Java EE Architektur

Simon Martinelli - simas GmbH - www.simas.ch

simas

Ausgangslage

- Die Java Enterprise Edition definiert einen Standard für die Entwicklung komponenten-basierter, verteilter Applikationen. Dazu gehört die Definition einer Referenzarchitektur und die Bereitstellung einer ganzen Palette von Mechanismen (Container, verteilte Kommunikation, Sicherheit, Transaktionen, etc.).

- Als Entwickler in einem konkreten Projekt müssen wir beurteilen können, welche dieser Mechanismen in welcher Form und in welchen Kombinationen eingesetzt werden sollen. Dabei werden unterschiedliche Anforderungen (Projektgrösse, Performance, Verfügbarkeit, etc.) zu unterschiedlichen Lösungen führen.

Architektur ist das Fundament

Erfolgreiche Architektur

Die Cheops Pyramide, ca. 2680 v. Chr

Weniger erfolgreiche Architektur

Der Schiefe Turm von Pisa, 1173

Definition

Unter dem Begriff Softwarearchitektur versteht man eine strukturierte oder hierarchische Anordnung der Systemkomponenten sowie die Beschreibung ihrer Beziehungen.

Balzert H, Lehrbuch der Softwaretechnik,
Spektrum Akademischer Verlag, 2. Auflage, 2001

Anforderungen an ein System

- **Funktionale Anforderungen**
  - Use Cases, Szenarios etc.

- **Nichtfunktionale Anforderungen**
  - Zuverlässigkeit
  - Security
  - Einhaltung von Standards
  - Verfügbarkeit (Beispiele: 7*24, 5*10, 99,99% pro Jahr)
  - Ausfallsicherheit (Beispiel: Fail Over, Stand By)
  - Performance (Beispiel: Antwortzeit <= 0,5 Sec)
  - Skalierbarkeit (Beispiel: Erweiterung von 10 auf 100 parallele Nutzer)

Komponenten

- Eine Komponente ist eine abgeschlossene Einheit einer Software, bestehend aus einer Folge von Verarbeitungsschritten und Datenstrukturen

- Komponenten bieten eine Kapselung (Encapsulation) durch die Trennung von Schnittstelle und Implementierung

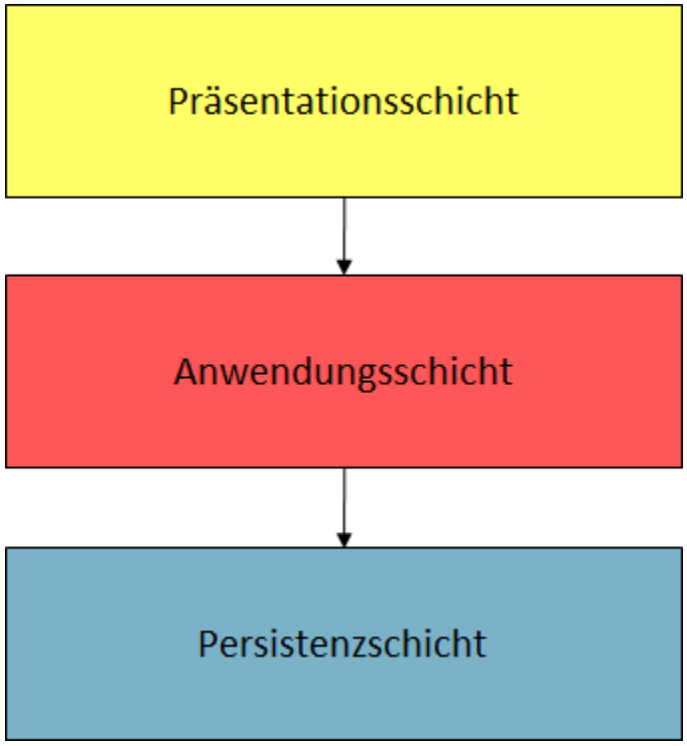
Separation of Concerns

- Separation of concerns aus "On the role of scientific thought” von Edsger W. Dijkstra, 1974
- Trennung der Verantwortlichkeiten
- Häufig verwendet um funktionale von nichtfunktionalen Anforderungen zu trennen
- Der Businesscode soll nicht “verschmutzt” werden
- Kann durch Aspekt Orientierte Programmierung (AOP) erreicht werden

Logische Schichten

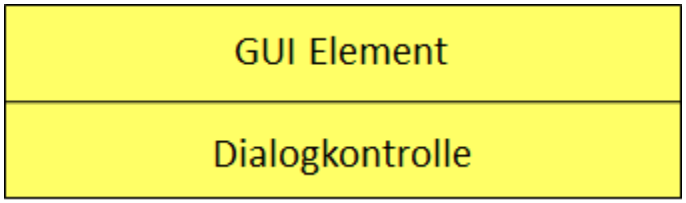
- Vorteile von verteilten Systemen
  - Zentrale Datenhaltung
  - Skalierbarkeit
  - Fehlertoleranz

