apachelogs

Release 0.1.0

John Thorvald Wodder II

2019 May 06

CONTENTS

GitHub | PyPI | Documentation | Issues

CHAPTER

ONE

PARSING

class apachelogs.**LogParser**(*format*, *encoding='iso-8859-1'*, *errors=None*)

A class for parsing Apache access log entries in a given log format. Instantiate with a log format string, and then use the *parse()* and/or *parse_lines()* methods to parse log entries in that format.

Parameters

- **format** (*str*) – an Apache log format
- **encoding** (*str*) – The encoding to use for decoding certain strings in log entries (see *Supported Directives*); defaults to `'iso-8859-1'`. Set to `'bytes'` to cause the strings to be returned as `bytes` values instead of `str`.
- **errors** (*str*) – the error handling scheme to use when decoding; defaults to `'strict'`

Raises

- ***InvalidDirectiveError*** – if an invalid directive occurs in `format`
- ***UnknownDirectiveError*** – if an unknown directive occurs in `format`

parse(*entry*)

Parse an access log entry according to the log format and return a *LogEntry* object.

Parameters **entry** (*str*) – an access log entry to parse

Return type *LogEntry*

Raises ***InvalidEntryError*** – if `entry` does not match the log format

parse_lines(*entries*, *ignore_invalid=False*)

Parse the elements in an iterable of access log entries (e.g., an open text file handle) and return a generator of *LogEntry*s. If `ignore_invalid` is `True`, any entries that do not match the log format will be silently discarded; otherwise, such an entry will cause an *InvalidEntryError* to be raised.

Parameters

- **entries** – an iterable of `str`
- **ignore_invalid** (*bool*) – whether to silently discard entries that do not match the log format

Return type *LogEntry* generator

Raises ***InvalidEntryError*** – if an element of `entries` does not match the log format and `ignore_invalid` is `False`

class apachelogs.**LogEntry**

A parsed Apache access log entry. The value associated with each directive in the log format is stored as an attribute on the *LogEntry* object; for example, if the log format contains a `%s` directive, the *LogEntry* for a

parsed entry will have a `status` attribute containing the status value from the entry as an `int`. See *Supported Directives* for the attribute names & types of each directive supported by this library.

If the log format contains two or more directives that are stored in the same attribute (e.g., `%D` and `%{us}T`), the given attribute will contain the first non-`None` directive value.

The values of date & time directives are stored in a `request_time_fields: dict` attribute. If this `dict` contains enough information to assemble a complete (possibly naïve) `datetime.datetime`, then the *LogEntry* will have a `request_time` attribute equal to that `datetime.datetime`.

entry = None
The original logfile entry with trailing newlines removed

format = None
The entry's log format string

apachelogs.**parse**(*format*, *entry*, *encoding='iso-8859-1'*, *errors=None*)
A convenience function for parsing a single logfile entry without having to directly create a *LogParser* object.

`encoding` and `errors` have the same meaning as for *LogParser*.

apachelogs.**parse_lines**(*format*, *entries*, *encoding='iso-8859-1'*, *errors=None*, *ignore_invalid=False*)
A convenience function for parsing an iterable of logfile entries without having to directly create a *LogParser* object.

`encoding` and `errors` have the same meaning as for *LogParser*. `ignore_invalid` has the same meaning as for *LogParser.parse_lines()*.

CHAPTER

TWO

UTILITIES

apachelogs.**parse_apache_timestamp**(*s*)

Parse an Apache timestamp into a `datetime.datetime` object. The month name in the timestamp is expected to be an abbreviated English name regardless of the current locale.

```
>>> parse_apache_timestamp('[01/Nov/2017:07:28:29 +0000]')
datetime.datetime(2017, 11, 1, 7, 28, 29, tzinfo=datetime.timezone.utc)
```

Parameters **s** (*str*) – a string of the form `DD/Mon/YYYY:HH:MM:SS +HHMM` (optionally enclosed in square brackets)

Returns an aware `datetime.datetime`

Raises **ValueError** – if `s` is not in the expected format

2.1 Log Format Constants

The following standard log formats are available as string constants in this package so that you don't have to keep typing out the full log format strings:

apachelogs.**COMBINED = '%h %l %u %t "%r" %>s %b "%{Referer}i" "%{User-agent}i"'**

NCSA extended/combined log format

apachelogs.**COMBINED_DEBIAN = '%h %l %u %t "%r" %>s %O "%{Referer}i" "%{User-agent}i"'**

