haralyzer Documentation

Release 2.0.0

Justin Crown

Oct 20, 2021

Contents

Haralyzer is a Python framework for using HAR files to analyze web pages.

CHAPTER 1

Overview

The haralyzer module contains three classes for analyzing web pages based on a HAR file. `HarParser()` represents a full file (which might have multiple pages). `HarPage()` represents a single page from said file. `HarEntry()` represents an entry in a `HarPage()`, and there are are multiple entries per page. Each `HarEntry` has a request and response that contains items such as the headers, status code, timings, etc

`HarParser` has a couple of helpful methods for analyzing single entries from a HAR file, but most of the pertinent functions are inside of the page object.

`haralyzer` was designed to be easy to use, but you can also access more powerful functions directly.

CHAPTER 2

Basic Usage

2.1 HarParser

The `HarParser` takes a single argument of a `dict` representing the JSON of a full HAR file. It has the same properties of the HAR file, EXCEPT that each page in HarParser.pages is a HarPage object.

```
import json
from haralyzer import HarParser, HarPage

with open('har_data.har', 'r') as f:
    har_parser = HarParser(json.loads(f.read()))

print har_parser.browser
# {u'name': u'Firefox', u'version': u'25.0.1'}

print har_parser.hostname
# 'humanssuck.net'

for page in har_parser.pages:
    assert isinstance(page, HarPage, None)
    # returns True for each
```

2.2 HarPage

The `HarPage` object contains most of the goods you need to easily analyze a page. It has helper methods that are accessible, but most of the data you need is in properties for easy access. You can create a HarPage object directly by giving it the page ID (yes, I know it is stupid, it's just how HAR is organized), and either a `HarParser` with *har_parser=parser*, or a `dict` representing the JSON of a full HAR file (see example in HarParser) with *har_data=har_data*.

```
import json
from haralyzer import HarPage

with open('har_data.har', 'r') as f:
    har_page = HarPage('page_3', har_data=json.loads(f.read()))

### GET BASIC INFO
har_page.hostname
# 'humanssuck.net'
har_page.url
# 'http://humanssuck.net/about/'

### WORK WITH LOAD TIMES (all load times are in ms) ###

# Get image load time in milliseconds as rendered by the browser
har_page.image_load_time
# 713

# We could do this with 'css', 'js', 'html', 'audio', or 'video'

### WORK WITH SIZES (all sizes are in bytes) ###

# Get the total page size (with all assets)
har_page.page_size
# prints 2423765

# Get the total image size
har_page.image_size
# prints 733488
# We could do this with 'css', 'js', 'html', 'audio', or 'video'

# Get duplicate requests (requests to the same URL 2 or more times) if any
# har_page.duplicate_url_request
# Returns a dict where the key is a string of the URL and the value is an int of the␣
↪number
# of requests to that URL. Only requests with 2 or more are included.
# {'https://test.com/': 3}

# Get the transferred sizes (works only with HAR files, generated with Chrome)
har_page.page_size_trans
har_page.image_size_trans
har_page.css_size_trans
har_page.text_size_trans
har_page.js_size_trans
har_page.audio_size_trans
har_page.video_size_trans
```

IMPORTANT NOTE - Technically, the *page_id* attribute of a single entry in a HAR file is optional. As such, if your HAR file contains entries that do not map to a page, an additional page will be created with an ID of *unknown*. This "fake page" will contain all such entries. Since it is not a real page, it does not have attributes for things like time to first byte or page load, and will return *None*.

2.3 HarEntry

The `HarEntry()` object contains useful information for each request. The main purpose is to have easy of use as it has a lot of attributes. Each entry also contains a `Request()` and `Response()` which are styled off of the requests

