Konzepte der Datenbanktechnologie

Prof. Dr. U. Hoffmann
FH Wedel

Einführung in objekt-relationales Mapping

Gliederung

Java DB-Zugriff

- ODBC
- Enterprise Java Beans

Hibernate Intro

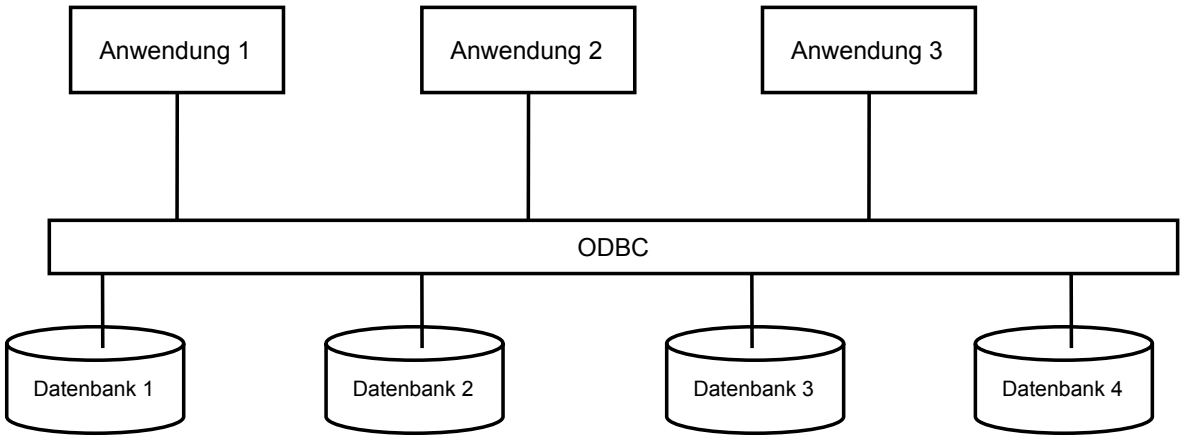
Können nicht relationale Datenbanken als Objektdatenbanken benutzt werden?

Welche Möglichkeiten gibt es in Java Objekte persistent in relationalen Datenbanken abzulegen?

- ODBC/JDBC
- JDO
- EJB 2
- EJB 3 (Java Persistence API)
- Hibernate (JPA)

