Eclipse MicroProfile

22. Juni 2018

Eclipse MicroProfile Community

Microservices

- Small lightweight services
- Functional section from the entire application

Microservices

Pros

- Independently deployable
- Scalability
- Safety
- Use of optimal technologies

Microservices

	Developer Anarchy	Netflix	Self-contained System
Größe eines Microservice	10 bis 100 Zeilen	Von einem Team gut zu bewältigen	1 SCS = 1…n Microservices
Integration	Asynchrone Kommunikation/ Messaging	REST	Bevorzugt Webintegration, ansonsten asynchrone Kommunikation
Programmiersprachen	Beliebig	Beliebig, aber Fokus auf Java	Beliebig

What is Eclipse MicroProfile?

- Eclipse MicroProfile is an open-source community specification for Enterprise Java microservices
- Optimizes Enterprise Java for a microservices architecture and delivers application portability
- 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

Get Started

Choose your own adventure

- Payara Micro
- TomEE
- WildFly Swarm
- Open Liberty
- KumuluzEE
- Hammock

Get Started

Red Hat Wildfly Swarm

WildFly Swarm offers an innovative approach to packaging and running Java EE applications by packaging them with just enough of the server runtime to "java -jar" your application. It's MicroProfile compatible, too. And, it's all much, much cooler than that …

-http://wildfly-swarm.io/-

MicroProfile 1.0 (Sep, 2016)

Context and Dependency Injection

- Module should not be dependent on submodules
- Dependence should be "injected" from the outside
- Dependency no concrete object instance (interface, class)
- Definition dependent on context

JSON-P

- Processing of JSON Messages
  - Create
  - Parse
  - Change
  - Search

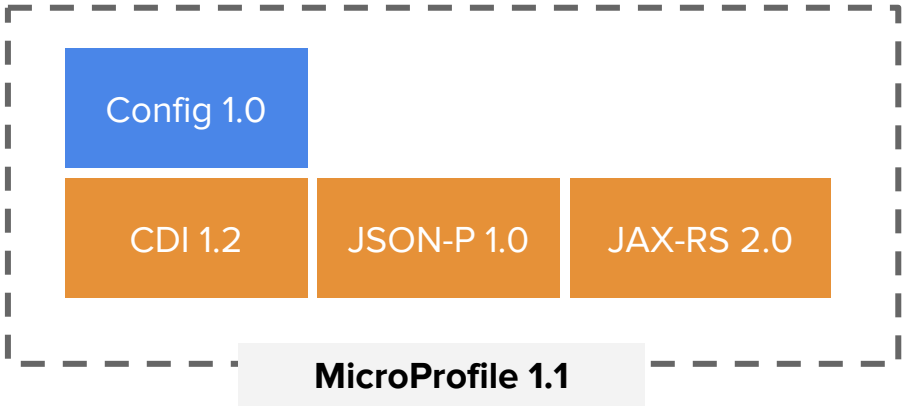
JAX-RS

- Definition of REST Interfaces
  - Set of Java interfaces and annotations

DEMO

DEMO CDI, JSON-P, JAX-RS

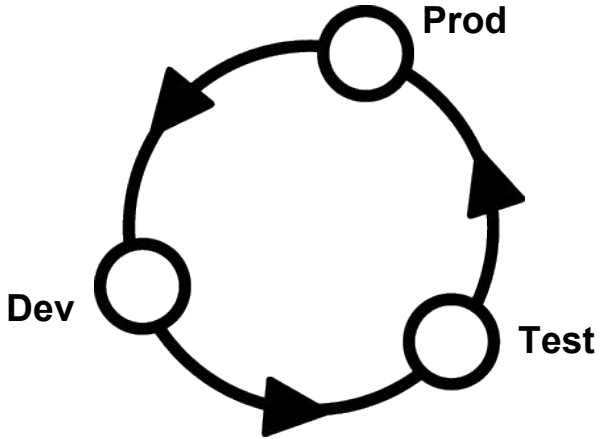
Eclipse MicroProfile 1.1 (Aug, 2017)



= 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

configuration from the outside of the project

```
com.acme.myproject.someserver.url = http://some.server/some/endpoint
com.acme.myproject.someserver.port = 9085
com.acme.myproject.someserver.active = true
com.acme.other.stuff.name = Karl
com.acme.myproject.notify.onerror = Karl@mycompany.com
some.library.own.config = some value
```

