Bringing Reactive to Enterprise Java Developers

Julien Ponge, Thomas Segismont and Clement Escoffier
Red Hat
Hello!

We work on Eclipse Vert.x, a toolkit for writing asynchronous and reactive applications on the JVM.
Vert.x in Action

Julien Ponge

MEAP

(shameless plug)

50% off with m1pong3 code
<table>
<thead>
<tr>
<th>Rank</th>
<th>Framework</th>
<th>Performance (higher is better)</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>vertex-postgres</td>
<td>43,245</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>vertex-web-postgres</td>
<td>39,219</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>http4k</td>
<td>27,860</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>actix-raw</td>
<td>27,842</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>http4k-apache</td>
<td>27,662</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>light-4j</td>
<td>26,985</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>kelp</td>
<td>26,747</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>http4k-undertow</td>
<td>26,713</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>actix-pg</td>
<td>26,674</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>h2o</td>
<td>26,250</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>act-hibernate-pgsql</td>
<td>25,756</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>actix-diesel</td>
<td>25,755</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>fasthttp-pgsql</td>
<td>25,668</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>chi-goaj</td>
<td>25,475</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>jooby</td>
<td>25,207</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>chi</td>
<td>24,866</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>chi-json</td>
<td>24,027</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>undertow-pgsql</td>
<td>22,767</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>ulib-postgres</td>
<td>22,661</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>http4k-netty</td>
<td>22,643</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>pedestal</td>
<td>22,536</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>act-hibernate-mysql</td>
<td>22,160</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>cpoll_cppsp-raw</td>
<td>21,806</td>
<td>0</td>
</tr>
</tbody>
</table>
Introducing reactive to the EE world
...without breaking everything!

Vert.x  ➔  MicroProfile  ➔  Jakarta EE
Times change quickly
From Java EE to Jakarta EE

- JAX-RS: '07
- CDI (Java EE 6): '09
- Kafka: '11
- Vert.x: '12
- Java EE 7: '13
- Docker: '13
- Reactive manifesto: '14
- Microservices: '14
- Kubernetes: '14
- Serverless: '14
- Vert.x: '12
- Node.js: '11
- Microservices: '14
- Docker: '13
- Reactive manifesto: '14
- Vert.x: '12
- Node.js: '11
- Reactor manifesto: '14
- Serverless: '14
# Java EE: a feet in the past

<table>
<thead>
<tr>
<th>Typical use-case</th>
<th>Development model</th>
<th>Execution model</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRUD</td>
<td>CDI (injection)</td>
<td>Servlets</td>
</tr>
<tr>
<td>Database with ORM</td>
<td>JAX-RS (REST API)</td>
<td>1 thread per request</td>
</tr>
<tr>
<td></td>
<td>JPA, Bean Validation, JTA, Management...</td>
<td></td>
</tr>
</tbody>
</table>
Not *cloud-friendly*, not *reactive*

Based on the concept of application server

➔ 1 server = *many* applications

Does not fit the “*cloud model*”

➔ Small, lightweight applications
➔ Each service in its own process
Not *cloud-friendly*, not *reactive*

Most of the model is synchronous except

- Asynchronous servlets
- CDI events

Also:

- No stream support
- No back-pressure
A new hope?

**MicroProfile** - the rogue child
- A new set of specs
- Microservice-friendly
- Very iterative
- Reactive working group!

**Jakarta EE**
- New spec model
- Faster processes
- Growing interest for reactive
Reactive and MicroProfile
Reactive Microprofile specifications

Reactive Streams Operators
Reactive Messaging
Concurrency

Traction from: Lightbend, Red Hat, Oracle, IBM
JAX-RS and reactive

```java
@Inject
private Neo neo;

@Produces(MediaType.SERVER_SENT_EVENTS)
@GET
@Path("/neo")
public Publisher<String> stream() {
    return neo.state();
}
```
Reactive Messaging (and data streaming)

```java
@Incoming("health")
@Outgoing("heartbeat")
@Multicast
public JsonObject filtered(JsonObject input) {
    LOGGER.info("Received {}", input.encode());
    return input.getJsonObject("heartbeat");
}
```
Reactive Messaging (and data streaming)

```java
@Incoming("kafka-state")
@Outgoing("state")
@Acknowledgment(POST_PROCESSING)
@Multicast
public JsonObject forwardState(JsonObject json) {
    LOGGER.info("Forwarding state {}", json.encode());
    return json;
}
```

Manuel, pre and post processing
Reactive Stream Operators

```java
@Incoming("health")
@Outgoing("heartbeat")
@Multicast
public PublisherBuilder<Message<JsonObject>> process(PublisherBuilder<MqttMessage> input) {
    return input
        .map(Message::getPayload)
        .map(bytes -> Buffer.buffer(bytes).toJsonObject())
        .flatMapCompletionStage(json -> invokeStoreService(json).thenApply(x -> json))
        .map(json -> json.getJSONObject("heartbeat"))
        .map(Message::of);
```
Demos

* no rabbits were harmed in the making of this.
Step #1
RestEasy, server-sent events and reactive streams

{sleeping, awake, eating}
Step #2
Introducing MicroProfile Reactive Messaging

Neo
(sleeping, awake, eating)

HealthDataBean
(temperature, heartbeat, pressure)

Processor

Health data

state

Heartbeat

Health data

Heartbeat

(temperature, heartbeat, pressure)
Step #3
Introducing MicroProfile Reactive Streams Operators

HealthDataBean
(temperature, heartbeat, pressure)

HTTP Client (Vert.x)

Processor

Snapshot service (latest data)

Heartbeat
Step #4
Health data comes from MQTT

MQTT

HTTP Client (Vert.x)

Processor

Snapshot service (latest data)

Heartbeat

Health data

Health data

Health data
Step #5
Event-driven and asynchronous microservices

MQTT
Health data
Collector service

Kafka
Events split by topic

SSE
Events

HTTP
Snapshot service
Web application
Before we wrap up...
SMALLRYE

COMMUNITY DRIVEN IMPLEMENTATIONS OF ECLIPSE MICROPROFILE.

TELL ME MORE
# SmallRye

<table>
<thead>
<tr>
<th>JAX-RS</th>
<th>Reactive Messaging</th>
<th>Reactive Stream Ops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vert.x</td>
<td><em>Vert.x</em>-powered</td>
<td>Vert.x + RxJava2</td>
</tr>
<tr>
<td>CDI (Weld)</td>
<td>MQTT</td>
<td></td>
</tr>
<tr>
<td>RestEasy</td>
<td>Apache Kafka</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Apache Camel</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AMQP 1.0</td>
<td></td>
</tr>
</tbody>
</table>
Roadmap

- **MicroProfile Reactive Stream Operators**
  - Approaching 1.0
  - Starting to infuse in the other MicroProfile specifications
  - Integration in the next MicroProfile release (February 2019)

- **MicroProfile Reactive Messaging**
  - Data streaming and Event-driven microservice
  - Integration in the next MicroProfile release (February 2019)

- **MicroProfile Concurrency**
  - Async context propagation
50% off with ml ponge code

(shameless plug #2)
Q&A

@jponge  @tsegismont  @clementplop
THANK YOU

plus.google.com/+RedHat
linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHat