
ORACLE®

Java Day™

ORACLE®

Java EE 8

Linda DeMichiel
Java EE Specification Lead
Oracle

Java Day Tokyo 2015
April 8, 2015

Java Day ORACLE

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.

Program Agenda

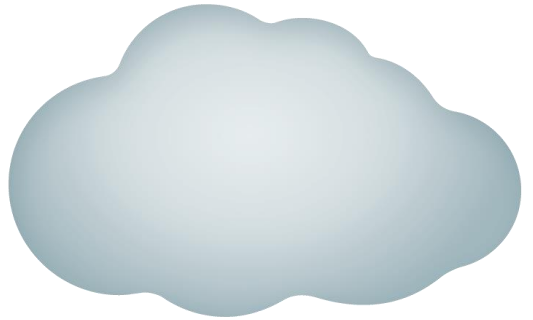
Overview of Java EE 8

1. How did we decide on content and goals of Java EE 8 ?
2. What are we planning?
3. How can you get involved?

Industry Trends We're Seeing

Cloud

HTTP/2



Mobile

User Experience

Reactive Programming

SECURITY

Java Day ORACLE

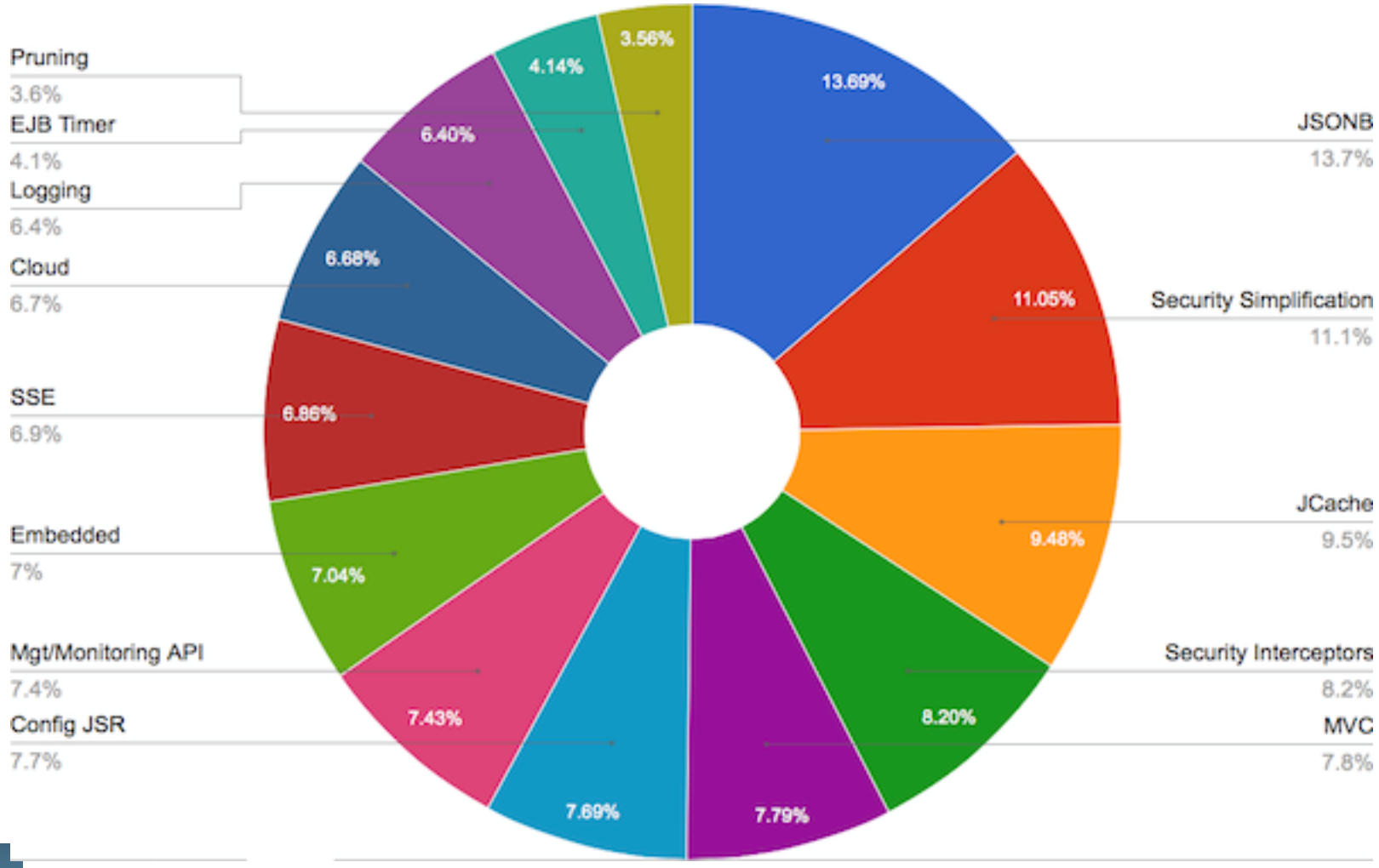
Feedback from the Community

- Many sources of input
  - Users lists of java.net projects
  - JIRAs
  - JavaOne 2013 Java EE BOF and Java EE EG meeting
  - Outreach by evangelists
- Consolidated into Community Survey

Java EE 8 Community Survey

- 3 parts over 3½ months
  - 47 questions
  - 15 fill-ins
  - 1000's of comments
- 4500+ respondents
- Prioritization of most-popular features
- https://java.net/projects/javaee-spec/downloads/download/JavaEE8_Community_Survey_Results.pdf

Community-Prioritized Features



Java EE 8 Themes

- HTML5 / Web Tier Enhancements
