Python: работа с XML-файлами и модуль xml.etree.ElementTree

Источник: https://rtfm.co.ua/python-rabota-s-xml-fajlami-i-modul-xml-etree-elementtree/

В стандартной библиотеке **Python** имеется две реализации этого модуля – xml.etree.ElementTree и xml.etree.cElementTree.

xml.etree.ElementTree – реализация **API** для работы с **XML** файлами на чистом **Python**, а xml.etree.cElementTree – то же, но на **C**, и даёт существенный прирост производительности при обработке больших файлов.

Можно импортировать их так:

|  |  |  |
| --- | --- | --- |
| 1 | #!/usr/bin/env python | |
| 2 |  |

|  |  |
| --- | --- |
| 3 | try: |
| 4 | import xml.etree.cElementTree as ET | |

|  |  |
| --- | --- |
| 5 | except ImportError: |
| 6 | import xml.etree.ElementTree as ET | |

|  |  |
| --- | --- |
| 7 |  |
| 8 | print ET | |

Результат:

|  |  |
| --- | --- |
| 1 | $ ./xml\_par.py |
| 2 | <module 'xml.etree.cElementTree' from '/usr/lib64/python2.6/xml/etree/cElementTree.pyc'> | |

В **Python** версии 3.3 и выше необходимости в такой try/except нет, т.к. интерпретатор самостоятельно будет выполнять поиск cElementTree при импорте ElementTree (да и в **Python** 2.6 и 2.7 cElementTree импортируется без проблем).

Contents

* [Парсинг XML-файла](https://rtfm.co.ua/python-rabota-s-xml-fajlami-i-modul-xml-etree-elementtree/#_XML)
* [Поиск элементов в файле](https://rtfm.co.ua/python-rabota-s-xml-fajlami-i-modul-xml-etree-elementtree/#i)
* [Добавление записей в файл](https://rtfm.co.ua/python-rabota-s-xml-fajlami-i-modul-xml-etree-elementtree/#i-2)
* [Similar posts](https://rtfm.co.ua/python-rabota-s-xml-fajlami-i-modul-xml-etree-elementtree/#Similar_posts)

**Парсинг XML-файла**

Для примера возьмём простой **XML**-файл с таким содержимым:

|  |  |
| --- | --- |
| 01 | <TreeRoot> |
| 02 | <Element1 value1="Value1"> | |

|  |  |
| --- | --- |
| 03 | <SubElement1>SubElement1</SubElement1> |
| 04 | <SubElement2>SubElement2</SubElement2> |

|  |  |  |
| --- | --- | --- |
| 05 | </Element1> | |
| 06 |  |

|  |  |
| --- | --- |
| 07 | <Element2 value2="Value2"> |
| 08 | <SubElement3>SubElement3</SubElement3> | |

|  |  |  |
| --- | --- | --- |
| 09 | <SubElement4>SubElement4</SubElement4> | |
| 10 | </Element2> |

|  |  |
| --- | --- |
| 11 | </TreeRoot> |

Изменим скрипт:

|  |  |  |
| --- | --- | --- |
| 01 | import os | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | import xml.etree.cElementTree as ET | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | XML\_FILE = os.path.join(os.environ['HOME'], 'xmlfile.xml') | |
| 06 |  |

|  |  |
| --- | --- |
| 07 | try: |
| 08 | tree = ET.ElementTree(file=XML\_FILE) | |

|  |  |
| --- | --- |
| 09 | print help(tree) |
| 10 | except IOError as e: |

|  |  |
| --- | --- |
| 11 | print 'nERROR - cant find file: %sn' % e |

В результате – мы должны получить список доступных методов для объекта tree:

|  |  |  |
| --- | --- | --- |
| 01 | Help on instance of ElementTree in module \_\_builtin\_\_: | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | class ElementTree(xml.etree.ElementTree.ElementTree) | |
| 04 | |  Methods defined here: |

|  |  |
| --- | --- |
| 05 | | |
| 06 | |  parse(self, source, parser=None) | |

|  |  |
| --- | --- |
| 07 | | |
| 08 | |  ---------------------------------------------------------------------- | |

