
ORACLE®

ORACLE

JavaEE.Next(): Java EE 7, 8, and Beyond

Reza Rahman
Java EE/GlassFish Evangelist

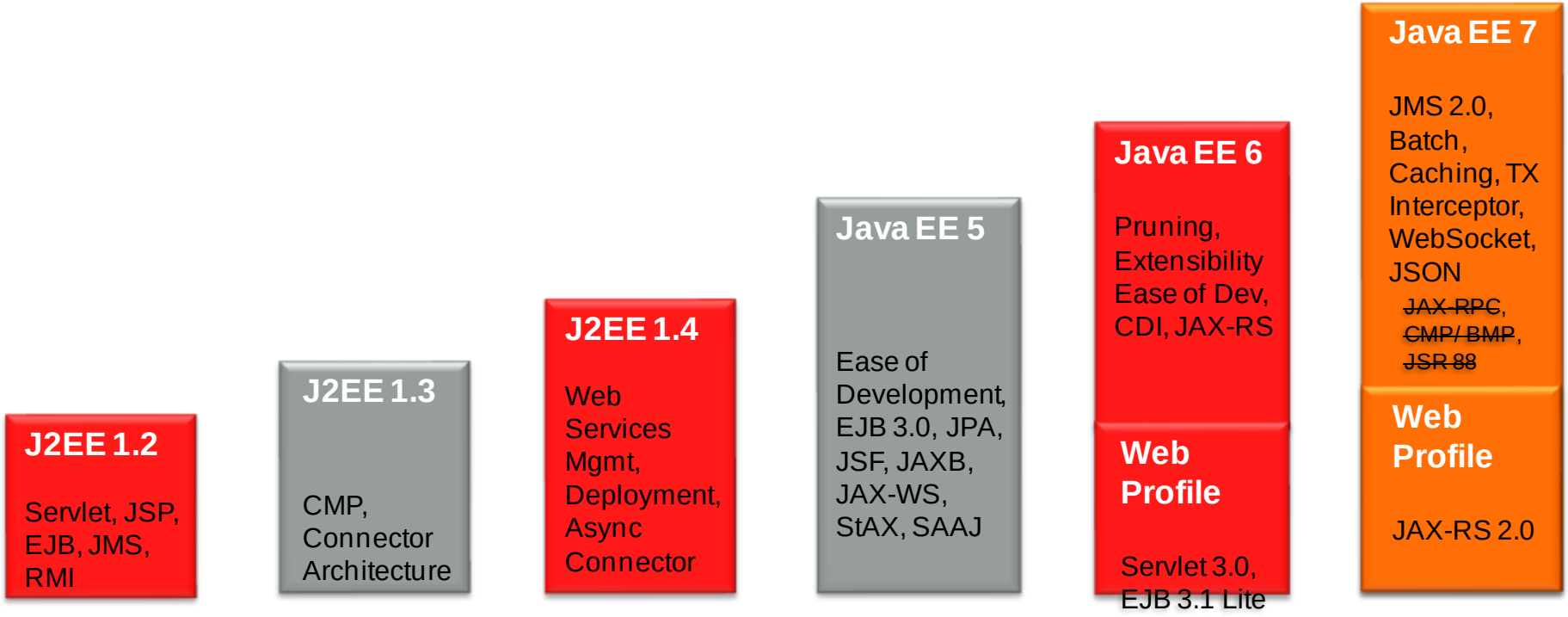
JavaDAYS

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.

