


# Enterprise Java Beans

Fachhochschule Hagenberg
WS 2001 / 2002

Silbergrau Consulting & Software GmbH
Dr. Andreas Erlach

# Inhaltsübersicht

- Application Server
- J2EE Architektur bzw. EJB Spezifikation
- Session, Entity und Message-Driven Beans
- EJB Deployment
- EJB Clients (Thin / Rich)
- Best Practices und Probleme
- EJB Application Server Auswahl
- Technologie-Vergleich (CORBA, COM+)

© Silbergrau Consulting & Software GmbH, 2000/2001

# Entity Bean XXII

| | Container Managed Persistence (CMP) | Bean Managed Persistence (BMP) |
|---|---|---|
| Class Definition | abstrakt | Nicht abstrakt |
| Datenbank-Zugriff | Während des Deployments generiert | Durch den Entwickler implementiert |
| Persistenter Zustand | Durch „virtuelle“ persistente Fields (Deployment) | Instanzvariablen |
| Zugriffsmethoden auf den Zustand | Abstrakte Zugriffsmethoden sind erforderlich | beliebig |
| Finder Methoden | Mit EJB QL beschrieben und während des Deployments generiert | Durch den Entwickler implementiert |
| Select Methoden | Siehe Finder Methoden | Nicht vorhanden |
| Return Value von ejbCreate Methoden | null | Primary Key |

© Silbergrau Consulting & Software GmbH, 2000/2001

# EJB Query Language I

- EJB QL für Entity Beans mit Container-managed Persistence
- Portable auf beliebigem EJB Container
- Syntax basiert auf SQL92
  - Besteht aus Select, From und Where Clause
- EJB QL erlaubt Pfadausdrücke
  - In SQL werden Joins „über“ Tabellen erstellt
  - In EJB QL gibt es Abfragen über Relationships der Entity Beans

© Silbergrau Consulting & Software GmbH, 2000/2001

# EJB Query Language II

- finder bzw. select Methoden
  - werden nicht implementierten, sondern mit EJB QL beschrieben
  - Methoden-Parameter werden mittels ?<i> referenziert
- Unterschied von select zu finder Methoden
  - Wird dem Client nicht über Local oder Remote Interfaces zugänglich gemacht
  - Kann auch persistente Fields liefern

© Silbergrau Consulting & Software GmbH, 2000/2001

# EJB Query Language III

- EJB QL ::= select_clause from_clause [where_clause]
  - select_clause definiert die Typen der Objekte bzw. Werte die von der Query geliefert werden
    - Local interface, Remote interface oder persistentes Field
  - from_clause definiert den Scope der Query mit einer oder mehreren Identification variables
    - Werden in der select_clause und/oder where_clause referenziert
    - Stellt entweder Abstract Schema Name (~ logischer Tabellenname) oder ein Element aus einer *:N Beziehung dar
  - where_clause definiert den Ausdruck der die Ergebnismenge der Query beschränkt

© Silbergrau Consulting & Software GmbH, 2000/2001

# EJB Query Language IV

- Einschränkungen für EJB Query Language
  - Erlaubt keine Kommentare
  - Date und Timestamp müssen als long dargestellt werden
  - Time muss als int dargestellt werden
  - Vererbung wird nicht unterstützt
    - Gilt generell für CMP 2.0
  - Erlaubt keine dynamischen Queries

© Silbergrau Consulting & Software GmbH, 2000/2001

# EJB Query Language (Beispiele) V

- SELECT OBJECT(p) FROM Person AS p
  - Finder Methode findAll()
  - Person ist der Abstract Schema Name
  - Keyword AS bei der Definition der Identification Variable ist optional
  - Liefert alle Personen
- SELECT OBJECT(p) FROM Person p
  WHERE p.age > ?1
  - Finder Methode findOlderThan(int age)
  - ?1 entspricht dem Parameter age der Finder Methode
  - Liefert alle Personen deren Alter größer, als der bei der Finder Methode übergebene Wert ist

© Silbergrau Consulting & Software GmbH, 2000/2001

# EJB Query Language (Beispiele) VI

- SELECT OBJECT(p) FROM Person p
  WHERE p.age > ?1 AND
  p.address.city = ?2
  - Finder Methode findOlderThanAndResidentIn(int age, String city)
  - address ist eine Container Managed Relationship (CMR)
  - „.“ stellt zwischen p und age bzw. zwischen address und city einen Delimiter dar
  - „.“ stellt zwischen p und address einen Navigation Operator dar
  - Liefert alle Personen deren Alter größer und deren Wohnort gleich den bei der Finder Methode übergebenen Werten ist

