GLASS

Transparent Persistence for Web Applications

GemStone•Linux•Apache•Seaside•Smalltalk

Version1.0 alpha 1

GemStone/S™ 64

But first, a Word from our Sponsors…

- Some highlights since last year
  - TimeZone patch for 6.x
  - GemStone/S 64 Bit releases
    - 1.1.7 to 1.1.14
    - 2.0.4, 2.0.5
    - 2.1.0 to 2.1.5
    - 2.2.0 to 2.2.2 (August 2007)

GemStone 64 Bit Enhancements - 1

- Multi-threaded stone, shared page cache monitor
- Polymorphic method lookup caches
- Segment (security) support
- Larger signal buffer for gem-to-gem signals
- Linux supports faster performance using optimized POSIX Asynchronous I/O
- Signal when transaction logs are full
- Ten application write lock queue with timeout
- Locale and new extended character set support
- System class>>#logout now available
- Several float operations now in primitives

GemStone 64 Bit Enhancements - 2

- RcQueue improvements
  - Add time is now millisecond-based
  - Improved performance by preconfiguring max size
  - Multiple retries on commit conflict
- Reduced conflict equality indexes
- Added tranlog analysis abilities (auditability)
- DateTime now supports millisecond resolution
- Better handling of LostOT situation
- Collection optimizations
- Optimization of #and: and #or:
- Intel Macintosh client libraries (level “B” support)
- Logging enhancements (ms times, etc.)

Announcement of Free Edition

- 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 Intel hardware
  - Uses only one CPU on one host
  - GemBuilder for Smalltalk is disabled
  - Community support

What is Seaside and Why the Interest?

- What is Seaside?
  - Where were you during Lucas Renggli’s presentation yesterday?
- Why the interest?
  - Buzz from Ruby on Rails
  - Grow interest in 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!

Challenge of Persistence

- 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!

Challenge of Multi-User Coordination

- 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!

Challenge of Scalability

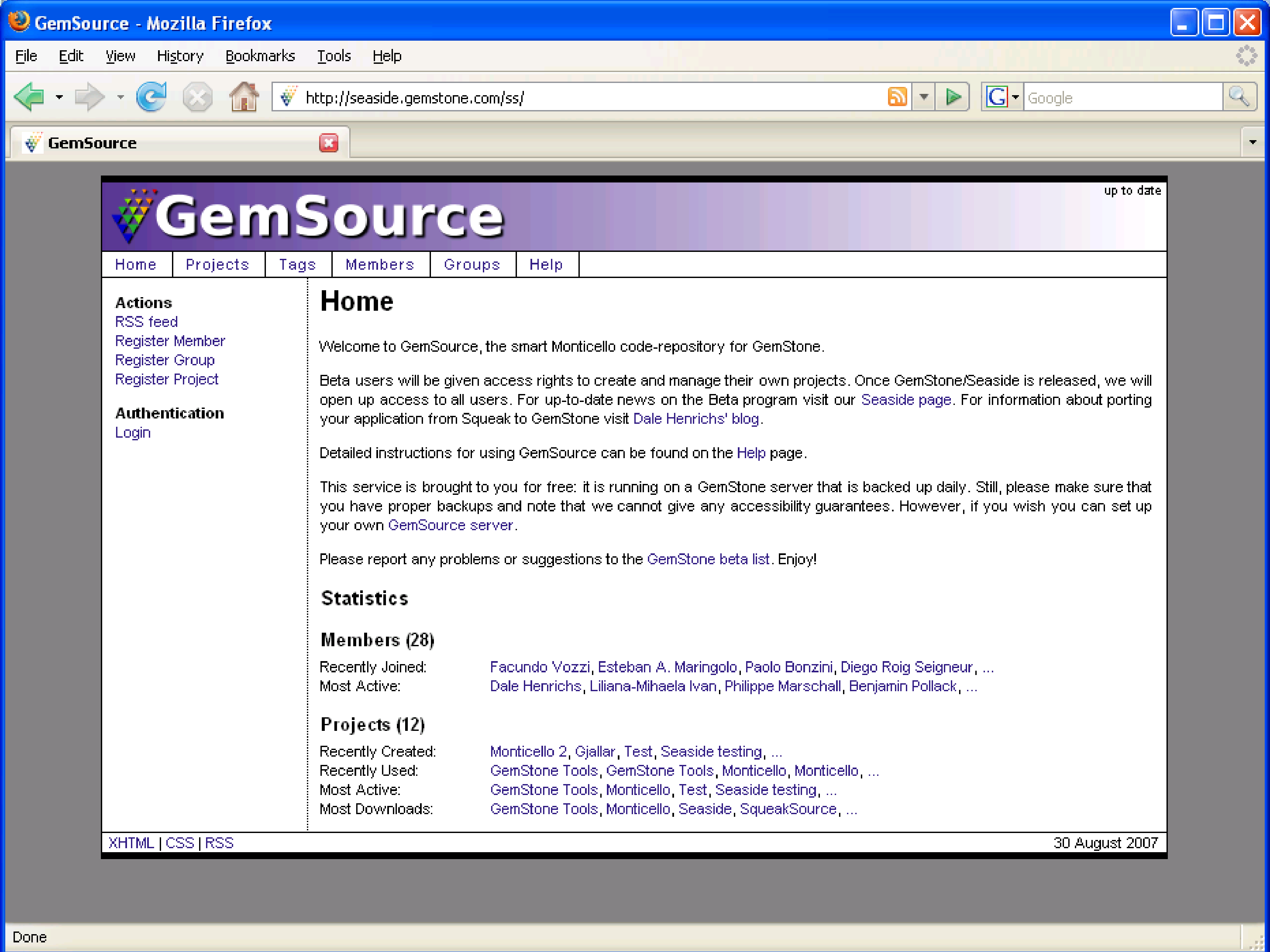
- Scalability approaches in Smalltalk
  - Add hardware
    - Still basically single-threaded
  - Run more images
    - Presents all the persistent/multi-user issues
- GemStone/S solves this problem!

