



# Module 2-9

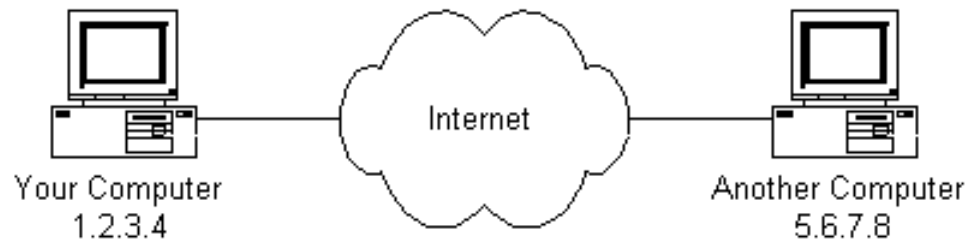
HTTP and Web APIs

# Objectives

- Explain the purpose of: IP Addresses, DNS, Ports, HTTP, TLS
- Identify and explain the purpose of the main components of HTTP
- Explain the steps of a typical HTTP request between a web browser and a server
- Explain what a GET request is used for
- Recognize that a 2xx Status Code indicates "success"
- Make an HTTP GET request using Postman and inspect the result
- Explain what JSON is and how to use it in a Java program
- Make an HTTP GET request to a RESTful web service using Java and process the response

# What is the Internet?

- **Global network of computers**
- Each has a unique address (IP Address – Internet Protocol)
- IPv4 (32 – bit)  
nnn.nnn.nnn.nnn -- nnn must be a number between 0 – 255  
Ex. -- 198.185.159.145
- IPv6 (128 – bit)  
8 groups of 4 hexadecimal digits, groups are separated by colons



# Anatomy of a URL

Here is a URL that uses an IP number:

**https://127.0.0.1:3000**

- **protocol:** others - http, ftp
- **ip address:** This is the unique address of a machine on a network.
- **port:** Number allocated for a specific type of service.

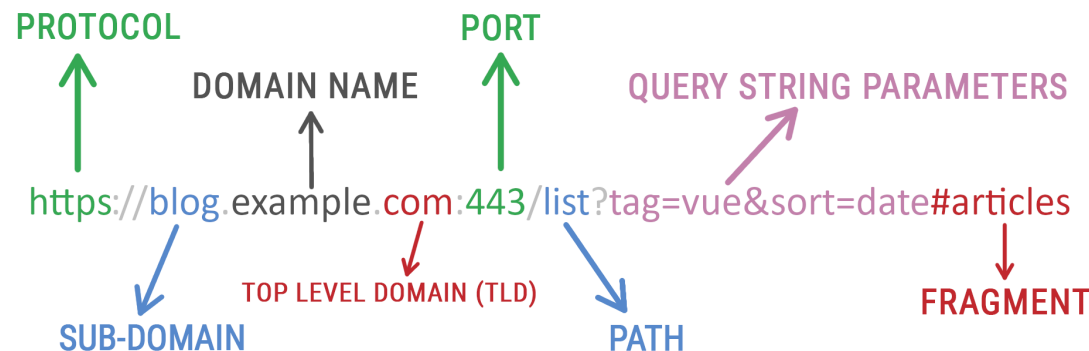
# Anatomy of a URL

Here is another URL that uses hostnames, this is certainly easier to remember than a bunch of numbers.

`https://skynet.wecomeinpeace.com`

- host name: A physical name assigned to your machine.
- **domain name**: Defines a specific “region of control” on the internet, also, .com is referred to as the top-level domain name.

The above URL is an example of a fully qualified domain name (FQDN).



