

Architekturen

Von der DB basierten zur Multi-Tier Anwendung

Lernziele

- Sie kennen Design und Architektur Patterns, welche beim Datenbankzugriff in verteilten Systemen verwendet werden
- Sie können ODBC und JDBC gegeneinander abgrenzen
- Sie wissen, dass Architekturen langlebiger als konkrete Produkte sind
 - Architekturen lassen sich längerfristig standardisieren
=> Konzeptionelle Modelle
 - Produkte wechseln zum Teil sehr schnell
=> Lebenszyklen bei den Investitionen beachten
(bei auslaufenden Modellen eventuell Miete, Leasing, ...)

- Remote Database Access
 - Einführung in die Problematik
- ODBC
 - Architektur
- JDBC
 - Architektur
- Enterprise Lösungen
 - Multi Tier Applikationen : Applikations- Szenarios
 - Client Tier
 - Web Tier
 - Applikations Tier

Warum?

- Kernthema einer Applikations-Entwicklung
 - Wie kann meine Applikation oder Teile davon möglichst lange leben (Investitionsschutz)?
- Lösungsansatz
 - Architekturen basierte Informationssysteme planen und auswählen
 - MVC Pattern als Gedankenstütze
 - Darstellung (View)
 - Logik (Model)
 - Koordination (Controller)
 - View
 - Upper / Front End : GUI, Thin Client, DB unabhängig
 - Model
 - Lower / Back End : DBMS abhängig (so universell wie möglich)

Mögliche Ansätze

- Trennen der View vom Model und dem Controller
 - Ansatz 1
 - ODBC : Open Database Connection
 - oder JDBC (Java Version von ODBC)
 - Ansatz 2
 - DB unabhängige Tools für die Entwicklung der View, des Frontends einsetzen
 - Ansatz 3
 - Multi Tier Architektur festlegen
 - Praxis
 - Kombination aller drei

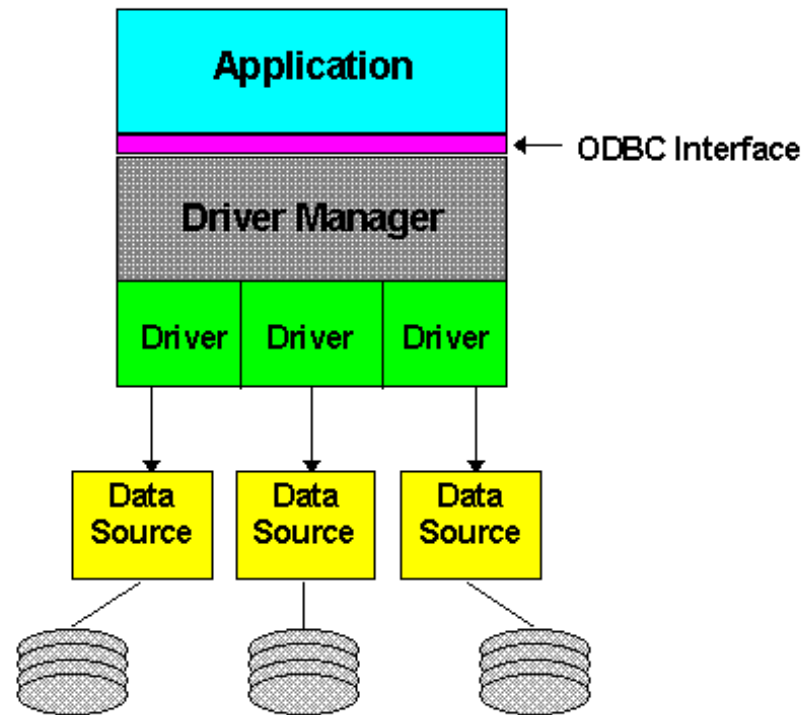
ODBC - Open Database Connectivity

- ODBC ist der Desktop Standard!
 - Für PCs, Workgroup und unternehmensweite Datenbank-Anwendungen
- ODBC ist herstellerunabhängig,
 - obschon der Standard von Microsoft um 1990 entwickelt wurde
- ODBC basiert auf
 - CLI (Call Level Interface) von X/Open und SQL Access Group

ODBC - Einschränkungen

- ODBC benutzt das CLI für SQL, um eine einheitliche SQL Syntax zu definieren
- Da jeder DBMS Hersteller einzigartig sein möchte, gibt es unterschiedliche „Dialekte“ von SQL

ODBC Architektur



ODBC - Architektur

- Basis
 - Daten : Files, relationale, Datenbanken
- ODBC Driver
 - auf jedem PC installiert
 - setzt ODBC gerechte Abfragen ab
 - setzt SQL Anfrage so um, dass die Datenquelle sie versteht (update, delete, select, rollback)

ODBC - Architektur

- ODBC Driver Manager
 - liefert eine Liste mit möglichen (ODBC) Datenquellen
- ODBC Query Tool (Desktop Anwendung)
 - erlaubt es dem Benutzer die Abfragen zu formulieren und die Ergebnisse auszuwerten

ODBC Spezifikation

- Es gibt drei Konformitäts-Levels:
 - Core Level API
 - Ressourcen zuordnen, mit Datenquellen verbinden, Abfragen vorbereiten und ausführen
 - Level 1 API
 - Core API + Möglichkeit Informationen über die Daten zu liefern (Datenkatalog)
 - Level 2 API
 - Level 1 + Möglichkeit Verbindungsparameter abzufragen, ...

