ORACLE

E-Neuauflage - Die Java Enterprise Edition 7 und GlassFish

Wolfgang Weigend
Systemberater Fusion Middleware
Java Technologie und Architektur

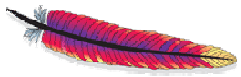
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Java: Broadest Industry Adoption

9,000,000

JAVA DEVELOPERS

DEPLOYING TO 18 COMPLIANT APPLICATION SERVERS






Java™ ORACLE

Make the Future Java

- Platform Completeness
- Modernization and Innovation
- Developer Productivity
- Open and Transparent Evolution
- Active Community Involvement
- Quality and Security

Java™ ORACLE

Java EE 7 Themes

- More annotated POJOs
- Less boilerplate code
- Cohesive integrated platform

- WebSockets
- JSON
- Servlet 3.1 NIO
- REST

- Batch
- Concurrency
- Simplified JMS

Top Ten Features in Java EE 7

1. WebSocket client/server endpoints
2. Batch Applications
3. JSON Processing
4. Concurrency Utilities (asynchronous capabilities)
5. Simplified JMS API
6. @Transactional and @TransactionScoped (on any POJO)
7. JAX-RS Client API
8. Default Resources (JDBC DataSource, JMS ConnectionFactory)
9. More annotated POJO's (@JMSDestinationDefinition)
10. Faces Flow in JSF

Java API for WebSocket 1.0

- Server and Client WebSocket Endpoint
  - Annotated: @ServerEndpoint, @ClientEndpoint
  - Programmatic: Endpoint
- Lifecycle methods
- Packaging and Deployment

```
@ServerEndpoint("/chat")
 public class ChatServer {
   @OnMessage
   public void chat(String m) {
     . . .
   }
 }
```

Java API for WebSocket 1.0

Chat Server

```
@ServerEndpoint("/chat")
public class ChatBean {
    static Set<Session> peers = Collections.synchronizedSet(…);

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

    @OnClose
    public void onClose(Session peer) {
        peers.remove(peer);
    }

    . . .
```

Java API for WebSocket 1.0

Chat Server (contd.)

```
    . . .

    @OnMessage
    public void message(String message) {
        for (Session peer : peers) {
            peer.getRemote().sendObject(message);
        }
    }
}
```

JSON Processing 1.0

- API to parse and generate JSON
- Streaming API
  - Low-level, efficient way to parse/generate JSON
  - Similar to StAX API in XML world
- Object Model API
  - Simple, easy to use high-level API
  - Similar to DOM API in XML world

Java API for JSON Processing 1.0

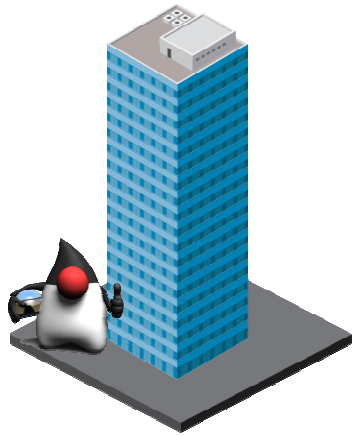
Streaming API

```
{
    "firstName": "John", "lastName": "Smith", "age": 25,
  "phoneNumber": [
    { "type": "home", "number": "212 555-1234" },
    { "type": "fax", "number": "646 555-4567" }
  ]
}
```

```
JsonParser p = Json.createParser(…);
JsonParser.Event event = p.next();          // START_OBJECT
event = p.next();                           // KEY_NAME
event = p.next();                           // VALUE_STRING
String name = p.getString();                // "John”
```

Batch Applications for Java Platform 1.0

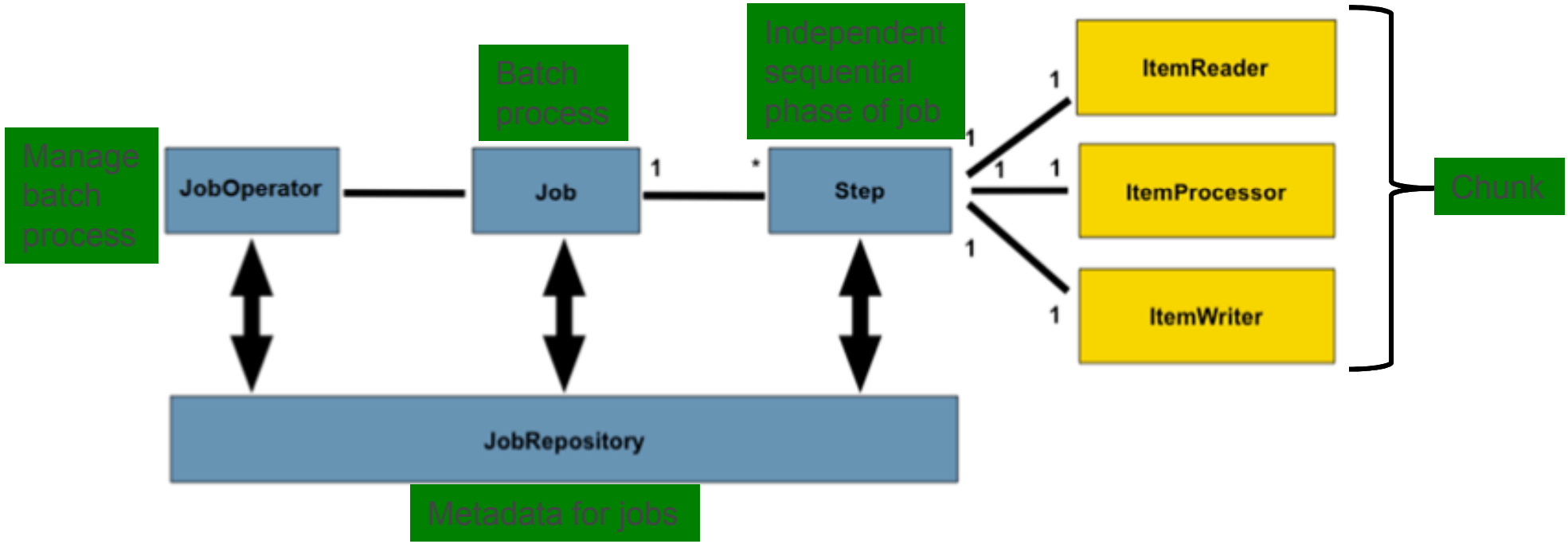
- Suited for non-interactive, bulk-oriented, and long-running tasks
- Batch execution: sequential, parallel, decision-based
- Processing Styles
  - Item-oriented: Chunked (primary)
  - Task-oriented: Batchlet



Java™ ORACLE®

Batch Applications 1.0

Concepts



Batch Applications 1.0

Chunked Job Specification

```
<step id=”sendStatements”>
  <chunk item-count=“3”>
    <reader ref=”accountReader”/>
    <processor ref=”accountProcessor”/>
    <writer ref=”emailWriter”/>
</step>
```

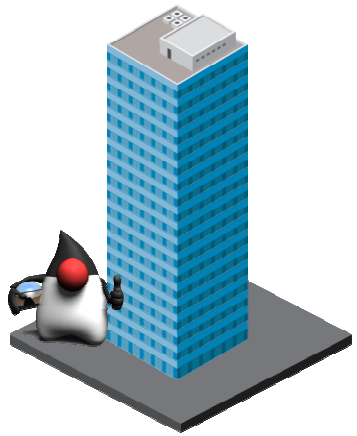
```
…implements ItemReader {
public Object readItem() {
    // read account using JPA
}
```

```
…implements ItemProcessor {
Public Object processItems(Object account) {
    // read Account, return Statement
}
```

```
…implements ItemWriter {
public void writeItems(List accounts) {
    // use JavaMail to send email
}
```


Concurrency Utilities for Java EE 1.0

- Extension of Java SE Concurrency Utilities API
- Provide asynchronous capabilities to Java EE application components
- Provides four types of managed objects
  - ManagedExecutorService
  - ManagedScheduledExecutorService
  - ManagedThreadFactory
  - ContextService
- Context Propagation



