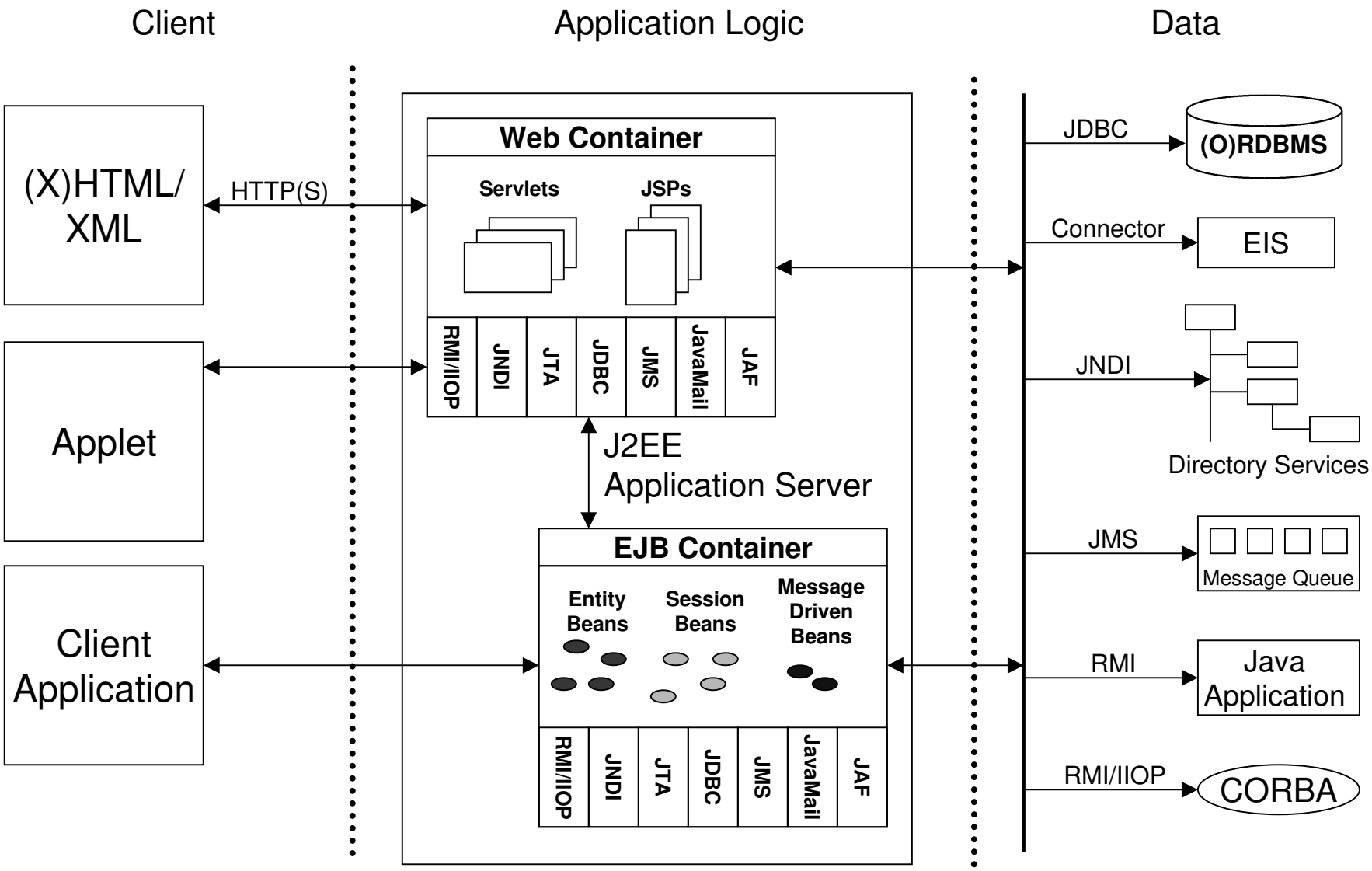
J2EE
Enterprise Information Systems & Enterprise Java Beans

