From Java EE to Jakarta EE

A user experience

A few words about me

@jefrajames

Worldline
EXPERT
community

Speaker me = SpeakerOf.setLastName("**James**")
.setFirstName("**Jean-François**")
.setBackgroundInYears(**32**)
.setMindset("**DevOps**")
.addSkill("**Unix**", LocalDate.of(**1988,** MONTH.January, 1))
.addSkill("**Software Architect**", LocalDate.of(**1990,** MONTH.January, 1))
.addSkill("**Java**", LocalDate.of(**1997,** MONTH.January, 1))
.build();

User perspective?

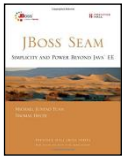
Agenda

- The past
- The transition
- The future

The Past

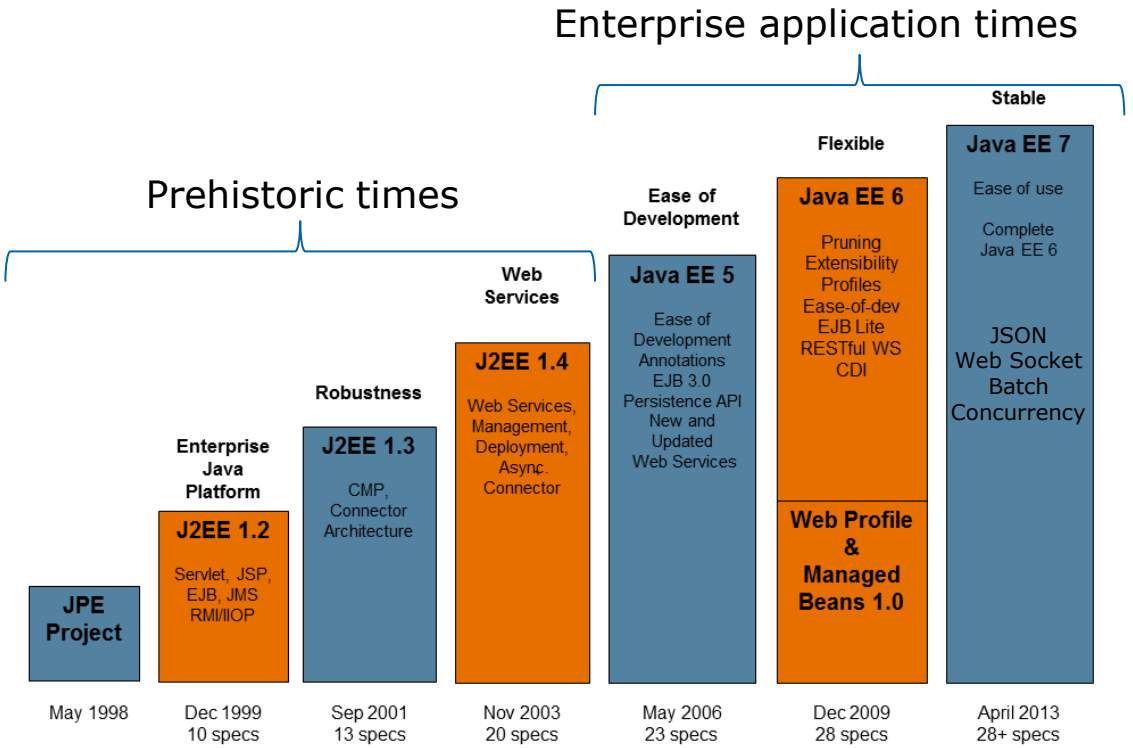
Where do we come from?

From J2EE to Java EE



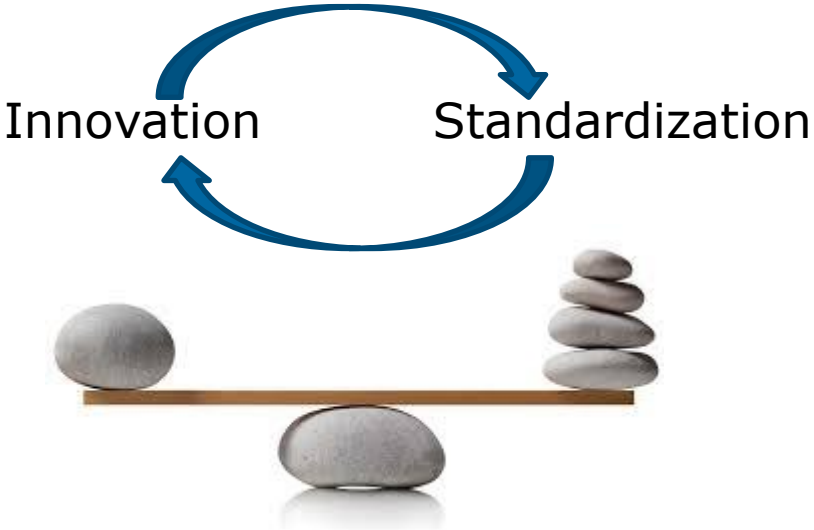
2007

2004



worldline
an atos company

Innovation vs standardization



"Java EE is the place to standardize innovation in the Enterprise Java space"

