

# MicroProfile

microservices made easy



# # whoami

- Martin Štefanko
- Software engineer, Red Hat
- MicroProfile contributor
-  @xstefank

# Enterprise Java in past 20 years

- Java EE (currently Jakarta EE)
  - Java EE 5 - May 11, 2006
  - Java EE 6 - December 10, 2009
  - Java EE 7 - June 12, 2013
  - Java EE 8 - August 31, 2017



# 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



# Eclipse MicroProfile 2.2 (Feb 2019)



■ = New

■ = Updated

■ = No change from last release (MicroProfile 2.1)

# +Under discussion

- Long Running Actions (LRA)
- Reactive Streams Operators (standalone release)
- Reactive messaging
- Service mesh
- Context propagation
- GraphQL
- ...

# Community - individuals, organizations, vendors



ORACLE®

[HTTPS://MICROPROFILE.IO/](https://microprofile.io/) | [HTTPS://PROJECTS.ECLIPSE.ORG/PROJECTS/TECHNOLOGY.MICROPROFILE](https://projects.eclipse.org/projects/technology.microprofile)

3

#openhouseredhat #microprofile

@xstefank @RedHat

# Current MicroProfile implementations



4

[HTTPS://MICROPROFILE.IO/](https://microprofile.io/) | [HTTPS://PROJECTS.ECLIPSE.ORG/PROJECTS/TECHNOLOGY.MICROPROFILE](https://projects.eclipse.org/projects/technology.microprofile)

#openhousereredhat #microprofile

@xstefank @RedHat

# Differences from Java EE

- open source and open community
- code first approach
- 3 releases per year (Feb, Jun, Oct)
  - MP 1.0 - Sep 2016
  - MP 1.1 - Aug 2017
  - MP 1.2 - Sep 2017
  - MP 1.3 - Jan 2018
  - MP 1.4 / MP 2.0 - Jun 2018
  - MP 2.1 - Oct 2018

# MicroProfile 3.0

📅 Due by June 05, 2019 0% complete

The MicroProfile 3.0 release is targeted for June 2019 (with additional releases in Feb and October of 2019). We will use this Milestone to help track the content for this platform release. The expected Release Announce date will be Tuesday, June 11. Note: Based on the "Breaking API Changes" discussion, this release will be renamed MicroProfile 3.0 to all...

Show more ▾

① 7 Open ✓ 0 Closed
≡ ⓘ <b>Include GraphQL 1.0 (stand alone)</b> #102 opened 7 days ago by kwsutter 
ⓘ <b>Include Rest Client 1.3</b> #101 opened 7 days ago by kwsutter 
ⓘ <b>Include Health Check 2.0</b> #69 opened on Nov 27, 2018 by kwsutter  2
ⓘ <b>Include Metrics 2.0</b> #71 opened on Nov 27, 2018 by kwsutter  4
ⓘ <b>Include Context Propagation 1.0 (stand alone, used to be known as Concurrency)</b> #73 opened on Nov 27, 2018 by kwsutter 
ⓘ <b>Include Reactive Messaging 1.0 (stand alone)</b> #76 opened on Nov 27, 2018 by kwsutter 
ⓘ <b>Include LRA 1.0 (stand alone)</b>  6

# Eclipse MicroProfile

Optimizing Enterprise Java  
for a Microservices Architecture

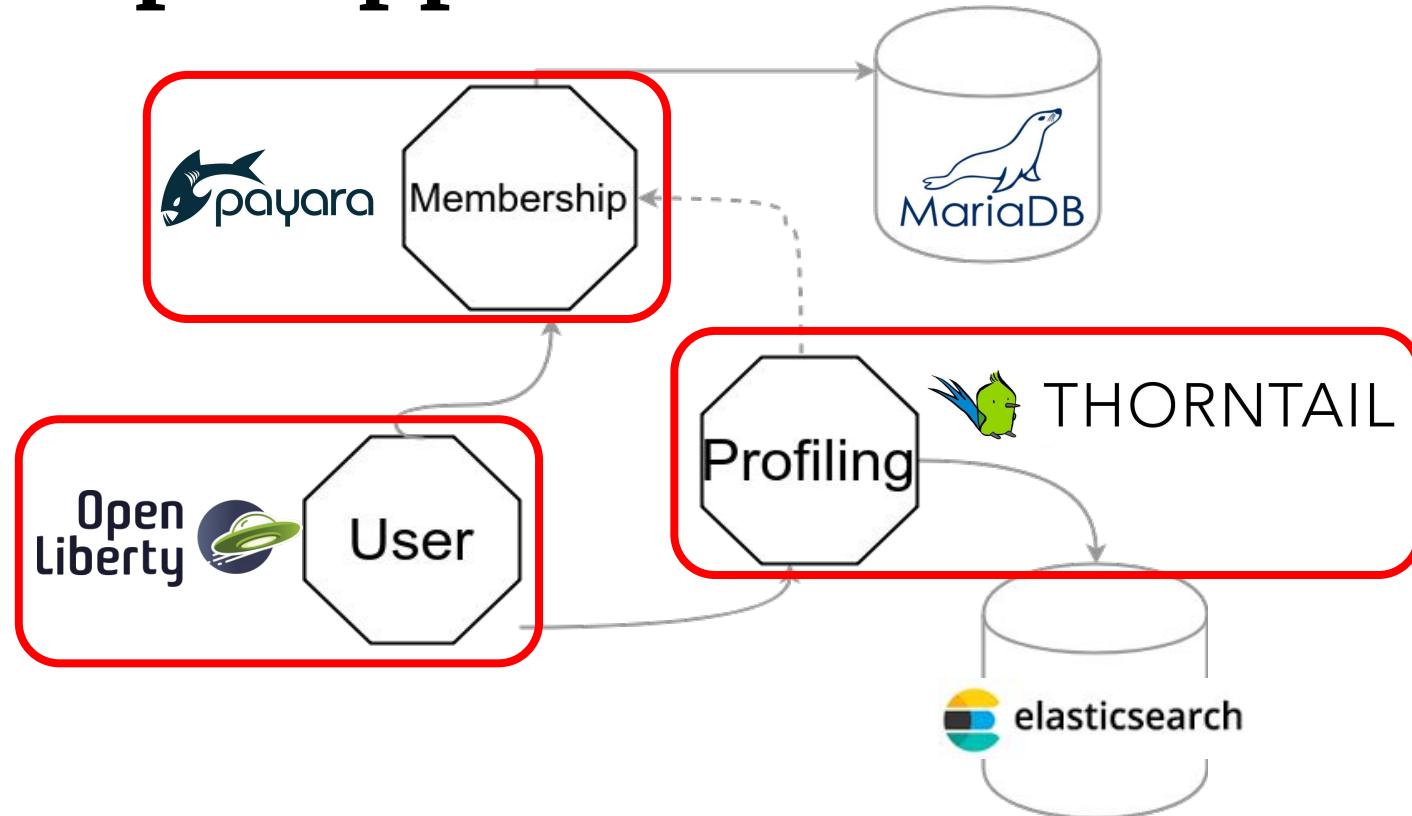


[microprofile.io](http://microprofile.io)

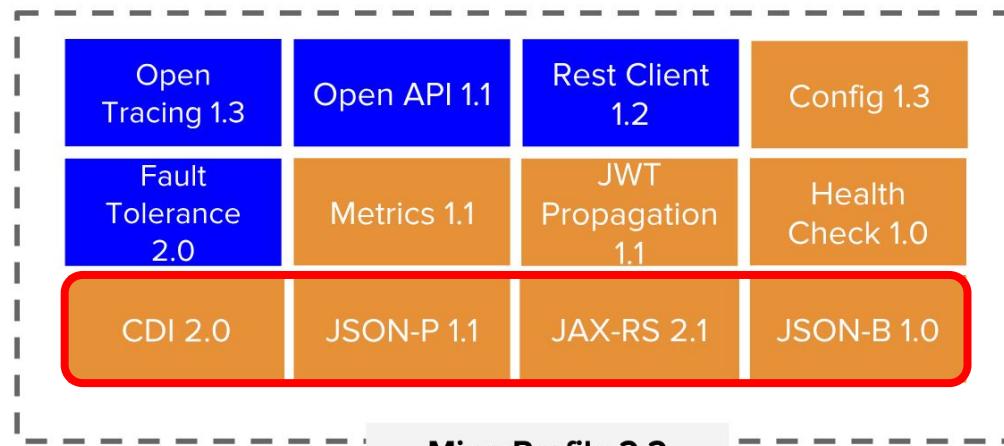
#openhouseredhat #microprofile

@xstefank @RedHat

# Example application



# Eclipse MicroProfile 2.2 (Feb 2019)



- = New
- = Updated
- = No change from last release (MicroProfile 2.1)

6

[HTTPS://MICROPROFILE.IO/](https://microprofile.io/) | [HTTPS://PROJECTS.ECLIPSE.ORG/PROJECTS/TECHNOLOGY.MICROPROFILE](https://projects.eclipse.org/projects/technology.microprofile)

# JAX-RS

```
@ApplicationPath("/api")
public class ApplicationConfig extends Application {
}
```

host:port/api/path1/path2/...

```
@Path("/")
@Consumes(MediaType.APPLICATION_JSON) @Produces(MediaType.APPLICATION_JSON)
public class ProfileService {

    @POST
    public Response logEvent(...) {
        return Response.accepted(event).build();
    }

    @GET
    @Path("user/{userId}")
    public Response getUserEvents(@PathParam("userId") int userId) {

        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);
    }

    ...
}
```

# JSON-P

```
JsonObject json = Json.createObjectBuilder()
    .add("name", "Iron Man")
    .add("realName", "Tony Stark")
    .add("alive", "true")
    .build();
```

# JSON-P

- Creating, reading and writing JSON
- Parsing JSON
- Stream support
- JSON pointers
- JSON patching

# JSON-B

```
public class Membership implements Serializable {  
    private int membershipId;  
    private Person owner;  
    private Type type;  
}  
  
public class Person implements Serializable {  
    private int id;  
    private List<String> names;  
    private String surname;  
    private String email;  
}
```



```
{  
    "membershipId": 4,  
    "owner": {  
        "email": "minki@gmail.com",  
        "id": 4,  
        "names": [  
            "Minki"  
        ],  
        "surname": "van der Westhuizen"  
    },  
    "type": "FREE"  
}
```

# JSON-B

```
Jsonb jsonb = JsonbBuilder.create();
String json = jsonb.toJson(Avenger.name("Iron Man")
    .realName("Tony Stark")
    .alive(true)
    .build());
Avenger ironMan = jsonb.fromJson(json, Avenger.class);
```

```
@Path("/")
@Consumes(MediaType.APPLICATION_JSON) @Produces(MediaType.APPLICATION_JSON)
public class MembershipService {

    @GET
    public List<Membership> getAllMemberships() {
        ...
    }

    @GET
    @Path("{id}")
    public Membership getMembership(@NotNull @PathParam(value = "id") int id) {
        ...
    }
}
```

# CDI

```
@RequestScoped
public class EventLogger {
    @Inject
    private Client client;

    @Inject @Successful
    private Event<UserEvent> successfulBroadcaster;

    public Future<Void> logEvent(String token, @NotNull UserEvent event){
        IndexResponse response = client.prepareIndex(IndexDetails.PROFILING_INDEX,
IndexDetails.TYPE).setSource(json.toString(), XContentType.JSON).get();

        if(response.status().getStatus() == 201){
            successfulBroadcaster.fire(event);
        }

        return CompletableFuture.completedFuture(null);
    }
}
```

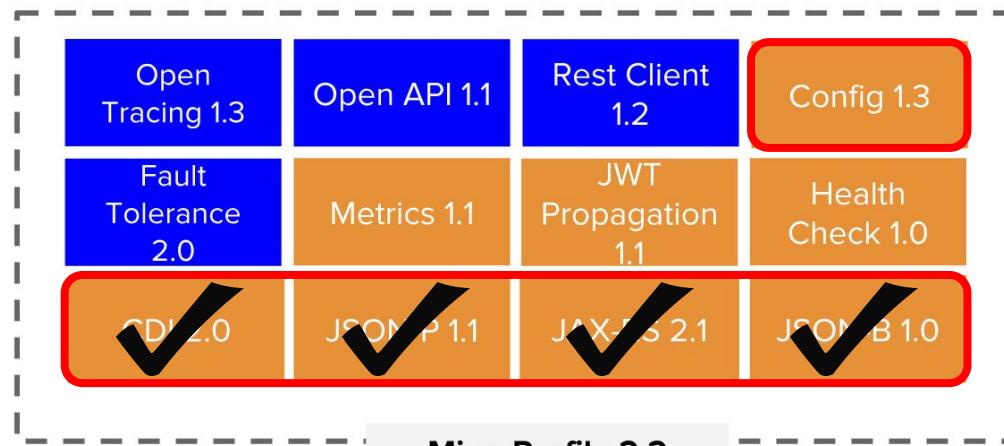
# CDI

- `@RequestScoped`
- `@ApplicationScoped`
- `@SessionScoped`
- `@Dependent`
- `@ConversationScoped`

```
@ApplicationScoped
public class BootstrapService {

    @Produces
    public Client getClient() throws ClientNotAvailableException{
        return node.client();
    }
}
```

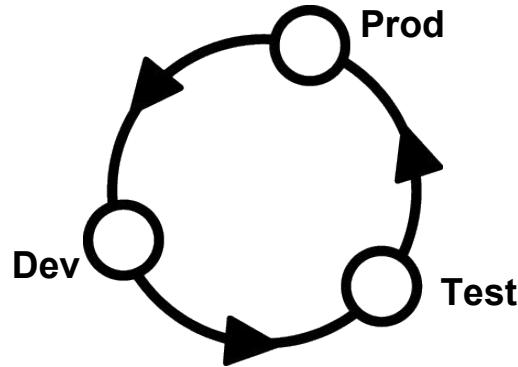
# Eclipse MicroProfile 2.2 (Feb 2019)



[HTTPS://MICROPROFILE.IO/](https://microprofile.io/) | [HTTPS://PROJECTS.ECLIPSE.ORG/PROJECTS/TECHNOLOGY.MICROPROFILE](https://projects.eclipse.org/projects/technology.microprofile)

# 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



```
@ApplicationScoped
public class BootstrapService {

    @Inject @ConfigProperty(name = "java.io.tmpdir", defaultValue = "/tmp")
    private String tempDir;

    @Inject @ConfigProperty(name = "elasticsearch.cluster.name", defaultValue = IndexDetails.CLUSTER_NAME)
    private String clusterName;

    private void startElastic(){
        if(!isRunning){

            String homePath = tempDir + SLASH + appName + SLASH;

            Settings.Builder settingsBuilder = Settings.builder()
                .put("path.home", homePath)
                .put("cluster.name", clusterName)
                .put("node.name", "internal")
                .put("client.transport.sniff", true)
                .put("node.max_local_storage_nodes",3);

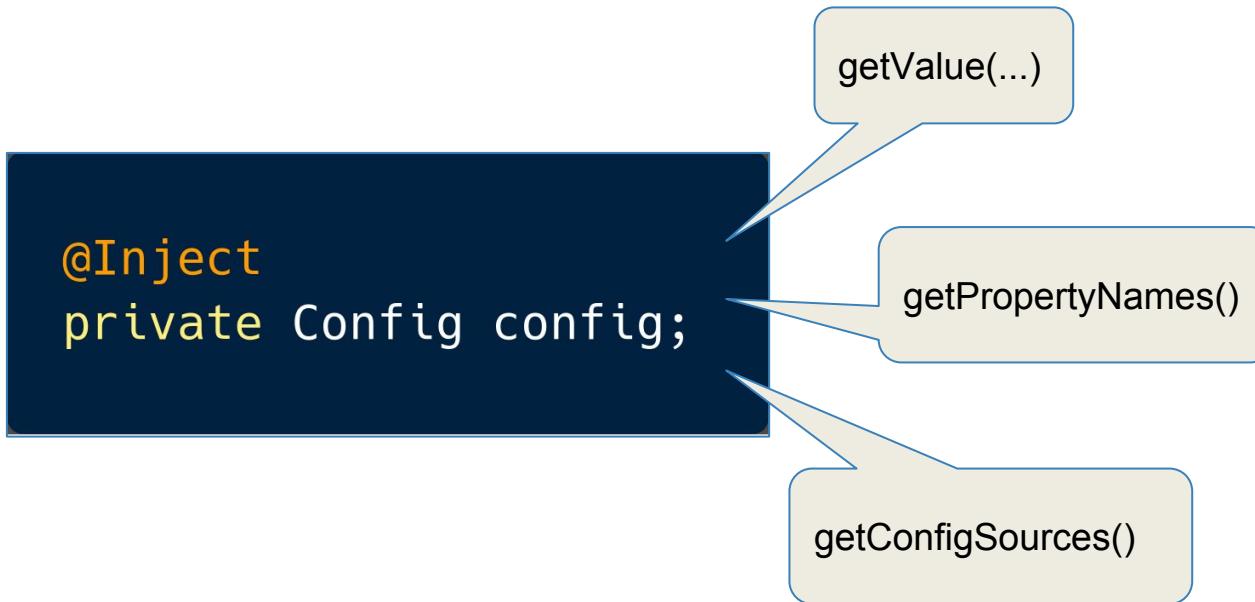
            ...
        }
    }
}
```

```
@Inject  
@ConfigProperty(name = "requiredProp", defaultValue = "default")  
private String required;
```

```
@Inject  
@ConfigProperty(name = "optionalProp", defaultValue = "default")  
private Optional<String> optional;
```

```
@Inject  
@ConfigProperty(name = "alwaysReloadedProp", defaultValue = "default")  
private Provider<String> alwaysReloaded;
```

# Configuration



# Configuration

By default there are 3 default config sources

Your own source...

