Veille de l'écosystème Java – Avril 2025

Avril 2025 a été un mois riche en actualités pour l'écosystème Java. La plateforme Java SE a connu des avancées notables, avec la sortie de **Java 24** apportant un nombre record de nouveautés, tout en préparant le terrain pour la prochaine version **Java 25 LTS**. Les principaux frameworks Java ont également évolué : **Spring** anticipe sa prochaine génération (Spring Framework 7, Spring Boot 3.5, etc.), **Jakarta EE 11** finalise ses derniers profils, et des mises à jour significatives sont sorties pour **Micronaut**, **Quarkus** et d'autres. Ce mois a aussi vu l'arrivée de nouvelles bibliothèques open source innovantes (ex. exécution de scripts shell en Java, threads virtuels, etc.), sans oublier des événements communautaires majeurs comme Devoxx France, qui ont mis en avant les tendances du moment (intégration de l'**IA** dans les tests, nouvelles pratiques de **testabilité** sans mocks, …).

En synthèse, on observe une consolidation des avancées introduites avec Java 17 et 21 (LTS), et une forte impulsion vers les technologies émergentes : **optimisations de la JVM** (GC, démarrage, etc.), **virtual threads** et programmation concurrente structurée, amélioration de la **productivité développeur** (outillage, nouveaux API), ainsi qu'une attention croissante à la **sécurité** et à l'**observabilité**. Les meilleures pratiques continuent d'évoluer – adoption progressive des nouveaux patterns du langage, outillage pour les tests plus robustes – ce qui aura un impact direct sur la feuille de route des projets Java dans les mois à venir.

Mises à jour du JDK et de la plateforme Java SE

- **Sortie de Java 24 (JDK 24)** – La version **JDK 24**, publiée le 18 mars 2025, est la plus riche en fonctionnalités depuis Java 9, embarquant *24 JEPs (Java Enhancement Proposals)* (JDK 24: The new features in Java 24 | InfoWorld). C'est une version à support court terme (6 mois) avant le prochain LTS, mais elle apporte de nombreux changements intéressants. Parmi les nouveautés notables figurent :
    - *Quatrième preview de la Concurrence Structurée* (Structured Concurrency) facilitant la gestion de tâches concurrentes en groupe (JDK 24: The new features in Java 24 | InfoWorld).
    - *Améliorations du support pour l'IA* : intégration des types primitifs dans les motifs (pattern matching) et *module import declarations*, utiles pour interfacer du code Java avec des bibliothèques de Machine Learning (JDK 24: The new features in Java 24 | InfoWorld) (JDK 24: The new features in Java 24 | InfoWorld). De plus, l'API vectorielle (Vector API) est enrichie, bénéfique pour les calculs intensifs en IA.
    - *Optimisations JVM expérimentales* : **Shenandoah générationnel** (un mode expérimental du garbage collector Shenandoah visant à améliorer le throughput) (Java 24 Delivers New Experimental and Many Final Features - InfoQ), et **en-têtes d'objets compacts** (Compact Object Headers) pour réduire la taille mémoire des objets (Java 24 Delivers New Experimental and Many Final Features - InfoQ). Ces fonctionnalités sont désactivées par défaut en attendant leur maturité.

- *Pipeline de traitement de flux extensible* : les **Stream Gatherers** permettent de créer des opérations intermédiaires personnalisées dans les streams, fonctionnalité finalisée après deux préviews (Java 24 Delivers New Experimental and Many Final Features - InfoQ).
- *Performance et démarrage* : le **chargement et lien des classes en avance** (Ahead-of-Time Class Loading & Linking) pour accélérer le démarrage des applications en pré-chargeant des classes lors d'une exécution de profilage (Java 24 Delivers New Experimental and Many Final Features - InfoQ). Également, divers affinements de la JVM comme l'expansion tardive des barrières pour G1 GC, etc.
- *Sécurité renforcée* : Java 24 implémente des algorithmes de cryptographie résistants aux ordinateurs quantiques, notamment une **signature digitale post-quantique** (Module-Lattice-Based Digital Signature, ML-DSA) (Java 24 Delivers New Experimental and Many Final Features - InfoQ) et un mécanisme d'encapsulation de clé post-quantique (KEM), conformément aux standards NIST FIPS.

(Java 24 Delivers New Experimental and Many Final Features - InfoQ) *Nombre de JEPs par version de Java depuis JDK 8. Java 24 introduit 24 nouvelles fonctionnalités, marquant un pic d'évolutions comparé aux versions précédentes (Java 9 en avait 91 avec la modularisation, puis ~5–17 JEPs par version)*

- **Versions LTS et correctifs** – Oracle a publié en avril les mises à jour critiques trimestrielles pour les versions LTS : **JDK 17.0.14** et **JDK 21.0.6** (ainsi que JDK 8u441, 11.0.26, et même 23.0.2 pour ceux sur JDK 23) (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ). Ces correctifs incluent des patchs de sécurité (CVE) et corrections de bugs. Parallèlement, les distributions tierces ont suivi : par exemple BellSoft (Liberica JDK) a livré ses builds 17.0.14.0.1, 21.0.6.0.1, etc., intégrant au total **740 backports et correctifs** pour éliminer 38 problèmes sur l'ensemble des releases (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ) (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ). Les développeurs utilisant Java 17 ou Java 21 sont encouragés à appliquer ces mises à jour pour rester à jour en matière de sécurité.