Enterprise applications

Moderate
Predictable
scalability

Synchronous & imperative programming model
Strict constency

Slow release
cadence

App server

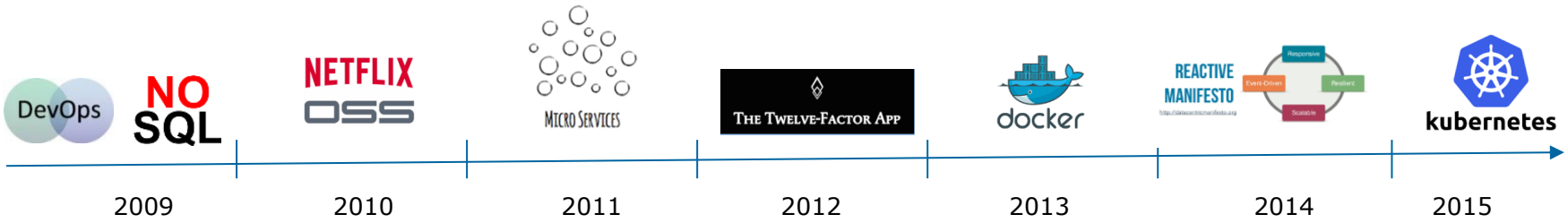
On Premise

worldline
an atos company

The transition

From enterprise to cloud-native applications

Shifting to cloud-native applications



worldline
an atos company

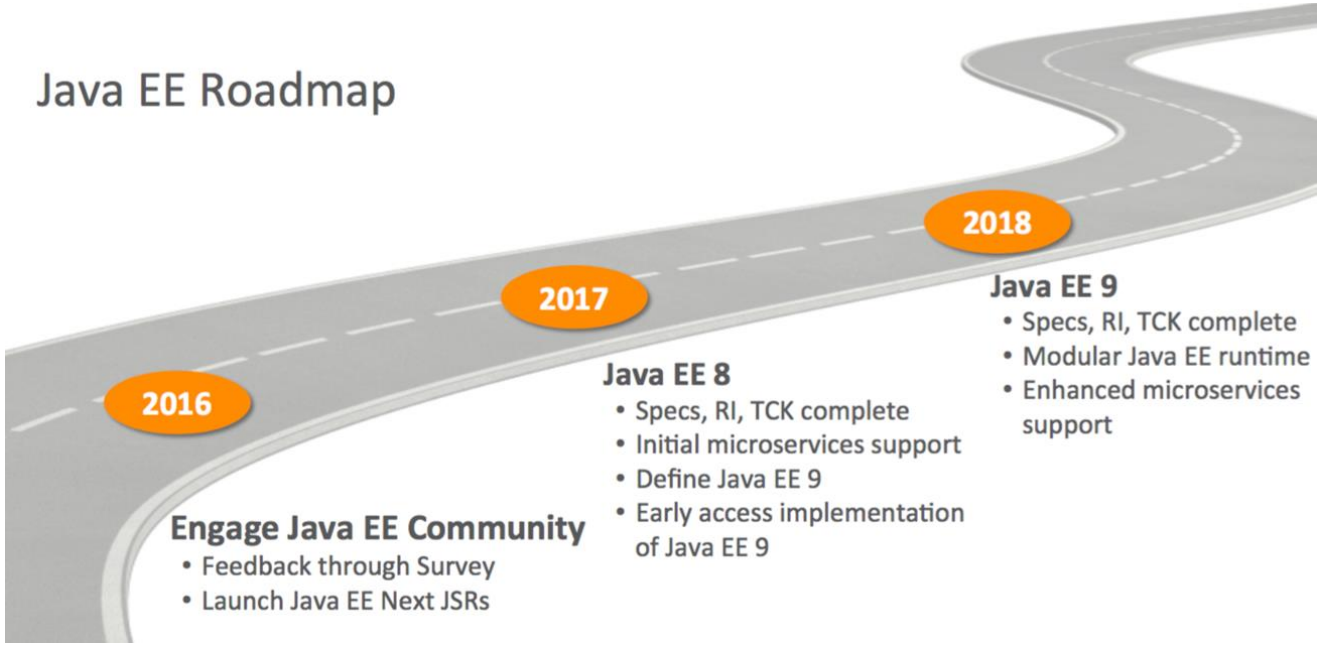
Digitalization in action

- Digital platform vs traditional application
- Time-to-market vs release plan
- DevOps vs silos
- API Economy vs standalone application
- Diversity of devices
- Cloud vs on-premise-only infrastructure
- Infrastructure As Code vs bare-metal
- Polyglot persistency vs SQL-only
- Microservices vs monolith
- Reactive & functional vs synchronous & imperative programming
- Eventual vs strict consistency

The long road to Java EE 8

- Started in 2014
- Inactive until Java One 2016
- Two initiatives from the community in S1-2016:

JavaOne 2016: on the road again



What's new with Java EE™ 8?

- Released on 21 Sept. 2017
- Repository moved to **GitHub** (java.net decommissioned)
- **Glassfish 5.0** Reference Implementation

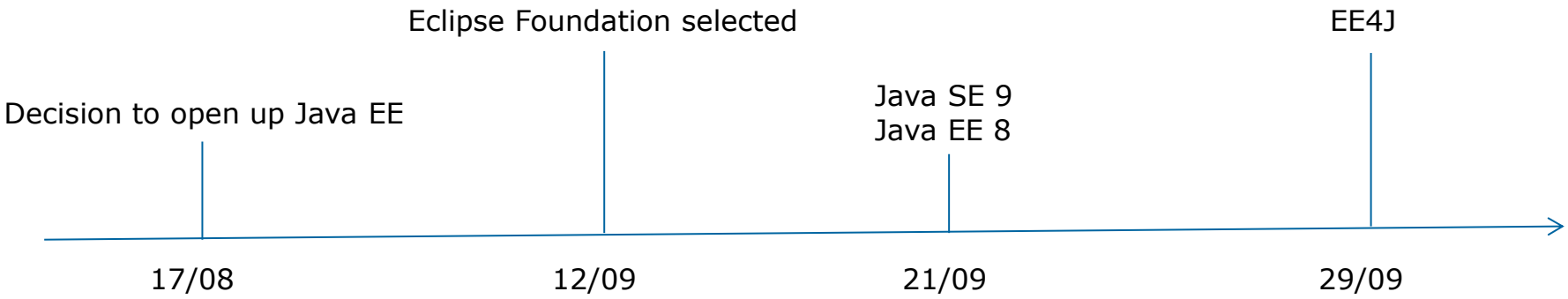
