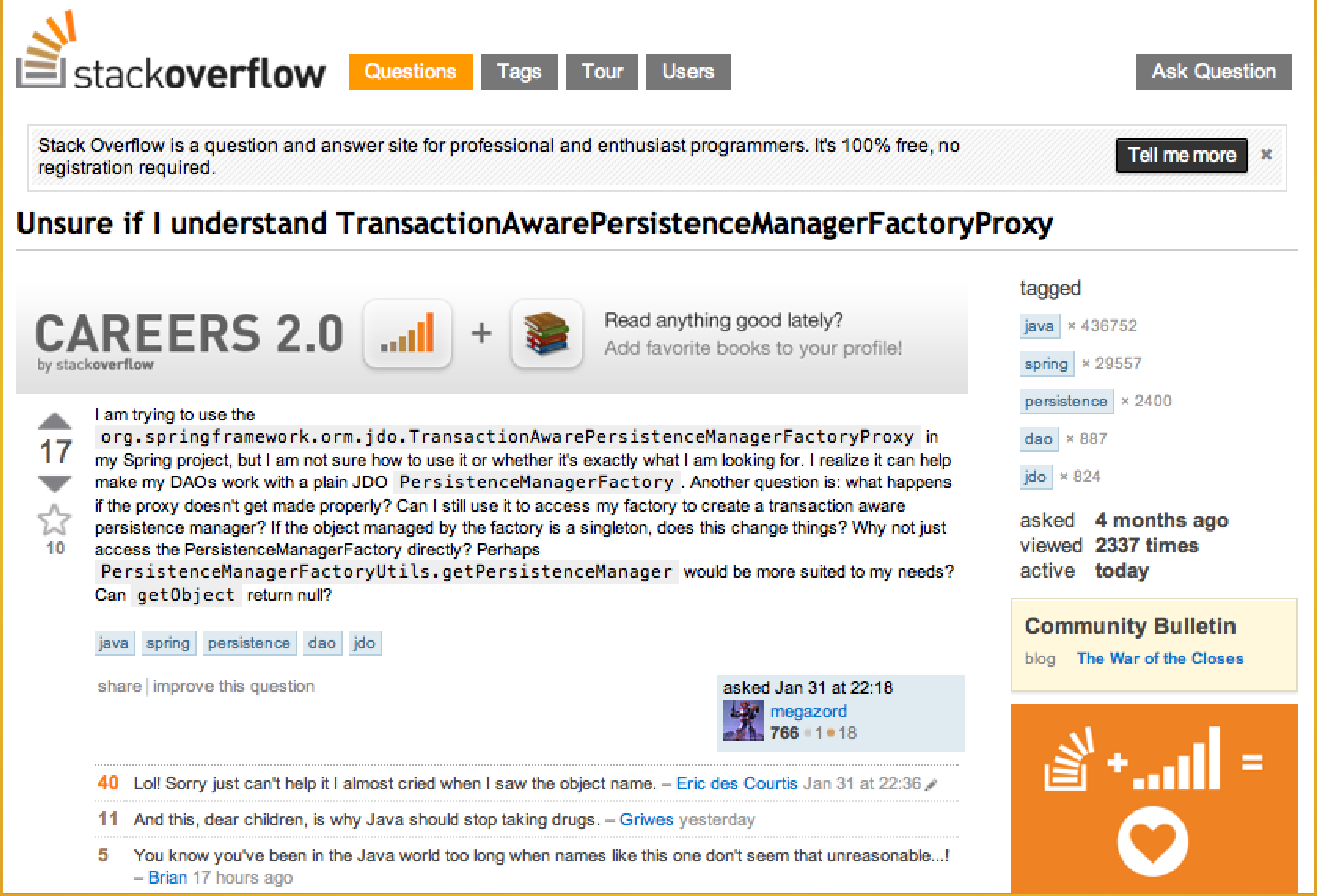
A State-Driven, Service-Oriented Dynamic Web Development Framework

Steve Huntley
stephen.huntley@alum.mit.edu

The Rails Apocalypse:

Motivation:

- 2013 has been an “Apocalyptic” year for object-oriented web frameworks
- Not just new exploits, but new classes of exploits
- Expected to take years to flush out and fix, if ever
- Total server pwnage is the new normal
- Bonus: object-relational mapping is a disaster for performance and scalability



Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

Unsure if I understand TransactionAwarePersistenceManagerFactoryProxy

Read anything good lately?
Add favorite books to your profile!