Scaling

- 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

What’s Done?

- Code in GemStone Smalltalk
  - HTTP Server (Hyper) and FastCGI Server
  - Monticello (File & HTTP)
  - Seaside 2.6 & Seaside 2.8
  - SqueakSource (in 2.6)
- Squeak-based tools
  - GS/S login & workspace
  - View & edit GS/S code & objects
    - OmniBrowser, OB Standard, OB Tools



GemSource

Actions
RSS feed
Register Member
Register Group
Register Project

Authentication
Login

Home

Welcome to GemSource, the smart Monticello code-repository for GemStone.

Beta users will be given access rights to create and manage their own projects. Once GemStone/Seaside is released, we will open up access to all users. For up-to-date news on the Beta program visit our Seaside page. For information about porting your application from Squeak to GemStone visit Dale Henrichs' blog.

Detailed instructions for using GemSource can be found on the Help page.

This service is brought to you for free: it is running on a GemStone server that is backed up daily. Still, please make sure that you have proper backups and note that we cannot give any accessibility guarantees. However, if you wish you can set up your own GemSource server.

Please report any problems or suggestions to the GemStone beta list. Enjoy!

Statistics

Members (28)

Recently Joined:	Facundo Vozzi, Esteban A. Maringolo, Paolo Bonzini, Diego Roig Seigneur, ...
Most Active:	Dale Henrichs, Liliana-Mihaela Ivan, Philippe Marschall, Benjamin Pollack, ...

Projects (12)

Recently Created:	Monticello 2, Gjallar, Test, Seaside testing, ...
Recently Used:	GemStone Tools, GemStone Tools, Monticello, Monticello, ...
Most Active:	GemStone Tools, Monticello, Test, Seaside testing, ...
Most Downloads:	GemStone Tools, Monticello, Seaside, SqueakSource, ...

XHTML | CSS | RSS

30 August 2007

Where Do We Get Seaside?

- 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

Get Seaside Directly from the Source!

