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

1. EJB – History, Goals, Evolution
2. CDI – History, Goals, Evolution
3. Advantages/disadvantages – when to use EJB/CDI
4. Java EE – Managed Bean Alignment
5. What is our strategy for the future
EJB

Some Background and History

• EJB 1.0 (begun 1996)
  – Origins in TP monitors and component-based systems (e.g. Microsoft MTS)
  – Designed for remote access and coarse-grained components

• EJB 1.1 (J2EE 1.2)
  – First "real" EJB

• EJB 2.0 (J2EE 1.3)
  – Updated Entity Beans; EJB QL; Message-driven Beans; IIOP Interoperability

• EJB 2.1 (J2EE 1.4)
  – Web services support; Timer service
EJB

Some Background and History

- **EJB 3.0 (Java EE 5)**
  - First "modern" EJB – Theme is "Ease of Development"
  - Annotations; resource injection; simple interfaces; Java Persistence API; interceptors

- **EJB 3.1 (Java EE 6)**
  - No-interface view; asynchronous methods; singletons; EJB Lite; Embedded EJB
  - Interceptors separated into own spec; JPA separated into own JSR

- **EJB 3.2 (Java EE 7)**
  - EJB Entity Beans and JAX-RPC now optional
EJB Today

- EJBs are POJOs
- Container provides services for ease-of-development
  - Injection
  - Transactions (declarative with defaults or programmatic)
  - Security (declarative with defaults or programmatic)
  - Thread management; synchronization; async support
  - Timed notifications (declarative or programmatic)
  - Integration with JMS, Web Services, IIOP-based clients and services
Program Agenda

1. EJB – History, Goals, Evolution
2. CDI – History, Goals, Evolution
3. Advantages/disadvantages – when to use EJB/CDI
4. Java EE – Managed Bean Alignment
5. What is our strategy for the future
CDI

Some Background and History

• CDI 1.0 (Java EE 6)
  – Original goal: direct JSF and EJB integration
  – Dependency injection with typed injection points; typesafe resolution
  – Annotations, qualifiers, stereotypes for strong typing
  – Scopes, contexts for automatic bean lifecycle management
  – Producers
  – Interceptor bindings; decorators
  – Events and observers
  – Rich SPI for portable extensions
CDI

Some Background and History

• CDI 1.1 (Java EE 7)
  – Implicit bean archives
  – Globally enabled interceptors, decorators, alternatives (with @Priority)
  – JTA transactional interceptors; @AroundConstruct interceptors
  – Enhancements to SPI and portable extensions
CDI Today

• Managed Bean POJOs are central
  – @Named qualifier allows direct use in JSF, EL, ...

• Container provides services
  – Type-safe injection
  – Lifecycle management; contexts; scopes; lifecycle callbacks
  – Event firing and delivery
  – Producers, Alternatives to configure available beans
  – Extensible model and rich SPI
Program Agenda

1. EJB – History, Goals, Evolution
2. CDI – History, Goals, Evolution
3. Advantages/disadvantages – when to use EJB/CDI
4. Java EE – Managed Bean Alignment
5. What is our strategy for the future
CDI Advantages

What does CDI give you that EJB alone does not?

• Annotation-based programming model; stereotypes
• Type-safe injection, interceptors, decorators
• Context management, scopes, conversations
• Events and observers
• Producers and disposers
• Extensibility
  – Custom scopes; programmatically defined beans; etc.

→ Higher level of abstraction
CDI Advantages

Higher level of abstraction vs EJB

• Automatic context and lifecycle management \(\leftrightarrow\) remove/pooling
• Events+observers \(\leftrightarrow\) callbacks
• Conversations \(\leftrightarrow\)
• Metalevel programming / extensibility \(\leftrightarrow\)
EJB Advantages

What does EJB give you that CDI does not?

- Remote access
  - RMI / CORBA
  - Web Services
- MDBs / JMS
- Timers; scheduled events
- Asynchronous methods
- Security integration
- JPA integration; container-managed extended persistence context
- Locking for concurrent access
EJB Advantages

What else does EJB give you?