- **Préparation de Java 25 (LTS)** – La prochaine version majeure **Java 25** (prévue LTS en septembre 2025) commence à se préciser. En avril, Mark Reinhold a annoncé le *planning officiel de sortie* de JDK 25 (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ) : phase de *Rampdown* à partir du 5 juin 2025, Release Candidate initial le 7 août, et disponibilité générale (GA) ciblée au **16 septembre 2025** (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ). Plusieurs fonctionnalités candidate ont été promues pour inclusion dans Java 25 : par exemple la **suppression complète de l'API Applet** (JEP 504), proposée étant donné l'obsolescence totale des applets (plus supportés par les navigateurs) (Java News Roundup: Jakarta EE 11 Web Profile, GlassFish, TornadoVM, Micronaut, JHipster, Applet API - InfoQ). Huit nouveaux JEPs sont passés en statut *Candidate* ce mois-ci, dont la plupart seront finalisés après quelques cycles de preview (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ). Deux JEPs ont même atteint le statut *Proposed to Target* pour Java 25 :

- **JEP 512: Compacteur de sources et méthodes main d'instance**, qui vise à permettre des *fichiers source Java plus concis* et la définition de méthodes `main` au niveau instance (non statiques) (Java News Roundup: Gradle 8.14, JBash Jash, Hibernate, Open Liberty, Spring Cloud Data Flow - InfoQ).
- **JEP 511: Déclarations d'import de module**, qui introduit la possibilité d'importer des modules (fichiers JAR modulaires) de manière déclarative, simplifiant l'usage de modules externes dans le code (Java News Roundup: Gradle 8.14, JBash Jash, Hibernate, Open Liberty, Spring Cloud Data Flow - InfoQ).

*Exemple de nouvelle syntaxe (Java 25 preview) – **Constructeurs flexibles** :*

```
class Person { Person(String nom) { /*...*/ } }
class Employe extends Person {
    int id;
    Employe(String nom, int id) {
        System.out.println("Attribution de l'id " + id);
        super(nom);               // Appel du constructeur parent après le println
        this.id = id;
    }
}
```

Grâce à JEP 447/492 (Flexible Constructor Bodies), ce code compile : il est désormais permis d'exécuter des instructions avant le `super(...)` tant qu'on ne référence pas l'instance en cours de construction (JEP 492: Flexible Constructor Bodies (Third Preview)) (on peut toutefois initialiser ses champs (JEP 492: Flexible Constructor Bodies (Third Preview))). Cela évite d'avoir à déplacer cette logique dans des méthodes statiques auxiliaires ou dans des constructeurs intermédiaires.

- **Performance et compatibilité** – À noter, JDK 24 *déprécie* officiellement le portage Windows x86 32-bit en vue de son retrait (JDK 24: The new features in Java 24 | InfoWorld) (cette architecture n'ayant plus de support long terme à l'avenir). De même, le Security Manager (sécurité au niveau JVM) est désormais *définitivement désactivé* par défaut, après avoir été obsolète depuis Java 17, signant la fin de ce mécanisme historique (JDK 24: The new features in Java 24 | InfoWorld). Les développeurs sont encouragés à tester dès maintenant leurs applications sur Java 24 pour repérer d'éventuelles régressions, et à remonter les bugs sur le bug tracker officiel (Java News Roundup: Jakarta EE 11 Web Profile, GlassFish, TornadoVM, Micronaut, JHipster, Applet API - InfoQ) en prévision de Java 25.

Évolutions des frameworks Java majeurs

- **Spring Framework & Spring Boot** – L'écosystème Spring prépare une **nouvelle génération majeure**. En avril, la *milestone 4* de **Spring Framework 7.0.0** est sortie (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ), apportant des fonctionnalités de confort pour les développeurs. Par exemple, Spring 7 intègre un convertisseur automatique des `Optional<T>` en leur valeur sous-jacente, ce qui permet de passer

