5

다양한 형식의 파일처리를 지원하는 모듈

개요

본 장에서는 여러 가지 형식의 파일을 파싱하는데 사용할 수 있는 파이썬 모듈을 설명한다.

마크업 언어

파이썬은 XML(Extensible Markup Language)과 HTML(Hypertext Markup Language)을 처리하는 확장 모듈과 SGML을 처리하는 위한 기본 모듈을 제공한다. 이 세 가지 파일 형식은 기본적으로 구조가 동일하다. 그 이유는 XML, HTML 모두 기본적으로는 SGML로부터 파생되었기 때문이다.

이들 문서는 아래의 예와 같이 시작 태그, 종료 태그, 문자 데이터 그리고 속성 참조(Entity References)를 포함하고 있다.

```
<document name="sample.xml">
    <header>This is a header</header>
    <body>This is the body text.  The text can contain
    plain text (&quot;character data&quot;), tags, and
    entities.
    </body>
</document>
```

위의 예제에서 <document>, <header>와 <body>는 시작 태그이며, 모든 시작 태그에는 대응하는 종료 태그가 있다. 종료 태그의 형식은 시작 태그 이름 앞에 슬래시('/')를 붙이기만 하면 된다. 시작 태그는 위 예제의 name과 같이 속성(attributes)이 하나 이상 있을 수 있다.

시작 태그와 종료 태그 사이에 위치하는 모든 것을 요소(element)라 부른다. 위 예제에서 document는 header와 body라는 두 개의 요소를 포함한다.

마지막으로 "는 문자 항목으로 텍스트 영역에서 예약 문자(reserved character)를 표현하는 데 사용된다. 이러한 경우 항목을 표현하는 문자열 앞에는 반드시 앰퍼샌드(&)를 붙여야 한다. 이 밖에 많이 쓰는 항목으로 '보다 작은(<)'을 표현하는 <' 보다 큰(>)'을 표현하는 > 등이 있다.

XML, HTML, SGML은 구조가 유사한 것처럼 보이지만 그들 사이에는 중요한 차이가 있다. XML은 반드시 시작 태그와 그에 대응하는 종료 태그가 있어야 하고, 태그는 중첩(nesting)이 정확하게 이루어져야 한다. 이처럼 중첩이 잘 된 XML 문서를 웰-폼드(well-formed)라고 부른다. 또한 XML은 대,소문자를 구별하기 때문에 <document>와 <Document>는 두 개의 다른 요소로 간주된다.

반면에 HTML은 XML에 비해 훨씬 융통적이다. 즉, 중첩구조를 만족하지 않아도 된다. 왜냐하면HTML 파서가 생략된 태그를 채워주기 때문이다. 예를 들어, HTML에서 새로운 문장을 시작할 경우 앞 문장의 태그를 종료하지 않고, 새로운 태그를 사용하여도 자동적으로 태그를 만들어 채워준다. 그러므로 문법을 적용하는데 융통적이다. 그리고HTML은 대소문자를 구별하지 않는다. 그러나XML이 새로운 요소를 직접 정의하여 사용할 수 있는데 반하여 HTML은 사양에 정의되어 있는 고정된 요소 집합만 사용해야 한다.

SGML은 HTML보다 훨씬 더 융통적이다. 구현의 측면에서 보면 원문을 요소 구조로 어떻게 번역할 것인지를 사용자가 정의할 수 있으며, DTD(Document Type Definition)를 적용하여 문서 구조의 적정성과 아울러 생략되거나 빠진 태그를 채워 넣을 수 있다. 기술적으로HTML과 XML은 모두 SGML의 응용이라고 볼 수 있다. 이들은 각각 SGML 선언을 보유하고 있으며, HTML에는 표준 DTD가 있다.

파이썬에는 모든 마크업(markup)에 대한 파서가 있다. SGML의 경우 형식이 가장 복잡함에도 불구하고 파이썬의 SGML 처리 모듈인 sgmllib는 실제로 매우 간단하다. 그 이유는 HTML을 처리할 수 있을 정도의 SGML만을 지원하면 충분하다는 판단 하에, 다른 대부분의 문제들은 다루지 않기 때문이다. sgmllib 모듈은 DTD도 처

리하지 않는다. 대신 서브 클래싱을 이용하여 파서를 개인이 원하는 형태로 재구성하는 기능을 지원한다.

파이썬은 SGML 파서를 기반으로 HTML을 지원한다. 파이썬의 HTML 지원 모듈인 htmllib 파서는 실제 번역 작업을 formatter 객체에게 위임한다. formatter 모듈에는 몇 가지의 표준 형식처리기(Formatter)가 있다.

파이썬의 XML 지원은 좀더 복잡하다. 파이썬 1.5.2에서는 내장 모듈인 xmllib 파서로 제한되어 있으며 이 파서는 sgmllib 모듈과 매우 유사하다. 단지 다른 점이 있다면 xmllib는 XML 전체 표준만 지원하도록 구성되었다는 점이다. 파이썬 2.0은 XML 처리를 위한 몇 가지 향상된 도구가 포함되었다. 이 도구들은 expat 파서를 기반으로 한다.

설정 파일

ConfigParser 모듈은 윈도우의 INI 파일과 같은 간단한 설정 파일을 읽고 쓰는 기능을 제공한다. netrc 파일은 .netrc 파일의 내용을 읽는다. 그리고 shlex 모듈은 셸 스크립트 형태의 모든 설정 파일 내용을 읽어오는데 사용한다.