ODBC: Open Database Connectivity

Ziel: Standardisierter Zugriff auf beliebige Datenquellen mit SQL-Befehlen

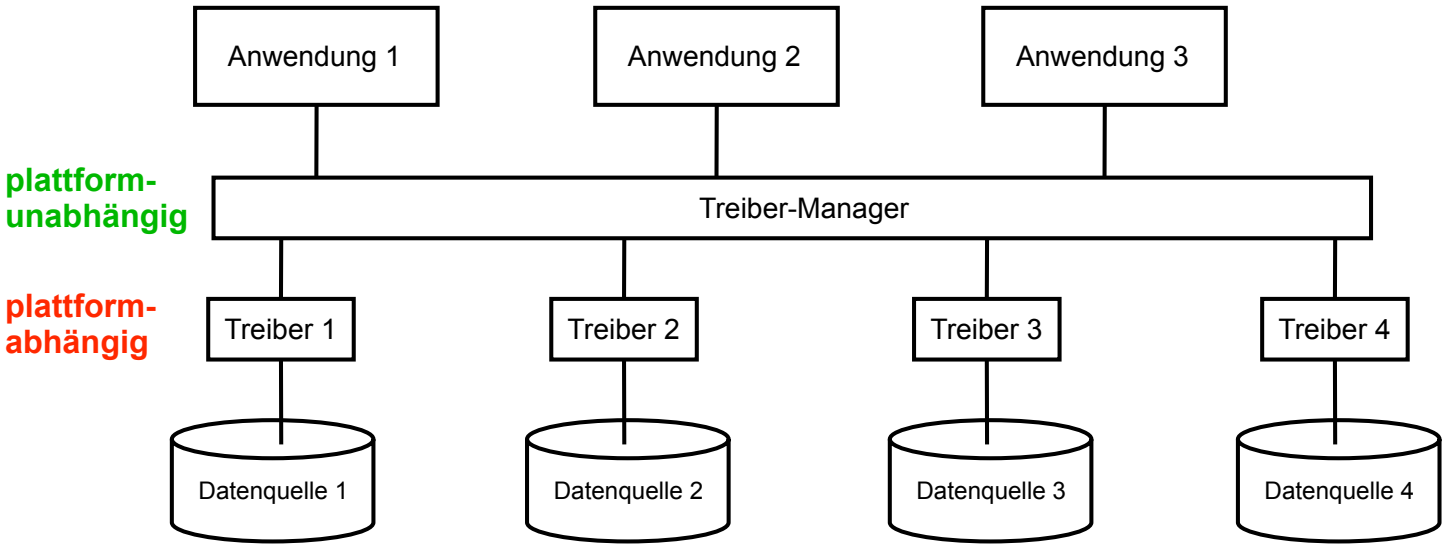


Geschichte

- entwickelt Anfang der 90'er Jahre mit Microsoft-Unterstützung
- Anwendungsfokus: C- / C++- Anwendungen

ODBC: Open Database Connectivity

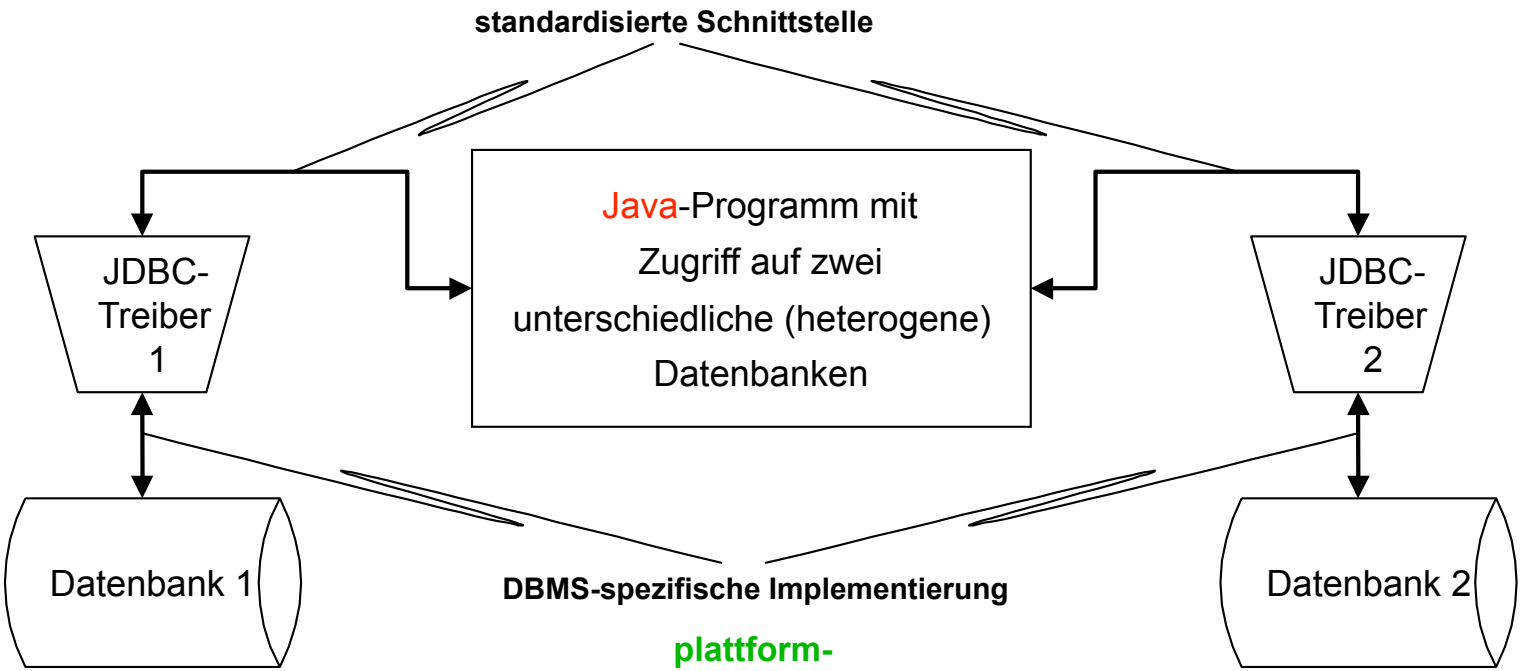
Architektur



Mögliche Datenquellen: (müssen keine Datenbanken sein)

- Tabellenkalkulationen
- Textdateien in beliebigem Format
- Desktop-Datenbanken (für Einzelbenutzer)
- Server-Datenbanken (für parallelen Mehrbenutzerbetrieb)

