

Reactive messaging

Evento

```
package com.redhat.training;

public class UserWasCreated {
    public Long id;
    public String username;

    public UserWasCreated() {}

    public UserWasCreated(Long id, String username) {
        this.id = id;
```

Eclipse Microfile Reactive Messaging Spec 2.0

- Quarkus usa SmallRye Reactive Messaging que soporta Kafka, AMQP o MQTT.
- Conceptos:
 - Messages (org.eclipse.microprofile.reactive.messaging.Message)
 - Channels (Incoming channels [consumo], outgoing channels [producción])
 - Connectors (Connector Kafka, RabbitMQ, Pulsar, etc)

```

@Incoming("user-creation-channel") ①
public CompletionStage<Void> consumeUserWasCreatedEvents(
    Message<UserWasCreated> message
) { ②
    UserWasCreated event = message.getPayload(); ③

    if (event.profile == "business") {
        notifyMarketingAboutNewBusiness(event.id);
    }

    return message.ack(); ④
}

```

Consumo

```

@Incoming("user-creation-channel") ②
@ActivateRequestContext ③
public Uni<Void> processEvent(UserWasCreated event) { ④
    String assignedAccountRegion = calculateAccountRegion(event.country);

    return session.withTransaction( ⑤
        t -> User.<User>findById(event.id)
            .onItem()
            .ifNotNull()
            .invoke( ⑥
                entity -> entity.region = assignedAccountRegion
            ).replaceWithVoid() ⑦
        ).onTermination().call(() -> session.close()); ⑧
}

```

De Mensajes - Payload

```
@Outgoing("incidents-ratio-channel") ①
public Message<Double> notifyRatioofIncidents() { ②
    Uni<Long> high = Speaker.count("status", "HIGH"); ③
    Uni<Long> total = Speaker.count(); ④

    return Message.of( ⑤
        high.await().atMost(Duration.ofSeconds(1)).doubleValue() ⑥
        /
        total.await().atMost(Duration.ofSeconds(1)).doubleValue()
    );
}
```

Producir
Mensajes

```
@Outgoing("incidents-ratio-channel") ①
public Double notifyRatioofIncidents() { ②
    Uni<Long> high = Speaker.count("status", "HIGH");
    Uni<Long> total = Speaker.count();

    return high.await().atMost(Duration.ofSeconds(1)).doubleValue() ③
        /
        total.await().atMost(Duration.ofSeconds(1)).doubleValue();
}
```

Código imperativo

Emitter

```
package com.redhat.training;

import org.eclipse.microprofile.reactive.messaging.Channel;
import org.eclipse.microprofile.reactive.messaging.Emitter;

@ApplicationScoped
public class TemperatureResource {

    @Channel("temperatures-channel") ❶
    Emitter<Double> emitter; ❷

    // Some business logic

    public void sendTemperature(double measurement) {
        emitter.send(measurement); ❸
    }
}
```

```
package com.redhat.training;

import org.eclipse.microprofile.reactive.messaging.Channel;
import org.eclipse.microprofile.reactive.messaging.Emitter;

@ApplicationScoped
public class SomeImperativeLogic {

    @Channel("temperatures-channel") ❶
    Emitter<Double> emitter; ❷

    // Some business logic

    public void sendTemperature(double measurement) {
        emitter.send(Message.of(measurement)); ❸
    }
}
```

Kafka

quarkus-messaging-kafka

```
<dependency>
  <groupId>io.quarkus</groupId>
  <artifactId>quarkus-messaging-kafka</artifactId>
</dependency>
```

Serializers: io.quarkus.kafka.client.serialization.ObjectMapperSerializer

Deserializers: io.quarkus.kafka.client.serialization.ObjectMapperDeserializer

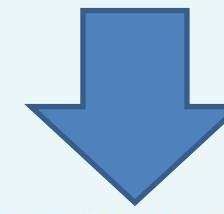
```
package com.redhat.training;

import io.quarkus.kafka.client.serialization.ObjectMapperDeserializer;

public class UserWasCreatedDeserializer extends
    ObjectMapperDeserializer<UserWasCreated> {
    public UserWasCreatedDeserializer() {
        super(UserWasCreated.class);
    }
}
```

Kafka

Configuration

```
kafka.bootstrap.servers = localhost:9092 ①  
  
# Incoming Channel  
mp.messaging.incoming.[channel-name].connector = smallrye-kafka ②  
mp.messaging.incoming.[channel-name].topic = a-topic ③  
mp.messaging.incoming.[channel-name].auto.offset.reset = earliest ④  
mp.messaging.incoming.[channel-name].value.deserializer =  
org.apache.kafka.common.serialization.DoubleDeserializer ⑤  
  
# Outgoing Channel  
mp.messaging.outgoing.[channel-name].connector = smallrye-kafka ⑥  
mp.messaging.outgoing.[channel-name].topic = another-topic ⑦  
mp.messaging.outgoing.[channel-name].value.serializer =  
io.quarkus.kafka.client.serialization.ObjectMapperSerializer ⑧
```

Recursos

<https://smallrye.io/smallrye-reactive-messaging/3.21.0/>

<https://quarkus.io/version/2.13/guides/kafka-reactive-getting-started>

<https://quarkus.io/version/2.13/guides/amqp>

<https://kafka.apache.org>