XML parsing in Python

This article focuses on how one can parse a given XML file and extract some useful data out of it in a structured way.

XML: XML stands for eXtensible Markup Language. It was designed to store and transport data. It was designed to be both human- and machine-readable.That's why, the design goals of XML emphasize simplicity, generality, and usability across the Internet.
The XML file to be parsed in this tutorial is actually a RSS feed.

RSS: RSS(Rich Site Summary, often called Really Simple Syndication) uses a family of standard web feed formats to publish frequently updated informationlike blog entries, news headlines, audio, video. RSS is XML formatted plain text.

- The RSS format itself is relatively easy to read both by automated processes and by humans alike.
- The RSS processed in this tutorial is the RSS feed of top news stories from a popular news website. You can check it out here. Our goal is to process this RSS feed (or XML file) and save it in some other format for future use.

Python Module used: This article will focus on using inbuilt xml module in python for parsing XML and the main focus will be on the ElementTree XML API of this module.

Implementation:

```
#Python code to illustrate parsing of XML files
# importing the required modules
import csv
import requests
import xml.etree.ElementTree as ET

def loadRSS():

    # url of rss feed
    url = 'http://www.hindustantimes.com/rss/topnews/rssfeed.xml'

    # creating HTTP response object from given url
    resp = requests.get(url)

    # saving the xml file
    with open('topnewsfeed.xml', 'wb') as f:
        f.write(resp.content)


def parseXML(xmlfile):

    # create element tree object
    tree = ET.parse(xmlfile)

    # get root element
    root = tree.getroot()

    # create empty list for news items
    newsitems = []

    # iterate news items
    for item in root.findall('./channel/item'):

        # empty news dictionary
        news = {}

        # iterate child elements of item
        for child in item:

            # special checking for namespace object content:media
            if child.tag == '{http://search.yahoo.com/mrss/}content':
                news['media'] = child.attrib['url']
            else:
                news[child.tag] = child.text.encode('utf8')

        # append news dictionary to news items list
        newsitems.append(news)

    # return news items list
    return newsitems


def savetoCSV(newsitems, filename):

    # specifying the fields for csv file
    fields = ['guid', 'title', 'pubDate', 'description', 'link', 'media']

    # writing to csv file
    with open(filename, 'w') as csvfile:

        # creating a csv dict writer object
        writer = csv.DictWriter(csvfile, fieldnames = fields)

        # writing headers (field names)
        writer.writeheader()

        # writing data rows
        writer.writerows(newsitems)


def main():
    # load rss from web to update existing xml file
    loadRSS()
```

```
    # parse xml file
    newsitems = parseXML('topnewsfeed.xml')

    # store news items in a csv file
    savetoCSV(newsitems, 'topnews.csv')


if __name__ == "__main__":

    # calling main function
    main()
```

Above code will:

- Load RSS feed from specified URL and save it as an XML file.
- Parse the XML file to save news as a list of dictionaries where each dictionary is a single news item.
- Save the news items into a CSV file.

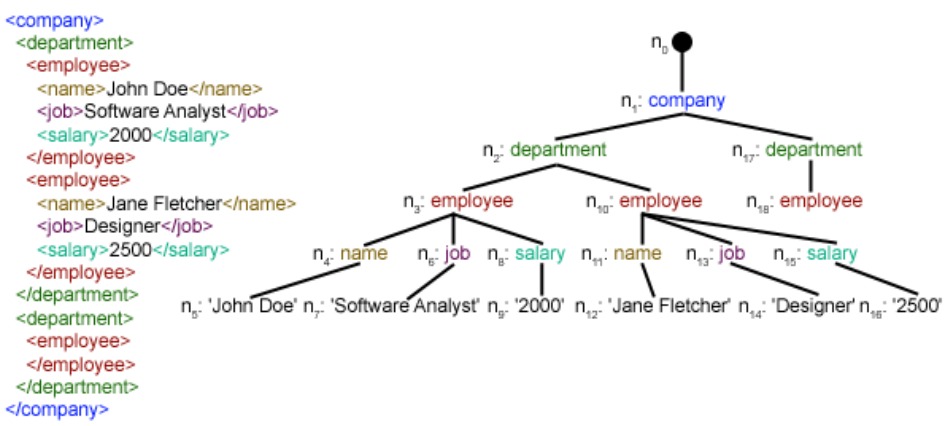
Let us try to understand the code in pieces:

- **Loading and saving RSS feed**

```
def loadRSS():
    # url of rss feed
    url = 'http://www.hindustantimes.com/rss/topnews/rssfeed.xml'
    # creating HTTP response object from given url
    resp = requests.get(url)
    # saving the xml file
    with open('topnewsfeed.xml', 'wb') as f:
        f.write(resp.content)
```

