Γλωσσική Τεχνολογία

HTML/XML Processing – HTTP Services

Περιεχόμενα

- HTML processing
- XML processing
- HTTP services
  - URL parsing
  - URL opening
  - Content fetching
- Project

HTML Processing

- 3 built-in modules:
  - HTMLParser – Simple HTML parser
  - **sgmllib** – Simple SGML parser
  - htmllib – A parser for HTML documents
- Άλλες βιβλιοθήκες:
  - BeautifulSoup

HTMLParser

- HTMLParser methods:
  - HTMLParser.reset()
  - HTMLParser.feed(*data*)
  - HTMLParser.close()
  - HTMLParser.handle_starttag(*tag*, *attrs*)
  - HTMLParser.handle_endtag(*tag*)
  - HTMLParser.handle_data(*data*)
  - HTMLParser.handle_comment(*data*)

HTMLParser - Παράδειγμα

```
from HTMLParser import HTMLParser

class MyHTMLParser(HTMLParser):

    def handle_starttag(self, tag, attrs):
        print "Encountered the beginning of a %s tag" % tag

    def handle_endtag(self, tag):
        print "Encountered the end of a %s tag" % tag
```

sgmllib

- Αποτελεί την βάση για επεξεργασία οποιουδήποτε κειμένου σε SGML (Standard Generalized Mark-up Language) μορφή.
- Στην πραγματικότητα υλοποιήθηκε ως βάση για την επεξεργασία HTML και του module htmllib.
- Παρόμοιο σύνολο συναρτήσεων με αυτό του HTMLParser.
- Ευρεία χρήση αντί του HTMLParser.

BaseHTMLProcessor 1/2

```
from sgmllib import SGMLParser

class BaseHTMLProcessor(SGMLParser):
    def reset(self):
        # extend (called by SGMLParser.__init__)
        pass

    def unknown_starttag(self, tag, attrs):
        # called for each start tag
        # attrs is a list of (attr, value) tuples
        # e.g. for <pre class="screen">, tag="pre", attrs=[("class", "screen")]\
        pass

    def unknown_endtag(self, tag):
        # called for each end tag, e.g. for </pre>, tag will be "pre"
        pass

    def handle_charref(self, ref):
        # called for each character reference, e.g. for " ", ref will be "160"
        pass
```

BaseHTMLProcessor 2/2

```
def handle_entityref(self, ref):
    # called for each entity reference, e.g. for "&copy;", ref will be "copy"
    pass

def handle_data(self, text):
    # called for each block of plain text, i.e. outside of any tag and
    pass

def handle_comment(self, text):
    # called for each HTML comment, e.g. <!-- insert Javascript code here -->
    pass

def handle_pi(self, text):
    # called for each processing instruction, e.g. <?instruction>
    pass

def handle_decl(self, text):
    # called for the DOCTYPE, if present, e.g.
    # <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    #     "http://www.w3.org/TR/html4/loose.dtd">
    pass
```

SubHTMLProcessor

```
from BaseHTMLProcessor import BaseHTMLProcessor

class SubHTMLProcessor(BaseHTMLProcessor):

    def reset(self):
        # extend (called from __init__ in ancestor)
        pass

    def start_pre(self, attrs):
        # called for every <pre> tag in HTML source
        pass

    def end_pre(self):
        # called for every </pre> tag in HTML source
        pass

    def handle_data(self, text):
        # override
        # called for every block of text in HTML source
        pass
```

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

XML modules

- xml.parsers.expat - Fast XML parsing using Expat
- 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 parsers
- **xml.etree.ElementTree - The ElementTree XML API**

Other Libraries

- BeautifulSoup
- lxml
- PyXML
- libxml2
- and more…

The **ElementTree** XML API

- Element:
  - Αντικείμενο για την ιεραρχική αποθήκευση δεδομένων στην μνήμη. Κάθε element σχετίζεται με:
    - ένα tag (string), ονομάζεται element type και προσδιορίζει τον «τύπο» δεδομένων (στο XML) που διατηρεί το element
    - ένα πλήθος attributes, αποθηκευμένα σε dictionary
    - ένα string (το κείμενο μεταξύ των tags)
    - ένα προεραιτικό tail string και
    - ένα πλήθος από child elements, αποθηκευμένα σε sequence.

