Jakarta EE – Present and Future

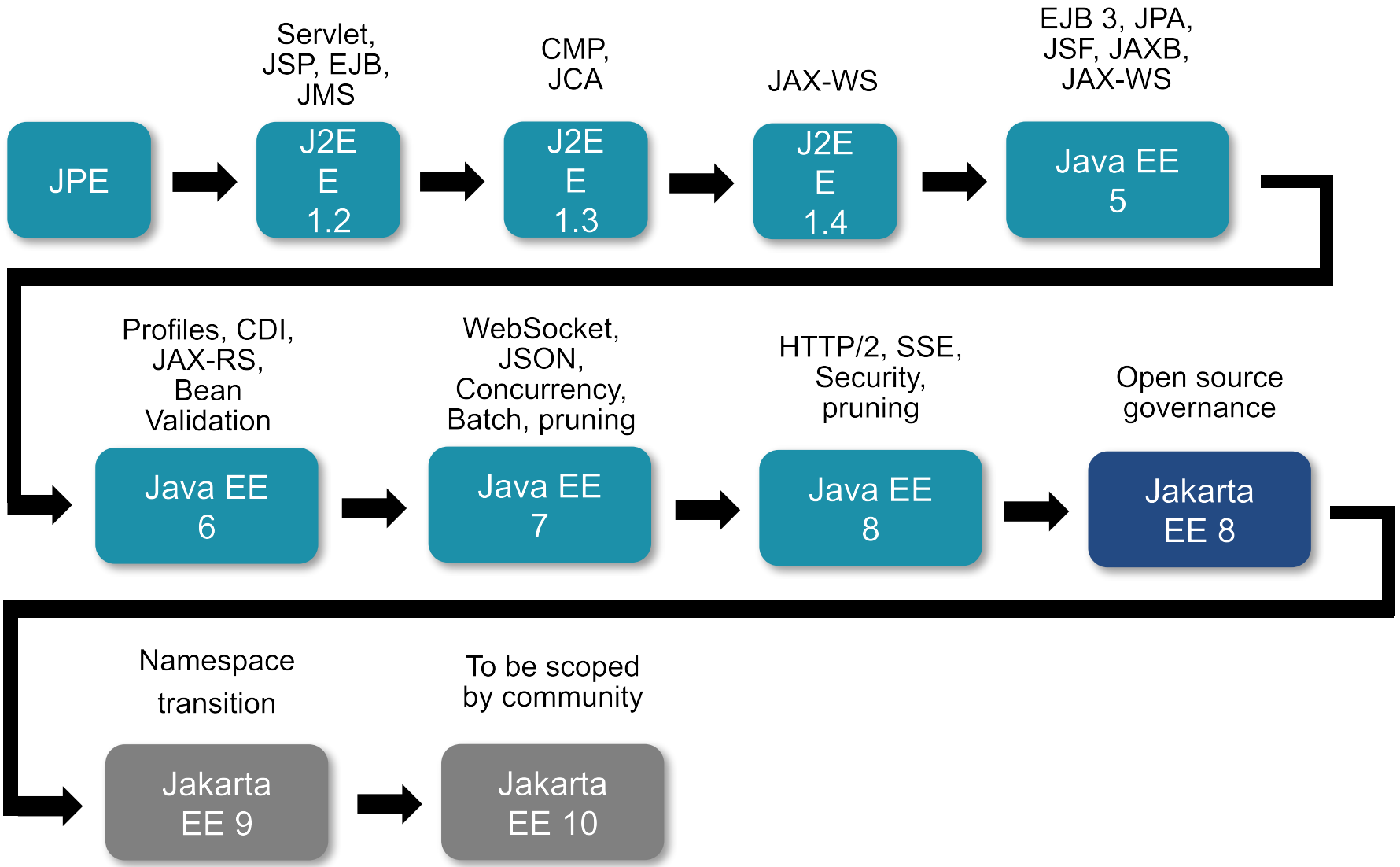
Michael P. Redlich
Senior Research Technician
mike@redlich.net
@mpredli

