pyramid$_{pages}$

Release

July 06, 2015

Contents

1 Installing **3**
- 1.1 github . . . 3
- 1.2 PyPi . . . 3
- 1.3 source . . . 3

2 Configuration **5**
- 2.1 Custom model for tree pages . . . 5
- 2.2 Configure *pyramid_pages* . . . 5
- 2.3 Custom resource . . . 6
- 2.4 Generate menu . . . 6

3 API **11**
- 3.1 common . . . 11
- 3.2 models . . . 11
- 3.3 routes . . . 12
- 3.4 views . . . 13
- 3.5 security . . . 13
- 3.6 Module contents . . . 13

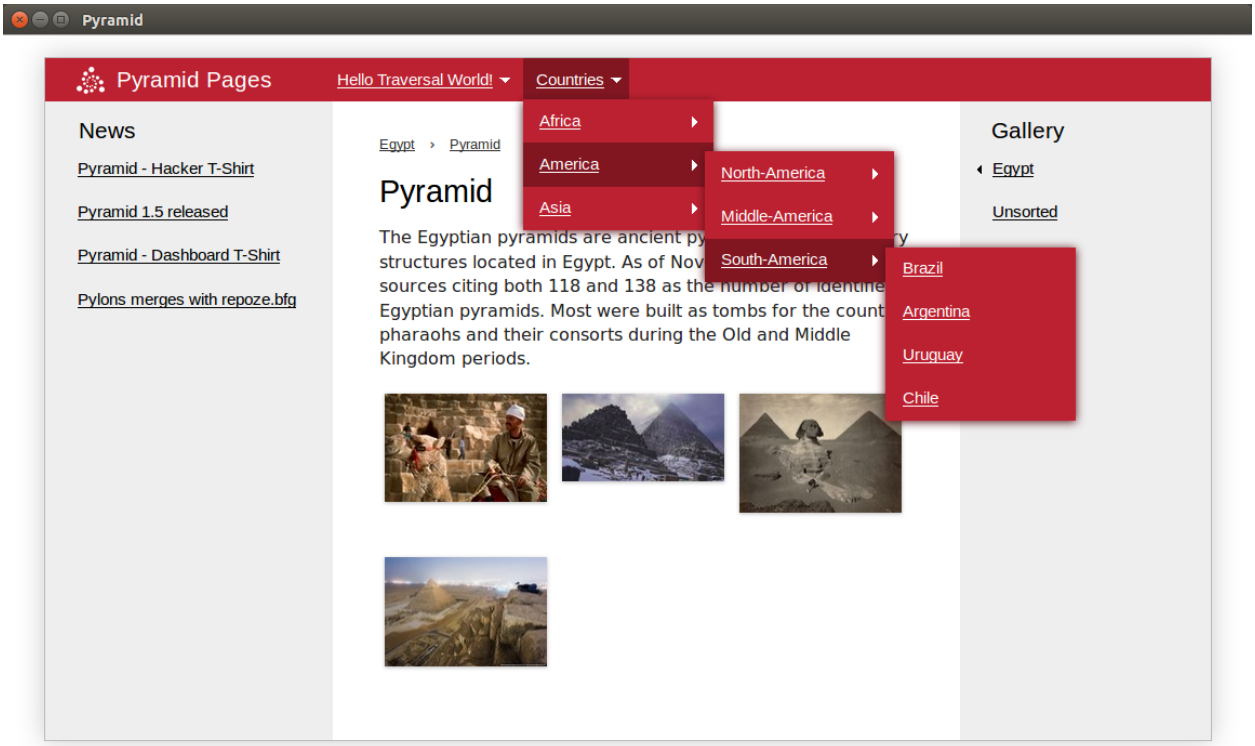
4 Tutorials and Cookbook Recipes **15**
- 4.1 Simple Web-site with tree pages . . . 15

5 Support and Development **19**

6 Indices and tables **21**

Python Module Index **23**

pyramid_pages provides a collections of pages to your Pyramid application. This is very similar to **django.contrib.flatpages** but with a tree structure and traversal algorithm in URL dispath.



