Enterprise JavaBeans™ Komplett

Dezember 2004

Enterprise JavaBeans -

Neuerungen der Gegenwart (EJB 2.1) und Zukunft (EJB 3.0)

Präsentation Arbeitskreis Objekttechnologie
Oliver Ihns, Stefan M. Heldt
Resco GmbH

Über uns

Enterprise JavaBeans™ Komplett

Oliver Ihns (oliver.ihns@resco.de)

- Leiter Competence Area „Enterprise Application Architecture“
- Senior Consultant & Trainer bei der IT-Unternehmensberatung Resco in Hamburg
- Mitglied der EJB 3.0 Expert Group im JCP bei Sun Microsystems
- Co-Autor und Herausgeber Fachbuch „Enterprise JavaBeans komplett – EJB 2.1“
- Autor diverser Fachartikel
- Schwerpunkte
  - Software-Architekturen,
  - Verteilte (objekt-orientierte) Systeme auf Basis J2EE / EJB, CORBA, MOM…

Stefan M. Heldt (stefan.heldt@resco.de)

- Senior Consultant bei Resco in der Competence Area „Enterprise Application Architecture“
- Trainer bei der IT-Unternehmensberatung Resco in Hamburg
- Co-Autor Fachbuch „Enterprise JavaBeans komplett – EJB 2.1“
- Autor diverser Fachartikel
- Schwerpunkte
  - Software-Architekturen
  - J2EE
  - Internet-Technologien

Enterprise JavaBeans 2.1 (JSR 153)

Veröffentlicht: Final Release November 2003

J2EE™ 1.4 SDK

Überblick

- Web Service-Integration (WSI) in EJB
- Connector-based Message-driven Beans
- Jederzeit zu Ihren Diensten – Der EJB Timer Service
- EJB QL, die Zweite
- Resümee – EJB 2.1
- Ausblick auf EJB 3.0

Web Service-Integration in EJB

Enterprise JavaBeans Komplett

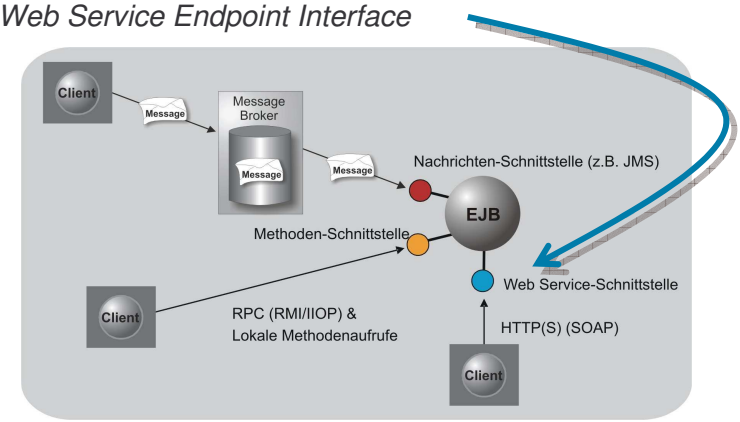
- EJB-Komponenten können ab EJB Version 2.1 als **Web Service** angesprochen werden
- Ein weitere Schritt Richtung integrativer Cross-Plattform
- Mit Integration der Web Service-Fähigkeit halten
  - SOAP 1.1
  - WSDL 1.1 und
  - JAX-RPC (Java API for XML RPC)

Einzug in die EJB-Welt.

Web Service-Integration in EJB

Enterprise JavaBeans Komplett

- Zusätzlich zu *Remote Interface*, *Local Interface* und JMS-Schnittstelle (mit EJB 2.1 genereller: JCA-Schnittstelle) gibt es nun die Web Service-Schnittstelle:
  - *Web Service Endpoint Interface*



Web Service-Integration in EJB

Enterprise JavaBeans™ Komplett

Können alle EJB-Typen als Web Service fungieren?

- Stateless Session Bean — ja
- Stateful Session Bean — nein
- Entity Bean — nein
- Message-driven Bean — ja – aber…

Web Service-Integration in EJB

Enterprise JavaBeans™ Komplett

- Stateless Session Beans können uneingeschränkt als Web Service fungieren
  - Web Services sind per se statuslos.
  - Web Services verwenden hauptsächlich HTTP als Protokoll
  - HTTP ist zustandslos und kennt keine Objektidentitäten
- Stateful Session Beans und Entity Beans können nicht als Web Service eingesetzt werden
  - Setzen Objektidentität und Zustand voraus ➔ siehe oben
- Message-driven Beans sind ein Sonderfall
  - JAX-RPC, JAXM (Details folgen…)

Stateless Session Beans als WS

Enterprise JavaBeans Komplett

- Schritte einer Stateless Session Bean zum Web Service…
  - Muss ein *Web Service Endpoint-Interface* besitzen
  - Muss eine *WSDL-Beschreibung* der angebotenen Dienste (Methoden) besitzen
  - (Toolgestützte Generierung der Stub-Klassen)
- Web Service Endpoint Interface ist zusätzlich spezifiziert zu…
  - Home Interface (Remote Home und Local Home)
  - Component Interface (Remote und Local)

