Java EE 8 finally final! Now what?

Jfokus

February 2018

David Delabassee - @delabassee
Oracle

JavaYourNext (Cloud)

Safe Harbor Statement

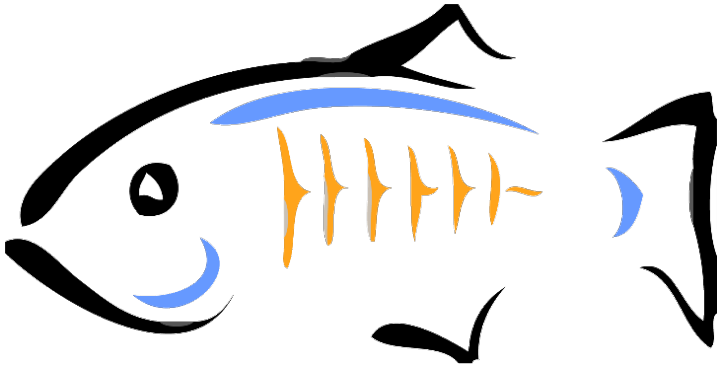
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

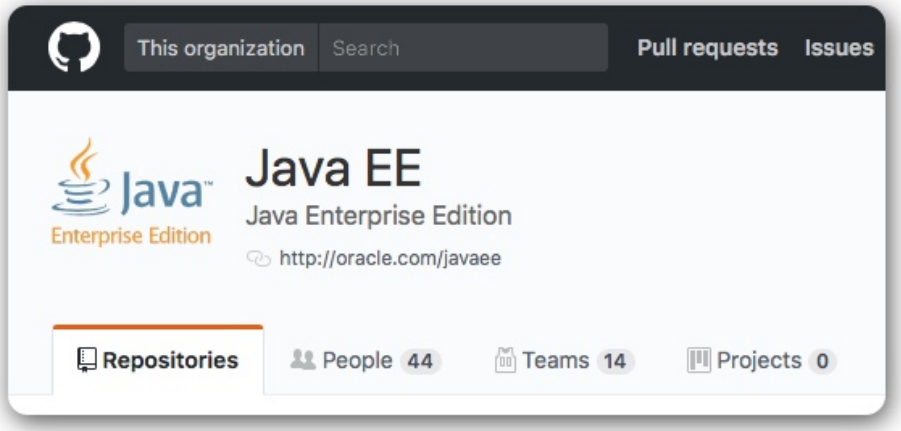
Jfokus 2017

> “ The reports of my death are greatly exaggerated.
>
> Java EE

Today – Java EE 8

- Released Sept. 21 2017
- GlassFish 5 - Open Source RI
  - https://javaee.github.io/glassfish/
  - https://hub.docker.com/r/oracle/glassfish/
- Open
  - https:/github.com/javaee/
  - https://javaee.groups.io/





Tomorrow - Eclipse Enterprise for Java

Moving Java EE to Eclipse Foundation

Technology

IBM

ORACLE

redhat

Community and Vendors

Sponsorship

eclipse

Enterprise for Java

- ✓ Nimble
- ✓ Flexible
- ✓ Open
- ✓ Compatible

https://projects.eclipse.org/projects/ee4j/

Why Eclipse ?

- Java and Java EE friendly
  - Eclipse Yasson
  - EclipseLink
- MicroProfile.IO
- Well defined processes
- Etc.

EE4J

Objectives – Short Term

- Ship a Java EE 8-compliant release as quickly as possible!
- Demonstrate that the EE4J projects are fully functional as open source projects
  - Eg. Build infrastrcuture
- Foster an ecosystem around the EE4J code

EE4J

Objectives – Mid Term

- Define processes to evolve EE4J technologies
  - Inc. Governance and IP flows
- Define compatibility rules and branding process for implementations to ensure application portability
- See EE.Next WG draft charter
  - https://www.eclipse.org/org/workinggroups/eclipse_ee_next_charter.php

EE4J

Report card

- PMC established
  - Ivar Grimstad, IBM, Oracle, Payara, Red Hat, Tomitribe & Eclipse
- Brand name
  - Stay tunned
- EE.Next WG draft charter
- Code?

