

---

# Restful Interfaces to Third-Party Websites with Python

---

Kevin Dahlhausen  
kevin.dahlhausen@keybank.com

---

# My (pythonic) Background

- learned of python in 96 ← Vim Editor
  - started pyFltk
  - PyGallery – an early online static photo gallery generators
  - wxGlade plugin– floatspin
  - hardware tester for embedded system
  - desktop – ErgMate, ErgTweeter
-

---

# Motivation – Why REST?

- consuming a SOAP web-service in J2EE
    - Simple call – get list of bank officers
    - Generated / modified: 33 files over 5 directories!
  - easy to get started (rest-like) creating interface
    - harder to be strict
  - lightweight
  - cross-platform
  - tools/libraries widely available
    - view headers and content – browser/proxy
-

---

# REST – Review

- architectural style
    - here (nearly everywhere) as applied to HTTP
  - Representational State Transfer
    - resources have unique URI
    - client retrieves representations (views) of resources via url
    - representations control state of the client
  - Roy Fielding dissertation
  - leverages existing web infrastructure and protocols
-

---

# REST – As Applied to HTTP

- Get
  - Head
    - last modified / meta / existence
  - Put
    - specify all data, complete resource
  - Delete
  - Post
    - update all/partial existing, server assigned
    - not idempotent
  - Options - discovery
-

---

# Adding API to Website that has None

- Web (Screen) Scraping
    - programmatic web browser
  - Some problems with this technique
    - brittle – tightly coupled
    - tedious
    - dynamic websites
-

---

# Managing Risks of Screen-Scraping

- brittle
    - testing
      - parser, domain, api
  - tedious
    - library/toolset choice
  - dynamic
    - library
    - browser automation
-

---

# Python Tools: Mechanize

- John J. Lee
  - programmatic browser in python
    - cookies
    - redirects / refresh
    - honors robots.txt
    - forms
    - proxies
-



---

# Python Tools: Beautiful Soup

- Leonard Richardson
- html/xml parser
  - accepts broken input
  - pure python (no native module)

*“You didn't write that awful page. You're just trying to get some data out of it. Right now, you don't really care what HTML is supposed to look like. Neither does this parser.”*

- Ruby port: [Rubyful Soup](#).
-

---

# Python Tools: Nose

- Jason Pellerin
  - unit-test runner
    - collects tests automatically – no need for suites
    - parametric tests using generators
      - example1.py
    - coverage, profile, debugger, attributes ...
    - plugins
      - <http://nose-plugins.jottit.com/>
-

---

# Python Tools: Nosy

- Jeff Winkler
  - watches files
  - runs nosy when code changes
  - simple script -> ~20 lines of code
-

---

# Python Tools: Nosed Reloaded

- Doug Latornell
  - enhanced Nosed script
  - supports config files
    - directories/file glob patterns to watch for changes
      - \*, ?, and character ranges expressed with []
    - list of options to send to nose
      - attributes -> id sets of tests to run
-

---

# Python Tools: django-piston

- framework helps with REST implementation
  - dispatch -> HTTP verbs
  - authorization
    - basic, open-auth, custom
  - many formats
    - json, yaml, xml, pickled python objects
  - throttling
    - user id, ip-address
    - global or grouped
-

---

# Python Tools: google app-engine

## ■ why?

- python
- cost
- scalability
- cloud computing – wanted to learn about it

## ■ challenges

- django -> app-engine patch
  - piston -> manual changes to model, django-piston
  - mechanize -> minor changes urllib, blog
  - no native modules (lxml)
-

# Case-Study: Concept 2 Online Logbook



## Use Cases:

- add workouts
- get total distance
- synchronize online log with spreadsheet

**concept 2**  
ROWING

Logbook Rankings History Challenges Ranked Workouts Profile Teams Logout

**Auggie Dog's Logbook**  
6,310,124m

**2010 Season**  
May 1, 2009-April 30, 2010

**Holiday Challenge**  
November 26-December 24

**Quick Links**

- Training Forum
- Workout of the day
- Affiliation Standings
- Ranking FAQs / Help
- Pace Calculator
- Weekly Winners
- Indoor Rower Product Information
- SkiErg Product Information
- Member of the Day