|  |  |  |
| --- | --- | --- |
| 09 | |  Methods inherited from xml.etree.ElementTree.ElementTree: | |
| 10 | | |

|  |  |  |
| --- | --- | --- |
| 11 | |  \_\_init\_\_(self, element=None, file=None) | |
| 12 | | |

|  |  |  |
| --- | --- | --- |
| 13 | |  find(self, path) | |
| 14 | | |

|  |  |  |
| --- | --- | --- |
| 15 | |  findall(self, path) | |
| 16 | | |

|  |  |  |
| --- | --- | --- |
| 17 | |  findtext(self, path, default=None) | |
| 18 | | |

|  |  |  |
| --- | --- | --- |
| 19 | |  getiterator(self, tag=None) | |
| 20 | | |

|  |  |  |
| --- | --- | --- |
| 21 | |  getroot(self) | |
| 22 | | |

|  |  |
| --- | --- |
| 23 | |  write(self, file, encoding='us-ascii') |

В случае ошибки синтаксиса **XML** – будет вызвана ошибка с указанием точного места:

|  |  |
| --- | --- |
| 1 | cElementTree.ParseError: mismatched tag: line 11, column 2 |

Тогда файл стоит проверить в **XML**-валидаторе, например – [тут>>>](https://rtfm.co.ua/goto/http:/www.xmlvalidation.com/index.php).

Для получения корневого элемента – используется метод getroot():

|  |  |
| --- | --- |
| 1 | ... |
| 2 | try: | |

|  |  |  |
| --- | --- | --- |
| 3 | tree = ET.ElementTree(file=XML\_FILE) | |
| 4 | print tree.getroot() |

|  |  |  |  |
| --- | --- | --- | --- |
| 5 | print type(tree.getroot()) | | |
| 6 | ... | |
| 1 | $ ./xml\_par.py |

|  |  |  |
| --- | --- | --- |
| 2 | <Element 'TreeRoot' at 0x6ffffd96120> | |
| 3 | <type 'Element'> |

Каждый элемент содержит несколько параметров:

* tag – строка, отображающая тип данных, которые представляет элемент;
* attrib – атрибуты элемента, которые сохраняются в словарь **Python**;
* text – текстовое значение элемента;
* дочерние элементы.

Например – получить тег корневого элемента можно так:

|  |  |
| --- | --- |
| 1 | ... |
| 2 | try: | |

|  |  |  |
| --- | --- | --- |
| 3 | tree = ET.ElementTree(file=XML\_FILE) | |
| 4 | root = tree.getroot() |

|  |  |
| --- | --- |
| 5 |  |
| 6 | print root.tag | |

|  |  |
| --- | --- |
| 7 | ... |
| 1 | $ ./xml\_par.py | |

|  |  |
| --- | --- |
| 2 | TreeRoot |

Что бы получить список прямых потомков корневого элемента – можно просто вызвать цикл:

|  |  |
| --- | --- |
| 1 | ... |
| 2 | try: | |

|  |  |  |
| --- | --- | --- |
| 3 | tree = ET.ElementTree(file=XML\_FILE) | |
| 4 | root = tree.getroot() |

|  |  |
| --- | --- |
| 5 |  |
| 6 | for child\_of\_root in root: | |

|  |  |  |
| --- | --- | --- |
| 7 | print child\_of\_root.tag, child\_of\_root.attrib | |
| 8 | ... |

Результат:

|  |  |  |
| --- | --- | --- |
| 1 | $ ./xml\_par.py | |
| 2 | TreeRoot |

|  |  |
| --- | --- |
| 3 | Element1 {'value1': 'Value1'} |
| 4 | Element2 {'value2': 'Value2'} |

Можно так же вывести только ключи, или *ключи:значения*:

|  |  |
| --- | --- |
| 1 | ... |
| 2 | for child\_of\_root in root: | |

|  |  |  |
| --- | --- | --- |
| 3 | print child\_of\_root.tag, child\_of\_root.keys(), child\_of\_root.items() | |
| 4 | ... |