CHAPTER 1

Installing

1.1 github

```
pip install git+http://github.com/ITCase/pyramid_pages.git
```

1.2 PyPi

```
pip install pyramid_pages
```

1.3 source

```
git clone git+http://github.com/ITCase/pyramid_pages.git
python setup.py develop
```

CHAPTER 2

Configuration

2.1 Custom model for tree pages

To build a tree, using the model from sqlalchemy_mptt.

Note:

Otherwise, it will look like flat pages.
Plans to add a recursive model with only *parent_id* field.

Create model of tree pages. For more detail see example pyramid_pages_example.

```
from pyramid_pages.models import FlatPageMixin, MpttPageMixin, RedirectMixin
from sqlalchemy_mptt import BaseNestedSets

...

class WebPage(Base, MpttPageMixin, RedirectMixin):
    __tablename__ = 'mptt_pages'

    id = Column('id', Integer, primary_key=True)


class NewsPage(Base, FlatPageMixin):
    __tablename__ = 'flat_news'

    id = Column('id', Integer, primary_key=True)
    date = Column(Date, default=func.now())
```

2.2 Configure *pyramid_pages*

Then add settings of *pyramid_pages*.

```
from youproject.models import MPTTPages, MPTTNews

...

settings['pyramid_pages.models'] = {
   '': WebPage,
   'pages': WebPage,  # available with prefix '/pages/'
```

```
    'news': NewsPage
}

# pyramid_pages - put it after all routes
# and after pyramid_pages configuration.
config.include("pyramid_pages")
```

If you use version of pyramid >= 1.6a1, there is a possibility put `config.include("pyramid_pages")` before pyramid_pages Configuration.

```
from youproject.models import MPTTPages, MPTTNews

...

config.include("pyramid_pages")
settings['pyramid_pages.models'] = {
    '': WebPage,
    'pages': WebPage,  # available with prefix '/pages/'
    'news': NewsPage
}
```

2.3 Custom resource

Base resource for pages can be found in the module *pyramid_pages.routes*.

Just inherit your resource from *PageResource*.

And add it to config.

```
settings['pyramid_pages.models'] = {
    '': WebPage,
    'pages': WebPage,  # available with prefix '/pages/'
    'news': NewsPage,
    'gallery': GalleryResource
}
```

2.4 Generate menu

Make menu object and pass it in template context.

```
from pyramid_pages.common import Menu


class Gallery(Base, MpttPageMixin):
    __tablename__ = 'mptt_gallery'

    menu_template = 'myproject/templates/my_custom_menu.mako'

    id = Column('id', Integer, primary_key=True)

page_menu = Menu(DBSession, WebPage).mptt
news_menu = Menu(DBSession, NewsPage).flat
gallery_menu = Menu(DBSession, Gallery).mptt
```

Just include menu template.

Listing 2.1: Base resource for pages.

```
class PageResource(object):

    template = 'pyramid_pages/index.jinja2'
    view = PageView
    attr = 'page_with_redirect'

    def __init__(self, node, prefix=None):
        self.node = node
        self.prefix = prefix

    @property
    def __name__(self):
        if self.node and not self.node.slug == '/':
            return self.node.slug
        elif self.node and self.node.slug == '/':
            return ''
        elif self.prefix:
            return self.prefix
        return None

    @property
    def __parent__(self):
        if hasattr(self.node, 'parent'):
            return self.__class__(self.node.parent, self.prefix)
        elif self.node:
            return self.__class__(None, self.prefix)
        return None

    def __getitem__(self, name):
        children = {str(child.slug or ''): self.__class__(child, self.prefix)
                    for child in self.node.children}
        return children[name]

    def __resource_url__(self, request, info):
        separator = '/' if self.prefix else ''
        # XXX: I feel a dissonance here
        info['virtual_path'] = re.sub('/+', '/', info['virtual_path'])
        url = info['app_url'] + separator + info['virtual_path']
        return url

    def __repr__(self):
        if not hasattr(self.node, 'name'):
            return '{}'.format(self.node)
        return "<{}>".format(self.node.name.encode('utf-8'))

    def get_prefix(self, request, node=None):
        if not node:
            node = self.node
        node = node.__class__
        settings = request.registry.settings
        models = settings[CONFIG_MODELS]
        reversed_models = dict(zip(models.values(), models.keys()))
        prefix = reversed_models.get(node, None)
        if prefix:
            return prefix

        # search prefix for resource
        for model in models.values():
            if not hasattr(model, '__table__') and hasattr(model, 'model')\
                    and model.model == node:
                return reversed_models.get(model, None)
        return None
```

