piwikapi Documentation

Release 0.3

Nicolas Kuttler

Sep 26, 2017

Contents

`piwikapi` is a Python implementation of the Piwik tracking and the analytics API. You can use it to track visitors, ecommerce, actions, goals, generate reports and much more.

The package was originally written for Django and expects a Django HttpRequest object. However, you don't need to use Django, you can create a *mockup object* instead.

As `piwikapi` only implements a Python interface to the Piwik APIs you'll want to check their official documentation as well.

Tracking API:

- http://piwik.org/docs/tracking-api/
- http://piwik.org/docs/tracking-api/reference/
- http://piwik.org/docs/tracking-goals-web-analytics/

Analytics API:

- http://piwik.org/docs/analytics-api/
- http://piwik.org/docs/ecommerce-analytics/
- http://piwik.org/docs/analytics-api/reference/
- http://piwik.org/docs/analytics-api/segmentation/
- http://piwik.org/docs/analytics-api/metadata/

Misc:

- http://piwik.org/docs/custom-variables/

The project is in alpha status and the API interface might change in the future. The full source is available at https://github.com/piwik/piwik-python-api.

CHAPTER 1

Advantages over client-side tracking

My first implementation of the Piwik tracking API was written for a client who needed to track 301 and 302 redirects on a Django site. So JavaScript logging obviously wouldn't work. Doing the tracking API requests from the server instead of the browser has the big advantage of making it much easier to intertwine business logic and tracking info. Ecommerce, actions and goals can be logged without any JavaScript.

Another advantage can be that the browser has one less request to do, and that you don't depend on any client-side code at all. If you care much about performance it would probably be a good idea to feed your tracking requests into some task queue.

CHAPTER 2

Disadvantages

You can't check all client-side features from the server, such as plugin support, screen resolution, click goals etc. This could be accomplished by using some JavaScript code though if necessary.

CHAPTER 3

Quickstart

At the moment only Django is supported out of the box. If you use a different framework/library you'll have to use a fake request class that mimics Django's. This is easy, but that requirement will be dropped in the future:

```
from piwikapi.tracking import PiwikTracker
from piwikapi.tests.request import FakeRequest

headers = {
    'HTTP_USER_AGENT': 'Fancy Browser 17.4',
    'REMOTE_ADDR': '203.0.113.4',
    'HTTP_REFERER': 'http://referer.com/somewhere/',
    'HTTP_ACCEPT_LANGUAGE': 'en-US',
    'SERVER_NAME': 'www.example.com',
    'PATH_INFO': '/path/to/page/',
    'QUERY_STRING': 'something=bar',
    'HTTPS': False,
}
request = FakeRequest(headers)
piwiktracker = PiwikTracker(PIWIK_SITE_ID, request)
piwiktracker.set_api_url(PIWIK_TRACKING_API_URL)
piwiktracker.set_ip(headers['REMOTE_ADDR']) # Optional, to override the IP
piwiktracker.set_token_auth(PIWIK_TOKEN_AUTH)  # Optional, to override the IP
piwiktracker.do_track_page_view('My Page Title')
```

As you can see the interface is very bad at the moment. This was partially inherited from the PHP reference implementation, and rewriting the interface is the next step for this project.

With Django you can skip the FakeRequest and simply pass Django's HttpRequest object to the API.

CHAPTER 4

News

The API code itself isn't very nice, and if I find enough time I'll rewrite it entirely. It is my intention to continue supporting the old API interface for a while and to raise deprecation warnings when necessary.

The testing method will change with the 0.3 release. The previous way of using a setup class won't be supported any more. The project was also moved into the Piwik project on github, but this shouldn't affect you unless you were checking out the code from the old repository.

CHAPTER 5

Todo

- Abstraction for various request models (Django, webob, ...?)
- API rewrite
- CLI client (?)

Supported

- Normal tracking
- Custom variables
- Ecommerce
- Goals
- Actions

Not supported yet

- Custom variables through cookies
- Attribution info through cookies
- probably more

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Install

Installing the API

You can get the source from https://github.com/piwik/piwik-python-api or install it with pip:

```
pip install piwikapi
```

On Python 2.5 you'll also have to install simplejson.

