

Reactive messaging

Eclipse Microfile Reactive Messaging Spec 2.0

Quarkus usa SmallRye Reactive Messaging que soporta Kafka, AMQP o MQTT.

Messages (`org.eclipse.microprofile.reactive.messaging.Message`)

Channels (Incoming channels, outgoing channels)

Connectors (Connector Kafka, RabbitMQ, Pulsar, etc)

```

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

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

    return message.ack(); ❹
}

```

Consumo
De Mensajes
- Payload

```

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

    return session.withTransaction( ❺
        t -> User.<User>findById(event.id)
            .onItem()
            .ifNotNull()
            .invoke( ❻
                entity -> entity.region = assignedAccountRegion
            ).replaceWithVoid() ❼
    ).onTermination().call(() -> session.close()); ❽
}

```



```

@Outgoing("incidents-ratio-channel") ❶
public Message<Double> notifyRatioOfIncidents() { ❷
    Uni<Long> high = Speaker.count("status", "HIGH"); ❸
    Uni<Long> total = Speaker.count(); ❹

    return Message.of( ❺
        high.await().atMost(Duration.ofSeconds(1).doubleValue() ❻
        /
        total.await().atMost(Duration.ofSeconds(1).doubleValue()
    );
}

```

Producir

Mensajes

```

@Outgoing("incidents-ratio-channel") ❶
public Double notifyRatioOfIncidents() { ❷
    Uni<Long> high = Speaker.count("status", "HIGH");
    Uni<Long> total = Speaker.count();

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

```

Coding 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") 1
    Emitter<Double> emitter; 2

    // Some business logic

    public void sendTemperature(double measurement) {
        emitter.send(measurement); 3
    }
}
```

```
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") 1
    Emitter<Double> emitter; 2

    // Some business logic

    public void sendTemperature(double measurement) {
        emitter.send(Message.of(measurement)); 3
    }
}
```


Kafka

quarkus-smallrye-reactive-messaging-kafka

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 ❶

# Incoming Channel
mp.messaging.incoming.[channel-name].connector = smallrye-kafka ❷
mp.messaging.incoming.[channel-name].topic = a-topic ❸
mp.messaging.incoming.[channel-name].auto.offset.reset = earliest ❹
mp.messaging.incoming.[channel-name].value.deserializer =
    org.apache.kafka.common.serialization.DoubleDeserializer ❺

# Outgoing Channel
mp.messaging.outgoing.[channel-name].connector = smallrye-kafka ❻
mp.messaging.outgoing.[channel-name].topic = another-topic ❼
mp.messaging.outgoing.[channel-name].value.serializer =
    io.quarkus.kafka.client.serialization.ObjectMapperSerializer ❽
```

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>