



J2EE 1.4 – Neue Features im Überblick

Version 1.0 - JAX 2003

Lars Röwekamp & Jens Schumann

Agenda

- Motivation
- J2EE - was bisher geschah ... ?
- Ausrichtung der neuen Spezifikation
- Web Services
- JSP und Java Servlet API
- Enterprise JavaBeans
- Deployment, Management & Monitoring

- Fazit & Ausblick
- Q & A



Motivation

- Unternehmen müssen heute ...
 - flexibel sein,
 - Kosten reduzieren,
 - Reaktionszeiten gegenüber Kunden und Lieferanten verkürzen.
- Applikationen müssen daher ...
 - Enterprise Information Systems (EIS)
 - und neue Business Anforderungen kombinieren.
- Notwendige Services müssen ...
 - hoch verfügbar,
 - sicher,
 - verlässlich und
 - skalierbar sein.

Motivation

- Zitate aus der J2EE Spezifikation:
 - *„The Java 2 Platform Enterprise Edition reduces the cost and complexity of developing multitier, enterprise services.“*
 - *„J2EE Applications can be rapidly deployed and easily enhanced as the enterprise responds to competitive pressure.“*

Agenda

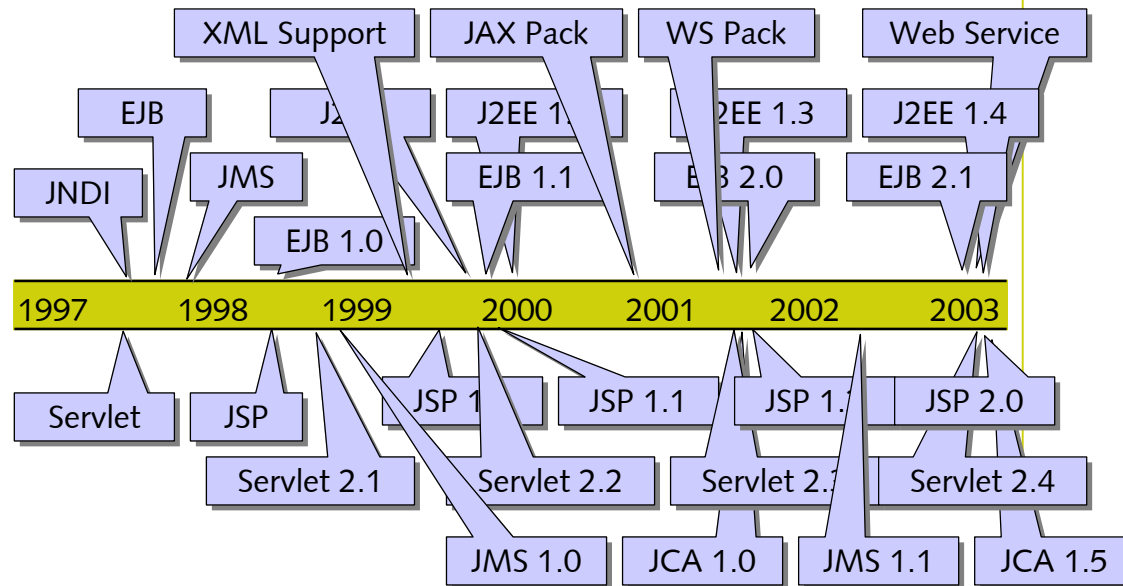
- Motivation
- J2EE - was bisher geschah ... ?
- Ausrichtung der neuen Spezifikation
- Web Services
- JSP und Java Servlet API
- Enterprise JavaBeans
- Deployment, Management & Monitoring

- Fazit & Ausblick
- Q & A



Was bisher geschah ...

■ Die J2EE Zeitachse



Agenda

- Motivation
- J2EE - was bisher geschah ... ?
- Ausrichtung der neuen Spezifikation
- Web Services
- JSP und Java Servlet API
- Enterprise JavaBeans
- Deployment, Management & Monitoring

- Fazit & Ausblick
- Q & A



Ausrichtung der neuen Spezifikation

- J2EE 1.4 ist ein Pool für diverse Enterprise Computing Spezifikationen:
 - Web Service
 - Enterprise Java Beans 2.1
 - Java Servlets 2.4
 - Java Server Pages 2.0
 - Java Standard Tag Library 1.0
 - Java Message Service 1.1
 - J2EE Connector Architecture 1.5
 - J2EE Management
 - J2EE Deployment

Ausrichtung der neuen Spezifikation

- J2EE 1.4 Schwerpunkte und Highlights:
 - Web Services
 - EJB Timer Service
 - Advanced Message Driven Beans
 - JSP Expression Language
 - JSTL & besserer Custom Tag Support
 - Servlet Listener Konzept
 - JCA Erweiterung
 - JMS Vereinfachung
 - Deployment & Management

Agenda

- Motivation
- J2EE - was bisher geschah ... ?
- Ausrichtung der neuen Spezifikation
- **Web Services**
- JSP und Java Servlet API
- Enterprise JavaBeans
- Deployment, Management & Monitoring

- Fazit & Ausblick
- Q & A



- Möglichkeiten des J2EE Web Service Handlings via ...
 - direkter SOAP Abarbeitung innerhalb eines Java Servlets,
 - JAX RPC Endpoint und Java Servlet,
 - Service Endpoint Interface (SEI) innerhalb einer Stateless Session Bean oder Message Driven Bean

Web Services – direktes SOAP Handling im Servlet

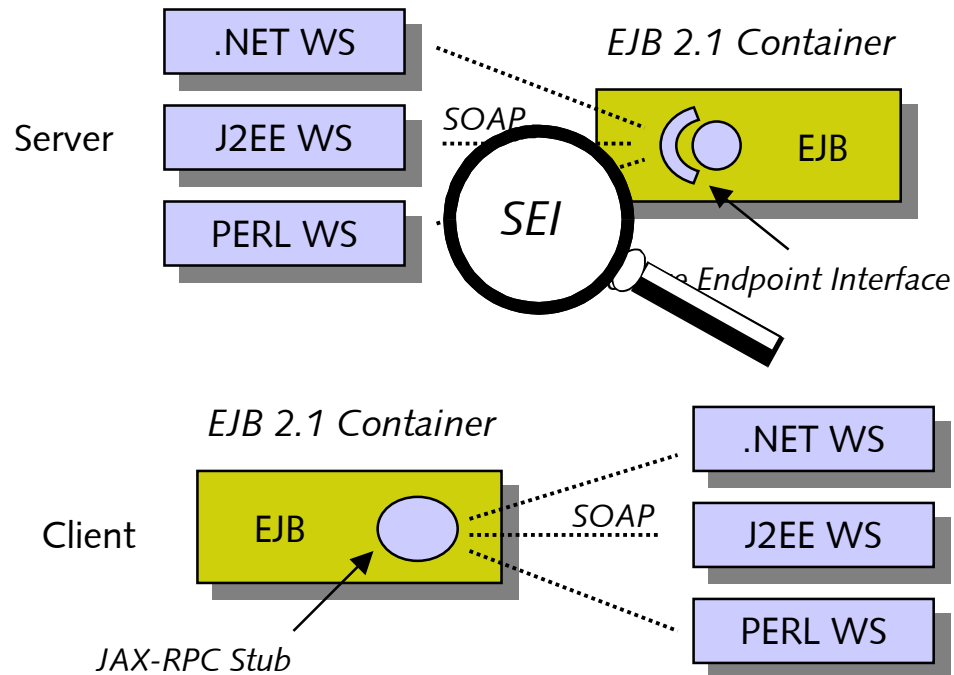
- SOAP Message wird direkt im Servlet bearbeitet
- Vorteile:
 - extrem flexibel
 - Unterstützung von NON-HTTP Protokollen
 - Unterstützung mehrerer Rückgabeparameter
- Nachteile:
 - sehr aufwendig
 - schwer zu warten
 - keine Interaktion durch den Container

Web Services - JAX RPC Endpoint

- Service Endpoint Interface als Subset der öffentlichen Servlet Methoden definieren
- Vorteile:
 - toolgestützte Generierung von WSDL bzw. Interfaces
 - JAX RPC Runtime Environment übernimmt größten Teil der Arbeit
 - Deploymentunterstützung durch **webservice.xml**
 - Servlet Features, wie z.B. Filter, können weiter genutzt werden
- Nachteile:
 - RPC Ansatz ist nur Subset der SOAP Spezifikation
 - begrenzte Datentypen

Web Services – SEI & Session Bean

- Stateless Session Bean kann als Web Service Client und Server fungieren



Web Services – SEI & Session Bean

- SEI als Subset der öffentlichen Session Bean Methoden definieren
- Vorteile:
 - Service Methoden stehen gleichzeitig als Web Services zur Verfügung
 - Web Service kann EJB Vorteile nutzen
 - Deploymentunterstützung durch `webservice.xml` und `ejb-jar.xml`
 - JAX RPC Runtime Environment übernimmt größten Teil der Arbeit
- Nachteile:
 - RPC Ansatz ist nur Subset der SOAP Spezifikation
 - begrenzte Datentypen

Web Services – SEI & Session Bean

- EJB SEI im Detail – lediglich 5 Schritte:
 - Service Endpoint Interface definieren
 - SEI in Bean Implementierung einbinden
 - SEI in `ejb-jar.xml` deklarieren
 - Zusätzlichen Deskriptor `webservices.xml` erzeugen
 - Web Service nutzen

- WSDL des Endpoints sowie SOAP Handling Klassen werden automatisch generiert

Web Services – SEI & Session Bean

- EJB SEI im Detail – Schritte 1 von 5:
 - Service Endpoint Interface definieren

```
public interface OpenKnowledge extends
    javax.rmi.Remote {

    /** openKnowledge web service */
    public String getKnowledge (String name)
        throws javax.rmi.RemoteException;

}
```

Web Services – SEI & Session Bean

- EJB SEI im Detail – Schritte 2 von 5:
 - SEI in Bean Implementierung einbinden

Business
Interface

```
public class OpenKnowledgeBean implements
    OpenKnowledge, SessionBean {

    /** openKnowledge web service */
    public String getKnowledge(String name) {...}

    /** EJB Life Cycle Methods */
    public void ejbCreate() throws CreateException {...}

    ...
}
```

Web Services – SEI & Session Bean

- EJB SEI im Detail – Schritte 3 von 5:
 - SEI in `ejb-jar.xml` deklarieren

```
<session>
  <ejb-name> ... <ejb-name>
  ...
  <remote> ... <remote>
  <service-endpoint>
    de.openknowledge.webservice.OpenKnowledge
  </service-endpoint>
  ...
</session>
```

Endpoint
Deklaration

Web Services – SEI & Session Bean

- EJB SEI im Detail – Schritte 4 von 5:
 - Deskriptor `webservices.xml` erzeugen

```
<webservices>
  <web-service-description>
    <web-service-description-name>OpenKnowledgeService
      ...
    <port-component>
      <port-component-name>OpenKnowledge ...
      <service-endpoint-interface>
        de.openknowledge.webservice.OpenKnowledge
      </service-endpoint-interface>
      <service-impl-bean>
        <ejb-link>OpenKnowledgeBean</ejb-link>
      </service-impl-bean>
      ...
    
```

Bean-WS
Mapping

Web Services – SEI & Session Bean

- EJB SEI im Detail – Schritte 5 von 5:
 - Web Service nutzen

WS Referenz –
kann wie andere
Referenzen auch
im Environment
platziert werden

```
String serviceName =
    "java:comp/env/service/OpenKnowledgeService
";

/** retrieve OpenKnowledgeService via JNDI */
javax.xml.rpc.Service = (javax.xml.rpc.Service)
    new InitialContext().lookup(serviceName);

/** retrieve OpenKnowledge via Service */
OpenKnowledge openKnowledge =
    okService.getPort(OpenKnowledge.class);

/** use openKnowledge business method */
String knowledge = openKnowledge.getKnowledge();
```

Agenda

- Motivation
- J2EE - was bisher geschah ... ?
- Ausrichtung der neuen Spezifikation
- Web Services
- JSP und Java Servlet API
- Enterprise JavaBeans
- Deployment, Management & Monitoring
- Fazit & Ausblick
- Q & A



Java Server Pages und Java Servlet API

- Neues allgemein
 - web.xml DD als XML Schema
- Neues in der Servlet API 2.4 ...
 - Request und Request Attribute Listener
 - Servlets als „Welcome Files“ erlaubt
 - Feinere Kontrolle über Filteraufruf beim Aufruf von RequestDispatcher's include() und forward() Methode
- Neues in den Java Server Pages 2.0 ...
 - Expression Language (EL)
 - Custom Tag Entwicklung ohne Java
 - Selbst dokumentierende TLDs durch neue Tags
 - JSTL

Java Server Pages und Java Servlet API

- Expression Language (EL)
 - vereinfachter Zugriff auf Daten innerhalb von JavaBeans
 - kann pro Seite aktiviert/deaktiviert werden
 - EL unterstützt arithmetische, logische, relationale und leere Operatoren
 - Beispiel:

```
/** test for empty shopping cart */  
<c:if test="${sessionScope.cart.numberOfItems > 0}">  
    ...  
</c:if>
```

- Expression Language Funktionen
 - Definition in TLD

```
<function>
  <name>myFunction</name>
  <function-class>MyClass</function-class>
  <function-signature>
    String myFunction(String myParam)
  </function-signature>
</function>
```

- Nutzung in JSP

```
...
Hello, ${ok:myFunction(myParam)}!
...
```

myFunction muss eine öffentliche Methode in der öffentlichen und nicht abstrakten Klasse **MyClass** sein.

Java Server Pages und Java Servlet API

- Simple Tag Extension
 - JSP Syntax Custom Tags
 - Tag Handler ist deutlich einfacher als der Standard Tag Handler
 - nur doTag() Methode
 - Wiederverwendung von text/HTML innerhalb eines Tags
 - JSP Page Autoren benötigen keine Java Kenntnisse oder Unterstützung durch Java Programmierer
 - ablegen in .tag Datei bzw. tagf
 - Packen wie normale Tags mit TLD in JAR oder unter WEB-INF/tags mit impliziter TLD

Java Server Pages und Java Servlet API

- Simple Tag Extension – Beispiel 1 von 2
 - Tag File – tableTag.tag

```
<%@ tag name="tableTag" %>
<%@ attribute name="tableItems" %>
<table width= ... >
  ...
  <c:forEach var="iter" items="${tableItems}">
    <tr>
      <td>${iter.name}</td>
      <td>${iter.price}</td>
    </tr>
  </c:forEach>
</table>
```

JSTL
Element

- Simple Tag Extension – Beispiel 2 von 2
 - Anwendung innerhalb einer JSP

```
...  
Shopping Cart:  
<ok:tableTag tableItems="${shoppingCart}" />  
  
Wunschliste:  
<ok:tableTag tableItems="${wishList}" />  
...
```

Java Server Pages und Java Servlet API

- Java Standard Tag Library – Tags für ...
 - Attribut Handling
 - Exception Handling
 - Bedingungen
 - Schleifen
 - URL Behandlung
 - Ressourcen Import
 - l18n
 - Formatierung
 - SQL DBMS Zugriff
 - XML, XPath und XSLT

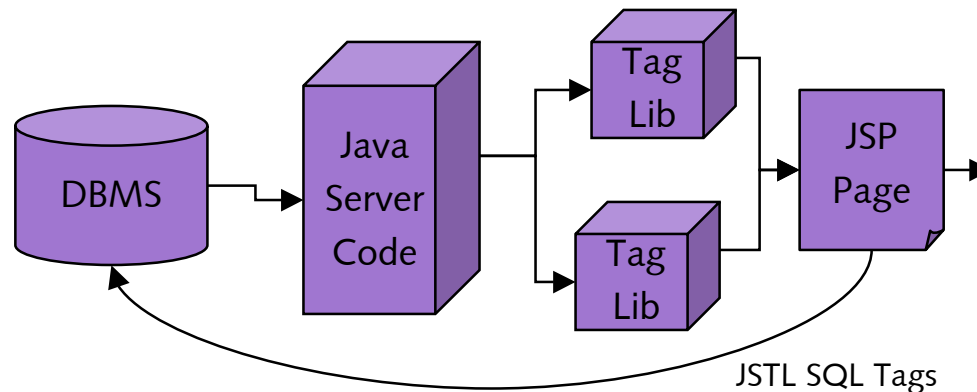
- optional

Java Server Pages und Java Servlet API

- Java Standard Tag Library – Beispiel:
 - SQL Zugriff

```
...  
<sql:query sql="SELECT * FROM USERS" var="result" />  
<c:forEach items="${result.rows}">  
...  
</c:forEach>  
...
```

- Problem



Agenda

- Motivation
- J2EE - was bisher geschah ... ?
- Ausrichtung der neuen Spezifikation
- Web Services
- JSP und Java Servlet API
- **Enterprise JavaBeans**
- Deployment, Management & Monitoring

- Fazit & Ausblick
- Q & A



Enterprise JavaBeans

- Enterprise JavaBeans
 - Timer Service
 - siehe folgendes Beispiel
 - Erweiterte Message Driven Beans
 - nicht mehr beschränkt auf JMS
 - können durch JCA Ressource Adapter aufgerufen werden
 - Erweiterung der EJB Query Language
 - Sortierung möglich durch „order by“
 - zusätzliche Funktionen, wie z.B. min, max, sum, count, avg und mod

- EJB Timer Services ...
 - sind nutzbar durch Stateless Session Beans und Message Driven Beans
 - bieten Single-Action und Interval-Based Callback Methoden
 - sind persistent

- Timer Service nutzen ...
 - EJB Implementierung muss `javax.ejb.TimerObject` und somit die Methode `ejbTimeout()` implementieren
 - Zugriff erfolgt über den `ejbContext`
 - Timer „scharf schalten“ durch Aufruf von `TimerService.createTimer()`
 - Timer ruft bei Ablauf die Callback Methode `ejbTimeout()` auf, wobei die EJB eventuell neu erstellt wird

■ Timer Service - Beispiel

```
...
public void ejbTimeout(Timer timer){
    cancelOrder();
}
public void cancelOrder() {
    ... // do something
    cancelTimer();
}
private void cancelTimer() {
    TimerService ts = ejbCtx.getTimerService();
    Iterator timers = ts.getTimers().iterator();
    if (timers.hasNext()) {
        Timer timer = (Timer) timers.next();
        timer.cancel();
    }
} ...
```

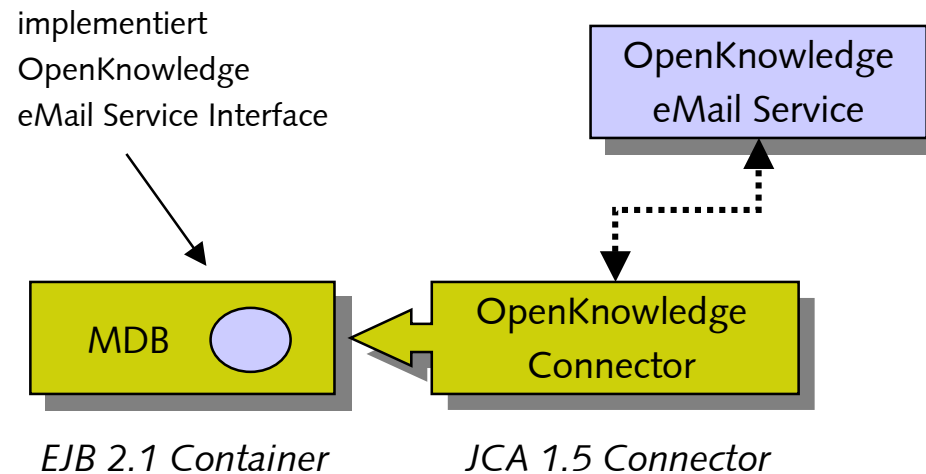
- Timer Service - Nachteile
 - Keine deklarative Konfiguration - muss innerhalb der Beaninstanz konfiguriert werden
 - Kein Gruppenmanagement von Timern möglich
 - Es gibt keinen Weg alle aktiven Timer im System aufzufinden

 - Nicht zu vergleichen mit cron Jobs

- Advanced Message Driven Beans
 - Können neben JMS auch durch JCA Resource Adapter angestoßen werden
 - Müssen dazu entsprechenden Message Listener implementieren
 - Neue Flexibilität führt zu Änderungen im Deployment Descriptor
 - Activation Properties als eine Art Key/Value Mechanismus

Enterprise JavaBeans

- Message Driven Beans
 - Beispiel mit JCA – eMail Service
 - Szenario



Agenda

- Motivation
- J2EE - was bisher geschah ... ?
- Ausrichtung der neuen Spezifikation
- Web Services
- JSP und Java Servlet API
- Enterprise JavaBeans
- Deployment, Management & Monitoring
- Fazit & Ausblick
- Q & A



Deployment, Management & Monitoring

- Bestandteil der „Tools Standardization Initiatives for J2EE Platform“
- Ziele
 - Vereinfachung der J2EE Entwicklung
 - Verkürzen der „Time-to-Market“ Periode
 - bessere Kontrolle zur Laufzeit
- APIs
 - J2EE Management API JSR 77
 - J2EE Deployment API JSR 88
 - J2EE Debugging API ???

Deployment, Management & Monitoring

■ Management API

- Management Applikationen können die zu verwaltenden Objekte einer J2EE Applikation auffinden und interpretieren.
- Management Protokoll Spezifikationen versichern eine einheitliche Sicht für SNMP und WBEM Management Stationen
- J2EE Management Model als Basis
 - Managed Objects
 - Events
 - State
 - Performance

Discovery und Navigation

Notification

State Management

Performance Monitoring

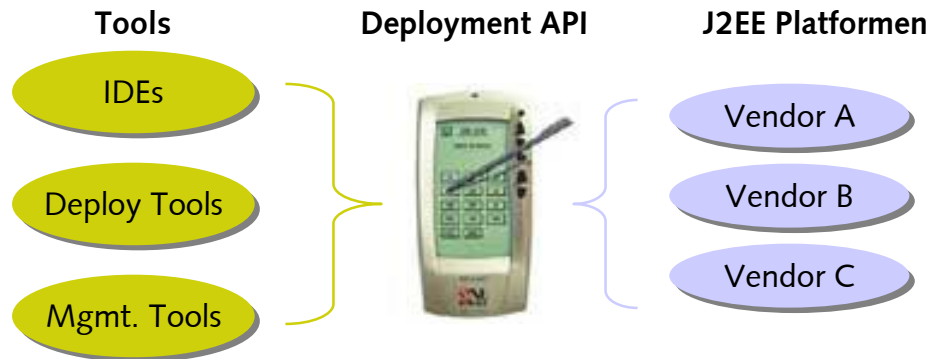
Deployment, Management & Monitoring

- Management API
 - J2EE Management EJB Komponenten
 - Zugriff auf das Management Model
 - JNDI Binding
 - pro Domain eine EJB notwendig
 - Protokoll Interoperabilität
 - CIM/WBEM durch CIM Model
 - SNMP durch SNMP MIB
 - offene Bereiche
 - Security Management
 - Accounting
 - Configuration Management
 - Fault Recovery

Deployment, Management & Monitoring

- Deployment API

- als universelle „Remote Control“



- Verantwortlichkeiten nach „Rollen“

- J2EE Deployer
 - J2EE Product Provider
 - J2EE Tool Provider

Deployment, Management & Monitoring

- Deployment
 - J2EE Deployer ...
 - nutzt den Deployment Manager
 - J2EE Product Provider ...
 - implementiert DeploymentFactory
 - implementiert DeploymentManager
 - J2EE Tool Provider ...
 - stellt Verbindung zum Deployment Manager her
 - stellt UI für den Deployment Manager zur Verfügung
 - stellt Laufzeitinformationen dar

Agenda

- Motivation
- J2EE - was bisher geschah ... ?
- Ausrichtung der neuen Spezifikation
- Web Services
- JSP und Java Servlet API
- Enterprise JavaBeans
- Deployment, Management & Monitoring

- Fazit & Ausblick
- Q & A



Fazit

- Fazit
 - Sun hat sehr viel getan, um eine *echte* Enterprise Spezifikation zu erzeugen
 - Teile der neuen Spezifikation sind lediglich Folgen aktueller Markttrends

 - J2EE Spezifikation und Entwicklungsprozess erreicht langsam für den Entwickler/Deployer unüberschaubare Dimensionen
 - Extremes Spezialwissen oder aber Vertrauen in externe Tools ist notwendig

Wie geht es weiter ... ?

- Wie geht es weiter ... ?
 - XML Data Binding
 - JNLP Support (Web Start) für das Deployment von Client Applikationen
 - J2EE Service Provider Interface (SPI)
 - Security APIs
 - SQLJ Part 0 – embedded SQL
 - Verbesserung der User Registrierung
 - Auditing Unterstützung für sicherheitsrelevante Events
 - Instanz basierte Zugriffskontrolle

Agenda

- Motivation
- J2EE - was bisher geschah ... ?
- Ausrichtung der neuen Spezifikation
- Web Services
- JSP und Java Servlet API
- Enterprise JavaBeans
- Deployment, Management & Monitoring

- Fazit & Ausblick
- Q & A



Question & Answers

Fragen ?

■ Links

- J2EE Home Page
 - java.sun.com/j2ee
- The Server Side
 - www.theServerSide.com
- Java Server Developer JumpPoint
 - www.javaskyline.com/jumppoint.html
- Java Performance Tuning
 - www.javaperformancetuning.com/tips
- UrbanCode EJB Benchmark
 - urbancode.com/projects/ejbbenchmark

■ Literatur

- Enterprise JavaBeans -
Monson-Haefel, O'Reilly
- Mastering Enterprise JavaBeans -
Ampler, Jewell, Roman - Wiley
- Patterns of Enterprise App. Architecture -
Martin Fowler - Addison-Wesley

Kontakt zu OpenKnowledge

OpenKnowledge GmbH

Bismarckstrasse 13

26122 Oldenburg

Tel. +49 441 4082-0

Fax +49 441 4082-111

<http://www.openknowledge.de>