Jürgen Göres

Überblick


Client
Application Logic
Data
(X)HTML/
XML
HTTP(S)
Applet
Client
Application
Web Container
Servlets
JSPs
RMI/IIOP
JNDI
JTA
JDBC
JMS
JavaMail
JAF
J2EE
Application Server
EJB Container
Entity
Beans
Session
Beans
Message
Driven
Beans
RMI/IIOP
JNDI
JTA
JDBC
JMS
JavaMail
JAF
JDBC
(O)RDBMS
Connector
EIS
JNDI
Directory Services
JMS
Message Queue
RMI
Java
Application
RMI/IIOP
CORBA

Enterprise Information Systems

- Oberbegriff für verschiedenartige Systeme:
  - Datenbanken
  - *Enterprise Resource Planing* -Systeme (ERP)
  - TP-Monitore
  - „Altlasten“ (*Legacy*-Systeme)
- individuelle Einschränkungen der Systeme
- Entwicklung neuer Anwendungen (*E-Commerce*) auf Daten dieser Systeme

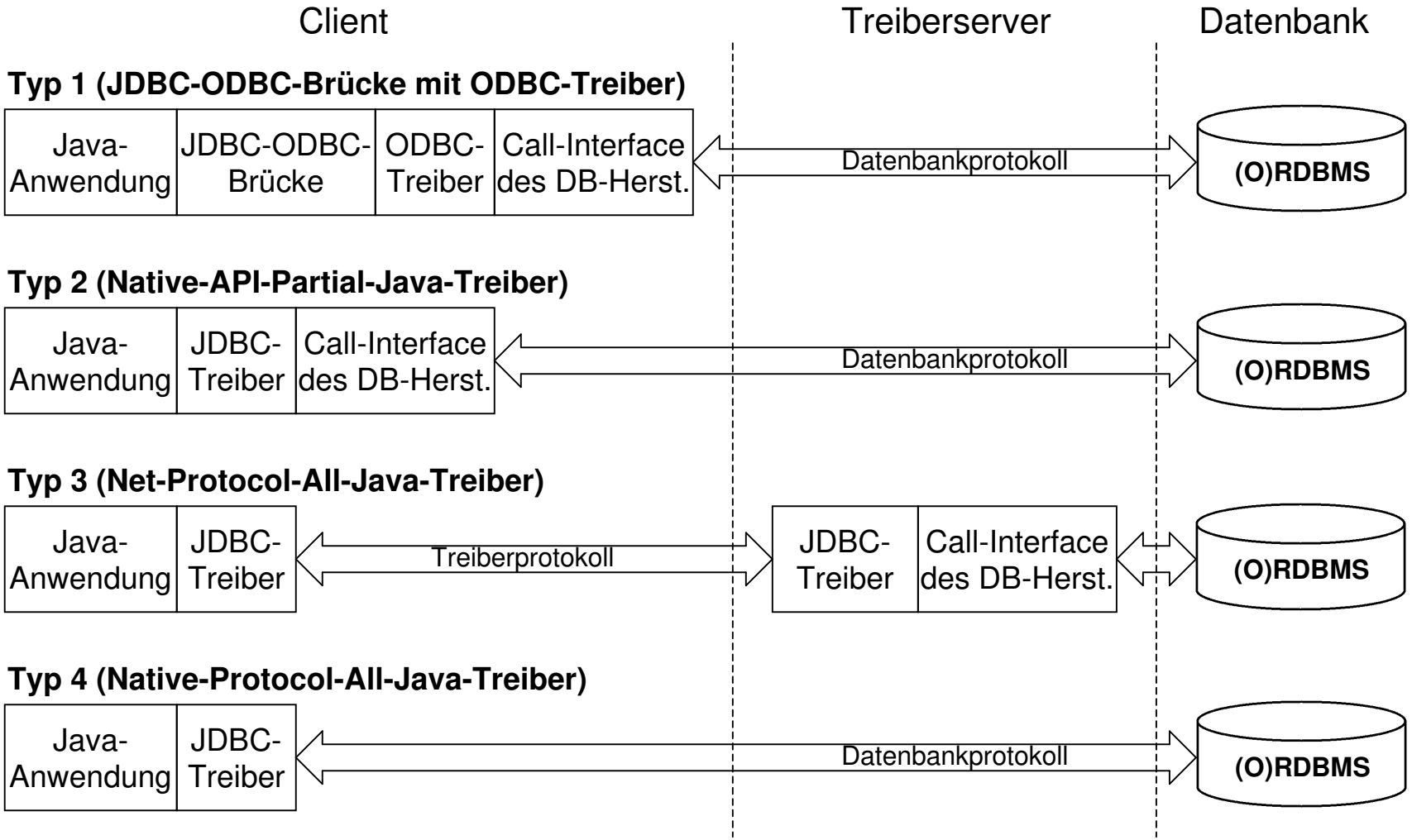
  ➔ Bedarf nach standardisiertem Zugriff

