



## **EJB 2.x – Neuerungen in der EJB-Welt**

Version 1.0 - JAX 2003  
Lars Röwekamp & Jens Schumann

# Agenda

- Motivation
- EJBs - was bisher geschah ... ?
- Ausrichtung der EJB Version 2.x
  
- Local Components
- Container Managed Persistence
- Message Driven Beans
- Web Services
- Timer Service
  
- 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 EJB 2.1 Spezifikation:
  - *„The Enterprise JavaBeans architecture is a component architecture for the development and deployment of component based distributed business applications.“*
  - *„Applications written using the Enterprise JavaBeans architecture are scalable, transactional and multi-user secure.“*

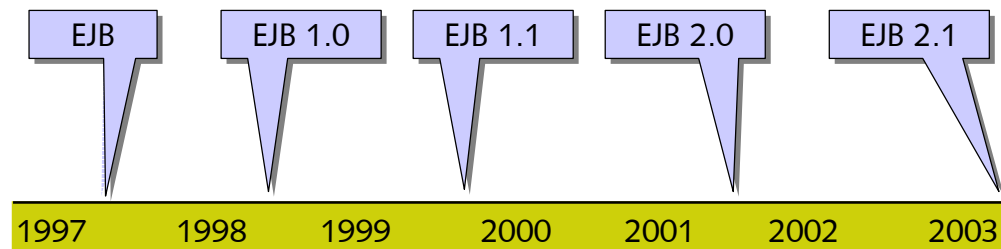
# Agenda

- Motivation
- EJBs - was bisher geschah ... ?
- Ausrichtung der EJB Version 2.x
- Local Components
- Container Managed Persistence
- Message Driven Beans
- Web Services
- Timer Service
- Fazit & Ausblick
- Q & A



## Was bisher geschah ...

### ■ Die EJB Zeitachse



- ~~Definition des Application Assembly~~
- ~~Standard Component Architecture~~
- ~~Verteilte Objekte, Remote-Objekte~~
- Definition als EJB Developer Views
- Applikationen
- Definition des <ejb-jar> Formats
- Adressierung von Development, Deployment und Runtime Aspekten

# Agenda

- Motivation
- EJBs - was bisher geschah ... ?
- Ausrichtung der EJB Version 2.x
- Local Components
- Container Managed Persistence
- Message Driven Beans
- Web Services
- Timer Service
- Fazit & Ausblick
- Q & A



## Ausrichtung der EJB Version 2.x

- Einführung einer Bean für asynchrones Processing – Message Driven Bean
- Einführung einer lokalen Sichtweise für bessere Performance bei lokalen Zugriffen
- Deutliche Verbesserung der Container Managed Persistence durch neues Konzept - Persistence Manager
- Verbesserung der Unterstützung von Relationen zwischen Entity Beans
- Einführung einer deklarativen Abfragesprache - EJB QL
- Unterstützung von Web Services
- Einführung eines „Schedulers“

# Agenda

- Motivation
- EJBs - was bisher geschah ... ?
- Ausrichtung der EJB Version 2.x
- Local Components
- Container Managed Persistence
- Message Driven Beans
- Web Services
- Timer Service
- Fazit & Ausblick
- Q & A



## Local Components

- EJB 1.x
  - Home und Remote Interfaces
- EJB 2.x
  - Local Home und Local Interface
  - Remote Home und Remote Interface
- Grundidee – Unterstützung verschiedener Clienttypen
  - Remote Clients, die in einer anderen VM als der EJB Container laufen
  - Local Clients, die in der selben VM wie der EJB Container laufen
- Vorteil (theoretisch) – Local Client Kommunikation kann optimiert werden, da kein RMI notwendig ist.

## Local Components

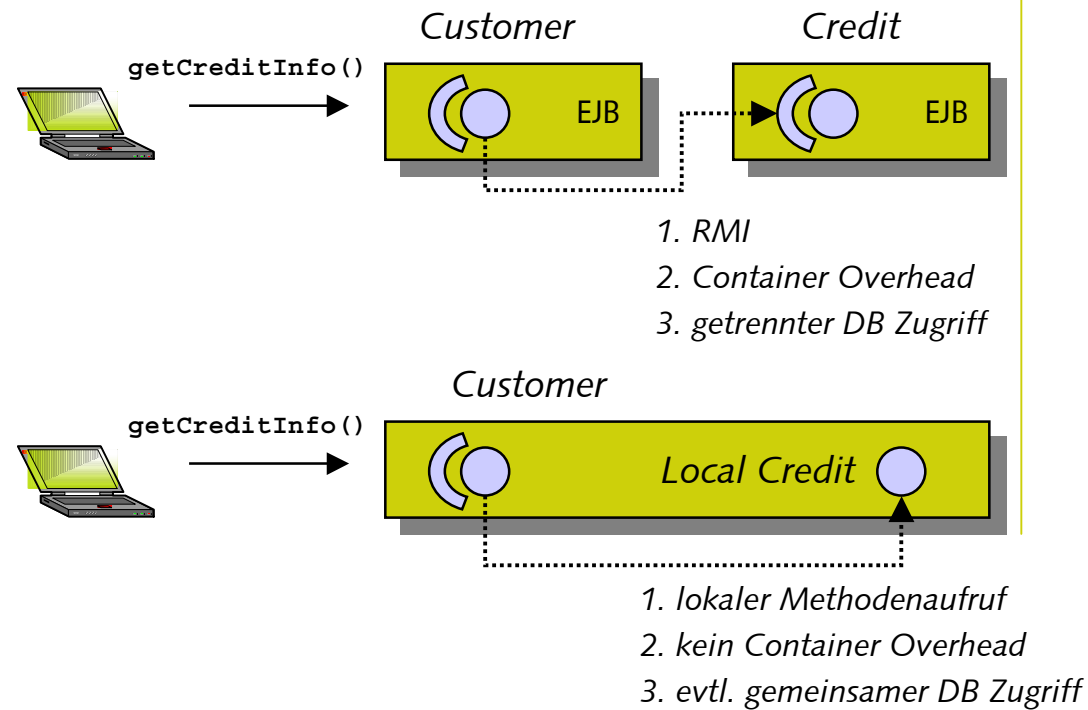
- Remote Clients
  - Overhead durch Kommunikation / RMI
  - Call by Value
  - grobgranular und schwergewichtig
- Local Clients
  - Optimierte Kommunikation
  - Call by Referenz
  - feingranular und leichtgewichtig
- Programmierung wie Remote, aber ...
  - kein werfen der RemoteException
  - Local Home Interface implementiert  
EJBLocalHome statt EJBHome
  - Local Interface Implementiert  
EJBLocalObject statt EJBObject

## Local Components

- Local Components bilden die Grundlage für die neue CMP Entity Bean Version
- Local Components können „Dependent Objects“ von Remote Components abbilden
- Achtung:
  - Local Components müssen immer in der selben VM wie der aufrufende Client liegen
  - Einsatz von Local Components ist eine Designentscheidung

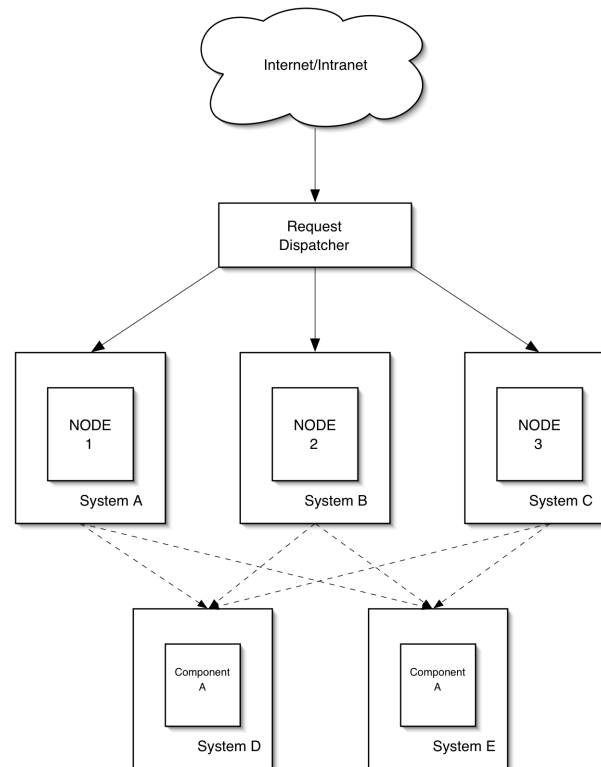
## Local Components

- Beispiel – Entity Beans für Kunden- und Kreditkarteninformationen



# Local Components

## ■ Problem des Distributed Computing



# Agenda

- Motivation
- EJBs - was bisher geschah ... ?
- Ausrichtung der EJB Version 2.x
- Local Components
- **Container Managed Persistence**
- Message Driven Beans
- Web Services
- Timer Service
- Fazit & Ausblick
- Q & A



## Container Managed Persistence

- CMP 2.x unterscheidet sich dramatisch von CMP 1.x
- CMP auch für Entity Beans die Relationen zu anderen EJBs oder „Dependent Objects“ besitzen
- Persistence Manager als zentrale Instanz
- Home und Select Methoden möglich
- CMP 2.x und CMP 1.x sind nicht kompatibel *aber* EJB 2.x Server Provider müssen beide Modelle unterstützen

## Container Managed Persistence

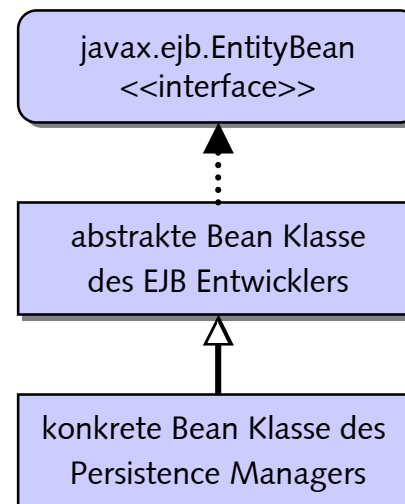
- CMP 1.x
  - Bean Developer ist verantwortlich für die persistenten Felder
  - persistente Felder können Java Primitive oder serialisierbare Klassen sein
  - Mapping zwischen persistenten Entity Bean Attributen und RDBMS erfolgt über DD mittels `<cmp-field>` Tag
  - in der Regel mapped eine Entity Bean auf genau eine Tabelle
  - Probleme bei Relationen zwischen mehreren Entity Beans
  - Fazit – Möglichkeiten der Entity Bean Spec stimmen nicht mit CMP Möglichkeiten überein.

## Container Managed Persistence

- CMP 2.x - Persistence Manager
  - neuer Contract zwischen Persistence Manager und CMP Entity Bean
  - Persistence Manager übernimmt die Aufgabe früherer BMP Generatoren (z.B. CocoBase von Thought Inc.)
  - komplexe Relationen möglich
    - Entity EJB zu Entity EJB
    - Entity EJB zu Dependent Objects
    - Dependent Object zu Dependent Object innerhalb einer Entity EJB
  - Mapping Informationen durch
    - EJB Abstract Persistence Schema
    - Deployment Descriptor

# Container Managed Persistence

- CMP 2.x - Persistence Manager
  - Entity Bean Implementierung ist eine abstrakte Klasse
  - persistente Felder sind abstrakte Klassen Attribute mit Zugriffsmethoden



# Container Managed Persistence

- CMP 2.x Beispiel – Address Bean 1/5
  - Entity Bean „Implementierung“

```
public abstract AddressBean implements
    javax.ejb.EntityBean {

    /* instance field */
    EntityBeanContext ejbContext;

    /* container-managed persistent fields */
    public abstract void setFirstName(String firstName);
    public abstract void setLastName(String lastName);

    /* container-managed relations */
    public abstract void setPhoneInfo(PhoneInfo info);
    ...
}
```

## Dependent Object:

- abstrakte Klasse
- nur via EJB
- Persistenz via PM

## Container Managed Persistence

- CMP 2.x Beispiel – Address Bean 2/5
  - Dependent Object „Implementierung“

```
public abstract PhoneInfo {  
  
    /* container-managed persistent fields */  
    public abstract void setCountyCode(String cc);  
    public abstract void setAreaCode(String ac);  
    public abstract void setPhoneNumber(String pn);  
  
    public abstract String getCountyCode();  
    public abstract String getAreaCode();  
    public abstract String getPhoneNumber ();  
}
```

## Container Managed Persistence

- CMP 2.x Beispiel – Address Bean 3/5
  - XML Deployment Descriptor

```
<ejb-jar>
  <enterprise-beans>
    <entity>
      <ejb-name>AddressEJB</ejb-name>
      ...
      <persistence-type>Container</persistence-type>
      ...
      <cmp-field>
        <field-name>firstName</field-name>
      </cmp-field>
      ...
    </entity >
  ...                               <!-- siehe naechste Folie -->
```

## Container Managed Persistence

- CMP 2.x Beispiel – Address Bean 4/5
  - XML Deployment Descriptor

```
...                               <!-- Fortsetzung -->
</enterprise-beans>
<dependents>
  <dependent>
    <dependent-class>PhoneInfo</dependent-class>
    <dependent-name>PhoneInfo</dependent-name>
    <cmp-field>countryCode</cmp-field>
    <cmp-field>areaCode</cmp-field>
    <cmp-field>phoneNumber</cmp-field>
    ...
  </dependent>
</dependents>
...                               <!-- siehe naechste Folie -->
```

# Container Managed Persistence

- CMP 2.x Beispiel – Address Bean 5/5
  - XML Deployment Descriptor

```
...                               <!-- Fortsetzung -->
<relationships>
  <ejb-relation>
    </ejb-relation-name>Address-PhoneInfo ...
    <ejb-relationship-role> ....
    </ejb-relationship-role>
  </ejb-relation>
</relationships>
...
</ejb-jar>
```

Source:  
AdressEJB

Relation:  
1:1, 1:n, n:m

CRM-Field:  
Name & Typ

Relationship:  
Adr\_has\_Tel

mehrere  
Relationships  
erlaubt

## Container Managed Persistence

- CMP 2.x - Home und Select Methoden
  - Home Methoden als quasi-statische Instanzmethoden
  - Home Methoden als Ersatz für Query Access via Stateless Session Bean
  - Select Methoden als „private“ Finder Methoden zum Aufsuchen von
    - EJB Local Objects
    - EJB Objects
    - CMP Fields

# Agenda

- Motivation
- EJBs - was bisher geschah ... ?
- Ausrichtung der EJB Version 2.x
- Local Components
- Container Managed Persistence
- **Message Driven Beans**
- Web Services
- Timer Service
- Fazit & Ausblick
- Q & A



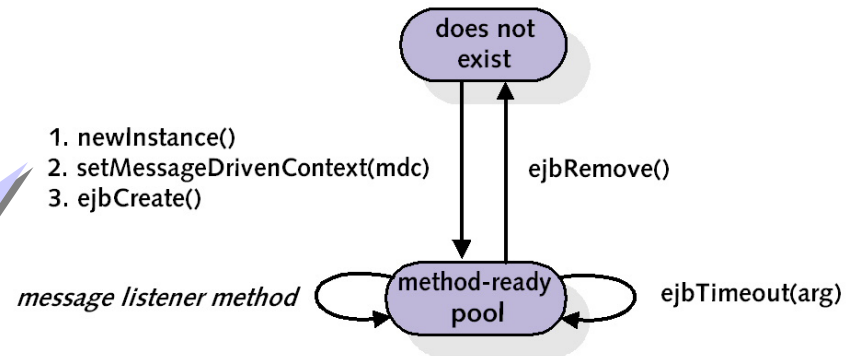
## Message Driven Beans

- asynchrone Enterprise JavaBeans
- werden durch Nachrichten aktiviert
  - nur JMS Nachrichten in EJB Spec 2.0
  - auch andere Messaging Systeme via Java Connector Architecture 1.5 seit EJB Spec 2.1
    - Transaktionen des EIS können übernommen werden
- zustandslos wie Stateless Session Beans
- nutzen alle Vorteile des EJB Containers, wie Sicherheit, Transaktionen, ...
- Message Driven Beans bestehen ...
  - aus Bean Implementierung und XML DD
  - nicht aus Komponenten Interfaces, da sie *nie* direkt angesprochen werden

# Message Driven Beans

## ■ Message Driven Bean - Life Cycle

Zugriff auf  
Security und  
Transaction  
Informationen  
via `ejbContext`



<i>message listener method</i>	Aktion die aus einer Client Message resultiert
<code>ejbCreate()</code>	Aktion durch Container angestoßen

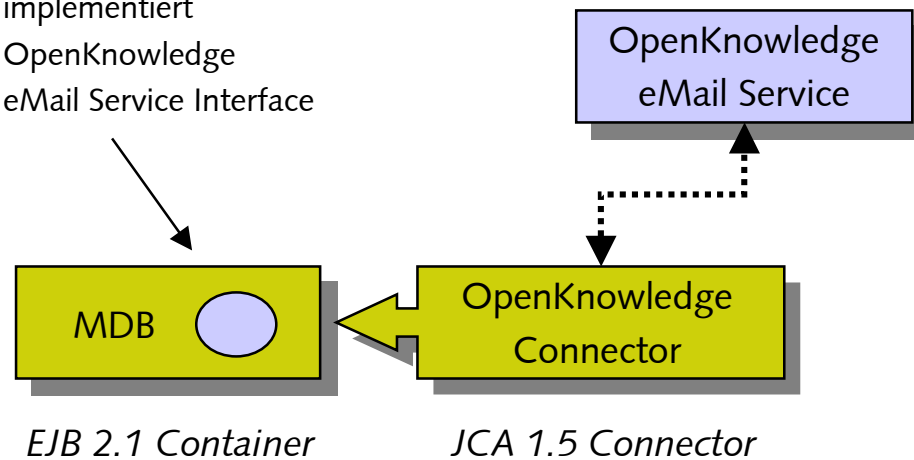
## Message Driven Beans

- JMS MDBe implementieren
  - `javax.ejb.MessageDrivenBean`
  - `javax.jms.MessageListener`
- JCA MDBs implementieren
  - `javax.ejb.MessageDrivenBean`
  - `com.vendorxyz.ABCListener`
- Nachrichtenverarbeitung via Callback:
  - JMS - `onMessage()`
  - sonst je nach Interface
- Problem im DD durch neu gewonnene Flexibilität – keine JMS spezifischen Tags mehr sondern *Activation Properties*

# Message Driven Beans

- MDB Beispiel mit JCA – eMail Service
  - Szenario

implementiert  
OpenKnowledge  
eMail Service Interface



# Message Driven Beans

- MDB Beispiel mit JCA – eMail Service
  - Interface definieren

```
package de.openknowledge.email;

import javax.mail.Message;

/** eMail Service Listener Interface */
public interface EMailListener {
    public void receiveMessage(Message msg);
}
```

# Message Driven Beans

- MDB Beispiel mit JCA – eMail Service
  - Interface implementieren

```
import javax.mail.Message;
import javax.mail.MimeMessage;

public class EmailServiceBean implements
    javax.ejb.MessageDrivenBean,
    de.openknowledge.email.EMailListener {

    ...

    /** handle incoming email */
    public void receiveMessage(Message msg) {
        MimeMessage mimeMsg = (MimeMessage)msg;
        Address[] addressArray = mimeMsg.getForm();
        ...
    }
}
```

# Agenda

- Motivation
- EJBs - was bisher geschah ... ?
- Ausrichtung der EJB Version 2.x
  
- Local Components
- Container Managed Persistence
- Message Driven Beans
- **Web Services**
- Timer Service
  
- Fazit & Ausblick
- Q & A

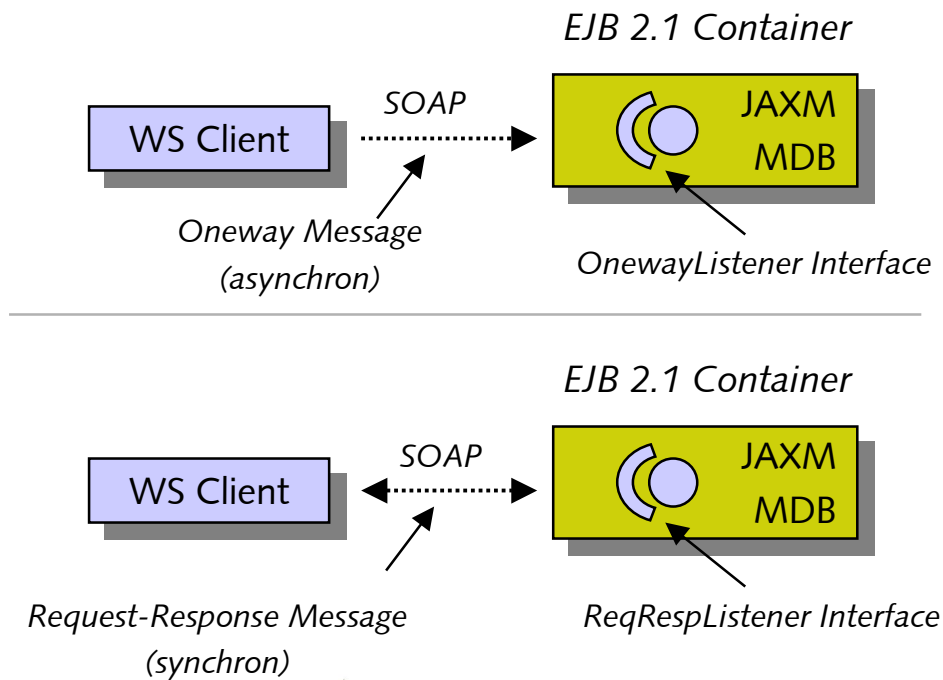


## Web Services

- SOAP basierte Web Services via ...
  - Stateless Session Bean (automatisch)
  - Message Driven Beans (manuelle)
- SOAP 1.1 kompatibel
  - .NET,
  - Apache Axis,
  - PHP ...
- Die Basis bilden J2EE XML APIs
  - JAXM und SAAJ für Message Driven Beans – Document Based
  - JAX-RPC als Service Endpoint Interface (SEI) für Stateless Session Beans durch Java RMI over SOAP – Remote Procedure Call

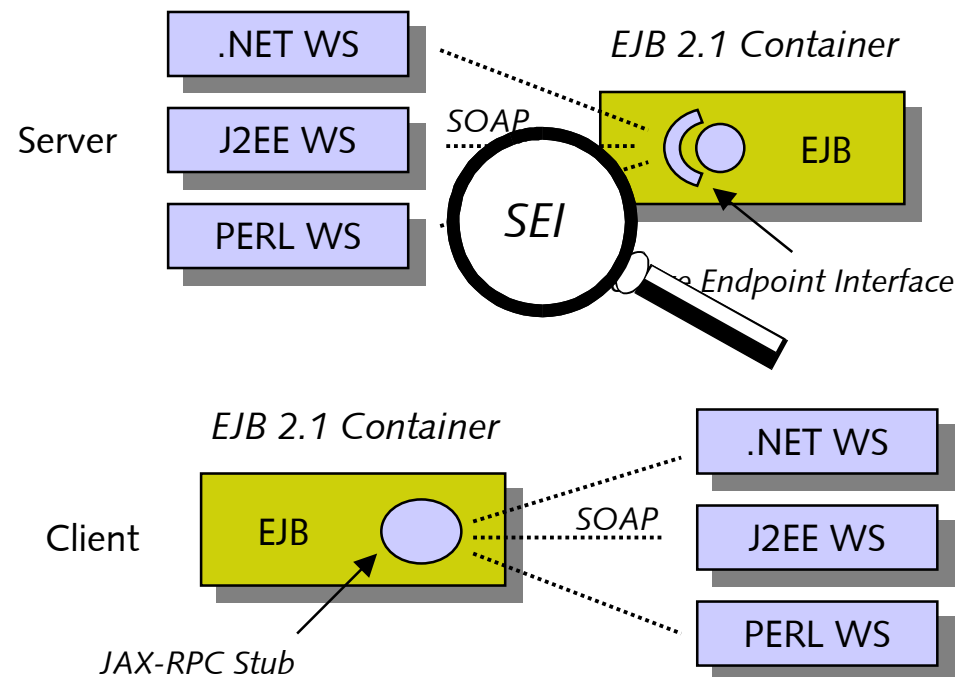
## Web Services – EJBs und JAXM / SAAJ

- MDBs Beans können Dokumenten basierte SOAP Messages als „Oneway“ oder „Request/Response“ Listener verarbeiten



# Web Services – EJBs und JAX-RPC

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



## Web Services – Service Endpoint Interface

- 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 `webservices.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 – Beispiel:
  - Web Service „OpenKnowledge“ ansprechen und dort via RPC die Service Operation „getKnowledge“ zum Erlangen wertvollen Wissens aufrufen

## 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 okService = (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
- EJBs - was bisher geschah ... ?
- Ausrichtung der EJB Version 2.x
  
- Local Components
- Container Managed Persistence
- Message Driven Beans
- Web Services
- Timer Service
  
- Fazit & Ausblick
- Q & A



## Timer Service

- EJB Timer Services ...
  - als Zeitplanungswerkzeug
  - Support von zeitbasierten Workflows
  - sind nutzbar durch
    - Stateless Session Beans
    - Message Driven Beans
    - Entity Beans via Primary Key
  - benachrichtigen via Callback Methoden
    - in Intervallen - Interval-Based
    - zu einem festen Zeitpunkt - Single-Action
  - sind persistent und überleben somit einen Server Crash

## Timer Service

- 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

### ■ Timer Service Beispiel – Timer erstellen:

```
...  
  
/** Timer auf 10 Tage ab aktuelles Datum setzen **/  
Calendar time = Calendar.getInstance();  
time.time.add(Calendar.DATE, 30);  
Date date = time.getTime();  
  
TimerService timerService =  
    ejbContext.getTimerService();  
timerService.createTimer( date, null);  
  
...
```

## Timer Service

### ■ Timer Service Beispiel – Timer nutzen:

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

## Timer Service

- 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

# Agenda

- Motivation
- EJBs - was bisher geschah ... ?
- Ausrichtung der EJB Version 2.x
  
- Local Components
- Container Managed Persistence
- Message Driven Beans
- Web Services
- Timer Service
  
- Fazit & Ausblick
- Q & A



## Fazit

- Fazit
  - EJB 2.x ist ein deutlicher Fortschritt gegenüber 1.1
  - flexibleres CMP Model
    - „beliebige“, portable Relationen
    - Relationen zwischen EJBs, Depending Objects und serialisierbaren Klassen
  - Message Driven Beans
    - JMS seit EJB 2.0
    - JCA Connetoren seit EJB 2.1
  - Web Service Support seit EJB 2.1
  - Timer Service Support seit EJB 2.1
  - und .... lokale Komponenten

## Wie geht es weiter ... ?

- Wie geht es weiter ... ?
  - Erweiterung der EJB QL
    - Unterabfragen
  - Read Only EntityBeans mit CMP
  - Pluggability für den Persistence Manager
  - Unterstützung von Message Interceptoren zur Interaktion mit Message vor deren Verarbeitung
  - Vererbung auf Komponentenebene

# Agenda

- Motivation
- EJBs - was bisher geschah ... ?
- Ausrichtung der EJB Version 2.x
- Local Components
- Container Managed Persistence
- Message Driven Beans
- Web Services
- Timer Service
- Fazit & Ausblick
- Q & A



## Question & Answers

Fragen ?

## Links & Literatur

### ■ Links

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

## Links & Literatur

### ■ Literatur

- Enterprise JavaBeans -  
*Monson-Haefel, O'Reilly*
- Mastering Enterprise JavaBeans -  
*Ampner, 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>