ODBC Support

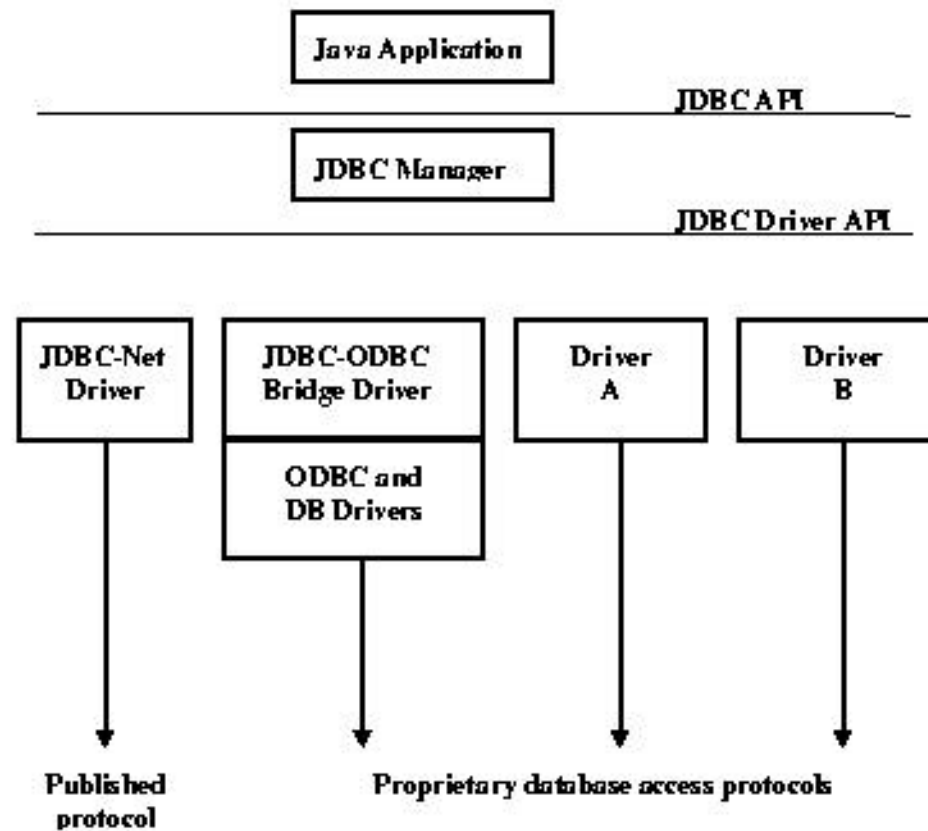
- Alle grösseren Datenbank-Anbieter unterstützen ODBC
 - Oracle
 - Sybase
 - Microsoft
 - IBM
 - ...
- Praxis
 - Excel als ODBC Datenquelle
 - später wird eine andere Datenquelle unter dem selben ODBC Namen (und der selben Tabellenstruktur) angeboten
=> keine Programmänderung!

JDBC - Java Database Connectivity

- 1996 von Javasoft entwickelt
- Idee von JDBC ist analog zu ODBC
- basiert auch auf CLI von X/Open
- ermöglicht die Datenbankbindung von Applets und Java Applikationen
 - diese ist aber auch über ODBC möglich
 - JDBC bietet auch eine „Brücke“ zu ODBC