Listing 2.2: Custom resource for gallery.

```
class GalleryResource(PageResource):
    model = Gallery
    template = 'gallery/index.jinja2'
```

Listing 2.3: Model for GalleryResource.

```
class Gallery(BasePage, MpttPageMixin):
    __tablename__ = 'mptt_gallery'

    id = Column(Integer, ForeignKey('base_pages.id'), primary_key=True)

    __mapper_args__ = {
        'polymorphic_identity': 'gallery_page',
    }
```

```
{% with menu=news_menu() %}
  {% include menu.template with context %}
{% endwith %}

{% with menu=page_menu(from_lvl=1, to_lvl=6, trees=(1, 2, 3)) %}
  {% include menu.template with context %}
{% endwith %}
```

Or write your own.

If you want show mptt menu with `to_lvl==max-2` or similar, just use `to_lvl=-2`.

```
{% with menu=page_menu(from_lvl=1, to_lvl=-2, trees=(1, 2, 3)) %}
  {% include menu.template with context %}
{% endwith %}
```

Listing 2.4: Model photo for Gallery.

```
class Photo(Base):
    __tablename__ = 'photos'

    id = Column('id', Integer, primary_key=True)
    path = Column('path', Text)
    gallery_id = Column(Integer, ForeignKey('mptt_gallery.id'))
    gallery = relationship('Gallery', backref='photos')
```

Listing 2.5: Flat menu template.

```
<div class="main-menu-list">
    {% for node in menu %}
        <div class="main-menu-list__item
                {% if node == context.node or node == page %}
                  main-menu-list__item_state_current
                {% endif %}">
            <a href="{{ request.resource_url(
                        context.__class__(node,
                        context.get_prefix(request, node=node))) }}"
              class="main-menu-list__item-link">
              {{ node.name }}
            </a>
        </div>
    {% endfor %}
</div>
```

Listing 2.6: MPTT menu template.

```
<div class="main-menu-list">
  {% for node, children in menu recursive %}
    <div class="main-menu-list__item
                {% if node == context.node or node == page %}
                  main-menu-list__item_state_current
                {% endif %}
                {% if children|length %}
                  main-menu-list__item_state_has-child
                {% endif %}">
        <a href="{{ request.resource_url(
                      context.__class__(node,
                      context.get_prefix(request, node=node))) }}"
          class="main-menu-list__item-link">
          {{ node.name }}
        </a>
      <div class="main-menu-list">{{ loop(children.items()) }}</div>
    </div>
  {% endfor %}
</div>
```

CHAPTER 3

API

3.1 common

Menu generator

class pyramid_pages.common.**Menu**(*session*, *model*)
Bases: object

flat()

mptt(*from_lvl=1*, ***kwargs*)

class pyramid_pages.common.**PageMenu**(*items*, *template*)
Bases: object

pyramid_pages.common.**recursive_node_to_dict**(*node*, *menu*, *json=None*, *json_fields=None*)

pyramid_pages.common.**sort_by_left**(*tree*)
Sort dict by left and tree_id field