- EJBs (session beans) are CDI managed beans
  - They have all the benefits of CDI
  - Stateless session beans – @Dependent scope
  - Stateful session beans – any scope
  - Singleton session beans – @ApplicationScoped
  - However, need to use @Inject, not @EJB to acquire

- Converse is not true
  - CDI beans are not EJBs
  - However: We are moving EJB benefits into CDI
Program Agenda

1. EJB – History, Goals, Evolution
2. CDI – History, Goals, Evolution
3. Advantages/disadvantages – when to use EJB/CDI
4. Java EE – Managed Bean Alignment
5. What is our strategy for the future
CDI and EJB: Bridging the Gap in Java EE

- @ManagedBean
- Managed Bean spec, introduced in Java EE 6
  - Identified commonalities among different components as "managed beans"
  - Identified points for extension
- "Managed Bean Alignment" was an important theme of Java EE 7
  - Generalize use of injection, interceptors, new scopes
## Managed Beans – Java EE 6

<table>
<thead>
<tr>
<th>MATRIX1</th>
<th>Java EE 6</th>
<th>Contextual</th>
<th>Support CDI injection into them</th>
<th>Support Java EE resource injection into them</th>
<th>Injectable</th>
<th>Injectable with @Inject</th>
<th>Support CDI interceptors</th>
<th>Support EE interceptors</th>
<th>Support Decorators</th>
<th>Support Observer methods</th>
<th>Can fire events</th>
<th>Support Producer methods</th>
<th>Can have Qualifiers</th>
<th>Can have Scope</th>
<th>Support CDI Constructor injection</th>
<th>Support Timers</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDI managed beans (non-EJB)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Session beans (CDI-enabled behavior via @Inject)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Java EE @ManagedBean beans (CDI-enabled behavior via @Inject)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Session beans obtained via @EJB or JNDI lookup (with CDI enabled)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Java EE @ManagedBean beans obtained via @Resource or JNDI lookup (with CDI enabled)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>MDBs</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>JAX-RS Resource classes</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Servlets</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Servlet filters, listeners</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Other Java EE component classes (Table EE. 5-2)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>JPA entities</td>
<td>Y(4)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>JPA entity listeners</td>
<td>Y(4)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Interceptors</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Decorators</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>
CDI and EJB: Bridging the Gap in Java EE 7

Enabling CDI by Default

- CDI is enabled by default in "implicit bean archives"
- Use of CDI bean-defining annotations results in implicit bean archives
  - @SessionScoped, @Dependent, ...
  - Bean archives include library jars, EJB jars, WEB-INF classes, ...
  - No beans.xml required
CDI and EJB: Bridging the Gap in Java EE 7

Injection

• CDI injection applies to all Java EE components when CDI is enabled
• Java EE components support resource injection + CDI injection
  – Support for constructor injection added as well
• CDI beans support CDI injection + resource injection
• CDI producers can "transform" resource injection into CDI injection, making it strongly typed

```java
@Produces
@Resource(lookup="java:global/env/jdbc/CustomerDatasource")
@CustomerDatabase
DataSource customerDatabase;
```
CDI and EJB: Bridging the Gap in Java EE 7

Interceptors

• CDI interceptor-binding interceptors apply to all Java EE components when CDI is enabled
• EJBs support "EJB interceptors" + CDI interceptor-binding interceptors
• CDI beans support CDI interceptor-binding interceptors + "EJB interceptors"
• Java EE components also support both
JTA Transactional Interceptors

Generalization of Container-managed Transactions (Java EE 7)

@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 ShoppingCart {...}
Bean Validation Interceptors

Method-level Validation (Bean Validation 1.1, Java EE 7)

@Stateless
public class OrderService {
    ...
    @ValidOrder
    public Order placeOrder(
        @NotNull String productName,
        @Max(10) int quantity,
        @NotNull String customerName,
        @Address String customerAddress) {
        ...
    }
}
**CDI and EJB: Bridging the Gap in Java EE 7**

**Scopes**

- CDI Scopes are extensible
  - `@TransactionScope`
    - Defined in JTA 1.2
    - Used by JMS 2.0
  - `@FlowScoped (JSF 2.2)`
  - WebSocket expected to define scope for WebSocket endpoints in Java EE 8
## Managed Beans – Java EE 7

