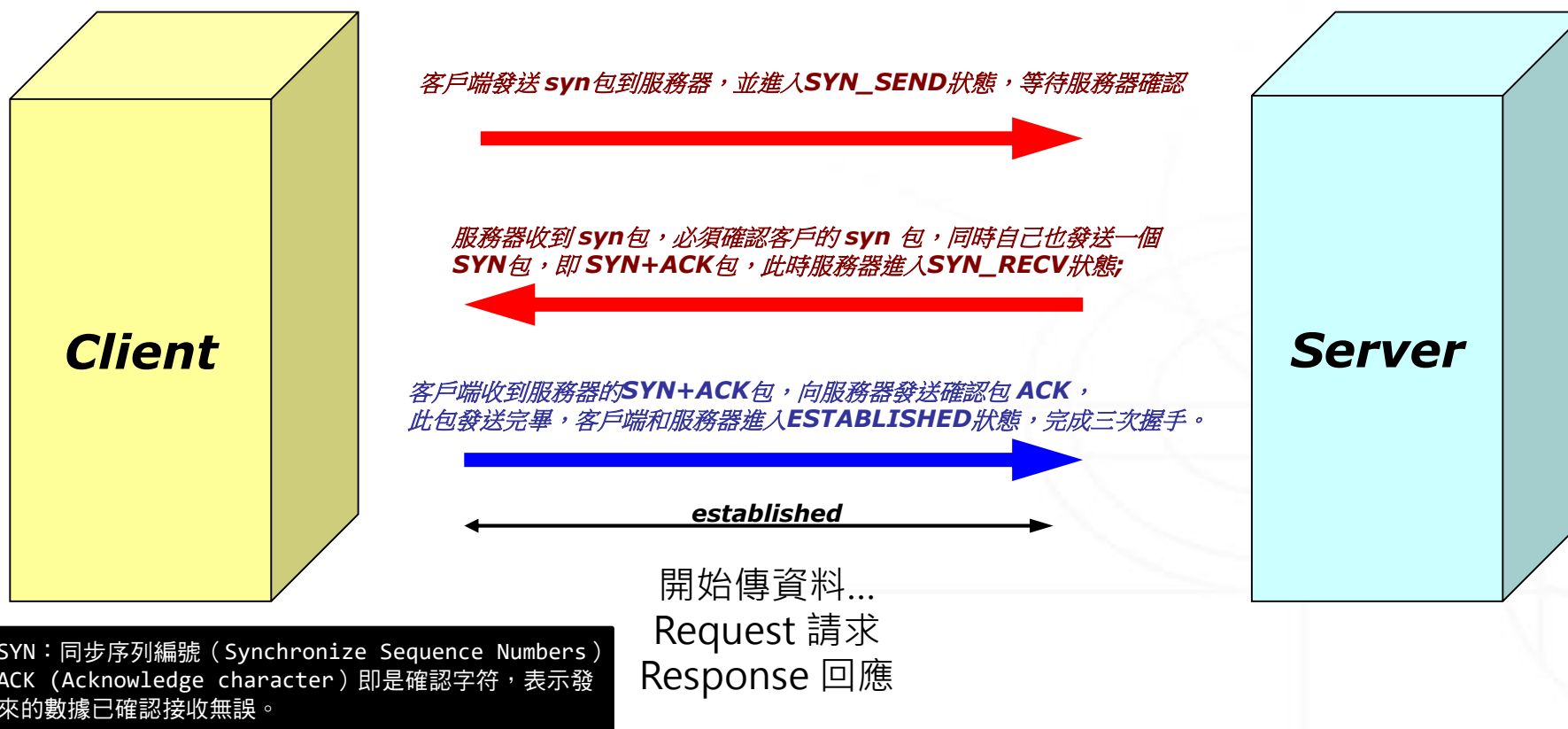
TCP/IP 網路模型

- IP 協議對應於網路層
- TCP 協議對應於傳輸層
- HTTP 協議對應於應用層
- 三者從本質上來說是完全不同。
 - 也可以說，TCP/IP協議是傳輸層協議，主要解決數據如何在網絡中傳輸，
- 而HTTP是應用層協議，主要解決如何包裝數據。



