Distributed Systems

Chun-Feng Liao

廖峻鋒

Department of Computer Science

National Chengchi University

Distributed Systems

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

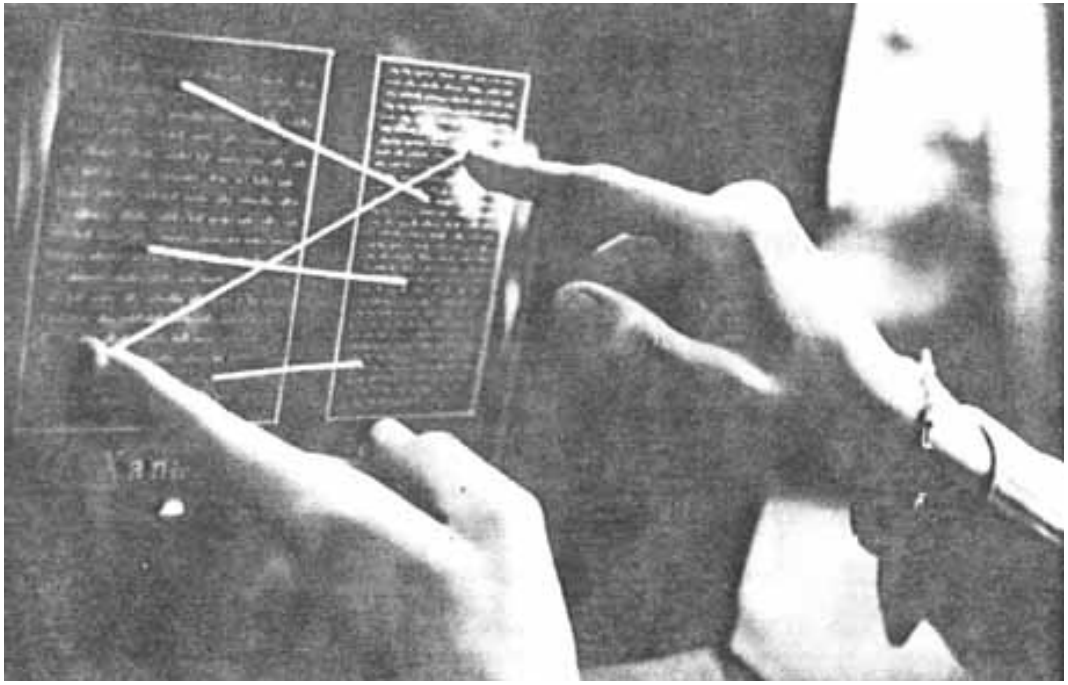
廖峻鋒

Dept. of Computer Science

National Chengchi University

Hypertext Concept

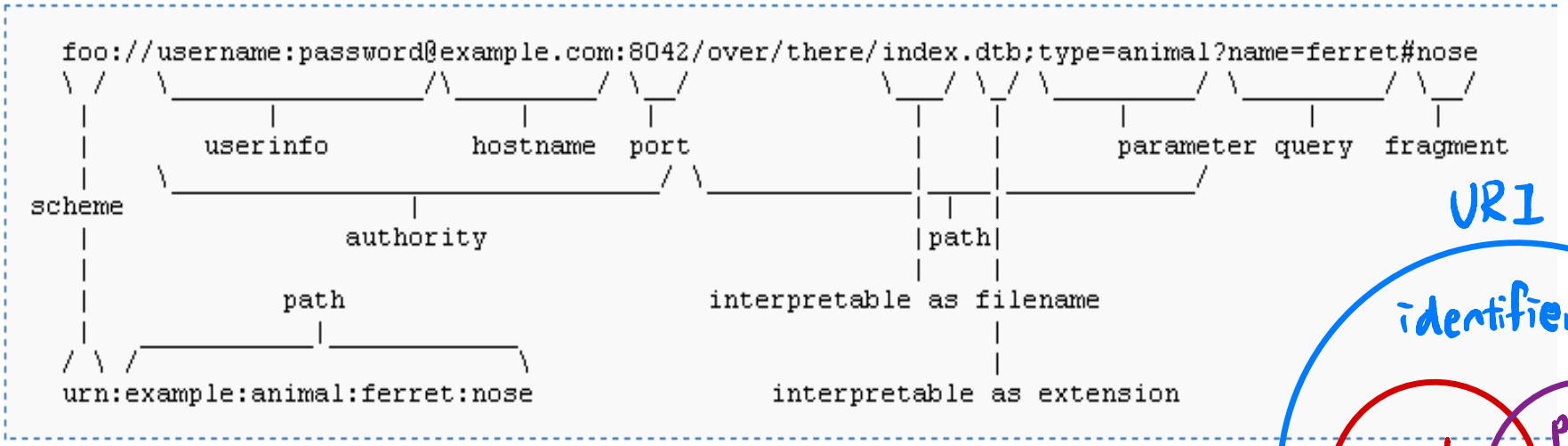
- 起源
  - Project Xanadu (1960s): The Original Hypertext Project
  - Founded by Theodor Holm Nelson

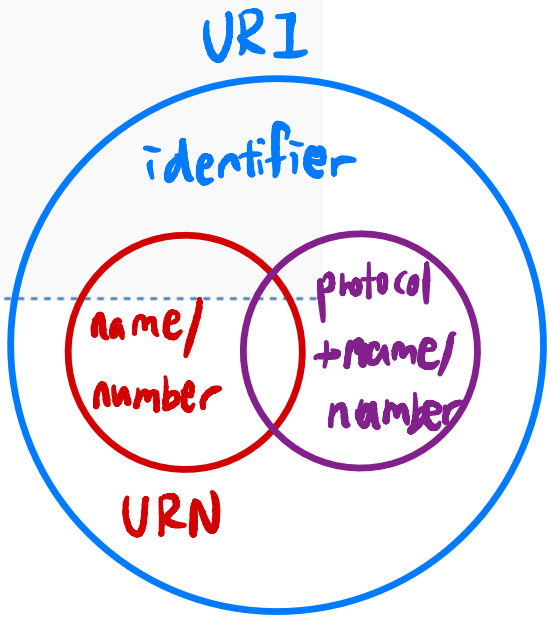


World Wide Web

- 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
    - Hypertext: 文件內文可透過hyperlink交互參考
- 三大元素
  - 文件定址:URL (Universal Resource Locator)
  - 文件格式:HTML (HyperText Markup Language)
  - 文件傳送:HTTP (HyperText Transfer Protocol)

URL





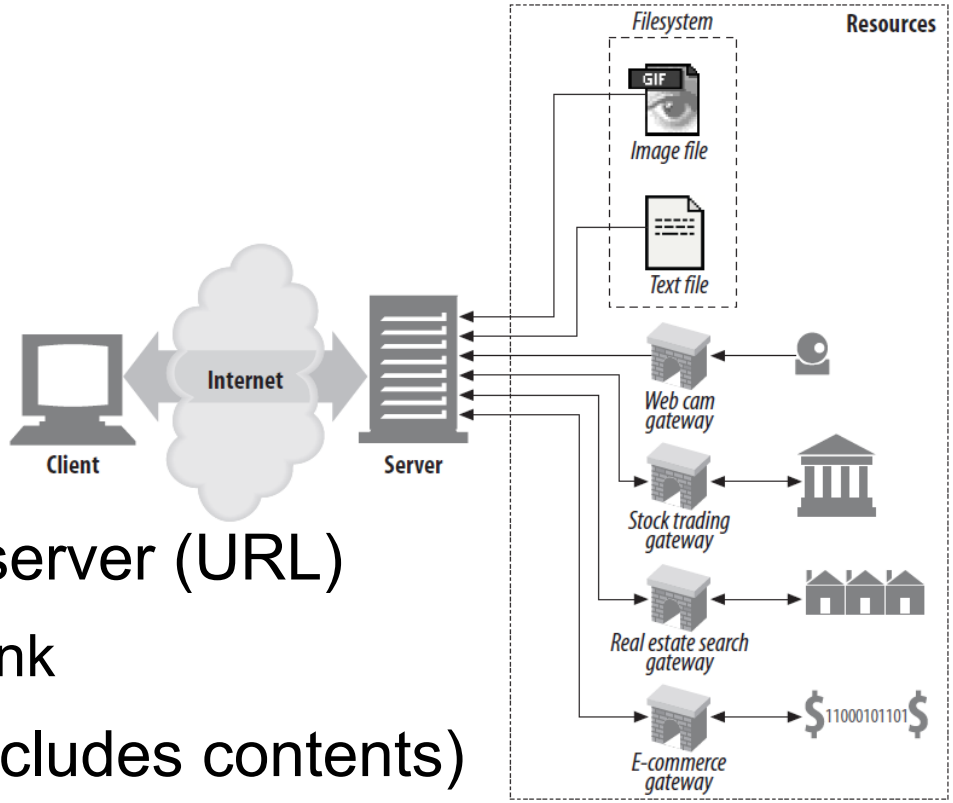
URI只用來辨識，不保證在網路上真正找得到

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

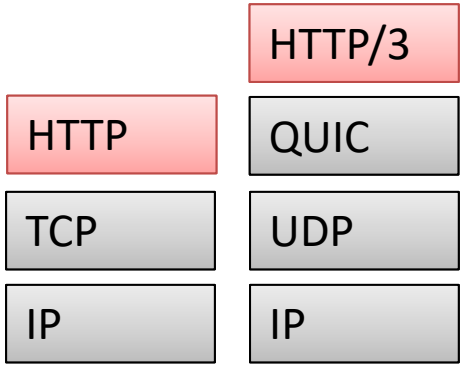
Ex:
http://dl.acm.org/citation.cfm?id=1967428.1967434&coll=DL&dl=ACM&CFID=70667179
2&CFTOKEN=10622052

Web主要元素

- Resource
  - Deliver 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



HTTP

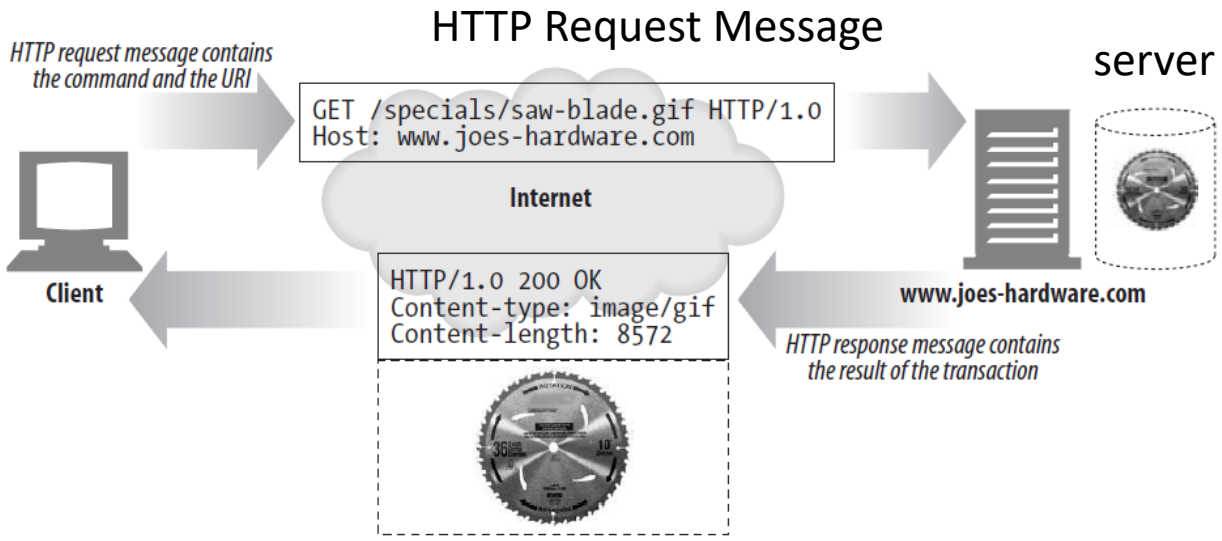


- Hypertext Transport Protocol
  - Application layer protocol
  - Based on TCP/IP
    - HTTP/2 SPDY: multiplexed connections
    - HTTP/3 將基於QUIC https://zh.wikipedia.org/zh-tw/HTTP/3
- Language of the Web
  - Protocol used for communication between Web clients and servers
- Is stateless (指server-side)
  - Server does not maintain the state of a session
  - Server的負擔較輕
  - 每次交談時，Client要自帶完整前後文

pronounced "speedy", obsolete

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

Request

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

[  optional request body  ]
```

header

body

Response

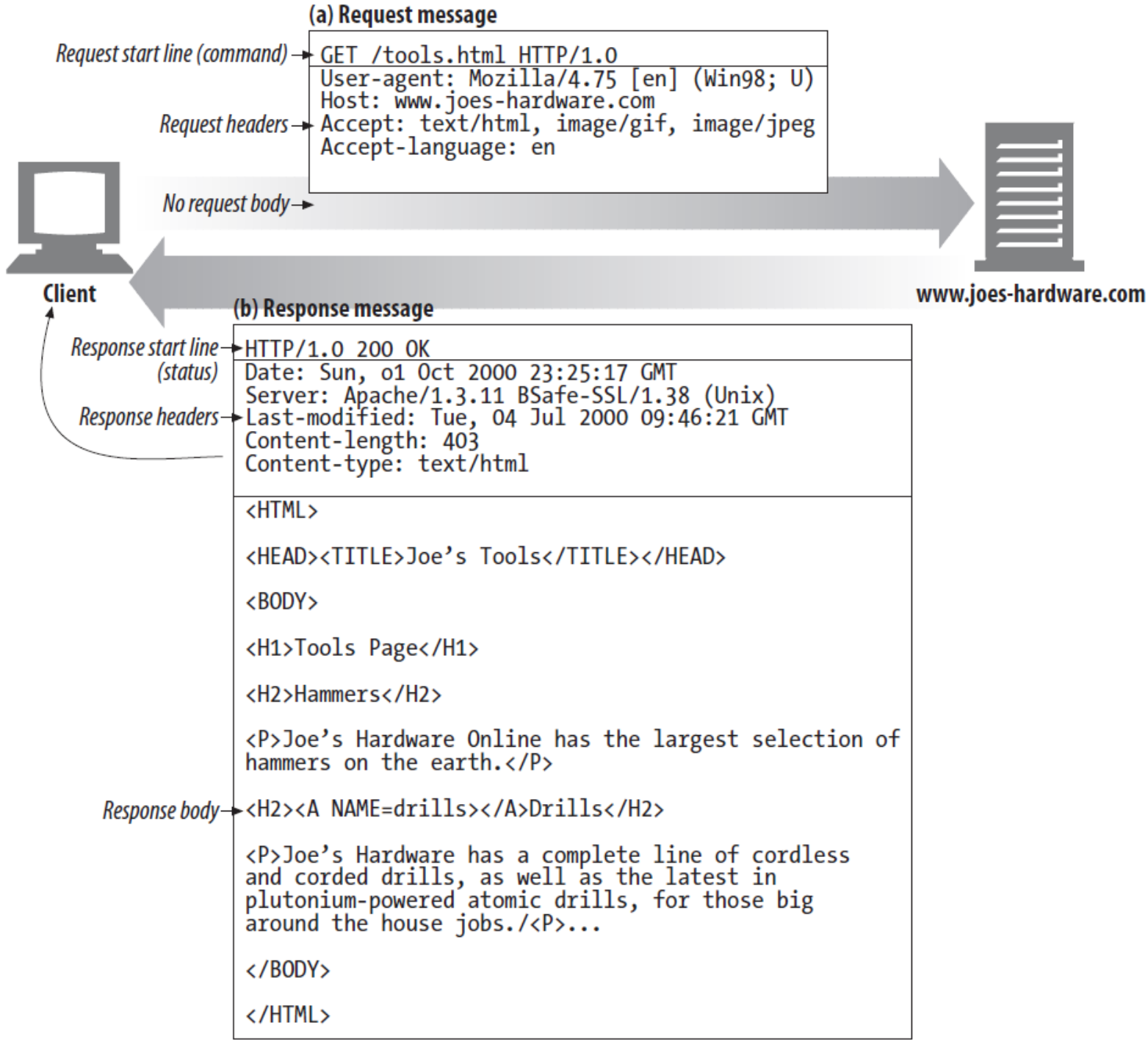
```
HTTP/1.1 200 OK
X-Powered-By: Express
Accept-Ranges: bytes
ETag: "149-1456280156295“
Date: Wed, 24 Feb …
Cache-Control: public, max-age=0
Last-Modified: Wed, 24 Feb …
Content-Type: text/html; charset=UTF-8
Content-Length: 149
Connection: keep-alive

