Software Reuse

6. Java

Java Beans und Enterprise Java Beans

Peter Sturm
Universität Trier

Java – Einführung

- Erfolgreicher virtueller Maschinenansatz der Gegenwart
  - Vorbilder
    - IBM: Virtualisierung der gesamten Rechnerhardware
    - UCSD Pascal (Apple][): Virtuelle stack-orientierte Maschine (P-Code)
- Vorteile
  - „Develop Once – Run Everywhere“
  - Erstmals sehr hoher Abstraktionsgrad in den Bibliotheken
  - Erleicherung des Entwicklers
- Nachteile
  - Performanz

Develop Once
Debug Everywhere ☺

Software Reuse

6. Java

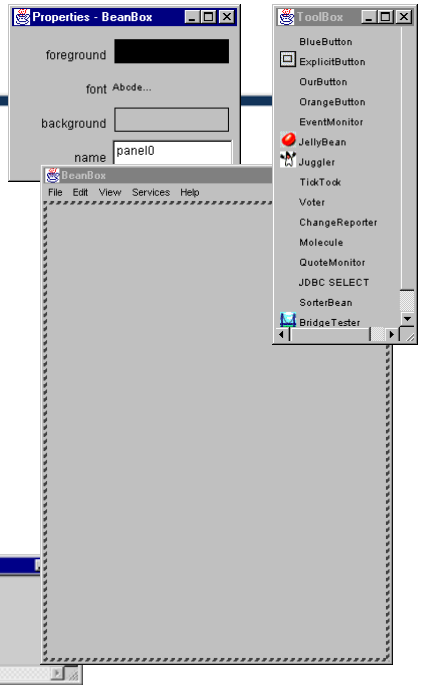
6.1 JavaBeans

JavaBeans

- Komponentenbasierte Programmierung in Java
- JavaBeans werden in der JVM der Anwendung ausgeführt
- JavaBeans sind Komponenten, die
  - über Zugriffsklassen verfügen
  - bestimmte Namenskonventionen bei den Methoden einhalten
  - über Dialogboxen konfigurierbar sind
  - neuen Programmiermethoden (Visual Programming) zugänglich sind
- Vergleichbar mit
  - Controls in OLE, COM, ActiveX (inproc)
  - Konfiguiermöglichkeiten in Visual Studio

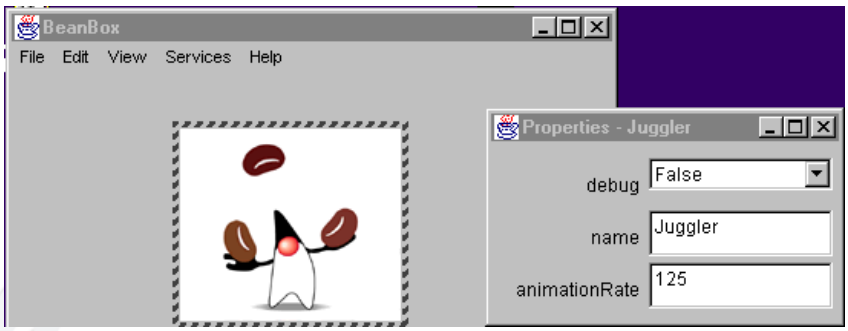
Die BeanBox

- Prototyp einer IDE mit visuellen Programmiereigenschaften
- 4 Fenster
  - BeanBox: Aktuelle Anwendung
  - ToolBox: Auflistung aller verfügbaren Beans
  - PropertiesBox: Eigenschaften der aktuellen Bean lesen und konfigurieren
  - MessageTracer



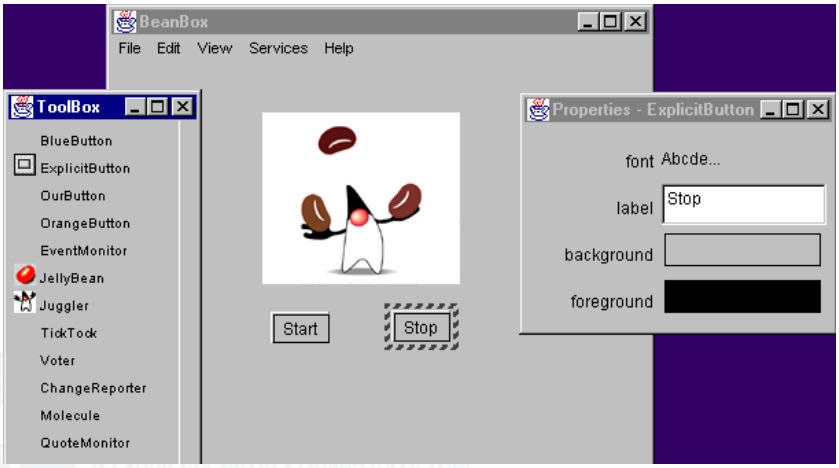
Beispiel: JugglerBean (1)

