

Microservicios En 30 minutos o gratis

26 de Junio, 2021

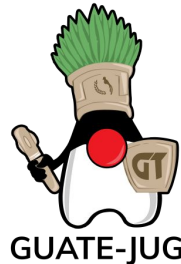


QUARKUS

¿Quién?



- JUG co-leader en GuateJUG
- Desarrollador Java certificado con +6 años de experiencia.
- Ganador del Duke Choice Award en 2016 con GuateJUG
- Parte del comité organizador del JConf Guatemala
- Conferencista desde 2012 en eventos nacionales e internacionales.
- Consultor en MangoChango S.A.



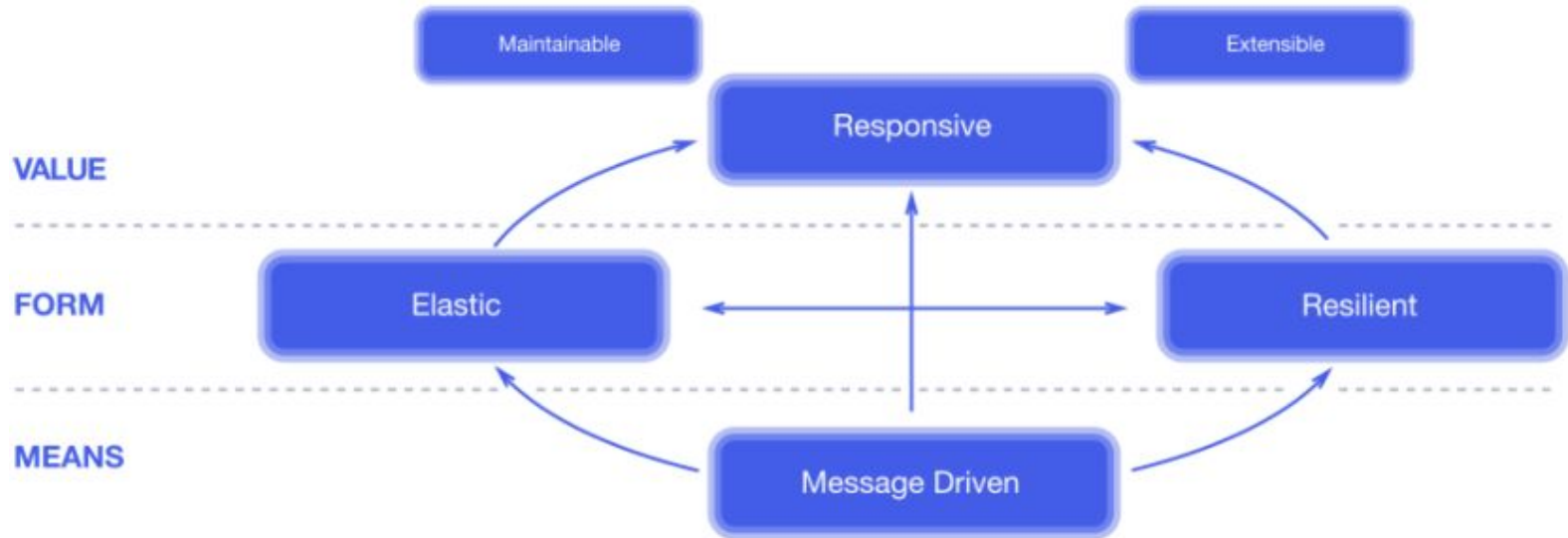
Contenido

- **JakartaEE & Eclipse MicroProfile**
- **Como empezar**
- **Quarkus**
- **Demo**

Java más vivo que nunca



Aplicaciones modernas



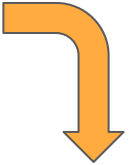
La comunidad de JavaEE



JAKARTA EE



JAVA EE
GUARDIANS



Eclipse MicroProfile



¿Porqué MicroProfile?



- Java se estaba quedando atrás en la nueva era de microservicios, por lo cual nacieron nuevos frameworks como vert.x, spring boot, entre otros, que ayudan al desarrollo de microservicios con Java.
- A pesar de esos esfuerzos, no existía una forma estandarizada de desarrollar microservicios, contrario a lo que diferencia el desarrollo de aplicaciones empresariales bajo los estándares de Java EE.
- Se buscaba una forma de reutilizar todo el conocimiento de JavaEE que ya tenemos y orientarlo a el desarrollo de microservicios.

JavaEE / JakartaEE

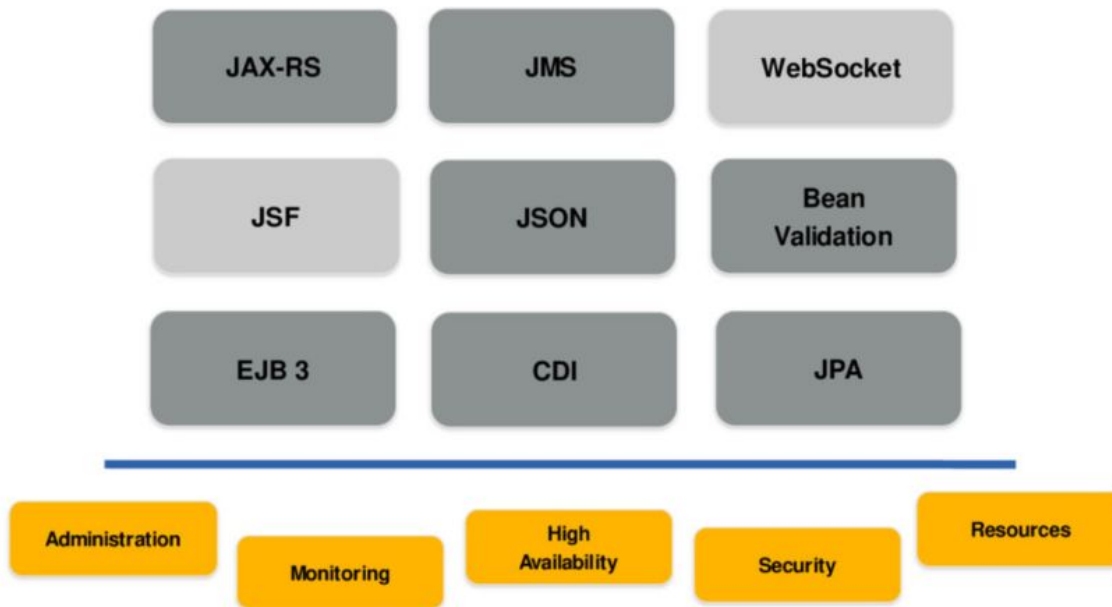
Java EE 8



Batch	Dependency Injection	JACC	JAXR	JSTL	Management
Bean Validation	Deployment	JASPIC	JMS	JTA	Servlet
CDI	EJB	JAX-RPC	JSF	JPA	Web Services
Common Annotations	EL	JAX-RS	JSON-P	JavaMail	Web Services Metadata
Concurrency EE	Interceptors	JAX-WS	JSP	Managed Beans	WebSocket
Connector	JSP Debugging	JAXB			
JSON-B	Security				

