sqlalchemy / mako Public

Code · Issues 62 · Pull requests 1 · Actions · Projects 1 · Wiki · ...

c2f392e0be

mako / mako / ext / extract.py / Jump to

zzzeek happy new year ... ✓ History

5 contributors

129 lines (112 sloc) | 4.55 KB

```python
# ext/extract.py
# Copyright 2006-2022 the Mako authors and contributors <see AUTHORS file>
#
# This module is part of Mako and is released under
# the MIT License: http://www.opensource.org/licenses/mit-license.php

from io import BytesIO
from io import StringIO
import re

from mako import lexer
from mako import parsetree


class MessageExtractor:
    use_bytes = True

    def process_file(self, fileobj):
        template_node = lexer.Lexer(
            fileobj.read(), input_encoding=self.config["encoding"]
        ).parse()
        yield from self.extract_nodes(template_node.get_children())

    def extract_nodes(self, nodes):
        translator_comments = []
        in_translator_comments = False
        input_encoding = self.config["encoding"] or "ascii"
        comment_tags = list(
            filter(None, re.split(r"\s+", self.config["comment-tags"]))
        )

        for node in nodes:
            child_nodes = None
            if (
                in_translator_comments
                and isinstance(node, parsetree.Text)
                and not node.content.strip()
            ):
                # Ignore whitespace within translator comments
                continue

            if isinstance(node, parsetree.Comment):
                value = node.text.strip()
                if in_translator_comments:
                    translator_comments.extend(
                        self._split_comment(node.lineno, value)
                    )
                    continue
                for comment_tag in comment_tags:
                    if value.startswith(comment_tag):
                        in_translator_comments = True
                        translator_comments.extend(
                            self._split_comment(node.lineno, value)
                        )
                continue

            if isinstance(node, parsetree.DefTag):
                code = node.function_decl.code
                child_nodes = node.nodes
            elif isinstance(node, parsetree.BlockTag):
                code = node.body_decl.code
                child_nodes = node.nodes
            elif isinstance(node, parsetree.CallTag):
                code = node.code.code
                child_nodes = node.nodes
            elif isinstance(node, parsetree.PageTag):
                code = node.body_decl.code
            elif isinstance(node, parsetree.CallNamespaceTag):
                code = node.expression
                child_nodes = node.nodes
            elif isinstance(node, parsetree.ControlLine):
                if node.isend:
                    in_translator_comments = False
                    continue
                code = node.text
            elif isinstance(node, parsetree.Code):
                in_translator_comments = False
                code = node.code.code
            elif isinstance(node, parsetree.Expression):
                code = node.code.code
            else:
                continue

            # Comments don't apply unless they immediately precede the message
            if (
                translator_comments
                and translator_comments[-1][0] < node.lineno - 1
            ):
                translator_comments = []

            translator_strings = [
                comment[1] for comment in translator_comments
            ]

            if isinstance(code, str) and self.use_bytes:
                code = code.encode(input_encoding, "backslashreplace")

            used_translator_comments = False
            # We add extra newline to work around a pybabel bug
            # (see python-babel/babel#274, parse_encoding dies if the first
            # input string of the input is non-ascii)
            # Also, because we added it, we have to subtract one from
            # node.lineno
            if self.use_bytes:
                code = BytesIO(b"\n" + code)
            else:
                code = StringIO("\n" + code)

            for message in self.process_python(
                code, node.lineno - 1, translator_strings
            ):
                yield message
                used_translator_comments = True

            if used_translator_comments:
                translator_comments = []
            in_translator_comments = False

            if child_nodes:
                yield from self.extract_nodes(child_nodes)

    @staticmethod
    def _split_comment(lineno, comment):
        """Return the multiline comment at lineno split into a list of
        comment line numbers and the accompanying comment line"""
        return [
            (lineno + index, line)
            for index, line in enumerate(comment.splitlines())
        ]
```