

Microprofile

Specs from 1.0 to 3.0

✉ rafael.benevides@oracle.com

🐦 [@rafabene](https://twitter.com/rafabene)

Link



<http://bit.ly/slides-microprofile>



Rafael Benevides

Cloud-Native Development Advocate

 rafael.benevides@oracle.com

 @rafabene

Java Certifications:

SCJA / SCJP / SCWCD / SCBCD / SCEA

JBoss Certifications:

JBCD / JBCAA

Red Hat Certifications:

OpenShift / Containers / Ansible

Other Certifications:

SAP Netweaver / ITIL / IBM Software Quality



Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Break New Ground

<https://developer.oracle.com>

Break New Ground

<https://cloudnative.oracle.com>

Visite o nosso estande!

- Faça um trial e ganhe um brinde
- Faça um hands on e ganhe outro brinde

What is Eclipse Microprofile?

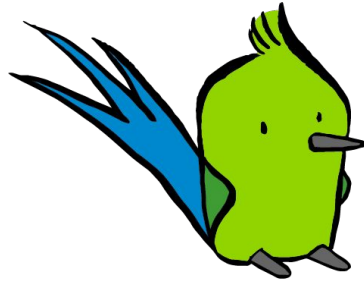


- Eclipse MicroProfile is an **open-source** community **specification** for Enterprise Java microservices
- A community of **individuals**, **organizations**, and **vendors** collaborating within an open source (Eclipse) project to bring microservices to the Enterprise Java community

Community - individuals, organizations, vendors



Current MicroProfile implementations



Apache TomEE



THORNTAIL

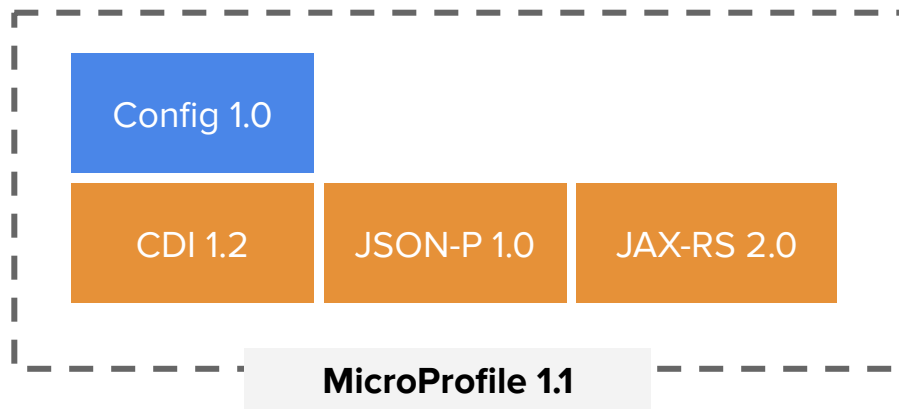




SMALLRYE

MicroProfile 1.0 (Sep, 2016)



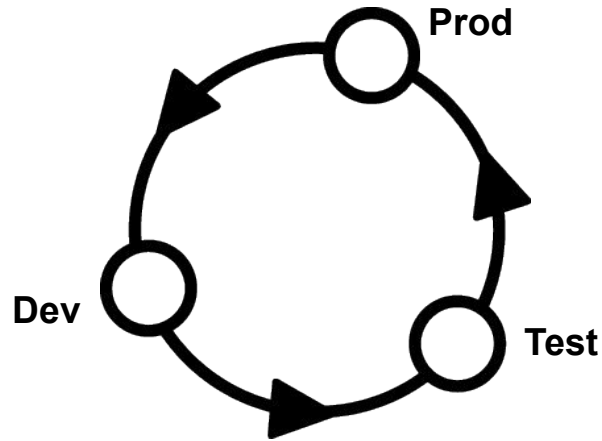
Eclipse MicroProfile 1.1 (Aug, 2017)



-  = New
-  = No change from last release

Configuration

Applications need to be configured based on a running environment. It must be possible to modify configuration data from outside an application so that the application itself does not need to be repackaged



Configuration

By default there are 3 default config sources:

- System.getProperties() (ordinal=400)
- System.getenv() (ordinal=300)
- all META-INF/microprofile-config.properties files. (default ordinal=100)

@Inject

@ConfigProperty(name = "**preference.api.url**",
 defaultValue = "**http://localhost:8180/**")