- JugglerBean aus Toolbox auswählen und plazieren



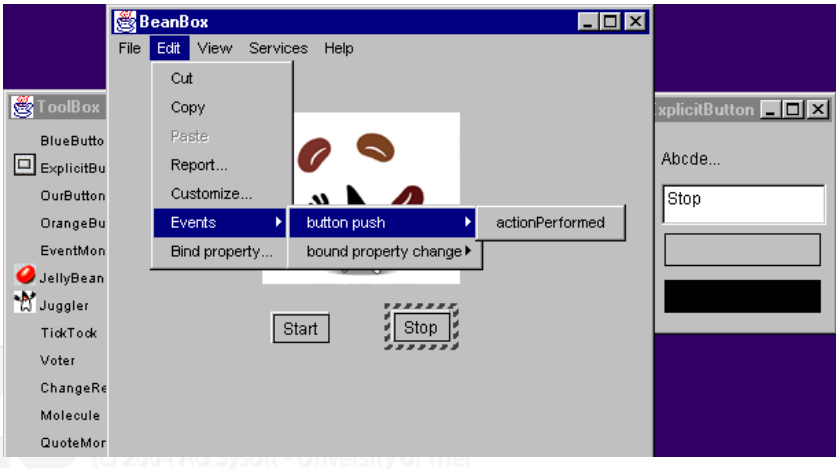
JugglerBean (2)

- 2 ExplicitButton mit „Start“ und „Stop“ hinzufügen



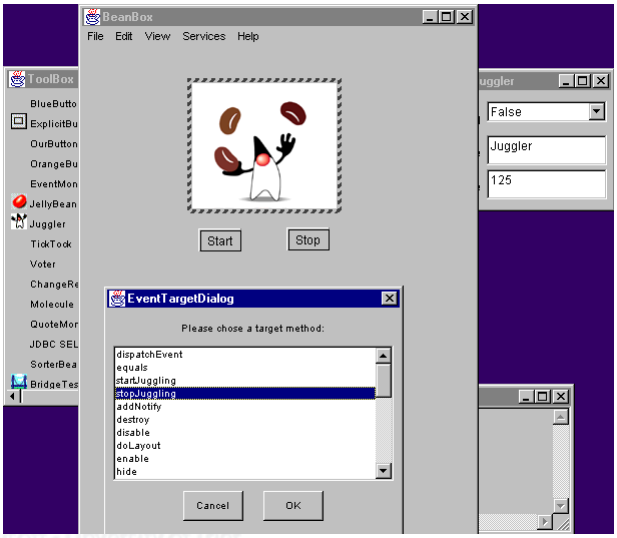
JugglerBean (3)

- „Stop“: Edit->Events->button push->actionPerformed



JugglerBean (4)

- ... mit JugglerBean verknüpfen



JugglerBean (5)

- Automatisch generierter Code

```
package tmp.sunw.beanbox;
import sunw.demo.juggler.Juggler;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class ___Hookup_1653fe1423 implements
java.awt.event.ActionListener, java.io.Serializable {
    public void setTarget(sunw.demo.juggler.Juggler t) {
        target = t;
    }

    public void actionPerformed(java.awt.event.ActionEvent
arg0) {
        target.stopJuggling(arg0);
    }

    private sunw.demo.juggler.Juggler target;
}
```

BeanProperties

- BeanBox erkennt Properties über Namen der Zugriffsfunktionen
- BeanProperty X
  - Verwendung sogenannter Decapitalization:
    - aus **`setMeineProp`** bzw. **`getMeineProp`** wird **`meineProp`**
- Angabe der Zugriffsfunktionen:
  - **`public X getX ()`**
    - Ausnahme: **`public boolean isX ()`**
  - **`public void setX ( X x )`**
- Nurleseeigenschaften: Keine **`setX()`**-Methode