Installing Piwik

To install Piwik please see http://piwik.org/docs/installation-optimization/.

Enabling the tracking API

For simple tracking you don't need to do anything. However, by default all tracking requests will originate from your server.

If you want your user's IP to be logged (you probably do) create a new user and make that user an admin of the site you want to track.

Keep in mind that the auth token will be submitted with each request, so you should consider using SSL to submit the data to your server.

Enabling ecommerce tracking

You need to enable ecommerce tracking inside Piwik, see http://piwik.org/docs/ecommerce-analytics/#toc-enable-ecommerce-for-the-website

See also *Enabling the tracking API*.

If you want to track goals you'll have to create them first. Either through the web interface or the API.

Enabling the analytics API

The analytics API works out of the box, no configuration is needed. To view reports the viewer must have access. You could just give anonymous access to the data but it's probably better to use the auth token.

Secure your install

First you should check the official docs at http://piwik.org/security/how-to-secure-piwik/.

If you use the Python API exclusively you could consider password-protecting your Piwik install or binding the httpd to local interfaces only etc.

Tracking API Usage

If you're not using Django you will have to write some custom wrapper code, see below. Feel free to send pull request.

Usage without Django

As this code was written for usage with Django it expects a Django HttpRequest object in some places. If you're not using Django you'll want to pass an object that looks like this:

class piwikapi.tests.request.**FakeRequest**(*headers*)

A replacement for Django's Request object. This is only used for unit tests at the moment. If you're not using Django and need to create such a class have a look at the source for the tests fake request class in in tests/request.py

Configure request object according to the headers we get

Parameters **headers** (*dict*) – see META

Return type None

COOKIES = False

Cookies... work in progress

__init__(*headers*)

Configure request object according to the headers we get

Parameters **headers** (*dict*) – see META

Return type None

META = {}

HTTP headers like in the PHP $_SERVER variable, see http://php.net/manual/en/reserved.variables.server.php

secure = False
Boolean, if the connection is secured or not

is_secure()
Returns a boolean, if the connection is secured

Return type bool

You can also have a look at the official Django documentation for the HttpRequest attributes and methods, though you'll only need a very small subset of this.

Once you have created a compatible request object you can do this:

```
from piwikapi.tracking import PiwikTracker
import FakeRequest # This is your own class

headerdict = {
    '<see unit test source or PHP documentation>',
}

request = FakeRequest(headerdict)
pt = PiwikTracker(1, request) # 1 is the Piwik site id
pt.set_api_url('http://example.com/piwik.php')
pt.do_track_page_view('Page title')
```

I think this code is a little odd, but that's how the PHP class was built.

Usage with Django

In your view code do something like this, assuming you use class based views:

```
from piwikapi.tracking import PiwikTracker

pt = PiwikTracker(1, self.request) # 1 is the Piwik site id
pt.set_api_url('http://example.com/piwik.php')
pt.do_track_page_view('Page title')
```

Basic examples

These examples assume that you're passing a Django-compatible request object. If you're not using django see above, *Usage without Django*.

The first example is probably the easiest thing to track:

```
from piwikapi.tracking import PiwikTracker

pt = PiwikTracker(1, request) # 1 is the side ID you want to log to
pt.set_api_url('http://yoursite.example.com/piwik.php')
pt.do_track_page_view("Some page title")
```

This will log a page view and set the title. The only problem with this is that the request will be logged to the IP of your server. To avoid this you'll want to set the auth token of an admin for that site:

```
from piwikapi.tracking import PiwikTracker

pt = PiwikTracker(1, request) # 1 is the side ID you want to log to
pt.set_api_url('http://yoursite.example.com/piwik.php')
```

```
pt.set_ip('192.0.2.134') # Your visitor's IP
pt.set_token_auth('YOUR_AUTH_TOKEN_STRING')
pt.do_track_page_view("Some page title")
```

That's all, happy tracking!

