# xmltool Documentation

***Release 0.4***

**Aurélien Matouillot**

April 20, 2016

Contents


**1 Getting started** **3**
- 1.1 Creating a XML file . . . 3
- 1.2 Loading a XML file . . . 3
- 1.3 Updating a XML file . . . 4
- 1.4 Accessing . . . 4
- 1.5 Traversing . . . 5

**2 Tutorial** **7**
- 2.1 Insert element in list . . . 8
- 2.2 Delete element . . . 8
- 2.3 Choice element . . . 9
- 2.4 List choice element . . . 9

**3 Migration** **11**
- 3.1 Updating a XML file after a dtd change . . . 11

**4 Form** **13**

**5 Changelog** **15**

xmltool is a python package to manipulate XML files. It's very useful to update some XML files with the python syntax without using the DOM. You can find the source on github.

CHAPTER 1

# Getting started

For the following examples will use this dtd content:

```
<!ELEMENT movies (movie*)>
<!ELEMENT movie (title, date?, realisator, characters, (good-comment|bad-comment)*, (bad|good|awesome
<!ATTLIST movie idmovie ID #IMPLIED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT realisator (#PCDATA)>
<!ELEMENT characters (character+)>
<!ELEMENT character (#PCDATA)>
<!ATTLIST character idcharacter ID #IMPLIED>
<!ELEMENT good-comment (#PCDATA)>
<!ELEMENT bad-comment (#PCDATA)>
<!ELEMENT bad (#PCDATA)>
<!ELEMENT good (#PCDATA)>
<!ELEMENT awesome (#PCDATA)>
```

## 1.1 Creating a XML file

To create a XML we just need to call the method `create` like in the following example:

```
>>> import xmltool
>>> dtd_url = 'examples/movies.dtd'
>>> movies = xmltool.create('movies', dtd_url=dtd_url)
>>> print movies
<movies/>

>>> movies.write('examples/movies.xml')
```

You can see the content of the generated file and note that the required tags have been added automatically to make a valid XML file.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE movies SYSTEM "examples/movies.dtd">
<movies/>
```

## 1.2 Loading a XML file

For this example we load the file previously created:

```
>>> import xmltool
>>> filename = 'examples/movies.xml'
>>> movies = xmltool.load(filename)
>>> print movies
<movies/>
```

## 1.3 Updating a XML file

Now we will see that updating an XML is very easy:

```
>>> import xmltool
>>> filename = 'examples/movies.xml'
>>> movies = xmltool.load(filename)
>>> print movies
<movies/>

>>> movie = movies.add('movie')
>>> title = movie.add('title', 'Full Metal Jacket')
>>> print title
<title>Full Metal Jacket</title>

>>> print movies
<movies>
  <movie>
    <title>Full Metal Jacket</title>
    <realisator></realisator>
    <characters>
      <character></character>
    </characters>
  </movie>
</movies>

>>> movies.write()
```

## 1.4 Accessing

To access a property of the XML object you have to use the list and dictionnary syntax.

```
>>> import xmltool
>>> filename = 'examples/movies.xml'
>>> movies = xmltool.load(filename)
>>> # As you can see in the dtd, movies has only one child movie
>>> # which is a repeated element.
>>> # Here is the syntax to get the first movie
>>> print movies['movie'][0]
<movie>
  <title>Full Metal Jacket</title>
  <realisator></realisator>
  <characters>
    <character></character>
  </characters>
</movie>

>>> # You have the text property to access to the value of a tag
```

```
>>> print movies['movie'][0]['title'].text
Full Metal Jacket
```

To check if a XML property exists you can use if ... in ... or .get():

```
>>> import xmltool
>>> filename = 'examples/movies.xml'
>>> movies = xmltool.load(filename)
>>> movie1 = movies['movie'][0]
>>> print 'date' in movie1
False
>>> print movie1.get('date')
None
```

There is also a method to get or add element

```
>>> import xmltool
>>> filename = 'examples/movies.xml'
>>> movies = xmltool.load(filename)
>>> movie1 = movies['movie'][0]
>>> # Long version
>>> if 'date' not in movie1:
...     date = movie1.add('date')
>>> # Short version
>>> date = movie1.get_or_add('date')
```

We can also access to the attributes

```
>>> filename = 'examples/movies.xml'
>>> movies = xmltool.load(filename)
>>> movie1 = movies['movie'][0]
>>> movie1.add_attribute('idmovie', 'myid')
>>> print movie1
<movie idmovie="myid">
  <title>Full Metal Jacket</title>
  <realisator></realisator>
  <characters>
    <character></character>
  </characters>
</movie>

>>> print movie1.attributes['idmovie']
myid
```

## 1.5 Traversing

To find all elements from a tagname use `findall`

```
>>> import xmltool
>>> filename = 'examples/movies.xml'
>>> movies = xmltool.load(filename)
>>> # Add 2 new movies to improve this example
>>> for i in range(2):
...     movie = movies.add('movie')
...     title = movie.add('title', 'Title %i' % i)
>>> titles = movies.findall('title')
>>> titles_str = [t.text for t in titles]
```

```
>>> print titles_str
['Full Metal Jacket', 'Title 0', 'Title 1']
```

You can also go through all the elements by using `walk`

```
>>> import xmltool
>>> filename = 'examples/movies.xml'
>>> movies = xmltool.load(filename)
>>>
>>> tagnames = [elt.tagname for elt in movies.walk()]
>>> print tagnames
['movie', 'title', 'realisator', 'characters', 'character']
```

CHAPTER 2

# Tutorial

We will create a new XML file from scratch with 2 movies.

