Flasky Goodness

(Or Why Django Sucks?)

Kenneth Reitz

Hi.

@kennethreitz


heroku

Open Source

Python-Guide.org

The Hitchhiker's Guide to Python

- Documented best practices.
- Guidebook for newcomers.
- Reference for seasoned veterans.
- Don't panic & always carry a towel.

Requests

HTTP for Humans

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf-8'
>>> r.encoding
'utf-8'
>>> r.text
u'{"type":"User"...'
>>> r.json
{u'private_gists': 419, u'total_private_repos': 77, ...}
```

Httpbin.org

```
$ curl http://httpbin.org/get?test=1
{
  "url": "http://httpbin.org/get",
  "headers": {
    "Content-Length": "",
    "Connection": "keep-alive",
    "Accept": "*/*",
    "User-Agent": "curl/7.21.4 ...",
    "Host": "httpbin.org",
    "Content-Type": ""
  },
  "args": {
    "test": "1"
  },
  "origin": "67.163.102.42"
}
```

Et Cetera

- Legit: Git Workflow for Humans
- Envoy: Subprocess for Humans
- Tablib: Tabular Data for Humans
- Clint: CLI App Toolkit
- Autoenv: Magic Shell Environments
- OSX-GCC-Installer: Provokes Lawyers

275+ More

Open Source

All The Things!

Build for Open Source

- Components become concise & decoupled.
- Concerns separate themselves.
- Best practices emerge (e.g. no creds in code).
- Documentation and tests become crucial.
- Code can be released at any time.

Abstraction

Let’s build something.

We’ll use Django, of course.

Django Benefits

- Makes modular decisions for you.
- Makes security decisions for you.
- Excellent documentation available.
- Installable third-party Django apps.
- Tremendous resources & community.

Django Handles Things

- Admin & Management Interface
- Database Schema & Migrations
- User Profiles and Authentication
- User Sessions and Cookies
- Internationalization

Anything is possible*.

Django Application


Django Application
Tools & Utilities
Web Process
Worker Process

Django Application

Tools & Utilities

Web Process

Worker Process

Management Tools

Supporting Services

Django Application

- Tools & Utilities
  - Management Tools
  - Supporting Services
- Web Process
- Worker Process
  - Deferred Tasks
  - Scheduled Tasks

Django Application

Tools & Utilities

- Management Tools
- Supporting Services

Web Process

- User Interface
- API Service
- Data Persistence
- CRUD Admin
- Authentication

Worker Process

- Deferred Tasks
- Scheduled Tasks

Single Codebases are Great

- All the benefits of the Django stack.
- Figure out architecture as you go.
- Shared modules keep you DRY.
- Make broad and sweeping changes quickly.
- Only need to deploy once.

Django Application

Tools & Utilities

Management Tools

Supporting Services

Web Process

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Worker Process

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Management Tools

Supporting Services

Web Process

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Worker Process

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Management Tools

Supporting Services

Web Process

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Worker Process

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Management Tools

Supporting Services

Web Process

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Worker Process

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Management Tools

Supporting Services

Web Process

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Worker Process

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Management Tools

Supporting Services

Web Process

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Worker Process

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Management Tools

Supporting Services

Web Process

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Worker Process

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Management Tools

Supporting Services

Web Process

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Worker Process

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Management Tools

Supporting Services

Web Process

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Worker Process

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Management Tools

Supporting Services

Web Process

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Worker Process

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Management Tools

Supporting Services

Web Process