Property-Arten

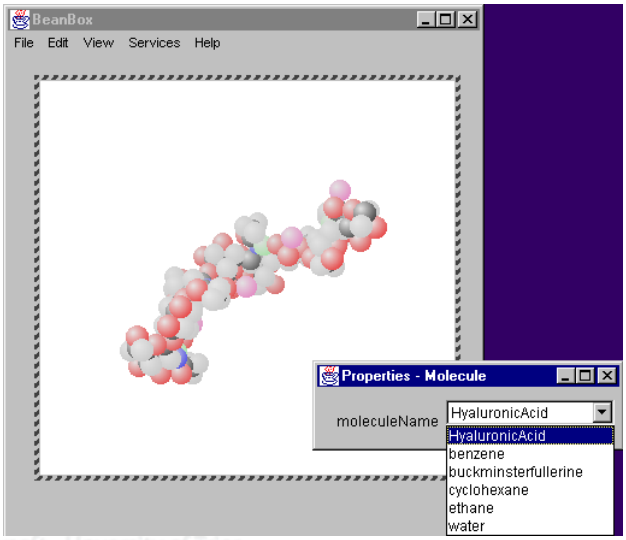
- Simple Property
  - Variable speichert einen einfachen Wert
- Indexed Property
  - Speicherung eines Feldes
  - Zugriffsfunktionen
    - **`X[] getX ()`**
    - **`void setX ( X[] x)`**
    - **`X getX ( int i )`**
      `void setX ( int i, X x )`

Property-Arten (contd.)

- Bound Property
  - Listener werden über Änderungen informiert
  - Zusätzlicher Implementierungsaufwand
    - Bean muß bei Änderung PropertyChange-Event senden
    - Verwaltung aller Listener:
      - void addPropertyChangeListener (...)
      - void removePropertyChangeListener (...)
    - Convenience-Klasse PropertyChangeSupport vorhanden
- Constraint Property
  - Bound Property mit Vetorecht der Listener

Property-Editoren

- BeanBox stellt für Grundtypen Editoren zur Verfügung
- Editoren für anwendungs-spezifische Methoden integrierbar



Alternativen

- BeanBox ist veraltete aber schöne Demonstration der wesentlichen Bean-Eigenschaften
  - Kleinere Inkompatibilitäten mit Java 1.4
- Neuere Fassung: Bean Builder
- Primärer Einsatz: Integrierte Entwicklungsumgebungen
  - JBuilder
  - NetBeans
  - …

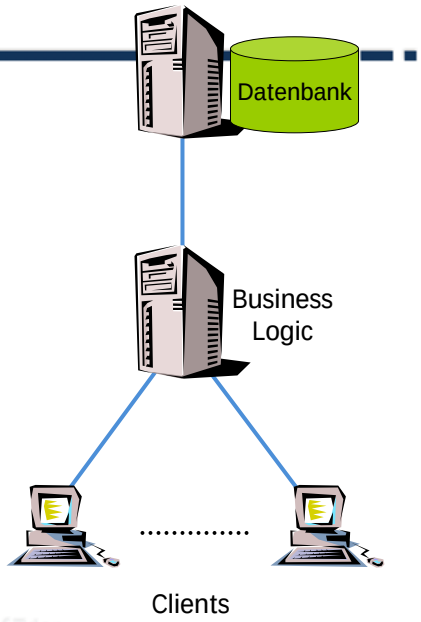
Software Reuse

6. Java

6.2 Enterprise Java Beans (EJB)

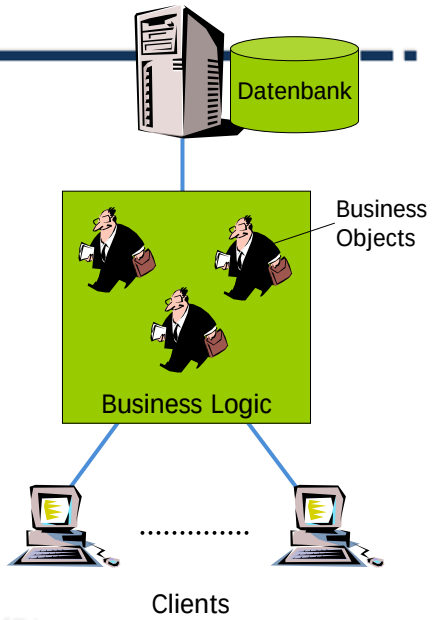
Motivation

- Bean = Komponente
- Zielgruppe
  - Kommerzielle Anwendungen
  - E-Commerce
- Evolution früherer
  - Client/Server-Systeme
  - Transaktionsmonitore
- 3-Tier-Applikationen
  - Client
  - Bussiness Application
  - Database



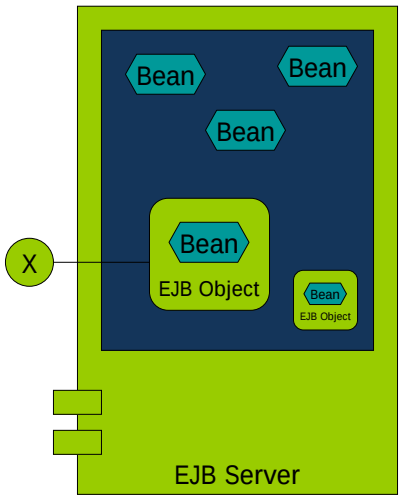
Business Objects

- Komponenten-basierte Anwendung
- Varianten
  - Eigenständige Komponenten
  - Wrapper für Legacy Software
  - Client-Transaktionen
- Laufzeitumgebung
  - EJB Server
  - vergleichbar MTS bei COM+