**Concept2 Online Community Statistics**  
Active Members This Season: 29,367  
Total Meters This Season: 6,047,733,863

**News**  
**CRASH-B Sprints Erg Sales**  
Concept2 CTS will be selling a limited number of Concept2 Indoor Rowers used at the 2010 CRASH-B Sprints World Indoor Rowing Championships at a discount. These demos are only used for racing at CRASH-Bs.  
[Learn more](#)

**Virtual Team Challenge**  
**January 1-January 31**  
The 2010 Virtual Team Challenge has begun, but there's still time to join in if you haven't already! Teams and team members have until January 15 to join the challenge. If you're already a member of a team, make sure you've confirmed your participation on your [Team](#) page.  
[Find out more about the VTC](#)  
[View the 2010 VTC Team Standings](#)

**The 10th Annual Holiday Challenge!**  
Thanks to everyone who participated in this year's Holiday Challenge to benefit Oxfam International, Slow Food USA, Feeding America and The Center for an Agricultural Economy. You have until **January 4, 2010**, to finish entering your meters, so please make sure you enter all your Holiday Challenge meters in your logbook by that date. Thanks!

**Facebook Application**  
If you're on Facebook, you can now have a box on your profile that shows your latest workout as well as a summary of your overall history. You can also add a Logbook tab that allows you to see the most recent pieces from friends on Facebook that have also added the application!  
[Go to the application](#)

**Add Personal Live Stats to your Google Home Page**  
Use this button to add a live stats feed to your [iGoogle.com](#) home page:  
  
You will need to know your Ranking ID#. It can be found on your Profile page.

Your Challenge Meters: 0  
Your Money Raised: \$0.00  
Your Charity: Not yet chosen  
Total Challenge Meters: 4,344,037,994  
Total Money Raised: \$28,050.00  
Complete at least 100k during the **Holiday Challenge** and Concept2 will donate \$.02 for every kilometer to your choice of four charities. Row or ski beyond 100k and we'll donate \$.04 per additional kilometer. Our goal is to raise a total of \$30,000.  
[Choose Your Charity](#)  
Check out the **Honor Boards** to see who has completed each challenge.

**Weekly Winners**  
**For week of Dec 19-Dec 25**  
20,000m+: Margaret Wright, USA (21,000)  
50,000m+: Tom Reed, USA (164,157)  
[More on the weekly winners](#)