Your own source...

`System.getProperties()`

`System.getenv()`

`META-INF/microprofile-config.properties`

```
public class MemoryConfigSource implements ConfigSource {  
  
    private static final Map<String, String> PROPERTIES = new HashMap<>();  
  
    @Override  
    public int getOrdinal() {  
        return 900;  
    }  
  
    @Override  
    public Map<String, String> getProperties() {  
        return PROPERTIES;  
    }  
  
    @Override  
    public String getValue(String key) {  
        if (PROPERTIES.containsKey(key)) {  
            return PROPERTIES.get(key);  
        }  
        return null;  
    }  
  
    @Override  
    public String getName() {  
        return "MemoryConfigSource";  
    }  
}
```

```
public class DynamicConfigSourceProvider implements ConfigSourceProvider {  
    @Override  
    public Iterable<ConfigSource> getConfigSources(ClassLoader forClassLoader) {  
        List<ConfigSource> configSources = new ArrayList<>();  
  
        Map<String, String> memoryMap = new HashMap<>();  
        memoryMap.put("test-prop", "test new memory prop");  
  
        configSources.add(new MemoryConfigSource(memoryMap));  
  
        return configSources;  
    }  
}
```

# Configuration - converters

- Boolean
  - Byte
  - Short
  - Int
  - Long
  - Float
  - Double
  - Character
  - Class
- Array
  - List
  - Set

```
@Inject  
@ConfigProperty(name = "avengers")  
private List<String> avengers;  
  
avengers=Iron man,Captain America,Thor
```

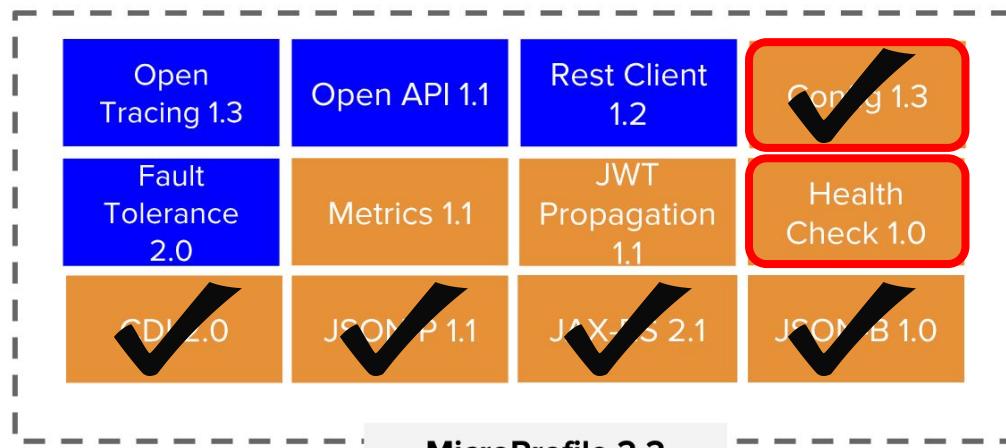
# Configuration - converters

```
@Inject  
@ConfigProperty(name = "avenger")  
private Avenger avenger;
```

```
avenger=Iron Man,Tony Stark,true
```

```
public class AvengerConverter implements Converter<Avenger> {
    @Override
    public Avenger convert(String value) {
        String[] split = value.split(",");
        return Avenger.name(split[0])
            .realName(split[1])
            .alive(Boolean.valueOf(split[2]))
            .build();
    }
}
```

# Eclipse MicroProfile 2.2 (Feb 2019)



■ = New

■ = Updated

■ = No change from last release (MicroProfile 2.1)

# 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



# Health

- MUST be compatible 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
@ApplicationScoped
public class MembershipHealthCheck implements HealthCheck {

    @Override
    public HealthCheckResponse call() {

        HealthCheckResponseBuilder responseBuilder = HealthCheckResponse.named("membership");
        try {
            Connection connection = datasource.getConnection();
            boolean isValid = connection.isValid(timeout);
            DatabaseMetaData metaData = connection.getMetaData();

            responseBuilder = responseBuilder
                .WithData("databaseProductName", metaData.getDatabaseProductName())
                .WithData("driverName", metaData.getDriverName())
                .WithData("isValid", isValid);

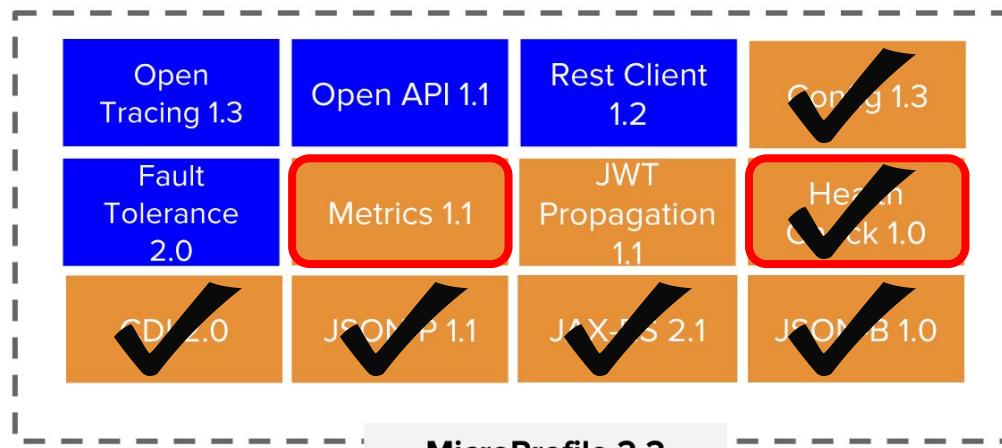
            return responseBuilder.state(isValid).build();
        } catch(SQLException e) {
            responseBuilder = responseBuilder.withData("exceptionMessage", e.getMessage());
            return responseBuilder.down().build();
        }
    }
}
```

# Health output - /health

```
{  
  "outcome": "UP",  
  "checks": [  
    {  
      "name": "heap-memory",  
      "state": "UP",  
      "data": {  
        "max %": "0.9",  
        "max": "7365197824",  
        "used": "185686984"  
      }  
    },  
  ],  
}
```

```
{  
  "name": "membership",  
  "state": "UP",  
  "data": {  
    "databaseProductVersion": "1.4.196 (2017-06-10)",  
    "databaseProductName": "H2",  
    "driverVersion": "1.4.196 (2017-06-10)",  
    "isValid": "true",  
    "driverName": "H2 JDBC Driver"  
  },  
  ...  
}  
]
```

# Eclipse MicroProfile 2.2 (Feb 2019)



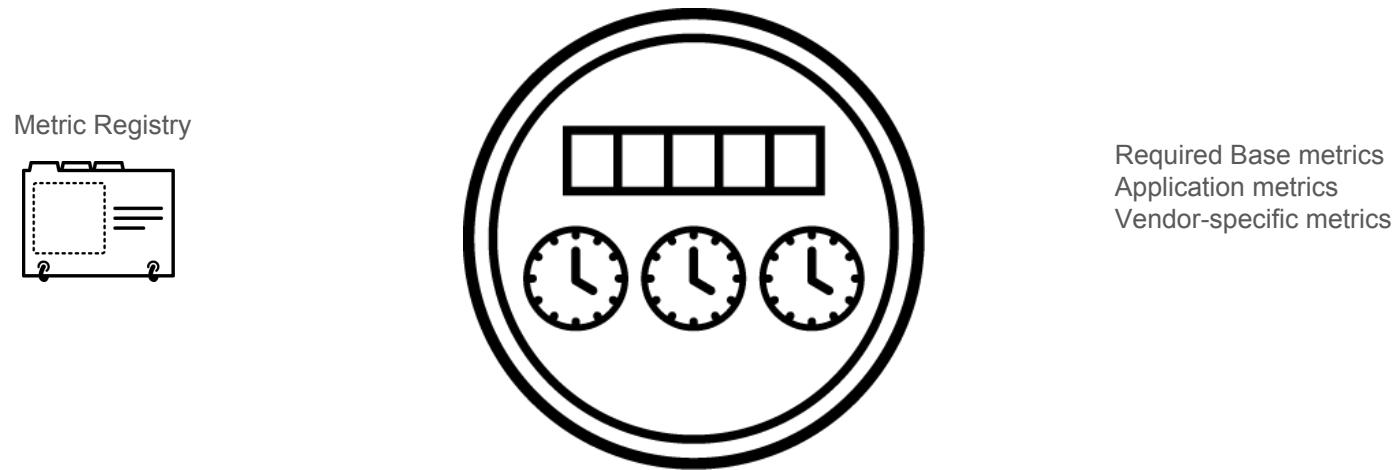
■ = New

■ = Updated

■ = No change from last release (MicroProfile 2.1)

# Metrics

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



# Metrics – scopes

- **Base**
  - metrics that all MicroProfile vendors have to provide
- **Vendor**
  - vendor specific metrics (optional)
- **Application**
  - application-specific metrics (optional)

# Metrics - base scope

## General

- UsedHeapMemory
- CommittedHeapMemory
- MaxHeapMemory
- GCCount & GCTime
- JVM Uptime

## Thread

- ThreadCount
- DaemonThreadCount
- PeakThreadCount
- ActiveThreads
- PoolSize

## ClassLoader

- LoadedClassCount
- TotalLoadedClassLoaded
- UnloadedClassCount

## Operating System

- AvailableProcessors
- SystemLoadAverage
- ProcessCpuLoad

# Metrics - application

```
@RequestScoped
@Path("/")
@Consumes(MediaType.APPLICATION_JSON) @Produces(MediaType.APPLICATION_JSON)
public class MembershipService {

    @GET
    @Timed(name = "Memberships requests time",absolute = true,unit = MetricUnits.MICROSECONDS)
    public List<Membership> getAllMemberships() {
        ...
    }

    @GET
    @Path("{id}")
    @Counted(name = "Membership requests",absolute = true,monotonic = true)
    public Membership getMembership(@NotNull @PathParam(value = "id") int id) {
        ...
    }

    ...
}
```

# Metrics - application

@Timed

@Gauge

@Counted

@Metered

@Metric

# Metrics - @Metric

```
@Path("ping")
@ApplicationScoped
public class PingResource {

    @Inject
    @Metric(name = "metric counter", absolute = true)
    private Counter metricCounter;

    @GET
    public Response ping() {
        metricCounter.inc();
        return Response.ok(metricCounter.getCount()).build();
    }
}
```

# Metrics - REST API

- /metrics
- /metrics/<scope>
  - /metrics/base
  - /metrics/vendor
  - /metrics/application
- /metrics/<scope>/<metric\_name>

# Metrics - REST API

/metrics/application/ping\_request\_counter

HTTP GET