library.:

```
import json
from haralyzer import HarPage

with open("humanssuck.net.har", 'r') as f:
    har_page = HarPage('page_3', har_data=json.loads(f.read()))

### GET BASIC INFO
print(har_page.hostname)
# 'humanssuck.net'
print(har_page.url)
# 'http://humanssuck.net/'

### GET LIST OF ENTRIES
print(har_page.entries)
# [HarEntry for http://humanssuck.net/, HarEntry for http://humanssuck.net/test.css, .
↪..]

### WORKING WITH ENTRIES
single_entry = har_page.entries[0]

### REQUEST HEADERS
print(single_entry.request.headers)
# {'host': 'humanssuck.net', 'user-agent': 'Mozilla/5.0 (X11; Linux i686 on x86_64;␣
↪rv:25.0) Gecko/20100101 Firefox/25.0', ...}

### RESPONSE HEADERS
print(single_entry.response.headers)
# {'server': 'nginx', 'date': 'Mon, 23 Feb 2015 03:28:12 GMT', ...}

### RESPONSE CODE
print(single_entry.response.status)
# 200

# GET THE VALUE OF A REQUEST OR RESPONSE HEADER
print(single_entry.request.headers.get("accept"))
# text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

# ALL ATTRIBUTES OF A ENTRY

single_entry.cache
# Dictionary of cached content
single_entry.cookies
# List of combined cookies for request and response
single_entry.pageref
# String of the pageref
single_entry.port
# Integer of the port number for the server
single_entry.request
# Request object
single_entry.response
# Response object
single_entry.secure
# Bool if secure is set
single_entry.serverAddress
# String of the server IP
single_entry.startTime
```

(continues on next page)

(continued from previous page)

```
# Datetime of the start time
single_entry.time
# Integer of total time for entry
single_entry.timings
# Dictionary of the timings for a request
single_entry.url
# String of the request url

# ALL ATTRIBUTES OF A REQUEST

single_entry.request.accept
# String of the ``Accept`` header
single_entry.request.bodySize
# Integer of the body size for the request
single_entry.request.cacheControl
# String of the ``Cache-Control`` header
single_entry.request.cookies
# List of cookies
single_entry.request.encoding
# String of the ``Accept-Encoding`` header
single_entry.request.headers
# Dictionary of headers
single_entry.request.headersSize
# Integer of the size of the headers
single_entry.request.host
# String of the ``Host`` header
single_entry.request.httpVersion
# String of the http version used
single_entry.request.language
# String of the ``Accept-Language`` header
single_entry.request.method
# String of the HTTP method used
single_entry.request.queryString
# List of query string used
single_entry.request.url
# String of the URL
single_entry.request.userAgent
# String of the User-Agent

### ALL ATTRIBUTES OF A RESPONSE
single_entry.response.bodySize
# Integer of the body size for the response
single_entry.response.cacheControl
# String of the `Cache-Control` header
single_entry.response.contentSecurityPolicy
# String of the `Content-Security-Policy`` header
single_entry.response.contentSize
# Integer of the content size
single_entry.response.contentType
# String of the ``content-type`` header
single_entry.response.date
# String of the ``date`` header
single_entry.response.headers
# Dictionary of headers
single_entry.response.headersSize
# Integer of the size of the headers
single_entry.response.httpVersion
```

(continues on next page)

(continued from previous page)

```
# String of the http version used
single_entry.response.lastModified
# String of the ``last-modified`` header
single_entry.response.mimeType
# String of the mimeType of the content
single_entry.response.redirectURL
# String of the redirect URL or None
single_entry.response.status
# Integer of th HTTP status code
single_entry.response.statusText
# String of HTTP status
single_entry.response.text
# String of content received

# You are still able to access items like a dictionary.
print(single_entry["connection"])
# "80"
```

CHAPTER 3

Advanced

3.1 Advanced HARPage

`HarPage` includes a lot of helpful properties, but they are all easily produced using the public methods of `HarParser` and `HarPage`:

```
import json
from haralyzer import HarPage

with open('har_data.har', 'r') as f:
    har_page = HarPage('page_3', har_data=json.loads(f.read()))

### ACCESSING FILES ###

# You can get a JSON representation of all assets using HarPage.entries #
for entry in har_page.entries:
    if entry['startedDateTime'] == 'whatever I expect':
        ... do stuff ...

# It also has methods for filtering assets #
# Get a collection of entries that were images in the 2XX status code range #
entries = har_page.filter_entries(content_type='image.*', status_code='2.*')
# This method can filter by:
# * content_type ('application/json' for example)
# * status_code ('200' for example)
# * request_type ('GET' for example)
# * http_version ('HTTP/1.1' for example)
# * load_time__gt (Takes an int representing load time in milliseconds.
#   Entries with a load time greater than this will be included in the
#   results.)
# Parameters that accept a string use a regex by default, but you can also force a␣
↪literal string match by passing regex=False

# Get the size of the collection we just made #
```

(continues on next page)

