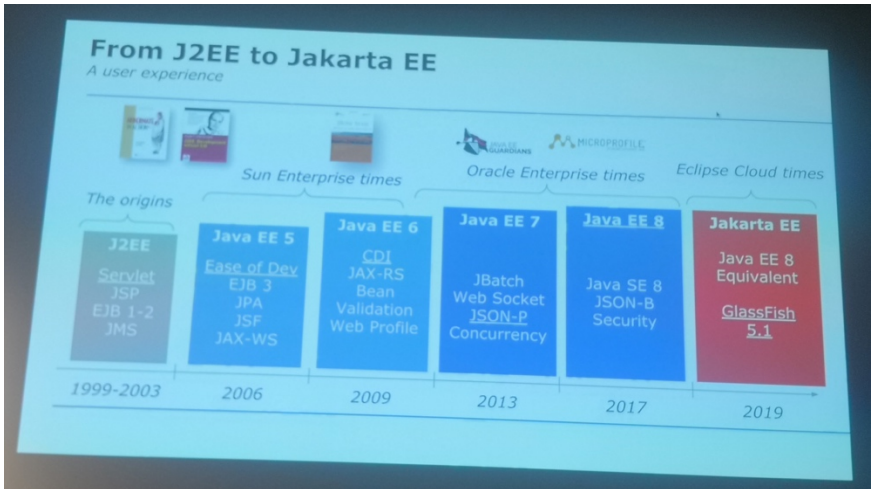
Oubliez Java EE, voilà Jakarta EE !

Speakers : Jean-François James (Atos Wordline), Sébastien Blanc (RedHat)
Format : Conférence
Date : 19 avril 2019

Jean-François et Sébastien comment par nous relater l'historique de Java EE.



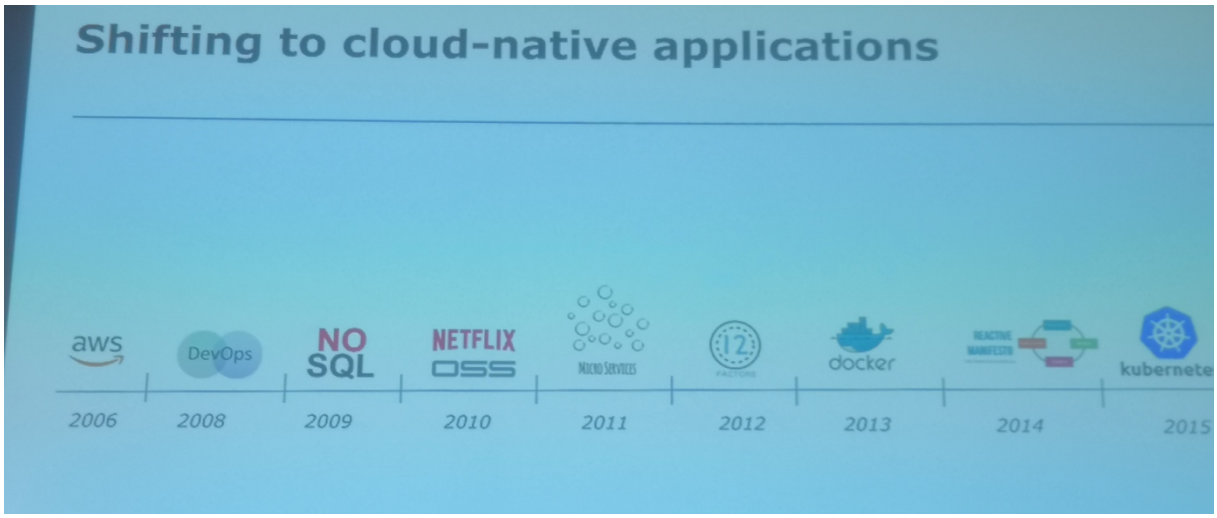
En voici les faits marquants :