```
>>> import xmltool
>>> dtd_url = 'examples/movies.dtd'
>>> movies = xmltool.create('movies', dtd_url=dtd_url)
>>> movie_fmj = movies.add('movie')
>>> t = movie_fmj.add('title', 'Full Metal Jacket')
>>> r = movie_fmj.add('realisator', 'Stanley Kubrick')
>>> characters = movie_fmj.add('characters')
>>> c1 = characters.add('character', 'Matthew Modine')
>>> c2 = characters.add('character', 'Vincent D\'Onofrio')
>>> print movies
<movies>
  <movie>
    <title>Full Metal Jacket</title>
    <realisator>Stanley Kubrick</realisator>
    <characters>
      <character>Matthew Modine</character>
      <character>Vincent D'Onofrio</character>
    </characters>
  </movie>
</movies>

>>> movie_tit = movies.add('movie')
>>> t = movie_tit.add('title', 'Titanic')
>>> r = movie_tit.add('realisator', 'Cameron James')
>>> characters = movie_tit.add('characters')
>>> c1 = characters.add('character', 'Leonardo DiCaprio')
>>> c2 = characters.add('character', 'Kate Winslet')
>>> print movies
<movies>
  <movie>
    <title>Full Metal Jacket</title>
    <realisator>Stanley Kubrick</realisator>
    <characters>
      <character>Matthew Modine</character>
      <character>Vincent D'Onofrio</character>
    </characters>
  </movie>
  <movie>
    <title>Titanic</title>
    <realisator>Cameron James</realisator>
    <characters>
```

```
      <character>Leonardo DiCaprio</character>
      <character>Kate Winslet</character>
    </characters>
  </movie>
</movies>

>>> movies.write('examples/tutorial1.xml')
```

## 2.1 Insert element in list

We will add a new movie after "Full Metal Jacket" so we have to pass `index=1`. If you want to add the movie at the last place don't pass index.

```
>>> import xmltool
>>> movies = xmltool.load('examples/tutorial1.xml')
>>> movie = movies.add('movie', index=1)
>>> print movies
<movies>
  <movie>
    <title>Full Metal Jacket</title>
    <realisator>Stanley Kubrick</realisator>
    <characters>
      <character>Matthew Modine</character>
      <character>Vincent D'Onofrio</character>
    </characters>
  </movie>
  <movie>
    <title></title>
    <realisator></realisator>
    <characters>
      <character></character>
    </characters>
  </movie>
  <movie>
    <title>Titanic</title>
    <realisator>Cameron James</realisator>
    <characters>
      <character>Leonardo DiCaprio</character>
      <character>Kate Winslet</character>
    </characters>
  </movie>
</movies>
```

## 2.2 Delete element

We will delete the movie "Full Metal Jacket" which is the first of the list.

```
>>> import xmltool
>>> movies = xmltool.load('examples/tutorial1.xml')
>>> movies['movie'][0].delete()
>>> print movies
<movies>
  <movie>
    <title>Titanic</title>
```

```
    <realisator>Cameron James</realisator>
    <characters>
      <character>Leonardo DiCaprio</character>
      <character>Kate Winslet</character>
    </characters>
  </movie>
</movies>

>>> movies.write()
```

## 2.3 Choice element

Add and delete choice element.

```
>>> import xmltool
>>> movies = xmltool.load('examples/tutorial1.xml')
>>> movie = movies['movie'][0]
>>> c1 = movie.add('good', 'Good movie')
>>> print movies
<movies>
  <movie>
    <title>Titanic</title>
    <realisator>Cameron James</realisator>
    <characters>
      <character>Leonardo DiCaprio</character>
      <character>Kate Winslet</character>
    </characters>
    <good>Good movie</good>
  </movie>
</movies>

>>> print movie['good'].text
Good movie
>>> movie.add('bad')
Traceback (most recent call last):
Exception: good is defined so you can't add bad
>>> movie['good'].delete()
>>> bad = movie.add('bad')
>>> print movies
<movies>
  <movie>
    <title>Titanic</title>
    <realisator>Cameron James</realisator>
    <characters>
      <character>Leonardo DiCaprio</character>
      <character>Kate Winslet</character>
    </characters>
    <bad></bad>
  </movie>
</movies>
```

## 2.4 List choice element

We will add a good-comment and a bad-comment

```
>>> import xmltool
>>> movies = xmltool.load('examples/tutorial1.xml')
>>> movie = movies['movie'][0]
>>> c1 = movie.add('good-comment', 'My comment 1')
>>> c2 = movie.add('bad-comment', 'My comment 2')
>>> print movies
<movies>
  <movie>
    <title>Titanic</title>
    <realisator>Cameron James</realisator>
    <characters>
      <character>Leonardo DiCaprio</character>
      <character>Kate Winslet</character>
    </characters>
    <good-comment>My comment 1</good-comment>
    <bad-comment>My comment 2</bad-comment>
  </movie>
</movies>

>>> movies.write()
```

Since {good,bad}-comment is a choice list it's a bit different to access, a property named list__tag1_tag2_... is automatically create. For this example it's list__good-comment_bad-comment.

