Exploring Beautiful Languages

Programming language exploration blog

Thursday, February 12, 2009

Writing Xml with IronPython, XmlWriter and the 'with' statement

This post shows a little example of wrapping calls to `System.Xml.XmlWriter` inside a Python's 'with' statement using IronPython.

Writing Xml

While reading some code examples from the XIST HTML/XML generation library I noticed the nice use of Python's `'with'` statement to represent the target HTML or XML.

The System.Xml.XmlWriter class provided with .NET already gives you a way to write well formed Xml documents. In this post I'm going to show how to use an XmlWriter instance in conjunction with Python's 'with' statement.

We want to write the following code:

```
from __future__ import with_statement

...

w = XmlWriter.Create(System.Console.Out,XmlWriterSettings(Indent=True))
x = XWriter(w)

with x.element('tableofcontents'):
    with x.element('section',{'page' : '10'}):
       x.text('Introduction')
    with x.element('section',{'page' : '12'}):
       x.text('Main topic')
    with x.element('section',{'page' : '14'}):
       x.text('Extra topic')
```

To generate the following Xml file:

```
<tableofcontents>
  <section page="10">Introduction</section>
  <section page="12">Main topic</section>
  <section page="14">Extra topic</section>
</tableofcontents>
```

The 'with' statement

The 'with' statement was introduced in Python 2.5 . This statement is used wrap the execution of a series of statements with some special code. For example it is used to implement the `try...except...finally` pattern.

As described in the documentation the following statement:

```
with context expression:
   statements...
```

Will be executed as follows:

1. Evaluate the context expression to obtain the context manager
2. Invoke the context manager's `__enter__()` method
3. Execute the statements

About Me

Luis Diego Fallas

Programming language

View my complete prof

Blog Archive

- ► 2021 (4)
- ► 2020 (1)
- ► 2019 (1)
- ► 2018 (1)
- ► 2017 (7)
- ► 2016 (4)
- ► 2015 (5)
- ► 2014 (1)
- ► 2012 (6)
- ► 2011 (5)
- ► 2010 (12)
- ▼ 2009 (16)
  - ► November (1)
  - ► October (3)
  - ► August (1)
  - ► July (1)
  - ► June (3)
  - ► May (1)
  - ► April (2)
  - ► March (2)
  - ▼ February (1)
    - Writing Xml with I XmlWriter and
  - ► January (1)
- ► 2008 (30)
- ► 2007 (43)

Labels

- .net (11)
- abcexplorationlib (2
- actionscript (9)
- air (1)
- apl (6)
- arrays (1)
- basic (1)
- boo (3)
- c (2)
- c# (13)
- c++ (1)
- continuations (1)
- dlr (2)
- eclipse (3)
- ecmascript (1)
- emacs (1)
- erlang (1)
- examples (1)
- f# (31)
- flash (1)
- flex (3)

4. When the execution of the statements finishes(even with an exception), the context manager's `__exit()__` method is called.

Given these steps we're going to implement a context manager that assist in the creation of Xml documents using the `System.Xml.XmlWriter` .NET class.

The following code shows a class that wraps the XmlWriter instance and helps with the creation of context managers:

```
class XWriter(object):
   def __init__(self,writer):
      self.writer = writer

   def element(self,name,atts = {}):
       return ElementCtxt(name,atts,self)

   def nselement(self,prefix,name,namespace,atts = {}):
       return NamespaceElementCtxt(prefix,name,namespace,atts,self)


   def text(self,text):
       self.writer.WriteString(text)

   def cdata(self,text):
       self.writer.WriteCData(text)
```

Notice that the `element` method creates an instance of the `ElementCtxt` class using the element name and an optional dictionary with the attributes. As the following listing shows this class performs the calls to WriteStartElement and WriteEndElement in the `__enter__` and `__exit__` methods.

```
class ElementCtxt(object):
   def __init__(self,elementName,atts,writer):
      self.elementName = elementName
      self.atts = atts
      self.writer = writer

   def processAttributes(self):
      for att in self.atts:
         self.writer.writer.WriteAttributeString(att,self.atts[att].__str__())

   def processStartTag(self):
      self.writer.writer.WriteStartElement(self.elementName)
      self.processAttributes()

   def __enter__(self):
      self.processStartTag()
      return self

   def __exit__(self,t,v,tr):
      self.writer.writer.WriteEndElement()
      return t == None
```

