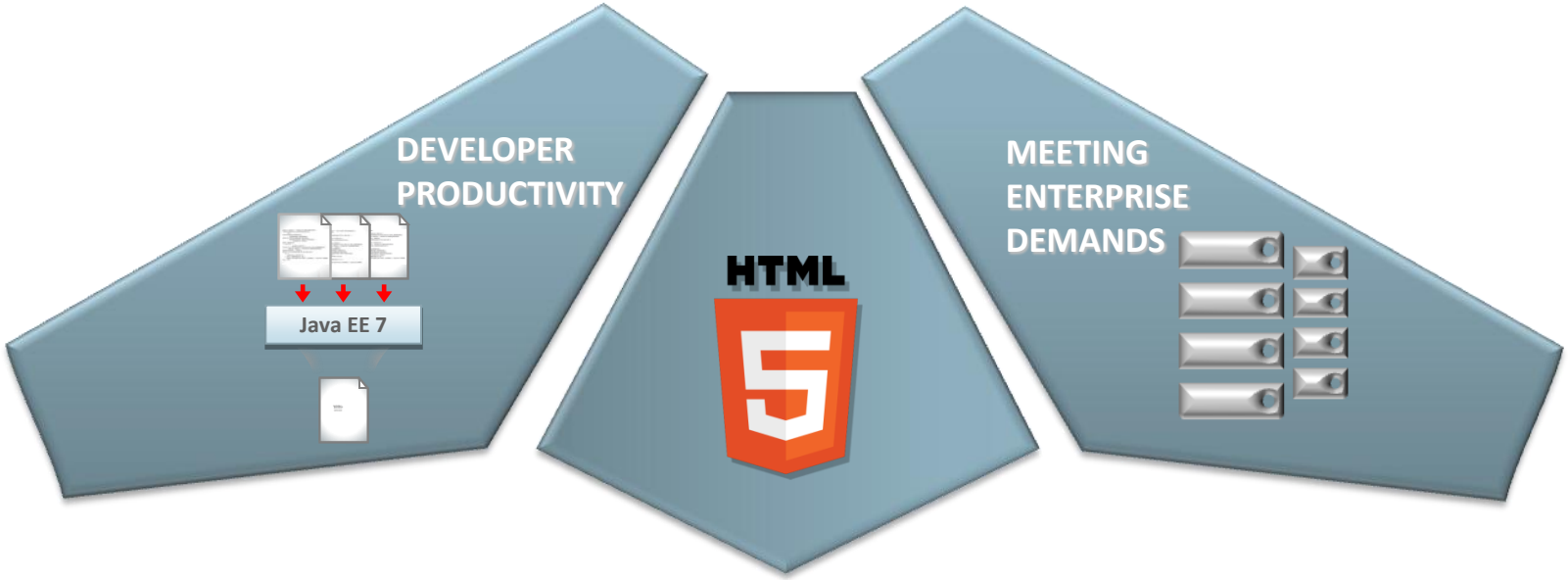
ORACLE

Java EE 8: A Community Update

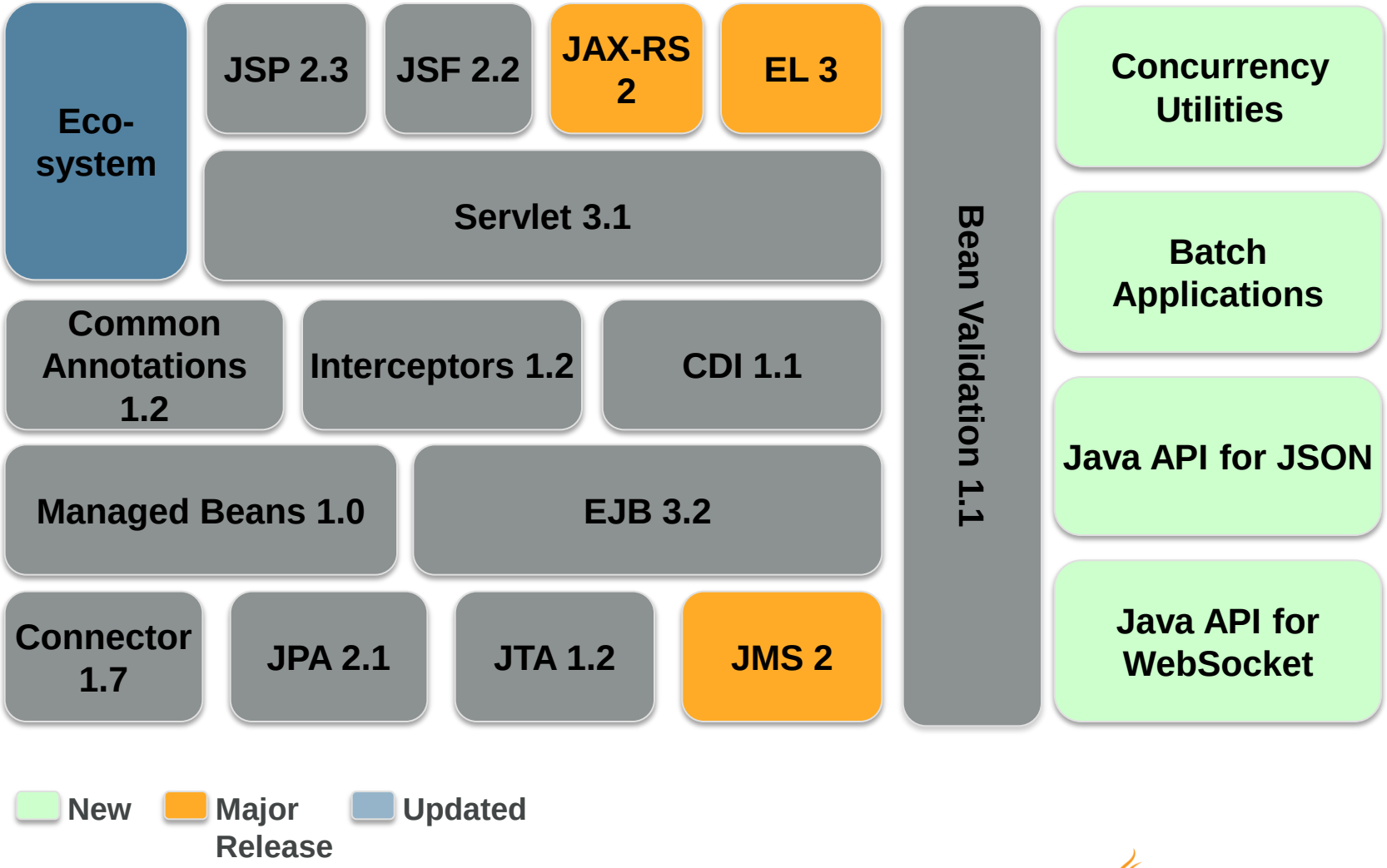
Reza Rahman
Java EE/GlassFish Evangelist
Reza.Rahman@Oracle.com
@reza_rahman

The preceding 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.