TCP/IP 網路模型

TCP 三向交握 (Three-way Handshake)



綱 要

- 一、Web Application
- 二、TCP/IP 網路模型
- 三、HTTP 協定

HTTP 協定

- HTTP

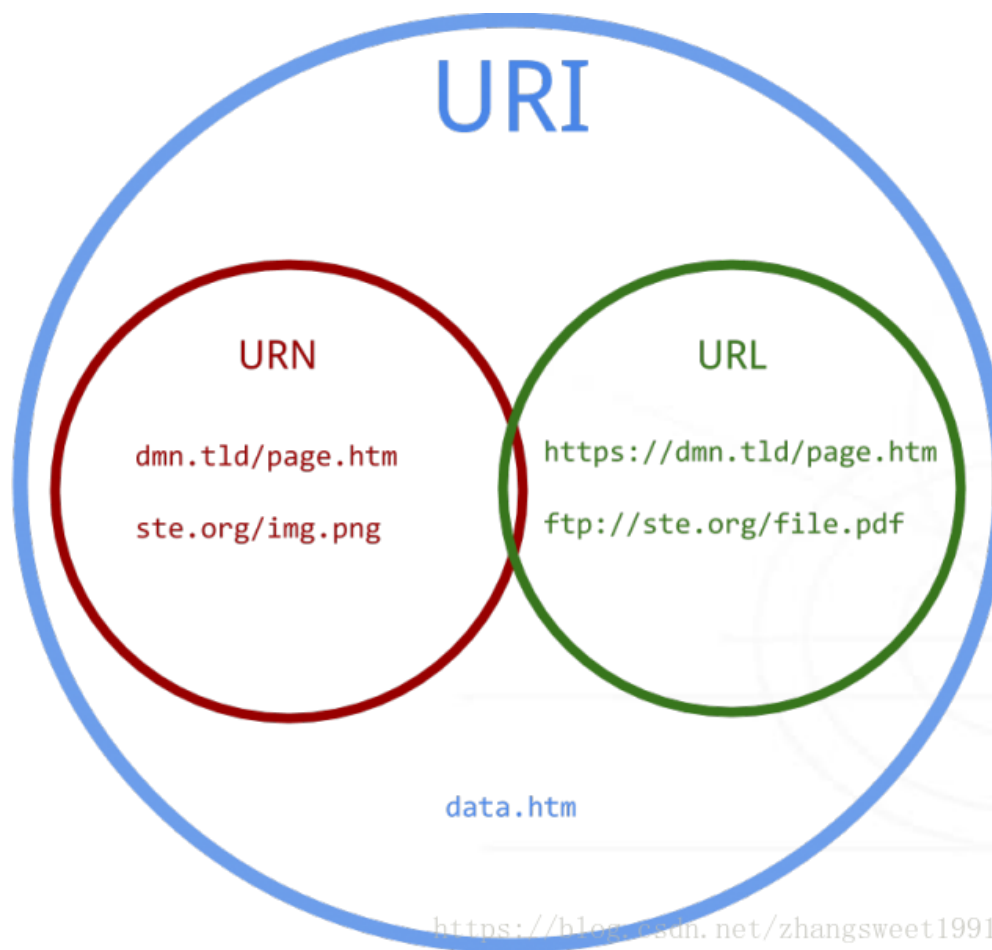
- Hypertext Transfer Protocol，是一種單純的要求與回應(request-response)服務且沒有連線狀態(stateless)的協定。

- Client 端與 Server 端在分析了 HTTP 訊息後會給予適當的服務(您也可以把 HTTP 想成是一種近似於 FTP 的服務，在 Client 與 Server 之間可以用來傳送檔案與資料)。

HTTP 協定

- HTTP 上 3 種常見的資源指定路徑：
 - URI : Uniform Resource Identifier
 - URL : Uniform Resource Locator
 - URN : Uniform Resource Name