(continued from previous page)

```
collection_size = har_page.get_total_size(entries)

# We can also access files by type with a property #
for js_file in har_page.js_files:
    ... do stuff ....

### GETTING LOAD TIMES ###

# Get the BROWSER load time for all images in the 2XX status code range #
load_time = har_page.get_load_time(content_type='image.*', status_code='2.*')

# Get the TOTAL load time for all images in the 2XX status code range #
load_time = har_page.get_load_time(content_type='image.*', status_code='2.*',␣
↪asynchronous=False)
```

All of the HarPage methods above leverage stuff from the HarParser, some of which can be useful for more complex operations. They either operate on a single entry (from a HarPage) or a `list` of entries:

```
import json
from haralyzer import HarParser

with open('har_data.har', 'r') as f:
    har_parser = HarParser(json.loads(f.read()))

for page in har_parser.pages:
    for entry in page.entries:
        ### MATCH HEADERS ###
        if har_parser.match_headers(entry, 'Content-Type', 'image.*'):
            print 'This would appear to be an image'
        ### MATCH REQUEST TYPE ###
        if har_parser.match_request_type(entry, 'GET'):
            print 'This is a GET request'
        ### MATCH STATUS CODE ###
        if har_parser.match_status_code(entry, '2.*'):
            print 'Looks like all is well in the world'
```

3.2 Asset Timeline

The last helper function of `HarParser` requires it's own section, because it is odd, but can be helpful, especially for creating charts and reports.

It can create an asset timeline, which gives you back a `dict` where each key is a `datetime` object, and the value is a `list` of assets that were loading at that time. Each value of the `list` is a `dict` representing an entry from a page.

It takes a `list` of entries to analyze, so it assumes that you have already filtered the entries you want to know about:

```
import json
from haralyzer import HarParser

with open('har_data.har', 'r') as f:
    har_parser = HarParser(json.loads(f.read()))

### CREATE A TIMELINE OF ALL THE ENTRIES ###
entries = []
```

(continues on next page)

(continued from previous page)

```
for page in har_parser.pages:
    for entry in page.entries:
        entries.append(entry)

timeline = har_parser.create_asset_timeline(entries)

for key, value in timeline.items():
    print(type(key))
    # <type 'datetime.datetime'>
    print(key)
    # 2015-02-21 19:15:41.450000-08:00
    print(type(value))
    # <type 'list'>
    print(value)
    # Each entry in the list is an asset from the page
    # [{u'serverIPAddress': u'157.166.249.67', u'cache': {}, u'startedDateTime': u
↪'2015-02-21T19:15:40.351-08:00', u'pageref': u'page_3', u'request': {u'cookies':....
↪.......................
```

With this, you can examine the timeline for any number of assets. Since the key is a `datetime` object, this is a heavy operation. We could always change this in the future, but for now, limit the assets you give this method to only what you need to examine.

3.3 haralyzer package

3.3.1 haralyzer.assets module

Provides all of the main functional classes for analyzing HAR files

class haralyzer.assets.**HarEntry**(*entry: dict*)

Bases: *haralyzer.mixins.MimicDict*

An object that represent one entry in a HAR Page

cache

Returns Cached objects

Return type str

cookies

Returns Request and Response Cookies

Return type list

pageref

Returns Page for the entry

Return type str

port

Returns Port connection was made to

Return type int

request

Returns Request of the entry

Return type *Request*

response

Returns Response of the entry

Return type *Response*

secure

Returns Connection was secure

Return type bool

serverAddress

Returns IP Address of the server

Return type str

startTime

Start time and date

Returns Start time of entry

Return type Optional[datetime.datetime]

status

Returns HTTP Status Code

Return type int

time

Returns Time taken to complete entry

Return type int

timings

Returns Timing of the page load

Return type dict

url

Returns URL of Entry

Return type str

class haralyzer.assets.**HarPage**(*page_id: str*, *har_parser: Optional[haralyzer.assets.HarParser] = None*, *har_data: dict = None*)

Bases: object

An object representing one page of a HAR resource

actual_page

Returns the first entry object that does not have a redirect status, indicating that it is the actual page we care about (after redirects).

