Oracle JDK 11 ist schon da

CREATE
THE
FUTURE

Wolfgang Weigend
Sen. Leitender Systemberater
Java Technology and Architecture

Java™ 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.

Agenda

1. Java SE Roadmap
2. JDK 11 Migration Guide
3. JDK 11 Features
4. Summary

JDK Version Numbers and Java Critical Patch Updates

- Rules for Java CPU's
  - Main release for security vulnerabilities
  - Covers all JDK families (11, 8, 7, 6)
  - CPU release triggers Auto-update
  - Dates published 12 months in advance
  - Security Alerts are released as necessary
  - Based off the previous (non-CPU) release
  - Released simultaneously on java.com and OTN

- JDK 11.0.1 - Security Baselines

JRE Family Version	JRE Security Baseline (Full Version String)
10	10.0.99
9	9.0.99
8	1.8.0_191-b12
7	1.7.0_201-b11
6	1.6.0_211-b11

Java SE Roadmap

Java SE Public Updates

http://www.oracle.com/technetwork/java/eol-135779.html

Java SE Public Updates				
Release	**GA Date**	**End of Public Updates Notification**	**Commercial User End of Public Updates**	**Personal User End of Public Updates**
7	July 2011	March 2014	April 2015	
8	March 2014	September 2017	January 2019****	December 2020****
9 (non-LTS)	September 2017	September 2017	March 2018	
10 (18.3^) (non-LTS)	March 2018	March 2018	September 2018	
11 and later		No longer Applicable		

End of Public Updates

Oracle will make available to Commercial Users and Personal Users updates to publicly available versions of Oracle Java SE in accordance with the table below. Once a Java SE version reaches "End of Public Updates", any further updates will be available only to Customers and accessible through My Oracle Support and via corporate auto update where applicable (Visit My.Oracle Support Note 1439822.1 - All Java SE Downloads on MOS – Requires Support Login).

Oracle does not plan to migrate desktops from Java SE 8 to later versions via the auto update feature. This includes the Java Plugin and Java Web Start. Instead of relying on a browser-accessible system JRE, we encourage application developers to use the packaging options introduced with Java SE 9 to repackage and deliver their Java applications as stand-alone applications that include their own custom runtimes.

Oracle will continue to provide Public Updates and auto updates of Java SE 8, until at least the end of December 2020 for Personal Users, and January 2019 for Commercial Users.

Oracle Java SE Support Roadmap*

http://www.oracle.com/technetwork/java/eol-135779.html

Oracle Java SE Support Roadmap*†

Release	GA Date	Premier Support Until** Notification	Extended Support Until**	Sustaining Support**
6	December 2006	December 2015	December 2018	Indefinite
7	July 2011	July 2019	July 2022	Indefinite
8	March 2014	March 2022	March 2025	Indefinite
9 (non-LTS)	September 2017	March 2018	Not Available	Indefinite
10 (18.3^) (non-LTS)	March 2018	September 2018	Not Available	Indefinite
11 (18.9^ LTS)	September 2018***	September 2023	September 2026	Indefinite
12 (19.3^ non-LTS)	March 2019***	September 2019	Not Available	Indefinite

Oracle Java SE Product Releases

Oracle provides Customers with Oracle Premier Support on Oracle Java SE products as described in the Oracle Lifetime Support Policy. For product releases after Java SE 8, Oracle will designate a release, every three years, as a Long-Term-Support (LTS) release. Java SE 11 (18.9 LTS) is the next planned LTS release. For the purposes of Oracle Premier Support, non-LTS releases are considered a cumulative set of implementation enhancements of the most recent LTS release. Once a new feature release is made available, any previous non-LTS release will be considered superseded. For example, Java SE 9 was a non-LTS release and immediately superseded by Java SE 10 (also non-LTS), Java SE 10 in turn is immediately superseded by Java SE 11. Java SE 11 however is an LTS release, and therefore Oracle Customers will receive Oracle Premier Support and periodic update releases, even after Java SE 12 is released.

† Excluding Deployment Technology and JavaFX

Oracle Java SE Support Roadmap*

http://www.oracle.com/technetwork/java/eol-135779.html

Oracle Java SE Support Roadmap^†				
Release	**GA Date**	**Premier Support Until** Notification**	**Extended Support Until****	**Sustaining Support****
6	December 2006	December 2015	December 2018	Indefinite
7	July 2011	July 2019	July 2022	Indefinite
8	March 2014	March 2022	March 2025	Indefinite
9 (non-LTS)	September 2017	March 2018	Not Available	Indefinite
10 (18.3^) (non-LTS)	March 2018	September 2018	Not Available	Indefinite
11 (18.9^ LTS)	September 2018***	September 2023	September 2026	Indefinite
12 (19.3^ non-LTS)	March 2019***	September 2019	Not Available	Indefinite

- **Das Oracle JDK 11 darf nur mit der Java SE Subscription produktiv eingesetzt werden, ansonsten ist es frei einsetzbar nur für Entwicklung, Testbetrieb, Prototyping und für Demozwecke:**

- Oracle JDK and OpenJDK builds from Oracle
  Starting with Java SE 9, in addition to providing Oracle JDK for free under the BCL, Oracle also started providing builds of OpenJDK under an open source license (similar to that of Linux). Oracle is working to make the Oracle JDK and OpenJDK builds from Oracle interchangeable - targeting developers and organizations that do not want commercial support or enterprise management tools. **Beginning with Oracle Java SE 11 (18.9 LTS), the Oracle JDK will continue to be available royalty-free for development, testing, prototyping or demonstrating purposes.** As announced in September 2017, with the OracleJDK and builds of Oracle OpenJDK being interchangeable for releases of Java SE 11 and later, **the Oracle JDK will primarily be for commercial and support customers** and OpenJDK builds from Oracle are for those who do not want commercial support or enterprise management tools.

