SMS for Pyramid Documentation

Release 0.1

Mikko Ohtamaa

September 09, 2016

Contents

Contents:

CHAPTER 1

SMS for Pyramid

SMS text messaging services for Pyramid web framework.

- Free software: ISC license
- Documentation: https://pyramid_sms.readthedocs.org.

1.1 Features

- Configurable SMS backend
- Send outbound SMS using Twilio or dummy backend for unit testing
- Asynchronous operations that do not block HTTP response with Websauna
- API type hinting

1.2 Author

Mikko Ohtamaa.

1.3 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Installing Python package

Python 3.x required.

At the command line:

```
pip install "pyramid_sms[twilio]"
```

2.2 Configuring outbound SMS with Twilio

In your Pyramid `INI` settings you can have

```
# Choose your SMS backend
sms.service = pyramid_sms.twilio.TwilioService

# Use this in test.ini for your unit test run
# sms.service = pyramid_sms.dummy.DummySMSService

# Twilio SMS number we have bought
sms.default_sender = +555123123

# Use Celery tasks fro async operating.
# If true doesn't block HTTP response.
# Requires Websauna.
sms.async = false

# Account SID in Twilio account settings
sms.twilio_account = xxx

# Auth Token in Twilio account settings
sms.twilio_token = yyy
```

CHAPTER 3

Usage

3.1 Setting up addon

In your Pyramid `__init__` add:

```
config.include("pyramid_sms")
```

3.2 API functions

- `pyramid_sms.outgoing.send_sms()`
- `pyramid_sms.outgoing.send_templated_sms()`
- `pyramid_sms.utils.normalize_us_phone_number()`
- `pyramid_sms.utils.normalize_international_phone_number()`

3.3 Validators

- `pyramid_sms.validators.valid_us_phone_number()`
- `pyramid_sms.validators.valid_international_phone_number()`

3.4 Events

See `pyramid_sms.events`

3.5 Self-contained command line script example

No INI settings file needed, you can copy-paste into Python shell:

```
from zope.interface import implementer

from pyramid.registry import Registry
from pyramid.interfaces import IRequest
```

```
from pyramid_sms.utils import normalize_us_phone_number
from pyramid_sms.outgoing import send_sms
from pyramid_sms.twilio import TwilioService
from pyramid_sms.interfaces import ISMSService

registry = Registry()
settings = registry.settings = dict()

# Twilio SMS number we have bought
settings["sms.default_sender"] = "+15551231234"

# Use Celery tasks fro async operating.
# If true doesn't block HTTP response.
# Requires Websauna.
settings["sms.async"] = False

# Account SID in Twilio account settings
settings["sms.twilio_account"] = "xxx"

# Auth Token in Twilio account settings
settings["sms.twilio_token"] = "yyy"

# Set up service backend
registry.registerAdapter(factory=TwilioService, required=(IRequest,), provided=ISMSService)

# Use request interface for send_sms
@implementer(IRequest)
class DummyRequest:
    registry = registry

request = DummyRequest()
to = normalize_us_phone_number("808 111 2222")
send_sms(request, to, "Hello there")
```

3.6 SMS login example

See this gist for a example how to implement Slack like "Magic link" like sign in with Websauna and Pyramid.

3.7 Testing

In `test.ini` or relevant set up for your test cases, configure `pyramid_sms.dummy.DummySMSService` backend according to *Installation*.

In your test case you can read back SMS sent to dummy backend.

Example:

```
import transaction
from sqlalchemy.orm.session import Session

from pyramid_sms.utils import get_sms_backend
from splinter.driver import DriverAPI
from websauna.system.user.models import User
from websauna.wallet.models.confirmation import UserNewPhoneNumberConfirmation
```

```
def test_ui_confirm_phone_number(require_phone_number, logged_in_wallet_user_browser: DriverAPI, dbse
    """User needs a confirmed phone number before entering the wallet."""

    # Run functional tests against a Waitress web server running in another thread
    b = logged_in_wallet_user_browser
    b.find_by_css("#nav-wallet").click()

    assert b.is_element_present_by_css("#heading-new-phone-number")
    b.fill("phone_number", "+15551231234")
    b.find_by_css("button[type='submit']").click()

    # We arrived to SMS code verification page
    assert b.is_element_present_by_css("#heading-confirm-phone-number")

    # We have a notification that SMS code was sent
    assert b.is_element_present_by_css("#msg-phone-confirmation-send")

    # Peek into SMS code
    with transaction.manager:
        user = dbsession.query(User).first()
        confirmation = UserNewPhoneNumberConfirmation.get_pending_confirmation(user)
        sms_code = confirmation.other_data["sms_code"]

    # Get a dummy SMS backend that's configured in test fixtures
    backend = get_sms_backend(test_request)

    # Make sure code got out to the user
    msg = backend.get_last_message()
    assert sms_code in msg

    # Enter the code
    b.fill("code", sms_code)
    b.find_by_css("button[type='submit']").click()

    # We arrived to wallet overview
    assert b.is_element_present_by_css("#heading-wallet-overview")

    # We have a notification for phone number verified
    assert b.is_element_present_by_css("#msg-phone-confirmed")
```

CHAPTER 4

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/websauna/pyramid_sms/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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/websauna/pyramid_sms/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 :)

4.2 Get Started!

Ready to contribute? Here's how to set up *pyramid_sms* for local development.

1. Fork the *pyramid_sms* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/pyramid_sms.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 pyramid_sms
$ cd pyramid_sms/
$ 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 pyramid_sms 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.

4.3 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 3.5, and for PyPy. Check https://travis-ci.org/miohtama/pyramid_sms/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

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

CHAPTER 5

Credits

5.1 Development Lead

- Mikko Ohtamaa <mikko@opensourcehacker.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.4 (unreleased)

- Nothing changed yet.

6.2 0.1.3 (2016-09-09)

- Update to Websauna new Celery APIs

6.3 0.1.2 (2016-06-29)

- Added validation and normalization for international phone number

6.4 0.1.1 (2016-06-25)

- Added validator `pyramid_sms.validators.valid_us_phone_number()`

6.5 0.1.0 (2016-04-28)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search