private String **preferenceURL**;

```
@Dependent
public class ElasticsearchClientProducer {

    @Produces
    public TransportClient getClient() throws ClientNotAvailableException{
        try {
            Settings settings = Settings.builder()
                .put("cluster.name", clusterName).build();

            InetAddress inetAddress = InetAddress.getByName(hostName);

            TransportAddress transportAddress = new TransportAddress(inetAddress, hostPort);

            return new PreBuiltTransportClient(settings)
                .addTransportAddress(transportAddress);
        } catch (UnknownHostException ex) {
            log.log(Level.SEVERE, null, ex);
            throw new ClientNotAvailableException(ex);
        }
    }

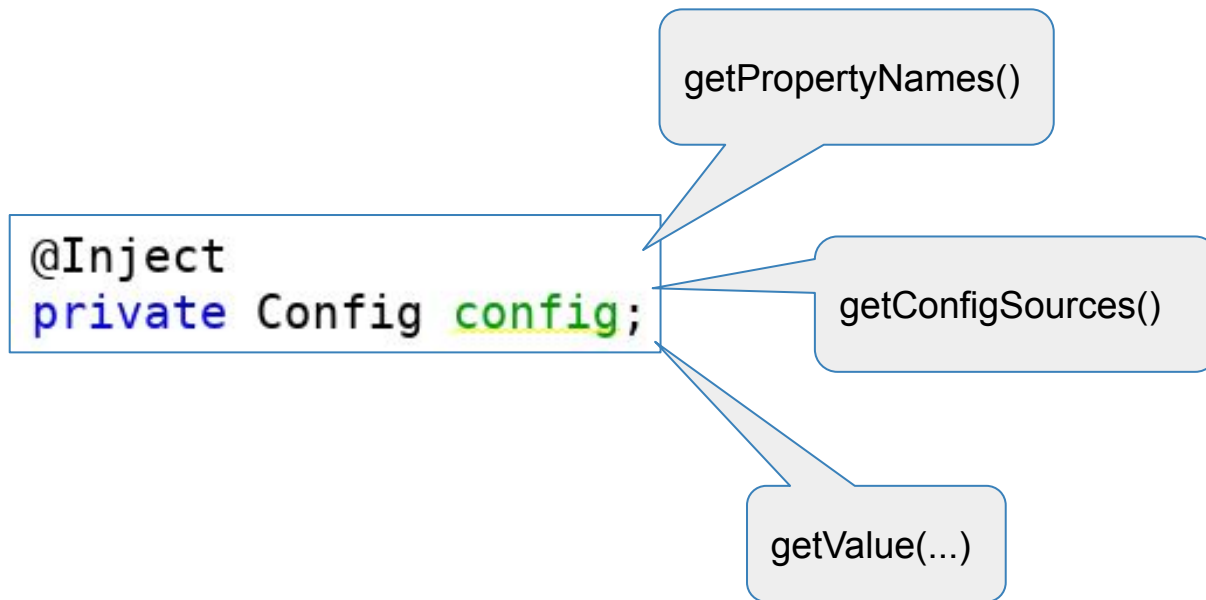
    @Inject @ConfigProperty(name = "elasticsearch.cluster.name", defaultValue = "the-red-cluster")
    private String clusterName;

    @Inject @ConfigProperty(name = "elasticsearch.host.name", defaultValue = "localhost")
    private String hostName;

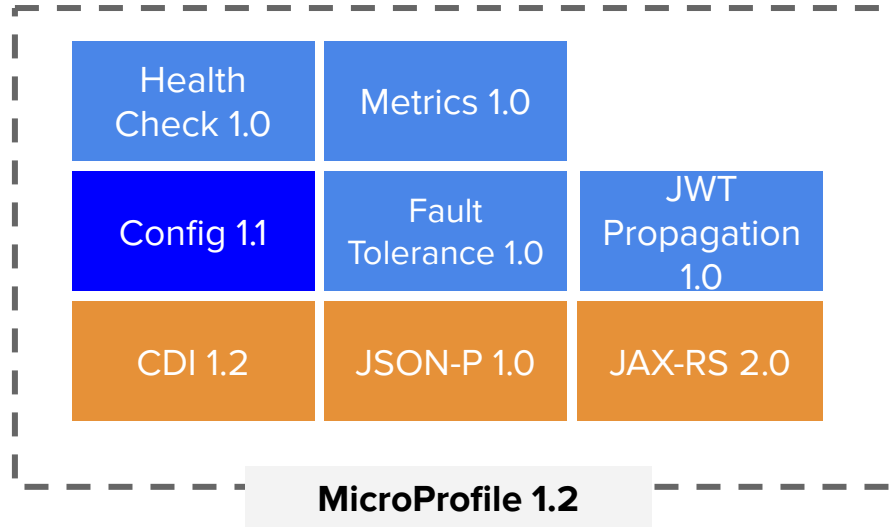
    @Inject @ConfigProperty(name = "elasticsearch.host.port", defaultValue = "9300")
    private int hostPort;
}
```

CDI

CDI



Eclipse MicroProfile 1.2 (Sep, 2017)



- = New
- = Updated
- = No change from last release

Health Check 1.0

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



Health Check 1.0 - Goals

- MUST be compatibility with well known **cloud** platforms (i.e. <http://kubernetes.io/docs/user-guide/liveness/>)
- MUST be appropriate for **machine-to-machine** communication
- SHOULD give enough information for a **human** administrator

Health Check 1.0 (Outcomes and Checks)

@Health

@ApplicationScoped

```
public class UserAvailabilityHealthCheck implements HealthCheck {
```

@Override

```
public HealthCheckResponse call() {
```

```
    HealthCheckResponseBuilder response = HealthCheckResponse.named("usersAvailable");
```

```
    try {
```

```
        // Try users microservices
```

```
        return response.up().build();
```

```
    } catch (Exception ex) {
```

```
        return response.down().build();
```

```
    }
```

```
}
```

```
}
```

<http://localhost:8080/health>

```
{
  "outcome": "UP",
  "checks": [
    {
      "name": "usersAvailable",
      "state": "UP",
      "data": {}
    }
  ]
}
```

Health Check 1.0 (Outcomes and Checks)

```
@Health
@ApplicationScoped
public class SubscribersListFullCheck implements HealthCheck {
```

```
@Override
```

```
public HealthCheckResponse call() {
    HealthCheckResponseBuilder response =
        HealthCheckResponse.named("subscribersListFull");
    try {
        // Try users microservices
        response.withData("Number of subscribers",
            numberOfSubscribers)
        return response.up().build();
    } catch (Exception ex) {
        return response.down().build();
    }
}
```

