Inspirage®

INSPIRING VALUE CHAIN ADVANTAGE

An open source approach to integrating with OTM using Python and XSLT

OTM SIG 2013
Eric Rosenbloom
eric.rosenbloom@inspirage.com

ORACLE Platinum Partner

Introduction

- This technically oriented talk focuses on the use of Python and XSLT to integrate with OTM.
- Python is used for transforming flat files into intermediate XML. XSL is used to transform intermediate XML into GLog XML.
- Python also provides data transport mechanisms.
- We present a working example that you can use as a template to build your own integrations via an open source approach.

Your Take-Away

- Get an introduction to Python and XSLT
- Learn to use Python to transform flat files to XML
- Learn to apply XSL transformations within a Python script
- Learn to transmit XML files to OTM from Python

Integration Diagram

Integration Server Design Points

- File extension of flat file indicates content type (and implies required data transformation)
  - Example: 2013043016402378.loc must contains location data
- File name itself is a timestamp of the form YYYYMMDDHHmmSSSS where YYYY = year, MM = month, DD = day, HH = hour, mm = minutes, SSSS = milliseconds
- Integration server processes files in FIFO sequence, as determined by file name (upstream system is responsible for processing sequence)
- Upstream system transfers file with a temporary extension of ".tmp" and then renames the file to ".loc" (for example)
  - This way integration server will ignore any file that is in the middle of being transferred.

Integration Server Design Points

- Upstream system is responsible for placing flat files in an agreed-to directory using secure ftp
- Integration server polls a given directory at a given polling interval
- When polling this directory:
  - The server builds a list of files to be processed
  - The sequence of the files in the list is based on the natural sort order of the file names
- The server processes each file in the list:
  - File name extension determines which data transformation is used

Integration Server Design Points

- Data transformations occur in two steps:
  1. Transform flat file to an “intermediate” XML format
  2. Transform intermediate XML to GLog XML using XSL

- Note: This two-step data transformation approach is a preference vs. a requirement. Some may prefer to avoid use of XSL altogether and transform directly from flat file to GLog XML 100% in Python code

- Integration Server pushes GLogXML to OTM via either
  - HTTP to WMServlet (which we will illustrate), or
  - Direct XML Insert (a performance enhancement)

Integration Server Design Points

- Reverse Flow (Planned Shipment, for example)
  - OTM is configured to push Planned Shipment XML to a given directory via ftp
  - Integration Server polls the directory for XML files (FIFO sequence)
  - GLog XML is flattened to upstream system's required format
  - Integration Server uses sftp to push flat file to agreed-to directory on upstream server