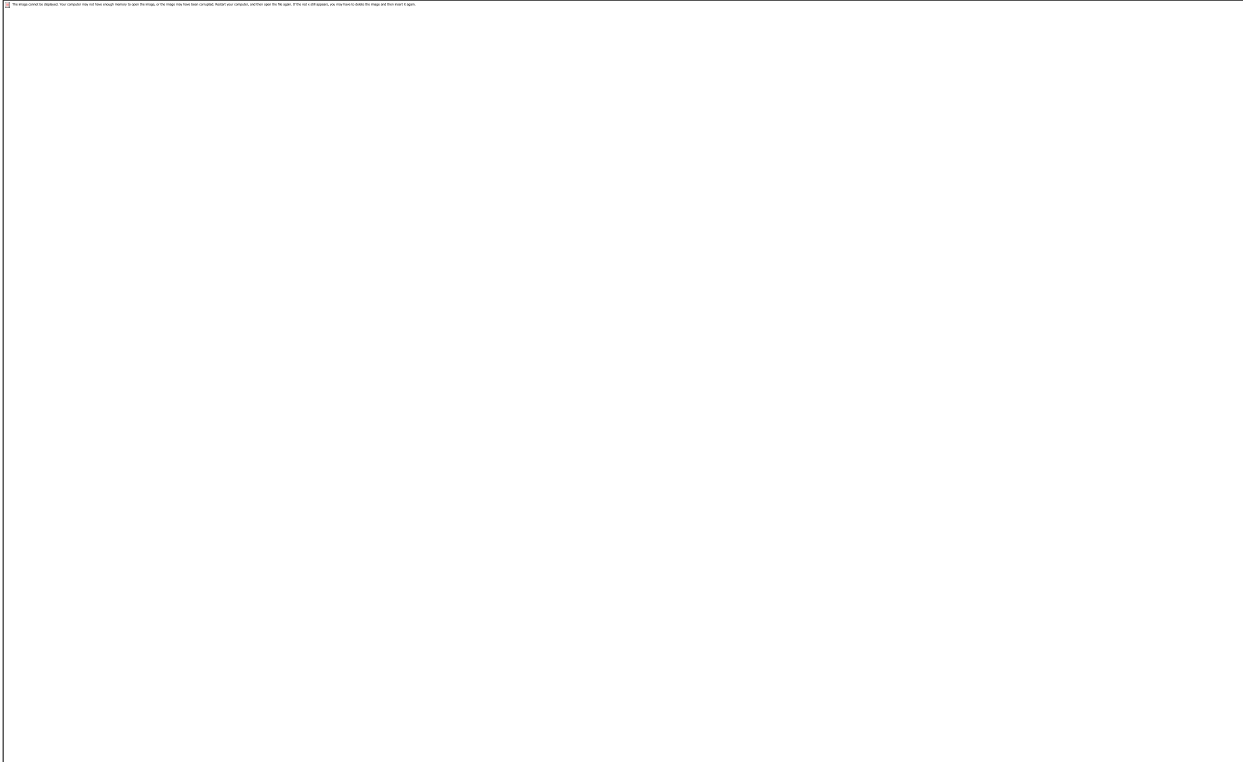
**Add new workout**

Distance: 5000 Meters  
Time: 21 : 5 : 1.Tenths  
Date: Jan 10, 2010  
Type: Indoor Rower  
Weight:  LWT  HWT  
Comments: You have 120 characters left

Add

---

# (Case Study) Example – Get All Workouts



**(authentication / api-key passed in headers or as parameters)**

---



# (Case Study) Example – Get All Workouts

## JSON:

[http://c2logapi.appspot.com/api/1/currentseason/workouts?api\\_key=xxxxxxxxx](http://c2logapi.appspot.com/api/1/currentseason/workouts?api_key=xxxxxxxxx)

