Do pônei ao faraó, Pyramid Web Framework

Quem sou eu?

- Mário Idival
- Desde 2011
- Time da documentação da linguagem Rust
- Rust, Python, Go, JS ...
- Dota pre-pro-player


WEB APPLICATION FRAMEWORK
PYRAMID
NOT BUILT BY ALIENS
Pyramid
Pylons
python

Por que aprender Pyramid?

Tem as melhores camisetas


WEB APPLICATION FRAMEWORK
PYRAMID
NOT BUILT BY ALIENS


PYRAMID


PYRAMID
WEB APPLICATION FRAMEWORK
PYLONSPROJECT.ORG


PYRAMID
• WEB FRAMEWORK • PYLONSPROJECT.ORG •

É feito por humanos

Apesar de ser uma brincadeira sobre o framework, essa frase é usada pois o source code do Pyramid é simples, diferente de outros frameworks.

Altamente testado e documentado

Os core commiters do projeto são rigorosos sobre testes e documentação. Desde seu início, o projeto nunca ficou com a cobertura de testes abaixo de 97% e todo nova feature só é mergeada se tiver documentada.

Comece pequeno, termine grande

Você precisa de um web framework Python que dê **suporte a suas decisões.**

Pyramid foi “pensado” para projetos ambiciosos.

Bora começar

Quick start

```
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response

def hello_world(request):
    return Response('Hello World!')

if __name__ == '__main__':
    with Configurator() as config:
        config.add_route('hello', '/')
        config.add_view(hello_world, route_name='hello')
        app = config.make_wsgi_app()
    server = make_server('0.0.0.0', 6543, app)
    server.serve_forever()
```

Features

- Views
- Routing
- Templates
- Static assets
- Scaffolding
- Configuration with .ini
- Security & Sessions
- Databases

views

o que são?

No Pyramid, views é uma maneira primária de receber web requests e retornar respostas

```python
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response

def hello_world(request):
    return Response('Hello World!')

if __name__ == '__main__':
    with Configurator() as config:
        config.add_route('hello', '/')
        config.add_view(hello_world, route_name='hello')
        app = config.make_wsgi_app()
    server = make_server('0.0.0.0', 6543, app)
    server.serve_forever()
```

```python
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response


def hello_world(request):
    return Response('Hello World!')


def goodbye_world(request):
    return Response('Goodbye World!')


if __name__ == '__main__':
    with Configurator() as config:
        config.add_route('hello', '/')
        config.add_view(hello_world, route_name='hello')
        config.add_route('sad', '/sad')
        config.add_view(goodbye_world, route_name='sad')
        app = config.make_wsgi_app()
    server = make_server('0.0.0.0', 6543, app)
    server.serve_forever()
```

function decorators

```python
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response
from pyramid.view import view_config


@view_config(route_name='hello')
def hello_world(request):
    return Response('Hello World!')


@view_config(route_name='sad')
def goodbye_world(request):
    return Response('Goodbye World!')


if __name__ == '__main__':
    with Configurator() as config:
        config.add_route('hello', '/')
        config.add_route('sad', '/sad')
        config.scan()

        app = config.make_wsgi_app()
    server = make_server('0.0.0.0', 6543, app)
    server.serve_forever()
```

routing

como criar?

```
config.add_route('hello', '/')
config.add_route('sad', '/sad/{name}')
```

```
config.add_route('hello', '/')
config.add_view(hello_world, route_name='hello')

config.add_route('sad', '/sad/{name}')
config.add_view(goodbye_world, route_name='sad')
```

```python
def hello_world(request):
    return Response('Hello World!')


def goodbye_world(request):
    return Response(f'Goodbye World! {request.matchdict["name"]}')


if __name__ == '__main__':
    with Configurator() as config:

        config.add_route('hello', '/')
        config.add_view(hello_world, route_name='hello')

        config.add_route('sad', '/sad/{name}')
        config.add_view(goodbye_world, route_name='sad')
```

Acessando partes da URL

templates

templates (engine)