Java™ ORACLE®

Concurrency Utilities for Java EE 1.0

Submit Tasks to ManagedExecutorService using JNDI

```
public class TestServlet extends HttpPServlet {
  @Resource(name="java:comp/DefaultManagedExecutorService")
  ManagedExecutorService executor;

  Future future = executor.submit(new MyTask());

  class MyTask implements Runnable {
    public void run() {
      . . .  // task logic
    }
  }
}
```

Java Message Service 2.0

Get More from Less

- New JMSContext interface
- AutoCloseable JMSContext, Connection, Session, …
- Use of runtime exceptions
- Method chaining on JMSProducer
- Simplified message sending

Java Message Service 2.0

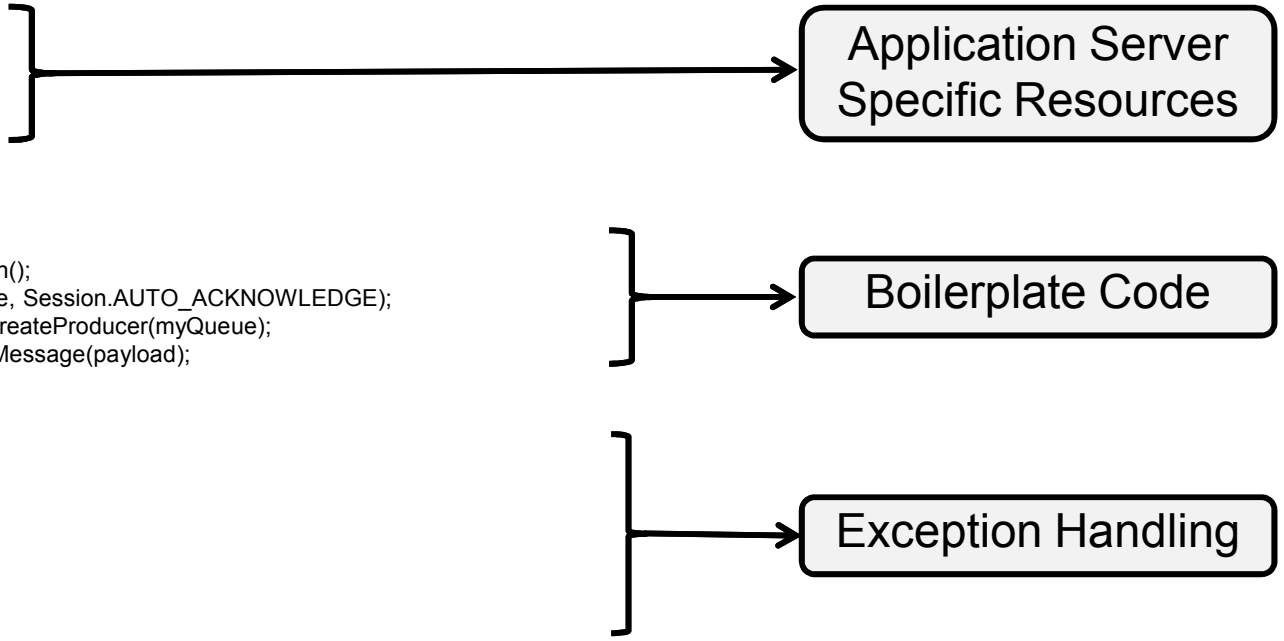
Sending a Message using JMS 1.1

```
@Resource(lookup = "myConnectionFactory")
ConnectionFactory connectionFactory;

@Resource(lookup = "myQueue")
Queue myQueue;

public void sendMessage (String payload) {
    Connection connection = null;
    try {
        connection = connectionFactory.createConnection();
        Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
        MessageProducer messageProducer = session.createProducer(myQueue);
        TextMessage textMessage = session.createTextMessage(payload);
        messageProducer.send(textMessage);
    } catch (JMSException ex) {
        //. . .
    } finally {
        if (connection != null) {
            try {
                connection.close();
            } catch (JMSException ex) {

                //. . .
            }
        }
    }
}
```



Java Message Service 2.0

Sending a Message

```
@Inject
JMSContext context;


@Resource(lookup = "java:global/jms/demoQueue”)
Queue demoQueue;


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

Java API for RESTful Web Services 2.0

- Client API
- Message Filters and Entity Interceptors
- Asynchronous Processing – Server and Client
- Common Configuration

Java API for RESTful Web Services 2.0

Client API

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

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

Contexts and Dependency Injection 1.1

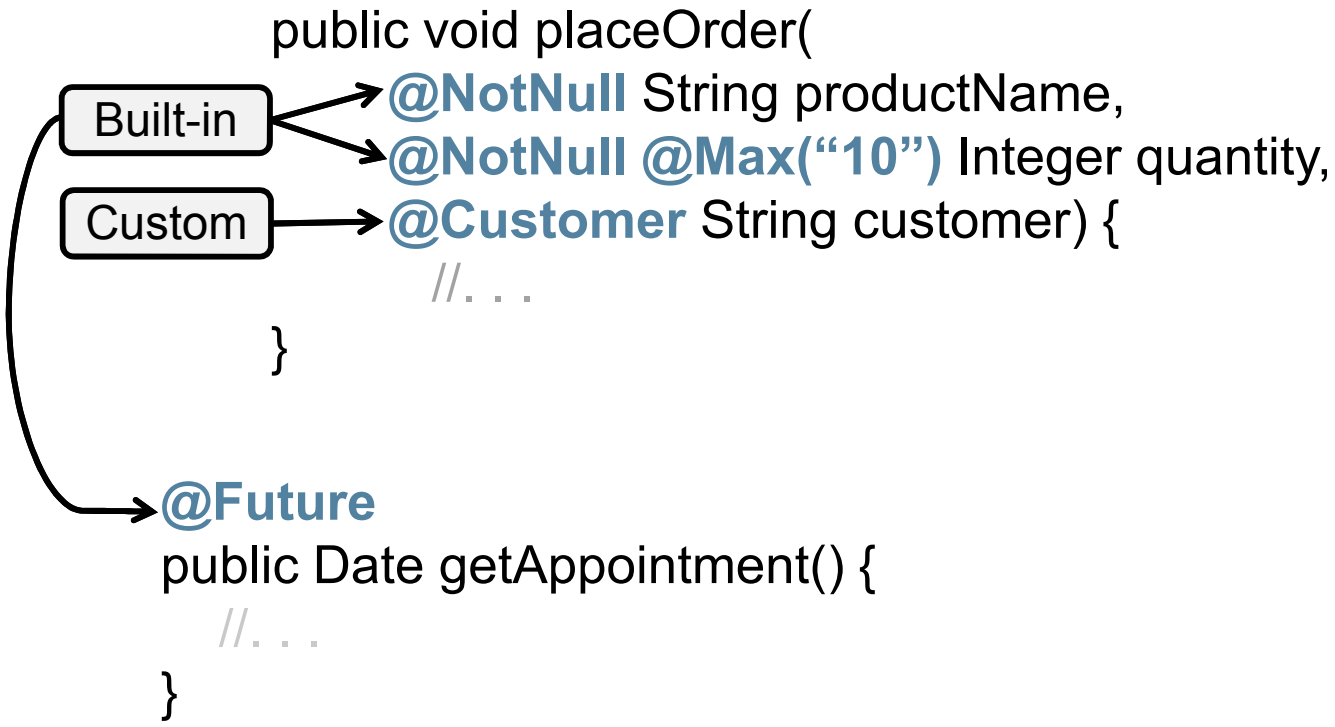
- Automatic enablement for beans with scope annotation and EJBs
  - “beans.xml” is optional
- Bean discovery mode
  - all: All types
  - annotated: Types with bean defining annotation
  - none: Disable CDI
- @Vetoed for programmatic disablement of classes
- Global ordering/priority of interceptors and decorators

Bean Validation 1.1

- Alignment with Dependency Injection
- Method-level validation
  - Constraints on parameters and return values
  - Check pre-/post-conditions
- Integration with JAX-RS

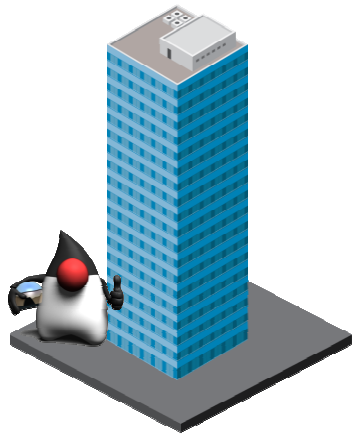
Bean Validation 1.1

Method Parameter and Result Validation



Java Persistence API 2.1

- Schema Generation
  - javax.persistence.schema-generation.* properties
- Unsynchronized Persistence Contexts
- Bulk update/delete using Criteria
- User-defined functions using FUNCTION
- Stored Procedure Query



Java™ ORACLE®

Servlet 3.1

- Non-blocking I/O
- Protocol Upgrade
- Security Enhancements
  - <deny-uncovered-http-methods>: Deny request to HTTP methods not explicitly covered

Servlet 3.1

Non-blocking I/O Traditional