des `Optional` dans les expressions SpEL ou appels de méthodes sans code supplémentaire (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ). De plus, Spring 7 adopte la nouvelle **API Class-File de Java 24** (JEP 484) pour analyser plus efficacement le bytecode des classes (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ) – une amélioration transparente qui remplace l'ancien usage d'ASM dans le scanner de composants Spring. Côté web, *Spring for GraphQL* 1.4.0 est passé en RC1, avec notamment un **support d'observabilité** amélioré : un nouvel *observateur* pour le DataLoader GraphQL afin d'affiner les traces de chargement de données, et l'annulation proactive des fetchs réactifs lorsque le client se déconnecte (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ). L'ensemble de l'écosystème Spring a été très actif sur la fin du mois : **Spring Boot**, **Spring Data 2025.0.0**, **Spring Security**, **Spring Authorization Server**, **Spring Session**, **Spring Integration**, **Spring Modulith** et **Spring Web Services** ont tous livré des **release candidates** de leur prochaine version alignée sur Spring Framework 7 (Java News Roundup: Gradle 8.14, JBash Jash, Hibernate, Open Liberty, Spring Cloud Data Flow - InfoQ). Cela s'accompagne de versions milestone pour certains sous-projets (Spring Data 2025.1.0, Spring for Apache Kafka, Spring Vault, etc.) (Java News Roundup: Gradle 8.14, JBash Jash, Hibernate, Open Liberty, Spring Cloud Data Flow - InfoQ). Cette synchronisation indique qu'une **sortie coordonnée** d'une nouvelle famille Spring (probablement Spring Boot **3.5** ou **3.6** nommée *2025.0.0*) est proche.

- **Évolution de Spring Cloud Data Flow** – À noter un changement important : l'équipe Spring a annoncé la **fin du support open-source de Spring Cloud Data Flow (SCDF)** et de projets associés comme Spring Cloud Deployer et Spring State Machine (Java News Roundup: Gradle 8.14, JBash Jash, Hibernate, Open Liberty, Spring Cloud Data Flow - InfoQ). SCDF, outil d'orchestration de pipelines de streaming et batch, ne sera dorénavant maintenu qu'en offre commerciale Tanzu chez VMware après les versions mineures en cours (2.11.x, 2.9.x, 4.0.x) (Java News Roundup: Gradle 8.14, JBash Jash, Hibernate, Open Liberty, Spring Cloud Data Flow - InfoQ). Cette décision, motivée par la difficulté de maintenir ces projets de façon pérenne, signifie que les futures versions ne seront disponibles qu'aux clients **VMware Tanzu** (Java News Roundup: Gradle 8.14, JBash Jash, Hibernate, Open Liberty, Spring Cloud Data Flow - InfoQ). Les utilisateurs open-source de SCDF devront soit rester sur les dernières versions communautaires stables, soit envisager une migration (par exemple vers des alternatives comme Apache Airflow, Argo, ou vers l'offre commerciale).

- **Jakarta EE 11 et serveurs d'applications** – L'écosystème Jakarta EE franchit une étape importante : la **spécification Jakarta EE 11 Web Profile a été finalisée et publiée** en avril (Java News Roundup: Jakarta EE 11 Web Profile, GlassFish, TornadoVM, Micronaut, JHipster, Applet API - InfoQ) (Java News Roundup: Jakarta EE 11 Web Profile, GlassFish, TornadoVM, Micronaut, JHipster, Applet API - InfoQ). D'après le compte-rendu d'Ivar Grimstad (responsable Jakarta EE à la Fondation Eclipse), l'effort a été colossal pour refactoriser le TCK (suite de tests de compatibilité) et valider cette nouvelle version de profil (Java News Roundup: Jakarta EE 11 Web Profile, GlassFish, TornadoVM, Micronaut, JHipster, Applet API - InfoQ). **Jakarta EE 11** vise une sortie de la plateforme complète d'ici la fin du 2ème trimestre 2025 (Java News Roundup: Jakarta EE 11 Web Profile, GlassFish,

TornadoVM, Micronaut, JHipster, Applet API - InfoQ), après un dernier Release Candidate. Le serveur de référence **Eclipse GlassFish 8.0.0** a d'ailleurs passé avec succès le TCK du Web Profile 11, se positionnant comme première implémentation compatible (Java News Roundup: Jakarta EE 11 Web Profile, GlassFish, TornadoVM, Micronaut, JHipster, Applet API - InfoQ). GlassFish a enchaîné les milestones (M10 en mars, M11 début avril) pour aligner ses fonctionnalités sur Jakarta EE 11, avec au menu des correctifs, des montées de versions de dépendances, et le support de JDK 24 (Java News Roundup: Milestone Releases of Spring Cloud, GlassFish and Grails, Devnexus 2025 - InfoQ). Par exemple, GlassFish 8 désactive désormais TLS 1.0/1.1 par défaut (protocoles dépréciés) et migre vers des classes non obsolètes pour l'intégration CDI (remplacement de `WeldListener` par `WeldInitialListener`) (Java News Roundup: Milestone Releases of Spring Cloud, GlassFish and Grails, Devnexus 2025 - InfoQ). Côté serveurs commerciaux/open source, la plateforme **Payara** (dérivée de GlassFish) a publié son édition d'avril 2025 : Payara Community 6.2025.4 et Payara Enterprise 6.25.0, apportant la personnalisation fine des logs envoyés vers des serveurs distants et de nouveaux réglages de pool de connexions JDBC (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ) (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ). Enfin, **WildFly 36** (serveur JBoss) a atteint sa première **version bêta** ce mois-ci : WildFly 36.0.0.Beta1 améliore les performances JPA en activant par défaut un transformateur de classes (`jboss.as.jpa.classtransformer`) et logue un avertissement si plusieurs systèmes de métriques sont activés en même temps (Java News Roundup: Jakarta EE 11 and Spring AI Updates, WildFly 36 Beta, Infinispan, JNoSQL - InfoQ) (Java News Roundup: Jakarta EE 11 and Spring AI Updates, WildFly 36 Beta, Infinispan, JNoSQL - InfoQ). Ces ajustements indiquent le travail en cours pour la compatibilité Jakarta EE 11 (WildFly 36 devrait supporter Jakarta EE 11 une fois finalisé).