Jakarta EE

- Java EE transitioned from JCP to Eclipse Foundation as Jakarta EE
- Open governance, open source, open compatibility testing
- Well defined specification process, clear IP flow, vendor-neutral open collaboration, level playing field
- Key stakeholders maintained if not expanded including Oracle, IBM, Payara and Pivotal
- Community participation and contribution key

https://jakarta.ee

Jakarta EE Evolution



Jakarta EE 8 At a Glance

- Web Standards Alignment
  - HTTP/2, Server-Sent Events, JSON Binding, JSON Pointer, JSON Patch
- CDI Alignment
  - CDI 2, Faces managed bean pruning, injecting Faces artifacts, CDI support in Persistence
- Simplicity
  - Security, EJB pruning
- Java SE Alignment
  - Repeatable annotations, Date-Time API, streams, completable futures
  - Jakarta Faces, Jakarta Persistence, Jakarta REST, Bean Validation

Jakarta Servlet 4

- Principal goal to support HTTP/2
  - Request/response multiplexing over single connection
  - Multiple streams, stream prioritization
  - Server push
  - Binary framing
  - Header compression

- Most of it done without major API changes

Jakarta JSON Binding

- API to marshal/un-marshal POJOs to/from JSON
  - Very similar to Jakarta XML Binding in the XML world
- Default mapping of classes to JSON
  - Annotations to customize default mappings
  - @JsonbProperty, @JsonbTransient
- Provides Jakarta REST a built-in way to support “application/json” for POJOs
  - Providers already supported non–standard binding APIs

JSON Binding Example

```
@GET ...
@Produces("application/json")
public Person getPerson(...)
{
  ...
  Person duke = new Person();
  duke.setName("Duke");
  duke.setGender("Male");
  phones = new HashMap<>();
  phones.put("home",
    "650-123-4567");
  phones.put("mobile",
    "650-234-5678");
  duke.setPhones(phones);

  return duke;
}
```

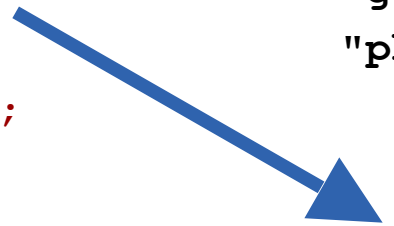
```
{
  "name":"Duke",
  "gender":"Male",
  "phones":{
    "home":"650-123-4567",
    "mobile":"650-234-5678"
  }
}
```

Jakarta JSON Processing 1.1

- Updates to JSON parsing API added in Java EE 7
- Adapted to new JSON standards
  - JSON Pointer
  - JSON Patch
- Java SE streams alignment

JSON Pointer Example

```
JsonArray contacts = ...;
JsonPointer pointer =
    new JsonPointer(
        "/0/phones/mobile");
JsonValue value =
  pointer.getValue(contacts);
```



```
[
  {
    "name":"Duke",
    "gender":"Male",
    "phones":{
      "home":"650-123-4567",
      "mobile":
          "650-234-5678"}},
  {
    "name":"Jane",
    "gender":"Female",
    "phones":{
       "mobile":
           "707-555-9999"}}
]
```

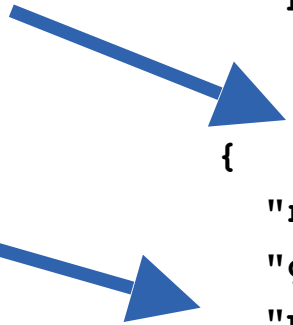
JSON Patch

- Modifying parts of a JSON document
- Patch itself a JSON document
  - add, replace, remove, move, copy, test operations
  - Must have "op" field and "path" field, may have “value” field

```
[
  {"op": "replace", "path":"/0/phones/mobile",
      "value":"650-111-2222"},
  {"op": "remove", "path":"/1"}
]
```

