XML Processing

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

# What is XML?

The Extensible Markup Language (XML) is a markup language much like HTML or SGML. This is recommended by the World Wide Web Consortium and available as an open standard.

XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

# XML Parser Architectures and APIs

## Simple API for XML (SAX)

Here, you register call backs for events of interest and then let the parser proceed through the document. This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

## Document Object Model (DOM) API

This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical (tree-based) form to represent all the features of an XML document.

<collection shelf="New Arrivals">

<movie title="Enemy Behind">

<type>War, Thriller</type>

<format>DVD</format>

<year>2003</year>

<rating>PG</rating>

<stars>10</stars>

<description>Talk about a US-Japan war</description>

</movie>

<movie title="Transformers">

<type>Anime, Science Fiction</type>

<format>DVD</format>

<year>1989</year>

<rating>R</rating>

<stars>8</stars>

<description>A schientific fiction</description>

</movie>

<movie title="Trigun">

<type>Anime, Action</type>

<format>DVD</format>

<episodes>4</episodes>

<rating>PG</rating>

<stars>10</stars>

<description>Vash the Stampede!</description>

</movie>

<movie title="Ishtar">

<type>Comedy</type>

<format>VHS</format>

<rating>PG</rating>

<stars>2</stars>

<description>Viewable boredom</description>

</movie>

</collection>

# Parsing XML with SAX APIs

* SAX is a standard interface for event-driven XML parsing.
* Parsing XML with SAX generally requires you to create your own ContentHandler by subclassing xml.sax.ContentHandler.
* Your ContentHandler handles the particular tags and attributes of your flavor(s) of XML. A ContentHandler object provides methods to handle various parsing events. Its owning parser calls ContentHandler methods as it parses the XML file.
* The methods startDocument and endDocument are called at the start and the end of the XML file. The method characters(text) is passed character data of the XML file via the parameter text.
* The ContentHandler is called at the start and end of each element. If the parser is not in namespace mode, the methods startElement(tag, attributes) and endElement(tag) are called; otherwise, the corresponding methods startElementNS and endElementNS are called. Here, tag is the element tag, and attributes is an Attributes object.

# The make\_parser Method

**xml.sax.make\_parser( [parser\_list] )**

Here is the detail of the parameters −

* parser\_list − The optional argument consisting of a list of parsers to use which must all implement the make\_parser method.

# The parseString Method

**xml.sax.parseString(xmlstring, contenthandler[, errorhandler])**

Here is the detail of the parameters −

* xmlstring − This is the name of the XML string to read from.
* contenthandler − This must be a ContentHandler object.
* errorhandler − If specified, errorhandler must be a SAX ErrorHandler object.

import xml.sax

import os

class MovieHandler( xml.sax.ContentHandler ):

def \_\_init\_\_(self):

self.CurrentData = ""

self.type = ""

self.format = ""

self.year = ""

self.rating = ""

self.stars = ""

self.description = ""

def startElement(self, tag, attributes):

self.CurrentData = tag

if tag == "movie":

print ("\*\*\*\*\*Movie\*\*\*\*\*")

title = attributes["title"]

print ("Title:", title)

def endElement(self, tag):

if self.CurrentData == "type":

print ("Type:", self.type)

elif self.CurrentData == "format":

print ("Format:", self.format)

elif self.CurrentData == "year":

print ("Year:", self.year)

elif (self.CurrentData == "rating"):

print ("Rating:", self.rating)

elif self.CurrentData == "stars":

print ("Stars:", self.stars)

elif self.CurrentData == "description":

print ("Description:", self.description)

self.CurrentData = ""

def characters(self, content):

if self.CurrentData == "type":

self.type = content

elif self.CurrentData == "format":

self.format = content

elif self.CurrentData == "year":

self.year = content

elif self.CurrentData == "rating":

self.rating = content

elif self.CurrentData == "stars":

self.stars = content

elif self.CurrentData == "description":

self.description = content

if ( \_\_name\_\_ == "\_\_main\_\_"):

# create an XMLReader

parser = xml.sax.make\_parser()

# turn off namepsaces

parser.setFeature(xml.sax.handler.feature\_namespaces, 0)

# override the default ContextHandler

Handler = MovieHandler()

parser.setContentHandler( Handler )

parser.parse(

os.path.join(r"D:\Workspace\PythonWorkspace\xmlprocess\com", "movies.xml")

)

Output:

\*\*\*\*\*Movie\*\*\*\*\*

Title: Enemy Behind

Type: War, Thriller

Format: DVD

Year: 2003

Rating: PG

Stars: 10

Description: Talk about a US-Japan war

\*\*\*\*\*Movie\*\*\*\*\*

Title: Transformers

Type: Anime, Science Fiction

Format: DVD

Year: 1989

Rating: R

Stars: 8

Description: A schientific fiction

\*\*\*\*\*Movie\*\*\*\*\*

Title: Trigun

Type: Anime, Action

Format: DVD

Rating: PG

Stars: 10

Description: Vash the Stampede!

\*\*\*\*\*Movie\*\*\*\*\*

Title: Ishtar

Type: Comedy

Format: VHS

Rating: PG

Stars: 2

Description: Viewable boredom

# Parsing XML with DOM APIs

* The Document Object Model ("DOM") is a cross-language API from the World Wide Web Consortium (W3C) for accessing and modifying XML documents.
* The DOM is extremely useful for random-access applications. SAX only allows you a view of one bit of the document at a time. If you are looking at one SAX element, you have no access to another.
* Here is the easiest way to quickly load an XML document and to create a minidom object using the xml.dom module. The minidom object provides a simple parser method that quickly creates a DOM tree from the XML file.
* The sample phrase calls the parse( file [,parser] ) function of the minidom object to parse the XML file designated by file into a DOM tree object.

from xml.dom.minidom import parse

import xml.dom.minidom

import os

# Open XML document using minidom parser

DOMTree = xml.dom.minidom.parse(os.path.join(r"D:\Workspace\PythonWorkspace\xmlprocess\com", "movies.xml"))

collection = DOMTree.documentElement

if collection.hasAttribute("shelf"):

print ("Root element : %s" % collection.getAttribute("shelf"))

# Get all the movies in the collection

movies = collection.getElementsByTagName("movie")

# Print detail of each movie.

for movie in movies:

print ("\*\*\*\*\*Movie\*\*\*\*\*")

if movie.hasAttribute("title"):

print ("Title: %s" % movie.getAttribute("title"))

type = movie.getElementsByTagName('type')[0]

print ("Type: %s" % type.childNodes[0].data)

format = movie.getElementsByTagName('format')[0]

print ("Format: %s" % format.childNodes[0].data)

rating = movie.getElementsByTagName('rating')[0]

print ("Rating: %s" % rating.childNodes[0].data)

description = movie.getElementsByTagName('description')[0]

print ("Description: %s" % description.childNodes[0].data)