# pyramid-katas Documentation

***Release 0.0***

**Atsushi Odagiri**

June 12, 2016

Contents

Contents:

CHAPTER 1

# Pyramid katas

## 1.1

- Python 3.4.3

## 1.2

pypa

UNIX

```
pyvenv .venv
. .venv/bin/activate
pip install -U pip
pip install -U setuptools
```

windows (powershell)

```
c:\python34\python c:\python34\tools\scripts\pyvenv.py .venv
.venv\scripts\activate.ps1
python -m pip install -U pip
python -m pip install -U setuptools
```

:

```
pip install wheel
pip wheel wheel
```

## 1.3 WSGI

waitress:

```
pip wheel -f wheelhouse waitress
pip install -f wheelhouse waitress
```

:

```
waitress-serve -h
```

wsgi

```
def application(environ, start_response):
    start_response("200 OK",
                   [('Content-type', 'text/plain')])
    return [b"Hello, world!"]
```

Web:

```
waitress-serve wsgiapp.application
```

## 1.4 webob

webob:

```
pip wheel webob -f wheelhouse
pip install webob -f wheelhouse
```

webobwsgi

```
from webob import Request, Response


def application(environ, start_response):
    request = Request(environ)
    response = Response(request=request)
    response.text = "Hello, world!"
    return response(environ, start_response)
```

webob.dec.wsgifywsgi

```
from webob.dec import wsgify


@wsgify
def application(request):
    request.response.text = "Hello, world!"
    return request.response
```

## 1.5 pyramid

pyramid:

```
pip wheel -f wheelhouse pyramid
pip install -f wheelhouse pyramid
```

pyramid

```
from pyramid.config import Configurator


def index(request):
    request.response.text = "Hello, world!"
    return request.response

config = Configurator()
```

```
config.add_view(index)
application = config.make_wsgi_app()
```

## 1.6 pyramid

- myapp1
  - __init__.py
  - wsgi.py
  - views.py

__init__.py:

```
from pyramid.config import Configurator


def main(global_conf, **settings):
    config = Configurator(
        settings=settings)
    config.scan()
    return config.make_wsgi_app()
```

views.py: web

```
from pyramid.view import view_config


@view_config()
def index(request):
    request.response.text = "Hello, world!"
    return request.response
```

wsgi.py: wsgi

```
from . import main

settings = {
}

application = main({}, **settings)
```

## 1.7 URL

- myapp2
  - __init__.py
  - wsgi.py
  - views.py

  __init__.py: add_routeURL

```
from pyramid.config import Configurator


def main(global_conf, **settings):
    config = Configurator(
        settings=settings)
    config.add_route('top', '/')
    config.add_route('user', '/users/{username}')
    config.scan()
    return config.make_wsgi_app()
```

views.py: view_configroute

```
from pyramid.view import view_config


@view_config(route_name="top")
def index(request):
    request.response.text = "Hello, world!"
    return request.response


@view_config(route_name="user")
def user(request):
    username = request.matchdict["username"]
    request.response.text = "Hello, {username}!".format(username=username)
    return request.response
```

wsgi.pymyapp1

## 1.8 HTML

pyramid_jinja2:

```
pip wheel pyramid_jinja2 -f wheelhouse
pip install pyramid_jinja2 -f wheelhouse
```

- myapp3
  - __init__.py
  - wsgi.py
  - views.py
  - templates
    - index.jinja2
    - user.jinja2

__init__.py: pyramid_jinja2 include

```
from pyramid.config import Configurator


def main(global_conf, **settings):
    config = Configurator(
        settings=settings)
```

```
    config.include("pyramid_jinja2")
    config.add_route('top', '/')
    config.add_route('user', '/users/{username}')
    config.scan()
    return config.make_wsgi_app()
```

views.py: view_configrenderer

```
from pyramid.view import view_config


@view_config(route_name="top",
             renderer='templates/index.jinja2')
def index(request):
    return dict()


@view_config(route_name="user",
             renderer='templates/user.jinja2')
def user(request):
    username = request.matchdict["username"]
    return dict(username=username)
```

(username)

```
Hello, {{ username }}!
```

## 1.9

pyramid_sqlalchemypyramid_tm:

```
pip wheel -f wheelhouse pyramid_sqlalchemy pyramid_tm
pip install -f wheelhouse pyramid_sqlalchemy pyramid_tm
```

- myapp4
  - __init__.py
  - wsgi.py
  - models.py
  - views.py
  - templates
    - index.jinja2
    - user.jinja2