Aufbau des EJB-Server

- Laufzeitumgebung
  - Namensverwaltung
  - Anbindung an DB
  - Nachrichtenkommunikation
- EJB Object = Bean Wrapper
  - Indirektionsstufe
  - Zugriff über Reflection
- Beantypen
  - Entity Bean
    - Zustand, Persistent
  - Session Bean
    - Transient
    - Stateless, Stateful



Aufgaben des Servers

- Resource Management
  - Instance Pooling: Beans vorinstanziiert
  - Instance Swapping: Wechsel der EJB-Object-Bindung
  - Activation: Stilllegen und Reaktivieren von Beans
- Primary Services
  - Concurrency: Single-Threaded Bean-Implementierung
  - Transactions: ACID-Prinzip
  - Persistence: Container-Managed, Bean-Managed
  - Distribution: RMI over JRMP oder RMI-IIOP
  - Naming: JNDI unterstützt LDAP, NIS+, DNS, ...
  - Security: Methodenbasierte Zugriffskontrolle

Entity Bean: Struktur

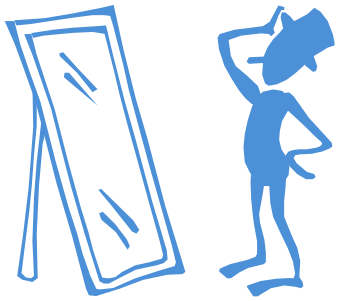
- Remote Interface
  - Die eigentlichen Zugriffsfunktionen der Bean
    - Setters und Getters :-)
  - `public interface X extends javax.ejb.EJBObject { ... };`
- Home Interface
  - Bean-Erzeugung: `public X create ( ... )`
  - Beans wiederfinden: `public X findByPrimaryKey ( key )`
  - `public interface XHome extends java.ejb.EJBHome { ... };`
- Primary Key
  - Speichern und Auffinden der Bean in Datenbank
  - `public class XPK implements java.io.Serializable { ... };`
- Bean Class
  - `public class XBean implements javax.ejb.EntityBean { ... };`

EntityBean: Funktionen

- ejbActivate
  - EJBObject-Bean-Bindung wiederhergestellt
- ejbPassivate
  - EJBObject-Bean-Bindung wird abgebaut
- ejbLoad
  - Bean wurde geladen
- ejbStore
  - Bean wird gespeichert
- ejbRemove
- setEntityContext
  unsetEntityContext
  - Informationen über EJBObject, Client, PK

Exkurs: Reflection

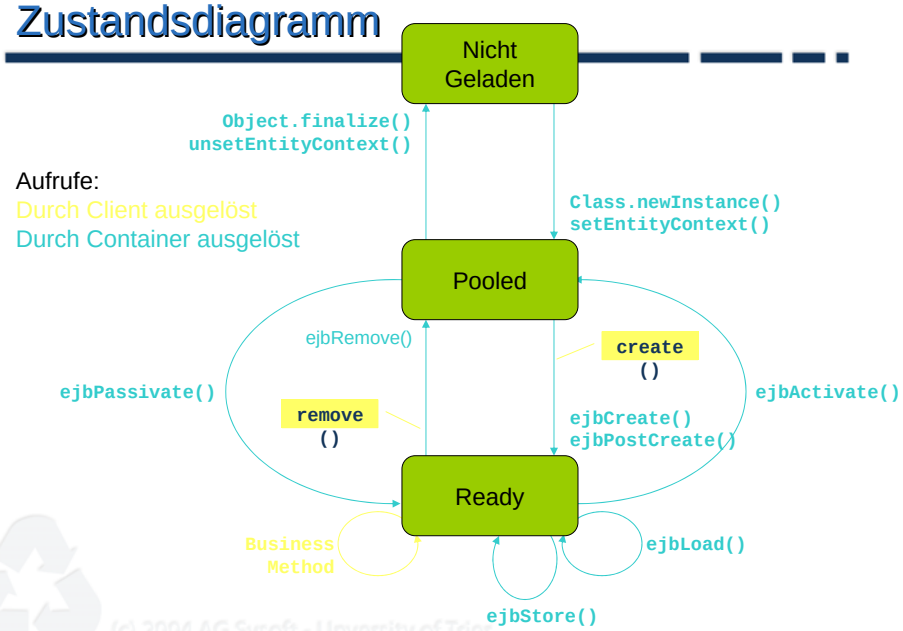
- Bean-Klasse implementiert weder Remote- noch Home-Interface
  - keine direkte Bindung zum Client vorhanden
- Bean-Klasse muß aber Methoden mit identischer Signatur enthalten
- EJB-Object greift über Reflection-Mechanismus darauf zu
  - vgl. JavaBeans