Configuration

configuration from the outside of the project

- Injecting config values at runtime
- Different config files possible
- Prioritization
- Use existing sources like system properties

Configuration

Config property over ConfigProvider

```
Config config = ConfigProvider.getConfig();
String serverUrl = config.getValue("com.acme.myproject.someserver.url", String.class);
```

Config property over injecting Config Object

```
@Inject
private Config config;
```

Config property over injecting configuration value

```
@Inject
@ConfigProperty(name="com.acme.myproject.someserver.url",
defaultValue="https://some-url.com")
private String someUrl;
```

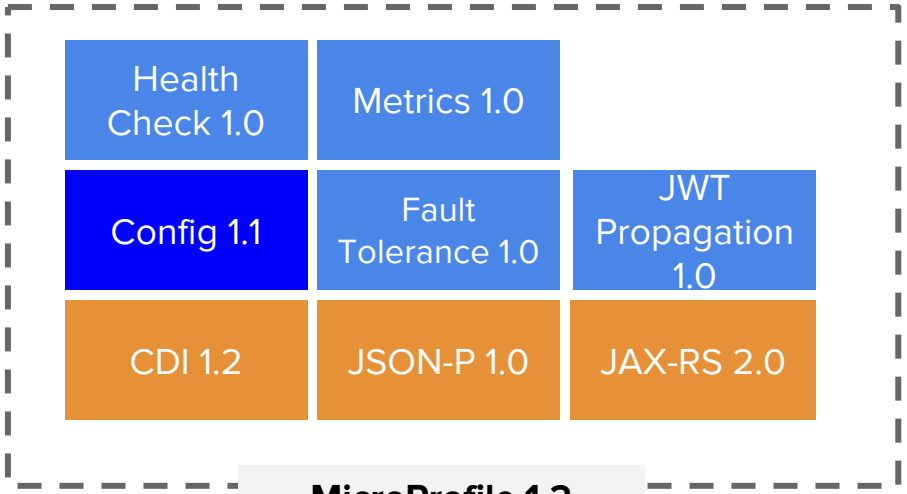
Eclipse MicroProfile 1.1 - release content

- Contexts and Dependency Injection (CDI 1.2)
- Java API for RESTful Web Services (JAX-RS 2.0)
- Java API for JSON Processing (JSON-P 1.0)

MicroProfile 1.0

- Eclipse MicroProfile 1.1 Specification PDF document
- Config 1.0:
  - Config 1.0 Specification PDF document
  - Javadocs
  - Test Compatibility Kit (TCK)

Eclipse MicroProfile 1.2 (Sep, 2017)



Health Check

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

- Runtime health check
- Creating and sending Health Report
- Better maintainability
- Faster reactions in case of a service failure
- Includable in existing infrastructure control systems

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

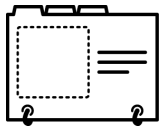
Health Check 1.0 - Goals

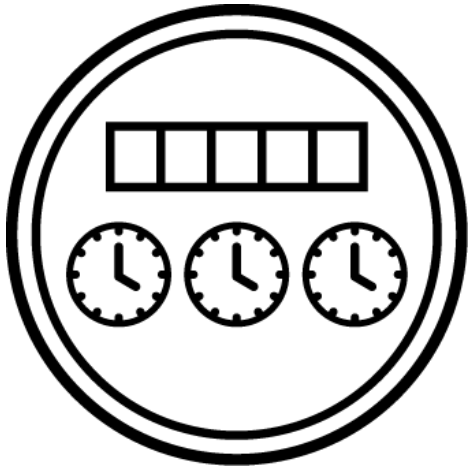
```
public class HealthCheckImpl implements HealthCheck{
        @Override
        public HealthCheckResponse call() {
            if(!System.getProperty("com.acme.myproject.someserver.name").equals("defaultServer")) {
                return HealthCheckResponse.named(SystemResource.class.getSimpleName())
                                .withData("default server", "not available")
                                .down().build();
            }
              return HealthCheckResponse.named(SystemResource.class.getSimpleName())
                        .withData("default server", "available").up().build();
        }
}
```

