Bambu Analytics Documentation

Release 2.0

Steadman

Sep 27, 2017

Contents

Provides a simple, pluggable system for analytics

CHAPTER 1

About Bambu Analytics

Bambu Analytics provides a simple system for implementing analytics tools like Google Analytics into your Django projects, so you can track page views, goals and events.

By default it supports Google's Universal Analytics programme, but you interact with the package within JavaScript via the `bambu.analytics` namespace. This way, you can change analytics providers (or write your own) without changing the code within the rest of your site.

This is massively a work-in-progress.

CHAPTER 2

About Bambu Tools 2.0

This is part of a toolset called Bambu Tools. It's being moved from a namespace of `bambu` to its own 'root-level' package, along with all the other tools in the set. If you're upgrading from a version prior to 2.0, please make sure to update your code to use `bambu_analytics` rather than `bambu.analytics`.

CHAPTER 3

Contents

Installation

Install the package via Pip:

```
pip install bambu-analytics
```

Add it to your `INSTALLED_APPS` list:

```
INSTALLED_APPS = (
    ...
    'bambu_analytics'
)
```

Next, install the tracking middleware:

```
MIDDLEWARE_CLASSES = (
    ...
    'bambu_analytics.middleware.AnalyticsMiddleware',
    ...
)
```

Also, make sure 'django.core.context_processors.request' is listed in your `TEMPLATE_CONTEXT_PROCESSORS` settings otherwise Bambu AJAX won't be able to access the current request object.

Finally, set your Google Analytics ID:

```
ANALYTICS_SETTINGS = {
    'UniversalAnalyticsProvider': {
        'ID': 'UA-XXXXXXXX-XX'
    }
}
```

Or, use the shortcut setting:

```
GOOGLE_ANALYTICS_IDS = ('UA-XXXXXXXX-XX',)
```

(This is a legacy setting that will be deprecated in a future release)

Usage

By default, all page views will be tracked once you include the `tracking` template tag in your base HTML template, like so:

```
<!DOCTYPE html>
<html>
    ...
    <body>
        ...
        {% load analytics %}{% tracking %}
    </body>
</html>
```

Tracking events are gathered by the middleware, as it allows trackable events to be defined server-side. For example, when you submit an enquiry form, you can add an event that will be tracked once the user is redirected to the 'thank you' page.

The workflow

1. The user requests a URL
2. The analytics middleware adds a page-view event to its tracking list
3. The view for that URL is rendered, and the script containing the analytics setup code and the tracked event from step 2 is rendered
4. The user submits a form on the page
5. The view for that form calls `bambu_analytics.track_event`
6. An HTTP redirect is issued
7. The middleware reads the redirect and stores the tracking event in a session variable
8. The user's browser is redirected to a 'thank you' page
9. When the 'thank you' page is rendered, the tracking event stored in the session variable are read into JavaScript and rendered

All of this sounds complex, but actually means you can track events more easily and in a pluggable, product-agnostic way. It also provides the option for server-side analytics events to be tracked.

In Google Analytics, the practical upshot is that it uses events rather than goals, meaning you don't have to manually define them in your Analytics property.

Tracking an event

If you're using a server-side provider - or you've written one, a: please let me know! but b: - this method should work fine.

Trackable events

Providers

Changing analytics provider

Bambu Analytics supports the legacy (ua.js) and new (analytics.js) scripts as provided by Google. ecommerce is setup to work with the old style (ua.js), so if you need to track ecommerce events, you should change the provider via your Django settings file:

```
ANALYTICS_PROVIDER = 'bambu_analytics.providers.google.GoogleAnalyticsProvider'
```

Writing your own provider

It's pretty easy to write your own provider. Start by taking a look at the two classes in `bambu_analytics.providers.google` to see how they're hooked up.

Essentially the job of a provider is to take Python objects that refer to events and turn them into JavaScript objects and function calls that your analytics library can understand.

Each provider needs to render a string. For client-side analytics tools this should contain HTML with a `<script>` tag. The first thing inside that tag should be:

```
{% include 'analytics/bambu.inc.js' %}
```

This exposes the `bambu.analytics` namespace. After all the code needed to hook up the analytics tool and track basic events, your provider should bind to the `track` event within `bambu.analytics` like this:

```
bambu.analytics.on('track',
    function(e) {
        // e.event contains the name of the event, which you can compare
        // against the constants in the bambu.analytics namespace (they're)
        // the same as the ones within the Python package.

        // e.args contains a dictionary of arguments that you can use to map
        // the Python-defined keyword args (like 'category' or 'option_value')
        // to arguments that your specific analytics library understands. See
        // the templates/analytics/universal.inc.html file for an idea of
        // how this works.
    }
);
```

This way you can write an analytics provider that works on all sites that use Bambu Analytics. Both of them!

Writing a server-side provider

If you want to track your own events or you have a server-side analytics tool that you want to hook into, you'll write a provider that focuses on teh back- rather than front-end. You'll still need to render something, but this can be an empty string, or some sort of tracking pixel if that's the route you want to go down.

AJAX and client-side tracking

If you want to track events client-side, or you're running a site that uses a lot of AJAX (like Poddle.fm), you'll get automatic access to the `bambu.analytics` namespace within JavaScript, and you can call `track()` to handle client-side events or AJAX page updates (ie: via `window.pushstate`).

Here's an example event used on Poddle.fm when a user clicks the Play button on an episode of a podcast:

```
<script>
    $('a.btn-play').on('click',
        function() {
            // Play the audio
            ...

            // Track the click event
            bambu.analytics.track(
                bambu.analytics.EVENT,
                {
                    category: 'Audio',
                    action: 'play'
                }
            );
        }
    );
</script>
```

Settings

CHAPTER 4

Todo

- Implement ecommerce into the Universal Analytics provider

CHAPTER 5

Questions or suggestions?

Find me on Twitter (@iamsteadman) or visit my blog.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search