XML Validation with Python, MSXML and comtypes

André Burgaud

2008-02-09

In XML Schema Validation with Python, MSXML and PyWin32, I described how to use PyWin32 and MSXML to validate XML contents against an XML Schema. This article is an adaptation of the previous, substituting `PyWin32` with `comtypes` to perform XML schema validation with Python on Windows.

The `comtypes` package is a pure Python COM implementation built on top of `ctypes` FFI (Foreign Function Interface). Thomas Heller created both packages, `comtypes` and `ctypes`.

Requirements

- Python is the first requirement, preferably Python 2.5, as it already includes `ctypes` (see below)
- Install comtypes
- The `ctypes` library is only needed for Python 2.3 or 2.4. Python 2.5 already includes `ctypes` in its standard library.
- Download and install Download MSXML 6.0 SP1. MSXML 4.0 SP2) would also work whereas MSXML 3.0 does not support XML Schema.
- Optional: py_tips_msxml_val_comtypes_1.0.zip, code source and files used to illustrate this article and available under the MIT License.

XML Files

The XSD, XML Schema Definition file, and the XML files used are: * `books.xsd`: an XML Schema * `books.xml`: a valid XML file (according to `books.xsd`) * `books_error.xml`: a non valid XML (according to `books.xsd`)

books.xsd (XML Schema)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: books.xsd 439 2008-01-26 02:08:18Z andre $ -->
<xsd:schema xmlns="http://www.burgaud.com/XMLSchema"
            targetNamespace="http://www.burgaud.com/XMLSchema"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified">

  <xsd:element name="Books">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="Book">
    <xsd:complexType>
```

```
      <xsd:sequence>
        <xsd:element ref="Title"/>
        <xsd:element ref="Authors"/>
      </xsd:sequence>
      <xsd:attribute name="isbn" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:pattern value="[0-9]{10}"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="Authors">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Author" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="Title" type="xsd:string" />
  <xsd:element name="Author" type="xsd:string" />

</xsd:schema>
```

books.xml (Valid XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: books.xml 439 2008-01-26 02:08:18Z andre $ -->
<Books xmlns="http://www.burgaud.com/XMLSchema">
  <Book isbn="2841771210">
    <Title>Developpement d'applications avec Objective Caml</Title>
    <Authors>
      <Author>Emmanuel Chailloux</Author>
      <Author>Pascal Manoury</Author>
      <Author>Bruno Pagano</Author>
    </Authors>
  </Book>
  <Book isbn="0201889544">
    <Title>C++ Programming Language</Title>
    <Authors>
      <Author>Bjarne Stroustrup</Author>
    </Authors>
  </Book>
  <Book isbn="0201710897">
    <Title>Programming Ruby</Title>
    <Authors>
      <Author>David Thomas</Author>
      <Author>Andrew Hunt</Author>
    </Authors>
  </Book>
```

```
</Books>
```

books_error.xml (Non valid XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: books_error.xml 439 2008-01-26 02:08:18Z andre $ -->
<Books xmlns="http://www.burgaud.com/XMLSchema">
  <Book isbn="2841771210">
    <Title>Developpement d'applications avec Objective Caml</Title>
    <Authors>
      <Author>Emmanuel Chailloux</Author>
      <Author>Pascal Manoury</Author>
      <Author>Bruno Pagano</Author>
    </Authors>
  </Book>
  <!-- Intentional error in the isbn format:
  3 digits instead of 10 (see XML Schema books.xsd) -->
  <Book isbn="123">
    <Title>C++ Programming Language</Title>
    <Authors>
      <Author>Bjarne Stroustrup</Author>
    </Authors>
  </Book>
  <Book isbn="0201710897">
    <Title>Programming Ruby</Title>
    <Authors>
      <Author>David Thomas</Author>
      <Author>Andrew Hunt</Author>
    </Authors>
  </Book>
</Books>
```

Testing in the Python Interpreter

The logic is similar to the one observed with PyWin32. The code below, in the Python command line, shows how to validate an XML document with `comtypes` and `MSXML`.

```
>>> from comtypes.client import CreateObject
>>> dom = CreateObject('Msxml2.DOMDocument.6.0')
>>> dom.async = 0
>>> schemas = CreateObject('Msxml2.XMLSchemaCache.6.0')
>>> schemas.add('http://www.burgaud.com/XMLSchema', 'books.xsd')
0
>>> dom.schemas = schemas
>>> dom.load('books.xml')
True
>>> dom.load('books_error.xml')
False
>>> dom.parseError.reason
u"pattern constraint failed.\r\nThe attribute: 'isbn' has an invalid value according to its data type.\
>>>
```

Complete Python Example

The following Python script takes an XML file, the data, and an XSD file, the schema, and validates the content of the XML file against the XML schema.

```
"""Validate an XML file (1st param) against an XML schema
(2nd param), using MSXML and comtypes.