- **Wer das Oracle JDK nicht einsetzen möchte, kann das äquivalente Oracle OpenJDK verwenden:**

- Java could be used without the Oracle Java SE Subscription with Oracle OpenJDK as an equivalent to the Oracle JDK. The OpenJDK release cadence is about two OpenJDK major versions per year. There would be almost no difference between Oracle JDK and Oracle OpenJDK but the OpenJDK comes without Java engineering support agreement for customers.

Oracle Java SE Support for Deployment Technology & JavaFX

http://www.oracle.com/technetwork/java/eol-135779.html

Support for Deployment Technology and JavaFX*				
Version	**GA Date**	**Java Web Start Support Until**	**Java Plugin (Applets) Support Until**	**JavaFX Support Until**
6	December 2006	October 2017	October 2017	N/A
7	July 2011	October 2017	October 2017	July 2019
8	March 2014	March 2025****	March 2019	March 2022
9 and later		N/A	N/A	N/A

* Oracle Java SE product dates are provided as **examples** to illustrate the support policies. Customers should refer to the Oracle Lifetime Support Policy for the most up-to-date information. Timelines may differ for Oracle Products with a Java SE dependency (My Oracle Support Note 1557737.1 - Support Entitlement for Java SE When Used As Part of Another Oracle Product – Requires Support Login).

** These support timelines apply to Java client and server deployments of Java with the exception of Web Deployment Technology and JavaFX. For more information on those features, see below.

*** LTS designation and dates, as noted in the above example, are subject to change.

**** Or later.

^ Starting with Java SE 10 (18.3), in March 2018, Oracle JDK includes, in the Java SE vendor version string, the year and month of the release as "YY.M".

† Excluding Deployment Technology and JavaFX, which is described in a separate section.

Web Deployment Technology and JavaFX

The Web Deployment Technology bundled with the JRE, consisting of the Java Plugin and Java Web Start has a shorter support lifecycle: only five years of Premier Support. The deployment stack was marked as deprecated and flagged for removal in Java SE 9 and Java SE 10. Oracle Java SE 11 and later versions will not include the Deployment Stack. As Java SE 8 will be the sunset release for the Deployment Stack (given that Java SE 9 and Java SE 10 are non-LTS releases) Oracle extended support of Java Web Start on Java SE 8 until the end of Java SE 8 Extended Support (March 2025). Support for the Java Plugin (Java Applets) remains available until March 2019.

Java SE 8 is the recommended and only supported version of the deployment stack, since versions earlier than Java SE 8 no longer include the deployment stack, while Java SE 9 has reached end of extended support, and Java SE 10 will reach end of extended support in September 2018. The Java SE 8 deployment stack may be used to run Java SE 6, or Java SE 7 applications on Windows platforms. The Java deployment technology will not be supported beyond Java SE 8. See the Oracle Lifetime Support Policy for details.

JavaFX has been Open Sourced and redesigned to be available as a stand-alone library rather than being included with the JDK. Starting with Java SE 11 (18.9 LTS), JavaFX will not be included in the Oracle JDK. Support for JavaFX on Java SE 8 will continue through the Premier Support term (until March 2022).

Oracle Java SE Subscription

- **Oracle Java SE Subscription for Desktops, Servers or Cloud deployments**

  Java SE Subscription is a simple, low-cost monthly subscription that includes Java SE Licensing and Support for use on Desktops, Servers or Cloud deployments, It follows a commonly used model, popular with Linux distributions. The subscription provides access to tested and certified performance, stability, and security updates for Java SE, directly from Oracle. It also includes access to My Oracle Support (MOS) 24x7, support in 27 languages, Java SE 8 Desktop management, monitoring, and deployment features, among other benefits.

- **Java SE Subscription cost**

  Desktop pricing is $2.50 per user per month, or lower with tiered volume discounts. Processor pricing for use on Servers and/or Cloud deployments is $25.00 per month or lower.

- **Java SE Subscription What is the length of terms**

  Standard term is one year, with two and three-year terms available.

- **Java SE Subscription support updates**

  Comprehensive Java SE Support is central to the Java SE Subscription and is provided via My Oracle Support (MOS)

- **Calculator**
  - Metric: Named User Plus, or Processor
  - Term: 1 Year .. 3 Years
  - Quantity: Anzahl

http://www.oracle.com/technetwork/java/javaseproducts/overview/javasesubscriptionfaq-4891443.html

Oracle Java SE Subscription Global Price List

Prices in USA (Dollar)

	Monthly Subscription Price	Subscription Metric	Volume
Java SE Platform Products			
Java SE Desktop Subscription	2.5000	Named User Plus	1-999
	2.0000	Named User Plus	1,000-2,999
	1.7500	Named User Plus	3,000-9,999
	1.5000	Named User Plus	10,000-19,999
	1.2500	Named User Plus	20,000-49,999
		Contact for Details	50,000+
Java SE Subscription	25.0000	Processor	1-99
	23.7500	Processor	100-249
	22.5000	Processor	250-499
	20.0000	Processor	500-999
	17.5000	Processor	1,000-2,999
	15.0000	Processor	3,000-9,999
	12.5000	Processor	10,000-19,999
		Contact for Details	20,000+

http://www.oracle.com/us/corporate/pricing/price-lists/java-se-subscription-pricelist-5028356.pdf