Please refer to the *PiwikTracker reference* and *PiwikTrackerEcommerce reference* for more information.

Tracking API Reference

Please check the official Piwik API documentation as well:

- http://piwik.org/docs/tracking-api/
- http://piwik.org/docs/tracking-api/reference/

PiwikTracker

class piwikapi.tracking.**PiwikTracker**(*id_site*, *request*)

The Piwik tracker class

Parameters

- **id_site** (*int*) – Site ID
- **request** (*A Django-like request object*) – Request

Return type None

LENGTH_VISITOR_ID = 16

Length of the visitor ID

KNOWN_PLUGINS = {'director': 'dir', 'quick_time': 'qt', 'silverlight': 'ag', 'java': 'java', 'gears': 'gears', 'real_player':

List of plugins Piwik knows

set_local_time(*datetime*)

Set the time

Parameters **datetime** (*datetime.datetime object*) – Time

Return type None

set_token_auth(*token_auth*)

Set the auth token for the request. The token can be viewed in the user management section of your Piwik install.

Parameters **token_auth** (*str*) – Auth token

Return type None

set_api_url(*api_url*)

Set which Piwik API URL to use

Parameters **api_url** (*str*) – API URL

Return type None

set_ip(*ip*)

Set the IP to be tracked. You probably want to use this as the request comes from your own server.

Requires setting the auth token.

Parameters **ip** (*str*) – IP

Return type None

set_browser_has_cookies()
Call this is the browser supports cookies

Return type None

set_browser_language(*language*)
Set the browser language. Piwik uses this to guess the visitor's origin when GeoIP is not enabled

Parameters **language** (*str*) – Accept-Language

Return type None

set_user_agent(*user_agent*)
Set the user agent. By default the original request's UA is used.

Parameters **user_agent** (*str*) – User agent

Return type None

set_resolution(*width*, *height*)
Set the visitor's screen width and height

Parameters

- **width** (*int or str*) – Screen width
- **height** (*int or str*) – Screen height

Return type None

set_visitor_id(*visitor_id*)

Parameters **visitor_id** (*str*) – Visitor I

Raises InvalidParameter if the visitor_id has an incorrect length

Return type None

set_debug_string_append(*string*)

Parameters **string** (*str*) – str to append

Return type None

set_url_referer(*referer*)
Set the referer URL

Parameters **referer** (*str*) – Referer

Return type None

set_url(*url*)
Set URL being tracked

Parameters **url** (*str*) – URL

set_attribution_info(*json_encoded*)
NOT SUPPORTED

Set the attribution info for the visit, so that subsequent goal conversions are properly attributed to the right referer, timestamp, campaign name and keyword.

This must be a JSON encoded string that you would normally fetch from the Javascript API, see function getAttributionInfo() in http://dev.piwik.org/trac/browser/trunk/js/piwik.js

Parameters **json_encoded** (*string*) – JSON encoded list containing attribution info

Raises InvalidParameter if the json_encoded data is incorrect

Return type none

set_force_visit_date_time(*datetime*)
Override the server date and time for the tracking request.

By default Piwik tracks requests for the "current" datetime, but this method allows you to track visits in the past. Time are in UTC.

Requires setting the auth token.

Parameters **datetime** (*datetime.datetime object*) – datetime

Return type None

get_visitor_id()
PARTIAL, no cookie support

If the user initiating the request has the Piwik first party cookie, this function will try and return the ID parsed from this first party cookie.

If you call this function from a server, where the call is triggered by a cron or script not initiated by the actual visitor being tracked, then it will return the random Visitor ID that was assigned to this visit object.

This can be used if you wish to record more visits, actions or goals for this visitor ID later on.

Return type str

get_attribution_info()
NOT SUPPORTED

To support this we'd need to parse the cookies in the request obejct. Not sure if this makes sense...

Return the currently assigned attribution info stored in a first party cookie.

This method only works if the user is initiating the current request and his cookies can be read by this API.

