WSGI

Web Server Gateway Interface

PEP 0333
PEP 3333

What is a Web app?

What is a Web app?

App = f(data) -> data'

What is a Web app?

App = f(data) -> data'

WebApp = f(Request) -> Response

What is a Web app?

App = f(data) -> data'

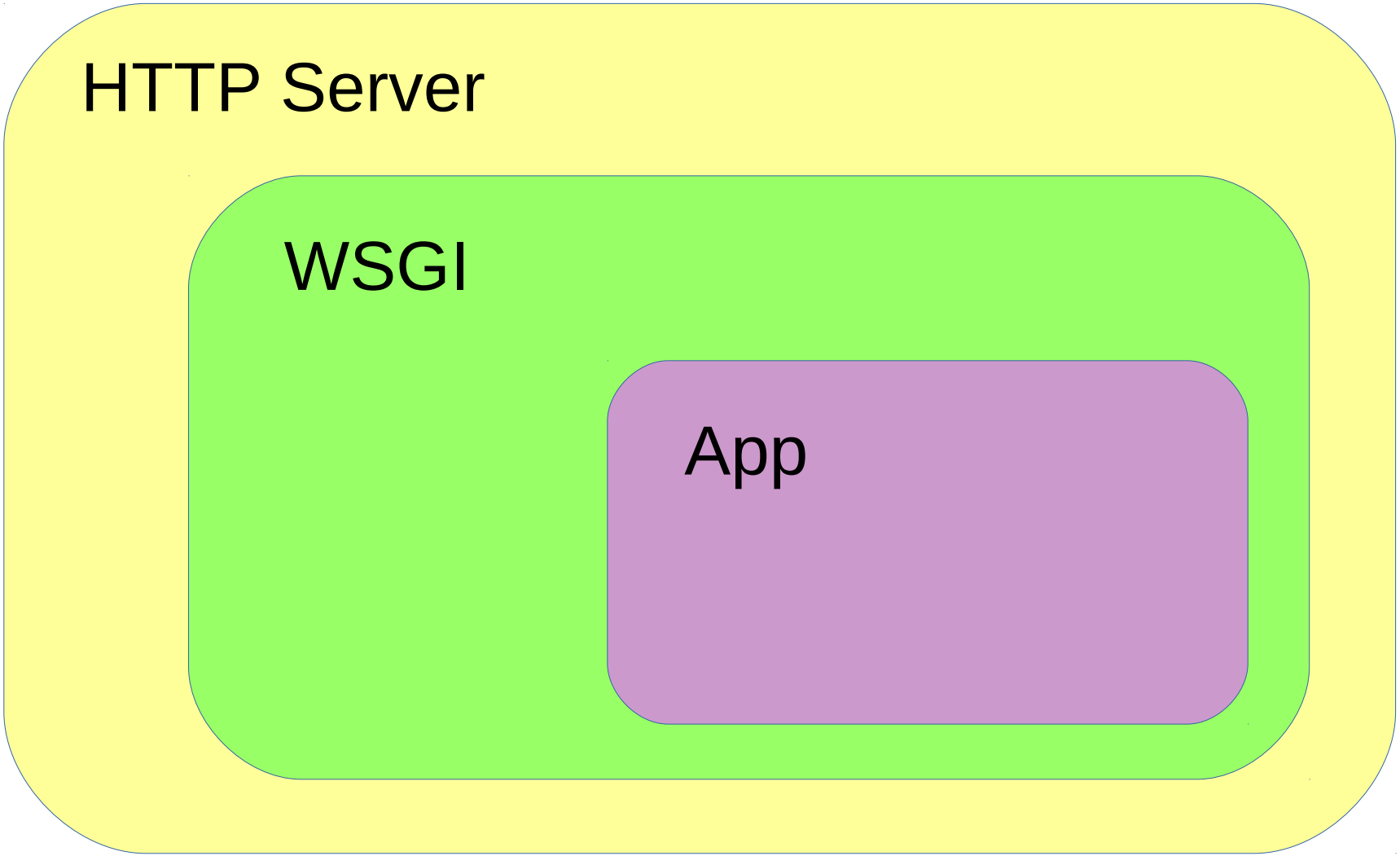
WebApp = f(Request) -> Response

WSGI = f(WebApp) -> WebApp'

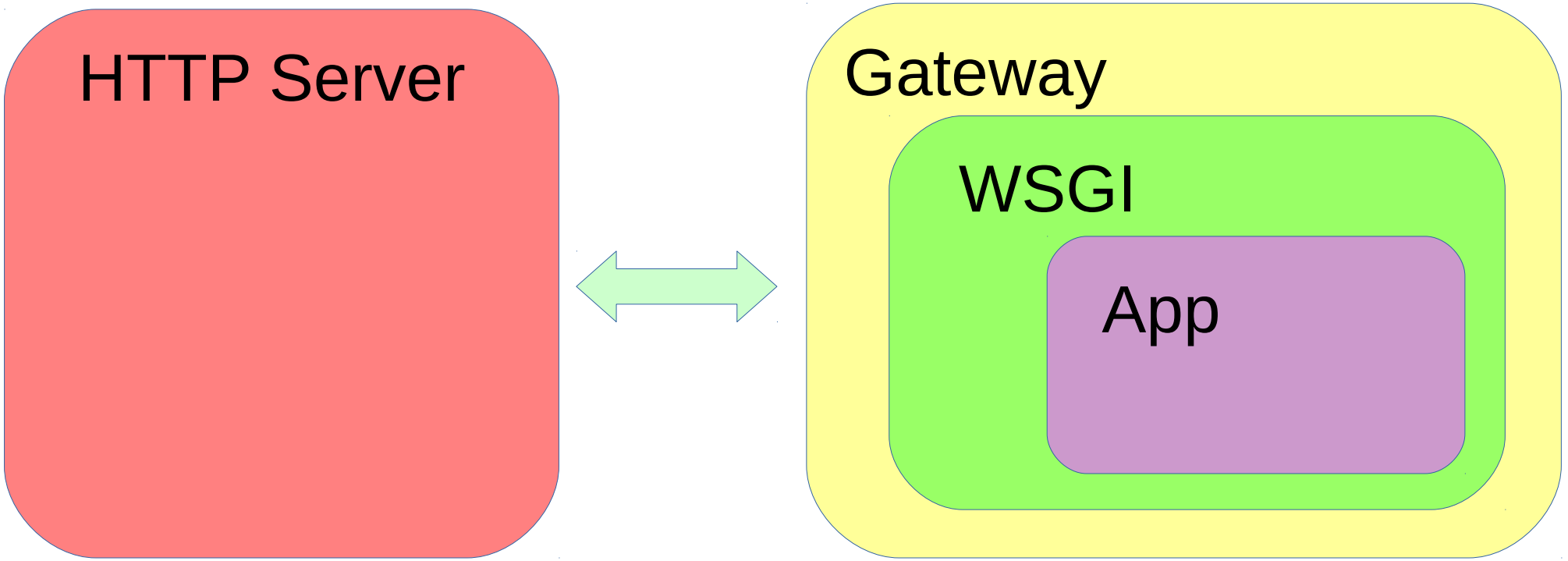
Why WSGI?

- Multitude of standards
- Interface easier than framework
- Allows boxing(middleware)