Stateless Session Beans als WS

Enterprise JavaBeans Komplett

- Web Service Endpoint Interface ist normales Java-Interface
- Muss von **`java.rmi.Remote`** erben
- Deklariert alle Geschäftsmethoden, die als Web Service-Operationen angeboten werden sollen
- Parameter und Rückgabetypen der deklarierten Methoden müssen JAX-RPC konform sein. (http://java.sun.com/xml/jaxrpc/index.html)

Stateless Session Beans als WS

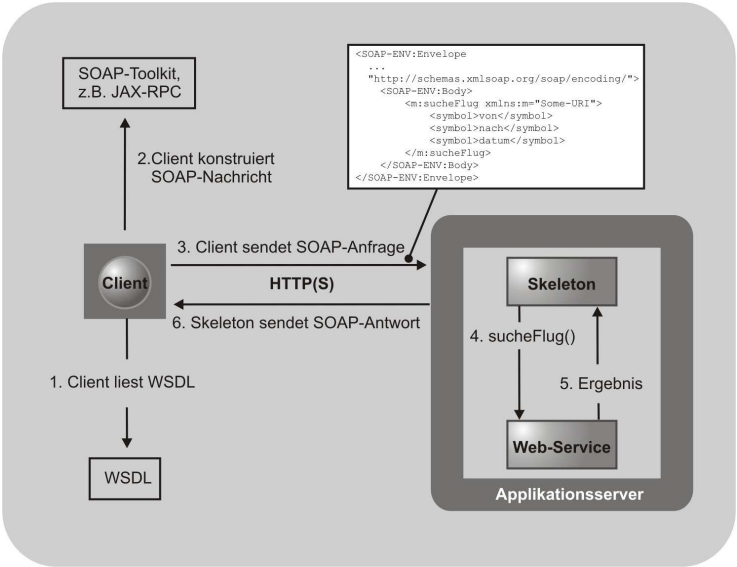
- Beispiel eines Web Service Endpoint Interfaces

```
public interface FlugSucheWS extends java.rmi.Remote {
  public String[] sucheFlug(String von, String nach, String datum)
      throws RemoteException;
}
```

- Beispiel der entsprechenden WSDL-Beschreibung

```
...
  <message name=„getVonRequest“> <part name=„von“ type=„xs:string“/>...
  <portType name=„FlugSucheWS“>
    <operation name=„sucheFlug“>
      <input message=„getVonRequest“/>
      ...
    </operation>
  </portType>
...
```


Ablauf einer Web Service-Kommunikation



Web Service-Integration in EJB

Enterprise JavaBeans™ Komplett

- Integration der Web Service-Fähigkeit führt zu Vorteilen:
  - Web Service-Schnittstelle einer EJB-Komponente fördert die lose Kopplung zwischen Client und EJB
  - Client besitzt keine Kenntnis darüber, dass der Dienstleister hinter einem Web Service eine EJB-Komponente ist
  - Nicht mehr nur Java-Clients sind Nutzer von EJB-Komponenten, sondern Clients fast aller gängigen Programmiersprachen
    - Voraussetzung: Unterstützung von Web Services in jeweiligen Sprache bzw. Technologie

Web Service-Integration in EJB

Enterprise JavaBeans™ Komplett

- Integrierte Kommunikationsparadigmen prädestinieren EJBs dazu, in der Mittelschicht integrierende Geschäfts-logik zu beherbergen
  - Kommunikation über RPC-Mechanismen
  - Kommunikation über Message-Oriented Middleware
  - Kommunikation über Web Service-Mechanismen (z.B. SOAP)
- „Kleber“ zwischen verschiedenen Welten
  - Programmiersprachen (Java, C#, …)
  - Kommunikationsparadigmen (MOM, RPC, SOAP)
  - Rechnersysteme (Hostwelt, dezentrale Server, Clients)

Web Service-Integration in EJB

Enterprise JavaBeans™ Komplett

- Weiterer Schritt zur integrativen Cross Plattform!
- Synergie-Effekte durch Verwebung von Komponenten-orientierter Architektur und Service-orientierter Architektur (SOA)
- EJBs weisen nun einen höheren Integrationsgrad für Service-orientierte Architekturen (SOA) auf.

Überblick

Enterprise JavaBeans™ Komplett

- Web Service-Integration (WSI) in EJB
- Connector-based Message-driven Beans
- Jederzeit zu Ihren Diensten – Der EJB Timer Service
- EJB QL, die Zweite
- Resümee – EJB 2.1
- Ausblick auf EJB 3.0

Connector-based MDBs

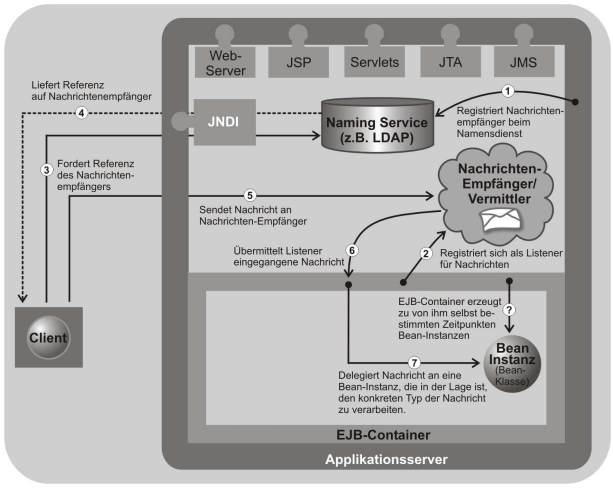
Enterprise JavaBeans™ Komplett

- Seit EJB 2.0 gibt es Message-driven Beans
  - Entkoppelte, für Clients transparente Komponente
  - Kommunikation erfolgt über Nachrichten (MOM)
  - Java Message Service (JMS) - basiert
- Mit EJB 2.1 Erweiterung von JMS- auf JCA-Anbindung
  - Können dadurch beliebige Nachrichtentypen verstehen
  - Verwendet wird JCA 1.5
- Registrieren sich als Empfänger von über JCA Konnektoren in Applikationsserver geroutete Nachrichten

