
ORACLE®

JavaOne™

ORACLE®

Java EE 8 Update

Linda DeMichiel
Java EE 8 Specification Lead
Oracle
September 19, 2016

JavaOne
ORACLE

Safe Harbor Statement

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.

Program Agenda

1. Road to Java EE 8
2. Java EE 8 JSRs (Original Proposal)
3. Proposed Shift in Focus
4. Where to Learn More at JavaOne

Java EE 7

Productivity

- Annotated POJOs
- Less boilerplate code
- Integrated
- Excellent tool ecosystem

HTML5-Ready

- JSON
- WebSockets
- JAX-RS

Meets Enterprise Demands

- Java Message Service
- Batch processing
- Distributed transactions

Scalable

- Multi-threaded
- Asynchronous APIs (Servlet, EJB, JAX-RS)
- Concurrency utilities for Java EE

Community Driven

- Java Community Process
- Adopt-a-JSR
- Open Source RI (GlassFish)

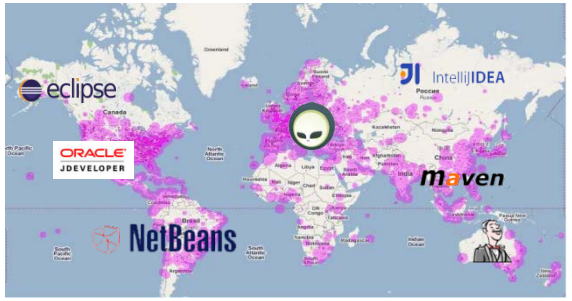
Industry Standard

- Lowers risk
- Vendor choice
- Implementation choice
- Operating system choice
- Portable applications

The Vibrant Java EE Community

Publications

Java EE Developers



Career Opportunity

indeed one search. all jobs.

careerbuilder

Dice The Career Hub for Tech Insiders™

glassdoor

Java EE Compatible Application Servers

NEC

OW2 Consortium

INFORS 中创中间件

TmaxSoft TmaxSoft Co., Ltd.

RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM

IBM

FUJITSU

Cosminexus

WildFly

ORACLE FUSION MIDDLEWARE WEBLOGIC SERVER

SAP

CAUCHO

TongTech

APACHE GERONIMO

User Groups

SouJava sociedade de usuários java

LJC

JUG

ceJug

Silicon Valley

CONGO JUG

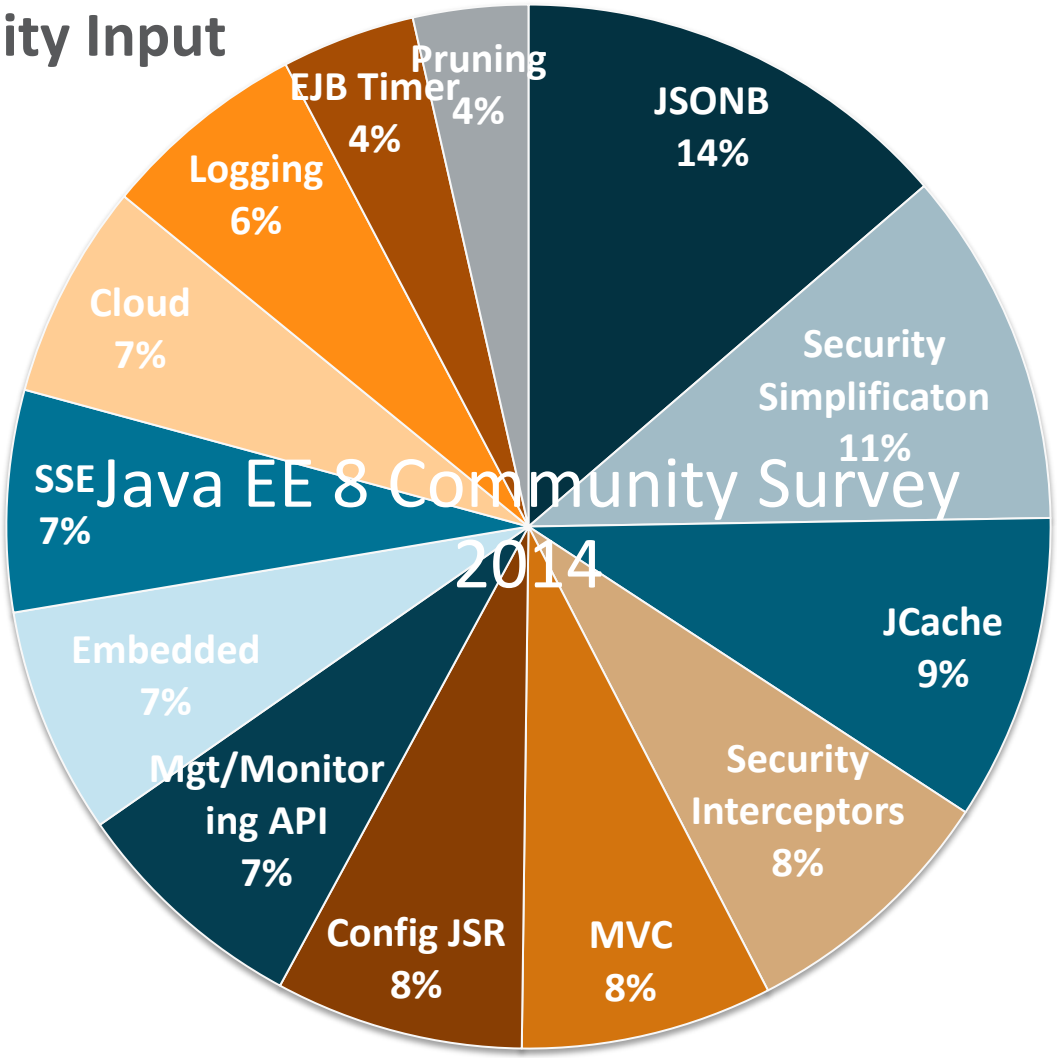
JavaOne ORACLE

Program Agenda

1. Road to Java EE 8
2. Java EE 8 JSRs (Original Proposal)
3. Proposed Shift in Focus
4. Where to Learn More at JavaOne

Java EE 8

Contents Driven by Community Input



Java EE 8

JSRs

- Java EE 8 Platform and Web Profile
- Contexts and Dependency Injection 2.0 (CDI)
- Java API for JSON Binding 1.0 (JSON-B)
- Java Message Service 2.1 (JMS)
- Java Servlet 4.0
- Java API for RESTful Web Services 2.1 (JAX-RS)
- Model-View-Controller 1.0 (MVC)
- JavaServer Faces 2.3 (JSF)
- Java EE Management API 2.0
- Java API for JSON Processing 1.1 (JSON-P)
- Java EE Security API 1.0
- Bean Validation 2.0

CDI 2.0

- Define behavior of CDI outside of a Java EE container
- Make CDI more modular to help other Java EE specs better integrate with it
- Spec split into 3 parts:
  - Core CDI
  - CDI in Java SE
  - CDI in Java EE
- API to bootstrap a CDI container in Java SE
- Observer ordering
- Asynchronous event firing

JSON-P 1.1

- Update JSON-P spec to stay current with emerging standards
- Support for IETF standards
  - JSON Pointer, JSON Patch, and JSON Merge Patch
- Add editing operations to JsonObject and JsonArray
- Helper classes and methods to better utilize Java SE 8 Stream operations

JSON-B 1.0

- JAXB-like API to marshal/unmarshal Java objects to/from JSON
- Default mapping between classes and JSON
- Customization APIs
- Standard support to handle “application/json” media type for JAX-RS
- Natural follow on to JSON-P – closes the JSON support gap

JAX-RS 2.1

- Server-sent events
- Non-blocking I/O in providers (filters, interceptors…)
- Reactive programming paradigm to improve JAX-RS asynchronous clients
- Hypermedia API enhancements
- Integration with other JSRs and frameworks

Servlet 4.0

- Support for HTTP/2
  - Request/response multiplexing
  - Server push
  - Upgrade from HTTP 1.1
- Compatibility with latest HTTP 1.1 RFCs
- Smaller community-requested improvements (JIRA issues)