The `XWriter.nselement` method is used to write elements with namespace and prefix. This call generates an instance of the following context manager:

```
class NamespaceElementCtxt(ElementCtxt):
   def __init__(self,prefix,elementName,namespace,atts,writer):
      ElementCtxt.__init__(self,elementName,atts,writer)
      self.namespace = namespace
      self.prefix = prefix
   def processStartTag(self):
      self.writer.writer.WriteStartElement(self.prefix,self.elementName,self.namespace)
      self.processAttributes()
```

Final example

The following code shows how to create a little SVG file:

- fortress (3)
- fsharp (5)
- functional (5)
- gameoflife (2)
- glsl (1)
- google (3)
- groovy (4)
- gwbasic (1)
- haskell (6)
- haxe (1)
- ironpython (11)
- ironruby (3)
- j (3)
- java (9)
- java3d (3)
- javafx (1)
- javascript (3)
- jruby (5)
- linq (6)
- lisp (3)
- list comprehensions
- logic (1)
- m (1)
- mef (1)
- mercury (5)
- metaprogramming (1
- mgrammar (1)
- mvvm (1)
- netbeans (4)
- newspeak (10)
- objective-c (2)
- oo (3)
- opengl (1)
- parallel (1)
- pattern matching (1
- perl (1)
- pharo (5)
- podcasts (1)
- powershell (2)
- prolog (5)
- python (11)
- racket (1)
- racklog (1)
- refactoring (2)
- relaxng (1)
- ruby (14)
- rust (1)
- scala (12)
- scheme (4)
- silverlight (9)
- smalltalk (11)
- snobol (4)
- squeak (2)
- tom (5)
- vb9 (1)
- vbnet (1)
- wpf (2)
- xaml (5)
- xml (10)
- xsd (3)

Work

Artinsoft

Open Source

- GitHub profile

```
from __future__ import with_statement
from xmlwriterw import XWriter

import clr

clr.AddReference('System.Xml')

from System.Xml import *
import System


w = XmlWriter.Create(System.Console.Out,\
                     XmlWriterSettings(Indent=True))
x = XWriter(w)

svgNs = 'http://www.w3.org/2000/svg'

with x.nselement('s','svg',svgNs,{'version': '1.1',
                                  'viewBox': '0 0 100 100',
                                  'style':'width:100%; height:100%; position:absolute; top:0; left:0; z-index:-1;'}):
   with x.nselement('s','linearGradient',svgNs, { 'id' : 'gradient' }):
       with x.nselement('s','stop',svgNs, {'class' : 'begin',
                                           'offset' : '0%',
                                           'stop-color':'red'}):
          pass
       with x.nselement('s','stop',svgNs, {'class' : 'end',
                                           'offset' : '100%'}):
          pass
   with x.nselement('s','rect',svgNs, { 'x':0,
                                        'y':0,
                                        'width':100,
                                        'height':100,
                                        'style':'fill:url(#gradient)'} ):
         pass
   for i in range(1,5):
      with x.nselement('s','circle',svgNs, { 'cx': 50,
                                             'cy': 50,
                                             'r': 30 - i*3,
                                             'style':'fill:url(#gradient)'} ):
          pass

w.Close()
```

Running this program shows:

```
<s:svg viewBox="0 0 100 100" style="width:100%; height:100%; position:absolute; top:0; left:0; z-index:-1;" version="1.1" xmlns:s="http://www
  <s:linearGradient id="gradient">
    <s:stop offset="0%" class="begin" stop-color="red" />
    <s:stop offset="100%" class="end" />
  </s:linearGradient>
  <s:rect x="0" height="100" width="100" style="fill:url(#gradient)" y="0" />
  <s:circle cx="50" cy="50" style="fill:url(#gradient)" r="27" />
  <s:circle cx="50" cy="50" style="fill:url(#gradient)" r="24" />
  <s:circle cx="50" cy="50" style="fill:url(#gradient)" r="21" />
  <s:circle cx="50" cy="50" style="fill:url(#gradient)" r="18" />
</s:svg>
```

Posted by Luis Diego Fallas at 5:23 AM
Labels: ironpython, python

Newer Post Home Older Post