libweb Documentation

Release 1.0.0

Hurricane Labs, LLC

Jul 03, 2017

Contents

libweb is, simply, a parsing engine for the web. The goal of the libweb project is to provide a library capable of parsing the vast majority of consumable content on the web. libweb strives to maintain compatibility with current versions of Python, and specifically tests against Python 2.7 and Python 3.3+.

CHAPTER 1

Documentation

User Guide

Introduction

libweb is a framework for interacting with web sites of all shapes and sizes. Parsers are included for the most common web formats, and new parsers are easy to add. libweb officially supports Python 2.7, Python 3.3, Python 3.4 and Python 3.5, making it easy to integrate into whatever your next web project may be.

Installation

Install from PyPI

python-libweb can be installed using pip3:

```
pip3 install libweb
```

Or, if you're feeling adventurous, can be installed directly from github:

```
pip3 install git+https://github.com/HurricaneLabs/python-libweb.git
```

Source Code

libweb lives on GitHub, making the code easy to browse, download, fork, etc. Pull requests are always welcome! Also, please remember to star the project if it makes you happy.

Once you have cloned the repo or downloaded a tarball from GitHub, you can install libweb like this:

```
$ cd python-libweb
$ pip3 install .
```

Or, if you want to edit the code, first fork the main repo, clone the fork to your desktop, and then run the following to install it using symbolic linking, so that when you change your code, the changes will be automagically available to your app without having to reinstall the package:

```
$ cd python-libweb
$ pip install -e .
```

Did we mention we love pull requests? :)

Quickstart

If you haven't done so already, please take a moment to *install* the libweb library before continuing.

Learning by Example

Here is a simple parser from libweb's README, showing how to get started interacting with the web:

```
# spamhaus.py

from libweb.dns import DnsblService

conf = {
    "rrname": "{target}.zen.spamhaus.org",
    "rrtype": "A",
}

for result in DnsblService(opts={"target": "127.0.0.2"}, **conf):
    print(result)
```

Then, to run the sample parser:

```
$ python3 spamhaus.py
OrderedDict([('name', '2.0.0.127.zen.spamhaus.org.'), ('type', 'A'), ('class', 'IN'),␣
↪('ttl', 60), ('rdata', '127.0.0.2')])
OrderedDict([('name', '2.0.0.127.zen.spamhaus.org.'), ('type', 'A'), ('class', 'IN'),␣
↪('ttl', 60), ('rdata', '127.0.0.10')])
OrderedDict([('name', '2.0.0.127.zen.spamhaus.org.'), ('type', 'A'), ('class', 'IN'),␣
↪('ttl', 60), ('rdata', '127.0.0.4')])
$
```

More Features

Here is a more involved example demonstrating the features available in all of the HTTP-based parsers:

```
# virustotal.py

import sys

from libweb.json import JsonService


conf = {
    "url": "https://www.virustotal.com/vtapi/v2/ip-address/report",
    "params": {
```

```
        "ip": "{target}"
    },
    "auth": {
        "name": "virustotal",
        "params": ["apikey"]
    },
    "jsonpath": {
        "url": "$.detected_urls[*].url",
        "pdns": "$.resolutions[*]",
        "asn": "$.asn",
        "country": "$.country",
        "as_owner": "$.as_owner",
    }
}

creds = {
    "virustotal": [sys.argv[1]],
}

opts = {
    "target": sys.argv[2]
}

for result in JsonService(opts=opts, creds=creds, **conf):
    print(result)
```

You will need a VirusTotal API key to run this sample. Feel free to borrow the key from our sister project, Machinae. You can run the sample like so:

```
$ python virustotal.py <apikey> 209.95.50.13
OrderedDict([('asn', '29854'), ('country', 'US'), ('as_owner', 'WestHost, Inc.'), (
↪'pdns', {'hostname': 'us-newyorkcity.privateinternetaccess.com', 'last_resolved':
↪'2016-03-13 00:00:00'})])
$
```

libweb Parsers

libweb.dns

DnsService

class libweb.dns.**DnsService**(*creds=None*, *opts=None*, ***conf*)

A simple service based on DNS requests

Keyword Arguments

- **rrname** (*str*) – The record name to lookup
- **rrtype** (*str*) – The DNS record type to request
- **split** (*str, optional*) – A string with which to split the result (for TXT records)

get_resolver()

