

2b8a44855a ▾

...

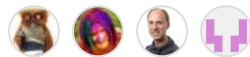
oathlib / oathlib / uri_validate.py / <> Jump to ▾



pauldekkers Use better regex for IPv6 to allow a lot more valid IPv6 addresses ✓

History

4 contributors



190 lines (136 sloc) | 5.96 KB

...

```

1  """
2  Regex for URIs
3
4  These regex are directly derived from the collected ABNF in RFC3986
5  (except for DIGIT, ALPHA and HEXDIG, defined by RFC2234).
6
7  They should be processed with re.VERBOSE.
8
9  Thanks Mark Nottingham for this code - https://gist.github.com/138549
10 """
11 import re
12
13 # basics
14
15 DIGIT = r"[\x30-\x39]"
16
17 ALPHA = r"[\x41-\x5A\x61-\x7A]"
18
19 HEXDIG = r"[\x30-\x39A-Fa-f]"
20
21 # pct-encoded = "%" HEXDIG HEXDIG
22 pct_encoded = r"%% %(HEXDIG)s %(HEXDIG)s" % locals()
23
24 # unreserved = ALPHA / DIGIT / "-" / "." / "_" / "~"
25 unreserved = r"(?: %(ALPHA)s | %(DIGIT)s | \- | \. | _ | ~ )" % locals()
26
27 # gen-delims = ":" / "/" / "?" / "#" / "[" / "]" / "@"
28 gen_delims = r"(?: : | / | \? | \# | \[ | \] | @ )"
29

```

```

30 # sub-delims      = "!" / "$" / "&" / "'" / "(" / ")"
31 #                / "*" / "+" / "," / ";" / "="
32 sub_delims = r"\""(?: ! | \$ | & | ' | \( | \) |
33             \* | \+ | , | ; | = )"\""
34
35 # pchar           = unreserved / pct-encoded / sub-delims / ":" / "@"
36 pchar = r"\""(?: %(unreserved)s | %(pct_encoded)s | %(sub_delims)s | : | @ )" % locals(
37 )
38
39 # reserved       = gen-delims / sub-delims
40 reserved = r"\""(?: %(gen_delims)s | %(sub_delims)s )" % locals()
41
42
43 # scheme
44
45 # scheme         = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )
46 scheme = r"\""%(ALPHA)s (?: %(ALPHA)s | %(DIGIT)s | \+ | \- | \. )*" % locals()
47
48
49 # authority
50
51 # dec-octet      = DIGIT                      ; 0-9
52 #                / %x31-39 DIGIT              ; 10-99
53 #                / "1" 2DIGIT                  ; 100-199
54 #                / "2" %x30-34 DIGIT           ; 200-249
55 #                / "25" %x30-35                ; 250-255
56 dec_octet = r"\"\"\"(?: %(DIGIT)s |
57             [\x31-\x39] %(DIGIT)s |
58             1 %(DIGIT)s{2} |
59             2 [\x30-\x34] %(DIGIT)s |
60             25 [\x30-\x35]
61             )
62 \"\"\" % locals()
63
64 # IPv4address    = dec-octet "." dec-octet "." dec-octet "." dec-octet
65 IPv4address = r"\"\"\"%(dec_octet)s \. %(dec_octet)s \. %(dec_octet)s \. %(dec_octet)s" % locals(
66 )
67
68 # IPv6address
69 IPv6address = r"\"\"\"([A-Fa-f0-9:]+:~)+[A-Fa-f0-9:~]+"
70
71 # IPvFuture      = "v" 1*HEXDIG "." 1*( unreserved / sub-delims / ":" )
72 IPvFuture = r"\"\"\"v %(HEXDIG)s+ \. (?: %(unreserved)s | %(sub_delims)s | : )+" % locals()
73
74 # IP-literal     = "[" ( IPv6address / IPvFuture  ) "]"
75 IP_literal = r"\"\"\"\[ (?: %(IPv6address)s | %(IPvFuture)s ) \]" % locals()
76
77 # reg-name       = *( unreserved / pct-encoded / sub-delims )
78 reg_name = r"\"\"\"(?: %(unreserved)s | %(pct_encoded)s | %(sub_delims)s )*" % locals()