Java Database Connectivity (JDBC) (1)

- Standardschnittstelle für den Zugriff auf (objekt-) relationale Datenbanksysteme aus Java-Anwendungen
- *Call Level Interface*
- basierend auf SQL:1999
- aktuelle Version: JDBC 2.1, im Beta-Stadium: JDBC 3.0
- zwei Teile:
  - *Core-Package*: Standardfunktionalität
  - *Optional-Package*: Erweiterungen für Business-Anwendungen
- Portabilität durch Treiber-Konzept:
  - werden z.B. vom Hersteller des DBMS zur Verfügung gestellt
  - bilden JDBC-Aufrufe auf die Schnittstelle des DBMS ab
  - verschiedene Formen der Realisierung

Java Database Connectivity (JDBC) (2)

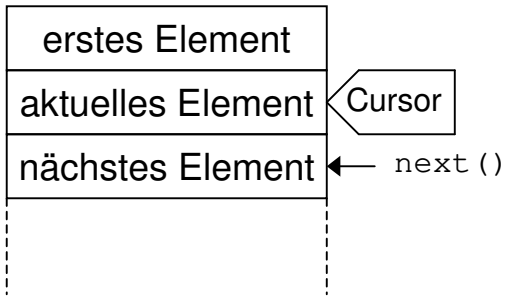
JDBC-Treiber-Varianten

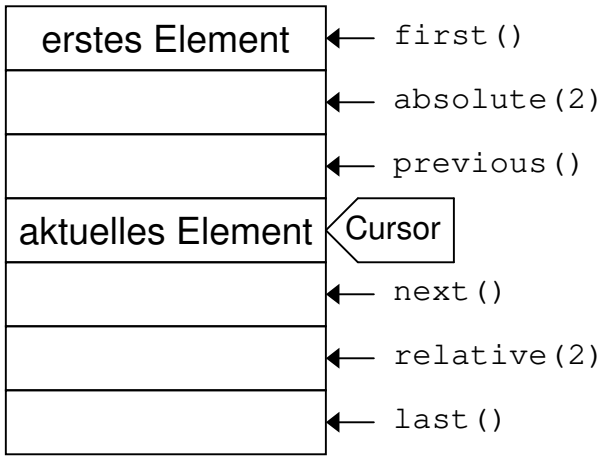


Java Database Connectivity (JDBC) (3)

Core Package

- JDBC-Basisfunktionalität:
  - Verbindungsaufbau
  - Absetzen von *Statements*
  - Erfragen von Metadaten
  - Umsetzung auf tupelorientierte Ergebnisverarbeitung mit Hilfe von *Result Sets*
- Erweiterungen seit JDBC 2.0:
  - Unterstützung von SQL:1999 -Datentypen
  - *Scrollable* und *Updatable Result Sets*:





