IST346: Web Services and API’s

Web Services

- The most important service in any organization.
- Beyond a company’s website, other business processes get “webified”
    - Webmail
    - Customer Relationship Portals
    - E-Commerce
- To support these same services outside the browser we “webify” the business logic into an API (Application Program Interface)

But First, The Basics

The World-Wide Web

- Information System on the Internet for displaying content resources.
- The world wide web is not the Internet; it is part of it!
- Built upon open standards

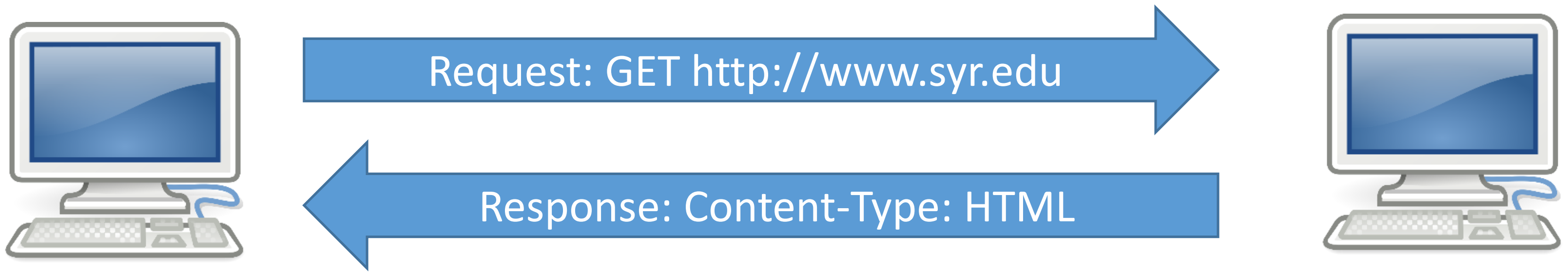
- *INTERESTING STAT*: Half a billion people accessed the internet via their mobile device in 2009!
- *MORE INTERESTING:* 1.2 billion mobile web users in 2011…roughly 17% of the world population!
  http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats#mobile-internet-access

Web Terminology

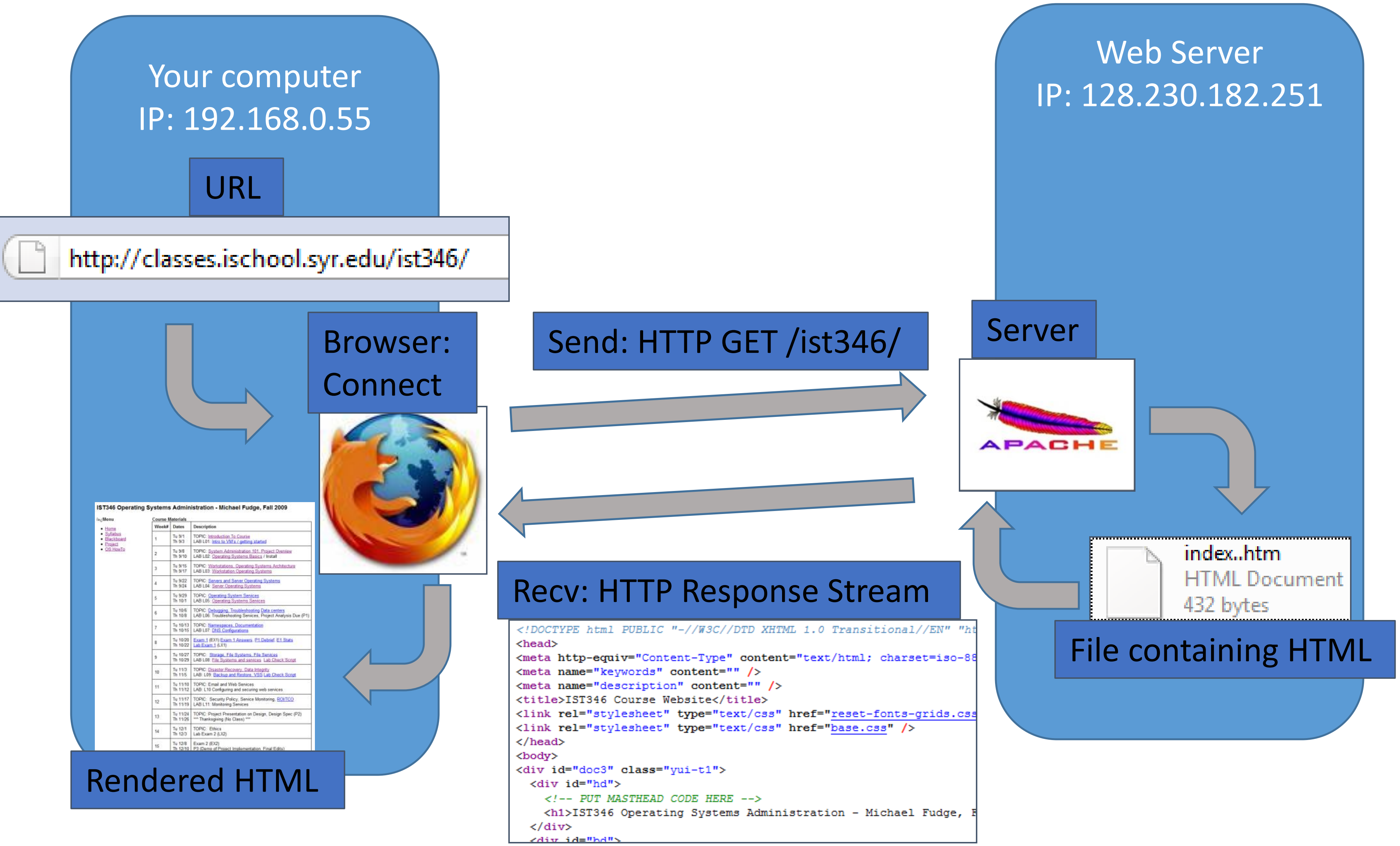
- **URL** – Uniform Resource Locator. A global name space which identifies a resource on the web.
- **HTML** – Hypertext Markup Language. A Markup language for rendering web pages.
- **Web Server** – A computer on the web which hosts resources.
- **Web Browser** – A computer on the web which consumes resources
- **Resource** – content at a URL, hosted on a web server and requested by a web browser.

How the web works

- Clients make requests to web servers, typically using a browser.
  - The client provides a request method and a URL
- The web server send a response to the client. In the response is the content based on the URL
  - The client renders (draws) the content in the web browser



