1.– 4. September 2014
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Blick in die Glaskugel

Die Oracle Glassfish Strategie und wie es mit Java EE 8 weitergeht

Wolfgang Weigend
ORACLE Deutschland B.V. & Co. KG

ORACLE

Blick in die Glaskugel

Die Oracle GlassFish-Strategie und wie es mit Java EE 8 weitergeht

Wolfgang Weigend
Java Technologie und Architektur

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

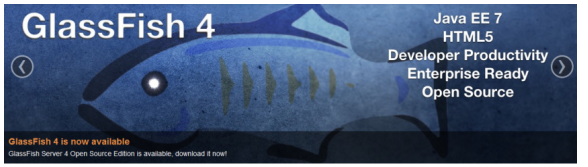
What happened

GlassFish Server commercial product being discontinued

- 4th of November 2013 we announced that the last release of the commercially supported version of GlassFish was Oracle GlassFish Server 3.1.2
- The open source community version will continue to be released, and will continue to serve as the RI for the Java EE platform
- We encourage all current GlassFish customers to migrate to the version of Oracle WebLogic Server that best fits their needs
- Oracle remains committed to Java EE

GlassFish Server Java EE Platform Strategy

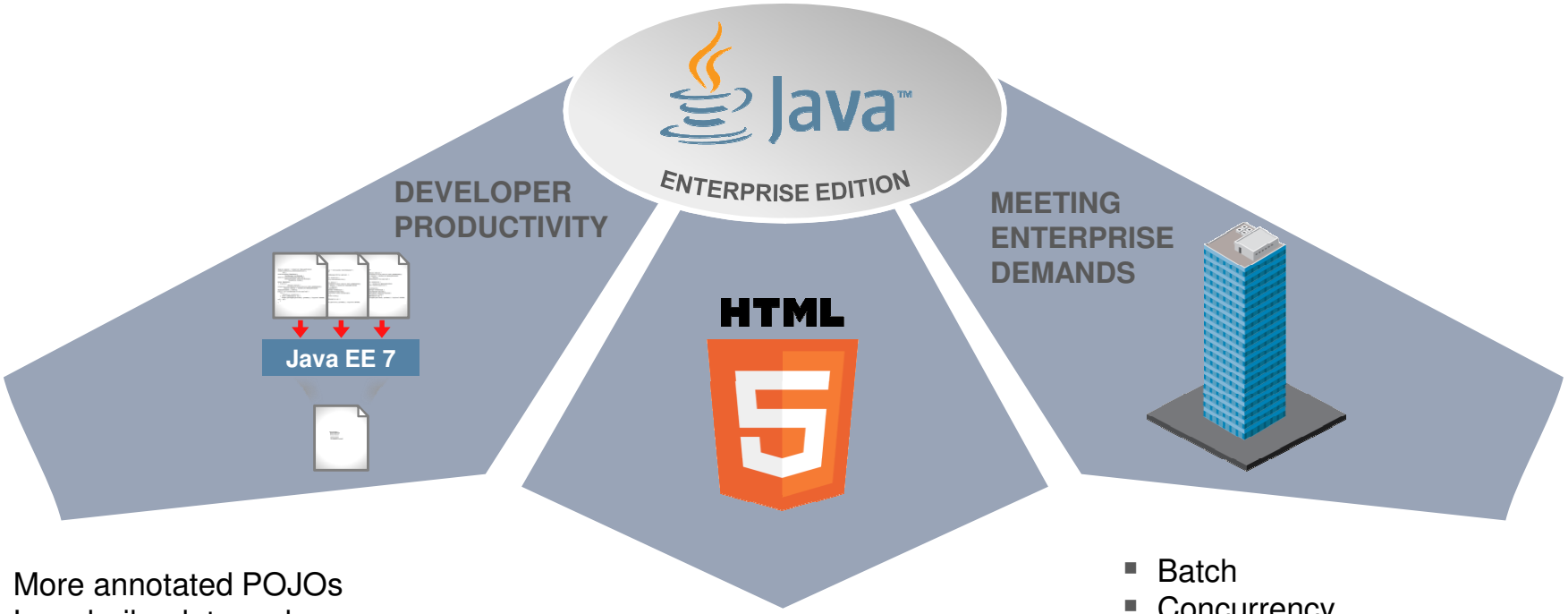
- GlassFish Server strategy is to drive adoption and development of the Java EE standard
- GlassFish (Java EE) is developed in open source and delivered in open source
- Latest version of GlassFish delivered Java EE 7
  - Released globally in June 2013 to wide acclaim by open source and commercial community
- Oracle's investment in GlassFish represents it's Java platform stewardship for Java EE
  - Similar to OpenJDK investment for Java SE



Java EE 7 Full Platform Compatible Implementations

GlassFish » Community

Java EE 7 Themes



- More annotated POJOs
- Less boilerplate code
- Cohesive integrated platform

- WebSockets
- JSON
- Servlet 3.1 NIO
- REST

- Batch
- Concurrency
- Simplified JMS

GlassFish Product Distributions

- Java EE Reference Implementation (RI)
  - Java Community Process requires specification lead to deliver a licensable implementation. Many licensees; 20+ Java EE implementations.
  - Java EE RI is a substantial subset of GlassFish
- Java EE SDK
  - Tutorial, samples and documentation for developers learning Java EE
- GlassFish Server Open Source Edition
  - Free open source, unsupported server deployment of Java EE applications
- Oracle GlassFish Server
  - Commercially supported deployment platform for Java EE / GlassFish

ORACLE

GlassFish Server Product Strategy

Java EE SDK and Java EE RI – Partners, Java EE Licensees Primarily

- Java EE RI
  - Java Community Process requires specification lead to deliver a licensable implementation
  - Released with each major update of the Java EE standard
  - Plan: Continue with future releases of Java EE
- Java EE SDK
  - Tutorials, samples and documentation for developers learning Java EE
  - Released with each major update of the Java EE standard
  - Plan: Continue with future releases of Java EE

GlassFish Server Product Strategy

GlassFish Server Open Source Edition

- GlassFish Server Open Source Edition
  - Developed in open source
  - Delivered in open source
  - Not commercially supported
- GlassFish Server Open Source Edition release plans
  - **GlassFish Server Open Source Edition 4.1 planned for 2014**
  - Additional updates as Java EE specification evolves
  - Plan: Regular patch updates will be delivered as needed
- GlassFish Server Open Source Edition Major Release
  - Delivered coincident with new versions of Java EE Platform

GlassFish Server Product Strategy

Oracle GlassFish Server Commercial Edition

- Oracle GlassFish Server 3.1.2.2 is last public production release of Oracle GlassFish with commercial support
  - Support timelines March 2016 Premier Support; March 2019 Extended Support
  - Available on Oracle pricelist and supported per the Premier and Extended Support timelines
  - Plan: No future releases beyond GlassFish 3.1.x
- Ongoing commercial support of future Java EE releases will be delivered from Oracle's strategic application server: WebLogic Server
  - WebLogic Server consistently adopting Java EE standards in a timely fashion
  - Java EE 6 in December 2011
  - **Full Java EE 7 certification planned for 2014/2015**
  - Goal - simplify decision for customers and enables investment in one server for high value development, deployment and management capabilities

Oracle GlassFish Server Lifetime Support Policy

On Existing Commercial Releases, Customers Continue to be Supported

Release	GA Date	Premier Support Ends	Extended Support Ends	Sustaining Support Ends
Sun GlassFish Enterprise Server 2.1.1	Jan 2009	Jan 2014	Jan 2017	Indefinite
Oracle GlassFish Server 3.0.x	Dec 2009	Dec 2014	Dec 2017	Indefinite
Oracle GlassFish Server 3.1.x	March 2011	March 2016	March 2019	Indefinite

Customers can continue to purchase incremental licenses of GlassFish Server and deploy within these support timelines

License Migration to WebLogic Server

- Customers can license migrate from Oracle GlassFish Server to Oracle WebLogic Server
  - Net to net license migration provided
- Choices
  - Oracle GlassFish Server to Oracle WebLogic Server SE
  - Oracle GlassFish Server to Oracle WebLogic Server EE
  - Oracle GlassFish Server to Oracle WebLogic Suite
- Customers should work with their Oracle account team for more specific details specific to their situation

Technical Migration to WebLogic Server

- WebLogic Server is certified Java EE compatible
  - Every release of WebLogic Server since Java EE created as a standard
- Oracle is investing to leverage shared components from GlassFish into WebLogic Server for additional ongoing GlassFish compatibility
  - WebLogic 12.1.2: JPA, JAX-WS, JAXB, WS-AT, CDI, JAX-RS, WebSockets, Bean Validation
  - WebLogic 12.1.3: JPA 2.1, JAX-RS 2.0 WebSockets, JSON Processing
- Java EE compatible applications from GlassFish can be deployed on WebLogic Server
  - WebLogic Server provides deployment descriptor compatibility for proprietary Oracle GlassFish Server deployment descriptors for easy application migration
  - Possible to develop and test on GlassFish and deploy on WebLogic Server

Summary

- Oracle committed to the future of Java EE
  - Delivered the most exciting of Java EE ever with Java EE 7 in June 2013
- GlassFish Server is the strategic reference implementation of Java EE
- GlassFish distributions for Java EE continue to be regularly updated with major releases of the Java EE specification
  - GlassFish SDK
  - GlassFish RI
  - GlassFish Open Source Edition
- Oracle provides one commercially supported strategic application server – Oracle WebLogic Server

Third Party Commercial Support for GlassFish by LodgON

- LodgON has been using GlassFish from first day, since Open-Sourced GlassFish at JavaOne 2005

  “Often, early adopters of new Java Enterprise features run into issues. The best way to understand a technology, and to fix issues, is to look at the source code. That is what we do with GlassFish. We run most of our Enterprise projects on GlassFish. Whenever we encounter an issue, strange behavior, or a performance problem, we dive into the code and find the cause of the problem. In case the solution requires a code change in GlassFish, we contribute code to GlassFish.”