Here, we first created a HTTP response object by sending an HTTP request to the URL of the RSS feed. The content of response now contains the XML file data which we save as **topnewsfeed.xml** in our local directory.
For more insight on how requests module works, follow this article:
GET and POST requests using Python

- **Parsing XML**
We have created **parseXML()** function to parse XML file. We know that XML is an inherently hierarchical data format, and the most natural way to represent it is with a tree. Look at the image below for example:



Here, we are using **xml.etree.ElementTree** (call it ET, in short) module. Element Tree has two classes for this purpose – **ElementTree** represents the whole XML document as a tree, and **Element** represents a single node in this tree. Interactions with the whole document (reading and writing to/from files) are usually done on the **ElementTree** level. Interactions with a single XML element and its sub-elements are done on the **Element** level.

Ok, so let's go through the **parseXML()** function now:

```
tree = ET.parse(xmlfile)
```

Here, we create an **ElementTree** object by parsing the passed **xmlfile.**

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

getroot() function return the root of **tree** as an **Element** object.

```
for item in root.findall('./channel/item'):
```

Now, once you have taken a look at the structure of your XML file, you will notice that we are interested only in **item** element.
./channel/item is actually XPath syntax (XPath is a language for addressing parts of an XML document). Here, we want to find all **item** grand-children of **channel** children of the **root**(denoted by '.') element.
You can read more about supported XPath syntax here.

```
for item in root.findall('./channel/item'):

        # empty news dictionary
        news = {}

        # iterate child elements of item
        for child in item:

            # special checking for namespace object content:media
            if child.tag == '{http://search.yahoo.com/mrss/}content':
```

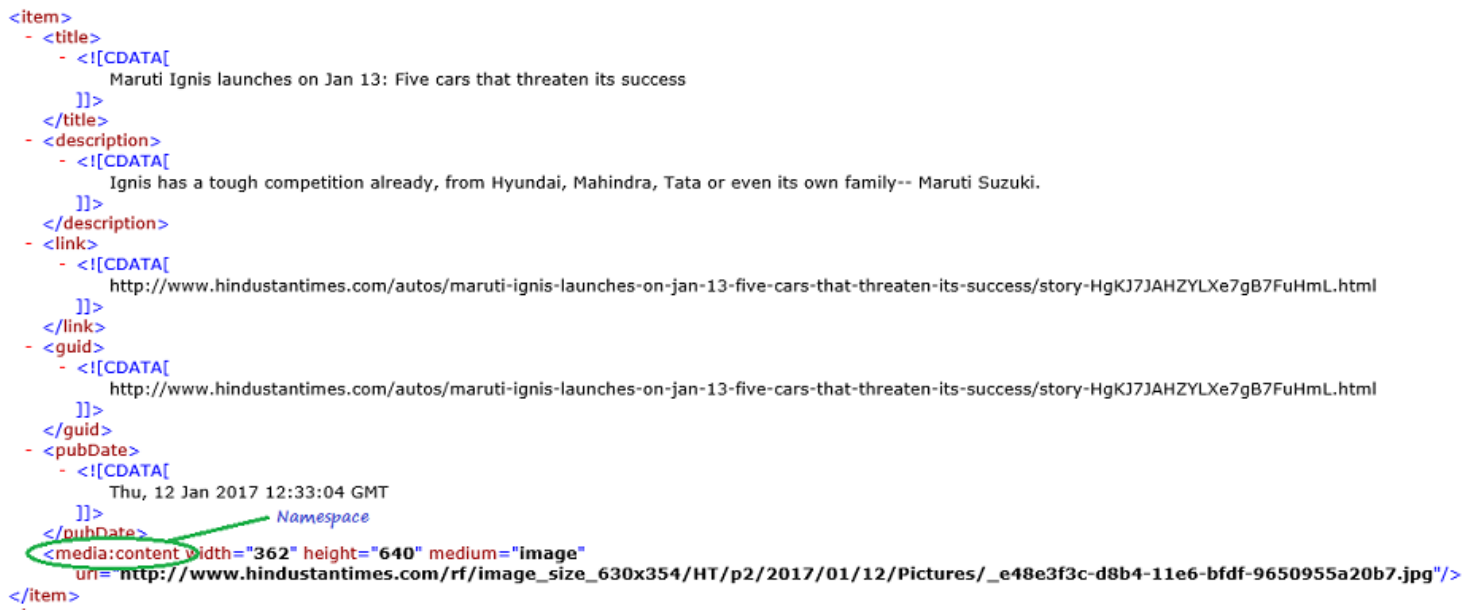
```
            news['media'] = child.attrib['url']
        else:
            news[child.tag] = child.text.encode('utf8')

    # append news dictionary to news items list
    newsitems.append(news)
```