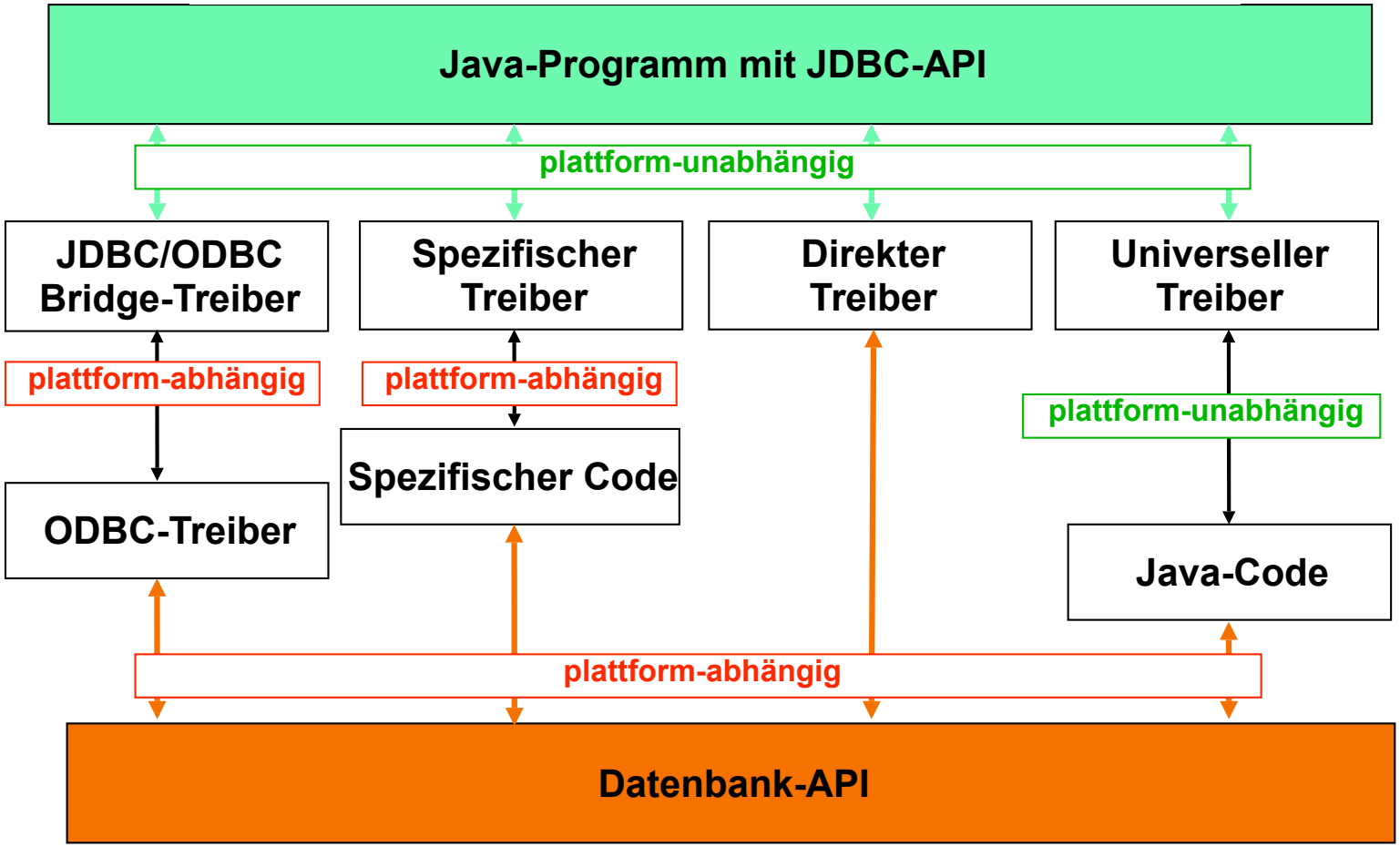
JDBC: Java Database Connectivity



plattform-unabhängig

(im Prinzip …)

