phantompy Documentation

Release 0.10

Andrey Antukh

May 16, 2016

Contents

Release v0.10.

phantompy is a BSD Licensed, headless WebKit engine with a powerful pythonic api.

CHAPTER 1

Introduction

This package has two main components:

- C/C++ Library which exposes some portions of the WebKit API from Qt5 (libphantompy).
- Python bindings for **libphantompy**

CHAPTER 2

Features

Note: One or more of the listed features are not implemented. And others are only a proof of concept and have a limited API.

- Live DOM access in a pythonic way. (Proof of concept API implemented)
- Totally configurable (currently only limited config options are exposed to Python)
- Access to a frames tree created by a page.
- Access to background requests of one page.

CHAPTER 3

User guide

3.1 Installation

3.1.1 Distribute & Pip

You can install **phantompy** with *pip* (see *C/C++ library installation notes*):

```
pip install phantompy
```

3.1.2 Get the Code

Also, you can download the latest version from github and install it manually:

```
git clone https://github.com/niwibe/phantompy
cd phantompy
python setup.py install
```

3.1.3 Additional notes

C/C++ Library Notes

The core part of **phantompy** is a c/c++ library that uses qt5 for access to WebKit engine (with Qt5WebKit).

Before use of python library/wrapper for libphantompy, you need install system-wide this c/c++ library.

For compile it, you need:

- Gcc >= 4.8 or clang++ >= 3.2 (simply not tested with previous versions)
- Qt5 (Core, Network, WebKit, Widgets)
- CMake >= 2.8.4

Compile and install instructions:

```
cd build
cmake ..
make
sudo make install
```

NOTES:

- This library has limited tested environtments. If you can compile in other environments would be helpful if you notified me of it.
- This library does not works properly on OSX, any help is welcome.

Ubuntu instalation notes

I have not been able to install all modules needed by phantompy on ubuntu. Ubuntu sucks. If you get compile, notify me so I can fill this gap with useful information.

Python Compatibility

This python bindings library is build with python3 in mind and has some layer of compatibility with python2.

CHAPTER 4

Developers Api

4.1 Python Api

This is the technical documentation of the python bindins **libphantompy**'s API.

4.1.1 Context & Config

Context class represents a context singleton pointer that contains an instance of a QT application, an interface for some WebKit engine configuration options and some actions (e.g. clear cache memory).

class phantompy.context.**Config**

WebKit engine configuration interface.

This class should onlu be accessed throught *Context* instance and can not be instanciated as is.

This config interface exposes these properties:

- 'load_images' (bool)
- 'javascript' (bool)
- 'dns_prefetching' (bool)
- 'plugins' (bool)
- 'private_browsing' (bool)
- 'offline_storage_db' (bool)
- 'offline_storage_quota' (int)
- 'frame_flattening' (bool)
- 'local_storage' (boo)

And some additional methods:

set_max_pages_in_cache(*num*)

Set webkit page number to maintain in cache.

set_object_cache_capacity(*min_dead_capacity*, *max_dead*, *total_capacity*)

Set webkit object cache capacity.

class phantompy.context.**Context**

clear_cookies()
Clear all cookies.

clear_memory_caches()
Clear all memory used by webkit engine.

conf
Return a *Config* instance.

get_all_cookies()
Get all available cookies.

process_events(*timeout=200*)
Method like a *time.sleep* but while waiths a timeout process qt events.

set_cookie(*name*, *value*, *domain*, *path='/'*, *expires=None*)
Generic method for set cookie to the cookiejar instance of WebKit Engine.

Parameters

- **name** (*str*) – cookie name.
- **value** (*str*) – cookie value.
- **domain** (*str*) – cookie domain.
- **path** (*str*) – cookie path (default '/')
- **path** – cookie expires date (this must be datetime, date, timedelta or str type.

Return type None

set_cookies(*cookies*)
Set a list of cookies.

phantompy.context.**context**()
Get or create instance of context (singleton).

phantompy.context.**destroy_context**()
Destroy context singleton instance.

4.1.2 Web Element

Live DOM manipulation and transversing api.

class phantompy.webelements.**WebElement**(*el_ptr*, *frame*)
Class that represents a live dom element on webkit engine.

append(*element*)
Append element or raw html to the current dom element.

Parameters **element** – Unicode string with html or WebElement instance.

