Creating XML with lxml.objectify

WE'LL COVER THE FOLLOWING

- Wrapping Up

The lxml.objectify sub-package is extremely handy for parsing and creating XML. In this section, we will show how to create XML using the lxml.objectify module. We'll start with some simple XML and then try to replicate it. Let's get started!

We will continue using the following XML for our example:

```
<?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>
    <appointment>
        <begin>1234360800</begin>
        <duration>1800</duration>
        <subject>Check MS Office website for updates</subject>
        <location></location>
        <uid>604f4792-eb89-478b-a14f-dd34d3cc6c21-1234360800</uid>
        <state>dismissed</state>
  </appointment>
</zAppointments>
```

Let's see how we can use lxml.objectify to recreate this XML:

```
from lxml import etree, objectify

def create_appt(data):
    """
    Create an appointment XML element
```

```python
    """
    appt = objectify.Element("appointment")
    appt.begin = data["begin"]
    appt.uid = data["uid"]
    appt.alarmTime = data["alarmTime"]
    appt.state = data["state"]
    appt.location = data["location"]
    appt.duration = data["duration"]
    appt.subject = data["subject"]
    return appt

def create_xml():
    """
    Create an XML file
    """

    xml = '''<?xml version="1.0" encoding="UTF-8"?>
    <zAppointments>
    </zAppointments>
    '''

    root = objectify.fromstring(xml)
    root.set("reminder", "15")

    appt = create_appt({"begin":1181251680,
                        "uid":"040000008200E000",
                        "alarmTime":1181572063,
                        "state":"",
                        "location":"",
                        "duration":1800,
                        "subject":"Bring pizza home"}
                       )
    root.append(appt)

    uid = "604f4792-eb89-478b-a14f-dd34d3cc6c21-1234360800"
    appt = create_appt({"begin":1234360800,
                        "uid":uid,
                        "alarmTime":1181572063,
                        "state":"dismissed",
                        "location":"",
                        "duration":1800,
                        "subject":"Check MS Office website for updates"}
                       )
    root.append(appt)

    # remove lxml annotation
    objectify.deannotate(root)
    etree.cleanup_namespaces(root)

    # create the xml string
    obj_xml = etree.tostring(root,
                             pretty_print=True,
                             xml_declaration=True)

    try:
        with open("example.xml", "wb") as xml_writer:
            xml_writer.write(obj_xml)
    except IOError:
        pass

if __name__ == "__main__":
    create_xml()
```

Let's break this down a bit. We will start with the **create_xml** function. In it we create an XML root object using the objectify module's **fromstring** function. The root object will contain **zAppointment** as its tag. We set the root's **reminder** attribute and then we call our **create_appt** function using a dictionary for its argument. In the **create_appt** function, we create an instance of an Element (technically, it's an **ObjectifiedElemen**t) that we assign to our **appt** variable. Here we use **dot-notatio**n to create the tags for this element. Finally we return the **appt** element back and append it to our **root** object. We repeat the process for the second appointment instance.

The next section of the **create_xml** function will remove the lxml annotation. If you do not do this, your XML will end up looking like the following:

```
<?xml version="1.0" ?>
<zAppointments py:pytype="TREE" reminder="15">
    <appointment py:pytype="TREE">
        <begin py:pytype="int">1181251680</begin>
        <uid py:pytype="str">040000008200E000</uid>
        <alarmTime py:pytype="int">1181572063</alarmTime>
        <state py:pytype="str"/>
        <location py:pytype="str"/>
        <duration py:pytype="int">1800</duration>
        <subject py:pytype="str">Bring pizza home</subject>
    </appointment><appointment py:pytype="TREE">
        <begin py:pytype="int">1234360800</begin>
        <uid py:pytype="str">604f4792-eb89-478b-a14f-dd34d3cc6c21-1234360800</uid>
        <alarmTime py:pytype="int">1181572063</alarmTime>
        <state py:pytype="str">dismissed</state>
        <location py:pytype="str"/>
        <duration py:pytype="int">1800</duration>
        <subject py:pytype="str">Check MS Office website for updates</subject>
    </appointment>
</zAppointments>
```

To remove all that unwanted annotation, we call the following two functions:

```
objectify.deannotate(root)
etree.cleanup_namespaces(root)
```

The last piece of the puzzle is to get lxml to generate the XML itself. Here we use lxml's **etree** module to do the hard work:

```
obj_xml = etree.tostring(root,
                         pretty_print=True,
                           xml_declaration=True)
```

The tostring function will return a nice string of the XML and if you set **pretty_print** to True, it will usually return the XML in a nice format too. The **xml_declaration** keyword argument tells the etree module whether or not to include the first declaration line (i.e. **<?xml version="1.0" ?>**.

Wrapping Up

Now you know how to use lxml's etree and objectify modules to parse XML. You also know how to use objectify to create XML. Knowing how to use more than one module to accomplish the same task can be valuable in seeing how to approach the same problem from different angles. It will also help you choose the tool that you're most comfortable with.