Erweiterung von JMS auf JCA

Enterprise JavaBeans™ Komplett

- MDBs reagieren als Empfänger auf JCA-konforme Konnektoren.



Erweiterung von JMS auf JCA

Enterprise JavaBeans™ Komplett

- MDB muss die JCA-Konnektor-spezifische Callback-Methode des jeweiligen Nachrichtentyps implementieren
  - analog der `onMessage()`–Methode für JMS
  - definiert im jeweiligen MessageListener-Interface des Nachrichtentyps

Erweiterung von JMS auf JCA

Enterprise JavaBeans™ Komplett

- Mit Einzug von JCA entfallen JMS Tags für MDBs, stattdessen werden die *ActiviationProperties* von JCA genutzt.

2.0

```
enterprise-beans>
 ...
 <message-driven>
  <ejb-name>MyMDB</ejb-name>
  <ejb-class>com.foo.MyMDBBean</ejb-class>
  <transaction-type>Container</transaction-type>
  <message-selector>MessageFormat = 'Version 3.4'
   </message-selector>
  <acknowledge-mode>Auto-acknowledge</acknowledge-mode>
  <message-driven-destination>
    <destination-type>javax.jms.Queue</destination-type>
  </message-driven-destination>
 </message-driven>
 ...
</enterprise-beans>
```

2.1

```
<enterprise-beans>
 ...
  <message-driven>
    <ejb-name>MyMDB</ejb-name>
    <ejb-class>com.foo.MyMDBBean</ejb-class>
    <messaging-type>javax.jms.MessageListener</messaging-type>
    <transaction-type>Container</transaction-type>
    <message-destination-type> javax.jms.Queue </message-destination-type>
    <activation-config>
      <activation-property>
        <activation-config-property-name>destinationType
        </activation-config-property-name>
        <activation-config-property-value>javax.jms.Queue
        </activation-config-property-value>
      <activation-property>
      <activation-property>
        <activation-config-property-name>messageSelector
        </activation-config-property-name>
        <activation-config-property-value>MessageFormat = 'Version 3.4'
       </activation-config-property-value>
      <activation-property>
        ...
    </activation-config>
    ...
  </message-driven>
 ...
</enterprise-beans>
```

Erweiterung von JMS auf JCA

Enterprise JavaBeans Komplett

- Beispiel 1
  Bean implementiert **`javax.jms.MessageListener`**, wenn sie auf JMS-Nachrichten reagieren soll

```
public interface MessageListener {
  void onMessage(javax.jms.Message message);
}
```

- Beispiel 2
  Bean implementiert **`javax.xml.messaging.OneWayListener`**, wenn sie auf JAXM-Nachrichten reagieren soll

```
public interface OneWayListener {
  void onMessage(javax.xml.soap.SOAPMessage message );
}
```

JAXM: Java API for XML messaging

Connector-based MDBs

Enterprise JavaBeans Komplett

- Neben JMS MDBs ist eine weitere Ausprägung die JAXM MDB
  - SOAP-basiertes Messaging auf Basis von JAXM
    (Java API for XML Messaging)
    - javax.xml.messaging.OneWayListener
    - javax.xml.messaing.ReqRespListener
  - Damit implizit MDB-basierte Web Services
- JAXM nicht in J2EE 1.4 Standard
  - JAXM überschneidet sich mit JAX-RPC, JMS, SAAJ…
  - Verfügbarkeit herstellerabhängig
- Ergo: Web Service fähige MDB nicht per se Standard

Überblick

Enterprise JavaBeans™ Komplett

- Web Service-Integration (WSI) in EJB
- Connector-based Message-driven Beans
- **Jederzeit zu Ihren Diensten – Der EJB Timer Service**
- EJB QL, die Zweite
- Resümee – EJB 2.1
- Ausblick auf EJB 3.0

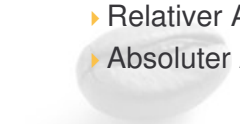
Jederzeit zu Ihren Diensten

Enterprise JavaBeans™ Komplett

- Zeitgesteuerte und automatische Ausführung von Geschäftsprozessen ist wichtig für Unternehmensanwendungen
  - z.B. nächtliche Erstellung von Reports
- Sicher kennen Sie das Scheduling-System *cron* aus der UNIX-Welt
- Bisher kein standardisiertes Scheduling-System in der J2EE-Welt
- Lediglich proprietäre Systeme
  - z.B. Timer Service des BEA WebLogic
- Dadurch Verlust der Portabilität