- Ease of Development / CDI alignment
- Infrastructure for running in the Cloud

HTML5 Support / Web Tier Enhancements

- JSON Binding
- JSON Processing enhancements
- Server-sent events
- Action-based MVC
- HTTP/2 support

JSON-B 1.0

Java API for JSON Binding

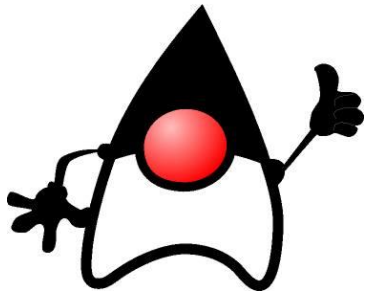
- API to marshal/unmarshal Java objects to/from JSON
  - Similar to JAXB runtime API in XML world
- Default mapping of classes to JSON
  - Annotations to customize the default mappings
  - JsonProperty, JsonTransient, JsonNillable, JsonValue, …
- Draw from best practices of existing JSON binding implementations
  - MOXy, Jackson, GSON, Genson, Xstream, …
  - Allow switch of JSON binding providers

JSON-B

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

Person duke = new Person();
duke.setName("Duke");
duke.setGender("M");
phones = new HashMap<String,String>();
phones.put("home", "650-123-4567");
phones.put("mobile", "650-234-5678");
duke.setPhones(phones);
```

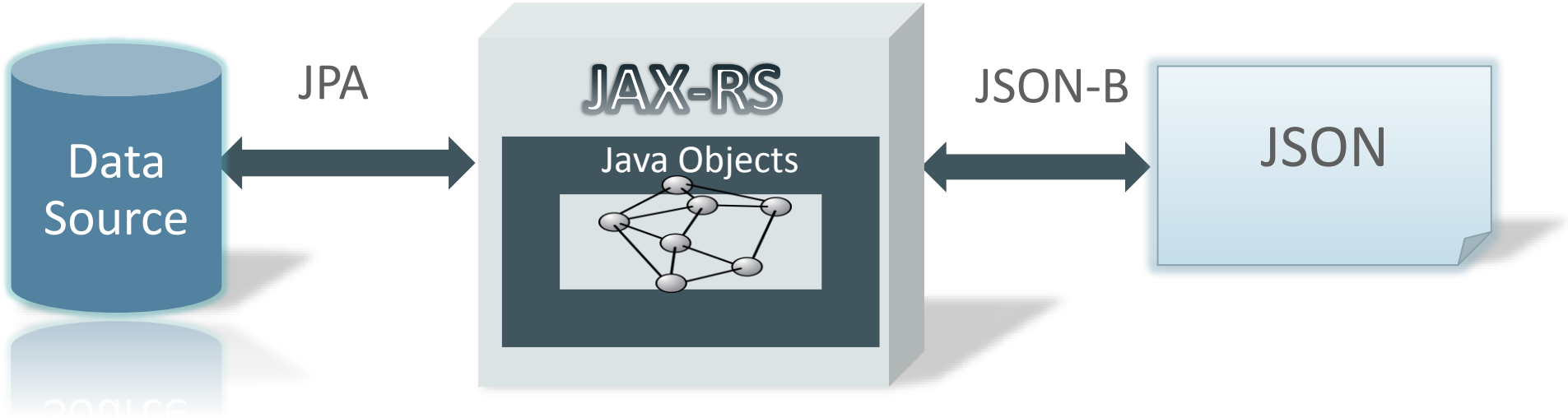
```
{
  "name":"Duke",
  "gender":"M",
  "phones":{
     "home":"650-123-4567",
     "mobile":"650-234-5678"}
}
```



