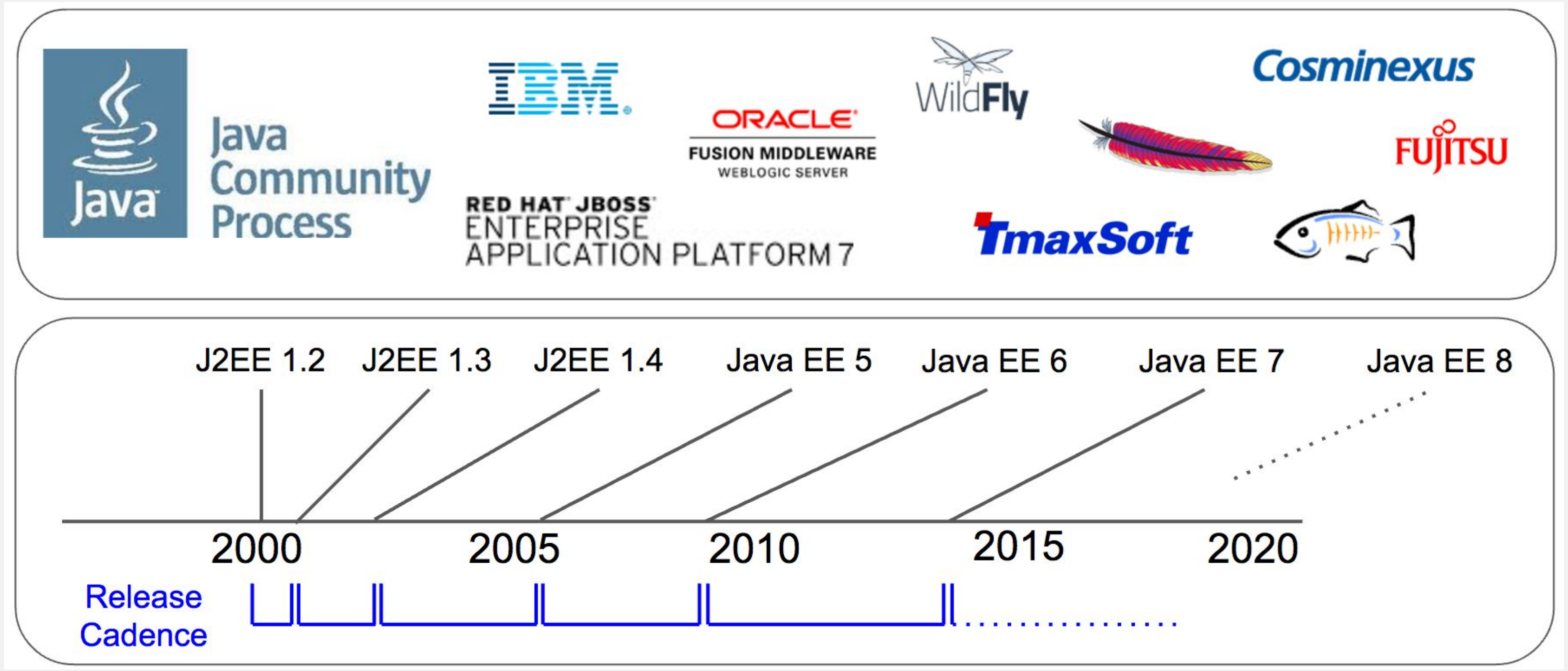
RED HAT SUMMIT

MicroProfile: Optimizing Java EE For a Microservices Architecture

John Clingan
Senior Principal Product Manager

Ken Finnigan
Principal Software Engineer

Enterprise Java Standards History



MicroProfile Background

- Many innovative “microservices” efforts in existing Java EE projects
    - WildFly Swarm
    - WebSphere Liberty
    - Payara
    - TomEE
  - Projects already leveraging both Java EE and non-Java EE technologies
  - Creating new features/capabilities to address microservices architectures

