Distributed Systems

Chun-Feng Liao

廖峻鋒

Department of Computer Science

National Chengchi University

Distributed Systems

HTTP 的通訊協定與應用

HTTP and REST

Chun-Feng Liao

廖峻鋒

Dept. of Computer Science

National Chengchi University

Distributed Systems

HTTP：網路通訊的角度看Web

Chun-Feng Liao

廖峻鋒

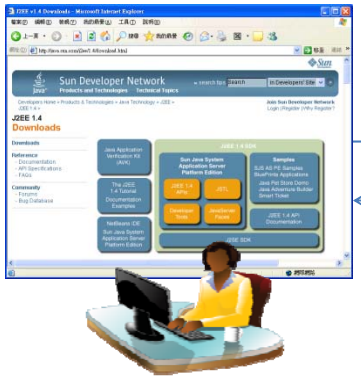
Dept. of Computer Science

National Chengchi University

Web架構

網路上有很多 Server

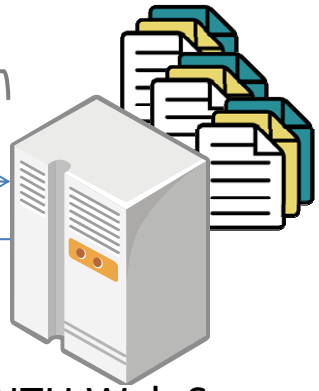
已經進展到 HTML 5
幾乎像一個小的 OS 在網路上



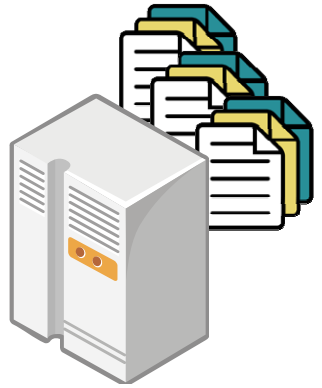
Browser

URL

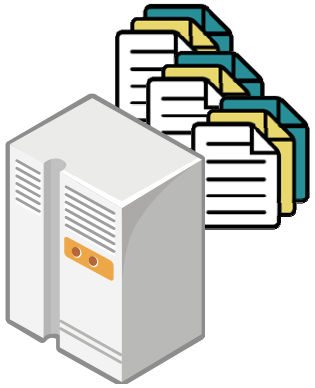
取得 http://www.ntu.edu.tw/index.html



NTU Web Server
http://www.ntu.edu.tw



NCCU Web Server
http://www.nccu.edu.tw



NCTU Web Server
http://www.nctu.edu.tw

Browser如何與Web Server交談以獲取文件的規範稱為HTTP
HTTP本身是建構在既有的TCP/IP網路上

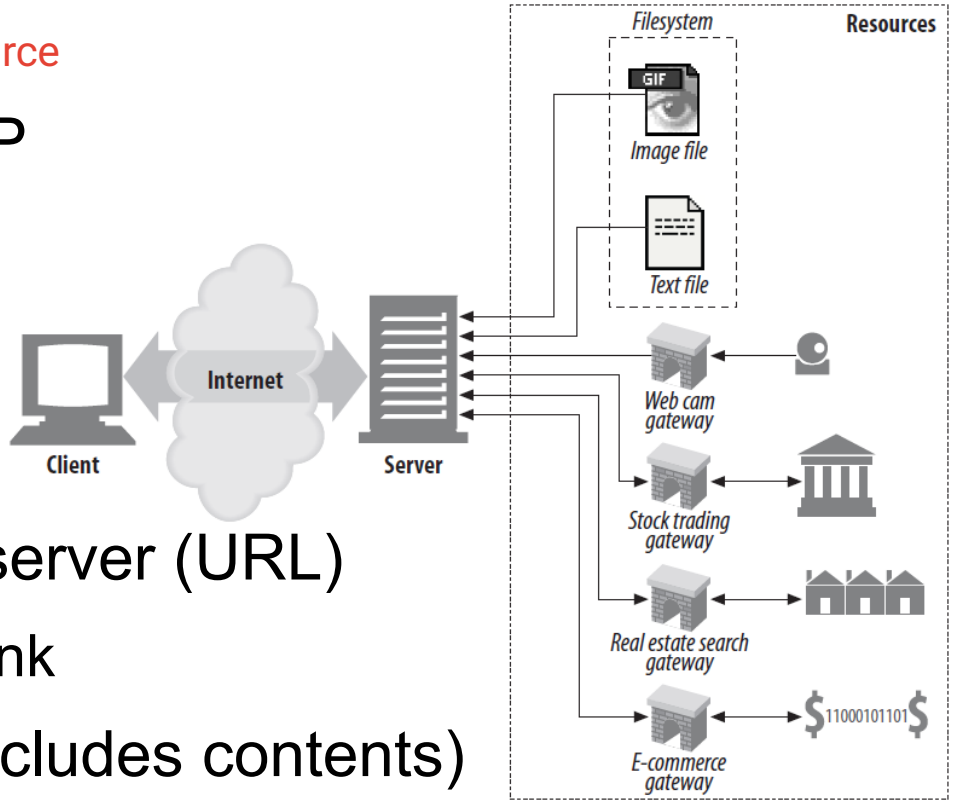
- HTTP
- TCP
- IP
- Ethernet

World Wide Web

- 歷史
  - 1989由Tim Berners-Lee提出
  - 最早主要目的做為學術文件交換
    - 每份文件都可定址 (用於存取的唯一表述格式)
    - 文件內文可透過hyperlink相互參考
- 三大元素
  - 文件定址:URL (Universal Resource Locator) 404 就是查無此頁
  - 文件格式:HTML (HyperText Markup Language)
  - 文件傳送:HTTP (HyperText Transfer Protocol)

Web主要元素

- Resource　Web 裡的東西都是 Resource
  - Provides contents via HTTP
  - Addressable via URL
  - Hosted by a Web server
- Web client and server
  - Client: send requests to a server (URL)
    - May be following a hyperlink
  - Server: send responses (includes contents) back to the client



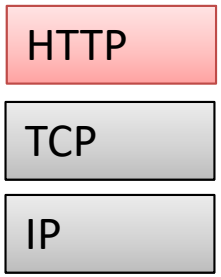
C to S => Request line, Headers, body
S to C 也類似

HTTP

所有東西都收斂進 HTTP，是溝通管道

- Hypertext Transport Protocol
  - Application layer protocol
  - Based on TCP/IP
    - HTTP/3 **將基於QUIC**

HTTP/3 不再用 TCP，而是使用 UDP
https://zh.wikipedia.org/zh-tw/HTTP/3



- Language of the Web
  - Protocol used for communication between Web clients and servers
- Is stateless (指server-side)　S 不會去區分 C 是誰
  - Server does not maintain the state of a session
  - Server**的負擔較輕**　C 能儲存就不要再給 S 去儲存 (因 S 端需要負擔所有的 C)
  - **每次交談時，Client要自帶完整前後文**

URL

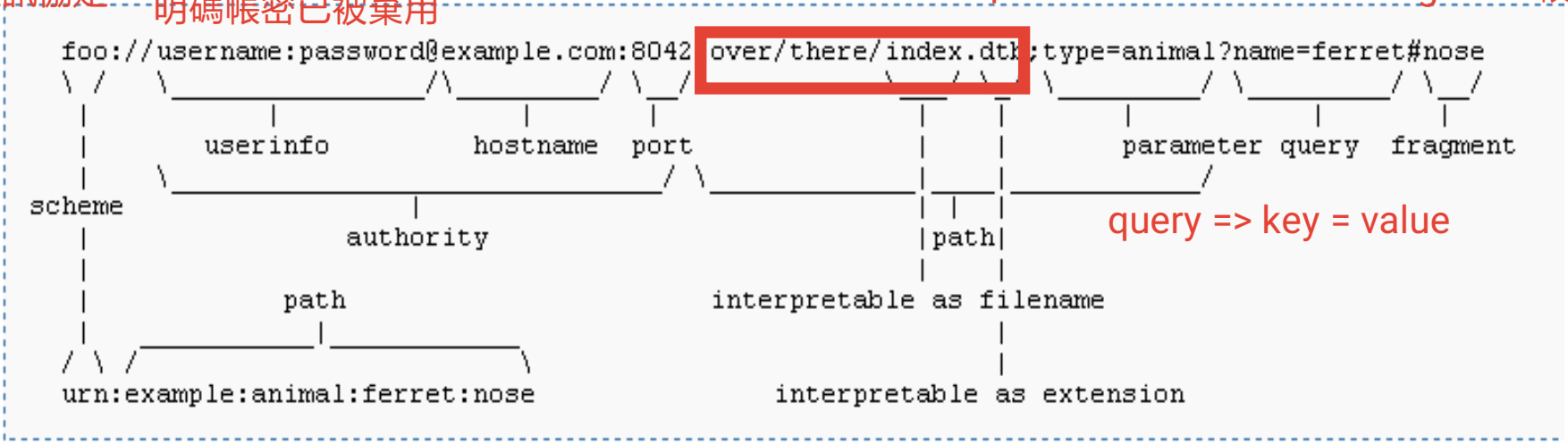
foo => 通訊協定

明碼帳密已被棄用

Path

parameter 較少用

fragment 較少用



query => key = value

URI只用來辨識，不保證在網路上真正找得到

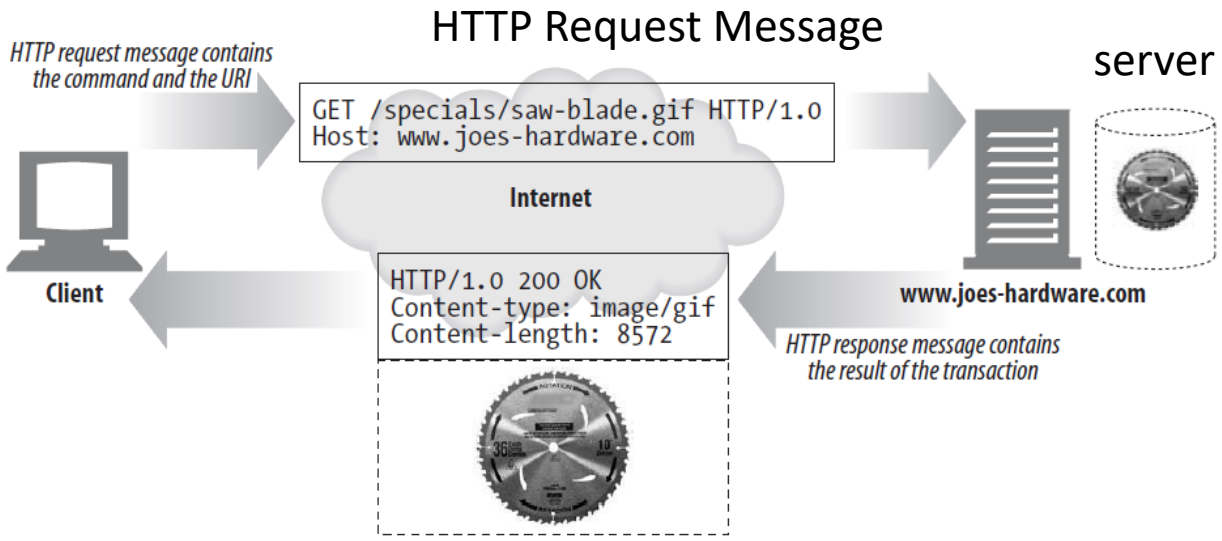
在網路上真正找得到的資源稱為URL (Universal Resource Locator)

Ex:
http://dl.acm.org/citation.cfm?id=1967428.1967434&coll=DL&dl=ACM&CFID=706671792&CFTOKEN=10622052

key value

HTTP Request and Response

- Is a request-response protocol
  - Client→Server 稱為Request
  - Server→Client 稱為Response



HTTP 訊息

- 無論是Request或Response，訊息都包含了
  - Header
  - Body

任何的通訊協定都有這兩者

2 開頭 => 正常
3 開頭 => 轉向
4 開頭 => client 的錯誤
5 開頭 => server 的錯誤
參照 24 頁

Response

```
HTTP/1.1 200 OK
X-Powered-By: Express
Accept-Ranges: bytes
ETag: "149-1456280156295"
Date: Wed, 24 Feb ...
Cache-Control: public, max-age=0
Last-Modified: Wed, 24 Feb ...
Content-Type: text/html; charset=UTF-8
Content-Length: 149
Connection: keep-alive

<!DOCTYPE html><html><head><meta
charset="BIG5"><title>Hello</title></hea
d><body><H1>Hello this is a static
page</H1></body></html>
```

header

body

Request

對哪個 path 做什麼 METHOD

request line

```
METHOD /path-to-resource  HTTP/version-number
Header-Name-1: value
Header-Name-2: value

[  optional request body  ]
```