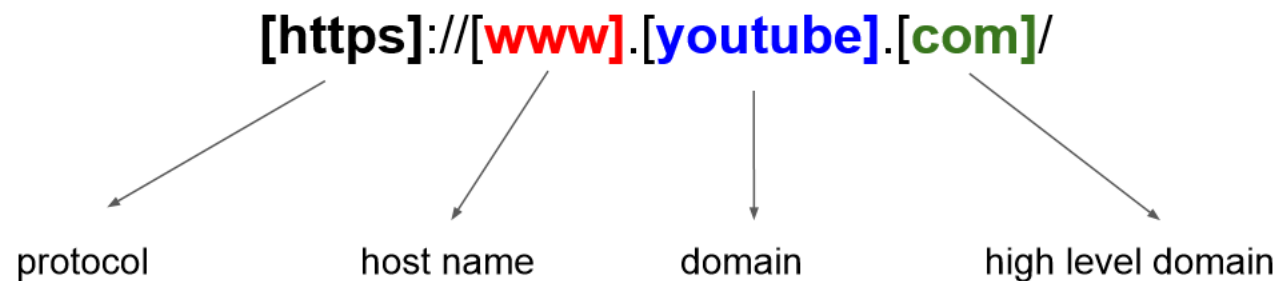
# DNS

- DNS is an acronym for Domain Name System.
- A DNS server is responsible for converting a URL containing human readable domain names (second example) to one containing an IP address (first example)

# WWW

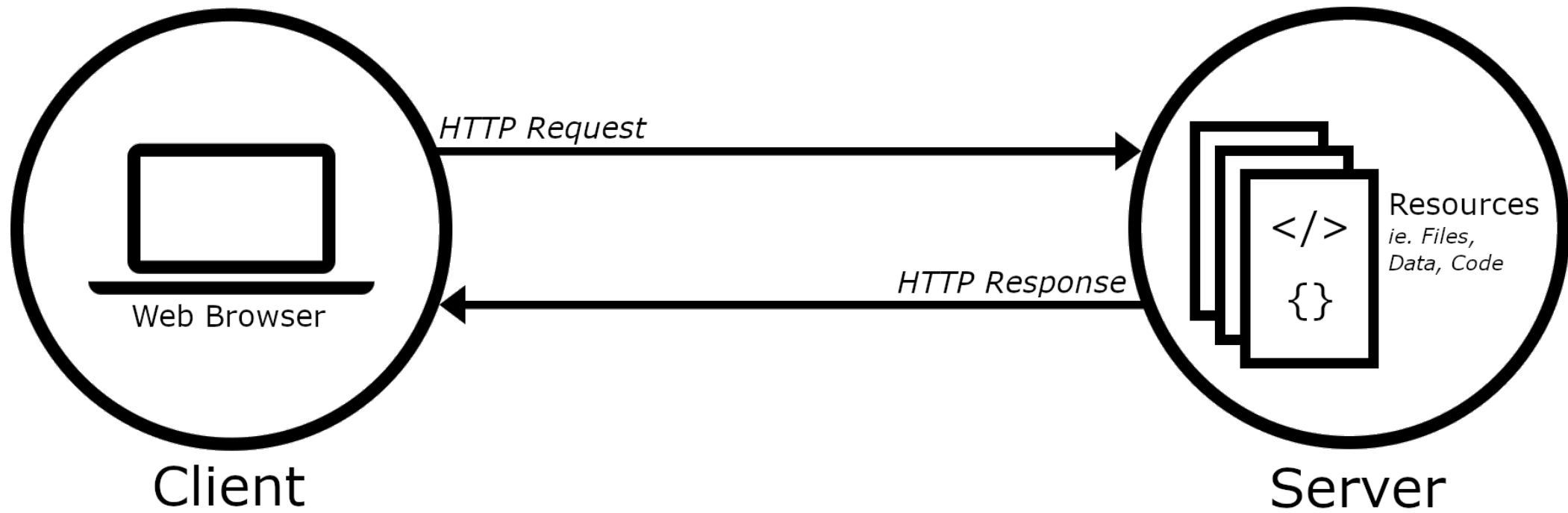
Because I'm sure you've wondered...