Jederzeit zu Ihren Diensten

- EJB 2.1 definiert einen Scheduling-Service
  - Der EJB Timer Service
- Standardisierte Schnittstelle
- Muss von jedem Hersteller implementiert werden
- Automatische Ausführung von EJBs zu bestimmten Zeitpunkten
  - Einmalausführung (single event timer)
  - Intervallausführung (interval timer)
  - Relativer Ausführungszeitpunkt (in Millisekunden)
  - Absoluter Ausführungszeitpunkt (Datum + Uhrzeit)



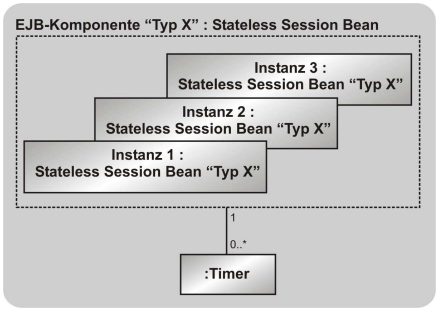
Jederzeit zu Ihren Diensten

Enterprise JavaBeans™ Komplett

- Nutzbar mit…
  - Stateless Session Beans
  - Message-driven Beans
  - Entity Beans

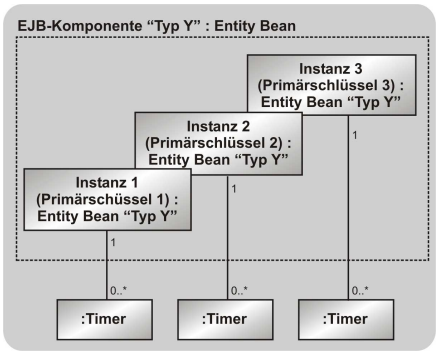
Timer Service – Anwendung (1)

- Stateless Session Beans und Message-driven Beans haben keinen Zustand
- Daher verwendet der Timer Service beliebige Instanzen
- Typische Aufgabe: Batch-Jobs
- Beziehen sich auf viele oder keine Entitäten
- z.B. Aufträge, die nachts abgearbeitet werden
- oder Versendung von Reports

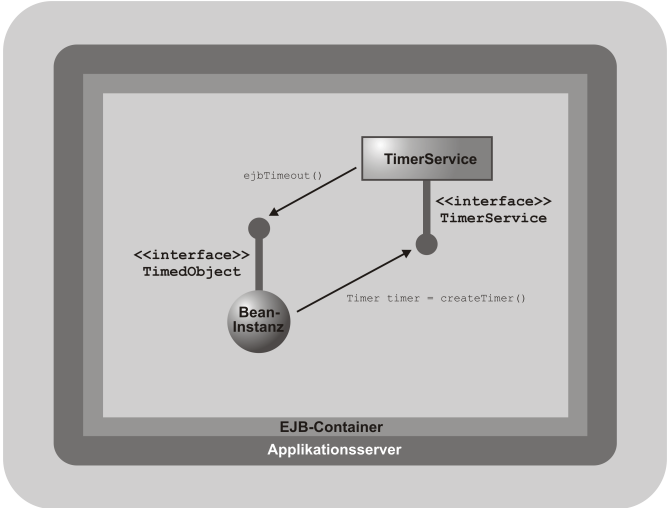


Timer Service – Anwendung (2)

- Entity Beans beziehen sich auf fachliche Entitäten
- Daher verwendet der Timer Service eine bestimmte Instanz
- bzw. eine bestimmte fachliche Entität
- Aufgabe bezieht sich auf fachliche Entität
- z.B. automatischer Kauf bzw. Verkauf von Wertpapieren
- Wertpapier überprüft periodisch seinen Kurs und platziert ggf. Order

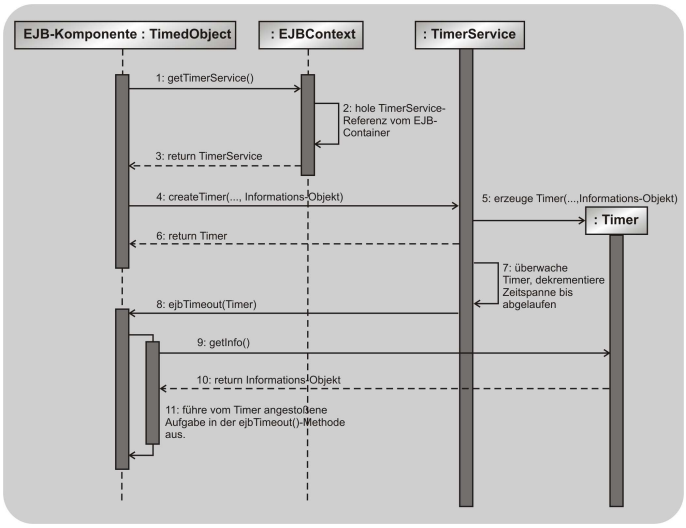


Timer Service - Architektur



- Installierte Timer sind persistent.