EE4J

Report card

- Done!
  - Eclipse Grizzly
  - Eclipse OpenMQ
  - Eclipse Project for JAX-RS
  - Eclipse Project for JMS
  - Eclipse Tyrus
  - Eclipse Project for WebSocket
  - Eclipse Project for JSON Processing
  - Eclipse Yasson (*)
  - EclipseLink (*)
- Ongoing
  - Eclipse Jersey
  - Eclipse Mojarra
- Next
  - JSON-B API
  - Concurrency
  - Security
  - JTA
  - JavaMail
  - JAX-B & JAX-WS
  - JSTL
  - UEL
  - JAF
  - etc.

https://github.com/eclipse-ee4j

JAX-RS

JavaYourNext (Cloud)

JAX-RS Client API

javax.ws.rs.client.Client interface

- Fluent API
  - Client Builder ➔ Client ➔ Web Target ➔ Request building ➔ Response

```
List<Forecast> forecast = ClientBuilder.newClient()
                                       .target("http://weath.er/cities")
                                       .request()
                                       .accept("application/json")
                                       .header("foo","bar")
                                       .get(new GenericType<List<Forecast>>() {});
```

JAX-RS Client API

Asynchronous invocation

```
Future<String> fCity = client.target("http://locati.on/api")
                             .queryParam("city", "Paris")
                             .request()
                             .async()
                             .get(String.class);

String city = fCity.get();
```

JAX-RS Client API

InvocationCallback

```
WebTarget myResource = client.target("http://examp.le/api/read");
Future<Customer> future = myResource.request(MediaType.TEXT_PLAIN)
        .async()
        .get(new InvocationCallback<Customer>() {
            @Override
            public void completed (Customer customer) {
                // do something with the customer
            }
            @Override
            public void failed (Throwable throwable) {
                // Oops!
            }
        });
```

...

JAX-RS Client API

New JAX-RS Reactive Invoker

```
// JAX-RS 2.0
Response response = client.target(recommandationService)
        .request()
        .get();

Future<Response> futureResponse = client.target(recommandationService)
        .request()
        .async()
        .get();

// JAX-RS 2.1
CompletionStage<Response> completionStageResp = client.target(recommandationService)
        .request()
        .rx()
        .get();
```

JAX-RS 2.1

```
CompletionStage<JsonObject> cfIp = client.target("http://api.ipify.org/")
                .queryParam("format", "json").request()
                .rx()
                .get(JsonObject.class);

Function<JsonObject, CompletionStage<JsonObject>> function = ip
                -> client.target("https://ipvigilante.com")
                        .path(ip.getString("ip")).request()
                        .rx()
                        .get(JsonObject.class);

cfIp.thenCompose(function)
    .thenAccept(System.out::println);
```

JAX-RS 2.1 – Client API

	Sync	Async	RX
Performance and scalability	✗✗	✔	✔
Easy to develop and maintain	✔	✗	✔
… complex workflow	✗	✗	✔
… error handling	✗	✗	✔
Leverage new Java SE feature	✗	✗	✔

JAX-RS 2.1 – RX Invoker

- Implementations **MUST** support an invoker for **CompletionStage**
- Implementations **MAY** support other reactive APIs
- Jersey
  - CompletionStageRxInvoker (Default)
  - RxListenableFutureInvoker – Guava
  - RxObservableInvoker – RxJava
  - RxFlowableInvoker – RxJava2

```
client.register(RxFlowableInvokerProvider.class);
client.target(...)...
      .rx(RxFlowableInvoker.class)
      .get();
```

https://github.com/jersey/jersey/tree/master/ext/rx

Server-Sent Events

- WHATWG standard
- Supported in all modern browsers
- Persistent, one-way communication channel
- Text protocol, special media type "text/event-stream"
- Server can send multiple messages (events) to a client
- Can contain id, name, comment, retry interval

Server-Sent Events

- javax.ws.rs.sse.**SseEvent** interface
- **OutboundSseEvent**
  - Server-side representation of a Server-Sent event
  - OutboundSseEvent.Builder()
- **InboundSseEvent**
  - Client-side representation of a Server-Sent event

Server-Sent Events

Server-side

- **SseEventSink**
  - Outbound Server-Sent Events stream

```
@GET
@Path ("sse")
@Produces(MediaType.SERVER_SENT_EVENTS)
public void eventStream(@Context SseEventSink eventSink, @Context SSE sse) {
    ...
    eventSink.send( sse.newEvent("an event") );
    eventSink.send( sse.newEvent("another event") );
    ...
    eventSink.close();
}
```

