XML Validation with Python, MSXML and PyWin32

André Burgaud

2008-01-19

Python ships with XML libraries, but none addressing XML Schema validation. Several options are available to make up for this missing feature. The solution demonstrated in this article is a basic implementation of XML schema validation using MSXML and PyWin32. This is only compatible with Windows.

Related article: XML Validation with Python, MSXML and comtypes

Requirements

https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms763742(v=vs.85)

In order to try the code and examples exposed in this article, it is assumed that you have Python already installed on your system and that you are familiar with this popular programming language. In addition you will need to the following software:

- MSXML 4.0 or better MSXML 6.0 (MSXML 3.0 does not support XML Schema)
- Python for Windows Extensions PyWin32
- Optional: py_tips_msxml_val_pywin32_11.zip, code source and files used to illustrate this article and available under the MIT License.

  Notes ActivePython is a Python distribution for Windows including PyWin32, therefore you don't need to install PyWin32 if you install the ActivePython distribution.

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)

The following file is an XML schema describing the content, type and constraints of the desired XML data.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: books.xsd 435 2008-01-20 02:15:28Z 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)

The following XML is a valid XML for the schema defined in the previous section.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: books.xml 435 2008-01-20 02:15:28Z 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)

The following XML file is intentionally incorrect and does not validate with `books.xsd`.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: books_error.xml 436 2008-01-20 02:33:53Z 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

`win32com` is a module from the Python for Windows Extensions, it wraps the Windows COM API's. The `win32com.client.Dispatch()` method creates COM objects from their ProgID's. The 2 ProgID's used are respectively: `Msxml2.DOMDocument.6.0` (DOM document), and `Msxml2.XMLSchemaCache.6.0` (schemas collection). You may have to use a different version if you have only MSXML 4.0 installed. After creating the DOM document COM object, set its `async` property to `false` (`dom.async = 0`) to use the `load` method in synchronous mode. Create a schema collection and add the XML schema and its namespace if any (if no namespace, an empty string is used instead). Assign the schema collection to the DOM document instance and load the document. The `load` method returns `True` if the validation is successful:

```
>>> import win32com.client
>>> dom = win32com.client.Dispatch('Msxml2.DOMDocument.6.0')
>>> dom.async = 0
>>> schemas = win32com.client.Dispatch('Msxml2.XMLSchemaCache.6.0')
>>> schemas.add('http://www.burgaud.com/XMLSchema', 'books.xsd')
>>> dom.schemas = schemas
>>> dom.load('books.xml')
True
>>>
```

If the validation fails, the `load` method returns `False`. In addition the `parseError` property of the DOM document contains information about the error:

```
...
>>> dom.load('books_error.xml')
False
>>> dom.parseError.errorCode
-1072897687
>>> dom.parseError.reason
u"'020188954' violates pattern constraint of '[0-9]{10}'.\r\nThe attribute 'isbn' with value '020188954
```

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 PyWin32.

Copyright 2008 (c) Andre Burgaud
MIT License

Version 1.1

$Id: msxml_schema_val.py 435 2008-01-20 02:15:28Z andre $
"""

from win32com.client import Dispatch
import os
import sys
from pywintypes import com_error

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
    # Loop true the possible MSXML versions
    # Pick the first one created successfully
    for version in versions:
      try:
        dom = Dispatch(MSXML_DOM % version)
        self.msxml_version = version
        print 'MSXML version %s: OK' % version
        break
      except com_error, msg:
        print 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 = Dispatch(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 com_error, 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 namespace is discovered from the XML schema file, using the XPath feature from MSXML. In a real context this approach might be questionable and the namespace could be provided as a parameter to the script for example.

This script takes 2 parameters, an XML file and an 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 of parsing the non valid XML:

```
C:\prompt>python msxml_schema_val.py books_error.xml books.xsd
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">
                                  ^
```

Source Code

You can download the source code released under the MIT License

Release Notes

- **01/26/2008**:
  - Updated source code and companion files (version 1.1)
- **01/19/2008**:
  - Initial publication of this blog post.