Java Client Roadmap Update (1)
Java Web Start and Pre-Installed Java Runtime Environments

- Java Web Start has been included in the Oracle Java Runtime Environment (JRE) since 2001. It is launched automatically when a Java application using Java Web Start technology is downloaded for the first time. Desktop shortcuts can also launch the application, providing the user with a similar experience to that of a native application.
- Java Web Start has become a migration path for developers as browser vendors continued to restrict plugin support over the past several years.
- Since it is predominantly a desktop technology, Web Start has some limitations. In particular, it requires a standalone JRE to be installed and maintained on the user's desktop.
- However, over the past decade, vendors of the most popular desktop operating systems have emphatically pushed for applications on their platforms to be delivered bundled with integrated, sandboxed runtimes. Increasingly they require desktop applications to be distributed through their own private "app stores."
- The notion of an application being distributed separately from a standalone JRE is, therefore, quickly fading.

Java Client Roadmap Update (2)

Java Web Start and Pre-Installed Java Runtime Environments - Consequently

- Oracle will extend support for Web Start in Java SE 8 from March, 2019, through at least March 2025.
- **Oracle products that have dependencies on Web Start will remain on Java SE 8 and continue with the support timelines as indicated by those products.**
- Oracle will not include Java Web Start in Java SE 11 (18.9 LTS) and later.
- Oracle will begin encouraging application developers and users to transition away from Java Web Start and encourage non-commercial consumers to remove any unused or non-supported Oracle JRE installations from their desktops.
- Developers who deploy desktop applications to individual consumers (eg, games, personal banking, or other B2C applications) will need to transition to other deployment technologies such as the jlink and/or third party packaging and deployment solutions before the end of 2020.
- Application developers who target applications for internal data processing, business, commercial, or production purposes, will either need to seek commercial license with Oracle, or transition to other deployment technologies by January 2019.

Oracle JDK Releases for Java 11 and later

Insights and updates on Java SE and OpenJDK from the Java Platform Group Product Management Team

Oracle JDK Releases for Java 11 and later (1)

Exec Summary

- Starting with Java 11, Oracle will provide JDK releases under the open source GNU General Public License v2, with the Classpath Exception (GPLv2+CPE), and under a commercial license for those using the Oracle JDK as part of an Oracle product or service, or who do not wish to use open source software. This combination of using an open source license and a commercial license replaces the historical "BCL" license, which had a combination of free and paid commercial terms.
- Different builds will be provided for each license, but these builds are functionally identical aside from some cosmetic and packaging differences, described in detail below.

Oracle JDK Releases for Java 11 and later (2)

From the BCL to the GPL

- The Binary Code License for Oracle Java SE technologies ("BCL") has been the primary license for Oracle Java SE technologies for well over a decade. The BCL permits use without license fees under certain conditions. To simplify things going forward, Oracle started providing open source licensed OpenJDK builds as of Java 9, using the same license model as the Linux platform.
- If you are used to getting Oracle Java SE binaries for free, you can simply continue doing so with Oracle's OpenJDK builds available at jdk.java.net.
- If you are used to getting Oracle Java SE binaries as part of a commercial product or service from Oracle, then you can continue to get Oracle JDK releases through My Oracle Support (MOS), and other locations.

Oracle JDK Releases for Java 11 and later (3)

Functionally identical and interchangeable ...

- Oracle's BCL-licensed JDK historically contained "commercial features" that were not available in OpenJDK builds. As promised, however, over the past year Oracle has contributed these features to the OpenJDK Community, including:
  - Java Flight Recorder,
  - Java Mission Control,
  - Application Class-Data Sharing, and
  - ZGC.
- From Java 11 forward, therefore, Oracle JDK builds and OpenJDK builds will be essentially identical.

Oracle JDK Releases for Java 11 and later (4)

... yet with some cosmetic and packaging differences

- There do remain a small number of differences, some intentional and cosmetic, and some simply because more time to discuss with OpenJDK contributors is warranted.
  - Oracle JDK 11 emits a warning when using the -XX:+UnlockCommercialFeatures option, whereas in OpenJDK builds this option results in an error. This option was never part of OpenJDK and it would not make sense to add it now, since there are no commercial features in OpenJDK. This difference remains in order to make it easier for users of Oracle JDK 10 and earlier releases to migrate to Oracle JDK 11 and later.
  - Oracle JDK 11 can be configured to provide usage log data to the "Advanced Management Console" tool, which is a separate commercial Oracle product. We will work with other OpenJDK contributors to discuss how such usage data may be useful in OpenJDK in future releases, if at all. This difference remains primarily to provide a consistent experience to Oracle customers until such decisions are made.

Oracle JDK Releases for Java 11 and later (5)

... yet with some cosmetic and packaging differences

- The ***`javac --release`*** command behaves differently for the Java 9 and Java 10 targets, since in those releases the Oracle JDK contained some additional modules that were not part of corresponding OpenJDK releases:
  - javafx.base
  - javafx.controls
  - javafx.fxml
  - javafx.graphics
  - javafx.media
  - javafx.web
  - java.jnlp
  - jdk.jfr
  - jdk.management.cmm
  - jdk.management.jfr
  - jdk.management.resource
  - jdk.packager.services
  - jdk.snmp
- This difference remains in order to provide a consistent experience for specific kinds of legacy use. These modules are either now available separately as part of OpenJFX, are now in both OpenJDK and the Oracle JDK because they were commercial features which Oracle contributed to OpenJDK (e.g., Flight Recorder), or were removed from Oracle JDK 11 (e.g., JNLP).