ORACLE

Quelle: http://www.adam-bien.com/roller/abien/entry/commercial_glassfish_support_is_back

Third Party Commercial Support for GlassFish by LodgON

- *What is LodgON Commercial GlassFish support?*
  - The different parts of code that together are bundles as the GlassFish server are all Open Source. Updates to this source code are freely available as well. Managing different versions of the different components can be cumbersome, though. Depending on the needs of your company, we provide different options:
    - We can help with the installation, configuration, tuning and operations of the GlassFish Application Server
    - We detect and analyze performance issues. Often, it is not clear whether a performance issue is due to a bottleneck in the Application or in the Application Server. We detect performance issues and either fix them or provide tips to fix them. In some cases, this involves making changes to the Open Source components that constitute GlassFish, and we will contribute those changes back to the GlassFish community
    - We can help with the implementation of Java EE 7 applications

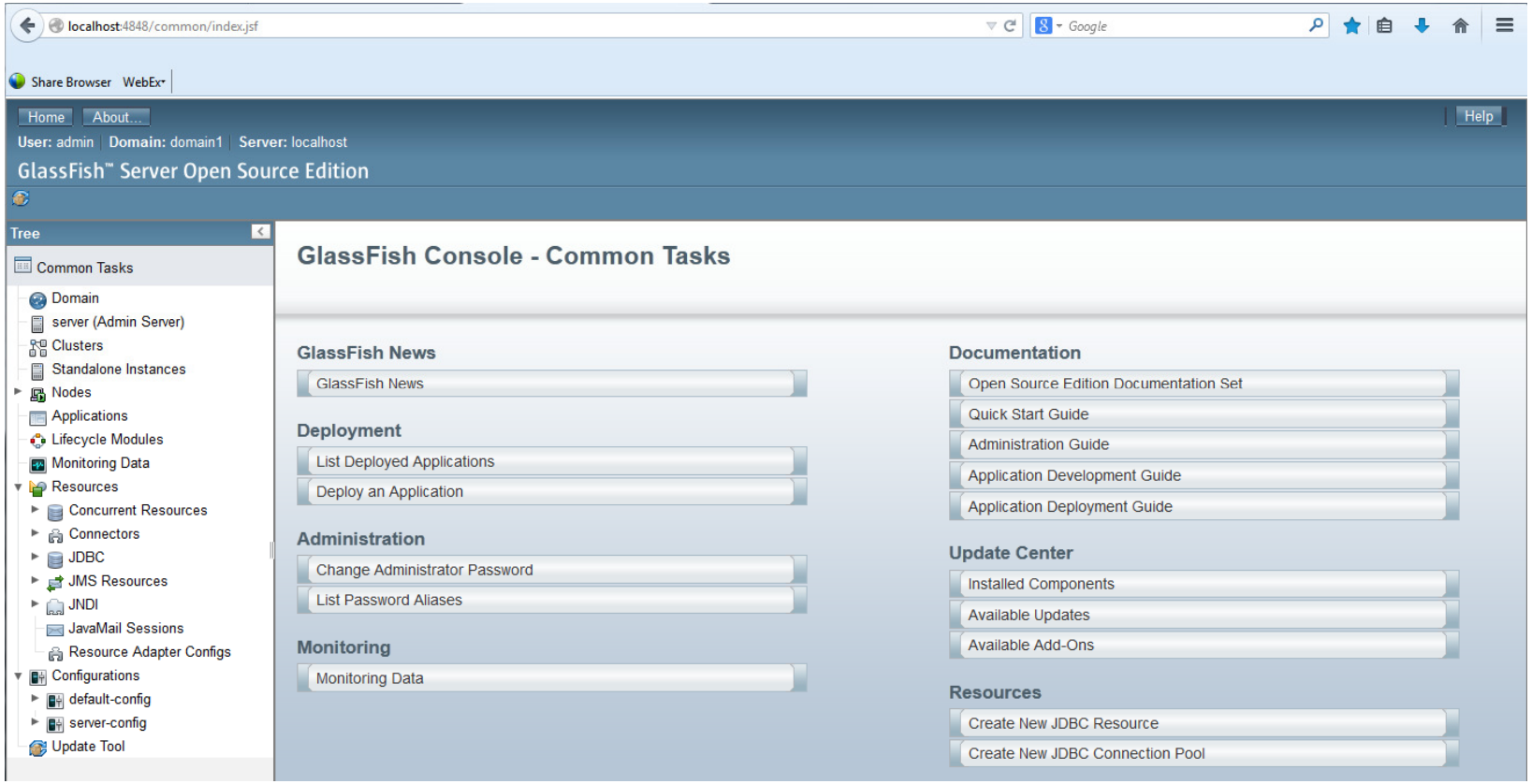
Quelle: http://www.adam-bien.com/roller/abien/entry/commercial_glassfish_support_is_back

GlassFish 4.1 next update

- Support for Java SE 8
- Incorporate bugs fixes addressed since GF 4.0, see the list of fixed bugs (recent fixes might not yet be incorporated into the Nightly Build)
- Update embedded sub-projects like Jersey, Tyrus, Weld, Mojarra, JavaDB, etc.
- NetBeans 8.0.1 alignment
- **Download the latest GlassFish 4.1 Nightly Build**
  - Web Profile latest-web.zip (69 MB)
  - Full Java EE Platform latest-glassfish.zip (118 MB)
  - http://dlc.sun.com.edgesuite.net/glassfish/4.1/nightly/

- **Clustering**
- **Improve modularization**
- **GlassFish Server Control**
- **Oracle GlassFish Server**
- **Bug fixes**

GlassFish Server Open Source Edition 4.1 b13

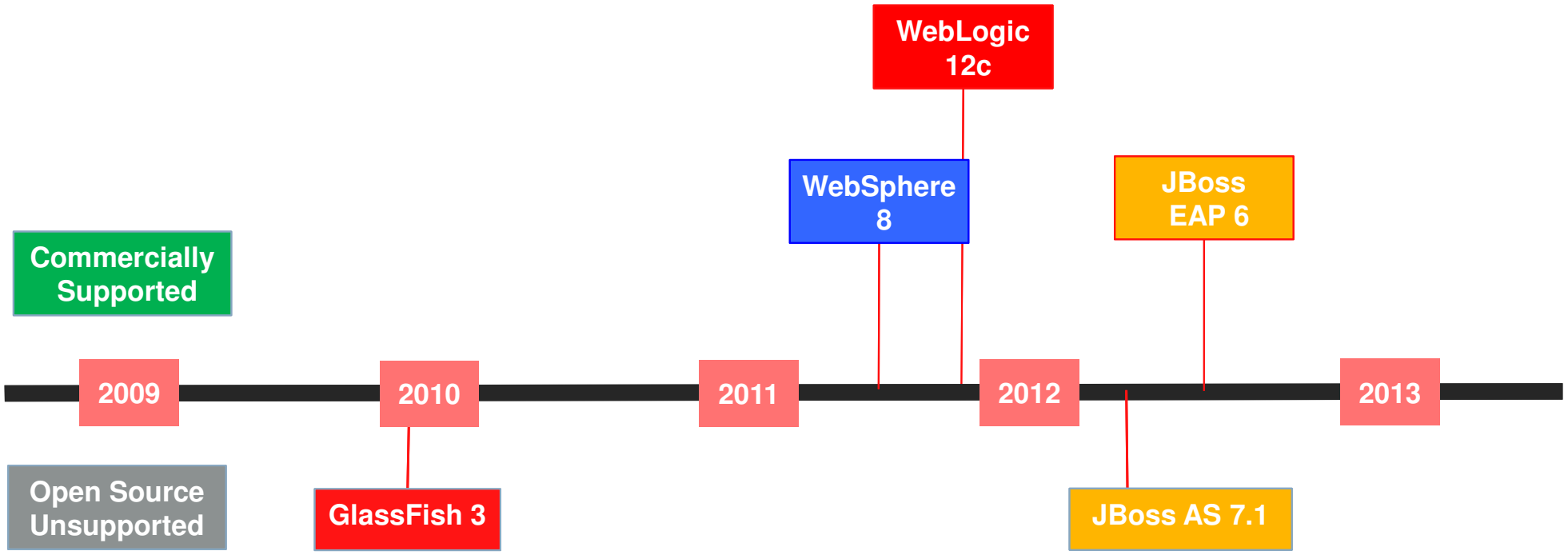


 |

Java EE Timeline to Adoption

Industry Java EE 6 Full Platform Support

Timeline to Adoption

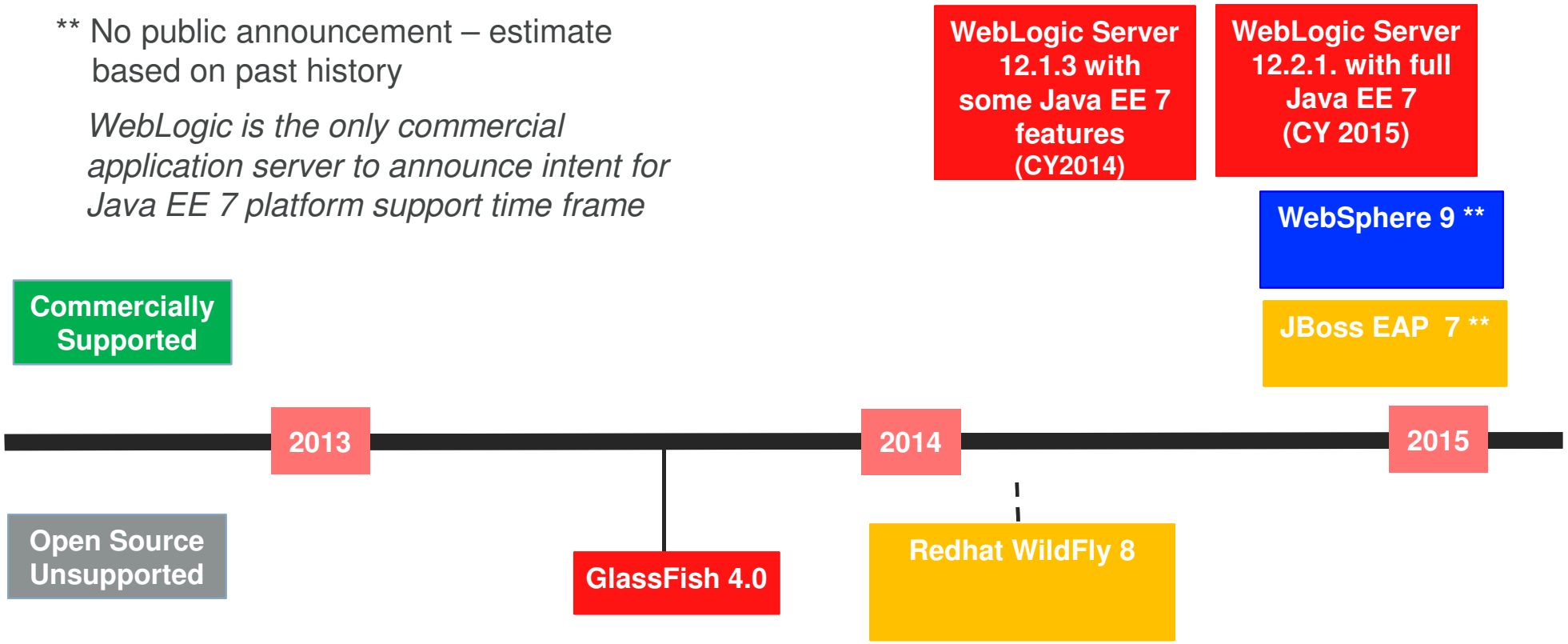


Industry Java EE 7 Full Platform Support

Timeline to Adoption - Based on Releases, Public Documentation

** No public announcement – estimate based on past history

WebLogic is the only commercial application server to announce intent for Java EE 7 platform support time frame