I am trying to use the `org.springframework.orm.jdo.TransactionAwarePersistenceManagerFactoryProxy` in my Spring project, but I am not sure how to use it or whether it's exactly what I am looking for. I realize it can help make my DAOs work with a plain JDO `PersistenceManagerFactory`. Another question is: what happens if the proxy doesn't get made properly? Can I still use it to access my factory to create a transaction aware persistence manager? If the object managed by the factory is a singleton, does this change things? Why not just access the PersistenceManagerFactory directly? Perhaps `PersistenceManagerFactoryUtils.getPersistenceManager` would be more suited to my needs? Can `getObject` return null?

java spring persistence dao jdo

asked Jan 31 at 22:18
megazord
766 ● 1 ● 18

40 Lol! Sorry just can't help it I almost cried when I saw the object name. – Eric des Courtis Jan 31 at 22:36

11 And this, dear children, is why Java should stop taking drugs. – Griwes yesterday

5 You know you've been in the Java world too long when names like this one don't seem that unreasonable...! – Brian 17 hours ago

tagged

java	× 436752
spring	× 29557
persistence	× 2400
dao	× 887
jdo	× 824

asked 4 months ago
viewed 2337 times
active today

Community Bulletin

blog The War of the Closes

Motivation:

- It's 2013 and creating small/medium size dynamic web sites is still TOO HARD
- Identity and policy-based access control techniques have hardly changed
- Easy to slip up and give away the farm
- “Best practice” advice for OO frameworks is often to steer away from the most powerful available options

Inspiration:

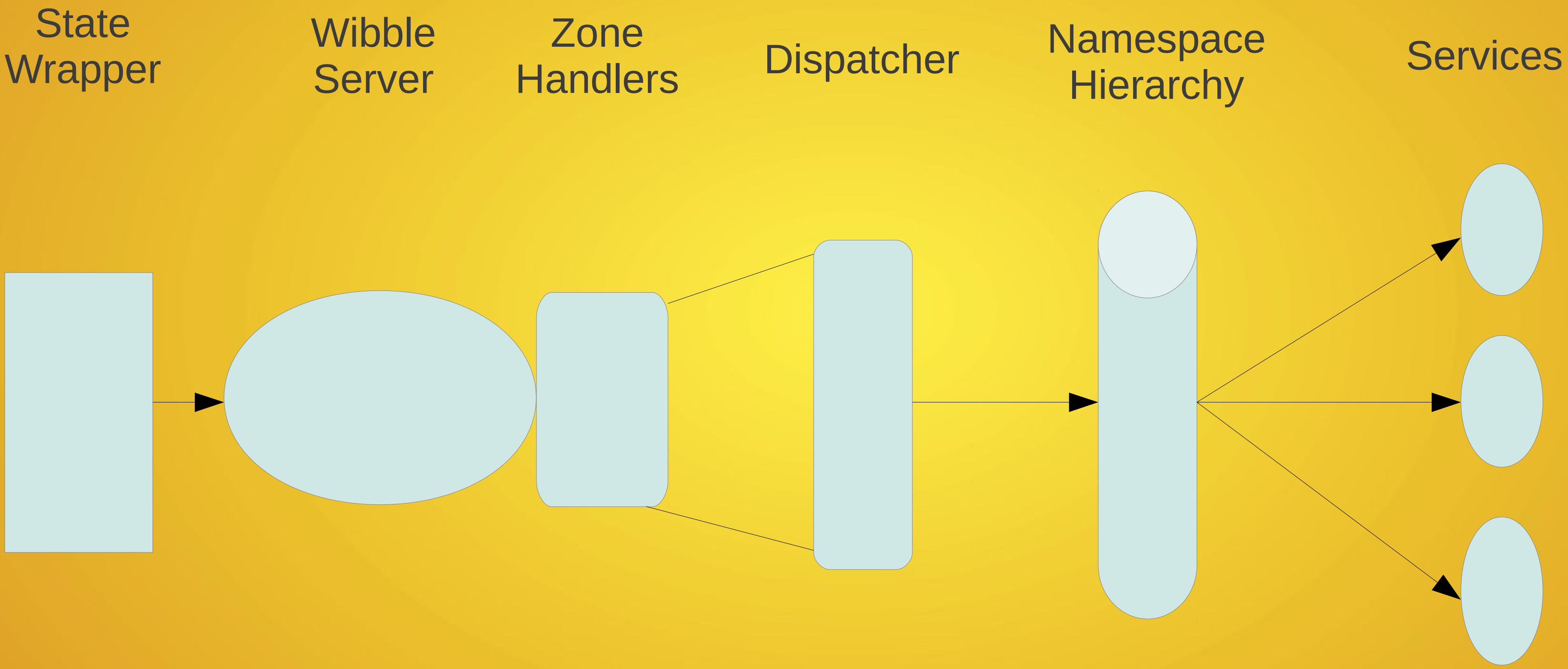
Service-oriented programming:

- “software modules are strictly encapsulated through well-defined service interfaces”
- “a service can be composed of other nested services in a hierarchical manner”
- Generalization of the “Unix way”
- No attempt here to adhere to any strict definition or protocol

Implementation:

- Little web framework in Tcl built on wibble
- Leverages unique Tcl features: hierarchical namespaces, call frame introspection, package modules
- Features delivered as set of packages

Overview:



State Wrapper

Wibble Server

Zone Handlers

Dispatcher

Namespace Hierarchy

Services

Package: pkgTree

- A URI is a hierarchically-structured service request
- Tcl namespaces are a hierarchically-structured tool for organizing code
- Tcl package modules are a hierarchically-structured tool for locating and loading procedure libraries
- A filesystem is a hierarchically-structured medium for storing files

Let's make them work together

pkgTree dispatcher

- Maps a URI path to a namespace ensemble
- Handles combination of REST-type arguments and query strings
- If suitable ensemble doesn't exist, looks in package module file space, loads candidate
- Access to arbitrary code prevented by requirement that candidate is an ensemble within defined namespace path

http://example.tk/document/statistics/wordcount/tutorial.txt

Or

http://example.tk/document/statistics/wordcount?doc=tutorial.txt

Maps to:

::API::document::statistics::wordcount $doc

Read from:

$lib/tm/API/document/statistics-0.1.tm

pkgTree Helper procs

- pkgTree::provide loads package in module file, is self-aware of filesystem location and creates corresponding namespace automatically
- pkgTree::public creates namespace ensemble and exports enumerated procs
- pkgTree::resource maps request maps requests for template files, images etc. to filesystem; is self-aware about namespace position, so only filename needed

Design follows theory of parallel service interfaces, with URI as index

$lib/tm/API/document/statistics-0.1.tm:

```
pkgTree::provide
pkgTree::public wordcount
pkgTree::add_service ::internal_service

proc wordcount {doc} {
    setState contenttype [mimestring [file extension $doc]]
    exec wc -w << [resource /$doc]
}
```

Enabling Workflow:

- Visibility of code to the dispatcher is controlled by function of **[namespace path]** and **[namespace which]** commands
- Visibility can be set dynamically on a per-connection basis
- Visitors can be required to qualify for access to code/resources
- Allowed namespace path can be part of persistent session state credentials, can be edited per visit
- Visitors can thus be moved through a state tree in successive transactions
- A table of namespace relations and transition conditions can function as an access policy, and turn the web site into a state machine

But how do you manage per-connection visitor state?

Wibble Router Table

- Wibble uses a table of “zone handlers” to match URI prefixes to code that will fulfill queries
- Key-value pairs of prefixes and procedure calls
- pkgTree dispatcher is called by a zone handler
- Matching handler doesn't have to return a result; has access to interrupts that allow passthrough to next handler
- Can configure “state” handlers that only set and read session credentials
- Can run multiple state handlers and dispatchers in succession

Access Control and Policy

- **[info frame]** is used to identify connection from anywhere in call frame
- Default connection state is read from server state
- Zone handlers manipulate connection state
- State determines visibility of services
- Service code has no knowledge of policy, if resource is not authorized it's simply invisible

Goals:

- Easy and quick to create dynamic web APIs (just drop in a package module)
- Easy to add internal complementary services (database, form validation)
- High confidence that resource leakage won't occur
- Easy to audit code for policy compliance via well-segregated discrete validation steps

Applications:

- User accounts
- Roles
- More fine-grained collaboration tools
- Workflow
- Virtual servers
- Abstract and adapt to policy-driven applications outside web context