- **Micronaut** – Le framework Micronaut (cloud-native, microservices) continue son rythme de releases fréquentes. En avril, **Micronaut 4.8** a vu deux mises à jour mineures (4.8.0 puis 4.8.2) apportant essentiellement des corrections de bugs et des mises à jour de dépendances. Micronaut 4.8.0 a introduit quelques améliorations internes et le support de **FP16 vers FP32** pour des calculs de matrices sur GPU via TornadoVM (Java News Roundup: Jakarta EE 11 Web Profile, GlassFish, TornadoVM, Micronaut, JHipster, Applet API - InfoQ). La version **4.8.2** (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ) a livré en particulier Micronaut Core 4.8.11 et mis à jour certains modules officiels (plugin Maven, module JSON Schema, Micronaut Micrometer pour la télémétrie, Micronaut Servlet, etc.). Rien de révolutionnaire fonctionnellement, mais la plateforme reste très active et stable, préparant vraisemblablement un futur Micronaut 5. Notons que Micronaut a servi de base d'expérimentation pour les threads virtuels (Project Loom) depuis la v4; ces versions récentes intègrent donc les derniers correctifs pour tirer parti de **Java 21+**.

- **Quarkus** – Le framework Quarkus (sponsorisé par Red Hat) consolide sa version 3.x avant un prochain support long terme. La branche Quarkus **3.20** est attendue comme LTS, et en amont la version **3.19.2** (mars) a préparé le terrain avec des améliorations de l'outil de mise à jour de Quarkus et une meilleure interopérabilité des tests JUnit (intégration de `QuarkusUnitTest` avec JUnit @TestFactory) (Java News Roundup: Milestone Releases of Spring Cloud, GlassFish and

Grails, Devnexus 2025 - InfoQ). En avril, Quarkus a progressé rapidement : une version **3.21.4** est sortie fin avril, corrigeant notamment un problème de `StackOverflowError` lié aux politiques de retry de MicroProfile Fault Tolerance (Java News Roundup: Gradle 8.14, JBash Jash, Hibernate, Open Liberty, Spring Cloud Data Flow - InfoQ). Quarkus 3.21 a également renforcé la validation de configuration de sécurité HTTP (alerte en cas de doublon de nom sur une policy) (Java News Roundup: Gradle 8.14, JBash Jash, Hibernate, Open Liberty, Spring Cloud Data Flow - InfoQ). Ces mises à jour incrémentales témoignent de la maturité de Quarkus 3, qui cible le support de **Java 21+** et l'optimisation du runtime natif via GraalVM. À noter, l'outil de planification de tâches **JobRunr** est sorti en version 7.5.0 en avril avec le support explicite de **Quarkus 3.20.0** (et Micronaut 4.8.0), ce qui confirme l'arrivée imminente de Quarkus 3.20 (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ) (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ). JobRunr 7.5 signale toutefois un *breaking change* pour Quarkus/Micronaut : le fallback automatique sur un storage en mémoire est désactivé, nécessitant de configurer explicitement le type de stockage des jobs (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ).

Nouveaux outils et bibliothèques open source

- **Langages et runtimes** – Le panorama des outils pour développeurs Java s'est enrichi ce mois-ci. Le build tool **Gradle 8.14** est sorti en version stable (Java News Roundup: Gradle 8.14, JBash Jash, Hibernate, Open Liberty, Spring Cloud Data Flow - InfoQ), apportant le support officiel de Java 24, l'introduction d'une initialisation *lazy* des configurations de dépendances (pour améliorer les performances mémoire lors de la configuration du build), et un nouveau mode de vérification d'intégrité du cache de configuration (Java News Roundup: Gradle 8.14, JBash Jash, Hibernate, Open Liberty, Spring Cloud Data Flow - InfoQ). Du côté des utilitaires, l'outil en ligne de commande **JBang** (qui facilite l'exécution de scripts Java) a introduit un nouveau composant baptisé **Jash** (Java + Shell) (Java News Roundup: Gradle 8.14, JBash Jash, Hibernate, Open Liberty, Spring Cloud Data Flow - InfoQ). **Jash** est une bibliothèque permettant d'exécuter des commandes shell de façon fluide et typée en Java, avec un style « pipe-like ». Par exemple, on peut écrire:

```
import static io.ongres.jash.Jash.$;
$("echo hello; echo world").stream()
    .forEach(System.out::println);
```

Ce code exécutera la commande shell entre guillemets et **streamera** la sortie standard ligne par ligne dans le programme Java (ici affichant "hello" puis "world"). Jash gère automatiquement les threads nécessaires en coulisse et offre des fonctionnalités avancées (chaînage de commandes avec `pipe()`, gestion des codes de retour, timeouts, etc.) (jbang-jash/README.md at main · jbangdev/jbang-jash · GitHub) (jbang-jash/README.md at main · jbangdev/jbang-jash · GitHub). Cet outil illustre l'effort de la communauté pour rendre l'interaction avec le système plus idiomatique en Java (un besoin fréquent pour les scripts et l'automatisation).