Deployment Descriptor

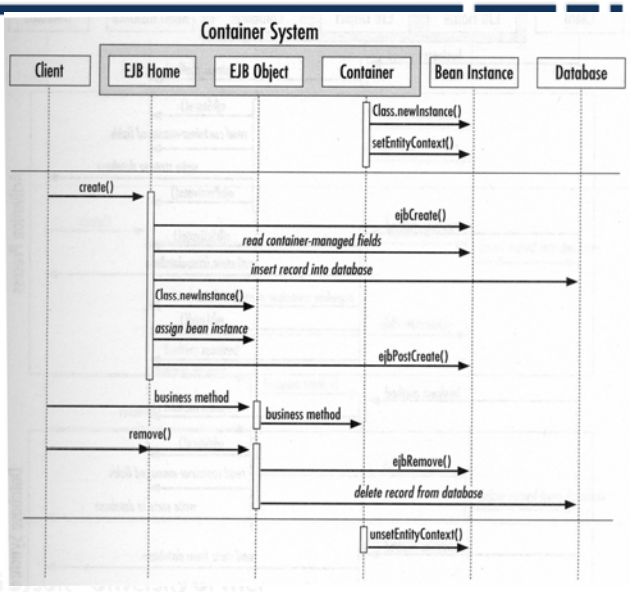
- XML-Dokument (EJB Version 1.1)
  - Bean-Beschreibung
    - Namen der Interfaces
    - Klassenname
    - Persistenztyp (Container oder Bean)
    - Welche Bean-Attribute werden persistent gespeichert
  - Zugriffsrollen von Clients beim Bean-Zugriff
  - Zugriffskontrolle auf Methoden
    - Welche Rollen dürfen zugreifen
  - Transaktionen
- Bean = jar-File bestehend aus XML-DD und *.class Dateien