The Web at work: The Details



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "ht
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-88
<meta name="keywords" content="" />
<meta name="description" content="" />
<title>IST346 Course Website</title>
<link rel="stylesheet" type="text/css" href="reset-fonts-grids.css
<link rel="stylesheet" type="text/css" href="base.css" />
</head>
<body>
<div id="doc3" class="yui-t1">
  <div id="hd">
    <!-- PUT MASTHEAD CODE HERE -->
    <h1>IST346 Operating Systems Administration - Michael Fudge, F
  </div>
  <div id="bd">
```

IST346 Operating Systems Administration - Michael Fudge, Fall 2009

ï»¿**Menu**

- Home
- Syllabus
- Blackboard
- Project
- OS HowTo

Course Materials

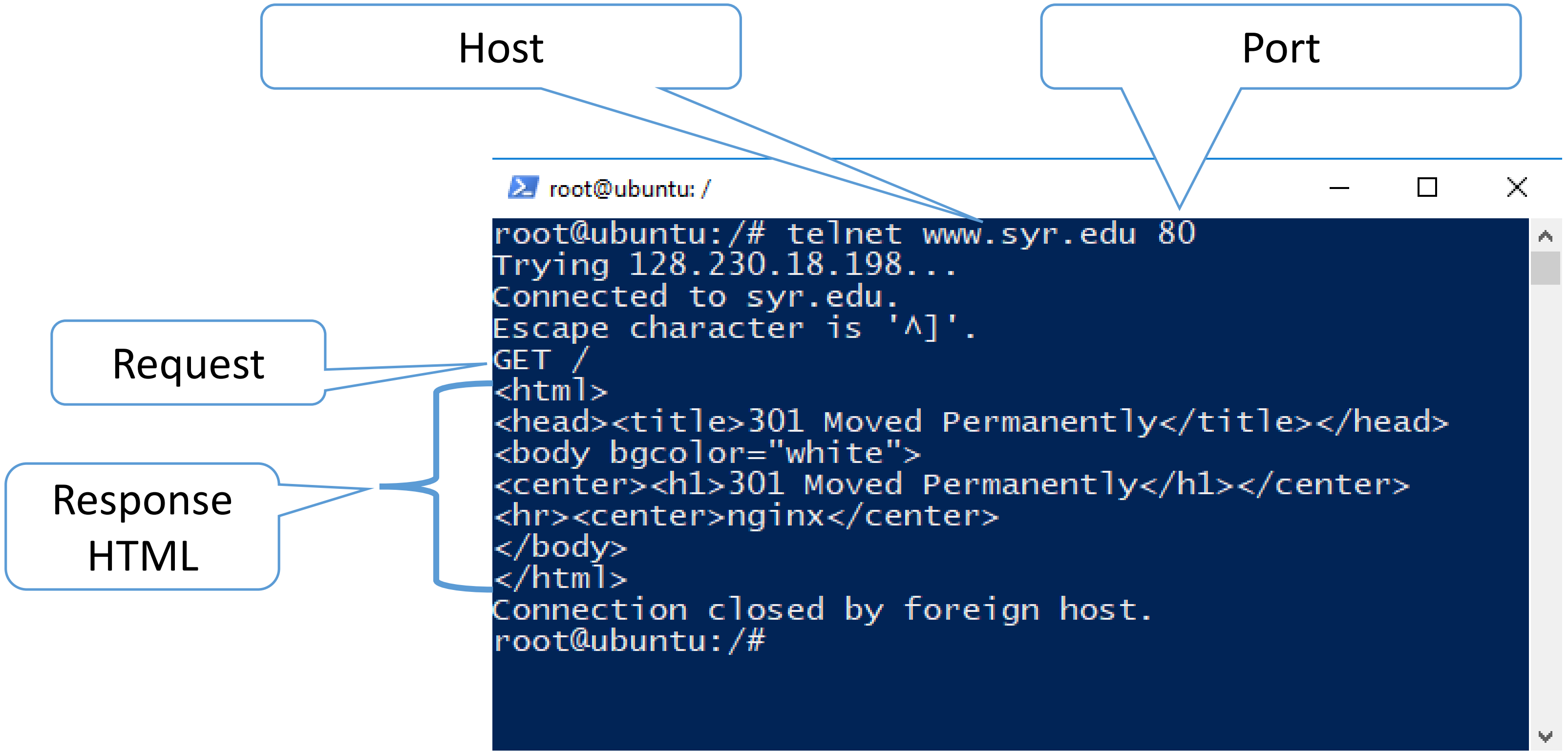
Week#	Dates	Description
1	Tu 9/1 Th 9/3	TOPIC: Introduction To Course LAB L01: Intro to VM's / getting started
2	Tu 9/8 Th 9/10	TOPIC: System Administration 101, Project Overview LAB L02: Operating Systems Basics / Install
3	Tu 9/15 Th 9/17	TOPIC: Workstations, Operating Systems Architecture LAB L03: Workstation Operating Systems
4	Tu 9/22 Th 9/24	TOPIC: Servers and Server Operating Systems LAB L04: Server Operating Systems
5	Tu 9/29 Th 10/1	TOPIC: Operating System Services LAB L05: Operating Systems Services
6	Tu 10/6 Th 10/8	TOPIC: Debugging, Troubleshooting Data centers LAB L06: Troubleshooting Services, Project Analysis Due (P1)
7	Tu 10/13 Th 10/15	TOPIC: Namespaces, Documentation LAB L07: DNS Configurations
8	Tu 10/20 Th 10/22	Exam 1 (EX1) Exam 1 Answers, P1 Debrief, E1 Stats Lab Exam 1 (LX1)
9	Tu 10/27 Th 10/29	TOPIC: Storage, File Systems, File Services LAB L08: File Systems and services Lab Check Script
10	Tu 11/3 Th 11/5	TOPIC: Disaster Recovery, Data Integrity LAB: L09: Backup and Restore, VSS Lab Check Script
11	Tu 11/10 Th 11/12	TOPIC: Email and Web Services LAB: L10 Configuring and securing web services
12	Tu 11/17 Th 11/19	TOPIC: Security Policy, Service Monitoring, ROI/TCO LAB L11: Monitoring Services
13	Tu 11/24 Th 11/26	TOPIC: Project Presentation on Design, Design Spec (P2) *** Thanksgiving (No Class) ***
14	Tu 12/1 Th 12/3	TOPIC: Ethics Lab Exam 2 (LX2)
15	Tu 12/8 Th 12/10	Exam 2 (EX2) P3 (Demo of Project Implementation, Final Edits)

HTTP

- HTTP, or the Hypertext Transport Protocol is the data transfer protocol of the web.
- It consists of requests, which contain a **verb** and **URL** and a response, which contains a **status code** and **content type**.
- HTTP is a stateless protocol, which means the current request knows nothing of the previous requests.
- The well-known port for HTTP is TCP/80

HTTP Protocol in Action

- Like SMTP and IMAP, you can use the HTTP protocol directly with telnet:



```
root@ubuntu:/# telnet www.syr.edu 80
Trying 128.230.18.198...
Connected to syr.edu.
Escape character is '^]'.
GET /
<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx</center>
</body>
</html>
Connection closed by foreign host.
root@ubuntu:/#
```

HTTP Request and Response

HTTP Request

- **Verb** – Nature of the request
- **URL** – The resource to request

HTTP Response

- **Status Code** – What happened?
- **Content Type** – The actual content

HTTP Request Verbs

- GET – Request a resource. Most common
- POST – Add to a resource. Used when sending data to the website, like submitting a form.
- Other Verbs:
- PUT – Update a resource
- DELETE – Remove a resource
- PATCH – Update part of a resource
- HEAD – No Response Body
- OPTIONS – Reserved.

HTTP Response Status Codes

- 1xx – Informational
- 2xx – Success
  - 200 – OK
  - 201 – Created
  - 202 – Accepted
- 3xx – Redirection
  - 301 – Moved Permanently
  - 304 – Not Modified
  - 307 - Redirect
- 4xx – Client error
  - 400 – Bad Request
  - 401 – Unauthorized
  - 403 – Forbidden
  - 404 – Not Found
- 5xx – Server Error
  - 500 – Internal Server Error
  - 501 – Not Implemented
  - 502 – Bad Gateway

HTTP Content Types

- These are Media Types. They instruct the client (usually a browser) what to do with the content.
- text/plain – plan text
- text/html – HTML text
- image/gif – gif image format
- image/jpeg – jpeg image format
- application/json – JSON data format
- application/xml – XML data format
- application/javascript - JavaScript

Web Servers

Serve up static content over HTTP, or execute code and return a response as content. (This is called CGI – Common Gateway Interface)

Popular Web Servers.

- **Apache** – Open source web server. Most Popular.
- **IIS** – Microsoft's web server
- **NGINX** – Engine X Open source webserver, commonly used for:
    - Load balancing
    - Reverse proxies

HTTP Dependent Services

- TCPIP Networking
- DNS (internal and root DNS servers)
  Resolve names like www.google.com to IP addresses

Webmaster vs. Web Administrator

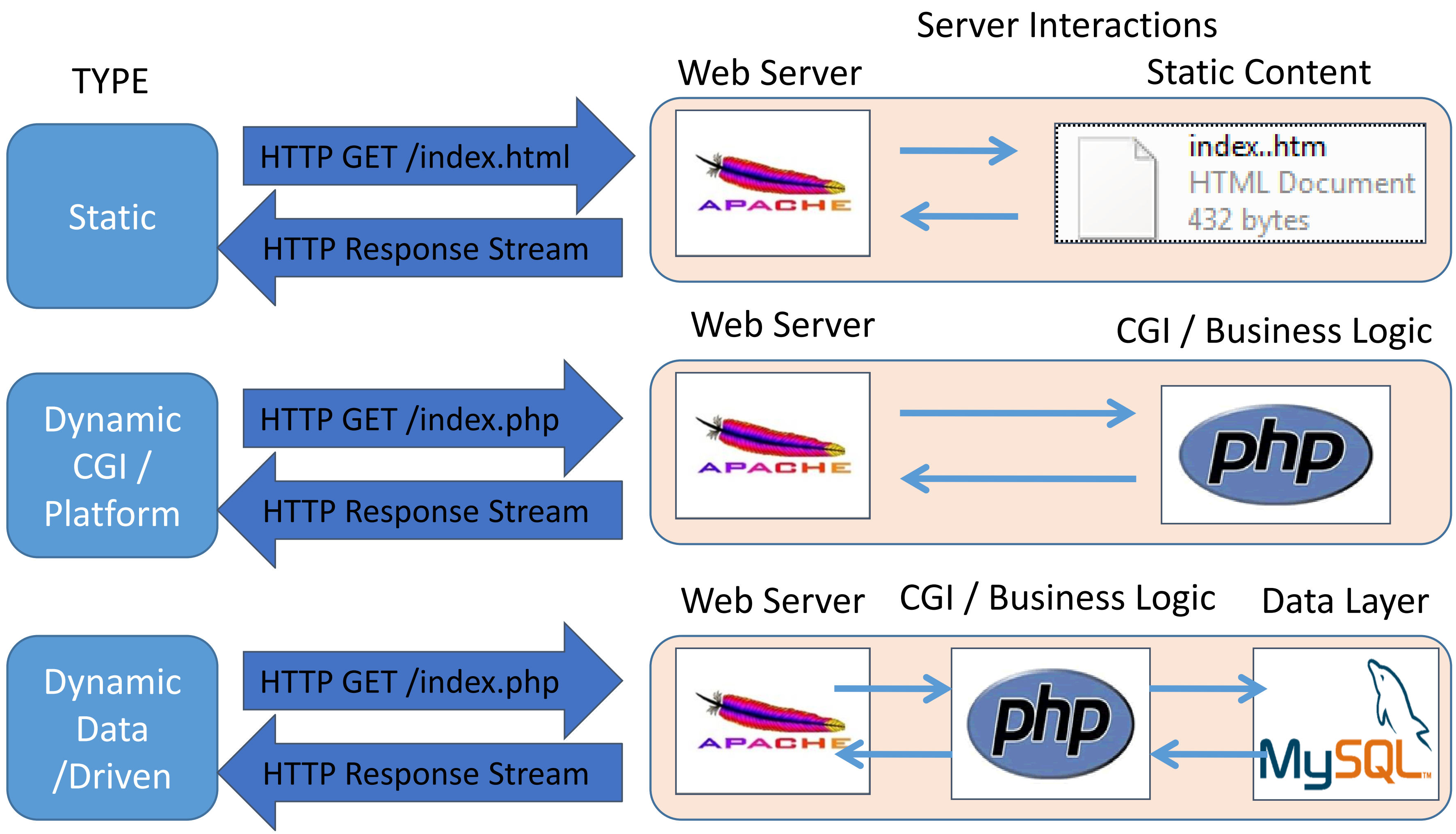
- Two major roles in the web
  - Webmaster (a very outdated term)
    - Person responsible for content, graphics, usability, etc
    - What is classically thought of when creating websites / webpages.
  - Web Administrator
    - Person responsible for administering webserver (machine or VM), create virtual directories, virtual sites, patching, backups, etc.
    - Basic skills required in administering any server
  - Generally the same person for small companies