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

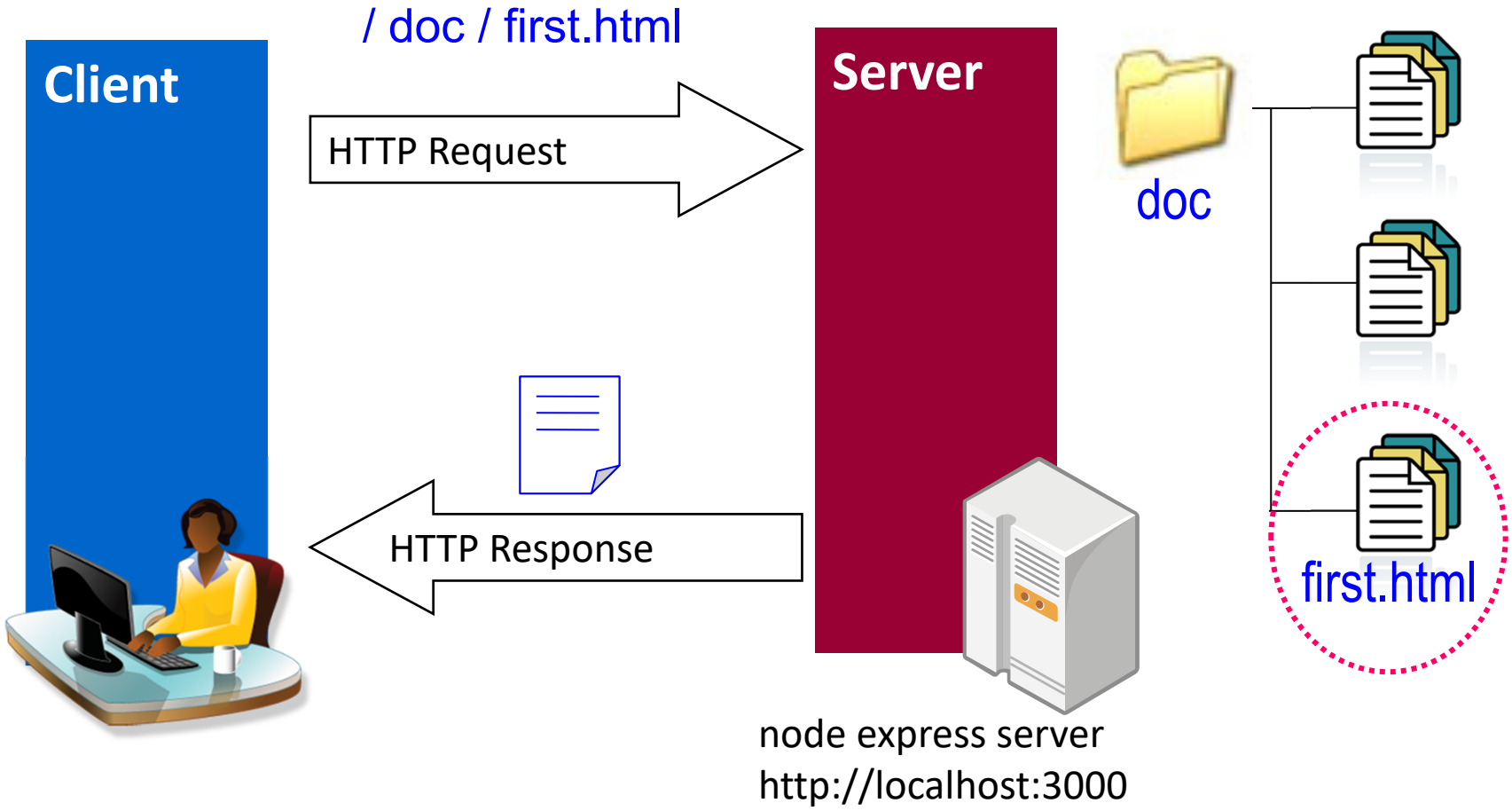
header

body

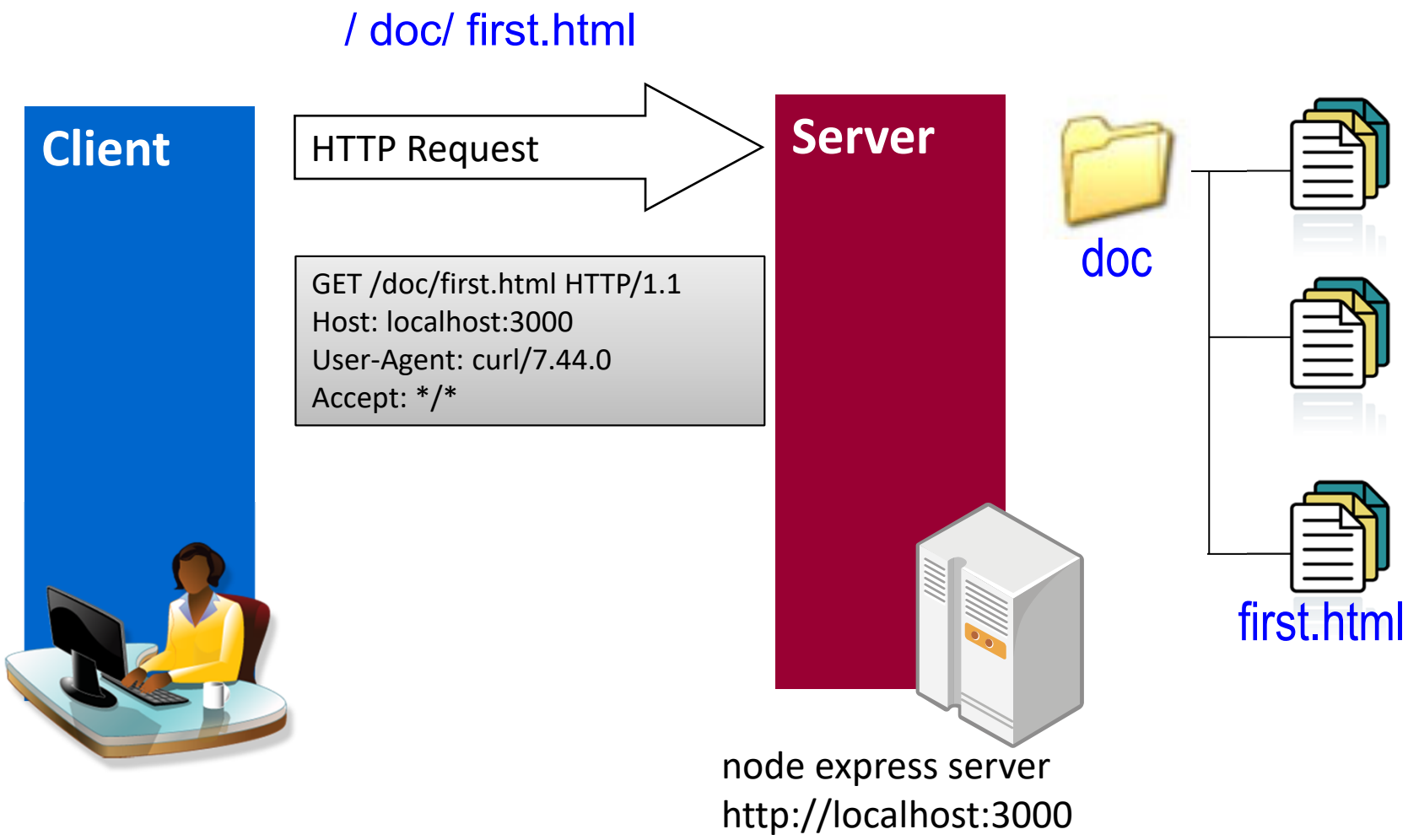
NOTIFY
M-SEARCH


(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>

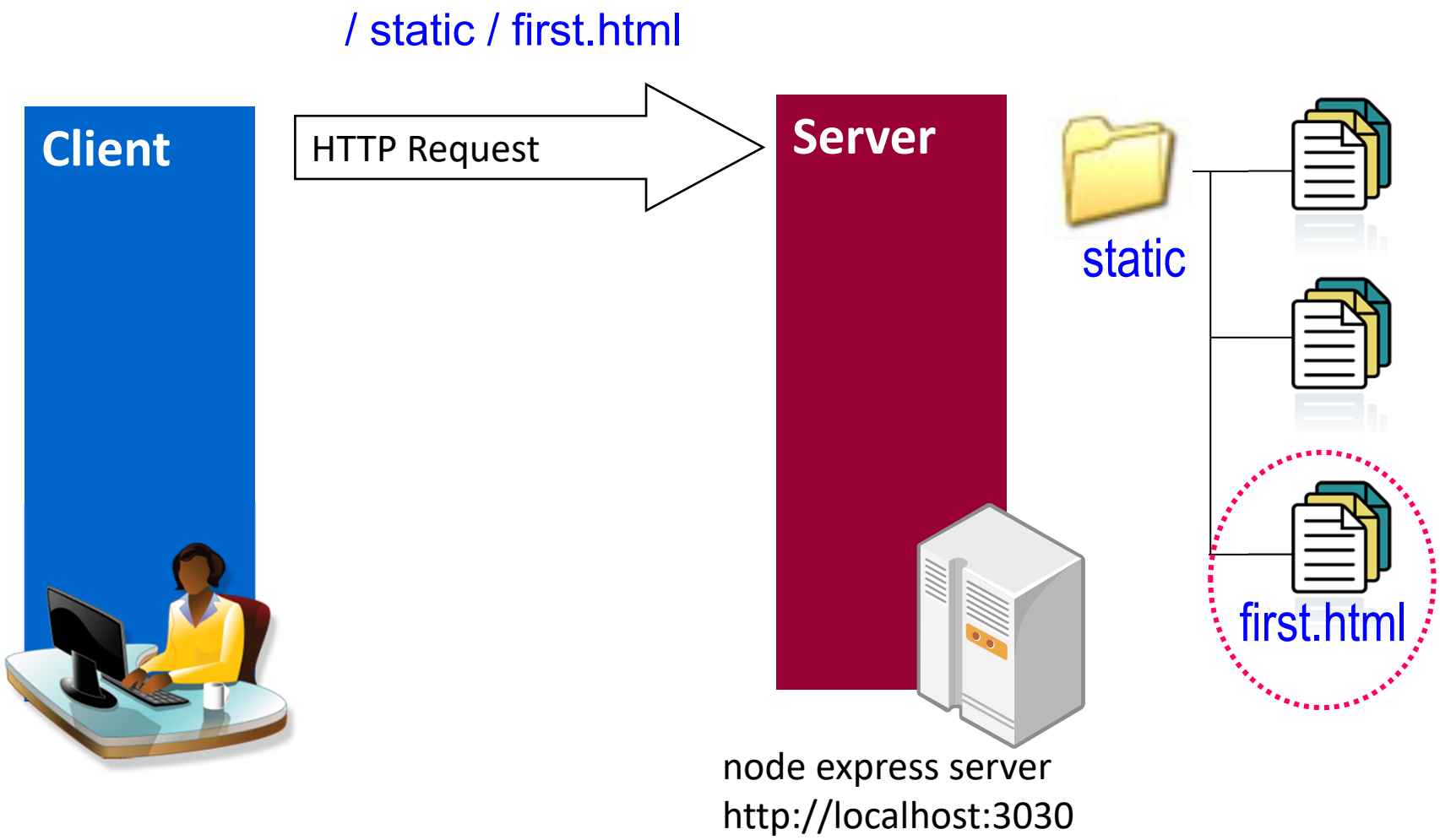
傳送靜態文件


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

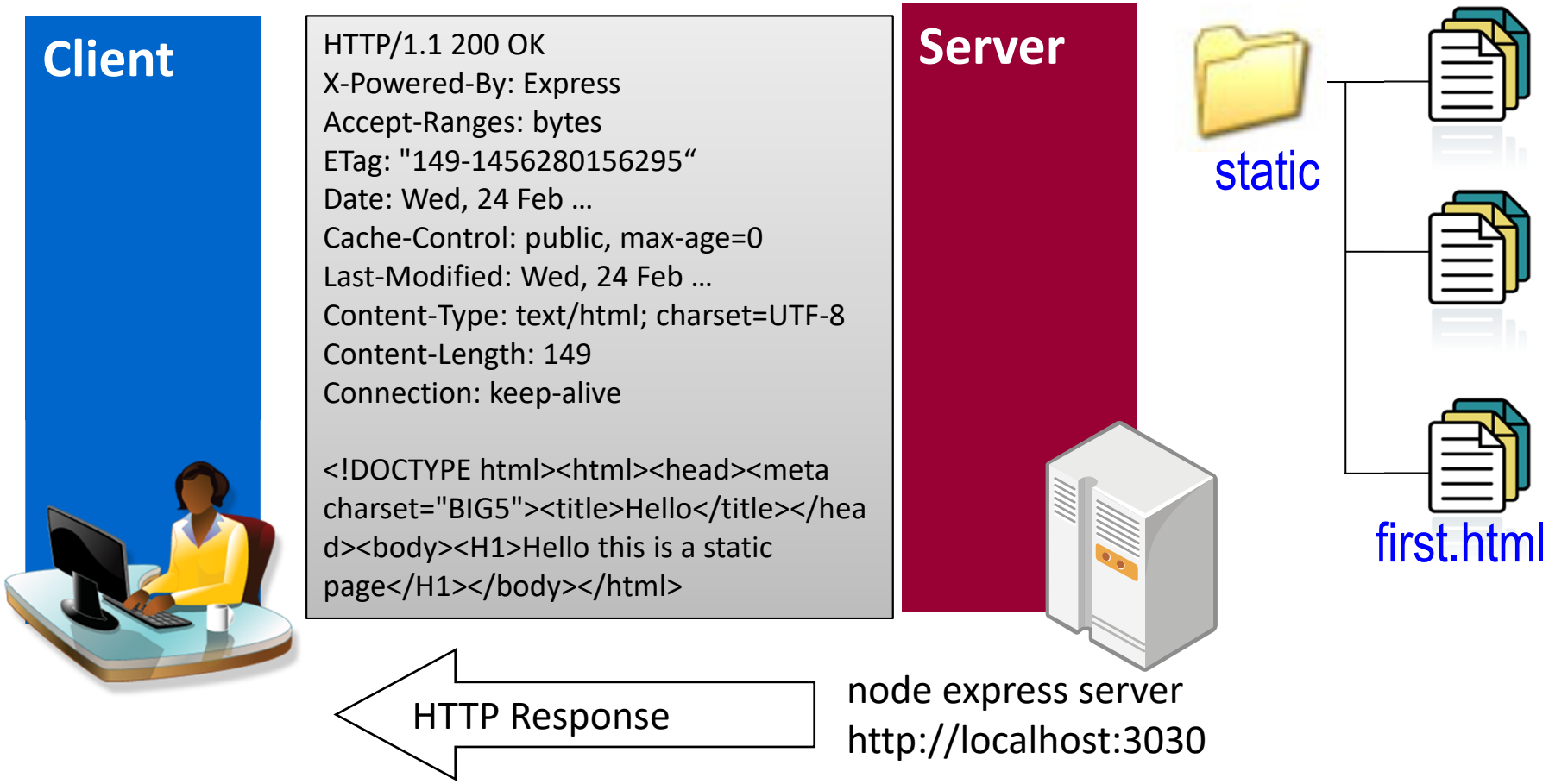
傳送靜態文件


/ 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

HTTP Request

- Request line
- Request header
- Request body

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

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

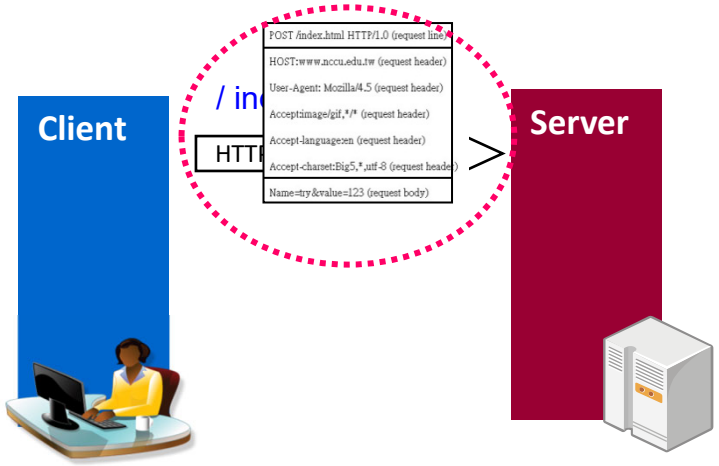
User-Agent: Mozilla/4.5 (request header)

