

python / cpython Public

<> Code Issues 5k+ Pull requests 2.2k Actions Projects Security and q

⚠ This commit does not belong to any branch on this repository, and may belong to a fork outside of the repository.

Commit 8d1b3df



serhiy-storchaka authored on Jul 3, 2025 · 68 / 75 · Verified

[3.9] [gh-135462](#): Fix quadratic complexity in processing special input in HTMLParser ([GH-135464](#)) ([GH-135486](#))

End-of-file errors are now handled according to the HTML5 specs -- comments and declarations are automatically closed, tags are ignored. (cherry picked from commit [6eb6c5d](#))

· v3.9.25 ... 3.9

1 parent [4f28afe](#) commit 8d1b3df

3 files changed +117 -23 lines changed

Top

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

3 files changed +117 -23 lines changed



```

Lib/html/parser.py
@@ -25,6 +25,7 @@
25 25 charref = re.compile('&#(?:[0-9]+|[xX][0-9a-fA-F+])[^0-9a-fA-F]')
26 26
27 27 starttagopen = re.compile('<[a-zA-Z]')
28 + endtagopen = re.compile('</[a-zA-Z]')
28 29 piclose = re.compile('>')
29 30 commentclose = re.compile(r'--\s*>')
30 31 # Note:

@@ -176,25 +177,43 @@ def goahead(self, end):
176 177         k = self.parse_pi(i)
177 178         elif startswith("<!", i):
178 179             k = self.parse_html_declaration(i)
179 -         elif (i + 1) < n:
180 +         elif (i + 1) < n or end:
180 181             self.handle_data("<")
181 182             k = i + 1
182 183         else:
183 184             break
184 185         if k < 0:
185 186             if not end:
186 187                 break
187 -         k = rawdata.find('>', i + 1)
188 -         if k < 0:
189 -             k = rawdata.find('<', i + 1)
190 -             if k < 0:
191 -                 k = i + 1
192 -             else:
193 -                 k += 1
194 -             if self.convert_charrefs and not self.cdata_elem:
195 -                 self.handle_data(unescape(rawdata[i:k]))
188 +         if starttagopen.match(rawdata, i): # < + letter
189 +             pass
190 +         elif startswith("</", i):
191 +             if i + 2 == n:
192 +                 self.handle_data("</")
193 +             elif endtagopen.match(rawdata, i): # </ + letter
194 +                 pass
195 +             else:

```

```

196 +             # bogus comment
197 +             self.handle_comment(rawdata[i+2:])
198 +             elif startswith("<!--", i):
199 +                 j = n
200 +                 for suffix in ("--!", "--", "-"):
201 +                     if rawdata.endswith(suffix, i+4):
202 +                         j -= len(suffix)
203 +                         break
204 +                 self.handle_comment(rawdata[i+4:j])
205 +             elif startswith("<![CDATA[", i):
206 +                 self.unknown_decl(rawdata[i+3:])
207 +             elif rawdata[i:i+9].lower() == '<!doctype':
208 +                 self.handle_decl(rawdata[i+2:])
209 +             elif startswith("<!", i):
210 +                 # bogus comment
211 +                 self.handle_comment(rawdata[i+2:])
212 +             elif startswith("<?", i):
213 +                 self.handle_pi(rawdata[i+2:])
196 214             else:
197 -                 self.handle_data(rawdata[i:k])
215 +                 raise AssertionError("we should not get here!")
216 +                 k = n
198 217             i = self.updatepos(i, k)
199 218             elif startswith("&#", i):
200 219                 match = charref.match(rawdata, i)

```

Lib/test/test_htmlparser.py

```

@@ -4,6 +4,8 @@
4 4     import pprint
5 5     import unittest
6 6
7 +   from test import support
8 +
7 9
8 10    class EventCollector(html.parser.HTMLParser):
9 11
@@ -391,28 +393,34 @@ def test_tolerant_parsing(self):
391 393                ('data', '<'),

```

```

392 394         ('starttag', 'bc<', [('a', None)]),
393 395         ('endtag', 'html'),
394 -         ('data', '\n", [('data', '<>')])
401 +         self._run_check("< >", [('data', '< >')])
402 +         self._run_check("< ", [('data', '< ')])
399 403         self._run_check("</>", [])
404 +         self._run_check("<$>", [('data', '<$>')])
400 405         self._run_check("</$>", [('comment', '$')])
401 406         self._run_check("</", [('data', '</')])
402 -         self._run_check("</a", [('data', '</a')])
407 +         self._run_check("</a", [])
408 +         self._run_check("</ a>", [('endtag', 'a')])
409 +         self._run_check("</ a", [('comment', ' a')])
403 410         self._run_check("<a<a>", [('starttag', 'a<a', [ ])])
404 411         self._run_check("</a<a>", [('endtag', 'a<a')])
405 -         self._run_check("<!", [('data', '<!')])
406 -         self._run_check("<a", [('data', '<a')])
407 -         self._run_check("<a foo='bar'", [('data', "<a foo='bar'")])
408 -         self._run_check("<a foo='bar", [('data', "<a foo='bar'")])
409 -         self._run_check("<a foo='>", [('data', "<a foo='>")])
410 -         self._run_check("<a foo='>", [('data', "<a foo='>")])
412 +         self._run_check("<!", [('comment', '!')])
413 +         self._run_check("<a", [])
414 +         self._run_check("<a foo='bar'", [])
415 +         self._run_check("<a foo='bar", [])
416 +         self._run_check("<a foo='>", [])
417 +         self._run_check("<a foo='>", [])
411 418         self._run_check("<a$a>", [('starttag', 'a$', [ ])])
412 419         self._run_check("<a$b>", [('starttag', 'a$b', [ ])])
413 420         self._run_check("<a$b/>", [('startendtag', 'a$b', [ ])])
414 421         self._run_check("<a$b >", [('starttag', 'a$b', [ ])])
415 422         self._run_check("<a$b />", [('startendtag', 'a$b', [ ])])
423 +         self._run_check("</a$b>", [('endtag', 'a$b')])

