dj-paddle Documentation

Release 0.1.1

Florian Purchess

Sep 06, 2020

Contents

Django + Paddle Made Easy

Contents:

CHAPTER 1

Getting Started

1.1 Get the distribution

Install dj-paddle:

```
pip install dj-paddle
```

1.2 Configuration

Add `djpaddle` to your `INSTALLED_APPS`:

```
INSTALLED_APPS =(
    ...
    "djpaddle",
    ...
)
```

Add to urls.py:

```
path("paddle/", include("djpaddle.urls", namespace="djpaddle")),
```

Tell paddle about the webhook (paddle webhook docs can be found here) using the full URL of your endpoint from the urls.py step above (e.g. `https://example.com/paddle/webhook/`).

Add your paddle keys and set the operating mode:

```
# can be found at https://vendors.paddle.com/authentication
DJPADDLE_VENDOR_ID = '<your-vendor-id>'

# create one at https://vendors.paddle.com/authentication
DJPADDLE_API_KEY = '<your-api-key>'
```

(continues on next page)

(continued from previous page)

```
# can be found at https://vendors.paddle.com/public-key
DJPADDLE_PUBLIC_KEY = '<your-public-key>'
```

djpaddle includes `vendor_id` template context processor which adds your vendor ID as `DJPADDLE_VENDOR_ID` to each template context

```
TEMPLATES = [
{
    ...
    'OPTIONS': {
        ...
        'context_processors': [
            ...
            'djpaddle.context_processors.vendor_id',
            ...
        ]
    }
}
```

Run the commands:

```
python manage.py migrate

# fetches all subscription plans from paddle
python manage.py djpaddle_sync_plans_from_paddle
```

1.3 Paddle Checkout

Next to setup a PaddleJS checkout page

First load in PaddleJS and initialise it by including the dj-paddle PaddleJS template in your own template to load PaddleJS:

```
{% include "djpaddle_paddlejs.html" %}
```

Next add a Paddle product or subscription plan into the page context. Below is an example of how to do this using a class based view where `plan_id` is passed through as a value from the URL

```
from django.conf import settings
from django.views.generic import TemplateView

from djpaddle.models import Plan


class Checkout(TemplateView):
    template_name = 'checkout.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

        context['paddle_plan'] = Plan.objects.get(pk=kwargs['plan_id'])
        # If you have not added 'djpaddle.context_processors.vendor_id' as a template␣
↪context processors
        context['DJPADDLE_VENDOR_ID'] = settings.DJPADDLE_VENDOR_ID
```

(continues on next page)

(continued from previous page)

```
        return context
```

Finally put a `Buy Now!` button for the plan subscription you added to the context

```
<a href="#!" class="paddle_button" data-product="{{ paddle_plan.id }}">Buy Now!</a>
```

Clicking this button will kick off the Paddle checkout process.

CHAPTER 2

Paddle Checkout

Once you have configured dj-paddle you will want to create a checkout page to process orders.

2.1 Paddle JS

To use Paddle checkout you need to load in Paddle JS. The easiest way to do this is to include the `dj-paddle.html` template on your checkout page.:

```
{% include "djpaddle_paddlejs.html" %}
```

Note: You need to have added the `djpaddle.context_processors.vendor_id` template context processor or manually add `DJPADDLE_VENDOR_ID` to your context.

If you want to customise the Paddle setup you can manually add PaddleJS and `Paddle.Setup` manually with:

```
<script src="https://cdn.paddle.com/paddle/paddle.js"></script>
<script type="text/javascript">
Paddle.Setup({
    vendor:  {{ DJPADDLE_VENDOR_ID }},
});
</script>
```

2.2 Checkout buttons

You can pass data to Paddle JS by add data attributes to the button. For example to set the users email you can use the `data-email` attribute

```
<a href="#!" class="paddle_button" data-product="{{ paddle_plan.id }}" data-email="{{␣
↪user.email }}" >Buy Now!</a>
```

Additional data can be saved against a order / subsciption using `passthrough` as the `data-passthrough` attribute

```
<a href="#!" class="paddle_button" data-product="{{ paddle_plan.id }}" data-email="{{
↪user.email }}" data-passthrough='{"user_id": {{ user.pk }}, "affiliation": "Acme
↪Corp"}'>Buy Now!</a>
```

See the Paddle checkout web sending additional user data page for more information.