- Wanted to avoid splitting into separate communities

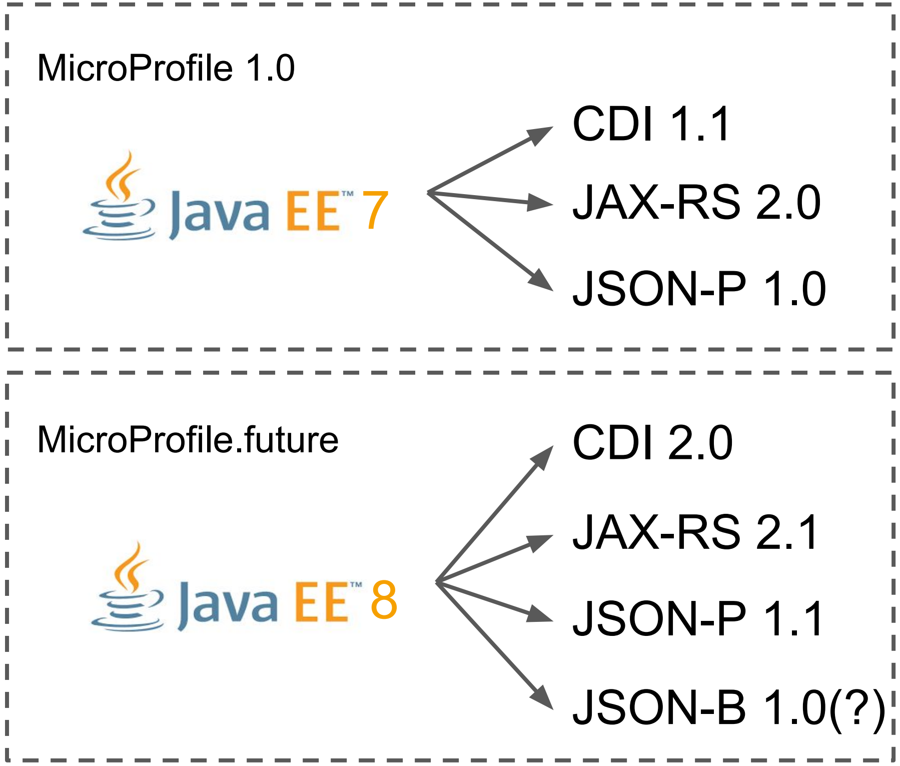
- *So we are collaborating in one community!*

An Eclipse Foundation Project

- Meritocracy; vendor neutrality
- MicroProfile leadership can change over time
- Legal and technical infrastructure
- Trademark Ownership
- Accepts Apache License

Paving a Path to Microservices
First: Leverage Java EE

- Leverages Java EE technologies most relevant to microservices
- Customers can leverage knowledge and expertise
- Facilitate customer, vendor, partner adoption



Paving a Path to Microservices
Second: Organic Innovation

- Begin with well-known microservices patterns
- Develop CDI-centric programming model to support them

