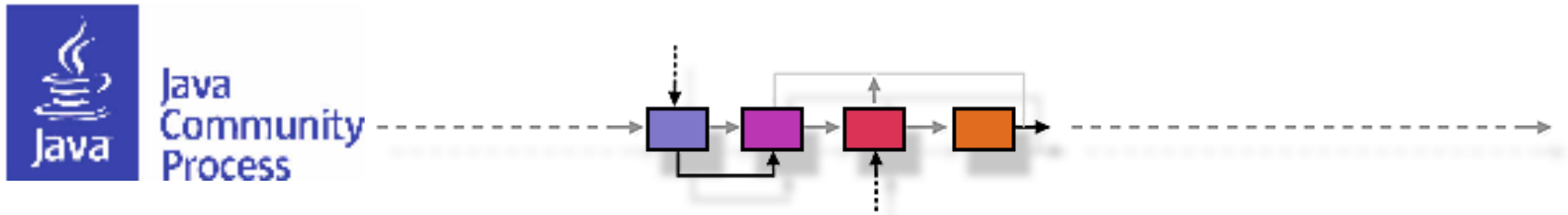


JSR 107 Review Process

3 September 2013

Greg Luck
Brian Oliver, Oracle America

Agenda

- Goals
- Information to be gathered
- Implementation notes
- Issues
- Questions, discussion, next steps

Information to be gathered

Scope

- **The Java Caching API provides a common way for Java programs to create, access, update and remove entries from caches.**

Objectives

The Java Caching API's objectives are to:

- provide applications with caching functionality, in particular the ability to cache Java objects;
- define a common set of caching concepts and facilities;
- minimize the number of concepts Java developers need to learn to adopt caching;
- maximize the portability of applications that use caching between caching implementations;
- support both in-process and distributed cache implementations;
- support caching Java objects by-value and optionally by-reference;
- define runtime cache annotations in accordance with JSR-175: A Metadata Facility for the Java Programming Language; so that Java developers making use of optionally provided annotation processors may declaratively specify application caching requirements; and

Introduction - Background

Open source caching projects and commercial caching vendors have been out there for over a decade. Each vendor uses a very similar map-like API for basic storage and retrieval yet each vendor uses their own API.

Java Community Process

Introduction - Target Platform

The Java Caching API is designed to be suitable for use by applications using the Standard and Enterprise Editions, versions 6 or newer.

A caching implementation:

- may choose to only work on a higher version of Java.
- may support its use by applications using Java EE, however this specification does not specify any standard for how that may be done.

Business/marketing/ecosystem justification

Why do this JSR?

- To standardise an important part of Java.

What's the need?

- With JSR107 specification developers can program to a standard API, instead of being tied to a single vendor, eliminating a major inhibitor to the mass adoption of in-memory technology.

- Indeed, the analyst firm Gartner reported last year that the lack of a standard in this area was the single biggest inhibitor to mass adoption*.

Business/marketing/ecosystem justification

How does it fit in to the Java ecosystem?

- Caching is used widely in production systems. Gartner has described it with the terms “Distributed Caching” and “In-Memory Data Grid”. Forrester uses the term “Elastic Caching”*.
- There are a wide range of commercial implementations in a market expected to reach $1bn by 2015*.
- By 2014 at least 40% of large organizations will have one or more advanced in-memory data grids*
- Will be added to Spring 4 in 2014 and will likely be added to Java EE8 via a maintenance release of JSR107.

*Source: In-memory Data Grid: Key Technology for 21st Century Computing, Massimo Pezzini, Gartner, 14 June 2012

Business/marketing/ecosystem justification

Is the idea ready for standardization?

- There are many commercial products: BigMemory, Coherence, Infinispan, Hazelcast, GridGain, Gemfire, Extreme Scale , ActiveSpaces and many open source ones.
- In particular, Ehcache has been very widely used for over a decade, with an estimated 2 million developers who have used it
- It is therefore possible to find common ground amongst implementations and standardise common features and the main map-like API.

