MicroProfile and Jakarta EE -- What’s Next?

Kevin Sutter (@kwsutter), MicroProfile and Jakarta EE Architect @ IBM
Ian Robinson (@ian_robinson), IBM Distinguished Engineer & WebSphere Chief Architect
What’s Next?

Kevin Sutter (@kwsutter), MicroProfile and Jakarta EE Architect @ IBM
Ian Robinson (@ian__robinson), IBM Distinguished Engineer & WebSphere Chief Architect

#1 Better Conference Scheduling
About Me

• IBM Senior Technical Staff Member (STSM)
• WebSphere MicroProfile and Jakarta EE Architect
• WebSphere development, open source, and enterprise Java standards
  • WebSphere, Liberty, Open Liberty, Java EE, Jakarta EE, MicroProfile, JPA, JCA, ...
• Based in IBM’s Rochester MN lab
JavaOne 2015: Java EE – Relevant or Elephant?

What can we do to advance microservice development in the Enterprise Java space?

- Java EE Community, early 2016
Characteristics

- Community Driven
- Lightweight, Iterative Processes
- No Reference Impl!
- Specs, APIs, TCKs
JavaOne 2016

MicroProfile 1.0 Announced!

Basic Building Blocks for Microservices

CDI 1.2  JAX-RS 2.0  JSON-P 1.0
Fast-forward two years...

MicroProfile 2.1

7 Platform Releases!

17 Component Releases!

<table>
<thead>
<tr>
<th>Open Tracing 1.2</th>
<th>Open API 1.0</th>
<th>Rest Client 1.1</th>
<th>Config 1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault Tolerance 1.1</td>
<td>Metrics 1.1</td>
<td>JWT Propagation 1.1</td>
<td>Health Check 1.0</td>
</tr>
<tr>
<td>CDI 2.0</td>
<td>JAX-RS 2.1</td>
<td>JSON-P 1.1</td>
<td>JSON-B 1.0</td>
</tr>
</tbody>
</table>

MicroProfile 2.1
Roadmap...

<table>
<thead>
<tr>
<th>Reactive Operators 1.0</th>
<th>Reactive Messaging 1.0</th>
<th>Service Mesh / ISTIO 1.0</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Tracing 1.2</td>
<td>Open API 1.0</td>
<td>Rest Client 1.1</td>
<td>Config 1.3</td>
</tr>
<tr>
<td>Fault Tolerance 1.1</td>
<td>Metrics 1.1</td>
<td>JWT Propagation 1.1</td>
<td>Health Check 1.0</td>
</tr>
<tr>
<td>CDI 2.0</td>
<td>JAX-RS 2.1</td>
<td>JSON-P 1.1</td>
<td>JSON-B 1.0</td>
</tr>
</tbody>
</table>

Component Updates
- Health Check 1.1
- Rest Client 1.2
- Config 1.4
- Metrics 2.0
- Fault Tolerance 1.2
- ...
Compatible Implementations!

https://wiki.eclipse.org/MicroProfile/Implementation
Meanwhile...
39 EE4J Projects and counting...

- 20% Project Proposal has been posted for community review
- 40% Project committers and resources have been provisioned
- 60% Initial Contribution provided to the Eclipse IP Team
- 80% Initial Contribution Pushed to Git Repository
- 100% Project has engaged in its first Release Review

https://www.eclipse.org/ee4j/status.php
EE4J CI/CD Progress

Eclipse Glassfish 5.1-RC1!

https://github.com/orgs/eclipse-ee4j/projects/1
Specification Process

Specification Process 1.0

- Replacement for JCP Standards Process
  - [link](https://docs.google.com/document/d/1ongNUyGvZPtrcuxA7RQNV2qBESFcAphQbzdk3pUIQ5s/edit#heading=h.o84um3qls283)
  - Comments accepted until Oct 31, 2018
- Guinea Pig Spec Project
  - JNoSQL
Additional Work Items in progress...

- TCK Compliance Process
- TCK License
- Specification License
- ...
Similarities between MicroProfile and Jakarta EE

- Focused on Microservice and Cloud Native Development
- Desire Agile Development Practices
- Code First Mentality
Characteristics

Community Driven

Lightweight, Iterative Processes

Specs, APIs, TCKs

No Reference Impls!
Differences between MicroProfile and Jakarta EE

- MicroProfile has an established release process and cycle
  - 7 major releases and 17 component releases in its 2 years of existence
  - Jakarta EE is planning for 1-2 major releases per year (components may have more)

- MicroProfile is not a Standards body
  - Jakarta EE needs to replace the JCP

- Jakarta EE brand is valuable and requires stringent compliance testing
  - MicroProfile’s compatibility statements are strictly on the honor system
Some positive signs...

Rest Client 1.x ➔ ??? ➔ JAX-RS 2.x
More positive signs...

Config JSR

???

Config Specification

Java Community Process

Jakarta EE
Reactive Platform...

- [link](https://www.lightbend.com/blog/how-the-microprofile-community-will-shape-jakarta-ee)
- Start in MicroProfile with the end goal of becoming part of Jakarta EE
Other Integrations...

