MicroProfile pour les microservices (contexte Quarkus)

Emmanuel BRUNO
Université de Toulon, LIS UMR CNRS 7020
emmanuel.bruno@univ-tln.fr

2026-11-01

Table des matières

1 Pourquoi MicroProfile ? 1
2 Démarrage rapide (Quarkus + MicroProfile) 2
3 MicroProfile Config (configuration) 2
3.1 Exemple : injection de config 2
4 MicroProfile Health (liveness / readiness) 3
4.1 Health check simple (Quarkus / SmallRye) 3
5 MicroProfile Metrics 3
5.1 Exemple : `@Counted` et `@Timed` 3
6 MicroProfile Fault Tolerance 4
6.1 Exemple : `@Retry`, `@Timeout`, `@Fallback` 4
6.2 `@CircuitBreaker` 4
7 MicroProfile Rest Client 5
7.1 Exemple : interface client 5
8 MicroProfile OpenAPI 5
9 Security / JWT (MicroProfile JWT / SmallRye) 6
9.1 Dépendances Quarkus 6
9.2 Exemple : sécuriser une ressource 6
10 Observability et Production 7
11 Exemples pratiques (endpoints) 7
12 Débogage et développement 7
13 Bonnes pratiques 7
14 Ressources et liens 7

1 Pourquoi MicroProfile ?

MicroProfile est un ensemble de spécifications (SmallRye / Eclipse) qui complète Jakarta EE pour les architectures microservices : configuration externalisée, tolérance de panne, métriques, santé, documentation OpenAPI et sécurité légère.

Quarkus fournit d'excellentes intégrations MicroProfile (SmallRye implementations), optimisées pour le démarrage rapide et la compilation native.

Objectifs du cours :

- Comprendre les principaux APIs MicroProfile
- Voir des exemples concrets sous Quarkus
- Savoir configurer et tester en `quarkus:dev`

2 Démarrage rapide (Quarkus + MicroProfile)

1. Créer un projet Quarkus (Maven) :

```
mvn io.quarkus:quarkus-maven-plugin:3.0.0.Final:create \
  -DprojectGroupId=com.example \
  -DprojectArtifactId=mp-quickstart \
  -DclassName="com.example.GreetingResource" \
  -Dextensions="resteasy-reactive, smallrye-openapi, smallrye-health, smallrye-
metrics, smallrye-config, smallrye-fault-tolerance, smallrye-jwt, rest-client"
```

2. Lancer en mode développement :

```
mvn quarkus:dev
# hot-reload, accessible à http://localhost:8080
```

Note : adaptez la version Quarkus selon votre projet. Les extensions SmallRye apportent les implémentations MicroProfile.

3 MicroProfile Config (configuration)

MicroProfile Config permet d'injecter des valeurs depuis `application.properties`, variables d'environnement, ou autres sources.

3.1 Exemple : injection de config

```java
import org.eclipse.microprofile.config.inject.ConfigProperty;

@ApplicationScoped
public class GreetingService {
    @ConfigProperty(name = "greeting.message", defaultValue = "Hello")
    String message;

    public String greet(String name) {
        return message + ", " + name;
    }
}
```

application.properties :

```
greeting.message=Bonjour
```

Dans Quarkus, `application.properties` est la source préférée mais MP Config s'intègre aussi aux variables d'environnement.

4 MicroProfile Health (liveness / readiness)

MicroProfile Health expose des endpoints permettant d'évaluer la santé des services.

4.1 Health check simple (Quarkus / SmallRye)

```java
import org.eclipse.microprofile.health.HealthCheck;
import org.eclipse.microprofile.health.HealthCheckResponse;
import org.eclipse.microprofile.health.Liveness;

@Liveness
@ApplicationScoped
public class LivenessCheck implements HealthCheck {
    @Override
    public HealthCheckResponse call() {
        return HealthCheckResponse.up("application");
    }
}
```

Endpoints fournis par Quarkus (SmallRye Health) :

- GET /q/health/live (liveness)
- GET /q/health/ready (readiness)

Exemples avancés : vérifier accès base de données, file d'attente, connexions externes.

5 MicroProfile Metrics

Expose des métriques applicatives (Prometheus, Micrometer adapter possible).

5.1 Exemple : `@Counted` et `@Timed`

```java
import org.eclipse.microprofile.metrics.annotation.Counted;
import org.eclipse.microprofile.metrics.annotation.Timed;

@ApplicationScoped
public class OrderService {
```

```
    @Counted(name = "orders_created_count")
    @Timed(name = "orders_create_time")
    public void createOrder(String id) {
        // logique
    }
}
```

Quarkus expose les métriques à :

- GET /q/metrics (raw)
- GET /q/metrics/application (application metrics)

Ces métriques peuvent être récupérées par Prometheus.