- J2EE a commencé en 1999. Jugé trop complexe, d'autres alternatives ont été proposées en 2003 : Spring et Hibernate. Les Servlets ont résisté à l'épreuve du temps.
- Avec la sortie Java EE 5 en 2006, les développeurs ont simplifié J2EE avec l'utilisation des annotations, la création de JPA (inspiré d'Hibernate) et de JSF. C'est l'âge d'or de JEE.
- En 2009, Java EE 6 a été inspiré par JBoss Seam. Java EE devient adulte.
- En 2013, Java EE 7 évolue guère.
- Passent alors 2 ans de silence d'Oracle. JEE n'avance plus. Des éditeurs de serveur d'application, des éditeurs de progiciels et des industriels se réunissent pour initier MicroProfile.
- Java EE 8 est une version de reprise.
- Oracle se désengage de la gouvernance de Java EE. 2018 a servi à transférer Java EE à la fondation Eclipse.
- En 2019, GlassFish 5.1 a été buildé avec les assets de la fondation Eclipse. Preuve que le transfert est effectif.

Alors que l'émergence de standards sont lents et qu'ils ne sont jamais là où on les attend, les speakers se posent l'intérêt des standards en 2019 ?
Les standards ont toutefois quelques avantages :

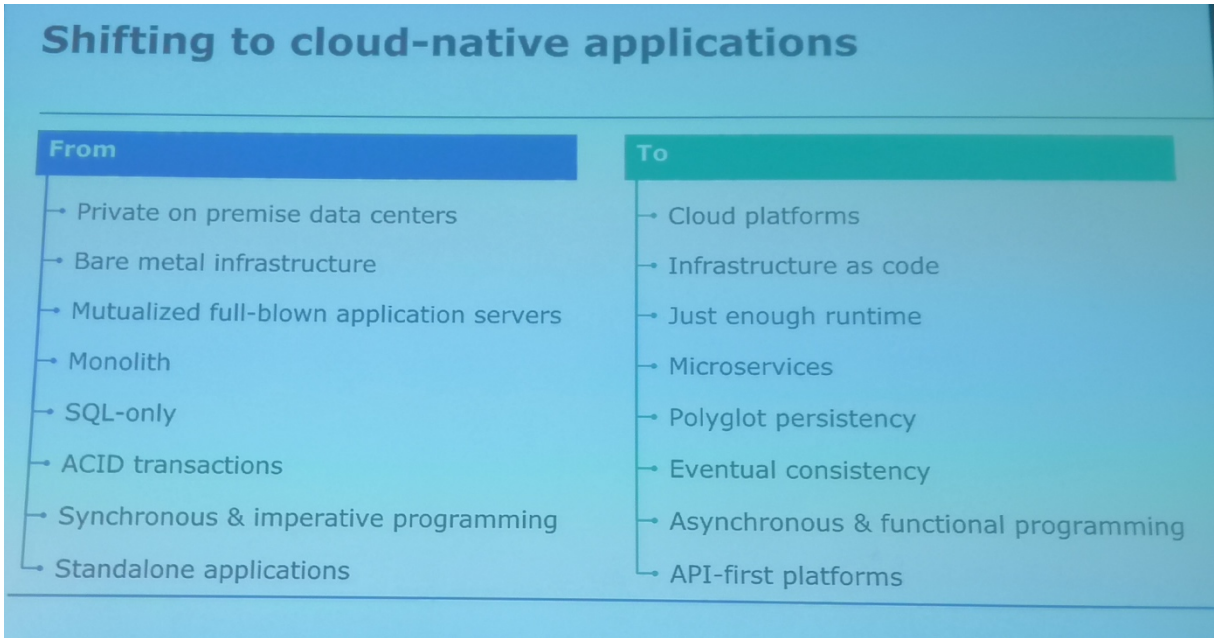
- D'un point de vue d'un utilisateur, cela protège du Vendor lockin
- Les utilisateurs peuvent participer et influencer les standards

En parallèle de J2EE, depuis 2006, on assiste à la naissance de la Cloudification



Là encore, voici quelques faits marquants :

- Netflix bascule son data center chez AWS
- Netflix open source ses librairies cloud Hystrix, Eureka …
- Le mouvement DevOps apparaît. On essaie de ne plus déployer de gros monolithes plusieurs fois par jour. Apparaissent alors les microservices
- Pour faciliter le déploiement des microservices, on les embarque dans des conteneurs. Et pour orchestrer le tout, on ajoute du Kubernetes qu'on saupoudre avec du Istio.