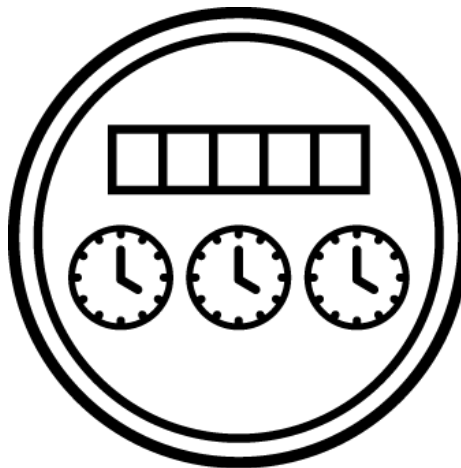
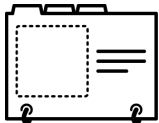
<http://localhost:8080/health>

```
{
  "outcome": "UP",
  "checks": [
    {
      "name": "subscribersListFull",
      "state": "UP",
      "data": {
        "Number of subscribers": "3",
        "Maximum subscriber": "10"
      }
    },
    {
      "name": "usersAvailable",
      "state": "UP",
      "data": {}
    }
  ]
}
```

Metrics 1.0

To ensure reliable operation of software it is necessary to **monitor** essential system parameters. Metrics adds well-known **monitoring endpoints** and metrics for each process

Metric Registry



Required Base metrics
Application metrics
Vendor-specific metrics

Metrics 1.0 - Goals

- Alternative to JMX but for a polyglot environment
- Standard
 - API path,
 - data types involved,
 - always available metrics
 - return codes used

Metrics 1.0 - Difference to health checks

Health Check	Metric
YES/NO Response	long term trend data for capacity planning
"Is my application still running ok?"	pro-active discovery of issues (e.g. disk usage growing without bounds)

Metrics can also help those scheduling systems decide when to scale the application to run on more or fewer machines.

Metrics 1.0 - Goals

<h2>General</h2> <ul style="list-style-type: none">● UsedHeapMemory● CommittedHeapMemory● MaxHeapMemory● GCCount & GCTime● JVM Uptime	<h2>ClassLoading</h2> <ul style="list-style-type: none">● LoadedClassCount● TotalLoadedClassLoaded● UnloadedClassCount
<h2>Thread</h2> <ul style="list-style-type: none">● ThreadCount● DaemonThreadCount● PeakThreadCount● ActiveThreads● PoolSize	<h2>Operating System</h2> <ul style="list-style-type: none">● AvailableProcessors● SystemLoadAverage● ProcessCpuLoad

Metrics 1.0 - Goals

The following three sets of sub-resource (scopes) are exposed.

- **base**: metrics that all MicroProfile vendors have to provide
- **vendor**: vendor specific metrics (optional)
- **application**: application-specific metrics (optional)

<http://localhost:8080/metrics>

<http://localhost:8080/metrics/base>

<http://localhost:8080/metrics/vendor>

<http://localhost:8082/metrics/application>

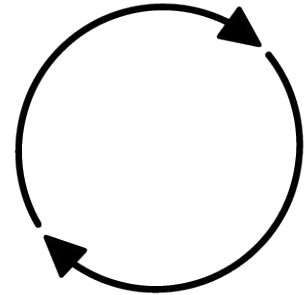
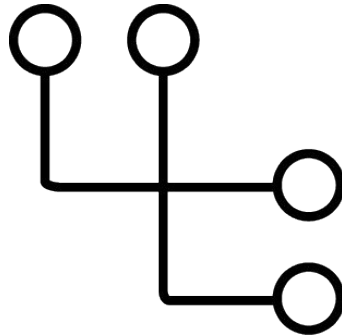
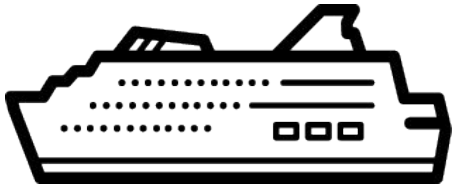
Metrics 1.0 - API

The easiest way is to annotate field, method or class with an annotation.

Annotation	Description	Default Unit
@Counted	Denotes a counter, which counts the invocations of the annotated object.	MetricUnits.NONE
@Gauge	Denotes a gauge, which samples the value of the annotated object.	none Must be supplied by the user
@Metered	Denotes a meter, which tracks the frequency of invocations of the annotated object.	MetricUnits.PER_SECOND
@Metric	An annotation that contains the metadata information when requesting a metric to be injected or produced	MetricUnits.NONE
@Timed	Denotes a timer, which tracks duration of the annotated object.	MetricUnits.NANOSECONDS

Fault Tolerance 1.0

Fault tolerance is about leveraging different strategies to guide the execution and result of some logic. **Retry policies**, **bulkheads**, and **circuit breakers** are popular concepts in this area. They dictate whether and when executions should take place, and fallbacks offer an alternative result when an execution does not complete successfully



Fault Tolerance 1.0

- @Timeout** - Define a duration for timeout
- @Retry** - Define a criteria on when to retry
- @Fallback** - Provide an alternative solution for a failed execution
- @Bulkhead** - Isolate failures in part of the system while the rest of the system can still function
- @CircuitBreaker** - Offer a way of fail fast by automatically failing execution to prevent the system overloading and indefinite wait or timeout by the clients