3.2 models

Models for page.

class pyramid_pages.models.**BaseSacrudMpttPage**
Bases: *pyramid_pages.models.SeoMixin*, *pyramid_pages.models.RedirectMixin*, *pyramid_pages.models.MpttPageMixin*

verbose_name = 'MPTT pages'

class pyramid_pages.models.**FlatPageMixin**
Bases: *pyramid_pages.models.PageMixin*

get_menu()

menu_template = 'pyramid_pages/menu/flat.jinja2'

class pyramid_pages.models.**MpttPageMixin**
Bases: sqlalchemy_mptt.mixins.BaseNestedSets, *pyramid_pages.models.PageMixin*

get_menu(***kwargs*)

menu_template = 'pyramid_pages/menu/mptt.jinja2'

class pyramid_pages.models.**PageMixin**
Bases: object

description = Column(None, UnicodeText(), table=None)

in_menu = Column(None, Boolean(), table=None)

name = Column(None, String(), table=None, nullable=False)

slug = Column(None, SlugType(), table=None, nullable=False)

visible = Column(None, Boolean(), table=None)

class pyramid_pages.models.**RecursionPageMixin**
Bases: *pyramid_pages.models.PageMixin*

model with single parent_id field

class pyramid_pages.models.**RedirectMixin**
Bases: object

redirect_type = Column(None, ChoiceType(), table=None)

redirect_url = Column(None, String(), table=None)

class pyramid_pages.models.**SeoMixin**
Bases: object

seo_description = Column(None, String(), table=None)

seo_keywords = Column(None, String(), table=None)

seo_metatags = Column(None, UnicodeText(), table=None)

seo_title = Column(None, String(), table=None)

3.3 routes

Routes for sacrud_pages

class pyramid_pages.routes.**PageResource**(*node*, *prefix=None*)
Bases: object

attr = 'page_with_redirect'

get_prefix(*request*, *node=None*)

template = 'pyramid_pages/index.jinja2'

view
alias of PageView

pyramid_pages.routes.**home_page_factory**(*request*)

pyramid_pages.routes.**includeme**(*config*)

pyramid_pages.routes.**page_factory**(*request*)

pyramid_pages.routes.**register**(**args*)

3.4 views

Views for pages

class `pyramid_pages.views.`**`PageView`**(*context*, *request*)

Bases: `object`

`page_with_redirect`()

3.5 security

3.6 Module contents

`pyramid_pages.`**`includeme`**(*config*)

CHAPTER 4

Tutorials and Cookbook Recipes

4.1 Simple Web-site with tree pages

This is an example of using *pyramid_pages* in just one file (pyramid_pages_example.py). Full code you can see there.

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-
# vim:fenc=utf-8
#
# Copyright © 2014 uralbash <root@uralbash.ru>
#
# Distributed under terms of the MIT license.

"""
Main for example
"""
import json
import os

import transaction
from pyramid.config import Configurator
from pyramid.events import BeforeRender
from pyramid.location import lineage
from pyramid.session import SignedCookieSessionFactory
from sqlalchemy import (Column, Date, ForeignKey, Integer, String, Text,
                        engine_from_config)
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship, scoped_session, sessionmaker
from sqlalchemy.sql import func
from zope.sqlalchemy import ZopeTransactionExtension

from pyramid_pages.common import Menu
from pyramid_pages.models import FlatPageMixin, MpttPageMixin, RedirectMixin
from pyramid_pages.routes import PageResource
from sqlalchemy_mptt import mptt_sessionmaker

Base = declarative_base()
DBSession = scoped_session(
    mptt_sessionmaker(
        sessionmaker(extension=ZopeTransactionExtension())
    )
)

```

