	Java SE	Java EE	Jakarta EE	Spring	Spring Boot	
1995	**Java 1.0 誕生** ・「Write Once, Run Anywhere」	-	-	-	-	Sun Microsystems が Java 発表。Web 時代の到来
1997	JDK 1.1 ・内部クラス追加 ・イベント処理改善	-	-	-	-	Java がブラウザ(Java Applet)で注目を集める
1998	J2SE 1.2 ・Java2 と呼ばれる ・コレクション API 導入	J2EE 1.2（初版） ・Servlet / JSP / EJB1.0 ・公式エンタープライズ API 誕生	-	-	-	Web アプリ需要増、企業向け Java の始まり
2001	J2SE 1.3 ・JIT 最適化 ・安定性向上	2EE 1.3 ・EJB2.0（さらに複雑に…）	-	-	-	Java EE が急速に複雑化し使いにくくなる。
2003	J2SE 1.4 ・assert 導入 ・Logging API	J2EE 1.4 ・Web サービス対応 ・EJB 依然として重い	-	Spring 1.0 ・DI/IoC を XML で実現 ・EJB を使わずに開発可能に ・軽量コンテナの革命	-	EJB の失敗。 より良い開発体験を求めて、Spring 誕生。
2004	Java SE 5 (1.5) - ジェネリクス導入 - アノテーション導入 - enum, 拡張 for, 可変長引数 → “近代 Java”の始まり	Java EE 5（準備段階） - 依然として複雑 (EJB2.x)	-	Spring 1.x → 2.0 準備 - DI コンテナ成熟 - XML 設定が主流	-	**Java 5 の躍進**
2006	Java SE 6 - パフォーマンス改善 - 開発ツール強化（JConsole など） - 安定版	Java EE 5 - アノテーションで EJB 簡略化 - POJO で EJB が書ける - Spring のアイデアをパクる → EE が突然モダン化	-	Spring 2.0 - AOP 強化 - XML ベースで柔軟な DI - EE より使いやすい	-	Java EE が巻き返しを狙い Spring を模倣。 だが既に多くの開発者は Spring へ流れていた。。。
2007-2008	Java SE 6 継続 - “黄金期の安定性”	Java EE 5 普及遅れ - サーバー実装(WebLogic 等)が重く導入コスト高	-	Spring 2.x - 実質 Java 業界のデファクト標準に - 多くの企業が Spring 採用	-	Java EE の衰退。 オープンソース＆軽量志向が主流に。
2009	Java SE 6 (末期) - Java 7 が遅延し不安視	Java EE 6 - CDI（DI の公式化） - JAX-RS（REST API） - EE がさらに Spring に寄せる	-	Spring 3.0 - JavaConfig（アノテーション DI） - XML 地獄からの脱却開始 - 強力な AOP/SpEL	-	Sun Microsystems 経営悪化。 Oracle が Sun を買収すると発表。
2010	Java SE 6（安定継続）	Java EE 6（そこそこ好評）	-	Spring 3.0（JavaConfig 登場）	-	Oracle、Sun 正式買収。 **OSS 文化の Sun 。商業志向の Oracle。**
2011	Java SE 7 - try-with-resources - switch on String - しかしバグ多発 & 信頼失墜	Java EE 6（変化なし）	-	Spring 3.1～3.2 - MVC 強化 - Annotation ベース開発が主流に	-	Java 7 品質問題の露呈。
2012	Java SE 7 継続	Java EE 6（停滞気味）	-	Spring 3.x 安定 - Spring が企業のデファクトに	-	クラウド時代の到来（AWS, Heroku） 「軽量に」「素早く」デプロイ。
2013	Java SE 7 継続 Java 8 準備	Java EE 7 - WebSocket - JSON-P - しかしまた重い…普及しない	-	Spring 4.0（Java 8 対応） - Functional スタイル導入	-	マイクロサービスの概念が世界で広がり始める
2014	Java SE 8（史上最大の進化） - ラムダ式 - Stream API - Optional - Date/Time API → “近代 Java 完全体”	Java EE 7（普及せず） - Spring に完全に遅れを取る	-	Spring 4.x	**Spring Boot 1.0 誕生** - “起動してすぐ動く” - 自動設定/スタータ依存 - Tomcat 内蔵 - DevTools、Actuator	Spring Boot 登場で Java 開発体験が激変。 「設定地獄からの解放」 「REST が爆速で作れる」
