django-sendgrid-events Documentation

Release 1.2

Eldarion, Inc

November 07, 2016

Contents

a simple app to provide and endpoind for and handle batch events from SendGrid's Event API

CHAPTER 1

Development

The source repository can be found at https://github.com/eldarion/django-sendgrid-events

1.1 Contents

1.1.1 ChangeLog

1.1

- added an admin

1.0

- initial release

1.1.2 Installation

- To install

```
pip install django-sendgrid-events
```

- Add 'sendgrid_events' to your INSTALLED_APPS setting:

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

- Add 'sengrid_events.urls' as an inclusion to the your main urls.py file. You will likely want to provide a bit of security through obscurity as you are essentially going to be trusting whatever is POSTed to this url.

  You can do this by generating a UUID and then copy and pasting this as part of your URL:

```
>>> import uuid
>>> print uuid.uuid4()
aeea4b34-e5cb-4b05-84d8-79bfee95ccf4
```

  Then in your urls.py:

```
    url("^aeea4b34-e5cb-4b05-84d8-79bfee95ccf4/", include("sendgrid_events.urls"))
```

- Finally, go to and setup the Events API app, pasting in the end point to this url, which will be:

```
    http(s)://<yoursite.com>/aeea4b34-e5cb-4b05-84d8-79bfee95ccf4/batch/
```

  Be sure to check the box that says "Batch event notifications"

1.1.3 Signals

batch_processed

Provides a single argument which is the `sendgrid_events.models.Event` instance for the single event within the batch. A single batch POST can contain one or more events.

1.1.4 Usage

Using this after it has been setup in your site depends on what you want to accomplish. Just installing it will enable you to receive events on every email sent through your Sendgrid account.

That by itself, however, likely produces much value as there is no context to those emails to derive any meaningful use out of.

One of the cool things you can do now that you have the Events API configured and django-sendgrid-events hooked up, is send identifying data in the headers of each email you send so that when you receive and Event API POST you can tied each email back to some event in your site.

To accomplish this you should make sure your email settings are configured to use your Sendgrid account:

```
EMAIL_BACKEND = "django.core.mail.backends.smtp.EmailBackend"
EMAIL_HOST = os.environ.get("EMAIL_HOST", "smtp.sendgrid.net")
EMAIL_PORT = os.environ.get("EMAIL_PORT", 587)
EMAIL_HOST_USER = os.environ.get("EMAIL_HOST_USER", "")
EMAIL_HOST_PASSWORD = os.environ.get("EMAIL_HOST_PASSWORD", "")
EMAIL_USE_TLS = True
```

And then whenever you are sending an email in your site that you want to be able to track the response to send an extra header like so:

```
email = EmailMultiAlternatives(
    "Your Subject Here",
    "Your plain text message here",
    "sender@email.com",
    ["recipient@email.com"],
    headers={
        "X-SMTPAPI": json.dumps({
            "unique_args": {
                "some_key": "some_value"
            },
            "category": "some_category"
        })
    }
)
email.attach_alternative("Your html rendered email", "text/html")
email.send()
```

You don't have to send HTML emails, but you can control the look a bit better since in order to get open and click event tracking with plain text emails, Sendgrids converts it to an HTML email, which can end up not looking how you'd like it to.

Now when your events POST to the endpoint you have setup, they'll have this extra bit of data which you can use to link to something in your site. For example, say, you want to link it to the instance of a specific content model you can pass in as a `unique_arg`:

```
"unique_args": {
    "my_model_pk": object.pk
}
```

Then hook up a signal receiver for `sendgrid_events.signals.batch_processed`:

```
from django.dispatch import receiver

from sendgrid_events.signals import batch_processed
from mysite.myapp.models import SomeContent


@receiver(batch_processed)
def handle_batch_processed(sender, events, **kwargs):
    for event in events:
        try:
            c = SomeContent.objects.get(pk=event.data.get("my_model_pk"))
            c.email_events.create(sendgrid_event=event)
        except SomeContent.DoesNotExist:
            pass
```

Where you have created a tracking model in your `mysite.myapps.models` for `SomeContent`:

```
class SomeContentEvent(models.Model):
    some_content = models.ForeignKey(SomeContent, related_name="email_events")
    sendgrid_event = models.ForeignKey(Event, related_name="+")

    class Meta:
        ordering = ["sendgrid_event__created_at"]
```

Now, you have access to email events for that object that you can use in your site:

```
{% for event in some_content.email_events.all %}
    <span class="label label-{{ event.kind }}" title="{{ event.created_at }}">
        {{ event.kind }}
    </span>
{% endfor %}
```