Java9 - Features abseits von Jigsaw und JShell

BED-Con 2017

22. September 2017

innoQ

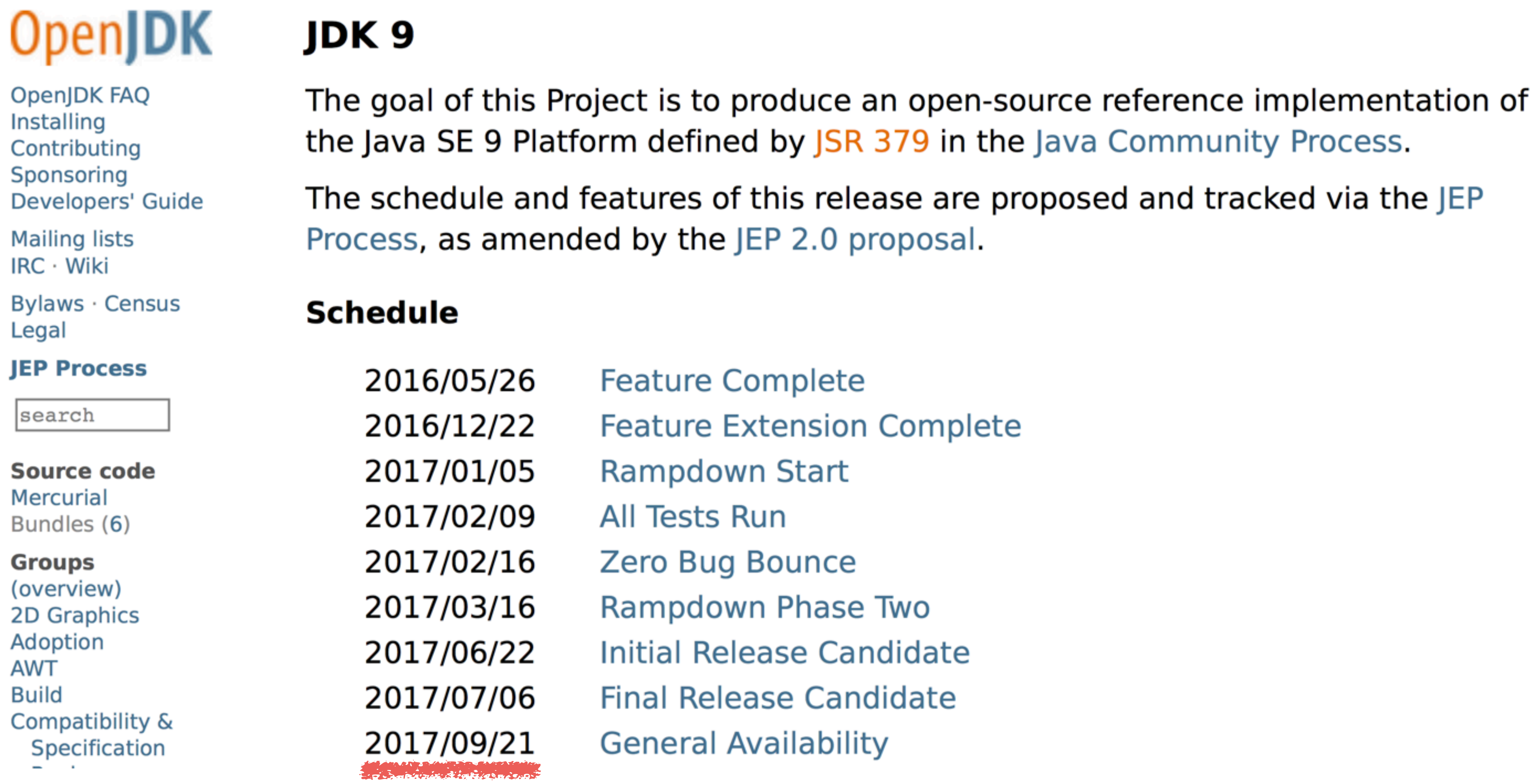
Michael Vitz

Senior Consultant @ innoQ

michael.vitz@innoq.com

@michaelvitz

Zeitplan



OpenJDK

OpenJDK FAQ
Installing
Contributing
Sponsoring
Developers' Guide

Mailing lists
IRC · Wiki

Bylaws · Census
Legal

JEP Process

search

Source code
Mercurial
Bundles (6)