Timer Service - Ablauf



Interface TimerService

- `public Timer createTimer(long duration, Serializable info)`

  Informationsobjekt

- `public Timer createTimer(long initialDuration, long intervalDuration, Serializable info)`
- `public Timer createTimer(Date expiration, Serializable info)`
- `public Timer createTimer(Date initialExpiration, long intervalDuration, Serializable info)`
- `public Collection getTimers()`

trotz Angabe von Millisekunden nicht echtzeitfähig

Interface Timer

- `public void cancel()`
- `public long getTimeRemaining()`
- `public Date getNextTimeout()`
- `public Serializable getInfo()`

  Informationsobjekt

- `public TimerHandle getHandle()`

Timer Service - Beispiel


```
public class ReporterBean implements SessionBean , TimedObject {
  public void setupDailyReport(String recipient) {
    TimerService theTimerService = ctx.getTimerService();
    theTimerService.createTimer(86400000, 86400000, recipient);
    //86400000 ms == 24 h
  }

  public void ejbTimeout(Timer timer) {
    Report oneReport = generateReport();
    String recipient = (String)timer.getInfo();
    sendReport(oneReport, recipient);
  }

  public void tearDownTimers() {
    TimerService theTimerService = ctx.getTimerService();
    Iterator iter = theTimerService.getTimers().iterator();
    while (iter.hasNext()) {
      Timer theTimer = (Timer)iter.next();
      theTimer.cancel();
    }
  }
}
```


Timer Service - Transaktionen

- Erzeugung und Löschung von Timern innerhalb des jeweiligen Transaktionskontextes
  - d.h. erzeugter Timer nach Rollback nicht mehr vorhanden
  - und gelöschter Timer nach Rollback wieder da
- Für **`ejbTimeout()`** kann ein beliebiges Transaktionsattribut angegeben werden
  - Wie bei jeder anderen Methode einer EJB

Überblick

- Web Service-Integration (WSI) in EJB
- Connector-based Message-driven Beans
- Jederzeit zu Ihren Diensten – Der EJB Timer Service
- EJB QL, die Zweite
- Resümee – EJB 2.1
- Ausblick auf EJB 3.0

EJB QL, die Zweite

- Einführung mit EJB 2.0
- Dient zur Beschreibung von Abfragen
  - Unabhängig von der Persistenzschicht
  - In Finder-Methoden
  - In Select-Methoden
- Erweiterung in EJB 2.1 um
  - ORDER BY
  - Aggregatfunktionen
  - Modulo-Funktion
  - Verwendung von Eingabeparametern in einigen Ausdrücken

EJB QL – ORDER BY

Enterprise JavaBeans Komplett

- Bekannt aus SQL, bisher vermisst in EJB QL
- Erlaubt die Sortierung der Ergebnismenge
- Folgt auf die WHERE Klausel einer Abfrage
- Angabe der Sortierreihenfolge durch
  - ASC, aufsteigend (Standardverhalten)
  - DESC, absteigend
- Sortierkriterien müssen in der SELECT Klausel vorhanden sein
- Mehrere Sortierkriterien werden durch Komma getrennt

ORDER BY – Beispiel 1

Enterprise JavaBeans Komplett

```
SELECT OBJECT(f)
FROM Flug f, IN (f.reservierungen) r
WHERE r.datum >= ?1
ORDER BY f.flugNr
```

- Sortierung der Flüge nach Flugnummer
- Sortierung nach Reservierungsdatum wäre nicht möglich
  - Reservierung ist nicht Bestandteil der SELECT Klausel

ORDER BY – Beispiel 2

Enterprise JavaBeans Komplett

```
SELECT OBJECT(f)
FROM Flug f, IN (f.reservierungen) r
WHERE r.datum >= ?1
ORDER BY f.start, f.flugNr DESC
```

- Sortierung der Flüge aufsteigend nach Startort des Fluges
- Mehrere Flüge mit dem selben Startort werden absteigend nach der Flugnummer sortiert

EJB QL - Aggregatfunktionen

Enterprise JavaBeans Komplett

- Aggregatfunktionen arbeiten auf einer Menge von Zwischenergebnissen
- Aggregatfunktionen liefern als Ergebnis einen Zahlenwert
- Finder-Methoden müssen immer Entitäten liefern
- Daher nur in Select-Methoden verwendbar

Funktion	Beschreibung	Beispiel
`AVG`	Durchschnitt aller Werte des selektierten Feldes (durchschnittlicher Flugdauer)	`SELECT AVG(f.dauer) FROM Flug f`
`MAX`	Maximum aller Werte des selektierten Feldes (längste Flugdauer)	`SELECT MAX(f.dauer) FROM Flug f`
`MIN`	Minimum aller Werte des selektierten Feldes (kürzeste Flugdauer)	`SELECT MIN(f.dauer) FROM Flug f`
`SUM`	Summe aller Werte des selektierten Feldes (Summe aller Flugdauern)	`SELECT SUM(f.dauer) FROM Flug f`
`COUNT`	Anzahl der selektierten Entitäten oder Felder (Anzahl der Flüge)	`SELECT COUNT(f) FROM Flug f`

EJB QL – Modulo-Funktion

Enterprise JavaBeans Komplett

- Verwendung ausschließlich in der WHERE Klausel
- Beispiel:
  - ```
    SELECT OBJECT(f)
    FROM Flug f
    WHERE MOD(f.dauer, 1) > 0
    ```
  - Selektiert alle Flüge, deren Dauer nicht in ganzen Stunden angegeben ist

EJB QL - Eingabeparameter

Enterprise JavaBeans Komplett

- Folgende Ausdrücke dürfen Eingabeparameter enthalten
- [NOT] IN
  - ```
    SELECT OBJECT(f)
    FROM Flug f
    WHERE f.start IN (?1, ?2, ?3)
    ```
- IS [NOT] NULL
  - ```
    SELECT OBJECT(f)
    FROM Flug f
    WHERE ?1 IS NOT NULL AND f.start = ?1
    ```
- LIKE
  - ```
    SELECT OBJECT(f)
    FROM Flug f
    WHERE f.start LIKE ?1
    ```