Результат:

|  |  |
| --- | --- |
| 1 | $ ./xml\_par.py |
| 2 | Element1 ['value1'] [('value1', 'Value1')] | |

|  |  |
| --- | --- |
| 3 | Element2 ['value2'] [('value2', 'Value2')] |

**Поиск элементов в файле**

Что бы перебрать все элементы в файле – можно воспользоваться методом iter():

|  |  |
| --- | --- |
| 1 | ... |
| 2 | try: | |

|  |  |  |
| --- | --- | --- |
| 3 | tree = ET.ElementTree(file=XML\_FILE) | |
| 4 | root = tree.getroot() |

|  |  |
| --- | --- |
| 5 |  |
| 6 | for child\_of\_root in root.iter(): | |

|  |  |  |
| --- | --- | --- |
| 7 | print 'Tag: %snKeys: %snItems: %snText: %sn' % (child\_of\_root.tag, child\_of\_root.keys(), child\_of\_root.items(), child\_of\_root.text) | |
| 8 | ... |

Результат:

|  |  |  |
| --- | --- | --- |
| 01 | $ ./xml\_par.py | |
| 02 | Tag: TreeRoot |

|  |  |
| --- | --- |
| 03 | Keys: [] |
| 04 | Items: [] | |

|  |  |  |
| --- | --- | --- |
| 05 | Text: | |
| 06 |  |

|  |  |
| --- | --- |
| 07 | Tag: Element1 |
| 08 | Keys: ['value1'] | |

|  |  |  |
| --- | --- | --- |
| 09 | Items: [('value1', 'Value1')] | |
| 10 | Text: |

|  |  |
| --- | --- |
| 11 |  |
| 12 | Tag: SubElement1 | |

|  |  |
| --- | --- |
| 13 | Keys: [] |
| 14 | Items: [] | |

|  |  |  |
| --- | --- | --- |
| 15 | Text: SubElement1 | |
| 16 |  |

|  |  |  |
| --- | --- | --- |
| 17 | Tag: SubElement2 | |
| 18 | Keys: [] |

|  |  |
| --- | --- |
| 19 | Items: [] |
| 20 | Text: SubElement2 | |

|  |  |
| --- | --- |
| 21 |  |
| 22 | Tag: Element2 | |

|  |  |
| --- | --- |
| 23 | Keys: ['value2'] |
| 24 | Items: [('value2', 'Value2')] | |

|  |  |  |
| --- | --- | --- |
| 25 | Text: | |
| 26 |  |

|  |  |  |
| --- | --- | --- |
| 27 | Tag: SubElement3 | |
| 28 | Keys: [] |

|  |  |
| --- | --- |
| 29 | Items: [] |
| 30 | Text: SubElement3 | |

|  |  |
| --- | --- |
| 31 |  |
| 32 | Tag: SubElement4 | |

|  |  |
| --- | --- |
| 33 | Keys: [] |
| 34 | Items: [] | |

|  |  |
| --- | --- |
| 35 | Text: SubElement4 |

А что бы найти только один элемент – его тег можно передать аргументом этому методу:

|  |  |
| --- | --- |
| 1 | ... |
| 2 | for child\_of\_root in root.iter('SubElement1'): | |

|  |  |  |
| --- | --- | --- |
| 3 | print 'Tag: %snKeys: %snItems: %snText: %sn' % (child\_of\_root.tag, child\_of\_root.keys(), child\_of\_root.items(), child\_of\_root.text) | |
| 4 | ... |

Результат:

|  |  |
| --- | --- |
| 1 | $ ./xml\_par.py |
| 2 | Tag: SubElement1 | |

|  |  |
| --- | --- |
| 3 | Keys: [] |
| 4 | Items: [] | |

|  |  |
| --- | --- |
| 5 | Text: SubElement1 |

Можно выполнить поиск с помощью [**XPath**](https://rtfm.co.ua/goto/http:/effbot.org/zone/element-xpath.htm).