    - But NOT the same person for midsized or larger companies.

Web servers

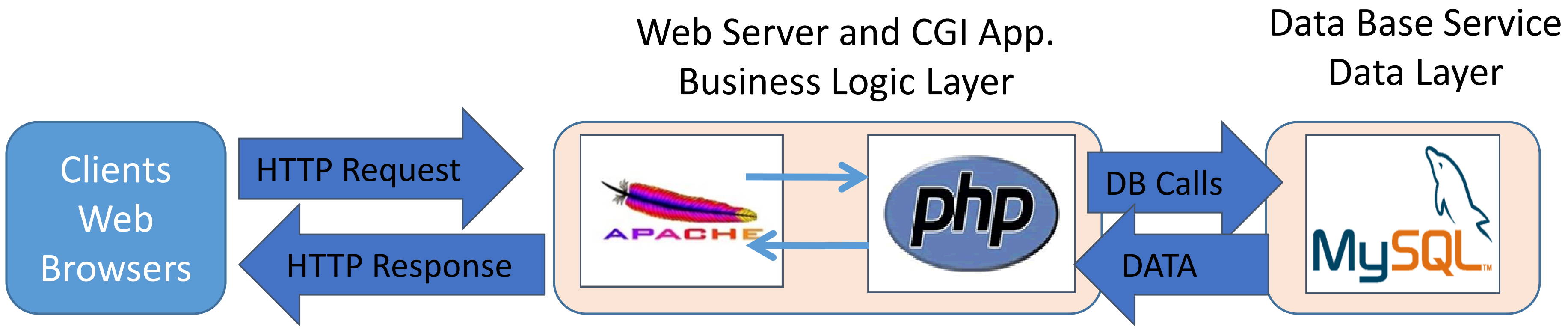
- Commonly used examples are LAMP (linux,apache,mysql,php), IIS (internet information server)
- Can be architected in different ways:
  - Single web server, single website
  - Single web server, multiple websites
    - Multiple TCP ports (80, 81, 8080, 85, etc..)
    - Multiple network interfaces/IP addresses
    - Host header values (multiple IP addresses and DNS records pointing to the same server)
  - Multiple web servers, single website
    - Or better known from previous topics as “horizontal scaling”

Web Architectures

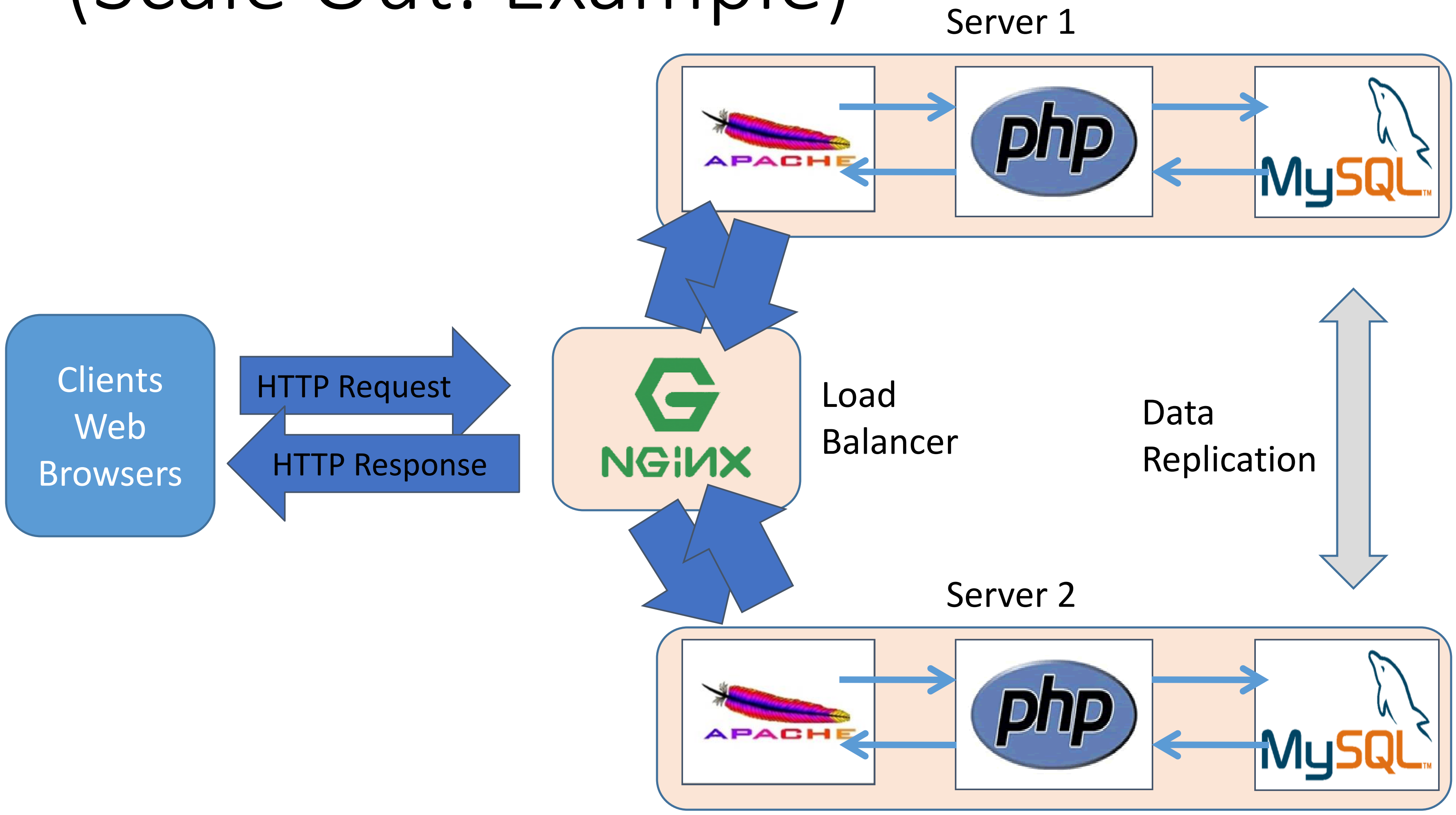
3 Types Of Web Service Architectures



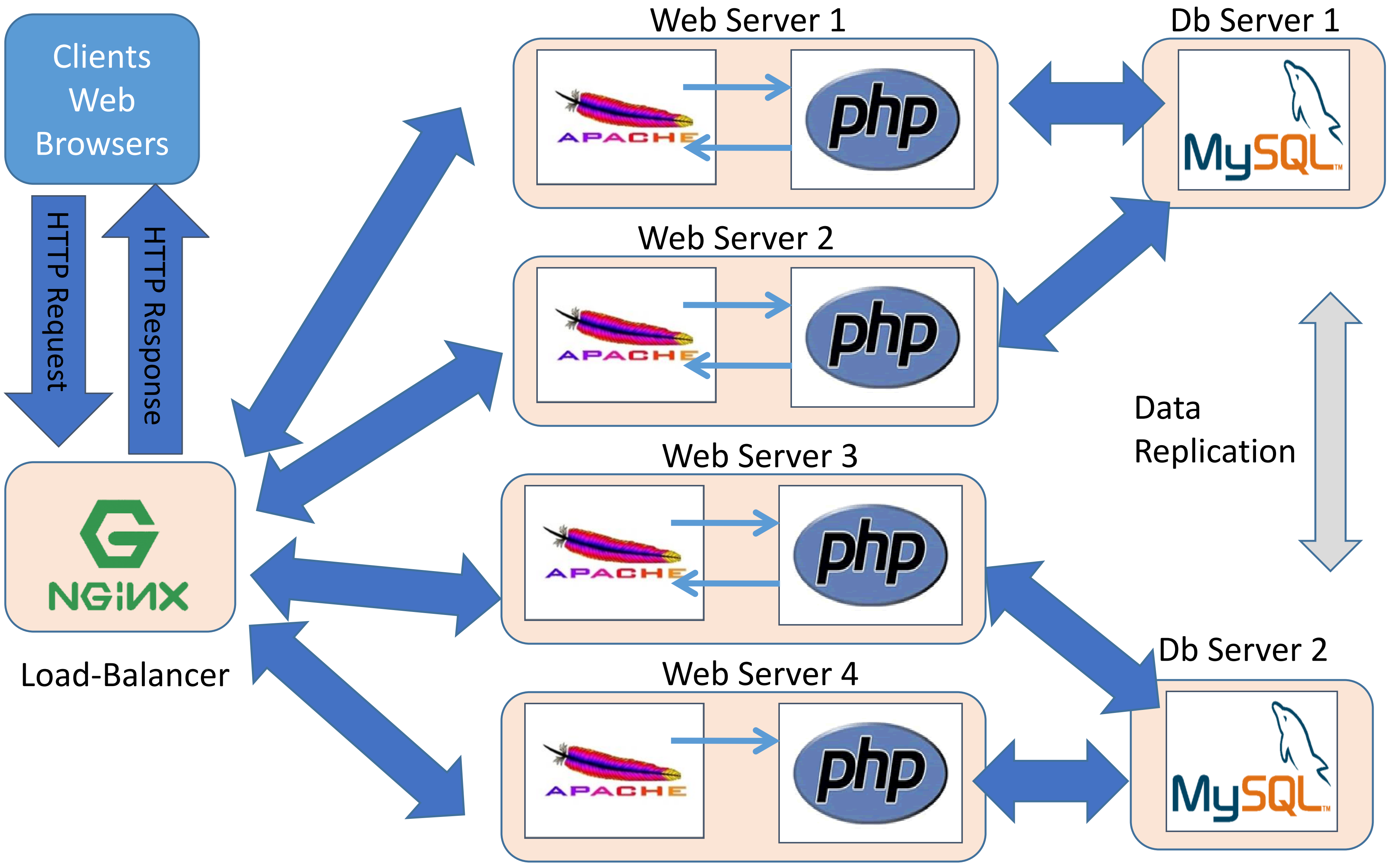
Web Scalability –Vertical (Scale Up Example)



Web Scalability – Horizontal (Scale Out: Example)

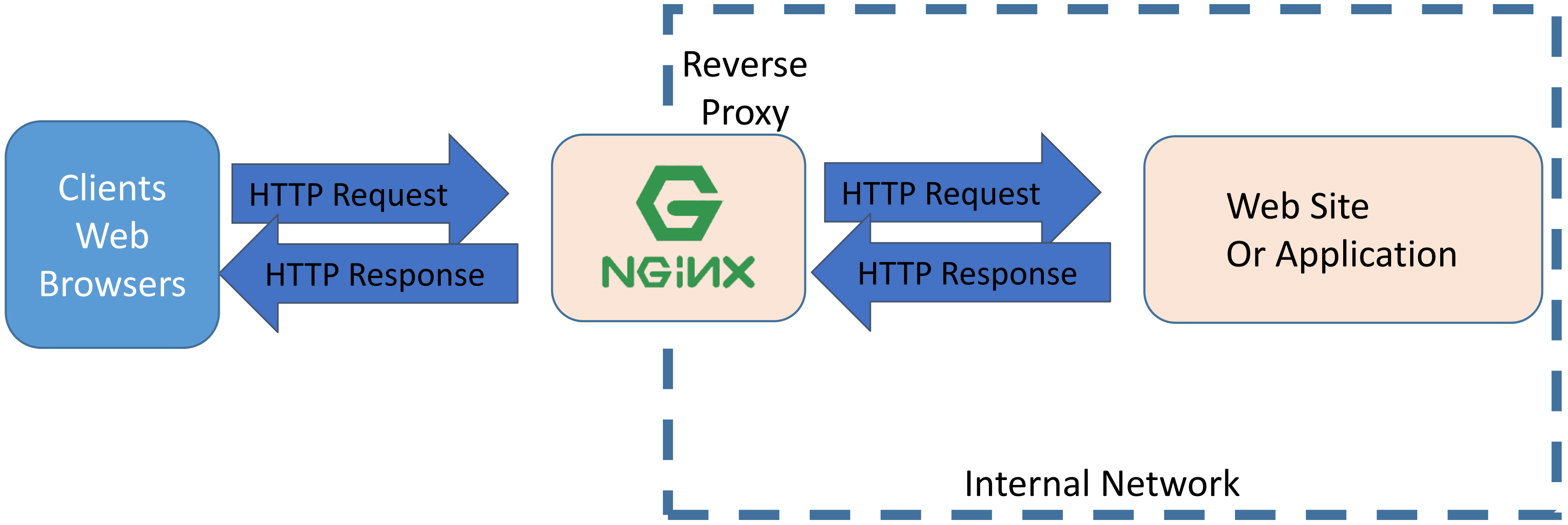


Web Scalability –Up and Out (ex)

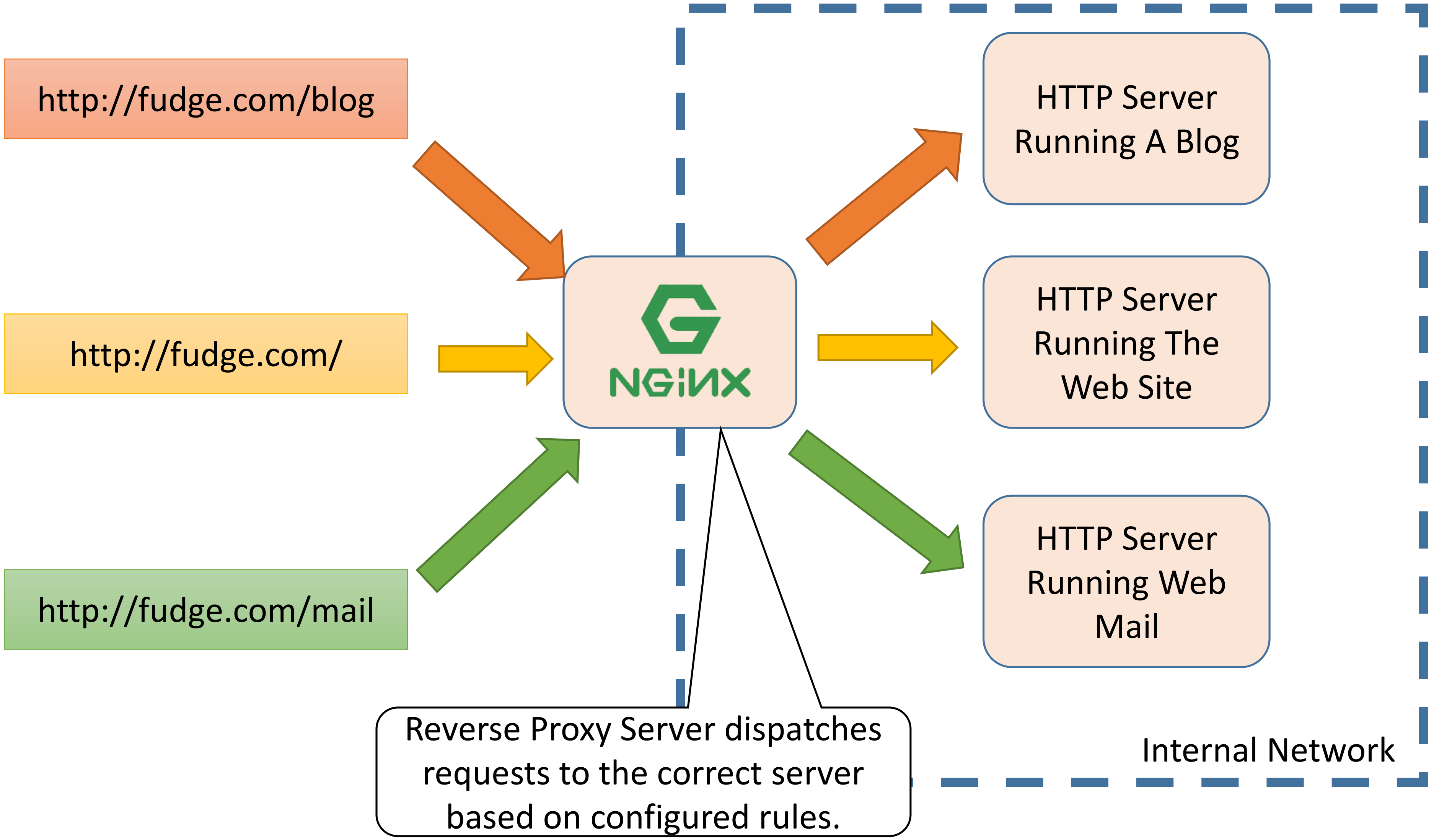


HTTP Reverse Proxy

- An HTTP Reverse Proxy is an HTTP server which retrieves resources from one or more servers on behalf of a client.
- Used to limit exposure of the web application



Reverse Proxies at work


http://fudge.com/blog
http://fudge.com/
http://fudge.com/mail
NGINX
HTTP Server Running A Blog
HTTP Server Running The Web Site
HTTP Server Running Web Mail
Reverse Proxy Server dispatches requests to the correct server based on configured rules.
Internal Network

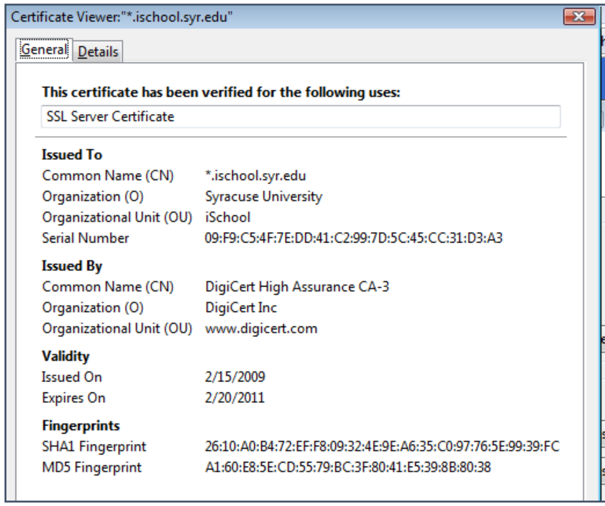
Websites and Security

Web Service Security

- Since virtually everyone can access your service, security is important.
- Rule #1 ALWAYS assume the worst.
- There are many layers of security, use them all:
  - Secure communication with TLS (Transport Layer Security)
  - Protect the server by service Hardening on the Web server. Only run the services that are required – nothing more.
  - Protect the web service itself
  - Secure the application running over the web

TLS – Transport Layer Security

- Encrypts traffic over the wire
- Protects against “Man in the Middle” attacks (sniffing data in transmission)
- Organizations acquire a certificate from an Authority
- Browsers “Trust” the Authority and encrypt the traffic
- **Clients request https:// instead of http:// to get the TLS encrypted site**
- **Moral:** Just because a site uses TLS doesn’t mean its “secure” it only means the traffic between you and the server is encrypted!!!!



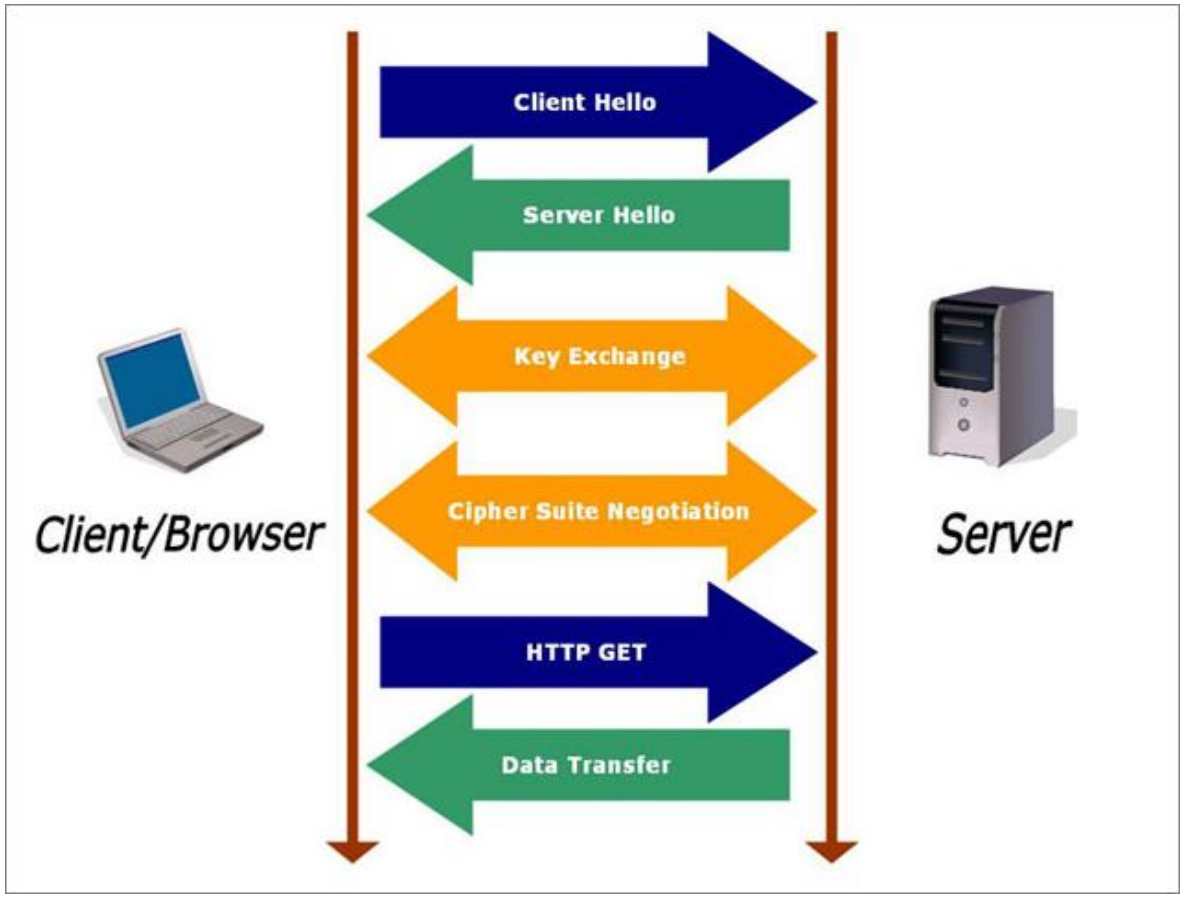
This certificate has been verified for the following uses:

SSL Server Certificate

Issued To	