Entity Beans: Zustandsdiagramm

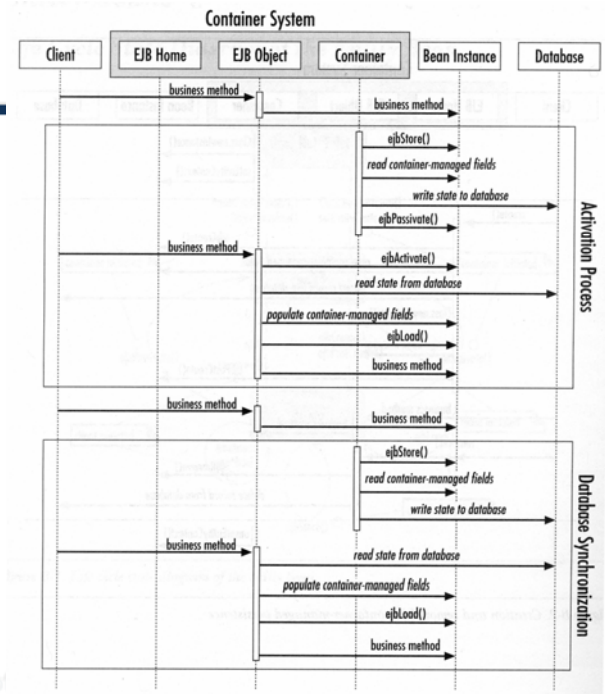


Container-Managed Persistence

- Erzeugen
- Löschen

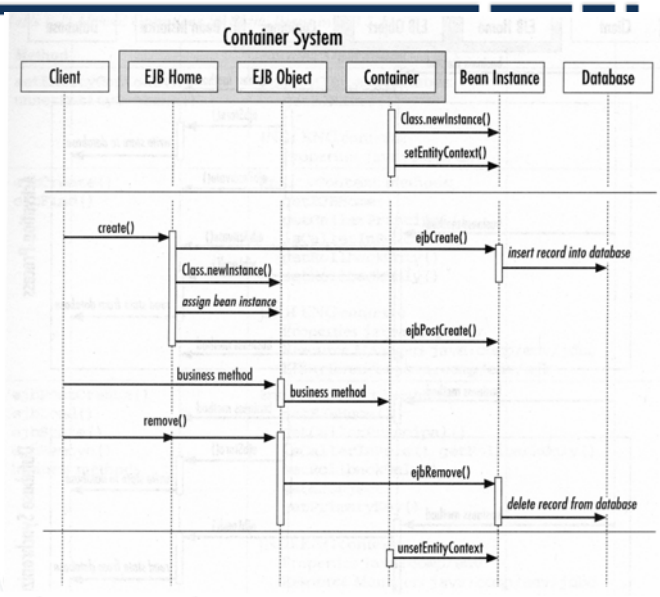


- Activation
- Synchronization

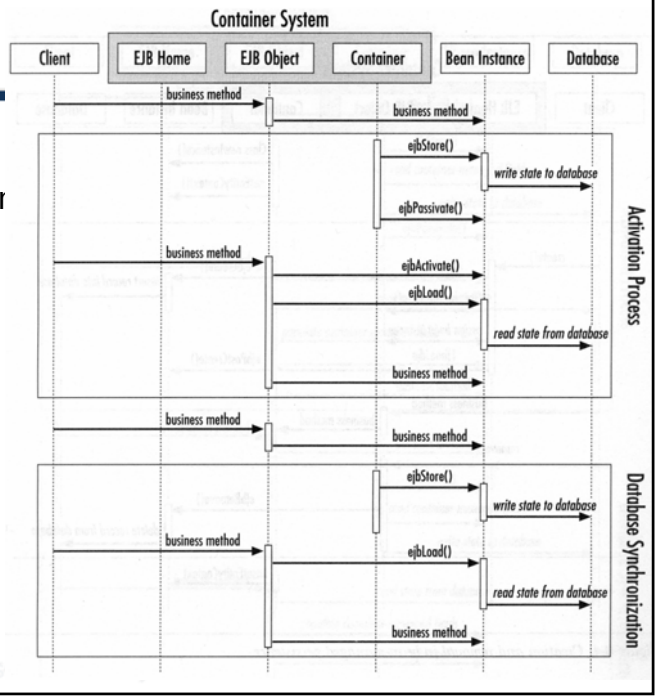


Bean-Managed Persistence

- Erzeugen
- Löschen



- Activation
- Synchronizatio



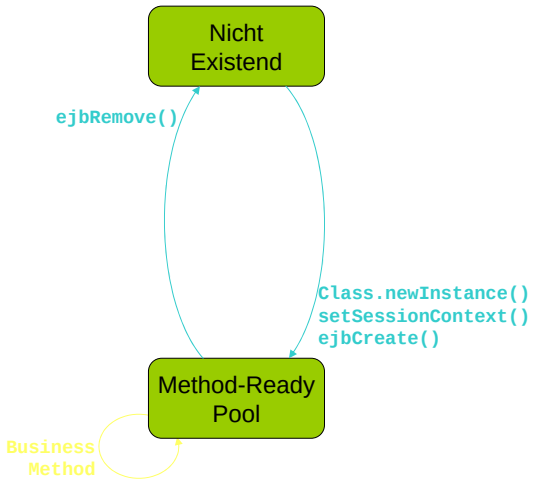
Session Beans

- Session Beans besitzen keinen persistent gespeicherten Zustand
- Zwei Varianten
- Stateless Session Bean
  - Kein sichtbarer Zustand zwischen Methodenaufrufen
  - Pooling und Swapping möglich
- Stateful Session Bean
  - Feste Bindung zu einem Client
  - Conversational State zwischen Methodenaufrufen
  - Swapping nicht möglich

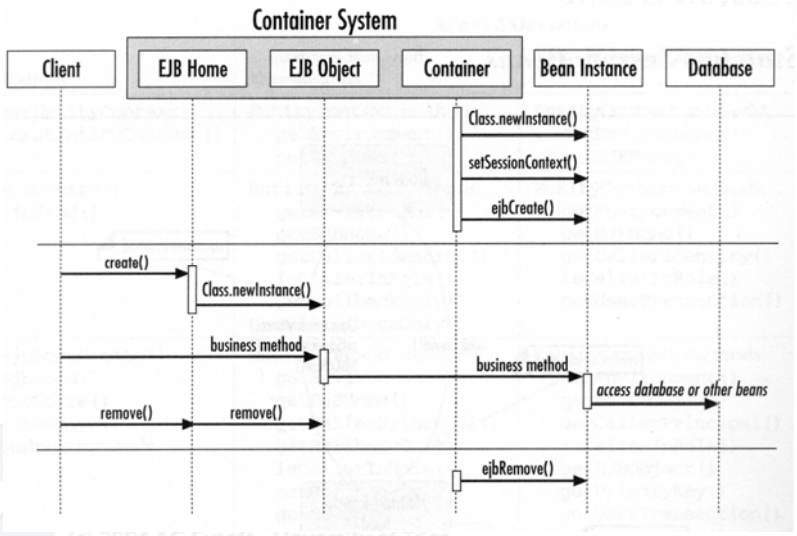
Bestandteile

- Remote Interface
  - Eigentlichen Interaktionsfunktionen
- Home Interface
  - Nur Erzeugung
  - Kein Auffinden über Primary Keys
- Bean-Klasse
  - Erweitert SessionBean
- Deployment Descriptor
  - XML-Dokument
    - u.a. Lokalisierungsinformation benötigter Beans
    - Zugriffskontrolle