압축파일 형식

파이썬 표준 라이브러리는 GZIP과 ZIP(2.0부터 지원) 같은 일반적으로 널리 쓰이는 압축 파일 형식을 처리하는 기능을 제공한다. gzip 모듈은 GZIP 파일을 읽고 쓰는 기능을, zipfile 모듈은 ZIP 파일을 읽고 쓰는 기능을 제공하며, 이 두 모듈은 모두 데이터 압축 모듈인 zlib에 의존한다.

xmllib 모듈

[예제 5-1]에서 본 것과 같이 정규 표현식을 XML 데이터로 분해하기 위하여 간단한 XML 파서인 xmllib 모듈을 제공한다. XML 파서는 최상위 요소가 하나만 있는지, 모든 태그는 정확히 균형을 맞추어 사용되었는지 같은 기본적인 문서 검사를 수행한다.

파서는 XML 데이터를 토큰 단위로 읽어들인다(네트워크를 통해서 들어오는 데이터들도 이렇게 한다). 그런 후에 시작 태그, 데이터 영역, 종료 태그, 항목(entity) 및 기타 사항 등을 처리하는 메소드를 호출한다.

만약 사용자가 몇 가지 태그에만 관심이 있다면, 특별한 start_tag와 end_tag 메소드를 정의할 수도 있다. tag 부분에는 tag 이름이 들어간다. start 함수는 사전에서 제공하는 속성과 함께 호출된다.

[예제 5-1] xmllib 모듈을 사용하여 요소에서 정보 추출하기

파일명 : xmllib-example-1.py

```
import xmllib

class Parser(xmllib.XMLParser):
    # 인용 번호를 얻는다.
    def __init__(self, file=None):
        xmllib.XMLParser.__init__(self)
        if file:
            self.load(file)

    def load(self, file):
        while 1:
            s = file.read(512)
            if not s:
                break
            self.feed(s)
        self.close()

    def start_quotation(self, attrs):
        print "id =>", attrs.get("id")
        raise EOFError

try:
    c = Parser()
    c.load(open("samples/sample.xml"))
except EOFError:
    pass
```

id => 031

[예제 5-2]는 간단한(완전하지 않은) 번역 엔진을 포함한다.

파서는 요소 스택을 유지하면서 이를 원문과 함께 번역기에 보낸다. 그러면 번역기는 스타일 사전에서 현재 태그의 계층 구조를 검사하고, 만약 스타일 사전에 없으면 새로운 스타일시트에 있는 조각들을 조합하여 스타일 표현기(style descriptor)를 만든다.

[예제 5-2] xmllib 모듈사용하기

파일명 : xmllib-example-2.py

```
import xmllib
import string, sys

STYLESHEET =
    # 각 요소는 하나 이상의 스타일 요소에 영향을 미친다.
   " q u o t a":"t i b y":"1 e t a"],i c
   " l a"n"gw e i"g"hb to"], d
   " n a"m"ew e i"g"hm te d"i u m


class Parser(xmllib.XMLParser):
    # 간단한 스타일링 엔진
    def _ _ i n_i(self, renderer):
        x m l l i b . X M_LiPnair(tsseerl.f )
        s e l_fdata = []
        s e l_ftags = []
        s e l_frenderer = renderer

    def load(self, file):
        while 1:
            s = file.read(8192)
            if not s:
                b r e a k
            s e l f . f e e d ( s )
        s e l f . c l o s e ( )

    def handle_data(self, data):
        s e l_fd.a t a . a p p e n d ( d a t a )

    def unknown_starttag(self, tag, attrs):
        if self._ _ d a t a :
            text = string.join(self.__data,   " ")
            s e l_fr.e n d e r e r .__tags,x ttext()s e l f .
        s e l_ft.a g s . a p p e n d ( t a g )
```

```
        self.__data = []

    def unknown_endtag(self, tag):
        self.__tags.pop()
        if self.__data:
            text = string.join(self.__data, "")
            self.__renderer.__tags, text)self.
        self.__data = []

class DumbRenderer:

    def __init__(self):
        self.cache = {}

    def text(self, tags, text):
        # 태그 스택에서 주어진 스타일에 따라 번역한다.
        tags = tuple(tags)
        style = self.cache.get(tags)
        if style is None:
            # 스타일을 확인한다.
            style = {}
            for tag in tags:
                s = STYLESHEET.get(tag)
                if s:
                    style.update(s)
            self.cache[tags] = style # 캐시를 갱신한다.
        # 표준출력에 인쇄한다.
        sys.stdout.write("%s: %r\n" % (style, text))
        sys.stdout.write("%r\n" % (text + "\n"))

#
# 프로그램 실행 모듈

r = DumbRenderer()
c = Parser(r)
c.load(open("samples/sample.xml"))

{'style': 'italic'} =>
 'I\'ve had a lot of developers come up to me and\012say,
 "I haven\'t had this much fun in a long time. It sure
  beats\012writing '
{'style': 'italic', 'weight': 'bold'} =>
 'Cobol'
{'style': 'italic'} =>
```

```
 '"  __'
'{ style': 'italic','weight': 'medium'} =>
 ' James Gosling'
'{ style': 'italic'} =>
 ' , on\012'
'{ weight': 'bold'} =>
 ' Java'
'{ style': 'italic'} =>
 ' .'
```

xml.parsers.expat 모듈

xml.parsers.expat 모듈(선택적이다)은 제임스 클라크(James Clark)의 Expat XML 파서의 인터페이스를 제공한다. 이 파서는 XML의 모든 사양을 지원하고 속도도 매우 빠르다. 그러므로 대부분의 XML 파서들은 이 Expat 파서를 기본으로 적용하고 있다. [예제 5-3]은 Expat 파서 모듈을 사용한 예다.