```
public class TestServlet extends HttpServlet
  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response)
                  throws IOException, ServletException {
    ServletInputStream input = request.getInputStream();
    byte[] b = new byte[1024];
    int len = -1;
    while ((len = input.read(b)) != -1) {
      . . .
    }
  }
}
```


Servlet 3.1

Non-blocking I/O: doGet

```
AsyncContext context = request.startAsync();
ServletInputStream input = request.getInputStream();
input.setReadListener(
    new MyReadListener(input, context));
```

Servlet 3.1

Non-blocking read

```
@Override
public void onDataAvailable() {
  try {
    StringBuilder sb = new StringBuilder();
    int len = -1;
    byte b[] = new byte[1024];
    while (input.isReady() && (len = input.read(b)) != -1) {
      String data = new String(b, 0, len);
      System.out.println("--> " + data);
    }
  } catch (IOException ex) {
    . . .
  }
}
```

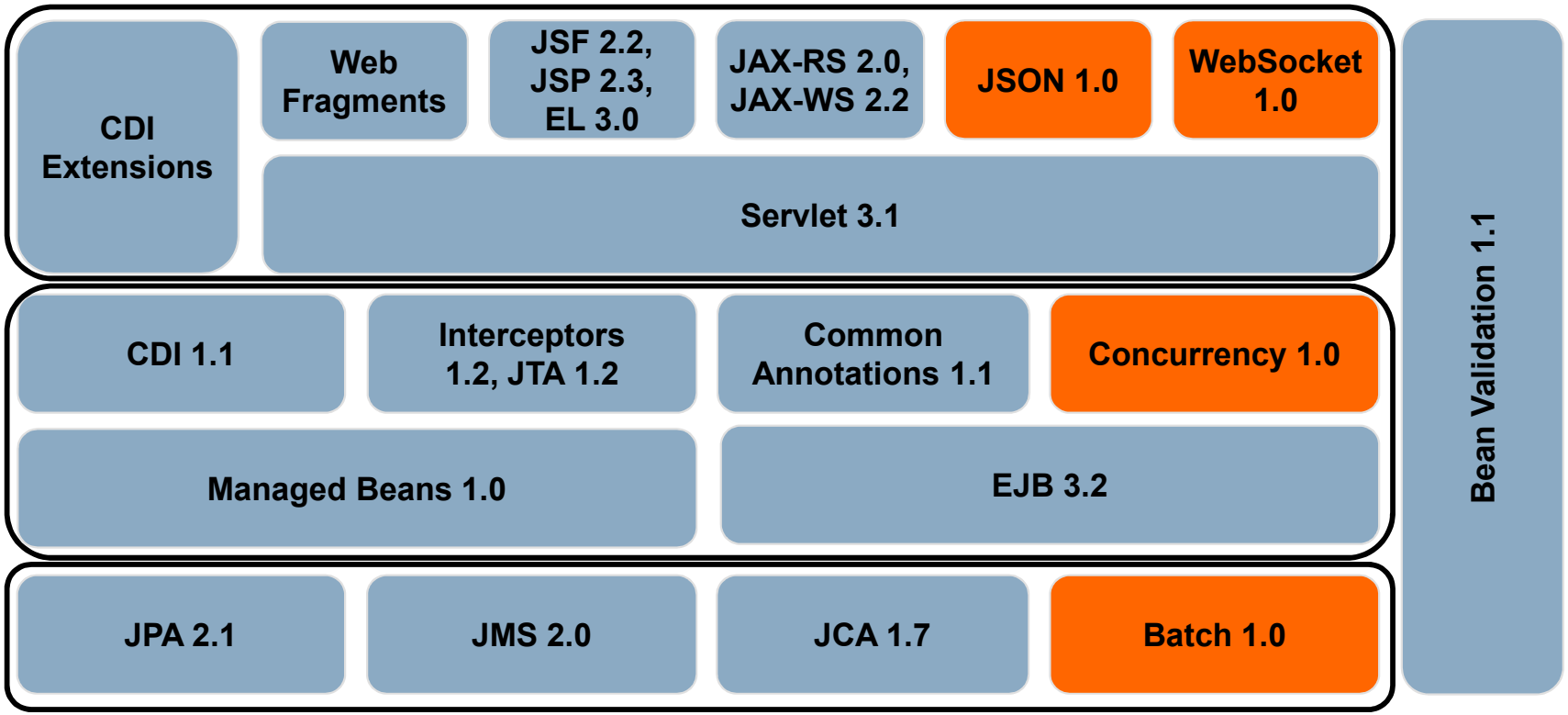
JavaServer Faces 2.2

- Faces Flow
- Resource Library Contracts
- HTML5 Friendly Markup Support
  - Pass through attributes and elements
- Cross Site Request Forgery Protection
- Loading Facelets via ResourceHandler
- h:inputFile: New File Upload Component

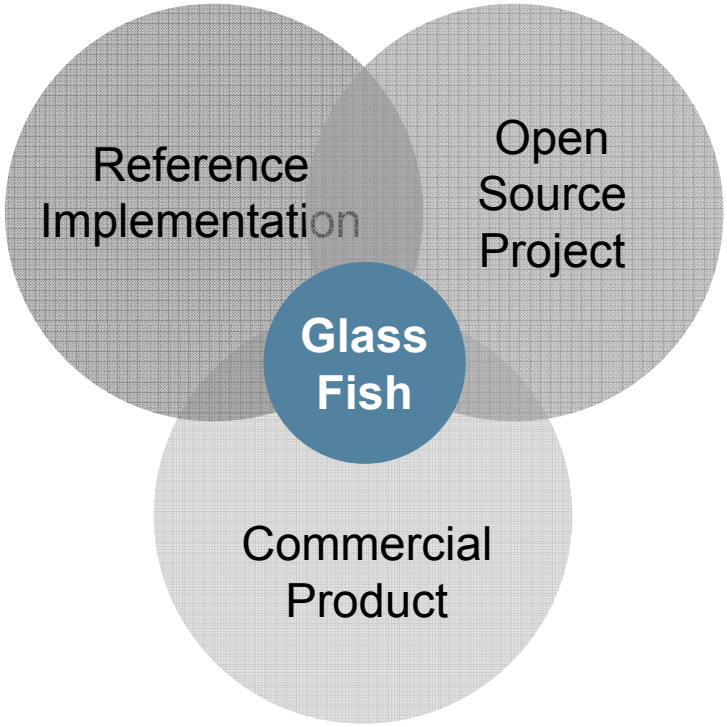
Java Transaction API 1.2

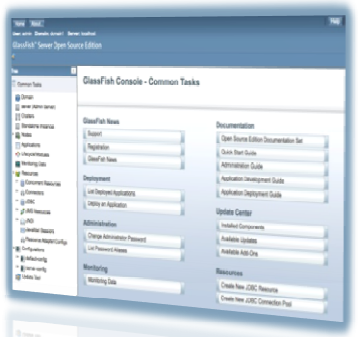
- @Transactional: Define transaction boundaries on CDI managed beans
- @TransactionScoped: CDI scope for bean instances scoped to the active JTA transaction

Java EE 7 Final Specification - JSR's



GlassFish Server

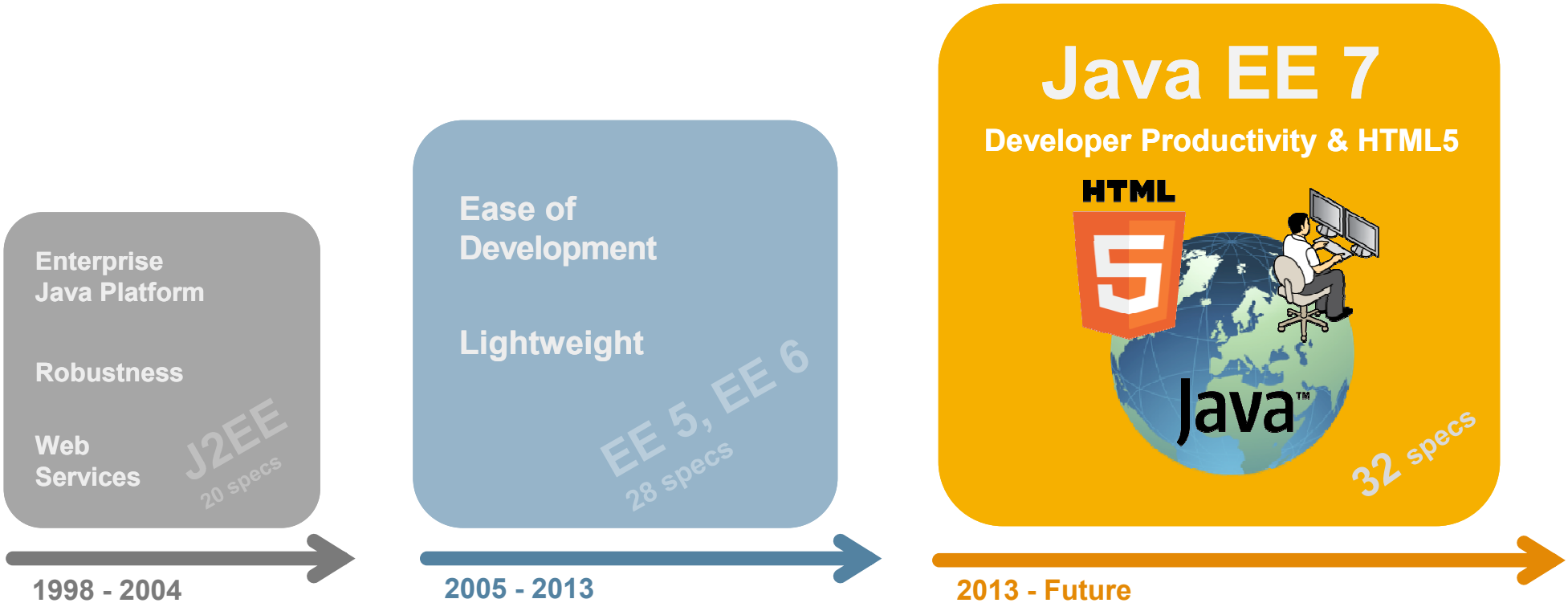




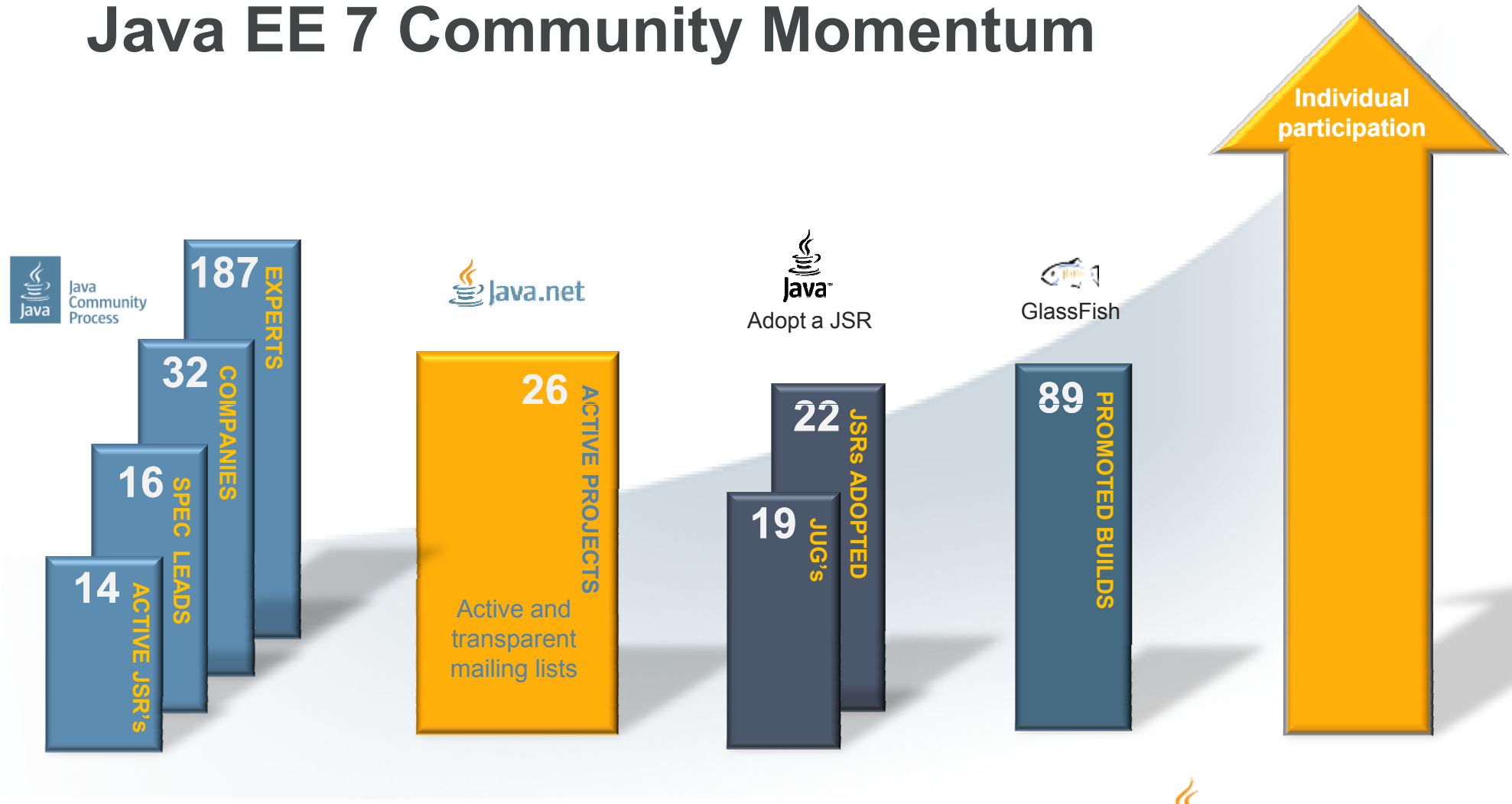
Java ORACLE

GlassFish Server -
Reference Implementation

Java EE – and GlassFish - Journey