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

Metric Registry





Required Base metrics
Application metrics
Vendor-specific metrics

Metrics

```
@Timed(name = "PropertiesRequestTime", absolute = true, description = "Time needed to get the properties of a system from the given hostname")
public Properties get(String hostname) {
        Properties properties = SystemClient.getPropertiesForHostname(hostname);
        return properties;
}
```

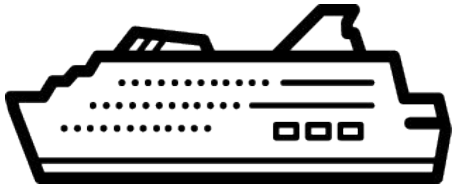
- @Timed
- @Counted
- @Gauge
- @Metered
- @Metric

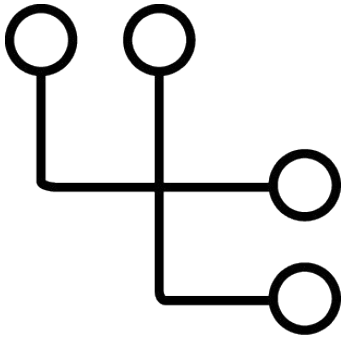
Metrics

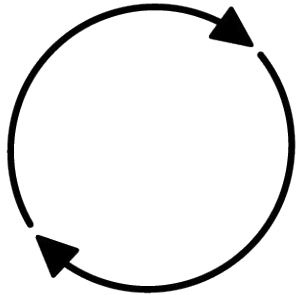
Demo Config, HealthCheck, Metrics

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 1.0 - Influence and goal

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

- Hystrix
- 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 1.0

- @Asynchronous
- @Retry
- @Fallback
- @CircuitBreaker
- @Bulkhead
- @Timeout

Fault Tolerance 1.0

Demo Fault Tolerance

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 1.0 - Influence and goals

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

- OAuth2
- OpenID Connect(OIDC), and
- JSON Web Tokens(JWT)

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

```
public class InventoryResource {

   @Inject
   InventoryManager manager;

   @GET
   @RolesAllowed({ "admin", "user" })
   @Produces(MediaType.APPLICATION_JSON)
   public Properties getProperties(@Context HttpHeaders httpHeaders) {

         String authHeader = httpHeaders.getRequestHeaders()
                                        .getFirst(HttpHeaders.AUTHORIZATION)
          return System.getProperties();
   }
}
```

Eclipse MicroProfile 1.2 - release content

- Eclipse MicroProfile 1.1
- Updated Config + Health Check + Metrics + Fault Tolerance + JWT Propag
- Eclipse MicroProfile 1.2 Specification PDF document

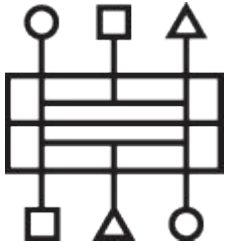
Config 1.1	Health Check 1.0	Metrics 1.0	Fault Tolerance 1.0	JWT Propagation 1.0
- Product page - Spec PDF doc - Spec HTML doc - Technology Compatibility Kit (TCK)	- Product page - Spec PDF doc - Javadocs - Technology Compatibility Kit (TCK)	- Product page - Spec PDF doc - Spec HTML doc - Technology Compatibility Kit (TCK)	- Product page - Spec PDF doc - Technology Compatibility Kit (TCK)	- Product page - Spec PDF doc - Javadocs - Technology Compatibility Kit (TCK)

Eclipse MicroProfile 1.3 (Q1 CY2018)

OpenAPI

MICROPROFILE™
OPTIMIZING ENTERPRISE JAVA

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

- 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

OpenAPI

@APIResponses = A container for multiple responses from an API operation. This annotation is optional, but it can be helpful to organize a method with multiple responses.

@APIResponse = Describes a single response from an API operation.

@Content = Provides a schema and examples for a particular media type.

@Schema = Defines the input and output data types.

@Operation = Describes a single API operation on a path.

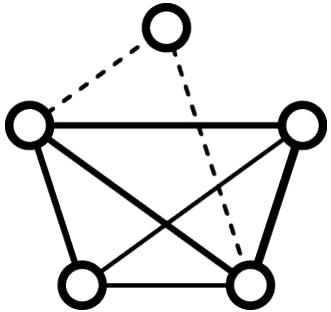
@Parameter = Describes a single operation parameter.