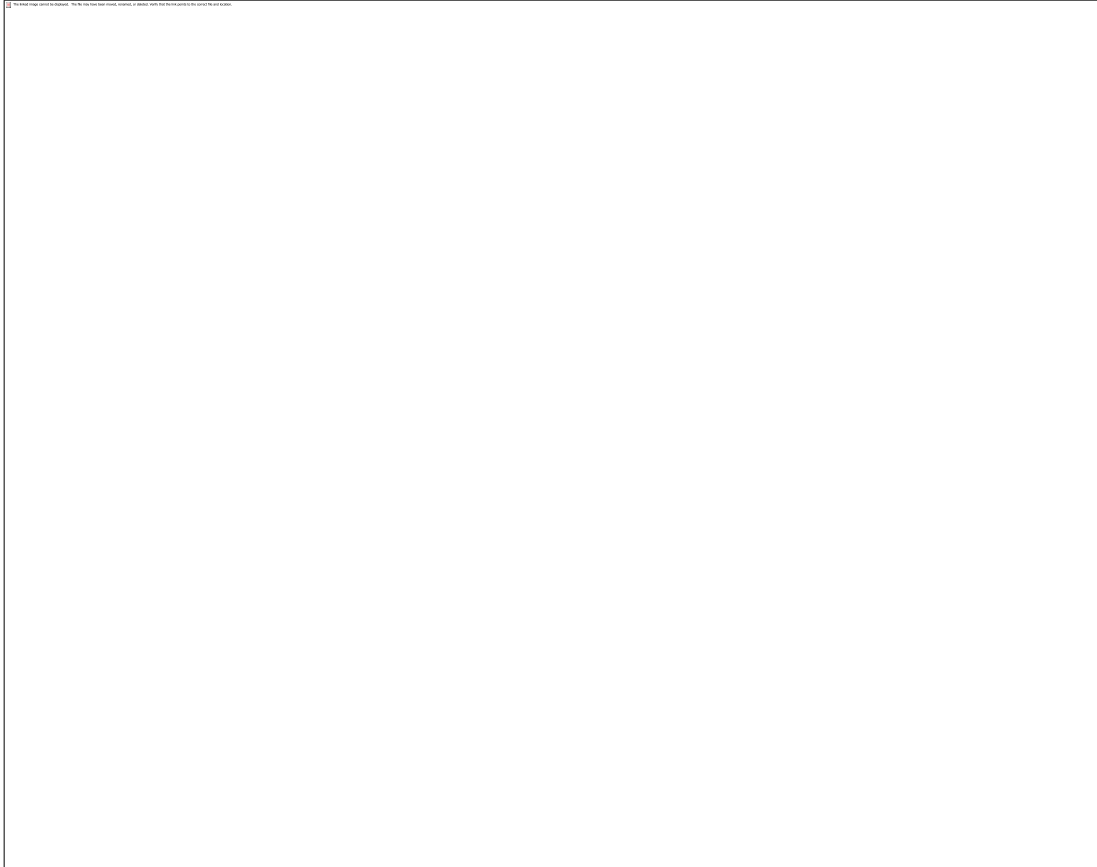
```
[
  {
    "distance": 5000,
    "seconds": 11,
    "age": 41,
    "day": 11,
    "comments": "fly and die - tired, dehydrated",
    "month": 1,
    "hours": 0,
    "weightClass": "H",
    "link": { "href": "http://c2logapi.appspot.com/api/1/currentseason/workout/11360463", "rel": "self" },
    "typeOfWorkout": "indoor rower",
    "year": 2010,
    "minutes": 21,
    "id": 11360463,
    "tenths": 3
  },
  {
    "distance": 5000,
    "seconds": 5,
    "age": 41,
    "day": 8,
    ...
  }
]
```

# (Case Study) Example – Get All Workouts

## **XML:**

[http://c2logapi.appspot.com/api/1/currentseason/workouts.xml?api\\_key=xxxxxxxx](http://c2logapi.appspot.com/api/1/currentseason/workouts.xml?api_key=xxxxxxxx)

[http://c2logapi.appspot.com/api/1/currentseason/workouts?format=xml&api\\_key=xxxxxxxx](http://c2logapi.appspot.com/api/1/currentseason/workouts?format=xml&api_key=xxxxxxxx)



---

# Preparation: accounts

- create non-user accounts
    - a development account
    - regression – 1+
      - data loading
  - might be obvious – but.... keep testing away from active user accounts
-

---

# Step 1: Script to store pages locally

- local copies
  - efficiency
  - good user
  - diff

example2.py

- avoid browser -> save
-

---

# Step 2: Write a unit test for the parser

- test
  - reads html from local file
  - passes to parser function
    - getTotalDistance(data)
      - parse data return value

example3.py

---

---

# Step 3: Implement the parsing

- idiom:
  - parser:
    - has html content passed to it
    - getTotalDistance(data)
      - parse data return value
  - domain object:
    - getTotalDistance()
      - uses mechanical browser to get page data
      - then calls parser

example4.py + HomePageLoggedIn.html

---

---

## Step 4: Add a domain-level test

- calls domain object against regression test account
- tag 'live' tests using Nose attributes

```
from nose.plugins.attrib import attr
```

```
@attr('live')
```

```
def testLiveGetDistance():
```

```
·
```

```
·
```

