

python / cpython Public

<> Code Issues 5k+ Pull requests 2.1k Actions Projects Security and q

Commit 6eb6c5d



serhiy-storchaka authored on Jun 13, 2025 · 109 / 113 · Verified

[gh-135462](#): Fix quadratic complexity in processing special input in HTMLParser (GH-135464)

End-of-file errors are now handled according to the HTML5 specs -- comments and declarations are automatically closed, tags are ignored.

main (#135464) · v3.15.0a8 ... v3.15.0a1

1 parent [14c1d09](#) commit 6eb6c5d

3 files changed

+111 -31

Top

- html
 - parser.py
 - test
 - test_htmlparser.py
 - Misc/NEWS.d/next/Security
 - 2025-06-13-15-55-22.gh-issue-135462.KBeJpc.rst



Lib/html/parser.py



@@ -27,6 +27,7 @@

```

27 27 attr_charref = re.compile(r'&#[0-9]+#[xX][0-9a-fA-F]+|[a-zA-Z][a-zA-Z0-9]*
    [;=?]')

28 28
29 29 starttagopen = re.compile('<[a-zA-Z]')
30 + endtagopen = re.compile('</[a-zA-Z]')
30 31 piclose = re.compile('>')
31 32 commentclose = re.compile(r'--\s*>')
32 33 # Note:
    @@ -195,25 +196,43 @@ def goahead(self, end):
195 196         k = self.parse_pi(i)
196 197         elif startswith("<", i):
197 198         k = self.parse_html_declaration(i)
198 -         elif (i + 1) < n:
199 +         elif (i + 1) < n or end:
199 200             self.handle_data("<")
200 201             k = i + 1
201 202         else:
202 203             break
203 204         if k < 0:
204 205             if not end:
205 206                 break
206 -         k = rawdata.find('>', i + 1)
207 -         if k < 0:
208 -             k = rawdata.find('<', i + 1)
209 -             if k < 0:
210 -                 k = i + 1
211 -         else:
212 -             k += 1
213 -         if self.convert_charrefs and not self.cdata_elem:
214 -             self.handle_data(unescape(rawdata[i:k]))
207 +         if starttagopen.match(rawdata, i): # < + letter
208 +             pass
209 +         elif startswith("</", i):
210 +             if i + 2 == n:
211 +                 self.handle_data("</")
212 +             elif endtagopen.match(rawdata, i): # </ + letter
213 +                 pass
214 +             else:
215 +                 # bogus comment

```

```

216 +         self.handle_comment(rawdata[i+2:])
217 +     elif startswith("<!--", i):
218 +         j = n
219 +         for suffix in ("--!", "--", "-"):
220 +             if rawdata.endswith(suffix, i+4):
221 +                 j -= len(suffix)
222 +                 break
223 +         self.handle_comment(rawdata[i+4:j])
224 +     elif startswith("<![CDATA[", i):
225 +         self.unknown_decl(rawdata[i+3:])
226 +     elif rawdata[i:i+9].lower() == '<!doctype':
227 +         self.handle_decl(rawdata[i+2:])
228 +     elif startswith("<!", i):
229 +         # bogus comment
230 +         self.handle_comment(rawdata[i+2:])
231 +     elif startswith("<?", i):
232 +         self.handle_pi(rawdata[i+2:])
215 233         else:
216 -         self.handle_data(rawdata[i:k])
234 +         raise AssertionError("we should not get here!")
235 +         k = n
217 236         i = self.updatepos(i, k)
218 237     elif startswith("&#", i):
219 238         match = charref.match(rawdata, i)

```

Lib/test/test_htmlparser.py

```

@@ -5,6 +5,7 @@
5 5     import unittest
6 6
7 7     from unittest.mock import patch
8 + from test import support
8 9
9 10
10 11     class EventCollector(html.parser.HTMLParser):
@@ -430,28 +431,34 @@ def test_tolerant_parsing(self):
430 431         ('data', '<'),
431 432         ('starttag', 'bc<', [('a', None)]),
432 433         ('endtag', 'html'),