Java EE 7 Community Momentum



Java ORACLE

GlassFish Server –
Open Source Project

Java™ ORACLE®

GlassFish Server

Open Source Project

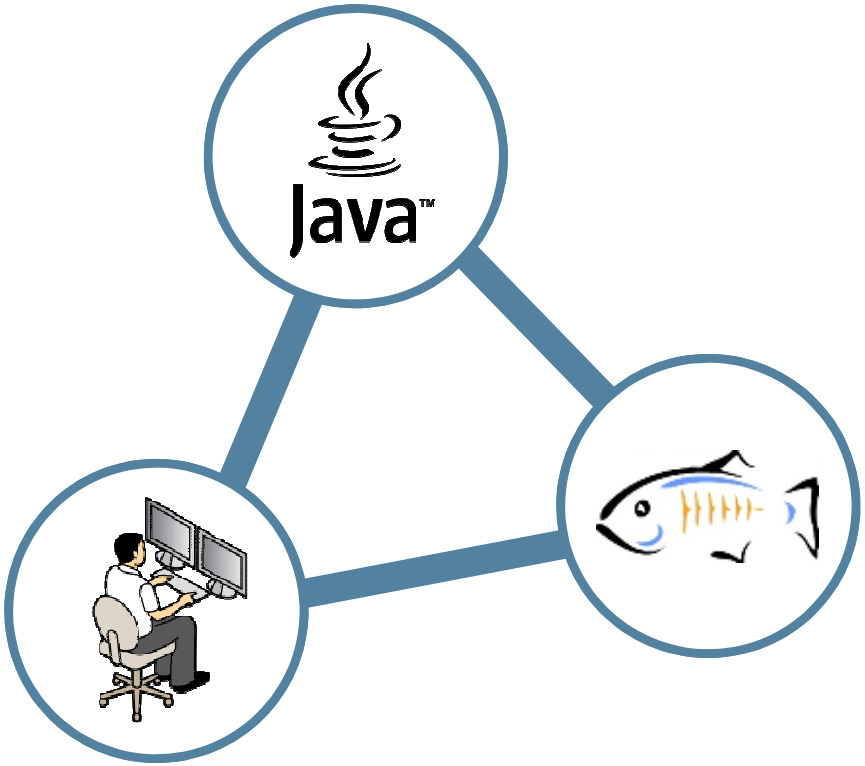
- Built in open source
- World's first Java EE 7 Application Server
- Lightweight / modular / easy to use
- Production ready

 | |

Java™ ORACLE

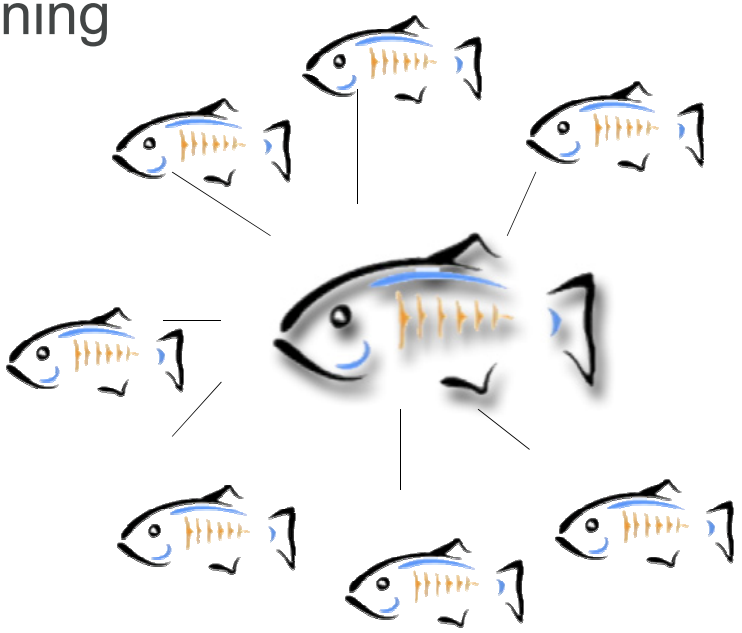
The GlassFish Community

- 10+ million annual downloads
- FishCAT - Early testing and fixes
  - Community Acceptance Testing program
- Active user forums
- Community contributions



Centralized Management

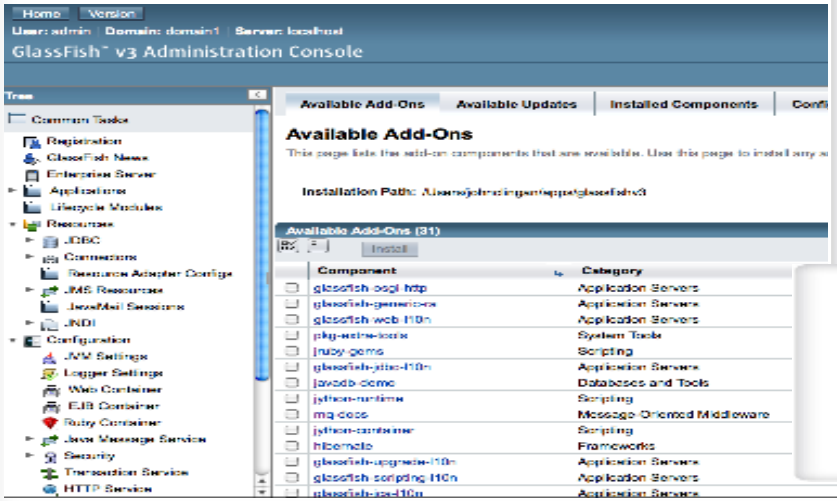
- Integrated GlassFish Server Provisioning
- Manage instance/cluster lifecycle
- Manage configurations
- Manage Java EE resources
- Manage application lifecycle



Flexible Administration



Web Browser



REST API

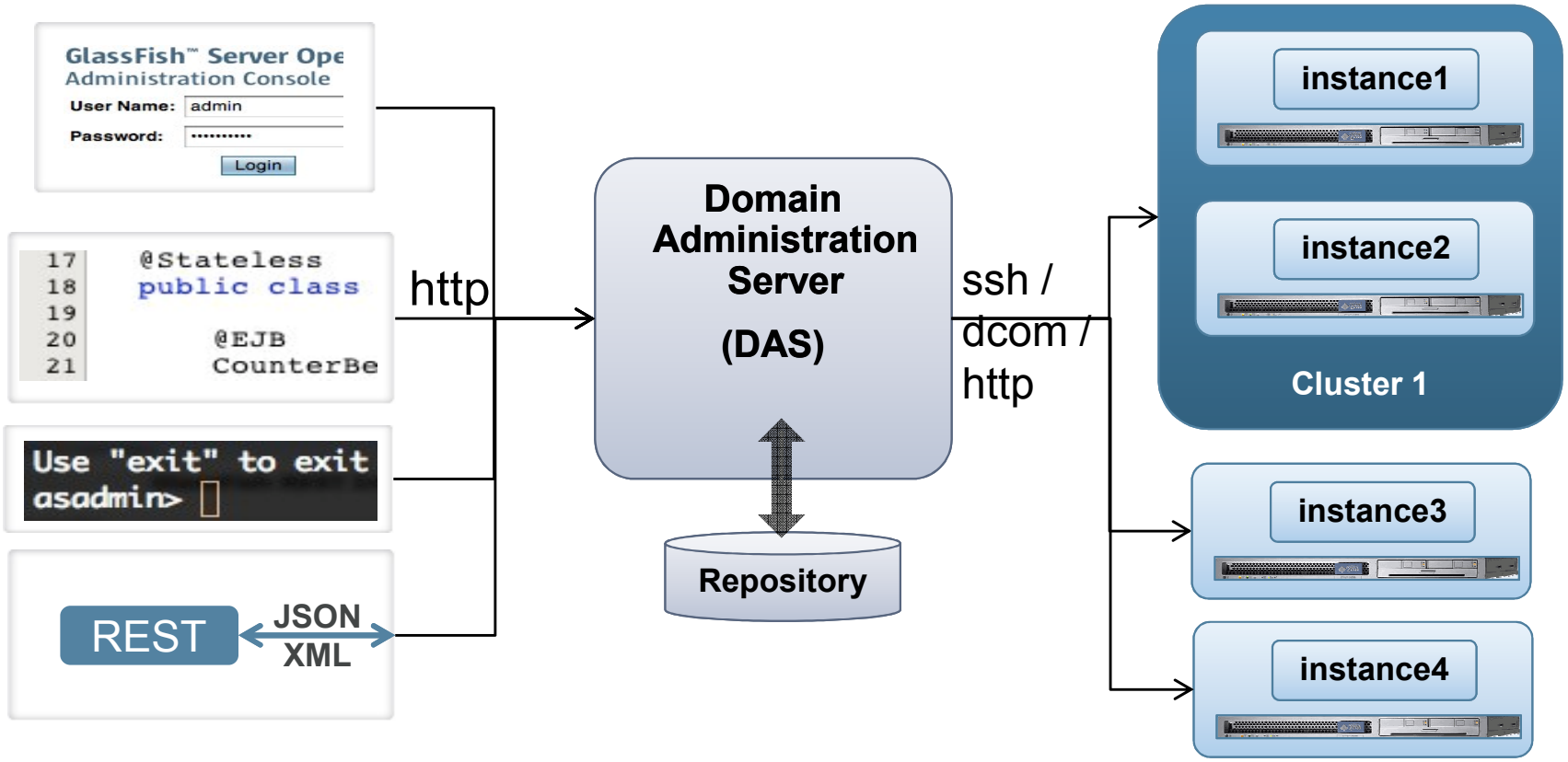
Command Line