Server-Sent Events

Client side

- **SseEventSource**
  - Client for processing incoming Server-Sent Events

```
WebTarget target = client.target("http://…");
try (SseEventSource source = SseEventSource.target(target)
                                           .reconnectingEvery(5, SECONDS)
                                           .build()) {
   source.register(System.out::println); //InboundSSEvent consumer
   ...
   source.open();
}
catch (InterruptedException e) {
   // Ooops
}
```

JAX-RS 2.1

- Resource method can return a CompletionStage
- @PATCH
- JSON-P & JSON-B support
- New ClientBuilder methods
  - #connectTimeout(long, TimeUnit);
  - #scheduledExecutorService(ScheduledExecutorService);
- Application provided providers priority

…

JSON

JavaYourNext (Cloud)

JSON-P 1.1

- Standard API to parse, generate, transform, query JSON
- Update JSON-P spec to stay current with emerging standards (RFC 7159)
  - **JSON Pointer** (RFC 6901)
  - **JSON Patch** (RFC 6902)
  - **JSON Merge Patch** (RFC 7396)
- Add editing/transformation operations to JSON objects and arrays
- Support JSON Collectors
- Support for processing big JSON

JSON Pointer

- IETF RFC 6901
- String syntax for identifying a specific value
  - Token(s) separated by "/"
    - specify key in object
    - or index into array
  - Ex. "/event/location", "/conferences/0"
- Special cases
  - Escape "/" with "~1" and "~" with "~0"
  - "/" points to the "" key in the root
  - "-" refers to the end of an array

JSON Pointer

```
JsonStructure jsonEvents = …

JsonPointer pt = Json.createPointer("/1/venue");

JsonValue preEvt = pt.getValue(jsonEvents);
// "Hilton"

JsonStructure newEvt = pt.replace(jsonEvents,
                Json.createValue("Moscone West"));


// + add, remove & containsValue
```

```
[
    {
        "event": "OpenWorld",
        "venue": "Moscone North"
    },
    {
        "event": "JavaOne",
        "venue": "Hilton"
    }
]
```

```
[
    {
        "event": "OpenWorld",
        "venue": "Moscone North"
    },
    {
        "event": "JavaOne",
        "venue": "Moscone West"
    }
]
```

JSON Patch

- IETF RFC 6902
- Modify Parts of JSON document
- Patch is a JSON document itself
- Operations
  - Add, replace, remove, move, copy & test
- HTTP PATCH method (application/json-patch+json)

JSON Patch

```
[
    {
        "op": "replace",
        "path": "/0/venue",
        "value": "Moscone West"
    },
    {
        "op": "add",
        "path": "/0/previousVenue",
        "value": "Hilton"
    }
]
```

```
[
    {
        "event": "JavaOne",
        "venue": "Hilton"
    }
]
```

JSON Patch

```
[
    {
        "op": "replace",
        "path": "/0/venue",
        "value": "Moscone West"
    },
    {
        "op": "add",
        "path": "/0/previousVenue",
        "value": "Hilton"
    }
]
```

```
[
    {
        "event": "JavaOne",
        "venue": "Moscone West"
    }
]
```

JSON Patch

```
[
    {
        "op": "replace",
        "path": "/0/venue",
        "value": "Moscone West"
    },
    {
        "op": "add",
        "path": "/0/previousVenue",
        "value": "Hilton"
    }
]
```

```
[
    {
        "event": "JavaOne",
        "venue": "Moscone West",
        "previousVenue":"Hilton"
    }
]
```

JSON Patch

```
JsonArray previousJ1 = …
JsonArray patch = …

JsonPatch jpVenue = Json.createPatch(patch);

JsonArray currentJ1 = jpVenue.apply(previousJ1);


JsonPatch patch2018 = Json.createPatchBuilder()
                .copy("/0/previousVenue","/0/venue")
                .replace("/0/venue", "Moscone North & South")
                .add("/0/days", 6)
                .build();

JsonArray nextJ1 = patch2018.apply(previousJ1);
```

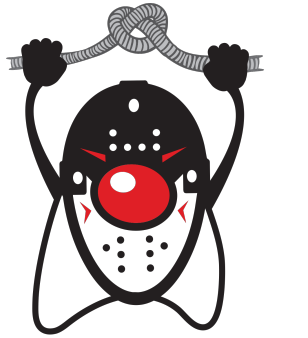
JSON-P 1.1

Misc.