Fault Tolerance 1.0

```
@Fallback(fallbackMethod = "defaultAuthor")
@Retry(maxRetries = 5)
@CircuitBreaker(requestVolumeThreshold = 10,
    failureRatio = 0.6,
    delay = 2000L,
    successThreshold = 2)
@Timeout(800)
public JsonObject findAuthorByEmail(String email) {
    // Call remote service
}

public JsonObject defaultAuthor(String email) {
    return Json.createObjectBuilder()
        .add("firstName", "")
        .add("lastName", "Unkown")
        .add("bio", "Try again later")
        .add("email", email)
        .build();
}
```

JWT Propagation

The security requirements that involve microservice architectures are strongly related with RESTful Security. In a RESTful architecture style, services are usually **stateless** and any **security state** associated with a client is sent to the target service on **every request** in order to allow services to **re-create** a security context for the caller and perform both **authentication** and **authorization** checks



What is JWT?

JSON Web Token (JWT) is an open standard ([RFC 7519](#)) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object

A client sends a HTTP request to Service A including the JWT as a bearer token:

```
GET /resource/1 HTTP/1.1  
Host: example.com  
Authorization: Bearer mF_9.B5f-4.1JqM
```

What is JWT?

An example minimal MP-JWT in JSON would be:

```
{
  "typ": "JWT",
  "alg": "RS256",
  "kid": "abc-1234567890"
}
{
  "iss": "https://server.example.com",
  "aud": "s6BhdRkqt3",
  "jti": "a-123",
  "exp": 1311281970,
  "iat": 1311280970,
  "sub": "24400320",
  "upn": "jdoe@server.example.com",
  "groups": ["red-group", "green-group", "admin-group", "admin"],
}
```


JWT Usage

```
@RequestScoped
@Path("/token")
@Produces(MediaType.TEXT_PLAIN)
@Tag(name = "Token service", description = "JWT Issuer")
@DeclareRoles({"user", "admin"})
@DenyAll
public class TokenService {

    @Context
    private SecurityContext securityContext;

    @Inject
    private TokenIssuer tokenIssuer;

    @Inject
    private TokenSigner tokenSigner;

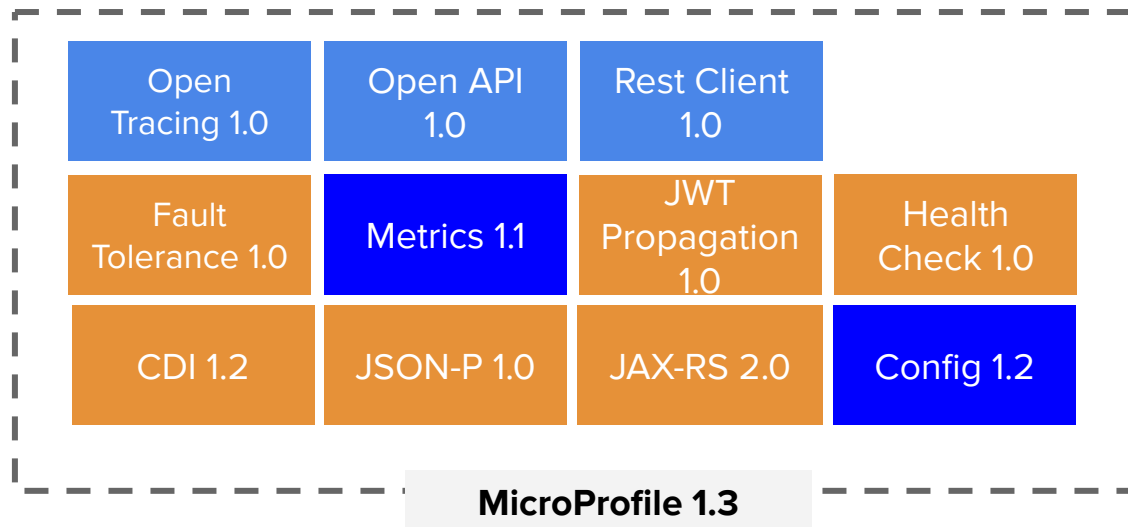
    @GET
    @Path("/issue")
    @Operation(description = "Issue a JWT Token for the logged in user")
    @APIResponse(responseCode = "200", description = "Get the signed token")
    @RolesAllowed({"admin", "user"})
    public Response issueToken(){




        Principal userPrincipal = securityContext.getUserPrincipal();
        String username = userPrincipal.getName();
        List<String> roles = getUserRoles();
        String token = tokenIssuer.issue(username, roles);
        String signToken = tokenSigner.signToken(token);

        return Response.ok(signToken).build();
    }
}
```

JAX-RS &
OpenAPI

Eclipse MicroProfile 1.3 (Jan, 2018)

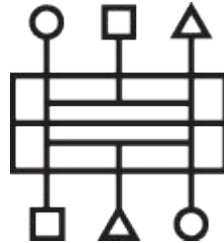


-  = New
-  = Updated
-  = No change from last release

OpenAPI 1.0

Management of microservices in an MSA can become unwieldy as the number of microservices increases. Microservices can be managed via their APIs.

Management, security, load balancing, and throttling are policies that can be applied to APIs fronting microservices. OpenAPI provides Java interfaces and programming models which allow Java developers to natively produce **OpenAPI v3 documents** from their **JAX-RS** applications.



