Django payments payu Documentation

Release 1.0.0

Petr Dlouhý

Oct 21, 2020

Contents

Contents:

CHAPTER 1

Installation

At the command line:

```
$ easy_install django-payments-payu
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-payments-payu
$ pip install django-payments-payu
```

CHAPTER 2

Django payments payu

NOTE: This project is still in development, so use with extreme caution.

PayU payments provider for django-payments. Uses the new PayU REST API. Supports normal, express and recurring payments.

2.1 Documentation

The full documentation is at https://django-payments-payu.readthedocs.io.

2.2 Quickstart

Install django-payments and set up PayU payment provider backend according to django-payments documentation:

class payments_payu.provider.**PayuProvider**(*client_secret*, *second_key*, *pos_id*[, *sandbox=False*, *endpoint="https://secure.payu.com/"*, *recurring_payments=False*, *express_payments=False*, *widget_branding=False*])

This backend implements payments using PayPal.com.

Parameters

- **client_secret** – PayU OAuth protocol client secret
- **pos_id** – PayU POS ID

- **second_key** – PayU second key (MD5)
- **sandbox** – if `True`, set the endpoint to sandbox
- **endpoint** – endpoint URL, if not set, the will be automatically set based on *sandbox* settings
- **recurring_payments** – enable recurring payments, only valid with `express_payments=True`, see bellow for additional setup, that is needed
- **express_payments** – use PayU express form
- **widget_branding** – tell express form to show PayU branding
- **store_card** – (default: False) whether PayU should store the card

Example:

```
# use sandbox
PAYMENT_VARIANTS = {
    'payu': ('payments_payu.provider.PayuProvider', {
        'pos_id': '123456',
        'second_key': 'iseedeadpeople',
        'client_secret': 'peopleiseedead',
        'sandbox': True,
        'capture': False,
    }),
}
```

Additional settings:

PayU requires users first name, last name and email. Override ether `get_user` or `get_user_email`, `get_user_first_name` and `get_user_last_name` methods from `BasePayment`.

NOTE: notifications about the payment status from PayU are requested to be sent to *django-payments process_payment* url. The request from PayU can fail for several reasons (i.e. it can be blocked by proxy). Use "Show reports" page in PayU administration to get more information about the requests.

Recurring payments:

If recurring payments are enabled, the PayU card token needs to be stored in your application for usage in next payments. The next payments can be either initiated by user through (user will be prompted only for payment confirmation by the express form) or by server. To enable recurring payments, you will need to set additional things:

NOTE: Recurring payments are not enabled by default even in Sandbox, you sould consult their helpdesk to enable this.

- In order to make payments recurring, the card token needs to be stored for the `Payment`'s user (not just the payment itself). Implement the `Payment.set_renew_token()` and `Payment.get_renew_token()`.
- Implement `Payment.get_payment_url()`.
- **For the server initiated recurring payments you will need to create the new payment and then call `payment.auto_`**
  - The method returns either string 'success' or url where the user can provide his CVV2 or 3D secure information.
  - The `'success'` string means, that the payment is waiting for notification from PayU, but no further user action is required.

Example of triggering recurring payment:

```
payment = Payment.objects.create(...)
redirect_url = payment.auto_complete_recurring()
if redirect_url != 'success':
    send_mail(
        'Recurring payment - action required',
        'Please renew your CVV2/3DS at %s' % redirect_url,
        'noreply@test.com',
        [user.email],
        fail_silently=False,
    )
```

2.3 Running Tests

Does the code actually work?

```
source <YOURVIRTUALENV>/bin/activate
(myenv) $ pip install tox
(myenv) $ tox
```

2.4 Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-djangopackage

CHAPTER 3

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

3.1 Types of Contributions

3.1.1 Report Bugs

Report bugs at https://github.com/PetrDlouhy/django-payments-payu/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

3.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

3.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

3.1.4 Write Documentation

Django payments payu could always use more documentation, whether as part of the official Django payments payu docs, in docstrings, or even on the web in blog posts, articles, and such.

3.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/PetrDlouhy/django-payments-payu/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

3.2 Get Started!

Ready to contribute? Here's how to set up *django-payments-payu* for local development.

1. Fork the *django-payments-payu* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-payments-payu.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-payments-payu
$ cd django-payments-payu/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 payments_payu tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

3.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/PetrDlouhy/django-payments-payu/pull_requests and make sure that the tests pass for all supported Python versions.

3.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_payments_payu
```

CHAPTER 4

Credits

4.1 Development Lead

- Petr Dlouhý <petr.dlouhy@email.cz>

4.2 Contributors

None yet. Why not be the first?

CHAPTER 5

History

5.1 0.3.0 (2020-05-30)

- fix amount quantization
- add store_card parameter
- fix base url parameter for express form

5.2 0.2.0 (2020-04-13)

- Second release
- Fixed testing matrix

5.3 0.1.0 (2020-04-06)

- First release on PyPI.
- Still in development.

Index

P