Ruby trunk - Feature #12886

URI#merge doesn't handle paths correctly

10/30/2016 11:08 PM - ioquatix (Samuel Williams)

Status:	Rejected
Priority:	Normal
Assignee:	
Target version:	

Description

I feel like this should work.

```
> URI.parse("/base/uri") + URI.parse("relative")
URI::BadURIError: both URI are relative
```

The result should be URI with path = "/base/relative".

But it doesn't. It fails with an exception.

There are two ways to fix this. The first is to change the meaning of URI#absolute? to relate to the absoluteness of the path, not whether or not there is a scheme.

The second way to fix this is to directly work around the issue in merge.

In my opinion

```
> URI.parse("a/b") + URI.parse("c")
URI::BadURIError: both URI are relative
```

should also work, with a result of "a/c".

The need for the LHS of the operation to contain a scheme is not a useful requirement in practice, and in addition, I'd like to state that URI("a/c") is actually a valid URI. So, it's purely the merge function being to limited in what it will handle for no obvious reason.

Situations where this comes up: parsing a website which contains relative URLS, and you want to construct absolute URLs.

History

#1 - 10/30/2016 11:45 PM - phluid61 (Matthew Kerwin)

This ticket should be re-cast as a feature request, to allow merging of two relative references.

Incidentally:

Samuel Williams wrote:

> [...] I'd like to state that URI("a/c") is actually a valid URI.

As discussed elsewhere, it's a "relative reference", not a "URI." It parses to a URI::Generic object for pragmatic reasons.

> So, it's purely the merge function being to limited in what it will handle for no obvious reason.

Except for Internet Standard 66 (RFC 3986), section 5.1 of which says "*The term "relative" implies that a "base URI" exists against which the relative reference is applied. Aside from fragment-only references, relative references are only usable when a base URI is known.*"

You have two relative references, not a (base) URI and a relative reference.

> Situations where this comes up: parsing a website which contains relative URLS, and you want to construct absolute URLs.

Except that this doesn't construct absolute URLs, it constructs different relative references.

#2 - 10/31/2016 05:47 AM - nobu (Nobuyoshi Nakada)

- Tracker changed from Bug to Feature

#3 - 11/03/2016 04:20 PM - duerst (Martin Dürst)

Samuel Williams wrote:

> I feel like this should work.

Feelings are not enough. As Matthew already said, https://tools.ietf.org/html/rfc3986.html#section-5.2 doesn't define this. I can see at least the following problems:

1) Based on experience with RFC 3986 and it's predecessors, there would be a lot of details to get right, where often the definition of 'right' is quite unclear.

2) Not all URI schemes that may contain '/' allow or define relative processing. As an example, mailto: allows '/' in some places, but doesn't do relative processing (except that you can combine something without the scheme name with a base that has the scheme). Because when you have "/base/uri" and "relative", you don't know what the scheme is/will be, combining them would have to assume too much.

3) The fact that it's not defined in RFC 3986 may be a strong indication that it's not an operation that can be assumed to work.

4) I wonder whether there are any other languages/libraries that implement anything like the operation you propose. Of course, there are libraries working on file system paths that will do this kind of operation, but these don't exactly count.

#4 - 04/17/2017 05:28 AM - hsbt (Hiroshi SHIBATA)

note: https://github.com/ruby/ruby/pull/1469

#5 - 04/17/2017 05:34 AM - akr (Akira Tanaka)

- Status changed from Open to Rejected

I agree with Martin-sensei.
Defining hierarchical path operation for all URIs (including non-hierarchical URI such as mailto:) is curious.

#6 - 04/17/2017 06:07 AM - ioquatix (Samuel Williams)

I'm not suggesting that the operation is defined for all URIs, just ones where it makes sense.

#7 - 04/17/2017 06:46 AM - ioquatix (Samuel Williams)

> I wonder whether there are any other languages/libraries that implement anything like the operation you propose.

This use case is already working in addressable gem:

```
Addressable::URI.parse("/foo/bar")
 => #<Addressable::URI:0x3fd99b4c0ba8 URI:/foo/bar>
Addressable::URI.parse("/foo/bar") + "/baz"
 => #<Addressable::URI:0x3fd99b4b95ec URI:/baz>
Addressable::URI.parse("/foo/bar") + "baz"
 => #<Addressable::URI:0x3fd99b4b1fcc URI:/foo/baz>
```

An actual use case is parsing URIs in an HTML document with a base URI and relative URIs. It should be possible to use the URI class to correctly produce valid URIs, IMHO. The operation should be something like:

```
base_uri = URI(base[:href]) # may be relative reference
a_href = base_uri + a[:href]
```

#8 - 04/17/2017 07:34 AM - naruse (Yui NARUSE)

URL Standard defines partially merging, but it is base (absolute URL) + relative as you may know.
And the base URL must be absolute.

You need to make absolute URI from request or something before join.

#9 - 04/17/2017 09:17 AM - duerst (Martin Dürst)

naruse (Yui NARUSE) wrote:

> URL Standard defines partially merging, but it is base (absolute URL) + relative as you may know.

Which is exactly the same in RFC 3986.

> And the base URL must be absolute.
> You need to make absolute URI from request or something before join.

In other words, in the end, only absolute URIs are actionable. So we need an absolute URI eventually. The way relative resolution is defined, this means that if we have absolute + relative1 + relative2, this is interpreted as (absolute + relative1) + relative2. If we could guarantee that (absolute + relative1) + relative2 == absolute + (relative1 + relative2), then we might implement the "relative + relative" case. But proving this equivalence might turn out to be quite hard.

#10 - 04/17/2017 11:21 PM - ioquatix (Samuel Williams)

> In other words, in the end, only absolute URIs are actionable. So we need an absolute URI eventually. The way relative resolution is defined, this means that if we have absolute + relative1 + relative2, this is interpreted as (absolute + relative1) + relative2. If we could guarantee that (absolute + relative1) + relative2 == absolute + (relative1 + relative2), then we might implement the "relative + relative" case. But proving this equivalence might turn out to be quite hard.

This is exactly what I was thinking - it should be associative - (a + b) + c should be equivalent to a + (b + c). The PR I submitted allows for this, but I haven't done any testing, I don't even know where to begin with adding unit tests to Ruby. All I did was relax it so that it's possible to add together two relative URIs.

#11 - 04/26/2017 11:32 PM - ioquatix (Samuel Williams)

By the way, it's absolutely feasible to implement this correctly, simply by removing this check.

https://github.com/ruby/ruby/pull/1469/commits/7a7a33c9486c7d7b2caa7f5095bcb28fba1fe095

#12 - 04/27/2017 11:47 AM - ioquatix (Samuel Williams)

I'm giving up on this ever being improved. But for anyone who is interested in what I wanted, I ended up making a gem.

https://github.com/ioquatix/build-uri

It solves all my problems - correct, transparent, handling of absolute URLs, relative (local) paths, and additionally SCP/GIT triples.