XML Handling

# ΓΛΩΣΣΙΚΉ ΤΕΧΝΟΛΟΓΊΑ

# XML related standards

- SAX (Simple API for XML)
  - event-driven interface
  - απλό
  - γρήγορο
  - διάτρεξη XML εγγράφου
- DOM (Document Object Model)
  - tree-based representation
  - αργό
  - μνημοβόρο
  - Αλλαγή δομής XML εγγράφου

# Python XML modules

- **xml.dom** - The Document Object Model API
- **xml.dom.minidom** - Lightweight DOM implementation
- **xml.dom.pulldom** - Support for building partial DOM trees
- **xml.sax** - Support for SAX2 parsers
- **xml.sax.handler** - Base classes for SAX handlers
- **xml.sax.saxutils** - SAX Utilities
- **xml.sax.xmlreader** - Interface for XML

# Χρήση SAX parser

```
# Define your specialized handler classes
from xml.sax import ContentHandler
from xml.sax import make_parser

# Create an XML parser
parser = make_parser()

# Create an instance of the handler classes
dh = docHandler()

# Tell the parser to use your handler instance
parser.setContentHandler(dh)

# Parse the file; your handler's methods will get called
parser.parse(document.xml)
```

# Κατασκευή SAX parser

```
class docHandler(ContentHandler):
        def __init__(self):
                self.inPage = 0
                self.inTitle = 0
                self.inText = 0

        def startElement(self, name, attrs):
                if name == 'page':
                        self.inPage = 1
                        # Get attribute (page's id).
                        self.page_id = attrs.get('id', None)
                if name == 'title':
                        self.inTitle = 1
                if name == 'text':
                        self.inText = 1

        def endElement(self, name):
                if name == 'page':
                        self.inPage = 0
                if name == 'text':
                        self.inText = 0
                if name == 'title':
                        self.inTitle = 0

        def characters(self, ch):
                if self.inText:
                        dest = open(self.new_file,"a")
                        dest.write(ch.encode("utf-8"))
                        dest.close()
```

# Κατασκευή DOM δέντρου

```
from xml.dom.ext.reader import Sax2

# create Reader object
reader = Sax2.Reader()

# parse the document
doc = reader.fromStream(document.xml)
```

# DOM attributes I

- nodeType: σταθερά που υποδηλώνει τον τύπο του κόμβου όπως ELEMENT_NODE, TEXT_NODE κτλ
- nodeName: το όνομα του κόμβου. ΠΡΟΣΟΧΗ διότι εξαρτάται από τον τύπο του κόμβου!
- nodeValue: η τιμή του κόμβου. ΠΡΟΣΟΧΗ διότι εξαρτάται από τον τύπο του κόμβου!

# DOM attributes II

- parentNode: ο πατέρας του κόμβου ή None αν πρόκειται για την ρίζα
- childNodes: μία λίστα με τα παιδιά του κόμβου (άδεια όταν δεν έχει)
- firstChild: το πρώτο παιδί του κόμβου (None αν δεν έχει)
- lastChild: το τελευταίο παιδί του κόμβου (None αν δεν έχει)

# DOM methods I

- appendChild(newChild): προσθέτει το newChild στον κόμβο, στο τέλος της λίστας των παιδιών του
- removeChild(oldChild): διαγράφει το oldChild από τον κόμβο
- replaceChild(newChild, oldChild): αντικαθιστά το oldChild με το newChild

# DOM methods II

- insertBefore(newChild, refChild): προσθέτει το newChild στον κόμβο αλλά πριν από το refChild και όχι στο τέλος
- HasChildNodes(): επιστρέφει True αν ο κόμβος έχει παιδιά
- cloneNode(deep): επιστρέφει αντίγραφο του κόμβου. Το deep ειναι boolean και υποδεικνύει την αντιγραφή των παιδιών του κόμβου

# Δημιουργία κόμβου

```
# The base of the entire tree is the Document node.
# Its documentElement attribute contains the Element
# node for the root element.

# building a DOM tree from scratch...
# create the root element
new = document.createElement('chapter')

# set attribute to new node
new.setAttribute('number', '5')

# append new node to root node
document.documentElement.appendChild(new)
```

# C# - Xml Handling

- Namespace: System.Xml
- Βασικές κλάσεις:
  - XmlDocument: Αντιπροσωπεύει ένα xml document. Παρέχει δυνατότητα προσθήκης, επεξεργασίας, διαγραφής κόμβων των xml δεδομένων.
  - XmlNode: Βασική κλάση που αντιπροσωπεύει έναν XML κόμβο. Από αυτή κληρονομούν οι ιδιαίτεροι τύποι κόμβων, όπως:
    - XmlElement (μέσω του XmlLinkedNode): Ο συνήθης τύπος κόμβου (πχ <item></item>)
    - XmlText: Το κείμενο μέσα στον κόμβο (πχ <item>txt</item>)
    - XmlAttribute: Ένα γνώρισμα κόμβου (πχ <item id="id1">)
  - XmlReader: Παρέχει streaming τύπου πρόσβαση στα XML δεδομένα (γρήγορο, forward only).
  - XmlWriter: Κλάση που παρέχει streaming τύπου δημιουργία XML δεδομένων (γρήγορο, forward only).