```
{  
    "ping_request_counter" : 3  
}
```

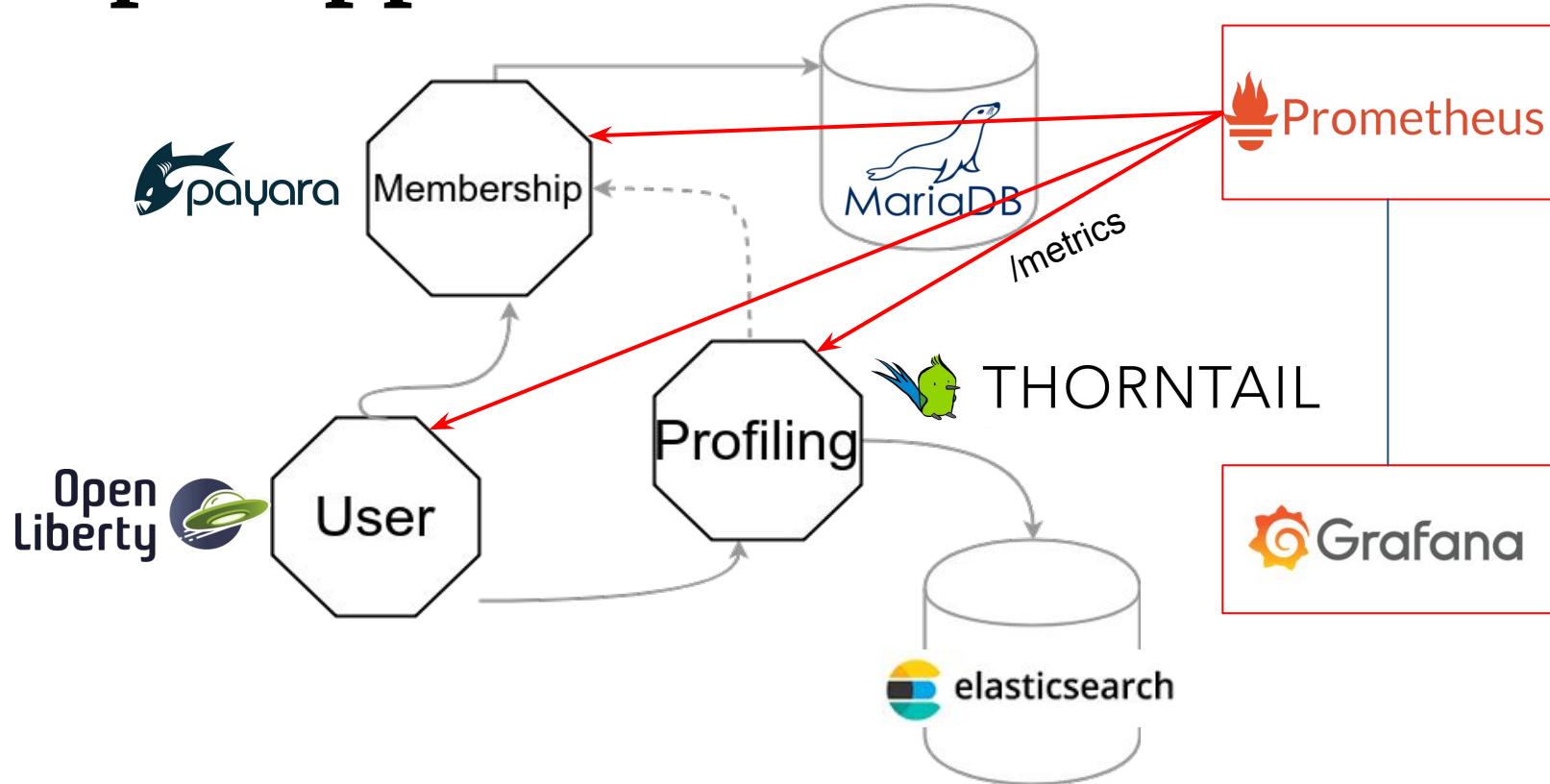
HTTP OPTIONS

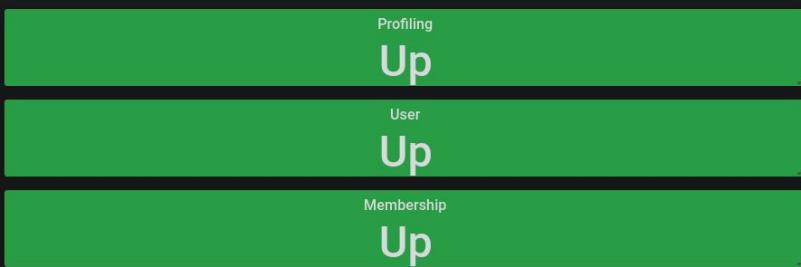
```
{  
    "ping_request_counter": {  
        "unit": "none",  
        "type": "counter",  
        "description": "Counter for the ping requests",  
        "displayName": "App ping counter",  
        "tags": "app=ping,deploy=new"  
    }  
}
```

# Metrics - Prometheus

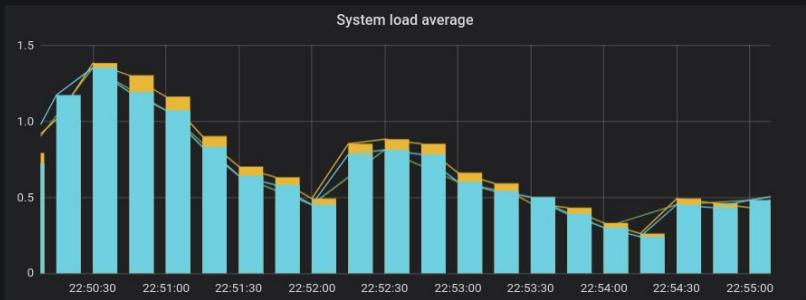
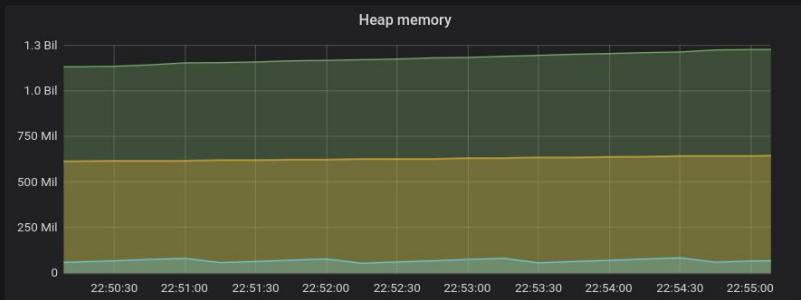
```
# HELP application:ping_request_counter Counter for the ping requests
# TYPE application:ping_request_counter counter
application:ping_request_counter{app="ping",deploy="new"} 3.0
```

# Example application





## ▼ Legoland

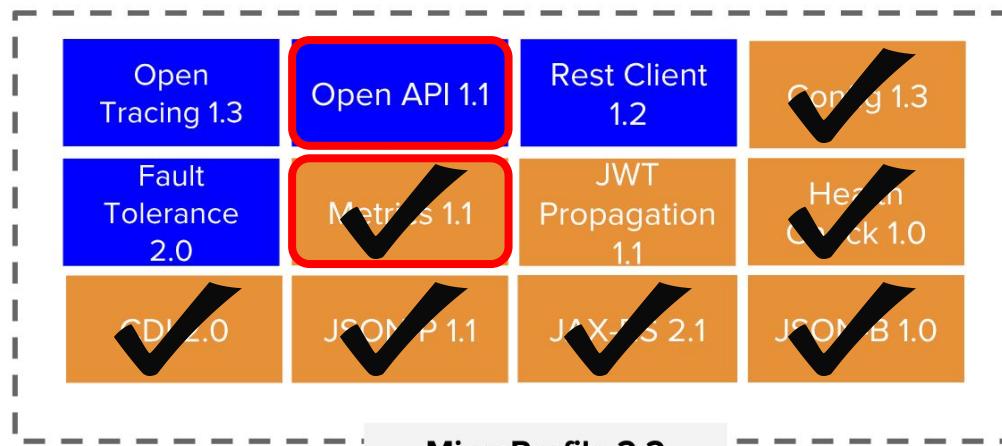


#openhouseredsat #microprofile



@xstefank @RedHat

# Eclipse MicroProfile 2.2 (Feb 2019)



■ = New

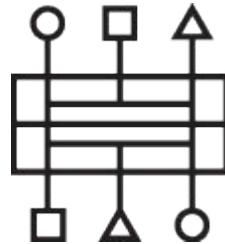
■ = Updated

■ = No change from last release (MicroProfile 2.1)

# Open API

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.





# Open API

- Enterprise Java Binding of the [OpenAPI v3](#) specification
- Based on [Swagger Core](#)
- OpenAPI
  - Defines a standard, programming language-agnostic interface description for REST APIs
  - Understandable by humans and machines



**ebay**

**P**  
**PayPal**

**Google**

# Open API

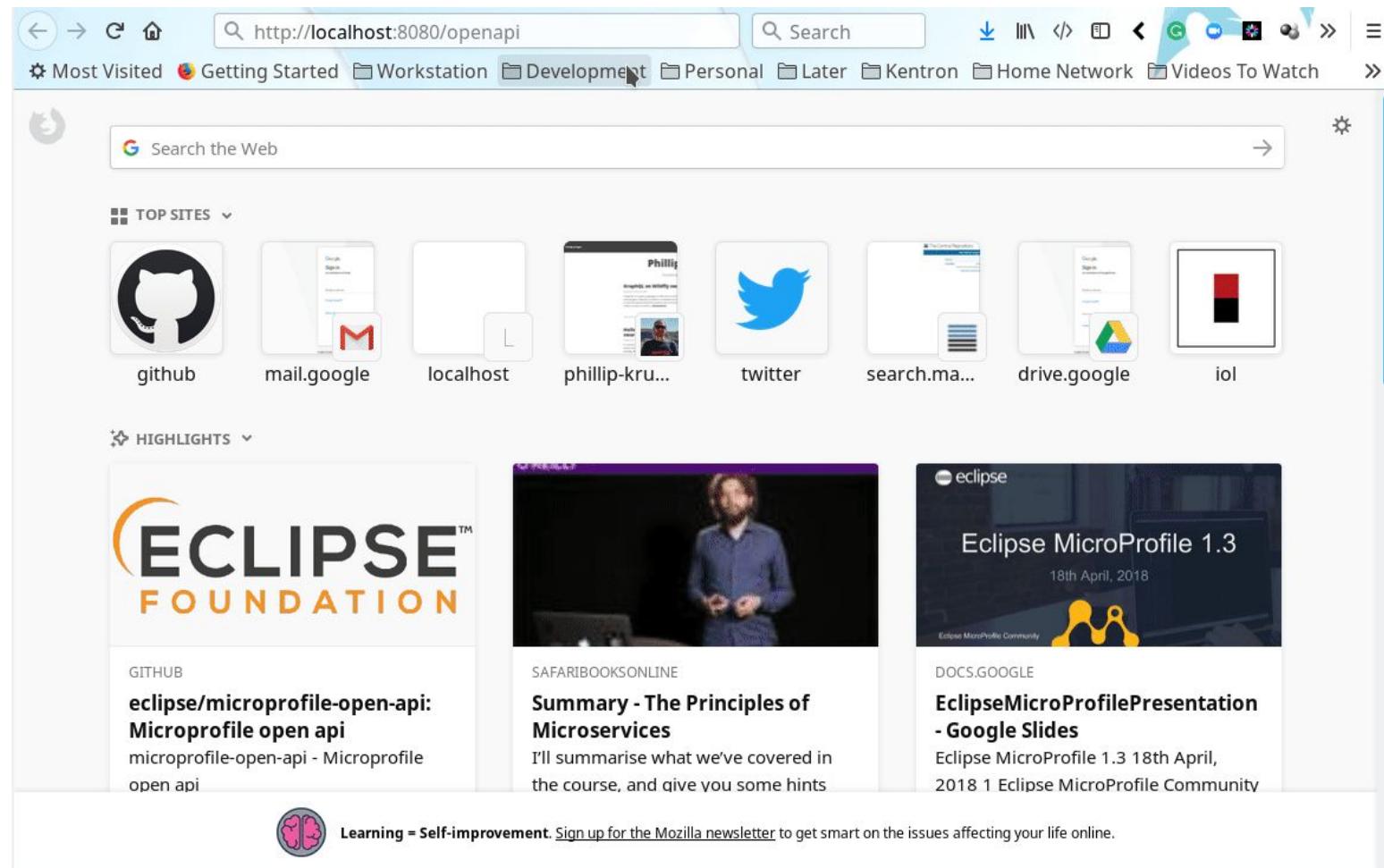
## /openapi

```
openapi: 3.0.1
info:
  title: Generated API
  version: "1.0"
paths:
  /ping:
    get:
      responses:
        200:
          description: OK
          content:
            '*/*': {}
```

```
@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 {

    @GET
    @Path("{id}")
    @Operation(description = "Get a certain Membership by id")
    @APIResponses({
        @APIResponse(responseCode = "200", description = "Successful, returning membership",
            content = @Content(mediaType = MediaType.APPLICATION_JSON,
                schema = @Schema(implementation = Membership.class))),
        @APIResponse(responseCode = "504", description = "Service timed out"),
        @APIResponse(responseCode = "401", description = "User not authorized")
    })
    public Membership getMembership(@NotNull @PathParam(value = "id") int id) {
        ...
    }
    ...
}
```



# #openhousereredhat #microprofile

@xstefank

@RedHat

A screenshot of a Mozilla Firefox browser window. The address bar shows "Search with Google or enter address". The toolbar includes icons for back, forward, search, and other browser functions. The menu bar shows "Most Visited", "Getting Started", "Workstation", "Development", "Personal", "Later", "Kenton", "Home Network", "Videos To Watch", and "Help". The main content area features a search bar with "Search the Web" and a "TOP SITES" section with icons for GitHub, mail.google, localhost, phillip-kru..., twitter, search.ma..., drive.google, and iol. Below this is a "HIGHLIGHTS" section with three cards: 1) Eclipse Foundation logo with links to GitHub repositories for Microprofile open api. 2) A video thumbnail for "SAFARIBOOKSONLINE Summary - The Principles of Microservices" by Phillip Kruger. 3) A presentation slide for "Eclipse MicroProfile 1.3" dated 18th April, 2018, by Eclipse MicroProfile Community.

<https://github.com/microprofile-extensions/openapi-ext/tree/master/swagger-ui>

#openhousereredhat #microprofile

@xstefank @RedHat

# Eclipse MicroProfile 2.2 (Feb 2019)



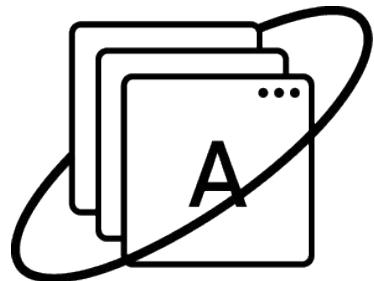
- = New
- = Updated
- = No change from last release (MicroProfile 2.1)

6

[HTTPS://MICROPROFILE.IO/](https://microprofile.io/) | [HTTPS://PROJECTS.ECLIPSE.ORG/PROJECTS/TECHNOLOGY.MICROPROFILE](https://projects.eclipse.org/projects/technology.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**. 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

```
Response response = ClientBuilder.newClient()
    .target("http://localhost:8080/membership/" + membershipId)
    .request()
    .header("Authorization", "Bearer " + token)
    .get();

if (response.getStatus() != Response.Status.OK.getStatusCode()) {
    throw new RuntimeException("Invalid return value - " + response.getStatus());
}

Membership membership = response.readEntity(Membership.class);
```



```
Membership membership = membershipProxy.getMembership("Bearer " + token, membershipId);
```

# REST Client - definition

```
@RequestScoped  
@RegisterRestClient  
@Consumes(MediaType.APPLICATION_JSON) @Produces(MediaType.APPLICATION_JSON)  
@RegisterProvider(RuntimeResponseExceptionMapper.class)  
@Path("/api")  
public interface MembershipProxy {  
  
    @GET @Path("/{id}")  
    public Membership getMembership(@HeaderParam("Authorization") String authorization,  
                                    @NotNull @PathParam(value = "id") int id);  
}
```

# REST Client - injection

```
@Inject  
@RestClient  
private MembershipProxy membershipProxy;
```

<class\_name>/mp-rest/url

```
com.github.phillipkruger.profiling.membership.MembershipProxy/mp-rest/url=http://localhost:8080/membership
```

# REST Client - builder