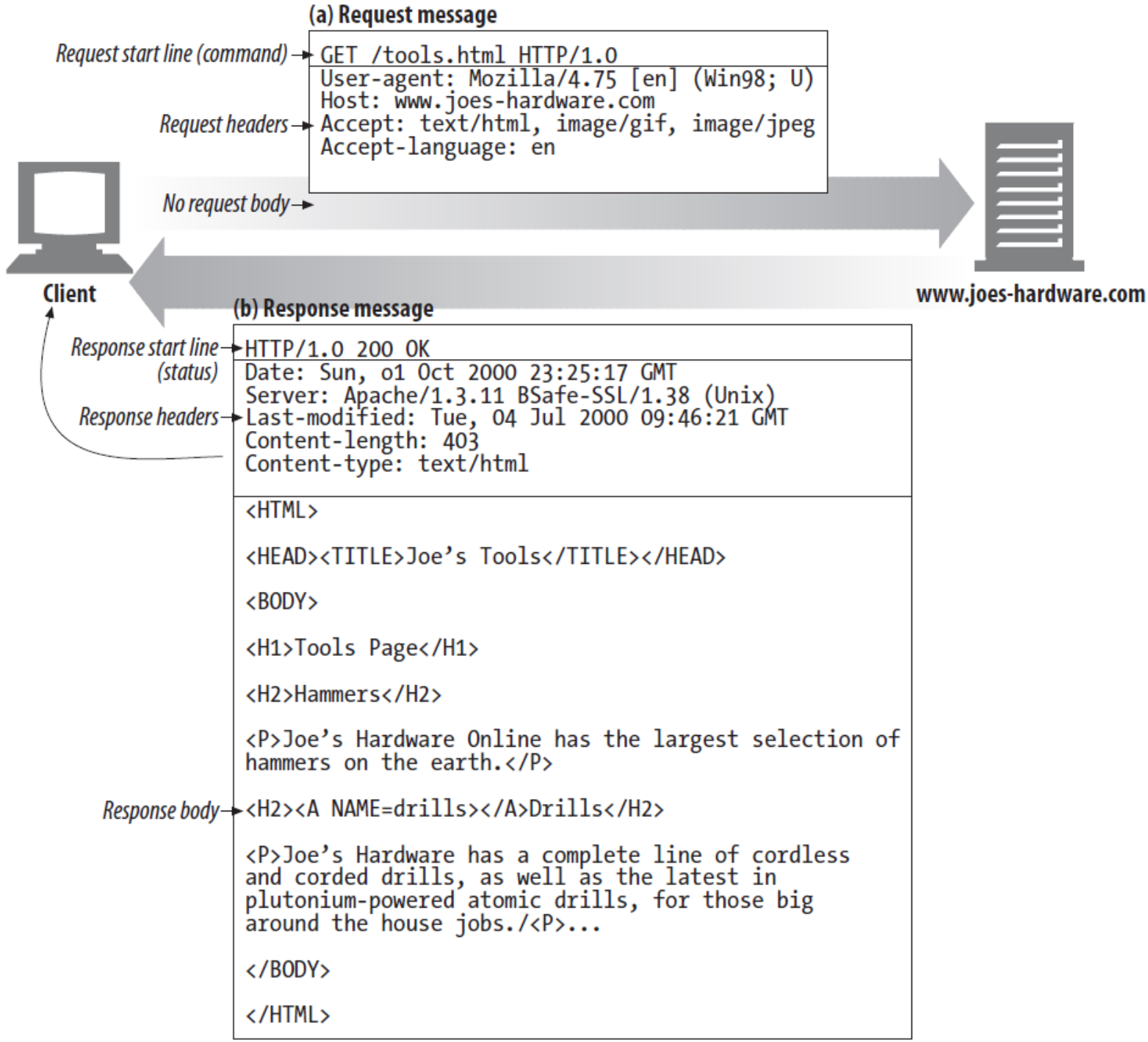
header

pair of key value

body

body 可為任何格式

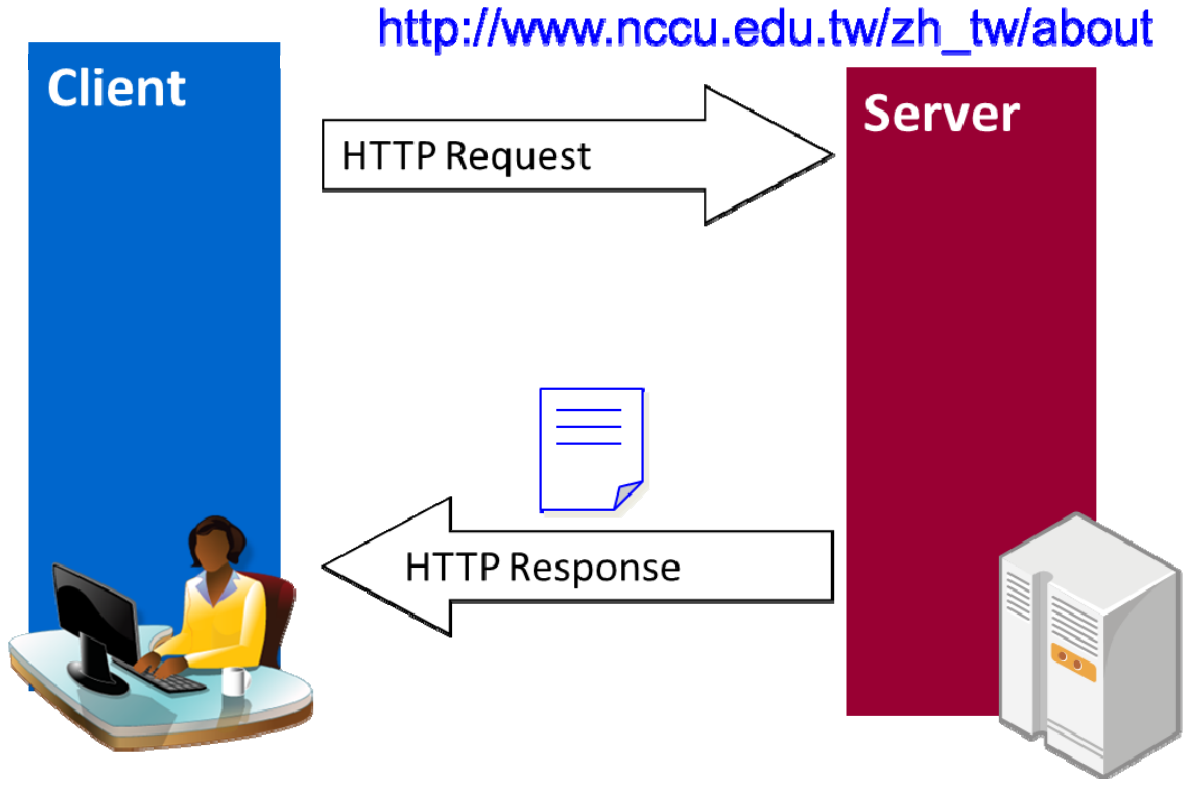
NOTIFY
M-SEARCH


(a) Request message
Request start line (command)
GET /tools.html HTTP/1.0
User-agent: Mozilla/4.75 [en] (Win98; U)
Host: www.joes-hardware.com
Request headers
Accept: text/html, image/gif, image/jpeg
Accept-language: en
No request body
Client
www.joes-hardware.com
(b) Response message
Response start line (status)
HTTP/1.0 200 OK
Response headers
Date: Sun, o1 Oct 2000 23:25:17 GMT
Server: Apache/1.3.11 BSafe-SSL/1.38 (Unix)
Last-modified: Tue, 04 Jul 2000 09:46:21 GMT
Content-length: 403
Content-type: text/html
Response body
<HTML>
<HEAD><TITLE>Joe's Tools</TITLE></HEAD>
<BODY>
<H1>Tools Page</H1>
<H2>Hammers</H2>
<P>Joe's Hardware Online has the largest selection of hammers on the earth.</P>
<H2><A NAME=drills></A>Drills</H2>
<P>Joe's Hardware has a complete line of cordless and corded drills, as well as the latest in plutonium-powered atomic drills, for those big around the house jobs./<P>...
</BODY>
</HTML>

