Django Whisperer

Oct 27, 2020

Contents:

Stay informed of it! Django Whisperer informs subscribed users via an URL when a specific event occurs. Currently only works on PostgreSQL.

Let's have a look:

```
from whisperer.events import WhispererEvent, registry
from whisperer.tasks import deliver_event
from django.db.models.signals import post_save

class PackageCreateEvent(WhispererEvent):
    serializer_class = PackageSerializer
    event_type = 'package-created'

registry.register(PackageCreateEvent)


def signal_receiver(instance, created=False, **kwargs):
    if created:
        deliver_event(instance, 'package-created')

post_save.connect(signal_receiver, Package)
```

This will informs subscribed users when a package created with the following payload:

```
>>> webhook_event.request_payload
'{"event": {"type": "package-created", "uuid": "da81e85139824c6187dd1e58a7d3f971"},
↪"data": {"...": "..."}}'
```

`data` contains serialized data of the instance which triggered the whisperer event.

CHAPTER 1

Introduction

1.1 Installation

You can install this package from PyPI:

```
pip install dj-whisperer
```

You need to append it to the `INSTALLED_APPS`:

```
INSTALLED_APPS = [
    ...
    'whisperer',
]
```

Then migrate your project:

```
python manage.py migrate whisperer
```

add cron to `CELERYBEAT_SCHEDULE` for undelivered event scanner:

```
CELERYBEAT_SCHEDULE = {
    ...
    'undelivered-event-scanner-cron': {
        'task': 'whisperer.tasks.undelivered_event_scanner',
        'schedule': get_rand_seconds(60 * 60, deviation_seconds=60 * 30),
        'args': (),
    },
}
```

Now you are ready to create your whisperer events, take a look at *Quick Start*.

CHAPTER 2

Quick Start

This page shows some examples of the basic usage.

2.1 Creating a Whisperer Event

Let's define a Whisperer Event.

```
from whisperer.events import WhispererEvent, registry

class PackageCreateEvent(WhispererEvent):
    serializer_class = PackageSerializer
    event_type = 'package-created'

registry.register(PackageCreateEvent)
```

2.2 Subscribing to a Whisperer Event

If `https://example.com/` wants to learn when a package created on `https://your-app.com/`, it can subscribe to `package-created` event like below.

```
import requests

requests.post(
    url='https://your-app.com/whisperer/hooks/',
    headers={
        'Authorization': 'Token <secret-login-token>',
    },
    json={
        'event_type': 'package-created',
        'secret_key': '<secret>',
```

(continues on next page)

(continued from previous page)

```
        'target_url': 'https://example.com/',
        'retry_countdown_config': {
            'choice': 'linear',
            'kwargs': {
                'base': 1 * 60,
                'limit': 10 * 60
            }
        }
    }
)
```

2.3 Delivering an event to subscribed users

A whisperer event can be delivered to subscribed users using `deliver_event` function

```
from django.db.models.signals import post_save
from whisperer.tasks import deliver_event
from foo.bar.app.models import Package

def signal_receiver(instance, created=False, **kwargs):
    if created:
        deliver_event(instance, 'package-created')

post_save.connect(signal_receiver, Package)
```

`dj-whisperer` will inform subscribed users as follows:

```
import requests

requests.post(
    url='https://example.com/',
    headers={
        'Content-Type': 'application/json',
        'X-Whisperer-Event': 'package-created'
    },
    json={
        'event': {
            'type': 'package-created',
            'uuid': 'da81e85139824c6187dd1e58a7d3f971',
        },
        'data': {
            'id': 61,
            'transfer_id': 49,
            'order_number': '248398923123',
            '.....': '......',
        }
    }
)
```

2.4 Cancelling a subscription

```
import requests

requests.delete(
    url='https://your-app.com/whisperer/hooks/<webhook-id>/',
    headers={
        'Authorization': 'Token <secret-login-token>'
    }
)
```

CHAPTER 3

Advanced Usage

Django Whisperer app is highly extensible.

3.1 Callback Function

When a callback function specified, it can be called after informing subscribed user with response, event_type, instance and request payload.

```
import logging

logger = logging.getLogger(__name__)

def callback(response, event_type, instance, payload):
    logger.info('this is a sweety callback function.')
    # some other codes
```

Subscribing to an event with callback function is as follows:

```
import requests

requests.post(
    url='https://your-app.com/whisperer/hooks/',
    headers={
        'Authorization': 'Token <secret-login-token>',
    },
    json={
        'event_type': 'package-created',
        'secret_key': '<secret>',
        'target_url': 'https://example.com/',
        'callback': 'foo.bar.app.callback',
        'retry_countdown_config': {'choice': 'exponential', 'kwargs': {'base': 2}}
    }
)
```

3.2 Deliver Event Optional Parameters

`deliver_event` function optional parameters:

3.2.1 _async

default: True

Set `_async=False` if you want the code to run sync

```
deliver_event(instance, 'package-created', _async=False)
```

3.2.2 event_uuid

default: None

Set `event_uuid=uuid` if you want process with spesific WebhookEvent

```
deliver_event(
    instance,
    'package-created',
    event_uuid='d960acbe-4193-44b8-b254-df115cf6d2e7'
)
```

3.3 Settings Paramaters

WHISPERER_REQUEST_TIMEOUT

Default: 10

If you specify a single value for the timeout, like this:

```
WHISPERER_REQUEST_TIMEOUT = 5
```

The timeout value will be applied to both the `connect` and the `read` timeouts. Specify a tuple if you would like to set the values separately:

```
WHISPERER_REQUEST_TIMEOUT = (5 , 10)
```

3.4 Custom Retry Countdown

When one of `linear`, `exponential`, `fixed` & `random` retry countdown patterns is not suitable and there is need for custom retry countdown pattern, it can be possible as below.

```
from whisperer.countdown import BaseRetryCountdown, countdown_classes


@countdown_classes.register(key='custom')
class CustomRetryCountdown(BaseRetryCountdown):
    def __init__(self, factor):
```

(continues on next page)

(continued from previous page)

```
        self.factor = factor

    def get_value(self, retry_count):
        if retry_count % 2 == 0:
            return 2 * self.factor * retry_count
        return self.factor * retry_count
```

There must be kwargs serializer class for that countdown

```
from rest_framework import serializers
from whisperer.validators import countdown_kwargs_serializers


@countdown_kwargs_serializers.register(key='custom')
class CustomRetryCountdownKwargsSerializer(serializers.Serializer):
    factor = serializers.IntegerField(min_value=1)
```

Subscribing to an event with this custom retry countdown is as follows:

```
import requests

requests.post(
    url='https://your-app.com/whisperer/hooks/',
    headers={
        'Authorization': 'Token <secret-login-token>',
    },
    json={
        'event_type': 'package-created',
        'secret_key': '<secret>',
        'target_url': 'https://example.com/',
        'retry_countdown_config': {
            'choice': 'custom',
            'kwargs': {
                'factor': 5
            }
        }
    }
)
```

3.5 Authentication Methods

Use can use authentication methods adding auth config on Webhook config field. Only basic authentication supported for now.

For basic authentication:

```
import requests

requests.post(
    url='https://your-app.com/whisperer/hooks/',
    json={
        'event_type': 'order-created',
        'target_url': 'https://example.com/order-created/',
    },
```

(continues on next page)

(continued from previous page)

```
    config={
        'auth': {
            'auth_type': 'basic',
            'username': 'username',
            'password': 'password'
        }
    }
)
```