Oracle JDK Releases for Java 11 and later (6)

... yet with some cosmetic and packaging differences

- The output of the ***`java --version`*** and ***`java -fullversion`*** commands will distinguish Oracle JDK builds from OpenJDK builds, so that support teams can diagnose any issues that may exist. Specifically, running java --version with an Oracle JDK 11 build results in:
  - java 11 2018-09-25
  - Java(TM) SE Runtime Environment 18.9 (build 11+28)
  - Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11+28, mixed mode)
- And for an OpenJDK 11 build:
  - openjdk version "11" 2018-09-25
  - OpenJDK Runtime Environment 18.9 (build 11+28)
  - OpenJDK 64-Bit Server VM 18.9 (build 11+28, mixed mode)

Oracle JDK Releases for Java 11 and later (7)

... yet with some cosmetic and packaging differences

- The Oracle JDK has always required third party cryptographic providers to be signed by a known certificate. The cryptography framework in OpenJDK has an open cryptographic interface, meaning it does not restrict which providers can be used. Oracle JDK 11 will continue to require a valid signature, and Oracle OpenJDK builds will continue to allow the use of either a valid signature or unsigned third party crypto provider.
- Oracle JDK 11 will continue to include installers, branding and JRE packaging for an experience consistent with legacy desktop uses. Oracle OpenJDK builds are currently available as zip and tar.gz files, while alternative distribution formats are being considered.

Oracle JDK Releases for Java 11 and later (8)

What should we call them?

- Ideally, we would simply refer to all Oracle JDK builds as the "Oracle JDK," either under the GPL or the commercial license depending on your situation. However, for historical reasons while the small remaining differences exist, we will refer to them separately as Oracle's OpenJDK builds, and the Oracle JDK.

JDK 11 Migration Guide

Migrating to Oracle JDK 11 - Migration Guide (1)

https://docs.oracle.com/en/java/javase/11/migrate/index.html

- **Migrating to JDK 11**
  - The purpose of this guide is to help you identify potential issues and give you suggestions on how to proceed as you migrate your existing Java application from JDK 8, or earlier version of the JDK, to JDK 11.
  - This guide is not significantly different than the JDK 10 Migration Guide.
  - Every new Java SE release introduces some binary, source, and behavioral incompatibilities with previous releases.
  - The modularization of the Java SE Platform that happened in JDK 9 brought many benefits, but also many changes.
  - Code that uses only official Java SE Platform APIs and supported JDK-specific APIs should continue to work without change.
  - Code that uses JDK-internal APIs should continue to run but should be migrated to use external APIs.

Migrating to Oracle JDK 11 - Migration Guide (2)

https://docs.oracle.com/en/java/javase/11/migrate/index.html

- **Prepare for Migration**
  - Download JDK 11
  - Run Your Program Before Recompiling
  - Update Third-Party Libraries
  - Compile Your Application if Needed
  - Run jdeps on Your Code

Migrating to Oracle JDK 11 - Migration Guide (3)

https://docs.oracle.com/en/java/javase/11/migrate/index.html

- **Migrating from JDK 8 to later JDK releases**
  - New Version-String Scheme
  - Understanding Runtime Access Warnings
  - Changes to the Installed JDK/JRE Image
  - Removed or Changed APIs
  - Deployment
  - Security Updates in JDK 9
  - Changes to Garbage Collection
  - Removed Tools and Components
  - Removed macOS-Specific Features

JDK 11 Features

JDK 11 Features – JEP's included

- 181: Nest-Based Access Control
- 309: Dynamic Class-File Constants
- 315: Improve Aarch64 Intrinsics
- 318: Epsilon: A No-Op Garbage Collector
- 320: Remove the Java EE and CORBA Modules
- 321: HTTP Client (Standard)
- 323: Local-Variable Syntax for Lambda Parameters
- 324: Key Agreement with Curve25519 and Curve448
- 327: Unicode 10
- 328: Flight Recorder
- 329: ChaCha20 and Poly1305 Cryptographic Algorithms
- 330: Launch Single-File Source-Code Programs
- 331: Low-Overhead Heap Profiling
- 332: Transport Layer Security (TLS) 1.3
- 333: ZGC: A Scalable Low-Latency Garbage Collector (Experimental)
- 335: Deprecate the Nashorn JavaScript Engine
- 336: Deprecate the Pack200 Tools and API

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 181 Nest-Based Access Control

- Summary: Introduce *nests*, an access-control context that aligns with the existing notion of nested types in the Java programming language. Nests allow classes that are logically part of the same code entity, but which are compiled to distinct class files, to access each other's private members without the need for compilers to insert accessibility-broadening bridge methods.

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 309 Dynamic Class-File Constants

- Summary: Extend the Java class-file format to support a new constant-pool form, CONSTANT_Dynamic. Loading a CONSTANT_Dynamic will delegate creation to a bootstrap method, just as linking an invokedynamic call site delegates linkage to a bootstrap method.