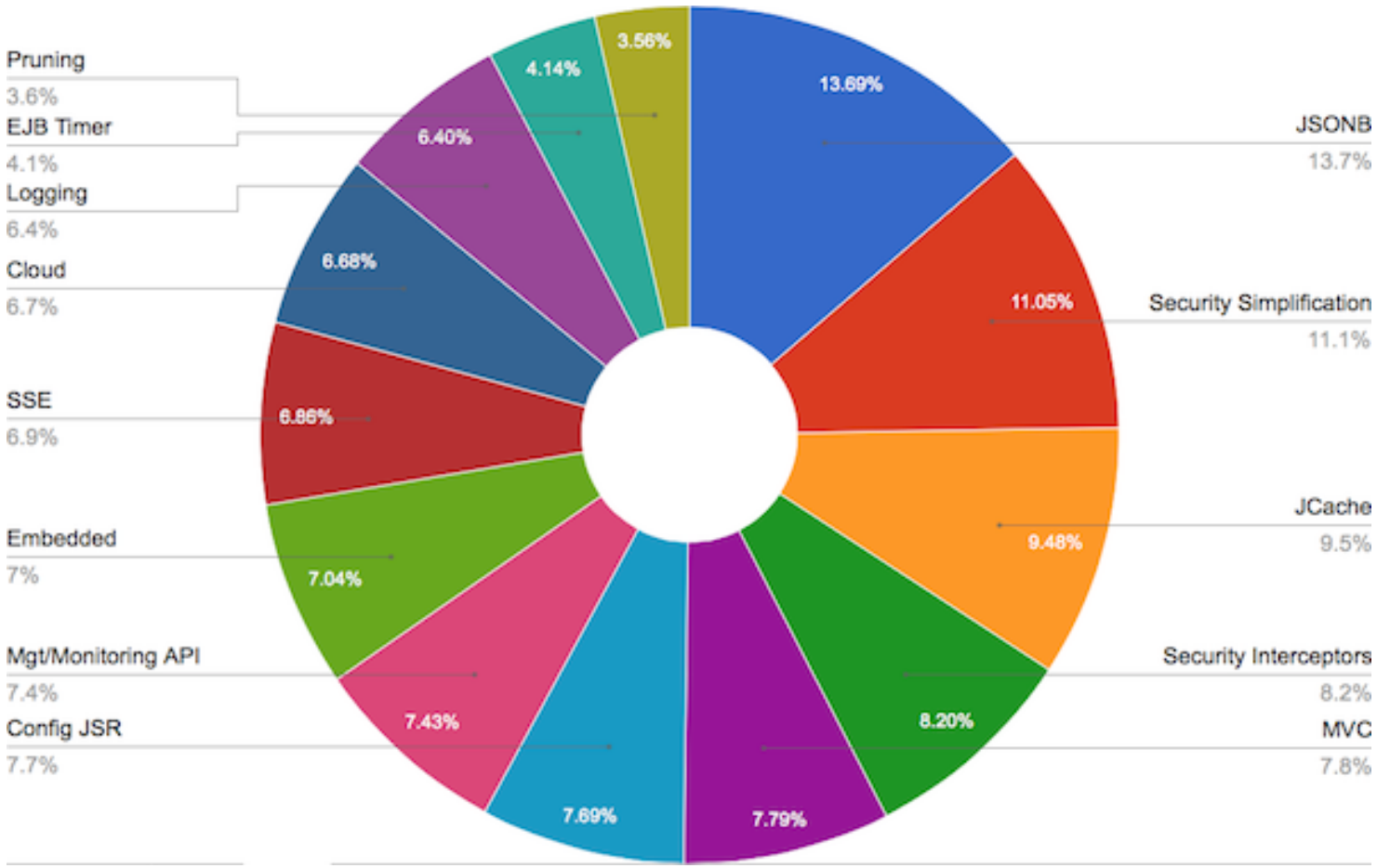
Java EE 7 Themes



Java EE 7



Java EE 8 Community Survey



https://blogs.oracle.com/ldemichiel/entry/results_from_the_java_ee

https://java.net/downloads/javaee-spec/JavaEE8_Community_Survey_Results.pdf

Java EE 8 Possibilities

- Web Standards/HTML5 Alignment
  - HTTP2, SSE, JSON-B, JSON-P, action-oriented web framework, hypermedia
- Cloud
  - Simple security providers, multitenancy, REST management/monitoring
- CDI Alignment
  - CDI 2, EJB services outside EJB, security interceptors, EJB pruning
- Enterprise
  - JCache, Configuration, JMS
- Java SE 8 alignment

Current Status

Already There

- Java EE 8 (JSR 366)
- CDI 2 (JSR 365)
- JSON-B (JSR 367)
- JMS 2.1 (JSR 368)
- Servlet 4 (JSR 369)
- JAX-RS 2.1 (JSR 370)
- MVC (JSR 371)
- JSF 2.3 (JSR 372)

Coming Soon

- Security
- Management and Monitoring
- JCache 1.1
- JSON-P 1.1

Farther Out

- Concurrency Utilities
- WebSocket
- JPA
- And more…

Servlet 4

- Principal goal to support HTTP 2
  - Request/response multiplexing over single connection
  - Multiple streams
  - Stream Prioritization
  - Server Push
  - Binary Framing
  - Header Compression