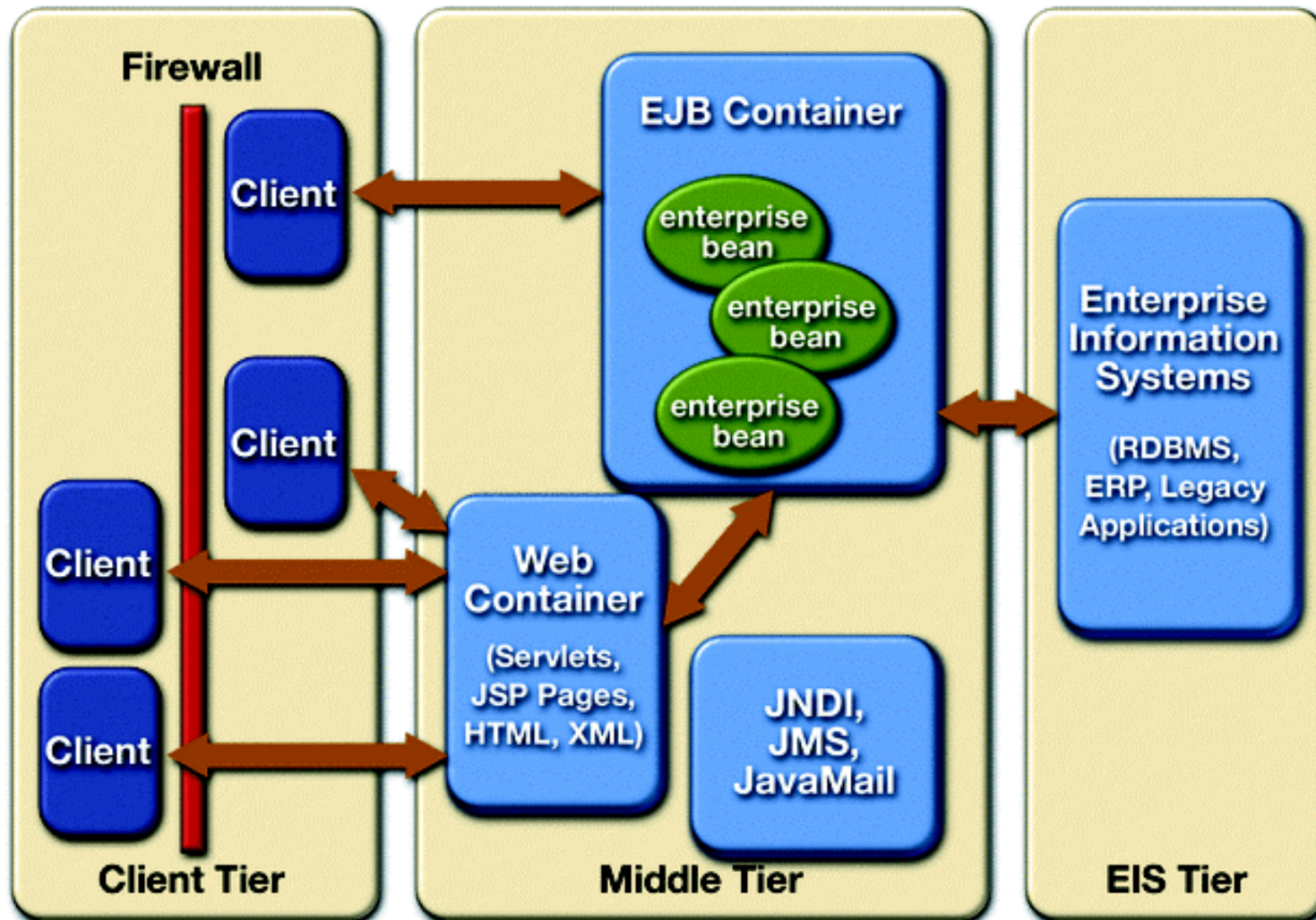


JDBC Software Struktur



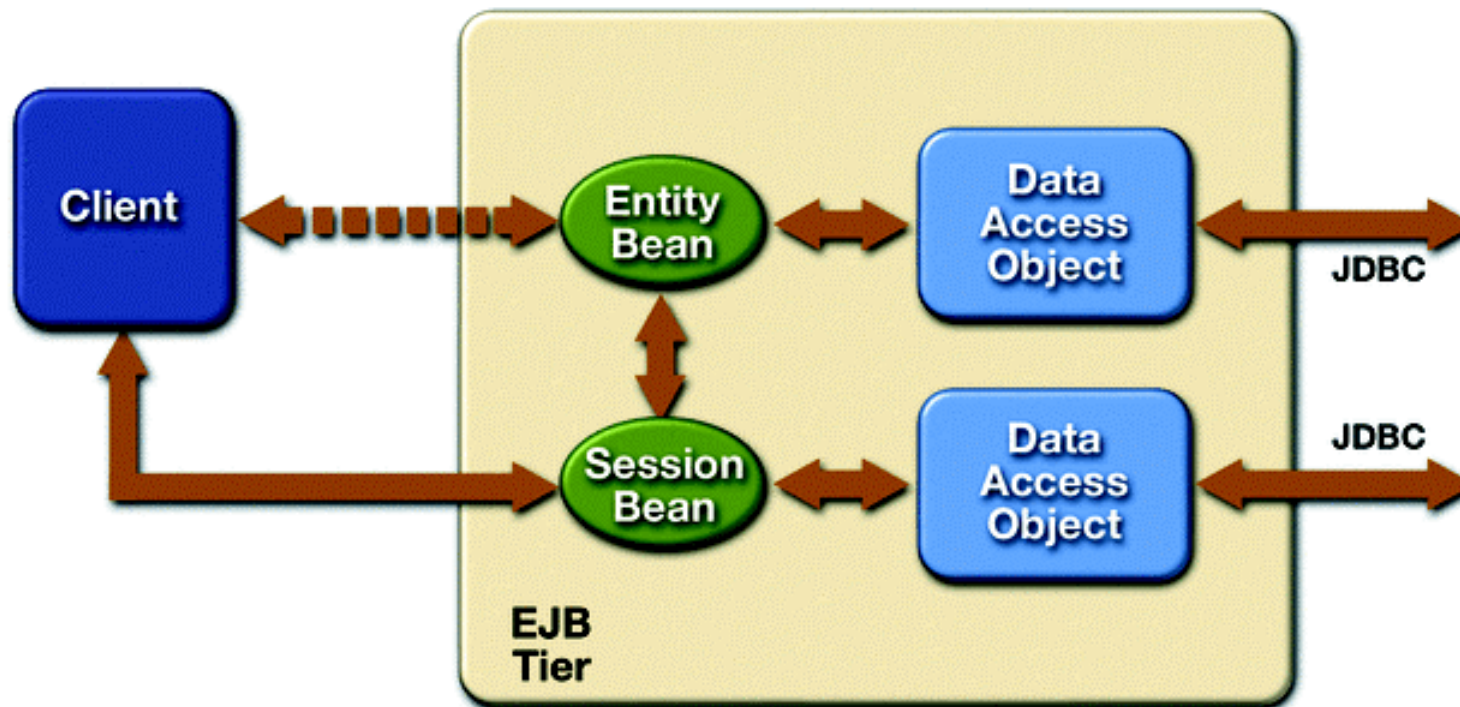
- Multi Tier Architekturen
 - Client Tier
 - HTML
 - Java Applet
 - Java Client (Handy, Palm, Notebook, ...)
 - Web Tier
 - handling der Web basierten Aufgaben
 - Applikations Tier
 - Applikations-Server (in der Regel mit DBMS)
 - Service Tier
 - Web Services (beispielsweise Message basiert)

