Network Working Group
Request for Comments: 1808
Category: Standards Track

R. Fielding
UC Irvine

June 1995

Relative Uniform Resource Locators

Status of this Memo

Abstract

A Uniform Resource Locator (URL) is a compact representation of the location and access method for a resource available via the Internet. When embedded within a base document, a URL in its absolute form may contain a great deal of information which is already known from the context of that base document's retrieval, including the scheme, network location, and parts of the url-path. In situations where the base URL is well-defined and known to the parser (human or machine), it is useful to be able to embed URL references which inherit that context rather than re-specifying it in every instance. This document defines the syntax and semantics for such Relative Uniform Resource Locators.

1. Introduction

This document describes the syntax and semantics for "relative" Uniform Resource Locators (relative URLs): a compact representation of the location of a resource relative to an absolute base URL. It is a companion to RFC 1738, "Uniform Resource Locators (URL)" [2], which specifies the syntax and semantics of absolute URLs.

A common use for Uniform Resource Locators is to embed them within a document (referred to as the "base" document) for the purpose of identifying other Internet-accessible resources. For example, in hypertext documents, URLs can be used as the identifiers for hypertext link destinations.

Absolute URLs contain a great deal of information which may already be known from the context of the base document's retrieval, including the scheme, network location, and parts of the URL path. In situations where the base URL is well-defined and known, it is useful to be able to embed a URL reference which inherits that context

rather than re-specifying it within each instance. Relative URLs can also be used within data-entry dialogs to decrease the number of characters necessary to describe a location.

In addition, it is often the case that a group or "tree" of documents has been constructed to serve a common purpose; the vast majority of URLs in these documents point to locations within the tree rather than outside of it. Similarly, documents located at a particular Internet site are much more likely to refer to other resources at that site than to resources at remote sites.

Relative addressing of URLs allows document trees to be partially independent of their location and access scheme. For instance, it is possible for a single set of hypertext documents to be simultaneously accessible and traversable via each of the "file", "http", and "ftp" schemes if the documents refer to each other using relative URLs. Furthermore, document trees can be moved, as a whole, without changing any of the embedded URLs. Experience within the World-Wide Web has demonstrated that the ability to perform relative referencing is necessary for the long-term usability of embedded URLs.

2. Relative URL Syntax

The syntax for relative URLs is a shortened form of that for absolute URLs [2], where some prefix of the URL is missing and certain path components ("." and "..") have a special meaning when interpreting a relative path. Because a relative URL may appear in any context that could hold an absolute URL, systems that support relative URLs must be able to recognize them as part of the URL parsing process.

Although this document does not seek to define the overall URL syntax, some discussion of it is necessary in order to describe the parsing of relative URLs. In particular, base documents can only make use of relative URLs when their base URL fits within the generic-RL syntax described below. Although some URL schemes do not require this generic-RL syntax, it is assumed that any document which contains a relative reference does have a base URL that obeys the syntax. In other words, relative URLs cannot be used within documents that have unsuitable base URLs.

2.1. URL Syntactic Components

The URL syntax is dependent upon the scheme. Some schemes use reserved characters like "?" and ";" to indicate special components, while others just consider them to be part of the path. However, there is enough uniformity in the use of URLs to allow a parser to resolve relative URLs based upon a single, generic-RL syntax. This generic-RL syntax consists of six components:

<scheme>://<net_loc>/<path>;<params>?<query>#<fragment>

each of which, except <scheme>, may be absent from a particular URL. These components are defined as follows (a complete BNF is provided in Section 2.2):

scheme ":" ::= scheme name, as per Section 2.1 of RFC 1738 [2].

"//" net_loc ::= network location and login information, as per Section 3.1 of RFC 1738 [2].

"/" path ::= URL path, as per Section 3.1 of RFC 1738 [2].

";" params ::= object parameters (e.g., ";type=a" as in Section 3.2.2 of RFC 1738 [2]).

"?" query ::= query information, as per Section 3.3 of RFC 1738 [2].

"#" fragment ::= fragment identifier.

Note that the fragment identifier (and the "#" that precedes it) is not considered part of the URL. However, since it is commonly used within the same string context as a URL, a parser must be able to recognize the fragment when it is present and set it aside as part of the parsing process.