Return type string, JSON encoded string containing the referer info for goal conversion attribution

get_random_visitor_id()
Return a random visitor ID

Return type str

disable_cookie_support()
NOT TESTED

By default, PiwikTracker will read third party cookies from the response and sets them in the next request.

Return type None

do_track_page_view(*document_title*)
Track a page view, return the request body

Parameters **document_title** (*str*) – The title of the page the user is on

Return type str

do_track_action(*action_url*, *action_type*)
Track a download or outlink

Parameters

- **action_url** (*str*) – URL of the download or outlink

- **action_type** (*str*) – Type of the action, either 'download' or 'link'

Raises InvalidParameter if action type is unknown

Return type str

set_custom_variable(*id*, *name*, *value*, *scope='visit'*)
Set a custom variable

See http://piwik.org/docs/custom-variables/

Parameters

- **id** (*int*) – Custom variable slot ID, 1-5
- **name** (*str*) – Variable name
- **value** (*str*) – Variable value
- **scope** (*str or None*) – Variable scope, either visit or page, defaults to visit

Return type None

set_plugins(***kwargs*)
Set supported plugins

```
>>> piwiktrackerinstance.set_plugins(flash=True)
```

See KNOWN_PLUGINS keys for valid values.

Parameters **kwargs** (*dict of {str: int}*) – A plugin: version dict, e.g. {'java': 6}, see also KNOWN_PLUGINS

Return type None

get_custom_variable(*id*, *scope='visit'*)
PARTIAL, no cookie support

Returns the current custom variable stored in a first party cookie.

Parameters

- **id** (*int*) – Custom variable slot ID, 1-5
- **scope** (*str*) – Variable scope, either visit or page

Return type mixed stuff TODO

PiwikTrackerEcommerce

class piwikapi.tracking.**PiwikTrackerEcommerce**(*id_site*, *request*)
The Piwik tracker class for ecommerce

add_ecommerce_item(*sku*, *name=False*, *category=False*, *price=False*, *quantity=1*)
Add an item to the ecommerce order.

This should be called before do_track_ecommerce_order() or before do_track_ecommerce_cart_update().

This method can be called for all individual products in the cart/order.

Parameters

- **sku** – Product SKU
- **name** (*str or None*) – Name of the product

- **category** (*str, list or None*) – Name of the category for the current category page or the product
- **price** (*int or None*) – Price of the product
- **quantity** – Product quantity, defaults to 1

Return type None

do_track_ecommerce_cart_update(*grand_total*)
Track a cart update (add/remove/update item)

On every cart update you must call add_ecommerce_item() for each item in the cart, including items which were in the previous cart. Items get deleted until they are re-submitted.

Parameters **grand_total** (*float*) – Grand total revenue of the transaction, including taxes, shipping, etc.

Return type str

do_track_ecommerce_order(*order_id*, *grand_total*, *sub_total=False*, *tax=False*, *shipping=False*, *discount=False*)
Track an ecommerce order

If the order contains items you must call add_ecommerce_item() first for each item.

All revenues will be individually summed and reported by Piwik.

Parameters

- **order_id** (*str*) – Unique order ID (required). Used to avoid re-recording an order on page reload.
- **grand_total** (*float*) – Grand total revenue of the transaction, including taxes, shipping, etc.
- **sub_total** (*float or None*) – Sub total amount, typicalle the sum of item prices for all items in this order, before tax and shipping
- **tax** (*float or None*) – Tax amount for this order
- **shipping** (*float or None*) – Shipping amount for this order
- **discount** (*float or None*) – Discount for this order

Return type str

do_track_goal(*id_goal*, *revenue=False*)
Record a goal conversion

Parameters

- **id_goal** (*int*) – Goal ID
- **revenue** (*int (TODO why int here and not float!?)*) – Revenue for this conversion

Return type str

set_ecommerce_view(*sku=False*, *name=False*, *category=False*, *price=False*)
Set the page view as an item/product page view, or an ecommerce category page view.

This method will set three custom variables of 'page' scope with the SKU, name and category for this page view.

