Portlets in Python

Using jython to write JSR-168 portlets

Santiago Gala

ApacheCon
Europe 06

Background

portals.apache.org

- jetspeed
- pluto
- graffito
- WSRP4J
- Portal Bridges

- strong java background
- (but) we want to do cross-language portals

Jetspeed

since around 1999

- portlet API (JSR-168) evolved out of proposals in jetspeed-dev@jakarta.apache.org
- early experiments
- “legacy” jetspeed-1.6 that can do JSR-168 portlets
  - using **fusion** -> embedded pluto portal container

Jetspeed

v 2.X re-designed with portlet API in mind

- Spring pipelines
- JAAS auth
- portlets (using internal attributes) for layout and decoration
- uses pluto as portlet container

Pluto

since 2004 or so

- reference implementation of JSR-168
- version 1.0.x out of original donation
- refactoring in 1.1

Cocoon portal

- uses pluto as portlet container too
- integrated with the cocoon xml framework: cocoon.apache.org

Graffito

- CMS portlets
- from Jetspeed to incubation
- waiting for enough community to join portals
  - uses jackrabbit
    - Java Content Repository API (JSR 170)

WSRP4J

- Oasis standard for remote portlets
- under incubation
- via WSRP (Web Services remoting of portlets, Oasis Standard)

Bridges

- for java frameworks such as JSF or Struts
- for CGI scripts (perl, PHP)
- Using JNI for PHP 4
- --> python, for jython integration

Why cross-language?

- safety network
  - cross breeding
  - java is beginning to show its age
- integrators want to “mix and match”
  - php pool of common knowledge/code
  - legacy CGI scripts in companies
  - “postmodern” frameworks

Portlets

Similar to servlets, but

- isolated (from other portlets) in the page
- with a richer (two way) interaction model
  - action -> changes in model
  - render -> pure view
- allow delivery of portlet applications (.war) with portlet.xml config

JSR-168

- PortletAPI
- Used mostly for highly dynamic intranet webapps
- Enterprise ready technology
- Currently being updated in JSR-286

where do portlets fit in MVC?

portlets typically act as controllers:

- perform actions (update the model)
- dispatch to a view/edit/help template during render
- can switch views according to preferences or state (minimized, maximized)
- partial help from the portal

portlet modes (VIEW, EDIT, HELP, ...)

Too much freedom?

- nothing impedes a portlet to be all three things (M, V and C)
- a very open technology, like servlets
  - much better (action/render)
- Needs Frameworks to shape its use
  - interaction with Java Server Faces
  - Good practices

Common practices

- views are JSP pages
- views using velocity
- bridging frameworks
  - JSF
  - struts

Dynamic languages

- no static typing (or optional)
- duck typing (pure Liskov substitution principle)
- php, perl, python, ruby, ecmascript, ... (even VB)

Why Dynamic?

- Looking for productivity gains
  - concise
  - expressive
- sweet spot for scripting/prototyping
  - changing (or)
  - unclear requirements

Python

- Very well managed evolution
- Mature yet fresh
- Has reasonable java (and C#) implementations

Why not XXX?

- I like the python syntax (YMMV)
- easy to learn
- simple code looks simple
- can do complex things, but it has entry barriers
- more “enteprisey” than Ruby (IMNSHO)
- very good for inter-language glue

Jython

- It is at the 2.2 alpha 1 level
- allows for tight java integration
  - python objects can inherit from java objects
  - python classes turn into java classes
  - bean-like properties
- moving again after migration to svn

Test case for classpath

- I wrote a quick and dirty jython testcase for a bug in classpath (java.security)
  - interactive
  - easy to understand
  - concise
- Also, jikesrvm hacking example

“postmodern” Web frameworks

- TLA (Three Letter Acronyms) packed:
  - DRY (Don't Repeat Yourself)
  - MVC (Model-View-Controller) *forced* on you
  - ORM (Object-Relational-Model)for simple DB generation
- advanced template engines
- fit for the task

back to PHP-world?

- Zero-training (Sam Ruby)
  - Slashcode or Duke-nuke
  - but also Moodle or Dokeos
- new frameworks are more structured, and abstract DB at the right level

TurboGears example

- wiki in 20 minutes
  - model -> (transparent DB access)
  - controller using annotations for view mapping
  - views use kid, neat if you like XML

Do we need continuations?

- they have been used to model non-linear navigation in webapps
  - “universal” Back button
- a tree of program “stacks” (really activation frames lists)
  - storing virtual PC, locals, globals
  - what scheme calls “environment”
- difficult to grok and tricky to use

for Wizards?

- The portlet API model decouples each portlet from the page behaviour
  - each portlet only sees actions addressed to it, and render **must** be idempotent
- portals don't deal very well with Back

Wizards

- Each portlet can do linear flow programming...
  - programmer controlled “back”
- which allows wizards
  - data validation
  - multi-step transactions

Python generators

- Basically implement the Iterator pattern
  - next(), StopIteration, yield <value>
  - changing in python 2.5
    - to be shaped exactly as we need them
  - in 2.2 need “from __future__ import generators”
- Similar to java Iterator model

generator example

```
def fibo(max=100):
    a,b=1,1
    while a <= max:
      yield a
      a,b=b,a+b
>>>fibo()
<generator object at 0x30085288>
>>>[i for i in fibo()]
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

RequestGenerator

```
class validator(requestProcessor):
    def process(self):
        env = {}
        yield initial_page
      for req in self.requests():
            if req.cancel: raise StopIteration("cancelled")
            if not valid(req):
                yield form % env.update({'error':req.error})
            env.update(req)
            page = do_something(env)
            yield page
      raise StopIteration
```

python modulo operator

for strings...

...makes for a poor man template engine

```
"%s => %s" % (key, value)
"%(key)s => %(value)s" % dict(key=key, value=value)
```

with printf-like formats

There are better things (i.e. kid, cheetah, ...)

Performance

- jython has a slow startup
- for process intensive tasks 2/3 times slower than java

  most portlets are not process intensive...

  ...or at least the controller is not

  rss portlet using Mark Pilgrim's feedparser.py is reasonable performance and packed with features

  +++ jython demo layouts clearly faster than current velocity ones

  using modulo-templates :)

Great for integration

- demoing a portlet around an existing business infrastructure
- quick changes to test different UI approaches
- alternate business processes simulation

...you name it

Where to go from here?

- prototype committed in portals.apache.org/bridges

  for you to play with...

  ...send patches

  ...or commit new stuff

Crazy ideas beyond

- write a pure python servlet/portlet API emulation layer
  - to enable lightweight portlet prototyping
    - in my laptop
    - java server env too CPU and memory intensive

References

http://portals.apache.org

http://www.python.org

http://jython.sf.net

Questions?

Answers not guaranteed!

Thanks for your attention

hope you enjoyed the presentation