Fault Tolerance
- Retry
- Timeout
- CircuitBreaker
- Bulkhead
- Fallback

Fault Tolerance
- Retry
- Timeout
- CircuitBreaker
- Bulkhead
- Fallback
Jakarta EE Community
MicroProfile Community
Both Communities
Anticipated Relationship

- Ongoing MicroProfile innovation on top of Jakarta EE, with desired adoption of foundational technologies into JakartaEE.next
Questions?

EclipseCon Europe 2018

Evaluate the Sessions

Sign in and vote at eclipsecon.org

-1    0    +1
References


- [https://microprofile.io](https://microprofile.io)
  - [https://projects.eclipse.org/projects/technology.microprofile](https://projects.eclipse.org/projects/technology.microprofile)
  - [https://microprofile.io/projects/](https://microprofile.io/projects/)
  - [https://wiki.eclipse.org/MicroProfile/Implementation](https://wiki.eclipse.org/MicroProfile/Implementation)
  - [https://openliberty.io/guides/](https://openliberty.io/guides/)

- [https://jakarta.ee](https://jakarta.ee)
  - [https://projects.eclipse.org/projects/ee4j/](https://projects.eclipse.org/projects/ee4j/)
  - [https://github.com/eclipse-ee4j/ee4j](https://github.com/eclipse-ee4j/ee4j)
  - [https://www.eclipse.org/ee4j/status.php](https://www.eclipse.org/ee4j/status.php)
  - [https://github.com/orgs/eclipse-ee4j/projects/1](https://github.com/orgs/eclipse-ee4j/projects/1)
MicroProfile Backup Material
Open Tracing

URL: https://github.com/eclipse/microprofile-opentracing
Latest Release: https://github.com/eclipse/microprofile-opentracing/releases

Tracing the flow of a request in a distributed environment has always been challenging but it is even more complex in a microservices architecture, where requests traverse across not just architectural tiers but also multiple services. The MicroProfile OpenTracing API provides a standard for instrumenting microservices for distributed tracing.
Open API

URL:  https://github.com/eclipse/microprofile-open-api
Latest Release:  https://github.com/eclipse/microprofile-open-api/releases

OpenAPI provides Java interfaces and programming models which allow Java developers to natively produce OpenAPI v3 documents from their JAX-RS applications.
Rest Client

URL: https://github.com/eclipse/microprofile-rest-client

In the Microservices world, we typically talk REST to other services. While the JAX-RS specification defines a fluent API for making calls, it is difficult to make it a true type safe client. MicroProfile Rest Client makes it easy to take an interface definition and create a JAX-RS client from it.
Applications need to be configured based on a running environment. Provides a standard way to define a hierarchy of config sources from which environmental config may be injected into a running application.
Fault Tolerance

URL: https://github.com/eclipse/microprofile-fault-tolerance

Fault tolerance is about leveraging different strategies to guide the execution and result of some logic. Defines annotations for container-brokered timeouts, retries, fallbacks, circuit-breakers and bulkheads. Simplifies the application-handling of failure in microservice architectures.
To ensure reliable operation of software it is necessary to monitor essential system parameters. Defines annotations to define metrics and well-known monitoring endpoints to gather them from.

- Required Base metrics
- Application metrics
- Vendor-specific metrics
JWT Propagation

URL: https://github.com/eclipse/microprofile-jwt-auth
Latest Release: https://github.com/eclipse/microprofile-jwt-auth/releases

The security requirements that involve microservice architectures are strongly related with RESTful Security. Defines standard JWT claims so the content of a JWT token to be used in Java EE RBAC – identifying user principle name and group names that a container can map to deployment roles.
Health Check

URL: [https://github.com/eclipse/microprofile-health](https://github.com/eclipse/microprofile-health)

Health checks are used to probe the state of a computing node from another machine (i.e. kubernetes service controller) with the primary target being cloud infrastructure environments where automated processes maintain the state of computing nodes.
Reactive Streams Operators (under construction)

URL: https://github.com/eclipse/microprofile-reactive-streams
Latest Release: https://github.com/eclipse/microprofile-reactive-streams/releases

Reactive Streams is an integration SPI - it allows two different libraries that provide asynchronous streaming to be able to stream data to and from each other.

Reactive Streams is not however designed to be used directly by application developers. The semantics defined by Reactive Streams are very strict, and are non trivial, particularly in the areas of thread safety, to implement correctly. Typically, application developers are expected to use third party libraries that provide the tools necessary to manipulate and control streams. Examples include Akka Streams, RxJava and Reactor.
Reactive Messaging (under construction)

URL:  https://github.com/eclipse/microprofile-reactive-messaging
Latest Release:  N/A

User annotates simple methods...container creates and manages Reactive Streams:
Service Mesh / ISTIO (under construction)

URL: https://github.com/eclipse/microprofile-service-mesh
Latest Release: N/A

MicroProfile defines programming model for developing cloud-native microservices. Cloud Native microservices developed with MicroProfile can take advantage of a Service Mesh by extracting many concerns away from the development of the microservice itself. It is important for MicroProfile to understand the capabilities of service mesh, so that MicroProfile can offer complimentary features for the infrastructure and avoid the conflicts.
HOW CAN THE ECLIPSE FOUNDATION BEST EVOLVE JAKARTA EE TO MEET YOUR CLOUD NEEDS? SELECT ALL THAT APPLY.