Evolving Your Framework Under Fire

Presented to the
PyCon 2010 Atlanta
20 February 2010

Tres Seaver
Agendaless Consulting, Inc.
tseaver@agendaless.com

BFG: What is it?

- “Pay only for what you eat”
- Draws on decade of using / maintaining large, DWIMish framework (Zope, Plone)
- Minimalism: URL dispatch, templating, and security

BFG: Do I Need It?

- Speed
- Learn only what you need
- “I'll have what he's having”

bfg

Why not Zope / Plone?

- Zope has cool concepts
  - Hierarchy
  - Fine-grained security
  - Easy extensibility
  - Component architecture
- “Pay for what you might need”
  - Steep learning curve
  - Extra “help” impedes performance

Why not Django / Pylons / TurboGears?

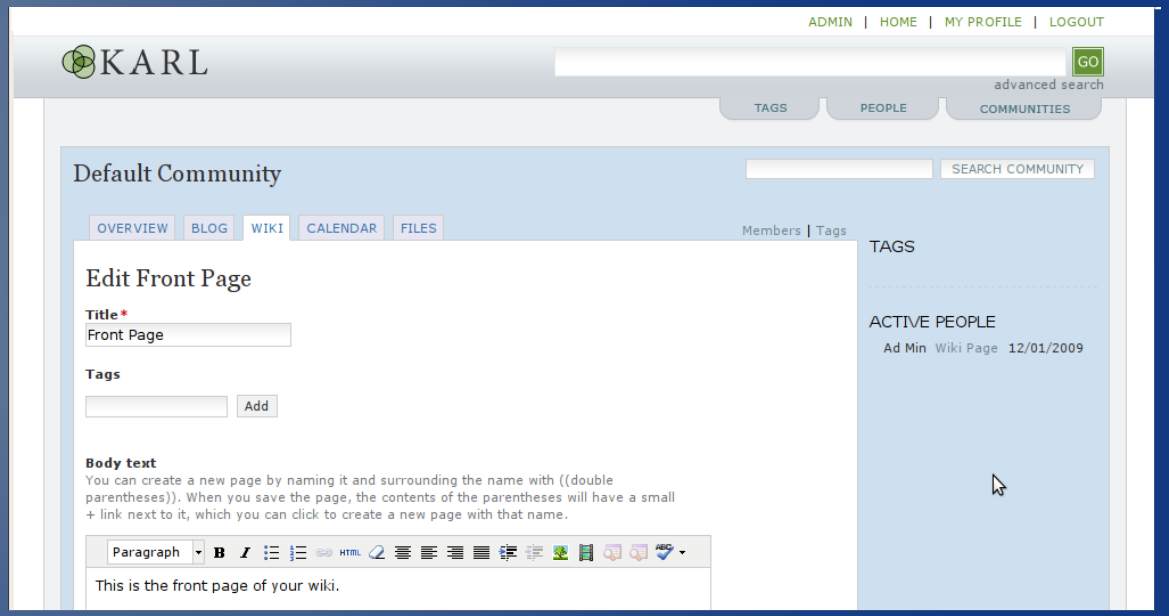
- Core Zope features harder to model
- Similar issues with “full stack” models
- Graph traversal rocks ;)

About KARL

- The Open Society Institute
- Mission-critical KM / extranet
- No anonymous requests
- I18N important
  - Content creation in users' languages of choice (French, Hungarian within OSI)

KARL UI Constraints

- “Coherent” UI choices
- Prefer consistency to features
- Aimed at casual users, rather than power users



bfg

“Community” is core abstraction

- Membership is invitation-based
- Members collaborate on content
  - Blog
  - Wiki
  - Calendar
  - File sharing
- Fierce security requirements

Prior Development

- Original development on Plone / Windows, 2006 - 2008
- Migrated to Plone / Linux, summer 2008
  - 2500 users
  - 400+ “communities”
  - 75,000 content pages

Why port KARL to BFG?

- Feature development slowed
- Stability issues
- Performance issues
  - 1 – 10 requests / second
- OSI actively marketing KARL to peer organizations

Doing the port

- “Transparent” port
  - No visible changes to user experience
- 4 – 5 developers dispersed across timezones
- Spread over 6 months
- Migration of content eased by prior migration from KARL1 to KARL2

Results of the port

- KARL now runs at 100+ requests / second on developer laptops
  - Fully-authenticated
  - Non-cached
  - Most expensive page
  - Faster on server hardware (~350 r/s)
- Stability / availability no longer issues
- Reduced hardware requirements

Evolving the framework to meet KARL's needs

- Move towards documentation-centric approach helped dispersed teams stay productive
- 100% test coverage as a minimum requirement
- Forcible conversion of encoded text at application boundaries
- Judicious use of component architecture to isolate customer-specific policies
- Move to make views more easily testable