Common Name (CN)	*.ischool.syr.edu
Organization (O)	Syracuse University
Organizational Unit (OU)	iSchool
Serial Number	09:F9:C5:4F:7E:DD:41:C2:99:7D:5C:45:CC:31:D3:A3
Issued By	
Common Name (CN)	DigiCert High Assurance CA-3
Organization (O)	DigiCert Inc
Organizational Unit (OU)	www.digicert.com
Validity	
Issued On	2/15/2009
Expires On	2/20/2011
Fingerprints	
SHA1 Fingerprint	26:10:A0:B4:72:EF:F8:09:32:4E:9E:A6:35:C0:97:76:5E:99:39:FC
MD5 Fingerprint	A1:60:E8:5E:CD:55:79:BC:3F:80:41:E5:39:8B:80:38

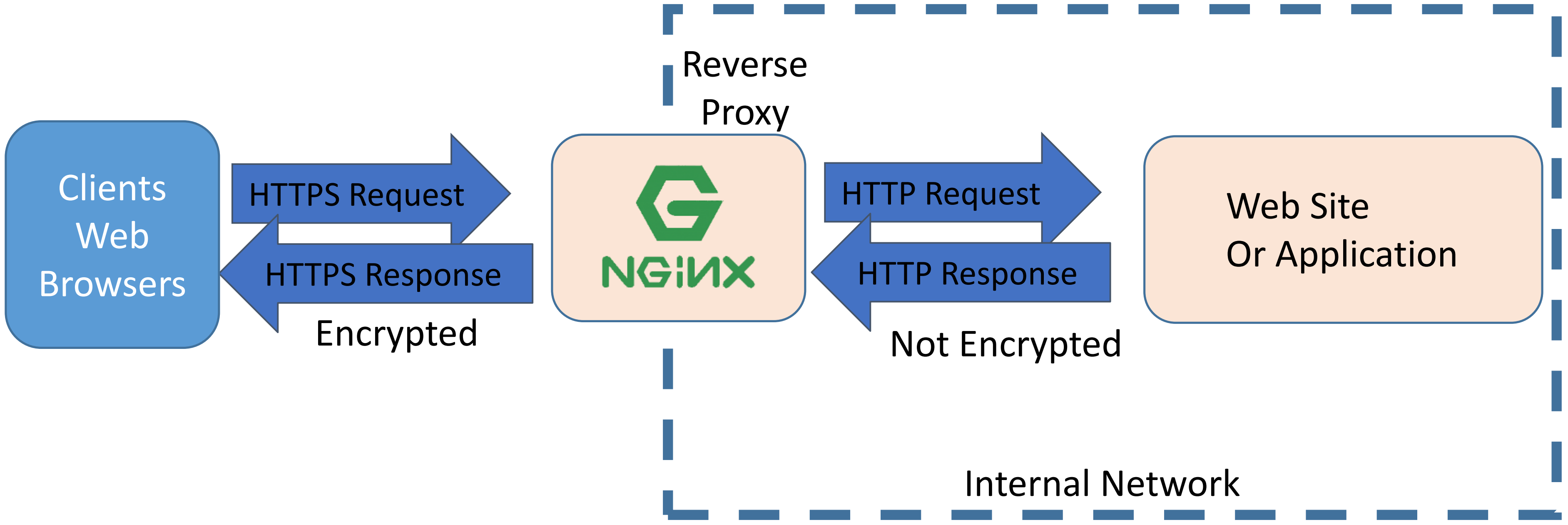
TLS – how it works on the web

1. Client request
2. Server response
3. Key exchange
4. Cipher negotiate
5. Client http get
6. Data transfer



TLS Termination Proxy

- We can configure our reverse proxy for TLS but do not require encryption over our internal network.

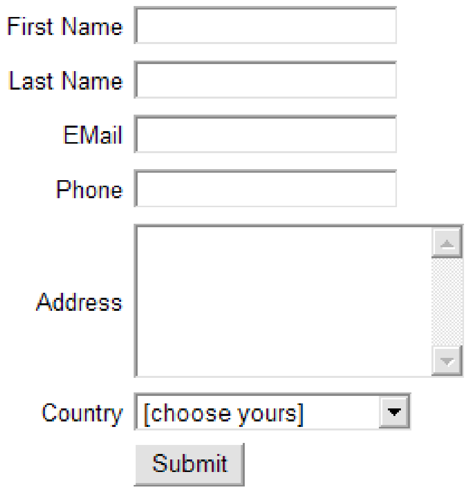


Protecting content

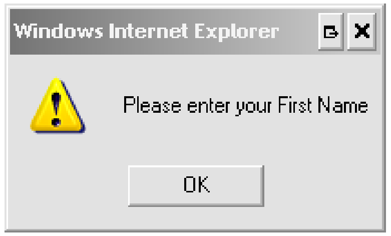
- We must take many steps in protecting website
- Common methods of attacks
  - *Directory Traversal*: using ../../ to go up or down a directory structure. Can obtain data that is otherwise unavailable
  - *Form field corruption*: using a websites forms to enter data or purchase items via hidden data fields. If you know what variables are being used to pass data, you can change the values.
  - *SQL injection:* inject SQL statements (select * from lastnames) to add, edit, or delete data in a database or even execute applications on the webserver.

Protecting data

- **Limited the potential damage.**
  - Connect to databases with read-only permissions if you are not updating or inserting data.
  - Validate form fields: verify the data the user typed before proceeding
  - Run web services with only the minimal level of permissions that is needed.
  - Use logging so if something does happen.
  - Use change control
  - ASSUME EVERYONE IS A BAD ACTOR!



First Name	
Last Name	
EMail	
Phone	
Address	
Country	[choose yours]



Please enter your First Name

Web API’s

What is an API?

- API Stands for Application Programming Interface
- It is a means to execute code other people wrote in our own programs.
- API's provide abstractions. You don't have to know how it works only how to use the API.
  - Think driving a car versus being a mechanic.
- Without API's we would have to write all our code from scratch every time.
- Imagine building a house but first you must build your own tools, cut your own wood from trees, make your own nails!

A Web API

- A Web API is an API which is executed over the HTTP or HTTPS protocols.