© Silbergrau Consulting & Software GmbH, 2000/2001

# EJB Query Language (Beispiele) VII

- SELECT OBJECT(p)
  FROM Person p, IN (p.friends) f
  WHERE f.address.city = ?1
  - Finder Methode findWithFriendsResidentIn(String city)
  - friends ist eine „mehrwertige“ Container Managed Relationship (CMR)
  - Über mehrwertige CMRs (Collections) kann in Expressions nicht navigiert werden
  - Liefert alle Personen deren Freunde im Ort wohnen der gleich dem bei der Finder Methode übergebenen Wert ist

© Silbergrau Consulting & Software GmbH, 2000/2001

# EJB Query Language (Beispiele) VIII

- SELECT OBJECT(p) FROM Person p
  WHERE f.friends IS EMPTY
  - Finder Methode findWithNoFriends()
  - friends ist eine „mehrwertige“ Container Managed Relationship (CMR)
  - Liefert alle Personen die keine Freunde haben

© Silbergrau Consulting & Software GmbH, 2000/2001

# EJB Query Language (Beispiele) IX

- SELECT DISTINCT p.name
  FROM Person p
  WHERE p.age BETWEEN ?1 AND ?2
  - Select Methode ejbSelectNameForAgeBetween(int from, int to)
  - DISTINCT eliminiert doppelte Namen
  - BETWEEN-AND entspricht p.age >= ?1 AND p.age <= ?2
  - Liefert die Namen aller Personen die in den angegebenen Altersbereich fallen

© Silbergrau Consulting & Software GmbH, 2000/2001

# EJB Query Language X

- Wegen der EJB Sprachdefinition siehe
  - Enterprise JavaBeans™ Specification, Version 2.0. Sun Microsystems. 2001
  - http://java.sun.com/products/ejb/

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean I

- Soll die Verwaltung eines Kontos realisieren
  - Kontoinformationen auslesen
  - Einzahlung bzw. Abhebung
  - Buchungszeilen sollen gespeichert werden
- Erstellt werden müssen folgende Entity Beans
  - Account und LineItem – mittels Local Interface
  - Account soll auch Remote verfügbar sein

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean II

```
import java.rmi.*;
import javax.ejb.*;

public interface AccountLocalHome extends EJBLocalHome {
   Account create(String number,
       double balance, String currency)
       throws CreateException;
   Account findByPrimaryKey(String number)
       throws FinderException;
}
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean III

```
import java.rmi.*;
Import java.util.*;
import javax.ejb.*;

public interface AccountLocal extends EJBLocalObject {
   AccountInfo getAccountInfo();
   void deposit(double amount, String currency)
       throws UnknownCurrencyException;
   void withdraw(double amount, String currency)
       throws InsufficientBalanceException,
       IllegalCurrencyException;
   Collection getLineItemsAfter(Date start);
}
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean IV

```
import javax.naming.*;
import java.rmi.*;
import javax.ejb.*;
import javax.rmi.*;

public abstract class AccountBean implements EntityBean {
   public void setEntityContext(EntityContext ctx) {}
   public void unsetEntityContext() {}
   public String ejbCreate(String number, double balance,
                           String currency) throws CreateException {
       setNumber(number);
       setBalance(balance);
       setCurrency(currency);
       return null;
   }
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean V

```
public String ejbPostCreate(String number, double balance,
            String currency) throws CreateException {}
public void ejbActivate() {}
public void ejbPassivate() {}
public void ejbRemove() {}
public void ejbLoad() {}
public void ejbStore() {}
public abstract String getNumber();
public abstract double getBalance();
public abstract String getCurrency();
public abstract void setNumber(String newNumber);
public abstract void setBalance(double newBalance);
public abstract void setCurrency(String newCurrency);
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean VI

```
public abstract Collection getLineItems();
public abstract void setLineItems(Collection items);
public AccountInfo getAccountInfo() {
    return new AccountInfo(getNumber(), getBalance(),
                                    getCurrency());
}
public void deposit(double amount, String currency)
   throws IllegalCurrencyException {
    if (!currency.equals(getCurrency()) {
       throw new IllegalCurrencyException();
    }
    addLineItemFor(amount);
    setBalance(getBalance() + amount);
}
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean VII