Java EE n'a malheureusement pas pris à temps le virage du Cloud : il n'a aucune JSR sur les métriques, le circuit breaker, la tolérance aux fautes … Oracle n'a rien souhaité faire. MicroProfile a été créé pour compenser les manques de Java EE.

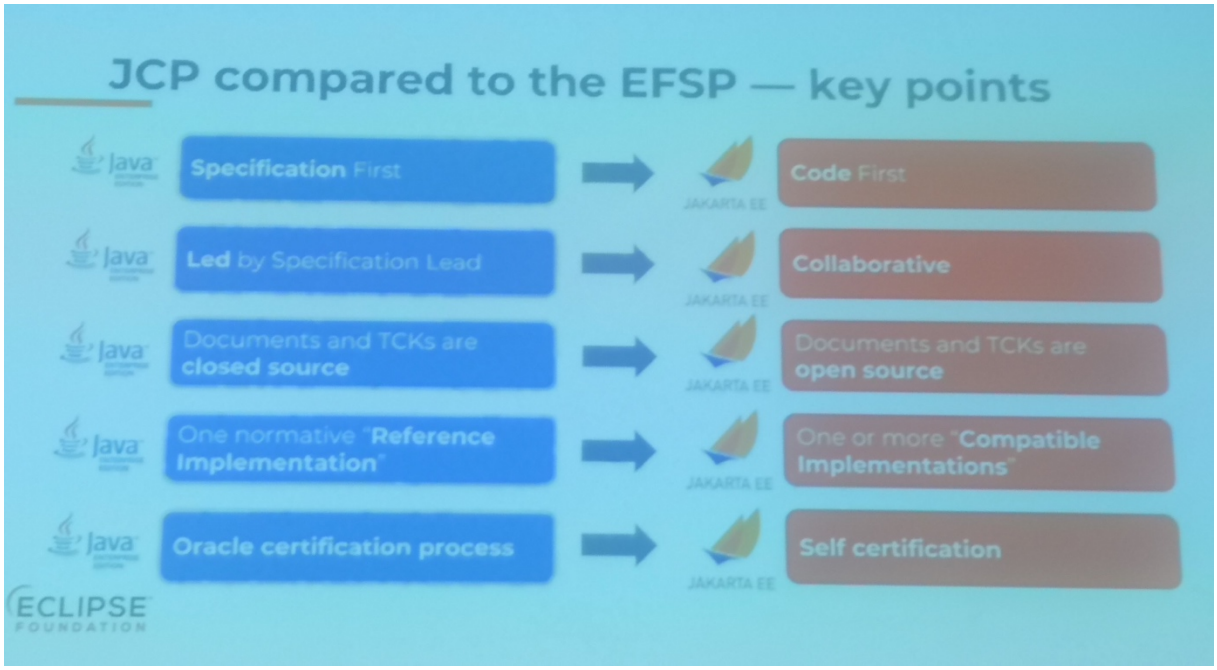
Avec le transfert de Java EE à la fondation Eclipse, ce sont 20 ans de travail de recherche et de spécifications qui est confié à la communauté. La fondation Eclipse accueille les 13,5 millions de LOC et les 95 000 fichiers.

EE4J est le top level project relatif à Java EE. On y retrouve Jakarta EE, le TCK et la documentation. Jakarta EE est le nouveau nom de Java EE.