On a category page you may set the category argument only.

Tracking product/category page views will allow Piwik to report on product and category conversion rates.

To enable ecommerce tracking see doc/install.rst

Parameters

- **SKU** (*str or None*) – Product SKU being viewed
- **name** (*str or None*) – Name of the product
- **category** (*str, list or None*) – Name of the category for the current category page or the product
- **price** (*float or None*) – Price of the product

Return type None

Analytics API Usage

Live data

Here's an example for getting live data for the last five minutes:

```
import json
from piwikapi.analytics import PiwikAnalytics

pa = PiwikAnalytics()
pa.set_api_url('http://yoursite.example.com/piwik.php')
pa.set_id_site(1) # 1 is the side ID you want to log to
pa.set_format('json')
pa.set_period('day')
pa.set_date('today')
pa.set_method('Live.getLastVisitsDetails')
pa.set_parameter('lastMinutes', 5)
visits = json.loads(self.a.send_request())
```

You can then inspect the data stored in `visits`. Please refer to the *PiwikAnalytics reference* for more details.

ImageGraphs

You can also get images from the API, so that you can save them or serve them to a webbrowser etc.:

```
from piwikapi.analytics import PiwikAnalytics

pa = PiwikAnalytics()
pa.set_api_url('http://yoursite.example.com/piwik.php')
pa.set_id_site(1) # 1 is the side ID you want to log to
pa.set_method('ImageGraph.get')
pa.set_parameter('apiModule', 'UserCountry')
pa.set_parameter('apiAction', 'getCountry')
image = pa.send_request()
```

Analytics API Reference

Please check the official Piwik API documentation as well:

- http://piwik.org/docs/analytics-api/

- http://piwik.org/docs/analytics-api/reference/
- http://piwik.org/docs/analytics-api/segmentation/
- http://piwik.org/docs/analytics-api/metadata/

PiwikAnalytics

class piwikapi.analytics.**PiwikAnalytics**
The Piwik analytics API class

Initalize the object

Return type None

set_parameter(*key*, *value*)
Set a query parameter

Parameters

- **key** (*str*) – The parameter to set
- **value** (*TODO, str?*) – The value you want to set

Return type None

remove_parameter(*key*)
Removes a query parameter

Parameters **key** – The parameter to remove

get_parameter(*key*)
Get a query parameter

Parameters **key** (*str*) – The parameter to return

Return type TODO mixed?

set_method(*method*)

Parameters **method** (*str*) – Method

Return type None

set_id_site(*id_site*)

Parameters **id_site** (*int*) – Site ID

Return type None

set_date(*date*)

Parameters **date** (*str*) – Date string TODO format

Return type None

set_period(*period*)

Parameters **period** (*str*) – Period TODO optinos

Return type None

set_format(*format*)

Parameters **format** (*str*) – Format TODO

Return type None

set_filter_limit(*filter_limit*)

Parameters **filter_limit** (*TODO ?*) – Filter limit TODO

Return type None

set_api_url(*api_url*)

Parameters **api_url** (*str*) – Piwik analytics API URL, the root of your Piwik install

Return type None

set_segment(*segment*)

Parameters **segment** (*str*) – Which segment to request, see http://piwik.org/docs/analytics-api/segmentation/

Return type None

get_query_string()
Return the query string

Raises ConfigurationError if the API URL was not set

Return type str

send_request()
Make the analytics API request, returns the request body

Return type str

Plugins Reference

These classes provide some wrappers around plugin methods.