Examples

Configuration 1.0

Security: JWT Token Exchange 1.0

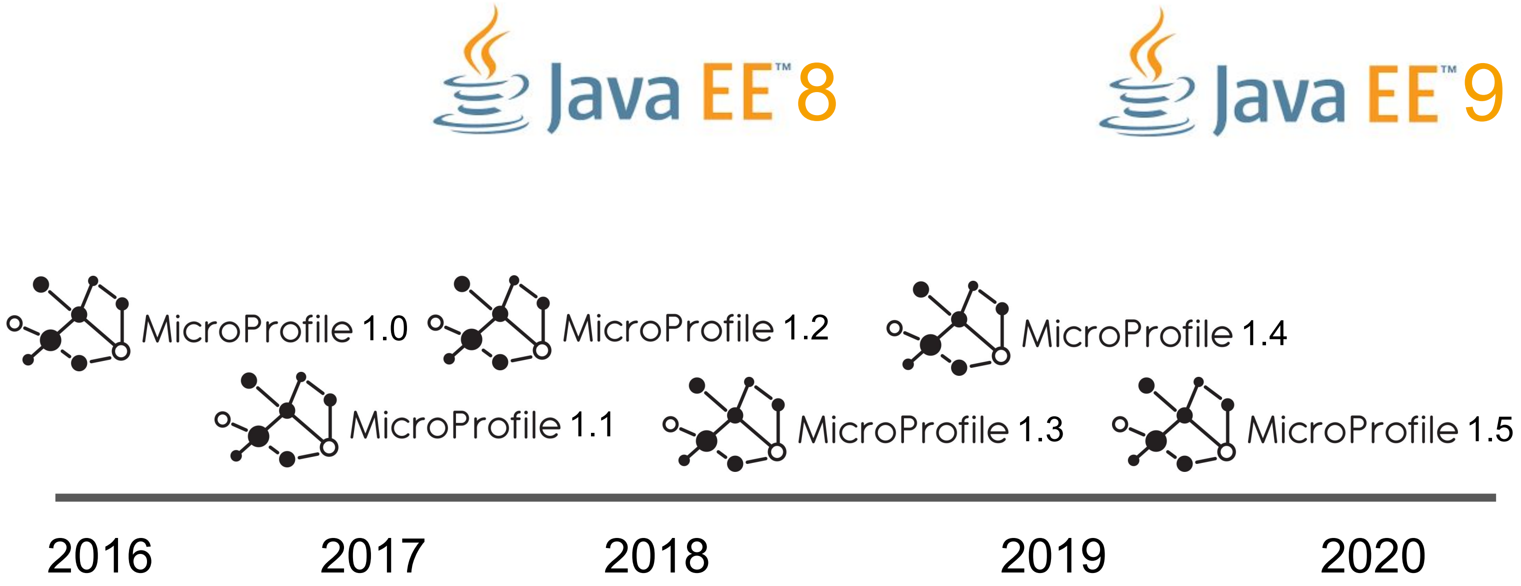
Health Check 1.0

Fault Tolerance 1.0

Paving a Path to Microservices
Third: Collaborate in Open Source

- Build a strong community
- Collaborate on specifications
- Encourage multiple implementations
- Standardize technologies when ready

Quickly Put Features in Developers Hands



* 2-4 releases per year

MicroProfile Roadmap (Under Review)

Releases

MicroProfile 1.1
(Q2, 2017)

MicroProfile 1.2
(Q3, 2017)

MicroProfile 2.0
(Q4, 2017)

MicroProfile Roadmap (Under Review)

Releases

MicroProfile 1.1
(Q2, 2017)

MicroProfile 1.2
(Q3, 2017)

MicroProfile 2.0
(Q4, 2017)

Feature Backlog

Configuration
Baseline Fault Tolerance
JWT Security Token Exchange
Health Check
Fault Tolerance w/event streams
Monitoring
OpenTracing
CDI 2.0
JAX-RS 2.1
JSON-P 1.1
JSON-B 1.0(?)

MicroProfile Roadmap (Under Review)

Releases

MicroProfile 1.1
(Q2, 2017)

MicroProfile 1.2
(Q3, 2017)

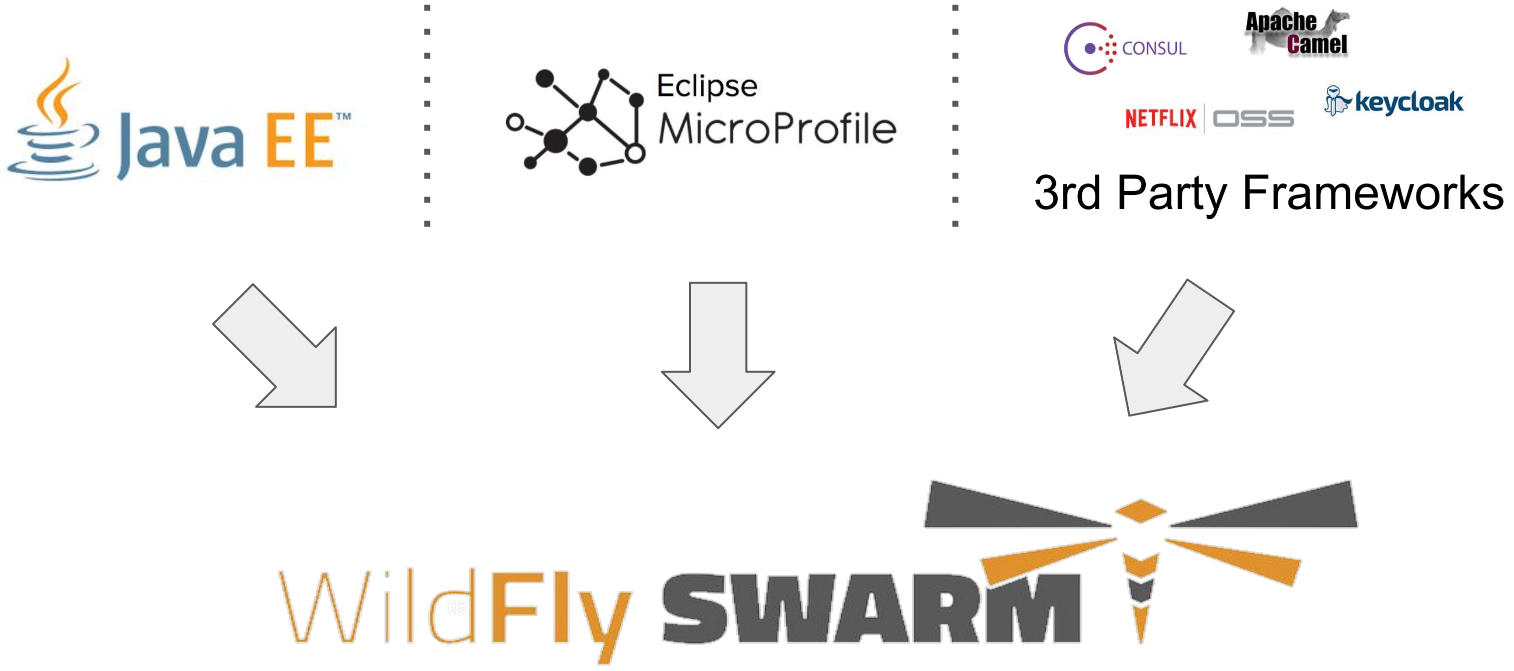
MicroProfile 2.0
(Q4, 2017)