Now, we know that we are iterating through **item** elements where each **item** element contains one news. So, we create an empty **news** dictionary in which we will store all data available about news item. To iterate though each child element of an element, we simply iterate through it, like this:

```
for child in item:
```

Now, notice a sample item element here:



We will have to handle namespace tags separately as they get expanded to their original value, when parsed. So, we do something like this:

```
if child.tag == '{http://search.yahoo.com/mrss/}content':
                news['media'] = child.attrib['url']
```

child.attrib is a dictionary of all the attributes related to an element. Here, we are interested in **url** attribute of **media:content** namespace tag.
Now, for all other children, we simply do:

```
news[child.tag] = child.text.encode('utf8')
```

child.tag contains the name of child element. **child.text** stores all the text inside that child element. So, finally, a sample item element is converted to a dictionary and looks like this:

```
{'description': 'Ignis has a tough competition already, from Hyun.... ,
 'guid': 'http://www.hindustantimes.com/autos/maruti-ignis-launch.... ,
 'link': 'http://www.hindustantimes.com/autos/maruti-ignis-launch.... ,
 'media': 'http://www.hindustantimes.com/rf/image_size_630x354/HT/... ,
 'pubDate': 'Thu, 12 Jan 2017 12:33:04 GMT ',
 'title': 'Maruti Ignis launches on Jan 13: Five cars that threa..... }
```

Then, we simply append this dict element to the list **newsitems**.
Finally, this list is returned.

- **Saving data to a CSV file**
  Now, we simply save the list of news items to a CSV file so that it could be used or modified easily in future using **savetoCSV()** function. To know more about writing dictionary elements to a CSV file, go through this article:
  Working with CSV files in Python

So now, here is how our formatted data looks like now:

	guid	title	pubDate	description	link	media
0	http://www.hindustantimes.com/business-news/ta...	Tata Sons appoints Natarajan Chandrasekaran as...	Thu, 12 Jan 2017 02:01:21 GMT	India's $100 billion salt-to-software conglome...	http://www.hindustantimes.com/business-news/ta...	http://www.hindustantimes.com/rf/image_size_63...
1	http://www.hindustantimes.com/assembly-electio...	Not funny: Bhagwant Mann under EC probe for sa...	Thu, 12 Jan 2017 01:32:23 GMT	The AAP, making its Punjab assembly poll debut...	http://www.hindustantimes.com/assembly-electio...	http://www.hindustantimes.com/rf/image_size_63...
2	http://www.hindustantimes.com/cricket/ajinkya-...	Ajinkya Rahane finds form, seals India A win o...	Thu, 12 Jan 2017 01:12:20 GMT	Ajinkya Rahane found some form ahead of the OD...	http://www.hindustantimes.com/cricket/ajinkya-...	http://www.hindustantimes.com/rf/image_size_63...
3	http://www.hindustantimes.com/autos/maruti-ign...	Maruti Ignis launches on Jan 13: Five cars tha...	Thu, 12 Jan 2017 12:33:04 GMT	Ignis has a tough competition already, from Hy...	http://www.hindustantimes.com/autos/maruti-ign...	http://www.hindustantimes.com/rf/image_size_63...
4	http://www.hindustantimes.com/delhi/delhi-cont...	Delhi temperature drop continues: City records...	Thu, 12 Jan 2017 11:18:22	A cold wave gripped Delhi and NCR as the	http://www.hindustantimes.com/delhi/delhi-cont...	http://www.hindustantimes.com/rf/image_size_63...

As you can see, the hierarchical XML file data has been converted to a simple CSV file so that all news stories are stored in form of a table. This makes it easier to extend the database too.

Also, one can use the JSON-like data directly in their applications! This is the best alternative for extracting data from websites which do not provide a public API but provide some RSS feeds.

All the code and files used in above article can be found here.

What next?

- You can have a look at more rss feeds of the news website used in above example. You can try to create an extended version of above example by parsing other rss feeds too.
- Are you a cricket fan? Then this rss feed must be of your interest! You can parse this XML file to scrape information about the live cricket matches and use to make a desktop notifier!

Quiz of HTML and XML

This article is contributed by **Nikhil Kumar.** If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Recommended Posts:

Important differences between Python 2.x and Python 3.x with examples
Python | Set 4 (Dictionary, Keywords in Python)
Python | Sort Python Dictionaries by Key or Value
try and except in Python
zip() in Python
SHA in Python
abs() in Python
chr() in Python
gcd() in Python
Any & All in Python
pow() in Python
Python Set | pop()
bin() in Python
max() and min() in Python
What other than Python and R?

Article Tags : GBlog Project Python python-utility

Be the First to upvote.

2.4

Based on **5** vote(s)

To-do Done

Feedback Add Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.