Name: Andi Dwi Saputro

Date: April 14th, 2020

---

1. **Apa bedanya Spring dan Springboot ?**

   Spring: framework yang menyuguhkan dukungan infrastruktur yang komperhensif untuk pengembangan aplikasi Java.

   Springboot: merupakan extension dari Spring framework sendiri namun memiliki keunggulan dimana tidak perlu lagi mengkonfigurasi code boilerplate yang dibutuhkan untuk mensetup Spring application.

   Table perbedaan Spring vs Spring Boot:

Cases	Spring	Spring Boot
Dependency	Membutuhkan banyak dependencies untuk satu jenis kebutuhan aplikasi seperti web misalnya.	Cukup dengan satu dependency starter untuk masing-masing kebutuhan aplikasi.
MVC	Membutuhkan setup dispatcher servlet, mappings, dan beberapa konfigurasi pendukung lainnya pada web.xml maupun initializer class. Serta perlu menambahkan anotasi **@EnableWebMvc** pada **@Configuration** class serta harus mendefine sebuah view-resolver.	Cukup dengan sedikit konfigurasi pada properties file seperti untuk resolve view controller. Semua konfigurasi sudah dikemas oleh Spring boot auto-configuration.
Template Engine	Selain harus menambahkan dependensi ***thymeleaf-spring5*** juga harus melakukan beberapa konfigurasi pada view resolver.	Pada spring boot 1 hanya membutuhkan dependency ***spring-boot-starter-thymeleaf***. Namun pada versi 3 dari thymeleaf kita harus menambahkan dependency ***thymeleaf-layout-dialec***t pada dependency management di web app spring boot 2. Tidak perlu konfigurasi apapun lagi.
Security	Diharuskan menambahkan dependency standard ***spring-security-web*** dan ***spring-security-config*** selanjutnya diperlukan konfigurasi dengan membuat class yang mengextends library **WebSecurityConfigurerAdapter** dan menambahkan anotasi **@EnableWebSecurity** untuk mengaktifkannya.	Cukup dengan menambahkan dependency ***spring-boot-starter-security*** saja dan semua dependency terkait yg dibutuhkan akan otomatis di tambahkan ke classpath. Dengan konfigurasi yang sama dengan Spring.
Bootstrapping	Spring melakukan bootstrapping app dengan menggunakan konfigurasi web.xml atau ***SpringServletContainerInitializer*** class sebagai bootstrap entry point.	Spring boot hanya menggunakan fitur Servlet 3 untuk memootstrap aplikasi. Dengan static main class yang memiliki anotasi

		@SpringBootApplication sebagai entry pointnya
Packaging & Deployment	Package management dengan maven/gradle. Tidak memiliki dukungan embedded container. Sering terjadi konflik jar dependency ketika dideploy di lingkungan external container.	Sama seperti Spring support maven/gradle. Namun, memiliki banyak kelebihan proses deployment diantaranya memiliki dukungan embedded container, option untuk exclude depndency2 untuk menghindari konflik jar ketika di deploy di external container. Random port generation utk test-test terintegrasi.
Conclusion	Dapat disimpulkan bahwasanya Spring Boot merupakan extension dari Spring itu sendiri untuk membuat proses development, testing, dan deployment lebih mudah dan efisien.	

2. Kenapa harus pakai Spring dan kenapa harus springboot?

a. Kenapa pakai Spring ?

Spring framework memudakan pemrograman aplikasi java menjadi jauh lebih cepat, mudah, dan aman ketimbang dengan cara konvensional. Spring framework focus pada speed, kesederhanaan, dan produktifitas yang membuat Spring menjadi java framework paling popular digunakan diseluruh dunia (JVM ecosystem report 2018).

b. Kenapa pakai Spring Boot?

Spring boot dapat memudahkan pembuatan stand-alone application yang berkualitas dan berstandar untuk production. Dengan spring boot dapat memudahkan developer dalam efisiensi proses development, testing, dan deployment aplikasi berbasis Spring dengan minimum konfigurasi.

Kenapa lebih mudah dengan Spring Boot? Karena sebelumnya based on my experience dalam membangun project Spring tanpa spring boot cukup kewalahan karena harus mengkonfigurasi banyak hal , mengulang berkali-kali create project baru dan menghadapi banyak error yang terbilang menyebalkan karena saya merasa setup segala macam sudah benar namun tetap error karena ada detil-detil kecil yang *terlewat* dari proses konfigurasi yang cukup banyak itu.