```
CONFIG_SQLALCHEMY_URL = 'sqlalchemy.url'
CONFIG_PYRAMID_PAGES_MODELS = 'pyramid_pages.models'
CONFIG_PYRAMID_PAGES_DBSESSION = 'pyramid_pages.dbsession'


class BasePage(Base, RedirectMixin):
    __tablename__ = 'base_pages'
    id = Column(Integer, primary_key=True)
    page_type = Column(String(50))

    __mapper_args__ = {
        'polymorphic_identity': 'base_page',
        'polymorphic_on': page_type
    }

    @classmethod
    def get_pk_name(cls):
        return 'id'

    @classmethod
    def get_pk_with_class_name(cls):
        return 'BasePage.id'


class WebPage(BasePage, MpttPageMixin):
    __tablename__ = 'mptt_pages'

    id = Column(Integer, ForeignKey('base_pages.id'), primary_key=True)

    __mapper_args__ = {
        'polymorphic_identity': 'web_page',
    }


class NewsPage(BasePage, FlatPageMixin):
    __tablename__ = 'flat_news'

    id = Column(Integer, ForeignKey('base_pages.id'), primary_key=True)
    date = Column(Date, default=func.now())

    __mapper_args__ = {
        'polymorphic_identity': 'news_page',
    }


class Gallery(BasePage, MpttPageMixin):
    __tablename__ = 'mptt_gallery'

    id = Column(Integer, ForeignKey('base_pages.id'), primary_key=True)

    __mapper_args__ = {
        'polymorphic_identity': 'gallery_page',
    }


class Photo(Base):
    __tablename__ = 'photos'

```

```
    id = Column('id', Integer, primary_key=True)
    path = Column('path', Text)
    gallery_id = Column(Integer, ForeignKey('mptt_gallery.id'))
    gallery = relationship('Gallery', backref='photos')


class GalleryResource(PageResource):
    model = Gallery
    template = 'gallery/index.jinja2'


class NewsResource(PageResource):
    model = NewsPage
    template = 'news/index.jinja2'


class Fixtures(object):

    def __init__(self, session):
        self.session = session

    def add(self, model, fixtures):
        here = os.path.dirname(os.path.realpath(__file__))
        file = open(os.path.join(here, fixtures))
        fixtures = json.loads(file.read())
        for fixture in fixtures:
            self.session.add(model(**fixture))
            self.session.flush()
        transaction.commit()


def add_globals(event):
    event['lineage'] = lineage
    event['page_menu'] = Menu(DBSession, WebPage).mptt
    event['news_menu'] = Menu(DBSession, NewsPage).flat
    event['gallery_menu'] = Menu(DBSession, Gallery).mptt


models = {
    '': WebPage,
    'pages': WebPage,
    'news': NewsResource,
    'gallery': GalleryResource,
}


def main(global_settings, **settings):
    config = Configurator(
        settings=settings,
        session_factory=SignedCookieSessionFactory('itsaseekreet')
    )
    config.include('pyramid_jinja2')
    config.add_jinja2_search_path('pyramid_pages_example:templates')
    config.add_static_view('pyramid_pages_example_static',
                           'static')

    # Database
    settings = config.get_settings()
```

```
    settings[CONFIG_SQLALCHEMY_URL] =\
        settings.get(CONFIG_SQLALCHEMY_URL,
                     'sqlite:///example.sqlite')
    engine = engine_from_config(settings)
    DBSession.configure(bind=engine)
    Base.metadata.drop_all(engine)
    Base.metadata.create_all(engine)
    fixture = Fixtures(DBSession)
    fixture.add(WebPage, 'fixtures/pages.json')
    fixture.add(WebPage, 'fixtures/country.json')
    fixture.add(NewsPage, 'fixtures/news.json')
    fixture.add(Gallery, 'fixtures/gallery.json')
    fixture.add(Photo, 'fixtures/photos.json')