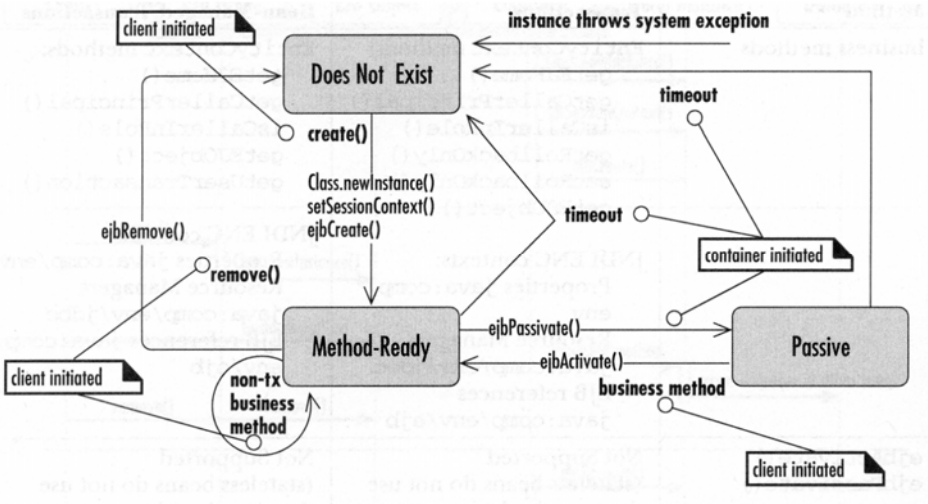
Session Bean: Stateless



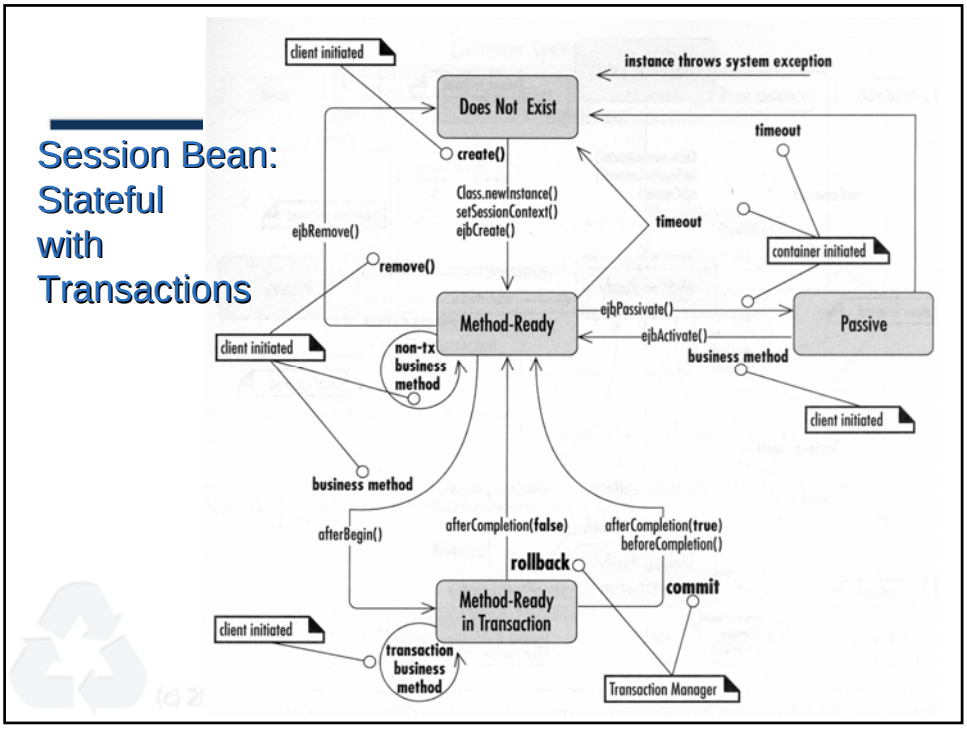
Stateless: `create()` und `remove()`



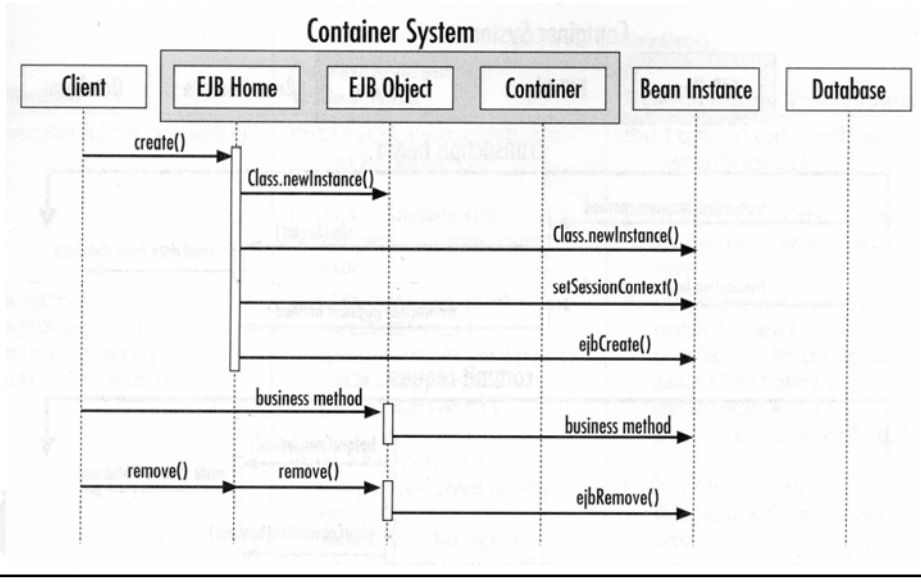
Session Bean: Stateful



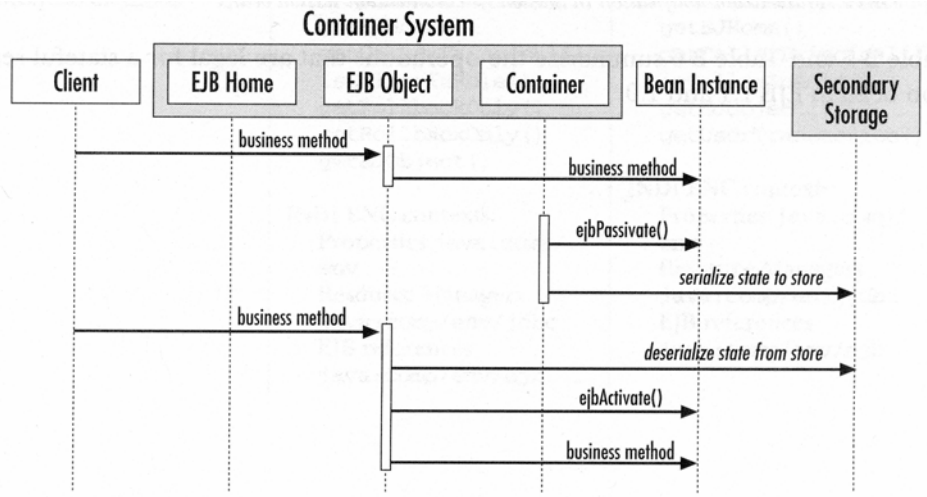
Session Bean: Stateful with Transactions