每個 URL 都是 URI，但不一定每個 URI 都是 URL



URI

- Universal Resource Identifier 統一資源標誌符
 - 在某一規則下能把一個資源獨一無二標示出來

```
https://developer.mozilla.org/en-US/docs/Learn  
tel:+1-816-555-1212  
git@github.com:mdn/browser-compat-data.git  
ftp://example.org/resource.txt  
urn:isbn:9780141036144
```

URL

- Universal Resource Locator 統一資源定位符
 - 類似一個人的住址
 - 標識一個網際網路資源，並指定對其進行操作或取得該資源的方法

```
https://developer.mozilla.org  
https://developer.mozilla.org/en-US/docs/Learn/  
https://developer.mozilla.org/en-US/search?q=URL
```

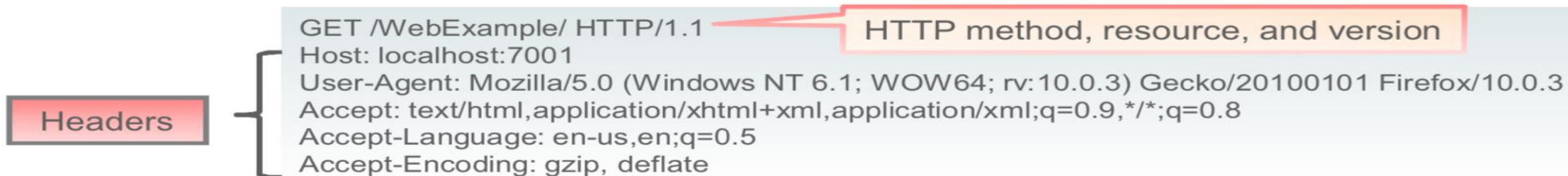
URN

- Universal Resource Name 統一資源名稱
 - 類似一個人的名字
 - 基於某命名空間通過名稱指定資源的 URI
 - 人們可以通過 URN 來指出某個資源，而無需指出其位置和獲得方式

```
urn:isbn:9780141036144  
urn:ietf:rfc:7230
```