PiwikGoals

class piwikapi.plugins.goals.**PiwikGoals**(*api_url*)
Initialize a PiwikAnalytics object

Parameters **api_url** (*str*) – Piwik API url, root of the install

Return type None

add_goal(*id_site*, *name*, *match_attribute*, *pattern*, *pattern_type*, *token_auth*, *case_sensitive=False*, *revenue=False*, *allow_multiple_conversions_per_visits=False*)
Create a goal

Parameters

- **id_site** – Site ID
- **name** – Name of the goal
- **match_attribute** – 'url', 'title', 'file', 'external_website' or 'manually'
- **pattern** – eg. purchase-confirmation.htm
- **pattern_type'** – 'regex', 'contains', 'exact'
- **case_sensitive** (*bool*) – Case sensitive
- **revenue** – Revenue per action

- **allow_multiple_conversions_per_visits** (*bool*) – By default, multiple conversions in the same visit will only record the first conversion. If set to true, multiple conversions will all be recorded within a visit (useful for Ecommerce goals)

Return type int, ID of the new goal

delete_goal(*id_site*, *id_goal*)
Delete a goal

Parameters

- **id_site** – Site ID
- **id_goal** – Goal ID

Hacking

If you want to work on the tracking code you should have a look at the original `PiwikTracking.php` in the Piwik source tree. There's also a `PiwikTracking.java` implementation in the release.

To run the unit tests you need to:

- Create a new Piwik site
- Configure your environment
- Enable tracker debugging

Creating a new site

You want to create a **new site** specifically for running the unit tests as to not pollute a real site with test data.

Configuring your environment

The tests assume that the following variables are set in your shell environment:

```
PIWIK_SITE_ID = '<ID of your test site>'
PIWIK_TRACKING_API_URL = '<Your Piwik tracker API URL>'
PIWIK_ANALYTICS_API_URL = '<Your Piwik API URL>'
PIWIK_TOKEN_AUTH = '<Your Piwik auth token>'
PIWIK_GOAL_ID = None
```

`PIWIK_TRACKING_API_URL` must contain the URL to your Piwik install's /piwik.php, something like 'http://example.com/piwik.php'. `PIWIK_TRACKING_API_URL` must contain the URL to your Piwik install's root, so 'http://example.com/'.

The auth token must have admin access for the site defined in `PIWIK_SITE_ID`. The `goal id` doesn't have to be set, the goal can be autogenerated by the tests.

Please let me know if you need a different way of defining those settings.

Enabling tracker debugging

Some unit tests parse the output of the tracker script, so you have to **enable debugging** in your Piwik install's `/piwik.php`:

```
$GLOBALS['PIWIK_TRACKER_DEBUG'] = true;
```

Test classes reference

The unit tests verify that data sent through the API was received by Piwik, either by parsing the debug output or by querying the analytics API.

Analytics API tests

These are just some very simple tests. The real testing happens in the tracking API tests, where the analytics API is used to verify the submitted data.

class piwikapi.tests.analytics.**AnalyticsTestCase**(*methodName='runTest'*)
Generic analytics API tests

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test_get_imagegraph()
Just a basic test to see if we can get an image

test_remove_parameter()

class piwikapi.tests.analytics.**AnalyticsLiveTestCase**(*methodName='runTest'*)
Useless for now

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

Tracking API tests

class piwikapi.tests.tracking.**TrackerBaseTestCase**(*methodName='runTest'*)
This sets up a more or less random visitor

In every test run all tests get the same testrun custom variable.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

setUp()
Set up a PiwikTracker instance

get_title(*title*)
Adds a timestamp to the action title"

Parameters **title** (*str*) – Action

Return type str

get_random_ip()
Returns an IP out of the test networks, see RFC 5735. Seemed to make sense to use such addresses for unit tests.

Return type str

get_random(*choices*)

get_random_ua()
Returns a random user agent string

Only return Desktop UAs as Piwik doesn't like big resolutions on devices it thinks are mobile.

Return type string

get_random_language()
Return a random language code

class piwikapi.tests.tracking.**TrackerClassTestCase**(*methodName='runTest'*)
PiwikTracker tests, without Piwik interaction

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test_set_visitor_id()
This is a little sloppy, we should probably create a custom exception

test_set_debug_string_append()

test_incorrect_custom_variables_invalid()

test_set_custom_variables()

test_set_user_agent()

test_set_visitor_id_exception()

test_do_track_action_exception()
Should probably also test that valid parameters pass

test_set_custom_variable_exception()

test_set_custom_variable_scope_exception()

test_get_custom_variable_exception()

test_get_custom_variable_scope_exception()

test_missing_api_url()

test_unknown_set_plugins()

class piwikapi.tests.tracking.**TrackerVerifyDebugTestCase**(*methodName='runTest'*)
These tests make sure that the tracking info we send is recognized by checking the debug output of Piwik.