[예제 5-3] xml.parsers.expat 모듈 사용하기

파일명 : xml-parsers-expat-example-1.py

```
from xml.parsers import expat

class Parser:

    def __init__(self):
        self._parser = expat.ParserCreate()
        self._parser.StartElementHandler = self.start
        self._parser.EndElementHandler = self.end
        self._parser.CharacterDataHandler = self.data

    def feed(self, data):
        self._parser.Parse(data, 0)

    def close(self):
        self._parser.Parse("", 1) # 데이터의 끝
        del self._parser          # 순환 참조를 제거한다.
```

```
    def start(self, tag, attrs):
        print " S T A",Rrepr(tag), attrs

    def end(self, tag):
        print " E N",Drepr(tag)

    def data(self, data):
        print " D A T",repr(data)

p = Parser()
p . f"e<etda(g > d a "t)a < / t a g >
p . c l o s e ( )

START u'tag' {}
DATA u'data'
END u'tag'
```

파서가 원문을 번역할 때 디폴트로 사용하는 코드는 UTF-8 (XML에서 표준으로 쓰임)이다. 여기에 다른 인코딩(encoding)을 적용하려면, XML 파일 안에 encoding 방법을 담고 있어야 한다. [예제 5-4]는 xml.parsers.expat 모듈을 사용해 ISO Latin-1 코드를 읽는 방법을 보여준다.

[예제 5-4] ISO Latin-1 코드로된 텍스트를 xml.parsers.expat 모듈을 사용하여 읽기

파일명 : xml-parsers-expat-example-2.py

```
from xml.parsers import expat

class Parser:

    def __ i n_i(t s e l f ) :
        self._parser = expat.ParserCreate()
        self._parser.StartElementHandler = self.start
        self._parser.EndElementHandler = self.end
        self._parser.CharacterDataHandler = self.data

    def feed(self, data):
        self._parser.Parse(data, 0)

    def close(self):
        s e l f . _ p a r"s"e, 1) # endaof dataa (
        del self._parser #    순환 참조를 제거한다.
```

```
    def start(self, tag, attrs):
        print " S T Á",,RrefPr(tag), attrs

    def end(self, tag):
        print " E N",,Drepr(tag)

    def data(self, data):
        print " D A T, Æepr(data)

p = Parser()
p . f ë"ë\par<?xml version= ' 1 .' @e n c o'diisnog-=8'8 5 9 - 1
< a u t h o r >
<name>fredrik lundh</name>
< c i t y > l i n k o p i n g < / c i t y >
< / a u t h o r >
"""
)
p . c l o s e ( )
```

```
START u'author' {}
DATA u'\012'
START u'name' {}
DATA u'fredrik lundh'
END u'name'
DATA u'\012'
START u'city' {}
DATA u'link\366ping'
END u'city'
DATA u'\012'
END u'author'
```

sgmllib 모듈

기본적인 SGML 파서는 [예제 5-5]에서와 같이 sgmllib 모듈을 사용하여 처리한다. sgmllib 모듈을 사용하는 방법은 xmllib 모듈의 사용법과 매우 유사하다. sgmllib 모듈은 xmllib보다 제한 사항은 적지만 완성도는 떨어진다.

xmllib와 마찬가지로, sgmllib 모듈도 자체 메소드를 호출하여 시작 태그, 데이터 영역, 종료 태그 그리고 항목을 처리한다. 태그가 몇 개 되지 않는 경우에는, start와 end 같은 특별 메소드를 정의하여 처리할 수도 있다.

[예제 5-5] sgmllib 모듈을 사용하여 타이를 요소추출하기

파일명 : sgmllib-example-1.py

```
import sgmllib
import string

class FoundTitle(Exception):
    p a s s

class ExtractTitle(sgmllib.SGMLParser):

    def __ i n__i(self, verbose=0):
        s g m l l i b . S G_ Mi LnPi(aself,sverbose)
        self.title = self.data = None

    def handle_data(self, data):
        if self.data is not None:
            s e l f . d a t a . a p p e n d ( d a t a )

    def start_title(self, attrs):
        self.data = []

    def end_title(self):
        self.title = string.join(self.data,     " ")
        raise FoundTitle     #     파싱을 중단한다.

def extract(file):
    # HTML/SGML  문장에서 제목 추출하기
    p = ExtractTitle()
    t r y :
        while 1:
            # 작은 조각 단위로 읽어들인다.
            s = file.read(512)
            if not s:
                b r e a k
            p . f e e d ( s )
        p . c l o s e ( )
    except FoundTitle:
```

```
        return p.title
    return None

#
# 실행을 해보자.

print "html=", extract(open("samples/sample.htm"))
print "sgml=", extract(open("samples/sample.sgm"))
```

```
html => A Title.
sgml => Quotations
```

모든 태그를 처리하기 위해 [예제 5-6] 처럼 unknown_starttag와 unknown_end 태그를 오버로딩하여 사용한다.