Enterprise Architekturen



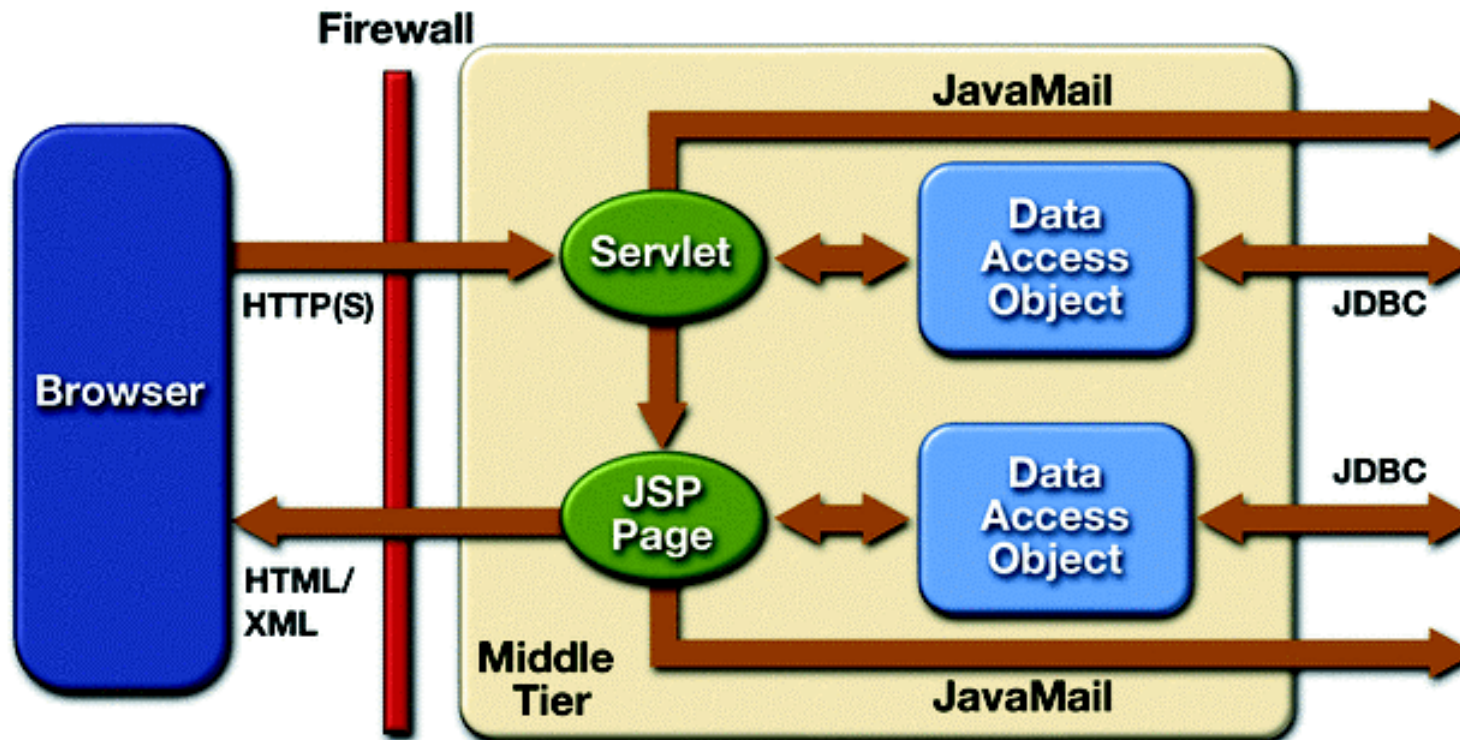
Enterprise Architekturen

Thin Client an DBMS über Applikationsserver



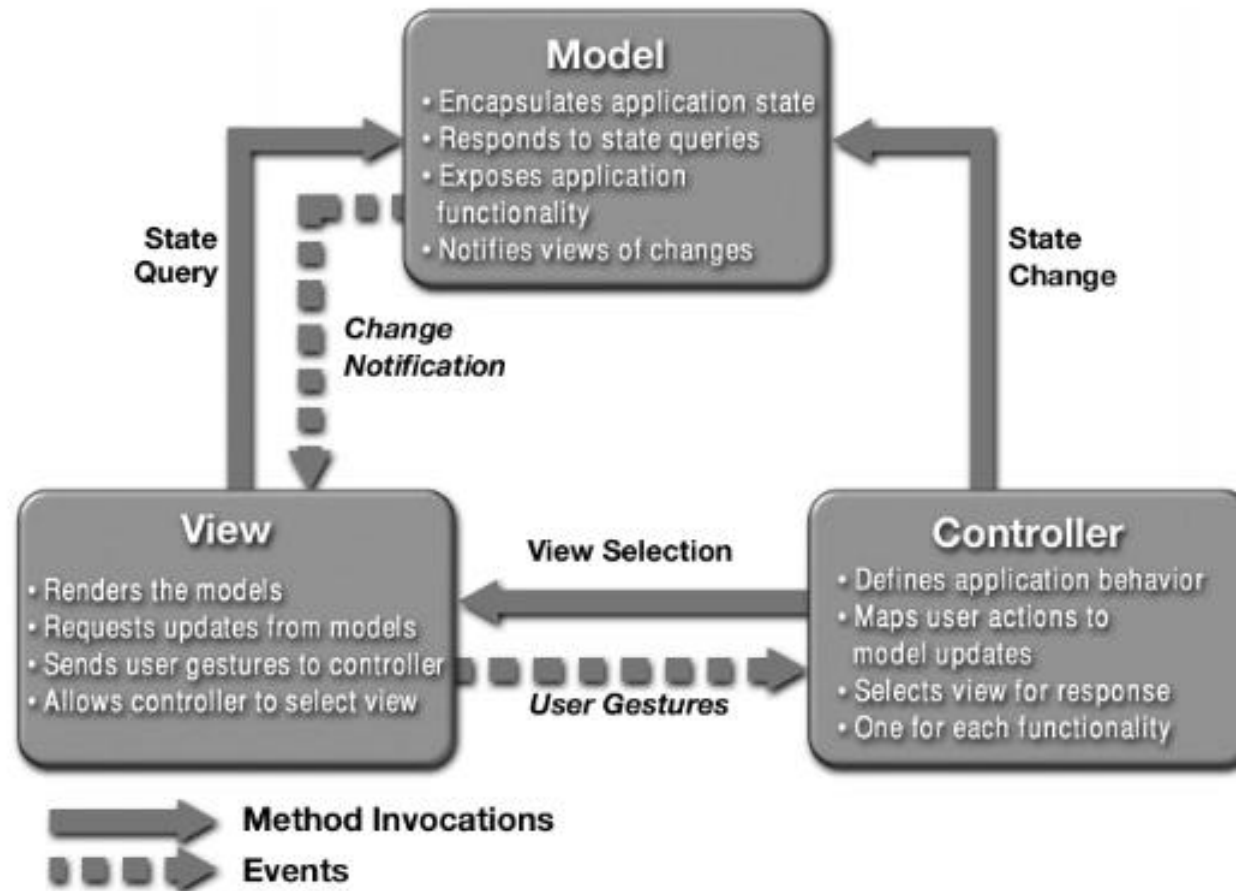