```
membershipProxy = RestClientBuilder.newBuilder()
    .baseUrl(new URL("http://localhost:8080/membership"))
    .build(MembershipProxy.class);
```

# REST Client - exception mapping

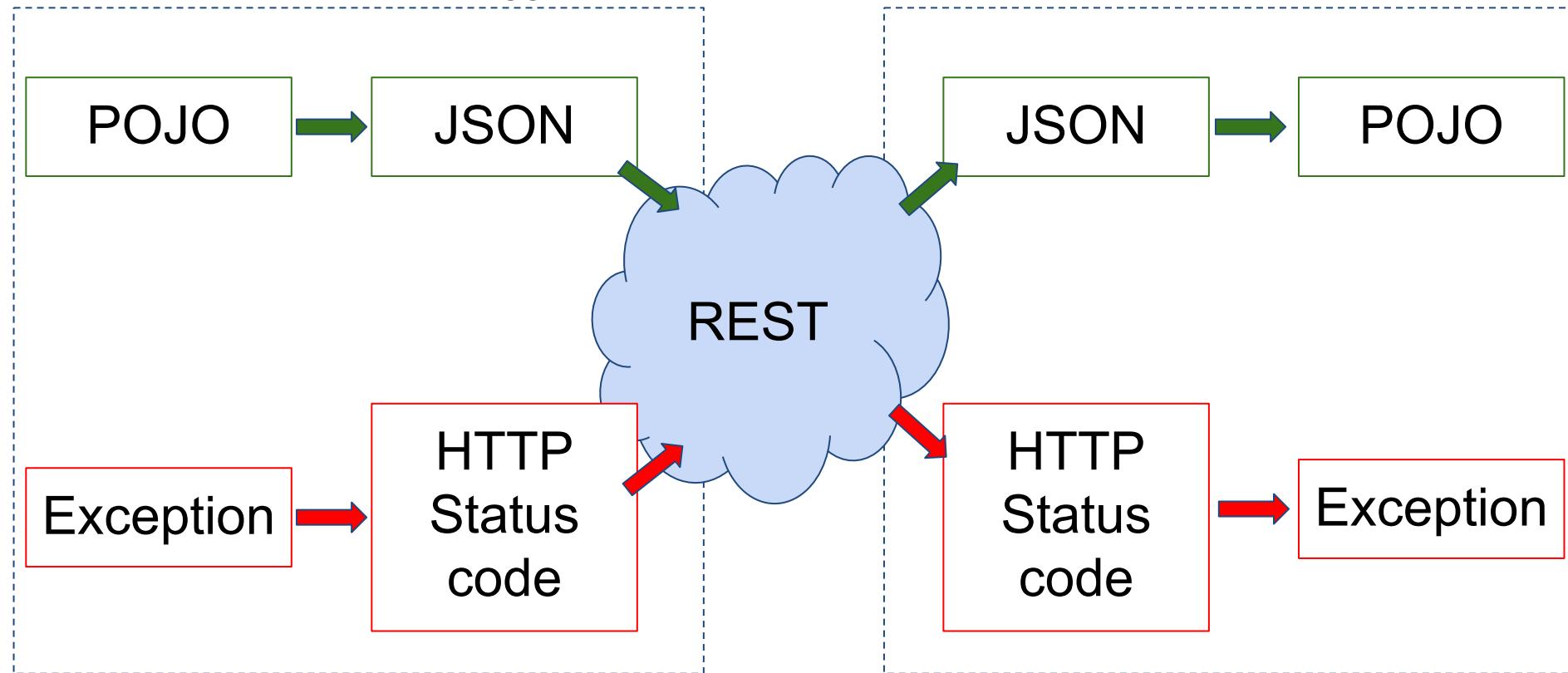
```
@Provider
public class RuntimeResponseExceptionMapper implements ResponseExceptionMapper<RuntimeException>{

    @Override
    public boolean handles(int status, MultivaluedMap<String, Object> headers) {
        return codes.contains(status);
    }

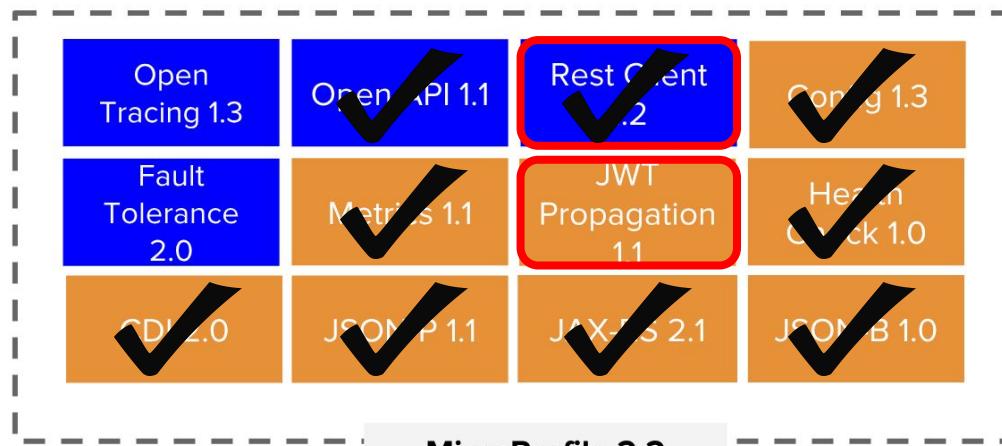
    @Override
    public RuntimeException toThrowable(Response response) {
        ...
        return new RuntimeException(...);
    }
    ...
}
```

JAX-RS + OpenAPI + Swagger UI

REST Client



# Eclipse MicroProfile 2.2 (Feb 2019)



- = New
- = Updated
- = No change from last release (MicroProfile 2.1)

6

[HTTPS://MICROPROFILE.IO/](https://microprofile.io/) | [HTTPS://PROJECTS.ECLIPSE.ORG/PROJECTS/TECHNOLOGY.MICROPROFILE](https://projects.eclipse.org/projects/technology.microprofile)

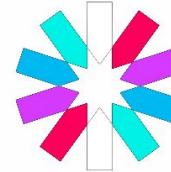
# JWT propagation

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



# JWT propagation

Multiple projects/standards directly influenced this proposal and acted as basis for this API, such as:



J W U T

Goal:

- One of the main strategies to propagate the security state from clients to services or even from services to services involves the use of **security tokens**.
- For RESTful based microservices, security tokens offer a very lightweight and interoperable way to propagate identities across different services.

# JWT propagation

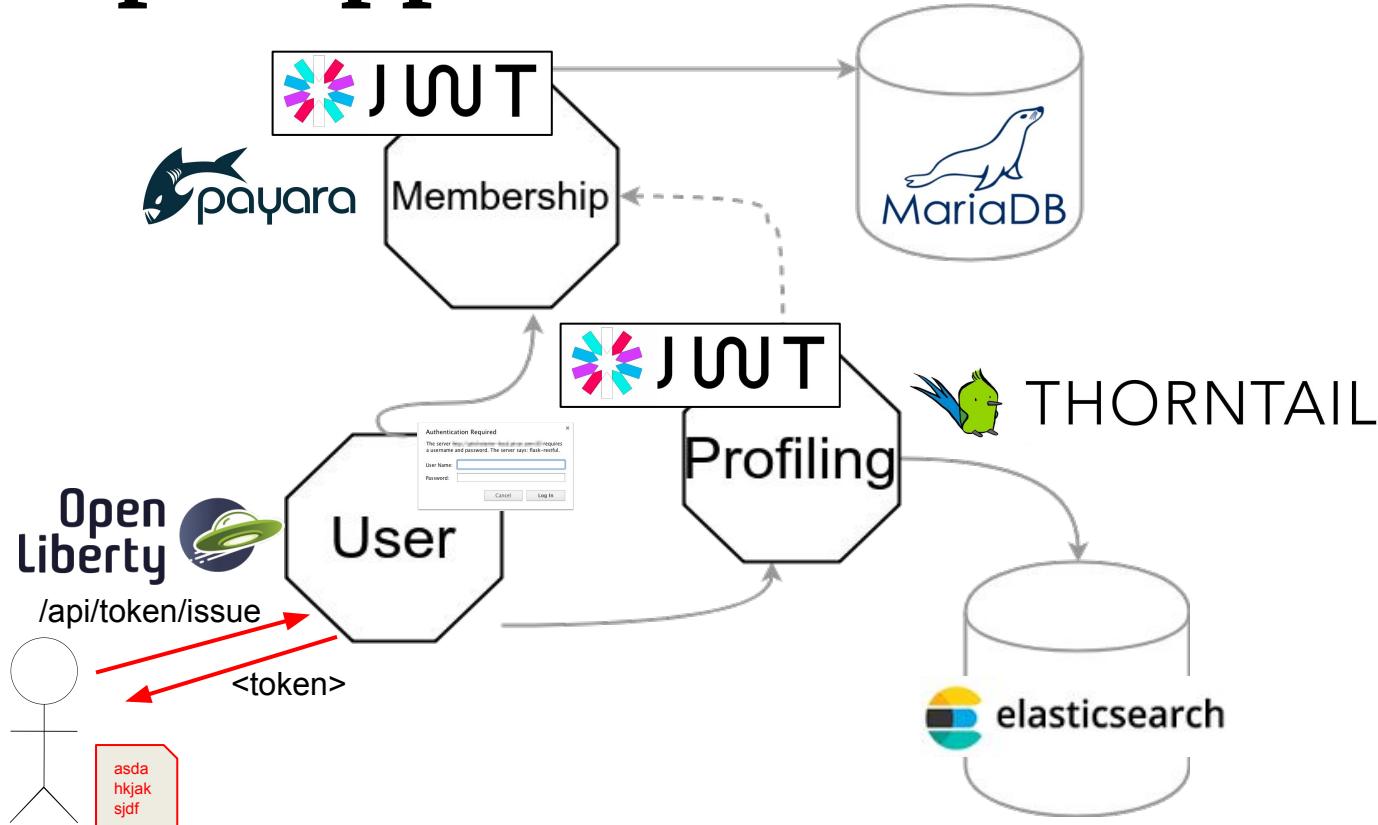
Deconstructing REST Security  
by David Blevins

[https://www.youtube.com/watch?v=9CJ\\_BAeOmW0](https://www.youtube.com/watch?v=9CJ_BAeOmW0)



David Blevins - Tomitribe

# Example application



A screenshot of a web browser window titled "Token service". The address bar shows "localhost:9080/openapi/ui/". The page content is the "Token service" documentation, featuring a dark header with the "Open Liberty" logo. Below the header, the title "Token service" is displayed with a "1.0.0" badge and an "OAS3" badge. A dropdown menu labeled "Servers" is open, showing "http://localhost:9080/user" as the selected option. The main content area is titled "Token service JWT Issuer". It shows a "GET /api/token/issue" endpoint. At the bottom of the page, there are links to "Phillip Kruger - Website" and "Contact Phillip Kruger".

```
mstefank@Set-FI: ~/GIT/xstefank/microprofile-demo
mstefank@Set-FI ~/GIT/xstefank/microprofile-demo (master) $ █
```

# JWT - token

```
eyJraWQiOiJcL3ByaXhdGVlZXkucGVtIiwidHlwIjoisldUIiwiYWxnIjoiUlMyNTYifQ.eyJhdWQ  
iOiJyYXNwYmVycnktcGkiLCJzdWIiOiJwaGlsbGwLmtydWdlckBwaGlsbGwLWtydWdlci5jb20iL  
CJ1cG4iOiJwaGlsbGwLmtydWdlckBnbWFpbC5jb20iLCJpc3MiOiJodHRwOlwvXC9sZWdvbGFuZC5  
waGlsbGwLWtydWdlci5jb20iLCJncm91cHMiOlsidXNlciiIsImFkbWluIl0sImV4cCI6MTU0ODI3N  
Dc10SwiaWF0IjoxNTQ4MjcyOTU5LCJqdGkiOiJmYTQxNjU2OS1jODNlLTTrmYTUt0WE5MC1kOGUwMGQ  
5YzFhNzcifQ.DB6TwtL3s6zwz_HASaO_TrnvvAM8--TqHbP545ibeBATpP4bkxs5WEvwmMnFadyPzb  
MECKeBHIjYA1Gc5v5kVof1No1KbexB9fbmxaA9tL-ZBl7dfugV9eTwskVnIcG6iqIx5jCaZD2i8q8C  
7In3Waq207TRc0i1XXPGotgP3ss-XYJ_GQSKWbQ8HMwSdQFmBS2t57dkdPl6LP9Zc062IAINUDiHEE  
yPI6rTWw9IIDXPIdkpjeJvSKJw950Pv5SkaGbb8NsZSiBh0hVAvaMoQjqr-jmpurJ_7Pcp1KhrgAHo  
C8fxQ0h5gHkhjTjsGhVWQRVtHgJtk34QfBrLK1Y2XQ
```

# JWT - token

```
{  
    "aud": "raspberry-pi",  
    "sub": "phillip.kruger@phillip-kruger.com",  
    "upn": "phillip.kruger@gmail.com",  
    "iss": "http://legoland.phillip-kruger.com",  
    "groups": [  
        "user",  
        "admin"  
    ],  
    "exp": 1548274759,  
    "iat": 1548272959,  
    "jti": "fa416569-c83e-4fa5-9a90-d8e00d9c1a77"  
}
```

```
@ApplicationPath("/api")
@OpenAPIDefinition(info = @Info(
    title = "Profile 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 = "/profiling",description = "localhost"),
    @Server(url = "http://red:7080/profiling",description = "Red Pi")
}
)
@SecurityScheme(description = "The JWT from User service",
    securitySchemeName = "Authorization",
    in = SecuritySchemeIn.HEADER,
    type = SecuritySchemeType.HTTP,
    scheme = "bearer",
    bearerFormat = "JWT")
@LoginConfig(authMethod = "MP-JWT",realmName = "jwt-domain")
@DeclareRoles({"user", "admin"})
public class ApplicationConfig extends Application {
```

OpenAPI

JWT

```
@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)),
    @APIResponse(responseCode = "503", description = "Problem with the connection to a downstream service",
        headers = @Header(name = REASON))
})
@SecurityRequirement(name = "Authorization")
@RolesAllowed({"admin", "user"})
public Response getUserEvents(
    ...
}
```

The code snippet illustrates the integration of three annotation sources:

- OpenAPI**: Annotations like `@APIResponse`, `@APIResponses`, and `@SecurityRequirement` are highlighted with a red box.
- JWT**: Annotations like `@RolesAllowed` are highlighted with a yellow box.
- Red Hat**: Annotations like `@GET`, `@Path`, `@Operation`, and `public Response getUserEvents` are highlighted with a blue box.

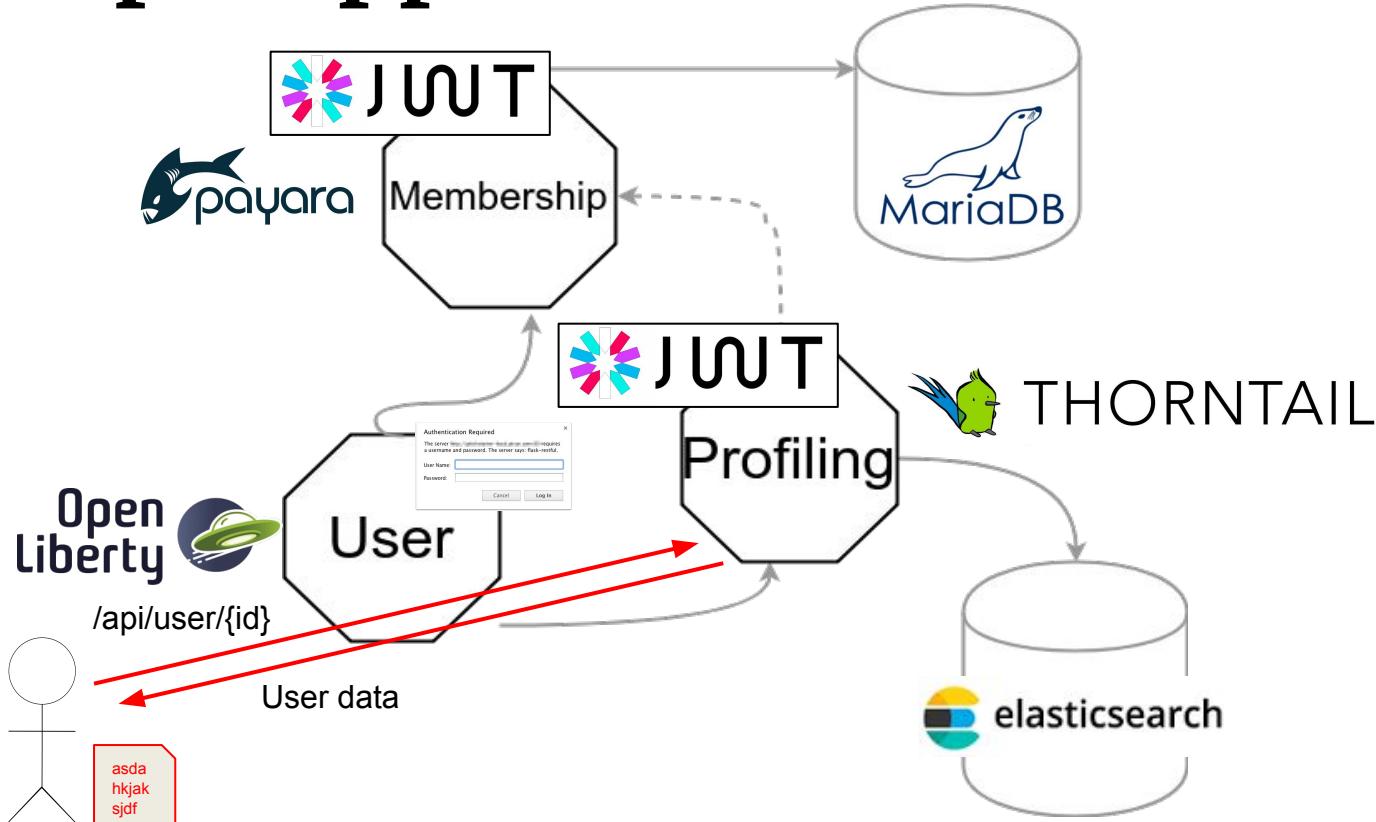
# JWT - public key

*mp.jwt.verify.publickey*

*mp.jwt.verify.publickey.location*

