Advanced Flask Patterns

PyCon Russia 2013

— a presentation by Armin Ronacher

@mitsuhiko

FIRETEAM™

-1

Who am I?

That's me

- Armin Ronacher
- @mitsuhiko
- Creator of Flask/Werkzeug/Jinja2

Focus & Caveats

Interrupt me

- Assumes some sense of Flask knowledge
- If too fast, interrupt me
- If not detailed enough, let me know

State Management

Flask States

- Setup State
- Application Context Bound
- Request Context Bound

Setup State

```
>>> from flask import g
>>> g.foo = 42
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
RuntimeError: working outside of application context
```

Application Bound

```
>>> ctx = app.app_context()
>>> ctx.push()
>>> g.foo = 42
>>> g.foo
42

>>> from flask import request
>>> request.args
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
RuntimeError: working outside of request context
```

Request Bound

```
>>> ctx = app.test_request_context()
>>> ctx.push()
>>> request.url
'http://localhost/'
>>> g.foo = 42
>>> g.foo
42
```

Lifetimes

- `flask.current_app` ⇝ application context
- `flask.g` ⇝ application context (*as of 0.10*)
- `flask.request` ⇝ request context
- `flask.session` ⇝ request context

Quick Overview

- Application contexts are fast to create/destroy
- Pushing request context pushes new application context
- Flask 0.10 binds `g` to the application context
- Bind resources to the application context

2 Resource Management

Basic Guide

- Create/Destroy Application Context == Task
- Bind resources task wise
- Resources: claimed database connections, caches

Teardown Illustrated

```
>>> from flask import Flask
>>> app = Flask(__name__)
>>> @app.teardown_appcontext
... def called_on_teardown(error=None):
...   print 'Tearing down, error:', error
...

>>> ctx = app.app_context()
>>> ctx.push()
>>>
>>> ctx.pop()
Tearing down, error: None

>>> with app.app_context():
...  1/0
...
Tearing down, error: integer division or modulo by zero
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
ZeroDivisionError: integer division or modulo by zero
```

Resource Management

```
def get_database_connection():
    con = getattr(g, 'database_connection', None)
    if con is None:
        g.con = con = connection_pool.get_connection()
    return con


@app.teardown_appcontext
def return_database_connection(error=None):
    con = getattr(g, 'database_connection', None)
    if con is not None:
        connection_pool.release_connection(con)
```

Responsive Resources

```
@app.teardown_appcontext
def return_database_connection(error=None):
    con = getattr(g, 'database_connection', None)
    if con is None:
        return
    if error is None:
        con.commit()
    else:
        con.rollback()
    connection_pool.release_connection(con)
```

Per-Task Callbacks

```
def after_commit(f):
    callbacks = getattr(g, 'on_commit_callbacks', None)
    if callbacks is None:
        g.on_commit_callbacks = callbacks = []
    callbacks.append(f)
    return f
```

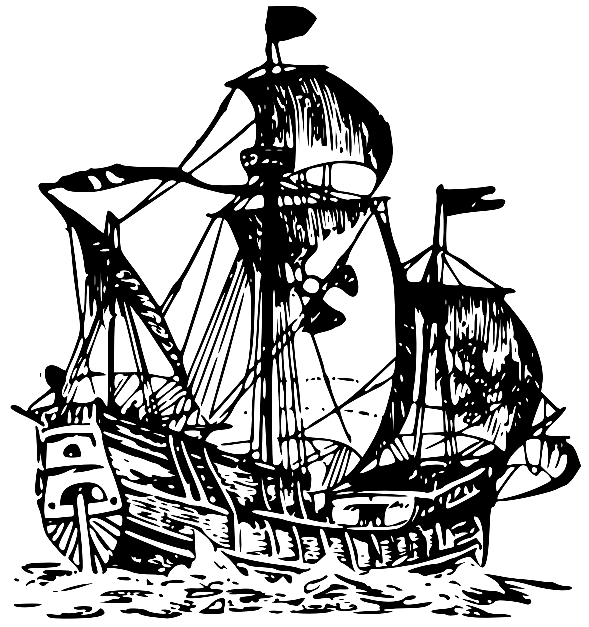
Per-Task Callbacks

```python
@app.teardown_appcontext
def return_database_connection(error=None):
    con = getattr(g, 'database_connection', None)
    if con is None:
        return
    if error is None:
        con.commit()
        callbacks = getattr(g, 'on_commit_callbacks', ())
        for callback in callbacks:
            callback()
    else:
        con.rollback()
    connection_pool.release_connection(con)
```

Per-Task Callbacks Example

```
def purchase_product(product, user):
    user.purchased_products.append(product)
    @after_commit
    def send_success_mail():
        body = render_template('mails/product_purchased.txt',
            user=user,
            product=product
        )
        send_mail(user.email_address, 'Product Purchased', body)
```

3 Response Creation



Response Object Passing

- One request object: read only
- Potentially many response objects, passed down a stack
- … can be implicitly created
- … can be replaced by other response objects
- there is no `flask.response`!