JSON Patch Example

```
[
  {
    "op":"replace",
    "path":"/0/phones/mobile",
    "value":"650-111-2222"},
  {
    "op":"remove",
    "path":"/1"}
]
```



```
[
  {
    "name":"Duke",
    "gender":"Male",
    "phones":{
      "home":"650-123-4567",
      "mobile":"650-234-5678"}},
  {
    "name":"Jane",
    "gender":"Female",
    "phones":{
       "mobile":"707-555-9999"}}
]
```

```
JsonPatchBuilder builder = new JsonPatchBuilder();
JsonArray result = builder.replace("/0/phones/mobile", "650-111-2222")
                          .remove("/1")
                          .apply(target);
```

Server-Sent Events (SSE)

- Lesser known part of HTML 5
- Server-to-client streaming
  - “Stock tickers”, monitoring applications
- Just plain long-lived HTTP
  - Between the extremes of vanilla request/response and WebSocket
  - Content-type ‘text/event-stream’
- Support via Jakarta REST 2.1
  - Pre-existing non-standard API in Jersey

SSE on the Server-Side

```
@Path("tickers")
public class StockTicker {
  @Resource ManagedExecutorService executor;

  @GET @Produces("text/event-stream")
  public void getQuotes(
      @Context SseEventSink sink,
      @Context Sse sse) {
    executor.execute(() -> {
      ...
      sink.send(sse.newEvent(stockqoute));
      ...
    });
  }
}
```

Jakarta Security

- Simplify security for Jakarta EE and improve portability
- Simple security providers
  - Database, LDAP
- Simple pluggability
- Universal security context

Simple Security Providers

```
@DataBaseIdentityStoreDefinition (
    dataSourceLookup="java:global/MyDB",
    callerQuery=
        "SELECT password FROM principals WHERE username=?",
    groupsQuery="SELECT role FROM roles where username=?", ...)

@LdapIdentityStoreDefinition(
    url="ldap://myldapserver:33389/",
    callerBaseDn="ou=caller,dc=acme,dc=com",
    groupBaseDn="ou=group,dc=acme,dc=com", ...)
```

Simple Security Pluggability

```
@ApplicationScoped
public class MyIdentityStore implements IdentityStore {
  @Inject UserService userService;

  @Override
  public CredentialValidationResult validate(
      Credential credential) {
    // Validate credentials using the user service.
  }
}
```

Security Context Example

```
@AccessLogged @Interceptor
public class AccessLoggingInterceptor {
  @Inject private SecurityContext security;

  @AroundInvoke
  public Object logMethodAccess(InvocationContext ctx)
      throws Exception {
    System.out.println("Entering method: "
        + ctx.getMethod().getName());
    System.out.println("Current principal: "
        + security.getCallerPrincipal());
    System.out.println("User is admin: "
        + security.isCallerInRole("admin"));
    return ctx.proceed();
  }
}
```

Jakarta CDI 2

- Java SE bootstrap, modularity
- Asynchronous events
- Portable extension SPI simplifications and enhancements
- Java SE alignment

Asynchronous Events

```
@Inject @CargoInspected Event<Cargo> cargoInspected;
...
public void inspectCargo(TrackingId trackingId) {
   ...
   cargoInspected.fireAsync(cargo);
}
```

```
public void onCargoInspected(
    @ObservesAsync @CargoInspected Cargo cargo) {
```

Adopting Key Java SE Innovations

- Most key Java SE innovations can already be used with Java EE
- Some APIs need to adopt features
- Repeatable Annotations
  - Jakarta Persistence, Jakarta Messaging, Jakarta Mail, Bean Validation
- Date-Time API
  - Jakarta Faces, Jakarta Persistence, Bean Validation
- Completable Future
  - Jakarta REST
- Streams
  - Jakarta Persistence, JSON Processing

Repeatable Annotations in Jakarta EE

```
@NamedQueries({
    @NamedQuery(name=SELECT_ALL, query="..."),
    @NamedQuery(name=COUNT_ALL, query="...")
})
public class Customer {
   ...
```

```
@NamedQuery(name=SELECT_ALL, query="...")
@NamedQuery(name=COUNT_ALL, query="...")
public class Customer {
   ...
```