- On a URL the appearance of www has no bearing on the means by which we are communicating with another machine on the network (the protocol is still http or https).
- www is simply a hostname:





# HTTP



# HTTP request

- Request message has several elements:
  - An HTTP method (GET, PUT, POST, DELETE)
  - Path to resource
  - Headers
  - Message body

# HTTP response

- **Request message has several elements:**
  - **A status-line**
  - **Zero or more headers**
  - **Empty line**
  - **Message body (optional)**

# HTTP methods

- GET
- POST
- PUT
- DELETE

## Example

A simple form using the default `application/x-www-form-urlencoded` content type:

```
1 POST /test HTTP/1.1
2 Host: foo.example
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 27
5
6 field1=value1&field2=value2
```

A form using the `multipart/form-data` content type:

```
1 POST /test HTTP/1.1
2 Host: foo.example
3 Content-Type: multipart/form-data;boundary="boundary"
4
5 --boundary
6 Content-Disposition: form-data; name="field1"
7
8 value1
9 --boundary
10 Content-Disposition: form-data; name="field2"; filename="example.txt"
11
12 value2
13 --boundary--
```

# HTTP status codes

Code Range	Description
100-199	Information Responses
200-299	Successful Responses
300-399	Redirection Messages
400-499	Client Error Responses
500-599	Server Error Responses

## HTTP Status Codes

This page is created from HTTP status code information found at [ietf.org](http://ietf.org) and [Wikipedia](http://Wikipedia). Click on the **category heading** or the **status code** link to read more.

### 1xx Informational

100 Continue

101 Switching Protocols

102 Processing (WebDAV)

### 2xx Success

★ 200 OK

203 Non-Authoritative Information

206 Partial Content

226 IM Used

★ 201 Created

★ 204 No Content

207 Multi-Status (WebDAV)

202 Accepted

205 Reset Content

208 Already Reported (WebDAV)

### 3xx Redirection

300 Multiple Choices

303 See Other

306 (Unused)

301 Moved Permanently

★ 304 Not Modified

307 Temporary Redirect

302 Found

305 Use Proxy

308 Permanent Redirect (experimental)

### 4xx Client Error

★ 400 Bad Request

★ 403 Forbidden

406 Not Acceptable

★ 409 Conflict

412 Precondition Failed

415 Unsupported Media Type

418 I'm a teapot (RFC 2324)

423 Locked (WebDAV)

426 Upgrade Required

431 Request Header Fields Too Large

450 Blocked by Windows Parental Controls (Microsoft)

★ 401 Unauthorized

★ 404 Not Found

407 Proxy Authentication Required

410 Gone

413 Request Entity Too Large

416 Requested Range Not Satisfiable

420 Enhance Your Calm (Twitter)

424 Failed Dependency (WebDAV)

428 Precondition Required

444 No Response (Nginx)

451 Unavailable For Legal Reasons

402 Payment Required

405 Method Not Allowed

408 Request Timeout

411 Length Required

414 Request-URI Too Long

417 Expectation Failed

422 Unprocessable Entity (WebDAV)

425 Reserved for WebDAV

429 Too Many Requests

449 Retry With (Microsoft)

499 Client Closed Request (Nginx)

### 5xx Server Error

★ 500 Internal Server Error

503 Service Unavailable

506 Variant Also Negotiates (Experimental)

509 Bandwidth Limit Exceeded (Apache)

598 Network read timeout error

501 Not Implemented

504 Gateway Timeout

507 Insufficient Storage (WebDAV)

510 Not Extended

599 Network connect timeout error

502 Bad Gateway

505 HTTP Version Not Supported

508 Loop Detected (WebDAV)

511 Network Authentication Required

★ "Top 10" HTTP Status Code. More REST service-specific information is contained in the entry.

# Web service

- Web server – hardware and software to display content
  - Hardware is computer where software and data files are stored
  - Software provides controls on how files are accessed
    - HTTP server
    - Other protocols – FTP, UDP, SNMP, SFTP, available but not for web content.