The order of the components is important. If both <params> and <query> are present, the <query> information must occur after the <params>.

2.2. BNF for Relative URLs

This is a BNF-like description of the Relative Uniform Resource Locator syntax, using the conventions of RFC 822 [5], except that "|" is used to designate alternatives. Briefly, literals are quoted with "", parentheses "(" and ")" are used to group elements, optional elements are enclosed in [brackets], and elements may be preceded with <n>* to designate n or more repetitions of the following element; n defaults to 0.

This BNF also describes the generic-RL syntax for valid base URLs. Note that this differs from the URL syntax defined in RFC 1738 [2] in that all schemes are required to use a single set of reserved characters and use them consistently within the major URL components.

```
URL         = ( absoluteURL | relativeURL ) [ "#" fragment ]

absoluteURL = generic-RL | ( scheme ":" *( uchar | reserved ) )

generic-RL  = scheme ":" relativeURL

relativeURL = net_path | abs_path | rel_path

net_path    = "//" net_loc [ abs_path ]
abs_path    = "/"  rel_path
rel_path    = [ path ] [ ";" params ] [ "?" query ]

path        = fsegment *( "/" segment )
fsegment    = 1*pchar
segment     =  *pchar

params      = param *( ";" param )
param       = *( pchar | "/" )

scheme      = 1*( alpha | digit | "+" | "-" | "." )
net_loc     =  *( pchar | ";" | "?" )
query       =  *( uchar | reserved )
fragment    =  *( uchar | reserved )

pchar       = uchar | ":" | "@" | "&" | "="
uchar       = unreserved | escape
unreserved  = alpha | digit | safe | extra

escape      = "%" hex hex
hex         = digit | "A" | "B" | "C" | "D" | "E" | "F" |
                      "a" | "b" | "c" | "d" | "e" | "f"

alpha       = lowalpha | hialpha
lowalpha    = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" |
              "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" |
              "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"
hialpha     = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |
              "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |
              "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"

digit       = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
              "8" | "9"

safe        = "$" | "-" | "_" | "." | "+"
extra       = "!" | "*" | "'" | "(" | ")" | ","
national    = "{" | "}" | "|" | "\" | "^" | "~" | "[" | "]" | "`"
reserved    = ";" | "/" | "?" | ":" | "@" | "&" | "="
punctuation = "<" | ">" | "#" | "%" | <">
```

2.3. Specific Schemes and their Syntactic Categories

Each URL scheme has its own rules regarding the presence or absence of the syntactic components described in Sections 2.1 and 2.2. In addition, some schemes are never appropriate for use with relative URLs. However, since relative URLs will only be used within contexts in which they are useful, these scheme-specific differences can be ignored by the resolution process.

Within this section, we include as examples only those schemes that have a defined URL syntax in RFC 1738 [2]. The following schemes are never used with relative URLs:

mailto	Electronic Mail
news	USENET news
telnet	TELNET Protocol for Interactive Sessions

Some URL schemes allow the use of reserved characters for purposes outside the generic-RL syntax given above. However, such use is rare. Relative URLs can be used with these schemes whenever the applicable base URL follows the generic-RL syntax.

gopher	Gopher and Gopher+ Protocols
prospero	Prospero Directory Service
wais	Wide Area Information Servers Protocol

Users of gopher URLs should note that gopher-type information is almost always included at the beginning of what would be the generic-RL path. If present, this type information prevents relative-path references to documents with differing gopher-types.

Finally, the following schemes can always be parsed using the generic-RL syntax. This does not necessarily imply that relative URLs will be useful with these schemes -- that decision is left to the system implementation and the author of the base document.

file	Host-specific Files
ftp	File Transfer Protocol
http	Hypertext Transfer Protocol
nntp	USENET news using NNTP access

NOTE: Section 5 of RFC 1738 specifies that the question-mark character ("?") is allowed in an ftp or file path segment. However, this is not true in practice and is believed to be an error in the RFC. Similarly, RFC 1738 allows the reserved character semicolon (";") within an http path segment, but does not define its semantics; the correct semantics are as defined by this document for <params>.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.