Package pyx12 :: Module params [hide private]

# Source Code for Module pyx12.params

```
  1  ######################################################################
  2  # Copyright (c) 2001-2007 Kalamazoo Community Mental Health Services,
  3  #   John Holland <jholland@kazoocmh.org> <john@zoner.org>
  4  # All rights reserved.
  5  #
  6  # This software is licensed as described in the file LICENSE.txt, which
  7  # you should have received as part of this distribution.
  8  #
  9  ######################################################################
 10
 11  #    $Id: params.py 1092 2007-05-05 17:43:41Z johnholland $
 12
 13  """
 14  Holds Run-time Parameters
 15
 16  Order of precedence:
 17   1. set(param) - Command line parameters
 18   2. Config files as constructor parameters
 19   3. self.params - Defaults
 20  """
 21  from os.path import dirname, abspath, join, isdir, isfile, expanduser
 22  import sys
 23  import libxml2
 24  import logging
 25
 26  from pyx12.errors import EngineError
 27
 28  NodeType = {'element_start': 1, 'element_end': 15, 'attrib': 2, 'text': 3,
 29      'CData': 4, 'entity_ref': 5, 'entity_decl':6, 'pi': 7, 'comment': 8,
 30      'doc': 9, 'dtd': 10, 'doc_frag': 11, 'notation': 12}
 31
 32  class ParamsBase(object):
 33      """
 34      Base class for parameters
 35      """
 36      def __init__(self):
 37          self.logger = logging.getLogger('pyx12.params')
 38          self.params = {}
 39          #First, try relative path
 40          base_dir = dirname(dirname(abspath(sys.argv[0])))
 41          map_path = join(base_dir, 'map')
 42          #Then look in standard installation location
 43          if not isdir(map_path):
 44              map_path = join(sys.prefix, 'share', 'pyx12', 'map')
 45          self.params['map_path'] = map_path
 46          self.params['pickle_path'] = map_path
 47          self.params['exclude_external_codes'] = None
 48          self.params['ignore_syntax'] = False
 49          self.params['charset'] = 'E'
 50          self.params['ignore_codes'] = False
 51          self.params['ignore_ext_codes'] = False
 52          self.params['skip_html'] = False
 53          self.params['skip_997'] = False
 54          self.params['simple_dtd'] = ''
 55          self.params['idtag_dtd'] = ''
 56          self.params['idtagqual_dtd'] = ''
 57          self.params['xmlout'] = 'simple'
 58          #self.params['xmlout'] = 'idtag'
 59          self.params['xslt_files'] = []
 60
 61      def get(self, option):
 62          """
 63          Get the value of the parameter specified by option
 64          @param option: Option name
 65          @type option: string
 66          """
 67          if option in self.params.keys():
 68              return self.params[option]
 69          else:
 70              return None
 71
 72      def set(self, option, value):
 73          """
 74          Set the value of the parameter specified by option
 75          @param option: Option name
 76          @type option: string
 77          @param value: Parameter value
 78          @type value: string
 79          """
 80          if value == '':
 81              self.params[option] = None
 82          else:
 83              self.params[option] = value
 84
 85      def _read_config_file(self, filename):
 86          """
 87          Read program configuration from an XML file
 88
 89          @param filename: XML file
 90          @type filename: string
 91          @return: None
 92          """
 93          if not isfile(filename):
 94              raise EngineError, 'Read of configuration file "%s" failed' % (filename)
 95          try:
 96              reader = libxml2.newTextReaderFilename(filename)
 97              ret = reader.Read()
 98              while ret == 1:
 99                  tmpNodeType = reader.NodeType()
100                  if tmpNodeType == NodeType['element_start']:
101                      cur_name = reader.Name()
102                      if cur_name == 'param':
103                          option = None
104                          value = None
105                          valtype = None
106                          while reader.MoveToNextAttribute():
107                              if reader.Name() == 'name':
108                                  option = reader.Value()
109                  elif tmpNodeType == NodeType['element_end']:
110                      if option is not None:
111                          self._set_option(option, value, valtype)
112                  elif tmpNodeType == NodeType['text']:
113                      if cur_name == 'value':
114                          value = reader.Value()
115                      elif cur_name == 'type':
116                          valtype = reader.Value()
117                  ret = reader.Read()
118          except:
119              self.logger.error('Read of configuration file "%s" failed' % \
120                  (filename))
121              raise
122
123      def _set_option(self, option, value, valtype):
124          """
125          Set the value of the parameter specified by option
126          @param option: Option name
127          @type option: string
128          @param value: Parameter value
129          @type value: string
130          @param valtype: Parameter type
131          @type valtype: string
132          """
133          if value == '':
134              value = None
135          if valtype == 'boolean':
136              if value in ('False', 'F'):
137                  self.params[option] = False
138              else:
139                  self.params[option] = True
140          else:
141              try:
142                  if self.params[option] != value:
143                      self.params[option] = value
144                      #self.logger.debug('Params: option "%s": "%s"' % \
145                      #    (option, self.params[option]))
146              except:
147                  self.params[option] = value
148                  #self.logger.debug('Params: option "%s": "%s"' % \
149                  #   (option, self.params[option]))
150          #self.logger.debug('Params: option "%s": "%s"' % \
151          #    (option, self.params[option]))
152
153
154  class ParamsUnix(ParamsBase):
155      """
156      Read options from XML configuration files
157      """
158      def __init__(self, config_file=None):
159          ParamsBase.__init__(self)
160          config_files = [join(sys.prefix, 'etc/pyx12.conf.xml'), \
161              expanduser('~/.pyx12.conf.xml')]
162          if config_file:
163              config_files.append(config_file)
164          for filename in config_files:
165              if isfile(filename):
166                  self.logger.debug('Read param file: %s' % (filename))
167                  self._read_config_file(filename)
168
169  class ParamsWindows(ParamsBase):
170      """
171      Read options from XML configuration files
172      """
173      def __init__(self, config_file=None):
174          ParamsBase.__init__(self)
175          config_files = [join(sys.prefix, 'etc/pyx12.conf.xml')]
176          if config_file:
177              config_files.append(config_file)
178          for filename in config_files:
179              if isfile(filename):
180                  self.logger.debug('Read param file: %s' % (filename))
181                  self._read_config_file(filename)
182
183  if sys.platform == 'win32':
184      params = ParamsWindows
185  else:
186      params = ParamsUnix
187
```