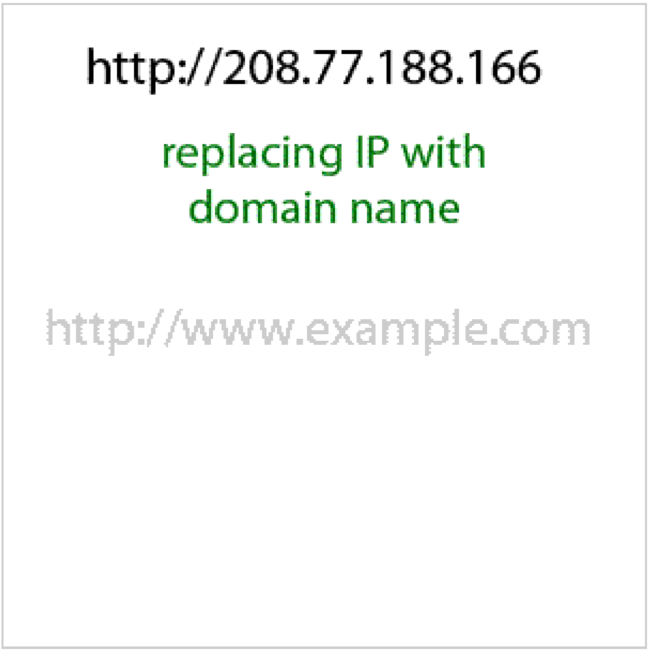
URL normalization

From Wikipedia, the free encyclopedia

URL normalization (or **URL canonicalization**) is the process by which URLs are modified and standardized in a consistent manner. The goal of the normalization process is to transform a URL into a normalized or canonical URL so it is possible to determine if two syntactically different URLs may be equivalent.

Search engines employ URL normalization in order to assign importance to web pages and to reduce indexing of duplicate pages. Web crawlers perform URL normalization in order to avoid crawling the same resource more than once. Web browsers may perform normalization to determine if a link has been visited or to determine if a page has been cached.



Types of URL normalization.

Contents

- 1 Normalization process
  - 1.1 Normalizations that preserve semantics
  - 1.2 Normalizations that usually preserve semantics
  - 1.3 Normalizations that change semantics
- 2 Normalization based on URL lists
- 3 See also
- 4 References

Normalization process

There are several types of normalization that may be performed. Some of them are always semantics preserving and some may not be.

Normalizations that preserve semantics

The following normalizations are described in RFC 3986 [1] to result in equivalent URLs:

- **Converting the scheme and host to lower case.** The scheme and host components of the URL are case-insensitive. Most normalizers will convert them to lowercase. Example:

`HTTP://www.Example.com/` → `http://www.example.com/`

- **Capitalizing letters in escape sequences.** All letters within a percent-encoding triplet (e.g., "%3A") are case-insensitive, and should be capitalized. Example:

  `http://www.example.com/a%c2%b1b` → `http://www.example.com/a%C2%B1b`

- **Decoding percent-encoded octets of unreserved characters.** For consistency, percent-encoded octets in the ranges of *ALPHA* (`%41`–`%5A` and `%61`–`%7A`), *DIGIT* (`%30`–`%39`), hyphen (`%2D`), period (`%2E`), underscore (`%5F`), or tilde (`%7E`) should not be created by URI producers and, when found in a URI, should be decoded to their corresponding unreserved characters by URI normalizers.[2] Example:

  `http://www.example.com/%7Eusername/` → `http://www.example.com/~username/`

- **Removing the default port.** The default port (port 80 for the "http" scheme) may be removed from (or added to) a URL. Example:

  `http://www.example.com:80/bar.html` → `http://www.example.com/bar.html`

Normalizations that usually preserve semantics

For http and https URLs, the following normalizations listed in RFC 3986 may result in equivalent URLs, but are not guaranteed to by the standards:

- **Adding trailing /** Directories are indicated with a trailing slash and should be included in URLs. Example:

  `http://www.example.com/alice` → `http://www.example.com/alice/`
  However, there is no way to know if a URL path component represents a directory or not. RFC 3986 notes that if the former URL redirects to the latter URL, then that is an indication that they are equivalent.

- **Removing dot-segments.** The segments ".." and "." can be removed from a URL according to the algorithm described in RFC 3986 (or a similar algorithm). Example:

  `http://www.example.com/../a/b/../c/./d.html` → `http://www.example.com/a/c/d.html`
  However, if a removed "`..`" component, e.g. "`b/..`", is a symlink to a directory with a different parent, eliding "`b/..`" will result in a different path and URL.[3] In rare cases depending on the web server, this may even be true for the root directory (e.g. "`//www.example.com/..`" may not be equivalent to "`//www.example.com/`".