```
java -jar movieservice.jar -Dmp.jwt.verify.publickey.location=orange.pem
```

# Example application



Token service Profiling API Membership API

localhost:7080/profiling/api/openapi-ui/index.html

Search

swagger

# Profile service 1.0.0 OAS3

/openapi

[Phillip Kruger - Website](#)  
[Send email to Phillip Kruger](#)

Authorize 

## Profile service Build up a profile of the user

POST /api



GET /api/event/{eventName}



GET /api/location/{location}



Created with MicroProfile Ext: OpenAPI . © 2018 Phillip Kruger | Running on Thorntail 2.0.0.Final (WildFly Core 3.0.8.Final) - 1.4.18.Final

#openhousederedhat #microprofile

@xstefank @RedHat

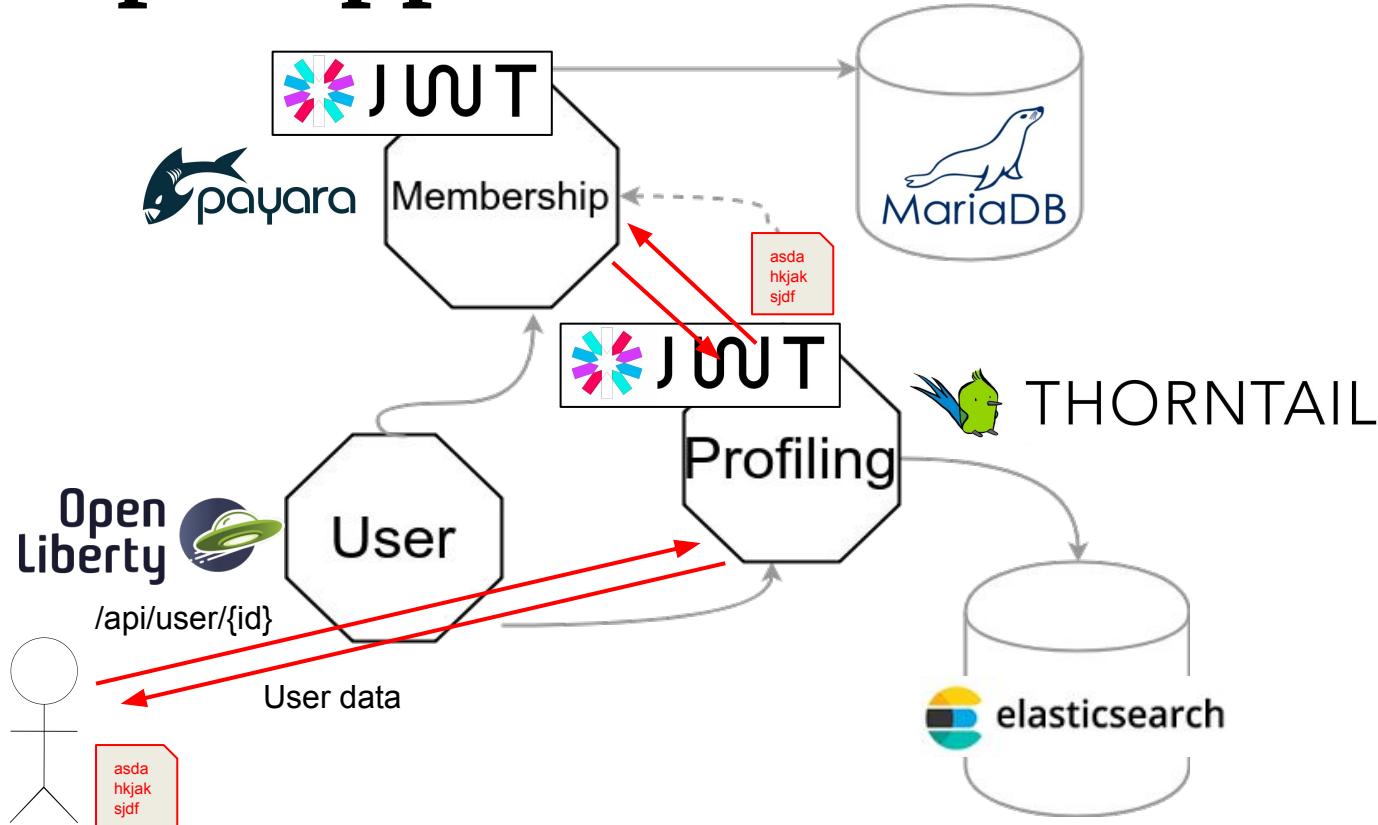
```
mstefank@Set-FI: ~/GIT/xstefank/microprofile-demo
```

```
mstefank@Set-FI ~/GIT/xstefank/microprofile-demo (master*) $ █
```

#openhousereredhat #microprofile

@xstefank @RedHat

# Example application



# Eclipse MicroProfile 2.2 (Feb 2019)



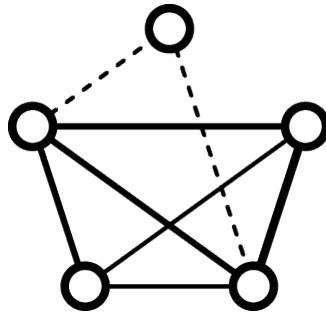
■ = New

■ = Updated

■ = No change from last release (MicroProfile 2.1)

# OpenTracing

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**.





OPENTRACING

# OpenTracing

- Enterprise Java Binding to [OpenTracing](#) specification
- Defines behaviors and an API for accessing an OpenTracing compliant Tracer object within your application
- Behaviors specify how incoming and outgoing requests will have OpenTracing Spans automatically created



INSTANA



sourcegraph



OPENTRACING



Hawkular



stage monitor



DATADOG

# Skywalking



JAEGER



<http://opentracing.io/>

87

#openhousereredhat #microprofile

@xstefank @RedHat

```
@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)),
    @APIResponse(responseCode = "503", description = "Proplem with a connection to a downstream service",
        headers = @Header(name = REASON))
})
@SecurityRequirement(name = "Authorization")
@RolesAllowed({"admin","user"})
@Traced(operationName = "GetUserEvents", value = true)
public Response getUserEvents() {
    ...
}
```

```
@Path("ping")
public class PingResource {

    @Inject
    private Tracer tracer;

    @GET
    public Response ping() {
        tracer.activeSpan().setTag("custom", "tag");
        tracer.activeSpan().setBaggageItem("host", System.getProperty("os.name"));

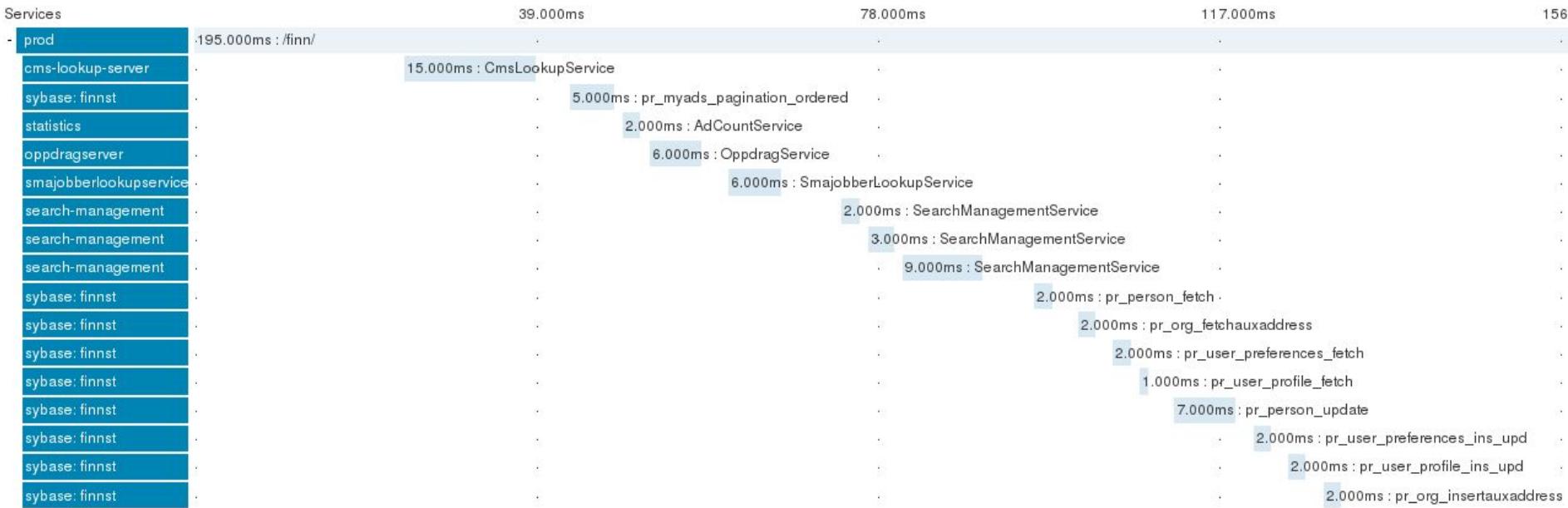
        return Response.ok("Application running successfully").build();
    }
}
```

Duration: 195.000ms

Services: 8

Depth: 2

Total Spans: 17

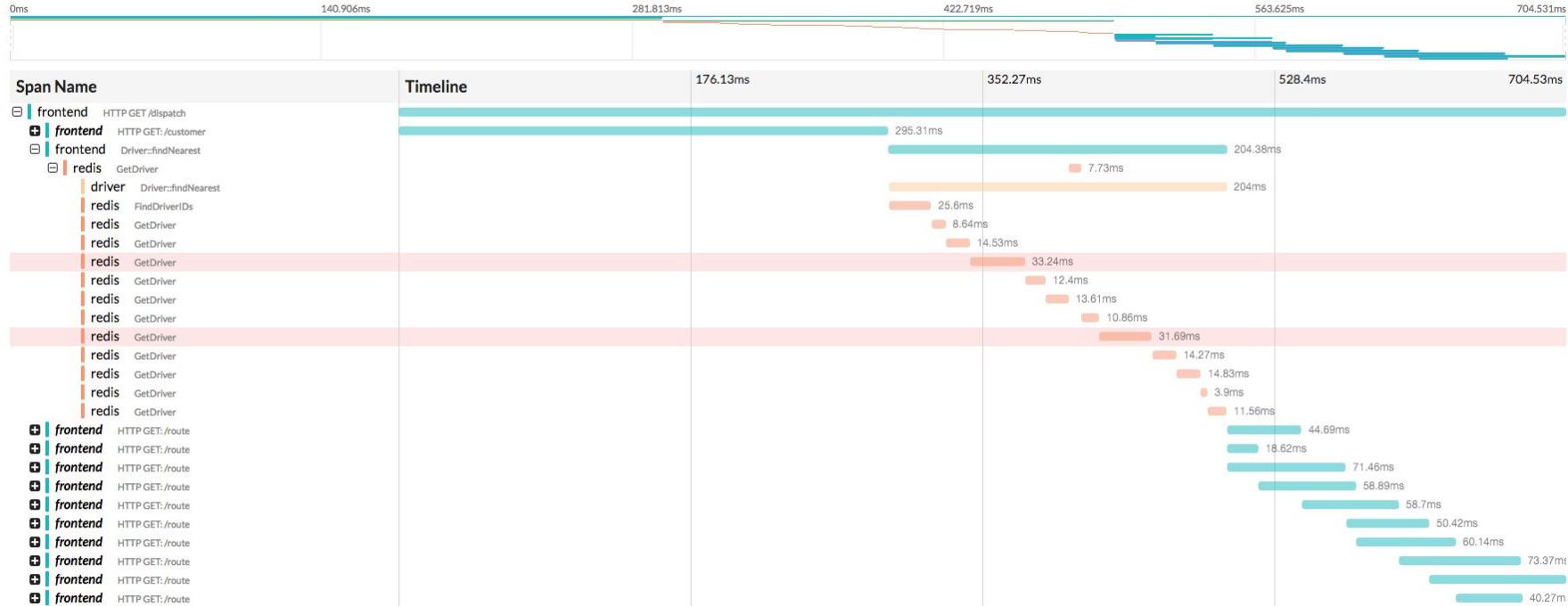
[Expand All](#)[Collapse All](#) Filter Service Search[cms-lookup-server x1](#) [oppdragserver x1](#) [prod x1](#) [search-management x3](#) [smajobber-thrift-server x1](#) [smajobberlookupservice x1](#) [statistics x1](#) [sybase: finnst x9](#)

❖ **frontend: HTTP GET /dispatch**

Trace Start: April 12, 2017 9:12 AM Duration: 704.531ms Services: 6 Depth: 5 Total Spans: 500

### **View Options**

Search...



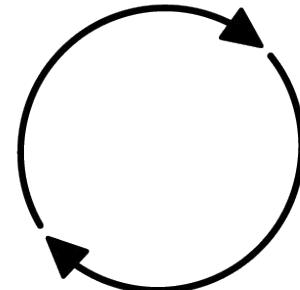
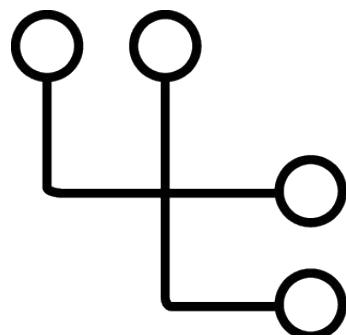
# Eclipse MicroProfile 2.2 (Feb 2019)



[HTTPS://MICROPROFILE.IO/](https://microprofile.io/) | [HTTPS://PROJECTS.ECLIPSE.ORG/PROJECTS/TECHNOLOGY.MICROPROFILE](https://projects.eclipse.org/projects/technology.microprofile)

# Fault Tolerance

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

Multiple projects directly influenced this proposal and acted as basis for this API, such as:



**HYSTRIX**  
DEFEND YOUR APP

## Failsafe

Goal:

- Separate the responsibility of executing logic (Runnables/Callables/etc) from guiding when execution should take place (through retry policies, bulkheads, circuit breakers)

# Fault Tolerance

- `@Timeout`
- `@Retry`
- `@Fallback`
- `@CircuitBreaker`
- `@Bulkhead`

# @Timeout