Groups
(overview)
2D Graphics
Adoption
AWT
Build
Compatibility &
Specification

JDK 9

The goal of this Project is to produce an open-source reference implementation of the Java SE 9 Platform defined by JSR 379 in the Java Community Process.

The schedule and features of this release are proposed and tracked via the JEP Process, as amended by the JEP 2.0 proposal.

Schedule

2016/05/26	Feature Complete
2016/12/22	Feature Extension Complete
2017/01/05	Rampdown Start
2017/02/09	All Tests Run
2017/02/16	Zero Bug Bounce
2017/03/16	Rampdown Phase Two
2017/06/22	Initial Release Candidate
2017/07/06	Final Release Candidate
2017/09/21	General Availability

Features

102: Process API Updates
110: HTTP 2 Client
143: Improve Contended Locking
158: Unified JVM Logging
165: Compiler Control
193: Variable Handles
197: Segmented Code Cache
199: Smart Java Compilation, Phase Two
200: The Modular JDK
201: Modular Source Code
211: Elide Deprecation Warnings on Import Statements
212: Resolve Lint and Doclint Warnings
213: Milling Project Coin
214: Remove GC Combinations Deprecated in JDK 8
215: Tiered Attribution for javac
216: Process Import Statements Correctly
217: Annotations Pipeline 2.0
219: Datagram Transport Layer Security (DTLS)
220: Modular Run-Time Images
221: Simplified Doclet API
222: jshell: The Java Shell (Read-Eval-Print Loop)
223: New Version-String Scheme
224: HTML5 Javadoc
225: Javadoc Search
226: UTF-8 Property Files
227: Unicode 7.0
228: Add More Diagnostic Commands
229: Create PKCS12 Keystores by Default
231: Remove Launch-Time JRE Version Selection
232: Improve Secure Application Performance
233: Generate Run-Time Compiler Tests Automatically
235: Test Class-File Attributes Generated by javac
236: Parser API for Nashorn
237: Linux/AArch64 Port
238: Multi-Release JAR Files
240: Remove the JVM TI hprof Agent
241: Remove the jhat Tool
243: Java-Level JVM Compiler Interface
244: TLS Application-Layer Protocol Negotiation Extension
245: Validate JVM Command-Line Flag Arguments
246: Leverage CPU Instructions for GHASH and RSA
247: Compile for Older Platform Versions
248: Make G1 the Default Garbage Collector
249: OCSP Stapling for TLS
250: Store Interned Strings in CDS Archives
251: Multi-Resolution Images
252: Use CLDR Locale Data by Default
253: Prepare JavaFX UI Controls & CSS APIs for Modularization
254: Compact Strings
255: Merge Selected Xerces 2.11.0 Updates into JAXP
256: BeanInfo Annotations
257: Update JavaFX/Media to Newer Version of GStreamer
258: HarfBuzz Font-Layout Engine
259: Stack-Walking API
260: Encapsulate Most Internal APIs
261: Module System
262: TIFF Image I/O
263: HiDPI Graphics on Windows and Linux
264: Platform Logging API and Service
265: Marlin Graphics Renderer
266: More Concurrency Updates
267: Unicode 8.0
268: XML Catalogs
269: Convenience Factory Methods for Collections
270: Reserved Stack Areas for Critical Sections
271: Unified GC Logging
272: Platform-Specific Desktop Features
273: DRBG-Based SecureRandom Implementations
274: Enhanced Method Handles
275: Modular Java Application Packaging
276: Dynamic Linking of Language-Defined Object Models
277: Enhanced Deprecation
278: Additional Tests for Humongous Objects in G1
279: Improve Test-Failure Troubleshooting
280: Indify String Concatenation
281: HotSpot C++ Unit-Test Framework
282: jlink: The Java Linker
283: Enable GTK 3 on Linux
284: New HotSpot Build System
285: Spin-Wait Hints
287: SHA-3 Hash Algorithms
288: Disable SHA-1 Certificates
289: Deprecate the Applet API
290: Filter Incoming Serialization Data
291: Deprecate the Concurrent Mark Sweep (CMS) Garbag
292: Implement Selected ECMAScript 6 Features in Nashor
294: Linux/s390x Port
295: Ahead-of-Time Compilation
297: Unified arm32/arm64 Port
298: Remove Demos and Samples
299: Reorganize Documentation