- This allows us to leverage services in the cloud into our own programs, such as:
  - Weather
  - Text to speech
  - Video playback
- Amazon Alexa is a simple device but seems intelligent because it simply is a voice activated means to execute Web API’s in the Cloud!

Why Web API’s

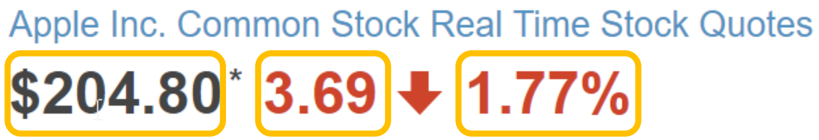
- The Web is transitioning:
  - From direct user-based consumption of data
  - To indirect user-based consumption of data through devices and also direct device-to-device consumption.
- Examples:
  - Do you read news in your browser or on your phone?
  - Do check the weather on weather.com or do you ask Alexa?

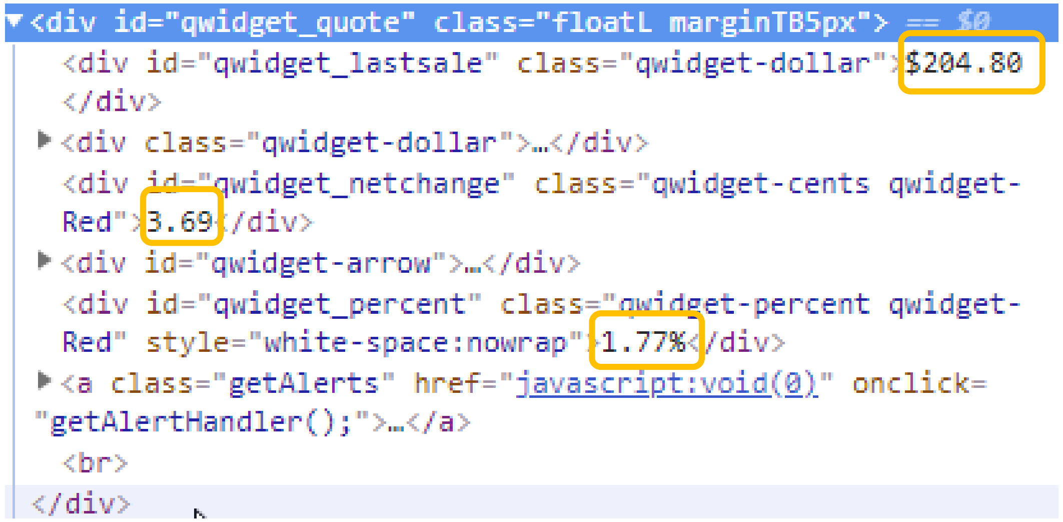
RESTful API’s

- When a web API embraces the HTTP semantics, it is considered a **RESTful** API.
- REST stands for “Representational State Transfer” and is a design pattern for API’s
- REST design uses URL’s and HTTP Verbs to make the intent clear:
- Examples:
  - Current Weather in Syracuse, NY:
    GET http://fudgeweather.com/weather/Syracuse,NY/current
  - Add Item to shopping cart:
    POST http://fudgeazon.com/cart?productid=1043

Web API Content formats

- HTML is not a desired content format for Web API's because another computer is the recipient of the output (as opposed to a user).
- While HTML **is** machine readable, it mixes data with layout and formatting making it difficult to find the information we wish to extract from the content.
- In the example there is HTML layout mixed in with the data, making the extraction of data difficult.

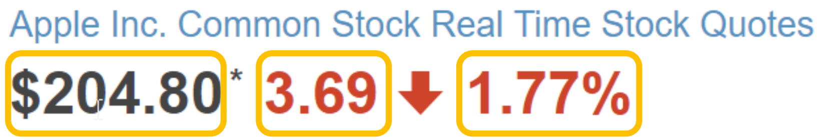




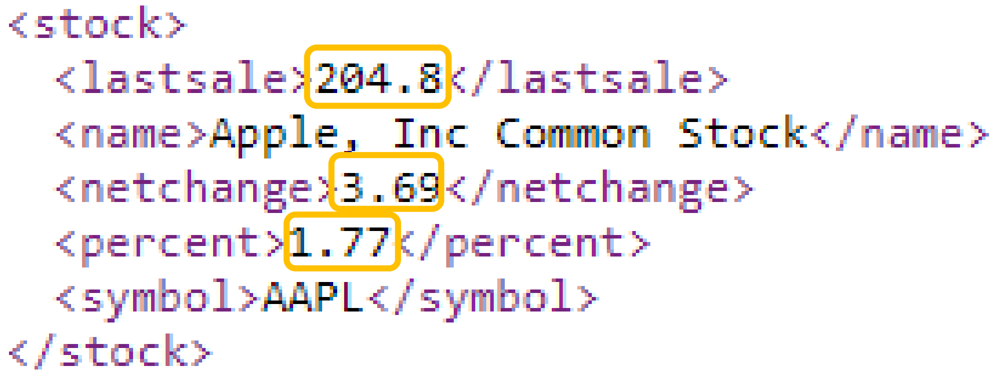
```
▼<div id="qwidget_quote" class="floatL marginTB5px"> == $0
   <div id="qwidget_lastsale" class="qwidget-dollar">$204.80
   </div>
 ▶<div class="qwidget-dollar">…</div>
   <div id="qwidget_netchange" class="qwidget-cents qwidget-
   Red">3.69</div>
 ▶<div id="qwidget-arrow">…</div>
   <div id="qwidget_percent" class="qwidget-percent qwidget-
   Red" style="white-space:nowrap">1.77%</div>
 ▶<a class="getAlerts" href="javascript:void(0)" onclick=
 "getAlertHandler();">…</a>
   <br>
</div>
```

XML Content Format

- XML – the Extensible Markup Language is a machine readable content format similar to HTML.
- XML allows for the design of schemas so that any data format can be represented. These schemas can then be validated to ensure the content
- In the example is trivial for a machine to extract the stock information because the XML only contains data and its structure.

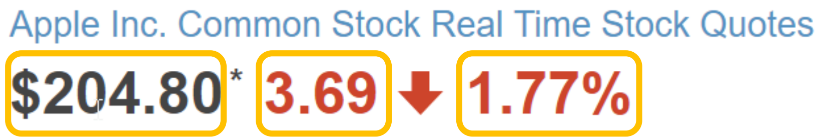

Apple Inc. Common Stock Real Time Stock Quotes

$204.80* 3.69 1.77%