Decided

Feature Backlog

Configuration
Baseline Fault Tolerance
JWT Security Token Exchange
Health Check
Fault Tolerance w/event streams
Monitoring
OpenTracing
CDI 2.0
JAX-RS 2.1
JSON-P 1.1
JSON-B 1.0(?)

Practical Usage of MicroProfile



Collaboration

- Discussions via Google Group
- Reach consensus, no “single power”
- Ideas, thoughts, views all welcome

Spec Proposal Process

- All proposals are submitted via Pull Request to a GitHub repository
  - https://github.com/eclipse/microprofile-evolution-process
- Recommend initial discussion on Google Group prior to PR
  - General view on proposal and interest

Spec Proposal Process

- Submit PR following template
  - https://github.com/eclipse/microprofile-evolution-process/blob/master/0000-template.md
- Don’t need to provide example APIs
- Define use cases that motivated the proposal
- Outline possible solution ideas, if applicable
  - Don’t need full solution to submit proposal

Spec Proposal Process

- Follow up with Google Group thread announcing PR has been made
- Various discussions will happen within the Pull Request
- Author of Pull Request needs to:
    - Reflect consensus view of feedback into updates of the proposal
    - Provide reasoning as to why a suggestion may not be applicable

Spec Proposal Process

- Depending on voracity of discussion, PR may remain open for a couple of weeks or month(s)
- When reasonable consensus reached
  - PR is merged
  - GitHub repository for proposal created in Eclipse organization
  - Work on specification document, APIs, and testsuite (TCK) commences

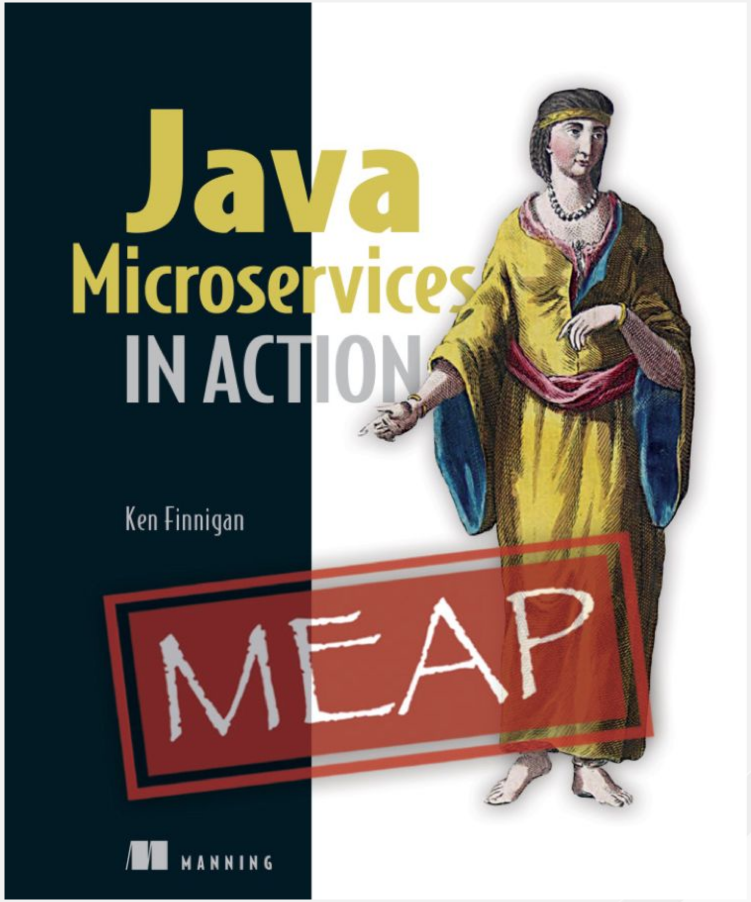
How To Get Involved?