JEP 102
Process API Updates

```
ProcessHandle.current().getPid()
```

```java
ProcessHandle
        .allProcesses()
        .map(ProcessHandle::getPid)
        .forEach(System.out::println);
```

```
ProcessHandle
    .allProcesses()
    .filter(p -> !p.parent().isPresent())
    .map(p -> p.info())
    .forEach(System.out::println);
```

```
ProcessHandle.current().destroy();
```

JEP 213
Milling Project Coin

@SafeVargs an privaten Instanzmethoden

```
public static void execute(Connection con)
          throws SQLException {
     try (con) {
          // ...
     }
}
```

```
private static Comparator<MyClass> C =
    new Comparator<>() {
        @Override
        public int compare(
                MyClass o1,
                MyClass o2) {
            return 0;
        }
};
```

```java
public interface Lambda {
    public int a(int _);
    public default void t(Lambda l) {
        t(_ -> 0);
    }
}
```

```
public interface Foo {
    private String bar() {
        return "FooBar";
    }
}
```

JEP 223
New Version-String Scheme

Versionsnummer

- [1-9][0-9]*((\.0)*\.[1-9][0-9]*)*
- $MAJOR.$MINOR.$SECURITY

Versionsstring

- $VNUM(-$PRE)?\+$BUILD(-$OPT)?
- $VNUM-$PRE(-$OPT)?
- $VNUM(+-$OPT)?