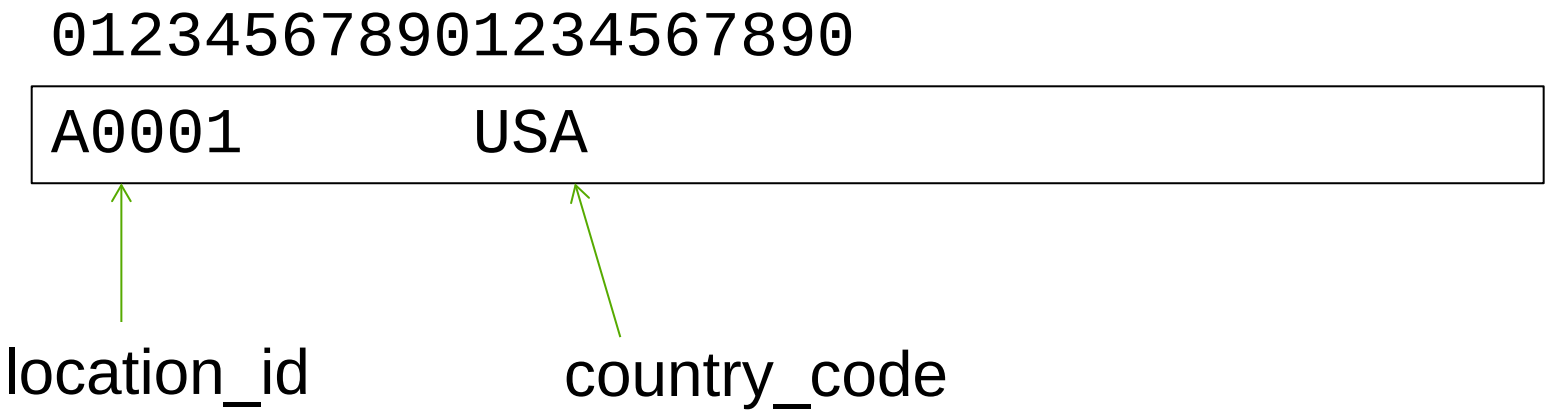
A Very Simple Location Example

- We will illustrate the inbound flow using the simplest OTM Location XML possible
- This way we illustrate the approach without getting overwhelmed with excessive detail

Our Very Simple Location Example

```
<?xml version="1.0"?>
- <Transmission>
  - <TransmissionHeader>
      <UserName>GUEST.ADMIN</UserName>
      <Password>CHANGEME</Password>
    </TransmissionHeader>
  - <TransmissionBody>
    - <GLogXMLElement>
      - <Location>
          <TransactionCode>IU</TransactionCode>
        - <LocationGid>
          - <Gid>
              <DomainName>GUEST</DomainName>
              <Xid>A0001</Xid>
            </Gid>
          </LocationGid>
        - <Address>
          - <CountryCode3Gid>
              <Xid>USA</Xid>
            </CountryCode3Gid>
          </Address>
        </Location>
      </GLogXMLElement>
    </TransmissionBody>
</Transmission>
```

Upstream Flat File (Fixed Field)



Intermediate XML

```
<?xml version="1.0"?>
- <doc>
  - <location>
      <location_id>A0001</location_id>
      <country_code>USA</country_code>
    </location>
</doc>
```

XSL Transformation

```
<?xml version="1.0" encoding="UTF-8"?>
- <Transmission xsl:version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  - <TransmissionHeader>
      <UserName>GUEST.ADMIN</UserName>
      <Password>CHANGEME</Password>
    </TransmissionHeader>
  - <TransmissionBody>
    - <xsl:for-each select="/doc/location">
      - <GLogXMLElement>
        - <Location>
            <TransactionCode>IU</TransactionCode>
          - <LocationGid>
            - <Gid>
                <DomainName>GUEST</DomainName>
              - <Xid>
                  <xsl:value-of select="location_id"/>
                </Xid>
              </Gid>
            </LocationGid>
          - <Address>
            - <CountryCode3Gid>
              - <Xid>
                  <xsl:value-of select="country_code"/>
                </Xid>
              </CountryCode3Gid>
            </Address>
          </Location>
        </GLogXMLElement>
      </xsl:for-each>
    </TransmissionBody>
</Transmission>
```

One Slide Intro to Python

- Invest a few hours reviewing the python tutorial at http://docs.python.org/2/tutorial/index.html
- Quote from that web page:

> Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Python Dictionary Structure

- Python dictionary is an associative array
- Extremely useful
- Example of how to create and use dictionary:

```
C:\Users\erosenbloom>python
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 3
32
Type "help", "copyright", "credits" or "license" for more i
>>> dict = {}
>>> dict["location_id"] = "A0001"
>>> dict["country_id"] = "USA"
>>> print dict
{'country_id': 'USA', 'location_id': 'A0001'}
>>> print dict["location_id"]
A0001
>>> print dict["country_id"]
USA
>>> dict = {'location_id' : 'A0001', 'country_id' : 'USA'}
>>> print dict
{'country_id': 'USA', 'location_id': 'A0001'}
>>>
```