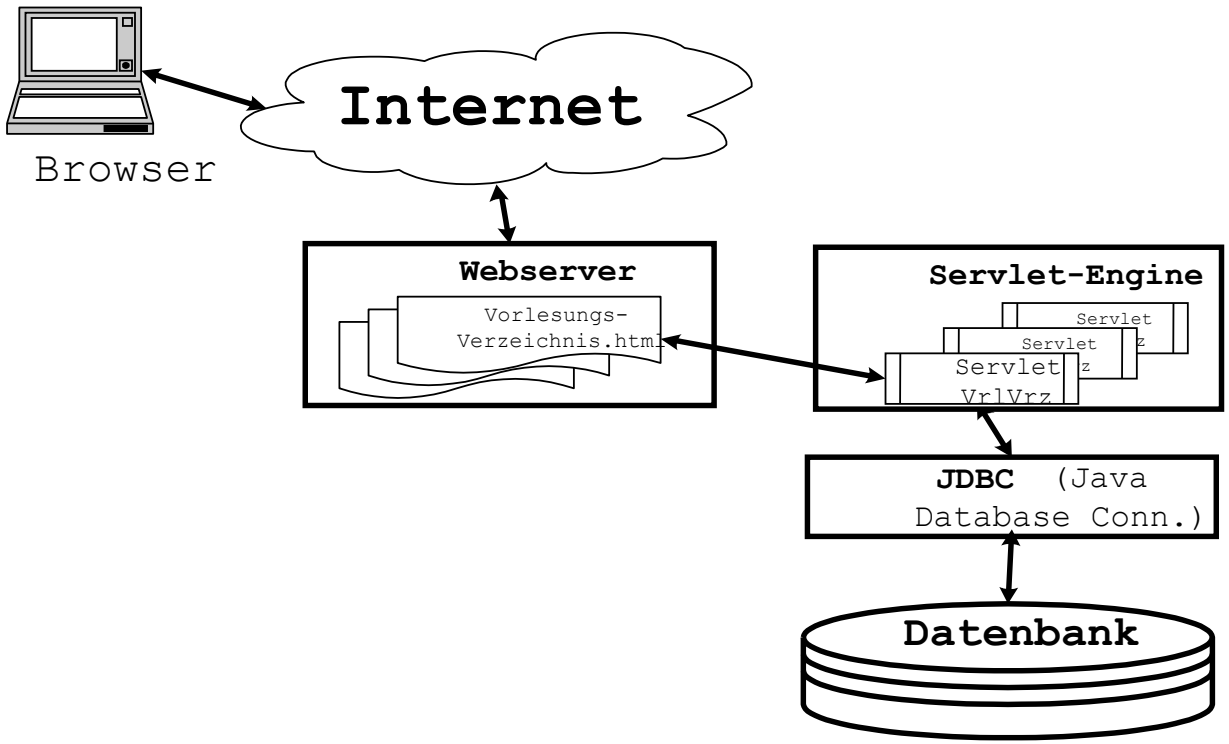
Anbindungsmöglichkeiten für JDBC



Unterschiedliche Wege zu einem JDBC-Treiber zu kommen

Aktuelle Verwendung von JDBC:

Anbindung von Datenbanken an das Internet



- Java Servlets als dynamische Erweiterung von Webservern
- Java Server Pages (JSP):
  HTML-Seiten mit eingebetteten Java Programmfragmenten

Java-API für JDBC

Laden des JDBC-Treibers

- dynamisch: `static Class Class.forName(String drivername)`
- statisch: durch Spezifikation im properties-File

Verbindungsaufbau zur Datenbank

- `static Connection DriverManager.getConnection(String url)`

Generierung eines Anfrageobjekts

- `Statement Connection.createStatement()`
- Hierdurch werden Eigenschaften der Antwort festgelegt.

Formulierung und Stellen der Frage

- `ResultSet Statement.executeQuery(String sqlQuery)`
- generiert Datenbanktabelle (mit Zeilen und Spalten)

Java-API für JDBC

Navigieren in der Antwort:

```
boolean ResultSet.next()
boolean ResultSet.previous()
```

navigiert zur nächsten bzw. vorigen Zeile der Antwort

```
String ResultSet.getString(int index)
int ResultSet.getInt(int index)
```

liest das Element in Spaltenposition index in der gegenwärtigen Zeile

```
String ResultSet.getString(String name)
int ResultSet.getInt(String name)
```

liest das Element in Spalte name in der gegenwärtigen Zeile
ResultSet bietet viele weitere nützliche Methoden.