- 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

Technical Challenges and Decisions

- Method namespaces
  - Base-class additions and overrides
  - Support for hosted sandbox
- Compiler changes for assignment
- Transient (non-persistent) objects
- Selective rollback

Aside on Namespaces

- Namespace problem:
  - Typically, isolating code from each vendor
  - Complexities of multi-user image
    - Each user can load different packages
    - A package can add/replace methods for classes in another package
    - What if other class/method is shared?
- Prior namespace not adequate
  - Need to isolate each user (session)
  - Add ‘Session Methods’ feature to GS/S

Session Methods-1

- Traditional method lookup:
  - Array then…
  - SequenceableCollection then…
  - Collection then…
  - Object then…
  - MessageNotUnderstood exception
- Dictionary subclass: SessionMethods
  - Keys: Class
  - Values: MethodDictionary instances

Session Methods-2

- New method search:
    - SessionMethods for Array
  - Array
    - SessionMethods for SequenceableCollection
  - SequenceableCollection
    - SessionMethods for Collection
  - Collection
    - SessionMethods for Object
  - Object
- Method is cached after first lookup

Namespace Proposal

- Dictionary subclass: Environment
  - Keys: Class
  - Values: MethodDictionary instances
- Each method, when compiled, can be associated with an Environment
  - Default to class’s Environment
  - Class’s Environment defaults to System
- Use Environment from method in place of SessionMethods from login

Make GemStone/S Accessible to Squeak Developers

- GemStone/S compiler was modified to allow underbar as the assignment statement:

```
url _ 'http://seaside.gemstone.com/'.
```

- This allows code to be loaded directly
  - No translation needed
- This allows use of Monticello tools to compare versions w/o clutter
  - Export back to Monticello and load in Squeak

New DbTransient Attribute

- A class can now have the new attribute DbTransient.
  - This is designed to be similar to Java “transient” variable attribute.
  - The instance variables of instances of a class that is DbTransient are not committed, but remain local to the session.
  - This allows you to reference objects that should not be persistent - such as semaphores - within data structures that are persistent and shared.
  - Typical pattern is to wrap such objects.

Multiple Dirty Lists - 1

- “Extreme Validation” (Leandro Caniglia)
  - Don’t rely exclusively on presentation layer for validation
  - Check domain objects for validity
  - Rollback domain objects if invalid
- Problem is that not all work should be discarded
  - Existing continuations need to be saved

Multiple Dirty Lists - 2

- Create separate buckets for objects
  - Seaside framework objects go in one bucket
  - Application domain objects go in another bucket
  - Objects in one bucket can be rolled back without losing data from objects in the other bucket
  - Not really “nested transactions,” but another solution that works here

How Does Apache Fit In?

- 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

Apache Forwards Requests

- ReverseProxy
  - Forward selected requests using HTTP
- FastCGI
  - Forward selected requests using FastCGI

Develop in Squeak, Deploy in GemStone

- 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

Port Process - 1

- Fix compile errors on load
  - Export application packages from Squeak to files or HTTP repository
  - Load packages into GemStone
  - Identify compile errors—typically {…}
  - Fix in Squeak image
  - Repeat until no compile errors

Port Process - 2

- Fix initialization errors on load
  - Export application packages from Squeak to files or HTTP repository
  - Load packages into GemStone
  - Identify initialization errors
    - References to missing classes or methods
  - Fix
    - Add missing classes (load more packages)
    - Add platform-specific or general-use methods
  - Repeat until no initialization errors

Port Process - 3

- Compare image to filed-out package
  - Should be no differences after load
  - Any differences are due to things that didn't load
  - Fix any differences
  - Repeat till no differences

Port Process - 4

- Fix Undeclared Globals
  - Don’t wait for a walkback
  - SymbolDictionary of undeclared globals
    - Key: Global name (referenced but not found)
    - Value: Set of Associations identifying methods
      - Key: Class with referencing method
      - Value: Selector for referencing method
  - Add missing classes (load more packages)
  - Repeat till no more undeclared globals

Port Process - 5

- Run Unit Tests
  - You do have tests don’t you?
  - Run application and verify operation
  - Fixes can be exported from GemStone/S and loaded into Squeak!

Jade: GemStone/S Tools in Squeak - 1

- 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

Jade: GemStone/S Tools in Squeak - 2

- User Interface available
  - Login
  - Workspace
- OmniBrowser
- Monticello Browsers
  - Thanks to Liliana Ivan for initial port

GLASS Appliance

- 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

Hosted Sandbox

- 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/

What’s Left?

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

Lessons Learned

- 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.

Question?

- 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
  - Dale.Henrichs@GemStone.com