HTTP Request-Response Model

HTTP Request



HTTP Response



HTTP 協定

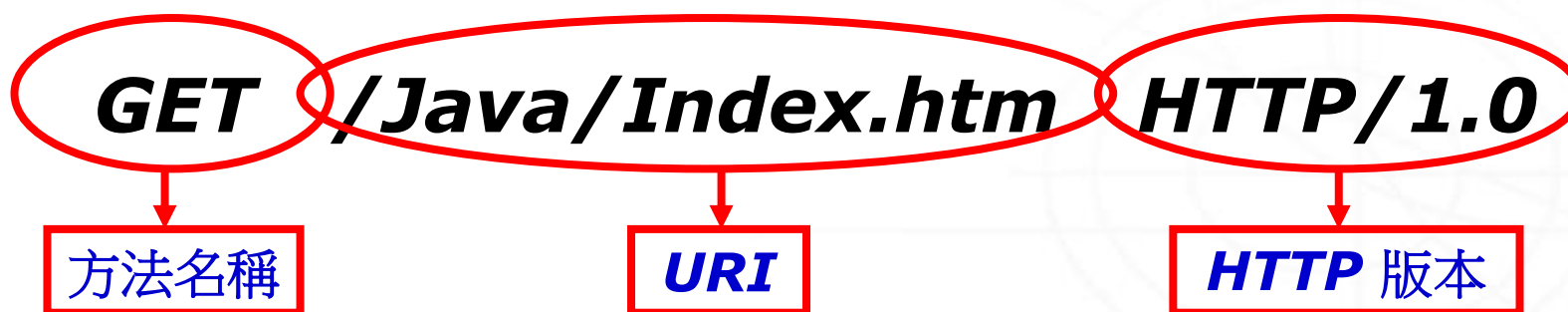
HTTP message 組成因子：

Message part	Description
<i>The initial(state) line</i>	裡面包含了 HTTP 的方法，URI，HTTP 版本與初始狀態
<i>The headers section</i>	根據 HTTP 的方法所包含的標頭資訊，meta：User-Agent、Content-Type 或 ContentLength 等 ...
<i>A blank line</i>	<i>Blank Line</i> 你可以當作是 Headers 資料的結束或者是 Headers 與 Message body 的分隔，例如當你要利用 GET 或 POST 的方法來送出其他的資料時，這些額外的資料就會包裝在 Message body 的區域而 Blank Line 就可以來區隔 Headers 與 Message body 的資料。
<i>An message body</i>	存放 Request 或 Response 的內容，例如：Request 中的參數或者是 Response 中的 HTML Tag。

HTTP 協定

Initial(state) Line

1. 方法名稱
2. *URI*
3. *HTTP* 版本



HTTP Methods

HTTP 1.1 methods are outlined by RFC 2616.

Method	Purpose
GET	Read, possibly cached
POST	Update or create without a known ID
PUT	Update or create with a known ID
DELETE	Remove
HEAD	Read headers; has the version changed?
OPTIONS	List the “Allow”ed methods

HTTP 協定

- 以下我們簡單的從 *Client* 來發出一個請求，來說明 *HTTP* 訊息內容：

Client 利用 *URL* 發出請求(*Request*)：

```
http://localhost:8080/demo/hello.jsp?examName=SCWCD
```

- *examName=SCWCD* 是 *QueryString (name-value)*
 - *QueryString* 就是查詢字串，他是位於 *URL* 末端，廣義的說他也是 *URL* 的一部份，一樣要被瀏覽器與 *Web Server* 解釋，”?” 代表 *URL* 主體與 *QueryString* 的分隔，所以在 ”?” 後面就可以代入參數名稱與參數內容，若有多個參數時則可以利用 ”&” 加以區隔。
 - `hello.jsp?examName=SCWCD&version=7&score=100`

HTTP 協定

Server 端所得到的 HTTP 訊息如下：

<i>initial(state) line</i>	<i>GET /demo/hello.jsp?examName=SCWCD HTTP/1.0</i>
<i>Headers</i>	<i>accept : image/gif, , application/x-shockwave ... accept-language : zh-tw user-agent : Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) host : localhost:8080 connection : Keep-Alive pragma : no-cache</i>
<i>Blank line</i>	
<i>Message body</i>	

HTTP 協定

Server 端處理完 *Client* 端的請求(*Request*) 之後將會回應(*Response*)給 *Client* 端，而 *Client* 端最後所得到的 *HTTP* 訊息如下：

<i>Initial line</i>	HTTP/1.0 200 OK
<i>Headers</i>	Date : Tuesday, 10-Feb-04 15:31:59 GMT Server : Tomcat Web Server / 5.0.18 MIME-version : 1.0 Content-type : text/html Content-length : 20
<i>Blank line</i>	
<i>Message body</i>	<html> SCWCD </html>

