**Novixys Software Dev Blog**

# Parsing XML in Python

**Contents** [hide]



## 1. Introduction

XML can be parsed in python using the *xml.etree.ElementTree* library. This article shows you how to parse and extract elements, attributes and text from XML using this library.

While this library is easy to use, it loads the whole XML document into memory and hence may not be suitable for processing large XML files or

where run-time memory is an issue.

## 2. Sample XML

Here is a short snippet of the sample XML we will be working with.

```
<?xml version="1.0"?>
<catalog>
  <book id="bk101">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
    <price>44.95</price>
    <publish_date>2000-10-01</publish_date>
    <description>An in-depth look at creating applications
    with XML.</description>
  </book>
  <book id="bk102">
    <author>Ralls, Kim</author>
    <title>Midnight Rain</title>
...
```

## 3. Getting Started

### 3.1 Parsing an XML File

It is very simple to parse an XML using *ElementTree*. The following returns an *ElementTree* object which contains all the XML artifacts.

```
import xml.etree.ElementTree as ET

tree = ET.parse('catalog.xml')
```

Once you have the *ElementTree* object, you can get the root *Element* object

using the *tree.getroot()* method:

```
root = tree.getroot()
```

### 3.2 Parsing an XML String

We use the *ElementTree.fromstring()* method to parse an XML string. The method returns root Element directly: a subtle difference compared with the *ElementTree.parse()* method which returns an ElementTree object.

```
print ET.fromstring('<a><b>1</b>2</a>')
# prints "<Element 'a' at ...>"
```

## 4. Working with Elements

From the *Element* object, you can get the XML tag name using the *tag* property.

```
print root.tag    # outputs "catalog"
```

Get a list of child *Element*s from any *Element* object using *element.findall('*')*. You can find specific children by name by passing the name of the tag e.g. *element.findall('book')*.

For example, the following code recursively processes all the elements in the XML and prints the name of the tag.

```
def show(elem):
    print elem.tag
```

```
    for child in elem.findall('*'):
        show(child)

show(root)
```

The above code can be modified to show nicely indented output of the tag names:

```
def show(elem, indent = 0);
    print ' ' * indent + elem.tag
    for child in elem.findall('*'):
         show(elem, indent + 1)

show(root)
```

To find a single element by name, use *elem.find(tagName)*:

```
print root.find('book').tag    # prints "book"
```

## 5. Working with Attributes

XML attributes can be extracted from an Element object using the *element.items()* method which returns a sequence of name, value pairs. (The name-value pairs are returned in random order, not in the order they appear in the XML.)

```
for attrName, attrValue in elem.items():
    print attrName + '=' + attrValue
```

To retrieve a single attribute value by name, use the *elem.get(attrName)* method:

```
print root.find('book').get('id')    # prints "bk101"
```

Get a list of all attribute names defined on the element using *elem.keys()*. Returns an empty list if no attributes are defined.

```
print root.find('book').keys()
# prints ['id']
```

To get all the attributes as a python dictionary, use the *elem.attrib* property:

```
print root.find('book').attrib
# prints {'id': 'bk101'}
```

## 6. Retrieving Element Text Content

You can retrieve an element's text content using the *elem.text* property as follows:

```
print ET.fromstring('<a>Hello<b>1</b>2</a>').text
# prints "Hello"

print ET.fromstring('<a>Hello<b>1</b>2</a>').find('b').text
#prints "1"
```

Text appearing after the element's end tag is retrieved using *elem.tail* property:

```
print ET.fromstring('<a>Hello<b>1</b>2</a>').find('b').tail
# prints "2"
```

## 7. Using Path to Extract Content

Some of the *Element* object methods support extracting content by using a syntax similar to XPath:

Retrieve a descendant element:

```
print root.find('book/author').text
# prints "Gambardella, Matthew"
```

To obtain the text of the first matching element, use the *elem.findtext()* method as follows:

```
print root.findtext('book/author')
#prints "Gambardella, Matthew"
```

Retrieve and process a list of matching elements using *elem.findall()*:

```
for e in root.findall('book/author'):
    print e.text

# prints the following
Gambardella, Matthew
Ralls, Kim
Corets, Eva
Corets, Eva
Corets, Eva
Randall, Cynthia
Thurman, Paula
Knorr, Stefan
Kress, Peter
O'Brien, Tim
O'Brien, Tim
Galos, Mike
```

Find a specific element by position. (Position indexes start with 1).

```
print root.find('book[2]/author').text
# prints "Ralls, Kim"
```

Here is an example to find and concatenate all text content using a reduce operation:

```
reduce(lambda x, y: x + '|' + y.text, root.findall("book/author"), '')

# prints "|Gambardella, Matthew|Ralls, Kim|Corets, Eva|Corets, Eva|Corets,
Eva|Randall, Cynthia|Thurman, Paula|Knorr, Stefan|Kress, Peter|O'Brien,
Tim|O'Brien, Tim|Galos, Mike"
```

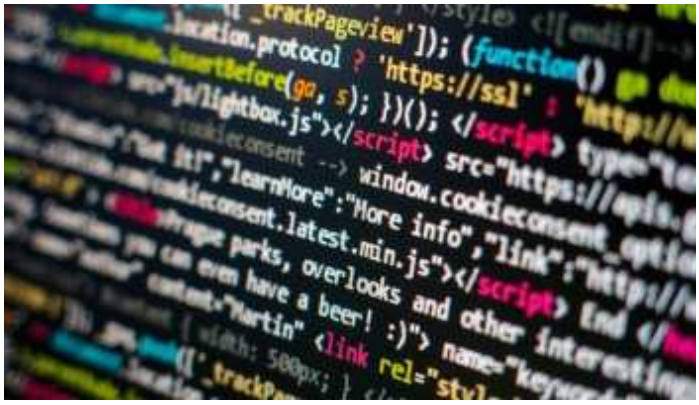
## Conclusion

This article demonstrated some aspects of parsing XML with python. We showed how to parse an XML file or an XML string and extract elements, attributes and text content.

## See Also

- xml.etree.ElementTree
- Element

See Also



### Load XML into Mysql Using Java

### How to Parse XML in Java using a DOM Parser

1. Introduction Let us learn how parse XML in Java using a DOM parser. DOM stands for Document Object Model. Parsing an XML file using a DOM Parser refers to obtaining a DOM Object from the XML data. The DOM object can then be queries for various XML artifacts like…

---

### How to Modify XML File in Java

1. Introduction Let us learn how to modify an XML file to remove unwanted information. One method to remove XML nodes is to use the XML DOM Api to search the XML structure and remove unwanted nodes. While this sounds easy, using the DOM Api is quite hard especially for anything more…

---

Jay Sridhar / December 31, 2016 / Uncategorized / python, xml