```
@GET  
@Timed(name = "Memberships requests time", absolute = true, unit = MetricUnits.MICROSECONDS)  
@Operation(description = "Get all the current memberships")  
@APIResponses({  
    @APIResponse(responseCode = "200", description = "Successful. returning members"),  
    @APIResponse(responseCode = "504", description = "Service timed out"),  
    @APIResponse(responseCode = "401", description = "User not authorized")  
})  
@RolesAllowed({"admin"})  
@SecurityRequirement(name = "Authorization")  
@Timeout(value = 3, unit = ChronoUnit.SECONDS)  
public List<Membership> getAllMemberships() {  
    // Some bad code went into production...  
    if(activateBadCode){  
        try {  
            log.severe("Sleeping for 10 seconds...");  
            TimeUnit.SECONDS.sleep(10L);  
        } catch (InterruptedException ex) {  
            log.severe(ex.getMessage());  
        }  
    }  
  
    TypedQuery<Membership> query = em.createNamedQuery(Membership.QUERY_FIND_ALL, Membership.class);  
    return query.getResultList();  
}
```

OpenAPI

Fault  
Tolerance



swagger

# Membership service 1.0.0 OAS3

[/openapi](#)[Phillip Kruger - Website](#)[Send email to Phillip Kruger](#)[Authorize](#)

## Membership service Managing the membership

[GET](#) /api[POST](#) /api[GET](#) /api/{id}

Created with MicroProfile Ext: OpenAPI . © 2018 Phillip Kruger | Running on Payara Micro #badassfish

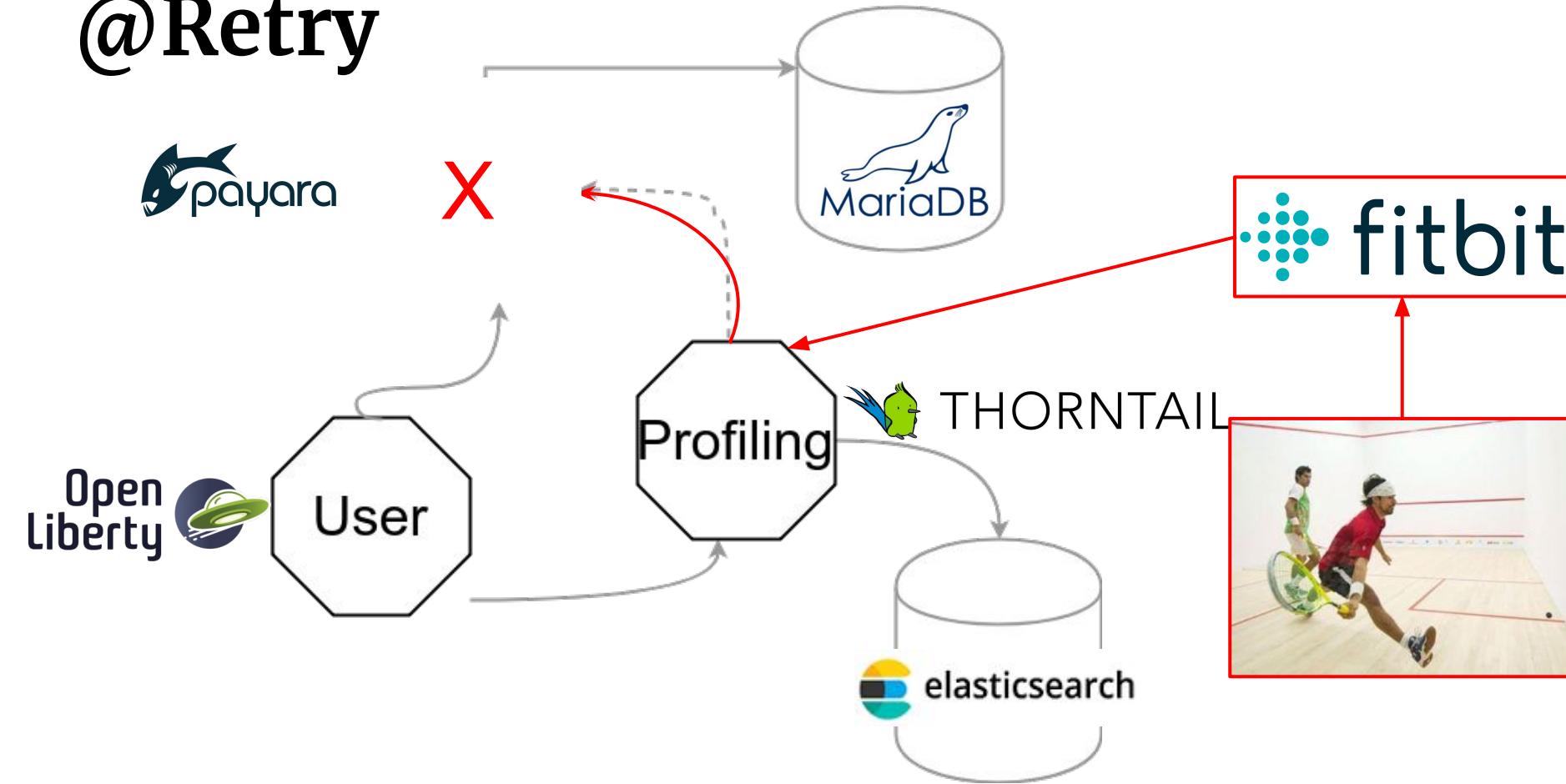
```
mstefank@Set-FI: ~/GIT/xstefank/microprofile-demo
```

```
mstefank@Set-FI ~ /GIT/xstefank/microprofile-demo (master*) $ █
```

```
mstefank@Set-FI: ~/GIT/xstefank/microprofile-demo
```

```
mstefank@Set-FI ~ /GIT/xstefank/microprofile-demo (master*) $ █
```

# @Retry



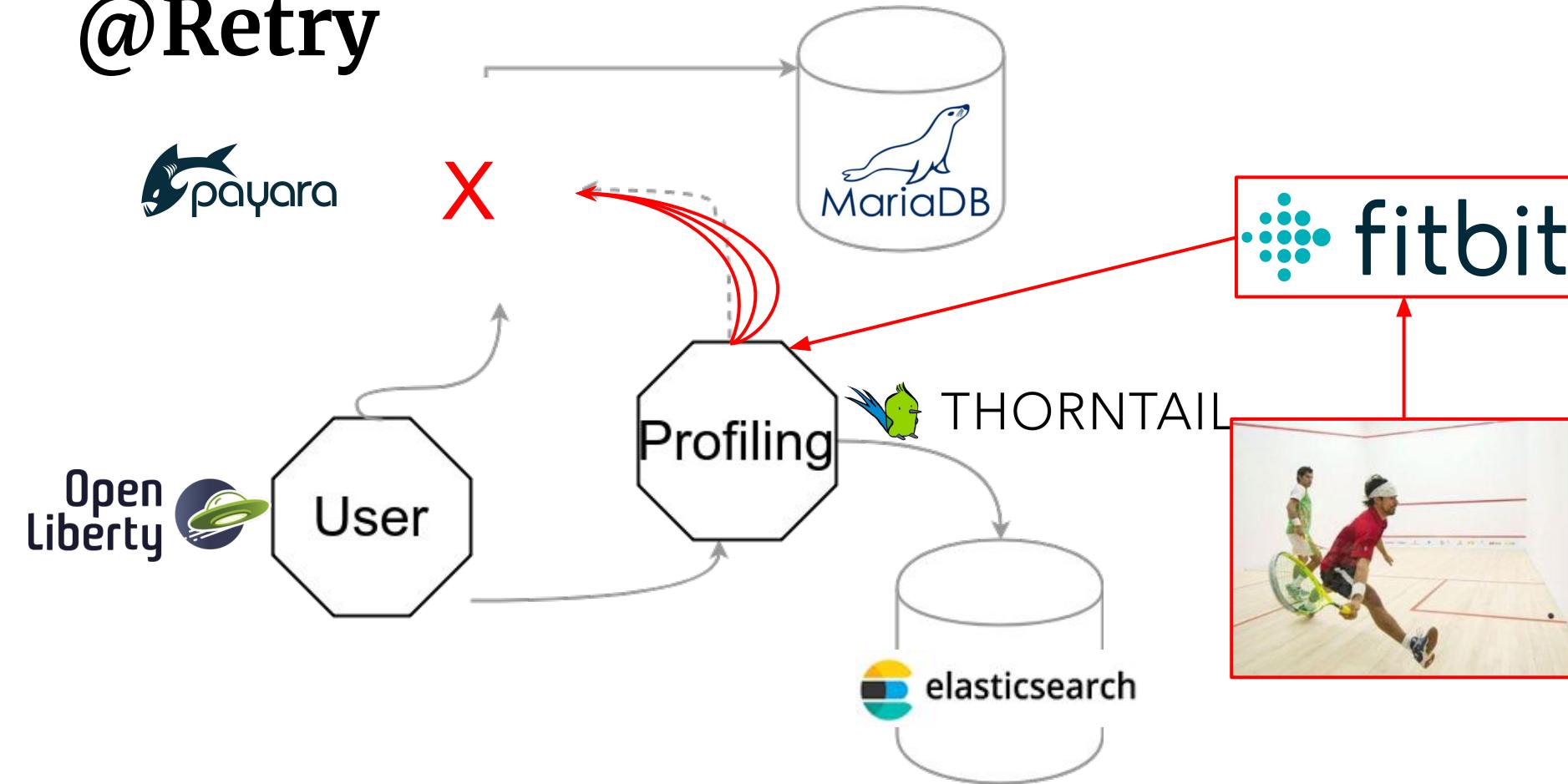
# @Retry

```
@Counted(name = "Events logged", absolute = true, monotonic = true)
@Asynchronous
@Retry(delay = 10, delayUnit = ChronoUnit.SECONDS, maxRetries = 5)
public Future<Void> logEvent(String token, @NotNull UserEvent event){
    log.log(Level.SEVERE, ">>> Now (trying to )log event [{0}] ...", event);
    JsonObject json = converter.toJsonObject(event);
    int membershipId = json.getInt("userId");

    validateMembership(token, membershipId);

    ...
}
```

# @Retry



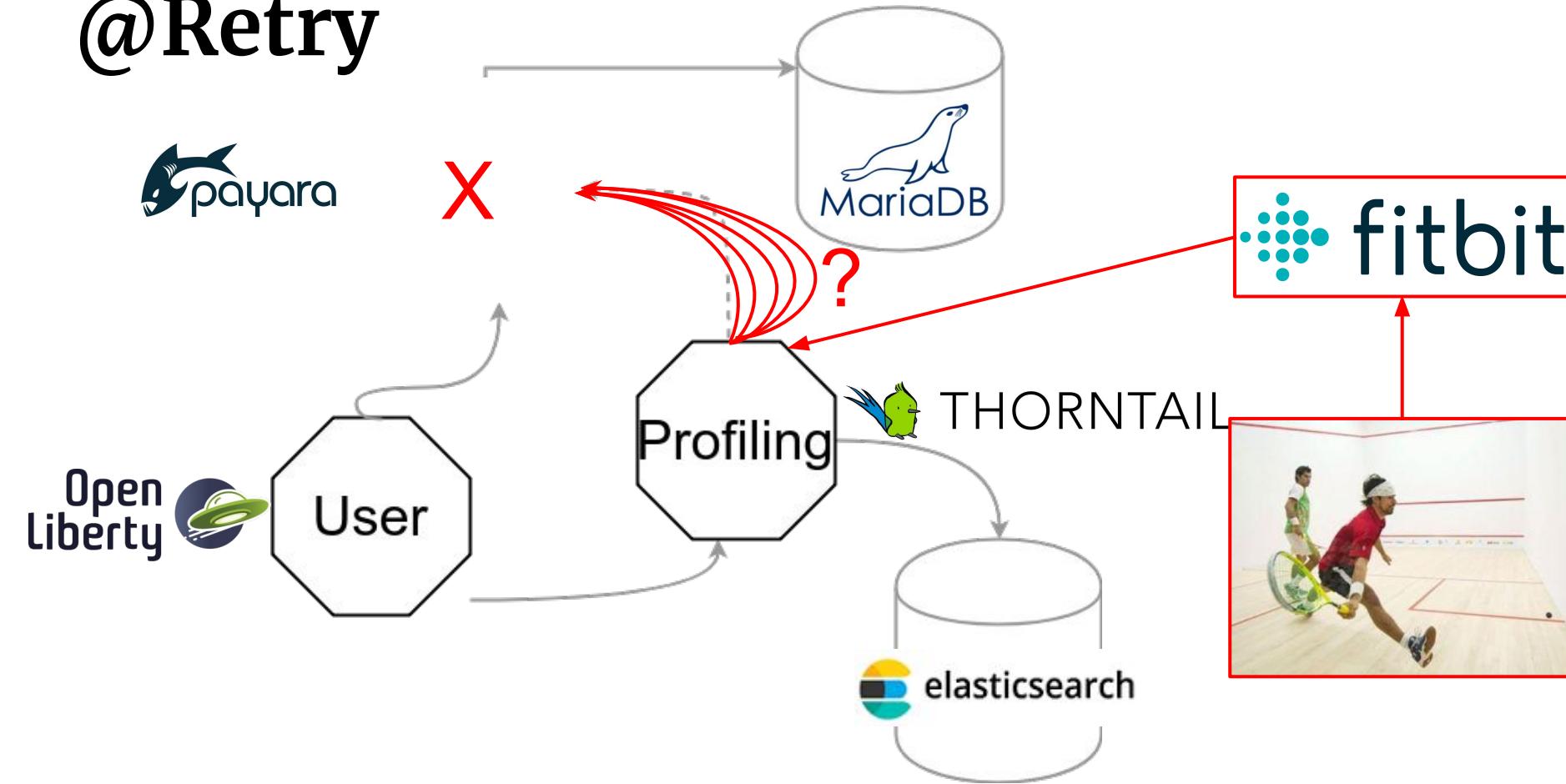
```
mstefank@Set-FI: ~/GIT/xstefank/microprofile-demo  
mstefank@Set-FI: ~/GIT/xstefank/microprofile-demo (master*) $
```

]

```
mstefank@Set-FI: ~/GIT/xstefank/microprofile-demo  
mstefank@Set-FI: ~/GIT/xstefank/microprofile-demo (master*) $
```

```
mvn-or-mvnw clean install -Prun  
2) OpenAPI document initialized: io.smallrye.openapi.api.models.OpenAPIImpl@554748a3  
2019-01-24 17:33:16,634 INFO [org.jboss.resteasy.resteasy_jaxrs.i18n] (ServerService Thread Pool -- 12 ) RESTEASY002225: Deploying javax.ws.rs.core.Application: class com.github.phillipkruger.profiling.ApplicationConfig$Proxy$ $$ WeldClientProxy  
2019-01-24 17:33:16,649 INFO [org.wildfly.swarm.mpopentracing.deployment.TracerProducer] (ServerService Thread Pool -- 12 ) Registering GlobalTracer[NoopTracer] to GlobalTracer and providing it as CDI bean.  
2019-01-24 17:33:16,710 INFO [org.wildfly.extension.undertow] (ServerService Thread Pool -- 12 ) WFLYUT0021: Registered web context: '/profiling' for server 'default-server'  
2019-01-24 17:33:16,735 INFO [org.jboss.as.server] (main) WFLYSRV0010: Deployed "profiling.war" (runtime-name : "profiling.war")  
2019-01-24 17:33:16,743 INFO [org.wildfly.swarm] (main) THORN99999: Thorntail is Ready
```

# @Retry



# @Fallback