Fehlerbehandlung: Alle Zugriffe auf Datenbank mit `try` ... `catch`

Die Interfaces `Connection`, `Statement`, `ResultSet` und die Klasse `DriverManager` befinden sich im `package java.sql`

Java-API für JDBC: Einfaches Beispiel

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Beispiel

```
try {
      Class.forName("oracle.jdbc.driver.OracleDriver");
      conn = DriverManager.getConnection
                ("jdbc:oracle:oci8:@lsintern-db", "nobody", "Passwort");
      sql_stmt = conn.createStatement();
} catch (Exception e) {
      System.err.println("Folgender Fehler ist aufgetreten: " + e);
      System.exit(-1);
}
try {
      ResultSet rset = sql_stmt.executeQuery(
        "select Name, Raum from Professoren where Rang = 'C4'");
      System.out.println("C4-Professoren:");
      while(rset.next())  {
        System.out.println
        (rset.getString("Name") + " " +  rset.getInt("Raum"));
       }
      rset.close();
} catch(SQLException e) { System.out.println("Error: " + e); }

try {
      sql_stmt.close(); conn.close();
} catch (SQLException e) {
      System.out.println("Fehler beim Schliessen der DB: " + e);
}
```

Visual-Basic-API für ODBC: Einfaches B

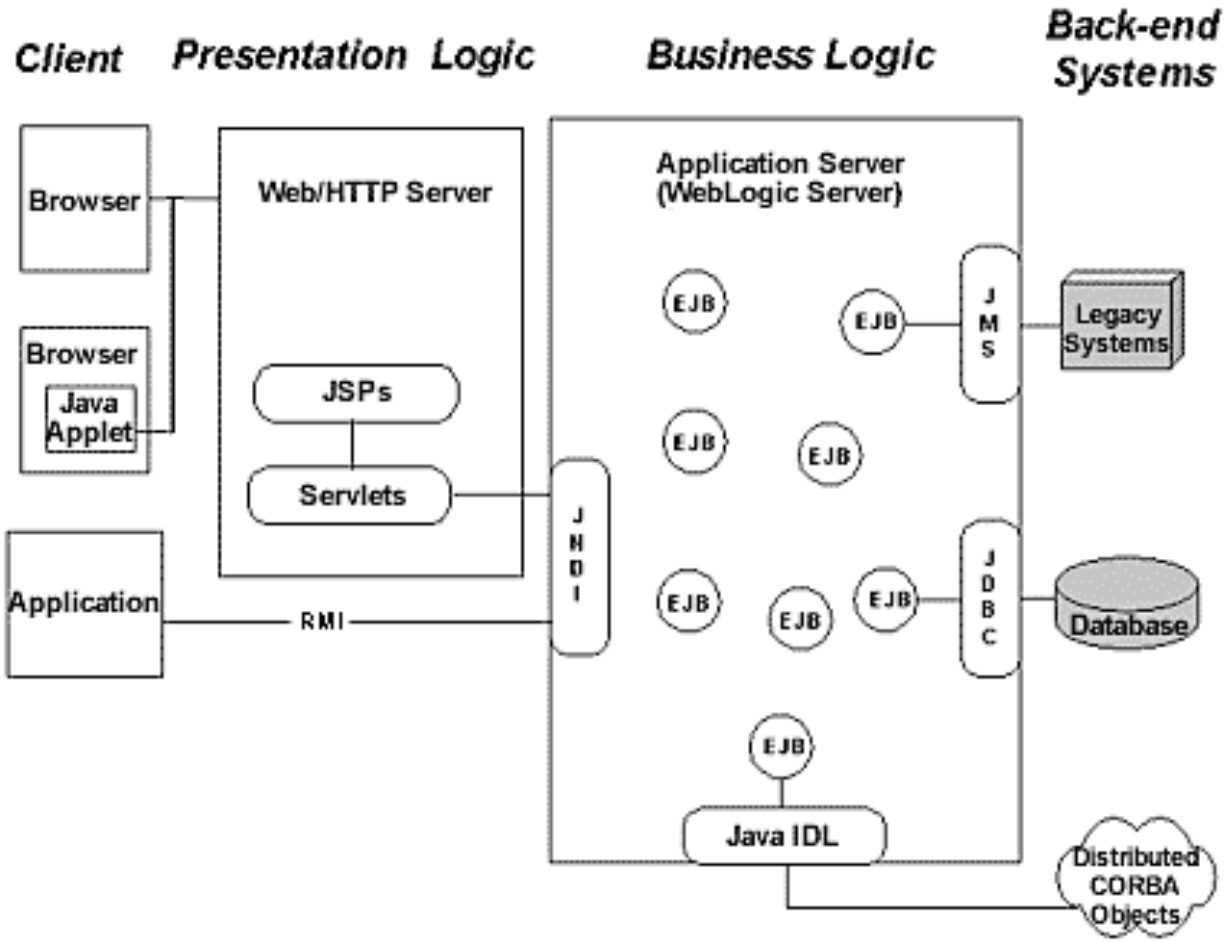
Beispiel ()

```
Try
  Dim ConString As String = "DRIVER={MySQL ODBC 3.51 Dri
      "SERVER=localhost;DATABASE=lsinter-db;" & _
      "UID=nobody;PASSWORD=Passwort;OPTION=3;"

  Dim Conn As New OdbcConnection(ConString) : Conn.Open()

Catch ex As Exception
   Console.WriteLine("Folgender Fehler ist aufgetreten:" & ex.ToString)
End Try

Try
  Dim SQL_stmt As New OdbcCommand() : SQL_stmt.Connection = Conn

  SQL_stmt.CommandText = "select Name, Raum from Professoren where Rang = 'C4'"

  Dim DataReader As OdbcDataReader
  DataReader = SQL_stmt.ExecuteReader
  While DataReader.Read
     Console.Write("Name = " & CStr(DataReader("Name")) & _
                   ", Raum = " & CStr(DataReader("Raum")))
  End While

Catch ex As OdbcException
   Console.WriteLine("Fehler beim Lesen aus der Datenbank" & ex.ToString)
End Try

Try
   Conn.Close
Catch ex As Exception
   Console.WriteLine("Fehler beim Schliessen der Datenbank" & ex.ToString)
End Try
```