- Google Group for discussions
  - https://groups.google.com/forum/#!forum/microprofile
- Eclipse MicroProfile
  - https://projects.eclipse.org/proposals/eclipse-microprofile
- MicroProfile site
  - http://microprofile.io/

Java Microservices Book

- Recently released into MEAP
- Uses WildFly Swarm
- 39% discount on all Manning books with code: **ctwrhsummit17**

https://www.manning.com/books/java-microservices-in-action



RED HAT SUMMIT

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos

RED HAT SUMMIT

LEARN. NETWORK.
EXPERIENCE
OPEN SOURCE.

redhat.

INSERT DIVIDER COPY

BACKUP SLIDES

Configuration API

Get all the configuration properties that are visible:

```
@Inject
Config config;
```

Configuration API

Get specific property value (static):

```
@Inject
@ConfigProperty(name = “myProp”, defaultValue = “defValue”)
String myProperty;
```

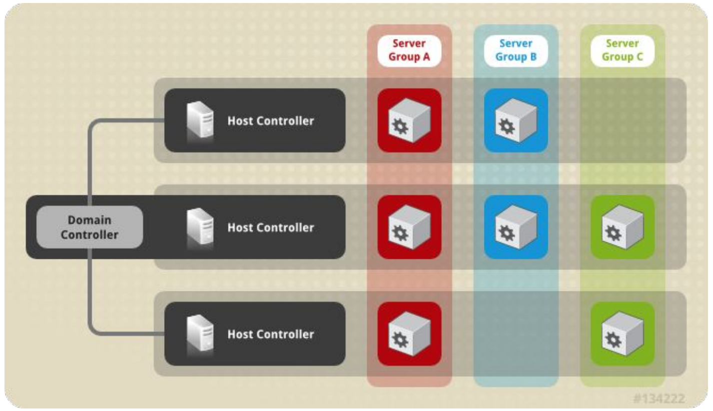
Configuration API

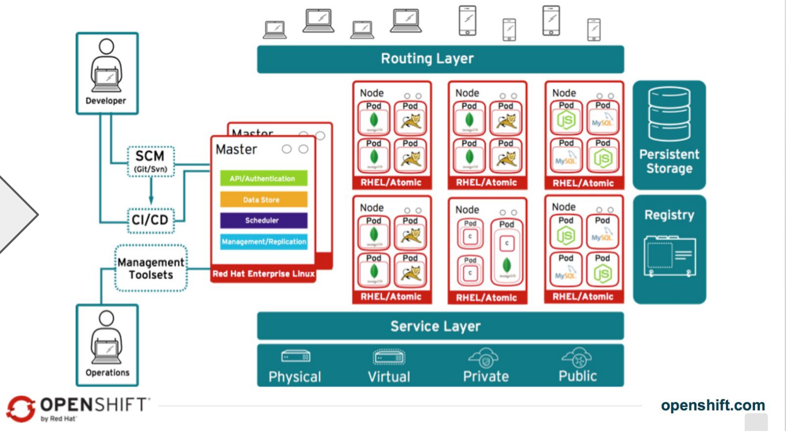
Get specific property value (dynamic):

```
@Inject
@ConfigProperty(name = “myProp”, defaultValue = “defValue”)
Provider<String> myProperty;

String getValue() {
    myProperty.get();
}
```

Changing Definition of “Platform”





May remove from this particular slide deck