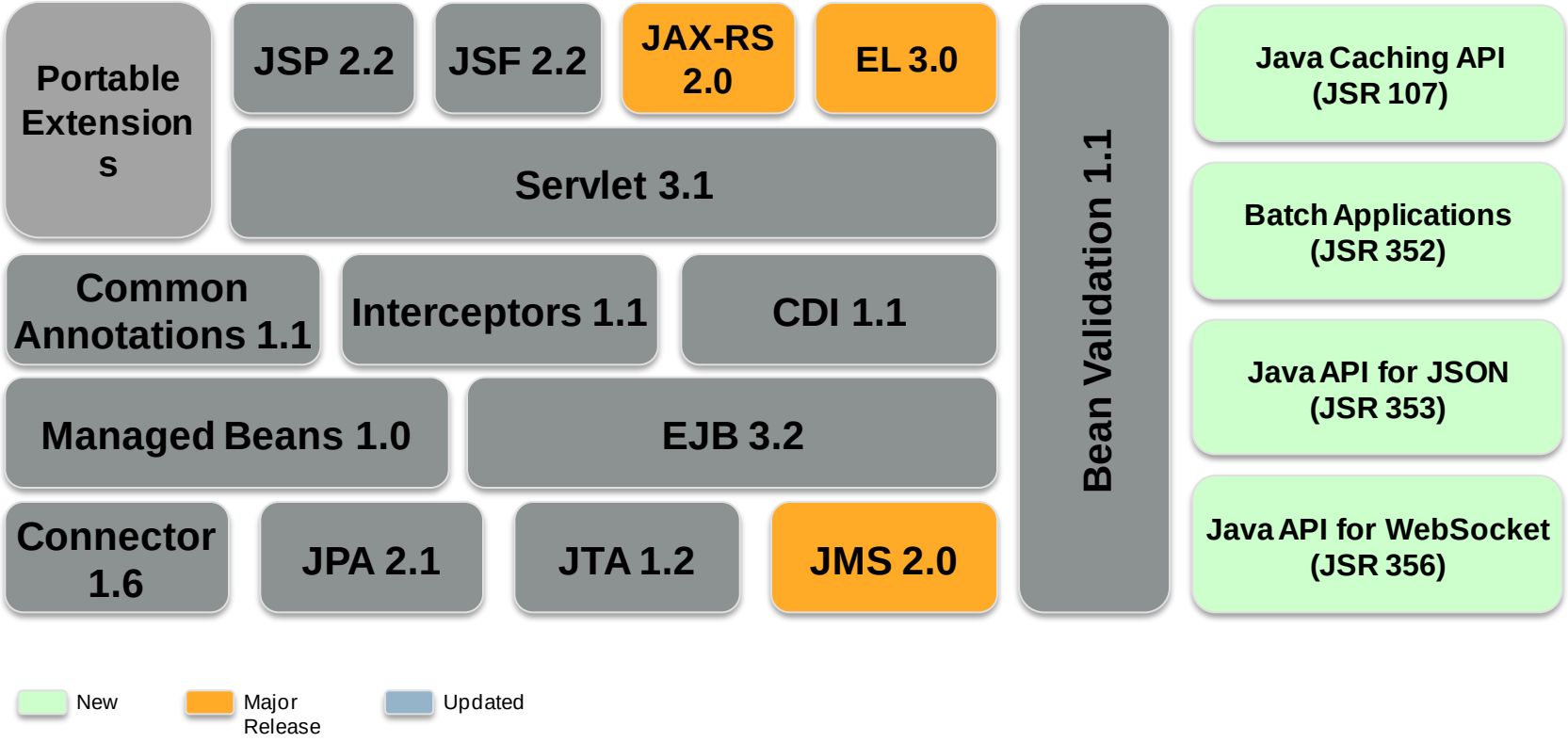
Program Agenda

- Overview
- A Taste of API Changes
- Your Feedback on Open Issues!

Java EE Past, Present, & Future



Java EE 7 – Candidate JSRs



JMS 2

- API modernization using dependency injection
- JCA adapter required
- Delivery delay, async send, delivery count, MDB alignment
- Fixes, clarifications

JMS 2

Old API

```
@Resource(lookup = "java:global/jms/demoConnectionFactory")
ConnectionFactory connectionFactory;
@Resource(lookup = "java:global/jms/demoQueue")
Queue demoQueue;

public void sendMessage(String payload) {
  try {
    Connection connection = connectionFactory.createConnection();
      try {
        Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
        MessageProducer messageProducer = session.createProducer(demoQueue);
        TextMessage textMessage = session.createTextMessage(payload);
        messageProducer.send(textMessage);
      } finally {
        connection.close();
      }
  } catch (JMSException ex) {
    Logger.getLogger(getClass().getName()).log(Level.SEVERE, null, ex);
  }
}
```


JMS 2

Simplified API

```
@Inject
private JMSContext context;

@Resource(mappedName = "jms/inboundQueue")
private Queue inboundQueue;

public void sendMessage (String payload) {
  context.createProducer().send(inboundQueue, payload);
}
```

JMS 2/Java EE 7

JMS Resource Definition

```
@JMSConnectionFactoryDefinition(
  name="java:global/jms/demoConnectionFactory",
  className= "javax.jms.ConnectionFactory",
  description="ConnectionFactory to use in demonstration")

@JMSDestinationDefinition(
  name = "java:global/jms/demoQueue",
  description = "Queue to use in demonstration",
  className = "javax.jms.Queue",
  destinationName="demoQueue")
```