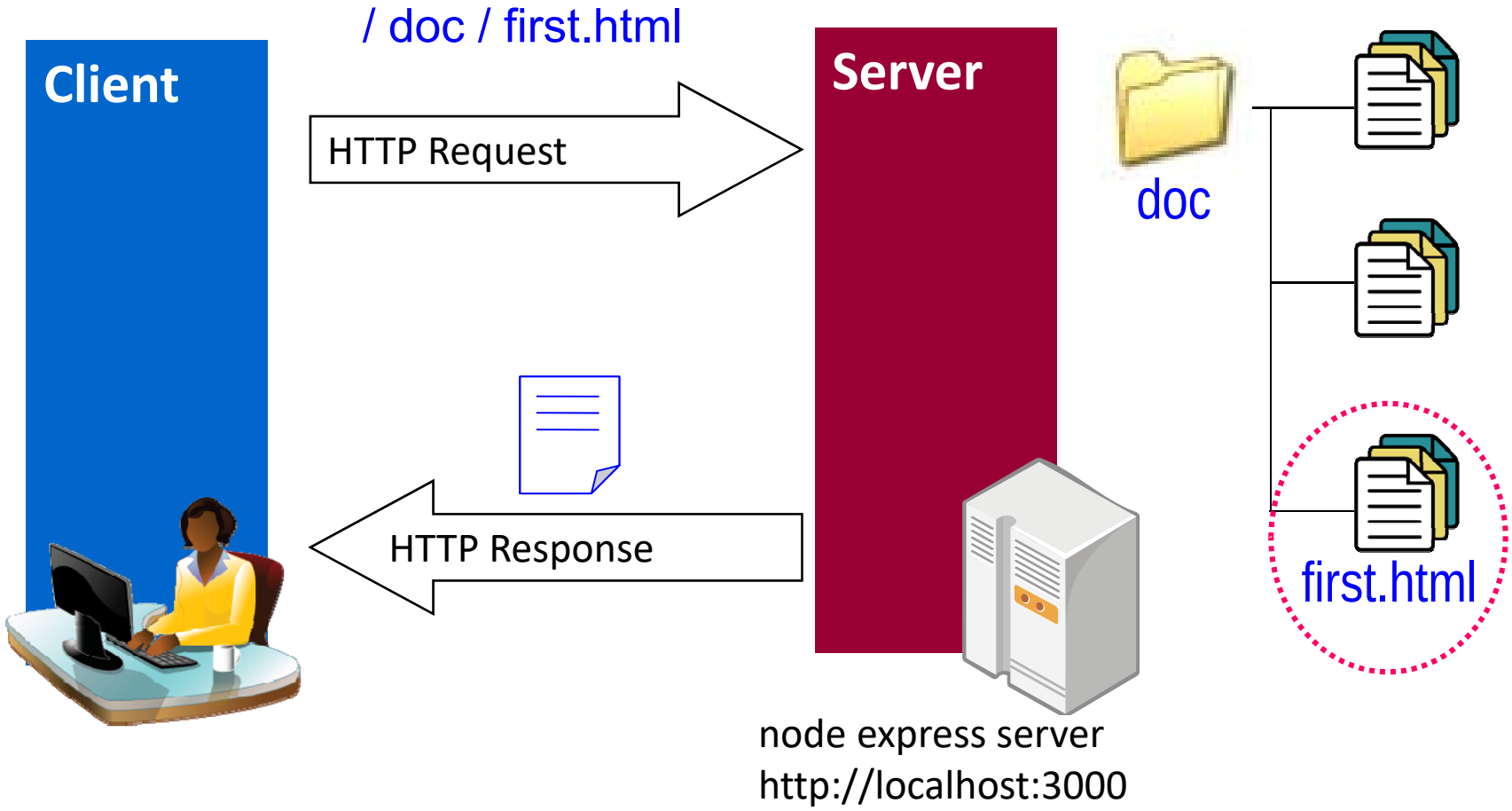
Demo

- 使用curl向www.nccu.edu.tw發出一個 HTTP GET Request

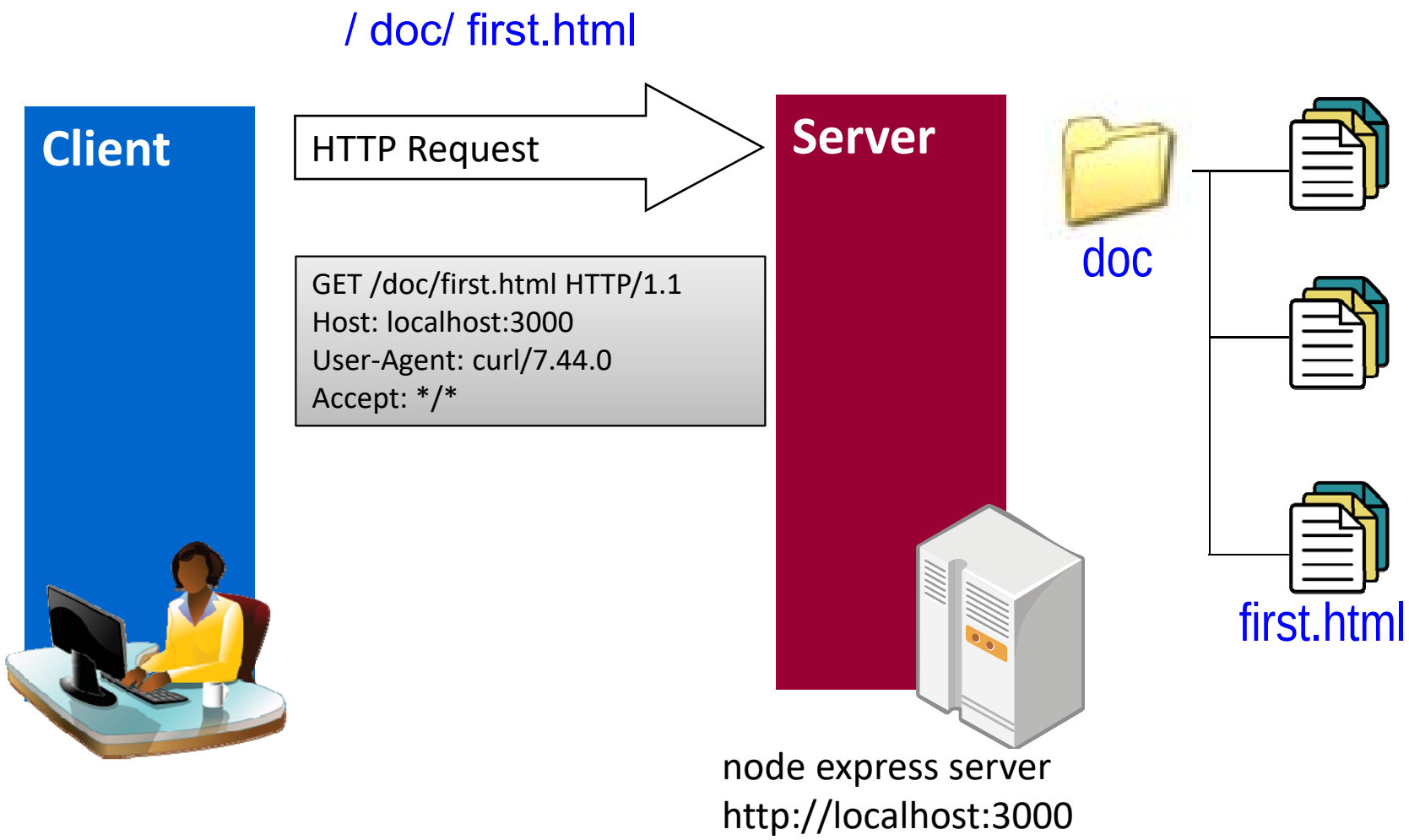
使用postman



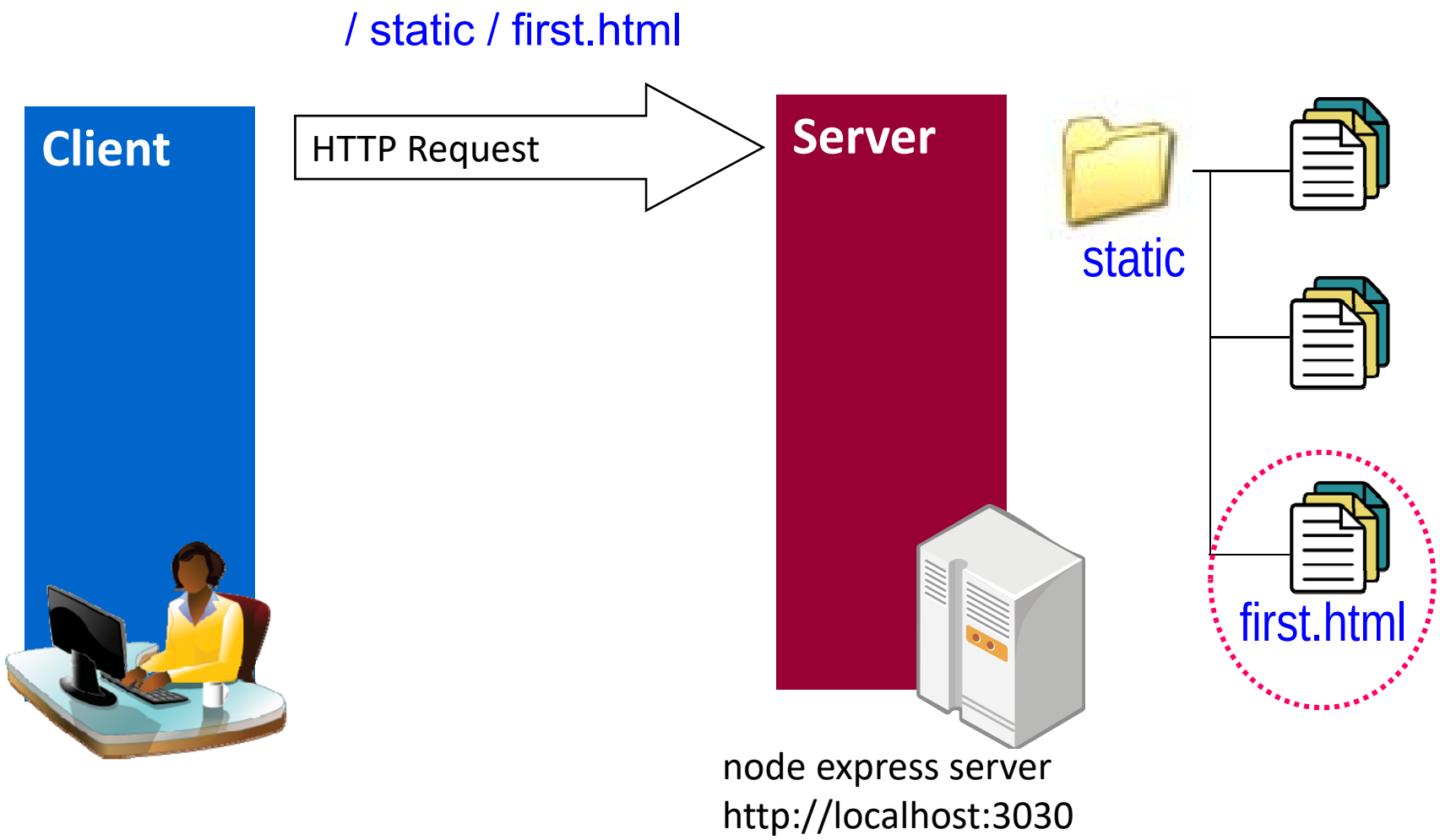
傳送靜態文件


/ doc / first.html
Client
HTTP Request
Server
doc
HTTP Response
first.html
node express server
http://localhost:3000

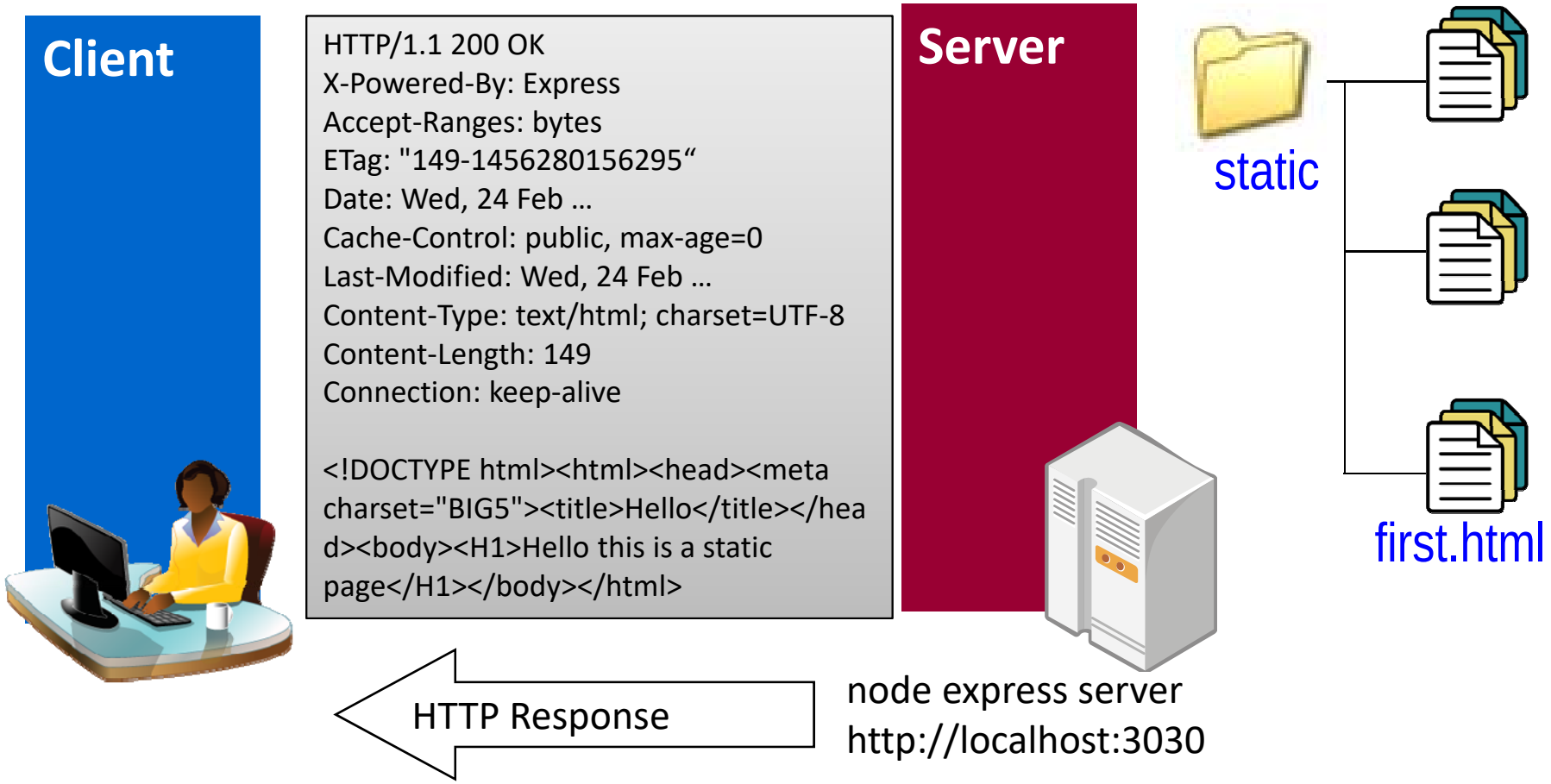
傳送靜態文件


/ doc/ first.html
Client
HTTP Request
GET /doc/first.html HTTP/1.1
Host: localhost:3000
User-Agent: curl/7.44.0
Accept: */*
Server
doc
first.html
node express server
http://localhost:3000

傳送靜態文件


/ static / first.html
Client
HTTP Request
Server
static
first.html
node express server
http://localhost:3030

傳送靜態文件


Client
HTTP/1.1 200 OK
X-Powered-By: Express
Accept-Ranges: bytes
ETag: "149-1456280156295“
Date: Wed, 24 Feb …
Cache-Control: public, max-age=0
Last-Modified: Wed, 24 Feb …
Content-Type: text/html; charset=UTF-8
Content-Length: 149
Connection: keep-alive
<!DOCTYPE html><html><head><meta charset="BIG5"><title>Hello</title></head><body><H1>Hello this is a static page</H1></body></html>
Server
static
first.html
HTTP Response
node express server
http://localhost:3030

Demo: HTTP Server寫作

```
// 取得express模組
const express = require('express');

//建立一個express application
const httpServer = express();

//指定當前目錄下的/doc為根目錄
httpServer.use(express.static(__dirname + '/doc'));

//指定8080 port
httpServer.listen(8080);
```

HTTP Request

- Request line
- Request header
- Request body

POST /index.html HTTP/1.0 (request line)

HOST:www.nccu.edu.tw (request header)

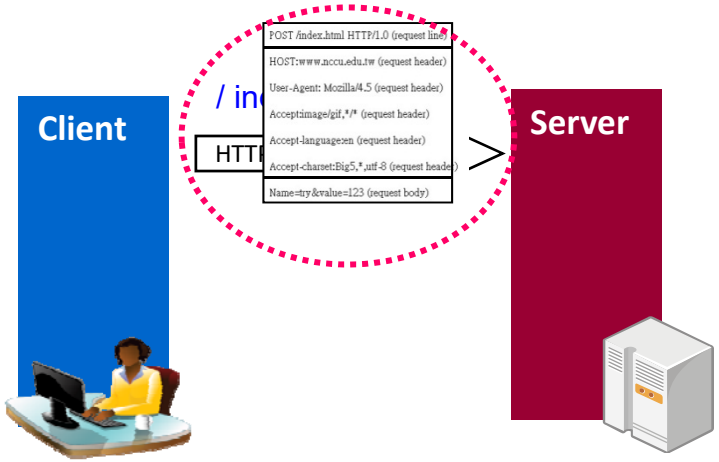
User-Agent: Mozilla/4.5 (request header)

Accept:image/gif,*/* (request header)

Accept-language:en (request header)

Accept-charset:Big5,*,utf-8 (request header)

Name=try&value=123 (request body)



curl –v http://...