[예제 5-6] sgmllib 모듈로 SGML 문서 포매팅하기

파일명 : sgmllib-example-2.py

```
import sgmllib
import cgi, sys

class PrettyPrinter(sgmllib.SGMLParser):
    # SGML 문서를 간단하게 인쇄해주는 객체

    def __init__(self):
        # 기본 객체 초기화
        sgmllib.SGMLParser.__init__(self)
        self.flag = 0

    def newline(self):
        # 필요하면 개행한다.
        if self.flag:
            sys.stdout.write("\n")
        self.flag = 0

    def unknown_starttag(self, tag, attrs):
        # 시작 태그를 요청한다.
        # attrs 인자는 (attr, value) 형태의 리스트 튜플이며
        # 이를 문자열로 변환한다.
        text = ""
        for attr, value in attrs:
            text = text + " %s='%s'" % (attr, cgi.escape(value))
```

```
        self.newline()
        sys.stdout.write("<%s>\n" % tag)

    def handle_data(self, text):
        # 텍스트 영역을 요청한다.
        sys.stdout.write(text)
        self.flag = (text[-1:] != "\n")

    def handle_entityref(self, text):
        # 속성(ent)을 요청한다.
        sys.stdout.write("&%s;" % text)

    def unknown_endtag(self, tag):
        # 종료 태그를 요청한다.
        self.newline()
        sys.stdout.write("</%s>\n" % tag)

#
# 실행하기

file = open("samples/sample.sgm")

p = PrettyPrinter()
p.feed(file.read())
p.close()
```

```
<chapter>
<title>
Quotations
<title>
<epigraph>
<attribution>
eff-bot, June 1997
<attribution>
<para>
<quote>
Nobody expects the Spanish Inquisition! Amongst
our weaponry are such diverse elements as fear, surprise,
ruthless efficiency, and an almost fanatical devotion to
Guido, and nice red uniforms &mdash; oh, damn!
<quote>
<para>
<epigraph>
<chapter>
```

[예제 5-7]은 XML의 관점에서 SGML 문서가 웰-폼드(well-formed)인지 검사한다. 웰-폼드 문서는 모든 요소가 적절히 중첩되어 있으며 하나의 시작 태그에 대해 반드시 하나의 종료 태그가 존재한다.

웰-폼드인지를 검사하는 방법은 매우 간단하다. 현재 열려있는 시작 태그의 리스트를 보유하고, 종료 태그와 하나씩 짝을 이루는지 살펴보면서 문서를 처리해 문서의 끝에 도착했을 때 남아있는 시작 태그와 종료 태그가 없으면 이는 웰-폼드이다.

[예제 5-7] sgmllib 모듈로 웰-폼드 문서인지 검사하기

파일명 : sgmllib-example-3.py

```
import sgmllib

class WellFormednessChecker(sgmllib.SGMLParser):
    # XML 관점에서 SGML 문서가 웰-폼드(well-formed)인지 검사하게
    def __init__(self, file=None):
        sgmllib.SGMLParser.__init__(self)
        self.tags = []
        if file:
            self.load(file)

    def load(self, file):
        while 1:
            s = file.read(8192)
            if not s:
                break
            self.feed(s)
        self.close()

    def close(self):
        sgmllib.SGMLParser.close(self)
        if self.tags:
            raise SyntaxError, "start tag %s not closed" % self.tags[-1]

    def unknown_starttag(self, start, attrs):
        self.tags.append(start)

    def unknown_endtag(self, end):
        start = self.tags.pop()
        if end != start:
```

```
                raise SyntaxError, " end tag %s does't match start tag %s" % \ p a r
(end, start)

t r y :
    c = WellFormednessChecker()
    c . l o a d ( o p e n ( " e x a m p l e s / s a m p l e . h t m
except SyntaxError:
    raise    # 오류를 일으킨다.
e l s e :
    print " document is well-formed "
```

```
Traceback (innermost last):
...
SyntaxError: end tag head does 't match start tag meta
```

마지막으로 [예제 5-8]은 SGML과 HTML의 필터링(filtering)을 위한 클래스를 소개한다. 이 클래스를 사용하려면 먼저 자신만의 기본 클래스를 생성하고 start와 end 메소드를 구현하면 된다.

[예제 5-8] sgmllib 모듈로 SGML 문서 필터링하기

파일명 : sgmllib-example-4.py

```
import sgmllib
import cgi, string, sys

class SGMLFilter(sgmllib.SGMLParser):
    # sgml 필터. 문서 내의 요소 처리를 위해
    # start 와 e n d를 오버라이드( o v e r 한다. d e )

    def __ i n__(self, outfile=None, infile=None):
        s g m l l i b . S G M L P a r s e r . _ _ i n i t _ _ ( s e l f )
        if not outfile:
            outfile = sys.stdout
        self.write = outfile.write
        if infile:
            s e l f . l o a d ( i n f i l e )

    def load(self, file):
        while 1:
            s = file.read(8192)
            if not s:
```

```
                b r e a k
            s e l f . f e e d ( s )
        s e l f . c l o s e ( )

    def handle_entityref(self, name):
        s e l f . "w&r%i"st%;eam(e)

    def handle_data(self, data):
        s e l f . w r i t e ( c g i . e s c a p e ( d a t a ) )

    def unknown_starttag(self, tag, attrs):
        tag, attrs = self.start(tag, attrs)
        if tag:
            if not attrs:
                s e l f . "w<r%i"st%>tag()
            e l s e :
                s e l f . "w<r%i"s%t tag)(
                for k, v in attrs:
                    s e l f . "w  %s i="%s (k, repr(v)))
                s e l f . "w>"r) i t e (

    def unknown_endtag(self, tag):
        tag = self.end(tag)
        if tag:
            s e l f . "w<r/i"% %s tag)

    def start(self, tag, attrs):
        return tag, attrs #   오버라이드( o v e r r i d e )

    def end(self, tag):
        return tag #  오버라이드

class Filter(SGMLFilter):

    def fixtag(self, tag):
        if tag == " e ":
            tag =" i"
        if tag == " s t r":i n g
            tag =" b"
        return string.upper(tag)

    def start(self, tag, attrs):
        return self.fixtag(tag), attrs
```

```
    def end(self, tag):
        return self.fixtag(tag)

c = Filter()
c.loads(open("samples/sample.htm").read())
```