MicroProfile - API's

JavaEE / Jakarta EE no es suficiente



Imágenes: @reza_rahman

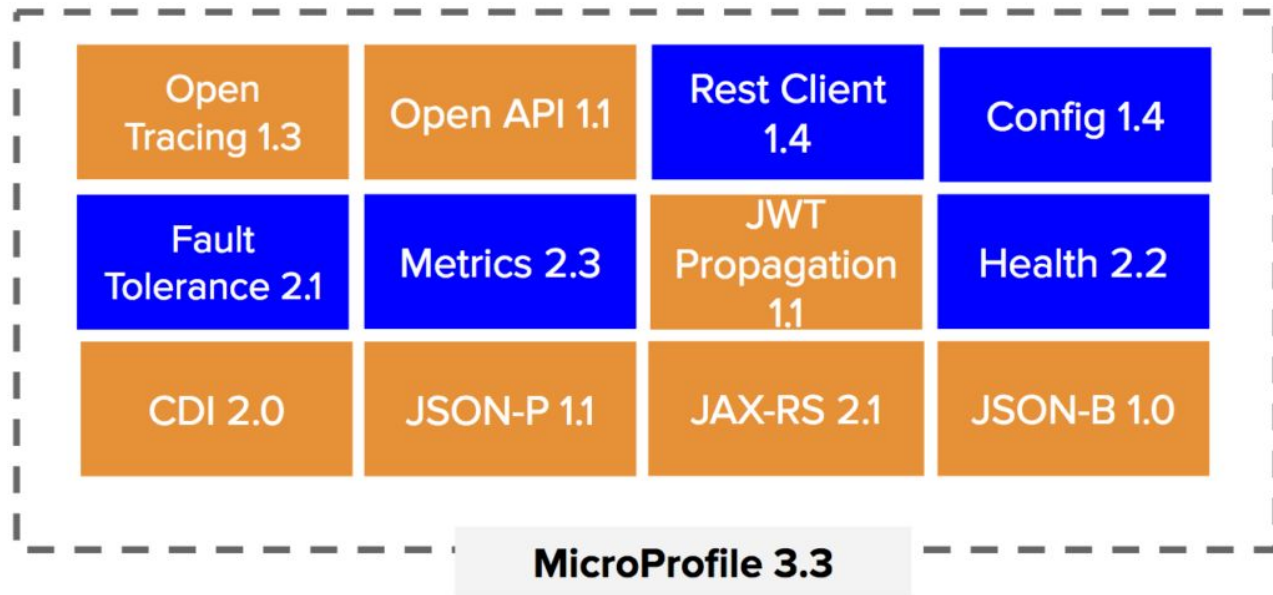
¿Quienes apoyan a MicroProfile?



THORNTAIL



MicroProfile - API's



= New

= Updated

= No change from last release (MicroProfile 3.2)

Tolerancia a Fallos

```
@GET
@Path("/{id}")
public Response findById(@PathParam("id") final String id) {
    //Logic and magic
    return Response.ok().build();
}
```

Tolerancia a Fallos

```
@GET
@Path("/{id}")
@Timeout(300)
public Response findById(@PathParam("id") final String id) {
    //Logic and magic
    return Response.ok().build();
}
```

Tolerancia a Fallos

```
@GET
@Path("/{id}")
@Timeout(300)
@Retry(maxRetries = 5, delay = 60)
public Response findById(@PathParam("id") final String id) {
    //Logic and magic
    return Response.ok().build();
}
```

Tolerancia a Fallos

```
@GET
@Path("/{id}")
@Timeout(300)
@Retry(maxRetries = 5, delay = 60)
@Fallback(fallbackMethod = "findByIdFallBack")
public Response findById(@PathParam("id") final String id) {
    //Logic and magic
    return Response.ok().build();
}

public Response findByIdFallBack(@PathParam("id") final String imdbId) {
    //more magic from cache here
    return Response.ok().build();
}
```


Metrics

```
@GET
@Path("/{id}")
@Timeout(300)
@Counted(name = "queries")
@Retry(maxRetries = 5, delay = 60)
@Fallback(fallbackMethod = "findByIdFallback")
public Response findById(@PathParam("id") final String id) {
    //Logic and magic
    return Response.ok().build();
}

public Response findByIdFallback(@PathParam("id") final String imdbId) {
    //more magic from cache here
    return Response.ok().build();
}
```

Metrics

```
@Inject
@Metric
Counter failures;

@GET
@Path("/{id}")
@Timeout(300)
@Counted(name = "queries")
@Retry(maxRetries = 5, delay = 60)
@Fallback(fallbackMethod = "findByIdFallback")
public Response findById(@PathParam("id") final String id) {
    //Logic and magic
    return Response.ok().build();
}

public Response findByIdFallback(@PathParam("id") final String imdbId) {
    //more magic from cache here
    failures.inc();
    return Response.ok().build();
}
```



QUARKUS

<https://github.com/cajasmota/slides/2021>

<https://github.com/cajasmota/quarkus-demo>

Microservicios

En 30 minutos o gratis

@cajasmota



QUARKUS