History

Phase	Start	Finish
Public Review Ballot	27 Aug, 2013	09 Sep, 2013
Public Review	05 Jul, 2013	26 Aug, 2013
Early Draft Review	23 Oct, 2012	22 Nov, 2012
Expert Group Formation	20 Mar, 2001	04 Apr, 2001
JSR Review Ballot	06 Mar, 2001	19 Mar, 2001

Technical scope and features

The fundamental concepts are there is a CacheManager that holds and controls a collection of Caches. Those Caches in turn have entries with keys and values.

The API can be thought of map-like with the following additional features:

- atomic operations, similar to java.util.ConcurrentMap
- read-through caching
- write-through caching
- cache event listeners
- statistics
- caching annotations
- full generics API for compile time safety
- storage by reference (applicable to on heap caches only) and storage by value

.

Technical scope and features – cont.

In addition there is one optional feature - that not all implementations might provide:

- storeByReference - storeByValue is the default

Whether they are present can be determined with a call to the capabilities API: CachingProvider.isSupported.

The Expert Group - Members

- Greg Luck
- Brian Oliver, Oracle
- Cameron Purdy, Oracle
- Manik Surtani, Red Hat, Inc.
- Nikita Ivanov, Grid Gain
- Chris Berry
- Jon Stevens
- Rick Hightower
- Ben Cotton, Credit Suisse
- David Mossakowski, Citigroup
- Bongjae Chang
- Steve Millidge
- Gabe Montero, IBM
- Brian Martin, IBM
- Eric Dalquist
- Pete Muir, Red Hat, Inc.
- William Newport, Goldman Sachs
- Ryan Gardner, Dealer.com
- Chris Dennis, Terracotta, Inc.

Java Community Process

The Expert Group

How diverse is the EG? Is it representative of the ecosystem?

- Open source projects represented: Greg Luck – Ehcache, Manik Surtani - InfiniSpan
- Wall Street represented: David Mossakowski - CitiGroup, Ben Cotton - Credit Suiss, Billy Newport – Goldman Sachs
- Caching Annotations/DI community represented: Eric Dalquist and Rick Hightower, Pete Muir
- Most commercial vendors on EG: Oracle, IBM, Terracotta, GridGain
- Original formation members still present: Chris Berry, Jon Stevens

The Expert Group

How does the EG operate? How often does it meet and how (teleconferences, online, f2f?):

- New topics are raised as GitHub issues. Discussion occurs there along with resolution
- Votes are done by the mailing list.
- The spec leads are in regular contact via skype video call
- At JavaOne each year there has been a get together

What collaboration tools are used to facilitate EG communications?

- GoogleGroups Forum and Mailing List: jsr107@googlegroups.com
- GitHub for home page, code repository and issues:
  - https://github.com/jsr107/jsr107spec
  - https://github.com/jsr107/jsr107spec/issues

Java Community Process

Other deliverables

- Other than Spec, RI, and TCK are you delivering, for example:
  - Additional documentation?
    No. Each implementation will do that.
  - User's guide?
    No. Each implementation will do that.
  - Sample code?
    Yes. Many examples are provided in our demo repository. See https://github.com/jsr107/demo
  - FAQ?
    No.
  - Other artifacts?
    Jars, JavaDoc and sources for each module distributed via Maven Central under the group javax.cache

Java Community Process

Publicity

Java Magazine

http://www.oraclejavamagazine-digital.com/javamagazine_open/20130102#pg59

Blog – gregluck.com

e.g.
http://gregluck.com/blog/archives/2013/07/jsr107-jcache-enters-public-draft-review/

Conferences

JavaOne Sessions on JSR107: 2011, 2012, 2013

JFocus, DevNexus

Java Community Process

Collaboration with other community groups

Are you working with other community groups or organizations?