```
<stock>
  <lastsale>204.8</lastsale>
  <name>Apple, Inc Common Stock</name>
  <netchange>3.69</netchange>
  <percent>1.77</percent>
  <symbol>AAPL</symbol>
</stock>
```

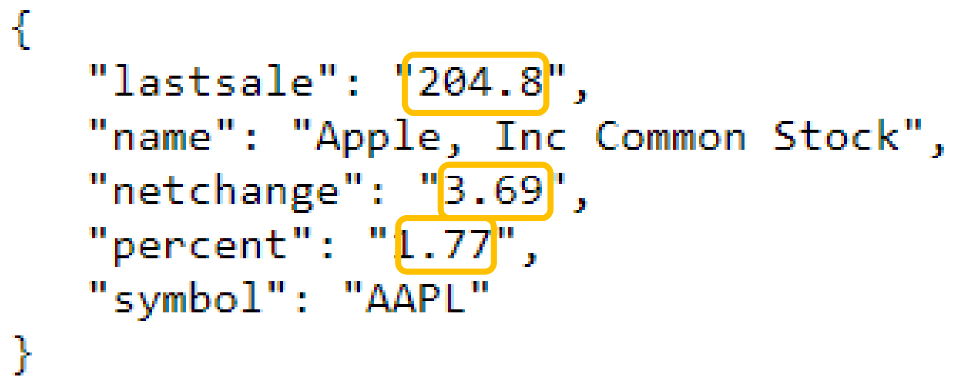
JSON Content Format

- JSON – JavaScript Object Notation is a lightweight data interchange format based on how JavaScript data is serialized to text.
- The JSON format is more compact than XML and requires little effort for many programming languages to parse (convert from text back into a workable object) easily.
- In the example is trivial for a machine to extract the stock information because the JSON only contains data and its structure.



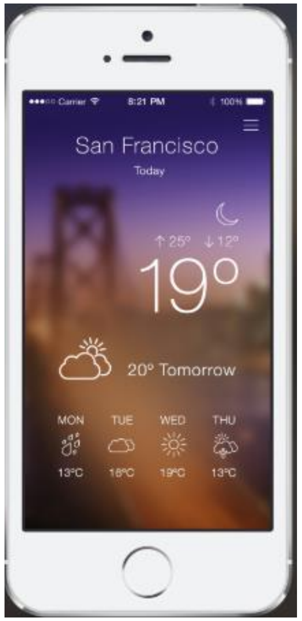
Apple Inc. Common Stock Real Time Stock Quotes

$204.80* 3.69 1.77%