- **Persistance et BDD NoSQL** – Dans le domaine des bases de données, la **spécification Jakarta Persistence 3.2** (JPA) a été implémentée par Hibernate ORM 7 (actuellement en RC) et commence à être supportée par les outils connexes. **Hibernate Search 8.0.0.Alpha3** est sorti, aligné sur Hibernate ORM 7.0.0.Beta5, et intègre la nouvelle API *Hibernate Models* (basée sur Jandex) pour la manipulation du modèle de données, en remplacement des anciennes annotations Commons (Java News Roundup: Jakarta EE 11 and Spring AI Updates, WildFly 36 Beta, Infinispan, JNoSQL - InfoQ) (Java News Roundup: Jakarta EE 11 and Spring AI Updates, WildFly 36 Beta, Infinispan, JNoSQL - InfoQ). Côté caches et in-memory data grid, la version **Infinispan 15.2.0.Final** (nom de code *Feelin' Blue*) est disponible (Java News Roundup: Jakarta EE 11 and Spring AI Updates, WildFly 36 Beta, Infinispan, JNoSQL - InfoQ) : elle apporte entre autres un support du protocole Redis **JSON API** (pour manipuler des documents JSON stockés dans Infinispan comme on le ferait dans Redis) et un rafraîchissement de son interface d'administration (nouveau thème basé sur PatternFly 6) (Java News Roundup: Jakarta EE 11 and Spring AI Updates, WildFly 36 Beta, Infinispan, JNoSQL - InfoQ).

- **Observabilité et monitoring** – L'univers de l'observabilité Java évolue avec les versions de la plate-forme. **Micrometer** – la bibliothèque de métriques utilisée par Spring Boot et d'autres – prépare sa version 1.15.0 (sortie en RC1). Micrometer 1.15 apporte de nouvelles métriques out-of-the-box, par exemple pour les *threads virtuels* : un compteur dédié suit le nombre de threads créés via `Executors.newVirtualThreadPerTaskExecutor()` (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ). L'API d'envoi OTLP (OpenTelemetry) a aussi été enrichie pour faciliter l'envoi de métriques (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ). En parallèle, **Micrometer Tracing** (ex Brave/Zipkin intégré, ex Spring Cloud Sleuth renaming) publie sa v1.5.0-RC1 qui se débarrasse d'une dépendance à une API OpenTelemetry incubée, anticipant les changements du standard OTEL (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ). Dans le même esprit de mise à niveau technologique, **Project Reactor** – la bibliothèque réactive utilisée par Spring WebFlux – a publié une *milestone 2* de sa prochaine version 2025.0.0, avec Reactor Core 3.8.0-M2, Reactor Netty 1.3.0-M2, etc. (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ). Un point notable : Reactor intègre désormais le support de **CRaC (Coordinated Restore at Checkpoint)**, une fonctionnalité expérimentale qui vise à améliorer le démarrage en restaurant l'état d'une application à partir d'un snapshot (pratique pour les FaaS et cloud cold starts) (Java News Roundup: Milestone Releases of Spring Cloud, GlassFish and Grails, Devnexus 2025 - InfoQ). Tous ces outils mettent l'accent sur la compatibilité avec Java 21+ et l'adaptation aux nouveaux paradigmes (threads virtuels, GraalVM, etc.).

- **Autres bibliothèques** – Parmi les autres sorties du mois, on peut citer : **Jox 1.0.0**, une librairie proposant une structure de données de type *Channel* optimisée pour les threads virtuels (inspirée des *Go channels* ou *Kotlin coroutines channels*) (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ). Jox vise à faciliter la communication entre tâches concurrentes de manière performante. Également, l'écosystème **JHipster** (générateur d'applications Java/JS) a livré JHipster Lite 1.30.0 avec des améliorations UX (couleurs et filtres dans l'interface de génération) (Java News Roundup: Jakarta EE 11 and Spring

AI Updates, WildFly 36 Beta, Infinispan, JNoSQL - InfoQ). Le framework open-source **OpenXava 7.5** (génération d'applications d'entreprise Java) a introduit le rechargement de code à chaud en dev sans impact prod et un nouveau thème UI modernisé (Java News Roundup: Jakarta EE 11 and Spring AI Updates, WildFly 36 Beta, Infinispan, JNoSQL - InfoQ). Enfin, un événement marquant pour la pérennité open source : le projet **Kroxylicious** (un proxy Kafka léger) a rejoint la **Commonhaus Foundation**, une fondation à but non lucratif dédiée au soutien des bibliothèques open source (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ). Cela lui assurera un support communautaire et financier pour accélérer son développement.