```
$ asadmin deploy --target prod_cluster website.war
Application deployed with name website.
Command deploy executed successfully.
```

 | |

Java™ ORACLE

GlassFish Server Administration Architecture

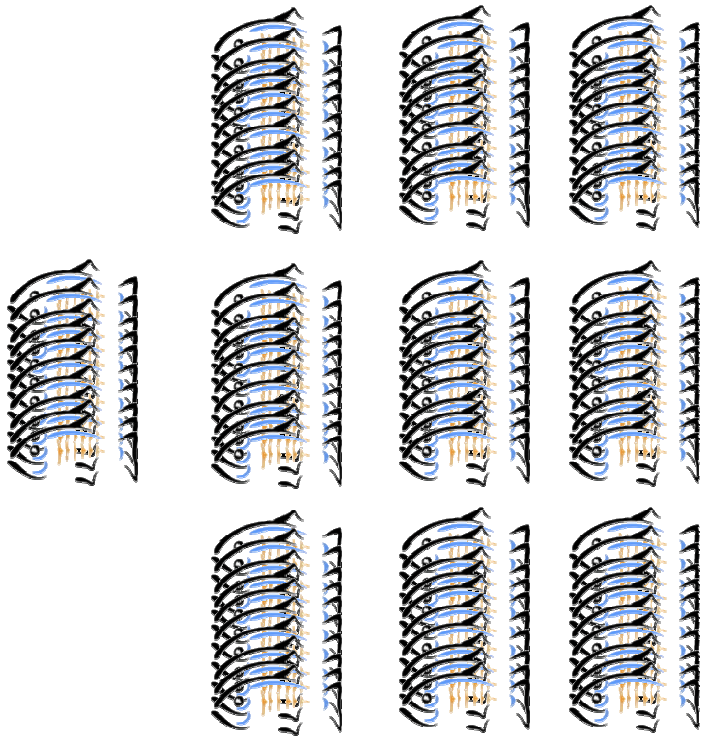


 | |

Java™ ORACLE

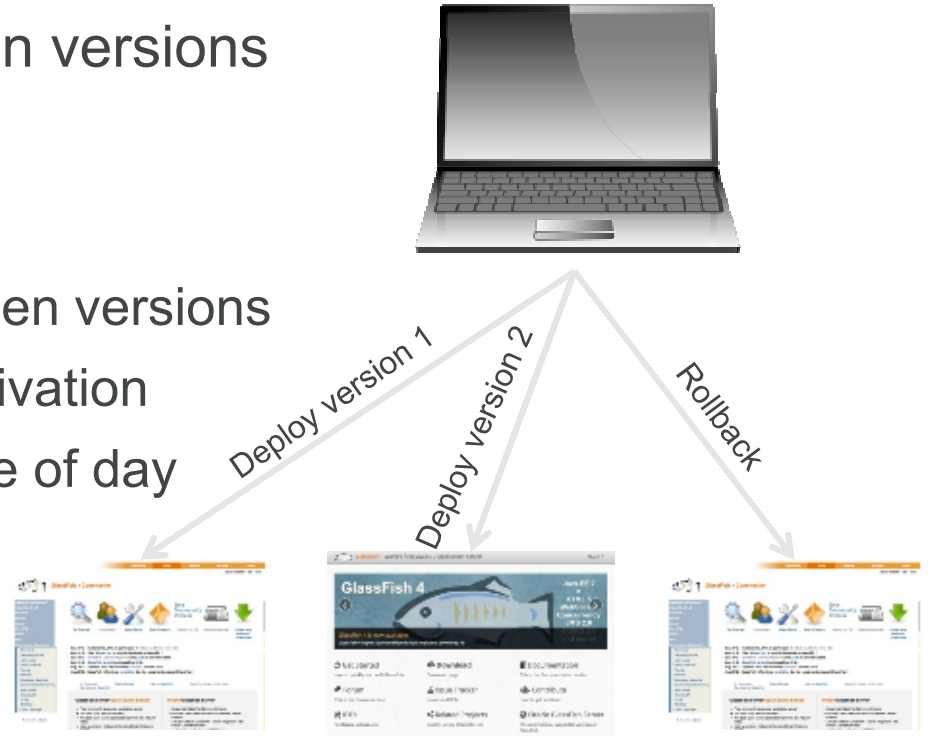
GlassFish Server Clustering

- Manage 100 instances per domain
  - Standalone
  - Clusters (max 10 per cluster)
- Dynamically resize cluster
- JMS Broker Clustering
- Load Balancing
  - Commercial plugin
  - AJP (mod_jk, mod_proxy)



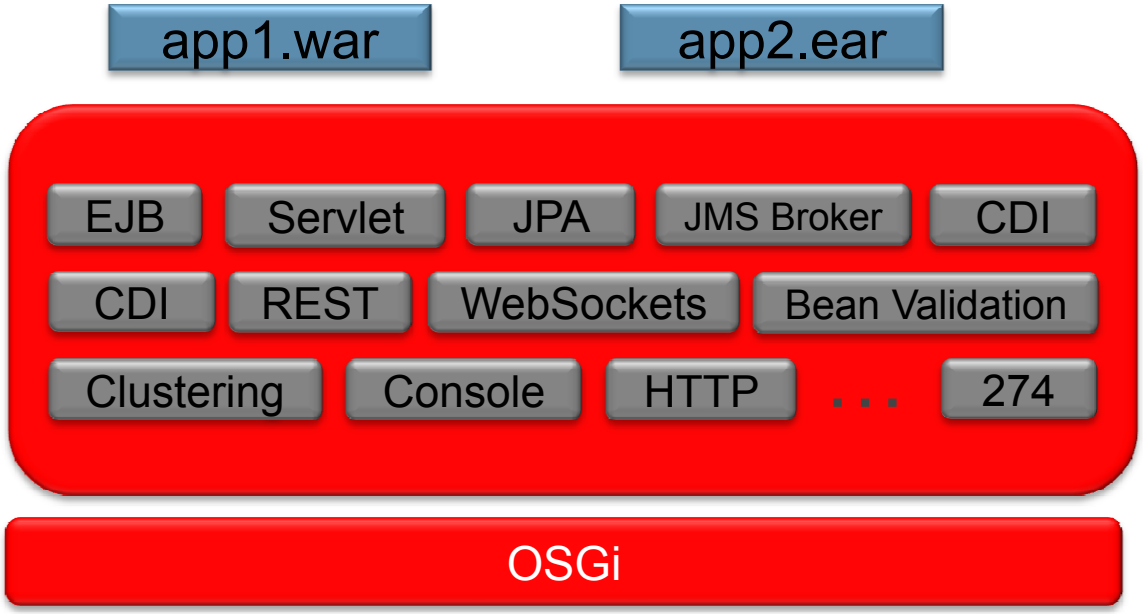
Application Versioning

- Deploy multiple versions application versions
- Activate any version
- For example:
  - Roll forward and backwards between versions
  - Pre-deploy application for later activation
  - Enable new version at specific time of day



GlassFish Modularity

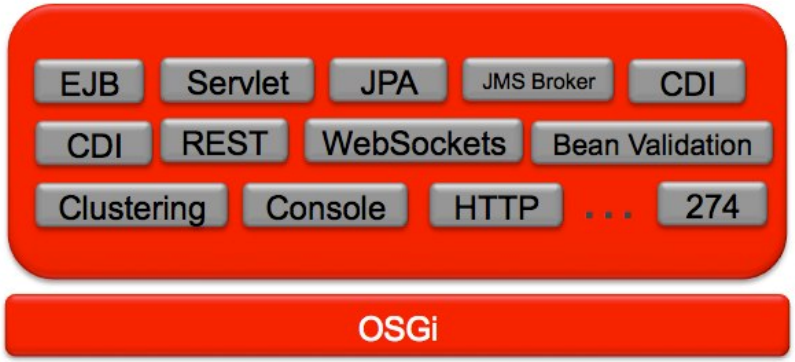
In a nutshell



- Starts in seconds
- Only loads required modules
- Including "infrastructure" features

OSGi Support

- OSGi Runtime (Apache Felix)
  - Also runs on others
  - Ships with ~275 modules
  - Can run without OSGi
  - Integrates OSGi management tools
- Hybrid Java EE / OSGi Applications
  - Expose EJBs as OSGi Services
  - Inject OSGi services into Java EE applications
- Run any OSGi bundle
- Supports OSGi Enterprise



Java™ ORACLE®

Extending GlassFish Server

- Define a new container
  - System-level logic
  - Implement administration commands
- Expose administration capabilities
  - Administration Console
  - Command Line
  - RESTful Interface
- Embeddable API
- Re-brand user interface

Built-in Developer Features

- Ultra-fast deployment
  - Hot deployment
  - Directory deployment
- Run as a single JAR

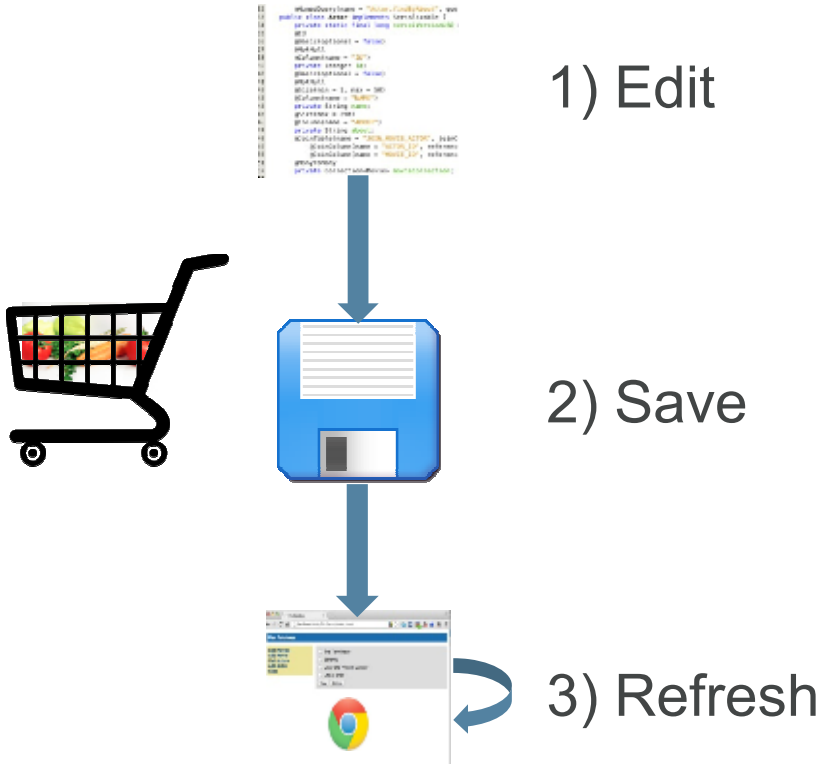
```
java –jar glassfish.jar app1.war
```

- Maven Plugin