Request Header

- 位在本文之前一連串標記
  - 以Key:Value的型式存在
  - 用來標記這個要求的一些屬性
  - server看不懂會自動忽略

Request 訊息範例

POST /index.html HTTP/1.0 (request line)

HOST:www.nccu.edu.tw (request header)

User-Agent: Mozilla/4.5 (request header)

Accept:image/gif,*/* (request header)

Accept-language:en (request header)

Accept-charset:Big5,*,utf-8 (request header)

Name=try&value=123 (request body)

demo

- 取得HTTP Request headers資訊
  - req.headers (陣列)

```
…
httpServer.all('/', (req, resp) => {
    const headers = req.headers;
    for (let headerName in headers) {
        console.log(`${headerName}: ${headers[headerName]}`);
    }
    resp.end();
});

…
```

HTTP Request Methods

- HTTP Request的二種較重要的Request方法
  - GET 從Server「取」文件
  - POST 將資訊「貼」到Server上

POST 將資料傳至 Server (例：表單)

Request 訊息範例

POST /index.html HTTP/1.0 (request line)

HOST:www.nccu.edu.tw (request header)

User-Agent: Mozilla/4.5 (request header)

Accept:image/gif,*/* (request header)

Accept-language:en (request header)

Accept-charset:Big5,*,utf-8 (request header)

Name=try&value=123 (request body)

除了GET / POST之外

- PUT /index.html HTTP/1.1

  Method URL HTTP version

  HEAD: Like GET, but ask that *only* a header be returned

  PUT: Request to store the entity-body at the URI

  DELETE: Request removal of data at the URI

  LINK: Request header information be associated with a document on the server

  UNLINK: Request to undo a LINK request

  OPTIONS: Request information about communications options on the server

  TRACE: Request that the entity-body be returned as received (used for debugging)

使用curl –X (Method_Name)

Response 訊息

HTTP/1.0 200 OK

Last-Modified Mon ,20 Dec 2001 23:26:42 GMT(上次更改)

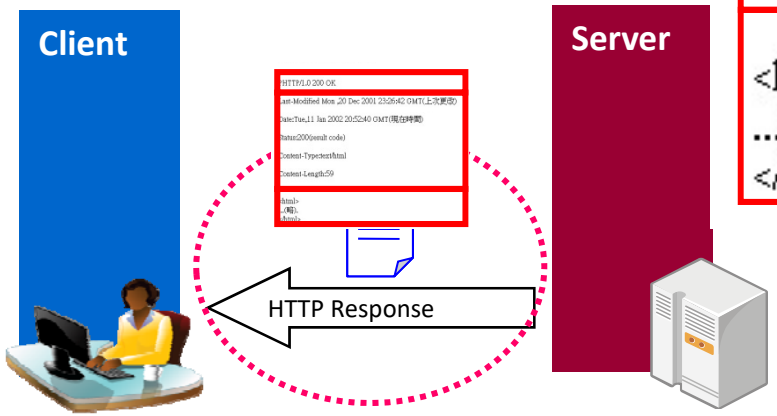
Date:Tue,11 Jan 2002 20:52:40 GMT(現在時間)

Status:200(result code)

Content-Type:text/html

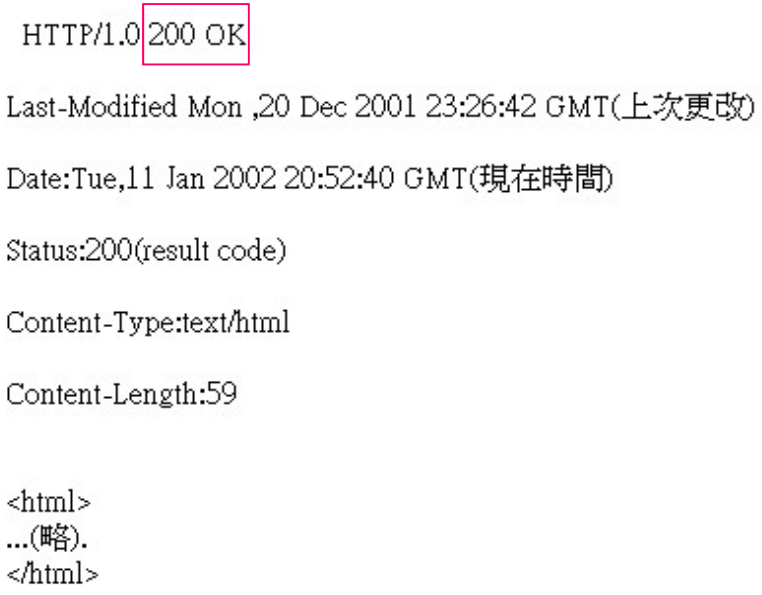
Content-Length:59

<html>
...(略).
</html>



Status Codes – 五大類

- 1XX Informational
- 2XX OK，Server能了解並允許Client的請求。
- 3XX 轉向。
- 4XX Client方面發生的錯誤。
- 5XX Server方面發生的錯誤。



Common status codes

- 200 OK
  - Everything worked, here's the data
- 301 Moved Permanently
  - URI was moved, but here's the new address for your records
- 302 Moved temporarily
  - URL temporarily out of service, keep the old one but use this one for now
- 400 Bad Request
  - There is a xyntax error in your request
- 403 Forbidden
  - You can't do this, and we won't tell you why
- 404 Not Found
  - No such document
- 408 Request Time-out, 504 Gateway Time-out
  - Request took too long to fulfill for some reason

Distributed Systems

REST：應用程式的角度看Web

把 Web 當成應用程式的平台
透過 HTTP 跟 Resource 提要求

Chun-Feng Liao

廖峻鋒

Dept. of Computer Science

National Chengchi University

當Web上不只放網頁…

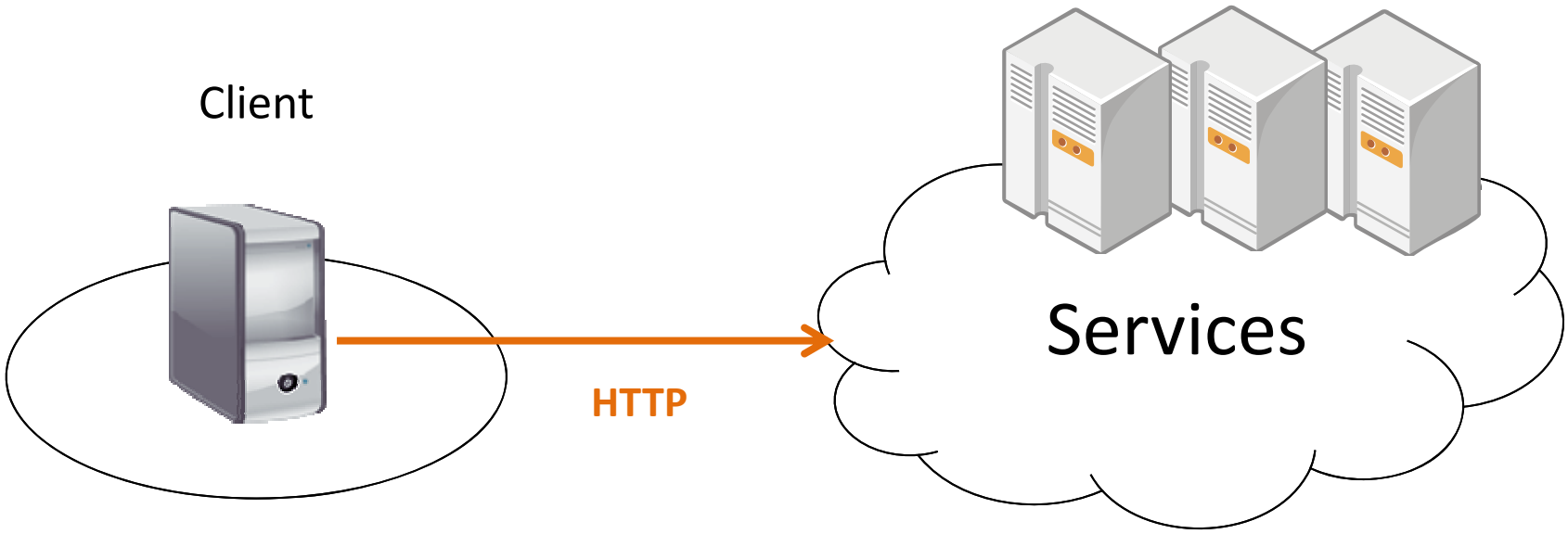
- Idea
  - 將各式計算以Resource形式放上Web，透過HTTP就能存取、運用
- 例如
  - GET /add?x=5&y=3
  - return: {“result”:8}

讓 HTTP 做 Remoting (遠端通訊)

simpleAddService
curl.exe -v http://localhost:3000/add?x=1"&"y=2

Web Service

- 可透過Internet (HTTP)存取的遠端業務邏輯



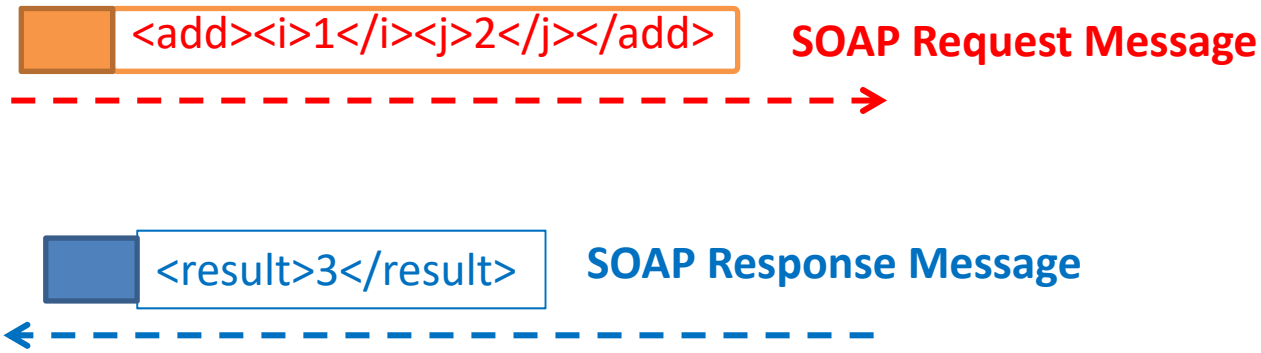
Web Services: 二種主要實現方式

- SOAP
  - 將服務視為遠端函式
  - 依照一定格式將呼叫/回應以XML編碼 (SOAP, Simple Object Access Protocol)
  - 只將HTTP拿來做為訊息運送工具
  - 使用WSDL(Web Service Description Language)描述服務內容
- REST
  - 將HTTP視為應用程式平台，將服務視為物件(資源)
  - 使用HTTP方法(GET/POST/…)操作資源
  - 不限定訊息格式 (XML, JSON或其它)
  - 有多種方式可用來描述服務
    - Swagger (Open API)
    - WADL

SOAP

- Simple Object Access Protocol
- An XML-based communication protocol
  - let applications exchange information over HTTP

型別就讓 user 自己去對



(在TCP/IP層，　　網路傳送訊息單元稱為Packet
在Application層，網路傳送訊息單元稱為Message)

實務做其實很複雜
=> 呼叫遠端服務的問題：型別

SOAP 封包結構


```
<soap:Envelope>
   <soap:Header/>
   <soap:Body>
      <add>
         <i>1</i>
         <j>1</j>
      </add>
   </soap:Body>
</soap:Envelope>
```

```
<soap:Envelope>
   <soap:Header/>
   <soap:Body>
      <addResponse>
         <return>2</return>
      </addResponse>
   </soap:Body>
</soap:Envelope>
```

xmlns:soap="http://www.w3.org/2001/12/soap-envelope"

RESTful Web Services

- R. Fielding在其博士論文提出之架構風格
  - 將HTTP的精神套用到Web Services上
  - “REST is not a standard, but it describes the use of standards”
    - HTTP/ URL/ XML
- 和傳統SOAP/WSDL Web Services各有優勢

SOAP 是強型別的遠端呼叫(強制規定如何傳送)

SOAP 思維 由operation (remote function)出發
REST 思維 由data出發

http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

REST

- Resource
  - typically represents a single “business object” (Domain Object)
  - Ex: Customer, Product
- HTTP verbs
  - GET query a resource
  - POST create, insert or update a resource
  - PUT create or update a resource
- Example
  - GET /orders/3
    - Retrieve the Order instance where order_id = 3

視萬物為 object

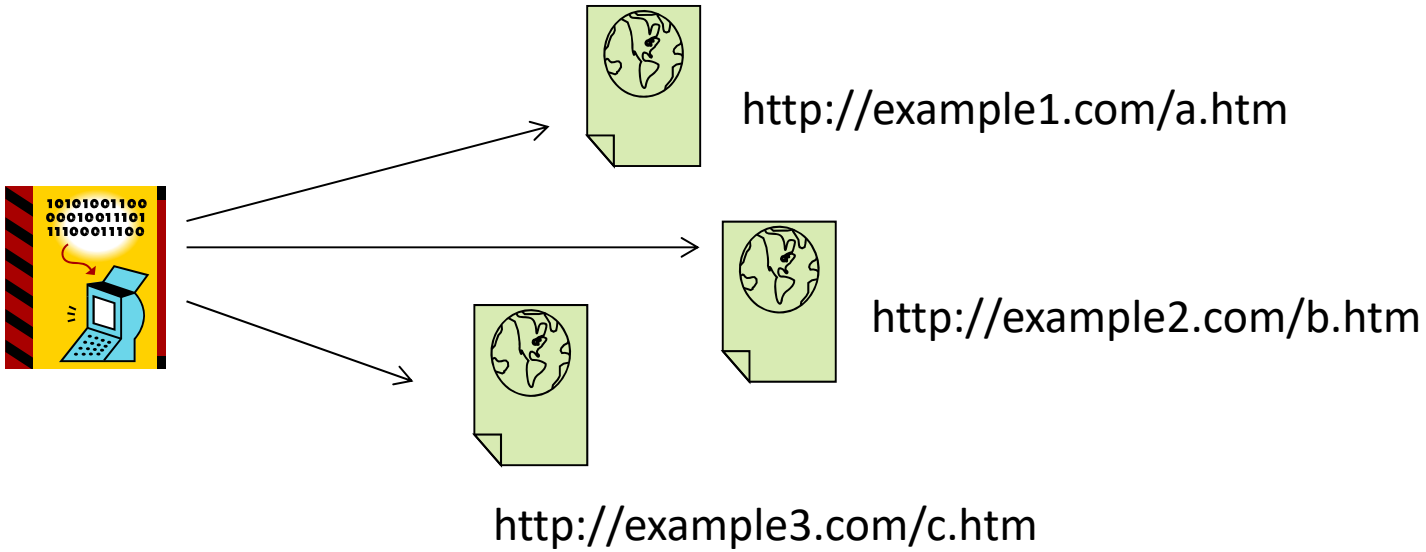
POST, GET, PUT, DEL => 對應 CRUD

RESTful Web Services

REST 風格的 Web Services

- A RESTful web service is a resource meant for a computer to request and process