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Worker Process

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Management Tools

Supporting Services

Web Process

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Worker Process

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Management Tools

Supporting Services

Web Process

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Worker Process

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Management Tools

Supporting Services

Web Process

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Worker Process

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Management Tools

Supporting Services

Web Process

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Worker Process

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Web Process

Worker Process

Management Tools

Supporting Services

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Web Process

Worker Process

Management Tools

Supporting Services

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Web Process

Worker Process

Management Tools

Supporting Services

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Web Process

Worker Process

Management Tools

Supporting Services

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Web Process

Worker Process

Management Tools

Supporting Services

User Interface

API Service

Data Persistence

CRUD Admin

Authentication

Deferred Tasks

Scheduled Tasks

Django Application

Tools & Utilities

Web Process

Worker Process

Management Tools

Supporting Services

User Interface

API Service

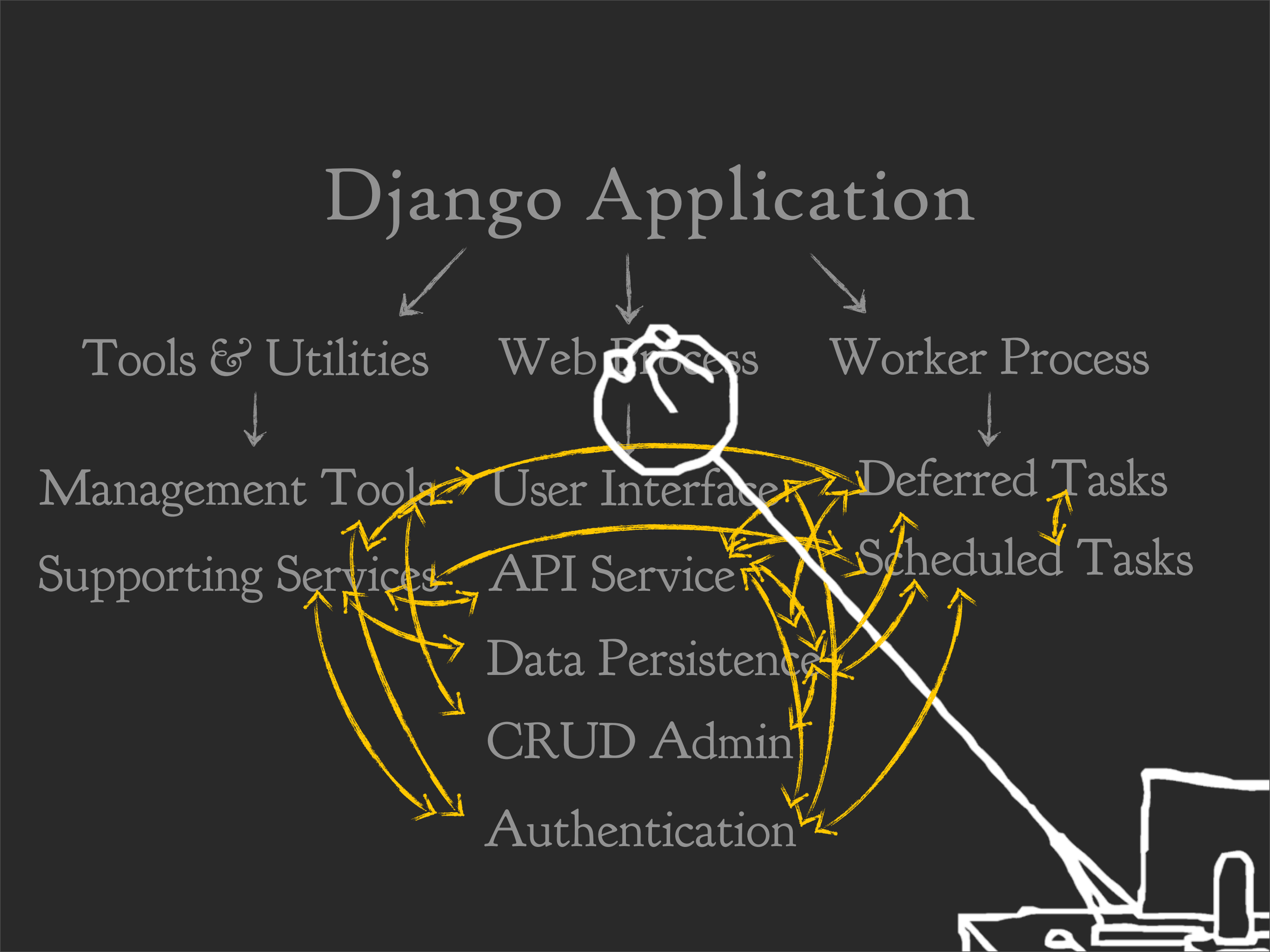
Data Persistence

CRUD Admin

Authentication

Deferred Tasks

Scheduled Tasks



Single Codebases are Evil!

- Components are tightly coupled.
- Broad tribal knowledge is required.
- Iterative change of components difficult.
- Technical debt has a tendency to spread.
- Forced to deploy everything at once.

Anything is possible*.

* But it'll often be a monolithic app.

CONSTRAINTS FOSTER CREATIVITY

Constraints are Good

- Text Editors vs IDEs
- Prime vs Zoom Lenses
- Mac OS X vs Desktop Linux
- Pen & Paper vs Digital Notes
- Monolithic Apps vs Services

Keep ‘em Separated

Developers

End Users

API Service ← Frontend

Data Persistence

Build for Open Source

- Components become concise & decoupled.
- Concerns separate themselves.
- Best practices emerge (e.g. no creds in code).
- Documentation and tests become crucial.
- Code can be released at any time.

Build for Services

- Components become concise & decoupled.
- Concerns seperate themselves.
- Best practices emerge (e.g. ideal tools).
- Documentation and contracts become crucial.
- Services can be scaled separately at any time
- Dogfood is delicious.

Composability

End Users

Developers

End Users

Developers

End Users

API Service

Developers

End Users

API Service

API Service

Developers

End Users

API Service → API Service →

Developers

End Users

API Service → API Service → API Service

Developers

End Users

(API Service → API Service → API Service)

Developers

End Users

(API Service → API Service → API Service)

Data Persistence

Developers

End Users

Internal

(API Service  API Service  API Service)

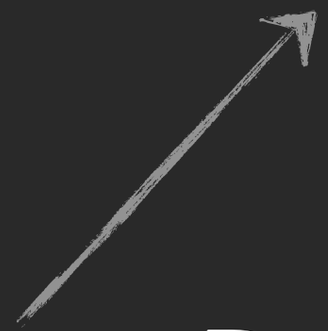
Data Persistence

Developers

End Users

Internal

(API Service → API Service → API Service)



Message Queue → Workers

Data Persistence

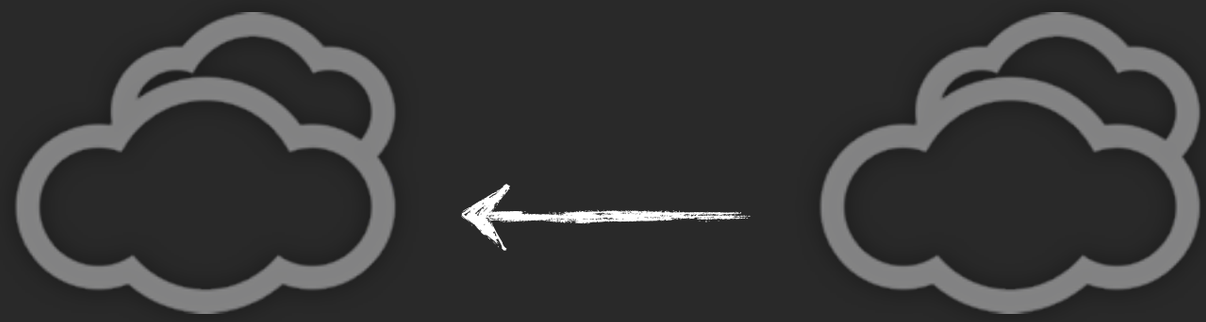
Let’s try it out.

We’ll use Django, of course.

Django: It Just Sits There™

Two Basic Components

API Service & Frontend Client



API Service	Frontend Client
Data Interface	User Interface
Business Logic	State and Sessions
Authentication	Public Face

Django for API Services

(and nothing else)

- Significant boilerplate code for simple views.
- No need for templates, tags, etc.
- API libraries are buggy; could use some love.
- `if request.method == 'POST'`

Django as API Consumer

(and nothing else)

- Keep in mind, database is handled by API.
- Makes modular decisions for you.
- Deals with the database for you.
- Installable third-party Django apps.

Django as API Consumer

(and nothing else)

- Everything is tied to the ORM.
- Third-party Django apps do too.
- User model requires sessions; isn't flexible.

What's Left?

Flask

Enter, Flask.

- HTTP Web Framework based on Werkzeug.
- Excellent for building web services.

Elegant & Simple

