

ORACLE

Event sourcing DB with MicroProfile Reactive Messaging

With Helidon over Oracle Advanced Queuing

Daniel Kec

Java developer

Oracle, Helidon team

May 27, 2021



Daniel Kec

Helidon developer at Oracle

 [@danielkec](https://twitter.com/danielkec)

 [@danielkec](https://github.com/danielkec)

 [@danielkec](https://www.linkedin.com/in/danielkec)

Agenda

- 1 Intro to Oracle Advanced Queueing
- 2 Intro to MicroProfile Messaging
- 3 Provisioning of Oracle DB in OCI
- 4 Setup example advanced queue
- 5 Create Helidon example project with reactive AQ connector
- 6 Flying Frank UI with Server Sent Events

Oracle Advanced Queueing

- Oracle DB build-in messaging
- Various API's for enqueueing and dequeuing with:
 - JMS 1.1
 - PL/SQL
 - ODP.NET
 - Python
 - C
 - Node.js
- Operational benefits of Oracle DB
 - high availability
 - scalability
 - security
 - reliability

Oracle Advanced Queueing

- Trigger as an event source
- Message transformations with SQL functions
- Rule based routing
- and a lot more ...

More info about Oracle AQ:

<https://www.oracle.com/database/technologies/advanced-queueing.html>

Oracle Advanced Queueing - Trigger enqueue example

```
CREATE TABLE FRANK.TEST (id NUMBER(5) PRIMARY KEY, val VARCHAR2(15) NOT NULL);
```

```
CREATE OR REPLACE TRIGGER ins_event_trigger
  BEFORE INSERT ON FRANK.TEST
  FOR EACH ROW
  DECLARE
    enqueue_options    DBMS_AQ.ENQUEUE_OPTIONS_T ;
    message_properties DBMS_AQ.MESSAGE_PROPERTIES_T ;
    message_handle     RAW(16);
    msg                SYS.AQ$_JMS_TEXT_MESSAGE ;
  BEGIN
    msg := SYS.AQ$_JMS_TEXT_MESSAGE.construct ;
    msg.set_int_property('tab_id', :new.id);
    msg.set_text(:new.val);
    DBMS_AQ.ENQUEUE(
      queue_name => 'FRANK.EXAMPLE_QUEUE_1',
      enqueue_options => enqueue_options,
      message_properties => message_properties,
      payload => msg,
      msgid => message_handle);
  END;
```

```
/
```

MicroProfile Reactive Messaging

- Convenient way to receive or send messages
- Support backpressure
- Build in acknowledge abstraction mechanism
- Designed as reactive pipeline makes it easy to connect to reactive API's
- Connector API makes it easy to create custom connectors

MicroProfile Reactive Messaging

- Consuming

```
@Incoming("from-aq")
public void receiveMethod(String msg) {
    System.out.println("Dequeued from AQ: " + msg);
}
```


MicroProfile Reactive Messaging

- Producing

```
@Outgoing("to-aq")
public PublisherBuilder<String> produceMessages() {
    return ReactiveStreams.of("message 1", "message 2", "message 3", "message 4");
}
```

```
@Outgoing("to-aq")
public Publisher<String> produceMessages() {
    return FlowAdapters.toPublisher(
        Multi.just("message 1", "message 2", "message 3", "message 4")
    );
}
```

MicroProfile Reactive Messaging

- Producing with third-party reactive operator implementations

```
@Outgoing("to-aq")
public Publisher<String> produceMessages() {
    // RxJava
    return Flowable.just("1", "2", "3", "4", "5");
}
```

```
@Outgoing("to-aq")
public Publisher<String> produceMessages() {
    // Reactor
    return Flux.just("1", "2", "3", "4", "5");
}
```

MicroProfile Reactive Messaging

- Producing from imperative code

```
private SubmissionPublisher<String> emitter = new SubmissionPublisher<>();

public int sendMessage(String message) {
    return emitter.submit(message);
}

@Outgoing("to-aq")
public PublisherBuilder<String> registerEmitterMethod() {
    return ReactiveStreams.fromPublisher(FlowAdapters.toPublisher(emitter));
}
```

Find more about MicroProfile Reactive Messaging

- <https://download.eclipse.org/microprofile/microprofile-reactive-messaging-1.0/microprofile-reactive-messaging-spec.html>
- <https://medium.com/helidon/reactive-messaging-with-helidon-2-0-f5de1ca5dc63>


DEMO

Provision Oracle DB and send some messages

Working demo project available on GitHub:

<https://github.com/danielkec/helidon-aq-demo>

Questions?

- daniel.kec@oracle.com
-  DM @danielkec
- <https://stackoverflow.com/questions/tagged/helidon>