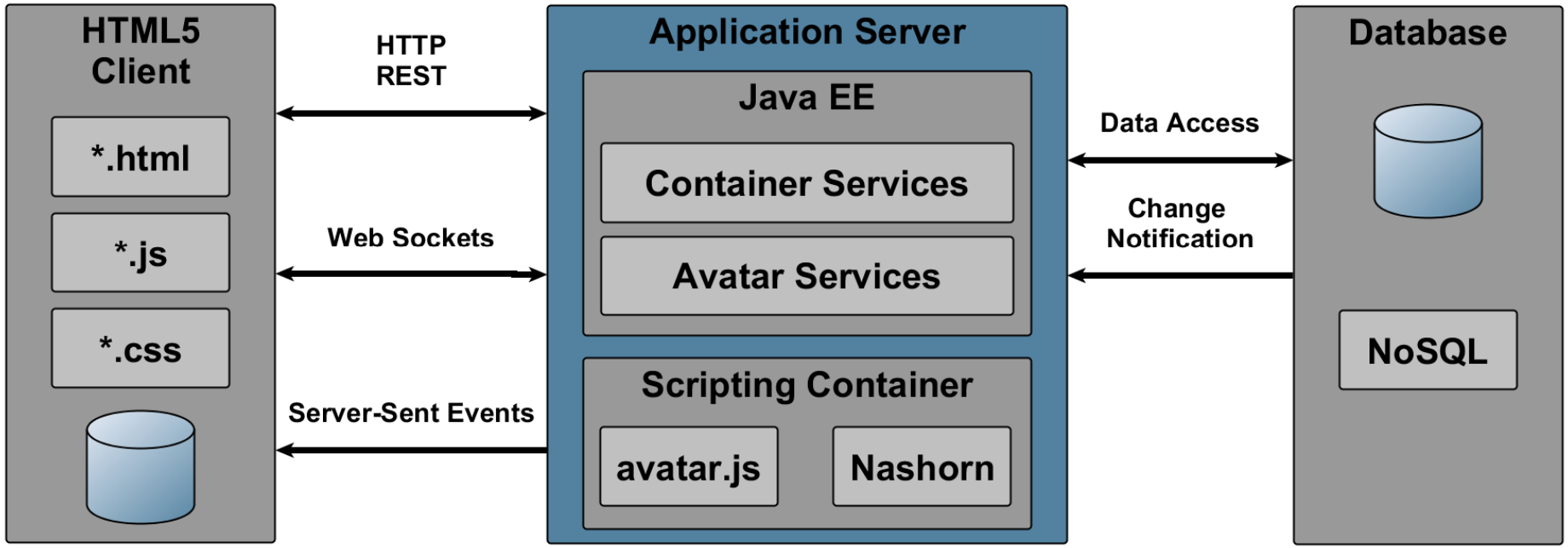


 |

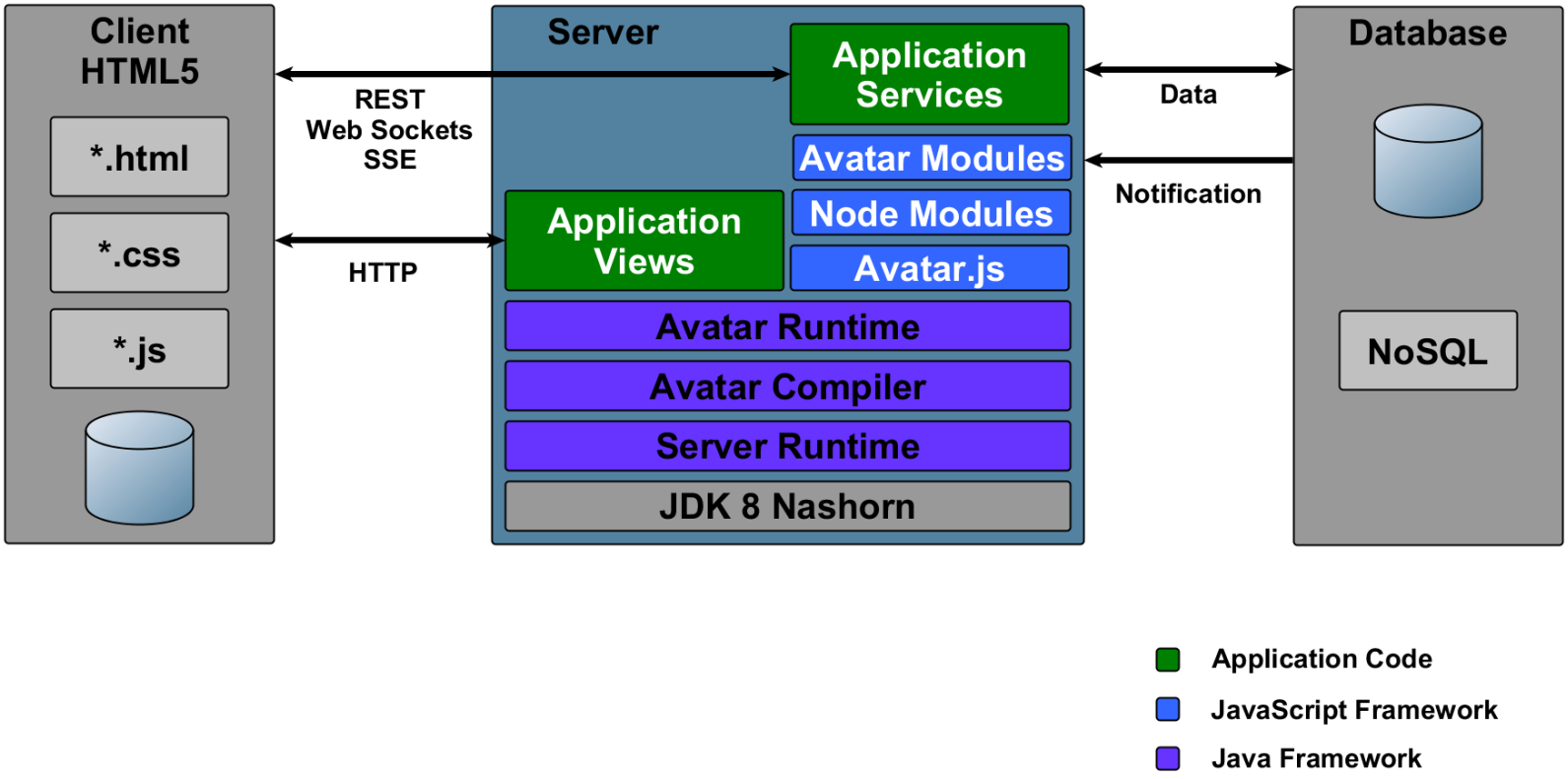
GlassFish und Project Avatar

ORACLE

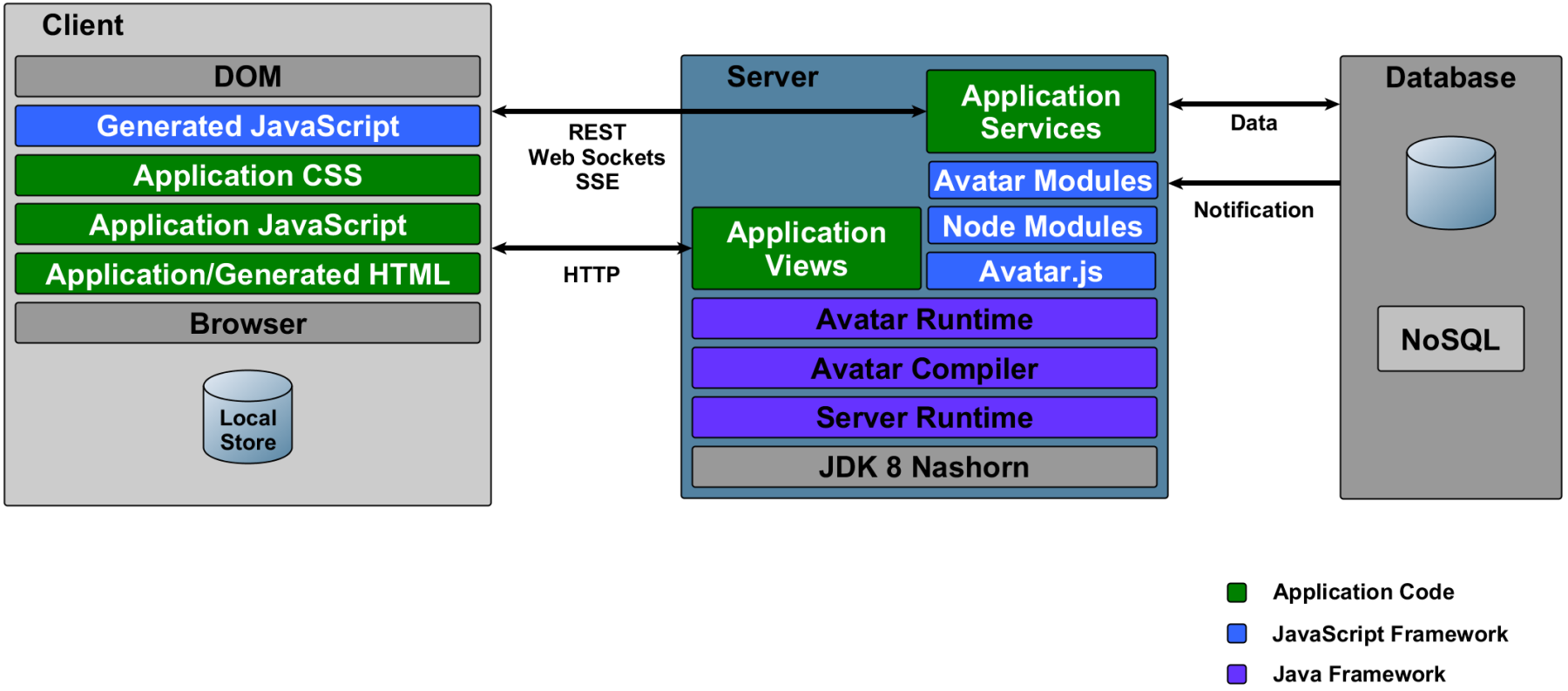
Project Avatar (1)



Project Avatar (2)



Project Avatar (3)



