Ruby master - Feature #18822

Ruby lack a proper method to percent-encode strings for URIs (RFC 3986)

06/09/2022 09:12 AM - byroot (Jean Boussier)

| Status:   | Closed       |
| Priority: | Normal       |
| Assignee: |              |
| Target version: |       |

**Description**

There are two fairly similar encoding methods that are often confused.

application/x-www-form-urlencoded which is how form data is encoded, and "percent-encoding" as defined by [RFC 3986](https://www.rfc-editor.org/rfc/rfc3986).

AFAIK, the only way they differ is that "form encoding" escape space characters as +, and RFC 3986 escape them as %20. Most of the time it doesn't matter, but sometimes it does.

**Ruby form and URL escape methods**

- URI.escape(" ") #=> "%20" but it was deprecated and removed (in 3.0 ?).
- ERB::Util.url_encode(" ") #=> "%20" but it's implemented with a gsub and isn't very performant. It's also awkward to have to reach for ERB
- CGI.escape(" ") #=> "+"
- URI.encode_www_form_component(" ") #=> "+"

**Unescape methods**

For unescaping, it's even more of a clear cut since URI.unescape was removed. So there's no available method that won't treat an unescaped + as simply +.

e.g. in Javascript: decodeURIComponent("foo+bar") #=> "foo+bar".

If one were to use CGI.unescape, the string might be improperly decoded: GI.unescape("foo+bar") #=> "foo bar".

**Other languages**

- Javascript encodeURI and encodeURIComponent use %20.
- PHP has urlencode using + and rawurlencode using %20.
- Python has urllib.parse.quote using %20 and urllib.parse.quote_plus using +.

**Proposal**

Since CGI already have a very performant encoder for application/x-www-form-urlencoded, I think it would make sense that it would provide another method for RFC3986.

I propose:

- CGI.url_encode(" ") #=> "%20"
- Or CGI.encode_url.
- Alias CGI.escape as CGI.encode_www_form_component
- Clarify the documentation of CGI.escape.

**Associated revisions**

Revision 3850113e - 08/16/2022 10:12 AM - byroot (Jean Boussier)

[ruby/cgi] Implement CGI.url_encode and CGI.url_decode

[Feature #18822]

Ruby is somewhat missing an RFC 3986 compliant escape method.
I forgot to mention that we'd need the decode method as well.

Proposed implementation: https://github.com/ruby/cgi/pull/26

This looks good to me and I think it's a good addition.

In the past, I've referred to this operation as "URL.encode_path" as it seems specifically related to how paths are created and escaped. However, I'd be interested to know if there is a better interpretation of that RFC w.r.t. the naming of the operation.

The counter point is, "url_encode" seems a little confusing, because it sounds like it's encoding a "URL". One could imagine an interface like url_encode(scheme, user, password, host, port, path, query, fragment).

We discussed this issue at the dev meeting. How about the following?

- Introduce CGI.escapeURIComponent(str) that behaves like CGI.escape, except that a space is encoded as %20 instead of + (as @byroot (Jean Boussier) proposed)
- Introduce CGI.unescapeURIComponent(str) that is a reverse operation.
- Introduce two aliases like CGI.escape_uri_component(str)
- Do not introduce CGI.encode_www_form_component (but improvement of the rdoc of CGI.escape is welcome)

(There was a very long discussion, but I didn't understand it due to my lack of knowledge. Please see the dev-meeting-log.)

Sounds good to me at first sight. I can work on a patch for it soon, if I notice any issue when implementing it I'll report it back here.

And thank you for the meeting notes, it's incredibly useful or me.

I forgot I already had a PR, I updated it to match the spec that was accepted during the meeting: https://github.com/ruby/cgi/pull/26

Turns out tons of projects these days rely on addressable for a more complete API, it would be nice only to need to lean on it in extreme outlier cases.

I wonder if there are other surfaces of addressable which should be pulled into core, at Discourse we lean on this:

https://github.com/discourse/discourse/blob/0df1c4eab2e1a5cd2414e88265fb9be329ac00b/lib/url_helper.rb#L21-L66

I think it is important to have "enough" tools in core MRI to deal with URLs such as https://ko.wikipedia.org/wiki/?abc%3A and somehow get a canonical URL out of it, just like web browsers are able to "figure out the right thing to do"

Since we just finished working around a nightmare scenario here @byroot (Jean Boussier), I think it is rather instructive to see a real world problem

The problem:

You get something, that is probably a URL from somewhere and need to be able to make requests to it.
- It can have a unicode domain that needs to run through an IDN converter
- It can have unicode chars that need percent encoding
- It can be unescaped, or it can be escaped (and in weird cases part escaped)

Ideally you want to normalize as well, so caching is "stronger" and does not break for identical URLs. (following the general guidelines in the RFC)

So we ended up with this monster and travesty, partly powered by URI in MRI, partly powered by addressable, 100% hack.


#9 - 08/16/2022 06:54 AM - byroot (Jean Boussier)
It's mostly just waiting for review. I'm not certain who's the maintainer though. @hsbt (Hiroshi SHIBATA)@nobu (Nobuyoshi Nakada) I added you both for review, but if someone is more suitable please let me know.

#10 - 08/16/2022 10:34 AM - ioquatix (Samuel Williams)
I'm positive on this feature, but I'm negative on the naming convention being introduced. Are we going to add aliases?

#11 - 08/16/2022 10:40 AM - byroot (Jean Boussier)
Ah right, I forgot to add the alias that were agreed upon. I'll open a followup: GCI.escape_uri_component and GCI.unescape_uri_component.

#12 - 08/16/2022 10:44 AM - byroot (Jean Boussier)
Here we go: https://github.com/ruby/cgi/pull/27

#13 - 08/16/2022 12:23 PM - ioquatix (Samuel Williams)
Should we also add aliases for escape_html and so on?

#14 - 08/16/2022 01:19 PM - byroot (Jean Boussier)
Maybe, but that would need approval. If you feel strongly about it please open a dedicated issue.

#15 - 08/16/2022 03:45 PM - byroot (Jean Boussier)
- Status changed from Open to Closed

Applied in changeset git|3850113e20b8c031529fc79de7202f61604425dd.

[ruby/cgi] Implement CGI.url_encode and CGI.url_decode
[Feature #18822]
Ruby is somewhat missing an RFC 3986 compliant escape method.
https://github.com/ruby/cgi/commit/c2729c7f33

#16 - 08/17/2022 08:06 AM - sam.saffron (Sam Saffron)
@byroot (Jean Boussier),
I am not sure the name is right here:
CGI.path_encode
with an alias of
CGI.params_encode
is far more correct.
Cause as it stands:
CGI.url_encode("https://i❤️.ws/❤️?test=❤️") will return an incorrect result.
CGI.url_encode("https://i❤️.ws") should return https://xn--i-7iq.ws/
Alternatively ... we "fix" url_encode?
Should I open a new ticket?
@sam.saffron (Sam Saffron) it's my fault for forgetting to update the commit message. CGI.url_encode was never implemented, what was is CGI.escapeURIComponent.