- Hopefully most of it can be done without major API changes

JSON-B

Java API for JSON Binding

- API to marshal/unmarshal POJOs to/from JSON
  - Very similar to JAXB in the XML world
- Default mapping of classes to JSON
  - Annotations to customize the default mappings
  - @JsonProperty, @JsonTransient, @JsonValue
- Draw from best of breed ideas in existing JSON binding solutions
  - MOXy, Jackson, GSON, Genson, Xstream, …
  - Allow switching providers
- Provide JAX-RS a standard way to support “application/json” for POJOs
  - JAX-RS currently supports JSON-P

JSON-B Possibilities

```
@Entity public class Person {
  @Id String name;
  String gender;
  @ElementCollection
  Map<String, String> phones;
  ...
}

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);
```

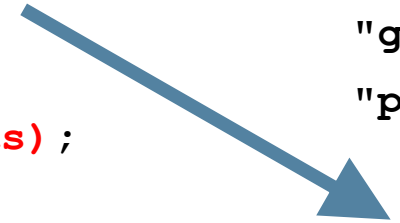
```
{
  "name":"Duke",
  "gender":"Male",
  "phones":{
    "home":"650-123-4567",
    "mobile":"650-234-5678"
  }
}
```

JSON-P 1.1

- Updates to new API in Java EE 7
- Adapt to new JSON standards
  - JSON-Pointer (IETF RFC 6901)
  - JSON-Patch (IETF RFC 6902)
- Helper classes and methods to better utilize SE 8's stream operations

JSON-Pointer Possibilities

```
JsonArray contacts = ...;
JsonPointer pointer =
    Json.createPointer(
        "/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
- JsonObject and JsonArray are immutable
  - Utilize Builder pattern for editing API?

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

JSON-Patch Possibilities

```
[
  {
    "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"}}
]
```

JSON Query using Lambda Operations

```
JsonArray contacts = ...;
List<String> femaleNames =
    contacts.getValuesAs(JsonObject.class).stream()
        .filter(x->"Female".equals(x.getString("gender")))
        .map(x->(x.getString("name"))
        .collect(Collectors.toList());
```

JSON Query Collecting Results in JsonArray

```
JsonArray contacts = ...;
JsonArray femaleNames =
    contacts.getValuesAs(JsonObject.class).stream()
        .filter(x->"Female".equals(x.getString("gender")))
        .map(x->(x.getString("name"))
        .collect(JsonCollectors.toJsonArray());
```

Server-Sent Events (SSE)

- Lesser known part of HTML 5
  - Standard JavaScript API on the browser
- Server-to-client streaming
  - “Stock tickers”, monitoring applications
- Just plain long-lived HTTP
  - Between the extremes of vanilla request/response and WebSocket
  - text/event-stream
- Support via JAX-RS.next()
  - Already supported in Jersey JAX-RS reference implementation

SSE on the Server-Side

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

  @GET @Produces("text/event-stream")
  public EventOutput getQuotes() {
    EventOutput output = new EventOutput();

    executor.execute(() -> {
      ...
      output.send(new StockQuote(...));
      ...
    });

    return output;
  }
}
```

SSE on the Client-Side

```
WebTarget target = client.target("http://example.com/tickers");

EventSource eventSource = new EventSource(target) {
  @Override
  public void onEvent(InboundEvent inboundEvent) {
    StockQuote quote = inboundEvent.readData(StockQuote.class);
    ...
  }
};

eventSource.open();
```

MVC

- Standard action-based web framework for Java EE
  - First class peer to JSF, JSF to continue on it's evolution path
- Model
  - CDI, Bean Validation, JPA
- View
  - Facelets, JSP
- Controller
  - Majority of work here

MVC Possibilities

```
@Path("/")
@Template("my-index.xhtml")
public class Bookstore {
   ...
   @GET
   public List<Item> getItems() {
      ...
      return items;
   }
}
```

Java EE Security

- Simplify security for Java EE and improve portability
- Simple security providers
- Simple pluggability and mappings
- Enabling existing security annotations (@RolesAllowed) for all beans
- EL enabled security annotations via interceptors
- Password aliasing
- OAuth/OpenID