```
  public void withdraw(double amount, String currency)
     throws InsufficientBalanceException,
       IllegalCurrencyException {
      if (!currency.equals(getCurrency()) {
         throw new IllegalCurrencyException();
      }
      if (getBalance() < amount) {
         throw new InsufficientBalanceException();
      }
      addLineItemFor(-amount);
      setBalance(getBalance() - amount);
  }
}
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean VIII

```
    protected void addLineItemFor(double amount) {
        try {
              InitialContext ctx = new InitialContext();
              LineItemLocalHome home = (LineItemLocalHome)
              ctx.lookup("java:comp/env/ejb/LIHome„);
              Collection items = getLineItems();
               items.add(home.create(
                    getNumber() + items.size(),
                    new Date(), amount));
        } catch (Exception ex) {
              throw new EJBException(ex.getMessage());
        }
    }
}
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean IX

```
import java.rmi.*;
import javax.ejb.*;

public interface LineItemLocalHome extends EJBLocalHome {
   LineItemLocal create(String id, Date date,
       double balance) throws CreateException;
   LineItemLocal findByPrimaryKey(String id)
       throws FinderException;
   Collection findLineItemsAfter(String accountId,
       long date) throws FinderException;
}
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean X

```
import java.rmi.*;
Import java.util.*;
import javax.ejb.*;

public interface LineItemLocal extends EJBLocalObject {
   String getId();
   long getDate();
   double getAmount();
   Account getAccount();
}
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean XI

```
import javax.naming.*;
import java.rmi.*;
import javax.ejb.*;
import javax.rmi.*;

public abstract class LineItemBean implements EntityBean {
   public void setEntityContext(EntityContext ctx) {}
   public void unsetEntityContext() {}
   public String ejbCreate(String id, Date date, double amount)
        throws CreateException {
        setId(id);
        setDate(date.getTime());
        setAmount(amount);
        return null;
   }
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean XII

```
public String ejbPostCreate(String id, Date date,
          double amount) throws CreateException {}
public void ejbActivate() {}
public void ejbPassivate() {}
public void ejbRemove() {}
public void ejbLoad() {}
public void ejbStore() {}
public abstract String getId();
public abstract long getDate();
public abstract double getAmount();
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean XIII

```
    public abstract void setId(String newId);
    public abstract void setDate(long newDate);
    public abstract void setAmount(double newAmount);
    public abstract LocalAccount getAccount();
    public abstract void setAccount(LocalAccount account);
}
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean XIV

- Warum wurde für das AccountBean Local Interface gewählt?
- Was ist erforderlich damit die AccountBean auch Remote verfügbar ist?
- Was muss beim Deployment festgelegt werden?
- Wie kann die finder Methode findLineItemsAfter() mit EJB QL beschrieben werden?
- Wie ist die „finder“ Methode getLineItemsAfter zu implementieren?
  - **`Collection getLineItemsAfter(Date start);`**

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean XV

- Warum wurde für das AccountBean Local Interface gewählt?
  - Wegen der bidirektionale Beziehung zwischen AccountBean und LineItemBean
- Was ist erforderlich damit die AccountBean auch Remote verfügbar ist?
  - Ein Remote Interface (Home und Remote) oder
  - zB. Session Bean mit einem Remote Interface

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean XVII

- Erforderliche Deployment-Schritte
  - Erstellen einer Enterprise Bean für Account und LineItem
  - Zuordnen der erforderlichen Klassen zu den entsprechenden EJB Jars
  - Bean Class und Local Interfaces für Account bzw. LineItem auswählen
  - Container-managed persistent Fields und Relationships festlegen

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean XVII

- Erforderliche Deployment-Schritte (Forts.)
  - DataSource festlegen
  - Transaction Attributes festlegen
  - Abstract Schema Name für Account und LineItem definieren
  - EJB QL Queries für finder bzw. select Methoden festlegen
  - SQL Statements für finder und Container Methoden (zB. Insert, Update bzw. Delete) erzeugen
  - ...

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean XVI

- Wie kann die finder Methode findLineItemsAfter() mit EJB QL beschrieben werden?

```
Collection findLineItemsAfter(String accountId,
      long date) throws FinderException;

SELECT OBJECT(l)
     FROM LineItem AS l
     WHERE id LIKE ?1 AND
          date > ?2
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 2.0 Entity Bean XVI

- Wie ist die „finder“ Methode getLineItemsAfter zu implementieren?

