Web Clients

Python on the Client Side: the urllib2 module

A Python program can also take the place of a browser, requesting and downloading HTML pages and other files. Your Python program can work like a web spider (for example visiting every page on a website looking for particular data or compiling data from the site), can visit a page repeatedly to see if it has changed, can visit a page once a day to compile information for that day, etc.

urllib2 is a full-featured module for making web requests. Although the **requests** module is strongly favored by some for its simplicity, it has not yet been added to the Python builtin distribution.

The **urlopen** method takes a url and returns a file-like object that can be **read()** as a file:

```
import urllib2
my_url = 'http://www.nytimes.com'
readobj = urllib2.urlopen(my_url)
text = readobj.read()
print text
readobj.close()
```

Alternatively, you can call **readlines()** on the object (keep in mind that many objects that can deliver file-like string output can be read with this same-named method:

```
for line in readobj.readlines():
  print line
readobj.close()
```

The text that is downloaded is HTML, Javascript, and possibly other kinds of data. It is not designed for human reading, but it does contain the info that we would need to retrieve from a page. To do so programmatically (instead of visually), we need to engage in *web scraping*. Most simply, this is done with regexes (coming up).

Encoding Parameters: urllib2.urlencode()

When including parameters in our requests, we must *encode* them into our request URL. The **urlencode()** method does this nicely:

```
import urllib
params = urllib.urlencode({'choice1': 'spam and eggs', 'choice2': 'spam, spam, bacon and spam'})
print "encoded query string: ", params
f = urllib.urlopen("http://i5.nyu.edu/~dbb212/cgi-bin/pparam.cgi?%s" % params)
print f.read()
```

this prints:

```
encoded query string: choice1=spam+and+eggs&choice2=spam%2C+spam%2C+bacon+and+spam

choice1:  spam and eggs<BR>
choice2:  spam, spam, bacon and spam<BR>
```

requests: even more convenient web requests

Requests is well regarded among developers discussing Python client modules like **urllib2**. It isn't part of the Python distribution but it is easily installed.

Requests is preferred mostly for its ease of use. Parameters, parameter encoding, and the various request types are easily handled.

```
import requests

# get a web page
resp = requests.get('http://www.mywebsite.com/user')

# send a post request with parameter input
userdata = {"firstname": "John", "lastname": "Doe", "password": "jdoe123"}
resp = requests.post('http://www.mywebsite.com/user', params=userdata)
```

The text of a response is easily accessed through an attribute. JSON data can also be easily decoded.

```
print resp.text
print resp.json()
```

Some other features of **requests**:

- International Domains and URLs
- Keep-Alive & Connection Pooling
- Sessions with Cookie Persistence
- Browser-style SSL Verification
- Basic/Digest Authentication
- Elegant Key/Value Cookies
- Automatic Decompression
- Unicode Response Bodies
- Multipart File Uploads
- Connection Timeouts

Beautiful Soup

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc, 'html.parser')
```

soup.title
#

soup.title.name
u'title'

soup.title.string
u'The Dormouse's story'

soup.title.parent.name
u'head'

soup.p
#

The Dormouse's story

soup.a # Elsie (http://example.com/elsie) soup.find_all('a') # [Elsie (http://example.com/elsie), # Lacie (http://example.com/lacie), # Tillie (http://example.com/tillie)] soup.find(id="link3") # Tillie (http://example.com/tillie) # all links in a page for link in soup.find_all('a'): print(link.get('href')) # pure text from a page print(soup.get_text()) ## Tag object soup.a # fetches the next tag

Sending email with smtplib

Sending mail is simple when you have an SMTP server running and available on your host computer. Python's **smtplib** module makes this easy:

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

# Import smtplib for the actual sending function
import smtplib

# Import the email modules we'll need
from email.mime.text import MIMEText

# Create a text/plain message formatted for email
msg = MIMEText('Hello, email.')

from_address = 'dbb212@nyu.edu'
to_address = 'david.beddoe@gmail.com'
subject = 'Test message from a Python script'

msg['Subject'] = subject
msg['From'] =    from_address
msg['To'] =      to_address

s = smtplib.SMTP('localhost')
s.sendmail(from_address, [to_address], msg.as_string())
s.quit()
```