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)

HTTP Request Methods

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

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 (idempotent)

  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) remote, application loop-back test

使用curl –X (Method_Name)

demo

- 回應trace method

```
httpServer.trace('/', function(req, res) {
        res.send(req.method);
});
```

curl –X TRACE http://HOST:PORT/

Response 訊息

HTTP/1.0 200 OK

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

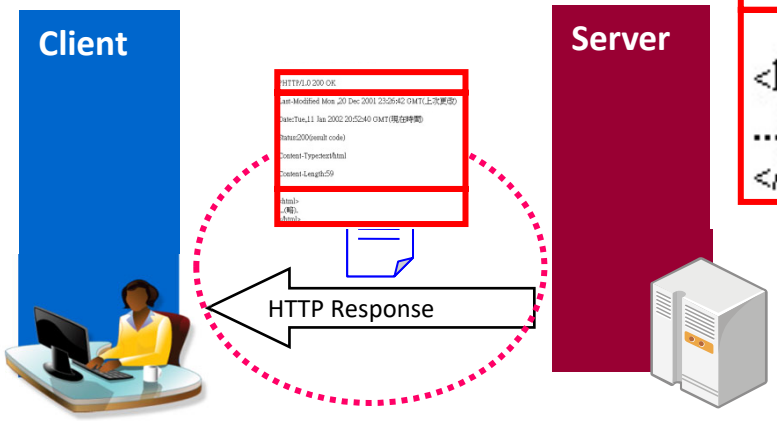
Date:Tue,11 Jan 2002 20:52:40 GMT(現在時間)

Status:200(result code)

Content-Type:text/html

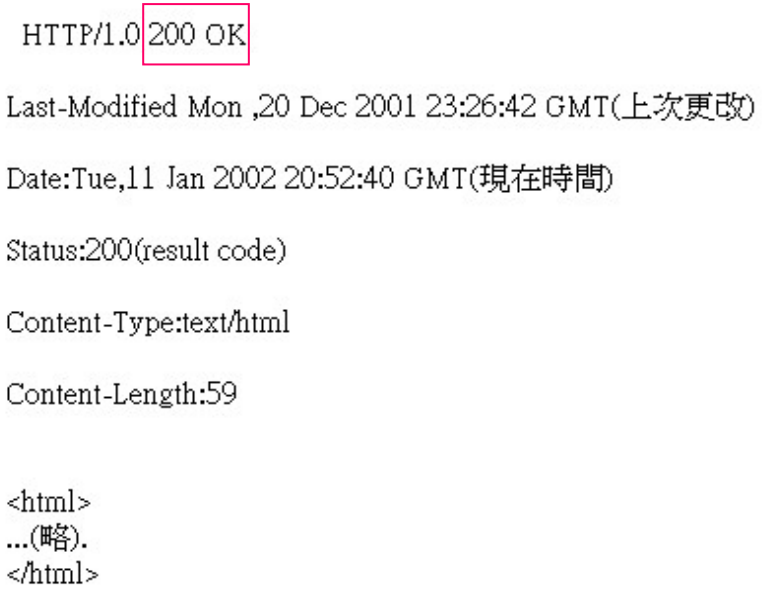
Content-Length:59

<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。 redirect
- 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

Representational State Transfer

Chun-Feng Liao

廖峻鋒

Dept. of Computer Science

National Chengchi University

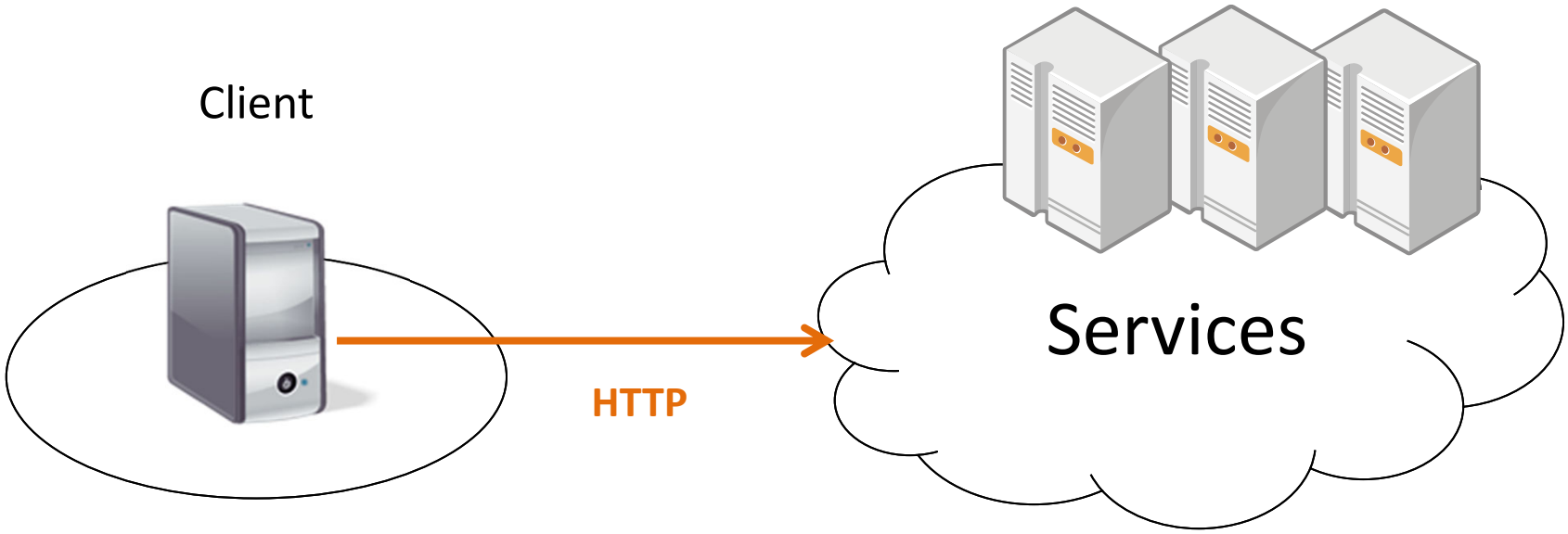
當Web上不只放網頁...

- 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}

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 (method, fnc)
  - 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 (複雜)
- REST
  - 將HTTP視為應用程式平台，將服務視為物件(資源) (object : apply CRUD)
  - 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 (Web Application Description Language)

gRPC (google)

← SOAP, gRPC

HTTP ← REST

TCP

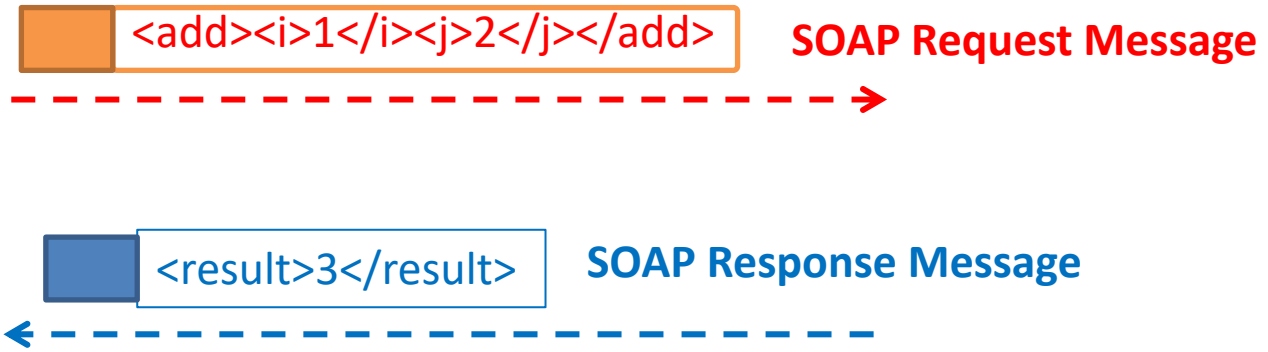
IP

Eth

SOAP

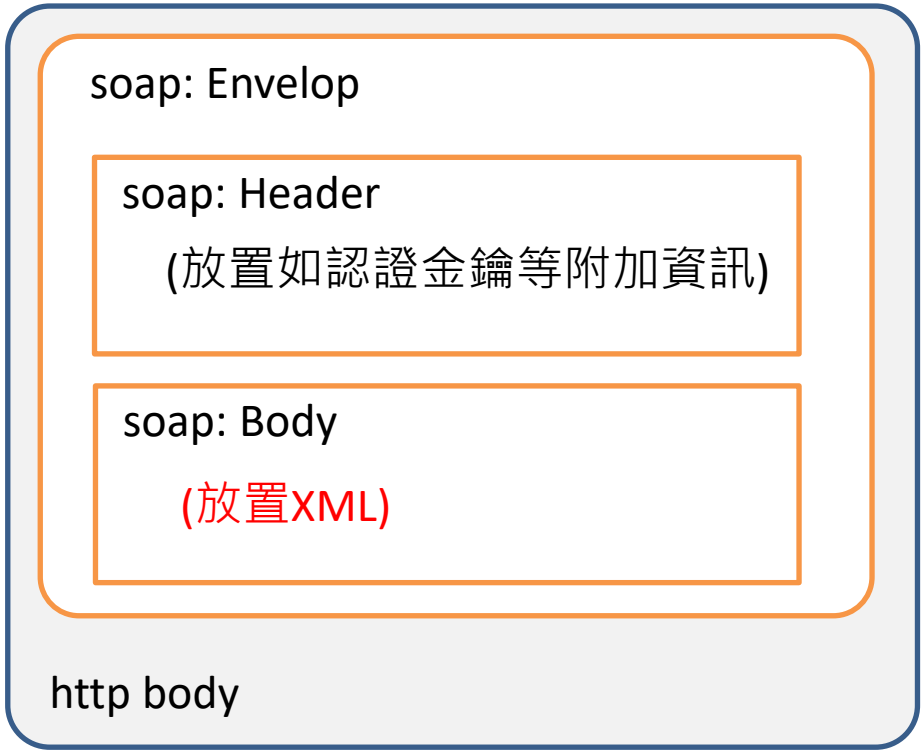
- Simple Object Access Protocol
- An XML-based communication protocol
  - let applications exchange information over HTTP

HTTP body



(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)

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

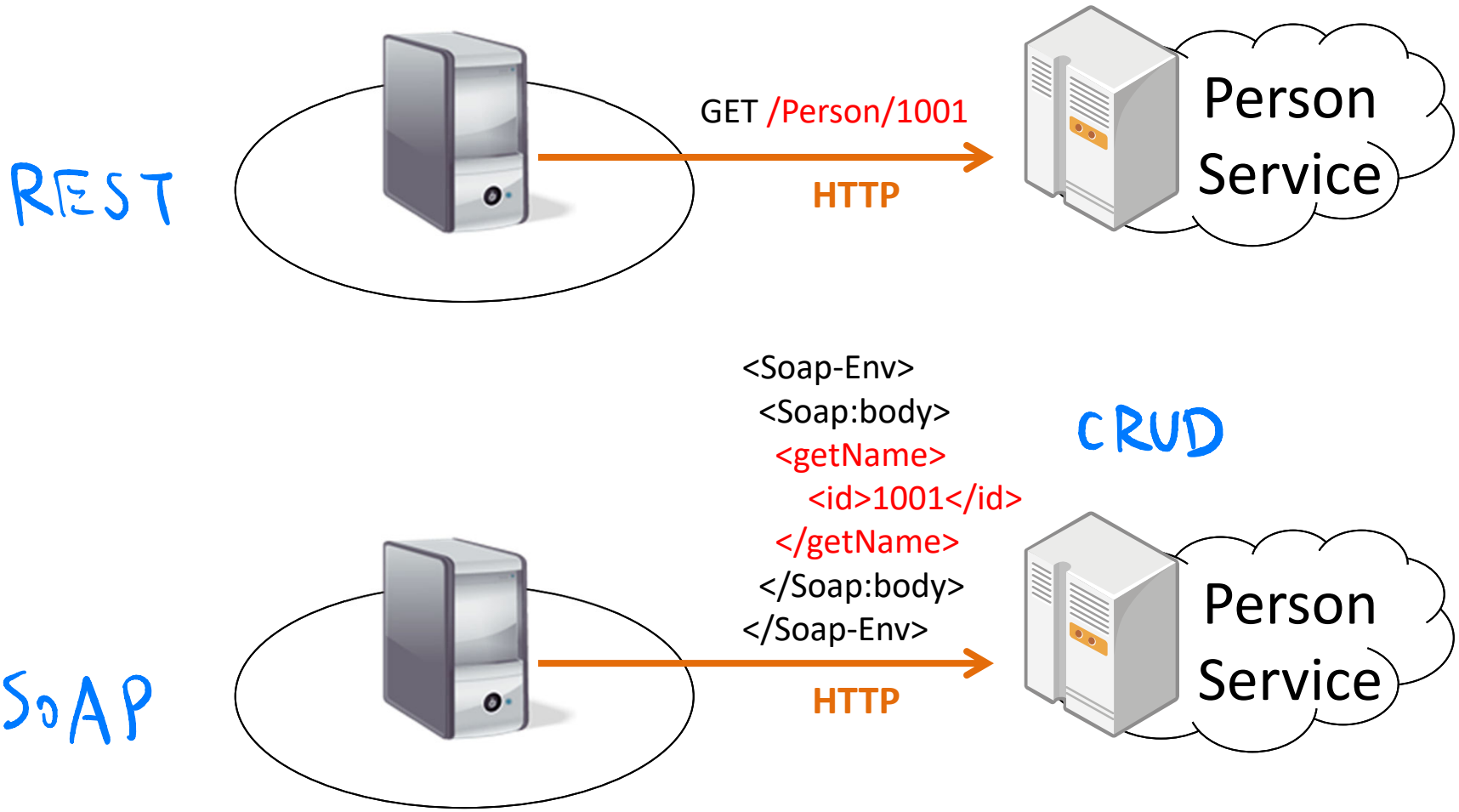
- R. Fielding在其博士論文提出之架構風格
  - 將HTTP的精神套用到Web Services上
  - “REST is not a standard, but it describes the use of standards”
    - HTTP/ URL/ XML
- 和傳統SOAP/WSDL Web Services各有優勢

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

REST

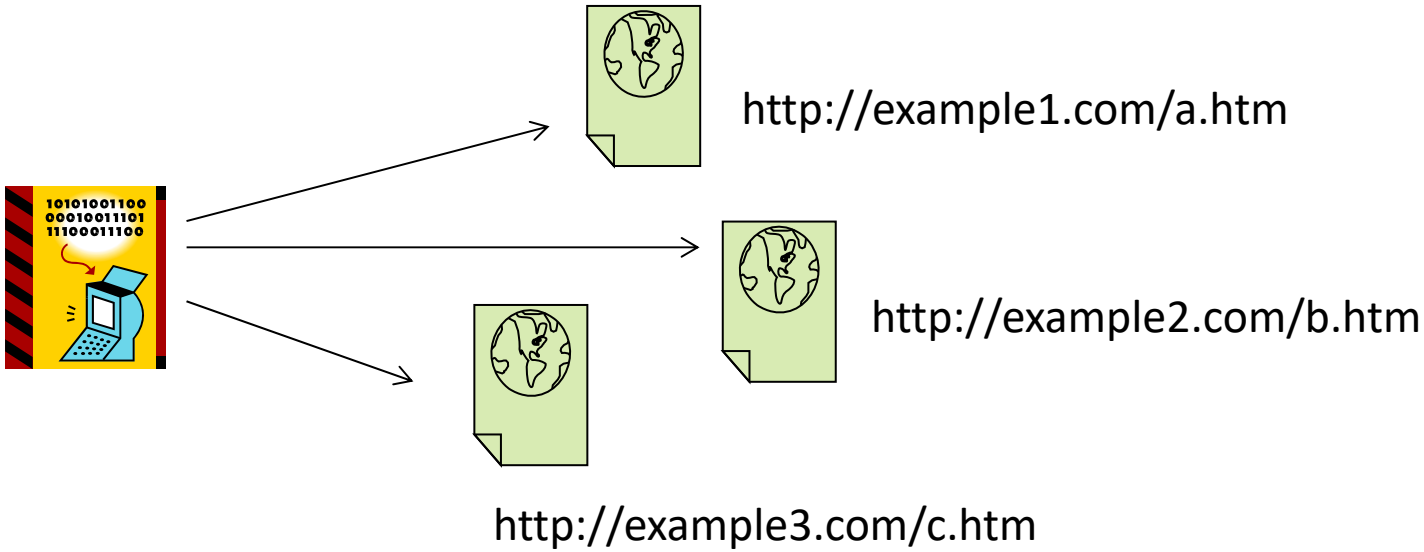
- Resource
  - typically represents a single “business object” (Domain Object)
  - Ex: Customer, Product
- HTTP verbs
  - GET query a resource
  - POST create, insert or update a resource
  - PUT create or update a resource
- Example
  - GET /orders/3
    - Retrieve the Order instance where order_id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

