Spring release model

Juergen Hoeller
Sébastien Deleuze
Tanzu Division, Broadcom

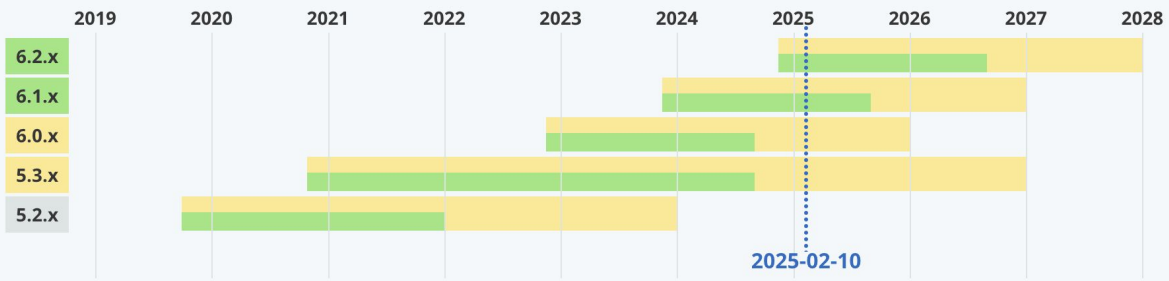
Disclaimer

This slidedeck does not represent the official feedback/opinion of Broadcom Inc. as a company/vendor. It is a pragmatic feedback from Spring engineers to the Java Ecosystem JCP Working Group.

Spring Framework release model on spring.io

Branch	Initial Release	End of OSS Support	End Enterprise Support *
6.2.x	2024-11-14	2026-08-31	2027-12-31
6.1.x	2023-11-16	2025-08-31	2026-12-31
6.0.x	2022-11-16	2024-08-31	2025-12-31
5.3.x	2020-10-27	2024-08-31	2026-12-31
5.2.x	2019-09-30	2021-12-31	2023-12-31

More



More

- 1 year cadence
- Drives infrastructure changes bottom-up
- Defines the JDK baseline
- Many library-like parts
- Common dependency, embedded in many stacks

OSS support

Free security updates and bugfixes with support from the Spring community.

See VMware Tanzu OSS support policy.

Enterprise support

Enterprise support from Spring experts during the OSS timeline, plus extended support after OSS End-Of-Life.

See Tanzu Spring for more details.

Future release

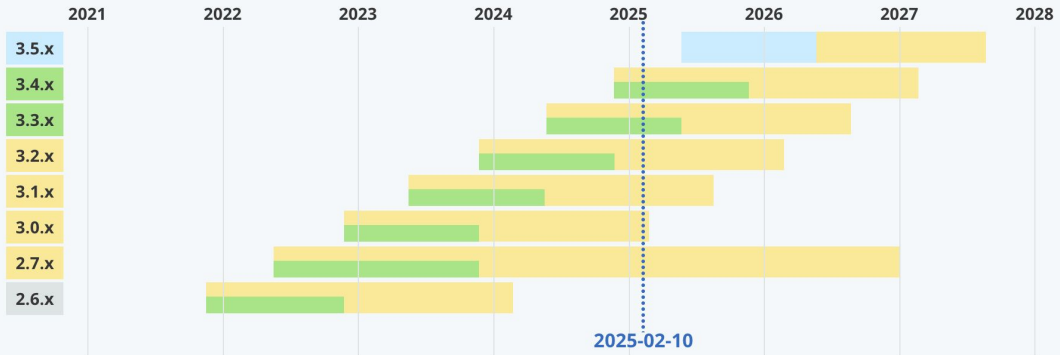
Generation not yet released, timeline is subject to changes.

spring

Spring Boot release model on spring.io (same for portfolio)