---

---

## Step 5: Add regression test for the API

- structure tests to run against local or remote server
- validate HTTP result code
- could serve as examples
  - but don't

example6.py

---



---

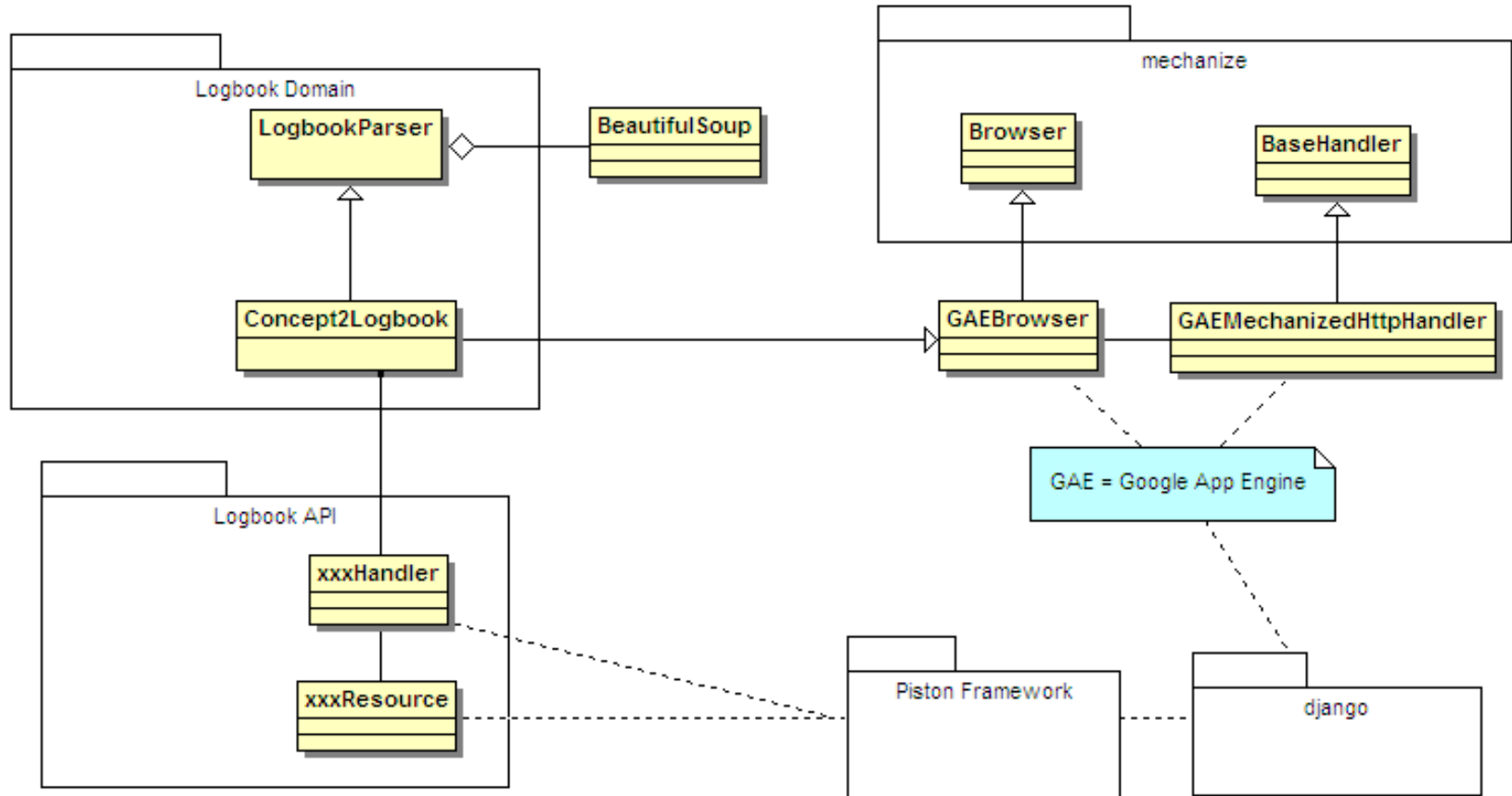
# Step 6: Expose domain object via your API