```
public Collection getLineItemsAfter(Date start) {
    try {
        InitialContext ctx = new InitialContext();
        LocalLineItemHome home = (LocalLineItemHome)
        ctx.lookup("java:comp/env/ejb/LIHome„);
        return home.findLineItemsAfter(
            getNumber() + "%", start.getTime());
    } catch (FinderException ex) {
        throw new EJBException(ex.getMessage());
    }
  }
}
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Entity Bean XXIII

| | CMP 2.0 | CMP 1.1 |
|---|---|---|
| Class Definition | abstrakt | Nicht abstrakt |
| Datenbank-Zugriff | Während des Deployments generiert | Während des Deployments generiert |
| Persistenter Zustand | Durch „virtuelle“ persistente Fields (Deployment) | Public persistent Fields |
| Zugriffsmethoden auf den Zustand | Abstrakte Zugriffsmethoden sind erforderlich | beliebig |
| Finder Methoden | Mit EJB QL beschrieben und während des Deployments generiert | EJB Container abhängig |
| Select Methoden | Siehe Finder Methoden | Nicht vorhanden |
| Return Value von ejbCreate Methoden | null | null |

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 1.1 Entity Bean I

- Soll die Verwaltung eines Kontos realisieren
  - Kontoinformationen auslesen
  - Einzahlung bzw. Abhebung
- Erstellt werden müssen folgende Interfaces bzw. Klassen
  - AccountHome – Home Interface
  - Account – Remote Interface
  - AccountBean – Entity Bean

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 1.1 Entity Bean II

```
import java.rmi.*;
import javax.ejb.*;

public interface AccountHome extends EJBHome {
   Account create(String number,
       double balance, String currency)
       throws CreateException, RemoteException;
   Account findByPrimaryKey(String number)
       throws FinderException, RemoteException;
}
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 1.1 Entity Bean III

```
import java.rmi.*;
import javax.ejb.*;

public interface Account extends EJBObject {
   void deposit(double amount, String currency)
        throws RemoteException, UnknownCurrencyException;
   AccountInfo getAccountInfo() throws RemoteException;
   String getNumber() throws RemoteException;
   double getBalance() throws RemoteException;
   String getCurrency() throws RemoteException;
   void withdraw(double amount, String currency)
        throws RemoteException,
        InsufficientBalanceException,
        IllegalCurrencyException;
}
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 1.1 Entity Bean IV

```
import javax.naming.*;
import java.rmi.*;
import javax.ejb.*;
import javax.rmi.*;

public class AccountBean implements EntityBean {
   public String currency;
   public double balance;
   public String number;
   public void setEntityContext(EntityContext ctx) {}
   public String ejbCreate(String newNumber, double newBalance,
   String newCurrency) throws CreateException {
        number = newNumber;
        balance = newBalance;
        currency = newCurrency;
        return null;
   }
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 1.1 Entity Bean V

```
public void ejbActivate() {}
public void ejbPassivate() {}
public void ejbRemove() {}
public void ejbLoad() {}
public void ejbStore() {}
public void unsetEntityContext() {}
public String getNumber() {
    return number;
}
public double getBalance() {
    return balance;
}
public String getCurrency() {
    return currency;
}
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 1.1 Entity Bean VI

```
public AccountInfo getAccountInfo() {
    return new AccountInfo(number, balance, currency);
}
public void deposit(double amount, String currency)
   throws IllegalCurrencyException {
    if (!currency.equals(getCurrency()) {
       throw new IllegalCurrencyException();
    }
    setBalance(getBalance() + amount);
}
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 1.1 Entity Bean VII

```
  public void withdraw(double amount, String currency)
     throws InsufficientBalanceException,
       IllegalCurrencyException {
     if (!currency.equals(getCurrency()) {
        throw new IllegalCurrencyException();
     }
     if (getBalance() < amount) {
        throw new InsufficientBalanceException();
     }
     setBalance(getBalance() - amount);
  }
}
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: CMP 1.1 Entity Bean VIII

- Erforderliche Deployment-Schritte
  - Erstellen einer Enterprise Bean für Account
  - Zuordnen der erforderlichen Klassen zu den entsprechenden EJB Jars
  - Bean Class und Interfaces für Account auswählen
  - Container-managed persistent Fields festlegen
  - DataSource festlegen
  - Transaction Attributes festlegen
  - ...