```

433	-		('data', '\n'),
434	-		('comment', '/img'),
435	-		('endtag', 'html<'))
434	+		('data', '\n'))
436	435		
437	436		def test_starttag_junk_chars(self):
437	+		self._run_check("<", [('data', '<')])
438	+		self._run_check("<>", [('data', '<>')])
439	+		self._run_check("< >", [('data', '< >')])
440	+		self._run_check("< ", [('data', '< ')])
438	441		self._run_check("</>", [])
442	+		self._run_check("<\$>", [('data', '<\$>')])
439	443		self._run_check("</\$>", [('comment', '\$')])
440	444		self._run_check("</", [('data', '</')])
441	-		self._run_check("</a", [('data', '</a')])
445	+		self._run_check("</a", [])
446	+		self._run_check("</ a>", [('endtag', 'a')])
447	+		self._run_check("</ a", [('comment', ' a')])
442	448		self._run_check("<a<a>", [('starttag', 'a<a', [])])
443	449		self._run_check("</a<a>", [('endtag', 'a<a')])
444	-		self._run_check("<!", [('data', '<!')])
445	-		self._run_check("<a", [('data', '<a')])
446	-		self._run_check("<a foo='bar'", [('data', "<a foo='bar'")])
447	-		self._run_check("<a foo='bar", [('data', "<a foo='bar'")])
448	-		self._run_check("", [('data', "")])
449	-		self._run_check("", [('data', "")])
450	+		self._run_check("<!", [('comment', '!')])
451	+		self._run_check("<a", [])
452	+		self._run_check("<a foo='bar'", [])
453	+		self._run_check("<a foo='bar", [])
454	+		self._run_check("", [])
455	+		self._run_check("", [])
450	456		self._run_check("<a\$>", [('starttag', 'a\$', [])])
451	457		self._run_check("<a\$b>", [('starttag', 'a\$b', [])])
452	458		self._run_check("<a\$b/>", [('startendtag', 'a\$b', [])])
453	459		self._run_check("<a\$b >", [('starttag', 'a\$b', [])])
454	460		self._run_check("<a\$b />", [('startendtag', 'a\$b', [])])
461	+		self._run_check("</a\$b>", [('endtag', 'a\$b')])
455	462		
456	463		def test_slashes_in_starttag(self):

```

457 464         self._run_check('<a foo="var"/>', [('startendtag', 'a', [('foo',
'var')]))])

```



```

@@ -576,21 +583,50 @@ def test_EOF_in_charref(self):

```

```

576 583         for html, expected in data:
577 584             self._run_check(html, expected)

```

```

578 585

```

```

579 -     def test_EOF_in_comments_or_decls(self):

```

```

586 +     def test_eof_in_comments(self):

```

```

580 587         data = [

```

```

581 -             ('<!', [('data', '<!')]),
582 -             ('<!--', [('data', '<!--')]),
583 -             ('<!---', [('data', '<!---')]),
584 -             ('<![', [('data', '<![')]),
585 -             ('<![CDATA[', [('data', '<![CDATA[')]),
586 -             ('<![CDATA[x', [('data', '<![CDATA[x')]),
587 -             ('<!DOCTYPE', [('data', '<!DOCTYPE')]),
588 -             ('<!DOCTYPE HTML', [('data', '<!DOCTYPE HTML')]),

```

```

588 +             ('<!--', [('comment', '')]),
589 +             ('<!---', [('comment', '')]),
590 +             ('<!---', [('comment', '')]),
591 +             ('<!---', [('comment', '-')]),
592 +             ('<!---', [('comment', '--')]),
593 +             ('<!--!', [('comment', '')]),
594 +             ('<!--!', [('comment', '-!')]),
595 +             ('<!--!>', [('comment', '-!>')]),
596 +             ('<!--foo', [('comment', 'foo')]),
597 +             ('<!--foo-', [('comment', 'foo')]),
598 +             ('<!--foo--', [('comment', 'foo')]),
599 +             ('<!--foo--!', [('comment', 'foo')]),
600 +             ('<!--<!--', [('comment', '<!')]),
601 +             ('<!--<!--!', [('comment', '<!')]),