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 315 Improve Aarch64 Intrinsics

- Summary: Improve the existing string and array intrinsics, and implement new intrinsics for the java.lang.Math sin, cos and log functions, on AArch64 processors.

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 318 Epsilon: A No-Op Garbage Collector

- Summary: Develop a GC that handles memory allocation but does not implement any actual memory reclamation mechanism. Once the available Java heap is exhausted, the JVM will shut down.

- Goals

Provide a completely passive GC implementation with a bounded allocation limit and the lowest latency overhead possible, at the expense of memory footprint and memory throughput. A successful implementation is an isolated code change, does not touch other GCs, and makes minimal changes in the rest of JVM.

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 320 Remove the Java EE and CORBA Modules

- Summary: Remove the Java EE and CORBA modules from the Java SE Platform and the JDK. These modules were deprecated in Java SE 9 with the declared intent to remove them in a future release.

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 321 HTTP Client (Standard)

- Summary: Standardize the incubated HTTP Client API introduced in JDK 9, via JEP 110, and updated in JDK 10.

- Goals

In addition to the goals of JEP 110, this JEP will:

- Take into account feedback received on the incubated API,
- Provide a standardized API, in the java.net.http package, based upon the incubated API, and
- Remove the incubated API.

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 323 Local-Variable Syntax for Lambda Parameters

- Summary: Allow var to be used when declaring the formal parameters of implicitly typed lambda expressions.

- Goals

Align the syntax of a formal parameter declaration in an implicitly typed lambda expression with the syntax of a local variable declaration.

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 324 Key Agreement with Curve25519 and Curve448

- Summary: Implement key agreement using Curve25519 and Curve448 as described in RFC 7748.

- Goals

RFC 7748 defines a key agreement scheme that is more efficient and secure than the existing elliptic curve Diffie-Hellman (ECDH) scheme. The primary goal of this JEP is an API and an implementation for this standard. Additional implementation goals are:

1. Develop a platform-independent, all-Java implementation with better performance than the existing ECC (native C) code at the same security strength.
2. Ensure that the timing is independent of secrets, assuming the platform performs 64-bit integer addition/multiplication in constant time. In addition, the implementation will not branch on secrets. These properties are valuable for preventing side-channel attacks.

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 327 Unicode 10

- Summary: Upgrade existing platform APIs to support version 10.0 of the Unicode Standard.

- Goals

Support the latest version of Unicode, mainly in the following classes:

- Character and String in the java.lang package,
- NumericShaper in the java.awt.font package, and
- Bidi, BreakIterator, and Normalizer in the java.text package.

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 328 Flight Recorder

- Summary: Provide a low-overhead data collection framework for troubleshooting Java applications and the HotSpot JVM.

- Goals

- Provide APIs for producing and consuming data as events
- Provide a buffer mechanism and a binary data format
- Allow the configuration and filtering of events
- Provide events for the OS, the HotSpot JVM, and the JDK libraries

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 329 ChaCha20 and Poly1305 Cryptographic Algorithms

- Summary: Implement the ChaCha20 and ChaCha20-Poly1305 ciphers as specified in RFC 7539. ChaCha20 is a relatively new stream cipher that can replace the older, insecure RC4 stream cipher.

- Goals

- Provide ChaCha20 and ChaCha20-Poly1305 Cipher implementations. These algorithms will be implemented in the SunJCE provider.
- Provide a KeyGenerator implementation that creates keys suitable for ChaCha20 and ChaCha20-Poly1305 algorithms.
- Provide an AlgorithmParameters implementation for use with the ChaCha20-Poly1305 algorithm.

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 330 Launch Single-File Source-Code Programs

- Summary: Enhance the java launcher to run a program supplied as a single file of Java source code, including usage from within a script by means of "shebang" files and related techniques.

- Non-Goals

It is not a goal to change either the Java Language Specification (JLS) or javac to accommodate shebang files. Likewise, it is not a goal to evolve the Java language into a general purpose scripting language.

It is not a goal of this JEP to change the Java Language Specification to accommodate simpler ways of writing small programs, such as eliminating the need for the standard public static void main(String[] args) method. However, it is expected that any such changes to the Java language will be usable in conjunction with this feature.

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 331 Low-Overhead Heap Profiling

- Summary: Provide a low-overhead way of sampling Java heap allocations, accessible via JVMTI.

- Goals

Provide a way to get information about Java object heap allocations from the JVM that:

- Is low-overhead enough to be enabled by default continuously,
- Is accessible via a well-defined, programmatic interface,
- Can sample all allocations (i.e., is not limited to allocations that are in one particular heap region or that were allocated in one particular way),
- Can be defined in an implementation-independent way (i.e., without relying on any particular GC algorithm or VM implementation), and
- Can give information about both live and dead Java objects.

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 332 Transport Layer Security (TLS) 1.3

- Summary: Implement version 1.3 of the Transport Layer Security (TLS) Protocol RFC 8446.

- Non-Goals

It is not a goal to support version 1.3 of the Datagram Transport Layer Security (DTLS) Protocol. It is also not a goal to support every feature of TLS 1.3; see the Description section for more details on what will be implemented.