9.0.0

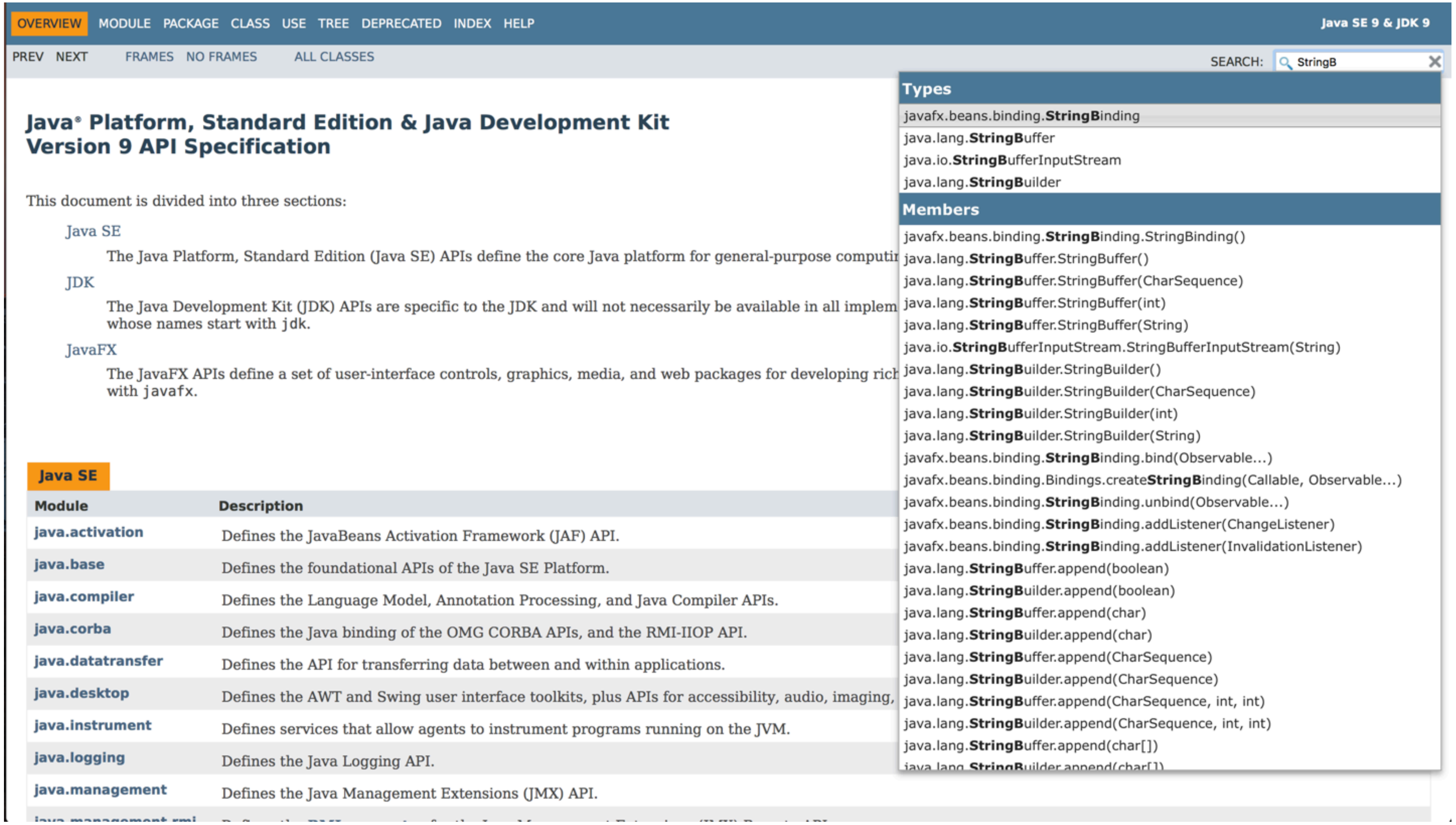
```java
final Runtime.Version version = Runtime.version();
System.out.println(version.toString());
System.out.println(version.major());
System.out.println(version.minor());
System.out.println(version.security());


final Runtime.Version newVersion =
    Runtime.Version.parse("9.1");

System.out.println(newVersion.compareTo(version));
```

```
Name                            Syntax
------------------------------  --------------
java.version                    $VNUM(\-$PRE)?
java.runtime.version            $VSTR
java.vm.version                 $VSTR
java.specification.version      $VNUM
java.vm.specification.version   $VNUM
```

JEP 225
Javadoc Search


OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
Java SE 9 & JDK 9
PREV NEXT FRAMES NO FRAMES ALL CLASSES
SEARCH: StringB

Java® Platform, Standard Edition & Java Development Kit Version 9 API Specification

This document is divided into three sections:

Java SE
: The Java Platform, Standard Edition (Java SE) APIs define the core Java platform for general-purpose computi

JDK
: The Java Development Kit (JDK) APIs are specific to the JDK and will not necessarily be available in all implem whose names start with `jdk`.

JavaFX
: The JavaFX APIs define a set of user-interface controls, graphics, media, and web packages for developing rich with `javafx`.

Java SE

Module	Description
java.activation	Defines the JavaBeans Activation Framework (JAF) API.
java.base	Defines the foundational APIs of the Java SE Platform.
java.compiler	Defines the Language Model, Annotation Processing, and Java Compiler APIs.
java.corba	Defines the Java binding of the OMG CORBA APIs, and the RMI-IIOP API.
java.datatransfer	Defines the API for transferring data between and within applications.
java.desktop	Defines the AWT and Swing user interface toolkits, plus APIs for accessibility, audio, imaging,
java.instrument	Defines services that allow agents to instrument programs running on the JVM.
java.logging	Defines the Java Logging API.
java.management	Defines the Java Management Extensions (JMX) API.