Например, что бы отобразить корневой элемент:

|  |  |
| --- | --- |
| 1 | ... |
| 2 | try: | |

|  |  |  |
| --- | --- | --- |
| 3 | tree = ET.ElementTree(file=XML\_FILE) | |
| 4 | root = tree.getroot() |

|  |  |
| --- | --- |
| 5 |  |
| 6 | for item in root.iterfind('.'): | |

|  |  |  |
| --- | --- | --- |
| 7 | print 'Find: %sn' % item.tag | |
| 8 | ... |

Результат:

|  |  |
| --- | --- |
| 1 | $ ./xml\_par.py |
| 2 | Find: TreeRoot |

С помощью // можно найти все вложенные элементы:

|  |  |
| --- | --- |
| 1 | ... |
| 2 | for item in root.iterfind('.//'): | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | print 'Find: %sn' % item.tag | | | |
| 4 | ... | | |
| 01 | | $ ./xml\_par.py |

|  |  |  |
| --- | --- | --- |
| 02 | Find: Element1 | |
| 03 |  |

|  |  |  |
| --- | --- | --- |
| 04 | Find: SubElement1 | |
| 05 |  |

|  |  |  |
| --- | --- | --- |
| 06 | Find: SubElement2 | |
| 07 |  |

|  |  |  |
| --- | --- | --- |
| 08 | Find: Element2 | |
| 09 |  |

|  |  |  |
| --- | --- | --- |
| 10 | Find: SubElement3 | |
| 11 |  |

|  |  |
| --- | --- |
| 12 | Find: SubElement4 |

Или вложенные элементы вложенного элемента:

|  |  |
| --- | --- |
| 1 | ... |
| 2 | for item in root.iterfind('./Element1//'): | |

|  |  |  |
| --- | --- | --- |
| 3 | print 'Find: %sn' % item.tag | |
| 4 | ... |

Результат:

|  |  |
| --- | --- |
| 1 | $ ./xml\_par.py |
| 2 | Find: SubElement1 | |

|  |  |
| --- | --- |
| 3 |  |
| 4 | Find: SubElement2 | |

**Добавление записей в файл**

Для начала – добавим функцию [prettify](https://rtfm.co.ua/goto/http:/pymotw.com/2/xml/etree/ElementTree/create.html):

|  |  |  |
| --- | --- | --- |
| 1 | from xml.dom import minidom | |
| 2 |  |

|  |  |
| --- | --- |
| 3 | def prettify(elem): |
| 4 | """Return a pretty-printed XML string for the Element. | |

|  |  |
| --- | --- |
| 5 | """ |
| 6 | rough\_string = ET.tostring(elem, 'utf-8') | |

|  |  |  |
| --- | --- | --- |
| 7 | reparsed = minidom.parseString(rough\_string) | |
| 8 | return reparsed.toprettyxml(indent='t') |

Теперь – отредактируем код:

|  |  |  |
| --- | --- | --- |
| 01 | #!/usr/bin/env python | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | import os | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | import xml.etree.cElementTree as ET | |
| 06 | from xml.dom import minidom |

|  |  |
| --- | --- |
| 07 |  |
| 08 | def prettify(elem): | |

|  |  |  |
| --- | --- | --- |
| 09 | """Return a pretty-printed XML string for the Element. | |
| 10 | """ |

|  |  |
| --- | --- |
| 11 | rough\_string = ET.tostring(elem, 'utf-8') |
| 12 | reparsed = minidom.parseString(rough\_string) | |

|  |  |  |
| --- | --- | --- |
| 13 | return reparsed.toprettyxml(indent='t') | |
| 14 |  |

|  |  |  |
| --- | --- | --- |
| 15 | XML\_FILE = os.path.join(os.environ['HOME'], 'xmlfile.xml') | |
| 16 |  |

|  |  |
| --- | --- |
| 17 | try: |
| 18 | tree = ET.parse(XML\_FILE) | |

|  |  |  |
| --- | --- | --- |
| 19 | root = tree.getroot() | |
| 20 |  |

|  |  |
| --- | --- |
| 21 | new\_element = ET.Element('NewElement') |
| 22 | new\_subelement = ET.SubElement(new\_element, 'NewSubelement') | |