```

```

416 424
417 425     def test_slashes_in_starttag(self):
418 426         self._run_check('<a foo="var"/>', [('startendtag', 'a', [('foo',
'var')]))])

```



```
@@ -537,13 +545,56 @@ def test_eof_in_charref(self):
```

```

537 545         for html, expected in data:
538 546             self._run_check(html, expected)
539 547

```

```

540 -     def test_broken_comments(self):
541 -         html = ('<! not really a comment >'

```

```

548 +     def test_eof_in_comments(self):
549 +         data = [
550 +             ('<!--', [('comment', ' ')]),
551 +             ('<!---', [('comment', ' ')]),
552 +             ('<!---', [('comment', ' ')]),
553 +             ('<!---', [('comment', '- ')]),
554 +             ('<!---', [('comment', '- - ')]),
555 +             ('<!--!', [('comment', ' ')]),
556 +             ('<!--!', [('comment', '-! ')]),
557 +             ('<!--!>', [('comment', '-!> ')]),
558 +             ('<!--foo', [('comment', 'foo ')]),
559 +             ('<!--foo-', [('comment', 'foo ')]),
560 +             ('<!--foo--', [('comment', 'foo ')]),
561 +             ('<!--foo--!', [('comment', 'foo ')]),
562 +             ('<!--<!--', [('comment', '<! ')]),
563 +             ('<!--<!--!', [('comment', '<! ')]),
564 +         ]
565 +         for html, expected in data:
566 +             self._run_check(html, expected)
567 +
568 +     def test_eof_in_declarations(self):
569 +         data = [
570 +             ('<!', [('comment', ' ')]),
571 +             ('<!--', [('comment', '- ')]),
572 +             ('<![', [('comment', '[')]),
573 +             ('<![CDATA[', [('unknown decl', 'CDATA[ ')]),
574 +             ('<![CDATA[x', [('unknown decl', 'CDATA[x ')]),
575 +             ('<![CDATA[x]', [('unknown decl', 'CDATA[x ')]),
576 +             ('<![CDATA[x]]', [('unknown decl', 'CDATA[x]] ')]),

```

```

577 +         ('<!DOCTYPE', [('decl', 'DOCTYPE')]),
578 +         ('<!DOCTYPE ', [('decl', 'DOCTYPE ')]),
579 +         ('<!DOCTYPE html', [('decl', 'DOCTYPE html')]),
580 +         ('<!DOCTYPE html ', [('decl', 'DOCTYPE html ')]),
581 +         ('<!DOCTYPE html PUBLIC', [('decl', 'DOCTYPE html PUBLIC')]),
582 +         ('<!DOCTYPE html PUBLIC "foo', [('decl', 'DOCTYPE html PUBLIC
"foo')]),
583 +         ('<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "foo',
584 +         [('decl', 'DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"foo')]),
585 +     ]
586 +     for html, expected in data:
587 +         self._run_check(html, expected)
588 +
589 +     def test_bogus_comments(self):
590 +         html = ('<!ELEMENT br EMPTY>'
591 +                '<! not really a comment >'
592 +                '<! not a comment either -->'
593 +                '<! -- close enough -->'
594 +                '<!><!-- this was an empty comment>'
595 +                '<!!! another bogus comment !!!>')
596 +         expected = [
597 +             ('comment', 'ELEMENT br EMPTY'),
598 +             ('comment', ' not really a comment '),
599 +             ('comment', ' not a comment either --'),
600 +             ('comment', ' -- close enough --'),
601 +
602 +             @@ -598,6 +649,26 @@ def test_convert_charrefs_dropped_text(self):
603 +
604 +             ('endtag', 'a'), ('data', ' bar & baz')]
605 +         )
606 +
607 +     @support.requires_resource('cpu')
608 +     def test_eof_no_quadratic_complexity(self):
609 +         # Each of these examples used to take about an hour.
610 +         # Now they take a fraction of a second.
611 +         def check(source):
612 +             parser = html.parser.HTMLParser()
613 +             parser.feed(source)
614 +             parser.close()
615 +         n = 120_000

```

```
661 +     check("<a " * n)
662 +     check("<a a=" * n)
663 +     check("</a " * 14 * n)
664 +     check("</a a=" * 11 * n)
665 +     check("<!--" * 4 * n)
666 +     check("<!" * 60 * n)
667 +     check("<?" * 19 * n)
668 +     check("</ $" * 15 * n)
669 +     check("<![CDATA[" * 9 * n)
670 +     check("<!doctype" * 35 * n)
671 +
```

```
601 672
```

```
602 673     class AttributesTestCase(TestCaseBase):
```

```
603 674
```



...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