Flask for Fun and Profit

Armin @mitsuhiko Ronacher

Hello I'm Armin,
Hailing from wonderful Vienna Austria

I do Open Source Things :)

Flask

web development,
one drop at a time


SENTRY

Where does Flask come from?

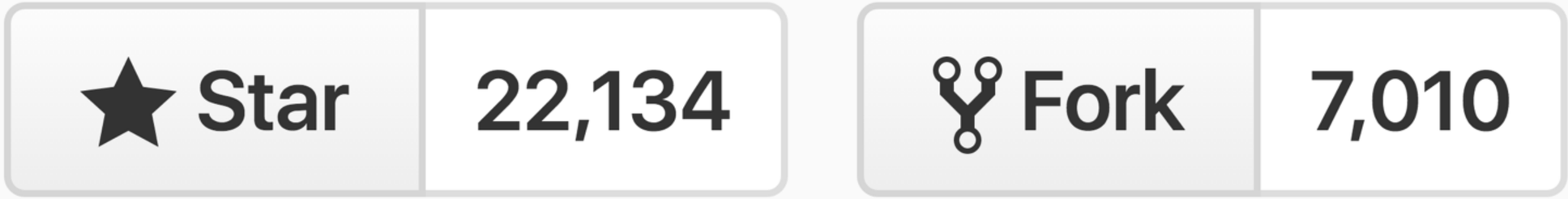
Iteration ...

- Before *Flask* there was *Werkzeug*
- Before *Werkzeug* there was *WSGITools*
- Before *WSGITools* there was *Colubrid*
- Before *Colubrid* there was a lot of *PHP* and *"Pocoo"*

Why?

- I wanted to build software to distribute
- Originally I wanted to write a version of phpBB
- The inspiration was utilities to build “trac” and never “django”
- Put programmer into control of configuration, do not impose configuration on the framework users

Why do people like it?


Star
22,134
Fork
7,010

The API seems to resonate with people

Small overall footprint

What's it good at

small HTML heavy CRUD sites

JSON APIs :)

Iteration Speed

Testing :)

What's it bad at

High Performance Async IO

My Favorite Flask App Structure

create_app

```
from flask import Flask

def create_app(config=None):
    app = Flask(__name__)
    app.config.update(config or {})
    register_blueprints(app)
    register_other_things(app)
    return app
```

register_blueprints

```
from werkzeug.utils import find_modules, import_string

def register_blueprints(app):
    for name in find_modules('myapp.blueprints'):
        mod = import_string(name)
        if hasattr(mod, 'blueprint'):
            app.register_blueprint(mod.blueprint)
```

Optional Contained App

```
from flask import Flask

class MyThing(object):

    def __init__(self, config):
        self.flask_app = create_app(config)
        self.flask_app.my_thing = self

    def __call__(self, environ, start_response):
        return self.flask_app(environ, start_response)
```

Development Runner

```
# devapp.py
from myapp import create_app
app = create_app({
    'DATABASE_URI': 'sqlite:////tmp/my-appdb.db',
})
```

Development Runner

```
$ export FLASK_APP=`pwd`/devapp.py
$ export FLASK_DEBUG=1
$ flask run
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 236-726-332
```

The Improved Runner

~~`app.run(debug=True)`~~

```
$ export FLASK_APP=/path/to/file.py
$ export FLASK_DEBUG=True
$ flask run
```

Context Locals

Basics

```
from flask import Flask, current_app

app = Flask(__name__)

with app.app_context():
    assert current_app.name == app.name
```

Other Context Objects

- request context bound:
  - flask.request
  - flask.session
- app context bound:
  - flask.g
  - flask.current_app

app context tears down end of request!

Cron Stuff

```
from myapp import create_app
from werkzeug.utils import import_string

def run_cron(import_name, config):
    func = import_string(import_name)
    app = create_app(config=config)
    with app.app_context():
        func()
```

Resource Management

```
import sqlite3
from flask import g

def get_db():
    db = getattr(g, '_database_con', None)
    if db is None:
        db = g._database_con = sqlite3.connect(DATABASE)
    return db

@app.teardown_appcontext
def close_connection(exception):
    db = getattr(g, '_database_con', None)
    if db is not None:
        db.close()
```

User Management

```
from flask import g

def get_user():
    user = getattr(g, 'user', None)
    if user is None:
        user = load_user_from_request()
        g.user = user
    return user
```