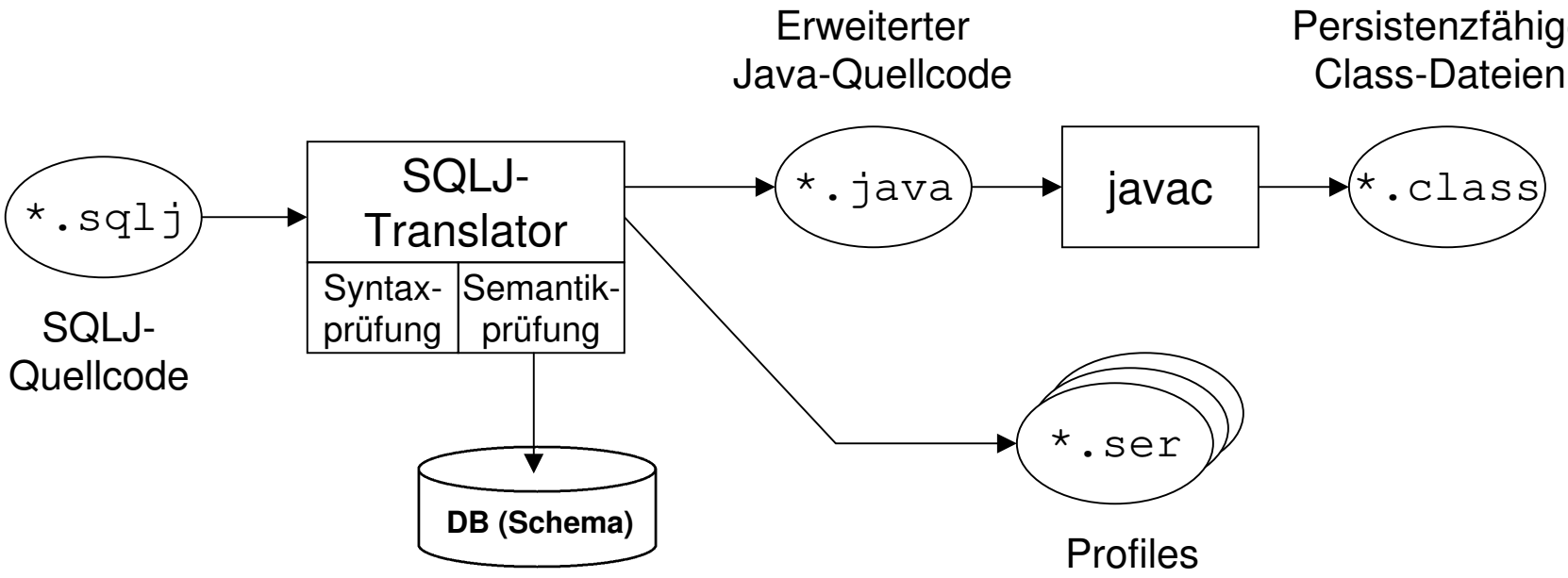
Java Database Connectivity (JDBC) (4)

Optional Package

- Verbessertes Verbindungsmanagement über `DataSource`-*Interface*
  - keine feste Kodierung von Treiber und DB-URL
  - stattdessen Zugriff auf die DB über deren logischen Namen (JNDI)
- *Connection Pooling*
  - erfolgt transparent über `DataSource`
  - freigegebene Verbindungen werden nicht zerstört, sondern wiederverwendet
- Verteilte Transaktionen
  - erfordert Unterstützung durch die Datenbank
  - keine Kontrolle der Transaktionsgrenzen durch Anwendungsprogrammierer

SQLJ (1)

- ANSI-Standard für eingebettetes SQL in Java (Abbildung auf JDBC)