```
mvn gf:run, gf:start, gf:deploy
```

Active Redeploy

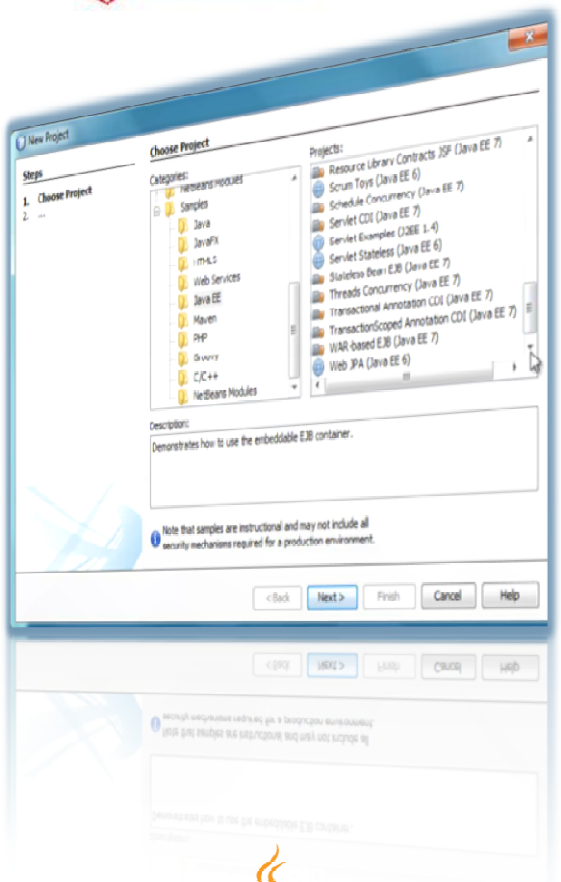
- Maintains state across redeploys
- Rapid Iterative development
  - Edit
  - Save
  - Refresh



Java™ ORACLE

NetBeans 7.3.1 + GlassFish 4.0

- NetBeans bundles GlassFish
- Incremental compilation; auto-deploy
- Complete Java EE 7 support
  - All Java EE 7 project types
  - Maven Support
  - Wizards
  - Advanced Wizards
    - Entity to REST generation
    - Database to JSF 2.2



 | |

Java™ ORACLE

Oracle GlassFish Server - Commercial Product

Java™ ORACLE®

Oracle GlassFish Server

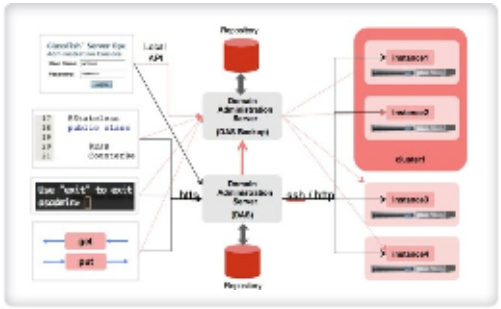
Java EE 6 Compliant

- **Commercial Distribution of GlassFish Server Open Source Edition**
- GlassFish Server Control
- Java SE support
- 24 x 7 x 365 premium support
- Regular patch releases
- Indemnification

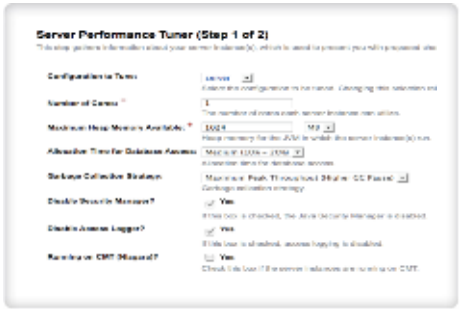
Java™ ORACLE®

