oauthlib / **oauthlib** Public

Code | Issues 72 | Pull requests 11 | Actions | Security 1 | Insights

2b8a44855a

oauthlib / oauthlib / **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

```python
"""
Regex for URIs

These regex are directly derived from the collected ABNF in RFC3986
(except for DIGIT, ALPHA and HEXDIG, defined by RFC2234).

They should be processed with re.VERBOSE.

Thanks Mark Nottingham for this code - https://gist.github.com/138549
"""
import re

# basics

DIGIT = r"[\x30-\x39]"

ALPHA = r"[\x41-\x5A\x61-\x7A]"

HEXDIG = r"[\x30-\x39A-Fa-f]"

#   pct-encoded   = "%" HEXDIG HEXDIG
pct_encoded = r" %% %(HEXDIG)s %(HEXDIG)s" % locals()

#   unreserved    = ALPHA / DIGIT / "-" / "." / "_" / "~"
unreserved = r"(?: %(ALPHA)s | %(DIGIT)s | \- | \. | _ | ~ )" % locals()

# gen-delims    = ":" / "/" / "?" / "#" / "[" / "]" / "@"
gen_delims = r"(?: : | / | \? | \# | \[ | \] | @ )"

```

```
#   sub-delims    = "!" / "$" / "&" / "'" / "(" / ")"
#                 / "*" / "+" / "," / ";" / "="
sub_delims = r"""(?: ! | \$ | & | ' | \( | \) |
                     \* | \+ | , | ; | = )"""

#   pchar         = unreserved / pct-encoded / sub-delims / ":" / "@"
pchar = r"(?: %(unreserved)s | %(pct_encoded)s | %(sub_delims)s | : | @ )" % locals(
)

#   reserved      = gen-delims / sub-delims
reserved = r"(?: %(gen_delims)s | %(sub_delims)s )" % locals()


# scheme

#   scheme        = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )
scheme = r"%(ALPHA)s (?: %(ALPHA)s | %(DIGIT)s | \+ | \- | \. )*" % locals()



# authority

#   dec-octet     = DIGIT                 ; 0-9
#                 / %x31-39 DIGIT         ; 10-99
#                 / "1" 2DIGIT            ; 100-199
#                 / "2" %x30-34 DIGIT     ; 200-249
#                 / "25" %x30-35          ; 250-255
dec_octet = r"""(?: %(DIGIT)s |
                    [\x31-\x39] %(DIGIT)s |
                    1 %(DIGIT)s{2} |
                    2 [\x30-\x34] %(DIGIT)s |
                    25 [\x30-\x35]
                )
""" % locals()

#  IPv4address   = dec-octet "." dec-octet "." dec-octet "." dec-octet
IPv4address = r"%(dec_octet)s \. %(dec_octet)s \. %(dec_octet)s \. %(dec_octet)s" % locals(
)

#   IPv6address
IPv6address = r"([A-Fa-f0-9:]+:+)+[A-Fa-f0-9]+"

#   IPvFuture     = "v" 1*HEXDIG "." 1*( unreserved / sub-delims / ":" )
IPvFuture = r"v %(HEXDIG)s+ \. (?: %(unreserved)s | %(sub_delims)s | : )+" % locals()

#   IP-literal    = "[" ( IPv6address / IPvFuture  ) "]"
IP_literal = r"\[ (?: %(IPv6address)s | %(IPvFuture)s ) \]" % locals()

#   reg-name      = *( unreserved / pct-encoded / sub-delims )
reg_name = r"(?: %(unreserved)s | %(pct_encoded)s | %(sub_delims)s )*" % locals()
```

```

#   userinfo      = *( unreserved / pct-encoded / sub-delims / ":" )
userinfo = r"(?: %(unreserved)s | %(pct_encoded)s | %(sub_delims)s | : )" % locals(
)

#   host          = IP-literal / IPv4address / reg-name
host = r"(?: %(IP_literal)s | %(IPv4address)s | %(reg_name)s )" % locals()

#   port          = *DIGIT
port = r"(?: %(DIGIT)s )*" % locals()

#   authority     = [ userinfo "@" ] host [ ":" port ]
authority = r"(?: %(userinfo)s @)? %(host)s (?: : %(port)s)?" % locals()

