Module 3: URL's and Understanding HTTP Request and Response

Edgardo Molina, PhD | Head Instructor

CUNY Tech Prep 2017-2018

Outline

Parts of a URL (IP's, TLD's, DNS, Ports)

Understanding HTTP Request - Response

More React

`fetch()` and Promises

Client-Server Communication

Client-Server Communication

In the context of web applications:

A client requests a web page, and the server responds with the requested web page or an error page.

In general:

Clients request a resource, and the server responds with that resource or an error status.

Client-Server Communication

In the context of web applications:

A client requests a web page, and the server responds with the requested web page or an error page.

In general:

Clients request a resource, and the server responds with that resource or an error status.

Networked Applications

The client and server are separate programs, they can run and connect on the same machine, or run and connect from different machines across a network.

In both cases, they use the same network protocols.

Client-Server Communication

What are clients?

Client-Server Communication

What are clients?

- Web Browsers
    - Desktop: Google Chrome, Firefox, Safari, Microsoft IE
    - Mobile: Safari, Chrome
- Other Programs
    - Instead of a user at a browser making a request, it could be another program (Apps, scripts, etc)
    - Google does this to crawl the internet
    - But our own applications can do it as well using API's (We will look at this later)

How does a client know where to find the server for a webpage URL?

`http://cunytechprep.nyc/index.html`

How does a client know where to find the server for a webpage URL?

```
http://cunytechprep.nyc/index.html
```

What is a URL?

How does a client know where to find the server for a webpage URL?

```
http://cunytechprep.nyc/index.html
```

What is a URL?

A *Uniform Resource Locator* also known as a web address.

Parts of the URL

```
http://cuny.edu/index.html
----   -------- ----------
  1        2         3
```

1. Application Protocol
2. Hostname
3. Resource Path

Parts of the URL

1. Application Protocols
    - HTTP - Hypertext Transfer Protocol
    - HTTPS - SSL Encrypted HTTP
2. Hostname
    - Registered Name
    - TLD - Top-Level Domain (.com, .edu, .org, .nyc, etc)
        - 1000's available
3. Path
    - Maps to a filesystem path for a specific document
    - Or, the path is passed to a Web Application as input

The Hostname

How does the Hostname get us to a specific server computer?

The Hostname

How does the Hostname get us to a specific server computer?

Computers on a network are assigned an IP Address for computer to computer communication.

For computers on the internet, your Internet Service Provider (ISP) assigns you the IP address.

An IPv4 address is a 32-bit number. Each byte separated by a (`.`) period.

`128.228.38.47`

Private IP Addresses

- `127.0.0.1` and `localhost` refer to the local machine
- `0.0.0.0` refers to the default address for local machine
- `10.0.0.0 - 10.255.255.255`
- `172.16.0.0 - 172.31.255.255`
- `192.168.0.0 - 192.168.255.255`

Private addresses can be used in closed networks (SOHO, Virtual Networking, WiFi Access Points)

The Hostname

How is `cuny.edu` translated to `128.228.38.47`?

The Hostname

How is `cuny.edu` translated to `128.228.38.47`?

Domain Name System

A decentralized hierarchy of servers provide domain name resolution.

Service maps registered hostnames (domain names) to IP Addresses.

General URL Format

`scheme:[//[user:pass@]host[:port]][/]path[?query][#fragment]`

- **scheme**: Application protocols (http/https)
- **[user:pass@]**: Optional. Authentication information
- **host**: A hostname or IP Address
- **[:port]**: Optional. HTTP/HTTPS default ports 80/443
- **path**: Resourse location. Can be **/**
- **[?query]**: Optional. Parameter and Argument list passed to web application
- **[#fragment]**: Optional. Identifier for secondary resource. Evaluated by client

Client-Server Communication

How does the web browser (client) retrieve webpages from a server using URL's?

```
http://cunytechprep.nyc/index.html
```

Client-Server Communication

How does the web browser (client) retrieve webpages from a server using URL's?

```
http://cunytechprep.nyc/index.html
```

1. Extract domain name from URL
2. Lookup IP for domain name via DNS service
3. Send the entire URL request to the IP address found in (2)
4. Receive the documents requested via path.

More Resources

https://en.wikipedia.org/wiki/Uniform_Resource_Locator

https://en.wikipedia.org/wiki/List_of_Internet_top-level_domains

Understanding HTTP Request - Response

Understanding HTTP Request - Response

What is happening when we call an HTTP URL?

- We send a HTTP Request
- We receive a HTTP Response
    - The payload may include: html, json, or nothing

Understanding HTTP Request - Response

What is happening when we call an HTTP URL?