Python List Structure

- Python list is a sequential array indexed by integer
- Also extremely useful
- Example of how to create and use list

```
>>> list = []
>>> list.append('apple')
>>> list.append('pear')
>>> print list
['apple', 'pear']
>>> list = ['apple', 'pear']
>>> print list
['apple', 'pear']
>>> print list[0]
apple
>>>
```

Python Code Files

- IntegrationServer.py
  - Main code file which implements our integration server
- CommonTools.py
  - Various utility methods such as httpPush()
- myxslt.py
  - Implements the xslt_transform() method
  - Allows for command-line testing of your xsl during development

IntegrationServer.py Classes

- GlobalConfig
  - Various configuration parameters
- FlatFileProcessor
  - Base class implementing methods to process flat files into GLog XML
- LocationProcessor
  - Location flat file processor which extends FlatFileProcessor
- IntegrationServer
  - Main class implementing the mainLoop and processInboundDir methods

IntegrationServer.py __main__

```
if __name__ == "__main__":
    myServer = IntegrationServer()
    myServer.mainLoop()
```

IntegrationServer Class

- IntegrationServer Class includes the following methods
  - mainLoop
    - Infinite loop calling processInboundDir() and then sleeps some
  - processInboundDir
    - Process known files in the inbound directory in FIFO sequence, where FIFO sequence is the responsibility of the upstream system via file naming convention. Known files have a known file name extension. Unknown files are ignored. After processing a known file, move it to the “done directory”.
  - calcSleepTime
    - Determine how much time to sleep at the end of each iteration of the main loop as a function of (a) the GlobalConfig.pollingInterval and (b) the duration of the most recent loop iteration. If the duration of the most recent loop iteration exceeds the pollingInterval, then we don’t sleep at all

IntegrationServer.mainLoop

```
def mainLoop (self):
    while 1:
        time1 = time.clock()
        self.processInboundDir()
        time.sleep(self.calcSleepTime(time1))
```

IntegrationServer.processInboundDir

```
def processInboundDir (self):
    """
    Process any files that the upstream
    system transferred into the inbound
    directory
    """
    fileList = os.listdir(GlobalConfig.inboundDir)
    for basename in fileList:
        if basename.endswith(".loc"):
            locationProcessor = LocationProcessor()
            locationProcessor.process(basename)
```

IntegrationServer.calcSleepTime

```
def calcSleepTime (self, time1):
    """
    Calculate how much time to sleep at the
    end of the main loop.
    time1 is the clock time at the start
    of the main loop.
    """
    time2 = time.clock()
    execTime = int(time2 - time1)
    sleepTime = GlobalConfig.pollingInterval - execTime
    if sleepTime < 0:
        sleepTime = 0
    log("calcSleepTime", {"sleepTime" : sleepTime})
    return sleepTime
```

LocationProcessor Class

- LocationProcessor class extends the more generalized FlatFileProcessor class, providing concrete implementation for the pure virtual methods defined by the FlatFileProcessor Class
- Pure virtual methods are
  - parseLine()
    - Parse a line from the location flat file, returning a dictionary
  - getXSL()
    - Return the full path file name of the XSL file responsible for converting the intermediateXML representation of the flat file to GLogXML

LocationProcessor Class

```
class LocationProcessor(FlatFileProcessor):

    """
    The LocationProcessor class implements methods specific to the
    processing of our Location flat file
    """

    def parseLine (self, line):
        """
        Parse a line from the location fixed format file,
        returning a dictionary with attributes
        "location_id" and "country_id"
        """
        return {"location_id":line[0:10].strip(),
                "country_code":line[10:13].strip()}

    def getXSL (self):
        return GlobalConfig.XSLdir + "/" + "location.xsl"
```