ElementTree methods

- xml.etree.ElementTree.fromstring(*text*):
  - Διαβάζει XML κείμενο από string (text).
- xml.etree.ElementTree.parse(*source*, *parser=None*):
  - Διαβάζει XML κείμενο από αρχείο (source).
- xml.etree.ElementTree.iselement(*element*):
  - Boolean συνάρτηση για τον έλεγχο επικύρωσης ενός element object.
- xml.etree.ElementTree.SubElement(*parent*, *tag*, *attrib*={}, ***extra*):
  - Δημιουργεί ένα αντικείμενο element και το προσθέτει στο υπάρχον element.

ElementTree Objects

- tag:
  - ένα string που προσδιορίζει τον τύπο δεδομένων (element type)
- text:
  - ένα string που διατηρεί το κείμενο μεταξύ start και close tags
- tail:
  - ένα string που διατηρεί οποιοδήποτε κείμενο που (αν) βρίσκεται μεταξύ του close tag και του επόμενου tag
- attrib:
  - ένα dictionary που διατηρεί τα attributes του element.
  - ΠΡΟΣΟΧΗ!! Μην χρησιμοποιείτε τις κλασσικές μεθόδους για dictionaries αλλά αυτές στην επόμενη διαφάνεια!!

dictionary-like methods

- clear():
  - «καθαρίζει» (reset) όλα τα objects ενός element
- get(*key*, *default*=*None*):
  - επιστρέφει την τιμή του attribute με όνομα key
- items():
  - επιστρέφει τα attributes ενός element σε μια ακολουθία από ζεύγη (name, value) σε τυχαία σειρά
- keys():
  - επιστρέφει με τυχαία σειρά τα ονόματα των attributes ενός element μέσα σε μία λίστα
- set(*key*, *value*):
  - θέτει το attribute με όνομα key ενός element στην τιμή value

methods on subelements

- append(*subelement*):
  - προσθέτει ένα subelement στην λίστα του element στο οποίο ανήκει
- find(*match*):
  - επιστρέφει το πρώτο subelement που ταιριάζει με το match. Το match μπορεί να είναι tag name ή path.
- findall(*match*):
  - επιστρέφει όλα τα matching subelements σε μία λίστα με την σειρά που εμφανίζονται
- insert(*index*, *element*):
  - εισάγει ένα subelement στο element στη θέση index
- remove(*subelement*):
  - απομακρύνει το subelement

ElementTree Objects

- getroot():
  - επιστρέφει το root element του δέντρου
- find(*match*):
  - ίδια με την *getroot().find(match)*
- findall(*match*):
  - ίδια με την *getroot().findall(match)*
- parse(*source*, *parser=None*):
  - φορτώνει ένα XML από το αρχείο source στην μνήμη σε μορφή element tree. Ως parser μπορείτε να αφήσετε τον default XMLParser.
- write(file, encoding="us-ascii", xml_declaration=None, method="xml"):
  - γράφει το element tree στο αρχείο file. Το default encoding είναι US-ASCII. Το xml_declaration ελέγχει την ύπαρξη δήλωσης του XML εγγράφου. Το None χρησιμοποιείται by default όταν το encoding δεν είναι US-ASCII. Το method είναι by default "xml", δεν το αλλάζετε.

ElementTree - Παράδειγμα

```
import elementtree.ElementTree as ET

# build a tree structure
root = ET.Element("html")

head = ET.SubElement(root, "head")

title = ET.SubElement(head, "title")
title.text = "Page Title"

body = ET.SubElement(root, "body")
body.set("bgcolor", "#ffffff")

body.text = "Hello, World!"

# wrap it in an ElementTree instance, and save as XML
tree = ET.ElementTree(root)
tree.write("page.xhtml")
```

HTTP Web Services

- 4 built-in modules:
  - **urlparse** - parse URLs into components
  - httplib - HTTP protocol client
  - urllib - open arbitrary resources by URL
  - **urllib2** - extensible library for opening URLs

urlparse 1/2

- urlparse.urlparse(*urlstring*[, *scheme*[, *allow_fragments*]]):
  - διάτρεξη του URL και «σπάσιμο» σε 6 κομμάτια. Το URL θεωρείται ότι είναι της μορφής: **scheme://netloc/path;parameters?query#fragment**
  - Μεγάλη ΠΡΟΣΟΧΗ στην χρήση της!! Μπορεί να έχετε απροσδόκητα αποτελέματα!!

```
from urlparse import urlparse
o = urlparse('http://www.cwi.nl:80/%7Eguido/Python.html')
print o   # doctest: +NORMALIZE_WHITESPACE
# ParseResult(scheme='http', netloc='www.cwi.nl:80', path='/%7Eguido/Python.html',
#             params='', query='', fragment='')
print o.scheme
# 'http'
print o.port
# 80
print o.geturl()
# 'http://www.cwi.nl:80/%7Eguido/Python.html'
```