對 resource 做操作，server 解讀後傳送給 client



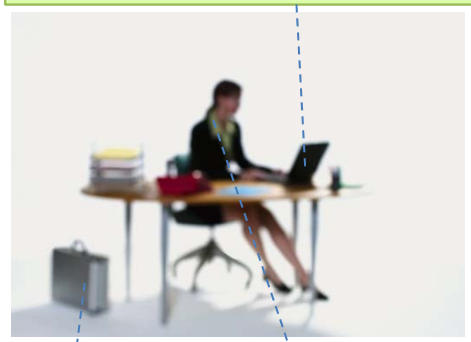
Example

URI 是唯一的
當 URI 能決定位置時 => URL

萬物皆視被為具有URI的資源(Resources)

http://acme.com/dep1/stapler

http://acme.com/dep1/notebook1

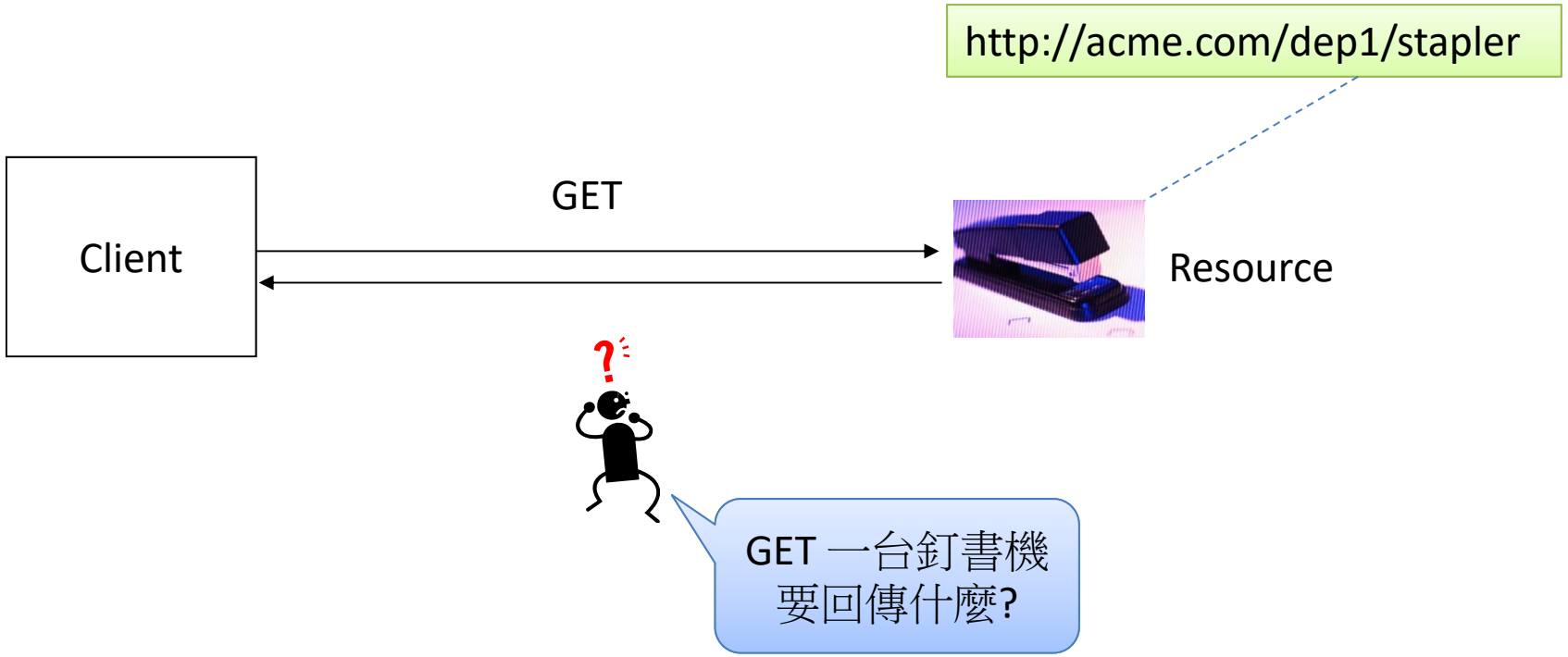


http://acme.com/dep1/phone1

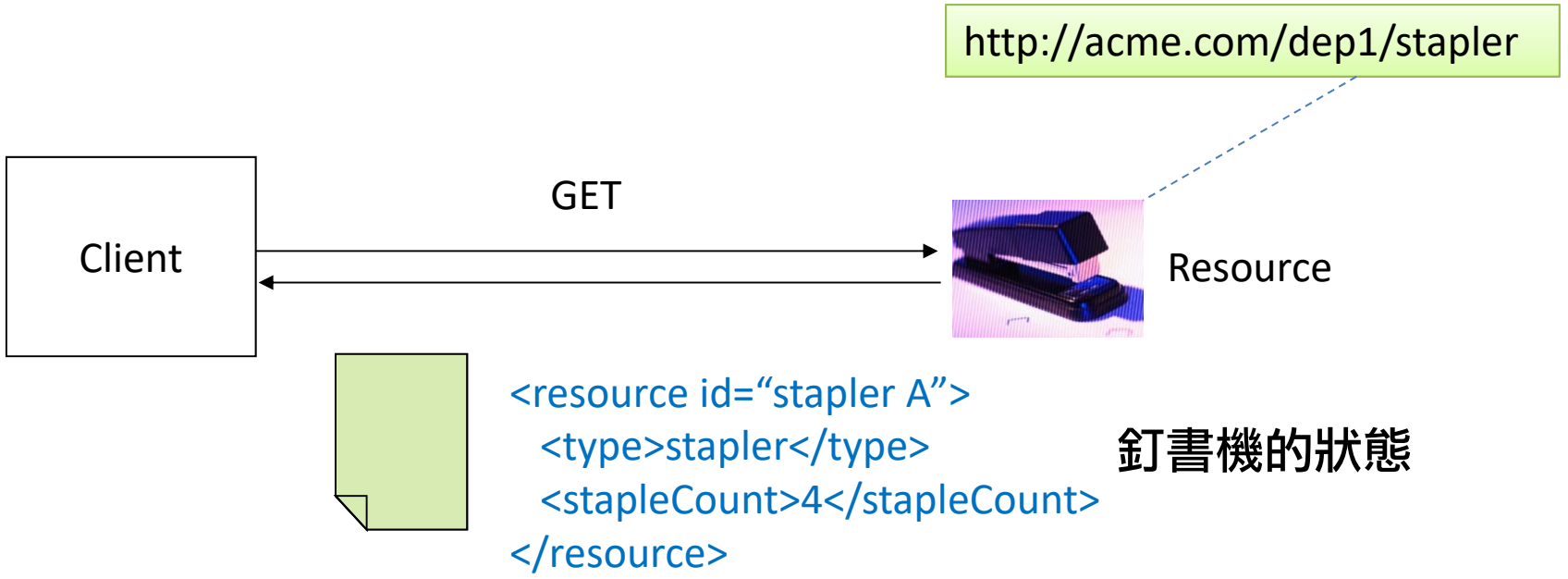
http://acme.com/dep1/person/Helen

http://acme.com/dep1/box1

Example


http://acme.com/dep1/stapler
Client
GET
Resource
GET 一台釘書機
要回傳什麼?

Example



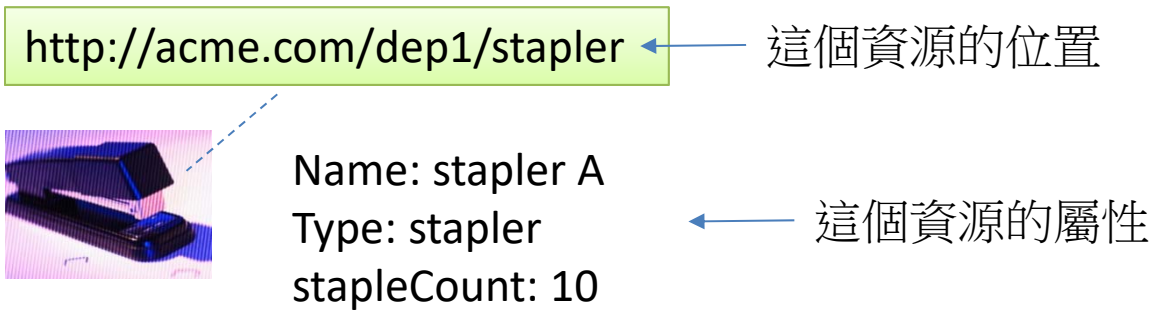
下達GET後可回傳多種表徵(Representation)，例如HTML, XML, JSON, …

透過HTTP Content Negotiation機制

MIME=Multipurpose Internet Mail Extensions

Elements of the Web

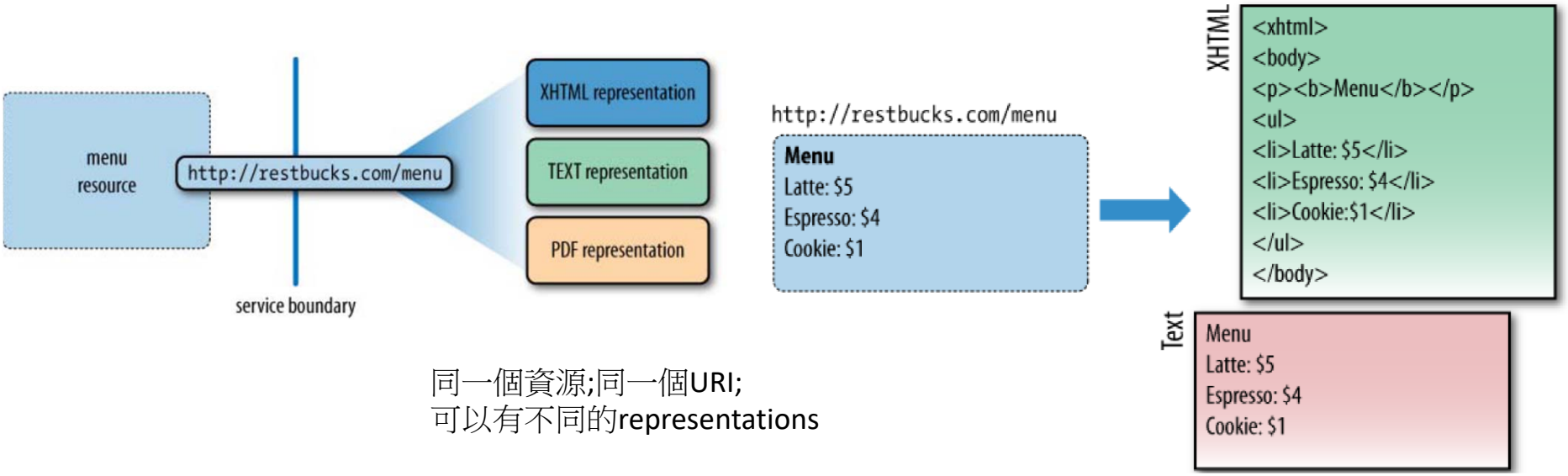
- Resource
  - Anything we exposed to the Web
  - We expose physical/virtual things to the Web by providing an abstraction of them
    - The attributes of the thing



Elements of the Web

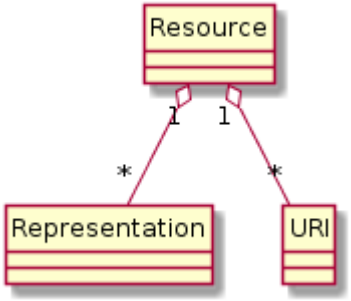
- Representation
  - A view of a resource's state at an instant in time
    - Various formats: HTML, XML, JSON,…
    - Mediated using content negotiation mechanism