FlatFileProcessor Class

- FlatFileProcessor Class provides the following methods
  - parseLine: Pure virtual described previously
  - getXSL: Pure virtual described previously
  - parseFile: Parse flat file into a list-of-dictionary, using the parseLine() method
  - createIntermediateXMLfile: Given the list-of-dictionary representation of the flat file, create a corresponding intermediate XML file
  - createGLogXMLfile: Convert a given intermediate XML file into a GLog XML file, using the XSL transformation returned from the getXSL method
  - pushGLogXMLfile: Given a GLog XML file, push it to the remote GLog WMServlet
  - process: process a given flat file and move it to the done directory

FlatFileProcessor.parseFile

```
def parseFile (self, fileName):
    """
    Parse file into list of dictionaries
    """
    list = []
    fh = open(fileName)
    while 1:
        line = fh.readline()
        if not line: break
        list.append(self.parseLine(line))
    fh.close()
    return list
```

FlatFileProcessor.createIntermediateXMLfile

```
def createIntermediateXMLfile (self, list):
    """
    Given list of dict, create intermediate
    XML file
    """
    doc = {}
    doc["location"] = list

    xmlString = xmltodict.unparse({"doc" : doc})

    tempFileName = \
    CommonTools.makeTempFile(".xml", GlobalConfig.tempDir)

    CommonTools.writeStringToFile(xmlString, tempFileName)
    return tempFileName
```

FlatFileProcessor.createGLogXMLfile

```
def createGLogXMLfile (self, intermediateXMLfileName):
    """
    Create a GLog XML file from an intermediate XML
    using an XSL transformation
    """
    glogXMLfileName = \
    CommonTools.makeTempFile(".xml", GlobalConfig.tempDir)

    myxslt.xslt_transform (intermediateXMLfileName,
                           self.getXSL(),
                           glogXMLfileName)
    return glogXMLfileName
```

FlatFileProcessor.pushGLogXMLfile

```
def pushGLogXMLfile (self, glogXMLfileName):
    """
    Given a GLogXML file, push it to the GLog WMServlet
    """
    reply = CommonTools.httpPush (glogXMLfileName,
                                  GlobalConfig.WMServletURL)
    return reply
```

FlatFileProcessor.process

```
def process (self, basename):
    """
    Process a flat file.
    basename: base name of file to be processed
    """
    fullPathName = GlobalConfig.inboundDir + "/" + basename
    list = self.parseFile(fullPathName)
    intermediateXMLfile = self.createIntermediateXMLfile(list)
    glogXMLfile = self.createGLogXMLfile(intermediateXMLfile)
    reply = self.pushGLogXMLfile(glogXMLfile)
    log("FlatFileProcessor.process", {"reply from GLog " : reply})
    CommonTools.moveFile(fullPathName, GlobalConfig.inboundDoneDir, "")
```

GlobalConfig class

```
class GlobalConfig:
    """
    Global configuration parameters
    """
    tempDir = "c:/temp/temp"
    WMServletURL = "http://hostname/GC3/glog.integration.servlet.WMServlet"
    pollingInterval = 3
    inboundDir = "c:/temp/inboundDir"
    inboundDoneDir = "c:/temp/inboundDir/done"
    XSLdir = "c:/temp/XSLdir"
```