|  |  |  |
| --- | --- | --- |
| 23 | new\_subelement.text = 'NewSubelement' | |
| 24 | root.append(new\_element) |

|  |  |
| --- | --- |
| 25 |  |
| 26 | print prettify(root) | |

|  |  |
| --- | --- |
| 27 |  |
| 28 | #  пока оставим так | |

|  |  |  |
| --- | --- | --- |
| 29 | #    tree.write(XML\_FILE) | |
| 30 |  |

|  |  |
| --- | --- |
| 31 | except IOError as e: |
| 32 | print 'nERROR - cant find file: %sn' % e | |

* с помощью ET.Element мы создаём новый элемент с именем NewElement;
* затем – с помощью SubElement – мы добавляем новый вложенный элемент с имменем NewSubelement внутрь элемента в объекте new\_element;
* далее – мы определяем параметр text объекта new\_subelement, и задаём занчение 'NewSubelement';
* после этого – мы добавляем новый элемент new\_element со всем содержимым к корню нашего файла;
* последним – вызываем print() и функцию prettify().

Результат:

|  |  |
| --- | --- |
| 01 | $ ./xml\_par.py |
| 02 | <?xml version="1.0" ?> | |

|  |  |
| --- | --- |
| 03 | <TreeRoot> |
| 04 | <Element1 value1="Value1"> | |

|  |  |
| --- | --- |
| 05 |  |
| 06 | <SubElement1>SubElement1</SubElement1> | |

|  |  |
| --- | --- |
| 07 |  |
| 08 | <SubElement2>SubElement2</SubElement2> | |

|  |  |
| --- | --- |
| 09 |  |
| 10 | </Element1> | |

|  |  |
| --- | --- |
| 11 |  |
| 12 | <Element2 value2="Value2"> | |

|  |  |
| --- | --- |
| 13 |  |
| 14 | <SubElement3>SubElement3</SubElement3> | |

|  |  |
| --- | --- |
| 15 |  |
| 16 | <SubElement4>SubElement4</SubElement4> | |

|  |  |
| --- | --- |
| 17 |  |
| 18 | </Element2> | |

|  |  |
| --- | --- |
| 19 | <NewElement> |
| 20 | <NewSubelement>NewSubelement</NewSubelement> | |

|  |  |  |
| --- | --- | --- |
| 21 | </NewElement> | |
| 22 | </TreeRoot> |

Для того, что бы в таком виде записать в файл – можно использовать функцию [indent()](https://rtfm.co.ua/goto/http:/effbot.org/zone/element-lib.htm#prettyprint):

|  |  |  |
| --- | --- | --- |
| 01 | def indent(elem, level=0): | |
| 02 | i = "n" + level\*"  " |

|  |  |
| --- | --- |
| 03 | if len(elem): |
| 04 | if not elem.text or not elem.text.strip(): | |

|  |  |
| --- | --- |
| 05 | elem.text = i + "  " |
| 06 | if not elem.tail or not elem.tail.strip(): | |

|  |  |  |
| --- | --- | --- |
| 07 | elem.tail = i | |
| 08 | for elem in elem: |

|  |  |
| --- | --- |
| 09 | indent(elem, level+1) |
| 10 | if not elem.tail or not elem.tail.strip(): | |

|  |  |  |
| --- | --- | --- |
| 11 | elem.tail = i | |
| 12 | else: |

|  |  |  |
| --- | --- | --- |
| 13 | if level and (not elem.tail or not elem.tail.strip()): | |
| 14 | elem.tail = i |