GlassFish Server Control

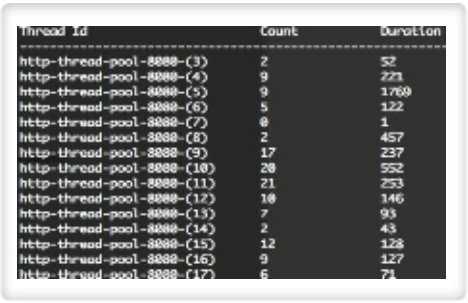
DAS Backup & Recovery



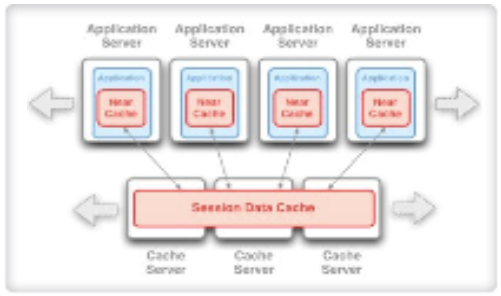
Performance Tuner



Monitoring Scripting Client

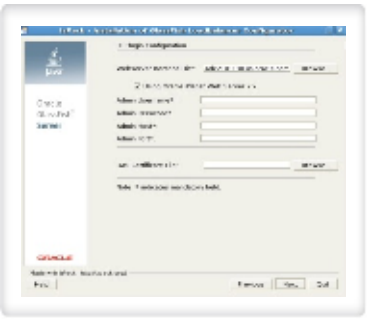


ActiveCache for GlassFish



Oracle Access Manager Integration

Load Balancer Plugin & Installer

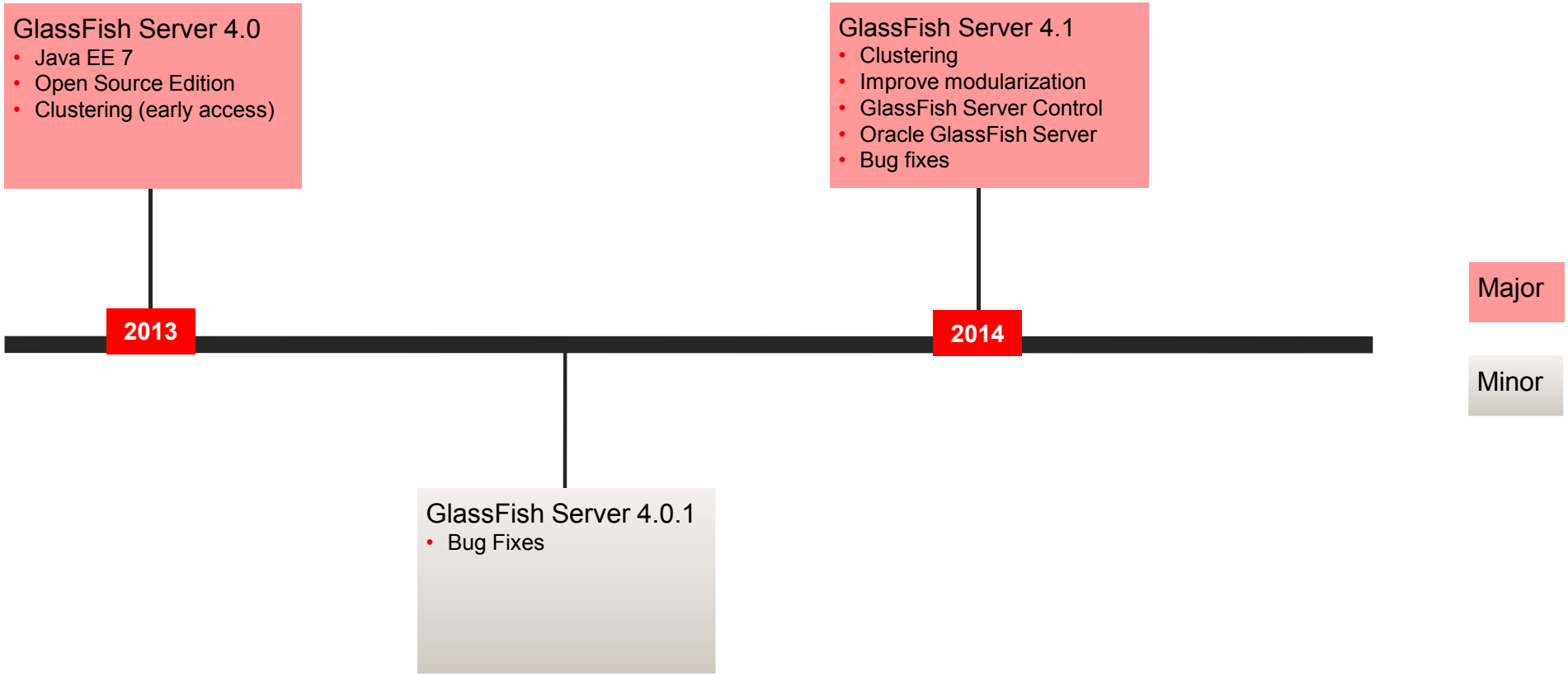


 | |

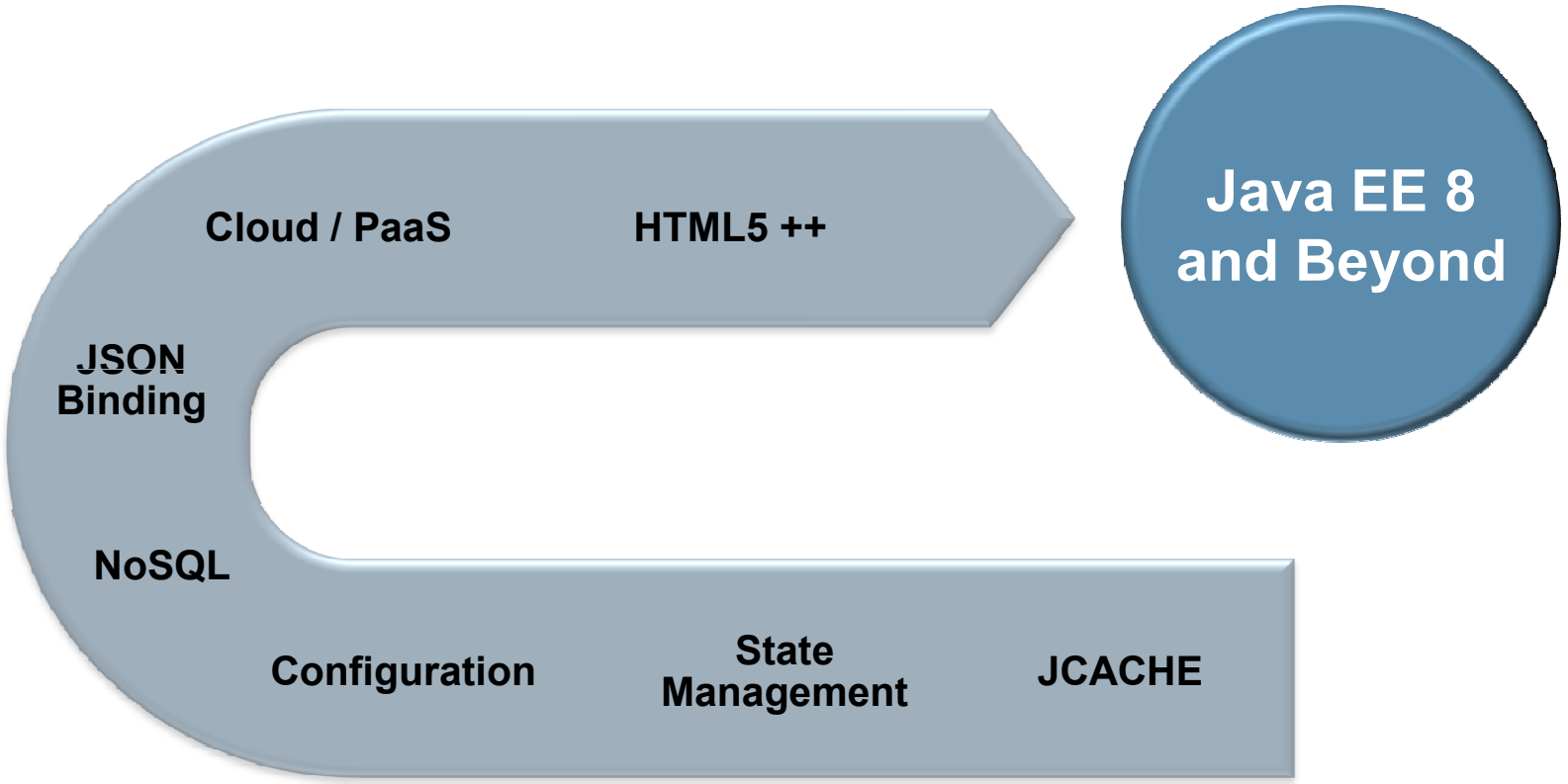
Java™ ORACLE

Roadmap

GlassFish Server 4.x



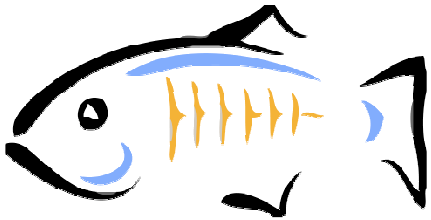
Future of Java EE



Java™ ORACLE®

Zusammenfassung

- **Java EE 7**
  - Neue API's
- **Java als Plattform nutzt Innovationen im Ökosystem und paßt sich dem Änderungprozess der IT an**



DOWNLOAD

Java EE 7 SDK

oracle.com/javaee

GlassFish 4.0

Full Platform or Web Profile

glassfish.org

Java™ ORACLE®

Vielen Dank für Ihre Aufmerksamkeit!

Wolfgang.Weigend@oracle.com

Twitter: @wolflook