Java API for WebSocket

- Higher level API for WebSocket
- Both client and server-side (Java SE and Java EE)
- Both declarative and programmatic

Java API for WebSocket

Connection Life Cycle

```
@WebSocketEndpoint(path="/chat")
public class ChatServer {
  Set<Session> peers = ...

  @WebSocketOpen
  public void onOpen(Session peer) {
    peers.add(session);
  }

  @WebSocketClose
  public void onClose(Session session) {
    peers.remove(session);
  }
  ...
```

Java API for WebSocket

WebSocket Communication

```
  ...
  @WebSocketMessage
  public void message(String message, Session client)
      throws IOException {
    for (Session session : peers) {
      if (!session.equals(client)) {
        session.getRemote().sendObject(message);
      }
    }
  }
}
```

Java API for JSON Processing

- API to parse, generate, transform, query JSON
- Object Model and Streaming API -- similar to DOM and StAX
- Binding JSON to Java objects forthcoming

Java API for JSON Processing

Building a JSON Object

```
"phoneNumber": [
  {
    "type": "home",
    "number": ”408-123-4567”
  },
  {
    "type": ”work",
    "number": ”408-987-6543”
  }
]
```

```
JsonObject jsonObject =
    new JsonBuilder()
  .beginArray("phoneNumber")
    .beginObject()
      .add("type", "home")
      .add("number", "408-123-4567")
    .endObject()
    .beginObject()
      .add("type", ”work")
      .add("number", "408-987-6543")
    .endObject()
  .endArray()
.build();
```

JAX-RS 2

- Client API
- Message Filters & Entity Interceptors
- Asynchronous Processing – Server & Client
- Hypermedia Support
- Common Configuration

JAX-RS 2

Client API

```
// Get instance of Client
Client client = ClientFactory.newClient();

// Get customer name for the shipped products
String name = client.target("../orders/{orderId}/customer")
                    .pathParam("orderId", "10")
                    .queryParam("shipped", "true")
                    .request()
                    .get(String.class);
```

Java Temporary Caching API

- Java caching standard
- Long awaited…
- Caching annotations

Java Temporary Caching API

Basic API

```
CacheManager cacheManager =
    CacheManagerFactory.getCacheManager();
Cache<String, Object> cache =
    cacheManager.getCache("testCache");
...
cache.put(key1, value1);
...
Object value2 = cache.get(key2);
...
cache.remove(key3);
```

Java Temporary Caching API

Caching Annotations

```
public class CustomerDao {
  @CachePut(cacheName="customers")
  public void addCustomer(
      @CacheKeyParam int cid,
      @CacheValue Customer customer) {
    . . .
  }
}
```

JPA 2.1

- Schema generation
- Stored procedures
- Unsynchronized persistence contexts
- Entity converters
- Fixes and enhancements

JPA 2.1

Schema Generation

```
@Table(indexes= {@Index(columnList="NAME")
                 @Index(columnList="DEPT_ID")})
@Entity public class Employee {
  @Id private Integer id;
  private String name;
  ...
  @ManyToOne
  private Department dept;
  …
}
```

JPA 2.1

Stored Procedures

```
@Entity
@NamedStoredProcedureQuery(name="topGiftsStoredProcedure”,
    procedureName="Top10Gifts")
public class Product {


StoredProcedreQuery query = EntityManager.createNamedStoredProcedureQuery(
    "topGiftsStoredProcedure");
query.registerStoredProcedureParameter(1, String.class, ParameterMode.INOUT);
query.setParameter(1, "top10");
query.registerStoredProcedureParameter(2, Integer.class, ParameterMode.IN);
query.setParameter(2, 100);
. . .
query.execute();
String response = query.getOutputParameterValue(1);
```

JTA 1.2

- Declarative transactions outside EJB
- Transaction scope - @TransactionScoped

JTA 1.2

@Transactional Annotation

```
@Inherited
@InterceptorBinding
@Target({TYPE, METHOD}) @Retention(RUNTIME)
public @interface Transactional {
  TxType value() default TxType.REQUIRED;
  Class[] rollbackOn() default {};
  Class[] dontRollbackOn() default {};
}

@Transactional(rollbackOn={SQLException.class},
    dontRollbackOn={SQLWarning.class})
public class UserService {...}
```

JSF 2.2

- HTML5 Support
- @FlowScoped
- @ViewScoped for CDI
- Managed beans deprecated/CDI alignment
- File upload component
- View actions
- Multi-templating
- Security
- Fixes and enhancements