- Jinja2
- Mako
- Chameleon

jinja2

```python
def hello_world(request):
    return {'message': 'Hello World!'}


def goodbye_world(request):
    return {'name': request.matchdict['name']}
```

```python
with Configurator() as config:
    config.include('pyramid_jinja2')

    config.add_route('hello', '/')
    config.add_view(
        hello_world,
        route_name='hello',
        renderer='templates/home.jinja2'
    )

    config.add_route('sad', '/sad/{name}')
    config.add_view(
        goodbye_world,
        route_name='sad',
        renderer='templates/sad.jinja2'
    )
```

jinja2

```
@view_config(route_name='hello', renderer='templates/home.jinja2')
def hello_world(request):
    return {'message': 'Hello World!'}


@view_config(route_name='hello', renderer='templates/sad.jinja2')
def goodbye_world(request):
    return {'name': request.matchdict['name']}


if __name__ == '__main__':
    with Configurator() as config:
        config.include('pyramid_jinja2')

        config.add_route('hello', '/')
        config.add_route('sad', '/sad/{name}')
        config.scan()

        app = config.make_wsgi_app()
    server = make_server('0.0.0.0', 6543, app)
    server.serve_forever()
```

static assets

pra que?

Usado para informar o ponto de onde você vai servir os arquivos estáticos com JS, CSS e Imagens.

```
with Configurator() as config:
    config.include('pyramid_jinja2')
    config.add_static_view(name='static', path='static')

    config.add_route('hello', '/')
    config.add_route('sad', '/sad/{name}')
    config.scan()
```

scaffolding

o que é?

Chega de colocar tudo em um arquivo só. Com scaffolds, você pode criar estruturas de projetos como quiser.

- pip install cookiecutter

Usando o pyramid-cookiecutter-starter

```
$ cookiecutter gh:Pylons/pyramid-cookiecutter-starter --checkout 1.9-branch

project_name [Pyramid Scaffold]: nomedoprojeto

repo_name [nomedoprojeto]:

Select template_language:

1 - jinja2
2 - chameleon
3 - mako

Choose from 1, 2, 3 [1]:
```

Usando o pyramid-cookiecutter-starter

Change directory into your newly created project.
```
cd nomedoprojeto
```

Create a Python virtual environment.
```
python3 -m venv env
```

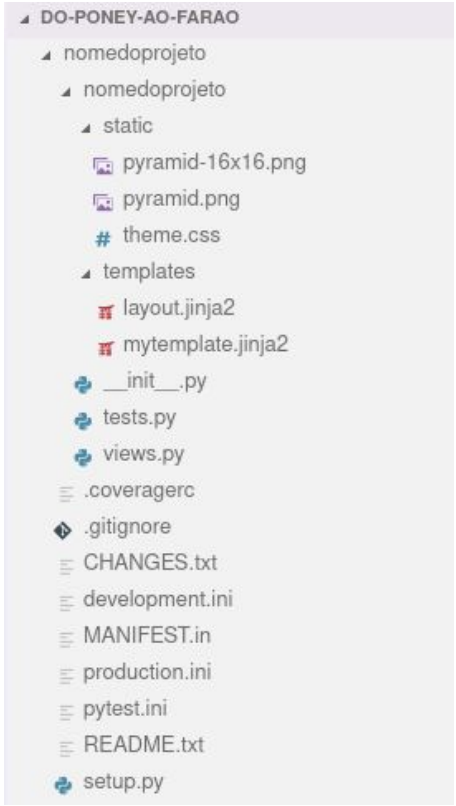
Upgrade packaging tools.
```
env/bin/pip install --upgrade pip setuptools
```

Install the project in editable mode with its testing requirements.
```
env/bin/pip install -e ".[testing]"
```

Run your project's tests.
```
env/bin/pytest
```

Run your project.
```
env/bin/pserve development.ini
```

Usando o pyramid-cookiecutter-started



configuration with ini files

o que é isso?

Basicamente o .ini é usado para configuração.

```
# app configuration
[app:main]
use = egg:nomedoprojeto

pyramid.reload_templates = true
pyramid.debug_authorization = false
pyramid.debug_notfound = false
pyramid.debug_routematch = false
pyramid.default_locale_name = en
pyramid.includes =
    pyramid_debugtoolbar

# wsgi server configuration
[server:main]
use = egg:waitress#main
listen = localhost:6543
```

```python
from pyramid.config import Configurator


def main(global_config, **settings):
    """ This function returns a Pyramid WSGI application.
    """
    config = Configurator(settings=settings)
    config.include('pyramid_jinja2')
    config.add_static_view('static', 'static', cache_max_age=3600)
    config.add_route('home', '/')
    config.scan()
    return config.make_wsgi_app()
```

Security & Sessions

Sessions

o que é?