JSF 2.3

- Better CDI integration
- WebSocket integration
- Ajax method invocation
- Class-level Bean Validation
- Java Date/Time support

MVC 1.0

- Provide action-based MVC framework
  - HTTP requests are routed to controllers and turned into actions by application code
  - Alternative/complement to JSF's component-based MVC framework
- Leverage existing technologies:
  - CDI, Bean Validation, Facelets, JSPs

JMS 2.1

- New API for receiving messages asynchronously
  - More flexible and general than MDBs
  - Alignment with CDI
- Improved portability of JMS providers between appservers
- Improved support for using JMS in XA transactions
- Improved handling of "bad" messages

Java EE Management API 2.0

- REST-based management APIs
  - Supersede current management EJB-based APIs of J2EE Management 1.0
  - Superset of functionality of J2EE Management 1.0
- Simple deployment API

Java EE Security 1.0

- API for managing users and groups
- Support for password aliasing
- API for role mapping
- Metadata and API for authentication
- Interceptors for authorization, with CDI support

Bean Validation 2.0

- Constraints applied to collection elements
- Support for new Date/Time API
- Integration with Optional wrappers
- Repeatable annotations
- Additional features requested from community

Where Are We Now?

Java EE 8 Progress to Date

CDI 2.0 (JSR 365)
- Bootstrap API for Java SE
- Async events
- Observer ordering

JSON-B 1.0 (JSR 367)
- JSON <-> object mapping

JMS 2.1 (JSR 368)
- Flexible JMS MDBs
- Improved XA support

Servlet 4.0 (JSR 369)
- HTTP/2 support

JAX-RS 2.1 (JSR 370)
- Reactive enhancements
- Server-sent events
- Non-blocking I/O

MVC 1.0 (JSR 371)
- Action-based MVC framework

JSF 2.3 (JSR 372)
- Small-scale new features
- Community-driven improvements

Management 2.0 (JSR 373)
- REST-based APIs

JSON-P 1.1 (JSR 374)
- JSON Pointer and Patch
- Java Lambda support

Security 1.0 (JSR 375)
- Authentication/authorization APIs

Bean Validation (JSR 380)
- Collection constraints
- Date/Time support
- Community-requested features

Where Are We Now?

The World Has Changed

- Focus on deployment into the Cloud
- Focus on microservices
- Emphasis on more rapid evolution of applications

Program Agenda

1. Road to Java EE 8
2. Java EE 8 JSRs (Original Proposal)
3. Proposed Shift in Focus
4. Where to Learn More at JavaOne

Revised Java EE 8 Proposal

Modernizing Java EE for Cloud and Microservices

Need to retarget Java EE to address these trends with 2-fold approach

1. Java EE 8 adjustment in focus
2. Java EE 9 – longer term, more in-depth work targeted at enhanced support for Cloud and microservices, leveraging work done in Java EE 8

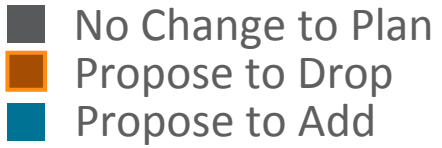
Revised Java EE 8 Proposal

Modernizing Java EE for Cloud and Microservices

Goals

- Migration path to cloud development and deployment models for Java EE customers
- Migration path to microservices-based architecture for Java EE applications
- Backwards compatibility with Java EE

Revised Java EE 8 Proposal



CDI 2.0 (JSR 365)
- Bootstrap API for Java SE
- Async events
- Observer ordering

JSON-B 1.0 (JSR 367)
- JSON <-> object mapping

JMS 2.1 (JSR 368)
- Flexible JMS MDBs
- Improved XA support

Servlet 4.0 (JSR 369)
- HTTP/2 support

JAX-RS 2.1 (JSR 370)
- Reactive enhancements
- Server-sent events
- Non-blocking I/O
- Client-side circuit breakers