Dengan Spring Boot konfigurasi menjadi minim dan effortless, juga tidak perlu risau dan stressful dengan setup server karena Spring boot sudah embedded servernya di include dari starter. Begitu juga untuk urusan dependency pada Spring Boot sudah tersedia starter packages yang siap pakai.

3. Apa itu Application Starter Package pada spring boot?

Application Starter Package merupakan paket starter spring boot yang berisi boilerplate dependency POM descriptor untuk masing-masing jenis starter. Spring boot memiliki beberapa jenis starter package diantaranya:

a. Web Starter: ***spring-boot-starter-web***
b. Test Starter: ***spring-boot-starter-test***
c. Data JPA Starter: ***spring-boot-starter-data-jpa***

Dan lain sebagainya.

4. **Bisakah kita mengubah web server pada springboot selain tomcat?**

   Setiap spring boot app memiliki embedded web server. Web server tersebut juga bisa diubah konfigurasi nya tidak hanya dengan tomcat. Spring boot memiliki starters package untuk web server lainnya yang sudah include default embedded container diantaranya:

   a. Untuk servlet stack applications ***(spring-boot-starter-web)*** dapat menggunakan diantaranya:
      - Tomcat: ***spring-boot-starter-tomcat***
      - Jetty: ***spring-boot-starter-jetty***
      - Undertow: ***spring-boot-starter-undertow***

   b. Untuk reactive stack applications ***(spring-boot-starter-webflux)*** dapat menggunakan diantaranya:
      - Reactor Netty: ***spring-boot-starter-netty***
        *Dapat juga menggunakan starter tomcat, jetty, ataupun undertow.

5. **POM pada spring apakah tetap dipakai di spring boot?**

   Ya hanya jika kita menggunakan maven sebagain dependency management. POM pada spring tetap dipakai di spring boot, karena POM (Project object model) dibutuhkan sebagai descriptor yang memudahkan untuk management dependency maven.

6. **Apa itu Yaml?**

   YAML (YAML Ain't Markup Language –recursive akronim) merupakan sebuah bahasa serialisasi data yang mudah dibaca oleh manusia yang umumnya digunakan sebagai file konfigurasi pada project software development dan tempat transmisi dan data storing pada suatu aplikasi. Mirip seperti XML namun memiliki syntax yang lebih minim. Menggunakan Python-style untuk indentation-nya.

7. **Bagaimana cara melindungi paket data yang kita akses dari app server, baik itu REST dan lainnya yang berjenis microservices. Apa saja strategi perlindungan data yang dikirim via microservices akses?**

   Terdapat banyak cara untuk melindungi paket data pada microservice yang diakses diantaranya dengan menggunakan metode authentication dimana data yang di kirim ataupun di terima pada lalulintas HTTP diverifikasi terlebih dahulu apakah yang mengakses kredensial data tersebut sama dan benar miliknya atau tidak dengan data kredensial yang berada pada database atau server.

   Strategi atau *pattern* untuk perlindungan data pada microservices rekomendasi menurut **Matt Raible** (https://developer.okta.com/blog/2020/03/23/microservice-security-patterns) diantaranya adalah:

   - **Be Secure By Design**
   - **Scan Dependencies**
   - **Use HTTPS Everywhere**
   - **Use Access and Identity Tokens**
   - **Encrypt and Protect Secrets**

- **Verify Security With Delivery Pipelines**
- **Slow Down Attackers**
- **Use Docker Rootless Mode**
- **Use Time-Based Security**
- **Scan Docker and Kubernetes Configuration for Vulnerabilities**
- **Know Your Cloud and Cluster Security**

8. **Apakah spring boot bisa dipakai membangun jenis aplikasi selain web MVC dan Microservices ? Mana lebih flexible, Spring atau SpringBoot dalam hal ini ?**