```
>>> import xmltool
>>> movies = xmltool.load('examples/tutorial1.xml')
>>> comments = movies['movie'][0]['list__good-comment_bad-comment']
>>> print comments[0].text
My comment 1
>>> print comments[1].text
My comment 2
```

CHAPTER 3

# Migration

When the dtd has been updated, if there are some new required elements, we have to update the XML files as well. Since we can load an existing XML file without validating it, it's very easy to update it.

## 3.1 Updating a XML file after a dtd change

We use the same dtd as before:

```
<!ELEMENT movies (movie*)>
<!ELEMENT movie (title, date?, realisator, characters, (good-comment|bad-comment)*, (bad|good|awesome
<!ATTLIST movie idmovie ID #IMPLIED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT realisator (#PCDATA)>
<!ELEMENT characters (character+)>
<!ELEMENT character (#PCDATA)>
<!ATTLIST character idcharacter ID #IMPLIED>
<!ELEMENT good-comment (#PCDATA)>
<!ELEMENT bad-comment (#PCDATA)>
<!ELEMENT bad (#PCDATA)>
<!ELEMENT good (#PCDATA)>
<!ELEMENT awesome (#PCDATA)>
```

We just put the date element as required

```
<!ELEMENT movies (movie*)>
<!ELEMENT movie (title, date, realisator, characters, (good-comment|bad-comment)*, (bad|good|awesome)
<!ATTLIST movie idmovie ID #IMPLIED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT realisator (#PCDATA)>
<!ELEMENT characters (character+)>
<!ELEMENT character (#PCDATA)>
<!ATTLIST character idcharacter ID #IMPLIED>
<!ELEMENT good-comment (#PCDATA)>
<!ELEMENT bad-comment (#PCDATA)>
<!ELEMENT bad (#PCDATA)>
<!ELEMENT good (#PCDATA)>
<!ELEMENT awesome (#PCDATA)>
```

The XML doesn't have the date defined:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE movies SYSTEM "examples/movies-new.dtd">
<movies>
  <movie>
    <title>Titanic</title>
    <realisator>Cameron James</realisator>
    <characters>
      <character>Leonardo DiCaprio</character>
      <character>Kate Winslet</character>
    </characters>
    <good-comment>My comment 1</good-comment>
    <bad-comment>My comment 2</bad-comment>
  </movie>
</movies>
```

```
>>> import xmltool
>>> filename = 'examples/migration.xml'
>>> # It fails since by default we validate the XML follows the DTD
>>> obj = xmltool.load(filename)
Traceback (most recent call last):
...
DocumentInvalid:
>>> obj = xmltool.load(filename, validate=False)
>>> # The date tag is automatically added when generating the XML
>>> print obj
<movies>
  <movie>
    <title>Titanic</title>
    <date></date>
    <realisator>Cameron James</realisator>
    <characters>
      <character>Leonardo DiCaprio</character>
      <character>Kate Winslet</character>
    </characters>
    <good-comment>My comment 1</good-comment>
    <bad-comment>My comment 2</bad-comment>
  </movie>
</movies>
```

CHAPTER 4

# Form

Xmltool can also generate an HTML interface with a tree for the navigation. This documentation wil come soon.

CHAPTER 5

# Changelog

**O.4:**

- Rewrite javascript and add unittest
- Cleanup python code
- Remove deprecated functions

**O.3.7:**

- Cache to get the dtd content
- CKEditor render

**O.3.6.2:**

- Support to have comment inside a TextElement
- Improve HTML rendering
- Bug fix: make numdict_to_list works when we have more than 9 elements

**O.3.6.1:**

- Be able to skip attributes and comments when we load object

**O.3.6:**

- Add functions to update existing object with data from other one
- Some cleanup and fixes

**O.3.5:**

- Use grunt to manage the webmedia
- Use bootstrap to make the HTML rendering
- javascript performance
- xpath support
- Only use the dict style to access elements
- Deprecate some attributes

**O.3.4:**

- Make xmltool works with old version of lxml
- Be able to pass attributes to the HTML form

**O.3.3:**

- Fix missing require

**O.3.2:**

- Better support for local dtd. It can a relative path from the XML filename.
- Bug fix: the EMPTY tags are support correctly!

**O.3.1:**

- Render the text element as HTML format when exporting in XML (no autoclose tag)
- Bug fix: make sure we add empty text element in the HTML form

**O.3:**

- Rewrite the core of the code
- Better performance to generate the HTML
- Bug fix

**O.2:**

- Update the project architecture
- Be able to access to the element properties like a dict
- Add functions to easily update or create XML file
- Fix missing README file in the package

**O.1:**

- Initial version: package to manipulate XML file and create HTML forms.