API Driven Development

How I Develop Things and Why.

Kenneth Reitz
Hi.
Open Source
Requests

HTTP for Humans

```python
>>> 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, ...}
```
$ 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
Purpose
We’re Diverse.

We come from many backgrounds.

- Product guys: non-tech cofounders
- Sales guys: hustlers
- Marketers: spammers
- Designers: pixel pushers
- Developers: code monkeys
What do we have in common?
We’re Makers.

We craft experiences & interfaces.

- Product guys: visionaries
- Sales guys: sustainability
- Marketers: communication
- Designers: experience and philosophy
- Developers: make the magic happen
Developers!

Developers, Developers, Developers.
People are going to be spending two or three hours a day with these machines — more than they spend with a car.

— Steve Jobs, 1983
Software design must be given at least as much consideration as we give automobiles today — if not a lot more.

— Steve Jobs, 1983
That worked.
Beautiful Interfaces.

Today, beautiful applications abound.

- Industrial Design
- Web Interfaces
- iOS, Android, Mobile Apps
- Desktop Clients & Applications
Hackers are the real Makers.
Developers spend 8+ hours a day with APIs.

Why are they treated differently?
Web Application
Web Application

Tools & Utilities  Web Process  Worker Process
Web Application

Tools & Utilities

Web Process

Worker Process

Management Tools

Supporting Services
Web Application

Tools & Utilities
Management Tools
Supporting Services

Web Process

Worker Process
Deferred Tasks
Scheduled Tasks
Web Application

Tools & Utilities
  Management Tools

Supporting Services

Web Process
  User Interface
  API Service
  Data Persistence
  CRUD Admin
  Authentication

Worker Process
  Deferred Tasks
  Scheduled Tasks
Web Application

- Tools & Utilities
- Web Process
- Worker Process

Management Tools
User Interface
Deferred Tasks
Supporting Services
API Service
Scheduled Tasks
Data Persistence
CRUD Admin
Authentication
Web Application

Tools & Utilities
Management Tools
Supporting Services

Web Process
User Interface
API Service
Data Persistence
CRUD Admin
Authentication

Worker Process
Deferred Tasks
Scheduled Tasks
Web Application

Tools & Utilities
Management Tools
Supporting Services

Web Process
User Interface
API Service
Data Persistence
CRUD Admin
Authentication

Worker Process
Deferred Tasks
Scheduled Tasks
End Users
Developers

End Users
Developers → API Service → End Users
Developers → API Service → End Users → API Service → API Service
Developers

API Service

End Users

API Service

API Service

Data Persistence
Developers

$\rightarrow$

API Service

$\rightarrow$

End Users

$\rightarrow$

API Service

$\rightarrow$

Internal

$\rightarrow$

API Service

$\rightarrow$

Data Persistence

Thursday, October 11, 12
Everything is a remix*.

* APIs Rule Everything Around Us.
How?
Step I: Have an Issue.
A Real, Tangible Problem.

You can’t solve a problem properly if you don’t experience it firsthand.
Example: OneNote.

The finest note-taking platform on earth.

• Hierarchical, freeform note-taking software that assumes nothing.
• Only available on Windows.
• I want to make OneNote for OS X.
• It would be incredible.
GitHub Success

Over two million people collaborating.

- GitHub wasn’t built for the developer community at large.
- Resonated with millions of developers.
- They themselves happen to be developers.
Other’s Success

- Gumroad, built for the founder.
- 37 Signals product, build for the team.
- Ruby on Rails, by Rubyists for Rubyists.
Optimization

What drives your decisions?

- Feature driven development?
- Profit driven development?
- Growth driven development?
- Problem driven development.
**pragmatic** | *prag'matik*, adj:

Dealing with things sensibly and realistically in a way that is based on practical rather than theoretical considerations
Requests Success

- Python is a language built for Humans.
- Why should HTTP be non-trivial?
- I explored and discovered what I really needed, and built it.
- I had a real problem that I solved for myself.
Requests Success

- At first, Requests was far from powerful.
- But, it deeply resonated with people.
- Features grew over time, but the API was never compromised.
- Quite popular.
Developers spend 8+ hours a day with APIs.

Build for yourself—a developer.
Step II: Respond.
Write the README.
• Before any code is written, write the README — show some examples.

• Write some code with the theoretical code that you’ve documented.
Achievement Unlocked!

- Instead of engineering something to get the job done, you interact with the problem itself and build an interface that reacts to it.

- You discover it. You respond to it.
Sculptures, Etc.

- Great sculptures aren’t engineered or manufactured—they’re discovered.
- The sculptor studies and listens to the marble. He identifies with it.
- Then, he responds.
- Setting free something hidden that inside all along.
Responsive Design

- It’s not about a design that will “work” on a phone, tablet, and desktop.
- It’s about making something that identifies itself enough to respond to the environment it’s placed in.
- Free of arbitrary constraints.
Readme-Driven Development?

Responsive API Design.
Step III: Build.
Responsive Design

- Once you discover the API: build it.
- Write all the code necessary to make exactly what you documented happen.
- Complex code? Layer your API.
- “Porcelain” layer is documented.
The API is all that matters.

Everything else is secondary.
Do unto others as you would have them do to you?

Build tools for others that you want to be built for you.
Pro Tips™
CONSTRAINTS
FOSTER
CREATIVITY
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.
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 separate themselves.
- Best practices emerge (e.g. ideal tools).
- Documentation and contracts become crucial.
- Services can be scaled separately at any time
- Dogfood is delicious.
Simplicity is always better than functionality.

— Pieter Hintjens
Questions?

github.com/kennethreitz