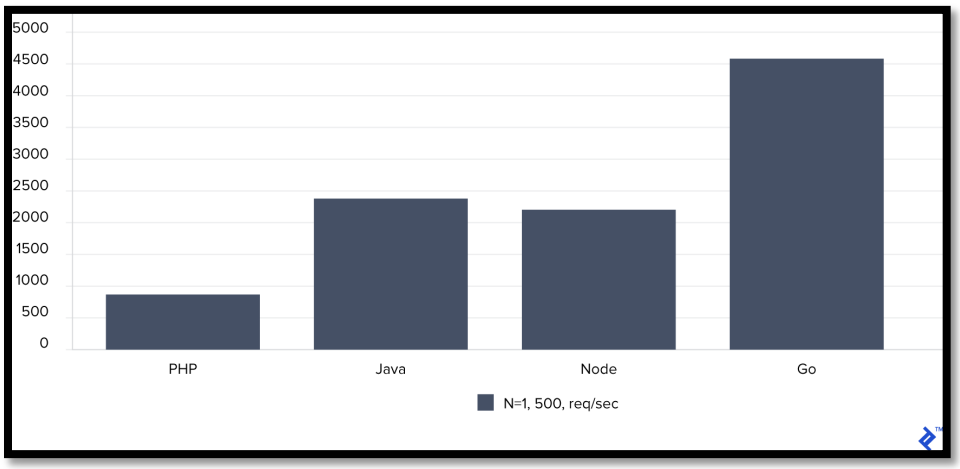
   Selain dipakai membangun aplikasi web MVC dan microservices, Spring Boot juga dapat digunakan untuk membuat **Scalable Serverless Application** yang dijalankan pada platform-platform ***PaaS (Platform as a Service)*** seperti Heroku, GoogleAppEngine, dsb. Dapat membuat aplikasi **Batch** dan **Message Streaming** integrated dengan berbagai software ***message broker*** seperti **RabbitMQ, Kafka, Redis**, dsb.

   Menurut saya Spring dan Spring Boot *seharusnya* sama-sama fleksibel karena Spring Boot mengadopsi keseluruhan fungsi dari framework spring itu sendiri jadi fleksibilitas nya harusnya bisa terbilang sama, namun yang membedakan antara keduanya hanyalah efisiensi proses development, testing, dan deployment nya saja. Untuk Spring Boot jauh lebih ringkas.

   Tetapi secara prioritas Spring framework bisa dikatakan lebih flexible dalam membangun jenis aplikasi lain yang membutuhkan dependency khusus karena Spring framework merupakan *core* yang dapat dikembangkan tools nya untuk keperluan pembuatan jenis aplikasi lain bahkan framework lain yang lebih beragam dimasa mendatang dan Spring Boot akan mengikuti arus implementasinya.

   Namun untuk fleksibilitas dalam memaintain suatu aplikasi maka Spring Boot lebih fleksibel. Itu menurut pendapat saya.

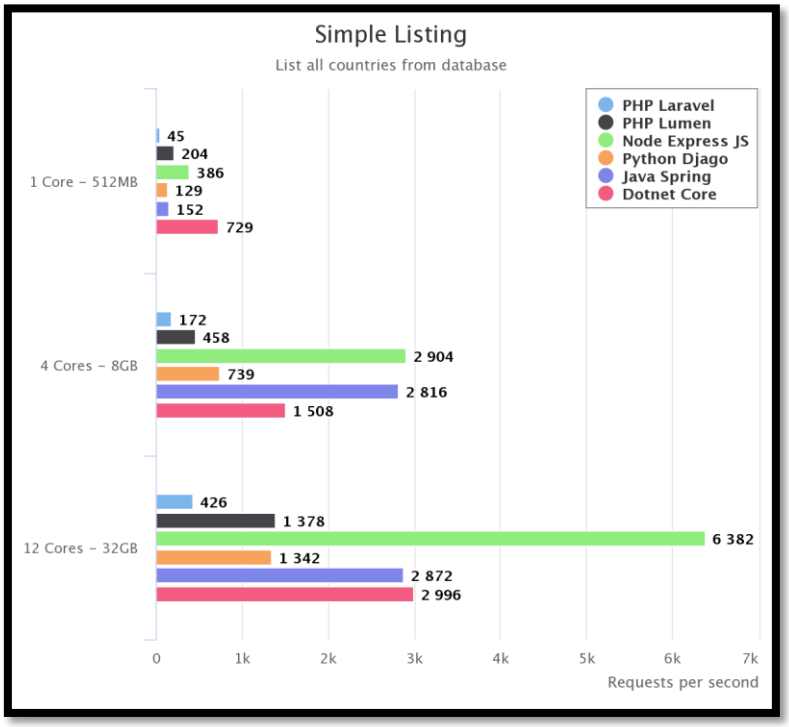
9. **Bagaimana perbandingan kinerja (Benchmark) web apps yang dbangun dari NodeJS, Go, Spring, dan PHP?**



Grafik benchmark Go, Java, PHP, Node.