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

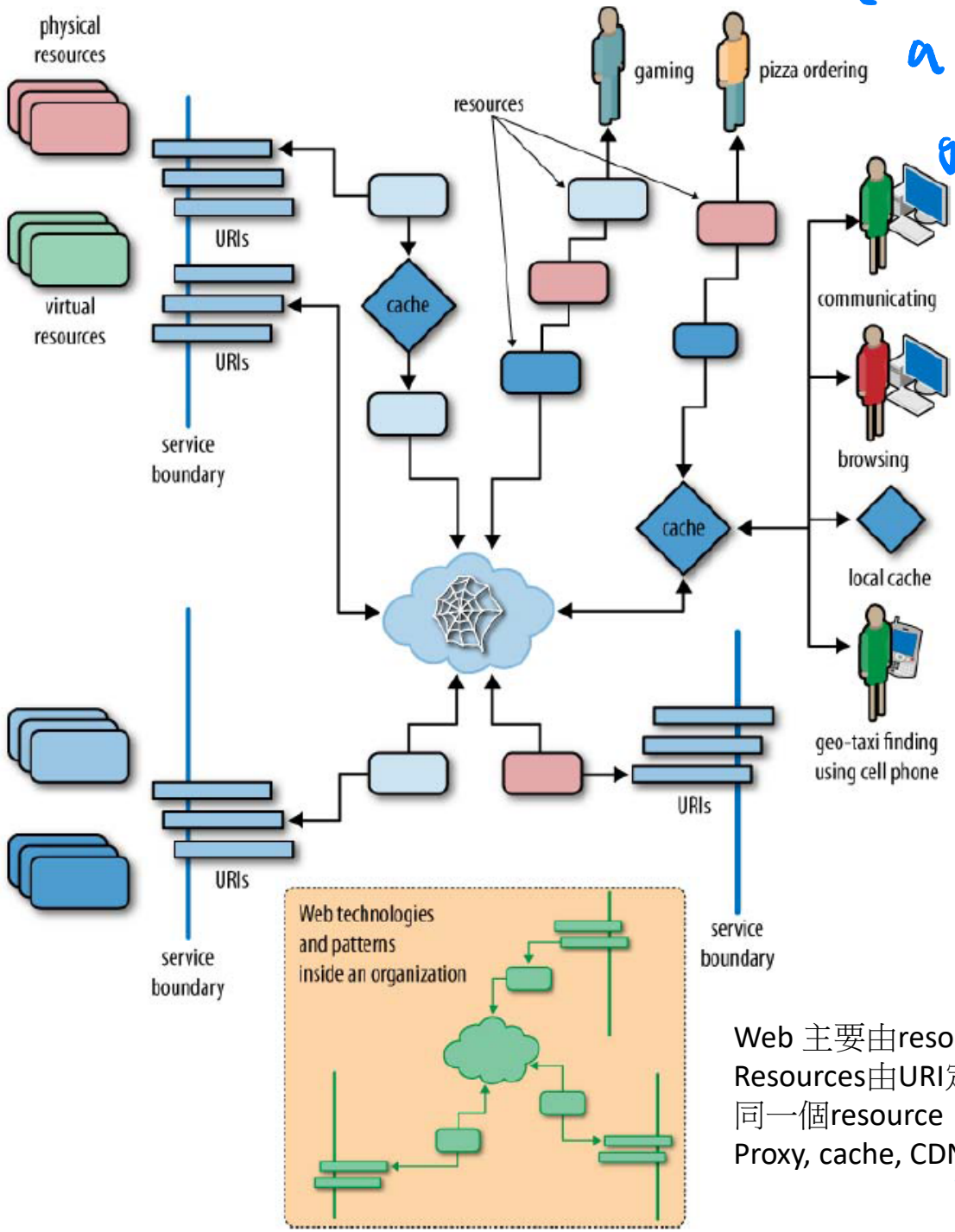


RESTful Web Services

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



The Web



* CDN (content delivery network): a geographically distributed group of servers that caches contents close to end users.

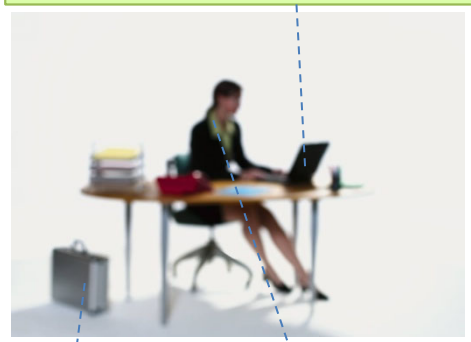
Web 主要由resources組成，
Resources由URI定址，多個URI可以指向同一個resource
Proxy, cache, CDN現在也是Web的一部份

Example

萬物皆視被為具有URI的資源(Resources)

http://acme.com/dep1/stapler

http://acme.com/dep1/notebook1

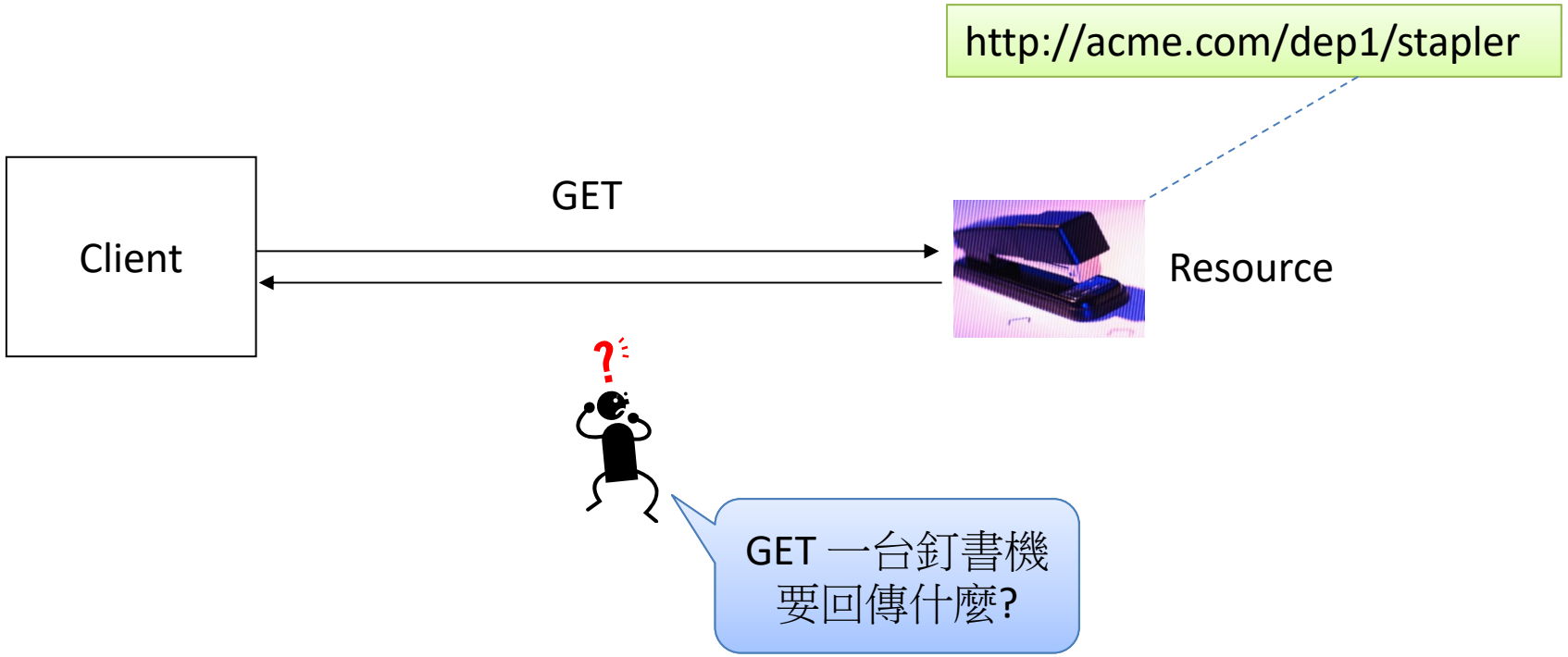


http://acme.com/dep1/phone1

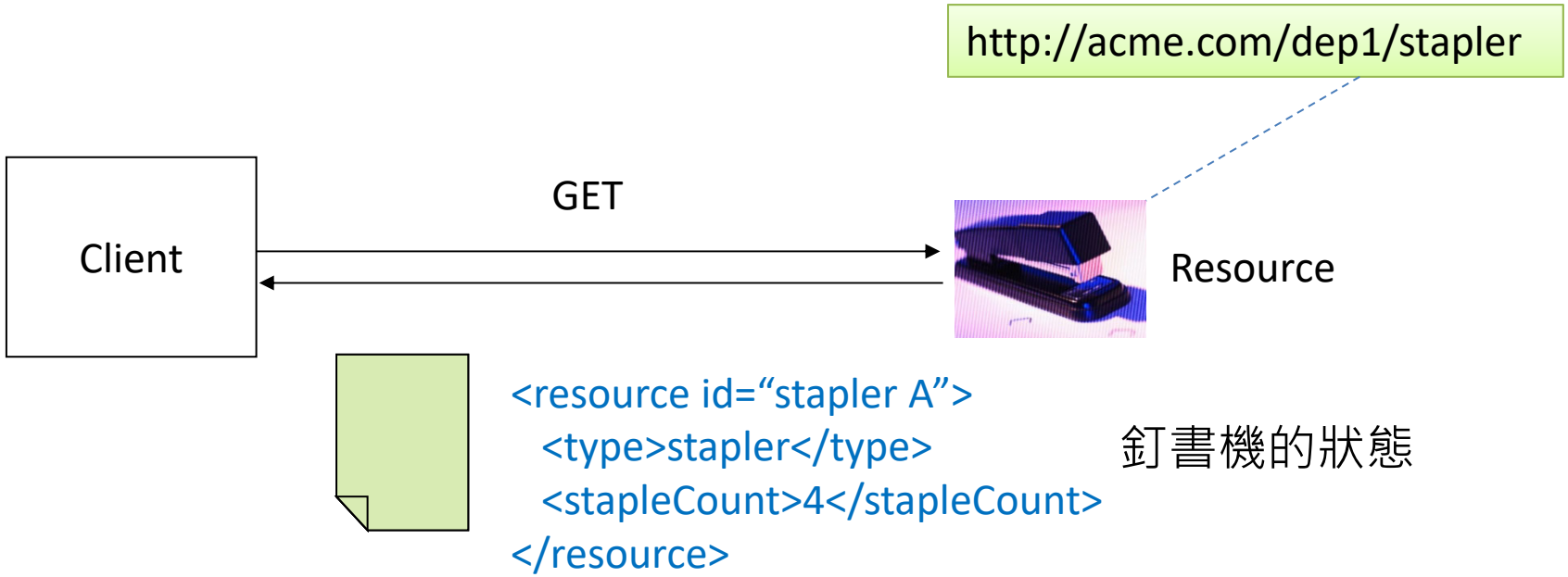
http://acme.com/dep1/person/Helen

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

in http headers

MIME=Multipurpose Internet Mail Extensions: an internet standard that support text in character sets as well as audios, videos, images & apps.

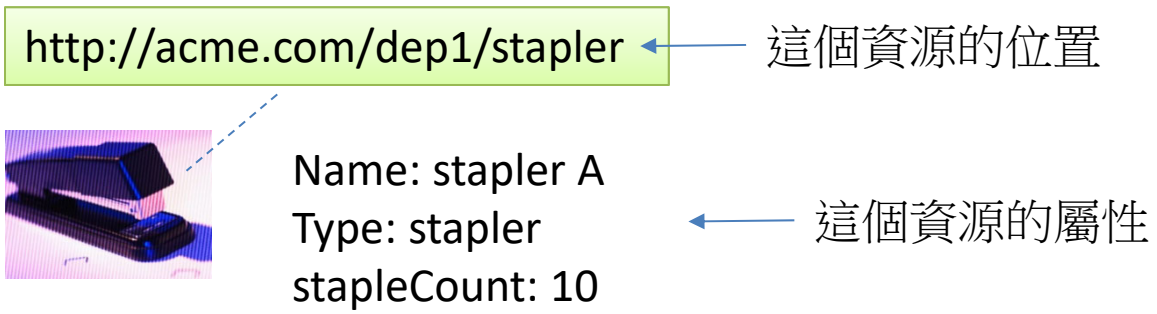
staplerService

- staplerService.js

curl.exe -X PUT -d count=4 http://localhost:3000/stapler
curl.exe -X POST http://localhost:3000/stapler

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

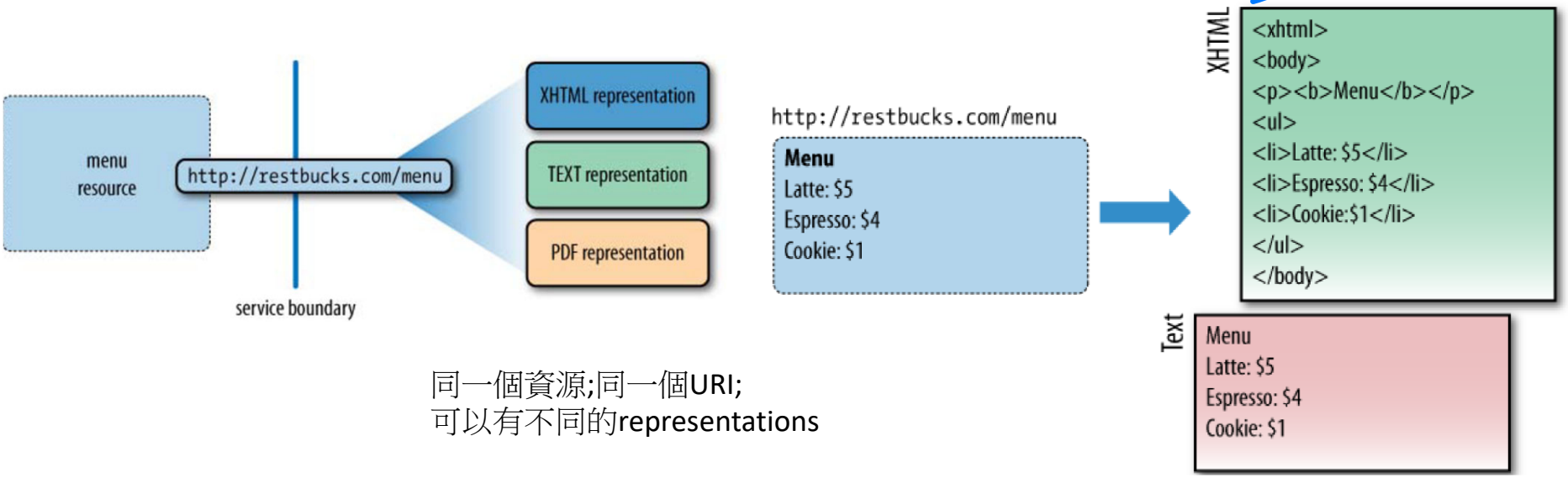


Elements of the Web

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

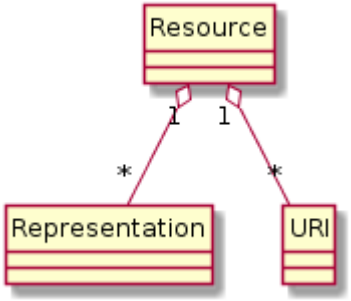
      Accept: text/html
  - Web components exchange representations of resources