6 MicroProfile Fault Tolerance

API pour gérer latences, échecs réseau, retries, timeouts, fallback et circuit breakers.

6.1 Exemple : `@Retry`, `@Timeout`, `@Fallback`

```
import org.eclipse.microprofile.faulttolerance.Retry;
import org.eclipse.microprofile.faulttolerance.Timeout;
import org.eclipse.microprofile.faulttolerance.Fallback;

@ApplicationScoped
public class RemoteService {

    @Retry(maxRetries = 3)
    @Timeout(2000)
    @Fallback(fallbackMethod = "fallbackResponse")
    public String callExternal() {
        // appel HTTP externe
    }

    public String fallbackResponse() {
        return "service-unavailable";
    }
}
```

6.2 `@CircuitBreaker`

```
import org.eclipse.microprofile.faulttolerance.CircuitBreaker;

@CircuitBreaker(requestVolumeThreshold = 5, failureRatio = 0.5)
public String callExternal() { ... }
```

Quarkus SmallRye Fault Tolerance implémente ces annotations et expose des métriques de tolérance.

7 MicroProfile Rest Client

Permet d'écrire des clients REST typés via une interface Java.

7.1 Exemple : interface client

```
import org.eclipse.microprofile.rest.client.inject.RegisterRestClient;
import javax.ws.rs.GET;
import javax.ws.rs.Path;

@RegisterRestClient(configKey = "inventory-api")
@Path("/inventory")
public interface InventoryClient {
    @GET
    List<Item> listItems();
}
```

Injection :

```
@Inject
@RestClient
InventoryClient inventoryClient;

public void check() {
    inventoryClient.listItems();
}
```

Configuration (application.properties) :

```
com.example.InventoryClient/mp-rest/url=http://inventory:8080
```

Quarkus supporte aussi la version reactive des Rest Clients.

8 MicroProfile OpenAPI

Génère automatiquement une documentation OpenAPI à partir des ressources JAX-RS et des annotations.

- Extension Quarkus : `smallrye-openapi`
- URI par défaut : `/q/openapi` (spec JSON) et `/q/swagger-ui` (interface)

Exemple minimal :

```java
@Path("/greet")
public class GreetingResource {
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String hello() { return "Hello"; }
}
```

OpenAPI sera exposé automatiquement ; on peut enrichir via `@Operation`, `@APIResponse`, etc.

9 Security / JWT (MicroProfile JWT / SmallRye)

MicroProfile JWT simplifie l'authentification basée sur tokens JWT.

9.1 Dépendances Quarkus

- `quarkus-smallrye-jwt`

9.2 Exemple : sécuriser une ressource

```java
import javax.annotation.security.RolesAllowed;
import org.eclipse.microprofile.jwt.JsonWebToken;

@Path("/secure")
public class SecureResource {

    @Inject
    JsonWebToken jwt;

    @GET
    @RolesAllowed("user")
    public String whoami() {
        return jwt.getName() + " - roles: " + jwt.getGroups();
    }
}
```

Configuration minimale (`application.properties`) :

```
mp.jwt.verify.publickey.location=META-INF/resources/publicKey.pem
quarkus.http.auth.permission.roles.paths=/secure/*
quarkus.http.auth.permission.roles.policy=roles
```

Quarkus fournit aussi des helpers pour (dev) générer des tokens via `SmallRye JWT` tooling.

10 Observability et Production

- Exposez metrics à Prometheus (`/q/metrics`)
- Exposez health checks (`/q/health`)
- Documentez API avec OpenAPI (`/q/openapi` + `/q/swagger-ui`)
- Utilisez Fault Tolerance pour circuit breaker & retries

11 Exemples pratiques (endpoints)

Un petit service demo :

- `GET /greet` → greeting message (MP Config)
- `GET /q/health/ready` → readiness
- `GET /q/metrics` → metrics
- `GET /q/openapi` → OpenAPI JSON
- `GET /secure` → resource protégée par JWT

12 Débogage et développement

- Mode dev : `mvn quarkus:dev` (hot reload)
- Activer logs MP :

```
quarkus.log.category."org.eclipse.microprofile".level=DEBUG
```

13 Bonnes pratiques

- Centralisez la configuration via MP Config
- Protégez les endpoints critiques avec JWT ou OAuth2
- Exposez métriques et health checks pour l'orchestrateur (Kubernetes)
- Testez Fault Tolerance via tests d'intégration (chaos)
- Documentez l'API avec OpenAPI et gardez-le synchronisé

14 Ressources et liens

- MicroProfile official : https://microprofile.io
- SmallRye (implémentations) : https://smallrye.io
- Quarkus guides : https://quarkus.io/guides
- Quarkus MicroProfile (OpenAPI, Health, Metrics, Fault Tolerance) guides