Professoren

PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

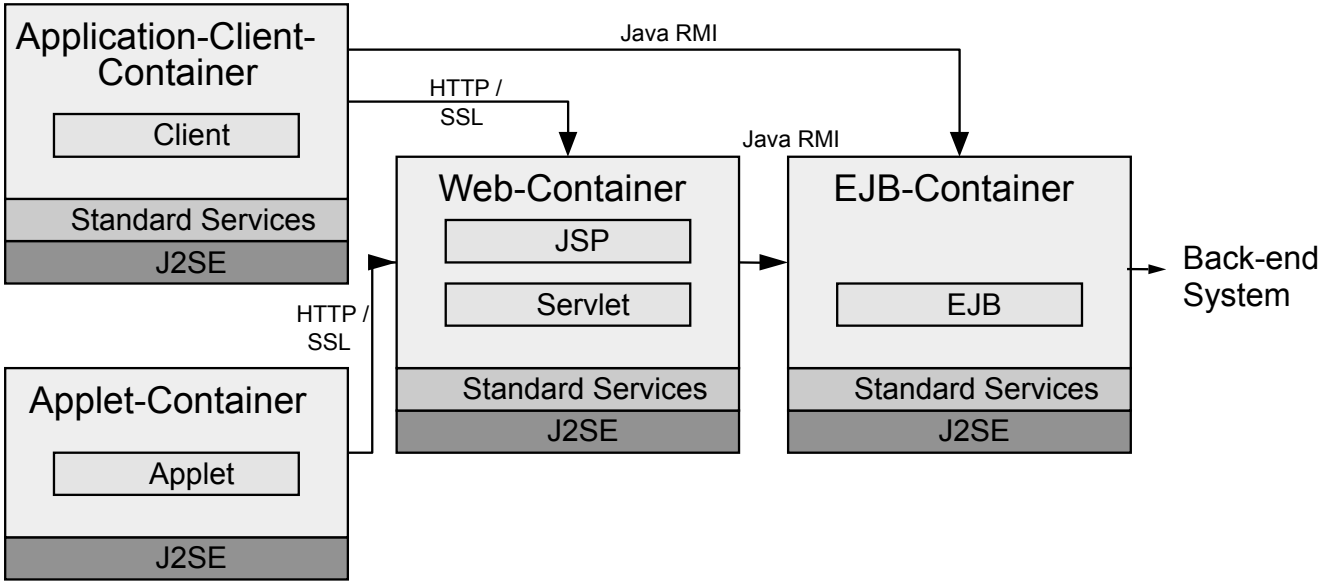
J2EE: Java 2 Enterprise Edition



J2EE: Bestandteile (einiges auch schon in J2SE)

- Namensraumverwaltung (JNDI: Java Naming Directory Interface)
- Transaktionsverwaltung (EJB: Enterprise Java Beans)
- Verbindung mit HTML-Code (JSP: Java Server Pages)
- Sicherheitsverwaltung
- Datenbankanbindung (JDBC: Java Database Connectivity)