2015	**Java SE 8** (支持率up)(**LTS**) - ラムダ/Stream 大普及	Java EE 7（停滞） - 進化なし - サーバーベンダーも低迷	-	Spring 4.x (安定)	Boot 1.x 普及開始	マイクロサービス時代の本格化 Netflix, AWS, Docker が台頭 REST + クラウドが常識に
2016	Java 8 続投（Java 9 遅延）	Java EE 7（放置状態）	-	Spring 4.x	Boot 1.x	Java EE 開発が止まる Oracle「EE に投資しない疑惑」浮上 開発者とコミュニティが激怒
2017	Java SE 9(**LTS**) - モジュールシステム(Jigsaw) - しかし複雑すぎて普及しない	Java EE 8 (最後の EE) - 3 年ぶりにリリース - しかし“遅すぎた”	-	Spring 5.0 - Java 8 以上必須 - リアクティブ対応	Boot 1.5 - Boot も安定成熟	OracleをJava EEを手放す。 EE の開発を Eclipse Foundation へ移管決定
2018	Java SE 10/11 - Java 11 が LTS（8 の後継） - var 導入（ローカル型推論） - Oracle JDK 有償化	Java EE → Eclipse へ移管完了	**Jakarta EE 8　誕生** - 中身は EE8 と同じ - まだ javax のまま	Spring 5.1	Spring Boot 2.0 - Spring 5 ベース - WebFlux 追加（リアクティブ） - Java 8 以上必須	Jakarta EE 誕生（名称変更の第一歩） "javax"商標は Oracle 所有 → Eclipse は使えない問題が判明 将来 "jakarta.*" への全面リネーム確定（互換地獄の予感）
2019	Java 12 / 13 - リリースサイクル半年ごとに変更 - 小刻みに進化	**(EE 8 で終了済)**	Jakarta EE 8 - 中身は Java EE 8 と同じ - まだ javax.* のまま	Spring 5.2	Boot 2.2	クラウド＆マイクロサービス全盛
2020	Java 14 / 15 - Records(プレビュー) - Pattern Matching	-	Jakarta EE 9 - javax.* → jakarta.* に全面変更 - 互換性地獄の始まり	Spring 5.3 - まだ javax.* を使用 - Jakarta 9 には非対応（様子見）	Boot 2.4	**javax.* の死**

	Java SE	Java EE	Jakarta EE	Spring	Spring Boot	
2021	Java 16 / 17(LTS) - sealed classes - Records 正式版 - Java 17 が"次の標準"に	-	Jakarta EE 9.1 - Java 11 対応版	Spring 5.3.x（長期サポート） -「Spring 6 で Jakarta 対応するから待ってて」宣言	Boot 2.5 / 2.6 - javax 継続使用	
2022	Java 18 / 19	-	Jakarta EE 10 - 内容も進化開始（EE として再起動）	Spring 6.0 - Java 17 以上必須 - 全面 Jakarta 化（jakarta.*） - javax.* 完全排除	Spring Boot 3.0 - Spring 6 ベース - Boot も全面 Jakarta 化 - javax.* 完全排除 - Java 17 以上必須	Boot 3 で javax 完全終了
2023	Java 21 リリース（LTS） Virtual Threads（仮想スレッド）正式リリース → 並行 Record Patterns などモダン化加速	-	Jakarta EE 10 安定 / EE 11 開発中 RESTful Web Services などの改善	Spring 6.x 安定運用 Micrometer, Observability, Native対応 etc. AOT, GraalVMへの対応が強化される	Spring Boot 3.1 / 3.2 AOT最適化、Native Image対応 Virtual Threadsサポート Oak / Micrometer Tracing デフォルト化	Quarkus, Micronaut, Helidonなど軽量Javaフレームワーク台頭 K8s / Cloud Native化がさらに進む
2024	Java 22 / 23 リリース Pattern Matching完成形へ LTSはJava 21のまま	-	Jakarta EE 11 発表準備 / リリース間近 モダンなAPI整理（No more legacy） MicroProfileとの連携強化	Spring 6.x 続く	Spring Boot 3.3 Boot 4 の計画が噂される（Java 21+ onlyになる可能性） Virtual Threads標準化が進む Native Imageが実用段階に	
2025						