Vie de la communauté Java en avril 2025

- **Conférences et meetups** – Le mois d'avril a été animé par des rassemblements de développeurs Java. En particulier, la conférence **Devoxx France 2025** s'est tenue du 16 au 18 avril à Paris (Palais des Congrès), réunissant plus de 240 présentations sur trois jours (Devoxx France 2025: Innovation and Networking in Technology - Orange Developer). Les thématiques à l'honneur ont reflété les tendances actuelles : beaucoup de sessions autour de **Jakarta EE 11**, des retours d'expérience sur **Spring Boot 3** en production et l'adoption de **Java 21**. Une session marquante portait sur "**AI for Testing**" (par Yann Helleboid) où l'on a démontré comment l'**IA générative** peut assister les tests logiciels, en générant des scénarios de test et en détectant des bugs à partir du code (Devoxx France 2025: Innovation and Networking in Technology - Orange Developer) (Devoxx France 2025: Innovation and Networking in Technology - Orange Developer). Une autre session "**Overcoming Mock Issues**" a mis en avant le **Contract Testing** (test de contrats) pour éviter l'abus de mocks en tests d'intégration, en s'appuyant sur l'architecture hexagonale pour tester les composants externes de manière réaliste (Devoxx France 2025: Innovation and Networking in Technology - Orange Developer) (Devoxx France 2025: Innovation and Networking in Technology - Orange Developer). Ces sujets montrent l'intérêt grandissant de la communauté pour améliorer la **fiabilité des tests** et adopter des outils modernes (y compris l'IA) dans le cycle de développement.

  Côté international, la conférence **Devnexus 2025** (organisée par le Atlanta JUG début mars) a publié en avril les vidéos de ses meilleurs talks. Devnexus, l'une des plus grandes conférences Java aux USA, a notamment proposé des keynotes sur l'état de **OpenJDK** (planning de Java 25, avancées du projet Loom) et sur **Quarkus en production serverless**. De même, les vidéos de **JFokus 2025** (Stockholm) diffusées ce mois-ci ont couvert des thèmes comme la **sécurité de la supply chain Java** et le futur de **Kotlin** côté JVM. La communauté Java continue ainsi de partager activement connaissances et retours d'expérience via ces événements.

- **Articles et publications marquants** – En avril, de nombreux articles de blog et news techniques ont alimenté la veille. Le site InfoQ a proposé chaque semaine un *Java News Roundup* synthétisant les sorties et JEPs en cours (une ressource précieuse que nous avons largement citée). Parmi les billets notables : un article détaillé sur la sortie de **Java 24** et ses **24 nouvelles features** a été très relayé (JDK 24: The new features in Java 24 | InfoWorld) (JDK 24: The new features in Java 24 | InfoWorld), notamment pour son analyse de l'impact sur le développement

d'applications d'IA en Java. L'annonce de **Spring Framework 7.0.0-M4** sur le blog officiel Spring (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ) a également fait du bruit, car elle donne un avant-goût de la direction prise par Spring (avec le support des nouveautés Java 24+, l'accent sur l'expérience développeur et l'ouverture vers l'IA via le projet Spring AI). Enfin, plusieurs experts de la communauté ont publié des contenus sur l'adoption de **Virtual Threads** en pratique : par exemple, un article sur *DZone* a exploré comment migrer un service web Spring Boot classique vers l'utilisation de threads virtuels pour améliorer sa scalabilité (montrant une réduction notable du coût de thread et de meilleurs temps de réponse sous forte charge). Sur le plan des bonnes pratiques de code, un billet de Nicolai Parlog a souligné les pièges courants lors de l'activation de *Preview Features* du langage en production, conseillant de bien cloisonner les usages expérimentaux et de surveiller les changements entre versions preview et finales. En somme, la littérature technique du mois a couvert aussi bien les nouvelles APIs que les retours terrain, aidant les développeurs à se projeter dans l'adoption de ces nouveautés.

Bonnes pratiques et conseils émergents

- **Adopter progressivement Java 17/21+** – Un message récurrent en avril : si ce n'est pas déjà fait, il est temps de planifier le **passage aux versions LTS récentes** (Java 17 ou idéalement Java 21). Les retours des conférences confirment que Java 21 est suffisamment stable et apporte un vrai plus (patterns, `Record` et `sealed`, threads virtuels, API de date améliorée, etc.) pour justifier une migration. De plus, les prochains frameworks majeurs (Spring 7, Jakarta EE 11, Quarkus 3.20) cibleront Java 21 minimum, donc rester sur Java 11 ou 8 va devenir un frein. Les équipes devraient profiter de 2025 pour mettre à jour leurs bases de code, en profitant des outils de migration (JDeps, etc.) et en activant les modes de compatibilité (`--enable-preview` le cas échéant pour tester les nouveautés).