<table>
<thead>
<tr>
<th>MATRIX Java EE 7</th>
<th>Contextual Support CDI injection into them</th>
<th>Support Java EE resource injection into them</th>
<th>Injectable with @Inject</th>
<th>Injectable with @Resource, @EJB (EE-defined)</th>
<th>Support CDI interceptors</th>
<th>Support EE interceptors (J)</th>
<th>Support Decorators</th>
<th>Support Observer methods</th>
<th>Can fine events</th>
<th>Support Producer methods</th>
<th>Can have Qualifiers</th>
<th>Can have Scope</th>
<th>Support CDI Constructor injection</th>
<th>Support Timers</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDI managed beans (non-EJB)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Session beans (CDI-enabled behavior via @Inject)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java EE @ManagedBean beans (CDI-enabled behavior via @Inject)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Session beans obtained via @EJB or INJ lookup (with CDI enabled)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java EE @ManagedBeans beans obtained via @Resource or INJ lookup (with CDI enabled)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MDBs</td>
<td>Y</td>
<td>Y</td>
<td>Y(6)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JAX-RS Resource classes</td>
<td>Y</td>
<td>Y</td>
<td>Y(6)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Servlets</td>
<td>Y</td>
<td>Y</td>
<td>Y(6)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Servlet Filters, listeners</td>
<td>Y</td>
<td>Y</td>
<td>Y(6)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other Java EE component classes (Table EE. 5-1)</td>
<td>Y</td>
<td>Y</td>
<td>Y(6)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JPA entities</td>
<td>Y(6)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JPA entity listeners (4)</td>
<td>Y</td>
<td>Y(6)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interceptors</td>
<td>Y</td>
<td>Y</td>
<td>Y(6)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decorators</td>
<td>Y</td>
<td>Y</td>
<td>Y(6)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Producer methods</td>
<td>Y(3)</td>
<td>Y(5)</td>
<td>Y(5)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y(2)(8)</td>
<td>Y(2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Producer fields</td>
<td>Y(2)</td>
<td>Y(2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disposer methods</td>
<td>Y(3)</td>
<td>Y(5)</td>
<td>Y(5)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y(3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Observer methods</td>
<td>Y(3)</td>
<td>Y(5)</td>
<td>Y(5)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y(3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Program Agenda

1. EJB – History, Goals, Evolution
2. CDI – History, Goals, Evolution
3. Advantages/disadvantages – when to use EJB/CDI
4. Java EE – Managed Bean Alignment
5. What is our strategy for the future
CDI and EJB: Bridging the Gap in Java EE 8

Expanded use of EJB Container Services

• Java EE 8 continues the theme of Managed-Bean Alignment
• Container-managed security: authorization
• Message-driven beans
Proposed: Authorization via Security Interceptors

Java EE Security 1.0 (Java EE 8)

@IsAuthorized("hasRoles('Manager') && schedule.officeHrs")
public void transferFunds() {...}

@IsAuthorized("hasRoles('Manager') && hasAttribute('directReports', employee.id)")
public double getSalary(int employeeId) {...}

@IsAuthorized(ruleSourceName="java:app/payrollAuthRules", rule="report")
public void displayReport() {...}
Message-driven Beans

JMS 2.1: New API to receive messages asynchronously (Java EE 8)

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

EJB MDBs Today (Java EE 7)

@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
    }
}
Proposed: Message-driven Beans

JMS 2.1 Tomorrow: Allow any Java EE bean to be a listener (Java EE 8)

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

Other Areas of Alignment

• JAX-RS Injection alignment
• Parameter injection
• Scope for WebSocket endpoints
• Extension of Timer Service and Timer notifications (@Schedule)
• ...
Some Gaps Remain

• Java EE components are CDI managed beans
  – They can be injected with @Inject
  – But, Java EE components other than session beans lose their Java EE "componentness"
  – E.g., you can inject a servlet into a managed bean, but the injected instance won't still service web requests
Summary

EJB Features made more broadly available through CDI

- Java EE 6
  - Interceptors

- Java EE 7
  - Container-managed transactions \(\rightarrow\) transactional interceptors

- Java EE 8
  - Container-managed authorization \(\rightarrow\) security interceptors
  - Message-driven beans \(\rightarrow\) simplified messaging with CDI-based MDBs

- Java EE 9
  - Timer Service (?)
  - Timed Events (?)
  - ... (?)
What is the Future of EJB?

• Part of EJB becoming Optional
  – EJB Entity Beans, EJB QL
    • Optional as of Java EE 7; superseded by JPA
  – Support for JAX-RPC
    • Optional as of Java EE 7; superseded by JAX-WS
  – IIOP Interoperability ??
    • Java EE 8 Experts will decide on "Proposed Optional" status
  – Remote interfaces ??
    • Java EE 8 Experts will decide on "Proposed Optional" status
  – Optionality process is slow
    • Takes 2 Java EE Platform release cycles
What is the Future of EJB?

• Important EJB features are being made more broadly available
  – Interceptors, container-managed txs and security, MDBs, ...
  – Availability is through mechanisms of CDI

• Will EJB still be relevant?

• EJB's long-term future depends on the future of remaining features
  – Remote access (RMI, Web Services)
  – @Schedule'd events and Timer Service
  – Singletons - @Startup; @DependsOn; container-managed concurrency
  – Asynchronous methods
  – Integration with JPA
How You Can Influence the Discussion

• 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