- authentication
  - consider for api and external site
- request throttling
  - piston – id, ipi, group
- ability to disable misbehaved clients
  - api key – exists, not disabled
- api versioning
- statelessness vs. efficiency

example7.py example8.py

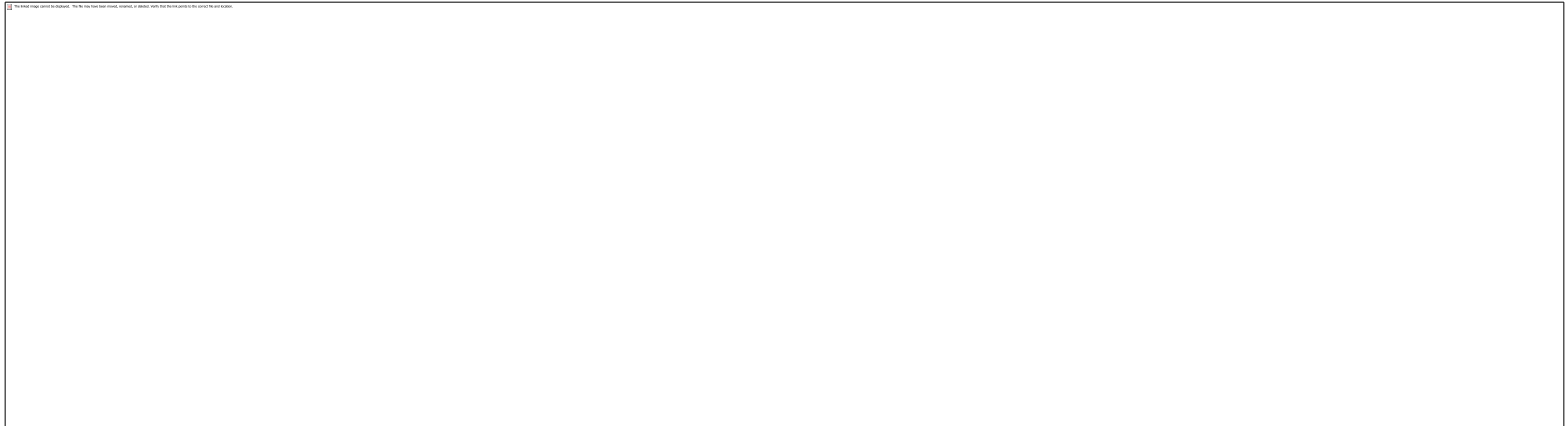
---

# (Case Study) Key Classes



---

# Deployment Diagram



# Case Study: implementation metrics

- 9 methods implemented
- domain package: 1178 loc (parsing ~26%)
- api package: 571 loc