© Silbergrau Consulting & Software GmbH, 2000/2001

# Message-Driven Bean I

- Message-driven beans (MDBs) sind stateless, server-side, transaktionale Komponenten
- Für die asynchrone Verarbeitung von Messages
- Stellen einen Java Message Service (JMS) Message Listener dar
  - Ist vergleichbar einem Event Listener
- EJB Container managed das Environment der MDB
  - Transaktion, Security, Ressourcen, Concurrency und Message acknowledgment
  - Wesentlicher Vorteil gegenüber einem „herkömmlichen“ JMS Client

© Silbergrau Consulting & Software GmbH, 2000/2001

# Message-Driven Bean II

- Message-driven bean hält keinen Conversational State
  - Eine einzige MDB kann Messages von verschiedenen „Clients“ verarbeiten
- Message-driven bean besteht aus
  - Enterprise Bean implementiert die onMessage(Message) Methode
- Message-driven bean hat kein
  - Home Interface
  - Remote/Local Interface

© Silbergrau Consulting & Software GmbH, 2000/2001

# Message-Driven Bean III

- Client-Zugriff auf eine MDB erfolgt über eine JMS Message Queue – nicht über RMI

© Silbergrau Consulting & Software GmbH, 2000/2001

# Message-Driven Bean IV

- Anforderungen an eine Message-Driven Bean Klasse
  - Muss javax.ejb.MessageDrivenBean und javax.jms.MessageListener implementieren
  - Muss public und darf weder final noch abstract sein
  - Muss einen public Default-Konstruktor implementieren
  - Darf die finalize Methode nicht implementieren
  - Muss die ejbCreate() und ejbRemove() Methode implementieren
  - Muss die onMessage(Message) Methode implementieren

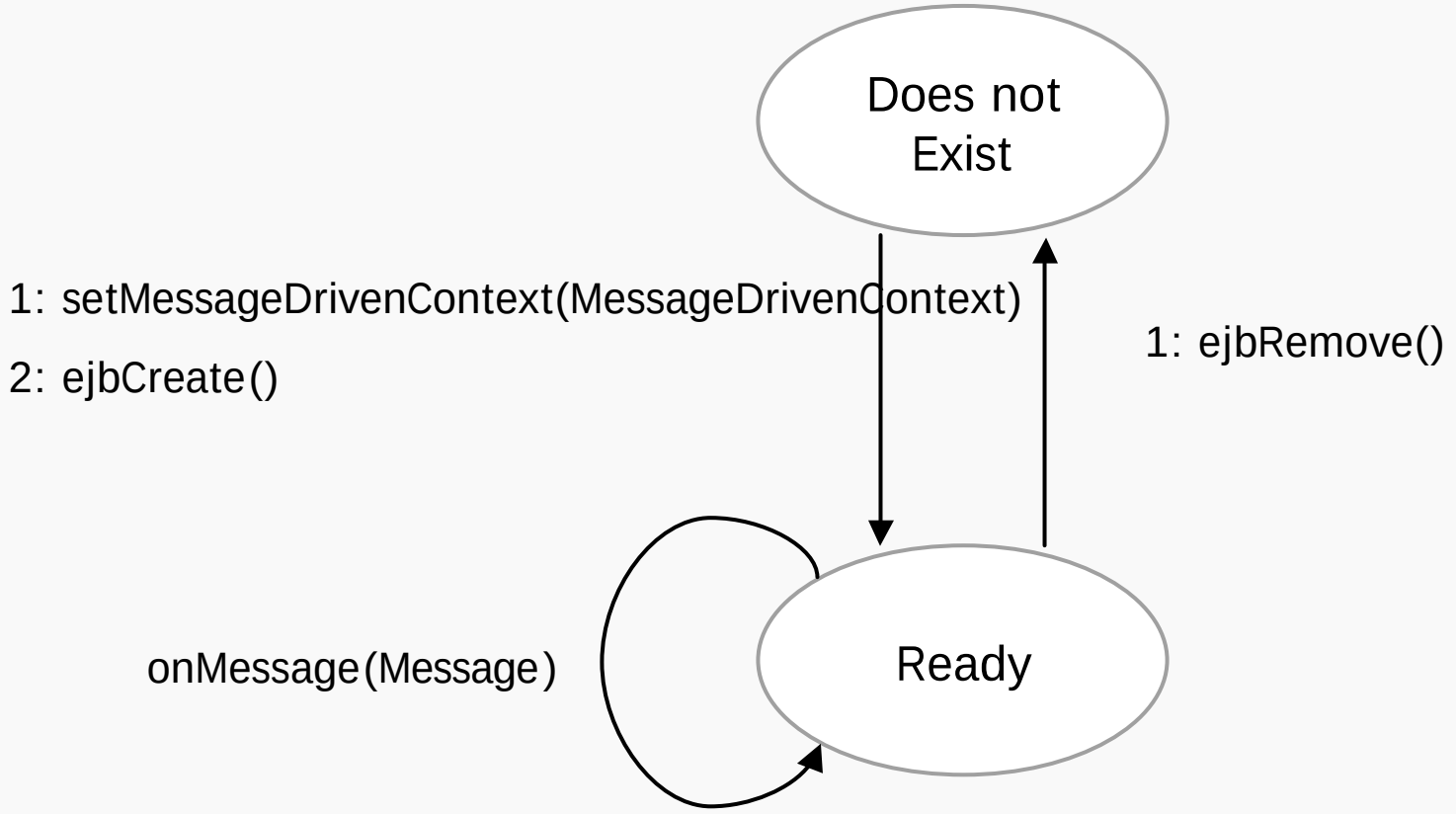
© Silbergrau Consulting & Software GmbH, 2000/2001