- We send a HTTP Request
- We receive a HTTP Response
    - The payload may include: html, json, or nothing

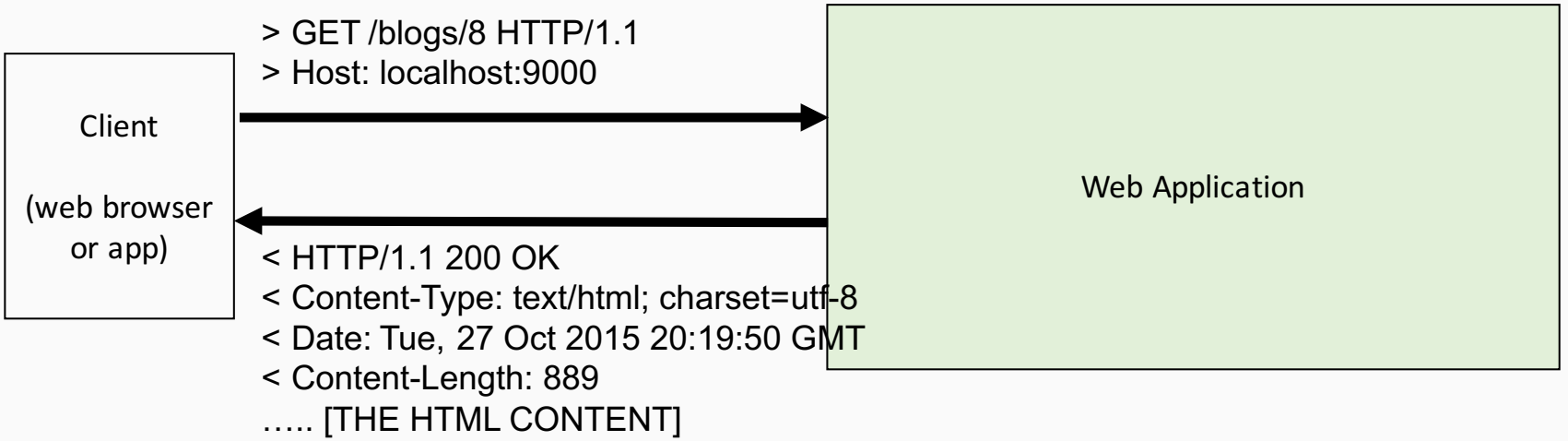

Figure 1: HTTP Request - Response

HTTP is Stateless

Wikipedia definition

> *HTTP is a stateless protocol. A stateless protocol does not require the HTTP server to retain information or status about each user for the duration of multiple requests. However, some web applications implement states or server side sessions using for instance HTTP cookies or Hidden variables within web forms.*

The browser opens a connection upon request, and it is closed once the response is received.

HTTP Verbs

When we load a URL in our browser we are making a **GET** request

HTTP supports other types of request methods

`https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Request_methods`

We call these HTTP Verbs

> *They do not explicitly change anything about the request except some headers. It is up to your web application to interpret them appropriately.*

The Request

HTTP Verbs

- GET
  - We typically use this request to retrieve a resource.
  - *GET* requests should not modify the content
- POST
  - Use this request type to create a new resource entity
- PUT
  - Use this request type to update an existing resource entity
- DELETE
  - Use this request type to delete an existing resource entity

The Response

HTTP returns a status code in the header

The status code lets the client know if the request succeeded, failed, or some other action is required.

In addition to the status code, content may be included in the response (the html, json, xml, file, etc).

The Response

Common Types of status codes

- 200 – Success
- 201 – Created
- 301 – Moved permanently
- 303 – See other (redirect)
- 400 – Bad request
- 404 – Not found
- 5xx – Server Errors

Full List:

`https://en.wikipedia.org/wiki/List_of_HTTP_status_codes`

Using cURL

`curl` is a command line tool for transferring data to and from URL's

> *Useful in testing web applications, API's, and for implementing web tools*

Some useful options

- `-v`
  - Makes the output verbose. Displays HTTP headers
- `-X [METHOD]`
  - Allows you to change the HTTP verb from the default GET to POST, PUT, DELETE, PATCH, etc
- `-d "fieldname=value"`
  - Allows you to submit parameters to the URL

Using cURL

Examples

curl -v http://ctp-zip-api.herokuapp.com/

curl -v -X POST -d “zipcode=10016” http://ctp-zip-api.herokuapp.com/zip

curl -v -X POST -d “test=hello” http://ctp-zip-api.herokuapp.com/zip