OpenAPI 1.0

```
@ApplicationPath("/api")
@OpenAPIDefinition(info = @Info(
    title = "Membership service",
    version = "1.0.0",
    contact = @Contact(
        name = "Phillip Kruger",
        email = "phillip.kruger@phillip-kruger.com",
        url = "http://www.phillip-kruger.com")),
    servers = {
        @Server(url = "/membership",description = "localhost"),
        @Server(url = "http://yellow:8080/membership",description = "Yellow Pi")
    }
)

public class ApplicationConfig extends Application {
}
```

RequestScoped

@Path("/")

@Consumes(MediaType.APPLICATION_JSON) @Produces(MediaType.APPLICATION_JSON)

@Tag(name = "Membership service", description = "Managing the membership")

public class MembershipService {

@PersistenceContext(name="MembershipDS")

private EntityManager em;

@GET

@Timed(name = "Memberships requests time", absolute = true, unit = MetricUnits.MICROSECONDS)

@Operation(description = "Get all the current memberships")

@APIResponse(responseCode = "200", description = "Successful, returning the memberships")

public List<Membership> getAllMemberships() {

TypedQuery<Membership> query = em.createNamedQuery(Membership.QUERY_FIND_ALL, Membership.class);

return query.getResultList();

}

@GET

@Path("/{id}")

@Counted(name = "Membership requests", absolute = true, monotonic = true)

@Operation(description = "Get a certain Membership by id")

@APIResponse(responseCode = "200", description = "Successful, returning the memberships",

content = @Content(mediaType = MediaType.APPLICATION_JSON, schema = @Schema(implementation = Membership.class)))

public Membership getMembership(@NotNull @PathParam(value = "id") int id) {

return em.find(Membership.class, id);

}

@POST

@Counted(name = "Membership created", absolute = true, monotonic = true)

@Operation(description = "Create a new Membership")

@APIResponse(responseCode = "200", description = "Successful, returning the created membership")

public Membership createMembership(@NotNull @RequestBody(description = "The membership",

content = @Content(mediaType = MediaType.APPLICATION_JSON, schema = @Schema(implementation = Membership.class)))

Membership membership){

membership = em.merge(membership);

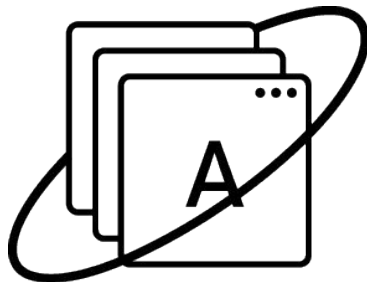
log.log(Level.INFO, "Created membership [{0}]", membership);

return membership;

}

REST Client 1.0

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**. Several JAX-RS implementations support the ability to take an interface definition and create a JAX-RS client from it (JBoss RestEasy, Apache CXF) as well as being supported by a number of service providers (Wildfly Swarm, OpenFeign). MicroProfile Rest Client API provides a **type-safe approach** to invoke RESTful services over HTTP in a **consistent** and **easy-to-reuse** fashion.



REST Client 1.0

- A type-safe approach to invoke RESTful services over HTTP
- More natural coding style
- Handles HTTP connectivity and serialization

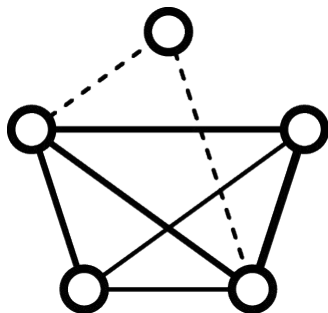
```
String apiUrl = "http://localhost:9080/movieReviewService";
MovieReviewService reviewSvc =
    RestClientBuilder.newBuilder()
        .baseUrl(apiUrl)
        .build(MovieReviewService.class);

Review review = new Review(3 /*stars*/, "Good Movie.");

reviewSvc.submitReview( movieId, review );
```

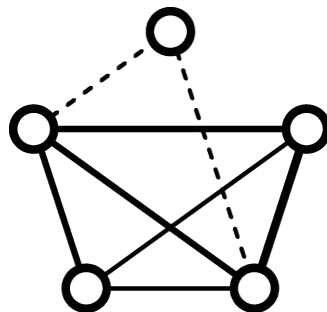
OpenTracing 1.0

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**.




```
@GET @Path("user/{userId}")
@Operation(description = "Getting all the events for a certain user")
@APIResponses({
    @APIResponse(responseCode = "200", description = "Successfull, returning events",
        content = @Content(schema = @Schema(implementation = UserEvent.class))),
    @APIResponse(responseCode = "401", description = "User not authorized"),
    @APIResponse(responseCode = "412", description = "Membership not found, invalid userId",
        headers = @Header(name = REASON))
})
@SecurityRequirement(name = "Authorization")
@RolesAllowed({"admin", "user"})
@Traced(operationName = "GetUserEvents", value = true)
public Response getUserEvents(
    @Parameter(name = "userId", description = "The User Id of the member", required = true, allowEmptyValue = false, example = "1")
    @PathParam("userId") int userId,
    @Parameter(name = "size", description = SIZE_DESC, required = false, allowEmptyValue = true, example = "10") @DefaultValue("-1")
    @QueryParam("size") int size){
    try {
        validateMembership(userId);
    } catch (NotFoundException nfe){
        return Response.status(Status.PRECONDITION_FAILED).header(REASON, "Membership [" + userId + "] does not exist").build();
    }
    return eventSearcher.search(UserEventConverter.USER_ID,userId,size);
}
```