JSF 2.2

Pass-Through HTML 5 Components

```
<html>
   ...
   <input type="color" jsf:value="#{colorBean.color2}" />
   <input type="date" jsf:value="#{calendarBean.date1}" />
   ...
</html>
```

JSF 2.2

Faces Flow

```
@Named
@FlowScoped(id="flow-a")
public class Flow_a_Bean implements Serializable {
    public String getName() {
        return "Flow_a_Bean";
    }
    public String getReturnValue() {
        return "/return1";
    }
    @Produces
    Flow getFlow(FlowBuilder builder) {
         builder.startNode("router1");
         builder.flowReturn("success").fromOutcome("/complete");
         builder.flowReturn("errorOccurred").fromOutcome("error");
         builder.switchNode("router1").
             navigationCase().condition("#{facesFlowScope.customerId == null}").
             fromOutcome("create-customer").
             defaultOutcome("view-customer");
         builder.viewNode("create-customer");
         builder.viewNode("maintain-customer-record");
         builder.methodCall("upgrade-customer").
           method("#{maintainCustomerBean.upgradeCustomer}").defaultOutcome("view-customer");
         builder.initializer("#{maintainCustomerBean.initializeFlow}");
         builder.finailzer("#{maintainCustomerBean.cleanUpFlow}");
         return builder.getFlow();
    }
}
```

JSF 2.2

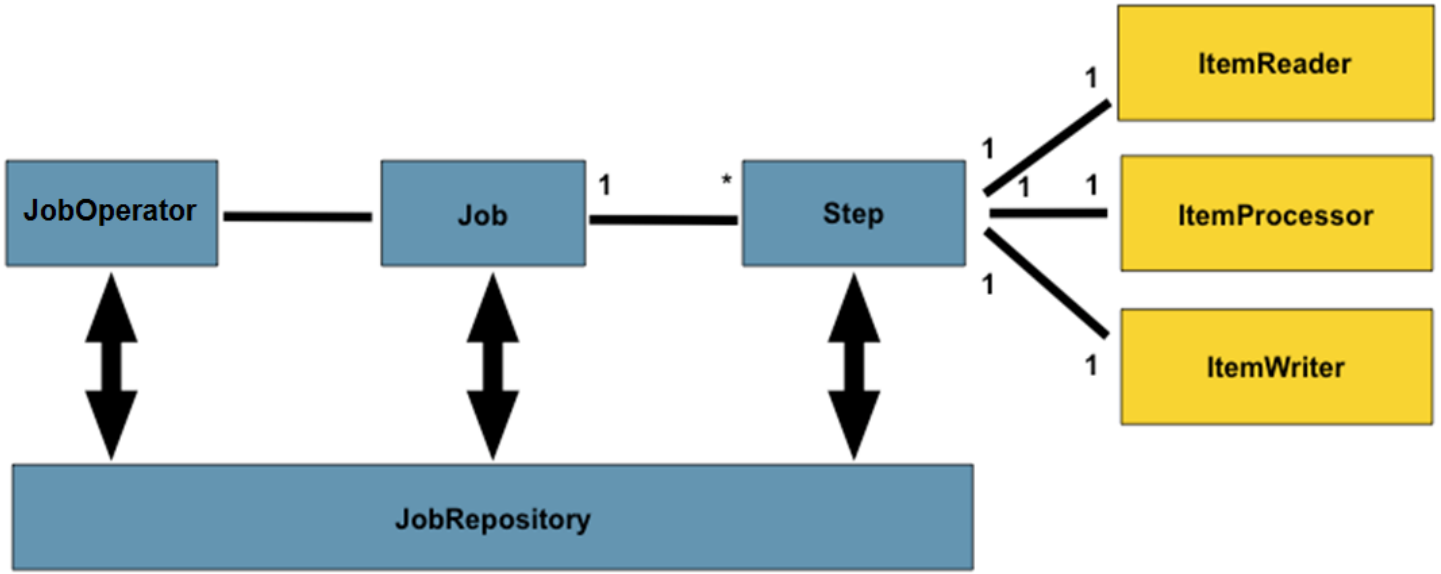
File Upload Component

```
<h:inputFile id="file" value="#{fileUploadBean.uploadedFile}">
  <f:validator validatorId="FileValidator" />
</h:inputFile>

@Named @RequestScoped
public class FileUploadBean {
  private Part uploadedFile; // getter/setter
  public String getFileText() {
    String text = "";
    if (null != uploadedFile) { try {
      InputStream is = uploadedFile.getInputStream();
      text = new Scanner( is ).useDelimiter("\\A").next();
    } catch (IOException ex) {} }
    return text;
  }
}
```