MVC 1.0 (JSR 371)
- Action-based MVC framework

JSF 2.3 (JSR 372)
- Small-scale new features
- Community-driven improvements

Management 2.0 (JSR 373)
- REST-based APIs

JSON-P 1.1 (JSR 374)
- JSON Pointer and Patch
- Java Lambda support

Security 1.0 (JSR 375)
- Authentication/authorization APIs
- OAuth, OpenID support
- Secret management

Bean Validation 2.0 (JSR 380)
- Collection constraints
- Date/Time support
- Community-requested features

Configuration
- Standard for externalizing application configuration

Health Checking
- Standard for client-side health reporting

Rationale for Proposed Changes

New Functionality

- Cloud apps make many remote REST calls. Need a **client-side circuit breaker** added to JAX-RS
- Need a **secret vault** because there's no way to do this today using **standards**
- **Need OAuth and OpenID** support because those technologies have rapidly emerged as standards
- Need **externalized configuration store** to make applications retargetable across environments
- Need basic **multi-tenancy** support to accommodate needs of more complex apps and offer higher density
- **Need standard way of health checking** Java-based apps

Dropped Functionality

- **JMS** is no longer very relevant in cloud. Proposed to stay at JMS 2.0 standard (vs. upgrading to JMS 2.1).
- Cloud apps often ship headless, making **MVC** largely irrelevant
- Current **Management** JSR not widely used

Technical Focus Areas

Programming Model

- Extend for reactive programming
- Unified event model
- Event messaging API
- JAX-RS, HTTP/2, Lambda, JSON-B, ...

Packaging

- Package applications, runtimes into services
- Standalone immutable executable binary
- Multi-artifact archives

Key Value/Doc Store

- Persistence and query interface for key value and document DB

Eventual Consistency

- Automatically event out changes to observed data structures

Serverless

- New spec – interfaces, packaging format, manifest
- Ephemeral instantiation

Configuration

- Externalize configuration
- Unified API for accessing configuration

Multi-tenancy

- Increased density
- Tenant-aware routing and deployment

State

- API to store externalized state

Resiliency

- Extension to support client-side circuit breakers
- Resilient commands
- Standardize on client-side format for reporting health

Security

- Secret management
- OAuth
- OpenID

Proposal for Reactive Programming

Problem Statements

- Need to incorporate evolving reactive/async-style programming model
- More common in Cloud because apps are distributed (split into microservices) and there is increased latency
  - Many remote calls.
  - Synchronous request-handling blocks threads with remote calls

Proposal

- Migration path to fuller reactive programming model in Java EE 9
- Improve JAX-RS to support reactive programming for client side (e.g., async "orchestration" as in RXJava or in Jersey)

Proposal for Circuit Breaker – Resiliency

Problem Statements

- Prevent request-handling threads from being consumed while making requests to remote systems
- Ease up on requests to remote system as it's having problems
- Allow system time to recover
- Prevent cascading failures. Isolates failures in the source system
- Use circuit breaker without writing extensive boiler-plate code.

Proposal

- Extension to JSR 339 - JAX-RS Client
- Several possible approaches:
  - Programmatic – change in JAX-RS Client API
  - Declarative – registering @Provider classes to the Client
  - Other…
- Configurable -- potential parameters might include:
  - Sampling frequency
  - Sampling time period
  - Performance threshold (milliseconds)
  - % error threshold
  - …

Proposal for Health Checking – Resiliency

Problem Statements

- No standard for health is being reported
- Applications, resources, servers, services, micro-services, etc. will report health differently
- Traditional health check just returns opaque up/down messages. If an instance is having problems, it should report the root cause(s) where possible to help diagnose/locate problems

Proposal

- Define standard for how individual instances should report health
  - REST API with structure format in JSON
- Define configurable context path – e.g. /healthcheck
- Define semantics for reporting health
  - Health status codes
  - Reason(s) for failures, warnings, etc.
  - Health of dependencies
- Enhancements to Java APIs to facilitate easier development
  - New annotations and descriptors to specify health endpoints
- Circuit breaker could poll /healthcheck rather than waiting for HTTP requests to fail first

Proposal for Security

Problem Statements

- No standard way of connecting an application to a key service
- Need to keep sensitive stored data secret
- Hard to use OAuth
- OpenID is emerging as the default authentication standard