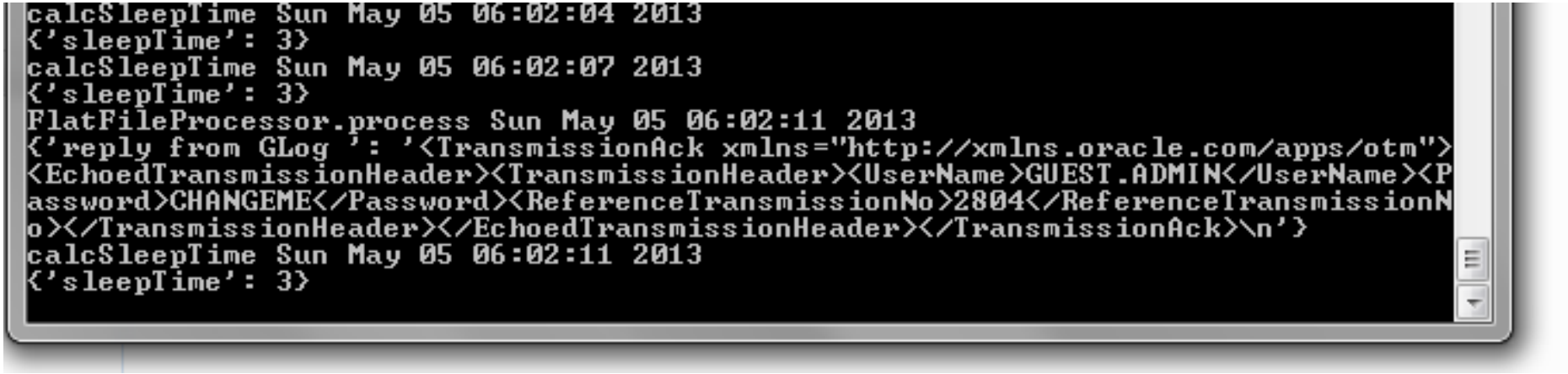
Testing the Integration Server

- Run IntegrationServer.py in a command session
- Move a location flat file into the inbound directory
- Observe the log
- View the intermediate XML file created in the temp directory
- View the GLog XML file created in the temp directory
- View the XML Transmission within OTM
- View the created location within OTM

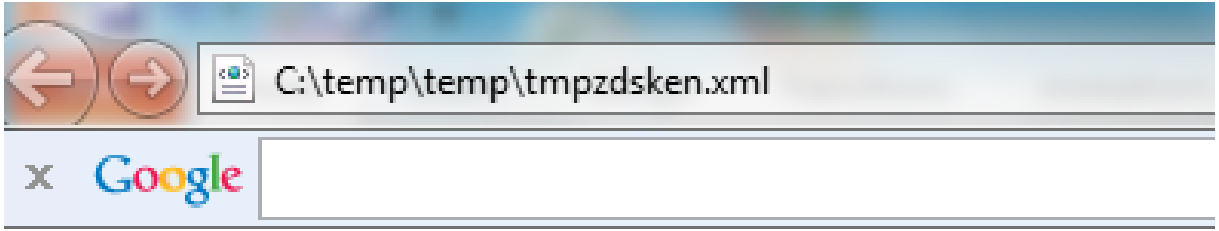
Run IntegrationServer.py

```
C:\inspirage\project\sig2013\python>python IntegrationServer.py
calcSleepTime Sun May 05 05:51:39 2013
{'sleepTime': 3}
calcSleepTime Sun May 05 05:51:42 2013
{'sleepTime': 3}
calcSleepTime Sun May 05 05:51:45 2013
{'sleepTime': 3}
calcSleepTime Sun May 05 05:51:48 2013
{'sleepTime': 3}
calcSleepTime Sun May 05 05:51:51 2013
{'sleepTime': 3}
calcSleepTime Sun May 05 05:51:54 2013
{'sleepTime': 3}
calcSleepTime Sun May 05 05:51:57 2013
{'sleepTime': 3}
calcSleepTime Sun May 05 05:52:00 2013
{'sleepTime': 3}
calcSleepTime Sun May 05 05:52:03 2013
{'sleepTime': 3}
calcSleepTime Sun May 05 05:52:06 2013
{'sleepTime': 3}
```

Move location flat file to inbound directory

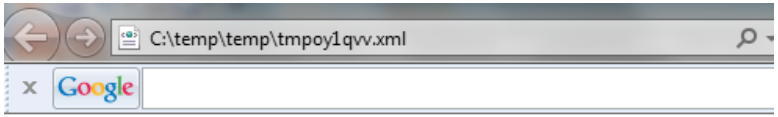


View Intermediate XML file in temp dir




```
<?xml version="1.0" encoding="UTF-8"?>
- <doc>
  - <location>
      <location_id>A0001</location_id>
      <country_code>USA</country_code>
    </location>
</doc>
```