É um “namespace” que é válido por um período de atividade contínua que pode ser usado para representar uma interação do usuário com a web app.

- carrinho de compras

```python
from pyramid.config import Configurator
from pyramid.session import SignedCookieSessionFactory


def main(global_config, **settings):
    """ This function returns a Pyramid WSGI application.
    """
    config = Configurator(settings=settings)
    my_factory_session = SignedCookieSessionFactory('isascretekey')
    config.set_session_factory(my_factory_session)

    config.include('pyramid_jinja2')
    config.add_static_view('static', 'static', cache_max_age=3600)
    config.add_route('home', '/')
    config.scan()
    return config.make_wsgi_app()
```

como adicionar

```python
from pyramid.view import view_config


@view_config(route_name='home', renderer='templates/mytemplate.jinja2')
def my_view(request):
    session = request.session
    if 'counter' in session:
        session['counter'] += 1
    else:
        session['counter'] = 0
    return {'project': 'nomedoprojeto'}
```

como usar

warning

Pyramid implementa uma versão insegura, pois ele utiliza os cookies do browser para isso, que por padrão não é encriptado e também pode ser visto por todos que tem acesso ao “cookie storage” do usuário.

Porém, ele é digitalmente assinado e portanto não será fácil adulterar esses dados.

Security

o que é?

Security no Pyramid é separado em Autenticação e Autorização. Eles se comunicam através de “identificadores principais”

Um principal é uma string ou um objeto Unicode representando uma entidade, geralmente um usuário ou grupo. Os principais são fornecidos por uma política de autenticação.

Por exemplo, se um usuário tiver o userid bob e for um membro de dois grupos denominados group foo e group bar, a solicitação poderá ter informações anexadas a ele, indicando que Bob foi representado por três principais: bob, group foo e group bar

Security

Autenticação

- é um mecanismo pelo qual credenciais são fornecidas por um request e são resolvidas para um ou mais “identificadores principais”

Autorização

- determina quem pode acessar determinado recurso, baseado no seu “principal”

adicionando autorização e autenticação

```python
from pyramid.config import Configurator
from pyramid.session import SignedCookieSessionFactory
from pyramid.authentication import AuthTktAuthenticationPolicy
from pyramid.authorization import ACLAuthorizationPolicy


def main(global_config, **settings):
    """ This function returns a Pyramid WSGI application.
    """

    authn_policy = AuthTktAuthenticationPolicy('seekrit', hashalg='sha512')
    authz_policy = ACLAuthorizationPolicy()
    config = Configurator(settings=settings)

    config.set_authentication_policy(authn_policy)
    config.set_authorization_policy(authz_policy)

    my_factory_session = SignedCookieSessionFactory('isascretekey')
    config.set_session_factory(my_factory_session)

    config.include('pyramid_jinja2')
    config.add_static_view('static', 'static', cache_max_age=3600)
    config.add_route('home', '/')
    config.scan()
    return config.make_wsgi_app()
```

adicionando autorização e autenticação como na vida real

```python
from pyramid.config import Configurator
from pyramid.authentication import AuthTktAuthenticationPolicy
from pyramid.authorization import ACLAuthorizationPolicy
from .security import groupfinder


def main(global_config, **settings):
    """ This function returns a Pyramid WSGI application.
    """

    authn_policy = AuthTktAuthenticationPolicy(
        'seekrit',
        callback=groupfinder,
        hashalg='sha512',
    )
    authz_policy = ACLAuthorizationPolicy()
    config = Configurator(settings=settings)

    config.set_authentication_policy(authn_policy)
    config.set_authorization_policy(authz_policy)

    config.include('pyramid_jinja2')
    config.add_static_view('static', 'static', cache_max_age=3600)
    config.add_route('home', '/')
    config.add_route('login', '/login')
    config.add_route('logout', '/logout')
    config.scan()
    return config.make_wsgi_app()
```

```python
import bcrypt


def hash_password(pw):
    pwhash = bcrypt.hashpw(pw.encode('utf8'), bcrypt.gensalt())
    return pwhash.decode('utf8')

def check_password(pw, hashed_pw):
    expected_hash = hashed_pw.encode('utf8')
    return bcrypt.checkpw(pw.encode('utf8'), expected_hash)


USERS = {
    'editor': hash_password('editor'),
    'viewer': hash_password('viewer')
}
GROUPS = {'editor': ['group:editors']}


def groupfinder(userid, request):
    if userid in USERS:
        return GROUPS.get(userid, [])
```