__init__.py: pyramid_sqlalchemy, pyramid_tm include

```
from pyramid.config import Configurator


def main(global_conf, **settings):
    config = Configurator(
        settings=settings)
    config.include("pyramid_tm")
    config.include("pyramid_sqlalchemy")
```

```
    config.include("pyramid_jinja2")
    config.add_route('top', '/')
    config.add_route('user', '/users/{username}')
    config.scan()
    return config.make_wsgi_app()
```

wsgi.py: sqlalchemy.url

```
import os
from . import main

here = os.getcwd()

settings = {
    'sqlalchemy.url': 'sqlite:///{here}/myapp4.sqlite'.format(here=here),
    'sqlalchemy.echo': True,
}

application = main({}, **settings)
```

models.py:

```
from sqlalchemy import (
    Column,
    Integer,
    Unicode,
)
from pyramid_sqlalchemy import (
    BaseObject,
    Session,
)


class User(BaseObject):
    __tablename__ = 'users'
    query = Session.query_property()
    id = Column(Integer, primary_key=True)
    username = Column(Unicode(255), unique=True)
```

views.py: User.query

```
from pyramid.view import view_config
from pyramid.httpexceptions import HTTPNotFound
from .models import User


@view_config(route_name="top",
             renderer='templates/index.jinja2')
def index(request):
    return dict()


@view_config(route_name="user",
             renderer='templates/user.jinja2')
def user(request):
    username = request.matchdict["username"]
    user = User.query.filter(User.username == username).first()
    if user is None:
        raise HTTPNotFound
```

```
    return dict(user=user)
```

user.jinja2: user

```
Hello, {{ user.username }}!
```

:

```
>>> from myapp4 import main
>>> import os
>>> here = os.getcwd()
>>> main({}, **{'sqlalchemy.url': 'sqlite:///{here}/myapp4.sqlite'.format(here=here)}
>>> from myapp4 import models
>>> models.BaseObject.metadata.create_all()
```

:

```
>>> user = models.User(username="aodag")
>>> models.Session.add(user)
>>> models.Session.flush()
>>> import transaction
>>> transaction.commit()
>>> models.User.query.all()
```

## 1.10

pyramid_deform colanderalchemy :

```
pip wheel -f wheelhouse "deform>=2.0dev" pyramid_deform colanderalchemy
```

- myapp5
  - __init__.py
  - wsgi.py
  - models.py
  - views.py
  - templates
    - index.jinja2
    - new_user.jinja2
    - users.jinja2
    - user.jinja2

__init__.py: pyramid_deform include

```
from pyramid.config import Configurator


def main(global_conf, **settings):
    config = Configurator(
        settings=settings)
    config.include("pyramid_tm")
    config.include("pyramid_sqlalchemy")
```

```
    config.include("pyramid_jinja2")
    config.include("pyramid_deform")
    config.add_route('top', '/')
    config.add_route('users', '/users')
    config.add_route('new_user', '/new_user')
    config.add_route('user', '/users/{username}')
    config.scan()
    return config.make_wsgi_app()
```

views.py: FormViewdeform

```
from pyramid.view import view_config
from pyramid.httpexceptions import (
    HTTPNotFound,
    HTTPFound,
)
from pyramid_deform import FormView
from colanderalchemy import SQLAlchemySchemaNode
from .models import User


@view_config(route_name="top",
             renderer='templates/index.jinja2')
def index(request):
    return dict()


@view_config(route_name="user",
             renderer='templates/user.jinja2')
def user(request):
    username = request.matchdict["username"]
    user = User.query.filter(User.username == username).first()
    if user is None:
        raise HTTPNotFound

    return dict(user=user)


@view_config(route_name="users",
             renderer="templates/users.jinja2")
def users(request):
    users = User.query.all()
    return dict(users=users)


@view_config(route_name="new_user",
             renderer="templates/new_user.jinja2")
class NewUserForm(FormView):
    schema = SQLAlchemySchemaNode(User,
                                  excludes=['id'])
    buttons = ('add',)

    def add_success(self, values):
        user = User(**values)
        user.query.session.add(user)
        return HTTPFound(self.request.route_url('users'))
```

new_user.jinja2:

```
{{ form|safe }}
```

## 1.11

pyramid_layout :

```
pip wheel -f wheelhouse pyramid_layout
pip install -f wheelhouse pyramid_layout
```

- myapp6
  - __init__.py
  - wsgi.py
  - models.py
  - views.py
  - layouts.py
  - templates
    - base.jinja2
    - index.jinja2
    - new_user.jinja2
    - users.jinja2
    - user.jinja2