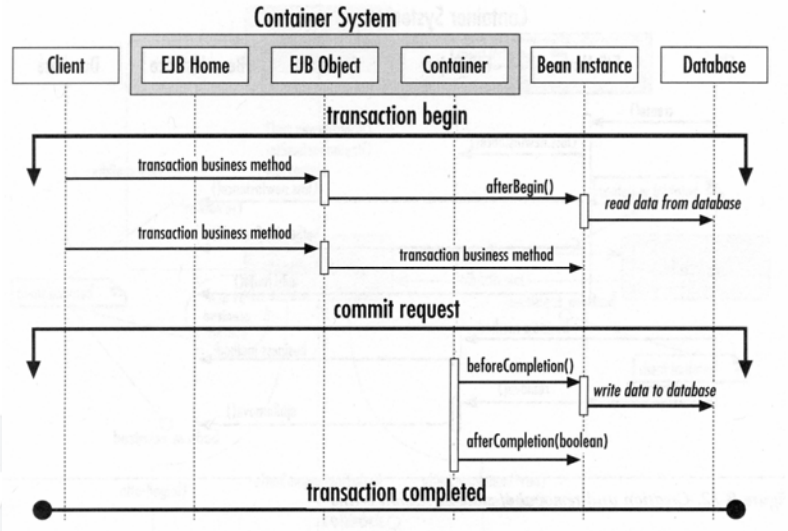
Stateful: `create()` und `remove()`



Stateful: Activation



Stateful: Transaction Notification



Literatur

- Richard Monson-Haefel
  Enterprise Java Beans
  O'Reilly, 2. Auflage, 2000
  - Quelle der gezeigten Zustands- und Interaktionsdiagramme