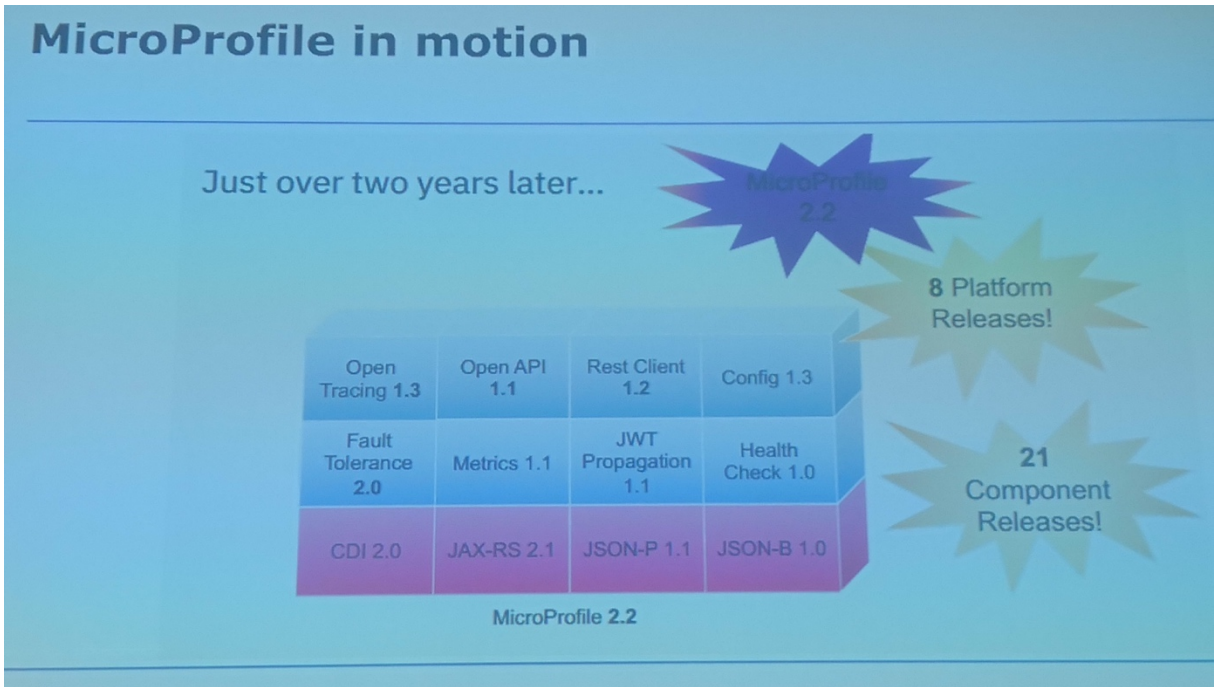
Fait intéressant : le projet MicroProfile est également hébergé par la Fondation Eclipse. Les nouvelles idées vont incubées dans MicroProfile puis seront ou non rebasculées dans Jakarta EE.
Connu par son IDE, la Fondation Eclipse héberge plus de 350 projets. Elle est financée par de grosses sociétés telles Google et Pivotal.
Les projets connus d'Eclipse sont Jetty, Eclipse IDE, Vert.X, JNOSQL (ressemble à Spring Data), Eclipse Link (implémentation de JPA), DL4J, Microprofile, OpenJ9 (alternative à HotSpot)

Le changement de gouvernance apporte plusieurs nouveautés :

- Approche Code First (vs Specfication first) des JCP, même si dans les faits le JCP l'avait déjà adopté.
- Le TCK était payant et son accès était opaque : il devient Open Source et fait partie des 13,5 millions de LOC transférées
- Il n'y a plus d'implémentation de référence, mais des implémentations compatibles.
- Le processus de certification n'est plus contrôlé par Oracle. Il y'a une self certification, ce qui permet d'accélérer les implémentations.



Microprofile s'appuie sur les briques usuelles de Java EE (ex : JAX-RS, JPA) et ajoute des briques pour la fault tolerance, la configuration, les métriques, le health check, le support d'Open API …



Certains éditeurs supportent Jakarta EE et Microprofiles (IBM). D'autres n'implémentent que Jakarta EE ou Microprofile.

Cette conférence se termine par 2 démos.
Dans une 1ière démo, Jean-François déploie 3 applications avec Payara. Il utilise 3 modes de déploiement : war, fat-jar (uber-jar) et all jars. Il nous montre comment le Health Check de MicroProfile permet de surveiller les threads, la data source, le filesystem …
La description de l'API REST est décrite à l'aide d'OpenAPI (Swagger 3).
Dans MicroProfile, il y'a un client REST : RegisterRestClient.
L'IHM GraphiQL permet d'interroger l'API GraphQL.

Sébastien conclue par une démo express de Quarkus. A ses dires, Jakarta EE devrait s’en inspirer.