- **Java SE 8** support: DateTime API, CompetableFuture, repeatable annotations
- **Servlet 4.0:** HTTP/2 support (Server Push)
- **CDI 2.0**: asynchronous events, events ordering, better integration in other specs
- **JAX-RS 2.1:** Server Sent Event, reactive extensions
- **JSON Processing 1.1** and **JSON Binding 1.0**
- **Security:** simplification, secret management, modernization, OAuth2, OpenId support

Entering a turbulent zone!

- Future of Java EE to be invented

- Disruptive changes in Java SE:
  - Java Platform Modular System:
    - Modularity at the heart of the JDK
    - And your applications?
  - New release cadence

2017: 43 days of change



worldline
an atos company

The future

Upcoming challenges

Challenge 1: open specification vs standard

"The mechanism for developing **standard technical specifications** for Java technology"

worldline
e-payment services

Challenge 1: open specification vs standard

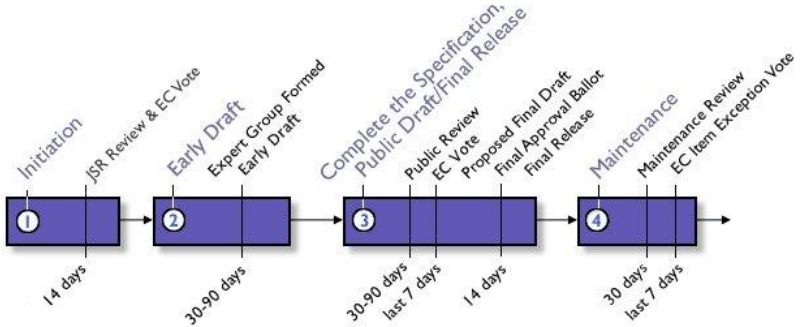
worldline
e-payment services

Challenge 2: strong & fast governance

- Each individual JSR managed by an Expert Group

- 3 deliverables:
  - A Specification Document
  - A Test Compatibility Kit
  - A Reference Implementation

