

Proposed Java 7 Features

This list contains *possible* features for Java 7. No Java SE 7 JSR has been created.

Modularity

JSR 294 specifies a mechanism for *superpackages*, which define a *development time* module above the package level. Superpackages contain multiple classes and packages and specify which are public.

JSR 277 defines a new *deployment module* with support for versioning, reflective access, and a distribution format (a JAM or Java Module) that is a JAR with more metadata.

Java Kernel

This effort would break apart the Java SE libraries to allow them to be distributed and started as a kernel plus dynamic modules loaded on demand. This would reduce startup time, memory footprint, and possibly encourage a more pluggable ecosystem.

Swing

There are three JSRs related specifically to Swing improvements. JSR 296 is the Swing Application Framework which provides a simple framework with infrastructure common to most desktop applications. JSR 295 is the bean binding framework, which makes it simple to bind properties in two Java Beans together so that they are automatically kept in sync. The JSP expression language can be used to define linkages. JSR 303 is a beans validation framework that allows you to declaratively specify the constraints on a bean via standard annotations (similar to Hibernate Validator).

Reified Generics

Generics were introduced in Java 5 and to support backward compatibility with the huge installed base of existing Java code they were implemented using type erasure. In other words, at runtime, the generic type information is not used in distinguishing types, an `ArrayList<String>` is really the same class as an `ArrayList<Integer>`. In many cases, this is not an issue, but is the source for many surprising errors and corner cases. “Reification” would make runtime

type information real and available at runtime, making it possible to distinguish between different parameterized classes of different types.

In addition, there is a notion called “type literals” that can be used to workaround some of the reification issues that has been developed and may be suitable for addition to the language proper.

Annotations on Java Types

JSR 308 extends annotations to support them on more elements of the Java parse tree, most importantly type declarations, but also possibly expressions and other elements. JSR 305 defines a standard set of annotations that can be used to help the compiler or static analysis tools verify the correctness of Java code at compile time. JSR 305 will not be part of Java SE, but relies on JSR 308.

Short Instance Creation

Short instance creation aims to reduce the amount of typing needed to instantiate a new object, particularly an instance created with generics.

NIO 2

NIO 2 (JSR 203) is an extension and completion of the original NIO project added in Java 5. NIO 2 provides completion of asynchronous APIs on sockets and files, a new filesystem API with better support for attributes and filesystem-specific features, and a number of smaller items such as multicast support, “big” buffers (larger than int size), etc.

Language-level XML

Language-level XML support would add support for literal XML construction, data conversion, navigation, and streaming.

JavaBean Property Support

Several proposals have been floated to formalize the idea of first class “property” support in JavaBeans. This would potentially add keywords for the definition of properties, which might automatically generate getters/setters or allow access to properties by either `.` syntax or method syntax.

Closures

Perhaps one of the most contentious and hotly debated issues thus far is the addition of closures to Java. Closures support the passing functions around that capture the bindings of free variables in their lexical context. Closures can be passed around as if they were objects and may actually provide the ability to create new syntax-like constructs.

invokedynamic

Interpreted loosely typed languages like Ruby, Python, JavaScript, etc are being implemented on the JVM and have a critical issue due to the dynamic nature of their classes and instances. To address this, Sun has proposed JSR 292 for the addition of a new Java bytecode `invokedynamic` that can be used to dynamically choose the target of a method invoke.

Additionally, it is likely that more script engines may be shipped with the JDK in addition to the Rhino Beanscript engine.

Date / Time API

JSR 310 is an overhaul of the Java date/time APIs aiming to clean up many problematic issues and inspired by the well-respected JodaTime library.

Units and Quantities

JSR 275 defines specifies packages for the programmatic handling of physical quantities and their expressions as units.

JMX

JSR 255 defines JMX 2.0, which updates JMX to improve usability, new annotations, and federated JMX server support. JSR 262 defines support for a new JMX Remote API connector using web services.

Javadoc

JSR 260 has been pushed through a couple Java SE releases now and is meant to improve the current old school look of javadoc to add information and improve readability.

Miscellaneous

- 🕒 **BigDecimal operator support** - allow BigDecimal (arbitrary precision) objects to be used with arithmetic operators just like other Java primitives.
- 🕒 Allow String literals in switch case blocks.
- 🕒 Comparison support to allow enums to work with range operators `<` and `>`.