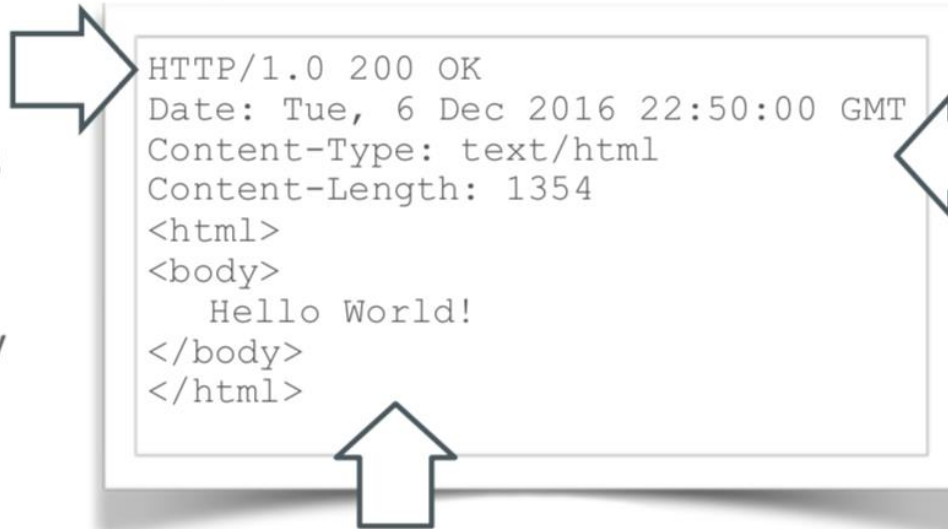
HTTP Status Codes

- 1xx – Informational
 - Request received, continuing process
- 2xx – Success
 - Action successfully received, understood, and accepted
- 3xx – Redirection
 - Client must take additional action to complete the request.
- 4xx – Client Error
 - Request contains bad syntax or cannot be fulfilled.
- 5xx – Server Error
 - Server failed to fulfill an apparently valid request.

HTTP Protocol Basics: Getting Responses

Status code that indicates the nature of the response to the client

- 1xx codes: Informational
Example: 101 Switching Protocols
- 2xx codes: Successful Responses
Example: 200 OK
- 3xx codes – Redirection
Example: 301 Moved Permanently
- 4xx codes: Client Errors
Example: 404 Not Found
- 5xx codes: Server Errors
Example: 503 Service Unavailable



```
HTTP/1.0 200 OK
Date: Tue, 6 Dec 2016 22:50:00 GMT
Content-Type: text/html
Content-Length: 1354
<html>
<body>
  Hello World!
</body>
</html>
```

Headers that describe metadata

- Context Type
- Content Length
- Character Encoding
- Date and Time
- And so on

Optionally Body of the Response

Example: Response contains 200 OK code and body will be dependent on the method used in the request.

- GET: Response body should contain requested resource content.
- HEAD: No response body is produced, just headers.
- POST: Response body should contain result of requested action.
- TRACE: Response body should echo the request message back.

Get v.s Post

- **Get**

- `/demo/Hello.jsp?examName=SCWCD`

- **Form-Data 數據限制 8K**

- **Post**

- `/demo/Hello.jsp`

- **夾帶數據 1 examName=SCWCD**

- **夾帶數據 2 ...**

- **夾帶數據 n ...**

- **Form-Data 數據限制 2G**

GET and POST Requests

	GET Request	POST Request
Type of Use	Default	Form submission
Method of Sending Form Data	<ul style="list-style-type: none">• Sent with the URI• Size limited (8K)	<ul style="list-style-type: none">• Sent in the request body• Size unlimited (2G)
Benefits and Drawbacks	<ul style="list-style-type: none">• Form data IS viewable in the browser's address bar• Form can be resubmitted with a bookmark.	<ul style="list-style-type: none">• Form data is not displayed in the browser's address bar.• Form cannot be resubmitted with a bookmark.

HTTP 協定

Initial line	Get /demo/hello.jsp?examName=SCWCD HTTP/1.0
Headers	accept : image/gif, , application/x-shockwave ... accept-language : zh-tw 略...
Blank line	
Message body	

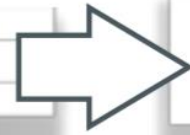
Initial line	POST /demo/hello.jsp HTTP/1.0
Headers	accept : image/gif, , application/x-shockwave ... accept-language : zh-tw 略...
Blank line	
Message body	examName=SCWCD

HTTP Protocol Basics: Sending Requests

HTTP GET method:

- Transmits parameters together with the URL
- Typically is used to request initial pages where parameters are not required

```
http://www.example.com/demos/something?p1=A&p2=B
```



```
GET /demos/something?p1=A&p2=B HTTP/1.1  
Host: www.example.com
```

HTTP POST method:

- Transmits parameters as separate name–value pairs
- Is used to submit user input data

```
<form action="something" method="POST">  
<input type="text" name="p1" value="A">  
  <input type="text" name="p2" value="B">  
  <input type="submit" value="OK">  
</form>
```



```
POST /demos/something HTTP/1.1  
Host: www.example.com  
p1=A&p2=B
```