Enterprise Architekturen

Thin Client an DBMS über Web Tier



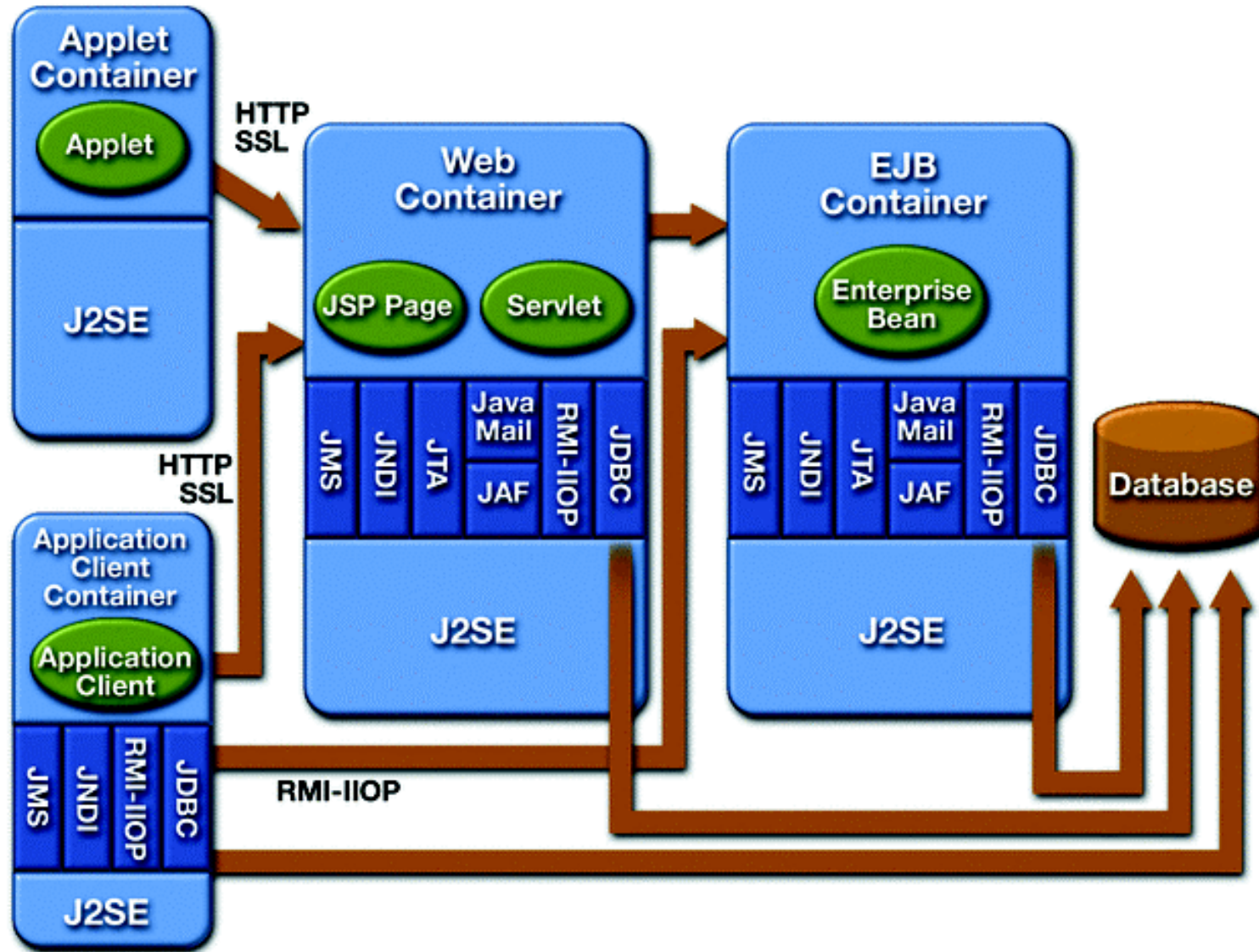
Enterprise Architekturen

Das MVC Pattern