OpenTracing 1.0



NOTE

Currently, MicroProfile OpenTracing does not integrate with MicroProfile RestClient, so you need to use a pure JAX-RS Client. This limitation will be removed in the future, see [MicroProfile OpenTracing issue #82](#).

https://docs.thorntail.io/2.3.0.Final/#tracing-a-complex-service_thorntail



JAEGER



Find Traces

Service

ordermgr

all

Tags

http.status_code:400|http.status_code:200

Lookback

Last Hour

Min Duration

e.g. 1.2s, 100ms, 50

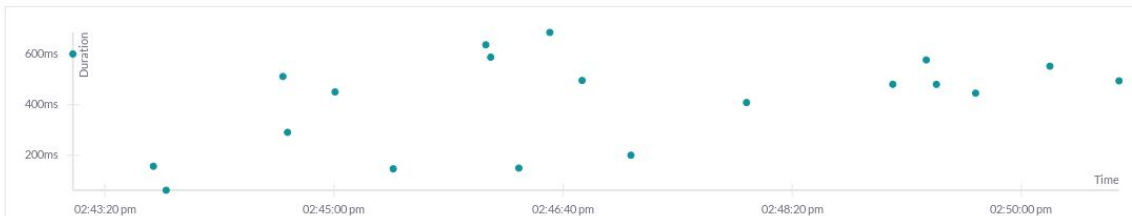
Max Duration

e.g. 1.1s

Limit Results

20

Find Traces

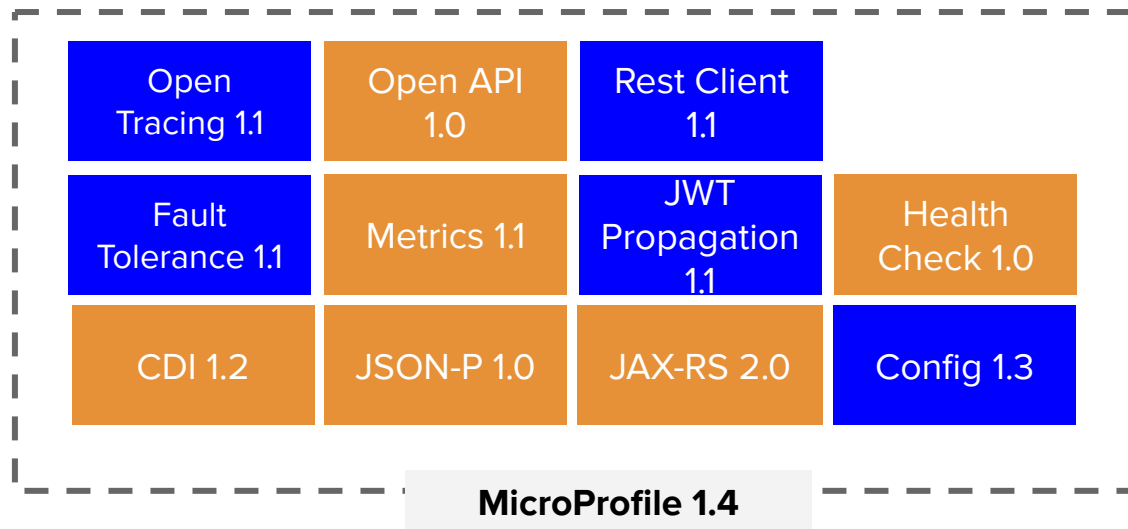





20 Traces

Sort Most Recent

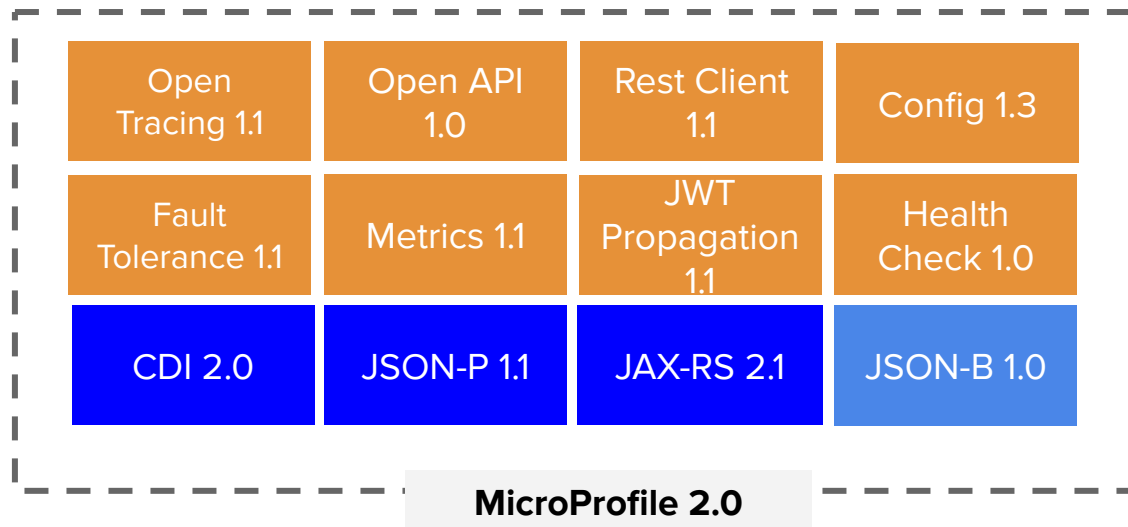
ordermgr: buy	494.67ms
5 spans 2 errors	accountmgr (2) ordermgr (3) 02:50:42 pm (a few seconds ago)
ordermgr: sell	552.97ms
3 spans 0	accountmgr (1) ordermgr (2) 02:50:12 pm (a minute ago)
ordermgr: fail	445.89ms
5 spans 1 error	accountmgr (2) ordermgr (3) 02:49:40 pm (2 minutes ago)
ordermgr: sell	480.8ms
3 spans 0	accountmgr (1) ordermgr (2) 02:49:22 pm (2 minutes ago)
ordermgr: buy	577.75ms
3 spans 0	accountmgr (1) ordermgr (2) 02:49:18 pm (2 minutes ago)
ordermgr: sell	481.07ms