Simple Security Providers

```
@EmbedddedSecurityProvider({
    @Credentials(username="reza", password="secret", roles="dad"),
    @Credentials(username="nicole", password="secret", roles="mom"),
    @Credentials(username="zehra", password="secret", roles="daughter"),
    @Credentials(username="xavier", password="secret", roles="son")})

@DatabaseSecurityProvider(
    lookup="java:global/MyDB",
    userQuery="SELECT password FROM principals WHERE username=?",
    rolesQuery="SELECT role FROM roles where username=?", ...)

@LdapSecurityProvider(url="...", dnPrefix="...", dnSuffix="...", ...)
```

EL Enabled Security Annotations

```
@IsAuthorized("hasRoles('Manager') && schedule.officeHours")
public void transferFunds();

@IsAuthorized(
    "hasRoles('Manager') && hasAttribute('directReports', employeeId)")
public double getSalary(long employeeId);

@IsAuthorized(ruleSourceName="java:app/payrollAuthRules")
public void displayReport();
```

Simple Security Pluggability

```
@SecurityProvider
public class MySecurityProvider {

  @Inject UserService userService;

  @OnAuthentication
  // The parameters should suit the credentials mechanism being
  // used.
  public Principal getPrincipal(
      String username, String password) {
    // Construct the principal using the user service.
  }

  @OnAuthorization
  public String[] getRoles (Principal principal) {
    // Construct an array of roles using the principal and user
    // service.
  }
}
```

Password Aliasing

```
@DataSourceDefinition(
    name="java:app/MyDataSource",
    className="com.example.MyDataSource",
    ...
    user="duke",
    password="${ALIAS=dukePassword}")
```

JMS 2.1

- Essentially continuation of JMS 2
- Declarative message listeners
  - Alternative to MDB
  - More powerful features
  - Available to all beans
- Improving JMS provider portability
- Minor features and corrections
  - Redelivery delay, redelivery limits, dead message queues

Declarative JMS Listeners

```
@ApplicationScoped
@MaxConcurrency(10)
public class HandlingEventRegistrationAttemptConsumer {
  @JmsListener(
     destinationLookup="jms/HandlingEventRegistrationAttemptQueue",
     selector="source = 'mobile'",
     batchSize=10, retry=5, retryDelay=7000,
     orderBy=TIMESTAMP)
  @Transactional
  public void onEventRegistrationAttempt(
      HandlingEventRegistrationAttempt... attempts) {
    ...
  }
}
```

CDI 2

- Java SE Bootstrap
- XML configuration
- Asynchronous events
- @Startup for CDI beans
- Portable Extension SPI simplification
- Small features and enhancements

Asynchronous CDI Events?

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

```
public void onCargoInspected(
     @Observes(async=true) @CargoInspected Cargo cargo) {
```

Java EE Management 2

- Revamp of dated JSR 77 (J2EE Management)
- REST/SSE instead of EJB 2.x remoting
- Not just management/monitoring but deployment as well

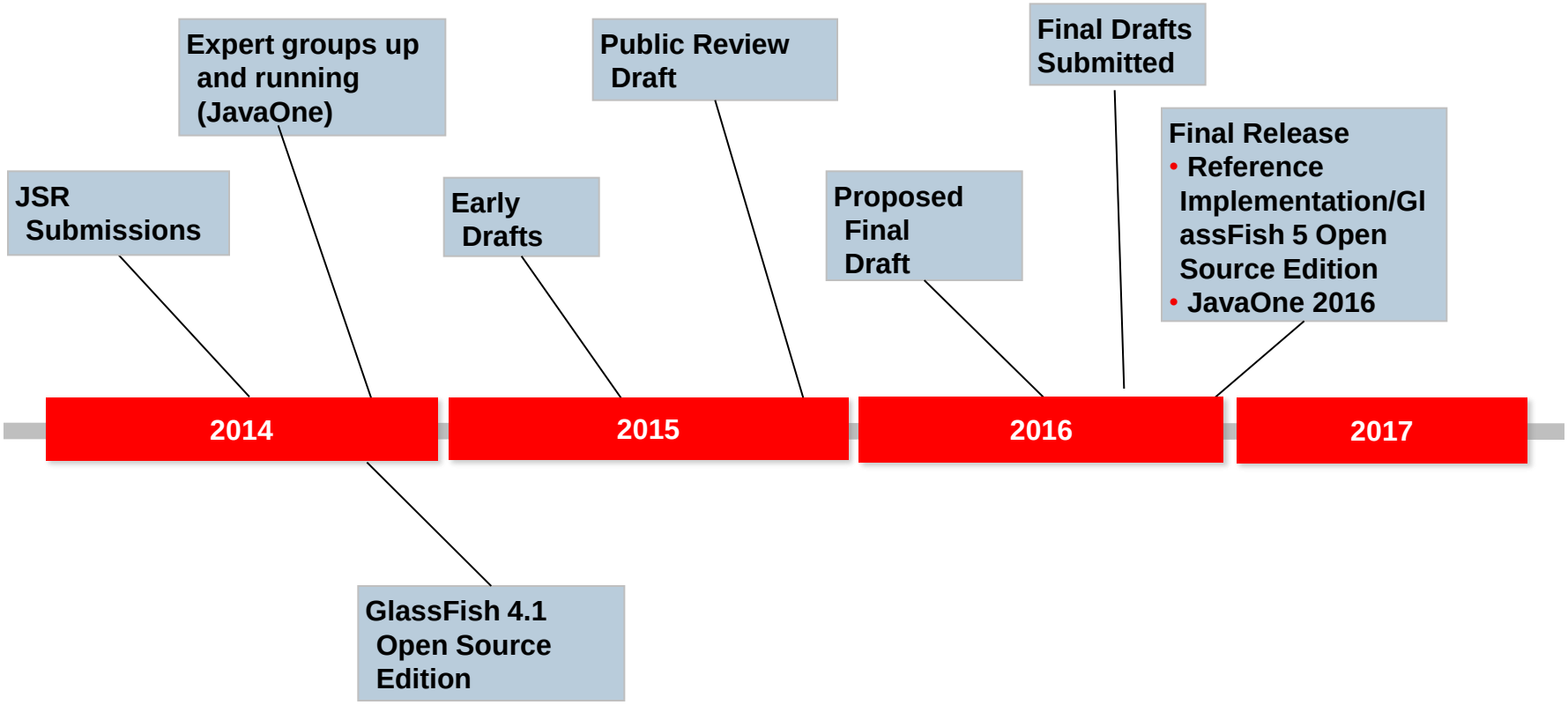
Adopting Java SE 8

- Most of Java SE 8 can already be used with Java EE
  - GlassFish, WildFly and WebLogic support JDK 8
- Some APIs could adopt features
  - Repeatable Annotations
  - Date-Time API/JDBC 4.2
  - Completable Future
  - Lambda expressions
  - Default methods

Others

- Reactive programming with JAX-RS client API
- Non-blocking I/O in JAX-RS
- More hypermedia support in JAX-RS
- Improving CDI integration with JAX-RS
- Improving CDI integration with WebSocket
- Ability to use @Schedule outside EJB
- Prune EJB 2 interfaces
- Prune CORBA interoperability

Java EE/GlassFish Roadmap



Adopt-a-JSR for Java EE 8

- Grassroots participation to shape Java EE
- Launched in Java EE 7 time-frame, key community element for Java EE 8
  - 19 Java user groups adopted a Java EE 7 JSR!

http://glassfish.org/adoptajsr

Java EE 8 JSRs Already Adopted!

User Group	Java EE 8.0	CDI 2.0	JSON-B 1.0	JMS 2.1	Servlet 4.0	JAX-RS 2.1	MVC 1.0	JSF 2.3
London Java Community	✓	✓	✓	✓		✓	✓	
Morocco JUG		✓	✓		✓	✓	✓	
Egypt JUG	✓		✓	✓			✓	✓
Hellenic Java User Group	✓						✓	
Santa Catarina Java User Group		✓			✓	✓	✓	
Japan Java User Group							✓	
JUG Chennai/Hyderabad		✓					✓	

Learning More

- Java EE 7 Transparent Expert Groups
  - http://javaee-spec.java.net
- Java EE 7 Reference Implementation
  - http://glassfish.org
- The Aquarium
  - http://blogs.oracle.com/theaquarium


Java™