# Message-Driven Bean V

- public interface MessageDrivenBean extends EnterpriseBean
  - The container uses the MessageDrivenBean methods to notify the enterprise bean instances of the instance's life cycle events
- public void ejbRemove() throws EJBException
  - A container invokes this method before it ends the life of the message-driven object
- public void setMessageDrivenContext(MessageDrivenContext ctx)
  throws EJBException
  - Set the associated message-driven context. The container calls this method after the instance creation
  - MessageDrivenContext extends EJBContext; but adds no additional methods

© Silbergrau Consulting & Software GmbH, 2000/2001

# Message-Driven Bean VI




Life Cycle einer Message-Driven Bean

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: Message-Driven Bean I

- Soll eine Message per Mail versenden
- Ein „einfache“ Client zum erzeugen/senden der Message.

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: Message-Driven Bean II

```
import javax.ejb.*;
import javax.jms.*;

public class MailerBean
   implements MessageDrivenBean, MessageListener {
   public void ejbCreate() {
      trace("MailerBean.ejbCreate()");
   }
   public void setMessageDrivenContext(MessageDrivenContext ctx) {
      trace("MailerBean.setMessageDrivenContext(MessageDrivenContext)");
   }
   public void ejbRemove() {
      trace(" MailerBean.ejbRemove()");
   }
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: Message-Driven Bean III

```
    /**
     * When a message arrives in the mailer queue, onMeesage is notified
     * with the message, which contains the emailadress, the subject,
     * and the contents of the message.
     */
    public void onMessage(Message message) {
       trace("MailerBean.onMessage(Message)");
       try {
          String messageText = ((TextMessage) message).getText();
          // send mail via Java Mail
       } catch (JMSException ex) {
          throw new EJBException("MailerBean.onMessage()" + ex);
       }
    }
}
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: Message-Driven Bean IV

- Erforderliche Deployment-Schritte
  - Erstellen einer Enterprise Bean für MailerBean
  - Zuordnen der erforderlichen Klassen zum EJB Jar
  - Bean Class für MailerBean auswählen
  - Destination Type (Queue oder Topic) festlegen
    - Queue ist ein Point-to-Point Messaging Domain
    - Topic ist ein Publish-Subscribe Messaging Domain
  - JNDI Name der Destination festlegen
  - Connection Factory (QueueConnectionFactory oder TopicConnectionFactory) festlegen

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: Message-Driven Bean IV

- Erforderliche Deployment-Schritte (Forts.)
  - Acknowledgement type bei Bean-Managed Transactions festlegen
    - Entweder Auto-Acknowledge oder Duplicates-OK
  - JMS Message Selector festlegen
    - Statement zum filtern der Messages
  - ...