Returns First entry of the page

Return type *HarEntry*

audio_files

All audio files for a page

Returns Audio entries for a page

Return type List[*HarEntry*]

audio_load_time
Audio load time

Returns Load time for audio on a page

Return type int

audio_size
Size of audio files from the page

Returns Size of audio files on the page

Return type int

audio_size_trans
Audio transfer size

Returns Size of transfer data for audio

Return type int

content_load_time
Content load time

Returns Load time for all content

Return type int

css_files
All CSS files for a page

Returns CSS entries for a page

Return type List[*HarEntry*]

css_load_time
CSS load time

Returns Load time for CSS on a page

Return type int

css_size
Size of CSS files from the page

Returns Size of CSS files on the page

Return type int

css_size_trans
CSS transfer size

Returns Size of transfer data for CSS

Return type int

duplicate_url_request
Returns a dict of urls and its number of repetitions that are sent more than once

Returns URLs and the amount of times they were duplicated

Return type dict

entries

Returns All entries that make up the page

Return type List[*HarEntry*]

filter_entries(*request_type: str = None*, *content_type: str = None*, *status_code: str = None*, *http_version: str = None*, *load_time__gt: int = None*, *regex: bool = True*) → List[haralyzer.assets.HarEntry]

Generate a list of entries with from criteria

Parameters

- **request_type** (*str*) – The request type (i.e. - GET or POST)
- **content_type** (*str*) – Regex to use for finding content type
- **status_code** (*str*) – The desired status code
- **http_version** (*str*) – HTTP version of request
- **load_time__gt** (*int*) – Load time in milliseconds. If provided, an entry whose load time is less than this value will be excluded from the results.
- **regex** (*bool*) – Whether to use regex or exact match.

Returns List of entry objects based on the filtered criteria.

Return type List[*HarEntry*]

get_load_time(*request_type: str = None*, *content_type: str = None*, *status_code: str = None*, *asynchronous: bool = True*, ***kwargs*) → int

This method can return the TOTAL load time for the assets or the ACTUAL load time, the difference being that the actual load time takes asynchronous transactions into account. So, if you want the total load time, set asynchronous=False.

EXAMPLE:

I want to know the load time for images on a page that has two images, each of which took 2 seconds to download, but the browser downloaded them at the same time.

self.get_load_time(content_types=['image']) (returns 2) self.get_load_time(content_types=['image'], asynchronous=False) (returns 4)

Parameters

- **request_type** (*str*) – The request type (i.e. - GET or POST)
- **content_type** (*str*) – Regex to use for finding content type
- **status_code** (*str*) – The desired status code
- **asynchronous** (*bool*) – Whether to separate load times

Returns Total load time

Return type int

get_requests

Returns a list of GET requests, each of which is a HarEntry object

Returns All GET requests

Return type List[*HarEntry*]

static get_total_size(*entries: List[HarEntry]*) → int

Returns the total size of a collection of entries.

Parameters **entries** – list of entries to calculate the total size of.

Returns Total size of entries

Return type int

static get_total_size_trans(*entries: List[HarEntry]*) → int
Returns the total size of a collection of entries - transferred.

NOTE: use with har file generated with chrome-har-capturer

Parameters **entries** – list of entries to calculate the total size of.

Returns Total size of entries that was transferred

Return type int

hostname

Returns Hostname of the initial request

Return type str

html_files
All HTML files for a page

Returns HTML entries for a page

Return type List[*HarEntry*]

html_load_time
HTML load time

Returns Load time for HTML on a page

Return type int

image_files
All image files for a page

Returns Image entries for a page

Return type List[*HarEntry*]

image_load_time
Image load time

Returns Load time for images on a page

Return type int

image_size
Size of image files from the page

Returns Size of image files on the page

Return type int

image_size_trans
Image transfer size

Returns Size of transfer data for images

Return type int

initial_load_time
Initial load time

Returns Initial load time of the page

Return type int

js_files
All JS files for a page

Returns JS entries for a page

Return type List[*HarEntry*]

js_load_time
JS load time

Returns Load time for JS on a page

Return type int

js_size
Size of JS files from the page

Returns Size of JS files on the page

Return type int

js_size_trans
JS transfer size

Returns Size of transfer data for JS

Return type int

page_load_time
Load time of the page

Returns Load time for the page

Return type int

page_size
Size of the page

Returns Size of the page

Return type int

page_size_trans
Page transfer size

Returns Size of transfer data for the page

Return type int

post_requests
Returns a list of POST requests, each of which is an HarEntry object

Returns All POST requests

Return type List[*HarEntry*]

text_files
All text files for a page

Returns Text entries for a page

Return type List[*HarEntry*]

text_size
Size of text files from the page

Returns Size of text files on the page

Return type int

text_size_trans
Text transfer size

Returns Size of transfer data for text

Return type int

time_to_first_byte

Returns Time to first byte of the page request in ms

Return type int

url
The absolute URL of the initial request.