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 333 ZGC A Scalable Low-Latency Garbage Collector (Experimental)

- Summary: The Z Garbage Collector, also known as ZGC, is a scalable low-latency garbage collector.
  - Scalable low latency garbage collector capable of handling heaps ranging from gigabytes to terabytes in size, with GC pause times not exceeding 10ms

- Goals

- GC pause times should not exceed 10ms
- Handle heaps ranging from relatively small (a few hundreds of megabytes) to very large (many terabytes) in size
- No more than 15% application throughput reduction compared to using G1
- Lay a foundation for future GC features and optimizations leveraging colored pointers and load barriers
- Initially supported platform: Linux/x64

We have strong ambitions to meet these goals for a large set of relevant workloads. At the same time, we want to acknowledge that we don't see these goals as hard requirements for every conceivable workload.

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 335 Deprecate the Nashorn JavaScript Engine

- Summary: Deprecate the Nashorn JavaScript script engine and APIs, and the jjs tool, with the intent to remove them in a future release.

- Non-Goals

This deprecation does not affect, in any way, the javax.script API.

http://openjdk.java.net/projects/jdk/11/

JDK 11 – JEP 336 Deprecate the Pack200 Tools and API

- Summary: Deprecate the pack200 and unpack200 tools, and the Pack200 API in java.util.jar.

- Description

Three types in the java.base module will be terminally deprecated, that is, annotated with @Deprecated(forRemoval=true):

- java.util.jar.Pack200
- java.util.jar.Pack200.Packer
- java.util.jar.Pack200.Unpacker

The jdk.pack module, which contains the pack200 and unpack200 tools, will also be terminally deprecated.

Running pack200 or unpack200 will display a warning about the planned removal of the tool. Running jar -c with the sub-option n (to normalize the archive) will display a warning about the planned removal of the sub-option. The documentation for all three tools will indicate the deprecation and planned removal.

A separate JEP will be filed for the actual removal of the types and module in a future JDK feature release.

http://openjdk.java.net/projects/jdk/11/

JDK 11 – Summary

- The Java platform development on OpenJDK is becoming more open
  - Contributing all commercial features (zGC, JFR, AppCDS, etc.)
  - GPL + CPE build
- The cloud is demanding a faster pace and continuous delivery
  - Uptake new Java releases every 6-month
- Let‘s continue to innovate and advance the Java SE Platform on OpenJDK together!
- Join and become an OpenJDK contributor
  - https://openjdk.java.net