eXtensibe Hyper Text Markup Language



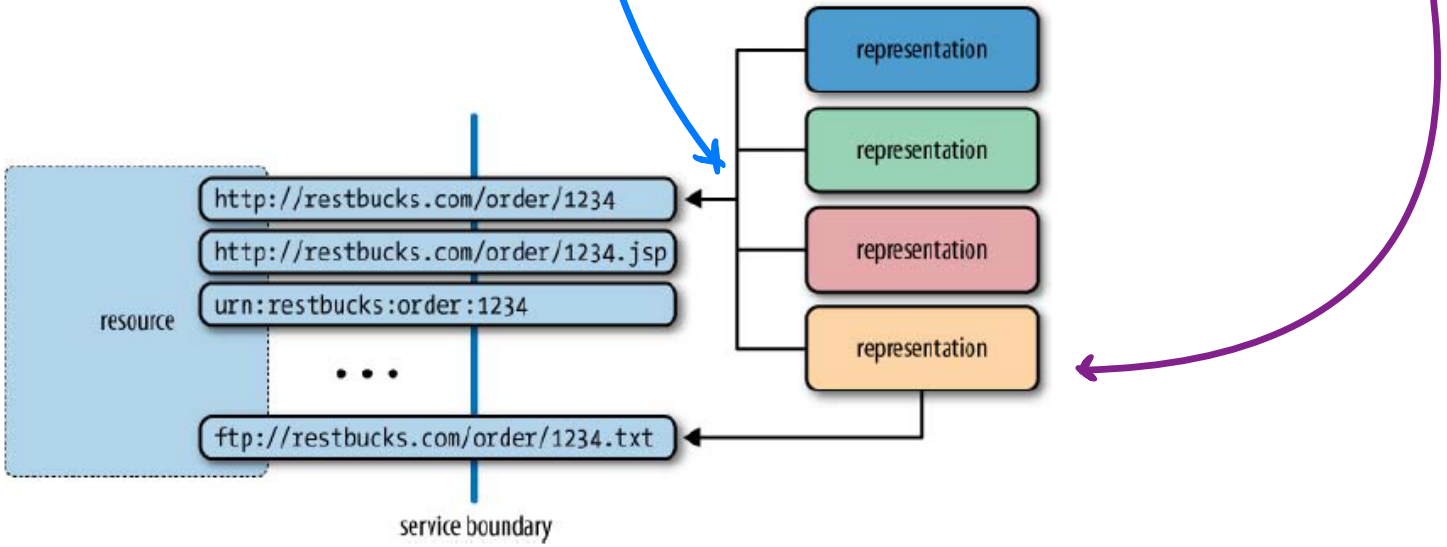
同一個資源;同一個URI;
可以有不同的representations

Elements of the Web



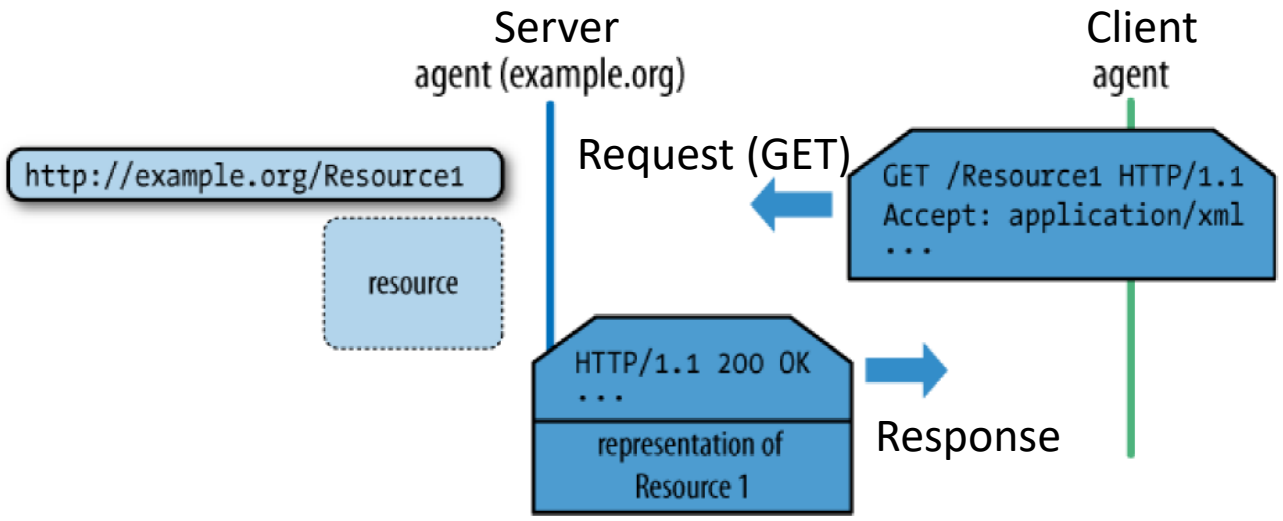
- 一個Resource可被多個URI對應
- 一個Resource可具有多種Representation

(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…



對資源做CRUD

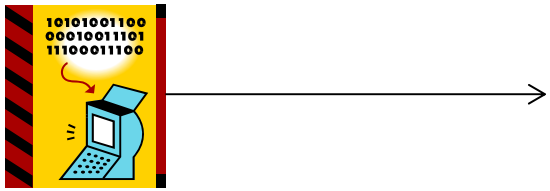
對一個Resource做出POST/GET/PUT/DELETE會代表什麼意義?

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

Create
Read
Update
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

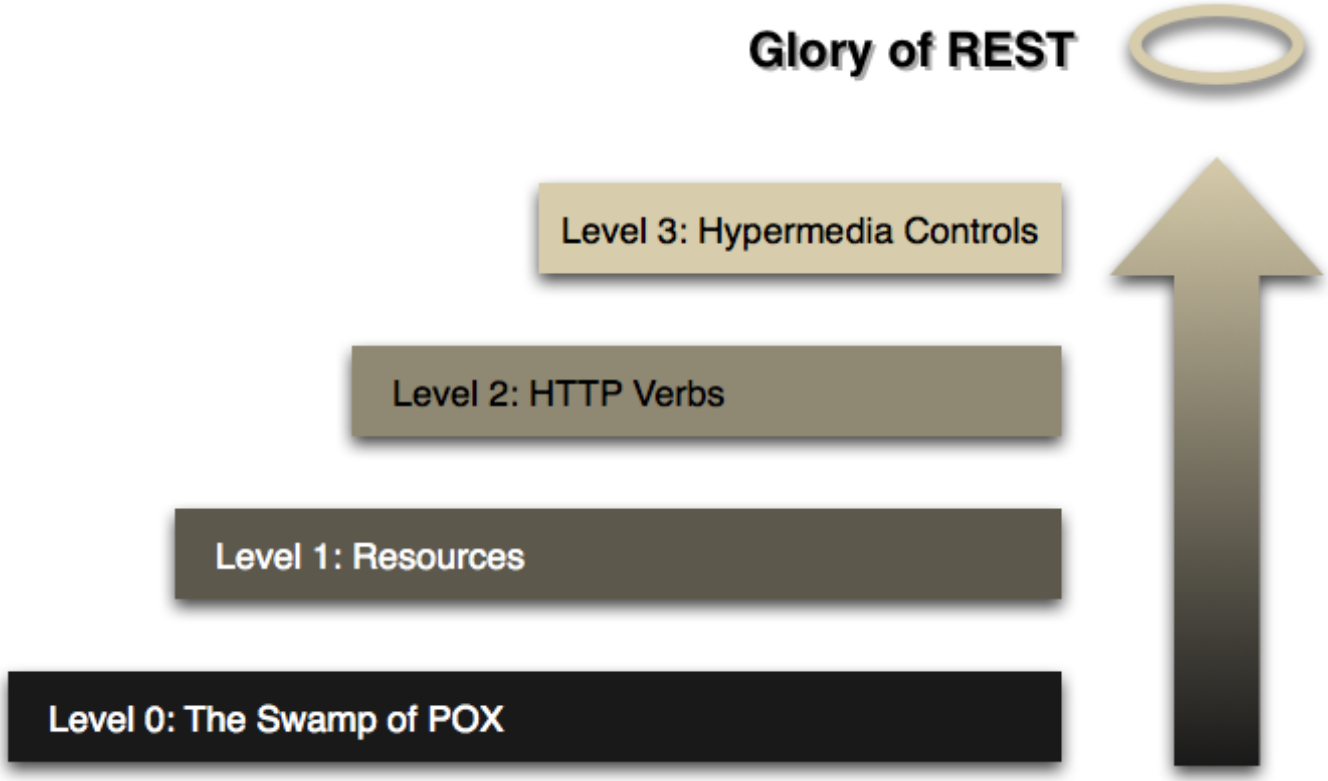
DELETE http://.../200102/aircon/3 (移除3號冷氣)

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 (HATEOAS)

Web Friendliness

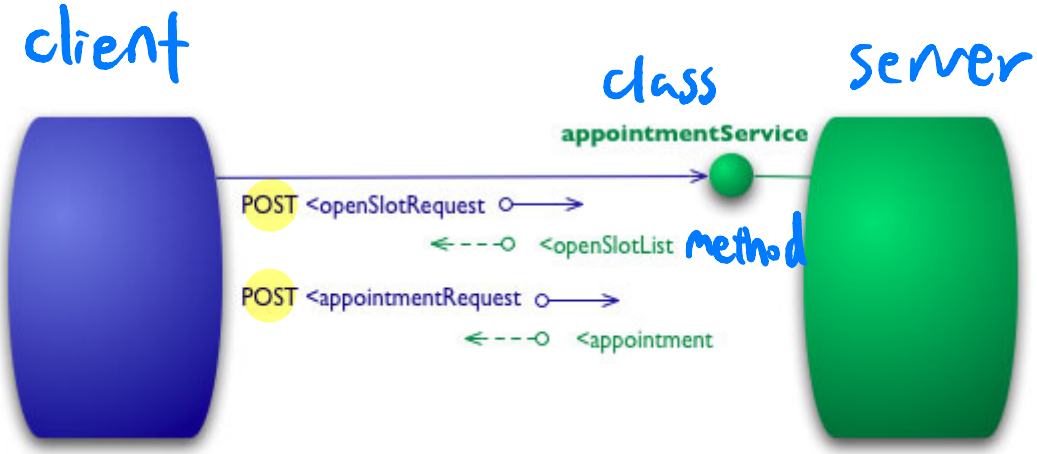
- Richardson Maturity Model (RMM)



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

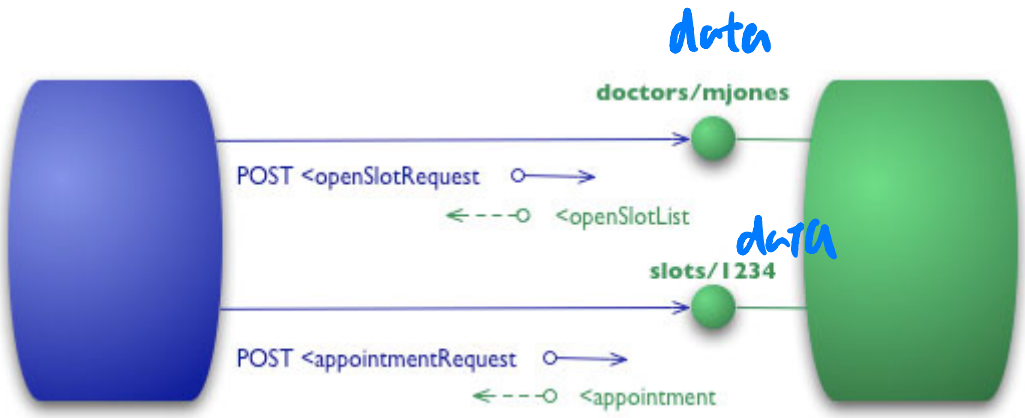
Level 0 POX

- 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



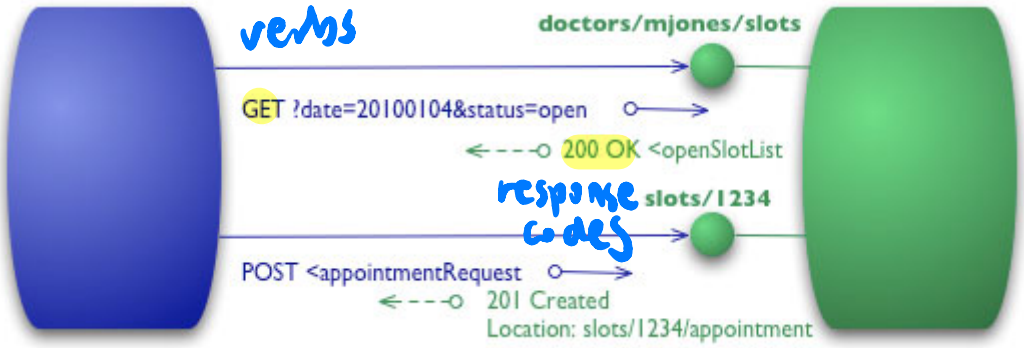
Level 1 Resources

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



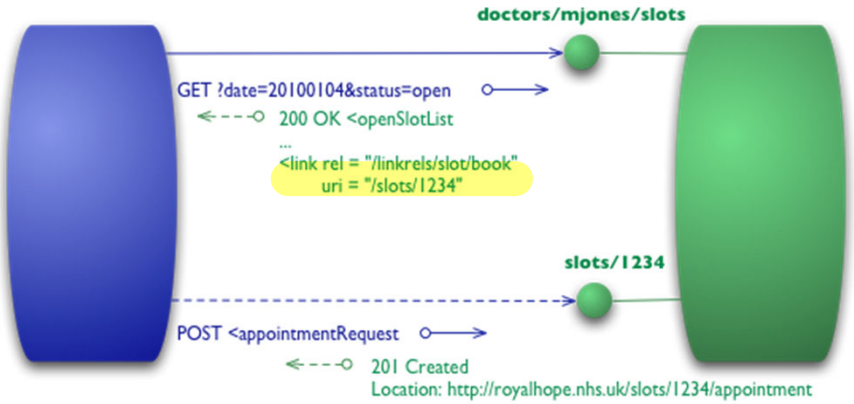
Level 2 HTTP Verbs

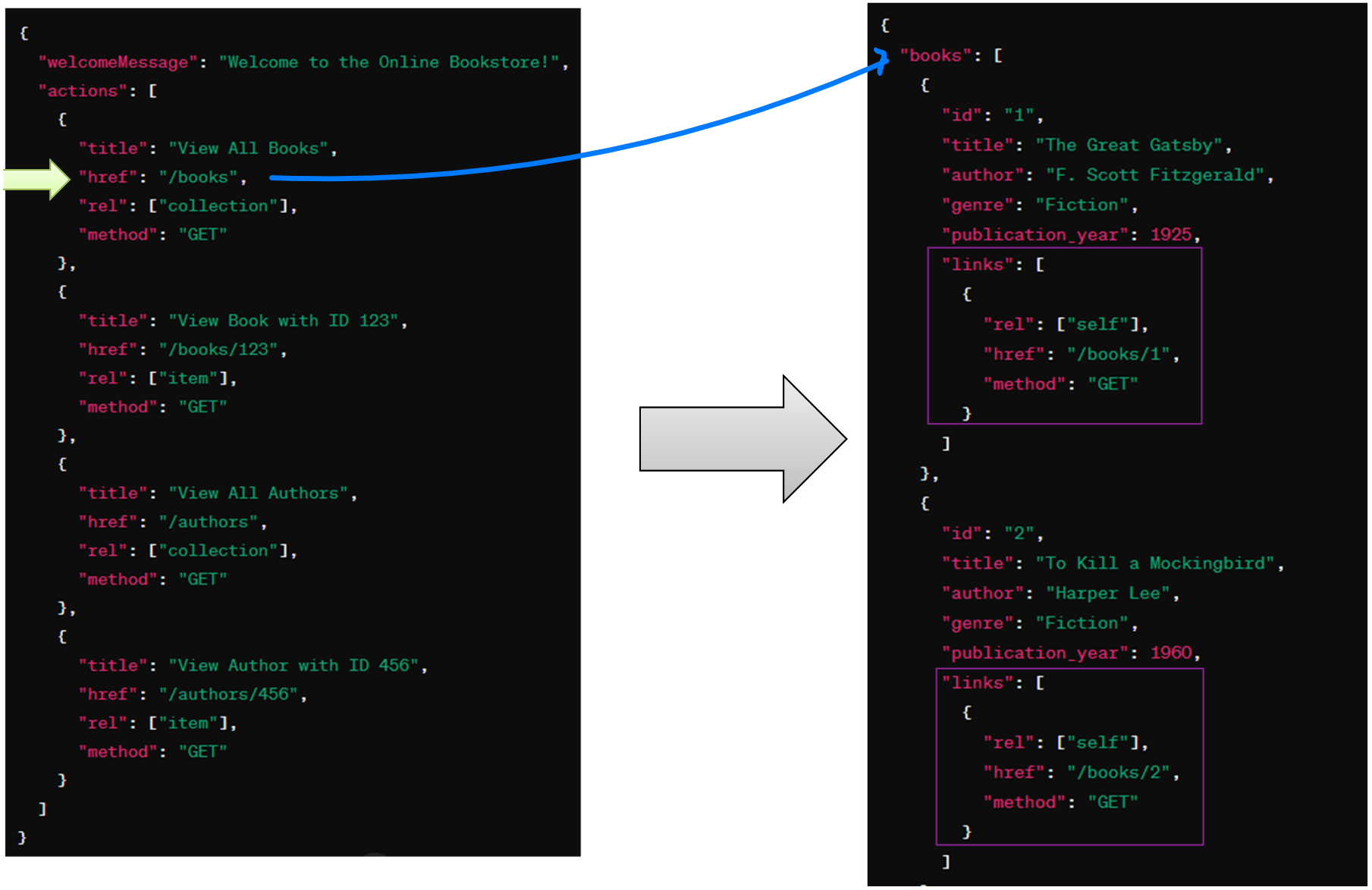
- Follows the semantics of HTTP Verbs
  - Using the HTTP verbs as closely as possible to how they are used in HTTP itself
- Follows the convention of HTTP response codes