Returns URL of first request

Return type str

video_files
All video files for a page

Returns Video entries for a page

Return type List[*HarEntry*]

video_load_time
Video load time

Returns Load time for video on a page

Return type int

video_size
Size of video files from the page

Returns Size of video files on the page

Return type int

video_size_trans
Video transfer size

Returns Size of transfer data for images

Return type int

class haralyzer.assets.**HarParser**(*har_data: dict = None*)
Bases: object

A Basic HAR parser that also adds helpful stuff for analyzing the performance of a web page.

browser
Browser of Har File

Returns Browser of the Har File

Return type str

static create_asset_timeline(*asset_list: List[HarEntry]*) → dict
Returns a *dict* of the timeline for the requested assets. The key is a datetime object (down to the millisecond) of ANY time where at least one of the requested assets was loaded. The value is a *list* of ALL assets that were loading at that time.

Parameters **asset_list** (*List[HarEntry]*) – The assets to create a timeline for.

Returns Milliseconds and assets that were loaded

Return type dict

creator
Creator of Har File. Usually the same as the browser but not always

Returns Program that created the HarFile

Return type str

hostname
Hostname of first page

Returns Hostname of the first known page

Return type str

static match_content_type(*entry: haralyzer.assets.HarEntry*, *content_type: str*, *regex: bool = True*) → bool
Matches the content type of a request using the mimeType metadata.

Parameters

- **entry** (HarEntry) – Entry to analyze
- **content_type** (*str*) – Regex to use for finding content type
- **regex** (*bool*) – Whether to use regex or exact match.

Returns Mime type matches

Return type bool

static match_headers(*entry: haralyzer.assets.HarEntry*, *header_type: str*, *header: str*, *value: str*, *regex: bool = True*) → bool
Function to match headers.

Since the output of headers might use different case, like:

'content-type' vs 'Content-Type'

This function is case-insensitive

Parameters

- **entry** (HarEntry) – Entry to analyze
- **header_type** (*str*) – Header type. Valid values: 'request', or 'response'
- **header** (*str*) – The header to search for
- **value** (*str*) – The value to search for
- **regex** (*bool*) – Whether to use regex or exact match

Returns Whether a match was found

Return type bool

static match_http_version(*entry: haralyzer.assets.HarEntry*, *http_version: str*, *regex: bool = True*) → bool
Helper function that returns entries with a request type matching the given *request_type* argument.

Parameters

- **entry** (HarEntry) – Entry to analyze
- **http_version** (*str*) – HTTP version type to match

- **regex** (*bool*) – Whether to use a regex or string match

Returns HTTP version matches

Return type bool

static match_request_type(*entry: haralyzer.assets.HarEntry*, *request_type: str*, *regex: bool = True*) → bool

Helper function that returns entries with a request type matching the given *request_type* argument.

Parameters

- **entry** (HarEntry) – Entry to analyze
- **request_type** (*str*) – Request type to match
- **regex** (*bool*) – Whether to use a regex or string match

Returns Request method matches

Return type bool

static match_status_code(*entry: haralyzer.assets.HarEntry*, *status_code: str*, *regex: bool = True*) → bool

Helper function that returns entries with a status code matching then given *status_code* argument.

NOTE: This is doing a STRING comparison NOT NUMERICAL

Parameters

- **entry** (HarEntry) – Entry to analyze
- **status_code** (*str*) – Status code to search for
- **regex** (*bool*) – Whether to use a regex or string match

Returns Status code matches

Return type bool

pages

This is a list of HarPage objects, each of which represents a page from the HAR file.

Returns HarPages in the file

Return type List[*HarPage*]

version

HAR Version

Returns Version of HAR used

Return type str

haralyzer.assets.**convert_to_entry**(*func*)

Wrapper function for converting dicts of entries to HarEnrty Objects

3.3.2 haralyzer.errors module

Custom exceptions for good ol haralyzer.

exception haralyzer.errors.**PageNotFoundError**

Bases: AttributeError

Error raised in the Page is not found

3.3.3 haralyzer.http module

Creates the Request and Response sub class that are used by each entry

class haralyzer.http.**Request**(*entry: dict*)