Überblick

- Web Service-Integration (WSI) in EJB
- Connector-based Message-driven Beans
- Jederzeit zu Ihren Diensten –
  Der EJB Timer Service
- EJB QL, die Zweite
- Resümee – EJB 2.1
- Ausblick auf EJB 3.0

Resümee – EJB 2.1

- Gravierendste Änderung ist die Web Service-Fähigkeit
  - Positioniert EJB als integrative Cross-Plattform
  - Öffnet EJB noch weiter gegenüber anderen Technologien
- Timer Service vereinfacht periodische oder langläufige Geschäftsprozesse
  - Kein Ersatz für komplexe Zeitsteuerung durch vollwertige Scheduling Systeme
- Alle anderen Änderungen dienen der Stabilisierung und Erweiterung der EJB-Technologie

Resümee – EJB 2.1

Enterprise JavaBeans™ Komplett

Applikationsserver, die EJB 2.1 implementieren… (Stand Ende September 2004)

JBoss 4.0

Borland® 6.x Enterprise Server

JEUS 5.0

Sun Java System Application Server Platform Edition 8

JOnAS 4.1.4

T4 Trifork

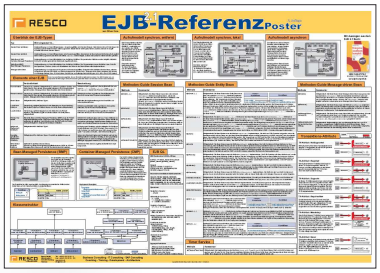
IBM WebSphere software 6.0

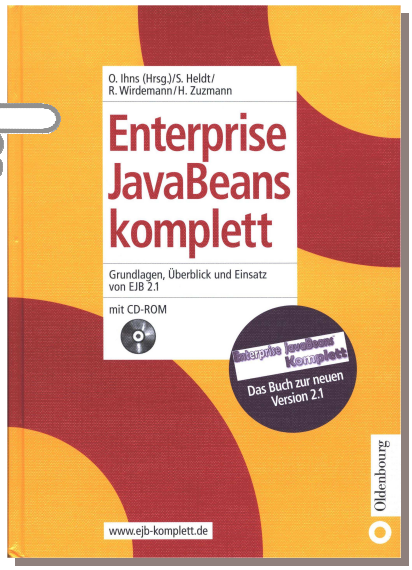
ORACLE 10g APPLICATION SERVER

Resümee – EJB 2.1

Enterprise JavaBeans™ Komplett

Mehr Infos zu EJB 2.1 gibt's im aktuellen Buch und in der Übersicht als DIN A0-Poster von Resco





Überblick

- Web Service-Integration (WSI) in EJB
- Connector-based Message-driven Beans
- Jederzeit zu Ihren Diensten – Der EJB Timer Service
- EJB QL, die Zweite
- Resümee – EJB 2.1
- Ausblick auf EJB 3.0

Ausblick auf EJB 3.0

Enterprise JavaBeans 3.0

JSR 220 im JCP

Java COMPATIBLE ENTERPRISE EDITION

J2EE ™ 5.0 SDK

Status:

Men at work!

Ausblick auf EJB 3.0

Enterprise JavaBeans Komplett

09/2003 EJB 3.0 Expert Group im September 2003 formiert
- Spezifikation in Arbeit…

08/2004 Erster öffentlicher Draft (Early Draft) August 2004

09/2004 JDO (JSR243) geht in EJB 3.0 (JSR 220) auf!!!
- Im Rahmen von EJB 3.0 separate Persistenzspezifikation
- Führende JDO-Experten stoßen zum EJB 3.0-Team

Q1/2006 Finale Veröffentlichung Q1 / 2006

- Kommt im Rahmen J2EE 5.0 (JSR244) vormals J2EE1.5

Zu verfolgen unter: Java Community Process http://www.jcp.org/en/jsr/detail?id=220

Ausblick auf EJB 3.0

Enterprise JavaBeans Komplett

- Hauptziele EJB 3.0

Vereinfachung^{n}

Ease of development
Ease of use

Ausblick auf EJB 3.0

Enterprise JavaBeans Komplett

- Hauptziele EJB 3.0
  - Vereinfachung des Entwicklungsprozesses
    - Z.B. Förderung der testgetriebenen Entwicklung
    - Verwendung von Meta-Annotationen… eine Codebasis
  - Vereinfachung der Nutzung von EJB-Komponenten
    - Z.B. Reduzierung der zu verwendenden APIs
    - Eliminieren der komplexen, aufwändigen und fehleranfälligen Deployment Deskriptoren

Ausblick auf EJB 3.0

Enterprise JavaBeans Komplett

- Hauptziele EJB 3.0
  - Vereinfachung/Reduzierung der Struktur der EJB-Komponenten (deren Fragmente)
    - Beispielsweise: Wegfall von `create()` und `remove()`-Methoden, wo sie nicht benötigt werden.
    - Am Beispiel Stateless Session Bean: `create()` erzeugt nichts und `remove()` löscht nichts…
    - Wegfall von Home Interface wo nicht benötigt
    - Wegfall von Component Interface wo nicht benötigt
    - Verwendung von POJOs/POJIs, statt komplexer Komponenten