htmllib 모듈

htmllib 모듈은 태그 기반의 HTML 파서를 포함하고 있는데, 이것은 데이터를 formatting 객체로 전달하는 역할을 한다. [예제 5-9]는 이 모듈의 사용 예다. 이 모듈을 사용하여 HTML 파일을 파싱하는 방법은 formatter 모듈 부문에서 더 자세히 알 수 있다.

[예제 5-9] htmllib 모듈 사용하기

파일명 : htmllib-example-1.py

```
import htmllib
import formatter
import string

class Parser(htmllib.HTMLParser):
    # 앵커 텍스트(anchor text) 와
    # 이에 해당하는 하이퍼링크를
    # 매핑(map)한 사전을 반환한다.

    def __init__(self, verbose=0):
        self.anchors = {}
        f = formatter.NullFormatter()
        htmllib.HTMLParser.__init__(self, f, verbose)

    def anchor_bgn(self, href, name, type):
        self.save_bgn()
        self.anchor = href

    def anchor_end(self):
        text = string.strip(self.save_end())
```

```
        if self.anchor and text:
            self.anchors[text] = self.anchors.get(text, []) + [self.anchor]

file = open("samples/sample.htm")
html = file.read()
file.close()

p = Parser()
p.feed(html)
p.close()

for k, v in p.anchors.items():
    print k, "=>", v

print
```

```
link => ['http://www.python.org']
```

만약 HTML 파일을 파싱만하고 출력 장치에 맞게 번역하지 않아도 되는 경우라면, sgmllib 모듈을 사용하는 것이 더 간단하다.

htmlentitydefs 모듈

htmlentitydefs 모듈은 HTML이 사용하는 많은 ISO Latin-1 문자 속성 사전을 포함하고 있다. [예제 5-10]은 이 모듈을 사용한 예다.

[예제 5-10] htmlentitydefs 모듈 사용하기

파일명 : htmlentitydefs-example-1.py

```
import htmlentitydefs

entities = htmlentitydefs.entitydefs

for entity in "amp","quot","copy","yuml":
    print entity, "=", entities[entity]
```

```
amp = &
quot = "
copy = \302\251
yen = \302\245
```

[예제 5-11]은 문자열 내의 속성을 번역하기 위해 htmlentitydef 사전과 정규 표현식을 결합하는 방법을 보여준다(이는 cgi.escape와는 반대되는 개념이다).

[예제 5-11] htmlentitydef 모듈을 사용하여 속성 번역하기

파일명 : htmlentitydefs-example-2.py

```
import htmlentitydefs
import re
import cgi

pattern = re.compile( "&(.+?);" )

def descape_entity(m, defs=htmlentitydefs.entitydefs):
    # callback: 속성을 ISO Latin 값으로 번역하기
    try:
        return defs[m.group(1)]
    except KeyError:
        return m.group(0)       # 현재의 것을 사용한다.

def descape(string):
    return pattern.sub(descape_entity, string)

print descape("&lt;spam&amp;eggs&gt;")
print descape(cgi.escape("<spam&eggs>"))
```

```
<spam&eggs>
<spam&eggs>
```

마지막으로 [예제 5-12]는 XML의 예약어와 ISO Latin-1 코드의 문자를 XML 문자열로 번역하는 방법을 알려준다. 이는 cgi.escape와 유사하다. 단지 여기서는 비-아스키 문자(non-ASCII characters)도 치환한다는 점이 다르다.

[예제 5-12] ISO Latin-1 속성을 회피하기

파일명 : htmlentitydefs-example-3.py

```
import htmlentitydefs
```

```
import re, string

# 이 패턴은 예약 문자 또는 비-아스키 문자의 일부 문자열과 부합한다.
pattern = re.compile(r  "[ & <\>x\8 0 - ")x f f ] +

# 문자맵을 생성한다.
entity_map = { }

for i in range(256):
    entity_map[chr(i)] =  " & %"d%; i

for entity, char in htmlentitydefs.entitydefs.items():
    if entity_map.has_key(char):
        entity_map[char] =  " & %"s%;entity

def escape_entity(m, get=entity_map.get):
    return string.join(map(get, m.group()),     "  ")

def escape(string):
    return pattern.sub(escape_entity, string)

print escape(" < s p a m &)e g g s >
print escape(" \303\245 I  \303\245a \303\244 e \303\266     ")
```

```
&lt;spam&amp;eggs&gt;
&aring; i &aring;a &auml; e &ouml;
```

formatter 모듈

formatter 모듈은 htmllib 모듈과 함께 사용할 수 있는 formatter 클래스를 제공한다.

formatter 모듈은 formatter와 writers 두 개의 클래스 계열을 제공한다. formatter는 HTML 파서로부터 오는 태그와 데이터 문자열 스트림(stream)을 출력장치에 적합한 이벤트 스트림(event stream)으로 변환한다. 그러면 Writer 클래스는 이를 출력장치에 전달한다. [예제 5-13]은 이에 대한 사례를 보여준다.