- Well-defined lifecycle:



worldline
e-payment services

Challenge 2: strong & fast governance

- Individual projects
- Overall coordination by top-level EE4J project
- Jakarta EE working group:
  - specification process
  - brand management

worldline
e-payment services

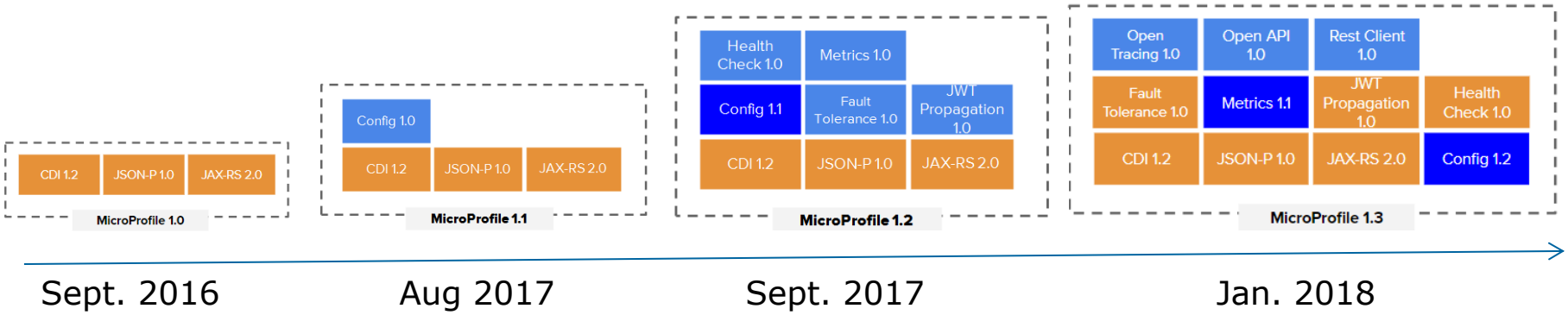
Challenge 3: cloud-native applications

- 3 years to release Java EE 8 and it is not cloud-native
- Competition has been more agile: cloud-native frameworks are here!
- Transitioning to Jakarta EE will take time:
  - 39 projects to transfer
  - Glassfish 5.1 scheduled in Q3-2018 (Java EE 8 certified)
  - Glassfish 5.2 scheduled in Q4-2018 (Jakarta EE 8 certified)

- Cloud-native work will start at the earliest by the end of next year

worldline
e-payment services

Challenge 3: cloud-native applications



worldline
e-payment services

Challenge 4: modular platform

- “Just-enough-runtime” is the new norm

- Modular specifications :
  - A set of independent but consistent specifications
  - Adopting JPMS:
    1. Automic-Module-Name in MANIFEST jar file
    2. Using the modular Java Service SPI (uses & provides with)

- Modular and flexible application servers:
  - Both standalone and embedded

worldline
e-payment services

Challenge 5: syncing with Java releases

OpenJDK

- Feature releases:
  - Bring innovation
  - 2 releases/year (March, Sept.)
  - Not production-ready: support limited to 6 months, no overlap

- Long Term Support releases:
  - Ensure production stability
  - 1 release/3 years
  - Commercial support required

worldline
e-payment services

Challenge 5: syncing with Java releases

- My 2 cents …
- Jakarta EE should leverage the current Java SE LTS
- 1 Jakarta EE release/year may be a good fit

Other challenges

- Overall consistency between specifications: EJB deprecation …
- Ease of testing
- Flexibility to quickly incorporate new technologies: NoSQL, Kafka, GraphQL …
- Migration path

Conclusion

- Java EE is dead but it has a promising follow-up: Jakarta EE
- A vendor-neutral open specification really makes sense
- Eclipse Foundation seems to be the place-to-be
- Eclipse Microprofile has already paved the way
- Let’s go for another 20 years!

As a (modest) user, you can be part of the story!

- The next two years will be decisive
- Stay tuned:
  - Visit the Jakarta EE Web Site: https://jakarta.ee/
  - Follow @JakartaEE on Twitter
  - Become a @javaee_guardian
- Learn Java EE 8 and Microprofile
- Test, provide feedback

Thanks!

an atos company

worldline
e-payment services