```
@Counted(name = "Events logged", absolute = true, monotonic = true)
@Asynchronous
@Retry(delay = 10, delayUnit = ChronoUnit.SECONDS, maxRetries = 5)
@Fallback(EventLoggerFallbackHandler.class)
public Future<Void> logEvent(String token, @NotNull UserEvent event){
    log.log(Level.SEVERE, ">>> Now (trying to )log event [{0}] ...", event);
    JsonObject json = converter.toJsonObject(event);
    int membershipId = json.getInt("userId");

    validateMembership(token, membershipId);

    ...
}
```

```
public class EventLoggerFallbackHandler implements FallbackHandler<Future<Void>>{  
  
    @Override  
    public Future<Void> handle(ExecutionContext context) {  
        UserEvent event = (UserEvent)context.getParameters()[1];  
  
        // Maybe log a JIRA ? Notify someone ?  
        // In our case we save in another index...  
        // We can later automatically save these events once the membership service is back up  
  
        log.log(Level.SEVERE, ">>> Save error: log event [{0}] ...", event);  
  
        JsonObject json = converter.toJsonObject(event);  
  
        IndexResponse response = client.prepareIndex(IndexDetails.FAILURE_INDEX, IndexDetails.TYPE)  
            .setSource(json.toString(), XContentType.JSON)  
            .get();  
  
        RestStatus status = response.status();  
  
        log.log(Level.SEVERE, ">>> Status [{0}] ...", status.getStatus());  
  
        return CompletableFuture.completedFuture(null);  
    }  
}
```

```
mstefank@Set-FI: ~/GIT/xstefank/microprofile-demo
xstefank@Set-FI ~/GIT/xstefank/microprofile-demo (master*) $ curl -X POST -H "Content-Type: application/json" -H "JWT AUTH localhost:7080/profiling/api -d "$(squash)" -i
HTTP/1.1 202 Accepted
Connection: keep-alive
Content-Type: application/json
Content-Length: 187
Date: Thu, 24 Jan 2019 17:00:54 GMT

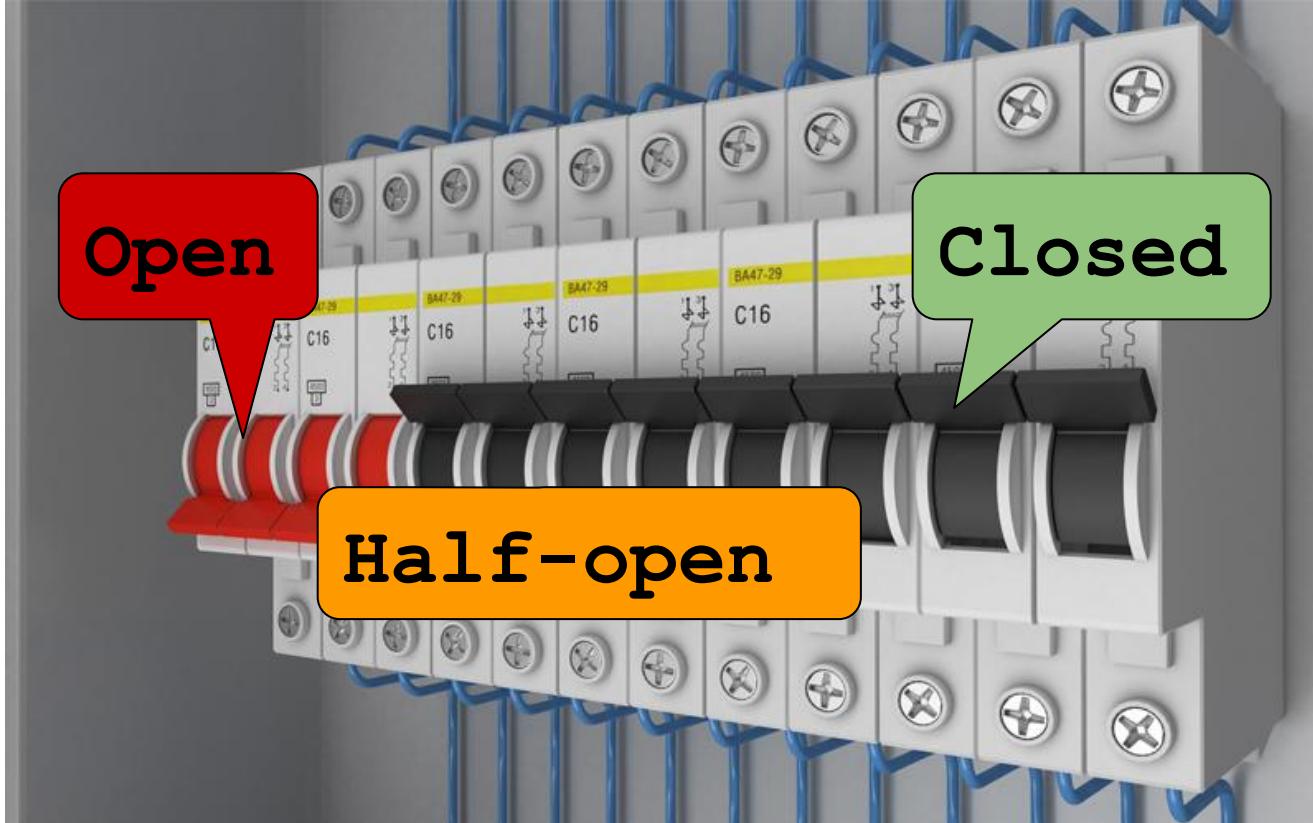
{"userId":1,"timeOccurred":1548423647000,"timeReceived":1548423647000,"eventName":"Squash","durationInMinutes":51,"location":"Knysna","partnerName":"Jawbone","metadata":{"calories":367}}]
xstefank@Set-FI ~/GIT/xstefank/microprofile-demo (master*) $
```

```
mstefank@Set-FI: ~/GIT/xstefank/microprofile-demo
xstefank@Set-FI ~/GIT/xstefank/microprofile-demo (master*) $ activateBadCode
Bad code active in membership service
xstefank@Set-FI ~/GIT/xstefank/microprofile-demo (master*) $
```

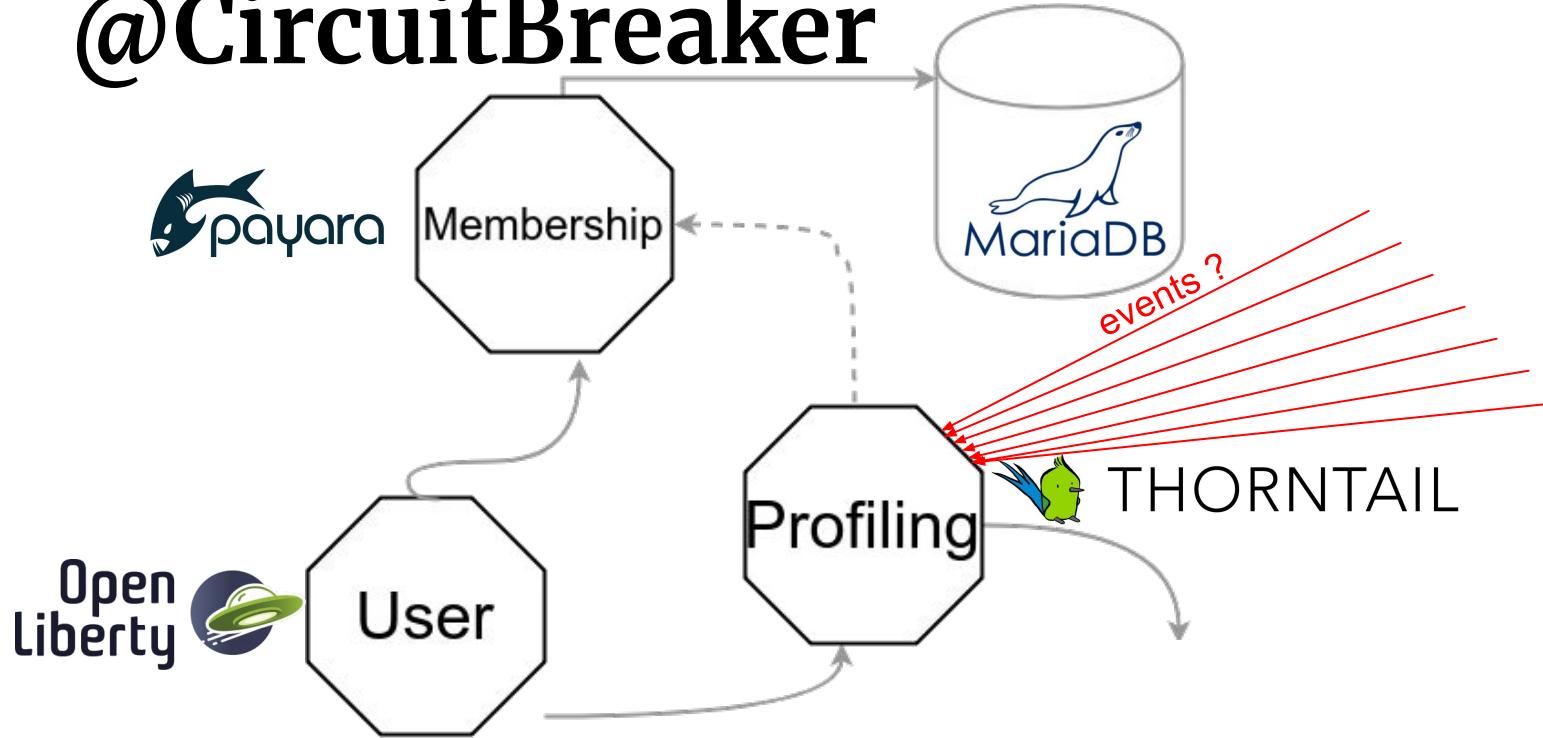
  

```
mvn-or-mvnw clean install -Prun
factory is initialized with configuration sources: com.netflix.config.ConcurrentCompositeConfiguration@3161e423
2019-01-24 18:00:54,180 INFO [io.smallrye.metrics.MetricsRegistryImpl] (EE-ManagedThreadFactory-default-Thread-1) Register metric [name: ft.com.github.phillipkruger.profiling.repository.EventLogger.logEvent.invocations.total, type: counter]
2019-01-24 18:00:54,209 SEVERE [com.github.phillipkruger.profiling.repository.EventLogger] (EE-ManagedThreadFactory-default-Thread-2) >>> Now (trying to )log event [UserEvent(userId=1, timeOccurred=Fri Jan 25 14:40:47 CET 2019, timeReceived=Fri Jan 25 14:40:47 CET 2019, eventName=Squash, durationInMinutes=51, location=Knysna, partnerName=Jawbone, metadata=(calories=367))] ...
2019-01-24 18:01:04,995 INFO [io.smallrye.metrics.MetricsRegistryImpl] (EE-ManagedThreadFactory-default-Thread-1) Register metric [name: ft.com.github.phillipkruger.profiling.repository.EventLogger.logEvent.retry.retries.total, type: counter]
2019-01-24 18:01:04,998 SEVERE [com.github.phillipkruger.profiling.repository.EventLogger] (EE-ManagedThreadFactory-default-Thread-3) >>> Now (trying to )log event [UserEvent(userId=1, timeOccurred=Fri Jan 25 14:40:47 CET 2019, timeReceived=Fri Jan 25 14:40:47 CET 2019, eventName=Squash, durationInMinutes=51, location=Knysna, partnerName=Jawbone, metadata=(calories=367))] ...
2019-01-24 18:01:18,135 SEVERE [com.github.phillipkruger.profiling.repository.EventLogger] (EE-ManagedThreadFactory-default-Thread-4) >>> Now (trying to )log event [UserEvent(userId=1, timeOccurred=Fri Jan 25 14:40:47 CET 2019, timeReceived=Fri Jan 25 14:40:47 CET 2019, eventName=Squash, durationInMinutes=51, location=Knysna, partnerName=Jawbone, metadata=(calories=367))] ...
2019-01-24 18:01:31,286 SEVERE [com.github.phillipkruger.profiling.repository.EventLogger] (EE-ManagedThreadFactory-default-Thread-5) >>> Now (trying to )log event [UserEvent(userId=1, timeOccurred=Fri Jan 25 14:40:47 CET 2019, timeReceived=Fri Jan 25 14:40:47 CET 2019, eventName=Squash, durationInMinutes=51, location=Knysna, partnerName=Jawbone, metadata=(calories=367))] ...
```

# @CircuitBreaker



# @CircuitBreaker



- Fail fast
- Fail pre-emptive

# @CircuitBreaker

```
@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)),
    @APIResponse(responseCode = "503", description = "Proplem with a connection to a downstream service",
                 headers = @Header(name = REASON))
})
@SecurityRequirement(name = "Authorization")
@RolesAllowed({"admin", "user"})
@Traced(operationName = "GetUserEvents", value = true)
@circuitbreaker(failOn = NoNodeAvailableException.class, requestVolumeThreshold = 1,
               failureRatio=1, delay = 10, delayUnit = ChronoUnit.SECONDS )
public Response getUserEvents() {
    ...
}
```

```
while true; do curl -H $JWT AUTH -i; sleep 1; echo ; done
```

```
itness", "timeOccurred": "2019-01-24T20:40:32+01", "timeReceived": "2019-01-24T20:40:32+01", "durationInMinutes": 61, "metaData": {"className": "Crossfit"}, {"userId": 1, "eventName": "Gym", "location": "Umhlanga Rocks", "partnerName": "Planet Fitness", "timeOccurred": "2019-01-24T20:37:08+01", "timeReceived": "2019-01-24T20:37:08+01", "durationInMinutes": 62, "metaData": {"className": "Crossfit"}, {"userId": 1, "eventName": "Cycling", "location": "Modimolle", "partnerName": "Fitbit", "timeOccurred": "2019-01-24T20:37:08+01", "timeReceived": "2019-01-24T20:37:08+01", "durationInMinutes": 86, "metaData": {"calories": "352"}}}
```

```
=====
```

```
HTTP/1.1 200 OK  
Connection: keep-alive  
x-number-of-hits: 6  
Content-Type: application/json  
Content-Length: 1316  
x-time-took-ms: 1  
Date: Thu, 24 Jan 2019 19:55:30 GMT
```

```
[{"userId": 1, "eventName": "Table tennis", "location": "Port Elizabeth", "partnerName": "Garmin", "timeOccurred": "2019-01-24T20:55:18+01", "timeReceived": "2019-01-24T20:55:18+01", "durationInMinutes": 32, "metaData": {"calories": "424"}}, {"userId": 1, "eventName": "Gym", "location": "Richards Bay", "partnerName": "Virgin Active", "timeOccurred": "2019-01-24T20:55:17+01", "timeReceived": "2019-01-24T20:55:17+01", "durationInMinutes": 87, "metaData": {"className": "Aerobics"}, {"userId": 1, "eventName": "Walk", "location": "Tembisa", "partnerName": "Polar", "timeOccurred": "2019-01-24T20:40:33+01", "timeReceived": "2019-01-24T20:40:33+01", "durationInMinutes": 42, "metaData": {"calories": "649"}}, {"userId": 1, "eventName": "Gym", "location": "Paarl", "partnerName": "Planet Fitness", "timeOccurred": "2019-01-24T20:40:32+01", "timeReceived": "2019-01-24T20:40:32+01", "durationInMinutes": 61, "metaData": {"className": "Crossfit"}, {"userId": 1, "eventName": "Gym", "location": "Umhlanga Rocks", "partnerName": "Planet Fitness", "timeOccurred": "2019-01-24T20:37:08+01", "timeReceived": "2019-01-24T20:37:08+01", "durationInMinutes": 62, "metaData": {"className": "Crossfit"}, {"userId": 1, "eventName": "Cycling", "location": "Modimolle", "partnerName": "Fitbit", "timeOccurred": "2019-01-24T20:37:08+01", "timeReceived": "2019-01-24T20:37:08+01", "durationInMinutes": 86, "metaData": {"calories": "352"}]}
```