BFG during KARL 3 development

- 40 BFG releases between initial “spike” (2008-09) and roll-out (2009-06)
- Dozens of releases of other supporting components
  - chameleon templating engine
  - repoze.who authentication
  - Other repoze middleware, libraries
- BFG documentation kept in sync with feature development

Since KARL 3.0 launch

- KARL now at 3.12 in production
- 5 customer instances (3 non-OSI)
- Dozens of BFG releases
  - BFG 1.0 (2009-07)
  - BFG 1.1 (2009-11)
  - BFG 1.2 (2010-02)
- Supporting packages continue active development

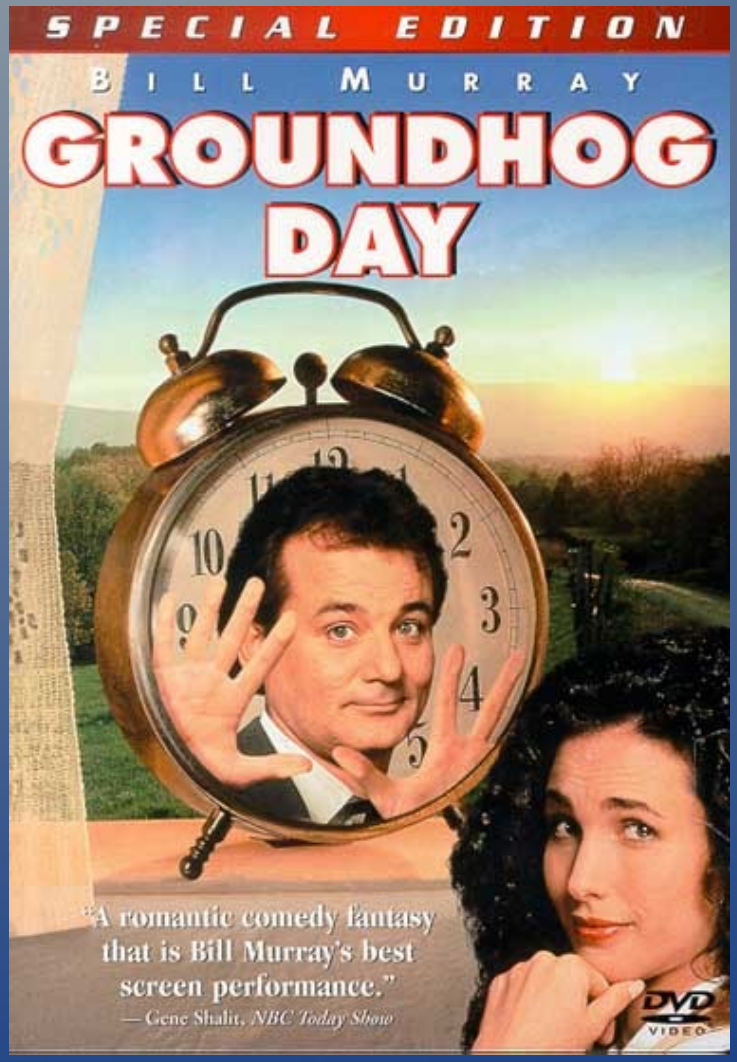
Lessons Learned

Lesson: Reuse can be overvalued

- Costs of reuse vs. benefits
- Fork now, merge later
- Criteria for dependability
  - Docs!
  - Test Coverage
  - Readability
  - Community
  - Docs!

bfg

Lesson: Repeatability wins



- Cultural: "works on my laptop" is not enough
- Frameworks can help by enabling / modeling repeatable testing, builds

bfg

Lesson: Optimize by removal

- Easiest code to optimize is the code which isn't there
- Remove indirections / abstractions which provide unneeded flexibility or features
- Framework which mandates features makes everybody pay

bfg

Lesson: Test coverage necessary, not sufficient

- “100%: the least you can do”
- Testing semantics may require more
- Unit testing: “Did I break anything?”
- Functional testing: “Are we there yet?”

bfg

Lesson: Backward compatibility can be broken

- Stripping away cruft wins
- Document required changes
- Automate evolution of persistent data

bfg

Lesson: Make measurements easy, obvious

- "Speedometer" in page header
- Test breakage reports
- "Complexity" (stack depth, function calls) as a measure of both speed, learning curve

bfg

Lesson: Documentation hurts so good

- Explain (possibly) dumb decisions before implementing
- Keep the team productive
- End up with a nice book
  - Buy it at the expo!

bfg

Resources

- KARL Project: http://karlproject.org/
- BFG Home: http://bfg.repoze.org/
- BFG Docs: http://docs.repoze.org/bfg/1.2/
- BFG Book: http://bfg.repoze.org/book
- IRC: irc://irc.freenode.net/#repoze
- BFG open space @ 4:00 PM today