      Accept: text/html
  - Web components exchange representations of resources



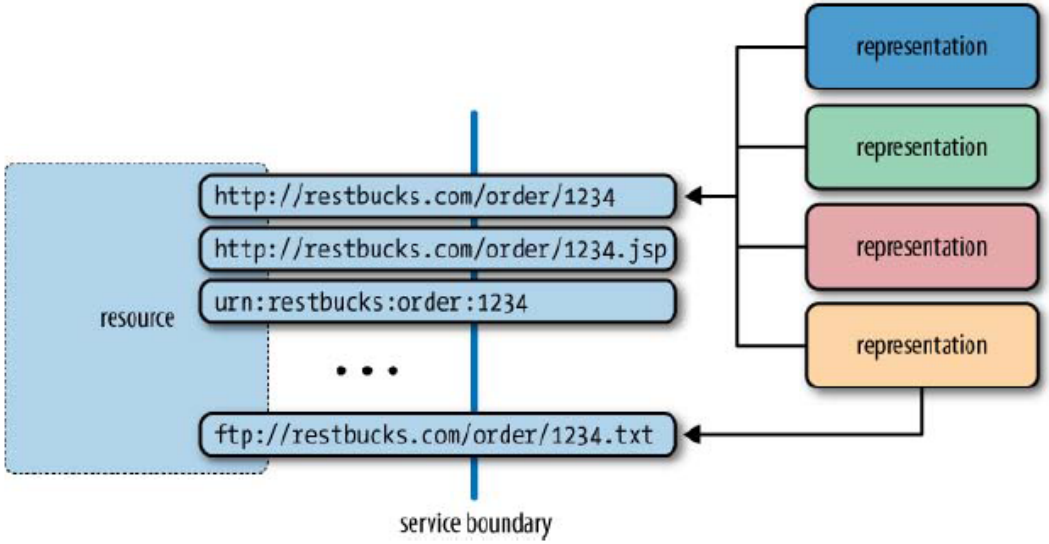
同一個資源;同一個URI;
可以有不同的representations

Elements of the Web



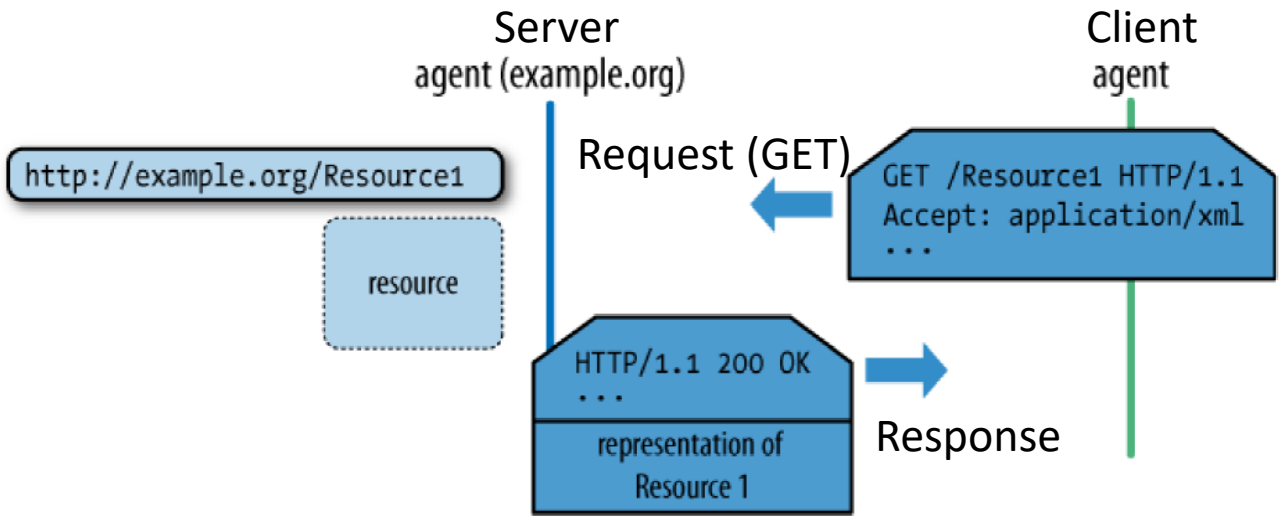
- 一個Resource可被多個URI對應
- 一個Resource可具有多種Representation

(如: html, xml, json…)



Uniform Interface

- 所有Web clients 具有一致的行為
  - HTTP Verbs: GET, POST, PUT, DELETE, …
- 所以resources只要對這些行為做出回應就好
  - On GET, on POST, on PUT, on DELETE…



staplerService

- staplerService.js

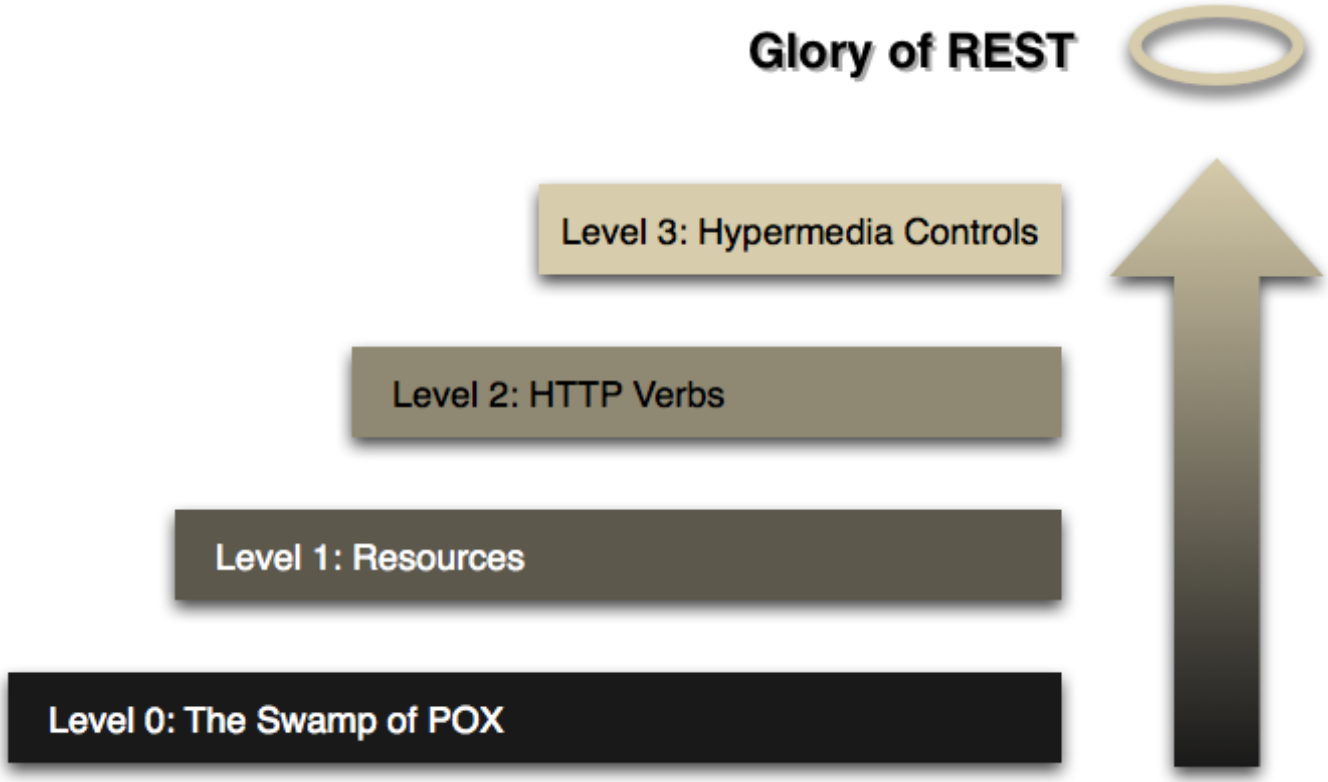
The REST style

- Identification of resources
  - Resource都至少具有一個URI
- Manipulation of resources through representations
- Self-descriptive message
  - 善用HTTP headers
  - 以key-value的型式存放meta-data來自我表述
- Hypermedia as the engine of application state

Web Friendliness

- Richardson Maturity Model

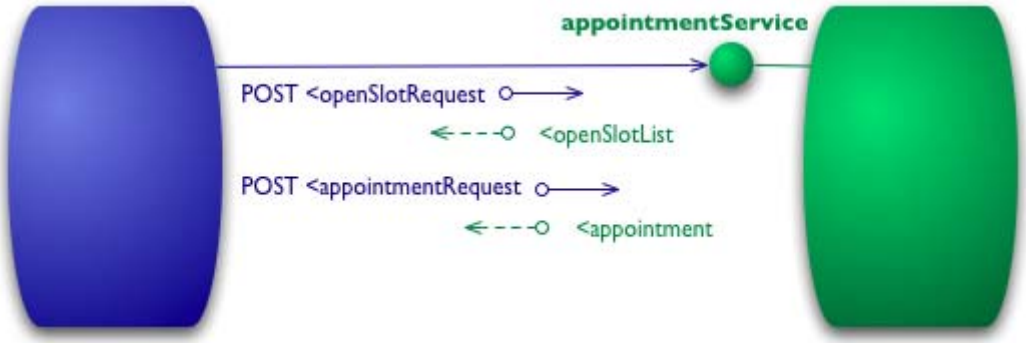
評價 REST



https://martinfowler.com/articles/richardsonMaturityModel.html

Level 0 POX

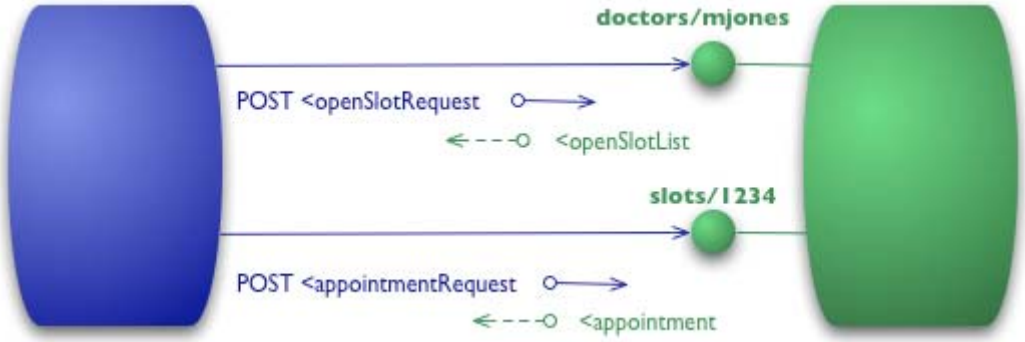
- POX=Plain Old XML
- Using HTTP as a transport system for Remote Procedure Invocation
- Ex: SOAP
  - Uses HTTP POST to transfer payloads
  - Ignoring the semantics of HTTP verbs



Level 1 Resources

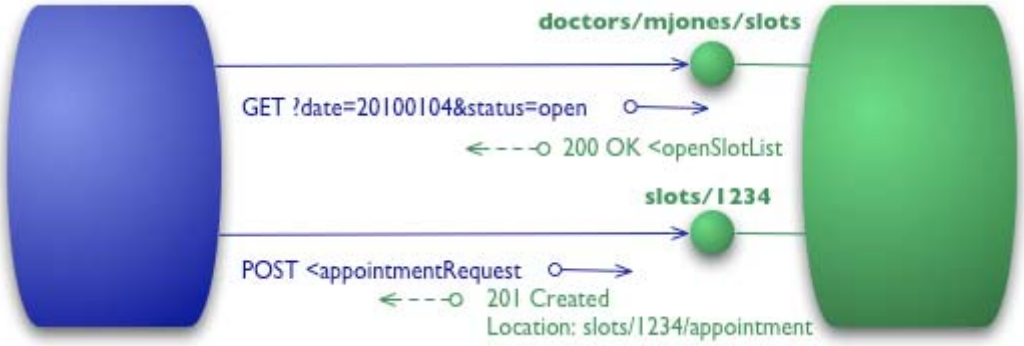
- Abstract remote services (data) as Resources
- Typically only use GET or POST
  - Does not strictly follow the semantics of HTTP verbs

有把東西視為 resource
沒有用到 HTTP



Level 2 HTTP Verbs

- Follows the semantics of HTTP Verbs
  - Using the HTTP verbs as closely as possible to how they are used in HTTP itself
- Follows the convention of HTTP response codes



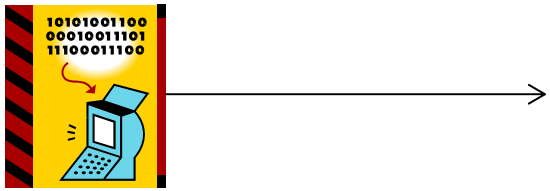
對資源做CRUD

對一個Resource做出POST/GET/PUT/DELETE會代表什麼意義?

