Migrer à Spring Boot 2 lorsqu'on a une « vraie » application

Speakers Julien Dubois (Ippon)
Format : Tools in action
Date : 18 avril 2018

Julien est le lead-dev du projet Open Source JHipster.
JHipster a permis de générer des centaines de milliers d'applications Spring Boot, de toute taille.
Au travers de ce Tools in action, Julien nous livre son retour d'expérience sur la migration Spring Boot 1.5 vers Spring Boot 2 sur les applications générées par JHipster.

La migration vers Spring Boot 2 de JHipster a duré 4 mois (migration partielle car sans Micrometer). Julien l'estime à 4 semaines de travail effectif. Le projet JHipster reçoit des contributions de personnes reconnues et particulièrement qualifiées.
Pour une application normale, prévoir 2 semaines de migration.

Le « Spring Platform BOM » vient d'être arrêté par Pivotal.
Nécessite de migrer vers le BOM de Spring Boot (ou celui de JHipster).
JHipster propose un BOM intermédiare.

Spring Boot 2 supporte Java 10 :

- Gain de démarrage de 10% sur le poste de dév de Julien. Davantage lié au JDK que Spring Boot 2.
- Image Docker assez volumineuse : alpine linux de 82 MB en Java 8 vs opendk10 de 300 ? MB

Migrer vers Spring Boot 2 entraine des montées de version de frameworks (librairies tierces). Elles sont responsables pour 50% de l'augmentation de la taille du build final.

La montée de version d'Elasticsearch est gênante car le mode embedded a disparu et nécessite désormais d'utiliser Docker (et pourquoi pas le testcontainers-java-module-elasticsearch de David Pilato).

Avant Spring 5, on pouvait utiliser les JDK Proxy. Désormais, l'utilisation de CGLIB devient.

Dans Spring Boot 2, de nombreuses propriétés YAML ont changé. Il est donc nécessaire d'étudier l'ensemble de ces migrations.
L'outil spring-boot-properties-migrator permet de migrer automatiquement les properties. Sinon intelliJ permet de montrer les propriétés qui ne sont plus utilisées.

Les formats JSON renvoyées par les Actuator ont changé. Il est donc nécessaire de modifier les outils qui exploitent les métriques renvoyées par les Actuators. Chez JHipster, cela a pris 1 semaine (charge conséquenque car plusieurs frontend React et Angular).

La montée de version des différents projets Spring impacte la migration.

Spring Data

- L'API a changé suite au Optional de Java 8. Ces changements sont détectés à la compilation. L'utilisation d'Optional peut poser problème sur d'anciennes versions de librairie (ex : MapStruct).
- Changements d'API : findOne() en findById().
- Impacts mineurs nécessitant de refactorer la couche DAO de l'application.
- Julien aurait préféré que les méthodes soient dépréciées.

Spring Cloud

- Spring Cloud est encore en Milestone
- Pas d'impact dans le code
- Les algos de load-balancing ont changé. Les tests de montée en charge et de tolérance aux pannes à refaire, ce qui prend du temps.

Spring WebFlux

- Les tests de l'équipe JHipster ont montré un intérêt limité de passer en réactif
- En JPA, l'impact est essentiellement négatif : le driver JDBC n'est pas réactif
- En NoSQL, l'impact dépend de plusieurs critères
- Intéressant pour les use-cases de streaming : streamer des blobs jusqu'à un client
- Support de Spring Data sur MongoDB/Couchbase/Cassandra pour les streams de données. Migration rapide si Spring Data est utilisé. Si pas de Spring Data, effort important pour migrer en réactif.
- Attention : les éléments dans les collections sont envoyés au fil de l'eau sous forme d'évènements JSON. Cela implique de faire évoluer le code du front.

Spring Security

- Un support de la sécurité a été ajouté pour les applis réactives. Ce support est transparent pour les utilisateurs même si gros travail a été réalisé sous le capot.
- OAuth2 intégré dans Spring Security 5
- Meilleur support d'OpenID Connect

- Attention, le bean AuthenticationManager va être créé automatiquement avec un utilisateur par défaut (avec mot de passe par défaut)
    - Nécessite de déclarer son propre bean
    - Attention à bien retester la sécurité des annotations @Secured => faille de sécurité potentielle

Micrometer

- Nouvelles librairies de monitoring de l'application. Comparaison à SLF4J pour le monitoring. Développé par un développeur de chez Pivotal (anciennement Netflix)
- Activé par défaut dans Spring Boot. Attention à l'impact sur les perfs si d'autres outils sont déjà utilisé (ex : Dynatrace)
- Fonctionne très différemment de Dropwizard Metrics. Migration coûteuse et complexe.
- Librairie très récente, pas forcément très stable.
- JHipster migrera d'ici 6 à 12 mois.

Bonus

- Les bannières Spring Boot peuvent maintenant être animées avec des Gif animés
    - Bloque la console sur le poste de dév
    - Pourri les logs de l'intégration continue

Conclusion

- Positif : nettoyage de code, nouvelles fonctionnalités
- Plus négatif : pas d'énorme gain de perf, coût assez élevé
- Julien conseille de migrer le plus vite possible car les versions antérieures de Spring Boot rentreront en mode maintenance. Migration de raison plus que de passion. Cette migration est toutefois nécessaire.

Zuul 2 et Eureka 2 de Netlix ne sortent pas. Les versions stagnent. Question sur l'avenir de ces outils ?