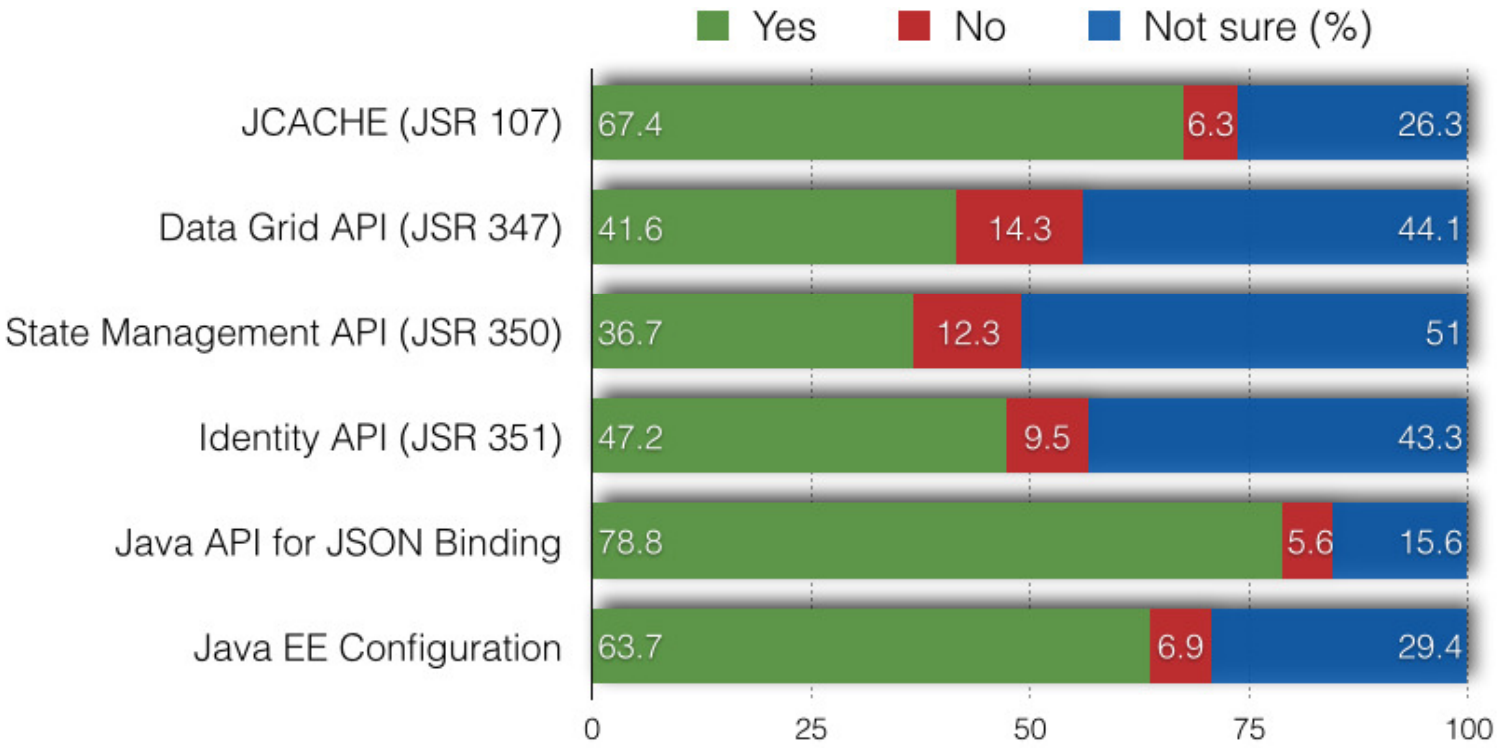
Java EE 8

ORACLE

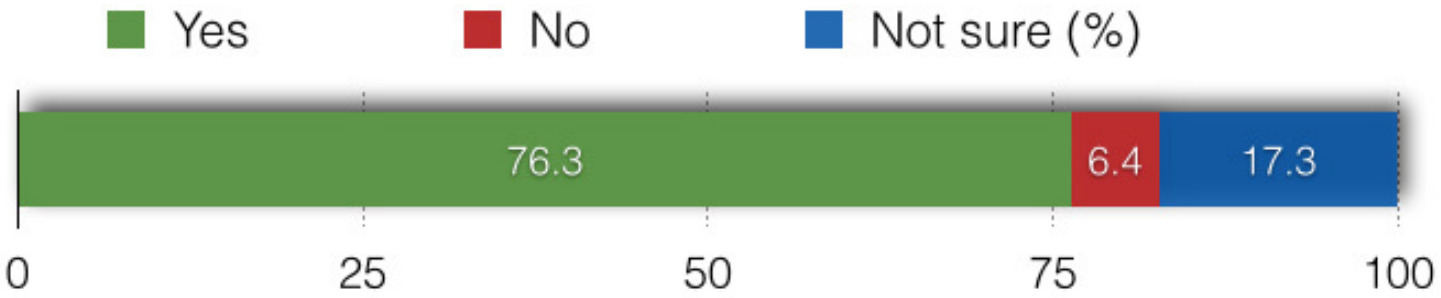
Java EE 8 Community Survey Results – Part 1

- New JSR's
- Web-Tier / HTML5
- CDI Alignment
- NoSQL

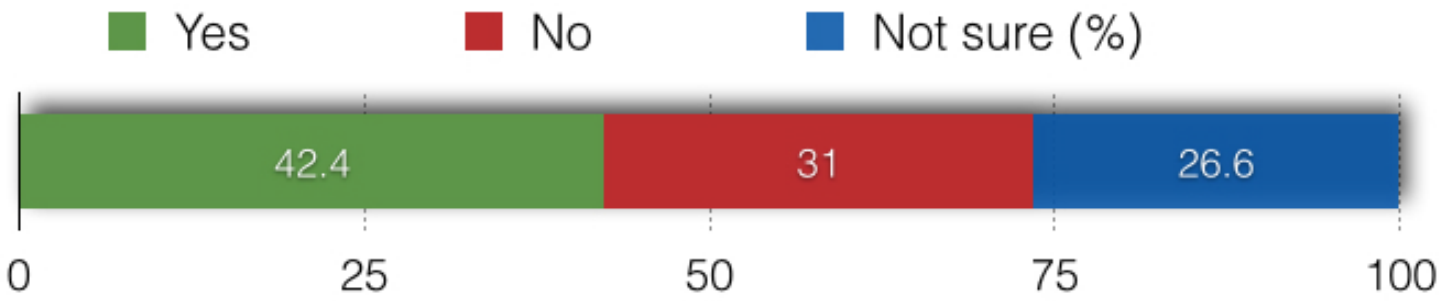
New JSR's - Which of these APIs do you think is important to be included into Java EE 8?



Should we also standardize a Java API for server-sent events?



Should we define APIs to support use of JavaScript on the server (e.g., Project Avatar)?

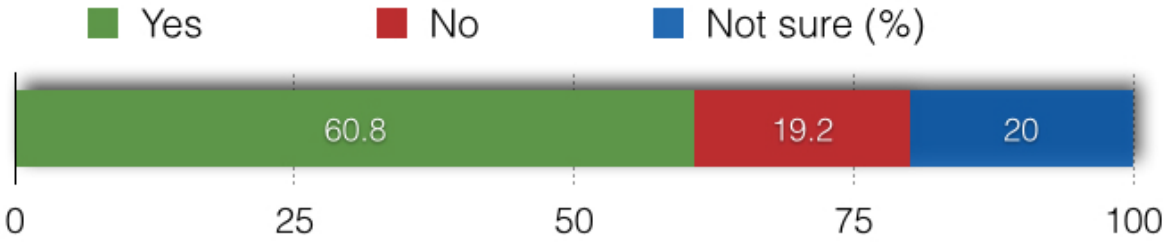


What should we add to better enable HTML5 mobile apps?

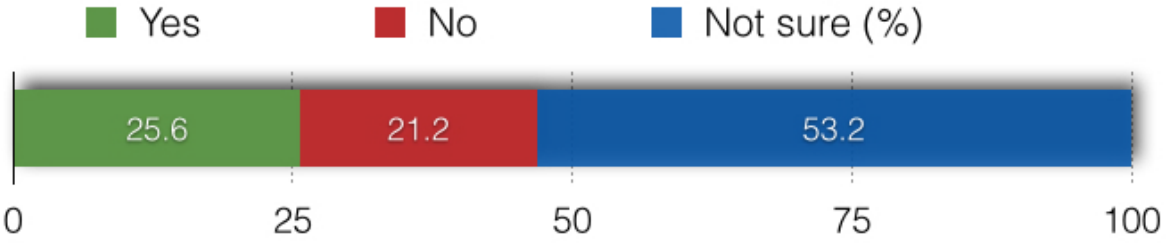
- Responses to this open-ended question were fairly diverse
- One group of responses focused on suggestions that take action mostly or entirely in the user agent. Within this group, one third suggested standardizing a JavaScript framework or anointing one particular framework and building integration that assumes that framework. The remainder were split among suggestions to offer some kind of direct binding between one or more particular pieces of Java EE and the user agent, suggestions to standardize one of the client adapter layers such as Sencha touch or Apache Cordova, and suggestions to put more Java on the user agent
- Another group of responses were focused on JSF. Most suggested building further on JSF's existing HTML5 support. Many liked the ability of JSF to cleanly address responsive web design concerns by virtue of its renderkit concept. The remainder suggested improvements to the markup generated by JSF

MVC Support

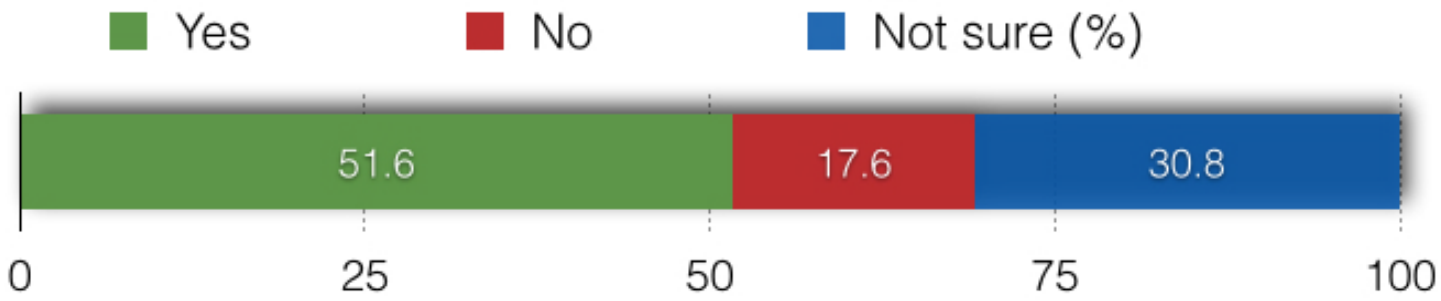
Should Java EE provide support for MVC, alongside JSF?



Is there any one de-facto standard technology in this space to which we should look for inspiration?

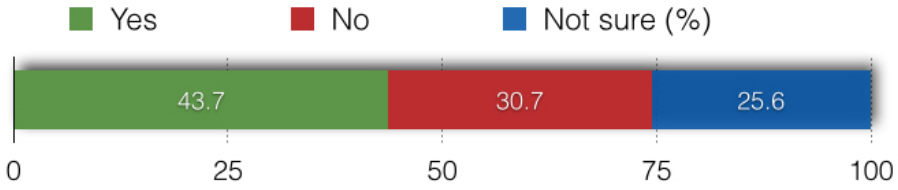


Client API for TSA: Should we investigate standardizing a client API for Thin Server Architecture?

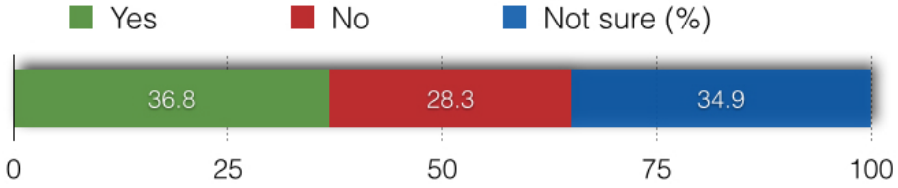


Templating

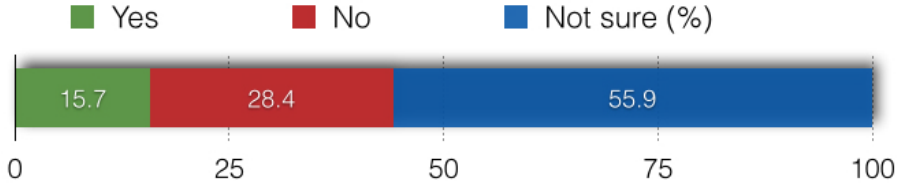
Should we define a (new) standard templating framework?



Should we extract Facelets from JSF as the standard template engine forJava EE?

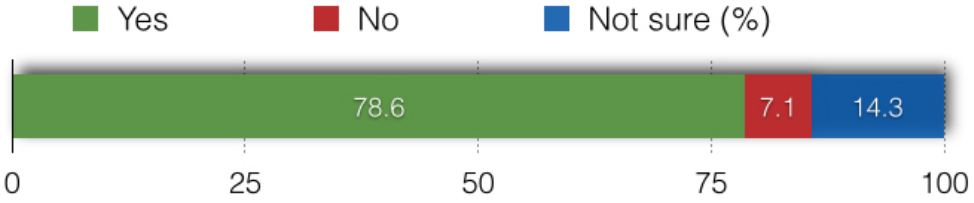


Should we standardize on another existing technology?