Method	CRUD
POST	Create, Insert or Update
GET	Read
PUT	Create or Update
DELETE	Delete

HTTP Methods in the RESTful Architecture

http://nccu.edu.tw/classrooms/200102/sensors/1



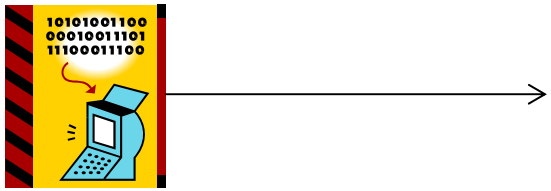
```
{
  id: “1”,
  type: “temperature”,
  value: 28.9,
  unit: celsius
}
```

Method	意義
POST	加入資源
GET	取得資源資料
PUT	更新資源資料
DELETE	刪除資源

GET http://.../200102/sensors/1/type
GET http://.../200102/sensors/1/value
GET http://.../200102/sensors/1/unit

(佈置新感測器)
POST http://.../200102/sensors/2
id=2&type=humidity&value=0&unit=percent

HTTP Methods in the RESTful Architecture



http://nccu.edu.tw/classrooms/200102/aircon/2

```
{
  id: “2”,
  type: “aircon”,
  powerOn:  true,
  fanSpeed: 5
}
```

Method	意義
POST	加入資源
GET	取得資源狀態
PUT	更新資源狀態
DELETE	刪除資源

GET http://.../200102/aircon/2/powerOn

PUT http://.../200102/aircon/2 (關電源)

powerOn=false

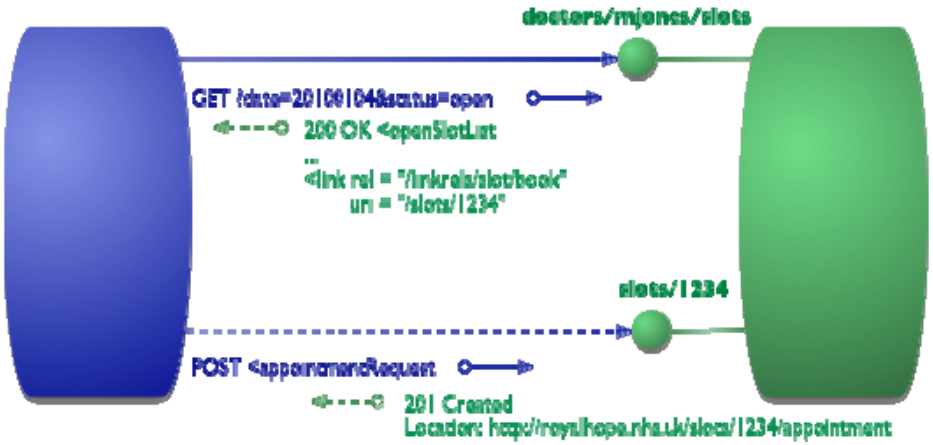
PUT http://.../200102/aircon/2 (調風扇)

fanSpeed=3

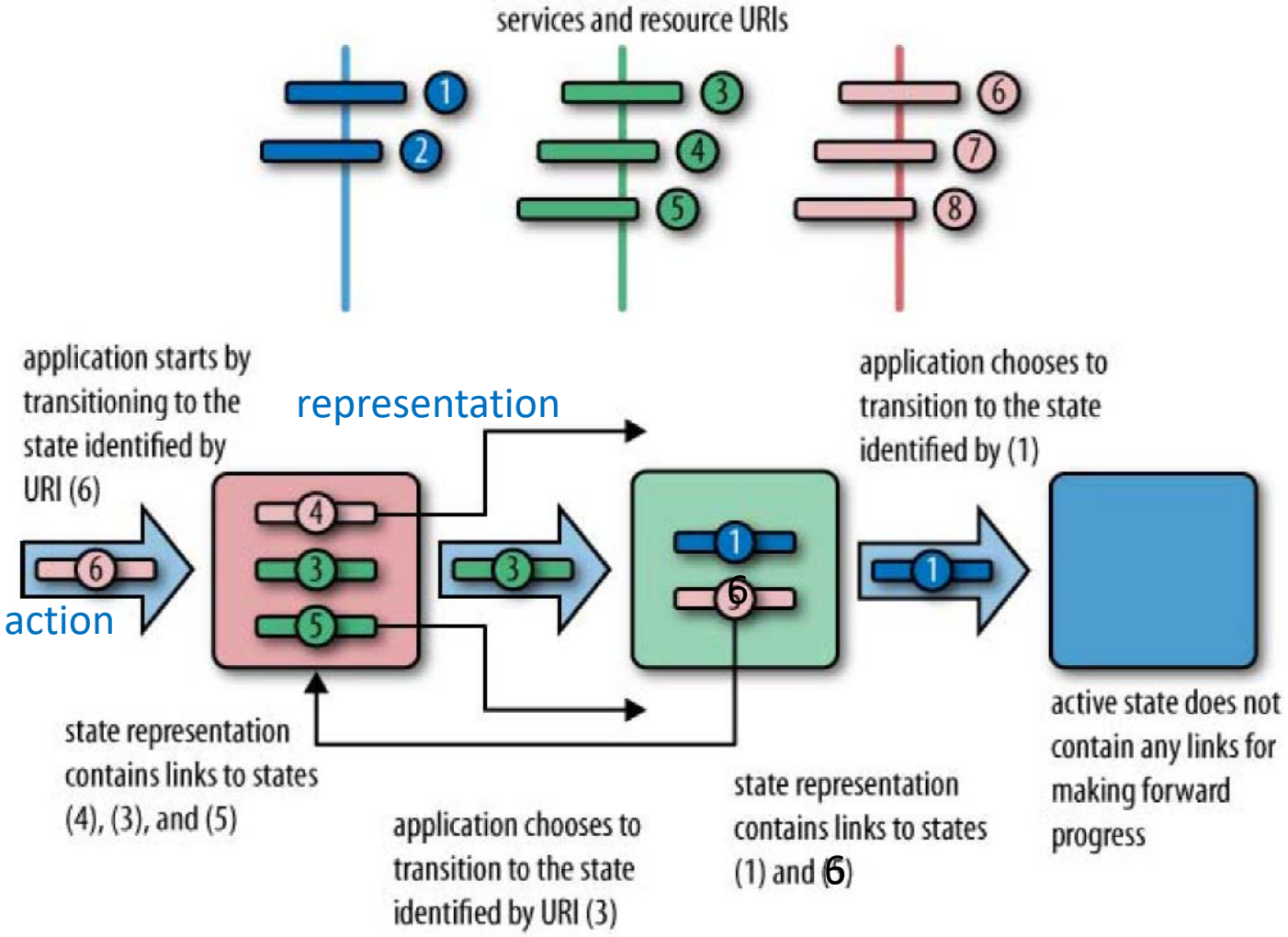
DELETE http://.../200102/aircon/3 (移除3號冷氣)

Level 3 HATEOAS

- HATEOAS
  - Hypertext As The Engine Of Application State
  - A distributed application makes forward progress by transitioning from one state to another
    - State transitions are not known in advance
    - As the application reaches a new state, the next possible transitions are discovered



HATEOAS


services and resource URIs
application starts by transitioning to the state identified by URI (6)
representation
application chooses to transition to the state identified by (1)
action
state representation contains links to states (4), (3), and (5)
application chooses to transition to the state identified by URI (3)
state representation contains links to states (1) and (6)
active state does not contain any links for making forward progress

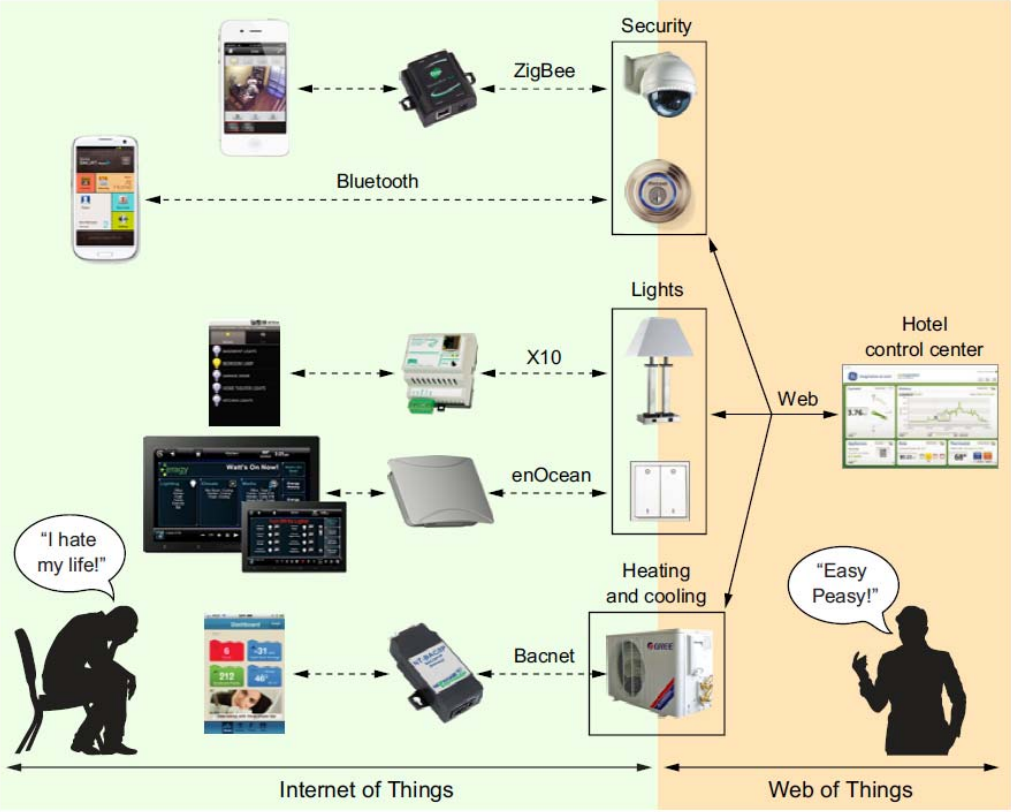
Summary

- Level 1 Resource
  - handling complexity by using divide and conquer, breaking a large service endpoint down into multiple resources.
- Level 2 HTTP Verbs
  - Introduces a standard set of verbs
  - Removes unnecessary variation
- Level 3 HATEOAS
  - Introduces discoverability (know what can do next)

Case: Web of Things (WoT)

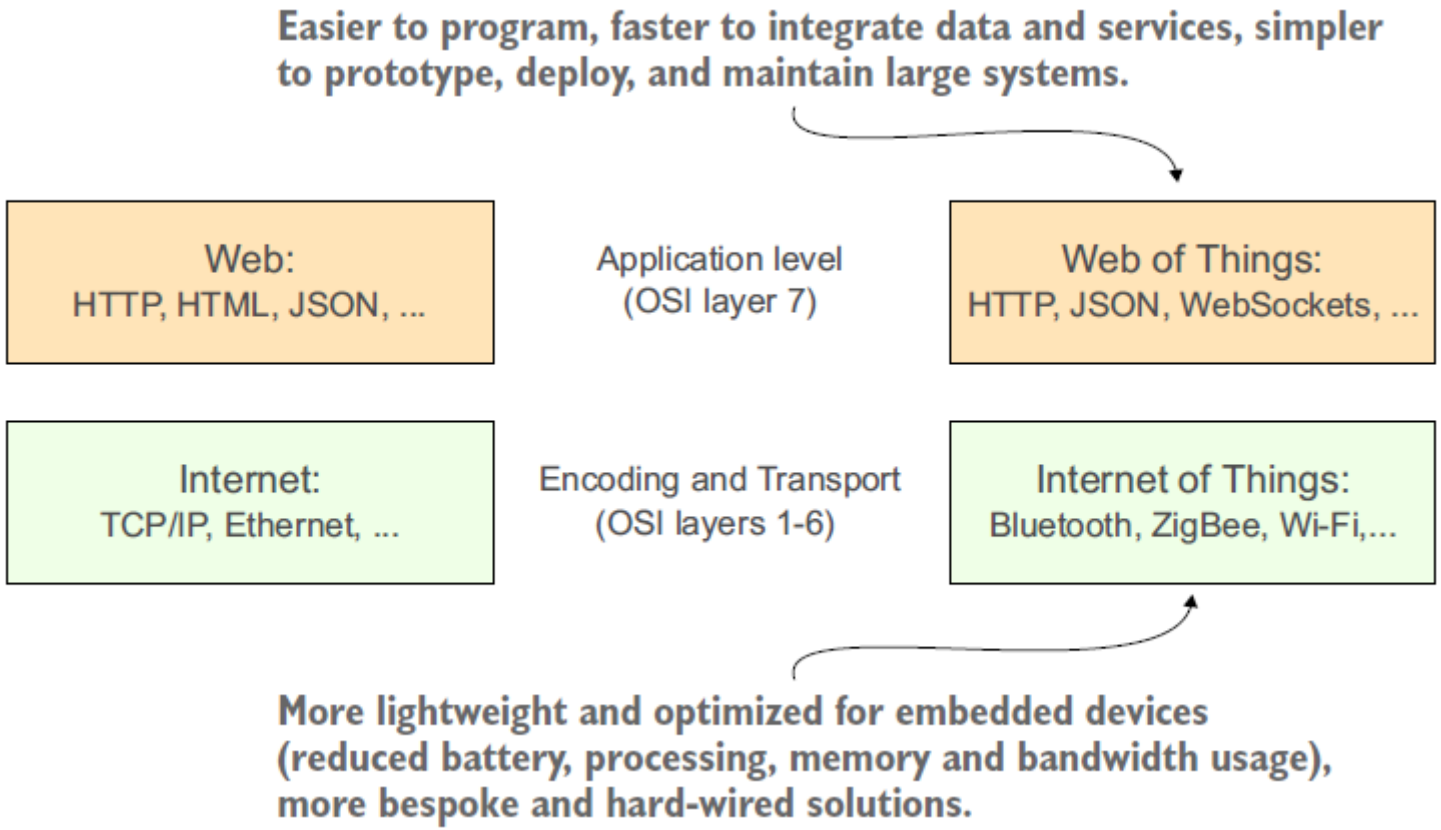
- 現況
  - 太多聯網標準、發展應用程式時，花太多時間在學習通訊協定與整合異質平台
- 願景
  - Web 技術目前已經是十分普及的網路技術
  - 只學習Web相關技術，就能開發應用程式?