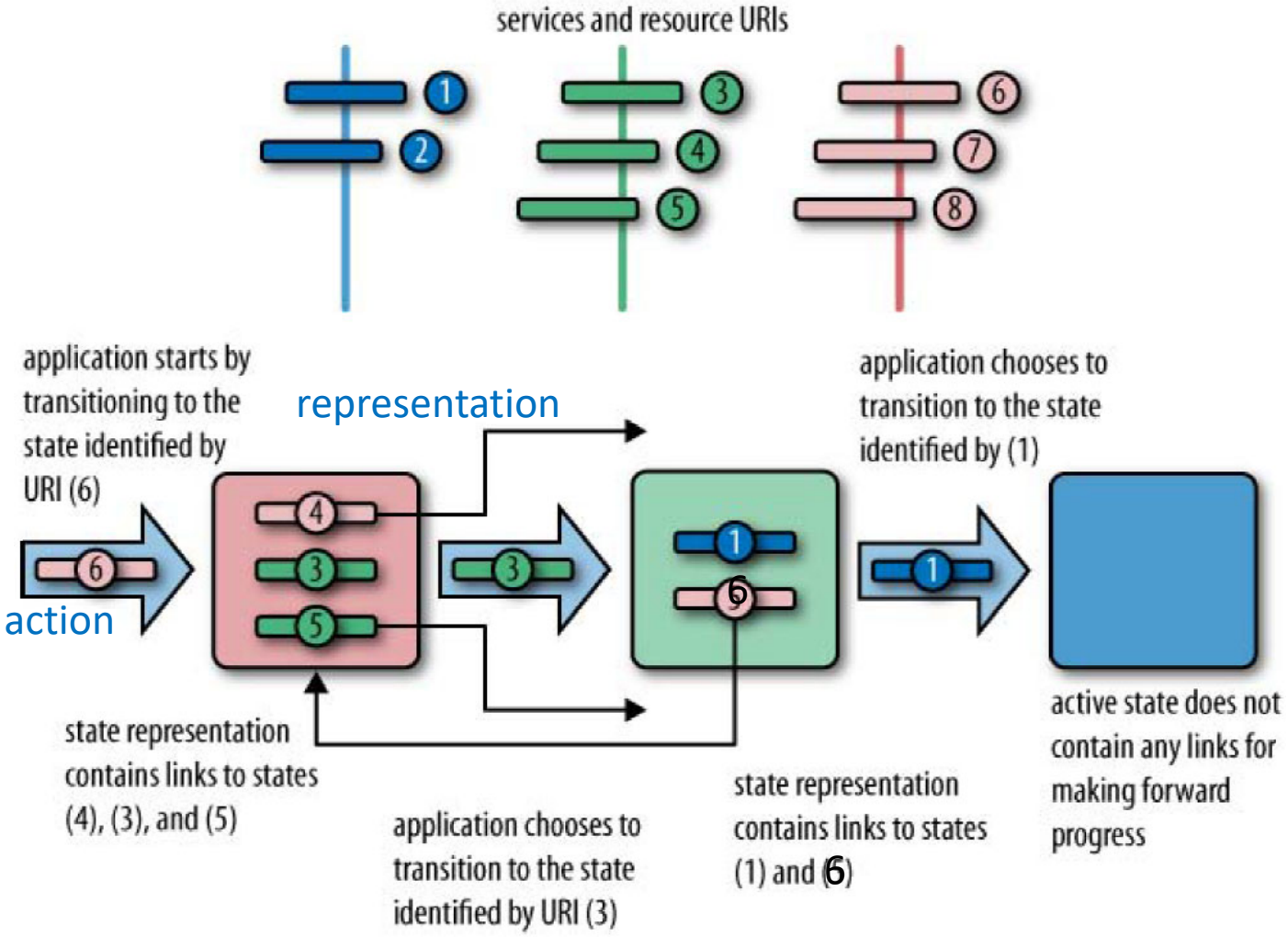
Level 3 HATEOAS

- HATEOAS (~~Hypertext~~ Hypermedia As The Engine Of Application State)
  - The application driven by self-descriptive links in the representation
  - Implications
    - Server states: presence / absence of links
    - Runtime discovery of possible operations
      - As the application reaches a new state, the next possible transitions are discovered (as the returned links)
    - Application makes progress by transitioning from one state to another




{
"welcomeMessage": "Welcome to the Online Bookstore!",
"actions": [
{
"title": "View All Books",
"href": "/books",
"rel": ["collection"],
"method": "GET"
},
{
"title": "View Book with ID 123",
"href": "/books/123",
"rel": ["item"],
"method": "GET"
},
{
"title": "View All Authors",
"href": "/authors",
"rel": ["collection"],
"method": "GET"
},
{
"title": "View Author with ID 456",
"href": "/authors/456",
"rel": ["item"],
"method": "GET"
}
]
}
{
"books": [
{
"id": "1",
"title": "The Great Gatsby",
"author": "F. Scott Fitzgerald",
"genre": "Fiction",
"publication_year": 1925,
"links": [
{
"rel": ["self"],
"href": "/books/1",
"method": "GET"
}
]
},
{
"id": "2",
"title": "To Kill a Mockingbird",
"author": "Harper Lee",
"genre": "Fiction",
"publication_year": 1960,
"links": [
{
"rel": ["self"],
"href": "/books/2",
"method": "GET"
}
]

HATEOAS


services and resource URIs
1
2
3
4
5
6
7
8
application starts by transitioning to the state identified by URI (6)
representation
application chooses to transition to the state identified by (1)
6
action
4
3
5
3
1
6
1
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)

REST API 相關標準

- Microsoft REST API Guidelines
  - https://github.com/microsoft/api-guidelines/blob/vNext/Guidelines.md
- PII (personally identifiable information) parameters
  - SHOULD NOT transmit PII parameters in the URL
    - inadvertently exposed via client, network, and server logs
  - SHOULD accept PII parameters transmitted as headers

REST API 相關標準

- Pagination : the process of separating a print of digital content into discrete pages
  - Server-driven paging
    - indicate a partial result by including a continuation token in the response
  - Client-driven paging
    - use $top and $skip query parameters to specify a number of results to return and an offset into the collection.

```
GET http://api.contoso.com/v1.0/people HTTP/1.1
Accept: application/json

HTTP/1.1 200 OK
Content-Type: application/json

{
  ...,
  "value": [...],
  "@nextLink": "{opaqueUrl}"
}
```

server-driven paging

```
GET http://api.contoso.com/v1.0/people?$top=5&$skip=2 HTTP/1.1
Accept: application/json

HTTP/1.1 200 OK
Content-Type: application/json

{
  ...,
  "value": [...]
}
```

client-driven paging

REST API 相關標準

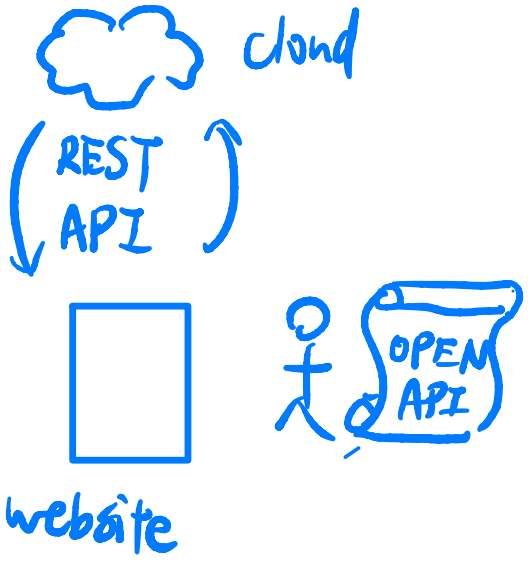
- Filter operations
  - GET https://api.contoso.com/v1.0/products?$filter=price lt 10.00

Operator	Description	Example
Comparison Operators		
eq	Equal	city eq 'Redmond'
ne	Not equal	city ne 'London'
gt	Greater than	price gt 20
ge	Greater than or equal	price ge 10
lt	Less than	price lt 20
le	Less than or equal	price le 100
Logical Operators		
and	Logical and	price le 200 and price gt 3.5
or	Logical or	price le 3.5 or price gt 200
not	Logical negation	not price le 3.5

REST API 相關標準

- OpenAPI Specification (OAS)
  - OpenAPI Initiative (a Linux Foundation Collaborative Project)
  - Defines a standard, programming language-agnostic interface description for HTTP APIs
  - RI: Swagger
- 功能
  - Code generation
  - Validation
  - Interactive documentation
    - Examples and Mocking

Swagger PetStore https://petstore.swagger.io/

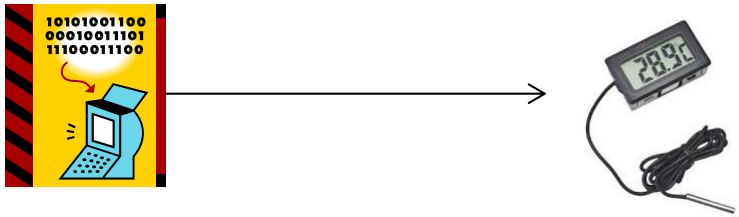


GraphQL

- A query language for your API (SQL for REST)
  - describe the capabilities and requirements of data models for the company's client/ server applications
- Origin
  - 2012 FB decided to rebuild the native FB app
    - FQL + RESTful servers
    - Performance was struggling and the apps often crashed
  - 2015 Lee Byron, Nick Schrock, and Dan Schafer form a team to build GraphQL
  - 2016 first GraphQL technical preview

React.js Conf 2015 - Data fetching for React applications at Facebook
https://www.youtube.com/watch?v=9sc8Pyc51uU

問題: REST需要更精緻的查詢機制



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

(overfetching: 資訊只能一次全要)

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

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

(underfetching: 不然就必須一個一個拿)

```
{
  type: “temperature”
}
```

```
{
  value: 28.9
}
```

GraphQL Example

.gql

```
{
  person(personID: 1) {
    name
    birthYear
    created
  }
}
```

```
{
  "data": {
    "person": {
      "name": "Luke Skywalker",
      "birthYear": "19BBY",
      "created": "2014-12-09T13:50:51.644000Z"
    }
  }
}
```

https://graphql.org/swapi-graphql/

Users

- Github
  - V4 of its public API uses GraphQL
  - “the ability to define precisely the data you want-and only the data you want-is a powerful advantage over the API v3”

- Proxy: "扮演" client
- Reverse Proxy: "扮演" server

API Gateway

- 定義
  - A management tool that sits at the edge of a system between a consumer and a collection of backend services and acts as a single point of entry for a defined group of APIs
- 主要功能
  - Cross-cutting concerns for APIs
    - Authentication
    - Observability (metrics, logs, traces)
    - Request rate limiting
    - Lifecycle of API

Feature	Reverse proxy	Load balancer	API gateway
Single Backend	*	*	*
TIS/SSL	*	*	*
Multiple Backends		*	*
Service Discovery		*	*
API Composition			*
Authorization			*
Retry Logic			*
Rate Limiting			*
Logging and Tracing			*
Circuit Breaking			*

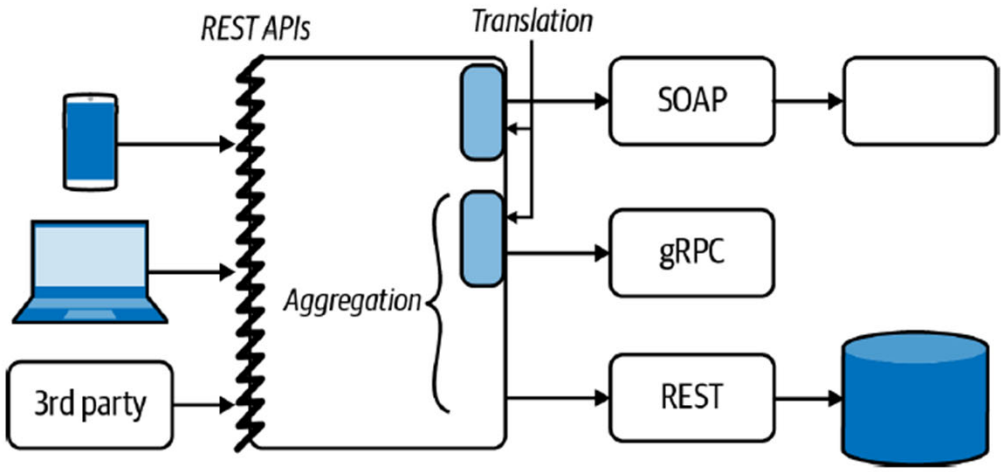
- TLS: Transport Layer Security
- SSL: Secure Sockets Layer

API Gateway

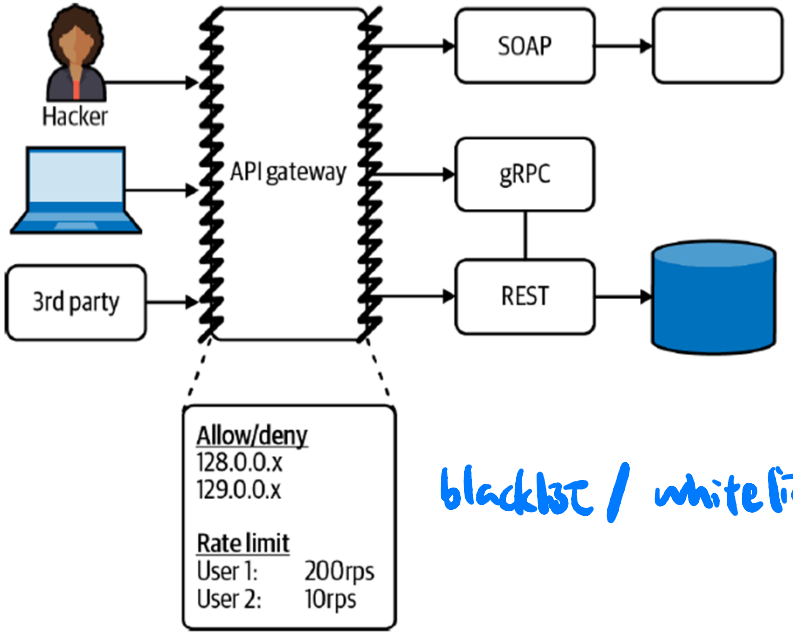
- 主要功能(細節)
  - Reducing coupling
    - As an adapter/facade between frontends and backends
  - Simplifying consumption
    - Aggregating/translating backend services
  - Protecting
    - Prevent APIs from overuse and abuse with threat detection and mitigation
  - Observability
    - Understanding how APIs are being consumed
  - Management
    - API lifecycle management
    - Account management, billing, and pay

