HTTP/URL research exercises

Tags	Tools
Created time	@April 22, 2023 11:31 PM
Last edited time	@April 23, 2023 9:01 PM

- The *fragment* part of a URL.

Every HTTP URL conforms to the syntax of a generic URI. The URI generic syntax consists of five *components* organized hierarchically in order of decreasing significance from left to right:

```
URI = scheme ":" ["//" authority] path ["?" query] ["#" fragment]
```

```
authority = [userinfo "@"] host [":" port]
```

The URI comprises:

- A non-empty **scheme** component followed by a colon (`:`), consisting of a sequence of characters beginning with a letter and followed by any combination of letters, digits, plus (`+`), period (`.`), or hyphen (`-`). Although schemes are case-insensitive, the canonical form is lowercase and documents that specify schemes must do so with lowercase letters. Examples of popular schemes include `http`, `https`, `ftp`, `mailto`, `file`, `data` and `irc`. URI schemes should be registered with the Internet Assigned Numbers Authority (IANA), although non-registered schemes are used in practice.

  [b]

- An optional **authority** component preceded by two slashes (`//`), comprising:
  - An optional **userinfo** subcomponent followed by an at symbol (`@`), that may consist of a user name and an optional password preceded by a colon (`:`). Use of the format `username:password` in the userinfo subcomponent is deprecated for security reasons. Applications should not render as clear text any data after the

first colon (`:`) found within a userinfo subcomponent unless the data after the colon is the empty string (indicating no password).

    - A **host** subcomponent, consisting of either a registered name (including but not limited to a hostname) or an IP address. IPv4 addresses must be in dot-decimal notation, and IPv6 addresses must be enclosed in brackets (`[]`).
    - An optional **port** subcomponent preceded by a colon (`:`), consisting of decimal digits.

- A **path** component, consisting of a sequence of path segments separated by a slash (`/`). A path is always defined for a URI, though the defined path may be empty (zero length). A segment may also be empty, resulting in two consecutive slashes (`//`) in the path component. A path component may resemble or map exactly to a file system path but does not always imply a relation to one. If an authority component is defined, then the path component must either be empty or begin with a slash (`/`). If an authority component is undefined, then the path cannot begin with an empty segment—that is, with two slashes (`//`)—since the following characters would be interpreted as an authority component.

By convention, in **http** and **https** URIs, the last part of a *path* is named **pathinfo** and it is optional. It is composed by zero or more path segments that do not refer to an existing physical resource name (e.g. a file, an internal module program or an executable program) but to a logical part (e.g. a command or a qualifier part) that has to be passed separately to the first part of the path that identifies an executable module or program managed by a web server; this is often used to select dynamic content (a document, etc.) or to tailor it as requested (see also: CGI and PATH_INFO, etc.).Example:URI: `"http://www.example.com/questions/3456/my-document"`where: `"/questions"` is the first part of the *path* (an executable module or program) and `"/3456/my-document"` is the second part of the *path* named *pathinfo*, which is passed to the executable module or program named `"/questions"` to select the requested document.An **http** or **https** URI containing a *pathinfo* part without a query part may also be referred to as a 'clean URL' whose last part may be a 'slug'.

Query delimiter	Example
Ampersand (`&`)	`key1=value1&key2=value2`
Semicolon (`;`)[d]	`key1=value1;key2=value2`

- An optional **query** component preceded by a question mark (`?`), consisting of a query string of non-hierarchical data. Its syntax is not well defined, but by convention is most often a sequence of attribute–value pairs separated by a delimiter.
- An optional **fragment** component preceded by a hash (`#`). The fragment contains a fragment identifier providing direction to a secondary resource, such as a section heading in an article identified by the remainder of the URI. When the primary resource is an HTML document, the fragment is often an `id` attribute of a specific element, and web browsers will scroll this element into view.

A web browser will usually dereference a URL by performing an HTTP request to the specified host, by default on port number 80. URLs using the `https` scheme require that requests and responses be made over a secure connection to the website.

- The `Accept` header sent by the HTTP client.

The `Accept` request HTTP header indicates which content types, expressed as MIME types, the client is able to understand. The server uses content negotiation to select one of the proposals and informs the client of the choice with the `Content-Type` response header. Browsers set required values for this header based on the context of the request. For example, a browser uses different values in a request when fetching a CSS stylesheet, image, video, or a script.

In this example, the client indicates that it can handle three MIME types: `text/html`, `application/xhtml+xml`, and `application/xml`. The `q` parameter specifies the relative priority of each type, with `q=0.9` indicating a higher priority than `q=0.8`. The `*/*` wildcard indicates that the client can handle any other MIME type with a lower priority than the ones explicitly listed.

When a server receives a request with an Accept header, it can use it to determine the best response format to send back to the client. The server can compare the MIME types that it can produce with the ones requested by the client and choose the most appropriate one based on the priority and other parameters specified in the Accept header. If the server cannot produce any of the requested MIME types, it can send back an error response, such as a 406 Not Acceptable status code.

```
Accept: <MIME_type>/<MIME_subtype>
Accept: <MIME_type>/*
Accept: */*

// Multiple types, weighted with the quality value syntax:
Accept: text/html, application/xhtml+xml, application/xml;q=0.9, image/webp, */*;q=0.8
--------------------------------------------------------------------------------

Accept: text/html

Accept: image/*

// General default
Accept: */*

// Default for navigation requests
Accept: text/html, application/xhtml+xml, application/xml;q=0.9, */*;q=0.8
```

- The `User-agent` header sent by the HTTP client. What does your browser send?

A user agent is a computer program representing a person, for example, a browser in a Web context.

Besides a browser, a user agent could be a bot scraping webpages, a download manager, or another app accessing the Web. Along with each request they make to the server, browsers include a self-identifying `User-Agent` HTTP header called a user agent (UA) string. This string often identifies the browser, its version number, and its host operating system.

Spam bots, download managers, and some browsers often send a fake UA string to announce themselves as a different client. This is known as *user agent spoofing*.

The user agent string can be accessed with JavaScript on the client side using the `NavigatorID.userAgent` property.

A typical user agent string looks like this: `"Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:35.0) Gecko/20100101 Firefox/35.0"`.

- How do you encode a path that contains a space in an URL? Which other characters are "special" and need to be treated this way in paths?

Answer: %20

- Which web server is the University of Bristol using for `www.bristol.ac.uk`, according to the headers? Read up online on this server and the organisation of the same name that maintains it.

Answer: Apache