대부분의 경우 포매팅을 위해서는 AbstractFormatter 클래스를 사용한다.

이 클래스는 서로 다른 종류의 포매팅 이벤트를 대표하는 writer 객체를 호출한다. 그러면 AbstractWriter 클래스는 각각의 메소드 호출에 따라 간단한 메시지를 출력한다.

[예제 5-13] formatter 모듈을 사용하여 HTML을 이벤트스트림으로 변환하기

파일명 : formatter-example-1.py

```
import formatter
import htmllib

w = formatter.AbstractWriter()
f = formatter.AbstractFormatter(w)

file = open("samples/sample.htm")

p = htmllib.HTMLParser(f)
p.feed(file.read())
p.close()

file.close()
```

```
send_paragraph(1)
new_font(('h1', 0, 1, 0))
send_flowing_data('A Chapter.')
send_line_break()
send_paragraph(1)
new_font(None)
send_flowing_data('Some text. Some more text. Some ')
send_flowing_data('')
new_font((None, 1, None, None))
send_flowing_data('emphasized ')
new_font(None)
send_flowing_data('text. A')
send_flowing_data('link')
send_flowing_data('[1]')
send_flowing_data('.')
```

formatter 모듈은 AbstractWriter 클래스뿐만 아니라, 전달되는 모든 이벤트를 무시해버리는 NullWriter 클래스, 그리고 이벤트 스트림을 단순한 텍스트로 변환시키는 DumbWriter 클래스를 포함하고 있다. 이들의 사용 예가 [예제 5-14]에 있다.

[예제 5-14] formatter 모듈을 사용하여 HTML 을 단순한 텍스트로 변환하기

파일명 : formatter-example-2.py

```
import formatter
import htmllib

w = formatter.DumbWriter()              #       단순 텍스트
f = formatter.AbstractFormatter(w)

file = open(" s a m p l e s / s")a m p l e . h t m

# html body 를 단순 텍스트로 출력한다.
p = htmllib.HTMLParser(f)
p . f e e d ( f i l e . r e a d ( ) )
p . c l o s e ( )

f i l e . c l o s e ( )

# 링크를 인쇄한다.
p r i n t
p r i n t
i = 1
for link in p.anchorlist:
    print i, " = ">, link
    i = i + 1
```

```
A Chapter.

Some text. Some more text. Some emphasized text. A link[1].

1 => http://www.python.org
```

[예제 5-15]에서는 DumbWriter 클래스를 서브 클래싱하여 새로운 Writer 클래스를 만든 사례를 제시하고 있다. 새로운 클래스는 현재의 폰트 스타일을 기억하면서 폰트에 따라 약간씩 출력을 수정하도록 만들었다.

[예제 5-15] 사용자가정의한 Write 클래스와 함께 formatter 모듈 사용하기

파일명 : formatter-example-3.py

```
import formatter
import htmllib, string
```

```
class Writer(formatter.DumbWriter):

    def __init__(self):
        formatter.DumbWriter.__init__(self)
        self.tag = ''
        self.bold = self.italic = 0
        self.fonts = []

    def new_font(self, font):
        if font is None:
            font = self.fonts.pop()
            self.tag, self.bold, self.italic = font
        else:
            self.fonts.append((self.tag, self.bold, self.italic))
            tag, bold, italic, typewriter = font
            if tag is not None:
                self.tag = tag
            if bold is not None:
                self.bold = bold
            if italic is not None:
                self.italic = italic

    def send_flowing_data(self, data):
        if not data:
            return
        atbreak = self.atbreak or data[0] in string.whitespace
        for word in string.split(data):
            if atbreak:
                self.file.write(' ')
            if self.tag in ('h1', 'h2', 'h3'):
                word = string.upper(word)
            if self.bold:
                word = '*' + word + '*'
            if self.italic:
                word = '_' + word + '_'
            self.file.write(word)
            atbreak = 1
        self.atbreak = data[-1] in string.whitespace

w = Writer()
f = formatter.AbstractFormatter(w)

file = open('samples/sample.htm')
```

```
# html body 를 단순 텍스트로 출력하기
p = htmllib.HTMLParser(f)
p . f e e d ( f i l e . r e a d ( ) )
p . c l o s e ( )
```

```
_A_ _CHAPTER._

Some text. Some more text. Some *emphasized* text. A link[1].
```

ConfigParser 모듈

ConfigParser 모듈은 설정 파일을 읽는 기능을 제공한다.

설정 파일은 반드시 윈도우의 INI 파일과 같은 형태로 표현되어야 한다. 여기에는 하나 이상의 섹션이 존재할 수 있고, 섹션의 구별은 대괄호를 사용한다. 각각의 섹션은 하나 이상의 환경 설정 항목을 포함한다.

아래 예제는 [예제 5-16]에서 샘플로 사용할 설정 파일의 내용이다.

```
[ b o o k ]
title: The Python Standard Library
author: Fredrik Lundh
email: fredrik@pythonware.com
version: 2.0-001115

[ e m a t t e r ]
pages: 250

[ h a r d c o p y ]
pages: 350
```

[예제 5-16] 은 ConfigParser 모듈을 사용하여 샘플 설정 파일을 읽어들이는 예다.

