Resilient Java Microservices with Eclipse MicroProfile

Emily Jiang – MicroProfile Development Lead @emilyfhjiang

Kevin Sutter – MicroProfile and Java EE Architect @kwsutter
MicroProfile Sessions

• **Open Java EE and Eclipse MicroProfile – A New Java Landscape for Cloud Native Apps**
  • *(old title) MicroProfile, Java EE, and the Application Server – Oh My!*
  • Wednesday, Oct 25, 15:45, Burgersaal 2
• **Resilient Java Microservices with Eclipse MicroProfile**
  • *(old title) Eclipse MicroProfile: What is it? Is it a Java EE Replacement?*
  • Wednesday, Oct 25, 16:30, Silchersaal
• **MicroProfile BOF**
  • Wednesday, Oct 25, 18:30, Seminarraum 5
• **The Monitoring Aspects of Eclipse MicroProfile**
  • Thursday, Oct 26, 11:00, Seminarraum 5
• **Eclipse MicroProfile Config and OSGi Config Admin**
  • Thursday, Oct 26, 13:00, Schubartsaal
What are Microservices?
- Small concise service – small piece of SOA
- Easy to maintain
- Easy to release
- Loosely coupled a number of them to form a system

Microservice best practice
- Configurable
- Fault tolerance
- Monitoring
- Secure
- Working well in the cloud

How to build it the right way?
- Use Eclipse MicroProfile
What is Eclipse MicroProfile?

- Eclipse MicroProfile is an open-source community specification for Cloud Native Java microservices

- A community of individuals, organizations, and vendors collaborating within an open source (Eclipse) project to bring microservices to the Enterprise Java community
Why Eclipse MicroProfile?

- Microservice portable among application servers
- No vendor locking unlike Spring
- Fast innovating
- Frequent release cycle
Robust Microservices

MicroProfile adds new enterprise Java capabilities for microservices

<table>
<thead>
<tr>
<th>Config</th>
<th>Fault Tolerance</th>
<th>Health Check</th>
<th>Health Metrics</th>
<th>JWT</th>
</tr>
</thead>
<tbody>
<tr>
<td>externalize</td>
<td>build robust behavior</td>
<td>ensure services are</td>
<td>understand the interactions between services while running</td>
<td>resolve problems in complex distributed</td>
</tr>
<tr>
<td>configuration</td>
<td>to cope with unexpected</td>
<td>running</td>
<td></td>
<td>systems</td>
</tr>
<tr>
<td>to improve</td>
<td>failures</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>portability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

New in Eclipse MicroProfile Release 1.2: [https://projects.eclipse.org/projects/technology.microprofile/releases/1.2](https://projects.eclipse.org/projects/technology.microprofile/releases/1.2)
for (int i = 0; i < 5; i++) {
    try {
        callServiceC();
        break;
        break;
    } catch (IOException e) {
    }
}
@Retry(retryOn=IOException.class,
   delay = 500,
   maxRetries=5
public void callServiceC() {
   // call the service
}
Dealing with slow services

```java
@Timeout(2000)
public void callServiceC() {
    // call the service
}
```
Don’t overload serviceC

@Bulkhead
def callServiceC()
    # call the service

Circuit Breaker Closed

service A

service C
Circuit Breaker Open

service A ➔ service C
@CircuitBreaker(failOn=IOException.class, delay = 500)
public void callServiceC() {
    // call the service
}
@Fallback(MyFallback.class)
public void callServiceC()
{
    // call the service
}

private class MyFallback
    implements FallbackHandler {
    public void handle(ExecutionContext c)
    {
        // perform fallback action
    }
}
MicroProfile Fault Tolerance with Istio

@Retry
@Timeout
@CircuitBreaker
@Bulkhead
@Fallback

Istio

Retry
Timeout
CircuitBreaker
Bulkhead
Resources

- http://microprofile.io/
- https://openliberty.io/
- https://github.com/NottyCode/fault-tolerance-examples
Thank you