By Joydip Kanjilal | April 20, 2022

JDK 18 is the first non-LTS release since the release of JDK 17. Note that LTS releases arrive after a span of two years – JDK 21 will be released in September 2023, the next LTS release in two years. JDK 19 will be released in September, after JDK 18. In this article, we will talk about the new features and enhancements in the Java 18 programming language.

What are the New Features in Java 18?

Below is a list of some of the newest features, updates, and deprecations made to the latest release of Java.

UTF-8 by Default

UTF-8 is a variable-width character encoding widely used on the web for electronic communication. Note that charset is a kind of character encoding capable of encoding all of the web's characters.

In Java 18, the default charset of the platform is now UTF-8. This is a change from Java's previous default charset, which was determined by the host operating system and locale. The main reason for this update is that UTF-8 has become the most widely used charset in the world, so making it an explicit choice in Java simplifies things quite a bit.

Read: Working with the new ValueType in Java

Simple Web Server

This proposal provides a minimalistic web server that can serve only static files. There is no CGI or servlet-like functionality provided. The tool will be handy for prototyping, ad hoc coding, and testing. The basic goals include the following:

- Providing an out-of-the-box static HTTP file server with a simple setup and minimal functionality.
- Reducing developer activation energy and making the JDK more approachable.
- Providing a default implementation via the command line alongside a small API for programmatic creation and customization.

Code Snippets in Java API Documentation

In addition to the fully-fledged sample applications, the API documentation sometimes contains small code snippets that Java developers can directly use in your application. The Java application programming interface (API) documentation includes code snippets written as comments in HTML and begins with the comment characters.

The Java API documentation provides code examples that illustrate the use of each feature. The examples are provided in a code snippet, which is a small piece of code that Java programmers can insert into a more extensive program.

Read: Tips to Improve Performance in Java

Vector API

Java coders can use the new Vector API in Java 18 to perform vector computations. A vector is simply an array of numbers that can be manipulated as a single entity. These vectors help perform numeric computations, especially complex ones that involve many different values. The Vector API is adept at increasing the speed of vector computations. A vector computation is a series of operations on vectors. For example, you could use vectors to perform scientific calculations or as part of machine learning algorithms. The Vector API allows you to do these sorts of calculations more quickly and with less effort than you could before.

The following are the fundamental objectives:

- Providing a ready-to-use static HTTP file server with minimum configuration and capabilities.
- Decreased developer activation energy and a more accessible JDK.
- Including a default command-line implementation in addition to a modest API.

Internet-Address Resolution SPI

The **Java.net.InetAddress** API converts hostnames to Internet Protocol (IP) addresses and back. This API currently uses the native resolver of your operating system. Java 18 specifies a service-provider interface (SPI) for hostname and address resolution, which java.net may use. **InetAddress** may use resolvers other than the built-in resolver of the platform. The new **InetAddress** API locates a resolution provider using a Service Loader.

You can use one of the following classes pertaining to the **java.net.spi** package:

- **InetAddressResolverProvider** — this is an abstract class that defines the service to be located by **java.util.ServiceLoader**.
- **InetAddressResolver** — this represents an interface that defines methods forward and reverse lookup operations.
- **InetAddressResolver.LookupPolicy** — this is another class whose instances describe the characteristics of a resolution request.
- **InetAddressResolverProvider.Configuration** — this is an interface that describes the built-in configuration for resolution operations of the platform.

The **InternetAddressResolver** class is a generic interface for resolving Internet address items. You can use this class to resolve hostnames from IP addresses, or hostname and port from an HTTP URI. The **InternetAddressResolverProvider** class is an abstract class that must be extended by concrete providers to define a custom lookup mechanism. It has three abstract methods that need to be overridden: **resolve()**, **getPriority()**, and **getSchemeName()**.

Foreign Function and Memory API

To access non-JVM systems, the Java Platform includes many libraries. JDBC Drivers, for example, may be used to connect to RDBMS. It is also feasible to use sockets to activate online services (HTTP client), serve distant clients (NIO channels), or connect with local processes.

The Java Platform includes many libraries that enable it to run on non-JVM systems. For instance, you can connect to JDBC drivers to work with RDBMS databases. Additionally, you can invoke web services and distant clients or even communicate with processes that run locally.

Read: The Top Java IDEs and Code Editors for Developers

Pattern Matching Improvements

Pattern matching simplifies the implementation of the common pattern of matching an object against a pattern. For example, you may want to check if a given object is an instance of a particular type and then access it in a type-safe manner. This feature improves readability and increases code safety. Pattern matching can be used in switch statements.

Deprecated Finalization for Removal

The finalize method is part of the legacy Java Object class and was intended to be called by the runtime system shortly before an object is made available for garbage collection. Finalizers are executed in a background thread, so they are explicitly not part of normal program execution. In addition, their invocation is unpredictable, especially when there is still enough memory to keep running. The JDK has deprecated finalizers for some time now, but

in JDK 18 the deprecation will become a warning. This move signals that finalizers might be removed from Java SE in a future release.

Final Thoughts on Java 18 Updates

In Java 18, the emphasis is on making the language more intuitive and accessible, encouraging programmers of all skill levels to use Java. The new version is sure to be a significant update, featuring new features like a simple web server, pattern matching improvements, and UTF-8 by default. In addition, with changes such as code snippets in Java API documentation and internet address resolution SPI, the focus has been placed more on improving the user experience of novices and experts alike.

Read more Java programming tutorials and Java developer news.