[예제 5-16] ConfigParser 모듈 사용하기

파일명 : configparser-example-1.py

```
import ConfigParser
```

```
import string

config = ConfigParser.ConfigParser()

c o n f i g . s o u r c e ( " s a m p l e s / s a m p l e . i n i " )

# 요약해서 인쇄하기
p r i n t
print string.upper(config.get( " b o o k " , " k t i t l e " ))
print " b y " , config.get( " b o o k " , " a u t h o r " )
print " ( " + config.get( " b o o k " , " e m a i l " ) + " ) "
p r i n t
print config.get( " e m a t t e r " , " p a g e s " ) , " s p a g e s "
p r i n t

# 환경 설정 파일 전체를 덤프( d u m p 한다. )
for section in config.sections():
    print section
    for option in config.options(section):
        print " " , option, " =", config.get(section, option)
```

```
THE PYTHON STANDARD LIBRARY
by Fredrik Lundh (fredrik@pythonware.com)

250 pages

book
  title = The Python Standard Library
  email = fredrik@pythonware.com
  author = Fredrik Lundh
  version = 2.0-001115
  __name__ = book
ematter
  __name__ = ematter
  pages = 250
hardcopy
  __name__ = hardcopy
  pages = 350
```

파이썬 2.0의 ConfigParser 모듈을 이용하면 환경 설정 데이터를 다른 파일에 기록할 수 있다. [예제5-17] 은 ConfigParser 모듈의 사용 예를 보여준다.

[예제5-17] ConfigParser 모듈로설정되어있는 데이터 출력하기

파일명 : configparser-example-2.py

```
import ConfigParser
import sys

config = ConfigParser.ConfigParser()

# 인자의 개수를 정한다.
c o n f i g . a d" db _o S)ekc t i o n (
c o n f i"gb.os"qé"ktt (i t",1 the python standard library ")
c o n f i"gb.os"qé"kta (u t"h,"ofnedrik lundh ")

c o n f i g . a d" de _m sa et")tteiro n (
c o n f i"ge.ms ae t",t(pe ar g",e250)

# 화면에 출력한다
c o n f i g . w r i t e ( s y s . s t d o u t )
```

```
[book]
title = the python standard library
author = fredrik lundh

[ematter]
pages = 250
```

netrc 모듈

netrc 모듈은 [예제 5-18]에서 보는 것처럼 .netrc 구성 파일을 파싱한다. .netrc 파일은 FTP 사용자 아이디와 패스워드를 사용자의 홈 디렉토리에 저장하는데 사용된다(이 때 주의할 것은 파일의 읽기 권한은 사용자에게만있다는 것이다. 즉," chmod 600 ~ /.netrc"로 되어 있다).

[예제 5-18] netrc 모듈 사용하기

파일명 : netrc-example-1.py

```
import netrc
```

```
# 디폴트는 $HOME/.netrc
info = netrc.netrc("samples/sample.netrc")

login, account, password = info.authenticators("secret.fbi")
print "login", "=>", repr(login)
print "account", "=>", repr(account)
print "password", "=>", repr(password)
```

```
login => 'mulder'
account => None
password => 'trustno1'
```

shlex 모듈

shlex 모듈은 유닉스의 셸 문법을 기반으로 하는 간단한 토큰 생성기(tokenizer)를 제공한다. [예제5-19]는 shlex 모듈의 사용 예를 보여주고 있다.

[예제 5-19] shlex 모듈 사용하기

파일명 : shlex-example-1.py

```
import shlex

lexer = shlex.shlex(open("samples/sample.netrc"))
lexer.wordchars = lexer.wordchars + "."

while 1:
    token = lexer.get_token()
    if not token:
        break
    print repr(token)
```

```
'machine'
'secret.fbi'
'login'
'mulder'
'password'
'trustno1'
```

```
' machine '
' non.secret.foo '
' login '
' scully '
' password '
' noway '
```

zipfile 모듈

2.0 버전의 새로운 모듈인 zipfile은 일반적인 ZIP 압축 파일에서 파일을 읽고 쓸 수 있는 기능을 제공한다.

내용 출력하기

압축 파일에 포함되어있는 내용을 출력하려면 [예제 5-20]에서와 같이 namelist와 infolist 메소드를 사용하면 된다. namelist는 파일 이름 리스트를, infolist는 ZipInfo 인스턴스 리스트를 반환한다.

[예제 5-20] zipfile 모듈을 사용하여 ZIP 파일에포함된 파일 출력하기

파일명 : zipfile-example-1.py

```
import zipfile

file = zipfile.ZipFile( " s a m p l e s / s " , " m " ) p l e . z i p

# 파일 이름 출력하기
for name in file.namelist():
    print name,
p r i n t

# 파일 정보 출력하기
for info in file.infolist():
    print info.filename, info.date_time, info.file_size
```

```
sample.txt sample.jpg
```

```
sample.txt (1999, 9, 11, 20, 11, 8) 302
sample.jpg (1999, 9, 18, 16, 9, 44) 4762
```

ZIP 파일에서 데이타 읽기

압축 파일에 담긴 데이터를 읽기 위해 [예제 5-21]과 같이 단순히 read 메소드만 사용하면 된다. read 메소드는 파일 이름을 인자로 전달하고, 결과로 압축된 데이터를 문자열로 반환한다.