- Spring: they will incorporate in Spring 4. We provide a Spring integration example implementation.
- Guice: we provide a Guice integration example implementation.

Implementations

How many implementations (apart from the RI) exist?

- Oracle Coherence – in development for V1 of the spec
- Terracotta BigMemory – in development. To be announced for early 2014 (press release to go out at JavaOne)
- Ehcache – up to 0.4 of the spec
- JBoss InfiniSpan – up to 0.8 of the spec
- Others expected including one from each organisation on the expert group once the spec is released

Schedule

Phase	Start	Finish
Final Release	28 Nov 2013	12 Dec 2013
Final Approval Ballot	14 Nov 2013	28 Nov 2013
Update the Deliverables post AB	7 Nov 2013	14 Nov 2013
7 Day Appeal Ballot	31 Oct 2013	7 Nov 2013
Est. First Level TCK Appeals Process	-	31 Oct, 2013
Complete the RI & TCK	-	31 Oct, 2013
Proposed Final Draft	-	30 Sep, 2013
Public Review Ballot	27 Aug, 2013	09 Sep, 2013
Public Review	05 Jul, 2013	26 Aug, 2013
Early Draft Review	23 Oct, 2012	22 Nov, 2012
Expert Group Formation	20 Mar, 2001	04 Apr, 2001
JSR Review Ballot	06 Mar, 2001	19 Mar, 2001

Legend
Actual in black Projected in blue

Java Community Process

IP flow

Provide pointers to the licenses for the the Spec, RI, and TCK.

Specification – usual specification license

RI – Apache 2 and CDDL

TCK – custom license

See http://jcp.org/en/jsr/detail?id=107

IP flow

How are you handling contributions from non JCP members?

- Spec, RI and TCK. Direct contributions prohibited. We take comments but do not accept code or literal text.

What Terms of Use apply to your collaboration tools?

- Free to Use

Do you have a Contributor Agreement?

Yes, any code contributions to the RI and TCK are only accepted if the contributor has signed a Terracotta Contributor Agreement. We have these on file.

Any legal issues or concerns?

- Yes. Software AG believes that the TCK license cost per “marketed product” is counter to the promotion of open standards. For their own part, Software AG will not be using JSR107 in internal components as it would cost almost $1m per year to do so. This issue however is not unique to JSR107 and applies to all JSRs used by Software AG. Software AG are in the process of removing all internal usage of JSRs.

RI and TCK development

- How are you developing the RI and TCK?
- If collaboratively (through an open-source project)
  - How many committers and who?
  - How many apart from the Spec Lead (organization)?
- Is the RI available for public download? (If so, provide URL.)
- Is the TCK available for public download? (If so, provide URL.)
- Do you have a source-code repository? (If so, provide URL.)

RI and TCK development

How are you developing the RI and TCK?

- By the spec leads and OSS contributors who have signed the Terracotta Contributor Agreement.
- If collaboratively (through an open-source project)
  - How many committers and who?

    See https://github.com/jsr107/RI/graphs/contributors.

    8 contributors
  - How many apart from the Spec Lead (organization)?

    4
- Is the RI available for public download? (If so, provide URL.)
- Yes. https://github.com/jsr107/RI

Java Community Process

RI and TCK development

- Is the TCK available for public download? (If so, provide URL.)

Yes, so far. https://github.com/jsr107/jsr107tck. It is possible that the final TCK not be publicly downloadable to comply with TCK licensing terms.

Do you have a source-code repository? (If so, provide URL.)

Yes. https://github.com/jsr107/

Participation and transparency

- Provide a pointer to the JSR page on JCP.org
  - http://jcp.org/en/jsr/detail?id=107[
- Provide a pointer to the “JSR project website" (eg, on Java.net.)
  - https://github.com/jsr107/jsr107spec

Java Community Process

Adopt-a-JSR

- Are you participating in the Adopt-a-JSR program?
  Yes. Houston JUG have adopted us.
- If so, what kind of contributions/help have you received and from whom/which JUGs?
  - We are finalising the adoption request.

