anafero Documentation

Release 1.1

Eldarion

March 24, 2015

Contents

Provides a site with the ability for users to publish referral links to specific pages or objects and then record any responses to those links for subsequent use by the site.

For example, one an object detail page, the site builder would use a template tag from anafero to display a referral link that the user of the site can send out in a Tweet. Upon clicking that link, a response to that referral code will be record as well any other activity that the site builder wants to track for that session.

It is also possilbe for anonymous referral links/codes to be generated which is useful in marketing promotions and the like.

CHAPTER 1

Development

The source repository can be found at https://github.com/eldarion/anafero

1.1 Contents

1.1.1 Installation

- To install anafero:

  ```
  pip install anafero
  ```

- Add `anafero` to your `INSTALLED_APPS` setting:

  ```
  INSTALLED_APPS = (
      # other apps
      "anafero",
  )
  ```

- See the list of *Settings* to modify anafero's default behavior and make adjustments for your website.
- Add `anafero.middleware.SessionJumpingMiddleware` in order to link up a user who registers and authenticate after hitting the initial referral link. Make sure that it comes after the `django.contrib.auth.middleware.AuthenticationMiddleware`:

  ```
  MIDDLEWARE_CLASSES = [
      ...
      "django.contrib.auth.middleware.AuthenticationMiddleware",
      ...
      "anafero.middleware.SessionJumpingMiddleware",
      ...
  ]
  ```

- Lastly you will want to add *anafero.urls* to your urls definition:

  ```
  ...
  url(r"^referrals/", include("anafero.urls")),
  ...
  ```

1.1.2 Usage

Referral.create

This is the factory method that the `create_referral` view calls but can be called directly in case you needed to integrate in a different way with anafero.

For example, you might want to automatically give every user a referral code that is emailed to them upon signup. In this case, you could created a one to one relationshiop between their `Profile` and anafero's `Referral` and create a signal receiver for when the `Profile` is created that calls:

```
referral = Referral.create(
    user=profile.user,
    redirect_to=reverse("home")
)
profile.referral = referral
profile.save()
```

Then you could, in the welcome email that you send them upon signup render their referral url that they could forward on to other users:

```
{{ user.get_profile.referral.url }}
```

The only required parameter for `Referral.create` is `redirect_to`. If you don't specify a user it will be recorded as `None`. This can be useful if you wanted to attach a relationship between some object in your system to a referral that might not have a user associated with it.

You can also pass in an optional `label` kwarg to `Referral.create` if you wanted to allow your users to create and manage multiple referrals so that labeling them became important to keep track of them.

Referral.record_response

The classmethod `record_response` will attempt to see if the current user or session has any previous responses to an initial referral and if so will then proceed to record the response action.

For example, say you want to record the fact that the user did some site activity after clicking on the referral link you tweeted and subsequently decided to register and login to the site:

```
from anafero.models import Referral


def my_view(request, **kwargs):
    ...
    referral_response = Referral.record_response(request, "SOME_ACTION")
    ...
```

In this case the `referral_response` will be None if the user on the request doesn't have any previously recorded referral responses. In addition, if the user has responded to more than one Referral code, then this will associate the activity with the most recent response.

In addition, this supports passing an options `target` keyword argument, if you wanted to record associations with specific objects. For example:

```
Referral.record_response(request, "SOME_ACTION", target=some_object)
```

This will record a generic foreign key to `some_object` that you can use elsewhere to identify activity from your referral at a deeper level than just based on the action label.

Referral.referral_for_request

This class method, will give you a referral object for the given request in case you needed to apply any business logic in your project. For example, to do any comparisons on the referral.target with another object you have in context for segmenting permissions or authorizations to make your referral system more fine grained.

1.1.3 Settings

ANAFERO_IP_ADDRESS_META_FIELD

Default "HTTP_X_FORWARDED_FOR"

This is the header value that is retrieved from *request.META* to record the ip address of the the respondent.

ANAFERO_SECURE_URLS

Default `False`

Setting this to `True` will enable produce urls with `https` instead of `http`.

ANAFERO_CODE_GENERATOR

Default "anafero.utils.generate_code"

Externalizes the logic that generates the referral code. *anafero* ships with a default that will generate a random 40-character alpha-numeric string that can also be used as a reference implementation. The callable defined by the fully qualified path is passed a single parameter that is the class of the referral model, or *Referral*, this is done as a pure convience so as to alleviate the need for you to have to import it should you need it (and you most likely will if you want to be certain of uniqueness).

ANAFERO_ACTION_DISPLAY

Default `{"RESPONDED": "Clicked on referral link"}`

Defines a dictionary mapping action codes for responses to user-friendly display text. Used by the `action_display` template filter.

1.1.4 Signals

`user_linked_to_response` is a signal that provides the single argument of a `response` object that has just been linked to a user. You can use this to provide further authomatic processing within your site, such as adding permissions, etc. to users that signup as a result of a referral.