Copyright 2008 (c) Andre Burgaud
MIT License

Version 1.0

$Id: msxml_schema_val.py 445 2008-01-27 05:13:47Z andre $
"""

from comtypes.client import CreateObject
from _ctypes import COMError
import os
import sys

FORMAT_ERROR = """
%s: Validation Error
- Error Code : %s
- Reason     : %s
- Character  : %s
- Line       : %s
- Column     : %s
- Source     : %s
               %s"""

FORMAT_SUCCESS = """
Namespace : %s
Schema    : %s
Valid XML : %s
"""

# MSXML versions to use
MSXML_VERSIONS = [6, 4]
MSXML_DOM = 'Msxml2.DOMDocument.%d.0'
MSXML_SCHEMAS = 'Msxml2.XMLSchemaCache.%d.0'

def find(xml_file, xsd_file):
  """Validate existence of files."""
  for xfile in (xml_file, xsd_file):
    if not os.path.exists(xfile):
      print 'File %s not found...' % xfile
      return False
  return True

class MsXmlValidator:
  """Encapsulate some basic MXSML functionalities to validate an XML
  file against an XML Schema.
  """
```

```
def __init__(self):
  """Instantiate DOMDocument and XMLSchemaCache."""
  self.dom = self.create_dom()
  self.schemas = self.create_schemas()
  self.msxml_version = 0

def create_dom(self):
  """Create and return a DOMDocument."""
  versions = MSXML_VERSIONS
  versions.sort(reverse=True)
  dom = None
  for version in versions:
    try:
      prog_id = MSXML_DOM % version
      dom = CreateObject(prog_id)
      self.msxml_version = version
      print 'MSXML version %s: OK' % version
      break
    except WindowsError, msg:
      print 'Error:', msg
      print 'MSXML version %d: problem on this system' % version
  if not dom:
    print 'No compatible MSXML versions found on this system'
    sys.exit()
  dom.async = 0 # false
  return dom

def create_schemas(self):
  """Create and return a Schema Cache."""
  schemas = CreateObject(MSXML_SCHEMAS % self.msxml_version)
  return schemas

def get_namespace(self, xsd_file):
  """Extract targetNamespace, if any,  from XML schema."""
  namespace = ''
  self.dom.load(xsd_file)
  self.dom.setProperty("SelectionLanguage", "XPath")
  path = "/*/@targetNamespace"
  node = self.dom.documentElement.selectSingleNode(path)
  if node:
    namespace = node.text
  return namespace

def add_schema(self, namespace, xsd_file):
  """Add schema and namespace to Schema Cache."""
  try:
    self.schemas.add(namespace, xsd_file)
  except COMError, msg:
    print 'Error in XML Schema: %s' % xsd_file
    print msg
    sys.exit()
  self.dom.schemas = self.schemas
```

```
  def validate_xml_file(self, xml_file, xsd_file):
    """Validate XML file against XML Schema file."""
    if not find(xml_file, xsd_file):
      sys.exit()
    namespace = self.get_namespace(xsd_file)
    self.add_schema(namespace, xsd_file)
    if self.dom.load(xml_file):
      print FORMAT_SUCCESS % (namespace, xsd_file, xml_file)
    else:
      error = self.dom.parseError
      print FORMAT_ERROR % (xml_file,
                            error.errorCode,
                            error.reason.strip(),
                            error.filepos,
                            error.line,
                            error.linepos,
                            error.srcText,
                            " " * (error.linepos - 1) + '^')

def main():
  """Handle parameters, create Validator and invoke validation."""
  if len(sys.argv) < 3:
    print 'Usage: %s xml_file xsd_file' % sys.argv[0]
    sys.exit()
  validator = MsXmlValidator()
  xml_file, xsd_file = sys.argv[1], sys.argv[2]
  validator.validate_xml_file(xml_file, xsd_file)

if __name__ == '__main__':
  main()
```

The main difference with the version using PyWin32 is the usage of `comtypes.client.CreateObject` instead of `win32com.client.Dispatch`. The exception handling is also slightly different.

This script takes 2 parameters (XML file and XML Schema file). Here is an example of the execution with the valid XML file:

```
C:\prompt>python msxml_schema_val.py books.xml books.xsd
MSXML version 6: OK
Namespace : http://www.burgaud.com/XMLSchema
Schema    : books.xsd
Valid XML : books.xml
```

And an example with the non valid XML:

```
MSXML version 6: OK
books_error.xml: Validation Error
- Error Code : -1072897687
- Reason     : '123' violates pattern constraint of '[0-9]{10}'.
The attribute 'isbn' with value '123' failed to parse.
- Character  : 534
- Line       : 14
- Column     : 20
- Source     :   <Book isbn="123">
                                ^
```

For more information about `comtypes`, check out the comtypes documentation

Source Code

You can download the source code released under the MIT License

History

- **02/08/2008**: Initial publication of this blog post.