Scripting Programming Languages and their Applications

Jan Gaura

October 22, 2012

XML[1]

- XML is the ’Extensible Markup Language’, a data format for structured document interchange.
- XML is a set of rules for encoding documents in machine-readable form.

- simplicity
- generality
- usability over the Internet

- textual data format
- Unicode support for the languages of the world
- can represent arbitrary data structures

[1] contains citations from http://en.wikipedia.org/wiki/XML

XML Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<painting>
  <img src="madonna.jpg"
       alt="Foligno Madonna, by Raphael"/>
  <caption>This is Raphael's "Foligno" Madonna, painted in
      <date>1511</date>–<date>1512</date>.
  </caption>
</painting>
```

XML Based Languages

- RSS
- Atom
- XML-RPC
- SOAP
- XHTML
- Microsoft Office (Office Open XML)
- OpenOffice.org (OpenDocument)
- Apple’s iWork

XML Nomenclature I

- Processor and Application
  - processor (XML parser) analyzes the markup and passes structured information to an application
- Markup and Content
  - strings which constitute markup either begin with the character "<" and end with a ">", or begin with the character "&" and end with a ";"
  - strings of characters which are not markup are content
- Tag
  - markup within "<" and ">"
  - start-tags: <section>
  - end-tags: </section>
  - empty-element tags: <line-break/>

XML Nomenclature II

- Element
  - logical component of a document within start-tag and matching end-tag (both included), or consists only of an empty-element tag
  - characters between the start- and end-tags, if any, are the element's content
  - may contain markup, including other elements, which are called child elements
- Attribute
  - construct consisting of name/value pair within a start-tag or empty-element tag

XML Well-formedness

- begin, end, and empty-element tags are correctly nested, with none missing and none overlapping
- element tags are case-sensitive
- there is a single "root" element which contains all the other elements

APIs for XML Manipulation in Python

- ElementTree - the xml.etree package
- event-driven XML parser implementing SAX, the Simple API for XML:
  - Sax - xml.sax package
- XML tree libraries that adhere to the W3C DOM standard:
  - MiniDom - xml.dom.minidom package
  - PullDom - xml.dom.pulldom package

XML Parsing Example I

```xml
<?xml version="1.0" encoding="UTF-8"?>
<menza>
    <datum den="Pondeli">
        <jidlo nazev="Bramborove placky">
            <ingredience nazev="brambory" />
            <ingredience nazev="mouka" />
            <ingredience nazev="vejce" />
        </jidlo>
        <jidlo nazev="Palacinky">
            <ingredience nazev="mleko" />
            <ingredience nazev="mouka" />
            <ingredience nazev="vejce" />
        </jidlo>
    </datum>
</menza>
```

ElementTree Example II

```
1 import xml.etree.ElementTree as ET

3 root = ET.parse('menza.xml') #ElementTree instance

5 datums = root.findall('datum') #list of Element s

7 for datum in datums:
      print datum.attrib['den']
9     jidla = datum.getiterator('jidlo')
      for jidlo in jidla:
11        print jidlo.attrib['nazev']
          for ingredience in jidlo.getiterator('ingredience'):
13            print ingredience.attrib['nazev']
```

ElementTree Example III - Output

```
Pondeli
    Bramborove placky
        brambory
        mouka
        vejce
    Palacinky
        mleko
        mouka
        vejce
```

Create XML from Application I

```
1  import xml.etree.ElementTree as ET

3  # build a tree structure
   root = ET.Element("html")
5
   head = ET.SubElement(root, "head")
7
   title = ET.SubElement(head, "title")
9  title.text = "Page Title"

11 body = ET.SubElement(root, "body")
   body.set("bgcolor", "#ffffff")
13
   body.text = "Hello, World!"
15
   # wrap it in an ElementTree instance, and save as XML
17 tree = ET.ElementTree(root)
   tree.write("page.xhtml")
```

Create XML from Application II

```
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body bgcolor="#ffffff">
    Hello, World!
  </body>
</html>
```

Convert XML to Python Object I

```
def xml2py(node):
    name = node.tag

    pytype = type(name, (object, ), {})
    pyobj = pytype()

    for attr in node.attrib.keys():
        setattr(pyobj, attr, node.get(attr))

    if node.text and node.text != '' and node.text != ' ' \
                and node.text != '\n':
        setattr(pyobj, 'text', node.text)

    for cn in node:
        if not hasattr(pyobj, cn.tag):
            setattr(pyobj, cn.tag, [])
        getattr(pyobj, cn.tag).append(xml2py(cn))

    return pyobj
```

Convert XML to Python Object II[2]

```
for datum in obj.datum:
    print datum.den
    for jidlo in datum.jidlo:
        print jidlo.nazev
        for ingredience in jidlo.ingredience:
            print ingredience.nazev
```

[2] I don't remenber anymore where I got the base for the previous code

Convert XML to Python Object III

- in CLR, we can do it using LINQ to XML, which is new language
- in Python, we just use dynamic language features
- we don’t need to ’create’ a new DSL (Domain Specific Language)

References

- Alex Martelli, Painless Python for Proficient Programmers
- Django documentation
- Adam Fast, intro to geodjango