Proposal

- Standard way of connecting an application to a key service
- Encryption service for stored data
- Improved OAuth support
  - Registration and Discovery of Resources to Request Scopes
  - Authorization Model
- OpenID support for authentication

Proposal for Configuration

Problem Statements

- No standard way of working with configuration in applications
- No easy way of moving applications between environments without hacking into their packages
- Changing application configuration without redeployment of app
- Reconfiguring multiple instances of an application at once
- Externalized configuration is the standard for cloud

Proposal

- Standardize a mechanism of defining, injecting and using configuration within an application
- Define configuration persistence mechanisms, formats, and bindings
- Provide an ability to externalize application configuration from the application package
- Support for merging, overriding, and federating configurations from different sources
- Provide a standard mechanism for working with mutable/dynamic configuration

Proposal for Multi-tenancy

Problem Statements

- No concept of customer-facing tenancy within Java today
- One customer writing a multi-tenant App or Service has to define the concept of tenancy from scratch – hard and error-prone

Proposal

- Optional feature for servers supporting multi-tenancy
- Specification for mapping an external inbound request back to a tenant
- API for applications to find out which tenant the current request corresponds to
- Tenant-aware routing and deployment within an instance of an app server

Java EE 7

Batch	Dependency Injection	JACC	JAXR	JSTL	Management
Bean Validation	Deployment	JASPIC	JMS	JTA	Servlet
CDI	EJB	JAX-RPC	JSF	Java Persistence	Web Services
Common Annotations	EL	JAX-RS	JSON-P	JavaMail	Web Services Metadata
Concurrency EE	Interceptors	JAX-WS	JSP	Managed Beans	WebSocket
Connector	JSP Debugging	JAXB			

Java EE 8 (Revised Proposal, 2016)

Batch	Dependency Injection	JACC	JAXR	JSTL	Management
Bean Validation	Deployment	JASPIC	JMS	JTA	Servlet
CDI	EJB	JAX-RPC	JSF	Java Persistence	Web Services
Common Annotations	EL	JAX-RS	JSON-B	JavaMail	Web Services Metadata
Concurrency EE	Interceptors	JAX-WS	JSON-P	Managed Beans	WebSocket
Connector	JSP Debugging	JAXB	JSP	Security	
		Configuration	Health Check		

Next Steps

Give us your feedback

- Take the survey
  - http://glassfish.org/survey
- Send technical comments to
  - users@javaee-spec.java.net
- Join the JCP – come to Hackergarten in Java Hub
  - https://jcp.org/en/participation/membership_drive
- Join or track the JSRs as they progress
  - https://java.net/projects/javaee-spec/pages/Specifications
- Adopt-a-JSR
  - https://community.oracle.com/community/java/jcp/adopt-a-jsr

Program Agenda

1. Road to Java EE 8
2. Java EE 8 JSRs (Original Proposal)
3. Proposed Shift in Focus
4. Where to Learn More at JavaOne

Where to Learn More at JavaOne

Session Number	Session Title	Day / Time
CON7975	Enterprise Java for the Cloud	Monday 4:00 p.m.
CON1558	What's New in the Java API for JSON Binding	Monday 5:30 p.m
BOF7984	Java EE for the Cloud	Monday 7:00 p.m
CON4022	CDI 2.0 Is Coming	Tuesday 11:00 a.m
CON7983	JAX-RS 2.1 for Java EE 8	Tuesday 12:30 p.m
CON7980	Servlet 4.0: Status Update and HTTP/2	Tuesday 4:00 p.m
CON7978	Security for Java EE 8 and the Cloud	Tuesday 5:30 p.m
CON7979	Configuration for Java EE 8 and the Cloud	Wednesday 11:30 a.m
CON7977	Java EE Next – HTTP/2 and REST	Wednesday 1:00 p.m
CON6077	The Illusion of Statelessness	Wednesday 4:30 p.m.
CON7981	JSF 2.3	Thursday 11:30 a.m

Q & A

Integrated Cloud

Applications & Platform Services

JavaOne™

ORACLE®


ORACLE®