OpenAPI

Demo OpenAPI

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 1.0

- 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

OpenTracing 1.0

- Enable MicroProfile OpenTracing

```
<feature>mpOpenTracing-1.0</feature>
<feature>usr:opentracingZipkin-0.30</feature>
```

- Define explicit span creation

```
@Traced(value = true, operationName = "InventoryManager.list")
public InventoryList list() {
  return invList;
}
```

OpenTracing

1. Point to the **http://localhost:9081/inventory/systems** URL
2. Check your tracing server, and sort the traces by newest first
3. You see a new trace record

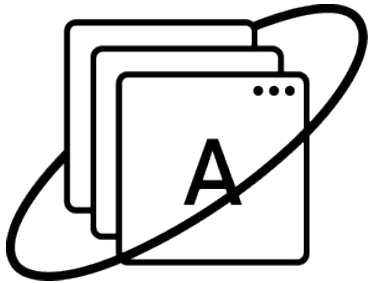
Span:

```
get:io.openliberty.guides.inventory.inventoryresource.listcontents

inventorymanger.list
```

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 with JAX-RS

```
@Path("/customers")
 @Produces("application/json")
public interface CustomerService {
   @GET
   @Path("/{customerId}")
   public Customer getCustomer(@PathParam( "customerId") String customerId);
                        …
 }
```

REST Client with JAX-RS

```
private static final String CUSTOMER_URI =
"http://localhost:9080/api/customers";

private Client client = ClientBuilder.newClient();

public Customer getCustomer(String customerId) {
    return client.request(MediaType.APPLICATION_JSON)
        .target(CUSTOMER URI)
        .path(customerId)
        .get(Customer.class);
```

Rest Client 1.0

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

Rest Client 1.0

```
private CustomerService customerService =
RestClientBuilder.newBuilder()
    .baseUrl(CUSTOMER URI)
    .build(CustomerService.class);

public Customer getCustomer(String customerId) {

    return customerService.getCustomer(customerId);

}
```

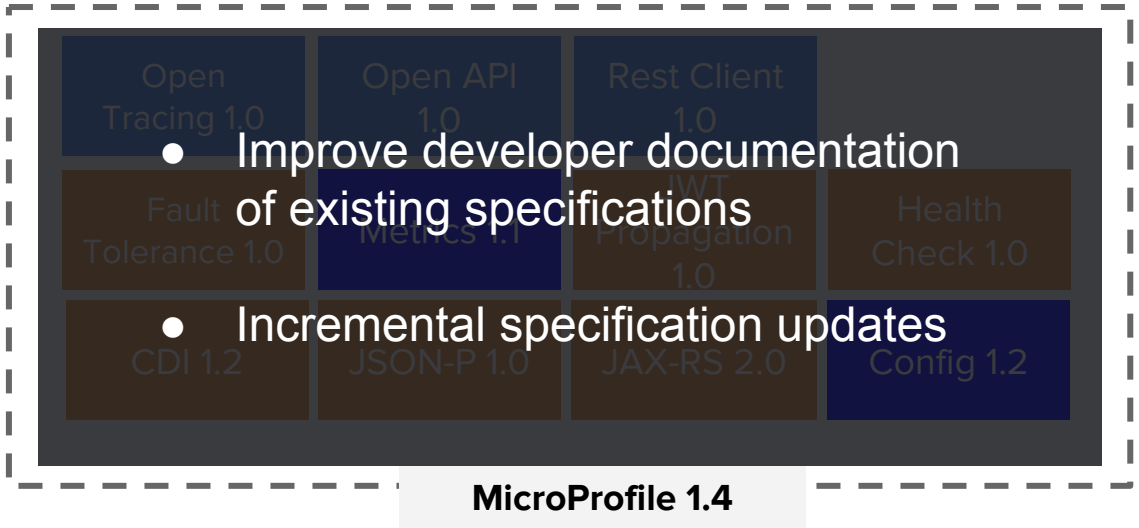
Eclipse MicroProfile 1.3 - release content

- Eclipse MicroProfile 1.2
- Updated Config & Metrics + Open Tracing + Open API + Rest Client
- Eclipse MicroProfile 1.3 Specification PDF document

