Pyramid™

Tutorial Outline

- The goal is to write a simple but complete Pyramid application and leave you with a base to start your own projects.
- We'll have four stages, where we'll first learn a few Pyramid concepts, then start looking at the code for that stage.
- Throughout the tutorial, I will answer any subject-related questions that may come up.

Welcome to the Pyramid Tutorial

Thanks for being here. Today we'll:

- Quickly go over some administrative stuff.
- Briefly introduce the tutorial.
- Learn about Pyramid.
- Write a simple application.
- Improve our application in stages.
- Get answers to some Pyramid questions.
- Have some fun.

Who is the Teacher?

- My name is Carlos de la Guardia.
- I live in Mexico and do Python web development for multiple clients.
- I've been doing web stuff since 1994.
- I'm a Pyramid contributor since it was (un)known as repoze.bfg (that means 2008 or so).
- I love Pyramid so much that I was willing to leave the side of my lovely three children and beautiful wife for a couple of days just to teach you about it.

Administrative notes

- Were you able to install Pyramid?
- If not, grab a CD, install and pass it on.
- Anyway, all the code is at:

https://github.com/cguardia/Pyramid-Tutorial

- Pyramid t-shirt give away.
- Tutorial structure.

What you bring to the table

- Familiarity with web development. General knowledge of how a web application works, some exposure to HTML, CSS and Javascript.
- At least some familiarity with Python. We'll not do anything complex here, but you should be able to look at some code and have some idea what's going on.
- Real interest in writing a real web application sometime, preferably right after the tutorial.

The Pylons Project

- The aim of the project is to develop a related set of web technologies
- It's an umbrella name, similar to how the Apache name is used today
- Pylons 1 is a stable and maintained web framework, but no further enhancements to it will be done.
- Pyramid is the current and future embodiment of Pylons-Style web development
- There will be other projects under this umbrella.

Is Pyramid for me?

- Like all web frameworks, Pyramid is better suited for some uses cases than others:
  - If you want more flexibility than 'rails'.
  - If you like to have control of all the parts of your application.
  - If you need a truly pluggable framework.
  - if you enjoy 'doing it yourself' when you need something special.
  - Certainly it's a great option if you want to use Python.

What is Pyramid?

A Python web application development framework that features:

- Simplicity
- Minimalism
- Documentation
- Openness
- Speed
- Reliability

Pyramid is primarily inspired by Django, Zope and Pylons.

Stage 0 – Installing Pyramid

- Pyramid runs on all popular UNIX-like systems, like Linux, Mac OSX and FreeBSD. It also can be used on Windows.
- The only requirement is a Python interpreter, version 2.6 or higher.
- Pyramid best practice suggests the use of virtualenv for development.

Where is Pyramid and where is it going?

- Version 1.3 is in beta just now.
- Now with **Python 3.2 compatibility**
- Other new features include configuration introspection and many convenience APIs for handling things like asset specifications, relative URLs, forbidden and not found views, etc.
- All new features are completely documented and we keep adding new documentation in the form of how-to's and tutorials.
- Many people have begun contributing Pyramid add-ons and applications.

Simplest Pyramid App

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

def hello_world(request):
   return Response('Hello %(name)s!' % request.matchdict)

