## 概念

### WSGI

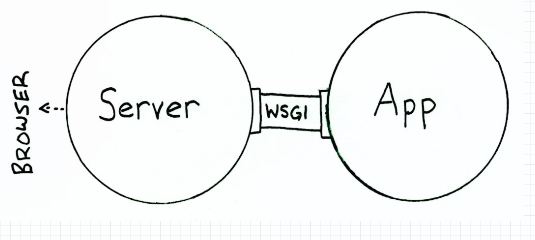
WSGI，全称 Web Server Gateway Interface，或者 Python Web Server Gateway Interface ，是为 Python 语言定义的 Web 服务器和 Web 应用程序或框架之间的一种简单而通用的接口。自从 WSGI 被开发出来以后，许多其它语言中也出现了类似接口。

是一个Gateway，也就是网关。网关的作用就是在协议之间进行转换。

WSGI 是作为 Web 服务器与 Web 应用程序或应用框架之间的一种低级别的[接口](http://www.nowamagic.net/academy/tag/%E6%8E%A5%E5%8F%A3" \t "http://www.nowamagic.net/academy/detail/_blank)，以提升可移植 Web 应用开发的共同点。WSGI 是基于现存的 CGI 标准而设计的。

很多框架都自带了 WSGI server ，比如 Flask，webpy，Django、CherryPy等等。当然性能都不好，自带的 web server 更多的是测试用途，发布时则使用生产环境的 WSGI server或者是联合 nginx 做 uwsgi 。

也就是说，WSGI就像是一座桥梁，一边连着web服务器，另一边连着用户的应用。但是呢，这个桥的功能很弱，有时候还需要别的桥来帮忙才能进行处理。WSGI 的作用如图所示：



**WSGI的作用**

WSGI有两方：“服务器”或“网关”一方，以及“应用程序”或“应用框架”一方。服务方调用应用方，提供环境信息，以及一个回调函数（提供给应用程序用来将消息头传递给服务器方），并接收Web内容作为返回值。

所谓的 WSGI中间件同时实现了API的两方，因此可以在WSGI服务和WSGI应用之间起调解作用：从WSGI服务器的角度来说，中间件扮演应用程序，而从应用程序的角度来说，中间件扮演服务器。“中间件”组件可以执行以下功能：

* 重写环境变量后，根据目标URL，将请求消息路由到不同的应用对象。
* 允许在一个进程中同时运行多个应用程序或应用框架。
* 负载均衡和远程处理，通过在网络上转发请求和响应消息。
* 进行内容后处理，例如应用XSLT样式表。

Pecan是一个轻量级的基于Python的Web框架，

Pecan的目标并不是要成为一个“full stack”的框架，

因此Pecan本身不支持类似Session和Databases.

Pecan框架的目标是实现一个**采用对象分发方式**进行URL路由的轻量级Web框架。它非常专注于自己的目标，它的大部分功能都和URL路由以及请求和响应的处理相关，而不去实现模板、安全以及数据库层，这些东西都可以通过其他的库来实现。

Pecan是一个基于对象路由的框架，即灵活又简单。Pecan主要实现了URL路由功能，支持RESTful API。Pecan没有实现模板、session管理和ORM等功能，但是这些功能可以通过其他的模块来实现。对于OpenStack来说，Pecan是一个很好的选择，因为OpenStack项目中统一使用sqlalchemy来实现ORM，API的实现也不需要模板功能，安全控制则基于Keystone体系。使用Pecan来开发REST服务，代码量很少，代码结构也清晰。Ceilometer项目就是使用了Pecan。

* 便捷的对象分发方式路由
* 完全支持restful风格的controller
* 可扩展的安全框架
* 可扩展的模板语言支持
* 可扩展的JSON支持
* 简单的基于Python的配置

## 2.安装

[root@localhost ~]# pip install pecan

### 2.1创建一个Pecan项目

[root@localhost lzhy]# pecan create test\_project

创建的项目包含如下文件：

$ ls

├── MANIFEST.in

├── config.py

├── public

│   ├── css

│   │   └── style.css

│   └── images

├── setup.cfg

├── setup.py

└── test\_project

   ├── \_\_init\_\_.py

   ├── app.py

   ├── controllers

   │   ├── \_\_init\_\_.py

   │   └── root.py

   ├── model

   │   └── \_\_init\_\_.py

   ├── templates

   │   ├── error.html

   │   ├── index.html

   │   └── layout.html

   └── tests

   ├── \_\_init\_\_.py

   ├── config.py

   ├── test\_functional.py

   └── test\_units.py

* app.py 一般包含了Pecan应用的入口，包含应用初始化代码
* config.py 包含Pecan的应用配置，会被app.py使用
* controllers/ 这个目录会包含所有的控制器，也就是API具体逻辑的地方
* controllers/root.py 这个包含根路径对应的控制器
* controllers/v1/ 这个目录对应v1版本的API的控制器。如果有多个版本的API，你一般能看到v2等目录。

**启动应用：**

$ pecan serve config.py

Starting server in PID 5800

serving on 0.0.0.0:8080, view at <http://127.0.0.1:8080>

**应用程序根目录**

Root Controller是应用程序的入口，你可以认为它相当于根URL（http://localhost:8080/）

class **RootController**(object):

*@expose*(generic=True, template=*'index.html'*)

def **index**(*self*):

return dict()

*@index.when*(method=*'POST'*)

def **index\_post**(*self*, q):

redirect(*'https://pecan.readthedocs.io/en/latest/search.html?q=%s'* % q)

*@expose*(*'error.html'*)

def **error**(*self*, status):

try:

status = int(status)

except ValueError: # pragma: no cover

status = 500

message = getattr(status\_map.get(status), *'explanation'*, *''*)

return dict(status=status, message=message)

**执行测试**

$ python setup.py test -q

running test

running egg\_info

writing requirements to sam.egg-info/requires.txt

writing sam.egg-info/PKG-INFO

writing top-level names to sam.egg-info/top\_level.txt

writing dependency\_links to sam.egg-info/dependency\_links.txt

reading manifest file 'sam.egg-info/SOURCES.txt'

reading manifest template 'MANIFEST.in'

writing manifest file 'sam.egg-info/SOURCES.txt'

running build\_ext

....

----------------------------------------------------------------------

Ran 4 tests in 0.009s

OK

### 2.2 Controllers和Routing

Pecan使用对象分发策略，将一个HTTP请求映射到controller。

from pecan import expose, redirect

from webob.exc import status\_map

class **BooksController**(object):

*@expose*()

def **index**(*self*):

return *"Welcome to book section!"*

*@expose*()

def **bestsellers**(*self*):

return *"We have 5 books in the top 10."*

class **CatalogController**(object):

*@expose*()

def **index**(*self*):

return *"Welcome to the catalog."*

books = BooksController()

class **RootController**(object)：

*@expose*()

def **index**(*self*):

return *"Welcome to store.example.com"*

*@expose*()

def **hours**(*self*):

return *"Open 24/7 on the web."*

catalog = CatalogController()

搜索路径：

└── /

   ├── /hours

   └── /catalog

   └── /catalog/books

   └── /catalog/books/bestsellers

路由到的方法：

└── RootController.index

   ├── RootController.hours

   └── CatalogController.index

   └── BooksController.index

   └── BooksController.bestsellers