login/logout views

```python
from pyramid.view import view_config
from pyramid.httpexceptions import HTTPNotFound, HTTPFound
from pyramid.security import remember, forget

from .security import USERS, check_password


@view_config(route_name='home', renderer='templates/mytemplate.jinja2')
def my_view(request):…


@view_config(route_name='login', renderer='templates/login.jinja2')
def login(request):
    login = request.params['login']
    password = request.params['password']
    hashed_pw = USERS.get(login)
    if hashed_pw and check_password(password, hashed_pw):
        headers = remember(request, login)
        return HTTPFound(location='/', headers=headers)
    return HTTPNotFound()


@view_config(route_name='logout', renderer=None)
def logout(request):
    headers = forget(request)
    url = request.route_url('home')
    return HTTPFound(location=url, headers=headers)
```

configurando autorização por grupos

```python
from pyramid.config import Configurator
from pyramid.authentication import AuthTktAuthenticationPolicy
from pyramid.authorization import ACLAuthorizationPolicy
from pyramid.security import Allow, Everyone
from .security import groupfinder


class Root(object):
    __acl__ = [
        (Allow, Everyone, 'view'),
        (Allow, 'group:editors', 'edit'),
    ]

    def __init__(self, request):
        pass


def main(global_config, **settings):
    """ This function returns a Pyramid WSGI application.
    """
    config = Configurator(settings=settings, root_factory=Root)
```

```python
from pyramid.view import view_config, forbidden_view_config


@view_config(route_name='protected', permission='edit')
@forbidden_view_config(renderer='templates/login.jinja2')
def protected(request):
    return {}


@view_config(route_name='free', permission='view')
def free(request):
    return {}
```

acl

Access control list

Database

usando pyramid-cookiecutter-alchemy

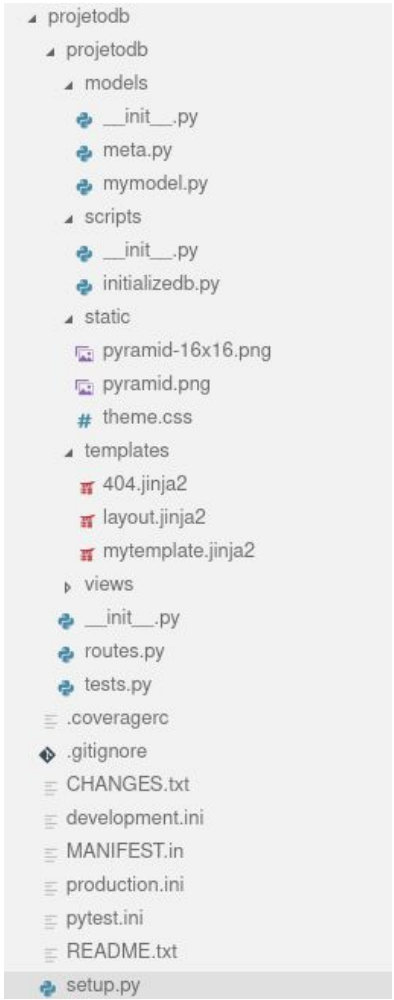
$ cookiecutter gh:Pylons/pyramid-cookiecutter-alchemy --checkout 1.9-branch
project_name [Pyramid Scaffold]: projetodb
repo_name [projetodb]:

- Já cria usando o pyramid_jinja2 por padrão.

usando pyramid-cookiecutter-alchemy

Change directory into your newly created project.
    cd projetodb
Create a Python virtual environment.
    python3 -m venv env
Upgrade packaging tools.
    env/bin/pip install --upgrade pip setuptools
Install the project in editable mode with its testing requirements.
    env/bin/pip install -e ".[testing]"
Configure the database:
    env/bin/initialize_projetodb_db development.ini
Run your project's tests.
    env/bin/pytest
Run your project.
    env/bin/pserve development.ini

usando pyramid-cookiecutter-alchemy



e é isso (:

ah

DeeperSystems contrata

email: jobs@deepersystems.com

Link de vagas: http://www.deepersystems.com/jobs.html

Requisitos:

- Git
- Mongo
- Pyramid

agora foi

são 03:15 da manhã.