urlparse 2/2

- urlparse.urlsplit(*urlstring*[, *scheme*[, *allow_fragments*]]):
  - παρόμοια με την urlparse() και χρησιμοποιείτε αυτήν αντί της urlparse(). Επιστρέφει μια 5-tuple: (addressing scheme, network location, path, query, fragment identifier). Δεν «σπάει» δηλαδή τις παραμέτρους για τις οποίες χρειάζεται μια επιπλέον συνάρτηση.
- ParseResult.geturl():
  - επιστρέφει το αρχικό URL. Μεγάλη ΠΡΟΣΟΧΗ στην χρήση της!! Το αποτέλεσμα που επιστρέφεται μπορεί να είναι διαφορετικό από το αρχικό URL.

urllib

- urllib.urlopen(*url*[, *data*[, *proxies*]]):
  - ο απλός αλλά ΟΧΙ προτεινόμενος τρόπος να κατεβάσετε περιεχόμενο από ένα URL

```
import urllib
data = urllib.urlopen('http://diveintomark.org/xml/atom.xml').read()
print data
# <?xml version="1.0" encoding="iso-8859-1"?>
# <feed version="0.3"
#    xmlns="http://purl.org/atom/ns#"
#    xmlns:dc="http://purl.org/dc/elements/1.1/"
#    xml:lang="en">
#    <title mode="escaped">dive into mark</title>
#    <link rel="alternate" type="text/html" href="http://diveintomark.org/"/>
#    <-- rest of feed omitted for brevity -->
```

urllib2

```
import urllib2
request = urllib2.Request('http://diveintomark.org/xml/atom.xml')
opener = urllib2.build_opener()
feeddata = opener.open(request).read()
```

- Διαδικασία 3 βημάτων:
  - Το πρώτο βήμα είναι να δημιουργήσετε μια αίτηση (Request object) του τι θέλετε να κατεβάσετε.
  - Το δεύτερο βήμα είναι να δημιουργήσετε έναν URL opener. Το πλεονέκτημα που αποκτάτε είναι ότι μπορείτε να ορίσετε τους δικούς σας handlers για το πως θα διαχειρίζεται κάθε ειδική περίπτωση, πχ: redirects.
  - Το τελευταίο βήμα είναι να εκτελέσετε τον opener έτσι ώστε να κατεβάσετε την πληροφορία που θέλετε.

Redirect Handlers

- *class* urllib2.HTTPRedirectHandler:
  - Ο default redirect handler για redirects. Μπορείτε να γράψετε τους δικούς σας για να διαχειριστείτε οποιοδήποτε redirect της σειράς 3xx (redirection).

```
class SmartRedirectHandler(urllib2.HTTPRedirectHandler):
    def http_error_301(self, req, fp, code, msg, headers):
        result = urllib2.HTTPRedirectHandler.http_error_301(
            self, req, fp, code, msg, headers)
        result.status = code
        return result
```

Error handlers

- *class* urllib2.HTTPDefaultErrorHandler:
  - Ο default error handler για HTTP error responses. Μπορείτε να γράψετε τους δικούς σας για να διαχειριστείτε οποιοδήποτε bad request της σειράς 4xx (client error) ή 5xx (server error).

```
class DefaultErrorHandler(urllib2.HTTPDefaultErrorHandler):
    def http_error_default(self, req, fp, code, msg, headers):
        result = urllib2.HTTPError(
            req.get_full_url(), code, msg, headers, fp)
        result.status = code
        return result
```

Σύνοψη

- Παρατηρείστε την διαχείριση των σελίδων σαν αρχεία (συναρτήσεις open(), read(), close()).
- Παρατηρείστε επίσης την χρήση handlers για τον χειρισμό ειδικών περιπτώσεων.
- Παρατηρείστε τέλος την χρήση timeout κατά την αίτηση ανοίγματος της σελίδας.

```
import urllib2
request = urllib2.Request('http://diveintomark.org/xml/atom.xml')
opener = urllib2.build_opener(SmartRedirectHandler(), DefaultErrorHandler())
datastream = opener.open(request, timeout=2)
datastream.read()
datastream.close()
```