Ausblick auf EJB 3.0

Enterprise JavaBeans™ Komplett

- „Zurück“ zu POJO/POJI (1/3)

  Beispiel: EJB 2.0/2.1

Component Interface

```
public interface Calculator extends EJBObject {
    int add (int a, int b) throws RemoteExcep
    int subtract (int a, int b) throws Remote      on;
}
```

Home Interface

```
ublic interface CalculatorHome extends EJBHome {
 alculator create() throws CreateException, RemoteException;
```

Bean-Klasse

```
public class CalculatorBean implements SessionBean {
    private SessionContext ctx;
    public void setSessionContext(SessionContext ctx) {
                this.ctx = ctx;
    }
    public void ejbCreate () {}
    public void ejbActivate () {}
    public void ejbPassivate () {}
    public void ejbRemove() {}
    public int add (int a, int b) {
                return a + b;
    }
    public int subtract ( int a, int b) {
                return a – b;
    }
}
```

oyment Deskriptor

```
<session>
  <ejb-name>CalculatorEJB</ejb-name>
  <home>com.example.CalculatorHome</home>
  <remote>com.example.Calculator</remote>
  <ejb-class>com.example.CalculatorBean</ejb-
        class>
  <session-type>Stateless</session-type>
  <transaction-type>Container</transaction-type>
</session>
...
```


Ausblick auf EJB 3.0

Enterprise JavaBeans™ Komplett

- „Zurück“ zu POJO/POJI (2/3)

  Beispiel: EJB 3.0 - Teil 1

Component Interface

```
public interface Calculator {
    int add (int a, int b);
    int subtract (int a, int b);
}
```

Home Interface

Bean-Klasse

```
@Stateless public class CalculatorBean implements Calculator {
    public int add (int a, int b) {
                return a + b;
    }
    public int subtract ( int a, int b) {
                return a – b;
    }
}
```

Deployment Deskriptor

Ausblick auf EJB 3.0

- „Zurück“ zu POJO/POJI (3/3)

  Beispiel: EJB 3.0 - Teil 2; noch einfacher

Component Interface …wird generiert

```
public interface Calculator {
    int add (int a, int b);
    int subtract (int a, int b);
}
```

Bean-Klasse

```
@Stateless public class CalculatorBean {
    public int add (int a, int b) {
            return a + b;
    }
    public int subtract ( int a, int b) {
            return a – b;
    }
}
```

Home Interface

Deployment Deskriptor

Ausblick auf EJB 3.0

- Verwendung von Meta-Annotationen im Java-Quellcode
  - Siehe JSR 175

    (A Metadata Facility for the JavaTM Programming Language)
  - XDoclet lässt grüßen

implements SessionBean ←hinweg

hinzu →

```
@Stateless public class ReserviererBean {
    void reserviereFlug(String flugNr, Date datum) {
        …
    }
}
```

Wird generiert →

```
public interface ReserviererBean {
    void reserviereFlug(…)
      throws RemoteExcpetion;
}
```

```
…
<session>
   <ejb-name>ReserviererEJB</ejb-name>
   <home>foo.ReserviererHome</home>
   <remote>foo.Reserviererer</remote>
   <ejb-class>foo.ReserviererBean</ejb-class>
   <session-type>Stateless</session-type>
   …
</session>
```

Ausblick auf EJB 3.0

Enterprise JavaBeans Komplett

- Kapselung von Umgebungszugriffen und JNDI (1/3)
  - …durch Nutzung von Meta-Annotationen
  - Vorteil: weniger Code für Zugriff auf Ressourcen benötigt
  - Vorteil: ermöglicht ENDLICH einfaches Testen von EJB-Komponenten
    - Diese sind nun einfache Java-Klassen, denen über set-Methoden die entsprechenden Informationen zur Verfügung gestellt werden können
    - …dadurch muss die Testumgebung nicht mehr ein J2EE-Applikationsserver sein
    - Bekannt als „Inversion of Control (IOC)“
      - Don‘t call us – we call you (Hollywood-Prinzip)

Ausblick auf EJB 3.0

Enterprise JavaBeans Komplett

- Kapselung von Umgebungszugriffen und JNDI (2/3)

Beispiele: Varianten des Zugriffs

```
@Stateless public class MySessionBean {
   Calculator myCalc;

   @EJB(name="calculator", jndiName=„com.foo.bean.Calculator“)
       public void setCalculator(Calculator c) {
         myCalc = c;
    }
}
```

@EJB

Referenzen auf EJB-Komponenten

```
@Stateless public class MySessionBean {
   @Resource(name="myDB") private DataSource customerDB;
   public void myMethod (String myString) {
   try {
                  Connection conn = customerDB.getConnection();
                  ...
   }
   catch (Exception ex) { ...}
   }
}
```

@Resource

Referenzen auf Ressourcen

Ausblick auf EJB 3.0

Enterprise JavaBeans™ Komplett

- Kapselung von Umgebungszugriffen und JNDI (3/3)

  Beispiele: Varianten des Zugriffs

```
@Stateless public class MySessionBean {
    private DataSource customerDB;
    @Resource(name="customerDB") private void setDataSource(DataSource myDB) {
                customerDB = myDB;
    }
    public void myMethod (String myString) {
                ...
                Connection conn = customerDB.getConnection();
                ...
    }
}
```

@Resource

Referenzen auf Ressourcen. Nutzung von set-Methoden
→ Externe Testbarkeit!

```
@Stateless public class MySessionBean {
    private DataSource customerDB;

    @Inject private void setCustomerDB(DataSource customerDB) {
                this.customerDB = customerDB;
    }

    public void myMethod (String myString) {
                ...
                Connection conn = customerDB.getConnection();
                ...
    }
}
```

@Inject

Referenzen auf Ressourcen, ohne expliziter Angabe von Meta-Daten (wie z.B. JNDI-Namen der Res.)