    # pyramid_pages
    settings[CONFIG_PYRAMID_PAGES_DBSESSION] =\
        settings.get(CONFIG_PYRAMID_PAGES_DBSESSION,
                     DBSession)
    settings[CONFIG_PYRAMID_PAGES_MODELS] =\
        settings.get(CONFIG_PYRAMID_PAGES_MODELS, models)
    config.include("pyramid_pages")
    config.add_subscriber(add_globals, BeforeRender)
    return config.make_wsgi_app()

if __name__ == '__main__':
    settings = {}
    app = main({}, **settings)

    from wsgiref.simple_server import make_server
    httpd = make_server('0.0.0.0', 6543, app)
    httpd.serve_forever()
```

CHAPTER 5

Support and Development

To report bugs, use the issue tracker or waffle board.

We welcome any contribution: suggestions, ideas, commits with new futures, bug fixes, refactoring, docs, tests, translations etc

If you have question, contact me sacrud@uralbash.ru or IRC channel #sacrud

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

p

pyramid_pages, 13
pyramid_pages.common, 11
pyramid_pages.models, 11
pyramid_pages.routes, 12
pyramid_pages.security, 13
pyramid_pages.views, 13

A

attr (pyramid_pages.routes.PageResource attribute), 12

B

BaseSacrudMpttPage (class in pyramid_pages.models), 11

D

description (pyramid_pages.models.PageMixin attribute), 12

F

flat() (pyramid_pages.common.Menu method), 11
FlatPageMixin (class in pyramid_pages.models), 11

G

get_menu() (pyramid_pages.models.FlatPageMixin method), 11
get_menu() (pyramid_pages.models.MpttPageMixin method), 11
get_prefix() (pyramid_pages.routes.PageResource method), 12

H

home_page_factory() (in module pyramid_pages.routes), 12

I

in_menu (pyramid_pages.models.PageMixin attribute), 12
includeme() (in module pyramid_pages), 13
includeme() (in module pyramid_pages.routes), 12

M

Menu (class in pyramid_pages.common), 11
menu_template (pyramid_pages.models.FlatPageMixin attribute), 11
menu_template (pyramid_pages.models.MpttPageMixin attribute), 11
mptt() (pyramid_pages.common.Menu method), 11
MpttPageMixin (class in pyramid_pages.models), 11

N

name (pyramid_pages.models.PageMixin attribute), 12

P

page_factory() (in module pyramid_pages.routes), 12
page_with_redirect() (pyramid_pages.views.PageView method), 13
PageMenu (class in pyramid_pages.common), 11
PageMixin (class in pyramid_pages.models), 11
PageResource (class in pyramid_pages.routes), 12
PageView (class in pyramid_pages.views), 13
pyramid_pages (module), 13
pyramid_pages.common (module), 11
pyramid_pages.models (module), 11
pyramid_pages.routes (module), 12
pyramid_pages.security (module), 13
pyramid_pages.views (module), 13

R

RecursionPageMixin (class in pyramid_pages.models), 12
recursive_node_to_dict() (in module pyramid_pages.common), 11
redirect_type (pyramid_pages.models.RedirectMixin attribute), 12
redirect_url (pyramid_pages.models.RedirectMixin attribute), 12
RedirectMixin (class in pyramid_pages.models), 12
register() (in module pyramid_pages.routes), 12

S

seo_description (pyramid_pages.models.SeoMixin attribute), 12
seo_keywords (pyramid_pages.models.SeoMixin attribute), 12
seo_metatags (pyramid_pages.models.SeoMixin attribute), 12
seo_title (pyramid_pages.models.SeoMixin attribute), 12

SeoMixin (class in pyramid_pages.models), 12
slug (pyramid_pages.models.PageMixin attribute), 12
sort_by_left() (in module pyramid_pages.common), 11

T

template (pyramid_pages.routes.PageResource attribute), 12

V

verbose_name (pyramid_pages.models.BaseSacrudMpttPage attribute), 11
view (pyramid_pages.routes.PageResource attribute), 12
visible (pyramid_pages.models.PageMixin attribute), 12