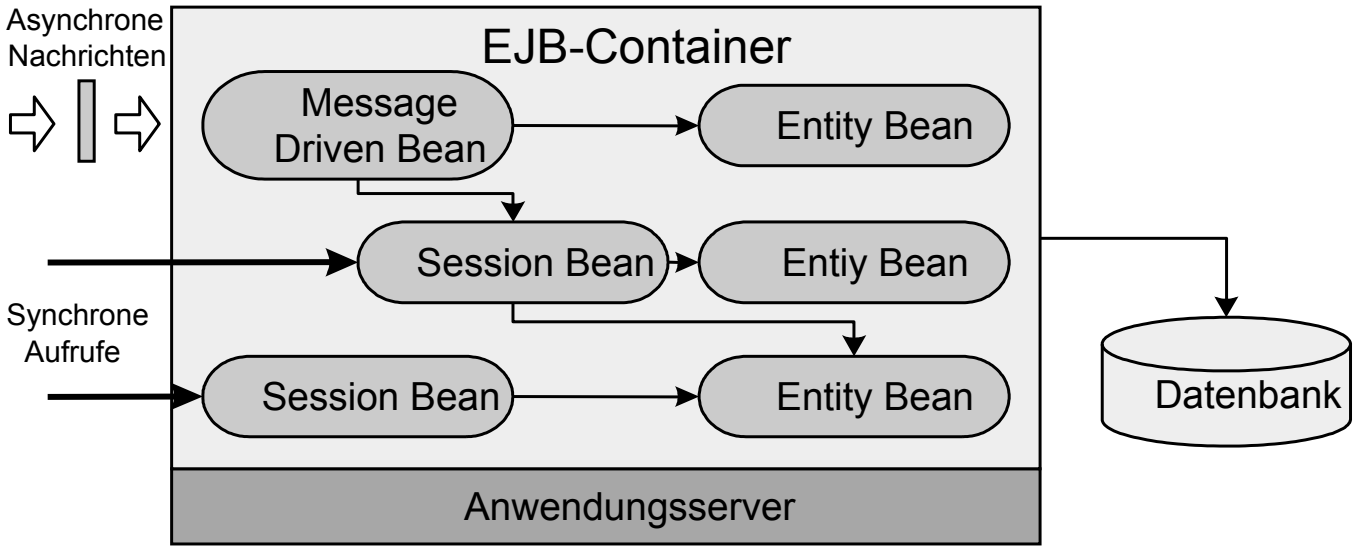
Architektur



- *Container-Architektur*
- vier Komponentenmodelle, jeweils mit Container und zugehörigen Komponenten

Quelle: *Verteilte Systeme und Anwendungen*, Ulrike Hammerschall

Funktion im EJB-Container



Die Struktur einer EJB-Anwendung orientiert sich an folgendem Aufbau:

- **Session Beans** bilden die Schnittstelle zum Client
- **Message Driven Beans** bilden die Schnittstelle zu einem JMS-Provider
- **Entity Beans** bilden die Schnittstelle zur Datenhaltung

Quelle: *Verteilte Systeme und Anwendungen*, Ulrike Hammerschall

EJB 2: Kritik aus der Praxis

Probleme mit EJB2:

- lange edit-compile-debug-Zyklen
- keine Trennung von Aufgaben gemäß der Schichten
- viel Code
- Arbeiten mit Data Transfer Objects

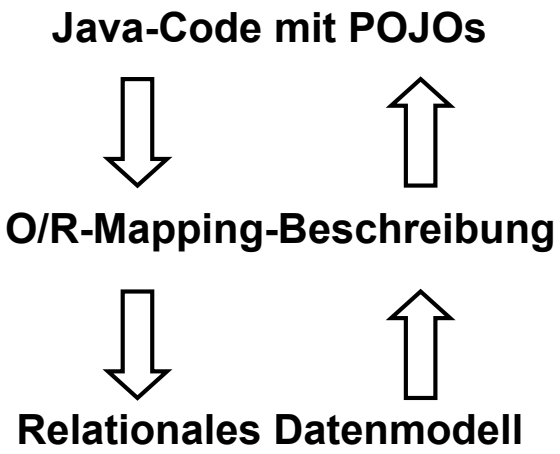
Sehnsüchtiger Ruf nach POJOs:
Plain Old Java Objects