JSON APIs

Result Wrapper

```
from flask import json, Response

class ApiResult(object):

    def __init__(self, value, status=200):
        self.value = value
        self.status = status

    def to_response(self):
        return Response(json.dumps(self.value),
                        status=self.status,
                        mimetype='application/json')
```

Response Converter

```
from flask import Flask

class ApiFlask(Flask):

    def make_response(self, rv):
        if isinstance(rv, ApiResult):
            return rv.to_response()
        return Flask.make_response(self, rv)
```

API Errors

```
from flask import json, Response

class ApiException(object):

    def __init__(self, message, status=400):
        self.message = message
        self.status = status

    def to_result(self):
        return ApiResult({'message': self.message},
                         status=self.status)
```

Error Handler

```
def register_error_handlers(app):
    app.register_error_handler(
        ApiException, lambda err: err.to_result())
```

Demo Api

```
from flask import Blueprint

bp = Blueprint('demo', __name__)

@bp.route('/add')
def add_numbers():
    a = request.args('a', type=int)
    b = request.args('b', type=int)
    if a is None or b is None:
        raise ApiException('Numbers must be integers')
    return ApiResult({'sum': a + b})
```

Validation / Serialization

Finding the Balance

- Most validation systems in Python are in a weird spot
- Either very powerful but opinionated and fun to use
- Or powerful and a pain to use
- Or weak and sooner or later shape your API a ton

Finding the Right Library

- There are so many
- jsonschema anyone?
- One that works for me: voluptuous

voluptuous 101

```
from flask import request
from voluptuous import Invalid

def dataschema(schema):
    def decorator(f):
        def new_func(*args, **kwargs):
            try:
                kwargs.update(schema(request.get_json()))
            except Invalid as e:
                raise ApiException('Invalid data: %s (path "%s")' %
                                   (e.msg, '.'.join(e.path)))
            return f(*args, **kwargs)
        return update_wrapper(new_func, f)
    return decorator
```

voluptuousified view

```
from voluptuous import Schema, REMOVE_EXTRA

@app.route('/add', methods=['POST'])
@dataschema(Schema({
    'a': int,
    'b': int,
}, extra=REMOVE_EXTRA))
def add_numbers(a, b):
    return ApiResult({'sum': a + b})
```

extensions vs hand rolled

Extensions

- They are nice for a lot of things (like database APIs)
- However they are very opinionated about data in/out
- Often these things fight with how I want APIs to work
- In particular serialization/deserialization/errors

Control the API: Pagination

```
from werkzeug.urls import url_join

class ApiResult(object):
    def __init__(self, …, next_page=None):
        …
        self.next_page = next_page

    def to_response(self):
        rv = Response(…)
        if self.next_page is not None:
            rv.headers['Link'] = '<%s>; rel="next"' % \
                url_join(request.url, self.next_page)
        return rv
```

Security!

Context, context, context

- Write good abstractions for security related APIs
- Make code aware of the context it's executed at

context for improved security

```
from myapp import db
from myapp.security import get_available_organizations

class Project(db.Model):
    …

    @property
    def query(self):
        org_query = get_available_organizations()
        return db.Query(self).filter(
            Project.organization.in_(org_query))
```

JSON Escaping

```
>>> from flask.json import htmlsafe_dumps
>>> print htmlsafe_dumps("<em>var x = 'foo';</em>")
"\u003cem\u003evar x = \u0027foo\u0027;\u003c/em\u003e"
```

Testing!

best paired with py.test

Basic Example

```
import pytest

@pytest.fixture(scope='module')
def app(request):
    from yourapp import create_app
    app = create_app(…)
    ctx = app.app_context()
    ctx.push()
    request.addfinalizer(ctx.pop)
    return app
```

Example Test

```
def test_app_name(app):
    assert app.name == 'mypackage'
```

More Fixtures

```
def test_client(request, app):
    client = app.test_client()
    client.__enter__()
    request.addfinalizer(
        lambda: client.__exit__(None, None, None))
    return client
```

Example View Test

```
def test_welcome_view(test_client):
    rv = test_client.get('/welcome')
    assert 'set-cookie' not in rv.headers
    assert b'Welcome' in rv.data
    assert rv.status_code == 200
```

Websockets and Stuff

no amazing answer

What I do:

- redis broker with pub/sub
- custom server that sends those events via SSE to the browser
- push events from the Flask backend to this redis broker
- use signing (`itsdangerous`) for authentication of the channel


Q&A