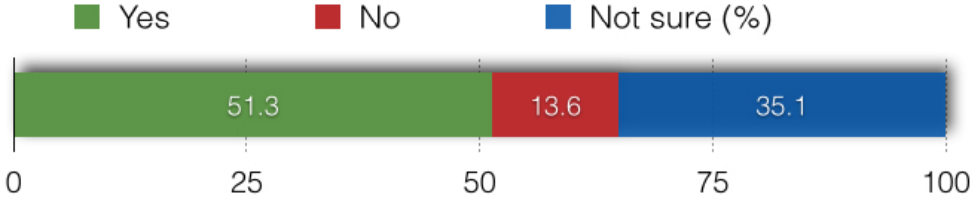


CDI

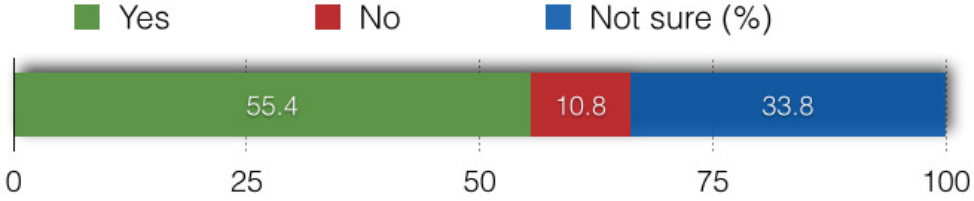
Should we consider adding Security Interceptors in Java EE 8?



Should we consider replacing @Resource with Implicit Producers?

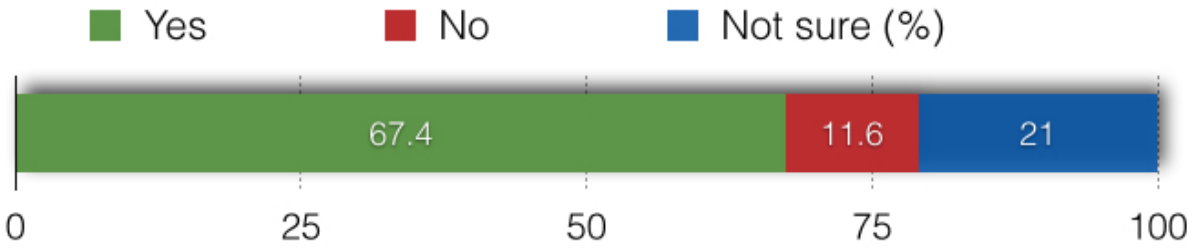


Should we consider expanding the use of CDI stereotypes?

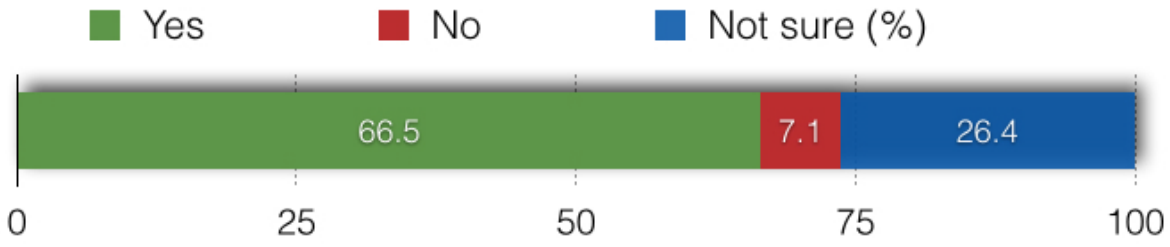


CDI - con't

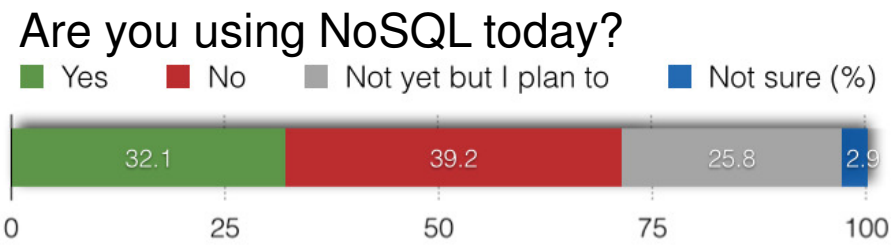
Should we consider generalization of the EJB Timer Service for use by other managed beans?

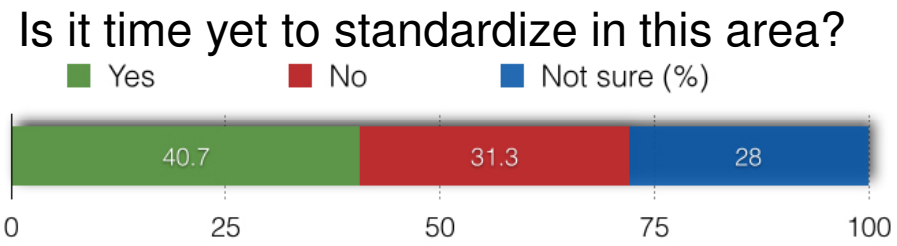


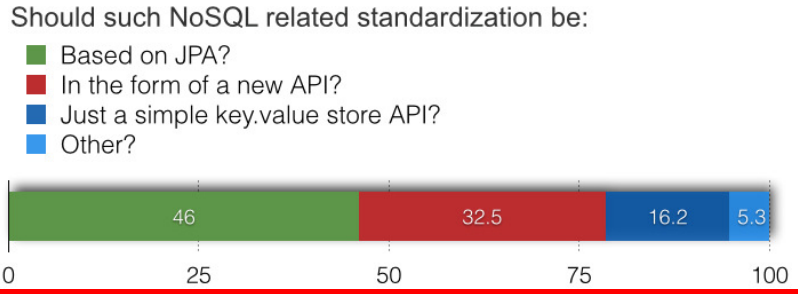
Should we consider expanding CDI events and observer methods for other services (e.g., timeout events, JMS messages, …)?



NoSQL





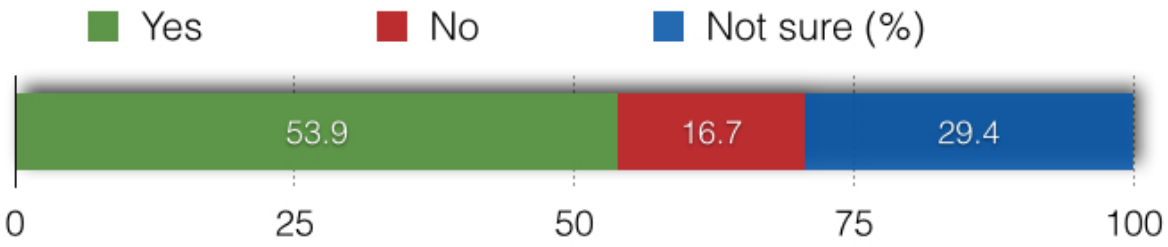


Java EE 8 Community Survey Results – Part 2

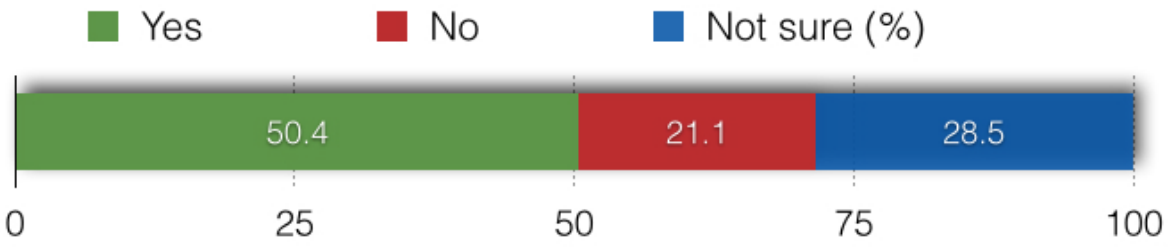
- Cloud (PaaS / SaaS / Multitenancy)
- Logging
- Security
- Testability
- Deployment and Management
- Profiles and/or Pruning

Cloud - PaaS, SaaS, and Multitenancy

Is it time to standardize support for PaaS?

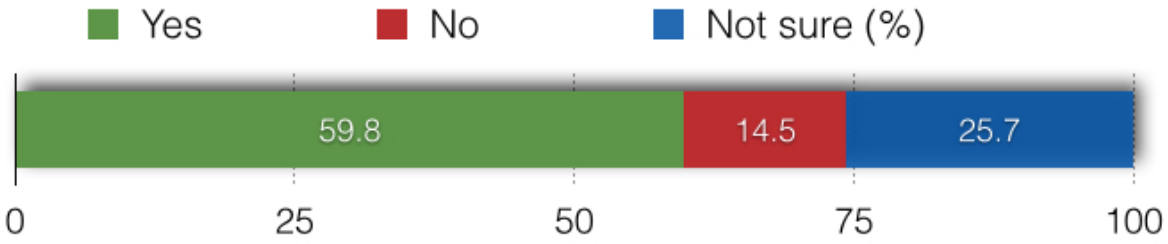


Is it time to standardize support for SaaS?

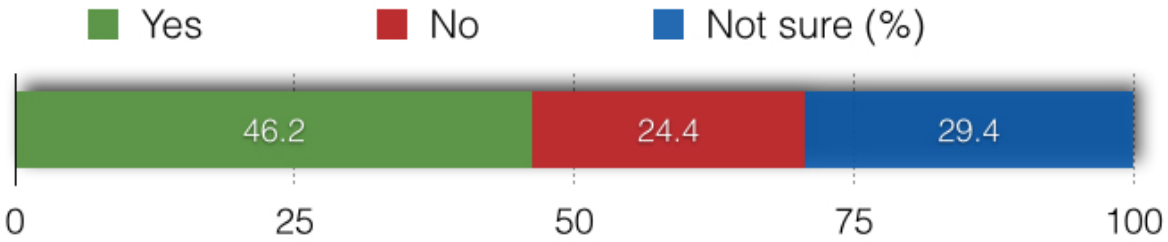


Cloud - PaaS, SaaS, and Multitenancy – con't

Is it time to standardize support for single application deployment with multiple application runtime instances in a single application server instance?

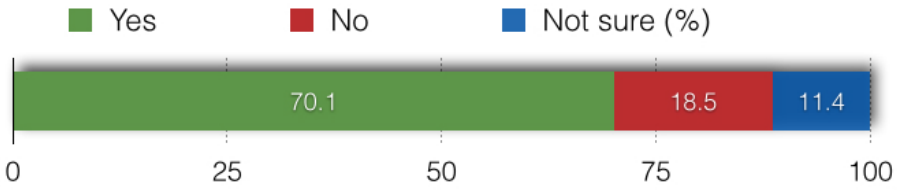


Is it time to standardize support for multiple tenants to run within a single application instance?

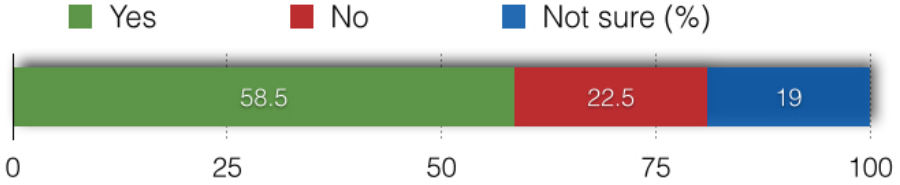


Logging

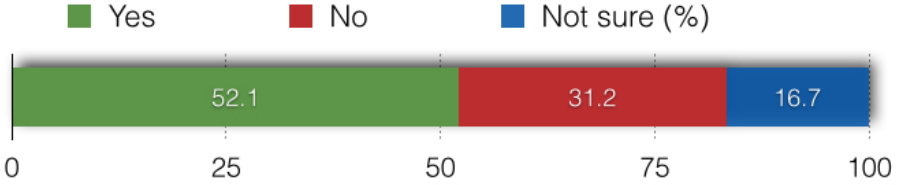
Should we standardize the use of java.util.logging by applications?



