Working with minidom

To start out, well need some actual XML to parse. Take a look at the following short example of XML:

```
<?xml version="1.0" ?>
<zAppointments reminder="15">
    <appointment>
        <begin>1181251680</begin>
        <uid>040000008200E000</uid>
        <alarmTime>1181572063</alarmTime>
        <state></state>
        <location></location>
        <duration>1800</duration>
        <subject>Bring pizza home</subject>
    </appointment>
</zAppointments>
```

This is fairly typical XML and actually pretty intuitive to read. There is some really nasty XML out in the wild that you may have to work with. Anyway, save the XML code above with the following name: *appt.xml*

Let's spend some time getting acquainted with how to parse this file using Python's **minidom** module. This is a fairly long piece of code, so prepare yourself.

```
import xml.dom.minidom
import urllib.request

class ApptParser(object):

    def __init__(self, url, flag='url'):
        self.list = []
        self.appt_list = []
        self.flag = flag
        self.rem_value = 0
        xml = self.getXml(url)
        self.handleXml(xml)

    def getXml(self, url):
        try:
            print(url)
```

```
            f = urllib.request.urlopen(url)
        except:
            f = url

        doc = xml.dom.minidom.parse(f)
        node = doc.documentElement
        if node.nodeType == xml.dom.Node.ELEMENT_NODE:
            print('Element name: %s' % node.nodeName)
            for (name, value) in node.attributes.items():
                print('    Attr -- Name: %s  Value: %s' % (name, value))

        return node

    def handleXml(self, xml):
        rem = xml.getElementsByTagName('zAppointments')
        appointments = xml.getElementsByTagName("appointment")
        self.handleAppts(appointments)

    def getElement(self, element):
        return self.getText(element.childNodes)

    def handleAppts(self, appts):
        for appt in appts:
            self.handleAppt(appt)
            self.list = []

    def handleAppt(self, appt):
        begin     = self.getElement(appt.getElementsByTagName("begin")[0])
        duration  = self.getElement(appt.getElementsByTagName("duration")[0])
        subject   = self.getElement(appt.getElementsByTagName("subject")[0])
        location  = self.getElement(appt.getElementsByTagName("location")[0])
        uid       = self.getElement(appt.getElementsByTagName("uid")[0])

        self.list.append(begin)
        self.list.append(duration)
        self.list.append(subject)
        self.list.append(location)
        self.list.append(uid)
        if self.flag == 'file':

            try:
                state     = self.getElement(appt.getElementsByTagName("state")[0])
                self.list.append(state)
                alarm     = self.getElement(appt.getElementsByTagName("alarmTime")[0])
                self.list.append(alarm)
            except Exception as e:
                print(e)

        self.appt_list.append(self.list)

    def getText(self, nodelist):
        rc = ""
        for node in nodelist:
            if node.nodeType == node.TEXT_NODE:
                rc = rc + node.data
        return rc

if __name__ == "__main__":
    appt = ApptParser("appt.xml")
    print(appt.appt_list)
```



This code is loosely based on an example from the Python documentation and I have to admit that I think my mutation of it is a bit ugly. Let's break this code down a bit. The url parameter you see in the **ApptParser** class can be either a url or a file. In the **getXml** method, we use an exception handler to try and open the url. If it happens to raise an error, than we assume that the url is actually a file path. Next we use minidom's **parse** method to parse the XML. Then we pull out a node from the XML. We'll ignore the conditional as it isn't important to this discussion. Finally, we return the **node** object.

Technically, the node is XML and we pass it on to the **handleXml** method. To grab all the appointment instances in the XML, we do this:

```
xml.getElementsByTagName("appointment").
```

Then we pass that information to the **handleAppts** method. That's a lot of passing information around. It might be a good idea to refactor this code a bit to make it so that instead of passing information around, it just set class variables and then called the next method without any arguments. I'll leave this as an exercise for the reader. Anyway, all the **handleAppts** method does is loop over each appointment and call the **handleAppt** method to pull some additional information out of it, add the data to a list and add that list to another list. The idea was to end up with a list of lists that held all the pertinent data regarding my appointments.

You will notice that the handleAppt method calls the **getElement** method which calls the **getText** method. Technically, you could skip the call to getElement and just call getText directly. On the other hand, you may need to add some additional processing to getElement to convert the text to some other type before returning it back. For example, you may want to convert numbers to integers, floats or decimal.Decimal objects.

Let's try one more example with minidom before we move on. We will use an XML example from Microsoft's MSDN website: http://msdn.microsoft.com/en-us/library/ms762271(VS.85).aspx. Save the following XML as *example.xml*

```
<?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>
      <genre>Fantasy</genre>
      <price>5.95</price>
      <publish_date>2000-12-16</publish_date>
      <description>A former architect battles corporate zombies,
      an evil sorceress, and her own childhood to become queen
      of the world.</description>
   </book>
   <book id="bk103">
      <author>Corets, Eva</author>
      <title>Maeve Ascendant</title>
      <genre>Fantasy</genre>
      <price>5.95</price>
      <publish_date>2000-11-17</publish_date>
      <description>After the collapse of a nanotechnology
      society in England, the young survivors lay the
      foundation for a new society.</description>
   </book>
</catalog>
```

For this example, we'll just parse the XML, extract the book titles and print them to stdout. Here's the code:

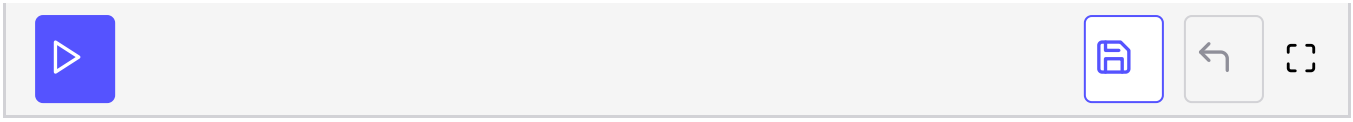
```
import xml.dom.minidom as minidom

def getTitles(xml):
    """
    Print out all titles found in xml
    """
    doc = minidom.parse(xml)
    node = doc.documentElement
    books = doc.getElementsByTagName("book")

    titles = []
    for book in books:
        titleObj = book.getElementsByTagName("title")[0]
        titles.append(titleObj)

    for title in titles:
        nodes = title.childNodes
        for node in nodes:
            if node.nodeType == node.TEXT_NODE:
                print(node.data)

if __name__ == "__main__":
    document = 'example.xml'
    getTitles(document)
```



This code is just one short function that accepts one argument, the XML file. We import the minidom module and give it the same name to make it easier to reference. Then we parse the XML. The first two lines in the function are pretty much the same as the previous example. We use the **getElementsByTagName** method to grab the parts of the XML that we want, then iterate over the result and extract the book titles from them. This actually extracts title objects, so we need to iterate over that as well and pull out the plain text, which is why we use a nested **for** loop.

Now let's spend a little time trying out a different sub-module of the xml module named **ElementTree**.