Python Requests

Sakshi Bansal
Amrita University, India
sakshi.april5@gmail.com

**Mini DebConf - Women
Barcelona**

Why use 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 get and parse RSS feeds.
- Can create a web spider to test your site or search other sites.
- Uses Beautifulsoup (Python module) for parsing HTML

and XML files.

Urllib

- Urllib/Urllib2 are the default Python modules used for opening HTTP URL’s.
- Urllib cannot be completely replaced by urllib2 since the former has methods that are absent in the later.   Eg: urlencode()
- The documentation for both urllib and urllib2 is extremely difficult to understand.
- Even for a simple GET request it is impossible to write a short script using urllib2.

Python Requests

Introduction

- Requests is a simple, easy-to-use HTTP library written in Python.
- Lead developer is Kenneth Reitz who is also a member of the **Python Software Foundation**.
- It can be used for various Operating Systems like Debian, Unix etc.

Parsing JSON

- Web pages usually have JSON embedded in their code.
- While receiving requests we often get response in JSON format.
- Requests have a built-in JSON decoder which helps in parsing JSON code.
- We can just import the JSON module.

a) How to know if the response is in JSON format

```
import requests

r = requests.get(“http://www.example.com”)
print r.status_code
print r.headers['content-type']
```

Output:

```
200
'application/json'
```

b) How to parse using JSON built-in module and Requests

```
import json
import requests

response = requests.get(url=url, params=params)
data = json.load(response)
```

json.load(response) - used for decoding the response

json.dump(request) - used for encoding request

Features

- 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.

- Cookies: We can get the cookies set by the server from the response

  -
    ```
    url = 'http://example.com/cookie'
    r = requests.get(url)
    r.cookies['cookie_name']
    ```
  - We can also send cookies to the server:

    -
      ```
      url = 'http://example2.com/cookies'
      cookies = dict(cookie1='This_is_a_cookie')
      r = requests.get(url, cookies=cookies)
      ```

- Requests can automatically decode the response based on the header values.
- Using .encoding method we can change the encoding type.
- Supports various types of exceptions such as DNS failure, Invalid HTTP response etc.
- Supports the entire restful API i.e, all its methods- PUT, GET, DELETE, POST.

Python Requests
v/s
Urllib/Urllib2

Example 1: Making a POST request

1.1 using urllib2/urllib

```
import urllib
import urllib2

url = "http://www.example.com"
values = {"firstname":" abc ", "lastname":" xyz "}

header = {"User-Agent":"Mozilla/4.0(compatible;MSIE 5.5;Windows NT)"}

values = urllib.urlencode(values)
request = urllib2.Request(url, values, header)
```

```
response = urllib2.urlopen(request)
html_content = response.read()
```

Note: In the above example 2.1 we had to make a use of both the urllib and urllib2 modules in order to write a script for a simple POST request.

1.2 using requests

import requests

```
values = {""firstname":" abc ", "lastname":" xyz "}
r = requests.post('https://www.example.com, data=values)


print r.status_code
print r.text
```

Thank You!!!