__init__.py: pyramid_layout include

```
from pyramid.config import Configurator


def main(global_conf, **settings):
    config = Configurator(
        settings=settings)
    config.include("pyramid_tm")
    config.include("pyramid_sqlalchemy")
    config.include("pyramid_jinja2")
    config.include("pyramid_deform")
    config.include("pyramid_layout")
    config.add_route('top', '/')
    config.add_route('users', '/users')
    config.add_route('new_user', '/new_user')
    config.add_route('user', '/users/{username}')
    config.scan()
    return config.make_wsgi_app()
```

layouts.py: BaseLayout

```
from pyramid_layout.layout import layout_config


@layout_config(template="templates/base.jinja2")
class BaseLayout(object):
    def __init__(self, context, request):
```

```
        self.context = context
        self.request = request
```

BaseLayout

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
          href="{{ request.static_url('deform:static/css/bootstrap.min.css') }}">
    {% if css_links %}
    {% for css in css_links %}
    <link rel="stylesheet"
          href="{{ request.static_url(css) }}">
    {% endfor %}
    {% endif %}

    <script src="{{ request.static_url('deform:static/scripts/jquery-2.0.3.min.js') }}"></script>
    <script src="{{ request.static_url('deform:static/scripts/bootstrap.min.js') }}"></script>

    {% if js_links %}
    {% for js in js_links %}
    <script src="{{ request.static_url(js) }}" ></script>
    {% endfor %}
    {% endif %}
  </head>
  <body>
    <div class="container">
      {% block main_contents %}{% endblock %}
    </div>
  </body>
</html>
```

()

```
{% extends main_template %}
{% block main_contents %}
Hello, world!
{% endblock %}
```

## 1.12

- myapp7
  - __init__.py
  - wsgi.py
  - models.py
  - views.py
  - layouts.py
  - templates
    - base.jinja2
    - index.jinja2

* new_user.jinja2
* users.jinja2
* user.jinja2

alembic:

```
pip wheel -f wheelhouse alembic
pip install -f wheelhouse alembic
```

:

```
alembic init alembic
```

alembic.ini sqlalchemy.url

```
# A generic, single database configuration.

[alembic]
# path to migration scripts
script_location = alembic

# template used to generate migration files
# file_template = %%(rev)s_%%(slug)s

# max length of characters to apply to the
# "slug" field
#truncate_slug_length = 40

# set to 'true' to run the environment during
# the 'revision' command, regardless of autogenerate
# revision_environment = false

# set to 'true' to allow .pyc and .pyo files without
# a source .py file to be detected as revisions in the
# versions/ directory
# sourceless = false

# version location specification; this defaults
# to alembic/versions.  When using multiple version
# directories, initial revisions must be specified with --version-path
# version_locations = %(here)s/bar %(here)s/bat alembic/versions

# the output encoding used when revision files
# are written from script.py.mako
# output_encoding = utf-8

sqlalchemy.url = sqlite:///%(here)s/myapp7.sqlite


# Logging configuration
[loggers]
keys = root,sqlalchemy,alembic

[handlers]
keys = console

[formatters]
keys = generic
```

```
[logger_root]
level = WARN
handlers = console
qualname =

[logger_sqlalchemy]
level = WARN
handlers =
qualname = sqlalchemy.engine

[logger_alembic]
level = INFO
handlers =
qualname = alembic

[handler_console]
class = StreamHandler
args = (sys.stderr,)
level = NOTSET
formatter = generic

[formatter_generic]
format = %(levelname)-5.5s [%(name)s] %(message)s
datefmt = %H:%M:%S
```

alembic/env.py: target_schemametadata

```
from myapp7 import models
target_metadata = models.BaseObject.metadata
```

:

```
alembic revision --autogenerate -m "first models"
```

:

```
alembic upgrade head
alembic history
```

models.py: (birthday)

```
from sqlalchemy import (
    Column,
    Integer,
    Date,
    Unicode,
)
from pyramid_sqlalchemy import (
    BaseObject,
    Session,
)


class User(BaseObject):
    __tablename__ = 'users'
    query = Session.query_property()
    id = Column(Integer, primary_key=True)
    username = Column(Unicode(255), unique=True)
    birthday = Column(Date)
```

:

```
alembic revision --autogenerate -m "user birthday"
```

:

```
alembic upgrade head
alembic history
```

CHAPTER 2

# Indices and tables

- genindex
- modindex
- search