```

```

589 602         ]

```

```

590 603         for html, expected in data:

```

```

591 604             self._run_check(html, expected)

```

```

605 +
606 +     def test_eof_in_declarations(self):

```

```

607 +         data = [
608 +             ('<!', [('comment', '')]),
609 +             ('<!--', [('comment', '-')]),

```

```

610 +         ('<![', [( 'comment', '[')]),
611 +         ('<![CDATA[', [( 'unknown decl', 'CDATA[')]),
612 +         ('<![CDATA[x', [( 'unknown decl', 'CDATA[x')]),
613 +         ('<![CDATA[x]', [( 'unknown decl', 'CDATA[x'])]),
614 +         ('<![CDATA[x]]', [( 'unknown decl', 'CDATA[x]]')]),
615 +         ('<!DOCTYPE', [( 'decl', 'DOCTYPE')]),
616 +         ('<!DOCTYPE ', [( 'decl', 'DOCTYPE ')]),
617 +         ('<!DOCTYPE html', [( 'decl', 'DOCTYPE html')]),
618 +         ('<!DOCTYPE html ', [( 'decl', 'DOCTYPE html ')]),
619 +         ('<!DOCTYPE html PUBLIC', [( 'decl', 'DOCTYPE html PUBLIC')]),
620 +         ('<!DOCTYPE html PUBLIC "foo', [( 'decl', 'DOCTYPE html PUBLIC
        "foo')]),
621 +         ('<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "foo',
622 +         [( 'decl', 'DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "foo')]),
623 +     ]
624 +     for html, expected in data:
625 +         self._run_check(html, expected)
626 +
592 627         def test_bogus_comments(self):
593 -         html = ('<! not really a comment >'
628 +         html = ('<!ELEMENT br EMPTY>'
629 +         '<! not really a comment >'
594 630         '<! not a comment either -->'
595 631         '<! -- close enough -->'
596 632         '<!><!-- this was an empty comment>'
@@ -604,6 +640,7 @@ def test_bogus_comments(self):
604 640         '<![CDATA]]>' # required '[' after CDATA
605 641         )
606 642         expected = [
643 +         ('comment', 'ELEMENT br EMPTY'),
607 644         ('comment', ' not really a comment '),
608 645         ('comment', ' not a comment either --'),
609 646         ('comment', ' -- close enough --'),
@@ -684,6 +721,26 @@ def test_convert_charrefs_dropped_text(self):
684 721         ('endtag', 'a'), ('data', ' bar & baz')]
685 722         )
686 723
724 +     @support.requires_resource('cpu')

```

```
725 +     def test_eof_no_quadratic_complexity(self):
726 +         # Each of these examples used to take about an hour.
727 +         # Now they take a fraction of a second.
728 +         def check(source):
729 +             parser = html.parser.HTMLParser()
730 +             parser.feed(source)
731 +             parser.close()
732 +             n = 120_000
733 +             check("<a " * n)
734 +             check("<a a=" * n)
735 +             check("</a " * 14 * n)
736 +             check("</a a=" * 11 * n)
737 +             check("<!--" * 4 * n)
738 +             check("<!" * 60 * n)
739 +             check("<?" * 19 * n)
740 +             check("</$" * 15 * n)
741 +             check("<![CDATA[" * 9 * n)
742 +             check("<!doctype" * 35 * n)
743 +
```

687 744

688 745 class AttributesTestCase(TestCaseBase):

689 746



...5-06-13-15-55-22.gh-issue-135462.KBeJpc.rst



... @@ -0,0 +1,4 @@

```
1 + Fix quadratic complexity in processing specially crafted input in
2 + :class:`html.parser.HTMLParser`. End-of-file errors are now handled according
3 + to the HTML5 specs -- comments and declarations are automatically closed,
4 + tags are ignored.
```

Comments 0