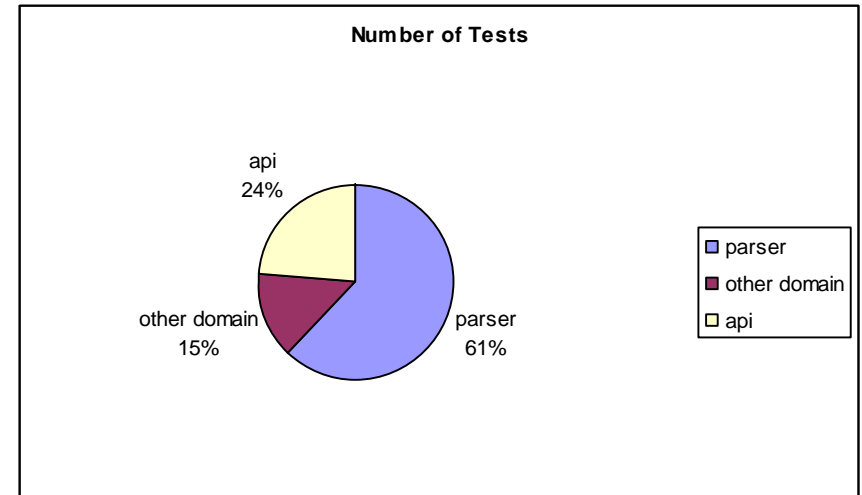
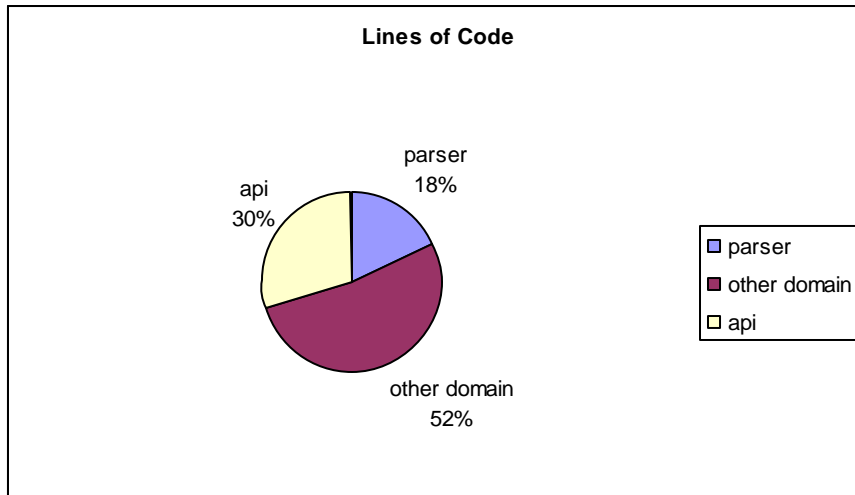
	LOC	% total
code	1749	59%
tests	1225	41%
	2974	

# Case Study: implementation metrics

	# tests	% total
parser*	68	62%
other domain	16	15%
api	26	24%
	110	

\*parser: 42 unique tests +  
(18 + 8) generated tests

# Case Study: implementation metrics



**testing emphasis on the html parsing**

---

# (Case Study) Lessons Learned

- REST is good
  - use nose 'attribute' plugin to segment tests
  - domain relationships
    - containment vs. inheritance
  - api regression tests as examples of usage
    - better to separate tests from examples
  - more logging – detailed usage
  - lxml for parsing -> Beautiful Soup
    - native library
-

---

# Resources

- *Roy Fielding's Dissertation (introduced REST):*  
[http://www.ics.uci.edu/~fielding/pubs/dissertation/rest\\_arch\\_style.htm](http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)
  - Allamaraju, *Subbu* and Mike Amundsen. RESTful Web Services Cookbook. O'Reilly, 2009 (Rough Cuts)
  - *Subbu Allamaraju's Publications :*  
<http://www.subbu.org/about/pubs>
  - *Python web-client programming FAQ:*  
<http://wwwsearch.sourceforge.net/bits/GeneralFAQ.html>
-



---

# Resources – python tools

- *mechanize* by John J.Lee  
<http://wwwsearch.sourceforge.net/mechanize/>
  - *Beautiful Soup* by Leonard Richardson  
<http://www.crummy.com/software/BeautifulSoup/>
  - *App-Engine Patch*  
<http://code.google.com/p/app-engine-patch/>
  - *Django Piston* by Jesper Noehr  
<http://http://bitbucket.org/jespern/django-piston>
  - *Nose/Nosy/Nosy Reloaded* by Jason Pellerin / Jeff Winkler / Doug Latornell  
<http://somethingaboutorange.com/mrl/projects/nose/0.11.1/>  
<http://jeffwinkler.net/2006/04/27/keeping-your-nose-green/>  
<http://douglatornell.ca/software/python/Nosy/>
-