```
Marshaller marshaller = new JsonContext().createMarshaller().setPrettyPrinting(true);
marshaller.marshal(duke, System.out);
```

JSON-B

- All the way from client to database
  - JSON-B will provide JAX-RS a standard way to support "application/json" media type



JSON-P 1.1

Java API for JSON Processing

- Keep JSON-P spec up-to-date
- Track new standards
- Add editing operations to JsonObject and JsonArray
- Helper classes and methods to better utilize SE 8’s stream operations

JSON-P

Tracking new standards

- JSON-Pointer – IETF RFC 6901
  - String syntax for referencing a value

    "/0/phones/mobile"

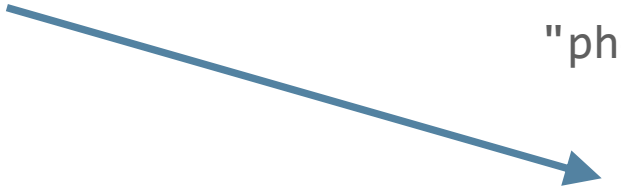
JSON-P

```
JsonArray contacts = Json.createArrayBuilder()
  .add(Json.createObjectBuilder()
     .add("name", "Duke")
     .add("gender", "M")
     .add("phones", Json.createObjectBuilder()
        .add("home", "650-123-4567")
        .add("mobile", "650-234-5678")))
  .add(Json.createObjectBuilder()
     .add("name", "Jane")
     .add("gender", "F")
     .add("phones", Json.createObjectBuilder()
        .add("mobile", "707-555-9999")))
  .build();
```

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

JSON-P

```
JsonArray contacts = ...;
JsonPointer p =
    Json.createPointer("/0/phones/mobile");
JsonValue v = p.getValue(contacts);
```



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

JSON-P

Tracking new standards

- JSON-Patch – IETF RFC 6902
- Patch is a JSON document
  - Array of objects / operations for modifying a JSON document
  - add, replace, remove, move, copy, test
  - Must have "op" field and "path" field

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

JSON-P

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

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

JSON-P

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

```
[
 {
    "name":"Duke",
    "gender":"M",
    "phones":{
       "home":"650-123-4567",
       "mobile":"650-111-2222"}},
  {
    "name":"Jane",
    "gender":"F",
    "phones":{
       "mobile":"707-555-9999"}}
]
```

JSON-P

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

```
[
 {
    "name":"Duke",
    "gender":"M",
    "phones":{
       "home":"650-123-4567",
       "mobile":"650-111-2222"}}
]
```

JSON-P

JSON Patch

- Issue: JsonObject and JsonArray are immutable
- Need editing capability to implement JSON patch
- Possible approach: use builder pattern
  - Builder creates mutable object for for temporary editing
  - Convert to immutable object when done

JSON-P

JSON Query using Lambda Operations

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

JSON-P

JSON query collecting results in JsonArray

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

Server-sent Events

- Part of HTML5 standardization
- Server-to-client streaming of text data
- Mime type is text/event-stream
- Long-lived HTTP connection
  - Client establishes connection
  - Server pushes update notifications to client
  - Commonly used for one-way transmission for period updates or updates due to events

Server-sent Events

- We evaluated several possibilities: Servlet; WebSocket; JAX-RS; standalone
  - And we polled the experts
- JAX-RS deemed most natural fit
  - Streaming HTTP resources already supported
  - Small extension
    - Server API: new media type; EventOutput
    - Client API: new handler for server side events
  - Convenience of mixing with other HTTP operations; new media type
  - Jersey (JAX-RS RI) already supports SSE

Server-sent events

JAX-RS resource class

```
@Path("tickers")
public class StockTicker {
    @Get @Produces("text/event-stream")
    public EventOutput getQuotes() {
        EventOutput eo = new EventOutput();
        new StockThread(eo).start()
        return eo;
    }
}
```

Server-sent events

JAX-RS StockThread class

```
class StockThread extends Thread {
  private EventOutput eo;
  private AtomicBoolean ab =
     new AtomicBoolean(true);

  public StockThread(EventOutput eo) {
     this.eo = eo;
  }
  public void terminate() {
     ab.set(false);
  }
```

```
@Override
public void run() {
  while (ab.get()) {
    try {
        // ...
      eo.send(new StockQuote("..."));
        // ...
     } catch (IOException e) {
        // ...
     }
   }
  }
}
```

Server-sent events

JAX-RS Client

```
WebTarget target = client.target("http://example.com/tickers");
EventSource eventSource = new EventSource(target) {
    @Override
    public void onEvent(InboundEvent inboundEvent) {
      StockQuote sq = inboundEvent.readData(StockQuote.class);
      // ...
     }
  };
eventSource.open();
```

Model View Controller (MVC)

2 Main Styles

- Component-based MVC
  - Style made popular by component frameworks
  - Controller provided by the framework
  - JSF, Wicket, Tapestry…
- Action-based MVC
  - Controllers defined by the application
  - Struts 2, Spring MVC…

MVC 1.0

- Action-based model-view-controller architecture
- Glues together key Java EE technologies:
  - Model
    - CDI, Bean Validation, JPA
  - View
    - Facelets, JSP
  - Controller
    - JAX-RS resource methods

MVC

JAX-RS controller

```
@Path("hello")
public class HelloController {
   @Inject
   private Greeting greeting;

   @GET
   @Controller
   public String hello() {
     greeting.setMessage("Hello there!");
     return "hello.jsp";
   }
}
```

MVC

Model

```
@Named("greeting")
@RequestScoped
public class Greeting{
   private String message;

   public String getMessage() { return message; }

   public void setMessage(String message) {
      this.message = message;
   }
}
```

MVC

JAX-RS controller

```
@Path("hello")
public class HelloController {
    @Inject
    private Greeting greeting;

    @GET
    @Controller
    public String hello() {
      greeting.setMessage("Hello there!");
      return "hello.jsp";
    }
}
```

MVC

View

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
   <title>Hello</title>
</head>
<body>
<h1>${greeting.message}</h1?
</body>
</html>
```

HTTP/2

Address the Limitations of HTTP 1.x

HTTP/2 standard now formally approved by IETF

- Reduce latency
- Address the HOL blocking problem
- Support parallelism (without requiring multiple connections)
- Retain semantics of HTTP 1.1
- Define interaction with HTTP 1.x

HTTP/2

- Request/Response multiplexing over single connection
  - Fully bidirectional
  - Multiple streams
- Stream Prioritization
- Server Push
- Binary Framing
- Header Compression
- Upgrade from HTTP 1.1

Servlet 4.0

HTTP/2 Features in Servlet API

- Request/response multiplexing
  - Servlet Request as HTTP/2 message
- Stream prioritization
  - Add stream priority to HttpServletRequest
- Server push
- Binary framing
  - Hidden from API
- Upgrade from HTTP 1.1

Ease of Development / CDI Alignment

- Security interceptors
- Simplified messaging through CDI-based “MDBs”
- JAX-RS injection alignment
- WebSocket scopes
- Pruning of EJB 2.x client view and IIOP interoperability

Java EE Security 1.0

Authorization via CDI Interceptors

```
@IsAuthorized("hasRoles('Manager') && schedule.officeHrs")
void transferFunds()

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

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

JMS 2.1

New API to receive messages asynchronously

- Alternative to EJB message-driven beans
- Simpler JMS-specific annotations
- Usable by any CDI bean
- No need for MessageListener implementation

JMS MDBs Today

EJB 3.2, JMS 2.0

```
@MessageDriven(activationConfig = {
  @ActivationConfigProperty(propertyName="connectionFactoryLookup", propertyValue="jms/myCF"),
  @ActivationConfigProperty(propertyName="destinationLookup", propertyValue="jms/myQueue"),
  @ActivationConfigProperty(propertyName="destinationType", propertyValue="javax.jms.queue")})

public class MyMDB implements MessageListener {
   public void onMessage(Message message) {
       // extract message body
       String body = message.getBody(String.class));
       // process message body
   }
}
```

Any Java EE bean as a listener

JMS 2.1

```
@RequestScoped
public class MyListenerBean {
   @JMSListener(destinationLookup="jms/myQueue")
   @Transactional
   public void myCallback(Message message) {
   ...
   }
}
```

Pruning

EJB 2.x Client View

```
public interface PayrollHome
   extends javax.ejb.EJBLocalHome {
     public Payroll create()
        throws CreateException;
    ...
   }
public interface Payroll
    extends javax.ejb.EJBLocalObject {
     public double getSalary(int empId);
    ...
   }
```

```
public interface Payroll {
    public double getSalary(int EmpId);
    ...
}
```

Modernize the Infrastructure

For On-Premise and for in the Cloud

- Java EE Management 2.0
  - REST-based APIs for Management and Deployment
- Java EE Security 1.0
  - Authorization
  - Password Aliasing
  - User Management
  - Role Mapping
  - Authentication
  - REST Authentication

Java EE Management 2.0

- Update to JSR 77 (J2EE Management)
- REST-based interfaces to augment (or replace) current Management EJB APIs
  - Currently used OBJECT_NAME to become URL
  - Define CRUD operations over individual managed objects
  - Server-sent events used for event support
- Simple deployment interfaces also to be considered as part of management API

Java EE Security 1.0

Candidate Areas to Enhance Portability, Flexibility, Ease-of-Use

- Password Aliasing
- User Management
- Role Mapping
- Authentication
- REST Authentication
- Authorization

Java EE Security 1.0

Password Aliasing

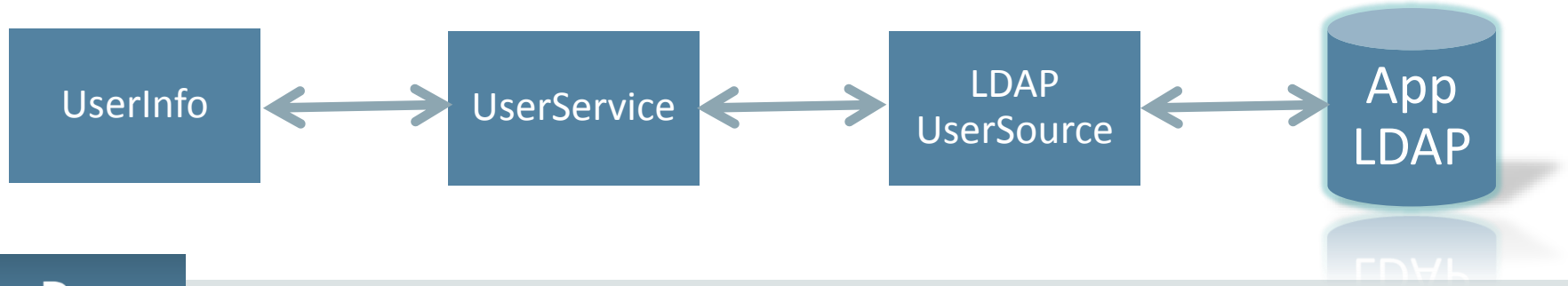
- Standardized syntax for password aliases
  - Avoids storing passwords in clear text in code, deployment descriptors, files
-
```
@DataSourceDefinition(
    name="java:app/MyDataSource",
    className="com.example.MyDataSource",
    ...
    user="duke",
    password="${ALIAS=dukePassword}")
```

- Standardized secure credentials archive for bundling alias and password with app
  - Used by platform as credential store for resolving alias

Java EE Security 1.0

User Management

- Allow application to manage its own users and groups
  - Without need to access server configuration
- Users stored in application-specified repository (e.g., LDAP)
- User service manipulates users from user source



Java EE Security 1.0

User Management

- UserSourceDefinition
- UserService
  - Create/delete users, create/delete groups, add user to group, load UserInfo by user name; etc…
- UserInfo
  - get user name, password, get user's roles, get user's attributes, …

Java EE Security 1.0

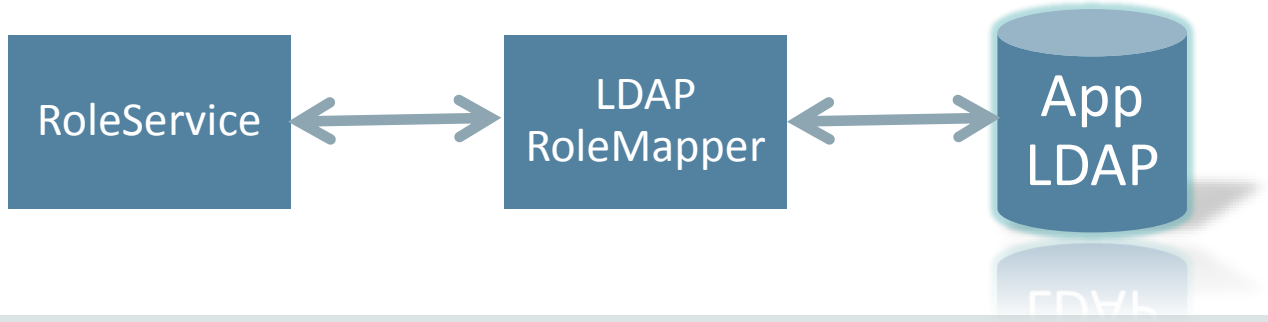
User Management