Types

javafx.beans.binding.**StringB**inding
java.lang.**StringB**uffer
java.io.**StringB**ufferInputStream
java.lang.**StringB**uilder

Members

javafx.beans.binding.**StringB**inding.StringBinding()
java.lang.**StringB**uffer.StringBuffer()
java.lang.**StringB**uffer.StringBuffer(CharSequence)
java.lang.**StringB**uffer.StringBuffer(int)
java.lang.**StringB**uffer.StringBuffer(String)
java.io.**StringB**ufferInputStream.StringBufferInputStream(String)
java.lang.**StringB**uilder.StringBuilder()
java.lang.**StringB**uilder.StringBuilder(CharSequence)
java.lang.**StringB**uilder.StringBuilder(int)
java.lang.**StringB**uilder.StringBuilder(String)
javafx.beans.binding.**StringB**inding.bind(Observable...)
javafx.beans.binding.Bindings.create**StringB**inding(Callable, Observable...)
javafx.beans.binding.**StringB**inding.unbind(Observable...)
javafx.beans.binding.**StringB**inding.addListener(ChangeListener)
javafx.beans.binding.**StringB**inding.addListener(InvalidationListener)
java.lang.**StringB**uffer.append(boolean)
java.lang.**StringB**uilder.append(boolean)
java.lang.**StringB**uffer.append(char)
java.lang.**StringB**uilder.append(char)
java.lang.**StringB**uffer.append(CharSequence)
java.lang.**StringB**uilder.append(CharSequence)
java.lang.**StringB**uffer.append(CharSequence, int, int)
java.lang.**StringB**uilder.append(CharSequence, int, int)
java.lang.**StringB**uffer.append(char[])
java.lang.**StringB**uilder.append(char[])

JEP 238
Multi-Release JAR Files

```
jar root
  - A.class
  - B.class
  - C.class
  - META-INF
    - versions
      - 8
        - A.class
        - B.class
      - 9
        - A.class
```

https://github.com/hboutemy/maven-jep238

JEP 247
Compile for Older Platform Versions

`--release`

JEP 269
Convenience Factory Methods for Collections

```java
List.of(1, 2, 3);

Set.of(1, 2, 1);

Map.of("foo", "bar");
Map.ofEntries(Map.entry("foo", "bar"));
```

JEP 277

Enhanced Deprecation

```
@Documented
@Retention(RetentionPolicy.RUNTIME)
@Target(value={CONSTRUCTOR, FIELD, LOCAL_VARIABLE, METHOD, PACKAGE, MODULE, PARAMETER,
TYPE})
public @interface Deprecated {
    /**
     * Returns the version in which the annotated element became deprecated.
     * The version string is in the same format and namespace as the value of
     * the {@code @since} javadoc tag. The default value is the empty
     * string.
     *
     * @return the version string
     * @since 9
     */
    String since() default "";

    /**
     * Indicates whether the annotated element is subject to removal in a
     * future version. The default value is {@code false}.
     *
     * @return whether the element is subject to removal
     * @since 9
     */
    boolean forRemoval() default false;
}
```