Jakarta EE MicroProfile Helidon

CREATE THE FUTURE

Peter Doschkinow
Principal Sales Consultant
Java Architect

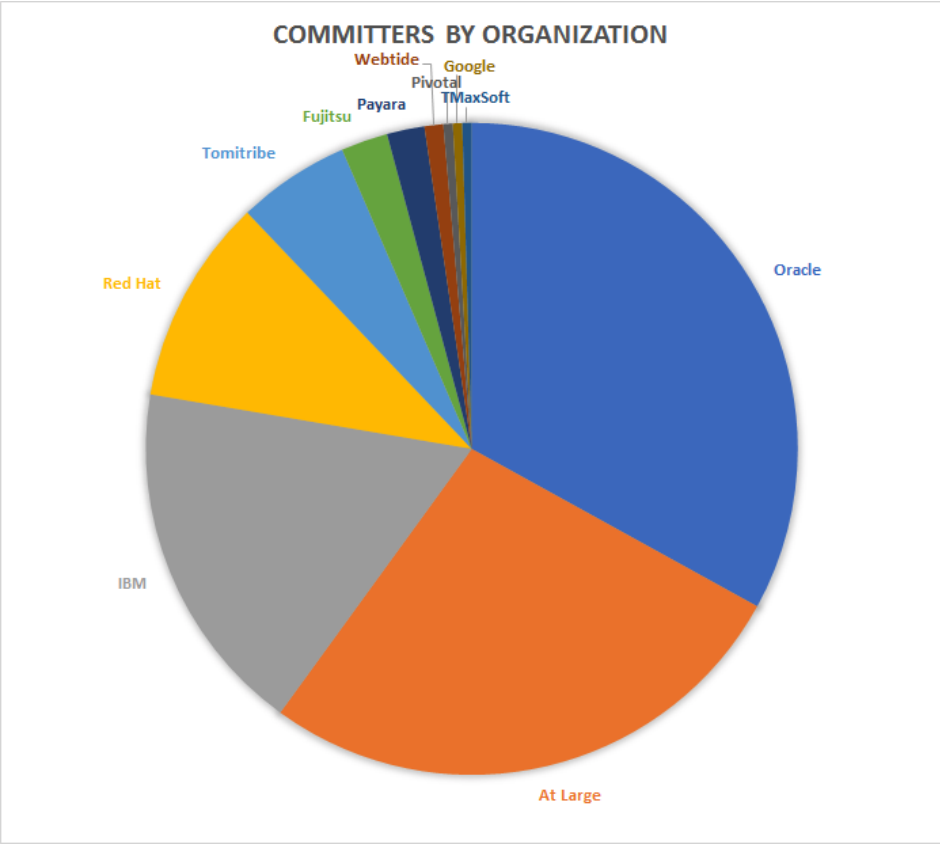
Java™ ORACLE

Jakarta EE

https://jakarta.ee/

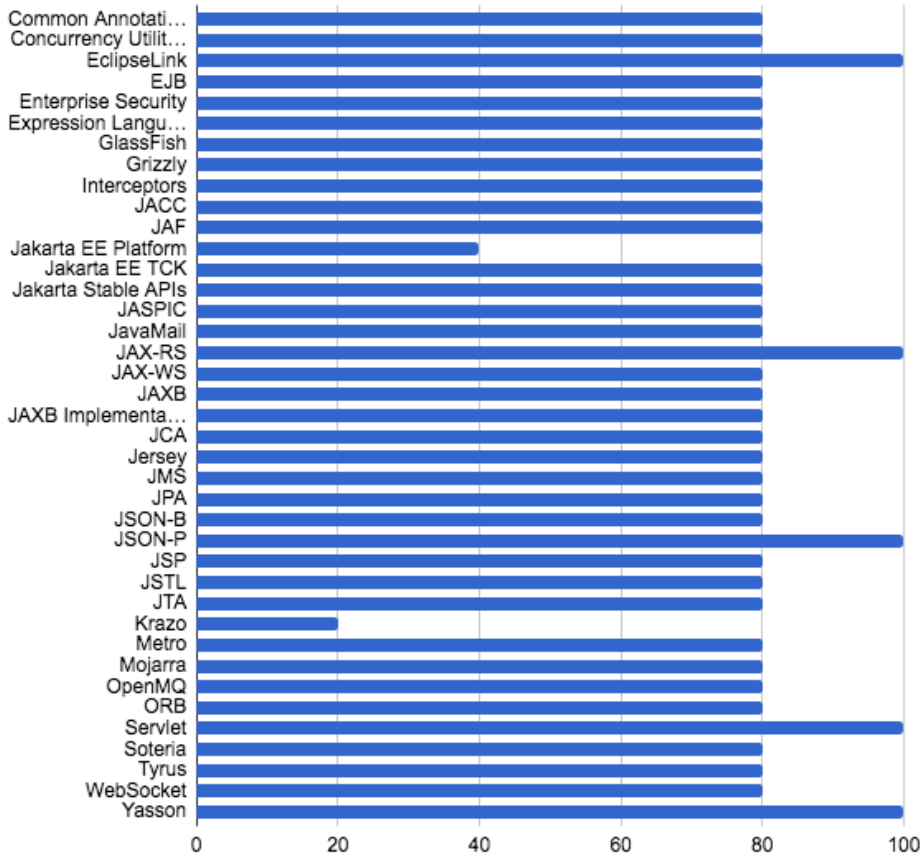
- The new name for Java EE
- Top level Eclipse project EE4J
- EE4J is the new home for Java EE and the home of Jakarta EE
  - Specification
  - Implementation
  - Test Compatibility Kits (TCK)
  - Documentation
- More flexible licensing – EPL 2.0 with secondary GPL 2.0 with CPE
  - This means EE4J results are GPL 2 compatible and when combined with GPL 2 artefacts could be provided with a GPL 2 license

Jakarta EE Contributions



Jakarta EE Status

- 20% Project Proposal has been posted for community review
- 40% Project committers and resources have been provisioned
- 60% Initial Contribution provided to the Eclipse IP Team
- 80% Initial Contribution Pushed to Git Repository
- 100% Project has engaged in its first Release Review



Project Helidon

https://helidon.io

- Oracle Java Microservices Framework
- Open source project
- A set of Java libraries for developing microservices
- Runs as a stand alone JVM. Application server is not required
- Use common tooling (Java SE, Maven, Docker, Kubernetes, etc)
- Implements Eclipse MicroProfile
- Two programming models
- Built-in integrations to facilitate using Oracle Cloud Services

Eclipse MP (Microprofile)

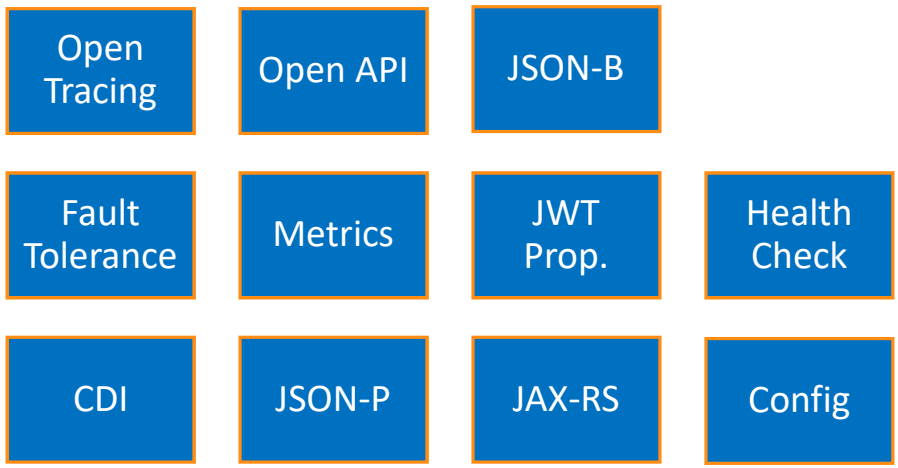
- Platform to optimize Enterprise Java for a microservices architecture
- Familiar to Java EE developers
- Key Java EE APIs + new MicroProfile APIs

Helidon SE (Standard Edition)

- Lightweight microframework
- Embedded Reactive Webserver to serve dynamic and static content
- Security support for different pluggable providers
- Built-in integrations to facilitate using Oracle Cloud Services

Eclipse MicroProfile 2.0

- Announced in 2016
- Leverage on Java EE knowledge to adopt cloud native architecture
- Initiated by RedHat, IBM, Tomitribe and Payara
- Supported by Oracle
- Key Java EE APIs + new MicroProfile APIs