```
{
    "lastsale": "204.8",
    "name": "Apple, Inc Common Stock",
    "netchange": "3.69",
    "percent": "1.77",
    "symbol": "AAPL"
}
```

Example: Web API



Weather App
On Phone

GET http://fudweather.com/San+Francicso,CA/current

fudweather
Web API

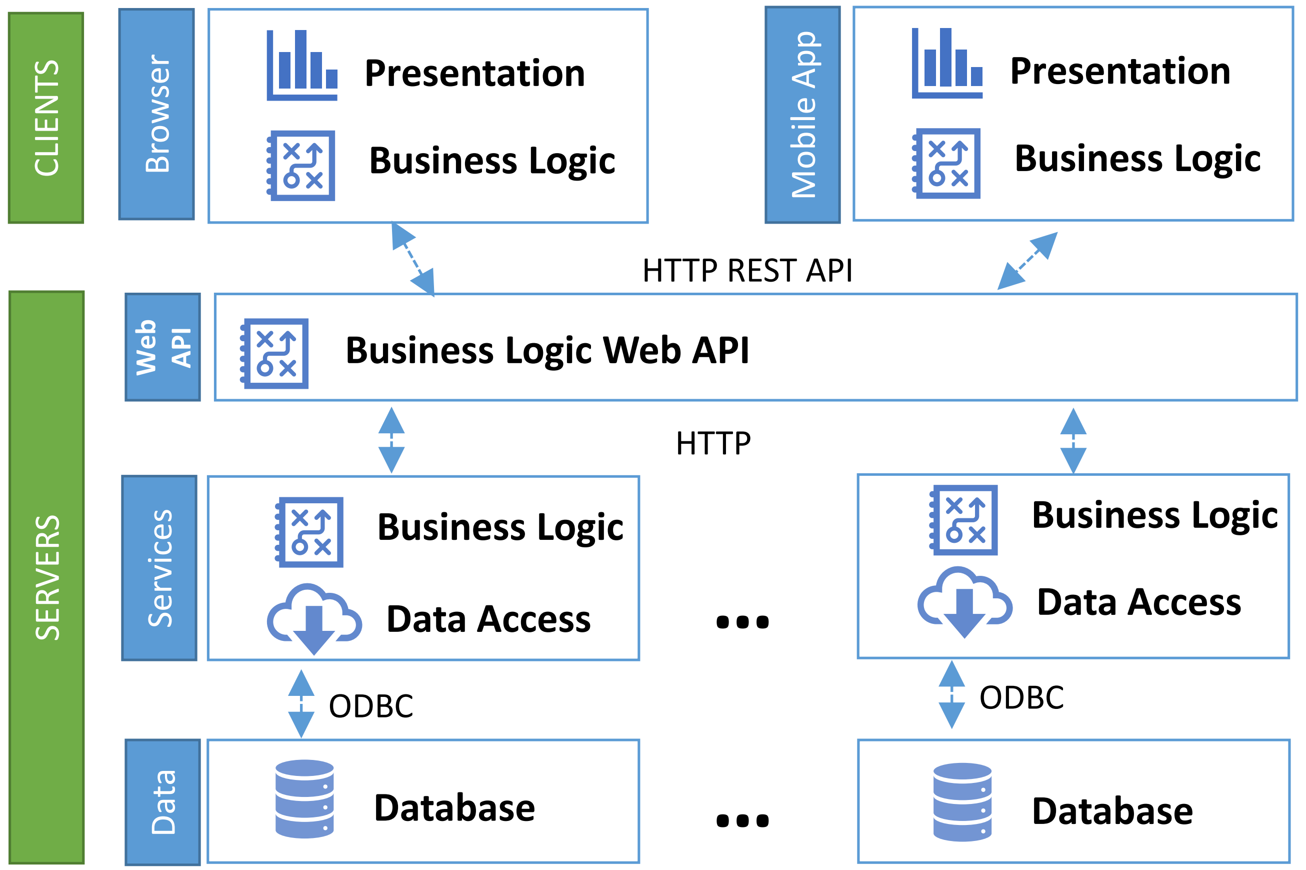
Response: Content-Type: application/json

```
{
    “temperature” : 19,
    “conditions” : “partly-cloudy”,
    “tomorrow” : 20
}
```

The Phone App is responsible for calling the API (requesting content) and drawing the API output on the screen

Recall: Micro Services

If you have a Web API, why not use it for any client whether it be Mobile or Web Browser or Anything else that comes along? Future-Proof your infrastructure!


CLIENTS
Browser
Presentation
Business Logic
Mobile App
Presentation
Business Logic
HTTP REST API
SERVERS
Web API
Business Logic Web API
HTTP
Services
Business Logic
Data Access
...
Business Logic
Data Access
ODBC
ODBC
Data
Database
...
Database