# C# - Παράδειγμα

- Έστω τα XML δεδομένα:

```
<?xml version="1.0" encoding="windows-1253"?>
<root id="r1">
 <child id="c1">
  <item id="i1">1</item>
  <item id="i2">2</item>
 </child>
 <child id="c2">
  <item id="i3">3</item>
  <item id="i4">4</item>
 </child>
</root>
```

- Χρησιμοποιώντας XmlDocument μπορούμε:
  - Να τα φορτώσουμε και να τα επεξεργαστούμε
  - Να τα δημιουργήσουμε
- Μπορούμε να τα διαβάσουμε γρήγορα με XmlReader
- Μπορούμε να τα δημιουργήσουμε γρήγορα με XmlWriter

# C#- XmlDocument Traversal

```
private static void TraverseXmlDocument()
{
    XmlDocument xmldoc = new XmlDocument();

    xmldoc.Load(@"C:\test.xml");
    TraverseSubtree(xmldoc.DocumentElement);

    /* prints
        root    r1
        child   c1
        item    i1
        1
        item    i2
        2
        child   c2
        item    i3
        3
        item    i4
        4
     * */
}

private static void TraverseSubtree(XmlNode parent)
{
    if (parent.NodeType == XmlNodeType.Element)
    {
        Console.WriteLine(parent.Name + "\t" + parent.Attributes["id"].Value);
    }
    else if (parent.NodeType == XmlNodeType.Text)
    {
        Console.WriteLine(parent.Value);
    }

    foreach (XmlNode node in parent.ChildNodes)
        TraverseSubtree(node);
}
```

# C# - XmlDocument Traversal

- xmldoc.Load(string path): διαβάζει τα xml δεδομένα από αρχείο και τα φορτώνει στο xml document.
- xmldoc.DocumentElement: ο κορυφαίος κόμβος του XmlDocument. Πρέπει να υπάρχει ακριβώς ένας κορυφαίος κόμβος, αλλιώς δεν είναι σωστό xml έγγραφο.
- node.ChildNodes: για έναν XmlNode επιστρέφει τη λίστα των παιδιών του. Προσοχή! Το κείμενο είναι επίσης παιδί, τύπου XmlText.
- node.Name: το όνομα του κόμβου
- node.NodeType: ο τύπος του κόμβου (element, text κλπ)
- node.Attributes: το dictionary με τα attributes του κόμβου
- node.Value: η τιμή του κόμβου, αν ο τύπος το επιτρέπει. Πχ. Για έναν XmlText κόμβο η τιμή είναι το κείμενο.

# C# - XmlDocument Creation

```
private static void CreateXmlDocument()
{
    XmlElement parent;
    XmlElement curr_element;
    XmlElement sub_element;
    XmlText text_element;
    XmlAttribute attr;
    XmlDocument xmldoc = new XmlDocument();
    int curr_item_id;

    //element creation
    parent = xmldoc.CreateElement("root");
    //attribute creation
    attr = xmldoc.CreateAttribute("id");
    attr.Value = "r1";
    parent.Attributes.Append(attr);
    //attach child to parent
    xmldoc.AppendChild(parent);

    //create 1st level
    curr_item_id = 1;
    for (int i = 1; i <= 2; i++)
    {
        curr_element = xmldoc.CreateElement("child");
        attr = xmldoc.CreateAttribute("id");
        attr.Value = "c"+ i;
        curr_element.Attributes.Append(attr);
        parent.AppendChild(curr_element);
        //create 2nd level
        for (int j = 1; j <= 2; j++)
        {
            sub_element = xmldoc.CreateElement("item");
            attr = xmldoc.CreateAttribute("id");
            attr.Value = "i" + curr_item_id;
            sub_element.Attributes.Append(attr);
            text_element = xmldoc.CreateTextNode(curr_item_id.ToString());
            sub_element.AppendChild(text_element);
            curr_item_id++;
            curr_element.AppendChild(sub_element);
        }
    }

    StreamWriter sw = new StreamWriter(@"C:\test.xml");
    sw.WriteLine(xmldoc.InnerXml);
    sw.Close();
}
```

# C# - XmlDocument Creation

- Δημιουργία στοιχείων
  - Οι αντίστοιχες συναρτήσεις καλούνται μέσω του XmlDocument αντικειμένου και επιστρέφουν το ανάλογο αντικείμενο (πχ XmlElement η CreateElement, XmlAttribute η CreateAttribute κλπ)
  - Μέσω του αντικειμένου που επιστρέφεται τίθενται οι επιθυμητές τιμές στον νέο κόμβο.
- Προσθήκη κόμβων
  - Αφού δημιουργηθεί ο κόμβος πρέπει να προστεθεί ως παιδί σε κάποιον ήδη υπάρχοντα με την κατάλληλη append.
  - Μόνο το DocumentElement γίνεται append στο XmlDocument!