For example:

@receiver(user_linked_to_response) def handle_user_linked_to_response(sender, response, **kwargs):

if response.action == "SOME_SPECIAL_ACTION": pass # do something with response.user and response.referral.target (object that referral was linked to)

1.1.5 Templates

anafero comes with a single template fragment for rendering a simple form that is used in creating a referral link.

_create_referral_form.html

This is a snippet that renders the form that upon submission will create the referral link. By default it is rendered with the class *referral* with the following context variables:

```
{
    "url": url,
    "obj": obj,
    "obj_ct": ContentType.objects.get_for_model(obj)
}
```

or if no object was passed into the *create_referral* template tag then the context would simply blank for *obj* and *obj_ct*.

1.1.6 Template Tags and Filters

create_referral

In order to use *anafero* in your project you will use the *create_referral* template tag wherever you'd like a user to be able to get a referral link to a page or a particular object:

```
{% load anafero_tags %}

{% create_referral object.get_absolute_url object %}
```

The use of *object* in this case is optional if you just want to record referrals to a particular url. In that case you can just do:

```
{% load anafero_tags %}

{% url my_named_url as myurl %}

{% create_referral myurl %}
```

This will render a form that is defined in *anafero/_create_referral_form.html* which will POST to a view and return JSON. The intent of this form is that it is to be used with an AJAX submission and handler.

The recommended way is to use *jquery.form* and to do the following:

```
$(function () {
    $('form.referral').each(function(i, e) {
        var form = $(e);
        options = {
            dataType: "json",
            success: function(data) {
                form.html('<input type="text" value="' + data.url + '" />');
                form.find("input[type=text]").select();
            }
        }
        form.ajaxForm(options);
    });
});
```

referral_responses

This template tag is an assignment tag that given a user sets an context variable with a queryset all all responses for all referrals owned by the user, in order of when they were created.

The use case for where this is useful is displaying all the activities associated with the user's different labeled referrals. Example:

```
{% load anafero_tags %}
{% referral_responses user as responses %}

{% for response in responses %}
    {# response is a ReferralResponse object #}
{% endfor %}
```

action_display

This filter converts a response code into a user friendly display of what that code means. The definitions exist in the setting `ANAFERO_ACTION_DISPLAY`.

> {% load anafero_tags %}
>
> **<p>** {{ response.action|action_display }}
>
> </p>

1.1.7 ChangeLog

1.0.1

- Fix deprecation warning in urls

1.0

- ENHANCEMENT: made GFK fields on *Referral* blankable

0.10.0

- ENHANCEMENT: added a signal to provide notifications of when a referred user authenticates

0.9.1

- BUG: Fix issue where target response is None

0.9

- FEATURE: added ability to record a specific object in reference to each response

Migration from 0.8.1

> **ALTER TABLE "anafero_referralresponse"** ADD COLUMN "target_content_type_id" integer REFERENCES "django_content_type" ("id") DEFERRABLE INITIALLY DEFERRED, ADD COLUMN "target_object_id" integer;

0.8.1

- ENHANCEMENT: switched over to use *django.utils.timezone.now* instead of *datetime.datetime.now*

0.8

- FEATURE: added ability to override filtering of responses

0.7

- ENHANCEMENT: Made admin a bit more usable

0.6.1

- ENHANCEMENT: Rewrote `referral_responses` to run on Django 1.3

0.6

- ENHANCEMENT: send full context to the `create_referral` template tag

0.5.2

- BUG: Fixed a stupid mistake

0.5.1

- BUG: fixed an issue with sessions in Django 1.4

0.5

- FEATURE: added ability to label referrals
- ENHANCEMENT: added support for bootstrap-ajax.js
- FEATURE: added a `referral_responses` assignment template tag
- FEATURE: added an `activity_display` template filter
- ENHANCEMENT: added a new classmethod on `Referral` for getting a referral object for a given `request` object.

Migration from 0.4

ALTER TABLE "anafero_referral" ADD COLUMN "label" varchar(100) NOT NULL DEFAULT '';

0.4

- FEATURE: Add ability to define code generators external to anafero
- ENHANCEMENT: Add ability to pass in a user to *referral.respond* in cases when *request.user* isn't set yet (e.g. during signup)
- FEATURE: Add ability to get a referral object from a request

0.3

- FEATURE: changed user on Referral to be nullable, thus enabling anonymous or site administered referral codes

0.2.1

- BUG: fixed target not being set in the *create_referral* ajax view

0.2

- DOC: fixed a typo in the docs
- ENHANCEMENT: added a response count property
- ENHANCEMENT: added the return of the referral code along with the URL in the ajax reponse of *create_referral*
- BUG: added the return of the proper mimetype in the *create_referral* ajax view
- ENHANCEMENT: moved the building of the URL for the referral code to a property on the Referral model
- FEATURE: added the ability to control referral code generation properties via settings at the site level
- BUG: fixed the url generation to include https if the site is configured to run SSL
- BUG: only delete cookies if user is present
- BUG: make sure to set a session value to prevent session key from changing with each request

0.1

- initial release