Helidon SE

Reactive WebServer

- Simple functional routing model with reactive Flow API
- Built on Netty
- OpenTracing and Prometheus Metrics support
- Static content support

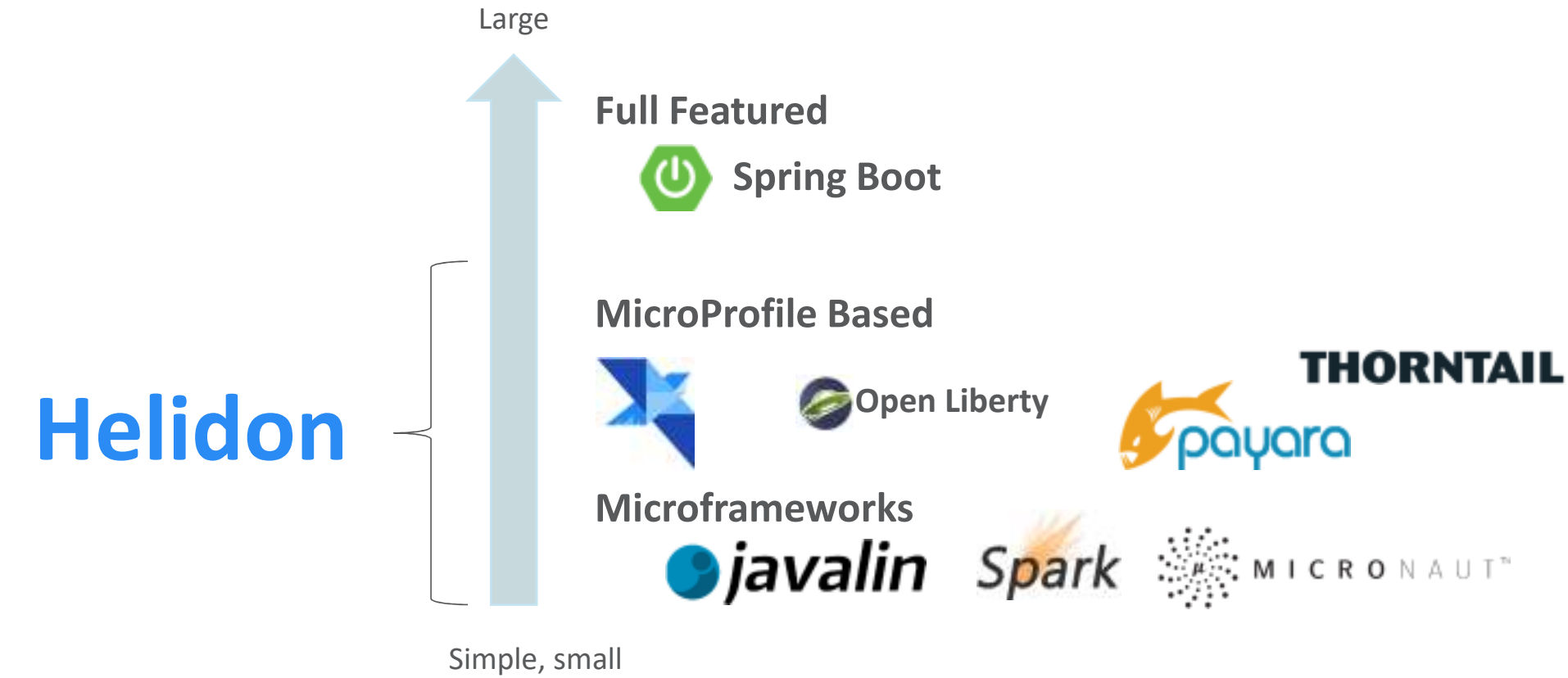
Config

- Flexible, typed config model
- Multiple data sources
- Hierarchical model
- Dynamic updates
- SPI

Security

- Authentication
- Roles and Authorization
- HTTP Signatures
- IDCS integration

Java Microservice Frameworks



Helidon Value

- Simple, functional, lightweight microservice framework
- Embedded fast Reactive WebServer, no Application server required
- Choice of programming models SE or MicroProfile
- Docker and Kubernetes support
- Oracle implementation of Eclipse MicroProfile open-source community specification
- Extensions for integration with Oracle Cloud services
- Transition path for Java (EE) customers looking for next Java Framework to develop cloud native applications

Danke!

Wolfgang.Weigend@oracle.com

Peter.Doschkinow@oracle.com

Informationsquellen

@OracleBUDB

MEET THE EXPERTS

Oracle Experten
beantworten Ihre Fragen

Oracle Stand im 3. OG

Oracle Espressi
(tinyurl.com/oespresso)

Oracle Cloud Security
Autonomous Data Warehouse Service
Oracle Cloud Access Security Broker
Oracle Unterstützung für agile Softwareentwicklung
Oracle Blockchain Cloud Service
Oracle NoSQL Datenbank

ORACLE BLOGS

Blogs

Core Tec & Cloud Technologies Deutschland
(blogs.oracle.com/cloudtec-de)

Deutschsprachiger Datenbank & Cloud Technologie Blog
(blogs.oracle.com/coretec)

Oracle Technology Monthly
(tinyurl.com/oratech-monthly)

Dojos
(tinyurl.com/dojos-online)

Absicherung einer Oracle Datenbank
Oracle Database 12.2 Spezial
Oracle Database (In-)Memory-Technologien
Arbeiten mit JSON und Oracle 12c
Private Cloud mit Cloud Control
Multitenant Option
: