Lightweight Language Saturday P3 発表資料
RSSビューアー

■Pytnon 2.2以上
　要JapaneseCodecs
　　http://www.python.jp/Zope/download/JapaneseCodecs
■使用モジュール
　xml.sax.handler　　XMLパーサのハンドラモジュール
　xml.sax　　SAXモジュール
■説明
　このプログラムは、http://slahdot.jp/slashdot.rdf を解析し、
　情報を整形してHTML出力をします。
　また、オリジナルをもとにしたRSS3形式のものも処理することができます。

rssget.py

```
#!/usr/bin/env python
"""Slashdot JapanのRSSを解析するモジュール

サイトにあるオリジナルのRSSやそれをもとにしたRSS3形式のものを解析する。
"""

from xml.sax.handler import ContentHandler
from xml.sax import make_parser
from cgi import escape
from StringIO import StringIO

class RSSHandler(ContentHandler):
    "RSSパーサハンドラ"

    # データのブロックとして認識する要素名
    BLOCK_ELEMENTS = ('channel', 'item')

    # ブロック内のデータとして認識する要素名
    ACCEPT_ELEMENTS = (
        'title', 'link', 'description', 'language', 'rights',
        'created', 'publisher', 'creator', 'subject',
        'updatePeriod', 'updateFrequency', 'updateBase')

    # エンコード時のコデック名
    ENCODING = 'utf-8'

    def __init__(self):
        self.blocks = []    #ブロックのリスト
        self.block = {}     #処理中のブロックデータがはいるバッファ
        self.nowtext = None #現在のテキストノード情報
        self.in_block = 0   #ブロック内かどうかのフラグ
        self.buf = StringIO()

    def characters(self, text):
        "文字列データが出現したときのハンドラ"

        if self.in_block:
            self.buf.write(escape(text.encode(self.ENCODING)).strip())

    def startElement(self, name, attrs):
        "開始タグが出現したときのハンドラ"

        if name in self.BLOCK_ELEMENTS:
            self.in_block = 1

    def endElement(self, name):
        "終了タグ出現したときのハンドラ"

        if self.in_block:
            name = name.encode(self.ENCODING).split(':', 1)[-1]
            if name in self.ACCEPT_ELEMENTS:
                self.block[name] = self.buf.getvalue()
                self.buf = StringIO()
        if name in self.BLOCK_ELEMENTS:
            self.in_block = 0
            if self.block:
                self.blocks.append(self.block)
                self.block = {}

    def getblocks(self):
        "処理済のブロックのリストを返す"

        return self.blocks

def parse_rss(fp):
    parser = make_parser()
    handler = RSSHandler()
    parser.setContentHandler(handler)
    parser.parse(fp)
    return handler.getblocks()


# RSS3形式のパース

class RSS3Parser:
    def parse_block(self, block):
        "ブロックを辞書オブジェクトのリストにする"

        dic = {}
        for key, val in [line.split(':', 1) for line in block.split('\n')]:
            dic[key] = val.strip()
        return dic

    def parse(self, fp):
        "ファイルオブジェクトを読みこんで辞書のリストに加工する"

        # 従来の表記方式
        # blocks = []
        # for block in open(RSSFILE).read().split('\n\n'):
        #     block = block.strip()
        #     if block: blocks.append(parse_block(block))

        # リスト内包表記
        return [self.parse_block(block)
                for block in fp.read().split('\n\n') if block.strip()]

def parse_rss3(fp):
    return RSS3Parser().parse(fp)
```

rssget.cgi

```
#!/usr/bin/env python
"""Slashdot JapanのRDFをHTMLに変換して表示するCGI

RSS解析モジュール(rssget.py)でRSSを解析し、表示する。
"""

from rssget import parse_rss
from urllib import urlopen

RSSFILE_URL = 'http://slashdot.jp/slashdot.rdf'

def render_html(title, value):
    if title == 'link':
        value = '<a href="%s">%s</a>' % (value, value)
    return '<tr><th>%s</th><td>%s</td></tr>' % (title, value)

# HTMLの出力

# ヘッダ部
print 'Content-Type: text/html; charset=utf-8'
print '''
<html>
<head>
<title>RSS feed</title>
</head>
<body>
'''

# サイト情報
blocks = parse_rss(urlopen(RSSFILE_URL))
header = blocks[0]
print '<table border="1">'
for key in header.keys():
    print render_html(key, header[key])
print '</table>'

print '<hr>'

# ニュースサマリ
bodies = blocks[1:]
for body in bodies:
    print '<table border="1">'
    for key in body.keys():
        print render_html(key, body[key])
    print '</table>'

# フッタ部
print '''
</table>
</body>
</html>
'''
```

Python Japan User's Group PyJUG Copyright (c) 2001-2002 Python Japan User Group

Lightweight Language Saturday P3 発表資料
Apacheログ解析

■Pytnon 2.2以上
■使用モジュール
time　　時間や時間の書式のフォーマットを処理するためのモジュール
■説明
このプログラムは、ApacheのCombinedアクセスログを解析し、IPアドレスごとの滞在時間と、全体の平均滞在時間を求めます

analyze.py

```
#!/usr/bin/python

from __future__ import generators
from time import strptime, strftime

LOGFILE = 'access_log_masked'

#
STILLTIME = 60 * 5

def parse_access_time(line):
    "It is parse to log line, so it returns timestamp and client address."

    tokens = line.split()
    return long(strftime(
        '%s', strptime(tokens[3][1:], '%d/%b/%Y:%H:%M:%S'))), tokens[0]

def make_accessdata1():
    items = []
    for line in open(LOGFILE).readlines():
        items.append(parse_access_time(line))
    return items

def make_accessdata2():
    return [parse_access_time(line) for line in open(LOGFILE).readlines()]

def make_accessdata3():
    for line in open(LOGFILE).readlines():
        yield parse_access_time(line)

make_accessdata = make_accessdata1

def analyze():
    ctable = {}
    exist_times = []
    accessdata = make_accessdata()
    accessdata.sort()
    for accesstime, client in accessdata:
        for exit_client in ctable.keys():
            lastaccess, firstaccess = ctable[exit_client]
            if STILLTIME < accesstime - lastaccess:
                del ctable[exit_client]
                print '%s:      %d sec' % (exit_client, lastaccess - firstaccess)
                exist_times.append(lastaccess - firstaccess)
        cur = ctable.get(client)
        if cur:
            lastaccess, firstaccess = cur
            ctable[client] = accesstime, firstaccess
        else:
            ctable[client] = accesstime, accesstime

    print 'Total AVG sec:', float(reduce(lambda a,b: a+b, exist_times)) / len(exist_times)

if __name__ == '__main__':
    analyze()
```

Python Japan User's Group PyJUG Copyright (c) 2001-2002 Python Japan User Group

Lightweight Language Saturday P3 発表資料
じゃんけんクライアント

■Pytnon 2.2以上
■使用モジュール
socket　　低レベルの socket モジュール
random　　乱数を扱うモジュール
thread　　低レベルの thread モジュール
■説明
このプログラムは、socket モジュールを使って、じゃんけんサーバとデータを送受信します。
tcp でじゃんけんサーバからの CALL を受けて手を送信する部分は thread を使って メインループのudpで勝敗を受ける部分から分離しています。
Python には SocketServer や asyncore という、よりハイレベルな socket プログラミング用の標準モジュールがあります。

jankenclient.py

```
#!/usr/bin/env python

import sys, socket, random, thread

HOST = sys.argv[1]
PORT = 2003
LISTEN_PORT = 2004


def sendClientInfo():
    addr = socket.gethostbyname(socket.gethostname())
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((HOST, PORT))
    s.send('%s:%s\r\n' % (addr, LISTEN_PORT))


def Janken():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(('', LISTEN_PORT))
    s.listen(1)
    while True:
        conn, addr = s.accept()
        f = conn.makefile("r+b")
        data = f.readline()
        if data=='CALL\r\n':
            hand = random.choice(['PAPER', 'SCISSORS', 'ROCK'])
            conn.send(hand+'\r\n')


thread.start_new_thread(Janken, ())

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(('', LISTEN_PORT))

sendClientInfo()

while True:
    data += s.recv(1)
    if data[-2:] == '\r\n':
        if data.startswith('100'):
            print '一人勝ち'
        elif data.startswith('110'):
            print '勝ち'
        elif data.startswith('120'):
            print '負け'
        elif data.startswith('130'):
            print ''
        data = ''
```

Python Japan User's Group PyJUG Copyright (c) 2001-2002 Python Japan User Group