```
@LdapUserSourceDefinition(
  name="java:app/ldapUserSource",
  ldapUrl="ldap://someURL",
  ldapUser="ElDap",
  ldapPassword="${ALIAS=LdapPW}",
  ...
)
public class MyAuthenticator {
  @Resource(lookup="java:app/ldapUserSource")
  private UserService userService;
  private boolean isAccountEnabled(String username) {
    return userService.loadUserByUsername(username).isEnabled();
  }
  ...
}
```

Java EE Security 1.0

Role Mapping

- Standardize role service
  - Role mappings can be stored in app-specified repository (e.g., LDAP)
  - Application can assign roles to users and groups, based on application-specific model
  - Without need to access server configuration



Java EE Security 1.0

Role Mapping

- RoleMapperDefinition
  - DataSource, Ldap, Memory/File, Custom, predefined
- RoleService
  - grant/revoke roles for user/group, get roles for user/group, ...

```
@Resource(lookup="java:app/devRoleMapper")
RoleService roleService;
List<String> getRoles(String username) {
   return roleService.getRolesForUser(username);
}
```

Java EE 8 JSRs

- Java EE 8 Platform (JSR 366)
- CDI 2.0 (JSR 365)
- JSON Binding 1.0 (JSR 367)
- JMS 2.1 (JSR 368)
- Java Servlet 4.0 (JSR 369)
- JAX-RS 2.1 (JSR 370)
- MVC 1.0 (JSR 371)
- JSF 2.3 (JSR 372)
- Java EE Management 2.0 (JSR 373)
- JSON-P 1.1 (JSR 374)
- Java EE Security 1.0 (JSR 375)
- … and more to follow …

Expected MRs and small JSRs

- EL
- Concurrency Utilities
- Connector Architecture
- WebSocket
- Interceptors
- JPA
- EJB
- JTA
- JCache
- Bean Validation
- Batch
- JavaMail
- … and more to follow …

Transparency

Commitment to JCP transparent processes

- Our Java EE 8 JSRs run in the open on java.net
  - http://javaee-spec.java.net
  - One project per JSR – jax-rs-spec, mvc-spec, servlet-spec,…
- Publically viewable Expert Group mail archive
  - Users observer lists gets all copies
- Publicly accessible download area
- Publicly accessible issue tracker / JIRA
- …

How to Get Involved

- Adopt a JSR
  - http://glassfish.org/adoptajsr
- Join an Expert Group project
  - http://javaee-spec.java.net
  - https://java.net/projects/javaee-spec/pages/Specifications
- The Aquarium
  - http://blogs.oracle.com/theaquarium
- Java EE 8 Reference Implementation
  - http://glassfish.org

Safe Harbor Statement

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.

Hardware and Software
Engineered to Work Together

Java Day™

ORACLE®


ORACLE®