HTTP

Consumer

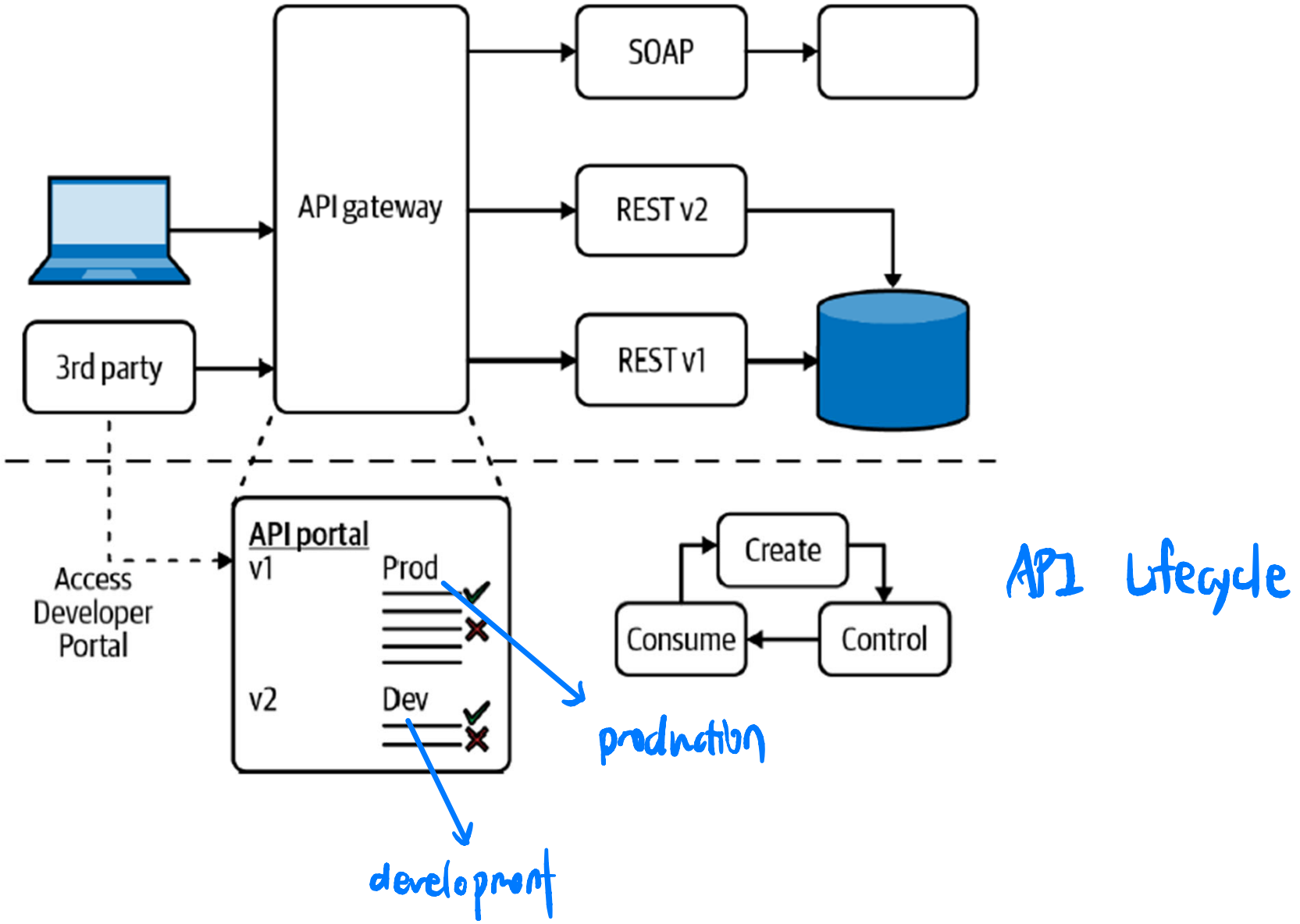


backend



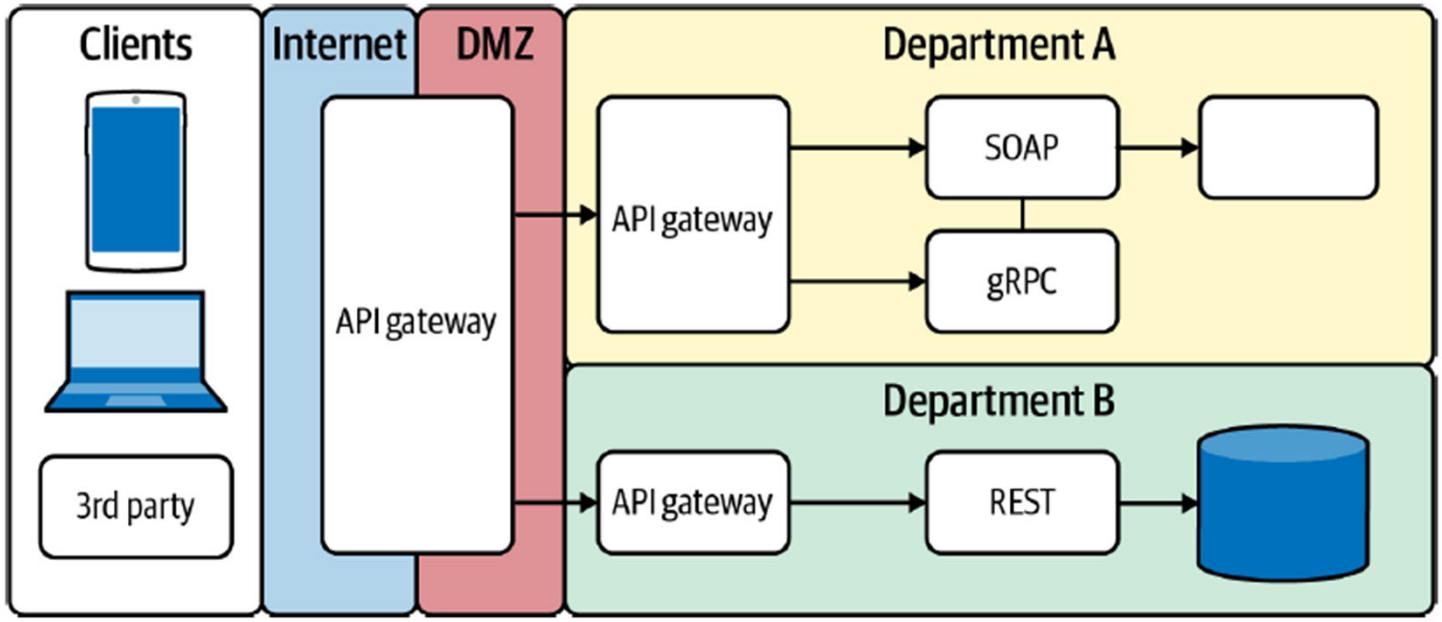
blacklist / whitelist

Life-cycle Management



API Gateway

- Where Is an API Gateway Deployed



Demilitarized Zone (DMZ): An area where firewall protection is forbidden.

API Gateway的演進

- 1990s Onward: Hardware Load Balancers
  - Basic health checks and dynamic routing
- Early 2000s Onward: Software Load Balancers
  - HAProxy and NGINX → web application firewall
  - CDN and WAF began to see increasing adoption
    ↳ content delivery network
  - ModSecurity Project: https://github.com/SpiderLabs/ModSecurity (open source WAF)
- Mid-2000s: Application Delivery Controllers (ADCs)
  - Proposed by F5 Networks, Citrix, and Cisco
  - Deliver SSL, media rich data payload
  - Different priority requests

API Gateway的演進

- Early 2010s: First-Generation API Gateways
  - Kong API Gateway: based on OpenResty + NGINX
  - WSO2: Cloud Services Gateway
  - RedHat: 3Scale
- 特色
  - Focus on Layer 7 (Application, HTTP) processing

Kong

Kong Gateway

3scale BY RED HAT

https://openresty.org/en/

API Gateway的演進

- 2015 Onward: Second-Generation API Gateways
  - Cloud native and microservices trends
  - Netflix: Zuul
  - Envoy Proxy
  - Traefik

- 特色
  - More focus on cross-cutting functions
  - Design to work with microservice and Kubernetes platform

- Future
  - Gateway for Service Mesh
  - Service Mesh將於後續SOA/Microservice章節詳述

Summary

Use case	Traditional enterprise API gateway	Microservices API gateway	Service mesh gateway
Primary Purpose	Expose, compose, and manage internal business APIs and associated services.	Expose, compose, and manage internal business services.	Expose internal services within the mesh.
Publishing Functionality	API management team or service team registers/updates gateway via admin API (in mature organizations this is achieved through delivery pipelines).	Service team registers/ updates gateway via declarative code as part of the deployment process.	Service team registers/ updates mesh and gateway via declarative code as part of the deployment process.
Monitoring	Admin and operations focused, e.g., meter API calls per consumer, report errors (e.g., internal 5XX).	Developer focused, e.g., latency, traffic, errors, saturation.	Platform focused, e.g., utilization, saturation, errors.

偏重SRE相關資訊

site reliability engineering

偏重開發相關資訊

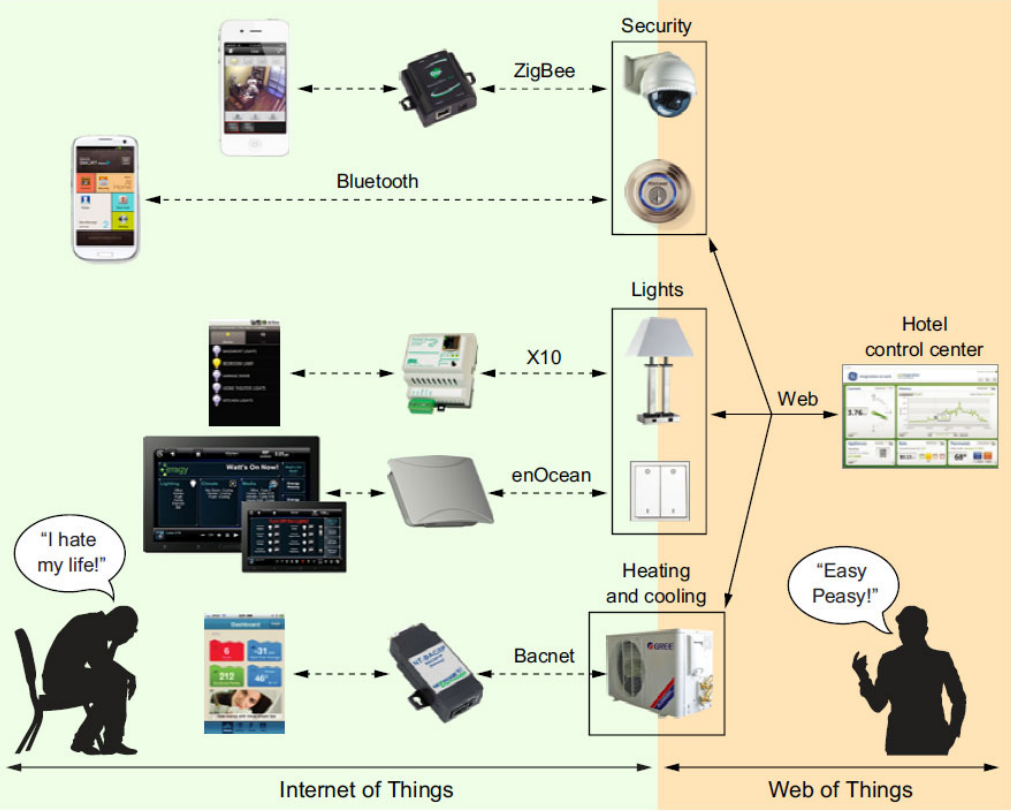
偏重平台本身資訊

Use case	Traditional enterprise API gateway	Microservices API gateway	Service mesh gateway
Handling and Debugging Issues	L7 error-handling (e.g., custom error page). For troubleshooting, run gateway/API with additional logging and debug issue in staging environment.	L7 error-handling (e.g., custom error page, failover, or payload). For debugging issues configure more detailed monitoring, and enable traffic shadowing and/or canarying to re-create the problem.	L7 error-handling (e.g., custom error page or payload). For troubleshooting, configure more detailed monitoring and/or utilize traffic "tapping" to view and debug specific service-to-service communication.
Testing	Operate multiple environments for QA, staging, and production. Automated integration testing, and gated API deployment. Use consumer-driven API versioning for compatibility and stability (e.g., semver).	Enables canary routing and dark launching for dynamic testing. Use contract testing for upgrade management.	Facilitate canary routing for dynamic testing.
Local Development	Deploy gateway locally (via installation script, Vagrant, or Docker), and attempt to mitigate infrastructure differences with production. Use language-specific gateway mocking and stubbing frameworks.	Deploy gateway locally via service orchestration platform (e.g., container, or Kubernetes).	Deploy service mesh locally via service orchestration platform (e.g., Kubernetes).
User Experience	Web-based administration UI, developer portal, and service catalog.	IaC or CLI-driven, with simple developer portal and service catalog.	IaC or CLI-driven, with limited service catalog.

named after the "canary in a coal mine" concept

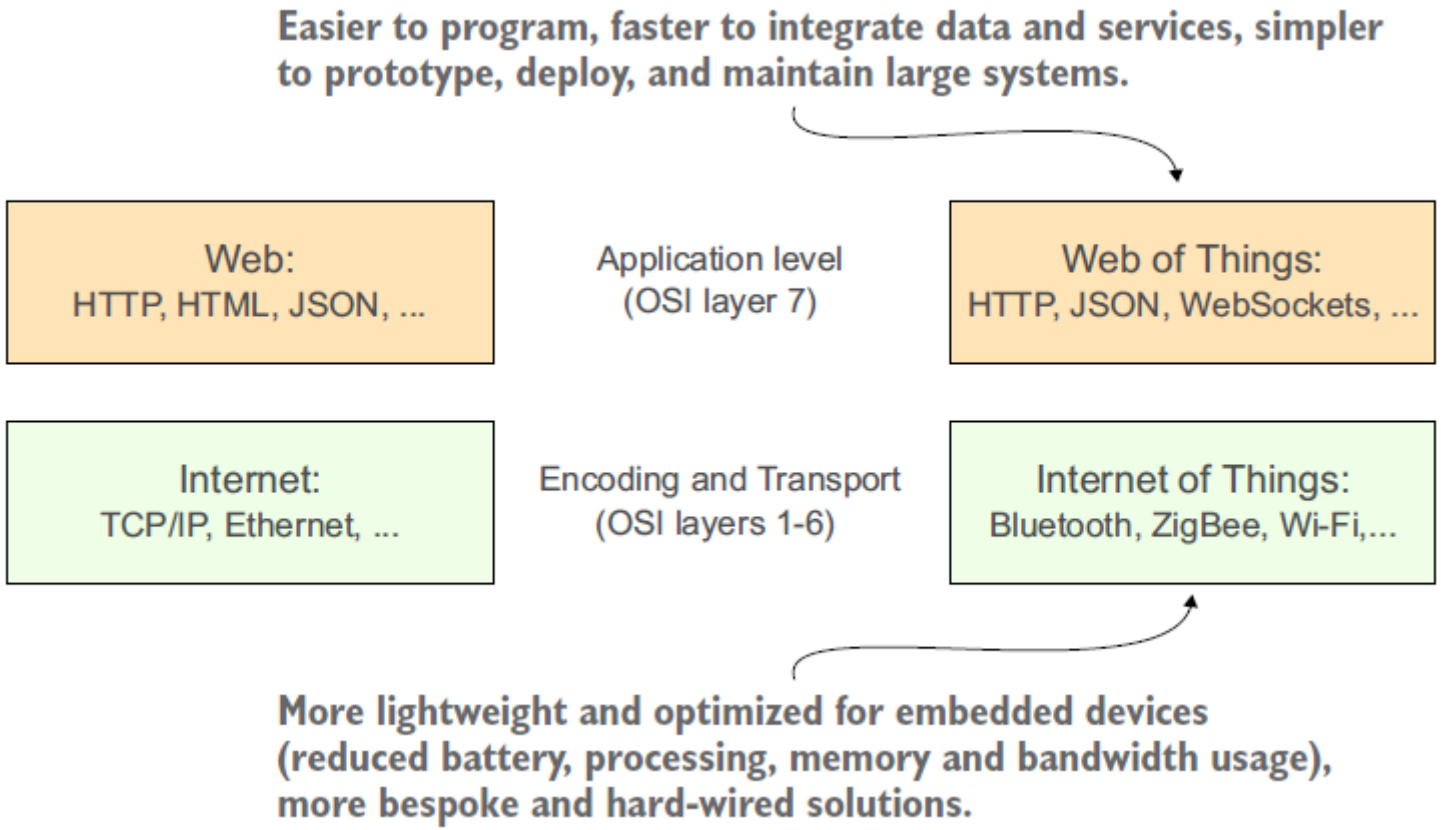
Case: Web of Things (WoT)

- 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?