Mailing lists or forums

How are you communicating with the public and how can they communicate with you?

- Either https://github.com/jsr107/jsr107spec/issues or jsr107@googlegroups.com

Provide pointers to public mailing list(s) and/or forum(s)

- jsr107@googlegroups.com is both a mailing list and forum https://groups.google.com/forum/#!forum/jsr107

Total number of messages, threads?

- Forum: 236 topics, 122 members, est. 2000 posts
  GitHub Issues: 235 issues, 94 stargazers, est. 1500 posts
- Total number of participants (EG members, non-EG members?)
- How many messages per month (from Spec Lead, EG

Mailing lists or forums

- Total number of participants (EG members, non-EG members?)
  Forum: 122 members
  GitHub Issues: 94 stargazers

  Almost all are non EG members.

Mailing lists or forums

How many messages per month (from Spec Lead, EG members, and non-EG members?)

All time stats are:

- Greg Luck (167) (spec lead)
- Yannis (125) (former spec lead)
- Ben.Cotton=jpmorgan.com (99)
- RickHightower (98)
- manik%su...@gtempaccount.com (92)
- Pete Muir (73)
- Brian Oliver (56) (spec lead)
- Manik Surtani (47)
- eric.d...@gmail.com (45)
- Steve Millidge (24)

Issue tracker

Total number of issues? @ 3 September 2013

- 235
- How many in each state (open, closed, deferred, etc?)
  226 closed, 9 open

Average number of issues logged per month?

- 20

Average number of issues resolved per month?

- 20 ie keeping up

How many different people logged them?

- 30 issue watchers.

How does this break down between Spec Lead, EG members, and non-EG members?

- 5 of the watchers are from Oracle or Terracotta. 25 watchers are not.

Document archive

Provide a pointer to your document archive.

- https://docs.google.com/document/d/1MZQstO9GJo_MUMy5iD5sxCrstunnQ1f85ekCng8LcqM/edit?usp=sharing

  Google Drive docs has a record of every change made since the start of the document.

Are meeting minutes and materials published?

- No. 2-3 meetings per week between spec leads makes this impractical. Every issue is created on GitHub and significant issues requiring a vote are circulated to the mailing list.

Document archive

What other materials are available for download?

- Maven Central artifacts
- Spec on Google Docs is findable and viewable by anyone

Everything else at GitHub: https://github.com/jsr107/

Total number of files available for download?

- Hundreds of files on GitHub and on Maven.

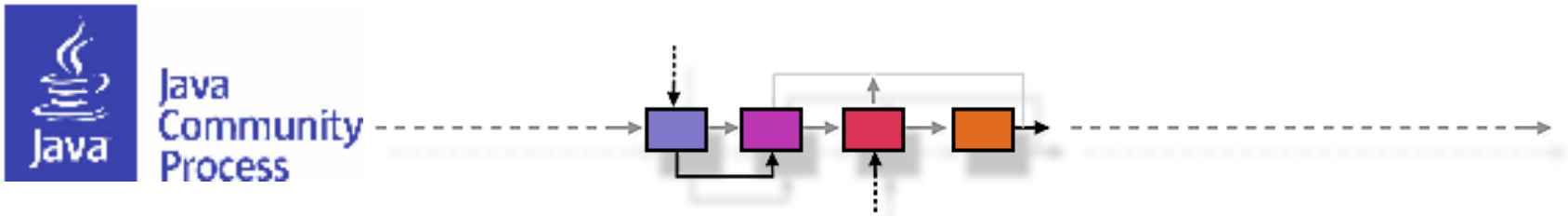
Average number of new files available for for download each month?

10

Other transparency and participation metrics

Provide information – including metrics – about any additional transparency and participation mechanisms you use.

All mechanisms discussed in previous slides



Thank you!
http://jcp.org