Eclipse MicroProfile 1.4 (Jun, 2018)



-  = New
-  = Updated
-  = No change from last release

Eclipse MicroProfile 2.0 (Jun, 2018)

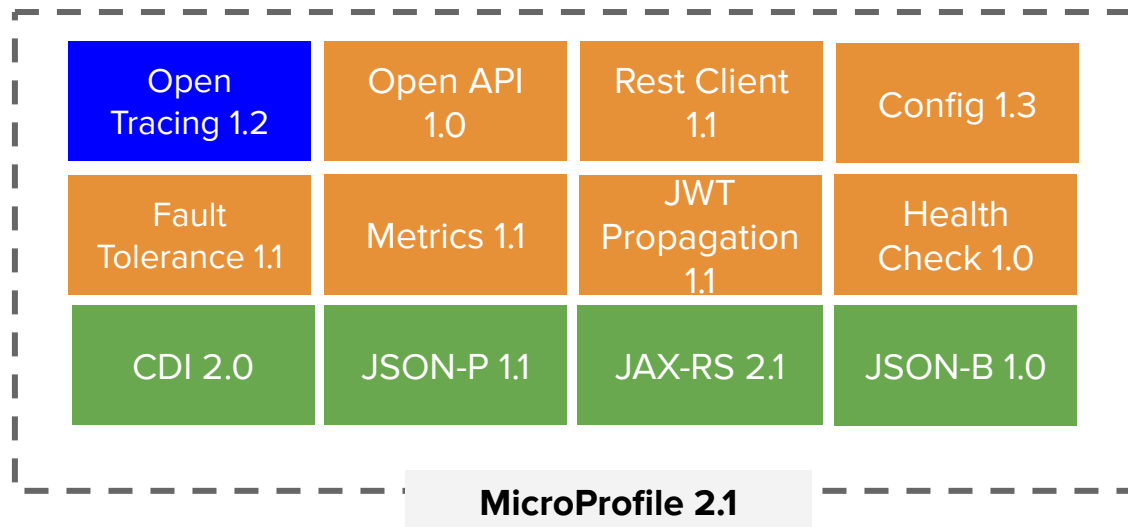






-  = New
-  = Updated
-  = No change from last release (MicroProfile 1.4)

Eclipse MicroProfile 2.0 - Goals

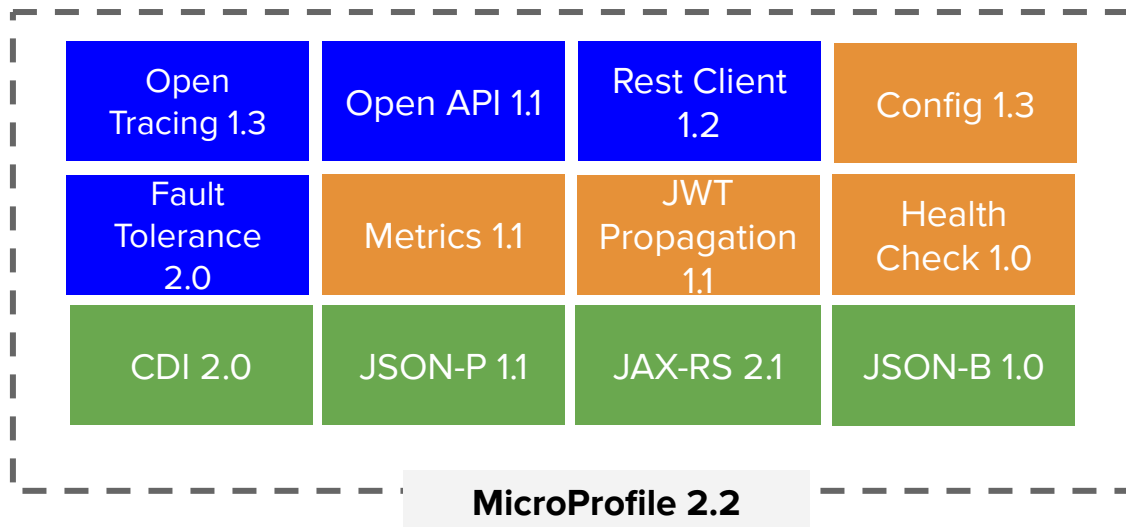
- Alignment of Java EE related APIs to Java EE 8 release:
- Updated CDI, JSON-P, JAX-RS, and added JSON-B





Eclipse MicroProfile 2.1 (Oct, 2018)