2.3 Checkout success

Without writing extra javascript, Paddle's checkout process does not give you any data back when a user completes a purchase. Paddle only sends information to your app via Webhooks. This can mean your app is unaware a purchase has happened until the Webhook is processed.

dj-paddle comes with an extra template which automatically adds very basic checkout data to the `Checkout` model via an API request when the checkout process is completed. This is so you don't have to write you own Post checkout javascript functions.

To use it you need to add the following to your template:

```
{% include "djpaddle_post_checkout.html" %}
```

This is done by automatically registering a Paddle `successCallback` onto any HTML element with the `paddle_button` class. `successCallback` then does an API request to a dj-paddle endpoint to save Checkout data.

2.3.1 Redirect after checkout

If you want to redirect the user after the checkout process is complete while using `djpaddle_post_checkout.html` you can add a context variable called `djpaddle_checkout_success_redirect`.

This will then redirect the user after the checkout data has been saved. The redirect will also include a checkout query parameter `?checkout={checkout_id}` so you can look up the checkout information on the redirected page.

2.3.2 Post-Checkout Order Information

Once the user has completed an order, you will probably want to display some kind of success message to you user with some information about the order. The best way to do this is redirecting the user to a success page using the `djpaddle_checkout_success_redirect` context variable above.

In the redirect view you can then get the basic order information from Paddle

---

Note: As Paddle Post Checkout Order Information states, order processing may take a few seconds after the transaction to complete. It's best to wait for the created / succeeded webhook to be processed before actually creating updating your model(s).

Note: dj-paddle does not yet support one-off purchases and does not do anything with `payment_succeeded` webhooks. This means there is currently no signal for one of purchases.

To get notified as soon as the `subscription_created` Webhook has been processed by dj-paddle you can listen to a `post_save` signal on the `Subscription` model.

```
from djpaddle.models import Subscription


def paddle_subscription_reciever(sender, instance, created, **kwargs):
    if created:
        ...

post_save.connect(paddle_subscription_reciever, sender=Subscription)
```

2.4 Keeping checkout information in sync

Due to Paddles checkout flow, it could be possible to miss checkout data and your system not to be in sync with Paddle. Because of this, you may want to ensure your data is in sync with Paddle.

2.4.1 Using the dj-paddle checkout model

If you have been using the `djpaddle_post_checkout.html` template you should have a record of each successful checkout in the djpaddle Checkout model. This model can then be used to compare each `Checkout.id` against each `Subscription.checkout_id` to ensure no Webhooks have been missed.

More info and management command coming soon

2.4.2 Using Paddle's Webhook history

Retrieving past events and alerts that Paddle has sent via webhooks using the Get Webhook History API. They should be replayed in the order they were created.

More info and management command coming soon

2.5 Other Paddle post checkout options

If you want to manually configure what happens after a checkout has been completed instead of using the `checkout_push.html` template please see:

- Order Information
- Paddles Post checkout page
- Paddles Checkout Events page

Note:

- Subscriptions currently do not have an option within Paddle to set a redirect URL via the seller dashboard
- For normal products, using the `successCallback` or `data-success-callback` will override any success redirect set in your Seller Dashboard. This includes using the `djpaddle_post_checkout` template above

- When redirecting using the `data-success` attribute (mentioned here), the redirect URL will **NOT** receive a checkout query parameter (`checkout={checkout_hash}`). Because of this, it is not advised to use this as the redirect provides no information about the checkout that has just been completed
- If you still want to use `data-success` ensure the value is set to the full URL of your application using `request.build_absolute_uri()`

CHAPTER 3

Indices and tables

- genindex
- modindex
- search