```

```

79
80 # userinfo      = *( unreserved / pct-encoded / sub-delims / ":" )
81 userinfo = r"(?: %(unreserved)s | %(pct_encoded)s | %(sub_delims)s | : )" % locals(
82 )
83
84 # host          = IP-literal / IPv4address / reg-name
85 host = r"(?: %(IP_literal)s | %(IPv4address)s | %(reg_name)s )" % locals()
86
87 # port         = *DIGIT
88 port = r"(?: %(DIGIT)s )" % locals()
89
90 # authority    = [ userinfo "@" ] host [ ":" port ]
91 authority = r"(?: %(userinfo)s @)? %(host)s (?: : %(port)s)?" % locals()
92
93 # Path
94
95 # segment      = *pchar
96 segment = r"%(pchar)s*" % locals()
97
98 # segment-nz   = 1*pchar
99 segment_nz = r"%(pchar)s+" % locals()
100
101 # segment-nz-nc = 1*( unreserved / pct-encoded / sub-delims / "@" )
102 #               ; non-zero-length segment without any colon ":"
103 segment_nz_nc = r"(?: %(unreserved)s | %(pct_encoded)s | %(sub_delims)s | @ )+" % locals()
104
105 # path-abempty = *( "/" segment )
106 path_abempty = r"(?: / %(segment)s )" % locals()
107
108 # path-absolute = "/" [ segment-nz *( "/" segment ) ]
109 path_absolute = r"/ (?: %(segment_nz)s (?: / %(segment)s )* )" % locals()
110
111 # path-noscheme = segment-nz-nc *( "/" segment )
112 path_noscheme = r"%(segment_nz_nc)s (?: / %(segment)s )" % locals()
113
114 # path-rootless = segment-nz *( "/" segment )
115 path_rootless = r"%(segment_nz)s (?: / %(segment)s )" % locals()
116
117 # path-empty    = 0<pchar>
118 path_empty = r"" # FIXME
119
120 # path          = path-abempty      ; begins with "/" or is empty
121 #               / path-absolute    ; begins with "/" but not "/"
122 #               / path-noscheme     ; begins with a non-colon segment
123 #               / path-rootless    ; begins with a segment
124 #               / path-empty       ; zero characters
125 path = r"""(?: %(path_abempty)s |
126              %(path_absolute)s |
127              %(path_noscheme)s |

```

```

128         %(path_rootless)s |
129         %(path_empty)s
130     )
131     """ % locals()
132
133     ### Query and Fragment
134
135     # query          = *( pchar / "/" / "?" )
136     query = r"(?: %(pchar)s | / | \? )" % locals()
137
138     # fragment       = *( pchar / "/" / "?" )
139     fragment = r"(?: %(pchar)s | / | \? )" % locals()
140
141     # URIs
142
143     # hier-part      = "//" authority path-abempty
144     #                / path-absolute
145     #                / path-rootless
146     #                / path-empty
147     hier_part = r"""(?: (?: // %(authority)s %(path_abempty)s ) |
148                        %(path_absolute)s |
149                        %(path_rootless)s |
150                        %(path_empty)s
151                    )
152     """ % locals()
153
154     # relative-part = "//" authority path-abempty
155     #                / path-absolute
156     #                / path-noscheme
157     #                / path-empty
158     relative_part = r"""(?: (?: // %(authority)s %(path_abempty)s ) |
159                            %(path_absolute)s |
160                            %(path_noscheme)s |
161                            %(path_empty)s
162                        )
163     """ % locals()
164
165     # relative-ref = relative-part [ "?" query ] [ "#" fragment ]
166     relative_ref = r"% (relative_part)s (?: \? %(query)s)? (?: \# %(fragment)s)?" % locals(
167     )
168
169     # URI           = scheme ":" hier-part [ "?" query ] [ "#" fragment ]
170     URI = r"^ (?: %(scheme)s : %(hier_part)s (?: \? %(query)s )? (?: \# %(fragment)s )? )" % locals(
171     )
172
173     # URI-reference = URI / relative-ref
174     URI_reference = r"^ (?: %(URI)s | %(relative_ref)s )" % locals()
175
176     # absolute-URI  = scheme ":" hier-part [ "?" query ]

```

```
177 absolute_URI = r"^(?: %(scheme)s : %(hier_part)s (?: \? %(query)s )? )$" % locals(
178 )
179
180
181 def is_uri(uri):
182     return re.match(URI, uri, re.VERBOSE)
183
184
185 def is_uri_reference(uri):
186     return re.match(URI_reference, uri, re.VERBOSE)
187
188
189 def is_absolute_uri(uri):
190     return re.match(absolute_URI, uri, re.VERBOSE)
```