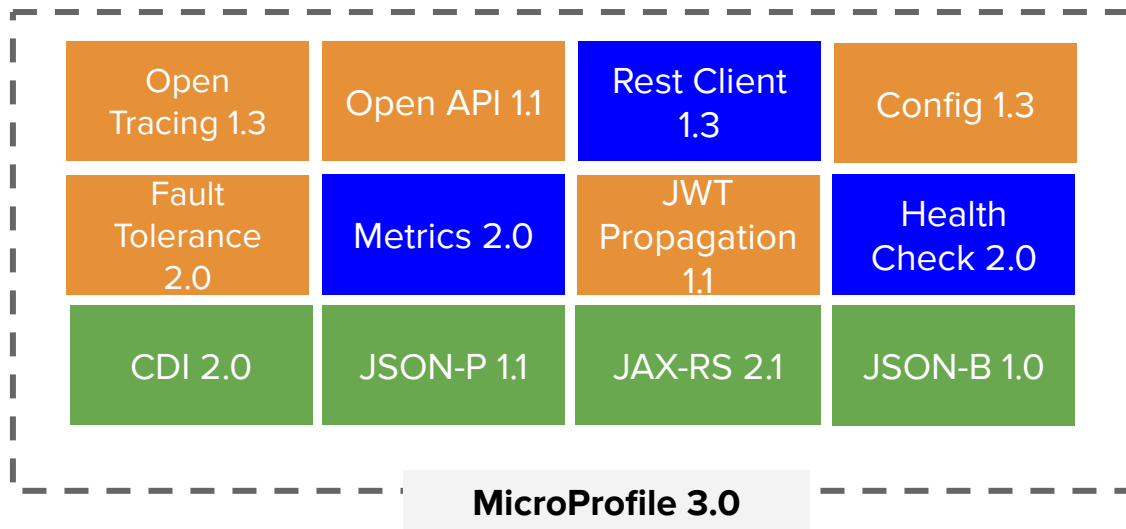
-  = New
-  = Updated
-  = No change from last release (MicroProfile 2.0)
-  = Java EE / Jakarta EE





Eclipse MicroProfile 2.2 (Feb, 2019)



-  = New
-  = Updated
-  = No change from last release (MicroProfile 2.1)
-  = Java EE / Jakarta EE

Eclipse MicroProfile 3.0 Released!



-  = New
-  = Updated
-  = No change from last release (MicroProfile 2.2)
-  = Java EE / Jakarta EE

Eclipse MicroProfile 3.0 Released!

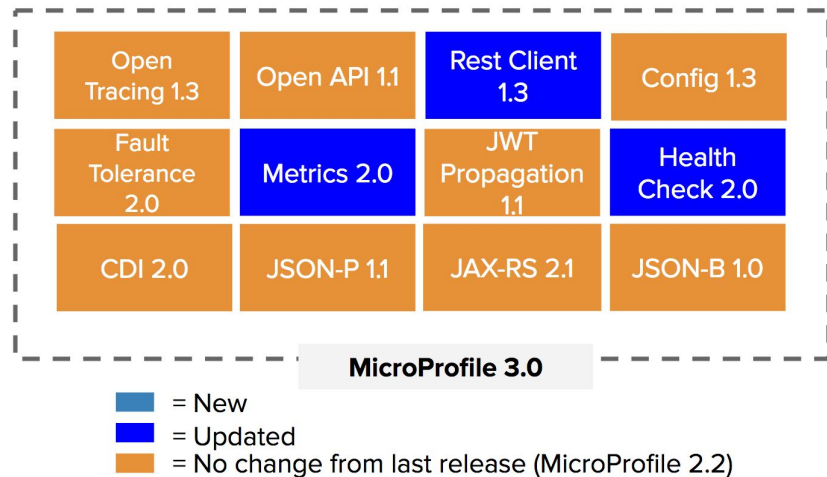
On June 11, 2019, MicroProfile 3.0 was released.

Offered in the release:

- Java EE 8 continued alignment
- A richer feature set for Rest Client, Metrics, and Health Check
- Metrics and Health Check have introduced breaking API changes
- CDI-based and programmatic interfaces
- Test Compatibility Kit (TCK), Javadoc, PDF doc for download

Other news:

- [Boost](#) project has been added to the MicroProfile sandbox
- start.microprofile.io planning to release command-line interface



Topics under discussion

- [Long Running Actions](#)
- [Reactive Messaging](#)
- [GraphQL](#)
- [Concurrency](#)
- Reactive Relational Database Access
- Event Data
- Service meshes



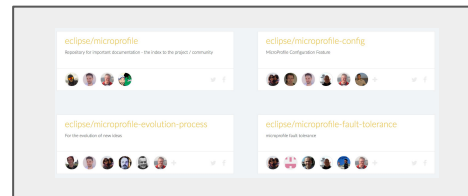
Demo

github.com/rafabene/microprofile-demo

Get Involved!



[Google Groups](#)



[MicroProfile Projects](#)



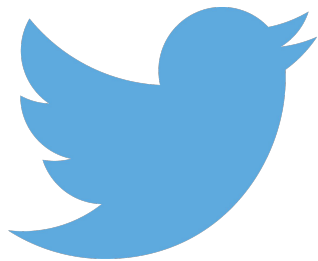
[Bi-Weekly & Quarterly
General community
Meetings](#)



[YouTube Channel](#)



[Video Hangouts](#)



@RAFABENE