View GLog XML file in temp dir




```
<?xml version="1.0"?>
- <Transmission>
  - <TransmissionHeader>
      <UserName>GUEST.ADMIN</UserName>
      <Password>CHANGEME</Password>
    </TransmissionHeader>
  - <TransmissionBody>
    - <GLogXMLElement>
      - <Location>
          <TransactionCode>IU</TransactionCode>
        - <LocationGid>
          - <Gid>
              <DomainName>GUEST</DomainName>
              <Xid>A0001</Xid>
            </Gid>
          </LocationGid>
        - <Address>
          - <CountryCode3Gid>
              <Xid>USA</Xid>
            </CountryCode3Gid>
          </Address>
        </Location>
      </GLogXMLElement>
    </TransmissionBody>
</Transmission>
```

View Transmission in OTM

Transmissions **Total Found:** 1 | View | Actions | Replace

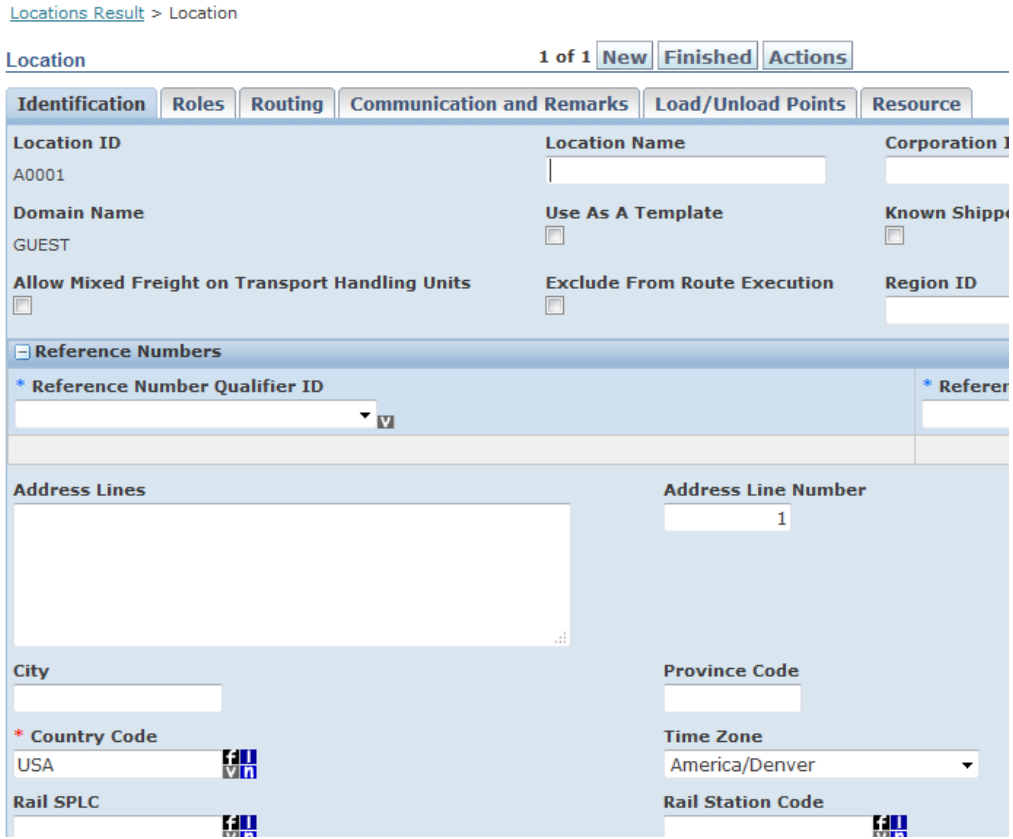
Pages 1 | **Selected** Page: 0 Total: 0

ID	Inbound	Sender Transmissio...	Status	Insert Time
2804			PROCESSED	2013-05-05 06:02 Ame...