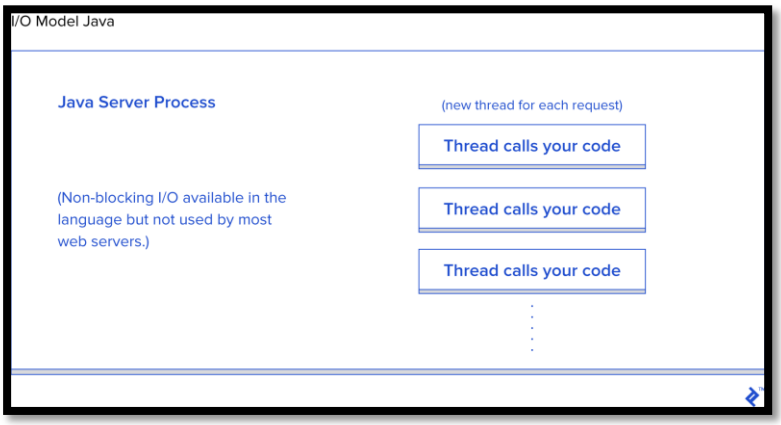
Sumber: https://www.toptal.com/back-end/server-side-io-performance-node-php-java-go

Grafik diatas berdasarkan jumlah request i/o per detik dengan iterasi N=1. Go menjadi MVP disini dengan raihan hingga 4500 req/sec disusul oleh Java (Spring/Boot) kemudian Node dan PHP. Hal ini menandakan bahwa server side technology yang menggunakan ***scripting language*** seperti Node.JS dan PHP memiliki kekurangan ketika menghadapi concurrency request yang banyak. Namun akan lain cerita ketika Node.JS digunakan dengan Express JS salah satu framework Node, maka masalah keunggulan dalam menangani massive concurrency requests bisa berkompetisi *a bit more fairly* dengan Java/Spring.



Grafik Komparasi Benchmark Node Express JS dgn Java,NET,PHP,Python
Sumber: https://medium.com/@mihaigeorge.c/web-rest-api-benchmark-on-a-real-life-application-ebb743a5d7a3

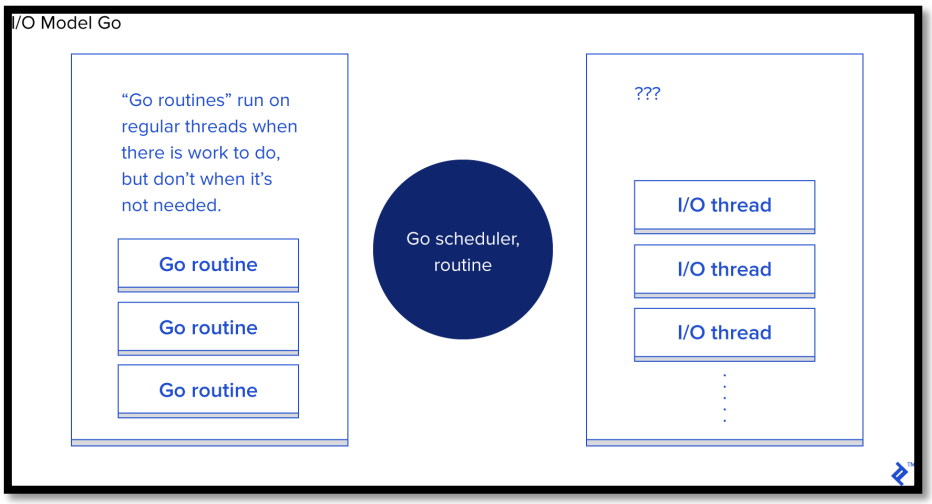
Kenapa Golang bisa lebih unggul dari Spring dan lain nya, padahal sama-sama menggunakan I/O Threading. Berikut adalah I/O model untuk Java/Spring:



Java I/O Model

Java I/O Model mengusung NIO (Non-Blocking Input/Output) tetapi ketersedian NIO tersebut tidak digunakan dibanyak web server java. Java mengandalkan multi thread nya yang digunakan untuk memory management. Saat program java melakukan suatu proses I/O maka **tiap requests** akan **mendapatkan/membuat thread baru** dan operasi I/O yang variatif tsb akan di block didalam thread sampai request tsb sudah ditangani sepenuhnya. Meskipun Thread tersebut memang akan hancur, tapi tetap saja apabila ada ribuan koneksi atau request maka akan menumpukkan ribuan thread yang mana hal tersebut tidak baik untuk Scheduler.

Dan berikut ini adalah I/O Model Golang:



Golang I/O model

Pada Golang untuk menangani proses I/O golang memiliki schedulernya sendiri. Daripada mengeksekusi masing-masing thread yang berkesinambungan dengan sebuah single OS thread, maka golang dapat menggunakan konsep ***GoRoutines*** dengan konsep ini Go runtime akan ototmatis memetakan Goroutines ke setiap OS threads yang dibuat berdasarkan logic dari schedulernya. Setiap requests yang datang dari Go HTTP server ditangani oleh Goroutine yang terpisah.