Настройки Pyramid.

Настройки Пирамиды можно задать при помощи различных способов:

1. Includes (при помощи сторонних модулей или собственных функций)
2. В виде словаря Python
3. В ini файле

Includes.

1. При помощи собственных функций.  
   Для Pyramid есть возможность создавать собственные функции с настройками. Они вызываются при помощи метода pyramid.config.Configurator.include().

Например, сперва прописываем собственную функцию myfunction, затем вызываем е в коде методом config.include(myfunction), который передает в функцию объект конфигуратора.

def **myfunction(config)**:

config.add\_route(‘hello’, ‘/hello’)

config.add\_view(hello, route\_name=’hello’)

config = Configurator()

config.add\_route(‘world’, ‘/’)

config.add\_view(hello\_world, route\_name=’world’)

**config.include(myfunction)**

app = config.make\_wsgi\_app()

1. При помощи сторонних модулей.  
   Для фреймворка Pyramid существует множество модулей, которые помогают расширить возможности приложения. Они устанавливаются так же, как и любые другие пакеты python.

Например, для установки пакета pyramid\_mailer, который предназначен для отправки электронной почты через Pyramid, теребуется прописать в командной строке следующее:

pip install pyramid\_mailer

А после, требуется подключить данный пакет при помощи того же метода pyramid.config.Configurator.include():

config.include(‘pyramid\_mailer’)

В виде словаря Python.

Так же настройки можно передать в приложение при помощи обычного словаря Python. Это выглядит так:

from pyramid.config import Configurator

if \_\_name\_\_ == ‘\_\_main\_\_’:

settings = {

‘default\_locale\_name': ‘ru’,

‘pyramid.includes’: ‘hypatia pyramid\_services pyramid\_sqlalchemy’,

}

config = Configurator(settings=settings)

print(config.registry.settings)

В ini файле.

Так же настройку как самого приложения, так и веб-сервера можно осуществлять при помощи ini файла. Т.е. мы сообщаем веб-серверу как требуется запустить наше приложение.

Файл ini состоит из нескольких разделов. В разделе [app:main] указываются настройки приложения. Например, подключим пакеты hypatia – система индексирования и поиска, pyramid\_sms - услуги SMS для веб-инфраструктуры и pyramid\_sqlalchemy – использование SQLAlchemy в Pyramid:

[app:main]

use = egg:Project

pyramid.includes = hypatia

pyramid\_sms

pyramid\_sqlalchemy

Данный код можно переписать в самом приложении так:

from pyramid.config import Configurator

def main(global\_config, \*\*settings):

config = Configurator(settings=settings)

config.include(‘hypatia’)

config.include(‘pyramid\_sms’)

config.include(‘pyramid\_ sqlalchemy)

Для настройки веб-сервера, требуется прописать настройки в разделе [server:main], там мы прописываем номер хоста и порта:

[server:main]

use = egg:waitress#main

host = 0.0.0.0

port = 6543

Запускаем приложение при помощи утилиты pserve, которая прочитает ini файл, запустит функцию Project.main, применив к ней все прописанные настройки. Затем получает WSGI приложение от этой функции и запускает его при помощи веб-сервера:

pserve development.ini

Так же есть пакеты, которые помогают управлять конфигурацией проектов:

pyramid\_zcml - Поддержка конфигурации языка разметки Zope для Pyramid.

pyramid\_services - определяет шаблон и вспомогательные методы для доступа к подключаемому слою сервиса из приложений Pyramid.

hupper - перезагрузка процесса для разработчиков, которые могут просматривать файлы изменений и перезапускать процесс.

Как и когда применяются настройки к приложению, определяет конфигурация. Она отличается от обычных настроек тем, что эти настройки могут изменяться по мере выполнения программы. Различают императивный и декларативный способ конфигурации.

Императивный.

При использовании императивного способа конфигурации команды Python будут выполняться по порядку, одна за другой

Декларативный.

При декларативном способе конфигурации, настройки располагаются наиболее близко к тому участку кода, в котором они требуются. А это не всегда требуется в самом начале.

Для того, чтобы применить настройки декларативно, потребуется декодер pyramid.view.view\_config. Однако, использование только его, не приведет к желаемому результату. Чтобы приложение смогло найти и применить эти настройки, нам потребуется выполнить метод pyramid.config.Configurator.scan(). Данный метод проходит по всем файлам, расположенным ниже в данной директории, затем ищет декларативное описание настроек и применяет их к проекту:

from pyramid.response import Response

from pyramid.view import view\_config

from wsgiref.simple\_server import make\_server

from pyramid.config import Configurator

# Декодер

**@view\_config(route\_name='myHelloRoute')**

def hello\_world(request):

return Response('Hello')

if \_\_name\_\_ == '\_\_main\_\_':

config = Configurator()

config.add\_route('myHelloRoute', '/hello/')

# Метод scan

**config.scan()**

# Создаем и запускаем WSGI приложение

app = config.make\_wsgi\_app()

server = make\_server('0.0.0.0', 8080, app)

server.serve\_forever()