И вызвать её, передав корневой элемент, перед запись в файл:

|  |  |
| --- | --- |
| 1 | ... |
| 2 | indent(root) | |

|  |  |
| --- | --- |
| 3 |  |
| 4 | tree.write(XML\_FILE) | |

|  |  |
| --- | --- |
| 5 | ... |

Результат:

|  |  |  |
| --- | --- | --- |
| 01 | $ cat xmlfile.xml | |
| 02 | <TreeRoot> |

|  |  |
| --- | --- |
| 03 | <Element1 value1="Value1"> |
| 04 | <SubElement1>SubElement1</SubElement1> | |

|  |  |  |
| --- | --- | --- |
| 05 | <SubElement2>SubElement2</SubElement2> | |
| 06 | </Element1> |

|  |  |
| --- | --- |
| 07 | <Element2 value2="Value2"> |
| 08 | <SubElement3>SubElement3</SubElement3> | |

|  |  |  |
| --- | --- | --- |
| 09 | <SubElement4>SubElement4</SubElement4> | |
| 10 | </Element2> |

|  |  |
| --- | --- |
| 11 | <NewElement> |
| 12 | <NewSubelement>NewSubelement</NewSubelement> | |

|  |  |  |
| --- | --- | --- |
| 13 | </NewElement> | |
| 14 | </TreeRoot> |

Файл с функциями можно скачать [тут>>>](https://rtfm.co.ua/goto/https:/github.com/edliaw/fungidb_tools/blob/master/fungidb_tools/elementlib/ElementLib.py).

Что бы добавить новый элемент внутрь уже имеющегося – можно указать его индекс в корне, например:

|  |  |
| --- | --- |
| 1 | ... |
| 2 | new\_element = ET.Element('NewElementInElement2') | |

|  |  |  |
| --- | --- | --- |
| 3 | new\_subelement = ET.SubElement(new\_element, 'NewSubelementInElement2') | |
| 4 | new\_subelement.text = 'NewSubelementInElement2' |

|  |  |  |
| --- | --- | --- |
| 5 | root[1].append(new\_element) | |
| 6 | ... |

Результат:

|  |  |
| --- | --- |
| 01 | $ ./xml\_par.py |
| 02 | <?xml version="1.0" ?> | |

|  |  |  |
| --- | --- | --- |
| 03 | <TreeRoot> | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | <Element1 value1="Value1"> | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | <SubElement1>SubElement1</SubElement1> | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | <SubElement2>SubElement2</SubElement2> | |
| 10 |  |

|  |  |  |
| --- | --- | --- |
| 11 | </Element1> | |
| 12 |  |

|  |  |  |
| --- | --- | --- |
| 13 | <Element2 value2="Value2"> | |
| 14 |  |

|  |  |  |
| --- | --- | --- |
| 15 | <SubElement3>SubElement3</SubElement3> | |
| 16 |  |

|  |  |  |
| --- | --- | --- |
| 17 | <SubElement4>SubElement4</SubElement4> | |
| 18 |  |

|  |  |  |
| --- | --- | --- |
| 19 | | <NewElementInElement2> |
| 20 | <NewSubelementInElement2>NewSubelementInElement2</NewSubelementInElement2> | | |

|  |  |  |
| --- | --- | --- |
| 21 | </NewElementInElement2> | |
| 22 | </Element2> |

|  |  |
| --- | --- |
| 23 |  |
| 24 | <NewElement> | |

|  |  |
| --- | --- |
| 25 |  |
| 26 | <NewSubelement>NewSubelement</NewSubelement> | |

|  |  |
| --- | --- |
| 27 |  |
| 28 | </NewElement> | |

|  |  |
| --- | --- |
| 29 |  |
| 30 | </TreeRoot> | |

Ссылки по теме

[http://effbot.org](https://rtfm.co.ua/goto/http:/effbot.org/zone/element.htm)

[https://docs.python.org](https://rtfm.co.ua/goto/https:/docs.python.org/2/library/xml.etree.elementtree.html)

[http://effbot.org](https://rtfm.co.ua/goto/http:/effbot.org/zone/element-lib.htm)

[http://eli.thegreenplace.net](https://rtfm.co.ua/goto/http:/eli.thegreenplace.net/2012/03/15/processing-xml-in-python-with-elementtree)

[http://pymotw.com](https://rtfm.co.ua/goto/http:/pymotw.com/2/xml/etree/ElementTree/create.html)