Transaction 4903

1. Location
   1. TransactionCode = IU
   2. LocationGid
      1. Gid
         1. DomainName = GUEST
         2. Xid = A0001
   3. Address
      1. CountryCode3Gid
         1. Xid = USA

View Created Location in OTM



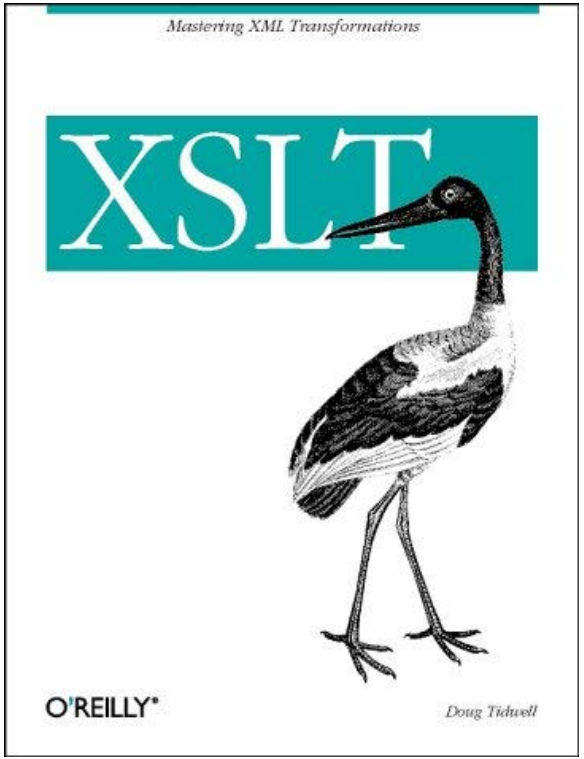
Testing Your XSL Changes Using myxslt.py

```
C:\inspirage\project\sig2013\python>dir
 Volume in drive C is Windows7_OS
 Volume Serial Number is 8CDC-AA51

 Directory of C:\inspirage\project\sig2013\python

05/22/2013  03:27 AM    <DIR>          .
05/22/2013  03:27 AM    <DIR>          ..
01/17/2013  10:29 AM             9,336 CommonTools.py
05/22/2013  03:11 AM            12,680 CommonTools.pyc
05/05/2013  08:01 AM             5,042 IntegrationServer.py
05/01/2013  06:48 AM             2,882 myxslt.py
05/22/2013  03:27 AM               434 out.xml
05/02/2013  04:06 AM               108 source.xml
05/04/2013  04:22 AM               675 trivial.xsl
05/22/2013  03:26 AM                81 xsl.bat
               8 File(s)         31,238 bytes
               2 Dir(s)  322,559,111,168 bytes free

C:\inspirage\project\sig2013\python>xsl

C:\inspirage\project\sig2013\python>python myxslt.py -sourceXML source.xml -tran
sformXSLT trivial.xsl -outXML out.xml

C:\inspirage\project\sig2013\python>
```

Windows Installation Procedure

- Download and Install Python 2.7 from www.python.org
- xmltodict
  - Download zip from https://github.com/martinblech/xmltodict
  - Extract zip to temp directory
  - Run “python setup.py install”
- Libxslt and libxml2
  - Download Windows installer from http://users.skynet.be/sbi/libxml-python/
  - Use libxml2-python-2.7.7.win32-py2.7.exe if you are using python 2.7
- Make required directories under c:/temp including c:/temp/inboundDir and c:/temp/XSLdir (these must match those defined in GlobalConfig class)
- Copy trivial-location-transformation-example.xsl to c:/temp/XSLdir/location.xsl

Recommended Book on XSL



Next steps for you

- Enhance the “trivial” location transformation example to handle the real-world with more than 2 attributes
- Add other interfaces such as item and order release
- Implement outbound interfaces such as planned shipment

Questions?

- Q&A