- **Exploiter les virtual threads de Loom** – Avec l'arrivée à maturité de **Project Loom** dans Java 21, les threads virtuels (aussi appelés fibres) constituent désormais une approche recommandée pour écrire du code concurrent massif de manière simple. Les discussions à Devoxx FR ont mis en avant des cas d'usage concrets : par exemple, remplacer un thread pool d'exécuteurs par des virtual threads permet de gérer des milliers de requêtes parallèles sans complexité ni tuning fin. Les **servlets** ou contrôleurs Spring Boot peuvent être annotés ou configurés pour utiliser un `VirtualThreadPerTaskExecutor`, ce qui, d'après des retours, **simplifie le code** (plus besoin de réacteur ou de programmation asynchrone complexe) et offre des performances équivalentes voire supérieures pour la plupart des workloads I/O. La **meilleure pratique** qui se dessine est : *utiliser les threads virtuels par défaut pour toute nouvelle application** (sauf contrainte spécifique), afin d'avoir un code plus lisible tout en conservant la scalabilité. Au niveau des frameworks, cela se traduit par exemple par Tomcat/Jetty/Netty qui travaillent à un support natif de Loom, ou par des libs comme Jox qui fournissent des outils adaptés aux virtual threads.

- **Tester sans mocks abusifs** – En matière de tests, un mantra refait surface : limiter le recours excessif aux mocks. Le Hands-on Lab sur les tests sans mocks à Devoxx a illustré que trop de mocks dans les tests unitaires peut conduire à des tests fragiles, peu fiables et coûteux à

maintenir. À la place, la communauté promeut l'utilisation de **tests d'intégration contractuels** (Contract Testing) ou de *fake implementations* simples. Par exemple, pour tester un service REST externe, au lieu de moquer le client HTTP, on peut utiliser un serveur stub qui répond avec des données réalistes, ou encore mieux, utiliser les frameworks de contract test (tels que Spring Cloud Contract) pour générer automatiquement des stubs conformes aux contrats d'API. L'architecture hexagonale (ports and adapters) a été conseillée, car elle facilite le remplacement des dépendances externes par des implémentations de test. Le résultat attendu est d'**améliorer la confiance** dans les tests (moins de faux positifs/negatifs dus à des mocks mal configurés) et de tester réellement le comportement de bout en bout du code.

- **Observabilité par défaut** – Un autre conseil qui se dégage des retours d'expérience : intégrer l'**observabilité** dès le début d'un projet Java. Concrètement, cela signifie activer les métriques (Micrometer) et le traçage distribué (OpenTelemetry) par défaut dans vos microservices, même en dev. Micrometer 1.15 va simplifier cela avec moins de configuration (plus d'auto-métriques, y compris sur les threads virtuels et executors). Les experts suggèrent de définir des dashboards et alertes standards dès la mise en production d'un service Java – par exemple un tableau de bord avec le **temps de pause GC**, le **throughput** de requêtes, le nombre de threads en exécution, etc., afin de détecter rapidement les problèmes de performance ou fuites de ressources. Les projets comme **OpenTelemetry** étant désormais supportés nativement par Spring Boot (actuator tracers) et Quarkus, il est recommandé d'en profiter pour instrumenter son code avec des *spans* personnalisés autour des traitements critiques, ce qui facilitera le débogage en cas d'incident en prod.

- **Sécurité et supply chain** – Suite à plusieurs alertes (Log4Shell, etc.) et à l'actualité réglementaire (ex: **Cyber Resilience Act** européen), la communauté Java met l'accent sur la sécurité de la chaîne logicielle. En avril, on a rappelé quelques bonnes pratiques : tenir à jour ses dépendances (outil comme OWASP Dependency Check, Renovate bot, etc.), préférer les distributions JDK sécurisées (les CPU d'avril ont corrigé des vulnérabilités, d'où l'importance d'adopter JDK 17.0.14/21.0.6 rapidement (Java News Roundup: JDK 25 Schedule, Spring 7.0-M4, Payara Platform, JobRunr 7.5, Jox 1.0, Commonhaus - InfoQ)), et activer **Security Manager** ou équivalents *sandboxing* pour les applications embarquant des plugins non fiables – même si le Security Manager traditionnel disparaît, des alternatives comme **OSP (Open Policy Agent)** ou des *Java Agents de monitoring* peuvent aider à contrôler les actions sensibles. Enfin, l'adoption de solutions comme **Sigstore** pour signer les artefacts Maven et vérifier leur provenance commence à être encouragée dans l'écosystème Java.