要寫很多通訊協定



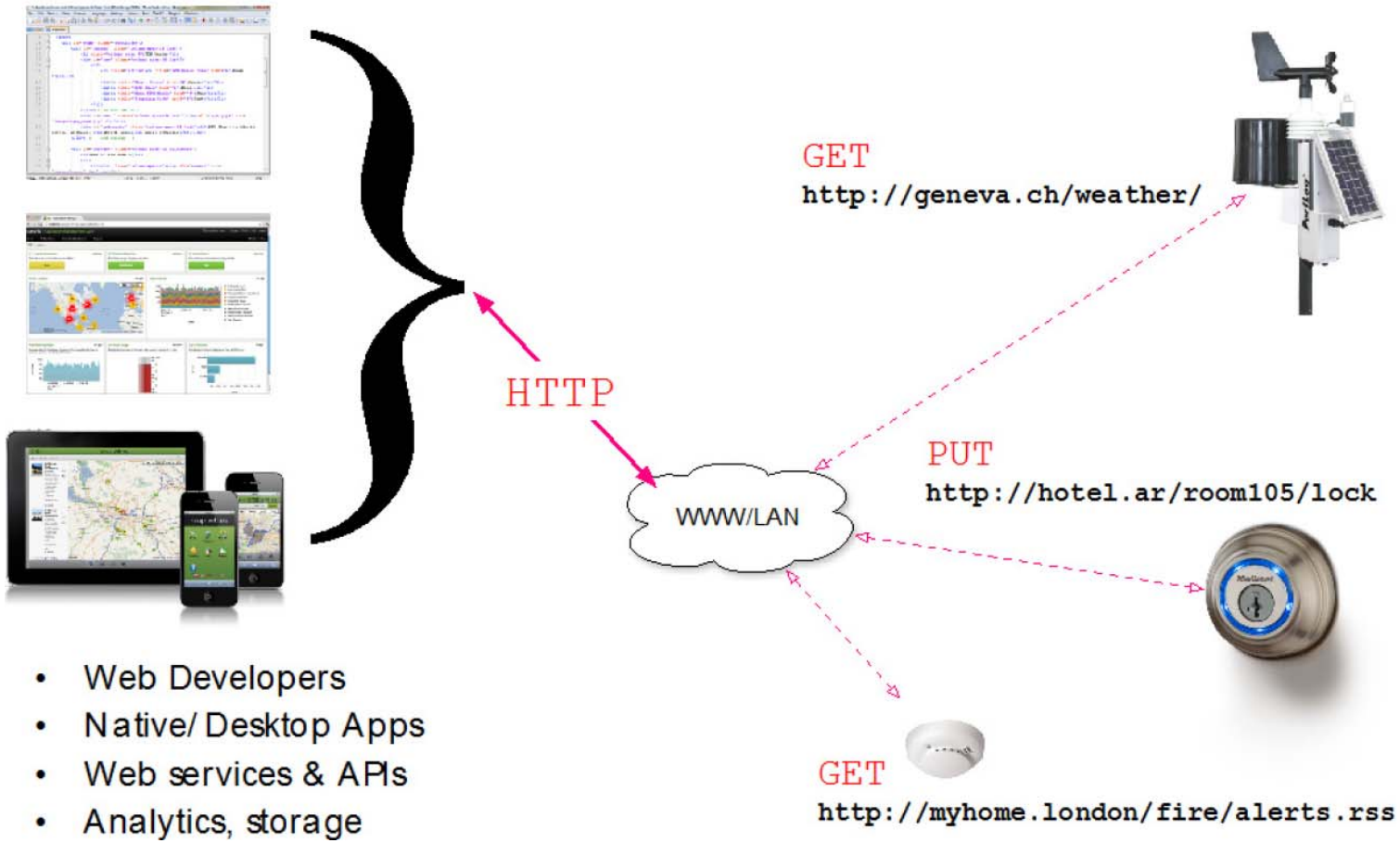
圖片來源: D.D. Guinard & V. M. Trifa, Building the Web of Things, Manning, 2015

IoT and WoT



圖片來源: D.D. Guinard & V. M. Trifa, Building the Web of Things, Manning, 2015

透過Web技術操作所有聯網裝置



Benefits of WoT

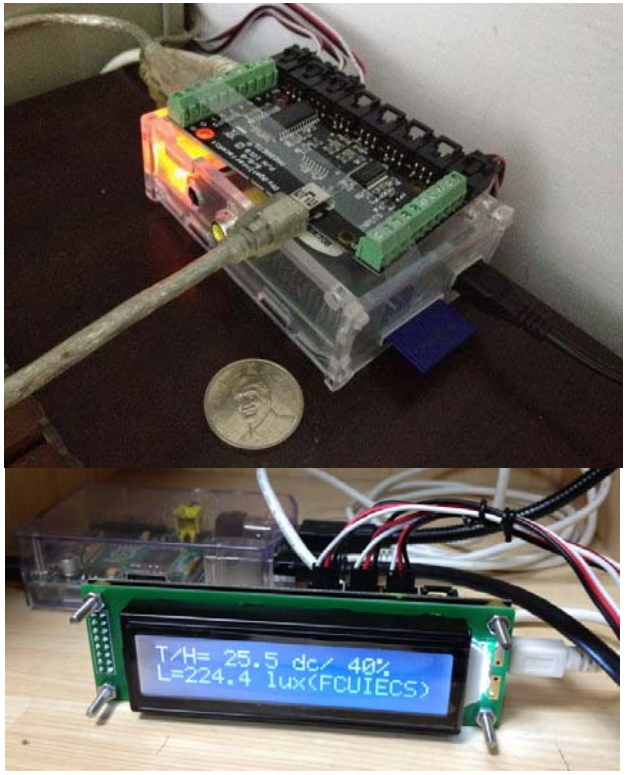
- Easier to program
- Open and extensible standards
- Easy to deploy, maintain, and integrate
- Loose coupling among things
- Widely used security and privacy mechanisms

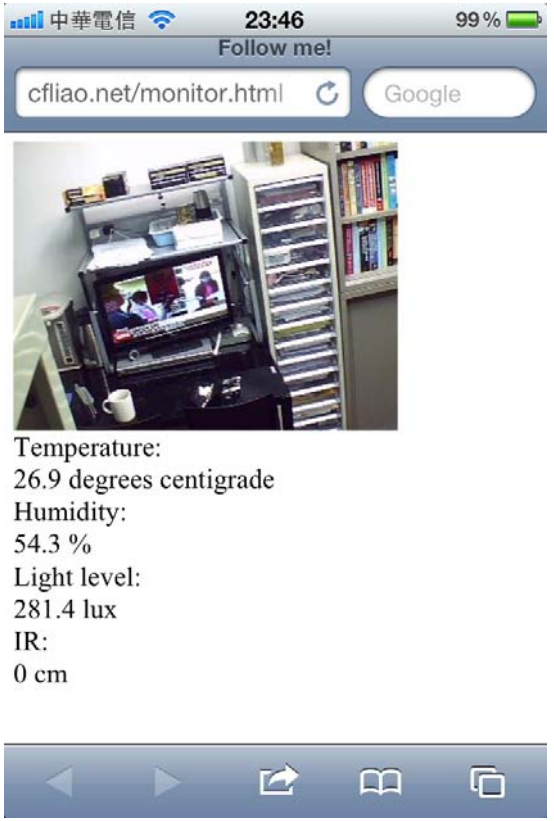
D.D. Guinard & V. M. Trifa, Building the Web of Things, Manning, 2015

智慧空間應用服務開發工具組

透過Web技術即可創作智慧空間應用服務

sMAP (Simple Measurement and Actuation Profile)
由美國Berkeley大學David Culler教授研究群所提出的WoT架構。

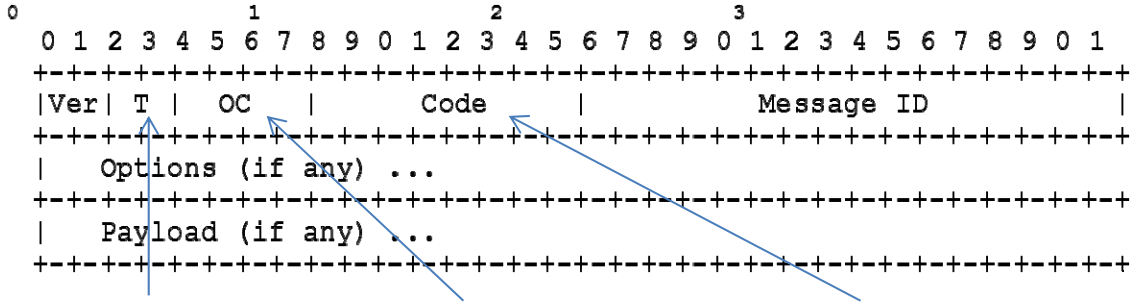




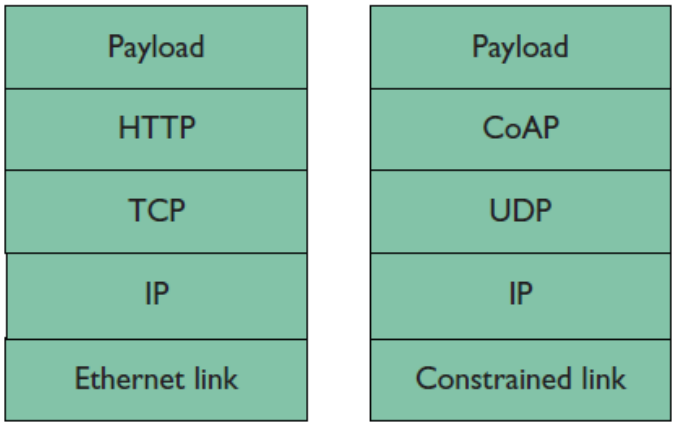
基於Phidget Interface Kit、Raspberry Pi 的硬體，在Raspbian Embedded Linux上使用JAX-RS實現的sMAP架構

CoAP : HTTP for Constrained Network

- CoAP (Constrained Application Protocol)
  - 基於REST架構，專為資源受限裝置所訂定之通訊協定
  - IETF CoRE小組所制定
    - CoRE = Constrained RESTful Environments
  - 約略可類比為是UDP上的HTTP

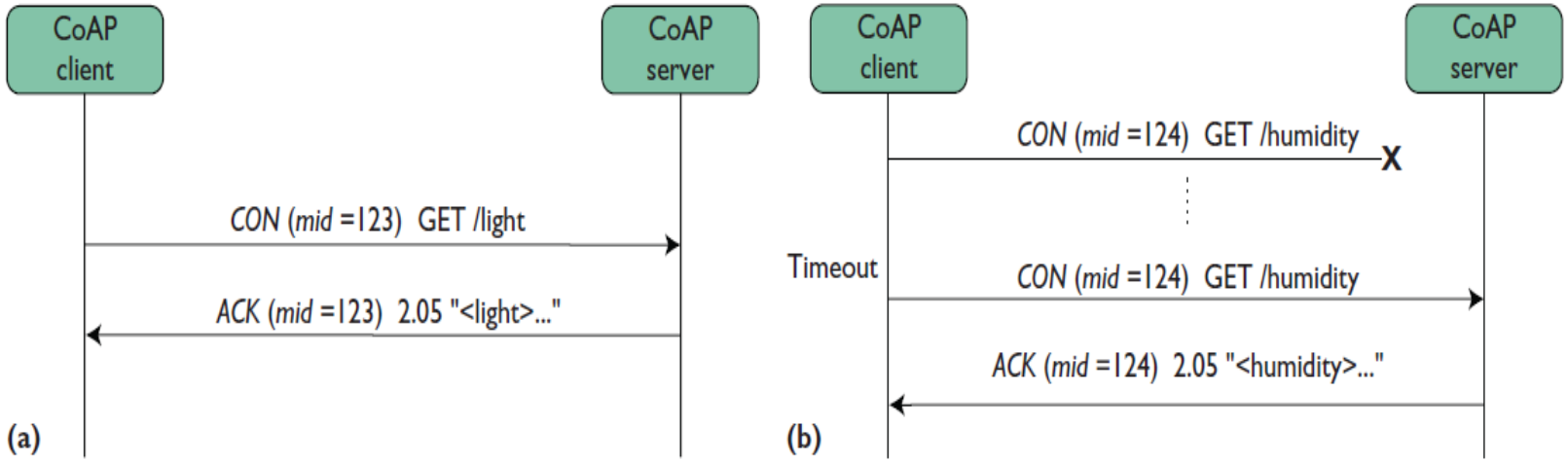


message type　option counts　(類似) http code



CoAP Message Type

- CON: client端發送request, server端要有回應
- NON: client端發送request, server端可不需回應
- ACK: server回應給client 表示收到訊息
- RST: server回應給client 請求重送訊息



REST

- Cons
  - Only supports the request/response style of communication
  - Reduced availability
    - Communicate directly without an intermediary to buffer messages
    - Client and server must both be running for when exchange
  - URL coupling
    - Client must know URL in advance
    - Can be alleviated by service discovery
  - Fetching multiple resources in a single request is challenging
    - 詳見下頁
  - Difficult to map multiple update operations to HTTP verbs
    - 詳見下頁
- Alternative
  - GraphQL: implements flexible, efficient data fetching
    https://www.howtographql.com/basics/1-graphql-is-the-better-rest/

只支援 request 後 respond

REST Cons (Details)

- Fetching multiple resource in a request
  - Hard to realize join-like queries
  - Alternatives for efficient data fetching: GraphQL or Falcor
- Mapping operations to HTTP verbs
  - PUT for update, but there can be multiple ways of update
    - Using sub-resources
    - Using parameters
  - No appropriate mapping for the control operations

https://netflix.github.io/falcor/