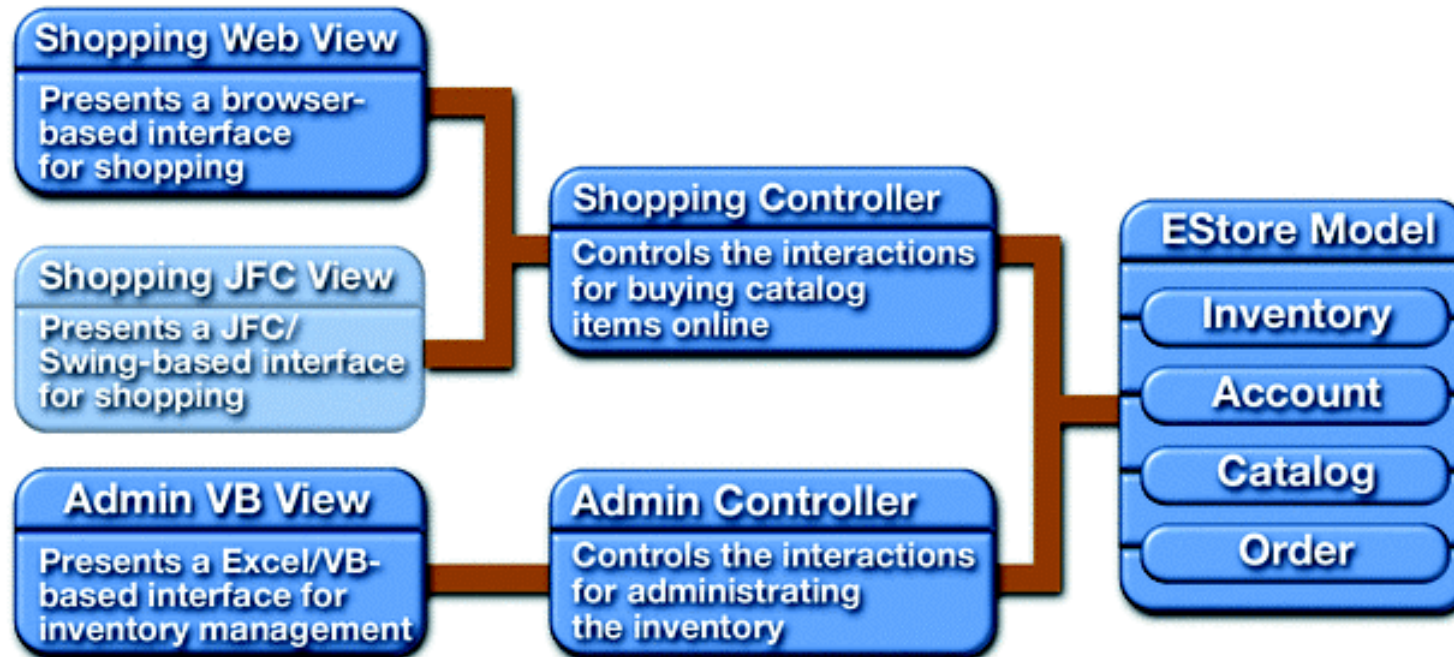


Enterprise Architekturen – Java Technologien



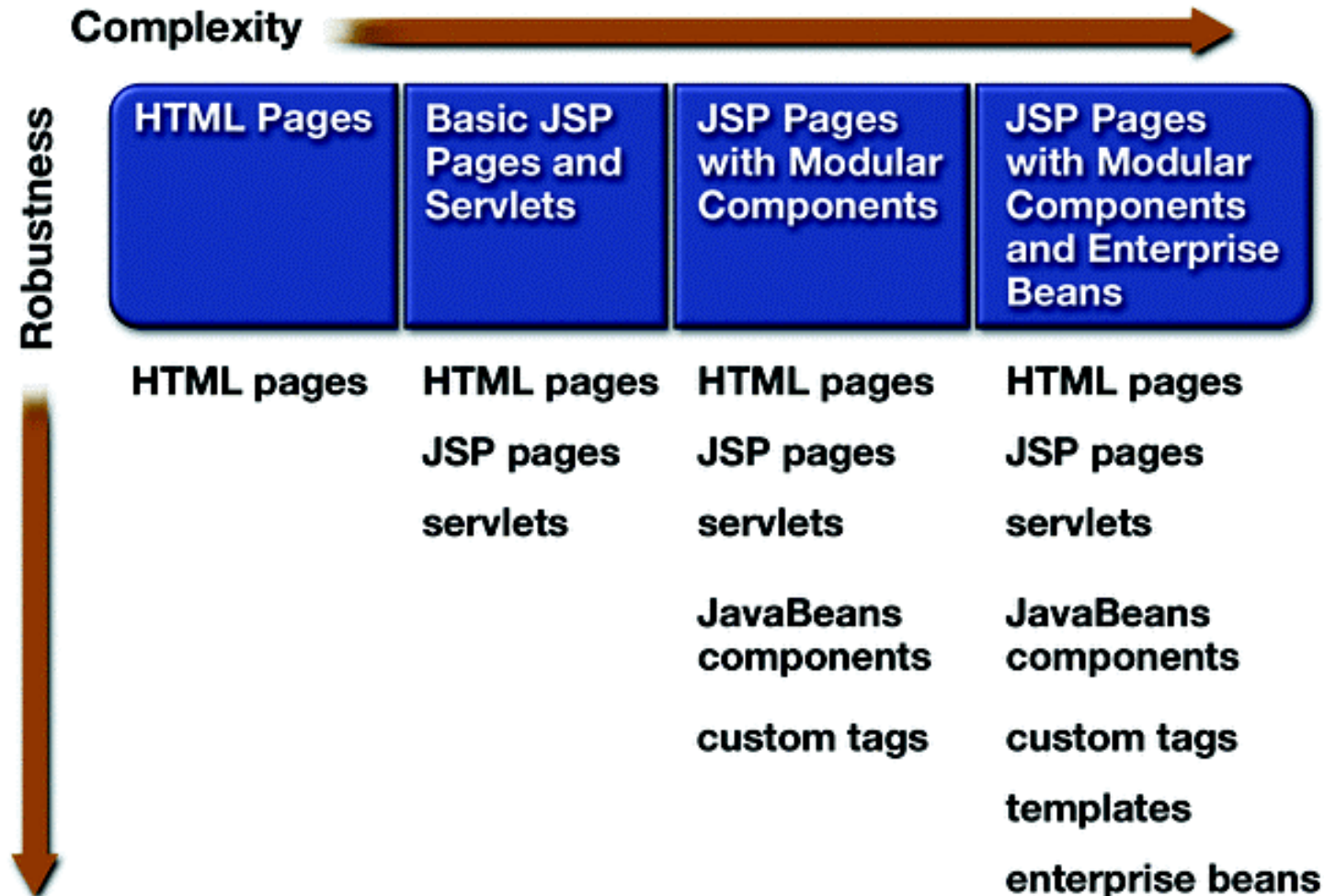


Enterprise Architekturen – Beispiel

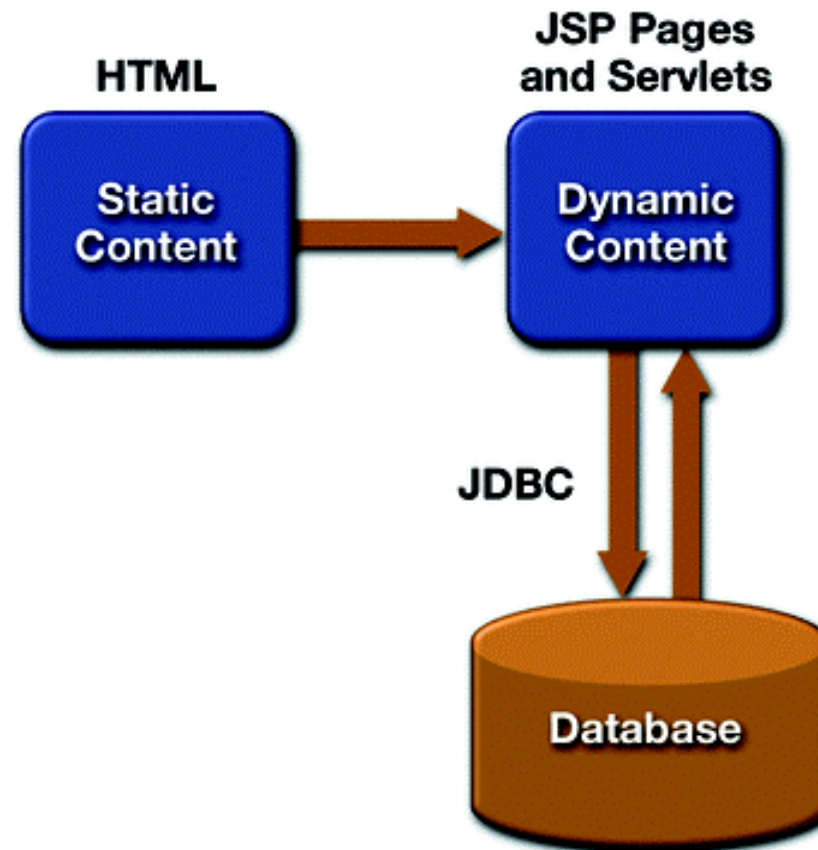