java.util.Stream
Erweiterungen

```
    /**
     * Returns, if this stream is ordered, a stream consisting of the longest
     * prefix of elements taken from this stream that match the given predicate.
     * Otherwise returns, if this stream is unordered, a stream consisting of a
     * subset of elements taken from this stream that match the given predicate.
     *
…
     *
     * <p>This is a <a href="package-summary.html#StreamOps">short-circuiting
     * stateful intermediate operation</a>.
     *
…
     *
     * @param predicate a <a href="package-summary.html#NonInterference">non-
interfering</a>,
     *                  <a href="package-summary.html#Statelessness">stateless</a>
     *                  predicate to apply to elements to determine the longest
     *                  prefix of elements.
     * @return the new stream
     * @since 9
     */
    default Stream<T> takeWhile(Predicate<? super T> predicate) {
```

```
    /**
     * Returns, if this stream is ordered, a stream consisting of the remaining
     * elements of this stream after dropping the longest prefix of elements
     * that match the given predicate.  Otherwise returns, if this stream is
     * unordered, a stream consisting of the remaining elements of this stream
     * after dropping a subset of elements that match the given predicate.
…
     * <p>Independent of whether this stream is ordered or unordered if all
     * elements of this stream match the given predicate then this operation
     * drops all elements (the result is an empty stream), or if no elements of
     * the stream match the given predicate then no elements are dropped (the
     * result is the same as the input).
     *
     * <p>This is a <a href="package-summary.html#StreamOps">stateful
     * intermediate operation</a>.
…
     * @param predicate a <a href="package-summary.html#NonInterference">non-
interfering</a>,
     *                  <a href="package-summary.html#Statelessness">stateless</a>
     *                  predicate to apply to elements to determine the longest
     *                  prefix of elements.
     * @return the new stream
     * @since 9
     */
    default Stream<T> dropWhile(Predicate<? super T> predicate) {
```

```
    /**
     * Returns a sequential ordered {@code Stream} produced by iterative
     * application of the given {@code next} function to an initial element,
     * conditioned on satisfying the given {@code hasNext} predicate.  The
     * stream terminates as soon as the {@code hasNext} predicate returns false.
…
     * <p>The resulting sequence may be empty if the {@code hasNext} predicate
     * does not hold on the seed value.  Otherwise the first element will be the
     * supplied {@code seed} value, the next element (if present) will be the
     * result of applying the {@code next} function to the {@code seed} value,
     * and so on iteratively until the {@code hasNext} predicate indicates that
     * the stream should terminate.
…
     * @param <T> the type of stream elements
     * @param seed the initial element
     * @param hasNext a predicate to apply to elements to determine when the
     *                stream must terminate.
     * @param next a function to be applied to the previous element to produce
     *             a new element
     * @return a new sequential {@code Stream}
     * @since 9
     */
    public static<T> Stream<T> iterate(T seed, Predicate<? super T> hasNext,
        UnaryOperator<T> next) {
```

```java
    /**
     * Returns a sequential {@code Stream} containing a single element, if
     * non-null, otherwise returns an empty {@code Stream}.
     *
     * @param t the single element
     * @param <T> the type of stream elements
     * @return a stream with a single element if the specified element
     *         is non-null, otherwise an empty stream
     * @since 9
     */
    public static<T> Stream<T> ofNullable(T t) {
        return t == null ? Stream.empty()
                         : StreamSupport.stream(new Streams.StreamBuilderImpl<>(t),
false);
    }
```