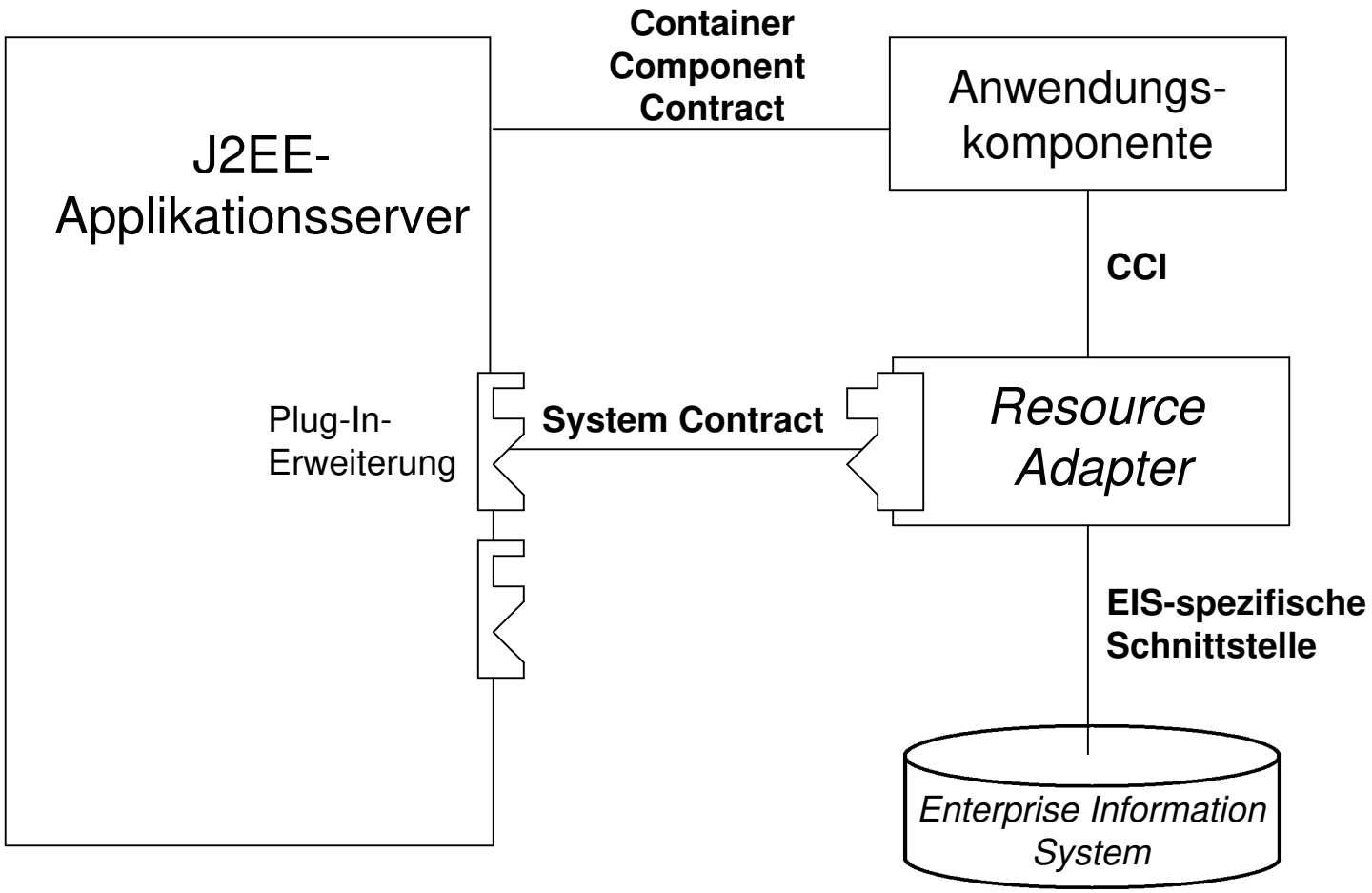


- Vorteile:
  - kurze, kompakte Schreibweise
  - Syntax- und Semantikprüfung zur Übersetzungszeit
  - größere Unabhängigkeit von verschiedenen SQL-Dialekten
- Nachteile:
  - nur statische Statements

SQLJ (2)

- zusätzliche Aspekte von SQLJ:
  - Java als Sprache für *Stored Procedures* („server-sided“ Java)
  - Verwendung von Java-Datentypen in der Datenbank
    - Verwendung wie herkömmliche SQL-Typen
    - automatische Generierung von *Wrapper*-Klassen für SQL-Typen
  - noch nicht verabschiedet
  - nicht-standardkonforme Implementierungen

J2EE Connector-Architektur



Java Transaktion API (JTA)