- JSON Merge Patch
- Merge & Merge Patch Diff
- JSON-P collectors
- Support for processing big JSON
  - Filters to JSON parsing : skipArray(), skipObject()
- …

JSON Binding

- API to serialize/deserialize Java objects to/from JSON documents
  - Similar to JAX-B
  - Standard API for existing framework (ex. Genson, Gson)
- Default mapping between classes and JSON
- Customization APIs
  - Annotations (@JsonbProperty, @JsonbNillable)
  - Runtime configuration builder
- Natural follow on to JSON-P
  - Closes the JSON support gap
  - Allows to change providers



JSON-B

```
// public Event(String name, int edition, String venue)

List<Event> events = new ArrayList<>();
h2.add(new Event("JavaOne", 2017, "SFO"));
h2.add(new Event("OpenWorld", 2017, "SFO"));
h2.add(new Event("Devoxx", 2017, "Antwerp"));

Jsonb jsonb = JsonbBuilder.create();
String nextUp = jsonb.toJson(events);
```

```
[
  {
    "edition": 2017,
    "name": "JavaOne",
    "venue": "SFO"
  },
  {
    "edition": 2017,
    "name": "OpenWorld",
    "venue": "SFO"
  },
  {
    "edition": 2017,
    "name": "Devoxx",
    "venue": "Antwerp"
  }
]
```

JSON-B

```
String q1Confs = "…";
Jsonb jsonb = JsonbBuilder.create();
Event event = jsonb.fromJson(q1Confs, Event.class);
```

```
{
  "edition": 2018,
  "name": "Oracle Code",
  "venue": "Global",
  "cost" : 0
}
```

JSON-B Customizations

- Annotation
  - @JsonbProperty
- Scope
  - Field
  - Getter/Setter
  - Parameter

```
public class Event{
    private int edition;
    @JsonbProperty("conference")
    private String eventName;
}

public class Customer {
    public int edition;
    public String venue;
    @JsonbProperty("conference")
    public String getEventName() {
       return eventName;
   }
}
```

JSON-B Customizations

- Naming strategies
  - IDENTITY : myProp
  - LOWER_CASE_WITH_DASHES : my-prop
  - LOWER_CASE_WITH_UNDERSCORES : my_prop
  - UPPER_CAMEL_CASE : MyProp
  - UPPER_CAMEL_CASE_WITH_SPACES : My Prop
  - CASE_INSENSITIVE : myprop
  - Or a custom strategy
- Property ordering
  - Any, Lexicographical, Reverse
- Binary Data Strategies
  - Base64, Base64 URL, Byte

JSON-B Customizations

- Property to ignore
- Null handling
- Custom instantiation
- Fields visibility
- Date/Number Formats
- …

JSON-B Customizations

```
//Ordering, naming strategy, encoding, Locale, …
JsonbConfig config = new JsonbConfig()
                                .withFormatting(true)
                                .withAdapters(new CarAdapter());

Jsonb jsonb = JsonbBuilder.create(config);



Jsonb jsonb = JsonBuilder.newBuilder("anotherProvider");
```

Web

JavaYourNext (Cloud)

Servlet 4.0

- Support for HTTP/2
  - Request/response multiplexing
  - Server push
  - Upgrade from HTTP 1.1
- Smaller community-requested improvements
  - Allow setting the default context-path without resorting to container specific config
  - Allow setting the jsp-file programmatically
  - Allow encoding to be set from deployment descriptor
  - Servlet Mapping API

HTTP/2

- Binary Framing over single TCP connection 
- Request/Response multiplexing
- Stream Prioritization
- Server Push
- Upgrade from HTTP 1.1
- Header Compression
- Preserve HTTP semantic 
- Flow Control

Servlet 4.0

```
PushBuilder builder = myRequest.newPushBuilder();

builder.addHeader("X-Pusher", …);

builder.path("resource")
       .push();
```

JSF 2.3

- HTTP/2 Server Push
- Better CDI Integration
  - Way more things are injectable
- Java Time support
- WebSocket Integration
- Ajax Method Invocation
- Class Level Bean Validation
- UIData and UIRepeat improvements

CDI 2.0 & BV 2.0

JavaYourNext (Cloud)

CDI 2.0

- Define behavior of CDI outside of a Java EE container
  - Inc. API to bootstrap a CDI container in Java SE
- Spec split into 3 parts
  - CDI Core
  - CDI for Java SE
  - CDI for Java EE
- Apply Interceptor on Producer
- Observers ordering
- Asynchronous events
- Alignment with Java SE 8, …