Ausblick auf EJB 3.0

Enterprise JavaBeans™ Komplett

- Default-Verhalten von EJB-Komponenten
  - Configuration by Exception
- Toolklassen, Muster und Idiome
- Weiterentwicklung Entity Beans / Persistenz
  - CMP/BMP Nutzung vereinfachen
  - Leistungsumfang erhöhen
  - Schwerpunkt CMP

Ausblick auf EJB 3.0

Enterprise JavaBeans™ Komplett

- Weiterentwicklung Entity Beans / Persistenz
  - Ziel: Simplifizierung von Entity Beans → POJOs
  - Konkrete Klasse; nicht mehr abstrakt
  - @Entity annotiert Bean-Klasse als Entity Bean
  - Leichtgewichtiges Domänen-Objekt
  - **Entity Beans sind nicht mehr direkt entfernt zugreifbar!**
  - Zugriff erfolgt nur noch über neu eingeführten javax.ejb.EntityManager

Ausblick auf EJB 3.0

Enterprise JavaBeans™ Komplett

- Weiterentwicklung Entity Beans / Persistenz
  - EntityManager
    - Erzeugen und Löschen von persistenten Entity-Instanzen
    - Finden von Entitäten über Primärschlüssel
    - Abfragen über Entitäten

```
@Stateless public class OrderEntry {
    EntityManager em;

    @Inject public void setEntityManager(EntityManager em) {
        this.em = em;
    }

    public int createOrder(String articleNo, int quantity, ...) {
        Order order = new Order(articleNo, quantity, ...);
        em.create(order);
        return order.getId();
    }

    public Order find(int id) {
        return em.find(Order.class, id);
    }

    public void enterOrder(int custID, Order newOrder) {
        Customer cust = (Customer)em.find("Customer",
    custID);
        cust.getOrders().add(newOrder);
        newOrder.setCustomer(cust);
    }
}
```

Ausblick auf EJB 3.0

Enterprise JavaBeans Komplett

- Weiterentwicklung Entity Beans / Persistenz
  - Entity Beans beherrschen nun Vererbung und Polymorphie
  - Polymorphe EJB QL-Abfragen sind nun zulässig
  - Kein Home Interface mehr nötig
  - Kein Business Interface mehr nötig (technisch betrachtet ;-)
  - JavaBean-Stil
    - Attribute sind nur per getter/setter-Methoden zugreifbar
    - Eliminieren der komplexen, aufwändigen und fehleranfälligen Deployment Deskriptoren
      - Beziehungen etc. werden in den Klassen per Annotationen deklariert

Ausblick auf EJB 3.0

Enterprise JavaBeans Komplett

- Weiterentwicklung EJB QL
  - Sub Queries
  - GroupBy
  - Having
  - Explizite Inner-Joins und Outer-Joins
  - Projections
  - Dynamische Abfragen, Named Queries
  - Polymorphe Abfragen
- …und sonst noch…
  - Interzeptoren
  - Callbacks
  - …

Ausblick auf EJB 3.0

Enterprise JavaBeans Komplett

- Fazit EJB 3.0 aus heutiger Sicht
  - Mit EJB 3.0 wird die Komponententechnologie erheblich einfacher und gleichzeitig mächtiger
  - EJB 3.0 wird die Entwicklung von komponenten-basierten Systemen beschleunigen
    → Konkurrenz .NET
  - Das leidige Thema der konkurrierenden Persistenz-strategien innerhalb von J2EE wird mit EJB 3.0 endlich behoben
    → es gibt „nur“ noch eine Lösung

Ausblick auf EJB 3.0

Enterprise JavaBeans Komplett

Objekt-orientiertes Programmieren OOP 2005 The essence of modern software engineering

EJB 3.0-Vortrag auf der OOP-Konferenz 2005 in München (www.oopconference.de)

„Enterprise JavaBeans 3.0 – erheblich einfacher und mächtiger“
Track: MI 11 (Mittwoch 26.01.2005, 16:00-17:00)
Sprecher: Oliver Ihns

Javamagazin

EJB 3.0 - Kolumne

Ab Ausgabe 01.05 wird es in **jeder** Ausgabe des Javamagazins einen Artikel zu EJB 3.0 geben.
Dauer: voraussichtlich bis zum finalen Release von EJB 3.0!

Ende - Vielen Dank!

Enterprise JavaBeans™ Komplett

Fragen & Antworten
Diskussion

Oliver.Ihns@resco.de Stefan.Heldt@resco.de

Weitere Informationen zu EJB 2.1 und EJB 3.0 unter
www.ejb-komplett.de