Branch	Initial Release	End of OSS Support	End Enterprise Support *
3.5.x	2025-05-22	2026-05-22	2027-08-22
3.4.x	2024-11-21	2025-11-21	2027-02-21
3.3.x	2024-05-23	2025-05-23	2026-08-23
3.2.x	2023-11-23	2024-11-23	2026-02-23
3.1.x	2023-05-18	2024-05-18	2025-08-18
3.0.x	2022-11-24	2023-11-24	2025-02-24
2.7.x	2022-05-19	2023-11-24	2026-12-31
2.6.x	2021-11-17	2022-11-24	2024-02-24

More



- 6 months cadence
- Closer to the application
- Manages many common third-party dependencies
- Provides build plugins
- Can be embedded but often used as a standalone stack

OSS support

Free security updates and bugfixes with support from the Spring community.
See VMware Tanzu OSS support policy.

Enterprise support

Enterprise support from Spring experts during the OSS timeline, plus extended support after OSS End-Of-Life.
See Tanzu Spring for more details.

Future release

Generation not yet released, timeline is subject to changes.

Spring Framework 6

- JDK baseline upgrade from Java 8 to Java 17
  - Spring tries to be a good JDK citizen while ensuring a reasonable level of upgrade disruption for developers
  - Java 17 baseline initially perceived as aggressive when announced but now well accepted and perceived as a good move for the ecosystem
  - Entire framework codebase upgraded to make optimal use of accumulated Java 17 language features: instanceof patterns, records, etc.
- In the same release, a hard upgrade from Java EE 8 to Jakarta EE 9/10
  - Jakarta package namespace as a breaking change has been difficult for users
  - Impacted Spring APIs like Spring MVC
  - This level of breakage is very unusual for Spring developers
- MRJAR allowed more flexibility for specific JDK API support
  - Virtual Thread support implemented with one specific Java 21 class in a single core module

Challenges preventing baseline upgrades

- Spring Framework currently shades ASM for several features:
  - Bytecode generation with CGLIB
  - Reading class metadata from bytecode
  - Spring Framework 7 will use the ClassFile API for reading class files on Java 24+ (via MRJAR)
- Gradle is usually the main blocker for our build

See the table below for the Java version supported by a specific Gradle release:

Table 1. Java Compatibility

Java version	Support for toolchains	Support for running Gradle
8	N/A	2.0
9	N/A	4.3
10	N/A	4.7
11	N/A	5.0
12	N/A	5.4
13	N/A	6.0
14	N/A	6.3
15	6.7	6.7
16	7.0	7.0
17	7.3	7.3
18	7.5	7.5
19	7.6	7.6
20	8.1	8.3
21	8.4	8.5
22	8.7	8.8
23	8.10	8.10
24	N/A	N/A

spring

Spring Framework 7 (WIP)

- For one more generation, staying on a Java 17 minimal requirement
  - Common industry consensus in 2025
  - Aligned with key third-party dependencies: Tomcat 11, Hibernate 7
  - First-class support for newer Java platform features through MRJAR
- Adopting JSpecify over Spring's own nullness annotations
  - For Java tooling (IDEA, NullAway) as well as for Kotlin support
- Jakarta EE 11 baseline
  - EE 11 APIs on a Java 17 baseline
  - Servlet 6.1, JPA 3.2
- At the same time, embracing the latest Java 24/25
  - Java 25 LTS recommended from an application perspective
  - A lot of goodness: ClassFile API, AOT Cache, etc.
  - Virtual Threads without pinning on synchronization:
    expecting a wave of VT adoption, more so than with Java 21

Feedback on the Tip & Tail release model

https://openjdk.org/jeps/14

- Spring mentioned as example for different strategies
  - Spring Framework for multiple release trains
  - Spring Boot for Tip & Tail
- Spring Framework traditionally operates with generations
  - Generational themes spanning multiple feature releases
  - Feature releases being rich but avoiding breakage
  - Baseline upgrades and removals only in new generation
- Pragmatic backporting to all active branches
  - Primarily CVEs and bug fixes
  - Platform compatibility issues
  - Selected performance enhancements

Thanks!