Returns a dns.resolver.Resolver instance

get_results()

Make the DNS requests and yield a structured response

get_rrname(*rrname*)
Formats the rrname using the options passed to the service

Parameters **rrname** (*str*) – A string template for rendering the rrname to be requested

make_requests()
Iterate over the requests for this service and yield the rrsets

DnsblService

class libweb.dns.**DnsblService**(*creds=None*, *opts=None*, ***conf*)
A DNS-based service where the service options are reversed for use in a DNSBL

Keyword Arguments

- **rrname** (*str*) – The record name to lookup
- **rrtype** (*str*) – The DNS record type to request
- **split** (*str, optional*) – A string with which to split the result (for TXT records)

get_rrname(*rrname*)
Formats the rrname using the options passed to the service. All options are split using "." as the separator and then the order reversed, as is required for DNSBL services (such as Spamhaus).

Parameters **rrname** (*str*) – A string template for rendering the rrname to be requested

libweb.http

HttpService

class libweb.http.**HttpService**(*creds=None*, *opts=None*, ***conf*)
A simple service based on HTTP requests. This class should not be used directly

build_request(*url*, *method='GET'*, ***kwargs*)
Apply request hooks to automatically transform request content

Override this if you need to customze the Request object generated.

get_auth(*auth*)
Find and apply authentication

Override this if you need to support additional styles of authentication

make_requests()
Iterate over configuration for multiple requests

prepare_request(*request*)
Applies session state to the request

process_params(*orig_params*)
Process parameters into usable pieces.

Override this if you provide any config parameters that may require interpreation, such as the relatime parameter

send_request(*request*, *verify_ssl=True*)
Suppress SSL if necessary and send the request

session
Return a requests Session object which sets a User-Agent header

unzip_content(*request*, **args*, ***kwargs*)
Automatically detect and decompress zip or gzip content

Override this to provide support for additional compressed content types

libweb.json

JsonService

class libweb.json.**JsonService**(*creds=None*, *opts=None*, ***conf*)
An HTTP-based service that speaks JSON. Uses jsonpath to parse the returned document.

Keyword Arguments **jsonpath** (*dict or list of dicts*) – JSONpath configuration to extract/parse data

get_data()
Make the HTTP requests and yield the data returned

get_results()
Parse the JSON and yield a structured response

libweb.regex

RegexService

class libweb.regex.**RegexService**(*creds=None*, *opts=None*, ***conf*)
An HTTP-based service that is scraped using regular expressions.

Keyword Arguments **parse** (*list*) – Regular expressions used to parse data from the service

get_html()
Make the HTTP request(s) and unescape the returned HTML

get_results()
Apply the configured regular expressions to the service's response

regexes
Compile the regular expressions provided in the configuration

libweb.xpath

XpathService

class libweb.xpath.**XpathService**(*creds=None*, *opts=None*, ***conf*)
A simple service based on HTTP requests (using XML as the reponse body)

Keyword Arguments **xpath** (*dict*) – key/value matches for extracting data

build_tree(*content*)
Uses defusedxml to parse the response into ElementTree

get_results()
Make the HTTP requests and yield a structured message

HtmlXpathService

class libweb.xpath.**HtmlXpathService**(*creds=None*, *opts=None*, ***conf*)
A simple service using XPATH with LXML to parse HTML.

Keyword argumnets:

build_tree(*content*)
Use the html5lib parser to parse HTML

Contribute to libweb

Thanks for your interest in the project! We welcome pull requests from developers of all skill levels. To get started, simply fork the master branch on GitHub to your personal account and then clone the fork into your development environment.

Steve McMaster (**iamthemcmaster** on Twitter) is the original creator of the libweb project, and currently maintains the project for Hurricane Labs.

Thanks!

Code style rules

Code style for the libweb project follows 3 simple rules:

1. Our code should be readable and easy to follow.
2. Our code should be well commented.
3. Our code should be well tested.

Tox tests include coverage testing and certain code quality tests, and it is expected that any PR's will maintain the same level of coverage and quality. No preference is given for line length, single vs double quotes, etc, as long as the code remains readable and understandable.

CHAPTER 2

libweb License

Index