```
from flask import Flask
app = Flask(__name__)

@app.route(”/”)
def hello():
    return ”Hello World!”

if __name__ == ”__main__”:
    app.run()
```

Flask Familiarities

- WSGI Application Framework.
- Built-in Templating System (Jinja2).
- Active community, extensions for everything.

Flask Philosphies

- Started as an April Fool's joke.
- Very minimal; 800 lines of code.
- Heavily tested; 1500 lines of tests.
- Exhaustively documented; 200 pages of docs.
- Layered API; built on Werkzeug, WSGI.

Flask Differences

- Explicit & passable app objects.
- Simple, elegant API. No boiler plate.
- BYOB: Bring Your Own Batteries.
- No built-in ORM or form validation.
- Context locals. Keeps things looking clean.

Flask Improvements

- Fewer batteries == greater flexibility.
- Jinja2 is an incredible template system.
- Everything harnesses actual references.
- Configuration is a simple dictionary.
- It's hard to build monolithic applications.
- Response objects are WSGI applications.

Flask Improvements

- Werkzueug debugger.
- No import-time side effects.
- Signals system outside of ORM.
- Tests are simpler with real app objects.
- `return (content, status)`

Popular Extensions

- Flask-SQLAlchemy: Database Mapper.
- Flask-Celery: Delayed Jobs.
- Flask-Script: Management Commands.
- Flask-WTF: Form Validation.

Shameless Plug

- Flask-SSLify: App HSTS (SSL) Policies
- Flask-GoogleFed: Google Federated Auth
- Flask-Heroku: Env Variable Configurations

Flask is a sharp tool for building sharp services.

Use the right tool for the job.

Torn?

Flask vs. Django

Alcohol

vs.

Ponies

Why not both?

Services are agnostic.

Just speak HTTP.

Questions?

github.com/kennethreitz