java.util.Optional
Erweiterungen

```java
/**
 * If a value is present, returns a sequential {@link Stream} containing
 * only that value, otherwise returns an empty {@code Stream}.
 *
 * @apiNote
 * This method can be used to transform a {@code Stream} of optional
 * elements to a {@code Stream} of present value elements:
 * <pre>{@code
 *     Stream<Optional<T>> os = ..
 *     Stream<T> s = os.flatMap(Optional::stream)
 * }</pre>
 *
 * @return the optional value as a {@code Stream}
 * @since 9
 */
public Stream<T> stream() {
    if (!isPresent()) {
        return Stream.empty();
    } else {
        return Stream.of(value);
    }
}
```

```java
/**
 * If a value is present, returns an {@code Optional} describing the value,
 * otherwise returns an {@code Optional} produced by the supplying function.
 *
 * @param supplier the supplying function that produces an {@code Optional}
 *        to be returned
 * @return returns an {@code Optional} describing the value of this
 *         {@code Optional}, if a value is present, otherwise an
 *         {@code Optional} produced by the supplying function.
 * @throws NullPointerException if the supplying function is {@code null} or
 *         produces a {@code null} result
 * @since 9
 */
public Optional<T> or(Supplier<? extends Optional<? extends T>> supplier) {
    Objects.requireNonNull(supplier);
    if (isPresent()) {
        return this;
    } else {
        @SuppressWarnings("unchecked")
        Optional<T> r = (Optional<T>) supplier.get();
        return Objects.requireNonNull(r);
    }
}
```

```java
/**
 * If a value is present, performs the given action with the value,
 * otherwise performs the given empty-based action.
 *
 * @param action the action to be performed, if a value is present
 * @param emptyAction the empty-based action to be performed, if no value is
 *        present
 * @throws NullPointerException if a value is present and the given action
 *         is {@code null}, or no value is present and the given empty-based
 *         action is {@code null}.
 * @since 9
 */
public void ifPresentOrElse(Consumer<? super T> action, Runnable emptyAction) {
    if (value != null) {
        action.accept(value);
    } else {
        emptyAction.run();
    }
}
```

java.util.Objects
Erweiterungen

```java
/**
 * Returns the first argument if it is non-{@code null} and
 * otherwise returns the non-{@code null} second argument.
 *
 * @param obj an object
 * @param defaultObj a non-{@code null} object to return if the first argument
 *                   is {@code null}
 * @param <T> the type of the reference
 * @return the first argument if it is non-{@code null} and
 *        otherwise the second argument if it is non-{@code null}
 * @throws NullPointerException if both {@code obj} is null and
 *        {@code defaultObj} is {@code null}
 * @since 9
 */
public static <T> T requireNonNullElse(T obj, T defaultObj) {
    return (obj != null) ? obj : requireNonNull(defaultObj, "defaultObj");
}
```

```java
    /**
     * Returns the first argument if it is non-{@code null} and otherwise
     * returns the non-{@code null} value of {@code supplier.get()}.
     *
     * @param obj an object
     * @param supplier of a non-{@code null} object to return if the first argument
     *                 is {@code null}
     * @param <T> the type of the first argument and return type
     * @return the first argument if it is non-{@code null} and otherwise
     *         the value from {@code supplier.get()} if it is non-{@code null}
     * @throws NullPointerException if both {@code obj} is null and
     *        either the {@code supplier} is {@code null} or
     *        the {@code supplier.get()} value is {@code null}
     * @since 9
     */
    public static <T> T requireNonNullElseGet(T obj, Supplier<? extends T> supplier) {
        return (obj != null) ? obj
                : requireNonNull(requireNonNull(supplier, "supplier").get(),
"supplier.get()");
    }
```

```java
/**
 * Checks if the {@code index} is within the bounds of the range from
 * {@code 0} (inclusive) to {@code length} (exclusive).
 *
 * <p>The {@code index} is defined to be out-of-bounds if any of the
 * following inequalities is true:
 * <ul>
 *  <li>{@code index < 0}</li>
 *  <li>{@code index >= length}</li>
 *  <li>{@code length < 0}, which is implied from the former inequalities</li>
 * </ul>
 *
 * @param index the index
 * @param length the upper-bound (exclusive) of the range
 * @return {@code index} if it is within bounds of the range
 * @throws IndexOutOfBoundsException if the {@code index} is out-of-bounds
 * @since 9
 */
@ForceInline
public static
int checkIndex(int index, int length) {
    return Preconditions.checkIndex(index, length, null);
}
```

Weitere Interessante JEPs

- JEP 11: Incubator Modules
- JEP 110: HTTP/2 Client (Incubator)
- JEP 226: UTF-8 Property Resource Bundles
- JEP 259: Stack-Walking API

Dankeschön!

Fragen?

Anmerkungen?

Michael Vitz | @michaelvitz

michael.vitz@innoq.com

https://www.innoq.com/de/talks/2017/09/bedcon-2017-java9-features/