TODO: Use the analytics API to verify the tests.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test_default_action_title_is_correct()

test_default_user_is_not_authenticated()

test_default_action_url_is_correct()

test_default_ip_is_not_changed()
This test can't fail, we use IPs from testing networks

test_default_repeat_visits_recognized()

test_token_auth_succeeds()

test_ip_not_changed_after_auth()
I think there was a bug in my earlier code. The IP should not be set or overridden just because we authenticated, Piwik should log the IP of the host that made the tracking request.

test_setting_ip_works_for_authed_user_only()

test_set_visitor_id()

class piwikapi.tests.tracking.**TrackerVerifyBaseTestCase**(*methodName='runTest'*)
The base class for tests that use the analytics API to verify

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

setUp()
To be able to verify against the analytics API each test gets a random custom variable. Segmentation is then used to query the submitted data.

get_v(*key*)
Get a variable from the last visit

get_av(*key*)
Get an action variable from the last visit

class piwikapi.tests.tracking.**TrackerVerifyTestCase**(*methodName='runTest'*)
Here are test we don't verify programmatically yet.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test_set_visitor_feature_resolution()

test_set_visitor_feature_single_plugin()

test_set_visitor_feature_plugins()

test_action_link()
Test out action

test_action_click()
Test download action

test_set_user_agent()
Piwik doesn't save the UA string but processes it.

class piwikapi.tests.ecommerce.**TrackerEcommerceBaseTestCase**(*methodName='runTest'*)
Base class for the ecommerce tests

Contains test products.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

products = {'car': {'sku': '2', 'category': ('car category', 'cars'), 'price': 525, 'name': 'Car', 'quantity': 3}, 'book': {'s

setUp()
Set up a PiwikTrackerEcommerce instance

get_cv(*number*)
Get a custom variable from the last visit

class piwikapi.tests.ecommerce.**TrackerEcommerceVerifyTestCase**(*methodName='runTest'*)
Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test_ecommerce_view()

test_add_ecommerce_item_do_track_ecommerce_cart_update()
Test add_ecommerce_item() together with do_track_ecommerce_cart_update(). Also make sure that an abandoned cart was logged.

test_track_ecommerce_order()
TODO We could test that each product was added, not only the sums

Plugin tests

class piwikapi.tests.goals.**GoalsTestCase**(*methodName='runTest'*)
Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

setUp()
Create a goal unless one was defined in the settings

tearDown()

test_track_goal_conversion()
Make sure goal conversions are logged

test_create_goal()
This is superfluous when we create a goal on the fly

Changelog

0.3 (2013-02-20)

- Python 3.2 support
- Switch to hashlib
- Test 'builds' on travis

0.2.2 (2013-01-20)

- Don't require anonymous view access for unit tests to pass
- Test against Piwik 1.10
- Fix readthedocs build

0.2.1 (2012-10-25)

- A few small improvements

0.2 (2012-04-15)

First release as piwikapi on pypi.

- Ecommerce tracking support
- Custom variables tracking support
- Action tracking support

- Goal tracking support
- Added unit tests
- Code refactoring
- Got rid of the Django dependency

0.1 (2012-04-03)

First release as django-piwik-tracker

- Written in a few hours for a client's project that had just gone live
- Very basic implementation

TODO

- Implement and test all the cookie stuff
- Refactor the tracking API code, it's not very pythonic
- Verify all unit tests through the analytics API
- Create sites etc. automatically if necessary for the tests
- TODO: SitesManager plugin
- TODO: ImageGraph plugin
- TODO: UserCountry plugin
- TODO: VisitsSummary plugin

Index

Symbols

A

C

D

F

G

I

K