```
=====
```

```
HTTP/1.1 200 OK  
Connection: keep-alive  
x-number-of-hits: 6  
Content-Type: application/json  
Content-Length: 1316  
x-time-took-ms: 1  
Date: Thu, 24 Jan 2019 19:55:31 GMT
```

```
[{"userId": 1, "eventName": "Table tennis", "location": "Port Elizabeth", "partnerName": "Garmin", "timeOccurred": "2019-01-24T20:55:18+01", "timeReceived": "2019-01-24T20:55:18+01", "durationInMinutes": 32, "metaData": {"calories": "424"}}, {"userId": 1, "eventName": "Gym", "location": "Richards Bay", "partnerName": "Virgin Active", "timeOccurred": "2019-01-24T20:55:17+01", "timeReceived": "2019-01-24T20:55:17+01", "durationInMinutes": 87, "metaData": {"className": "Aerobics"}, {"userId": 1, "eventName": "Walk", "location": "Tembisa", "partnerName": "Polar", "timeOccurred": "2019-01-24T20:40:33+01", "timeReceived": "2019-01-24T20:40:33+01", "durationInMinutes": 42, "metaData": {"calories": "649"}}, {"userId": 1, "eventName": "Gym", "location": "Paarl", "partnerName": "Planet Fitness", "timeOccurred": "2019-01-24T20:40:32+01", "timeReceived": "2019-01-24T20:40:32+01", "durationInMinutes": 61, "metaData": {"className": "Crossfit"}, {"userId": 1, "eventName": "Gym", "location": "Umhlanga Rocks", "partnerName": "Planet Fitness", "timeOccurred": "2019-01-24T20:37:08+01", "timeReceived": "2019-01-24T20:37:08+01", "durationInMinutes": 62, "metaData": {"className": "Crossfit"}, {"userId": 1, "eventName": "Cycling", "location": "Modimolle", "partnerName": "Fitbit", "timeOccurred": "2019-01-24T20:37:08+01", "timeReceived": "2019-01-24T20:37:08+01", "durationInMinutes": 86, "metaData": {"calories": "352"}]}]
```

```
mstefank@Set-FI: ~/GIT/xstefank/microprofile-demo
```

```
mstefank@Set-FI: ~/GIT/xstefank/microprofile-demo (master) $ █
```

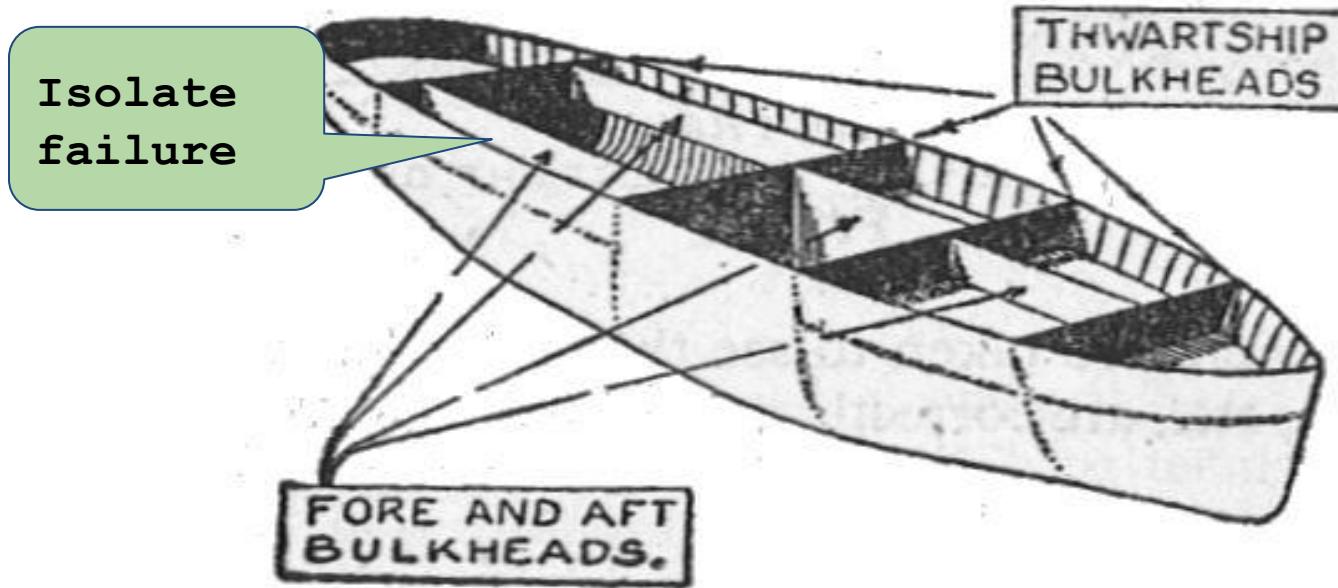
```
mvn-or-mvnw clean install -Prun
```

```
c [name: User events requests, type: counter]  
2019-01-24 20:55:27,400 INFO [io.smallrye.metrics.MetricsRegistryImpl] (default task-1) Register metric  
c [name: User events created, type: counter]  
2019-01-24 20:55:27,437 INFO [io.smallrye.metrics.MetricsRegistryImpl] (default task-1) Register metric  
c [name: ft.com.github.phillipkruger.profiling.ProfileService.getUserEvents.invocations.total, type: counter]  
2019-01-24 20:55:27,456 INFO [com.netflix.config.DynamicPropertyFactory] (default task-1) DynamicPropertyFactory is initialized with configuration sources: com.netflix.config.ConcurrentCompositeConfiguration@6d64caa6  
2019-01-24 20:55:27,887 INFO [io.smallrye.metrics.MetricsRegistryImpl] (default task-1) Register metric  
c [name: ft.com.github.phillipkruger.profiling.repository.EventSearcher.search.invocations.total, type: counter]  
2019-01-24 20:55:27,989 INFO [io.smallrye.metrics.MetricsRegistryImpl] (default task-1) Register metric  
c [name: ft.com.github.phillipkruger.profiling.repository.EventSearcher.search.timeout.callsNotTimedOut.total, type: counter]  
2019-01-24 20:55:27,989 INFO [io.smallrye.metrics.MetricsRegistryImpl] (default task-1) Register metric  
c [name: ft.com.github.phillipkruger.profiling.repository.EventSearcher.search.circuitbreaker.callsSucceeded.total, type: counter]  
2019-01-24 20:55:27,991 INFO [io.smallrye.metrics.MetricsRegistryImpl] (default task-1) Register metric  
c [name: ft.com.github.phillipkruger.profiling.repository.EventSearcher.search.timeout.executionDuration, type: histogram]  
2019-01-24 20:55:27,993 INFO [io.smallrye.metrics.MetricsRegistryImpl] (default task-1) Register metric  
c [name: ft.com.github.phillipkruger.profiling.ProfileService.getUserEvents.circuitbreaker.callsSucceeded.total, type: counter]
```

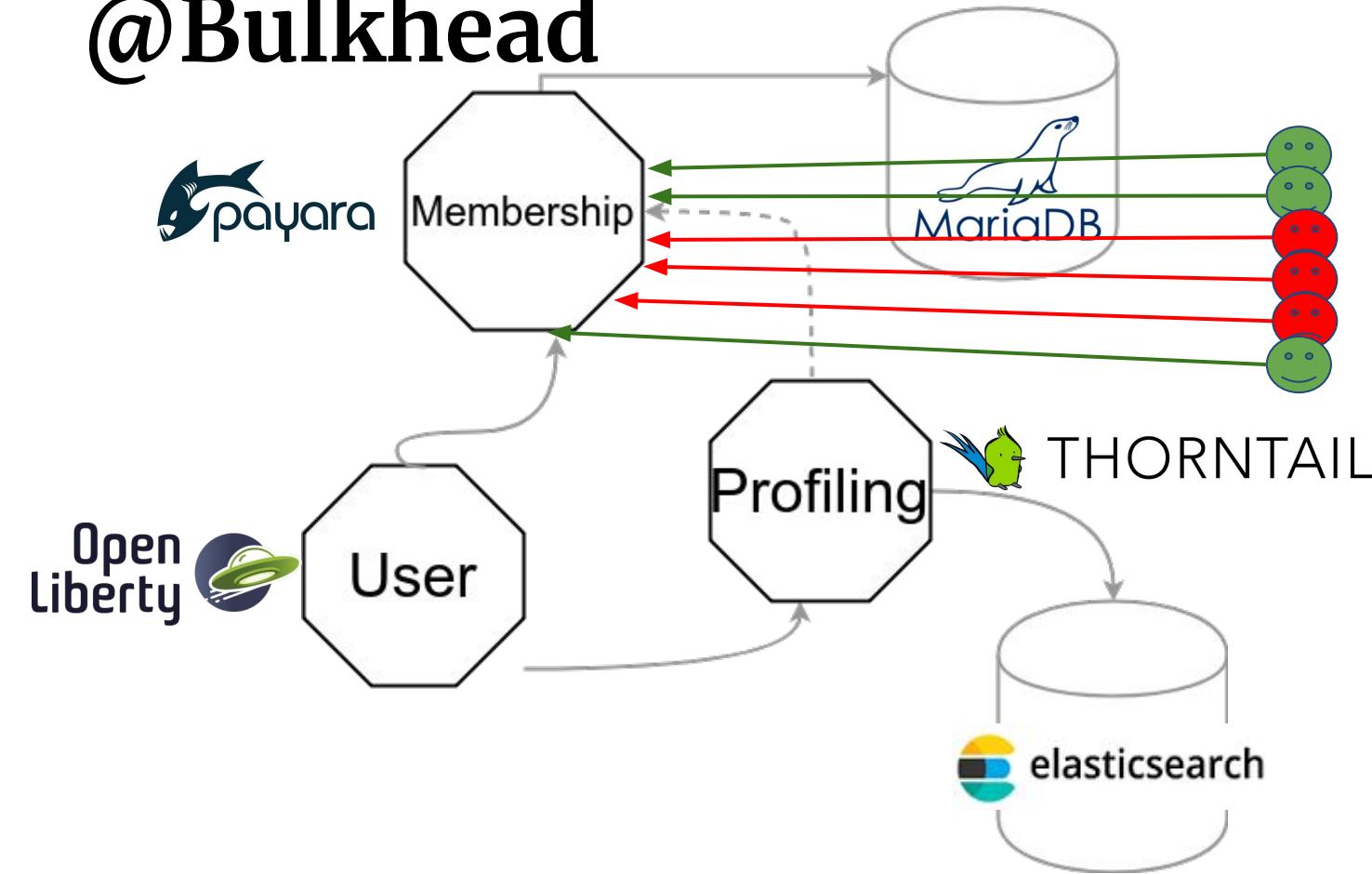
#openhousererdhat #microprofile

@xstefank @RedHat

# @Bulkhead



# @Bulkhead



# @Bulkhead

```
@GET  
@Path("{id}")  
@Counted(name = "Membership requests",absolute = true,monotonic = true)  
@Operation(description = "Get a certain Membership by id")  
@APIResponses({  
    @APIResponse(responseCode = "200", description = "Successful, returning membership",  
        content = @Content(mediaType = MediaType.APPLICATION_JSON,  
            schema = @Schema(implementation = Membership.class))),  
    @APIResponse(responseCode = "504", description = "Service timed out"),  
    @APIResponse(responseCode = "401", description = "User not authorized")  
)  
@SecurityRequirement(name = "Authorization")  
@RolesAllowed({"admin","user"})  
@Traced(operationName = "GetMembershipById", value = true)  
@CircuitBreaker(failOn = RuntimeException.class,requestVolumeThreshold = 1, failureRatio=1,  
    delay = 10, delayUnit = ChronoUnit.SECONDS )  
@Timeout(value = 3 , unit = ChronoUnit.SECONDS)  
@Bulkhead(2)  
public Membership getMembership(@NotNull @PathParam(value = "id") int id) {  
    ...  
}
```

MicroProfile demo

Membership

User service

Login request

Headers

Set token var

Get Membership

Headers

View Results Tree

### Thread Group

Name: Membership

Comments:

Action to be taken after a Sampler error

Continue  Start Next Thread Loop  Stop Thread  Stop Test  Stop Test Now

#### Thread Properties

Number of Threads (users): 100

Ramp-Up Period (in seconds): 0

Loop Count:  Forever 1

Delay Thread creation until needed

Scheduler

Scheduler Configuration

Duration (seconds)

Startup delay (seconds)

JMeter

# Eclipse MicroProfile 2.2 (Feb 2019)



= New

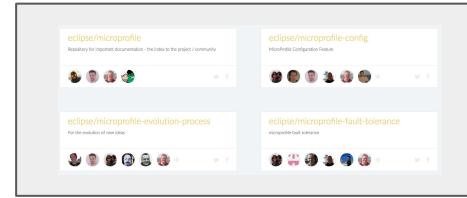
= Updated

= No change from last release (MicroProfile 2.1)

# MicroProfile - get involved!



[Google Groups](#)



[MicroProfile Projects](#)



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



[YouTube Channel](#)



[Video Hangouts](#)

# Known implementations



THORNTAIL



Apache TomEE



WildFly



Hammock  
CDI Based MicroServices



WebSphere  
Liberty



SMALLRYE



117

# Eclipse MicroProfile 2.2 (Feb 2019)



```
<dependency>
    <groupId>org.eclipse.microprofile</groupId>
    <artifactId>microprofile</artifactId>
    <version>${microProfile.version}</version>
    <type>pom</type>
</dependency>
```

↳ [Dependencies](#)

— NO Change from last release (MicroProfile 2.1)

6

[HTTPS://MICROPROFILE.IO/](https://microprofile.io/) | [HTTPS://PROJECTS.ECLIPSE.ORG/PROJECTS/TECHNOLOGY.MICROPROFILE](https://projects.eclipse.org/projects/technology.microprofile)

#openhousedredhat #microprofile

@xstefank @RedHat

# Thank you

-  @xstefank
-  xstefank
- mstefank@redhat.com

Code

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

Slides

<http://bit.ly/openhouse-mp>

# Resources

- <https://docs.google.com/presentation/d/1BYfVqnBIffh-QDlIrPyromwc9YSwIbsawGUECSsrSQBo/edit>
- [https://docs.google.com/presentation/d/1hvAWk8\\_7Mi52XemnF5IFqUDYCX3KwBrCfxqABWQ\\_PeY/edit](https://docs.google.com/presentation/d/1hvAWk8_7Mi52XemnF5IFqUDYCX3KwBrCfxqABWQ_PeY/edit)
- <https://github.com/microprofile-extensions/config-ext>
- <https://logz.io/wp-content/uploads/2017/03/prometheus-benchmark-dashboard.png>
- <https://docs.google.com/presentation/d/1BYfVqnBIffh-QDlIrPyromwc9YSwIbsawGUECSsrSQBo/edit>

Big thanks to [Phillip Krüger](#)  
for the original version of this presentation