- Steuerung des Transaktionsverhaltens auf zwei Arten:
  - *programmatic transaction demarcation*: Entwickler setzt Transaktionsgrenzen durch entsprechende Methodenaufrufe
  - *declarative transaction demarcation*: nur für Enterprise Java Beans, deskriptive Spezifikation durch Transaktionsattribute
- Transaktionsattribute
  - *Required*
  - *RequiresNew*
  - *NotSupported*
  - *Supports*
  - *Mandatory*
  - *Never*

Enterprise Java Beans Tier

Business Objects

- realisieren Geschäftsvorgänge (*Business-Logic*)
- gemeinsame Anforderungen:
  - Wiederverwendung
  - Teilnahme an Transaktionen
  - Erhaltung des Zustands
  - Operationen auf gemeinsamen Daten
  - Bedienung einer großen Zahl von Clients
  - Zugriffskontrolle
- Realisierung dieser Anforderungen z.T. sehr aufwendig

➔ Nutzung einer Laufzeitumgebung, die diese Dienste zur Verfügung stellt

Enterprise Java Beans (EJBs) (1)

- offener Standard für verteilte Komponenten
- zwei Haupttypen:
  - *Entity Beans*
  - *Session Beans*
- J2EE-Standard definiert Anforderungen an die Laufzeitumgebung
  - Realisierungen von zahlreichen Herstellern erhältlich
  - EJBs grundsätzlich in jedem standardkonformen System einsetzbar
- Verhalten von *Beans* nicht vollständig durch Implementierung definiert

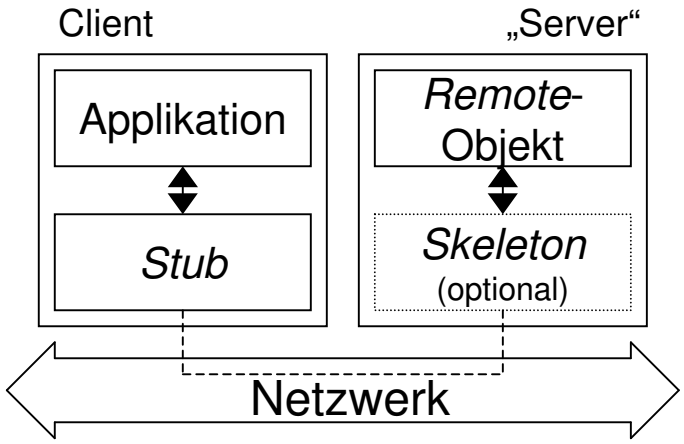
  ➔ *Deployment Descriptor*

Enterprise Java Beans (2)

- Ablauf in *Containern*:
  - erweiterte „virtuelle Maschine“ (Laufzeitumgebung)
  - zusätzliche spezielle Systemdienste
- Kontrolle durch den *Container* erfordert einheitliche Schnittstellen (*Container-component Contract*)
  - *Home-Interface* (`javax.ejb.EJBHome`)
  - *Remote-Interface* (`javax.ejb.EJBObject`)
  - *Bean*-Klasse
- *Client* kommuniziert nicht direkt mit der EJB
- *Container* reicht *Client*-Anfragen an *Bean*-Klasse weiter

Remote Method Invocation (RMI)

- Ermöglicht den Aufruf von Methoden auf entfernten Objekten über deren *Remote Interface*
- Erfragen eines Handles für das benötigte Objekt über einen *Naming Service*



RMI over IIOP (RMI-IIOP)

- RMI nur für Kommunikation zwischen Java-Klassen
- RMI over IIOP: Erweiterung von RMI um CORBA-Fähigkeiten:
  - transparente Kommunikation mit Nicht-Java-Komponenten

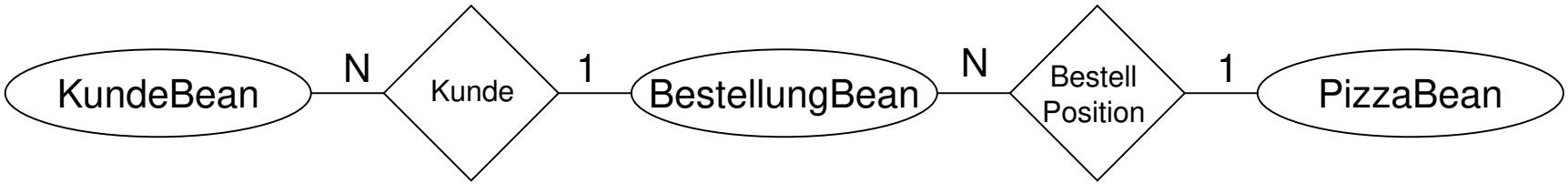
Entity Beans (1)