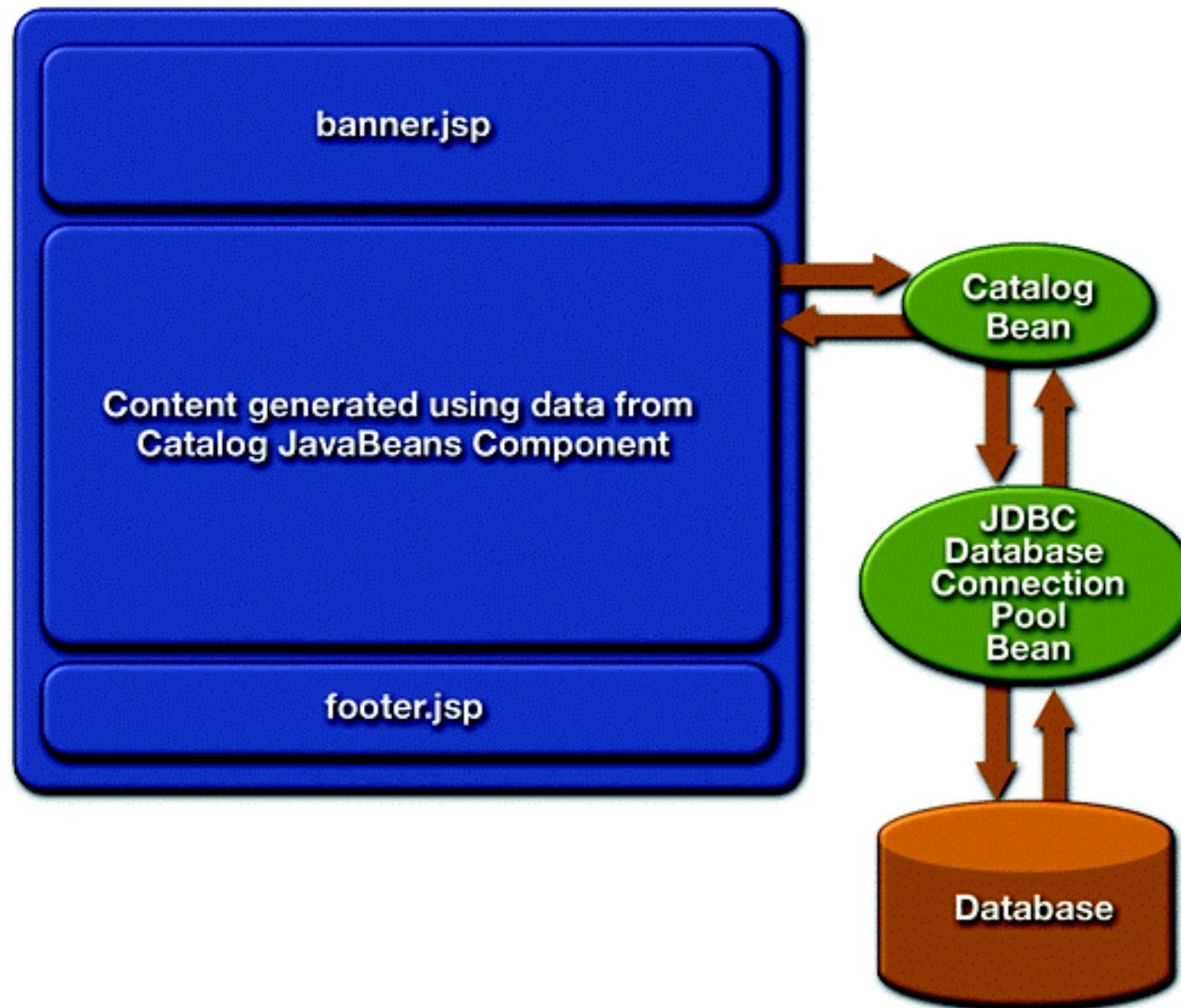
Enterprise Architekturen – Web Tier



Enterprise Architekturen – Web Tier DBMS als Basis für Web Content (*mySite*)