Like *COMBINED*, but with `%O` (total bytes sent including headers) in place of `%b` (size of response excluding headers)

apachelogs.**COMMON = '%h %l %u %t "%r" %>s %b'**

Common log format (CLF)

apachelogs.**COMMON_DEBIAN = '%h %l %u %t "%r" %>s %O'**

Like *COMMON*, but with `%O` (total bytes sent including headers) in place of `%b` (size of response excluding headers)

apachelogs.**VHOST_COMBINED = '%v:%p %h %l %u %t "%r" %>s %O "%{Referer}i" "%{User-Agent}i"'**

COMBINED_DEBIAN with virtual host & port prepended

apachelogs.**VHOST_COMMON = '%v %h %l %u %t "%r" %>s %b'**

COMMON with virtual host prepended

CHAPTER

THREE

EXCEPTIONS

exception apachelogs.**Error**
Bases: Exception

The base class for all custom exceptions raised by apachelogs

exception apachelogs.**InvalidDirectiveError**
Bases: apachelogs.errors.Error, ValueError

Raised by the *LogParser* constructor when given a log format containing an invalid or malformed directive

format = None
The log format string containing the invalid directive

pos = None
The position in the log format string at which the invalid directive occurs

exception apachelogs.**InvalidEntryError**
Bases: apachelogs.errors.Error, ValueError

Raised when a attempting to parse a log entry that does not match the given log format

entry = None
The invalid log entry

format = None
The log format string the entry failed to match against

exception apachelogs.**UnknownDirectiveError**
Bases: apachelogs.errors.Error, ValueError

Raised by the *LogParser* constructor when given a log format containing an unknown or unsupported directive

directive = None
The unknown or unsupported directive

CHAPTER

FOUR

SUPPORTED DIRECTIVES

The following table lists the log format directives supported by this library along with the names & types of the attributes at which their parsed values are stored on a *LogEntry*. The attribute names for the directives are based off of the names used internally by the Apache source code.

A directive with the < modifier (e.g., `%<s`) will be stored at `entry.original_`*`attribute_name`*, and a directive with the > modifier will be stored at `entry.final_`*`attribute_name`*

A type of `str` marked with an asterisk (*) means that the directive's values are decoded according to the `encoding` option to *LogParser*.

Any directive may evaluate to `None` when it is modified by a set of status codes (e.g., `%400,501T` or `%!200T`).

See the Apache documentation for information on the meaning of each directive.

Directive	*LogEntry* Attribute	Type
`%%`	N/A	N/A
`%a`	`entry.remote_address`	`str`
`%{c}a`	`entry.remote_client_address`	`str`
`%A`	`entry.local_address`	`str`
`%b`	`entry.bytes_sent`	`int` or `None`
`%B`	`entry.bytes_sent`	`int`
`%{`*`name`*`}C`	`entry.cookies[`*`name`*`]`	`str`* or `None`
`%D`	`entry.request_duration_microseconds`	`int`
`%{`*`name`*`}e`	`entry.env_vars[`*`name`*`]`	`str`* or `None`
`%f`	`entry.request_file`	`str`*
`%h`	`entry.remote_host`	`str`*
`%H`	`entry.request_protocol`	`str`* or `None`
`%{`*`name`*`}i`	`entry.headers_in[`*`name`*`]`	`str`* or `None`
`%I`	`entry.bytes_in`	`int`
`%k`	`entry.requests_on_connection`	`int`
`%l`	`entry.remote_logname`	`str`* or `None`
`%m`	`entry.request_method`	`str`* or `None`
`%{`*`name`*`}n`	`entry.notes[`*`name`*`]`	`str`* or `None`
`%{`*`name`*`}o`	`entry.headers_out[`*`name`*`]`	`str`* or `None`
`%O`	`entry.bytes_out`	`int`
`%p`	`entry.server_port`	`int`
`%{canonical}p`	`entry.server_port`	`int`
`%{local}p`	`entry.local_port`	`int`
`%{remote}p`	`entry.remote_port`	`int`
`%P`	`entry.pid`	`int`
`%{pid}P`	`entry.pid`	`int`

