Evolução do Spring Boot

O Spring Boot surgiu como uma solução para simplificar o desenvolvimento com o ecossistema Spring, eliminando a complexidade de configuração e permitindo o desenvolvimento rápido de aplicações. Ao longo dos anos, o framework evoluiu de maneira significativa, refletindo mudanças no ecossistema Java, especialmente com o lançamento de novas versões do JDK. Abaixo, revisamos a trajetória do Spring Boot, incluindo o esperado Spring Boot 4.

Spring Boot 1.x (2014 - 2017)

- Lançamento inicial: A primeira versão do Spring Boot trouxe a abordagem de 'convenção sobre configuração', facilitando o uso do ecossistema Spring com o mínimo de configuração. Incluía recursos como auto-configuration, Spring Initializr e servidores embutidos (Tomcat, Jetty).
- JDK: Suporte ao JDK 7 e JDK 8, com foco na nova abordagem funcional do Java 8 (como Lambdas e Streams).

Spring Boot 2.x (2018 - 2023)

- Evoluções importantes: Introdução de suporte nativo ao Reactive Programming com o WebFlux, melhorias no Spring Actuator para monitoramento de aplicações, e suporte a arquiteturas baseadas em microserviços.
- JDK: A série 2.x suportou o JDK 9 e, posteriormente, o JDK 11 como versão LTS. O Spring Boot 2.4 e posteriores passaram a recomendar o uso do JDK 11.

Spring Boot 3.x (2023 - presente)

- Principais mudanças: Com o lançamento do Spring Boot 3.0 em novembro de 2022, o suporte ao JDK 17 tornou-se o requisito mínimo. Esta versão também fez a transição completa para o Jakarta EE 9, incorporou mais integrações com GraalVM para builds nativos, e aprimorou a segurança e o desempenho.
- JDK: Com base no JDK 17, essa versão aproveitou os novos recursos da plataforma Java, como padrões de código mais modernos e melhorias na segurança.

Spring Boot 4.x (Previsto)

- O que esperar: O Spring Boot 4 ainda não foi lançado oficialmente, mas é esperado que ele traga suporte completo ao JDK 21 (lançado em setembro de 2023), aproveitando inovações como o Project Loom (threads leves) e melhorias em concorrência e desempenho.
- JDK: Espera-se que o JDK 21 seja a base mínima para o Spring Boot 4, proporcionando mais eficiência no desenvolvimento de aplicações empresariais.

Comparação com as versões do JDK

- JDK 8 (2014): A introdução de Streams e Lambda Expressions impactou diretamente o desenvolvimento do Spring Boot 1.x, com código mais funcional e enxuto.
- JDK 11 (2018): Com o Spring Boot 2.1, o JDK 11 passou a ser recomendado, com melhorias na execução de aplicações, segurança, e suporte a longo prazo (LTS).
- JDK 17 (2021): Tornou-se o requisito mínimo para o Spring Boot 3.x, trazendo estabilidade e novos padrões de código. Foi fundamental para a transição para o Jakarta EE 9.
- JDK 21 (2023): O JDK 21, lançado com grandes inovações como Project Loom e Project Panama, deve ser a base mínima para o Spring Boot 4.x, trazendo melhorias em concorrência e desempenho.

Vamos comparar de forma simples como os códigos evoluíram com o tempo, com foco em algumas funcionalidades e melhorias.

1. Spring Boot 1.x (JDK 7 e 8)

Com Spring Boot 1.x, o código era bastante simples, mas ainda focava em muitas configurações manuais. O JDK 8 trouxe grandes inovações, como expressões lambda e a API de Streams, permitindo escrever código mais conciso e legível.

Exemplo:

```
@RestController

Public class HelloController {

  @RequestMapping(“/hello”)

  Public String hello() {

    Return “Hello, World!”;
```

```
  }
}
```

Neste exemplo, ainda tínhamos suporte ao JDK 7 e 8, mas a configuração era mais manual.

2. Spring Boot 2.x (JDK 9, 11)

No Spring Boot 2.x, houve melhorias no suporte a microserviços e introdução de suporte a programação reativa. O JDK 9 introduziu módulos, mas foi com o JDK 11 que vimos a verdadeira estabilidade, com suporte a recursos como o var para inferência de tipos.

Exemplo com JDK 11:

```
@RestController
Public class HelloController {
  @GetMapping(“/hello”)
  Public String hello() {
    Return “Hello, Spring Boot 2!”;
  }
}
```

A mudança para @GetMapping é uma melhoria de simplicidade introduzida no Spring Boot 2.x. Com o JDK 11, o uso de var também pode ser aplicado em outras áreas do código, tornando-o mais conciso.

3. Spring Boot 3.x (JDK 17)

O Spring Boot 3.x exige JDK 17 como o mínimo. Agora temos suporte total para Jakarta EE 9 e builds nativos com GraalVM. O JDK 17 trouxe novos recursos como selagem de classes e padrões mais robustos para melhorar a performance e a segurança.

Exemplo com JDK 17:

```
@RestController
Public class HelloController {
  @GetMapping(“/hello”)
  Public String hello() {
    Return “Hello, Spring Boot 3!”;
  }
  Public sealed interface Greeting permits FriendlyGreeting, FormalGreeting {}
  Public final class FriendlyGreeting implements Greeting {}
  Public final class FormalGreeting implements Greeting {}
}
```

Aqui, usamos o recurso de classes seladas (sealed classes), introduzido no JDK 17, permitindo um controle mais rígido sobre a herança de classes.

4. Spring Boot 4.x (Previsão para JDK 21)

Espera-se que o Spring Boot 4.x aproveite o JDK 21, com foco em threads leves através do Project Loom, otimizando o gerenciamento de concorrência.

Exemplo com threads leves no JDK 21 (hipotético):

```
@RestController
Public class HelloController {
  @GetMapping(“/hello”)
  Public String hello() throws InterruptedException {
    Thread.startVirtualThread(() -> System.out.println(“Hello, Virtual Thread!”));
    Return “Hello, Spring Boot 4!”;
  }
}
```

Neste exemplo, o Project Loom introduzido no JDK 21 permite a criação de threads virtuais, que são mais leves e eficientes do que as threads tradicionais.

Conclusão

A evolução do Spring Boot e do JDK trouxe mais simplicidade, performance e escalabilidade ao código. A cada versão, vemos uma integração mais profunda com as inovações do Java, melhorando a eficiência do desenvolvimento e o desempenho das aplicações.

Links para consulta:

História e evolução do Spring Boot https://spring.io/blog/2022/11/24/spring-boot-3-0-released

Documentação oficial do Spring Boot https://docs.spring.io/spring-boot/docs/current/reference/html/

Notas de lançamento do JDK 21 https://www.oracle.com/java/technologies/javase/jdk-21-relnote.html

Comentários adicionais

O Spring Boot evoluiu ao longo dos anos para acompanhar as inovações da plataforma Java, adaptando-se às novas versões do JDK e introduzindo funcionalidades modernas para o desenvolvimento de aplicações escaláveis e eficientes. O esperado Spring Boot 4 deverá aproveitar as novidades do JDK 21, garantindo que o framework continue na vanguarda do desenvolvimento Java, otimizando o suporte a microserviços, containers e arquiteturas nativas.