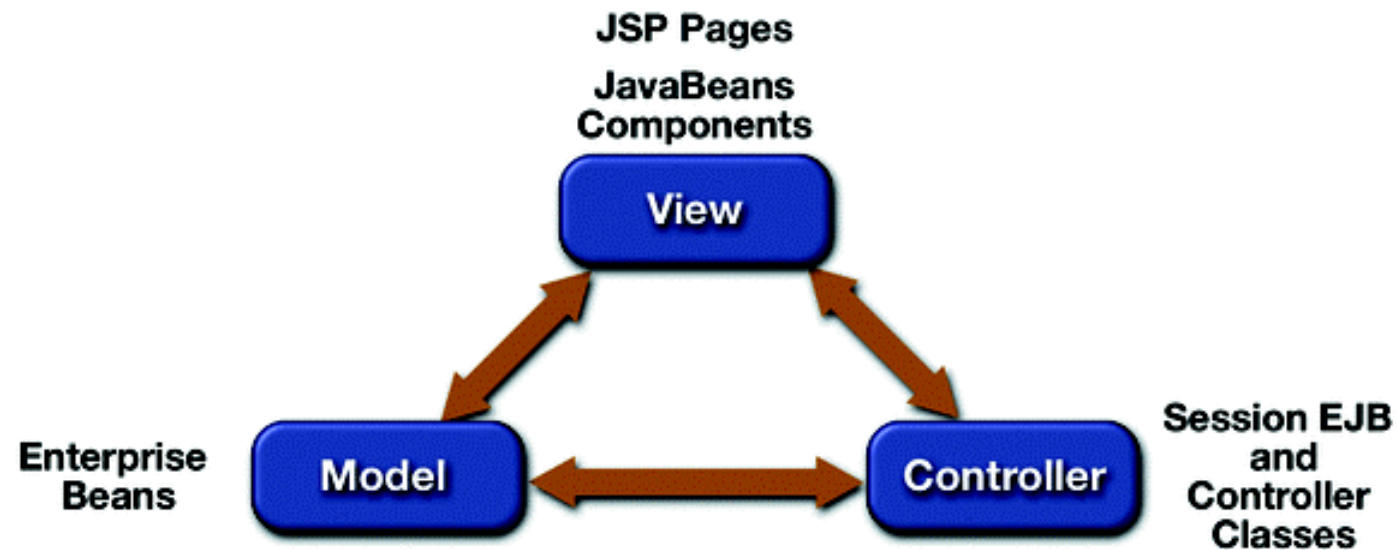


Enterprise Architekturen – Web Tier



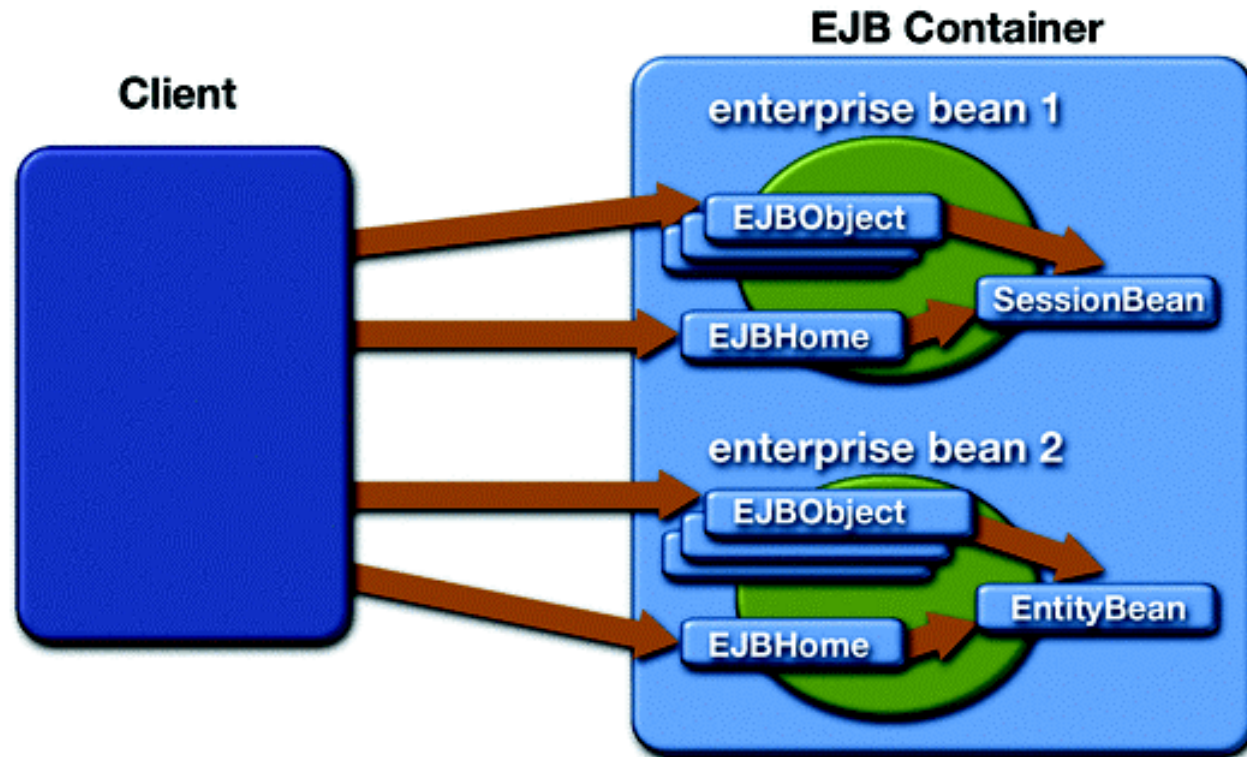
Enterprise Architekturen – Web Tier

MVC in Aktion



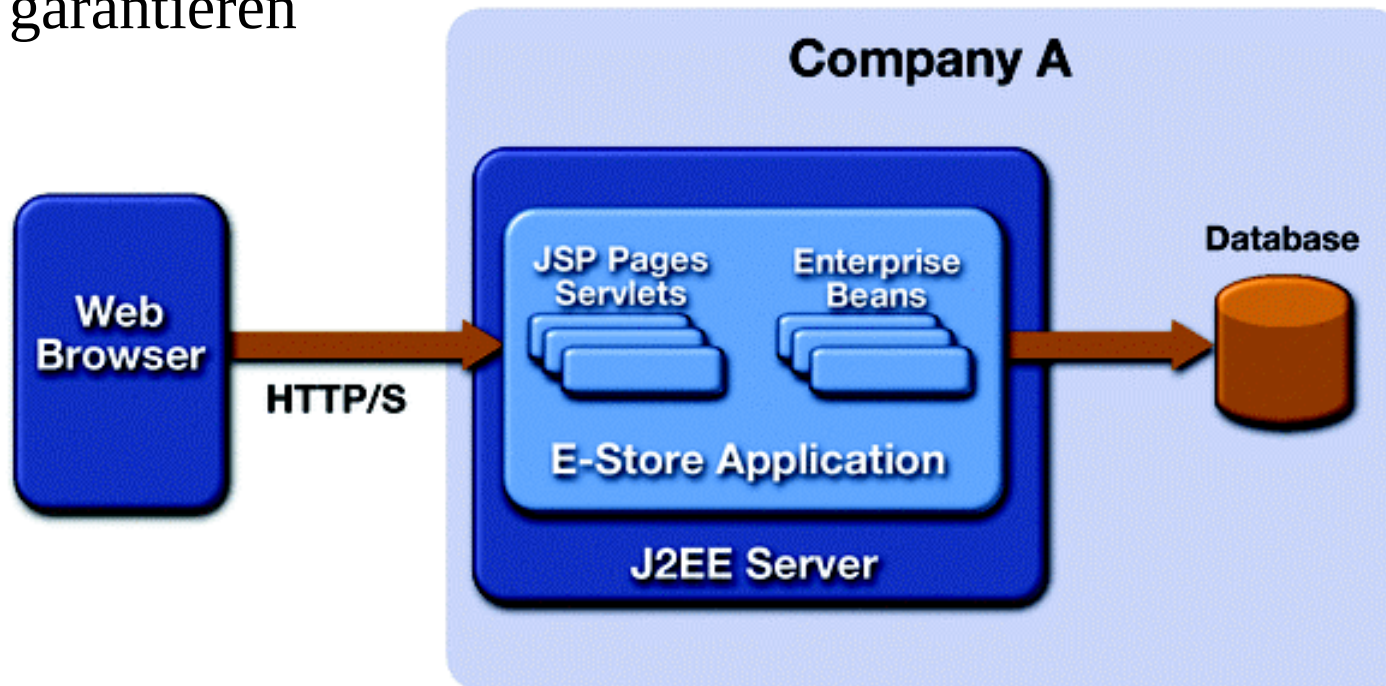
Enterprise Architekturen – Applikations Tier

- Der Applikations Tier bietet unterschiedliche Dienste auf der applikatorischen Ebene an
 - Finanz- und Rechnungswesen ...



Enterprise Architekturen – Enterprise Tier

- Der Enterprise Tier muss die Integration der Web basierten Technologie mit der Legacy Software anbieten / garantieren



Ausblick

Von der Datenbank zur Enterprise Applikation

- Datenbanken – Trends
 - DB Anbieter werden vermehrt in Applikations-Server Geschäft wachsen
 - XML ist ein „universelle“ Informatiksprache mit direkter Auswirkung auf die DBMS Szene
 - XML DBs
 - Ausgabe der Abfragen als XML
 - ...