

c2f392e0be ▾

...

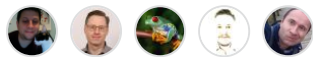
mako / mako / ext / extract.py / <> Jump to ▾



zzzeek happy new year ... ✓

History

5 contributors



129 lines (112 sloc) | 4.55 KB

...

```

1  # ext/extract.py
2  # Copyright 2006-2022 the Mako authors and contributors <see AUTHORS file>
3  #
4  # This module is part of Mako and is released under
5  # the MIT License: http://www.opensource.org/licenses/mit-license.php
6
7  from io import BytesIO
8  from io import StringIO
9  import re
10
11 from mako import lexer
12 from mako import parsetree
13
14
15 class MessageExtractor:
16     use_bytes = True
17
18     def process_file(self, fileobj):
19         template_node = lexer.Lexer(
20             fileobj.read(), input_encoding=self.config["encoding"]
21         ).parse()
22         yield from self.extract_nodes(template_node.get_children())
23
24     def extract_nodes(self, nodes):
25         translator_comments = []
26         in_translator_comments = False
27         input_encoding = self.config["encoding"] or "ascii"
28         comment_tags = list(
29             filter(None, re.split(r"\s+", self.config["comment-tags"])))

```

...

```

30     )
31
32     for node in nodes:
33         child_nodes = None
34         if (
35             in_translator_comments
36             and isinstance(node, parsetree.Text)
37             and not node.content.strip()
38         ):
39             # Ignore whitespace within translator comments
40             continue
41
42         if isinstance(node, parsetree.Comment):
43             value = node.text.strip()
44             if in_translator_comments:
45                 translator_comments.extend(
46                     self._split_comment(node.lineno, value)
47                 )
48                 continue
49             for comment_tag in comment_tags:
50                 if value.startswith(comment_tag):
51                     in_translator_comments = True
52                     translator_comments.extend(
53                         self._split_comment(node.lineno, value)
54                     )
55                 continue
56
57         if isinstance(node, parsetree.DefTag):
58             code = node.function_decl.code
59             child_nodes = node.nodes
60         elif isinstance(node, parsetree.BlockTag):
61             code = node.body_decl.code
62             child_nodes = node.nodes
63         elif isinstance(node, parsetree.CallTag):
64             code = node.code.code
65             child_nodes = node.nodes
66         elif isinstance(node, parsetree.PageTag):
67             code = node.body_decl.code
68         elif isinstance(node, parsetree.CallNamespaceTag):
69             code = node.expression
70             child_nodes = node.nodes
71         elif isinstance(node, parsetree.Controlline):
72             if node.isend:
73                 in_translator_comments = False
74                 continue
75             code = node.text
76         elif isinstance(node, parsetree.Code):
77             in_translator_comments = False
78             code = node.code.code

```

```

79         elif isinstance(node, parsetree.Expression):
80             code = node.code.code
81         else:
82             continue
83
84         # Comments don't apply unless they immediately precede the message
85         if (
86             translator_comments
87             and translator_comments[-1][0] < node.lineno - 1
88         ):
89             translator_comments = []
90
91         translator_strings = [
92             comment[1] for comment in translator_comments
93         ]
94
95         if isinstance(code, str) and self.use_bytes:
96             code = code.encode(input_encoding, "backslashreplace")
97
98         used_translator_comments = False
99         # We add extra newline to work around a pybabel bug
100        # (see python-babel/babel#274, parse_encoding dies if the first
101        # input string of the input is non-ascii)
102        # Also, because we added it, we have to subtract one from
103        # node.lineno
104        if self.use_bytes:
105            code = BytesIO(b"\n" + code)
106        else:
107            code = StringIO("\n" + code)
108
109        for message in self.process_python(
110            code, node.lineno - 1, translator_strings
111        ):
112            yield message
113            used_translator_comments = True
114
115        if used_translator_comments:
116            translator_comments = []
117        in_translator_comments = False
118
119        if child_nodes:
120            yield from self.extract_nodes(child_nodes)
121
122    @staticmethod
123    def _split_comment(lineno, comment):
124        """Return the multiline comment at lineno split into a list of
125        comment line numbers and the accompanying comment line"""
126        return [
127            (lineno + index, line)

```

```
128         for index, line in enumerate(comment.splitlines())
129     ]
```