ProxY Documentation

Release 1.0

João Vasconcelos

Jun 25, 2017

CONTENTS

CHAPTER

ONE

PROXY MODULE

The main module

Just starts the application. Run "python proxy.py" and configure the browser HTTP proxy to server port. By default the port is 12000, you can change on config.json file, is the port is been used the next will be try and on.

CHAPTER

TWO

LIB PACKAGE

2.1 Submodules

2.2 lib.cache module

This module writes and reads the cache DB. This module does not stops the execution flow, any error on this is transparent for the application, only is make a log entry for errors.

class `lib.cache.`**Cache**
This class has two attributes:

conn - is a connection with sqlite file **cursor** - is a cursor for DB executions

store_cache(*_request*, *_response*)
Store caches in DB. Based on requests make an entry to cache.

Parameters

- **_request** – a raw HTTP request message
- **_response** – a raw HTTP response message

Returns the return is empty

there_is_cache(*_request*)
Verify if **has**, in DB, **cache**, and returns this two informations, based on HTTP request.

Parameters **_request** – a raw HTTP request message

Returns this method return two informations: the boolean **has**, if there is cache, and a data of **cache**

2.3 lib.log module

Log module a simple class to save ProxY log in text files, located in log diretory.

class `lib.log.`**Log**(*_message*)
writes **_message** in log file.

Parameters **_message** – the log entry

2.4 lib.parser module

Parser module for HTTP messages, convert a data stream to dictionary and vice-versa.

class lib.parser.**Parser**

body_length(*_raw_http*)
Measures the lenght of body of HTTP message

Parameters **_raw_http** – a HTTP message

Returns length of HTTP body

dict_to_http(*_dict*, *_body*)
An dictionary to HTTP message converter.

Parameters

- **_dict** – dictonary with HTTP header parameters
- **_body** – data to HTTP body

Returns an HTTP message

http_to_dict(*_raw_http*)
An HTTP to dictonary converter.

Parameters **_raw_http** – a HTTP message

Returns **header_params** as dictonary with HTTP header parameters and **body** as HTTP body if exists

2.5 lib.server module

This module is the proxy server core, here the management of transmission flow is made.

class lib.server.**Server**(*_port*, *_auto_increment_port*, *_cache*)
The Server class has five attributes:

port - the server TCP port, brought from config.json file **auto_increment_port** - if 'True' the application will try next free port if the **port** is in use **do_cache** - if 'True' the application will try use cache **max_conn** - the number of simultaneous connections **buffer_size** used on receive method to control the TCP flow

conn_string(*conn*, *data*, *addr*)
This method is called when a request is received form server listening. This works de request message and pass the message to receiver.

Parameters

- **conn** – connection socket
- **data** – request data
- **addr** – socket address

proxy_server(*webserver*, *port*, *conn*, *data*, *addr*)
This method receive the request of **conn_string** method and sends to this destination. After receive the response return this to client.

Parameters

- **webserver** – host

- **port** – hosts port
- **conn** – connection socket
- **data** – request (HTTP message)
- **addr** – socket address

recvall(*sock*)
This method receive all data from TCP socke based on Server **bufer_size**.

Parameters **sock** – socket

Returns received data

recvall_old(*sock*)
Deprectaed method to receive data Based on **Contend-Length** HTTP header parameter.

Parameters **sock** – socket

Returns received data

start()
This method inicialize the server and start listening requests.

2.6 lib.setup module

This module is started by proxy.py to prepare all environment.

class lib.setup.**Setup**
When this class is instanced it verify if all environment is ready to run ProxY, if isn't this creates de necessary files:

- Log directory
- Config file
- Cache DB

config
property to get json config file in a dictionary

Returns config dictionary

lib.setup.**create_cache_db**(*_db_file*, *_log_msg*)
Creates a sqlite3 db if not exists and create CACHE table inside this

Parameters

- **_db_file** – the db file name
- **_log_msg** – message log for DB creation

lib.setup.**create_config**(*_port*, *_file*, *_log_msg*)
Create the config.json file if this don't exists.

Parameters

- **_port** – the port value for server
- **_file** – config file name
- **_log_msg** – log message for the config creation

lib.setup.**create_dir**(*_dir_name*, *_log_msg*)
This method verify if a given directory (**_dir_name**) exists and if isn't creates.

Parameters

- **_dir_name** – directory name
- **_log_msg** – log message for directory creation

2.7 Module contents

The **lib** package is just for organize the application code and contains only the classes coded exclusive for ProxY.

PYTHON MODULE INDEX

l

p

INDEX