JSR 336: Java SE 7

Final Release Specification

Mark Reinhold

2011/6/28 22:33 -0700 [36]

This specification defines the seventh edition of the Java Standard Edition platform.

As an “umbrella” JSR this document indirectly specifies significant features, which are themselves specified in related new component JSRs and in maintenance reviews of existing component JSRs. This JSR directly specifies smaller features, enhancements, and bug fixes that are not part of any component JSR. The specifications of those changes are contained in two accompanying documents: Annex 1 is a detailed change summary and Annex 2 is an annotated API specification showing the exact differences relative to Java SE 6.

This specification also includes updated versions of the Java Virtual Machine Specification and the Java Language Specification in Annex 3.

Contents

1 Themes

The feature set for Java SE 7 is driven, in large part, by a set of themes.

The themes describe the main focal points of the release. Some themes are fairly abstract guiding principles; others are more concrete in that they identify particular problem areas, significant new feature sets, or specific target market segments.

The themes are not prioritized, except that the first one is the most important.

Compatibility As the platform has matured, yet continued to evolve, many community members have naturally come to expect that their investments in Java-based systems, whether large or small, will be preserved. Any program running on a previous release of the platform must also run—unchanged—on an implementation of Java SE 7. (There are exceptions to this general rule but they are exceedingly rare, and they typically involve serious issues such as security.)

Productivity Java SE 7 will promote best coding practices and reduce boilerplate code by adding productivity features to the Java language and the Java SE APIs. These features will increase the abstraction level of most applications in a pragmatic way, with no significant impact on existing code and a minimal learning curve for all developers. We propose to enable, among other improvements, the automatic management of I/O resources, simpler use of generics, and more-concise exception handling.

Performance The Java SE platform has traditionally offered developers a range of features for writing scalable multi-threaded applications, for example with monitors in the Java language and VM and the concurrency utilities defined in JSR 166. To keep up with the inexorable trend toward multicore CPUs, Java SE 7 will add new concurrency APIs developed by Prof. Doug Lea and the JSR 166 community. These include, in particular, a Fork/Join Framework which can adaptively scale some types of application code to the available number of processors. Java SE 7 will further enable I/O-intensive applications by introducing a true asynchronous I/O API as part of JSR 203.

Universality Building upon the initial work in Java SE 6 to support scripting languages, Java SE 7 will introduce, via JSR 292, a new "invokedynamic" bytecode instruction and related APIs which will accelerate the performance of dynamic languages on the Java Virtual Machine.

Integration The Java SE Platform provides developers with a wealth of capabilities, but Java applications do not operate in isolation. A specific pain point for many years has been that of interacting with native filesystems, where a good user experience often requires exposing some details of the underlying platform. Java SE 7 will include a new, flexible filesystem API as part of JSR 203 which will provide portable access to common filesystem operations yet also allow platform-specific code to be written when desired.

Not every feature in the release is associated with a theme. The themes outline the primary focal points of the release, but they are not meant to be exhaustive. Not every theme, likewise, translates into a set of features. For example the first theme, Compatibility, captures a critical constraint on the further evolution of the platform.

2
Definitions

Additions to the Java SE platform specification are categorized as either features or enhancements. A *feature* is, roughly speaking, an addition of which at least one of the following statements is true:

- It requires two or more weeks of engineering effort to design and implement,
- It is significant in size (*e.g.*, a JSR), or
- It is in high demand by the Java community.

Any addition that is not a feature is considered an *enhancement*.

There is, obviously, quite a bit of room for interpretation in reading the above definition. In order to maximize the visibility of platform revisions we have generally tended to consider borderline items to be features rather than enhancements.

3
Component JSRs

Each large new component of the release is specified in its own separate JSR. All features, JSRs and otherwise, are described in more detail below, but for ease of reference the component JSRs are enumerated here along with their JCP status at the time of writing.

JSR 292: Support for dynamically-typed languages (InvokeDynamic)	*Proposed Final Draft*
JSR 334: Small language enhancements (Project Coin)	*Proposed Final Draft*
JSR 203: More new I/O APIs for the Java platform (NIO.2)	*Proposed Final Draft*

Consideration of the following JSRs, while of interest to many in the Java community, will be deferred to future releases:

- JSR 260: Javadoc Tag Technology Update
- JSR 295: Beans Binding
- JSR 296: Swing Application Framework
- JSR 305: Annotations for Software Defect Detection
- JSR 308: Annotations on Java Types
- JSR 310: Date and Time API

4 Related maintenance reviews

Several components of the previous version of the platform are still available separately or have significant specifications themselves. Changes to these components will therefore be covered in their own individual maintenance reviews:

- JSR 114: JDBC Rowsets
- JSR 221: JDBC 4.0
- JSR 224: Java API for XML Web Services
- JSR 269: Pluggable Annotation-Processing API
- JSR 901: Java Language Specification
- JSR 924: Java Virtual Machine Specification

5 Feature summary

vm	JSR 292: Support for dynamically-typed languages (InvokeDynamic)
	Strict class-file checking
lang	JSR 334: Small language enhancements (Project Coin)
core	Upgrade class-loader architecture
	Method to close a URLClassLoader
	Concurrency and collections updates (jsr166y)
i18n	Unicode 6.0
	Locale enhancement
	Separate user locale and user-interface locale
ionet	JSR 203: More new I/O APIs for the Java platform (NIO.2)
	TLS 1.2
jdbc	JDBC 4.1
client	Create new platform APIs for 6u10 graphics features
	Nimbus look-and-feel for Swing

	Swing JLayer component
web	Update the XML stack

6 Feature details

As a convenience, each feature description includes links to relevant online specifications when such documents exist. These links are not a normative part of this specification.