Config 1.2	OpenTracing 1.0	Metrics 1.1	Open API 1.0	Rest Client 1.0
- Product page - Spec PDF doc - Spec HTML doc - Technology Compatibility Kit (TCK)	- Product page - Spec PDF doc - Spec HTML doc - Technology Compatibility Kit (TCK)	- Product page - Spec PDF doc - Spec HTML doc - Technology Compatibility Kit (TCK)	- Product page - Spec PDF doc - Spec HTML doc - Javadocs - Technology Compatibility Kit (TCK)	- Product page - Spec PDF doc - Spec HTML doc - Javadocs - Technology Compatibility Kit (TCK)

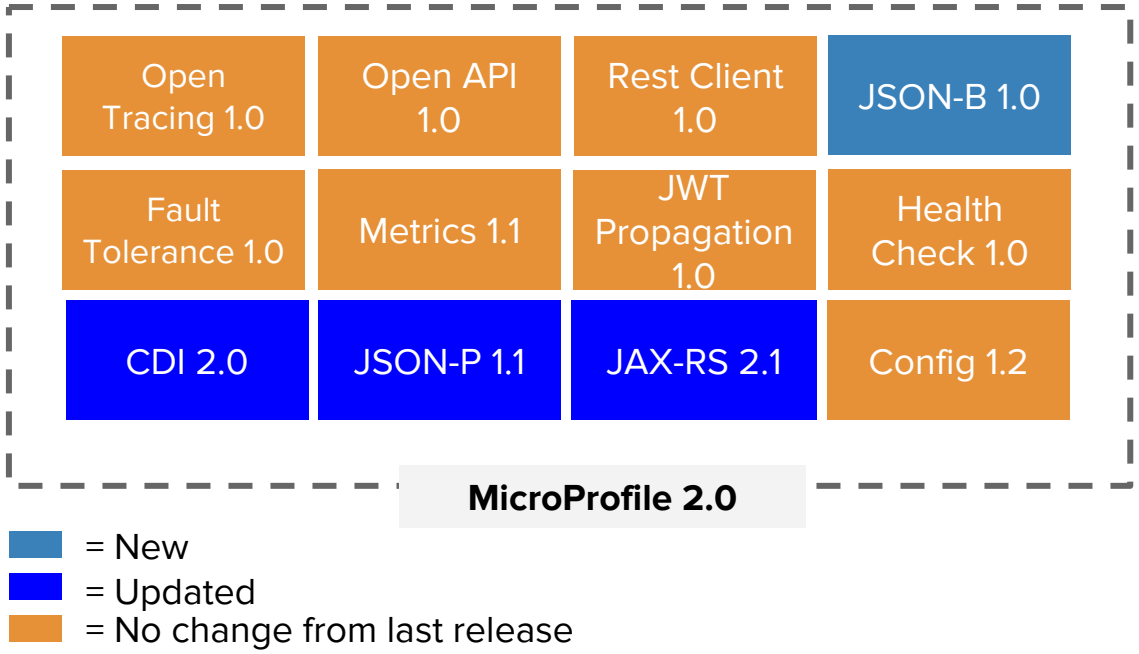
Eclipse MicroProfile 1.4 (Q2 CY2018?)

Roadmap



Eclipse MicroProfile 2.0 (1H 2018?)

Roadmap



JSON-B

```
public class Dog {
    public String name;
    public int age;
    public boolean bitable;
}

Dog dog = …;
dog. …
Jsonb jsonb = JsonbBuilder.create();
String result = jsonb.toJson(dog);
```

Ausgabe:

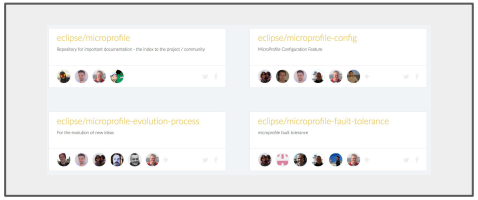
```
{
    "name":"Falco",
    "age":4,
    "bitable":false
}
```

Conclusion

- Don´t replace JavaEE, make it fit for the future instead
- Full community driven
- Independently from the manufacturer
- Well elaborated guides
- Ideal introduction to microservices

Get Involved!

Google Groups



MicroProfile Projects

Bi-Weekly & Quarterly General community Meetings

YouTube Channel

Video Hangouts

Quellen

openliberty.io
microprofile.io
heise.de
jaxenter.de
payara.fish


eclipse
Eclipse MicroProfile