Normalizations that change semantics

Applying the following normalizations result in a semantically different URL although it may refer to the same resource:

- **Removing directory index.** Default directory indexes are generally not needed in URLs. Examples:

  `http://www.example.com/default.asp` → `http://www.example.com/`

  `http://www.example.com/a/index.html` → `http://www.example.com/a/`

- **Removing the fragment.** The fragment component of a URL is never seen by the server and can sometimes be removed. Example:

  `http://www.example.com/bar.html#section1` → `http://www.example.com/bar.html`
  However, AJAX applications frequently use the value in the fragment.

- **Replacing IP with domain name.** Check if the IP address maps to a canonical domain name. Example:

  `http://208.77.188.166/` → `http://www.example.com/`
  The reverse replacement is rarely safe due to virtual web servers.

- **Limiting protocols.** Limiting different application layer protocols. For example, the "https" scheme could be replaced with "http". Example:

  `https://www.example.com/` → `http://www.example.com/`

- **Removing duplicate slashes** Paths which include two adjacent slashes could be converted to one. Example:

  `http://www.example.com/foo//bar.html` → `http://www.example.com/foo/bar.html`

- **Removing or adding "www" as the first domain label.** Some websites operate identically in two Internet domains: one whose least significant label is "www" and another whose name is the result of omitting the least significant label from the name of the first, the latter being known as a naked domain. For example, `http://example.com/` and `http://www.example.com/` may access the same website. Many websites redirect the user from the www to the non-www address or vice versa. A normalizer may determine if one of these URLs redirects to the other and normalize all URLs appropriately. Example:

  `http://www.example.com/` → `http://example.com/`

- **Sorting the query parameters.** Some web pages use more than one query parameter in the URL. A normalizer can sort the parameters into alphabetical order (with their values), and reassemble the URL. Example:

`http://www.example.com/display?lang=en&article=fred` →

`http://www.example.com/display?article=fred&lang=en`

However, the order of parameters in a URL may be significant (this is not defined by the standard) and a web server may allow the same variable to appear multiple times.[4]

- **Removing unused query variables.** A page may only expect certain parameters to appear in the query; unused parameters can be removed. Example:

  `http://www.example.com/display?id=123&fakefoo=fakebar` →

  `http://www.example.com/display?id=123`

  Note that a parameter without a value is not necessarily an unused parameter.

- **Removing default query parameters.** A default value in the query string may render identically whether it is there or not. Example:

  `http://www.example.com/display?id=&sort=ascending` →

  `http://www.example.com/display`

- **Removing the "?" when the query is empty.** When the query is empty, there may be no need for the "?". Example:

  `http://www.example.com/display?` → `http://www.example.com/display`

Normalization based on URL lists

Some normalization rules may be developed for specific websites by examining URL lists obtained from previous crawls or web server logs. For example, if the URL

`http://example.com/story?id=xyz`

appears in a crawl log several times along with

`http://example.com/story_xyz`

we may assume that the two URLs are equivalent and can be normalized to one of the URL forms.

Schonfeld et al. (2006) present a heuristic called DustBuster for detecting DUST (different URLs with similar text) rules that can be applied to URL lists. They showed that once the correct DUST rules were found and applied with a canonicalization algorithm, they were able to find up to 68% of the redundant URLs in a URL list.

See also

- Uniform Resource Locator
- Fragment identifier

- Web crawler

References

1. ^ RFC 3986, Section 6: Normalization and Comparison (http://tools.ietf.org/html/rfc3986#section-6)
2. ^ RFC 3986, Section 2.3.: Unreserved Characters (http://tools.ietf.org/html/rfc3986#section-2.3)
3. ^ "Secure Coding in C and C++" (https://www.securecoding.cert.org/confluence/download/attachments/26017980/08+File+System+Vulnerabilities.pdf). Securecoding.cert.org. Retrieved 2013-08-24.
4. ^ "jQuery 1.4 $.param demystified" (http://benalman.com/news/2009/12/jquery-14-param-demystified/). Ben Alman. 2009-12-20. Retrieved 2013-08-24.

- RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax
- Sang Ho Lee, Sung Jin Kim, and Seok Hoo Hong (2005). "On URL normalization" (http://dblab.ssu.ac.kr/publication/LeKi05a.pdf). Proceedings of the International Conference on Computational Science and its Applications (ICCSA 2005). pp. 1076–1085.
- Uri Schonfeld, Ziv Bar-Yossef, and Idit Keidar (2006). "Do not crawl in the dust: different URLs with similar text" (http://www2006.org/programme/item.php?id=p20). Proceedings of the 15th international conference on World Wide Web. pp. 1015–1016.
- Uri Schonfeld, Ziv Bar-Yossef, and Idit Keidar (2007). "Do not crawl in the dust: different URLs with similar text" (http://www2007.org/paper194.php). Proceedings of the 16th international conference on World Wide Web. pp. 111–120.

Retrieved from "http://en.wikipedia.org/w/index.php?title=URL_normalization&oldid=639756675"

Categories: Uniform resource locator | Internet search algorithms

---