[예제 5-21] zipfile 모듈을사용하여 ZIP 파일에서데이터 읽기

파일명 : zipfile-example-2.py

```
import zipfile

file = zipfile.ZipFile( " s a m p l e s / s" "a" "m" "p l e . z i p

for name in file.namelist():
    data = file.read(name)
    print name, len(data), repr(data[:10])
```

```
sample.txt 302 ' We will pe'
sample.jpg 4762 ' \377\307\377\340\000\020JFIF    '
```

ZIP 파일 만들기

압축 파일에 파일을 추가하는 것은 매우 간단하다. write 메소드를 이용하여 압축 파일에 파일 이름을 전달하기만 하면 된다. [예제 5-22]의 스크립트는 sample 디렉토리에 있는 모든 파일을 ZIP 파일로 압축하는 예다.

[예제 5-22] zipfile 모듈을사용하여 압축 파일만들기

파일명 : zipfile-example-3.py

```
import zipfile
import glob, os

# zip 파일을 쓰기 모드로 열고, 필요한 사항을 추가한다.
```

```
file = zipfile.ZipFile( " t e s t ","zw"i)p

for name in glob.glob( " s a m p l"i)e:s / *
    file.write(name, os.path.basename(name), zipfile.ZIP_DEFLATED)

f i l e . c l o s e ( )

# 다시 zip 파일을 열고, 안의 내용을 확인한다.

file = zipfile.ZipFile( " t e s t ","zr"i)p
for info in file.infolist():
    print info.filename, info.date_time, info.file_size, info.compress_size
```

```
sample.wav (1999, 8, 15, 21, 26, 46) 13260 10985
sample.jpg (1999, 9, 18, 16, 9, 44) 4762 4626
sample.au (1999, 7, 18, 20, 57, 34) 1676 1103
...
```

write 메소드의 세 번째 인자는 압축 방식 또는 압축의 가부를 알려준다. 일반적으로 디폴트 모드는 어떠한 압축도 하지 않고 데이터를 압축 파일에 저장하는 zipfile.ZIP_STORED 이다. zlib 모듈이 설치되어 있는 경우에는 zipfile.ZIP_DEFLATED 모드를 사용하여 'deflate' 압축을 할 수도 있다.

zipfile 모듈은 압축 파일에 문자열을 추가하는 기능도 제공한다 그러나 문자열을 압축 파일에 추가하는 것은 조금 복잡한데 단지 파일명과 데이터를 압축 파일에 전달하는게 아니라 ZipInfo 인스턴스를 생성하고 정확히 설정해야 한다. [예제 5-23]은 압축 파일에 문자열을 추가하는 사례다.

[예제 5-23] zipfile 모듈을 사용하여 ZIP 파일에문자열 추가하기

파일명 : zipfile-example-4.py

```
import zipfile
import glob, os, time

file = zipfile.ZipFile( " t e s t ","zw"i)p

now = time.localtime(time.time())[:6]

for name in (" l i "f,"eo "f," b r i")a:n
    info = zipfile.ZipInfo(name)
    info.date_time = now
```

```
    info.compress_type = zipfile.ZIP_DEFLATED
    file.writestr(info, name*1000)

file.close()

# 파일을 다시 열어서 그 안의 내용을 확인한다.

file = zipfile.ZipFile("test","zrip")

for info in file.infolist():
    print info.filename, info.date_time, info.file_size, info.compress_size
```

```
life (2000, 12, 1, 0, 12, 1) 4000 26
of (2000, 12, 1, 0, 12, 1) 2000 18
brian (2000, 12, 1, 0, 12, 1) 5000 31
```

gzip 모듈

[예제 5-24]에서 보듯이, gzip 모듈은 gzip으로 압축된 파일을 마치 일반 파일처럼 읽고 쓰는 기능을 제공한다.

[예제 5-24] gzip 모듈을 사용하여 압축 파일 읽기

파일명 : gzip-example-1.py

```
import gzip

file = gzip.GzipFile("samples/sample.gz")

print file.read()
```

```
Well it certainly looks as though we're in for
a splendid afternoon's sport in this the 127th
Upperclass Twit of the Year Show.
```

일반적인 gzip 모듈은 seek와 tell 메소드를 지원하지 않는다. 그래서 [예제 5-25]에서는 seek와 tell을 정의하여 전방 탐색(forward seeking) 하는 방법을 보여주고 있다.

[예제 5-25] seek와 tell 을 지원하도록 gzip 모듈 확장하기

파일명 : gzip-example-2.py

```
import gzip

class gzipFile(gzip.GzipFile):

    # GzipFile 에 s e 와 tell 지원기능 추가하기

    offset = 0

    def read(self, size=None):
        data = gzip.GzipFile.read(self, size)
        self.offset = self.offset + len(data)
        return data

    def seek(self, offset, whence=0):
        # 새로운 위치를 알아낸다(전방( f o r w만a탐색할) 수 있다) .
        if whence == 0:
            position = offset
        elif whence == 1:
            position = self.offset + offset
        e l s e :
            raise IOError, " Illegal argument "
        if position < self.offset:
            raise IOError, " Cannot seek backwards "

        # 16k 블록에서 앞으로 건너 뛴다.
        while position > self.offset:
            if not self.read(min(position - self.offset, 16384)):
                b r e a k

    def tell(self):
        return self.offset

#
# 실행하기

file = gzipFile(" s a m p l e s / "s) a m p l e . g z
f i l e . s e e k ( 8 0 )

print file.read()
```

this the 127th Upperclass Twit of the Year Show.