Should we standardize the names of loggers used by the application server?

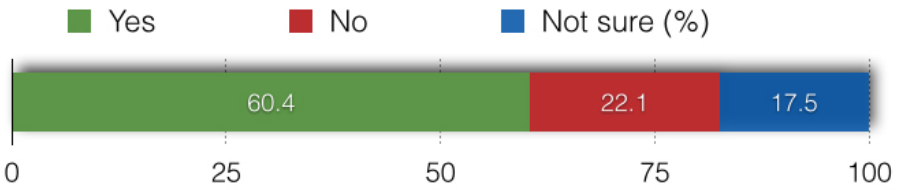


Should we standardize the format of log files?

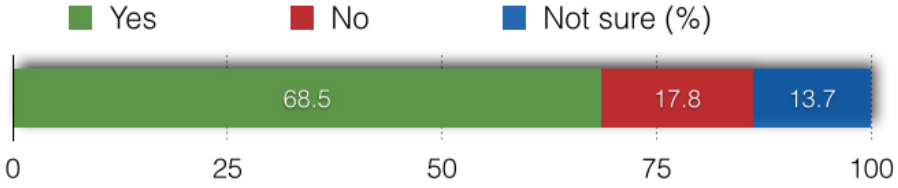


Logging – con't

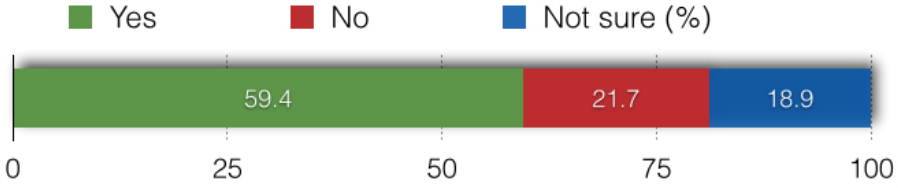
Should we standardize the management of log files?



Should we standardize the logging configuration for application loggers?

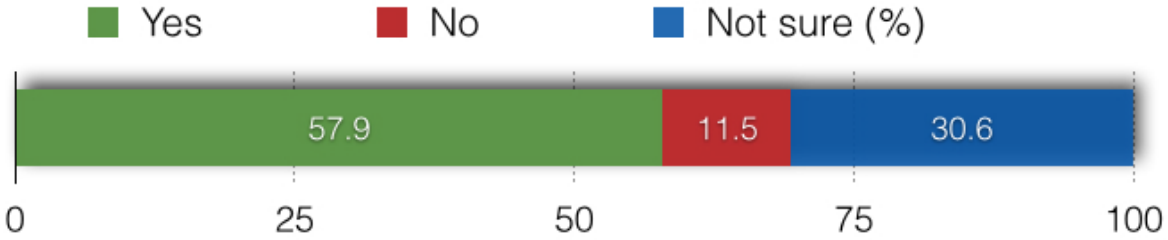


Should we standardize access to log messages by applications?

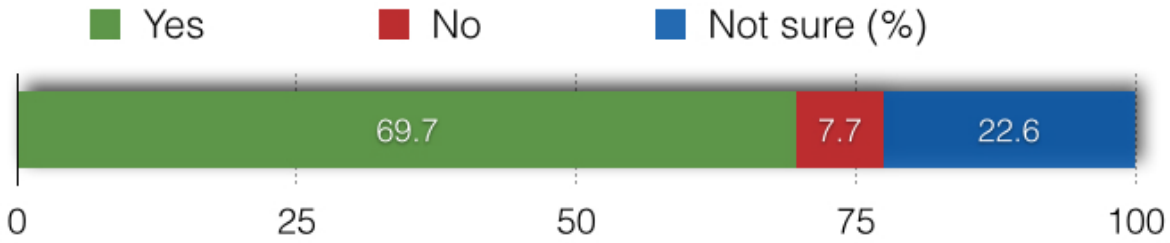


Security

Should we add support for password aliases (including the ability to provision credentials along with the application)?

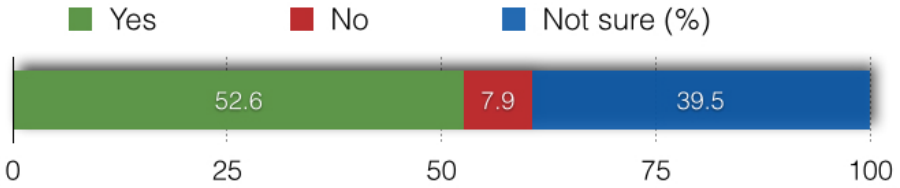


Should we standardize group-to-role mapping?

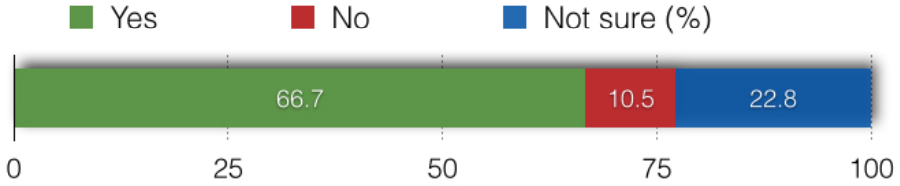


Security – con't

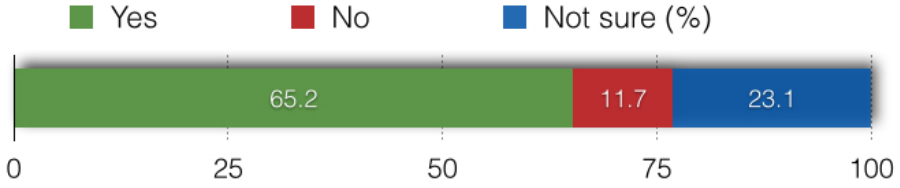
Should we simplify JASPIC?



Should we simplify authorization by introducing an EL-enabled authorization annotation?

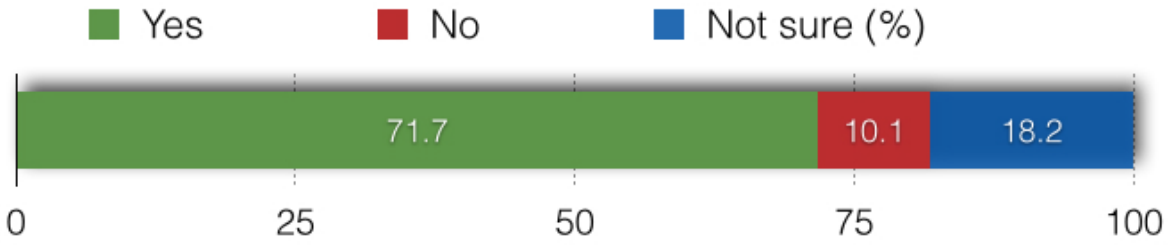


Should we standardize on requirements for simple security providers and their configuration?

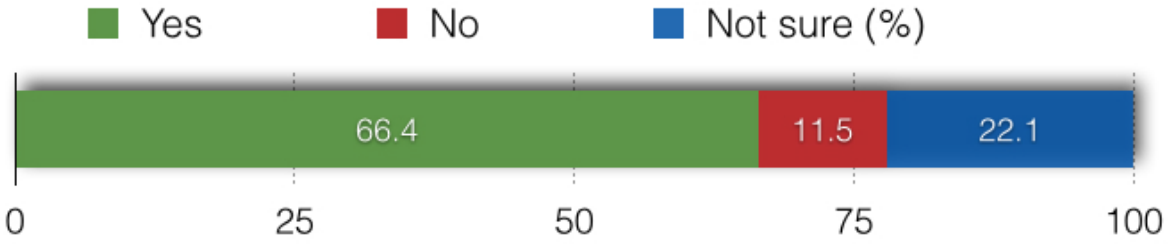


Embedded Containers and Testability

Should we define an embedded web container?

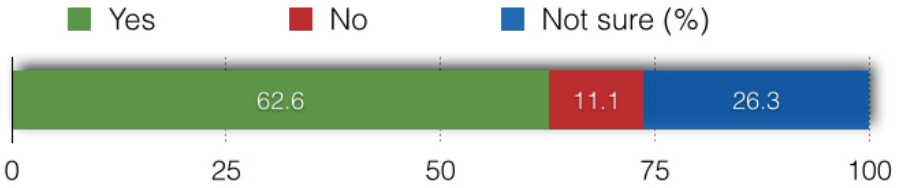


Should we define an embedded Java EE container?

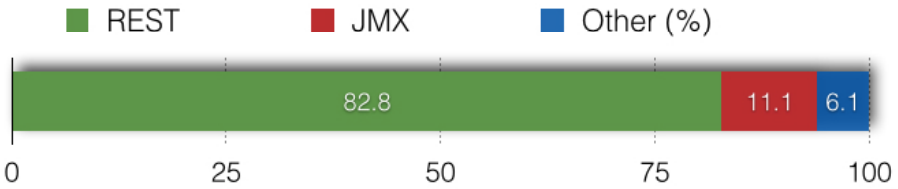


Deployment, Management, and Monitoring

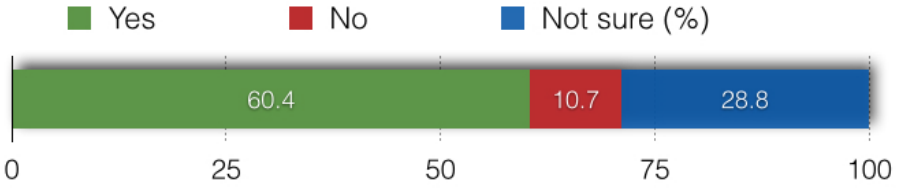
Should we define new API's to deploy and manage applications?



Should such new Deployment and Management API's be REST APIs or JMX API's?

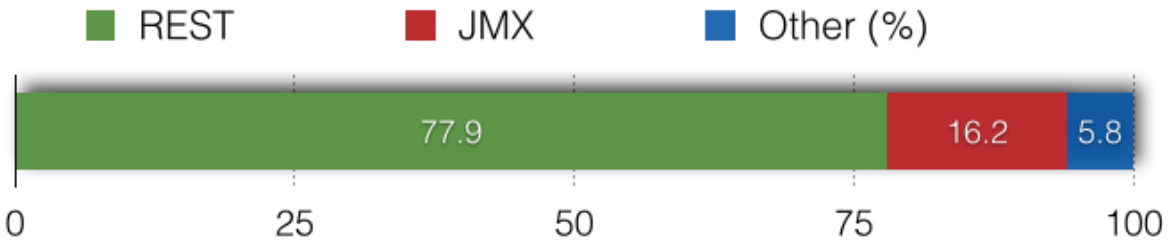


Should we define new API's to monitor applications?

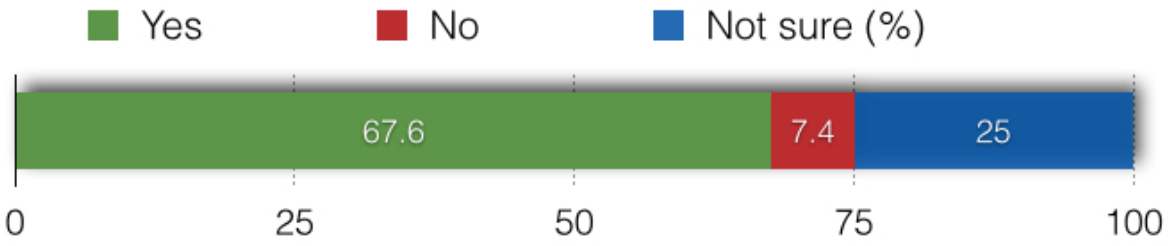


Deployment, Management, and Monitoring – con't

Should such new monitoring API's be REST APIs or JMX API's?