Implicit Response Creation

```
@app.route('/')
def index():
    return render_template('index.html')
```

Explicit Creation

```
from flask import make_response


@app.route('/')
def index():
    body = render_template('index.html')
    response = make_response(body)
    response.headers['X-Powered-By'] = 'Not-PHP/1.0'
    return response
```

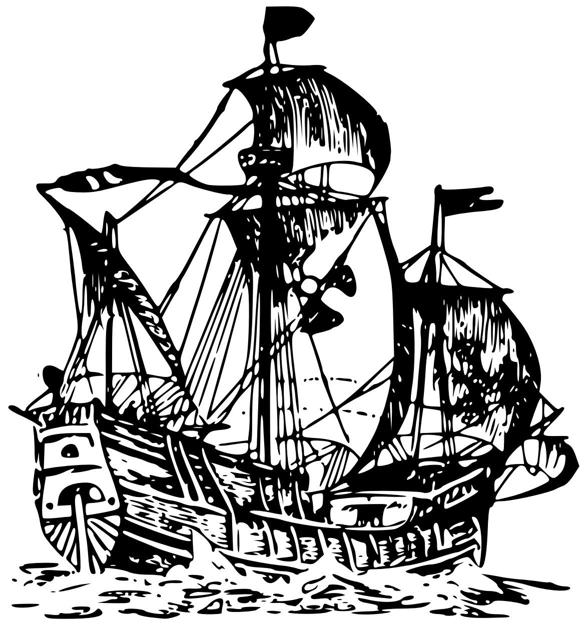
Customized Creation

```
from flask import Flask, jsonify


class CustomFlask(Flask):

    def make_response(self, rv):
        if hasattr(rv, 'to_json'):
            return jsonify(rv.to_json())
        return Flask.make_response(self, rv)
```

Customized Creation Example

```
class User(object):

    def __init__(self, id, username):
        self.id = id
        self.username = username

    def to_json(self):
        return {
            'id': self.id,
            'username': self.username
        }


app = CustomFlask(__name__)

@app.route('/')
def index():
    return User(42, 'john')
```

4 Server Sent Events



Basic Overview

- Open Socket
- Sends "`data: <data>\r\n\r\n`" packets
- Good idea for gevent/eventlet, bad idea for kernel level concurrency

Subscribing

```python
from redis import Redis
from flask import Response, stream_with_context


redis = Redis()


@app.route('/streams/interesting')
def stream():
    def generate():
        pubsub = redis.pubsub()
        pubsub.subscribe('interesting-channel')
        for event in pubsub.listen():
            if event['type'] == 'message':
                yield 'data: %s\r\n\r\n' % event['data']
    return Response(stream_with_context(generate()),
                    direct_passthrough=True,
                    mimetype='text/event-stream')
```

Publishing

```
from flask import json, redirect, url_for


@app.route('/create-something', methods=['POST'])
def create_something():
    create_that_thing()
    redis.publish('interesting-channel', json.dumps({
        'event': 'created',
        'kind': 'something'
    }))
    return redirect(url_for('index'))
```

Don't be Afraid of Proxying

- gunicorn/uwsgi blocking for main app
- gunicorn gevent for SSE
- nginx for unification

5 Worker Separation

supervisor config

```
[program:worker-blocking]
command=gunicorn -w 4 yourapplication:app -b 0.0.0.0:8000

[program:worker-nonblocking]
command=gunicorn -k gevent -w 4 yourapplication:app -b 0.0.0.0:8001
```

nginx config

```
server {
    listen 80;
    server_name example.com;

    location /streams {
        proxy_set_header Host $http_host;
        proxy_pass http://localhost:8001/streams;
    }

    location / {
        proxy_set_header Host $http_host;
        proxy_pass http://localhost:8000/;
    }
}
```

6 Signing Stuff

Basic Overview

- Use `itsdangerous` for signing information that roundtrips
- Saves you from storing information in a database
- Especially useful for small pieces of information that need to stay around for long (any form of token etc.)

User Activation Example

```
from flask import abort
import itsdangerous


serializer = itsdangerous .URLSafeSerializer(secret_key=app.config['SECRET_KEY'])
ACTIVATION_SALT = '\x7f\xfb\xc2(;\r\xa8O\x16{'


def get_activation_link(user):
    return url_for('activate', code=serializer.dumps(user.user_id, salt=ACTIVATION_SALT))


@app.route('/activate/<code>')
def activate(code):
    try:
        user_id = serializer.loads(code, salt=ACTIVATION_SALT)
    except itsdangerous.BadSignature:
        abort(404)
    activate_the_user_with_id(user_id)
```

7 Customization

Simple Cache Busting

```
from hashlib import md5
import pkg_resources


ASSET_REVISION = md5(str(pkg_resources.get_distribution(
    'Package-Name').version)).hexdigest())[:14]


@app.url_defaults
def static_cache_buster(endpoint, values):
    if endpoint == 'static':
        values['_v'] = ASSET_REVISION
```

Disable Parsing

```
from flask import Flask, Request


class SimpleRequest(Request):
    want_form_data_parsed = False
    data = None


app = Flask(__name__)
app.request_class = SimpleRequest
```

8

Secure Redirects

Redirect Back

```
from urlparse import urlparse, urljoin

def is_safe_url(target):
    ref_url = urlparse(request.host_url)
    test_url = urlparse(urljoin(request.host_url, target))
    return test_url.scheme in ('http', 'https') and \
           ref_url.netloc == test_url.netloc

def is_different_url(url):
    this_parts = urlparse(request.url)
    other_parts = urlparse(url)
    return this_parts[:4] != other_parts[:4] and \
        url_decode(this_parts.query) != url_decode(other_parts.query)

def redirect_back(fallback):
    next = request.args.get('next') or request.referrer
    if next and is_safe_url(next) and is_different_url(next):
        return redirect(next)
    return redirect(fallback)
```

Q&A