- Bereitstellung von Persistenz auf zwei Arten:
  - *bean-managed persistence*
  - *container-managed persistence*
- Verwendung:
  - Modellierung einzelner Tabellenzeilen
  - komplexere persistente Business-Objekte
- Spezielle Methoden (Auswahl)
  - `ejbLoad`/`ejbStore`: Synchronisation der Bean mit Ihrem Datensatz
  - `ejbPassivate`/`ejbActivate`: Aus-/Einlagern der Bean bei längerer Inaktivität
  - `ejbFind`: Suche innerhalb des Bean-Typs nach beliebigen Kriterien
  - `ejbSelect`: allgemeinere Anfragen an die Beans

Entity Beans (2)

EJB Query Language (EJB-QL)

- verfügbar seit EJB 2.0
- orientiert sich stark an SQL92
- gestattet die deklarative Definition der Semantik der `ejbFind`- und `ejbSelect`-Methoden
- Datenmodell für die Anfragen ergibt sich aus den Beziehungen zwischen den Beans, beschrieben im *Deployment Descriptor*:



ejbSelectKundenAdresse(String pizzaname)

```
SELECT  K.lieferadresse
FROM    B FOR BestellungBean
        BP IN b.BestellPosition
        P FOR BP.pizza
        K FOR BP.kunde
WHERE   P.name= pizzaname AND
        K.datum = today
```

Session Beans

- Stateful Session Beans
  - speichern client-spezifischen Zustand für die Dauer einer Session
  - Einsatzmöglichkeiten:
    - Realisierung von Aufgaben für einen Client
    - Modellierung kurzlebiger Objekte
- Stateless Session Beans
  - enthalten keinen client-spezifischen Zustand
  - Einsatzmöglichkeiten:
    - generische Dienste
    - Objekte mit hohen Performanzanforderungen
- Session Beans als *Facade*:
  - zentraler Einstiegspunkt für Clients

Message Driven Beans

Java Message Service API (JMS)

- synchroner und asynchroner Nachrichtenempfang
- Nachrichten werden über einen zentralen Punkt ausgetauscht:
  - *Point-to-Point- Messaging*
  - *Publish/Subscribe-Messaging*
- lediglich Einigung auf gemeinsames Nachrichtenformat notwendig