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

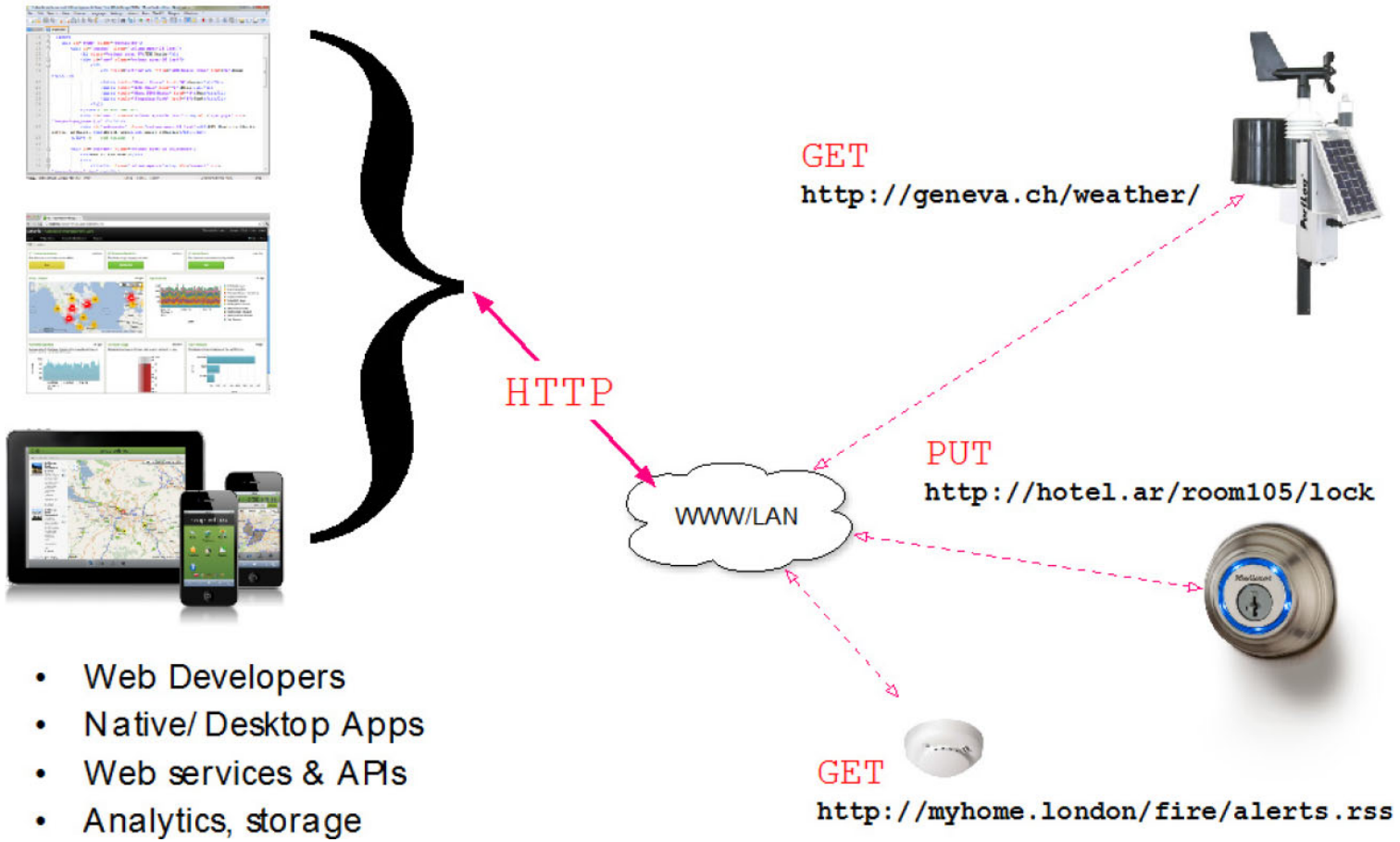
IoT and WoT



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

(a) made specifically for a particular person, organization or purpose

透過Web技術操作所有聯網裝置



- Web Developers
- Native/ Desktop Apps
- Web services & APIs
- Analytics, storage

Benefits of WoT

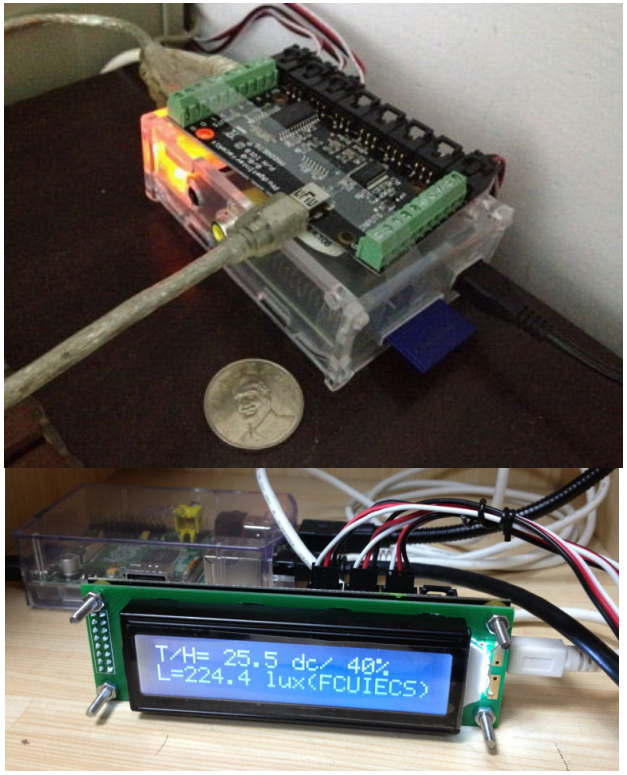
- Easier to program
- Open and extensible standards
- Easy to deploy, maintain, and integrate
- Loose coupling among things
- Widely used security and privacy mechanisms

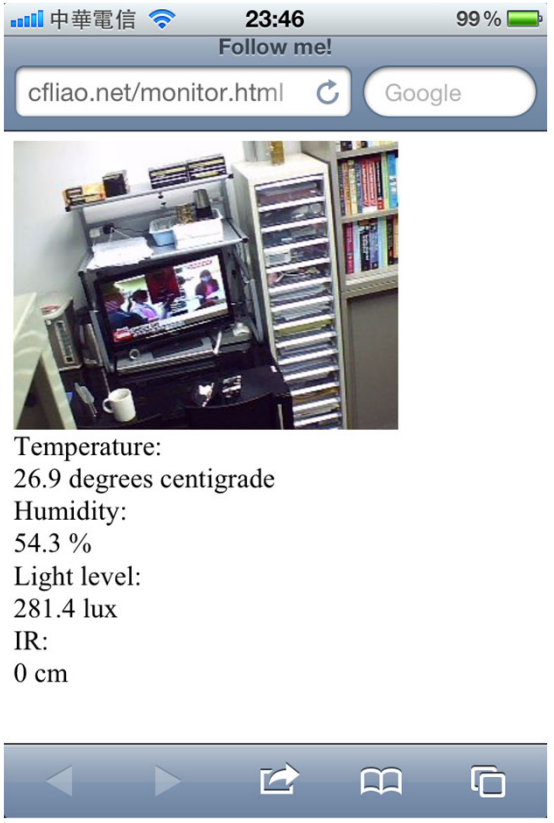
D.D. Guinard & V. M. Trifa, Building the Web of Things, Manning, 2015

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。

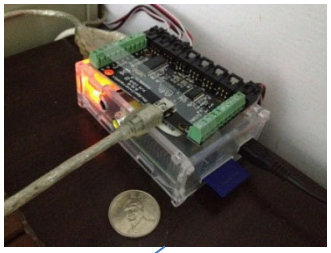




基於Phidget Interface Kit、Raspberry Pi 的硬體，在Raspbian Embedded Linux上使用JAX-RS實現的sMAP架構

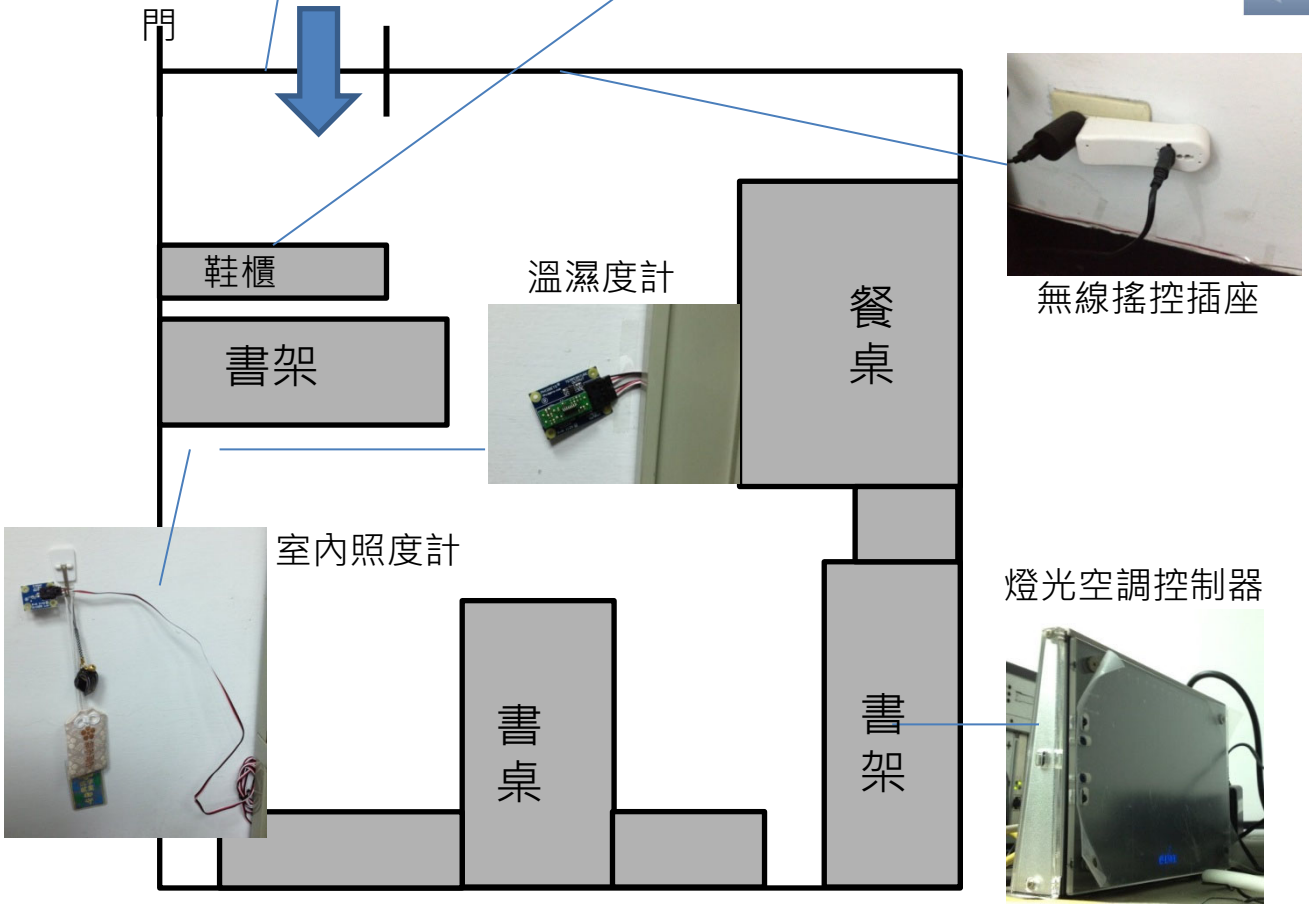
60cm 紅外線測距器

sMAP 小盒



無線搖控燈座

Mobile Sensor Web



無線搖控插座

燈光空調控制器

應用

- Mobile Sensor Web
- 門鈴
- 燈控
- 溫度調控
- 聲控服務
- 防盜監控

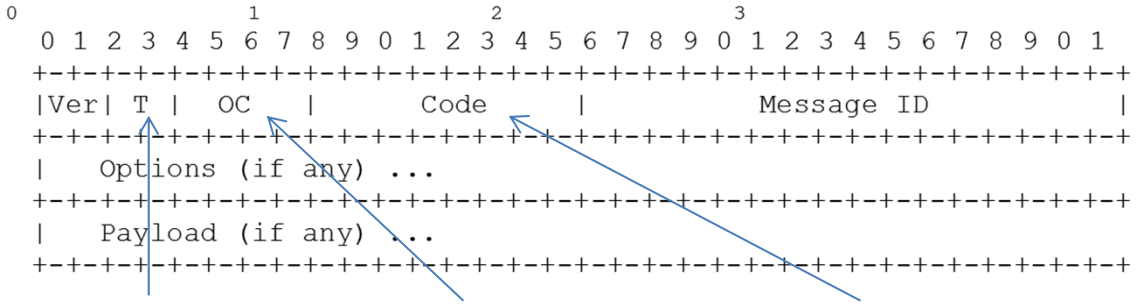
CoAP : HTTP for Constrained Network

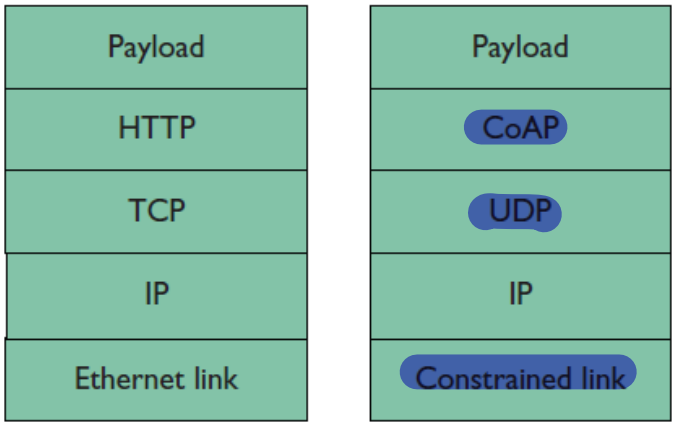
- CoAP (Constrained Application Protocol)
  - 基於REST架構，專為資源受限裝置所訂定之通訊協定
  - IETF CoRE小組所制定
    - CoRE = Constrained RESTful Environments
  - 約略可類比為是UDP上的HTTP

Constrained networks

Internet Engineering Task Force : a standards organization for the Internet

(精簡的 HTTP?)

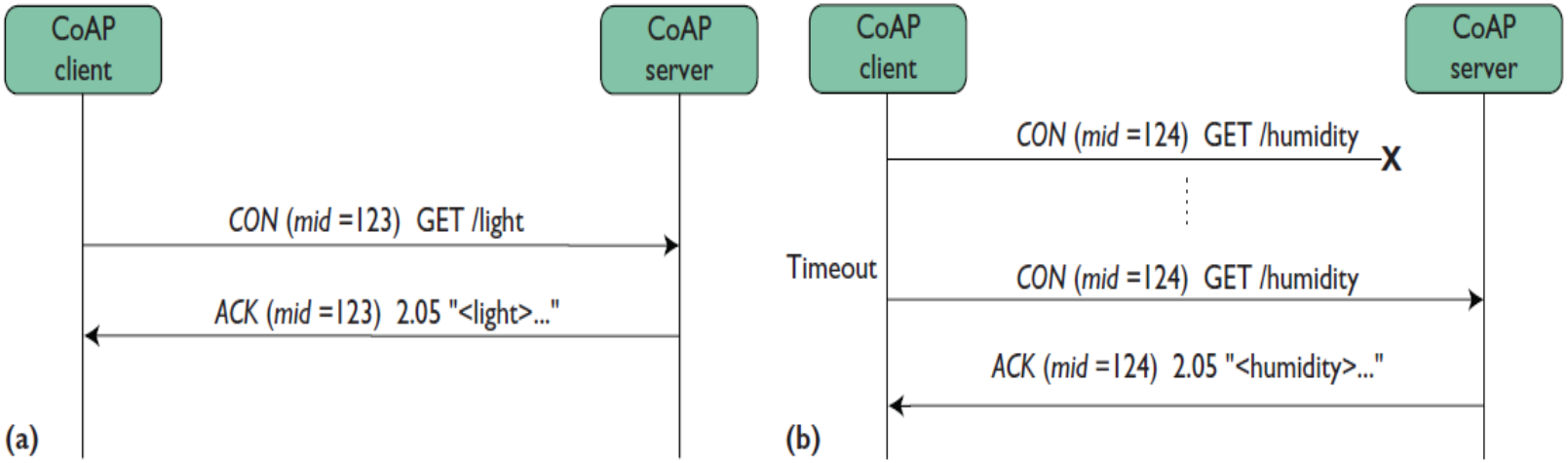




carry requests & responses

CoAP Message Type

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



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/

↗ the process of auto-detecting devices & services on a network. It aims to reduce the manual effort required from users & admins

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/