Impacts pour les roadmaps projets Java

Les évolutions d'avril 2025 se traduisent par plusieurs **impacts concrets sur les projets Java** en cours ou à venir :

- **Adoption de Java 21/24 et anticipation de Java 25** : Les équipes techniques doivent décider du timing de migration vers Java 21 (LTS) ou Java 24 (STS). Avec la feuille de route désormais claire pour Java 25 en septembre (LTS), un choix stratégique courant sera d'**attendre Java 25** pour

unifier l'effort de migration. Toutefois, reporter trop longtemps empêche de profiter des gains immédiats (par ex. threads virtuels, pattern matching étendu). Il peut être judicieux de commencer dès maintenant des **prototypes** ou tests sur Java 21 ou 24, tout en prévoyant une bascule en production une fois Java 25 sorti et stabilisé (d'ici fin 2025). Sur la compatibilité, peu de ruptures majeures sont signalées entre Java 17 -> 21 -> 24, hormis l'abandon des Applets et du Security Manager (déjà obsolètes) et la désactivation du 32-bit. Donc la **compatibilité applicative** ne devrait pas freiner la mise à jour, surtout si les applications utilisaient déjà un kit de build moderne (Gradle/Maven) et évitaient les API internes non supportées.

- **Mises à jour des frameworks** : La sortie prochaine de Spring Framework 6.1 / 7.0, Spring Boot 3.5+ et Jakarta EE 11 signifie que les projets devront planifier des **montées de version** de leurs frameworks au cours des 6 à 12 prochains mois. Par exemple, un projet Spring Boot 2.x ou 3.0 devra envisager de migrer vers Spring Boot 3.2/3.5 pour rester supporté et exploiter les nouvelles features (avec Java 17+ obligatoire). De même, les applications d'entreprise sur Jakarta EE 9/10 (Java EE 8) verront arriver Jakarta EE 11 : il faudra évaluer l'effort de migration (souvent essentiellement du renommage de package jakarta.* et quelques adaptations mineures). L'avantage est que Jakarta EE 11 apporte le support natif de Java 21, ce qui peut améliorer les performances et la stabilité. Il est recommandé d'**étudier les roadmaps officiels** (Spring, Quarkus, etc.) publiés en avril (Java News Roundup: Milestone Releases of Spring Cloud, GlassFish and Grails, Devnexus 2025 - InfoQ) (Java News Roundup: Gradle 8.14, JBash Jash, Hibernate, Open Liberty, Spring Cloud Data Flow - InfoQ) pour aligner la roadmap de votre application : par exemple, Quarkus 4 est peut-être en préparation chez Red Hat – si votre projet envisage Quarkus, anticipez les changements à venir. Globalement, intégrer ces mises à niveau dans la planification évite d'être pris de court par la fin de support d'une version ou l'arrivée d'une faille non corrigée sur un vieux stack.

- **Performances et coût** : Les améliorations de Java 24 et des outils peuvent se traduire par des **opportunités d'optimisation** pour vos projets. Par exemple, le GC Shenandoah générationnel ou les en-têtes compacts pourront à terme réduire la latence des applications à forte charge allocative (microservices intensifs en données). Sur le court terme, l'**activation des virtual threads** peut potentiellement réduire la consommation de ressources (mémoire par thread largement inférieure) et améliorer le throughput sur les serveurs applicatifs – avec possiblement une baisse de coûts cloud si moins d'instances sont nécessaires pour le même trafic. En parallèle, l'adoption d'outils comme JBang/Jash ou d'un build plus rapide (Gradle 8.14) peut accélérer la boucle de développement et de livraison. Les leads techniques devraient communiquer ces perspectives aux parties prenantes (management, ops) pour justifier les mises à jour : *migrer vers les dernières versions Java & frameworks n'est pas seulement pour "faire du neuf", c'est un investissement qui apporte des gains de performance, de sécurité et de maintenabilité*.

- **Innovations et futur-proofing** : Enfin, cette veille d'avril 2025 souligne l'importance de rester en veille sur les **tendances émergentes** (IA, nouvelles API, etc.). Un projet lancé en 2025 qui ignore ces mouvements risque de prendre du retard. Au contraire, intégrer dès maintenant quelques-uns de ces éléments peut apporter un avantage compétitif. Par exemple, ajouter une couche d'**IA générative** (via Spring AI ou d'autres SDK) pour aider les utilisateurs ou automatiser certaines

tâches dans vos applications, c'est désormais envisageable en Java de façon standardisée. De même, repenser l'architecture en intégrant l'observabilité et le contract testing améliore la qualité à long terme. Les roadmaps projets devraient donc inclure des **phases d'expérimentation** courtes sur ces sujets (POC sur l'IA, essais de nouveaux modules JDK en preview, etc.), afin d'évaluer leur pertinence. En somme, l'écosystème Java en avril 2025 démontre qu'il est en pleine évolution – les équipes qui embrassent ces changements de manière proactive seront mieux armées pour livrer des logiciels performants, sûrs et pérennes.

Sources : Blogs techniques d'experts, annonces officielles (Oracle, Spring.io), articles InfoQ (Michael Redlich), sites de projets open source (GitHub, Eclipse) et retours de conférences (Devoxx France 2025, Devnexus 2025). Les liens ci-dessous pointent vers ces sources pour approfondir chaque sujet abordé.