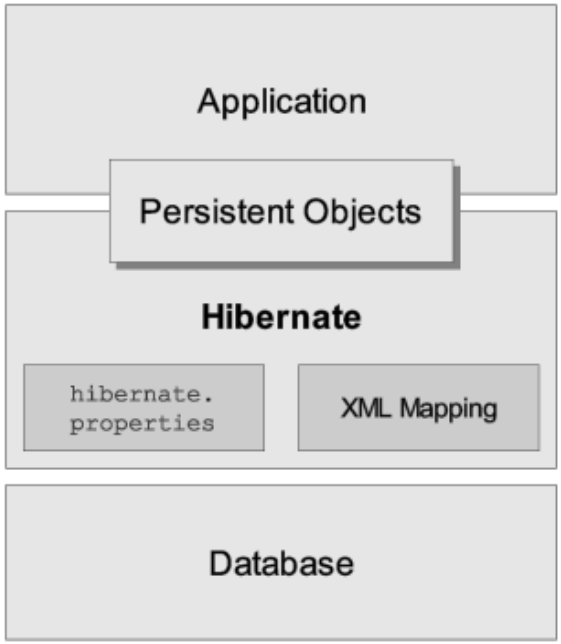
Wird in JPA / EJB3 / Hibernate aufgenommen.

Historie von Hibernate

- **Ende 2001:** initiiert von Gavin King
- weitergeführt als Open Source Projekt `www.hibernate.org`
- **Ende 2003:** aufgenommen von JBoss in J2EE-Entwicklung, vor allem als Alternative zum EJB2-Standard
- **2004:** 1. Auflage des Buchs *Hibernate in Action*
- **2004:** Aufnahme vieler Hibernate-Konzepte in EJB3-Standard angekündigt, inzwischen verwirklicht
- **2006:** 80000 Downloads / Monat
- **2006:** 2. Auflage des Hibernate-Buchs: *Java Persistence with Hibernate*
- **2007:** .NET-Version NHibernate

Prinzip von Hibernate





Gewünschte Funktionalität

- Automatische Überführung von einer Beschreibungsebene in die nächste

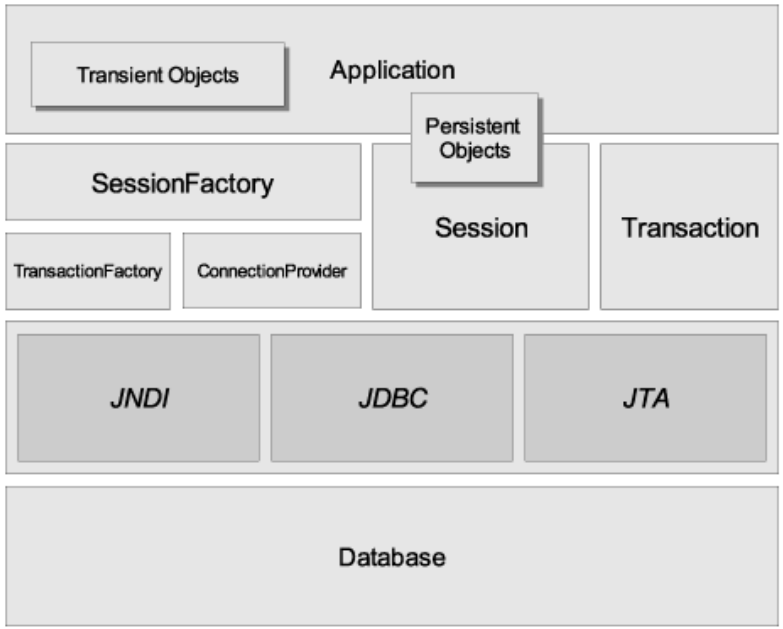
Prinzip von Hibernate

Java–Code mit POJOs

Hibernate-Schicht

Java–APIs

Relationales Datenmodell



Gewünschte Technologie

- Alle vorhandenen Java-APIs sollen genutzt werden

Wesentliche Eigenschaften von Hibernate

- Transparente Persistenz
- Transitive Persistenz (Persistenz per Erreichbarkeit)
- Detached Object Support
- Inheritence mapping strategies
- Intelligent fetching and caching
- Automatic dirty object checking
- Unterschiedliche Anfragekonzepte: Queries und Criteria

Zusammenfassung

Java DB-Zugriff

- ODBC
- Enterprise Java Beans

Hibernate Intro