  ➔ lose Kopplung zwischen Sender und Empfänger

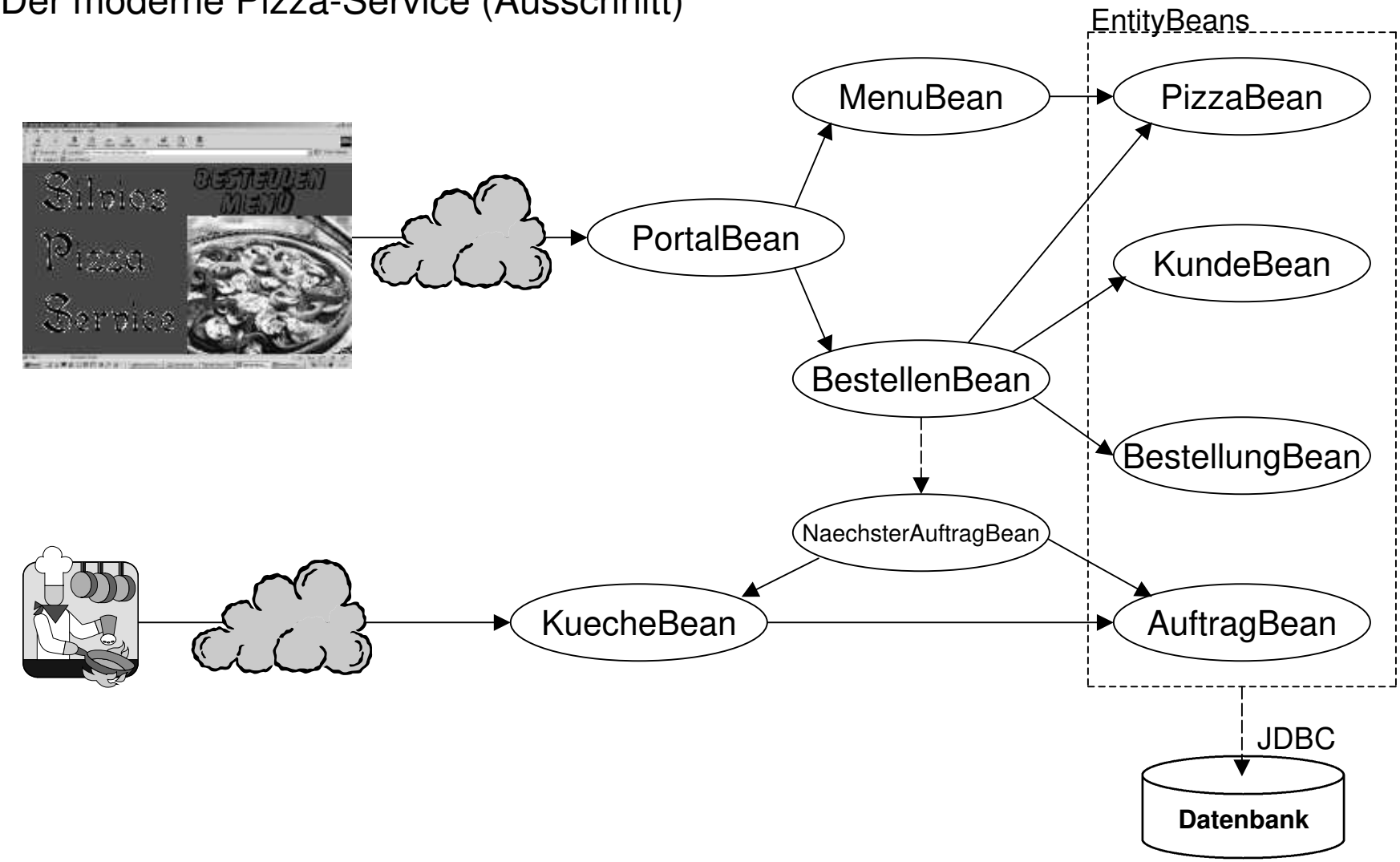
Message Driven Beans

- asynchroner Empfang von Nachrichten

  ➔ zeitliche Entkopplung vom Senden der Nachricht
- nebenläufige Verarbeitung mehrerer Nachrichten möglich
- vergleichbar mit *Stateless Session Beans*
- keine Interaktion mit Clients (kein Home- und Remote-Interface)

Nutzung von EJBs

Der moderne Pizza-Service (Ausschnitt)



Produkte mit EJB-Technologie (Auswahl)

J2EE-Applikationsserver

- BEA WebLogic
- Inprise Application Server
- IBM WebSphere Application Server
- Oracle Application Server 9i

J2EE-Komponenten

- IBM SanFrancisco

J2EE-Applikationen

- abaXX E-Business Suite
  - DekaBank
  - Dresdner Bank (Firmenfinanzportal)
  - Payback

Zusammenfassung

- Informationssysteme sind Grundlage für Geschäftsanwendungen
- Herausforderung: vereinheitlichter Zugriff
  - JDBC als ausgereifte Schnittstelle für den Zugriff auf (O)RDBMS
  - JDBC 3.0 verspricht weitere Verbesserungen
  - *Connector*-Architektur als zukünftiger Standard für den Zugriff auf beliebige Informationssysteme
- Realisierung von Geschäftsanwendungen mit Business Objekten
  - Frameworks zur Realisierung der gemeinsamen Anforderungen
  - EJBs als offener Standard