Continued on next page

Table 1 – continued from previous page

Directive	*LogEntry* Attribute	Type
%{tid}P	entry.tid	int
%q	entry.request_query	str*
%r	entry.request_line	str* or None
%R	entry.handler	str*
%s	entry.status	int or None
%S	entry.bytes_combined	int
%t	entry.request_time_fields["timestamp"]	aware datetime.datetime
%{sec}t	entry.request_time_fields["epoch"]	int
%{msec}t	entry.request_time_fields["milliepoch"]	int
%{usec}t	entry.request_time_fields["microepoch"]	int
%{msec_frac}t	entry.request_time_fields["msec_frac"]	int
%{usec_frac}t	entry.request_time_fields["usec_frac"]	int
%{*strftime_format*}t	entry.request_time_fields (See below)	(See below)
%T	entry.request_duration_seconds	int
%{ms}T	entry.request_duration_milliseconds	int
%{us}T	entry.request_duration_microseconds	int
%{s}T	entry.request_duration_seconds	int
%u	entry.remote_user	str* or None
%U	entry.request_uri	str* or None
%v	entry.virtual_host	str*
%V	entry.server_name	str*
%X	entry.connection_status	str
%^FB	entry.ttfb	int or None
%{*name*}^ti	entry.trailers_in[*name*]	str* or None
%{*name*}^to	entry.trailers_out[*name*]	str* or None

4.1 Supported `strftime` Directives

The following table lists the `strftime` directives supported for use in the parameter of a `%{*}t` directive along with the keys & types at which they are stored in the `dict` `entry.request_time_fields`. See any C documentation for information on the meaning of each directive.

A `%{*}t` directive with the `begin:` modifier (e.g., `%{begin:%Y-%m-%d}t`) will have its subdirectives stored in `entry.begin_request_time_fields` (in turn used to set `entry.begin_request_time`), and likewise for the `end:` modifier.

Directive	`request_time_fields` key	Type
%%	N/A	N/A
%a	"abbrev_wday"	str
%A	"full_wday"	str
%b	"abbrev_mon"	str
%B	"full_mon"	str
%C	"century"	int
%d	"mday"	int
%D	"date"	datetime.date
%e	"mday"	int
%F	"date"	datetime.date
%g	"abbrev_week_year"	int

Continued on next page

Table 2 – continued from previous page

Directive	request_time_fields key	Type
%G	"week_year"	int
%h	"abbrev_mon"	str
%H	"hour"	int
%I	"hour12"	int
%j	"yday"	int
%m	"mon"	int
%M	"min"	int
%n	N/A	N/A
%p	"am_pm"	str
%R	"hour_min"	datetime.time
%s	"epoch"	int
%S	"sec"	int
%t	N/A	N/A
%T	"time"	datetime.time
%u	"iso_wday"	int
%U	"sunday_weeknum"	int
%V	"iso_weeknum"	int
%w	"wday"	int
%W	"monday_weeknum"	int
%y	"abbrev_year"	int
%Y	"year"	int
%z	"timezone"	datetime.timezone or None
%Z	"tzname"	str

apachelogs parses Apache access log files. Pass it a log format string and get back a parser for logfile entries in that format. *apachelogs* even takes care of decoding escape sequences and converting things like timestamps, integers, and bare hyphens to datetime values, ints, and Nones.

CHAPTER

FIVE

INSTALLATION

`apachelogs` requires Python 3.5 or higher. Just use pip for Python 3 (You have pip, right?) to install `apachelogs` and its dependencies:

```
python3 -m pip install apachelogs
```

CHAPTER

SIX

EXAMPLES

Parse a single log entry:

```
>>> from apachelogs import LogParser
>>> parser = LogParser("%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"")
>>> # The above log format is also available as the constant `apachelogs.COMBINED`.
>>> entry = parser.parse('209.126.136.4 - - [01/Nov/2017:07:28:29 +0000] "GET / HTTP/
↪1.1" 301 521 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36␣
↪(KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36"\n')
>>> entry.remote_host
'209.126.136.4'
>>> entry.request_time
datetime.datetime(2017, 11, 1, 7, 28, 29, tzinfo=datetime.timezone.utc)
>>> entry.request_line
'GET / HTTP/1.1'
>>> entry.final_status
301
>>> entry.bytes_sent
521
>>> entry.headers_in["Referer"] is None
True
>>> entry.headers_in["User-agent"]
'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)␣
↪Chrome/57.0.2987.133 Safari/537.36'
```

Parse a file full of log entries:

```
>>> with open('/var/log/apache2/access.log') as fp:
...     for entry in parser.parse_lines(fp):
...         print(str(entry.request_time), entry.request_line)
...
2019-01-01 12:34:56-05:00 GET / HTTP/1.1
2019-01-01 12:34:57-05:00 GET /favicon.ico HTTP/1.1
2019-01-01 12:34:57-05:00 GET /styles.css HTTP/1.1
# etc.
```

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

a

INDEX