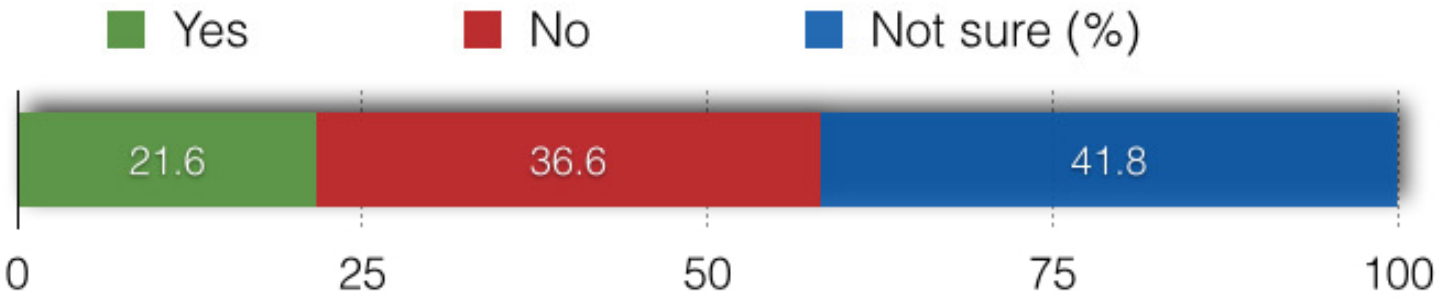


Should we define a trace context API to track a request as it goes through the system?



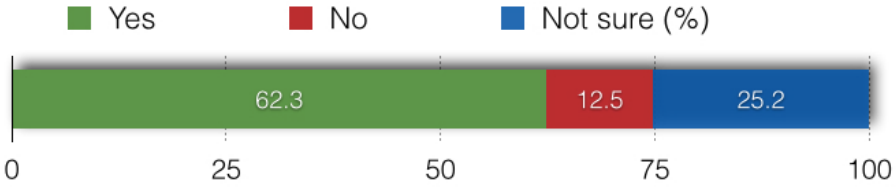
Profiles

Should we define any additional Java EE profiles?

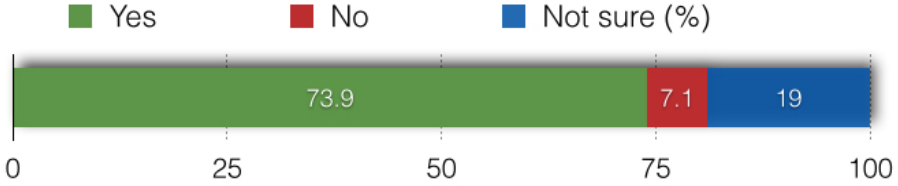


Pruning

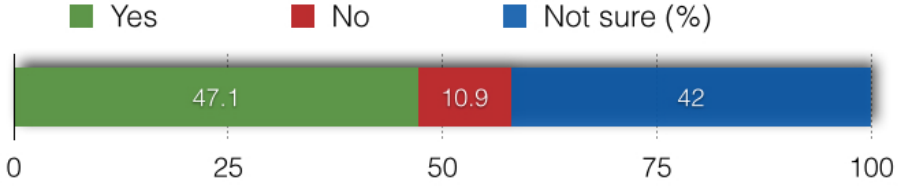
Should we prune CORBA, including support for interoperability by means of IIOP?



Should we prune EJB 2.x remote and local client view (EJBObject, EJBLocalObject, EJBHome, and EJBLocalHome interfaces)?



Should we prune JSR-77 (J2EE Management)?



Java EE 8 Specification (JSR-366)

- **Specification Lead Member:** Oracle America, Inc.
- **Specification Leads:** Linda DeMichiel, Bill Shannon
- **Who is supporting the Java EE 8 Specification:**
  - Credit Suisse
  - IBM
  - London Java Community
  - Oparco
  - OW2
  - Red Hat
  - TmaxSoft Inc
  - Adam Bien
  - David Blevins
  - Jeff Genender
  - Antonio Goncalves
  - Josh Juneau
  - Werner Keil
  - Kito Mann
  - Michael Remijan

Java EE 8 Specification (JSR-366) con't

- **Specification**
  - JSR Review to be finished by 8th of September 2014
  - JSR Review Ballot to be finished by 22nd of September 2014
- **Main focus of this release is**
  - on support for HTML5 and the emerging HTTP 2.0 standard
  - enhanced simplification and managed bean integration
  - and improved infrastructure for applications running in the cloud
- Since its inception, the Java EE Platform has been targeted at offloading the developer from common infrastructure tasks through its container-based model and abstraction of resource access
- In recent releases the platform has considerably simplified the APIs for access to container services while broadening the range of the services available
- In this release we aim to continue the direction of improved simplification, while extending the range of the Java EE platform to encompass emerging technologies in the web space and in cloud technology

ORACLE

Java EE 8 Specification (JSR-366) con't

- **Latest web standards**: Server-Sent Events, JSON Binding, HTTP 2.0, improvements to WebSocket and JSON-P, MVC
- **Ease of development**: Declarative security using CDI, notification of timed events using CDI event and observer mechanism
- **Infrastructure for cloud support**: Configuration of multiple tenants, simplified and improved security configuration; and REST-based APIs for monitoring and management
- **Alignment with Java SE 8**: Taking advantage of features available such as repeating annotations, Lambda expressions, the Date/Time API, Type annotations, Completable Futures and so on
- **Preliminary contents** as JSR candidates: JCache, JSON-B, Java Configuration
  - **Preliminary contents** which are already included & updated
- **Web profile**: JSON-B will likely be included
- **Pruning**: EJBLocalHome and support for CORBA IIOP interoperability

Java EE 8 - Latest web standards

- Java EE 7 delivered support for HTML5 dynamic and scalable applications with the Java API for WebSocket, the JSON Processing API, and Servlet NIO
  - Web standards continue to evolve, and it is critical that the Java EE platform support developments in this space
- In Java EE 8 we expect to augment our support for HTML5 applications by adding support for server-sent events, standardized binding between JSON text and Java objects, and improvements to the Java API for WebSocket and the Java API for JSON Processing
  - We expect to add support for the emerging HTTP 2.0 standard (scheduled for submission as a Proposed Standard in 2014) to the Servlet API
  - We also expect to add support for action-based MVC

Java EE 8 – Server-Sent Events

Latest web standards

- Standardized “Comet”
- Server-to-client streaming of text data
- Long-lived HTTP connection
- Client message notifications as DOM events
- Per-event data streams over same connection
- Browsers handle connection management and stream parsing
- Mime type is text/event-stream
- How to fit SSE API into Java EE 8

Java EE 8 – Server-Sent Events

Latest web standards - JSR Options Evaluated

- Servlet
- WebSocket
- Standalone
- JAX-RS

Java EE 8 – Server-Sent Events

JSR Options Evaluated: **Servlet**

- The common denominator for HTTP in Java EE
- SSE runs over an HTTP connection
- Async connections already supported
- SSE is HTTP streaming

✗ Right level of abstraction for HTML5 developers?

✗ Application scope vs. connection scope

✗ Will focus on HTTP/2

  ✗ Server push in HTTP/2 not for streaming data

✗ Operates at HTTP not resource level

ORACLE

Java EE 8 – Server-Sent Events

JSR Options Evaluated: **WebSocket**

- API similarities
- Lifecycle events and scopes (connection scope)
- One-way vs. two-way
- Interface and annotation sharing
  - Even if they come from javax.websocket?
  - Related being part of the “HTML5 stack”

✗ What to do with javax.websocket packages?

✗ Relative complexity of API:

  ✗ Programmatic vs. declarative endpoints

  ✗ Synchronous and asynchronous messaging

✗ Unclear semantics for @OnClose and @OnError

✗ Why bring WS implementation if only SSE is needed?

ORACLE

Java EE 8 – Server-Sent Events

JSR Options Evaluated: **Standalone**

- Small specification, small runtime footprint
  - No need for a large runtime just for SSE
- Declarative API using own set of annotations
  - @OnOpen, @OnClose, etc.

✗ Yet another JSR

✗ Resourcing, additional team

✗ Potential duplication of annotations and interfaces

✗ New client API from scratch?

✗ Unclear semantics for @OnClose

Java EE 8 – Server-Sent Events

JSR Options Evaluated: **JAX-RS**

- Ease of implementation - Simplicity
  - SSE is already supported in Jersey
  - JAX-RS supports async connections
- Combining regular HTTP and SSE connections
- Popularity of JAX-RS
- Streaming a REST resource with special media type
- Sharing client API concepts like configuration, targets, etc.

✗ Make JAX-RS even larger

✗ SSE as streaming resources may look “unrestful”

✗ Why bring JAX-RS implementation just for SSE

- Many applications will use JAX-RS anyway!

Java EE 8 – Server-Sent Events

JSR Options Evaluated: **JAX-RS is the Recommendation**

- SSE is streaming HTTP resources with special media type
  - Already supported in JAX-RS
- JAX-RS already popular for HTML5 applications
  - Footprint is a non-issue in practice
  - Angular JS and JAX-RS becoming quite popular
- Small extension
  - Server API: new media type, EventOutput (Broadcaster)
  - Client API: new handler for SSE events
- Convenience of mixing other HTTP operations with SSE GET's

Java EE 8 - JSON Binding (JSR 367)

Latest web standards

- Standard way to convert JSON into Java objects and vice versa
- JSON-B will leverage JSON-P and provide a conversion layer above it
- A default mapping algorithm will be defined for converting existing Java classes to JSON
  - The default mappings can be customized through the use of Java annotations and will be leveraged by the JSON-B runtime to convert Java objects to/from JSON
- Requirements from upper layers of JAX-RS
- JSON-B will have a similar feel to JAXB

Java EE 8 - Servlet 4.0 (JSR 369)

Latest web standards

- Expose support for the upcoming IETF standard HTTP/2 to users of the Servlet API
- Request/response multiplexing
- Stream prioritization
- Server push
- Upgrade from HTTP 1.1 -Refresh the Servlet API with to achieve compliance with new features in HTTP 1.1
- Responding to community input

Java EE 8 - MVC 1.0 (JSR 371)

Latest web standards

- Leverage existing Java EE technologies
- Model part should leverage CDI and Bean Validation
- View part should leverage existing view technologies like JSP's and Facelets
- Controller could be either JAX-RS based, invent a new technology, or defined a technology-independent way of defining
- Out of scope to define a new templating language, existing language should be evaluated, possibly define an SPI for additional ones to be integrated

Java EE 8 - JAX-RS 2.1 (JSR 370)

Latest web standards

- Adding support for Server-Sent Events
- Improving integration with CDI
- Exploring support for non-blocking I/O in providers (filters, interceptors, etc)
- Evaluating ways in which declarative security can be supported either directly in this JSR or by leveraging other EE-platform JSRs
- Making JAXB conditional on runtimes where it is available
- Providing integration with JSON-B
- Building upon the hypermedia API added in version 2.0
- Investigate the reactive programming paradigm as a way to improve the JAX-RS asynchronous client API
- Evaluating any requirements necessary to support the use of JAX-RS resource classes as controllers in the MVC 1.0 JSR

ORACLE

Java EE 8 - Ease of development

- We plan to enhance the managed bean model to make ease of use features that are currently available only to selected components available to all managed beans via the mechanisms provided by CDI

- In particular, we plan to consider enhancements for declarative security by means of CDI interceptors and for notifications for timed events by means of the CDI event and observer mechanism

Java EE 8 – CDI 2.0 (JSR 365)

Ease of development

- CDI split into two parts: Core CDI programming model and Java EE integrations for CDI
- Define a portable bootstrap API for CDI
- Passing thread-bound request-response context model, allow the application or container to portably push the active context to CDI when it requires the use of CDI
- Introduce modularity to CDI, CDI 2.0 Modularity Proposal
- Define a lightweight container

Java EE 8 - Infrastructure for cloud support

- We expect to augment the infrastructure added in Java EE 7 for cloud support. Areas that we plan to consider include more flexible support for **configuration**:
  - including support for configuration of multiple tenants (Multitenancy)
  - simplified and improved security configuration (Simplified security)
  - and REST-based API's for monitoring and management

Java EE 8 – Configuration

Infrastructure for cloud support

- Simple map based configuration abstraction
- Configuration Service API
- Configuration description template
- SPI to configure/back the implementation used by the accessor API

Java EE 8 - Alignment with Java SE 8

- Java EE 8 will build on Java SE 8. We will encourage component JSRs to review and improve their APIs with the Java SE 8 language changes in mind, so that developers will be able to take advantage of new features such as:
  - Repeating annotations
  - Lambda expressions
  - Date & Time API
  - Type annotations
  - CompletableFutures
  - etc.