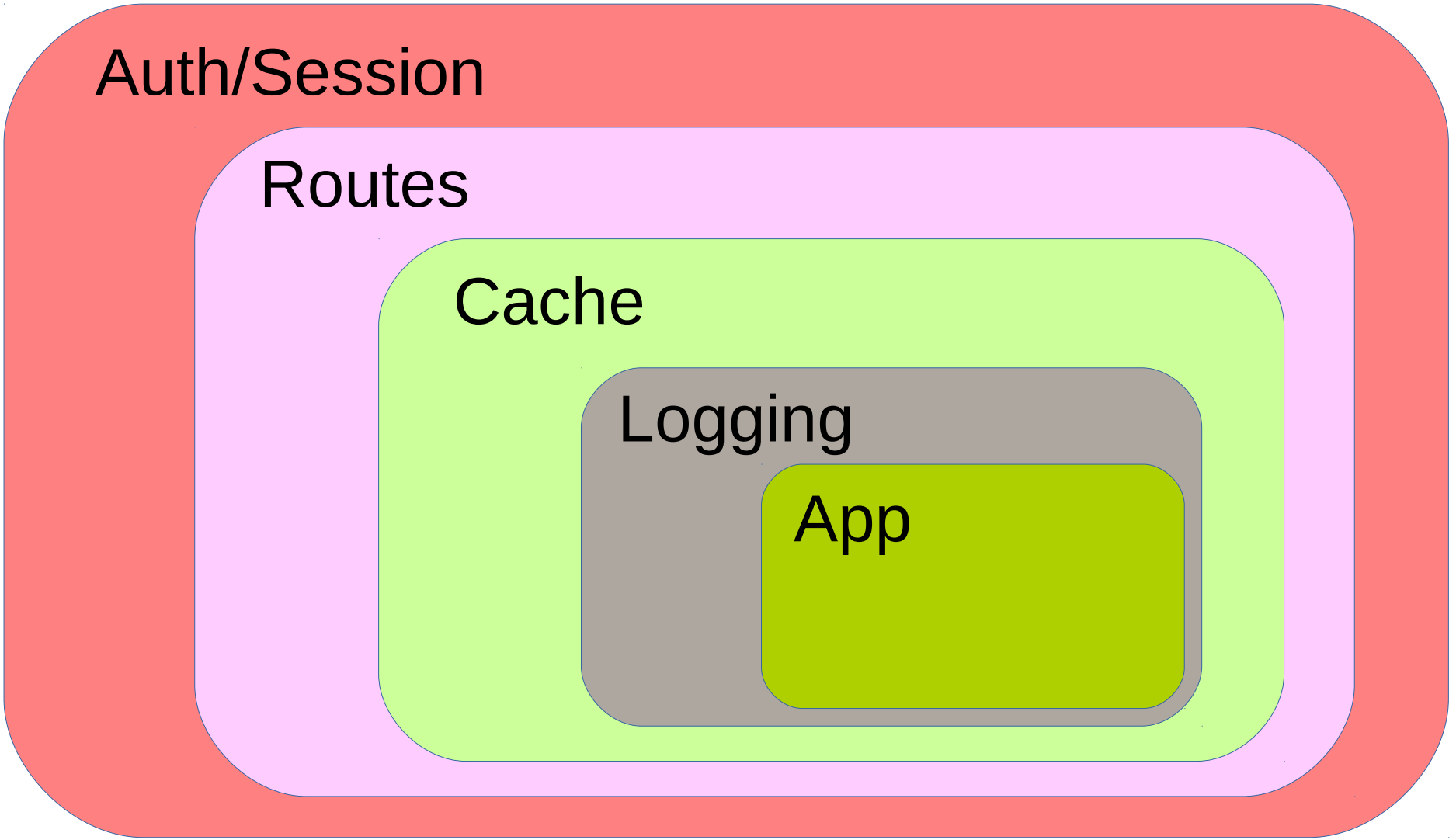
Deployment.0



Deployment.1



WSGI Middleware



Example WSGI app

```
from wsgiref.simple_server import make_server

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

make_server('', 8000, simple_app).serve_forever()
```

Example WSGI middleware

```
def reverse_middleware(app):
    def wrapped_app(environ, start_response):
        response = app(environ, start_response)
        return [s[::-1] for s in response]
    return wrapped_app
```

environ(CGI)

REQUEST_METHOD

SCRIPT_NAME

PATH_INFO

QUERY_STRING

CONTENT_TYPE

CONTENT_LENGTH

SERVER_NAME , SERVER_PORT

SERVER_PROTOCOL

HTTP_Variables

environ(WSGI)

wsgi.version

wsgi.url_scheme

wsgi.input

wsgi.errors

wsgi.multithread

wsgi.multiprocess

wsgi.run_once

start_response

f(status, headers, exc_info) -> iterable

status = HTTP Status (200 OK, 500 Internal Error)

headers = [("Content-Type", "text/plain")]

exc_info = sys.exc_info()

Why start with WSGI?

- Frameworks solve author's problems
- Unlimited library scope/interop
- Lightweight and transparent

Essential framework parts

- Request wrapper
- Response wrapper
- Controller wrapper
- Routing

Request Wrapper

```
from urlparse import parse_qs
from collections import namedtuple


def Request(environ):
    return namedtuple('Request', ['path', 'GET', ])(
        path=environ['PATH_INFO'],
        GET=parse_qs(environ['QUERY_STRING']),
    )
```

Response Wrapper

```
def Response(status, body):
    def wrapper(environ, start_response):
        http_status = {
            200: '200 OK',
            404: '404 Not Found'
        }[status]
        start_response(http_status, [('Content-type', 'text/plain')])
        return [body]
    return wrapper
```

Controller wrapper

```
def controller(func):
    def wrapper(environ, start_response):
        response = func(Request(environ))
        return response(environ, start_response)
    return wrapper
```

Routing

```
def Router(route_map):
    def router(environ, start_response):
        path = environ['PATH_INFO']
        default = Response(404, 'Page not found')
        action = route_map.get(path, default)
        return action(environ, start_response)
    return router
```

Come together

views.py

```
from util import controller, Response, Router

@controller
def index(request):
    return Response(200, "It works!")

@controller
def hello(request):
    name = request.GET.get('name', ["Anonymous"])[0]
    return Response(200, "Hello %s" % name)

router = Router({
    '/index': index,
    '/hello': hello,
})
```

app.py

```
from wsgiref.simple_server import make_server
from views import router
make_server('', 8000, router).serve_forever(
```

Gateways

- wsgiref
- werkzeug.serving
- uWSGI
- Green Unicorn
- CherryPy
- Tornado
- Twisted
- etc. etc. etc.

Utilities

- WebOb
- Routes
- Werkzeug
- Paste
- Jinja2
- Mako

Google -> I'm feeling lucky

PEP 0333

WSGI talks

webob framework

Also examples from talk

https://github.com/vharitonsky/firefly

Thank you, Any Questions?