Python Request

1. Using Python on Web

Python contains libraries which make it help to interact with web-sites to perform tasks like logging into gmail, viewing web pages, filling forms, viewing and saving cookies with nothing more than a few lines of code. Once a request is sent to a server, a Python script would return the information in almost the same way a browser would return. Some of the benefits of using Python on web:

- Can write scripts to automate interaction with a web-page.
- Can just use Python to fetch the HTML pages and process them.
- Can make GET and POST requests.
- Can make access external sites.

2. Urllib

Urllib is the default Python module used for opening HTTP URL's. It can accomplish other functions like Basic Authentication, getting Cookies, GET and POST requests, Error handling, viewing headers. But some of the functions like urllib.urlopen() have been replaced by urllib2.urlopen() in Python 3. Urllib2 provides additional functionalities such as fetching URLs using a variety of different protocols like HTTP, FTP and can also accept a request object to set the headers for a URL request. On the other hand urllib provides methods like urlencode() -used for the generation of GET query strings, which is absent in urllib2. So urllib cannot be completely replaced with urllib2 and vice versa. In spite of using both urllib and urllib2 there are various drawbacks of using this Python module:

- First and foremost it is unclear which module- urllib or urllib2 to be used. This is confusing especially for the beginners.
- Better choice seems urllib2 but it does not provide all the functionalities.
- The documentation or the API for both urllib and urllib2 is extremely difficult to understand. It is heavily over-engineered.

3. Python Request

Requests is a simple, easy-to-use HTTP library written in Python. Urllib3 is embedded within Requests providing additional features like Keep-alive and HTTP connection pooling to be 100% automatic. Requests makes interaction with web services seamless. It overcomes most of the difficulties faced while using urllib/urllib2 like manually adding query strings to your URLs, encoding of data while making GET/POST requests etc. We need to import a single Python module *import requests* before writing code. The main lead developer of Python-Requests is Kenneth Reitz.

3.1 Installation

Steps to install Python Request.

3.2 Parsing JSON

It is often found that web pages have JSON embedded in their code. Hence while receiving requests we might often get response in the JSON format. Hene requests have a built-in JSON decoder which helps in parsing JSON code. We can just import the JSON module.

Examples on how to parse JSON using Requests.

3.3 Features

- Sessions with Cookie Persistence: We can make a Session object and set certain parameters and cookie values. This allows to persist these parameters and cookies across all requests made from the Session instance. We can also give default data to the request methods.
- Keep-Alive & Connection Pooling: Keep-alive is available and automatic within a session. There is a pool of connections and a connection is released for only once all its data has been read.
- Browser-style SSL Verification: We can verify the SSL certificates for HTTPS.
- Response Content: Requests can automatically decode the response based on the header values. Using r.encoding we can change the encoding type.
- Basic/Digest Authentication: Requests provides two authentication scheme implementations- HTTPBasicAuth and HTTPDigestAuth. They are the subclasses of requests.auth.AuthBase.

3.4 Comparison

Examples comparing the length, complexity and size of codes written using Requests against that written using urllib and urllib2.

4. Advantages

- There is a method r.encoding and has a unicode object. Encoding in urllib2 is very difficult since we need to use regular expressions to get the response code from the headers and use that to encode the response to unicode.

- Requests encodes the parameters automatically so we just pass them as simple arguments unlike in urllib where we need to use urllib.encode() where we need to encode the parameters before passing them.
- Automatically decode the response into Unicode.
- Automatically save the content, enabling us to access it multiple times, unlike the read-once file-like object returned by urllib2.urlopen().