Java EE 8 - Preliminary contents (1)

- We expect that the following new JSRs will be candidates for inclusion in the Java EE 8 Platform:
  - JCache (JSR-107)
  - Java API for JSON Binding (JSR-367)
  - Model View Controller (MVC) (JSR-371)
  - The new component JSR's that we plan to propose for inclusion add significant capabilities to the platform:
    - JCache enables scaling of applications by standardizing a powerful caching layer accessible from all containers
    - the API for JSON Binding builds on the API for JSON Processing to provide for mapping between JSON text and Java objects
    - the Model View Controller JSR provides for action-based MVC, to complement the component-based approach of JSF

Java EE 8 – JCache Java Temporary Caching API (JSR 107)

Preliminary contents

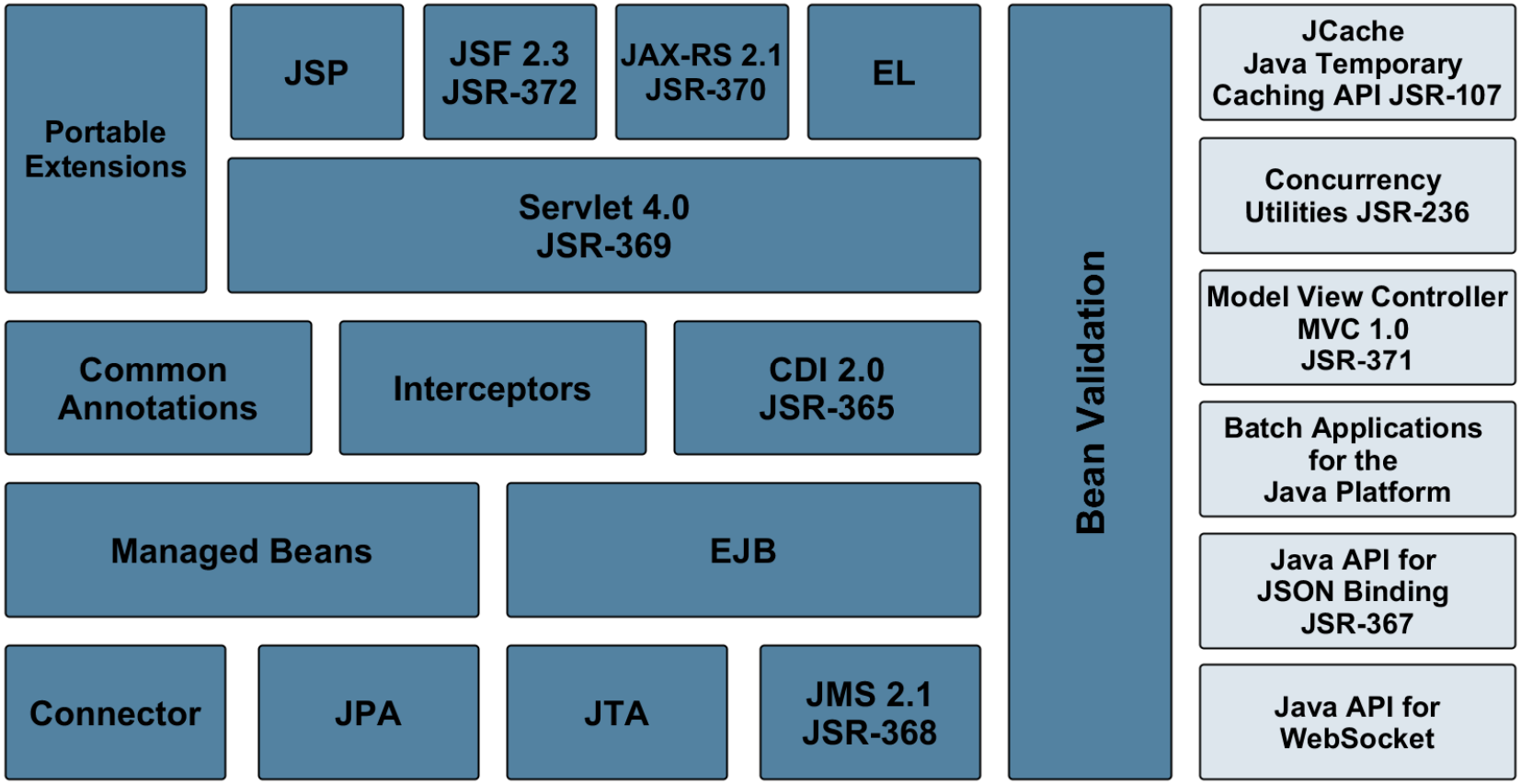
- Java Temporary Caching API provides a standard api for accessing caches and in-memory data grids from Java as well as providing advanced capabilities like on-grid processing and eventing
- JCache specification standardizes in process caching of Java objects in a way that allows an efficient implementation, and removes from the programmer the burden of implementing cache expiration, mutual exclusion, spooling, and cache consistency
- Objects whose type is not known until runtime can be cached, but only those which implement the serializable interface can be spooled

Java EE 8 - Preliminary contents (2)

- Several of the technologies already included in the Java EE platform are expected to be updated for the Java EE 8 release, including all or some of the following:

  - Java API for WebSocket
  - Java API for JSON Processing (JSON-P)
  - Java API for RESTful Web Services (JAX-RS)
  - JavaServer Faces (JSF)
  - Java Servlet
  - Expression Language (EL)
  - Interceptors
  - Java Message Service (JMS)
  - Concurrency Utilities for Java EE
  - Batch Applications for the Java Platform
  - Contexts and Dependency Injection for Java EE (CDI)
  - Bean Validation
  - Common Annotations
  - Java Connector Architecture
  - Java Transaction API (JTA)
  - Java Persistence API (JPA)
  - Enterprise JavaBeans (EJB)
  - JavaServer Pages (JSP)

Java EE 8 – JSR's



Java EE 8 – JSF 2.3 (JSR 372)

Preliminary contents

- Ajax method invocation: Allows invoking CDI managed bean methods directly from Ajax, all owing the response to be sent using standard JSON
- Community input: multi-field validation, @Inject FacesContext, EL performance optimizations, and cross-form Ajax clarifications
- Better integration with Java EE 8 and Java SE 8

ORACLE

Java EE 8 – JMS 2.1 (JSR 368)

Preliminary contents

- Simplifying and extending the API required to receive messages asynchronously
- Portability of JMS providers within Java EE servers will be improved
- Clarify “application server facilities” in chapter 11 of JMS 2.0
- Clarify JCA resource adapter relationship
- Improvements in JMS provider in Java EE transaction
- Corrections and minor enhancements

ORACLE

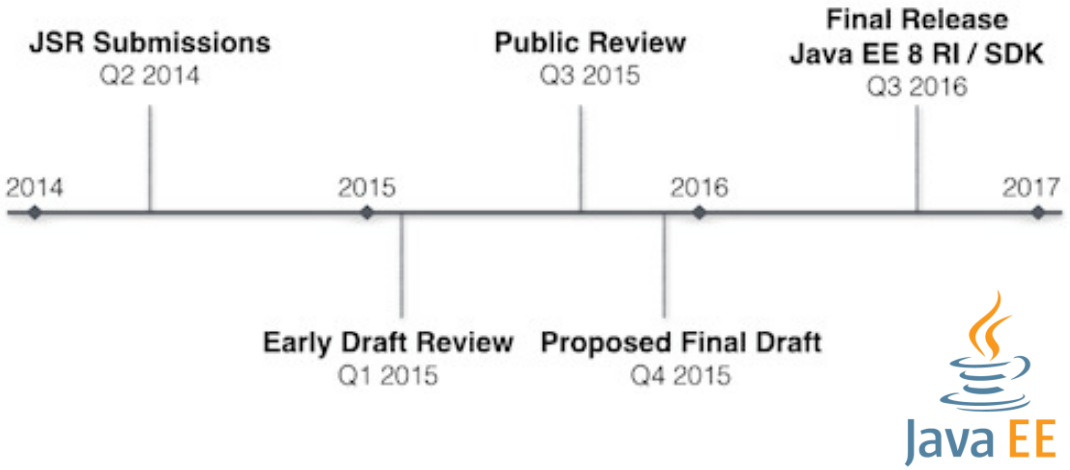
Java EE 8 - Web Profile

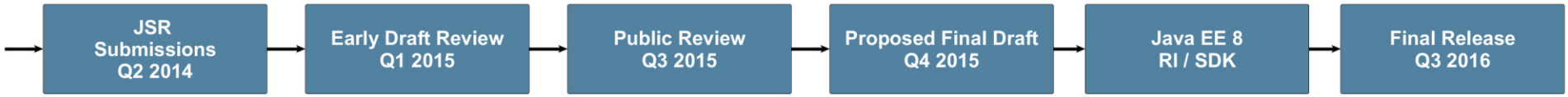
- This JSR will also update the Java Enterprise Edition Web Profile to version 8
- In addition to incorporating the latest versions of the technologies currently in the Web Profile, we expect to consider adding new required technologies to the Web Profile, such as the new API for JSON Binding

Java EE 8 - Pruning

- In accordance with the pruning process defined by the Java EE 6 specification, we will consider designating the following as Proposed Optional in this release:
  - the EJB 2.x client view APIs (EJBObject, EJBHome, EJBLocalObject, EJBLocalHome) and support for CORBA IIOP interoperability

Java EE 8 Roadmap





Summary

- Java EE 8 themes
  - Support for the latest web standards (HTTP 2.0)
  - Continue to work on ease of development
  - Improve the infrastructure for cloud support
  - Alignment with Java SE 8
- New JSR's added to the platform
  - JCache
  - Java API for JSON Binding
  - Java Configuration
- Updated JSR's
- Inside upcoming GlassFish Application Server

Danke!

Wolfgang.Weigend@oracle.com

Twitter: @wolflook