if __name__ == '__main__':
   config = Configurator()
   config.add_route('hello', '/hello/{name}')
   config.add_view(hello_world, route_name='hello')
   app = config.make_wsgi_app()
   server = make_server('0.0.0.0', 8080, app)
   server.serve_forever()
```

Pyramid starting points

Pyramid includes several 'scaffolds' that help you get started quickly:

- starter
- alchemy
- zodb (Python 2.x only)

In addition there are some third party scaffolds available. In particular Akhet might be useful for Pylons legacy users.

The alchemy scaffold

- Uses SQLAlchemy
- Uses a transaction manager
- Uses routes (URL dispatch)
- Uses ZPT
- Let's take a look at the code.

More about the scaffolds

- They are independent, but share a common structure.
- It's a good idea to try to keep that structure when developing using a scaffold.
- They include development and production configurations.
- They all use Waitress as a WSGI server, which permits Python 3 compatibility. It is intended for development only.

What Pyramid does

- Pyramid has a main method which is run by the calling WSGI infrastructure. In this case Waitress.
- This method creates a configuration object to hold Pyramid's routing and view configuration.
- Then a WSGI application is created, which is a router that uses the configuration to decide how to handle request.
- Pyramid also offers sessions, authorization, authentication and templating.

Typical Pyramid project

- A typical project might use routes and a relational database. We recommend but do not enforce using SQLAlchemy.
- Many of us started with Zope years ago and Pyramid also has features that are geared towards that kind of development.
- There are two templating languages. Most documentation uses ZPT, but you can use Mako out of the box as well. There are also third party bindings.

URL Dispatch

- A route consists basically of a name and a pattern, with some possible additional predicates.
- Our defined routes are checked one by one against every web request.
- If there's a match, Pyramid will find a view using a separate view lookup mechanism.
- Order is important when declaring routes.

Views

- A view is a callable (function, class or instance with a __call__ method) that accepts a request and returns a response.
- Views are registered at application start up.
- A particular view can be associated with a route and is called when the route matches and passed any parameters that it has.
- Views can be registered also for exceptions, allowing us to customize responses.

Templates

- Out of the box, Pyramid supports Chameleon and Mako, but there are third party bindings for other templating engines, like Jinja2.
- Templates can be used directly from a view by specifying their path or an asset specification.
- Templates can also be configured as renderers outside the view code itself (good for unit testing).
- Auto template reloading can be enabled in the package configuration.

View configuration parameters

- Name
- Context
- route_name
- Request type
- Request method
- Request parameter
- Container
- Xhr
- Accept
- Header
- Path info
- custom

Static assets

- An asset is a file which has an specification for locating it from application configuration and code.
- A static view allows an application to serve assets.
- The name of the static view represents a URL path to get the assets. It can contain other subdirectories.
- Static assets can be overridden from other packages.

Renderers

- Pyramid offers renderers that allow the user to automatically return one type of response without having to construct it.
- There are renderers for:
  - String
  - Json
  - Jsonp
  - Templates
- To use them just specify one when configuring the view and in the view return a dict with the parameters to pass to it.
- It's easy to create custom renderers if needed.
- You can still alter the response by using request.response to set headers.

Let's look at more code

- Our application is a twitter clone.
- At this point it has a single route.
- It has two views which use the same route.
- The view predicates are the difference.
- Note the use of renderers.
- We also have a static view for assets.

Security

- Pyramid provides a declarative authorization system that allows us to set permissions to views.
- It also provides a simple authentication policy to handle custom logins.
- The next stage of our sample application adds such a system to our Birdie app.

Looking at the sample app

- If we take a look at the code, the mode of configuration is now different.
- when all the views are defined one after the other this is known as imperative configuration.
- The other system is known as declarative configuration.
- Note that the decorators that Pyramid uses for views do not have side effects at import time, so it's necessary to use the scan function.

Declarative configuration

- The configuration for the views up to stage 2 of the sample app is done using a style of configuration that requires all views to be defined one after the other.
- Pyramid also offers another configuration mode which allows us to have the view defined right beside the code it uses.
- For larger systems, this might be a better strategy.

There's more

- But there's no time.
- Some important topics we didn't cover:
  - Traversal
  - Events
  - Sessions
  - Scripting
  - Hooks
  - Deployment

keeping configuration under control

- Get a list of all defined routes: bin/paster proutes development.ini
- Get a list of probable views for a given URL: bin/paster pviews development.ini /join
- Get a shell with the Pyramid configuration: bin/paster pshell development.ini

Sessions

- Pyramid allows configuration of sessions via a session factory.
- Pyramid includes a basic session factory, not secure, but digitally signed.
- You can create your own or use third party session factories.
- The session factory is added via configuration.
- A session factory provides a session object to the request.
- The session provides facilities for invalidation, CSRF protection and flash messages.

Events

- Pyramid allows us to subscribe to important events in the application lifecycle.
- Most importantly, we can be notified when the request is created and when the response is prepared.
- We can have our callbacks execute before these events.

Traversal

- Traversal is a resource finding mechanism that is very useful when you have 'contentish' objects.
- It's conceptually similar to navigating a filesystem tree.
- It's also very useful when you need row level security (different permissions per content object).
- Instead of looking views using routes, we traverse to a content object and then pick one of its views.

Really ran out of time

- Everything is documented, so you can go and check the docs today.
- We also have one of the most friendly live support channels on IRC. Please visit.
- More information at:

  http://www.pylonsproject.org

Thanks a lot

- To win a t-shirt please fill the survey at:

Http://pyramid.delaguardia.mx

- I'm cguardia@yahoo.com, cguardia on twitter and IRC.
- Find us in IRC, channel #pyramid