# C# - XmlReader

```
private static void ReadWithXmlReader()
{
    XmlReader xrdr = XmlReader.Create(@"C:\test.xml");
    while (xrdr.Read())
    {
        if (xrdr.NodeType == XmlNodeType.Element)
        {
            Console.WriteLine(xrdr.Name + "\t" + xrdr.GetAttribute("id"));
        }
        else if (xrdr.NodeType == XmlNodeType.Text)
        {
            Console.WriteLine(xrdr.Value);
        }
    }
    xrdr.Close();
    /* prints
       root    r1
       child   c1
       item    i1
       1
       item    i2
       2
       child   c2
       item    i3
       3
       item    i4
       4
    * */
}
```

# C# - XmlReader

- Σειριακή ανάγνωση ανά στοιχείο.
- read(): πραγματοποιεί την κατανάλωση του επόμενου στοιχείου. Υπάρχουν παραλλαγές πχ:
  - ReadToFollowing(string name): διαβάζει μέχρι το στοιχείο με το καθορισμένο όνομα
  - ReadSubtree(): καταναλώνει όλο το υπόδεντρο του τρέχοντα κόμβου και επιστρέφει νέο XmlReader του υπόδεντρου.
- xmlrdr.NodeType: ο τύπος του κόμβου στον οποίο βρίσκεται το stream.
- xmlrdr.Name: το όνομα του κόμβου στον οποίο βρίσκεται το stream.
- xmlrdr.GetAttribute (string name): επιστρέφει την τιμή του attribute με το συγκεκριμένο όνομα.
- xmlrdr.Value: αν ο τύπος του κόμβου επιτρέπει value, τότε επιστρέφει την τιμή.
- Είναι stream, το κλείνουμε!!!

# C# - XmlWriter

```
private static void CreateWithXmlWriter()
{
    XmlWriterSettings xsett = new XmlWriterSettings();
    xsett.Indent = true;
    xsett.Encoding = Encoding.Default;

    XmlWriter xw = XmlWriter.Create("C:\\test.xml", xsett);
    int curr_item_id = 1;

    xw.WriteStartElement("root");
    xw.WriteAttributeString("id", "r1");

    for (int i = 1; i <= 2; i++)
    {
        xw.WriteStartElement("child");
        xw.WriteAttributeString("id", "c" + i);

        for (int j = 1; j <= 2; j++)
        {
            xw.WriteStartElement("item");
            xw.WriteAttributeString("id", "i" + curr_item_id);
            xw.WriteValue(curr_item_id.ToString());
            xw.WriteEndElement();
            curr_item_id++;
        }

        xw.WriteEndElement();
    }

    xw.WriteEndElement();
    xw.Close();
}
```

# C# - XmlWriter

- Σειριακή εγγραφή ανά στοιχείο
- WriteStartElement(string name): γράφει το αρχικό tag ενός στοιχείου (πχ <item>)
- WriteEndElement(): γράφει το τελικό tag του στοιχείου του οποίου η WriteStartElement κλήθηκε τελευταία. (πχ </item>)
- Οτιδήποτε ανήκει σε αυτό το στοιχείο πρέπει να εγγραφεί ανάμεσα σε αυτές τις δύο εντολές. (θυγατρικοί κόμβοι, value, attributes)
- Προσοχή: ό,τι ανοίγει πρέπει να κλείσει με τη σωστή σειρά. Οι σχέσεις μεταξύ των κόμβων δεν καθορίζονται ρητά, εννοούνται από τη σειρά εγγραφής.
- XmlWriterSettings: κλάση που καθορίζει τη μορφή που θέλουμε να έχουν τα δεδομένα εξόδου. Πχ για να είναι με σωστά tabs, θέτουμε το Indent = true και αν θέλουμε συγκεκριμένο Encoding το ορίζουμε εδώ.
- Είναι stream, το κλείνουμε!!!

# Τι χρησιμοποιούμε?

- Ανάλογα με το πρόβλημα:
  - Τα streams είναι πολύ πιο γρήγορα.
  - Στις περισσότερες περιπτώσεις αποφεύγεται η αναδρομική κλήση.
  - Όμως:
    - Είναι μόνο μπροστά ανάγνωσης/εγγραφής.
    - Δεν προσφέρουν δυνατότητα επεξεργασίας δεδομένων που ήδη υπάρχουν.
    - Ο κώδικας καταλήγει να είναι λιγότερο κατανοητός από το XmlDocument.
    - Είναι εύκολο να συμβούν λάθη.
  - Γενικά, για απλά προβλήματα τα streams είναι η καλύτερη επιλογή, λόγω απόδοσης.