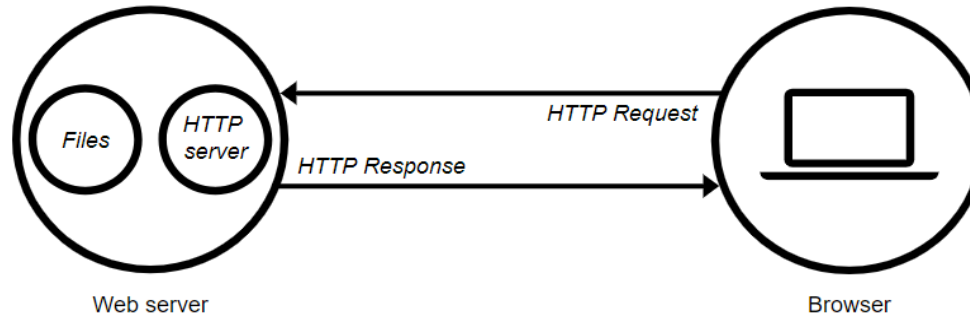


Image - Web Server

# Web service

- piece of software that makes itself available over the internet and uses a standardized XML messaging system.  
XML – is used to encode all communications to a web service
- Self contained, modular, distributed, dynamic applications  
Built on top of open standards such as TCP/IP, HTTP, Java, HTML and XML
- A collection of open protocols and standards used for exchanging data between applications or systems.

# What is an API?

- A set of functions and/or procedures designed to interact with an external system.
- Modern cloud architecture relies heavily on API's.
- Consuming an API means interacting with an API's code to product a desired result.



# APIs as a source of data

- We have explored various ways of obtaining data, starting from having Java read a text file, to building a sophisticated relational database like PostgreSQL.
- APIs could potentially be yet another source of data for other applications to consume.

# Request Types to an API

Recall that a REST controller can be configured to handle various types of requests. Let's review them:

- **GET**: Ideally suited to retrieve all the records from a REST endpoint.
- **GET** (with path variable): We can configure path variables (i.e. puppy/1 ) to retrieve a single record of data.
- **POST**: Ideally suited for inserting new data into the data source.
- **PUT**: Ideally suited for updating an existing record within a data source.
- **DELETE**: Ideally suited for removing an existing record from the data source.

Our focus today will be on GETs, in particular how we consume them in Java.

# Possible Responses from API


Once a request is made, the REST server can respond with specific status codes:

- **200**: All's well, the request was successful.
- **4XX**: The client (you or your application) has not structured the request correctly. Common examples of these are 400 Bad Request and 401 Unauthorized Request.
- **5XX**: The server has encountered some kind of error. The most common of these is the 500 Internal Server Error message

Postman!

# JSON

- JavaScript Object Notation
- Lightweight format for storing and transporting data
- Often used when data is sent from server to a web page
- “self – describing” and easy to understand
- Not language-specific (can be generated or read in any language)

```
Pretty Raw Preview Visualize JSON 
```

```
1 {
2   "count": 293,
3   "next": "https://pokeapi.co/api/v2/ability?offset=20&limit=20",
4   "previous": null,
5   "results": [
6     {
7       "name": "stench",
8       "url": "https://pokeapi.co/api/v2/ability/1/"
9     },
10    {
11      "name": "drizzle",
12      "url": "https://pokeapi.co/api/v2/ability/2/"
13    },
14    {
15      "name": "speed-boost",
16      "url": "https://pokeapi.co/api/v2/ability/3/"
17    },
18    {
19      "name": "battle-armor",
20      "url": "https://pokeapi.co/api/v2/ability/4/"
21    },
22    {
23      "name": "sturdy",
24      "url": "https://pokeapi.co/api/v2/ability/5/"
25    },
26    {
27      "name": "damp",
28      "url": "https://pokeapi.co/api/v2/ability/6/"
29    },
30    {
31      "name": "limber",
32      "url": "https://pokeapi.co/api/v2/ability/7/"
33    },
34    {
35      "name": "speed-veil"
```

# Making a GET request through Java

The RestTemplate class provides the means with which we can make a request to an API. Here is an example call:

```
private static final String API_BASE_URL = "http://helpful-site/v1/api/data";  
private static RestTemplate restTemplate = new RestTemplate();  
MyObj myobj = restTemplate.getForObject(API_BASE_URL, MyObj.class);
```

Note that we can specify the return type of the API call with the second parameter (MyObj.class). Alternatively, if you are getting an array of objects back, we can write the following:

```
MyObj [ ] myobj = restTemplate.getForObject(API_BASE_URL, MyObj[ ].class);
```

# Serialize vs. Deserialize

- When we convert our data from a JSON string into an object, we are deserializing.
- We won't cover this today, but the opposite of this is to serialize, which converts the object into a byte stream.

Let's code!