Date/Time API with Jakarta Persistence

```
@Entity
public class Accident {

  @Temporal(TemporalType.TIMESTAMP)
  @Past
  private Instant when;
}
```

CompletableFuture with Jakarta REST

```
CompletionStage<Assets> getAssets = client
                                    .target("assets/{ssn}")
                                    .resolveTemplate("ssn", person.getSsn())
                                    .request("application/json")
                                    .rx()
                                    .get(Assets.class);

CompletionStage<Liabilities> getLiabilities = client
                                    .target("liabilities/{ssn}")
                                    .resolveTemplate("ssn", person.getSsn())
                                    .request("application/json")
                                    .rx()
                                    .get(Liabilities.class);

Coverage coverage = getAssets.thenCombine(getLiabitities,
    (assets, liabilities) -> underwrite(assets, liabilities))
        .toCompletableFuture().join();
```

Streams with Jakarta Persistence

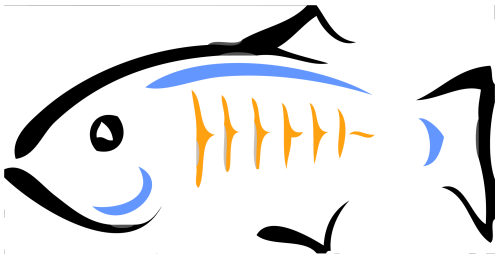
```
Stream<Book> books = entityManager.createQuery(
    "SELECT b FROM Book b", Book.class)
        .getResultStream();

books.map(b -> b.getTitle() + " was published on "
    + b.getPublishingDate())
        .forEach(s -> log.info(s));
```

Some Other Changes

- Jakarta Faces/WebSocket integration
- Jakarta Faces/CDI integration
- More Bean Validation constraints
  - @Email, @NotEmpty, @Past, @Future
- Broadcasting SSE events
- CDI event ordering
- Streams support in Jakarta Persistence

Jakarta EE 8 Implementations



Eclipse GlassFish

Open Liberty

Red Hat
JBoss Enterprise
Application Platform

https://jakarta.ee/compatibility/

Jakarta EE 9

- Move all relevant specifications from *javax* to *jakarta* namespace
- Remove older technologies
- Support both Java SE 8 and Java SE 11
- Jakarta XML Binding and Jakarta XML Web Services added to platform as Java SE has removed these technologies
- Primarily aimed for ecosystem to adapt to Jakarta
- Aimed to be done by Summer
- Your contribution is welcome and needed to get work done on time

https://jakartaee-ambassadors.io/getting-involved/guide-for-helping-deliver-jakarta-ee-9/

MicroProfile

Open Tracing	OpenAPI	Type-Safe REST Client	Configuration
Fault Tolerance	Metrics	JWT Propagation	Health Check

https://microprofile.io

Jakarta EE 10 Possibilities

- Jakarta NoSQL (formerly Eclipse JNoSQL)
- MVC
- Deprecating EJB
  - Move functionality to Jakarta Messaging/Concurrency/Security
- OpenID Connect, JWT
- Java SE alignment
  - Jakarta Concurrency, Jakarta Persistence
- CDI alignment
  - Batch, Jakarta REST
- Configuration

https://jakartaee-ambassadors.io

@jee_ambassadors

Summary

- Jakarta EE 8 very significant for the future of Java
- Jakarta EE 9 coming soon, followed by Jakarta EE 10 and beyond
- The time to get involved is now!

Resources

- JakartaOne Livestream recordings
  - https://jakartaone.org
- Jakarta EE Community alias
  - https://accounts.eclipse.org/mailing-list/jakarta.ee-community
- Jakarta EE Twitter handle
  - @JakartaEE
- Jakarta Tech Talks
  - https://www.meetup.com/jakartatechtalks_

JAKARTA™ EE AMBASSADORS