# Path

#   segment       = *pchar
segment = r"%(pchar)s*" % locals()

#   segment-nz    = 1*pchar
segment_nz = r"%(pchar)s+" % locals()

#   segment-nz-nc = 1*( unreserved / pct-encoded / sub-delims / "@" )
#                 ; non-zero-length segment without any colon ":"
segment_nz_nc = r"(?: %(unreserved)s | %(pct_encoded)s | %(sub_delims)s | @ )+" % locals()

#   path-abempty  = *( "/" segment )
path_abempty = r"(?: / %(segment)s )*" % locals()

#   path-absolute = "/" [ segment-nz *( "/" segment ) ]
path_absolute = r"/ (?: %(segment_nz)s (?: / %(segment)s )* )?" % locals()

#   path-noscheme = segment-nz-nc *( "/" segment )
path_noscheme = r"%(segment_nz_nc)s (?: / %(segment)s )*" % locals()

#   path-rootless = segment-nz *( "/" segment )
path_rootless = r"%(segment_nz)s (?: / %(segment)s )*" % locals()

#   path-empty    = 0<pchar>
path_empty = r""  # FIXME

#   path          = path-abempty    ; begins with "/" or is empty
#                 / path-absolute   ; begins with "/" but not "//"
#                 / path-noscheme   ; begins with a non-colon segment
#                 / path-rootless   ; begins with a segment
#                 / path-empty      ; zero characters
path = r"""(?: %(path_abempty)s |
               %(path_absolute)s |
               %(path_noscheme)s |
                %(path_rootless)s |
                %(path_empty)s
            )
""" % locals()

### Query and Fragment

#   query         = *( pchar / "/" / "?" )
query = r"(?: %(pchar)s | / | \? )*" % locals()

#   fragment      = *( pchar / "/" / "?" )
fragment = r"(?: %(pchar)s | / | \? )*" % locals()

# URIs

#   hier-part     = "//" authority path-abempty
#                 / path-absolute
#                 / path-rootless
#                 / path-empty
hier_part = r"""(?: (?: // %(authority)s %(path_abempty)s ) |
                    %(path_absolute)s |
                    %(path_rootless)s |
                    %(path_empty)s
                )
""" % locals()

#   relative-part = "//" authority path-abempty
#                 / path-absolute
#                 / path-noscheme
#                 / path-empty
relative_part = r"""(?: (?: // %(authority)s %(path_abempty)s ) |
                        %(path_absolute)s |
                        %(path_noscheme)s |
                        %(path_empty)s
                    )
""" % locals()

# relative-ref  = relative-part [ "?" query ] [ "#" fragment ]
relative_ref = r"%(relative_part)s (?: \? %(query)s)? (?: \# %(fragment)s)?" % locals(
)

# URI           = scheme ":" hier-part [ "?" query ] [ "#" fragment ]
URI = r"^(?: %(scheme)s : %(hier_part)s (?: \? %(query)s )? (?: \# %(fragment)s )? )$" % locals(
)

#   URI-reference = URI / relative-ref
URI_reference = r"^(?: %(URI)s | %(relative_ref)s )$" % locals()

#   absolute-URI  = scheme ":" hier-part [ "?" query ]
```

```python
absolute_URI = r"^(?: %(scheme)s : %(hier_part)s (?: \? %(query)s )? )$" % locals(
)


def is_uri(uri):
    return re.match(URI, uri, re.VERBOSE)


def is_uri_reference(uri):
    return re.match(URI_reference, uri, re.VERBOSE)


def is_absolute_uri(uri):
    return re.match(absolute_URI, uri, re.VERBOSE)
```