

GCLASS

Transparent Persistence for Web Applications

GemStone • Linux • Apache • Seaside • Smalltalk

Version 1.0 alpha 1

GEMSTONE  64

- ▶ Another Implementation of Smalltalk
 - + ANSI-98 compliant
 - + Large image
 - + Database semantics – commit/abort
 - + Many virtual machines and hosts share *the same image*
 - Expensive (?)
 - Text-only user interface (?)

- ▶ GemStone “Web Edition”
 - No-cost license, even for commercial use!
 - Up to 4 GB image (repository) size
 - Up to 1 GB shared page cache
 - Up to 64 million objects
 - Unlimited VMs (gems)
 - 64-bit Linux on 64-bit AMD/Intel hardware
 - Uses only one CPU on one host
 - GemBuilder for Smalltalk is disabled
 - Community support

- ▶ Where were you during Esteban Lorenzano's presentation yesterday?
- ▶ An application framework that simplifies the development of complex, dynamic web applications.

- ▶ Why the interest?
 - Buzz from Ruby on Rails
 - Promote Smalltalk
 - “We can do better” – Alan Knight
 - (Same as Cincom and Instantiations ;-)

Why Port Seaside to GemStone/S?

- ▶ Other dialects of Smalltalk are single-user and non-persistent
 - This means that a Seaside application needs to work around built-in limitations to handle multi-user persistence
- ▶ GemStone's value has always been providing Smalltalk developers with:
 - Built-in transactional **persistence**
 - Built-in **multi-user** capability
 - Built-in multi-CPU and multi-machine **scalability**.
- ▶ GemStone's lack of a GUI is okay!

- ▶ Persistence approaches in Smalltalk
 - In the image
 - Loss of data if image quits
 - Not shared across images
 - In a binary file-out
 - Limited size
 - Object identity is not be preserved
 - In an external database
 - Object/relational mapping overhead
 - Extra coding to foreign interface
- ▶ GemStone/S solves this problem!

- ▶ Multi-user approaches in Smalltalk
 - One image serving multiple clients
 - Requires layer directing query to image
 - Scalability limit
 - Coordinate through external database
 - Object/relational mapping issues
 - Extra coding to foreign interface
- ▶ GemStone/S solves this problem!

- ▶ Scalability approaches in Smalltalk
 - Add hardware
 - Still basically single-threaded
 - Run more images
 - Presents all the persistent/multi-user issues
- ▶ GemStone/S solves this problem!

- ▶ Multiple VMs
 - Each is a separate OS process
 - Each has full access to the database
 - Close to linear scaling
- ▶ Multiple hosts
 - Customer production systems
 - 1500 VMs; 200 hosts
- ▶ Tested
 - 3000 VMs; 1 terabyte data; 16 billion objects

- ▶ The “official” version is in Squeak
 - <http://www.squeaksource.com/Seaside>
- ▶ Existing Seaside ports to
 - Dolphin
 - VisualWorks
- ▶ Typical porting process is somewhat complex
 - Export from Squeak
 - Import into other dialect

- ▶ Typical port process is awkward
 - Any changes must be made in Squeak
 - Wait for someone to update port
- ▶ GemStone has the “advantage” of no native source code control
- ▶ Source is in Monticello repository
 - File and HTTP interface (among others)
- ▶ Port Monticello to GemStone/S

- ▶ **Method namespaces**
 - Base-class additions and overrides
 - Support for hosted sandbox
- ▶ **Compiler changes**
 - Assignment
 - Pragmas
 - Array constructors
- ▶ **Also:**
 - Transient (non-persistent) objects
 - Selective rollback

- ▶ Most Seaside applications use a Smalltalk HTTP server (e.g., Kom)
- ▶ Large-scale applications will generally want a separate web server
 - Serve static pages
 - Handle SSL (https requests)
 - Load balancing
 - Fail-over backup
 - Security (hacker-resistant)
 - Separate server administration

- ▶ ReverseProxy
 - Forward selected requests using HTTP
- ▶ FastCGI
 - Forward selected requests using FastCGI
- ▶ Both solutions need similar interface
 - Framework listens on TCP/IP port
 - Convert request to Seaside format
 - Convert response from Seaside format

- ▶ Squeak is more familiar for most Seaside developers
- ▶ Many Seaside applications are already developed
- ▶ Port to GemStone/S when ready for deployment, or when you need to scale

- ▶ Client access is through a shared library (DLL on Windows)
 - GemStone C Interface (GCI)
- ▶ Use Squeak's Foreign Function Interface (FFI) to access GCI
 - Wrap C functions with Smalltalk methods

- ▶ Avoid Setup, Configuration, and Management of GemStone Server
- ▶ VMware “appliance”
 - VMware Server free for Linux, Windows
 - VMware Fusion for Macintosh (US\$60)
- ▶ We create a full Ubuntu machine
 - Linux pre-installed and configured
 - Seaside starts when OS boots
 - Squeak image installed for tools

- ▶ Avoid Setup, Configuration, and Management of GemStone Server
- ▶ Shared Server on Internet
 - Added security features to reduce risk
- ▶ Apply for an Account
 - <http://seaside.gemstone.com/>

- ▶ Tools
 - Monticello browser performance
- ▶ Appliance
 - Start-up
 - Maintenance (GC, backup, etc.)
- ▶ Sandbox
 - Security setup
- ▶ Feedback from beta users

▶ Humility

- We don't know Seaside (Apache, etc.) very well, but we recognize the pain of Object-Relational mapping.
- We have learned a lot about web application needs from you, and have varied our approach based on early feedback (number of VMs, rollback, etc.).

▶ Community

- We aren't trying to steer the direction, but to give you, the community, good tools.

- ▶ Documentation (such as it is ;-)
 - <http://seaside.gemstone.com/>
- ▶ Mailing Lists
 - <http://www.seaside.st/Community/MailingList/>
 - subscribe-gemstone-smalltalk@earth.lyris.net
- ▶ Email
 - James.Foster@GemStone.com