Batch Applications for the Java Platform

- API for robust batch processing targeted to Java EE



Batch Applications for the Java Platform

Step Example

```
<step id="sendStatements">
  <chunk reader="AccountReader"
     processor="AccountProcessor"
     writer="EmailWriter"
     chunk-size="10" />
</step>
```

```
@ReadItem
public Account readAccount() {
  // read account using JPA
}
```

```
@ProcessItem
public Account processAccount(Account account) {
  // calculate balance
}
```

```
@WriteItems
public void sendEmail(List<Account> accounts) {
  // use JavaMail to send email
}
```

Bean Validation 1.1

- Method constraints
- Bean Validation artifacts injectable
- Fixes, clarifications and enhancements

Bean Validation 1.1

Method Level Constraints

```
public void placeOrder(
     @NotNull String productName,
     @NotNull @Max("10") Integer quantity,
     @Customer String customer) {
   . . .
}

@Future
public Date getAppointment() {
   . . .
}
```

Others

- **Servlet 3.1:** Non-blocking I/O, upgrade to WebSocket, security…
- **CDI 1.1:** Ordering of interceptors, Servlet events, @Veto…
- **EL 3.0:** Lambda expressions, Collection, operators, standalone API…
- **EJB 3.2:** Truncating CMP/BMP…

Java EE 8

- Cloud, PaaS, multitenancy/SaaS
- Modularity
- HTML5
- More CDI/EJB alignment?
- NoSQL?

Addition of New JSRs

	Add to Java EE Full Platform?			Add to Web Profile?		
	Yes	No	Not Sure	Yes	No	Not Sure
WebSocket 1.0	☐	☐	☐	☐	☐	☐
JSON-P 1.0	☐	☐	☐	☐	☐	☐
Batch 1.0	☐	☐	☐	☐	☐	☐
JCache 1.0	☐	☐	☐	☐	☐	☐

How important is CDI?

- Essential?
- Very Important?
- Important?
- So – so?
- Not Important?

Enable CDI more broadly?

- Currently, CDI must be enabled on a per-jar basis with beans.xml
- Should we enable CDI by default in Java EE?

Yes	No	Not Sure
☐	☐	☐

CDI @Inject Use

- Should new Java EE technologies invent their own annotations for injection, or just use CDI?

```
@JobListener @Named("MyJobListener")
public class MyJobListenerImpl {
  @BatchContext JobContext jctxt;
}
```

```
@JobListener @Named("MyJobListener")
public class MyJobListenerImpl {
  @Inject @BatchContext JobContext jctxt;
}
```

Expansion of CDI Stereotypes

```
@Secure
@RolesAllowed("Administrator")
@Stereotype
@Target({TYPE, METHOD}) @Retention(RUNTIME)
public @interface Admin {
}

@Stateless
@Stereotype @Target({TYPE}) @Retention(RUNTIME)
@Admin
public @interface AdminBean {
}

@AdminBean
public class AdminService {
   ...
}
```

Stereotypes

- Should we expand the use of Stereotypes?

Yes	No	Not Sure
☐	☐	☐

Use of CDI Interceptors

- CDI interceptors and Java EE interceptors are currently supported on
  - CDI managed beans, @ManagedBean beans, Session Beans, MDBs
- Should we support interceptors on:
  - All Java EE components (e.g., JAX-RS Resource classes, Servlets, …)?
  - Other Java EE managed classes?
- *Is there any place that CDI injection is supported that should not support interceptors?*

Use of CDI Observers

- CDI observer methods are currently supported on
  - CDI managed beans, Session beans, @ManagedBean beans
- Should we support observer methods on:
  - All JavaEE components (e.g., JAX-RS Resource classes, Servlets, …)?
  - Other Java EE managed classes?
- Should we support observer methods wherever CDI injection is supported?

Use of EJB Timers

- EJB Timers are currently supported on
  - Stateless Session Beans, Singleton Session Beans, MDBs
- Should we support timers on:
  - CDI managed beans?
  - Other Java EE components (MDBs, Servlets, JAX-RS Resource classes, . . .) ?
  - Other Java EE managed classes?
- Should we support timers wherever observer methods are supported ?

Try it Out!

download.java.net/glassfish/4.0/promoted/

Resources

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

Hardware and Software

ORACLE

Engineered to Work Together


ORACLE®