Bases: *haralyzer.mixins.HttpTransaction*

Request object for an HarEntry

accept

Returns HTTP Accept header

Return type str

bodySize

Returns Body size of the request

Return type int

cacheControl

Returns HTTP CacheControl header

Return type str

cookies

Returns Cookies from the request

Return type list

encoding

Returns HTTP Accept-Encoding Header

Return type str

headersSize

Returns Headers size from the request

Return type int

host

Returns HTTP Host header

Return type str

httpVersion

Returns HTTP version used in the request

Return type str

language

Returns HTTP language header

Return type str

method

Returns HTTP method of the request

Return type str

queryString

Returns Query string from the request

Return type list

url

Returns URL of the request

Return type str

userAgent

Returns User Agent

Return type str

class haralyzer.http.**Response**(*url: str*, *entry: dict*)

Bases: *haralyzer.mixins.HttpTransaction*

Response object for a HarEntry

bodySize

Returns Body Size

Return type int

cacheControl

Returns Cache Control Header

Return type str

contentSecurityPolicy

Returns Content Security Policy Header

Return type str

contentSize

Returns Content Size

Return type int

contentType

Returns Content Type

Return type str

date

Returns Date of response

Return type str

headersSize

Returns Header size

Return type int

httpVersion

Returns HTTP Version

Return type str

lastModified

Returns Last modified time

Return type str

mimeType

Returns Mime Type of response

Return type str

redirectURL

Returns Redirect URL

Return type Optional[str]

status

Returns HTTP Status

Return type int

statusText

Returns HTTP Status Text

Return type str

text

Returns Response body

Return type str

3.3.4 haralyzer.mixins module

Mixin Objects that allow for shared methods

class haralyzer.mixins.**GetHeaders**
Bases: object

Mixin to get a header

get_header_value(*name: str*) → Optional[str]
Returns the header value of the header defined in name

Parameters **name** (*str*) – Name of the header to get the value of

Returns Value of the header

Return type Optional[str]

class haralyzer.mixins.**HttpTransaction**(*entry: dict*)
Bases: *haralyzer.mixins.GetHeaders*, *haralyzer.mixins.MimicDict*

Class the represents a request or response

headers
Headers from the entry

Returns Headers from both request and response

Return type list

class haralyzer.mixins.**MimicDict**
Bases: collections.abc.MutableMapping

Mixin for functions to mimic a dictionary for backward compatibility

3.3.5 haralyzer.multihar module

Contains the mutlihar parse object

class haralyzer.multihar.**MultiHarParser**(*har_data*, *page_id=None*, *decimal_precision=0*)
Bases: object

An object that represents multiple HAR files OF THE SAME CONTENT. It is used to gather overall statistical data in situations where you have multiple runs against the same web asset, which is common in performance testing.

asset_types
Mimic the asset types stored in HarPage

Returns Asset types from HarPage

Return type dict

audio_load_time

Returns Aggregate audio load time for all pages. Can be an *int* or *float* depending on the self.decimal_precision

Return type int, float

css_load_time

Returns Aggregate css load time for all pages. Can be an *int* or *float* depending on the self.decimal_precision

Return type int, float

get_load_times(*asset_type: str*) → list
Just a list of the load times of a certain asset type for each page

Parameters **asset_type** (*str*) – The asset type to return load times for

Returns List of load times

Return type list

get_stdev(*asset_type: str*) → Union[int, float]
Returns the standard deviation for a set of a certain asset type.

Parameters **asset_type** (*str*) – The asset type to calculate standard deviation for.

Returns Standard deviation, which can be an *int* or *float* depending on the self.decimal_precision

Return type int, float

html_load_time

Returns Aggregate html load time for all pages. Can be an *int* or *float* depending on the self.decimal_precision

Return type int, float

image_load_time

Returns Aggregate image load time for all pages. Can be an *int* or *float* depending on the self.decimal_precision

Return type int, float

js_load_time

Returns Aggregate javascript load time. Can be an *int* or *float* depending on the self.decimal_precision

Return type int, float

page_load_time

Returns Average total load time for all runs (not weighted). Can be an *int* or *float* depending on the self.decimal_precision

Return type int, float

pages

Aggregate pages of all the parser objects.

Returns All the pages from parsers

Return type List[*haralyzer.assets.HarPage*]

time_to_first_byte

Returns The aggregate time to first byte for all pages. Can be an *int* or *float* depending on the self.decimal_precision

Return type int, float

video_load_time

Returns Aggregate video load time for all pages. Can be an *int* or *float* depending on the self.decimal_precision

Return type int, float

3.4 Indices and tables

- genindex
- modindex
- search

Python Module Index

h

Index

H

I

J

L

M

P

Q

R

S

T

U

V