Virtual machine

JSR 292: Support for dynamically-typed languages (InvokeDynamic)

Extensions to the JVM, the Java language, and the Java SE API to support the implementation of dynamically-typed languages at performance levels near to that of the Java language itself

»

JSR 292; java.lang.invoke

Strict class-file checking

As specified in JSR 202, which was part of Java SE 6, and in the recently-approved maintenance revision of JSR 924, class files of version 51 (SE 7) or later must be verified with the typechecking verifier; the VM must not fail over to the old inferencing verifier.

»

JSR 202 §4.11.1

Language

JSR 334: Small language enhancements (Project Coin)

A set of small language changes intended to simplify common, day-to-day programming tasks: Strings in switch statements, try-with-resources statements, improved type inference for generic instance creation (“diamond”), simplified varargs method invocation, better integral literals, and improved exception handling (multi-catch)

»

JSR 334

Core

Upgrade class-loader architecture

Modifications to the ClassLoader API and implementation to avoid deadlocks in non-hierarchical class-loader topologies

»

java.lang.ClassLoader: registerAsParallelCapable, getClassLoadingLock

Method to close a URLClassLoader

A method that frees the underlying resources, such as open files, held by a URLClassLoader

»

java.net.URLClassLoader.close

Concurrency and collections updates (jsr166y)

A lightweight fork/join framework, flexible and reusable synchronization barriers, transfer queues, concurrent linked double-ended queues, and thread-local pseudo-random number generators

»

java.util.concurrent: ForkJoinPool, Phaser, TransferQueue, ConcurrentLinkedDeque, ThreadLocalRandom

Internationalization

Unicode 6.0

Upgrade the supported version of Unicode to 6.0

»

Unicode 6.0; java.lang.Character

Locale enhancement

Upgrade the java.util.Locale class to support IETF BCP 47 (Tags for Identifying Languages) and UTR 35 (Local Data Markup Language)

»

IETF BCP 47 : java.util.Locale: forLanguageTag, toLanguageTag; UTR 35 : java.util.Locale: getUnicodeLocaleAttributes, getUnicodeLocaleType, getUnicodeLocaleKeys

Separate user locale and user-interface locale

Upgrade the handling of locales to separate formatting locales from user-interface language locales

»

java.util.Locale: getDefault, setDefault; Locale.Category

I/O and Networking

JSR 203: More new I/O APIs for the Java platform (NIO.2)

New APIs for filesystem access, scalable asynchronous I/O operations, socket-channel binding and configuration, and multicast datagrams

»

JSR 203

TLS 1.2

Add support for Transport Layer Security version 1.2 (RFC 5246)

»

RFC 5246, RFC 5289, RFC 5469; javax.net.ssl: ExtendedSSLSession, SSLSocket.getHandshakeSession

Database Connectivity

JDBC 4.1

Upgrade to JDBC 4.1 and Rowset 1.1

»

java.sql; javax.sql.rowset: RowSetFactory, RowSetProvider

Client

Create new platform APIs for 6u10 graphics features

Create new platform APIs for features originally implemented in the 6u10 release: Translucent and shaped windows, and heavyweight/lightweight component mixing

»

java.awt: Window: setShape, setOpacity; Component

Nimbus look-and-feel for Swing

A next-generation cross-platform look-and-feel for Swing

»

javax.swing.plaf.nimbus

Swing JLayer component

Add the SwingLabs JXLayer component decorator to the platform

»

javax.swing.JLayer

Web

Update the XML stack

Upgrade the components of the XML stack to the most recent stable versions: JAXP 1.4, JAXB 2.2a, and JAX-WS 2.2

»

Corresponding maintenance reviews of JSR 206 (JAXP), JSR 222 (JAXB), and JSR 224 (JAX-WS)

A
Change history

Proposed Final Draft

- Added Annex 3, which contains the updated JVMS and JLS

Public Review, 2011/5/16

- Features that are not whole JSRs, together with all enhancements and bug fixes, will be specified directly by this JSR. The EDR draft of this JSR stated that they would be specified in concurrent maintenance reviews of JSR 270, the umbrella JSR for Java SE 6.
- Updated the list of related maintenance reviews: Added JSR 114: JDBC Rowsets and JSR 221: JDBC 4.0; removed JSR 206: Java API for XML Processing 1.4 and JSR 222: Java Architecture for XML Binding, since they did not change.
- Clarified the summary description of JSR 292 to make it clear that the VM, language, and API specifications are all being extended by that JSR.