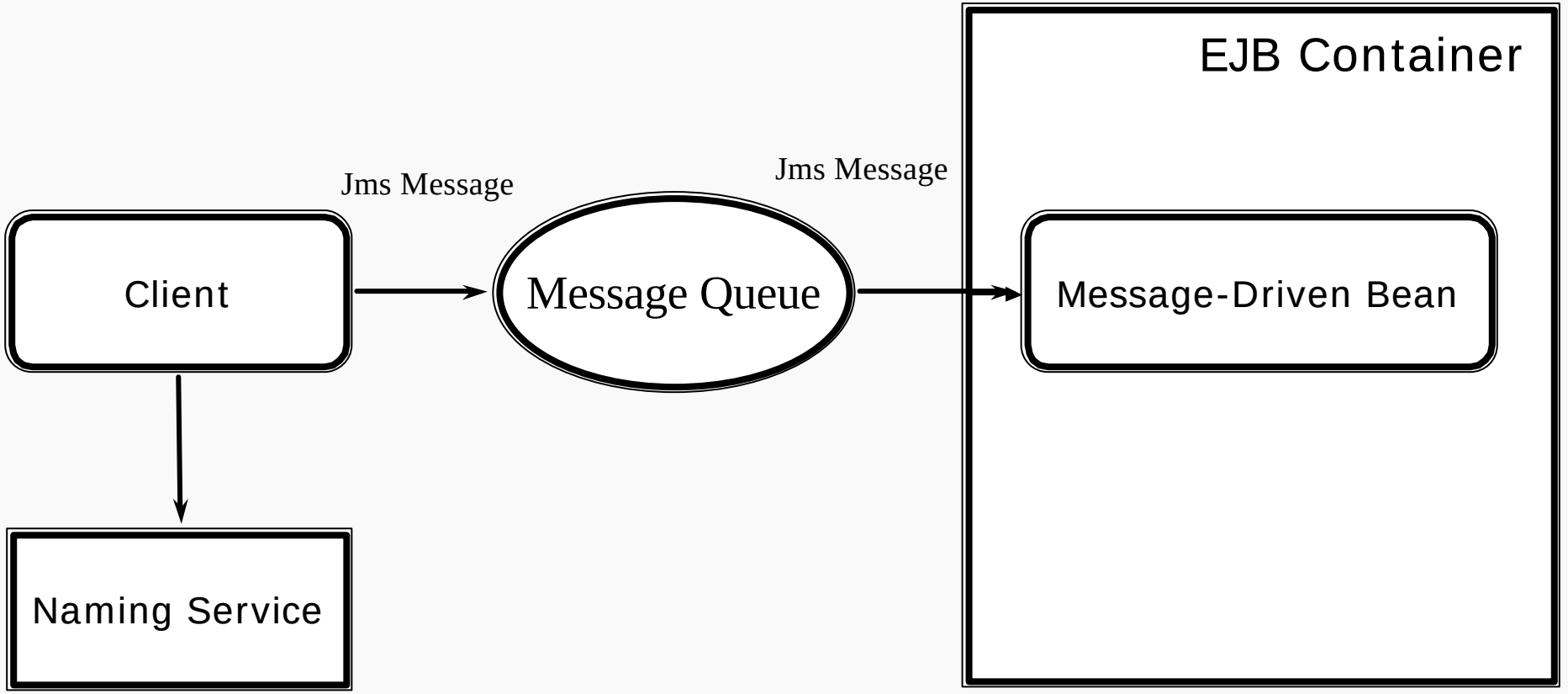
© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: Message-Driven Bean V

```
import javax.jms.*;
import javax.naming.*;
public class JmsClient {
    public static void main(String[] arguments) {
       try {
          InitialContext ctx = new InitialContext();
          QueueConnectionFactory qcf = (QueueConnectionFactory)
             ctx.lookup("java:comp/env/jms/MailerQueueConnectionFactory");
          Queue q = (Queue) ctx.lookup(java:comp/env/jms/MailerQueue);
          QueueConnection qc = qcf.createQueueConnection();
          QueueSession qs = qc.createQueueSession(true,0);
          qc.start();
          QueueSender s = qs.createSender(q);
          TextMessage msg = qs.createTextMessage();
          msg.setText(arguments[0]);
          s.send(msg);
          qc.close();
       } catch (Exception ex) {
          ex.printStackTrace();
       }
    }
}
```

© Silbergrau Consulting & Software GmbH, 2000/2001

# Beispiel: Message-Driven Bean VI




© Silbergrau Consulting & Software GmbH, 2000/2001

silbergrau
Consulting & Software GmbH

# Danke für die Aufmerksamkeit!

Silbergrau Consulting & Software GmbH
Dr. Andreas Erlach