Return type None

Example:

```
>>> element = p.cssselect("body > section")[0]
>>> element.append("<span>{0}</span>".format("FOO"))
```

append_after(*element*)
Same as *append()* but appends outside the current dom element.

cssselect(**args*, ***kwargs*)
Find all descendent elements by css selector like jQuery.

Parameters **selector** (*str*) – jQuery like selector

Return type list

cssselect_first(*selector*)

frame
Returns a frame instance of this element.

get_attr(*name*, ***kwargs*)

get_attrs(**args*, ***kwargs*)
Get all attributes as python dict. :rtype: dict

get_classes()
Returs a list of classes that hace current dom element.

Example:

```
>>> element = p.cssselect("section")[0]
>>> element.get_classes()
["main", "main-section"]
```

has_attr(*attrname*)
Method that checks the existence of one concrete attribute by name.

Parameters **attribute** (*str*) – attribute name

Return type bool

has_attrs()
Method that checkos of existence of any attrs. Returns a True value if a current dom element has any attribute.

Return type bool

has_class(*classname*)
Method that checks the existense of some class in a current dom element.

Parameters **classname** (*str*) – class name

Return type bool

Example:

```
>>> element = p.cssselect("section")[0].has_class("foo")
False
```

inner_html()
Get inner dom structure as html.

Return type str

inner_text()
Get inner dom structure as text, stripping all html tags.

Return type str

is_none()
Checks if a current dom element is empty or not.

Return type bool

name
Returns a tagname.

next()
Get a next element in the same level of dom.

Return type *WebElement*

prev()
Get a previous element in the same level of dom.

Return type *WebElement*

ptr
Returns a pointer to internal C++ instance object.

remove()
Remove the current element from the living dom and make this element as empty element.

remove_attr(*attrname*)
Remove attribute by name.

Parameters **attrname** (*str*) – attribute name.

Return type None

remove_childs()
Remove all childs of the current dom.

remove_class(*classname*)
Method that removes a class from a current dom node. If a class does not exists, this method does nothink.

Parameters **classname** (*str*) – class name

Return type None

replace(*element*)
Replace the current element with other.

Parameters **element** – Unicode string with html or *WebElement* instance.

Return type None

set_attr(*name*, *value*)

set_attrs(*attrs*)

wrap(*element*)
Wraps the current element with other element.

Parameters **element** – Unicode string with html or *WebElement* instance.

Return type None

Example:

```
>>> element = p.cssselect("a")[0]
>>> element.wrap("<div/>")
```

4.2 C Api

This is the technical documentation of the C api, compatible with ctypes. This API is an intermediate layer between the C++ library and Python. The python bindings use this API directly via ctypes.

4.2.1 Context

Context is a singleton object that mantains Qt5 application instance in memory and exposes some QtWebKit configuration options.

The current API is incomplete and in the near future it will expose lots of configuration options for the WebKit engine.

void* **ph_context_init**()

Return type pointer to a Context instance.

This method returns a new Context instance. Context is a singleton, and if you repeatedly call this method, it always returns a pointer to the same object.

void **ph_context_free**()
Destroy a current instance of Context. If you call this method repeatedly, the behavior is unspecified.

void **ph_context_clear_memory_cache**()
Clears the memory used by webkit for the current thread.

void **ph_context_set_object_cache_capacity**(int *cacheMinDeadCapacity*, int *cacheMaxDead*, int *totalCapacity*)
Specifies the capabilities of the memory cache for dead objects such as stylesheets or scripts.

Parameters

- **cacheMinDeadCapacity** (*int*) – specifies the minimum number of bytes that dead objects should consume when the cache is under pressure.
- **cacheMaxDead** (*int*) – is the maximum number of bytes that dead objects should consume when the cache is not under pressure.
- **totalCapacity** (*int*) – specifies the maximum number of bytes that the cache should consume overall.

void **ph_context_set_max_pages_in_cache**(int *num*)
Sets the maximum number of pages to hold in the memory page cache to pages.

Parameters

- **num** (*int*) – number of pages to hold in the memory.

char* **ph_context_get_all_cookies**()
Returns a cookies array with all the available cookies in a current cookiejar singleton, encoded as JSON.

void **ph_context_set_cookies**(const char **cookies*)
Add or overwrite cookies on the current cookiejar.

void **ph_context_clear_cookies**()
Clear all cookies available in a current cookiejar instance.

void **ph_context_set_boolean_config**(int *key*, int *value*)
Set WebKit configuration parameter.

void **ph_context_set_int_config**(int *key*, int *value*)
Set WebKit configuration parameter.

int32_t **ph_context_get_boolean_config**(int *key*)
Get WebKit configuration parameter value.

int32_t **ph_context_get_int_config**(int *key*)
Get WebKit configuration parameter value.

4.2.2 Web Page

This api exposes a **web page** and its frames functionality.

void* **ph_page_create**()
Create a new instance of a Page object and returns its pointer.

Return type pointer to a Page object instance.

void **ph_page_free**(void **page*)
Destroy a Page instance and frees the memory used by it.

Parameters

- **page** (*void**) – Page instance pointer returned by *ph_page_create()*

void **ph_page_set_viewpoint_size**(void **page*, int *x*, int *y*)
Set view point size to a page.

char ***ph_page_get_cookies**(void **page*)
Get the cookies generated by the page.

void **ph_page_set_initial_cookies**(void **page*, const char **cookies*)
Set initial cookies to the page.

int32_t **ph_page_load**(void **page*, char **url*)
Load contents for a current page.

int32_t **ph_page_is_loaded**(void **page*)
Checks if the Page is loaded.

char* **ph_page_get_requested_urls**(void **page*)
Get a list of URLs requested in background when the page is loaded. The result is encoded as JSON.

char* **ph_page_get_reply_by_url**(void **page*, const char **url*)
Get downloaded data from one of the background requests.

void* **ph_page_main_frame**(void **page*)
Get main frame from Page.

void **ph_frame_free**(void **frame*)
Release a frame memory.

char* **ph_frame_to_html**(void **frame*)
Get frame content as HTML.

char* **ph_frame_evaluate_javascript**(void **frame*, char* *js*)
Evaluate JavaScript in a current frame and return its result as string.

void* **ph_frame_capture_image**(void **frame*, const char **format*, int *quality*)

void **ph_image_free**(void **image*)

int64_t **ph_image_get_size**(void* *image*)

const char* **ph_image_get_format**(void* *image*)

void **ph_image_get_bytes**(void **image*, void **buffer*, int64_t *size*)

void* **ph_frame_find_first**(void **frame*, const char **selector*)

void* **ph_frame_find_all**(void **frame*, const char **selector*)

void* **ph_webcollection_get_webelement**(void **collection*, int32_t *index*)

void* **ph_webelement_find_all**(void **element*, const char **selector*)

void* **ph_webelement_take_from_document** (void **element*)

void* **ph_webelement_previous** (void **element*)

void* **ph_webelement_next** (void **element*)

void **ph_webcollection_free** (void **collection*)

void **ph_webelement_free** (void **element*)

char* **ph_webelement_tag_name** (void **element*)

char* **ph_webelement_inner_html** (void **element*)

char* **ph_webelement_inner_text** (void **element*)

char* **ph_webelement_get_classes** (void **element*)

char* **ph_webelement_get_attnames** (void **element*)

char* **ph_webelement_get_attr** (void **element*, const char **attrname*)

int32_t **ph_webcollection_size** (void **collection*)

int32_t **ph_webelement_has_class** (void **element*, const char **classname*)

int32_t **ph_webelement_has_attr** (void **element*, const char **attrname*)

int32_t **ph_webelement_has_attrs** (void **element*)

int32_t **ph_webelement_is_null** (void **element*)

void **ph_webelement_remove_attr** (void **element*, const char **attrname*)

void **ph_webelement_add_class** (void **element*, const char **classname*)

void **ph_webelement_set_attr** (void **element*, const char **attrname*, const char **value*)

void **ph_webelement_append_html** (void **element*, const char **htmldata*)

void **ph_webelement_append_element** (void **element*, void **elementement*)

void **ph_webelement_append_html_after** (void **element*, const char **htmldata*)

void **ph_webelement_append_element_after** (void **element*, void **elementement*)

void **ph_webelement_replace_with_html** (void **element*, const char **htmldata*)

void **ph_webelement_replace_with_element** (void **element*, void **elementement*)

void **ph_webelement_remove_all_child_elements** (void **element*)

void **ph_webelement_remove_from_document** (void **element*)

void **ph_webelement_wrap_with_html** (void **element*, const char **htmldata*)

void **ph_webelement_wrap_with_element** (void **element*, void **elementement*)

p

A

C

D

F

G

H

I

N

P

R

S

W