CDI 1.2

```
@Inject
Event<PaymentEvent> debitEvent;

// producer
debitEvent.fire(somePayload);


// consumer
public void anObesrver(@Observes Payload p) {
…
}
```

CDI 1.2

```
// consumer A
public void anObserver(@Observes Payload p) {
…
}

// consumer B
public void anotherObesrver(@Observes Payload p) {
…
}
```

CDI 2.0

```
// consumer A
public void anObesrver(@Observes @Priority(10) Payload p) {
…
}

// consumer B
public void anotherObesrver(@Observes @Priority(20) Payload p) {
…
}
```

CDI 2.0

```
@Inject
Event<PaymentEvent> debitEvent;

// async producer
CompletionStage<Payload> cs = debitEvent.fireAsync(somePayload);


// async consumer
public void anObesrver(@ObservesAsync Payload p) {
…
}
```

Bean Validation 2.0

- Embrace Java SE 8
  - Support for new Date/Time API
  - Constraints applied to collection elements
  - Optional wrappers
  - Repeatable annotations
- Introduce new constraints
  - @NotEmpty, @NotBlank, @Email
  - @PastOrPresent, @FutureOfPresent
  - @Positive, @PositiveOrZero, @Negative, @NegativeOrZero
- …

Security

JavaYourNext (Cloud)

Identity Store

- Provide a storage system where caller credentials and data are stored
  - LDAP, DataBase, ...
- Perform caller validation and details retrieval
  - In : Valid caller name & password
  - Out : (Possibly different) caller name and/or associated group(s)
- Does not interact with the caller!

Identity Store

```
@DatabaseIdentityStoreDefinition(
    dataSourceLookup = "${'java:global/MyDS'}",
    callerQuery = "#{'select password from caller where name = ?'}",
    groupsQuery = "select group_name from caller_groups where caller_name = ?",
    hashAlgorithm = Pbkdf2PasswordHash.class,
    priorityExpression = "#{100}",
    hashAlgorithmParameters = {
          "Pbkdf2PasswordHash.Iterations=3072", "${applicationConfig.dyna}"
    }
)
```

Authentication Mechanism

- CDI enabled version of **ServerAuthModule** that complies to the JASPIC Servlet Container Profile
- Encouraged to use an **IdentityStore**
  - Caller credential validation
  - Caller details retrieval
- Built-in
  - @BasicAuthenticationMechanismDefinition
  - @FormAuthenticationMechanismDefinition
  - @CustomFormAuthenticationMechanismDefinition

Authentication Mechanism

- New **HttpAuthenticationMechanism** interface (javax.security.enterprise.authentication.mechanism.http)
  - void **cleanSubject**(HttpServletRequest, HttpServletResponse, HttpMessageContext)
  - AuthenticationStatus **validateRequest**(Req ,Resp, MCtx) throws AuthException;
  - AuthenticationStatus **secureResponse**(Req ,Resp, MCtx) throws AuthException;

Security API for Java EE

```
@WebServlet("/protectedServlet")
@ServletSecurity(@HttpConstraint(rolesAllowed = "foo"))
public class ProtectedServlet extends HttpServlet {
   ...
}
```

```
@ApplicationScoped
public class myAuthMech implements HttpAuthenticationMechanism {

   @Inject
   private IdentityStoreHandler myIdentityStore;

   AuthenticationStatus status validateRequest(HttpServletRequest req,
        HttpServletResponse res, HttpMessageContext ctx) throws AuthenticationException {
        ...
```

Wrap-up

JavaYourNext (Cloud)

Java EE 8 – Modernization & Simplification

JAX-RS 2.1	Reactive Client API, Server-Sent Events, …
Servlet 4.0	HTTP/2, Server Push, …
JSON-B 1.0 (*)	Java <-> JSON binding
JSON-P 1.1	Updates to JSON standards, JSON Collectors, …
CDI 2.0	Async Event, Observers ordering, SE support, …
Bean Validation 2.0	Embrace Java SE 8, new constraints, …
JSF 2.3	Improved CDI, WebSocket, SE 8 integration, …
Security 1.0 (*)	Portable Identity Store, Authentication & Security Context

Tomorrow – EE4J

- "EE4J 1.0" will be compatible with Java EE 8
- Join the discussion
  - https://dev.eclipse.org/mhonarc/lists/ee4j-community/

Tack!


ORACLE®