- Klassische logische Aufteilung in 3 Schichten



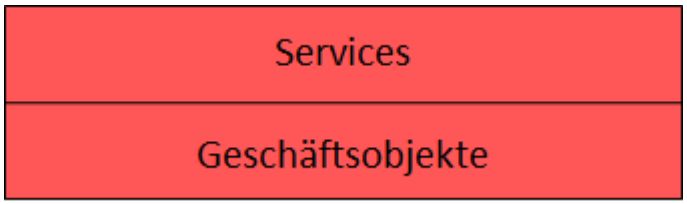
Präsentationsschicht

- Primäre Aufgabe
  - Benutzeroberfläche um Daten darzustellen und auf Benutzereingaben zu reagieren
  - Design Pattern Model View Controller (MVC)
  - Zur Erreichung einer losen Kopplung weiss die Präsentationsschicht möglichst wenig von der Geschäftslogik. Darüber hinaus soll sie – im Sinne einer Schichtenarchitektur – keinerlei Kenntnisse über die Persistenzschicht besitzen



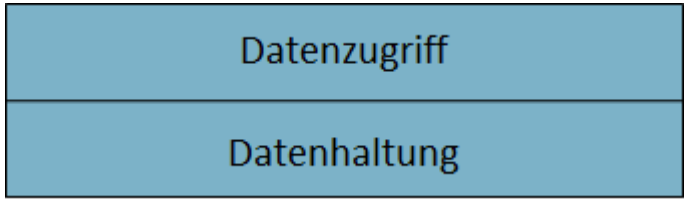
Anwendungsschicht

- Primäre Aufgabe
  - Fachliche Objekt und Logik um Geschäftsprozesse abzubilden
- Weiter Aufteilung möglich in
  - Services
    - Stellt Methoden und Dienste zur Verfügung um die Geschäftsprozesse zu realisieren
    - Erfüllt die Anforderungen der Präsentationsschicht
    - Meist Zustandslos
  - Geschäftsobjekte
    - In der Regel persistent

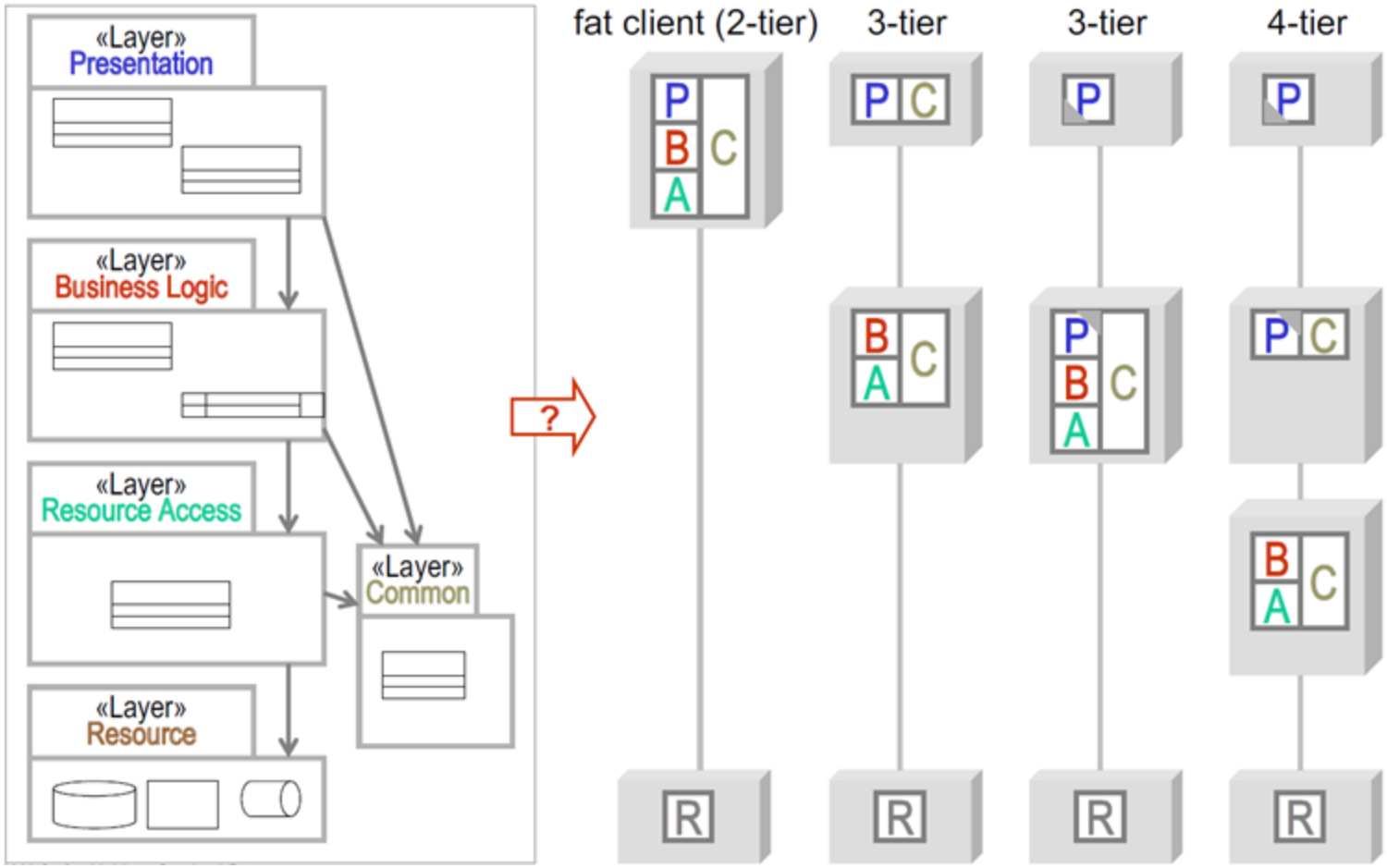


Persistenzschicht

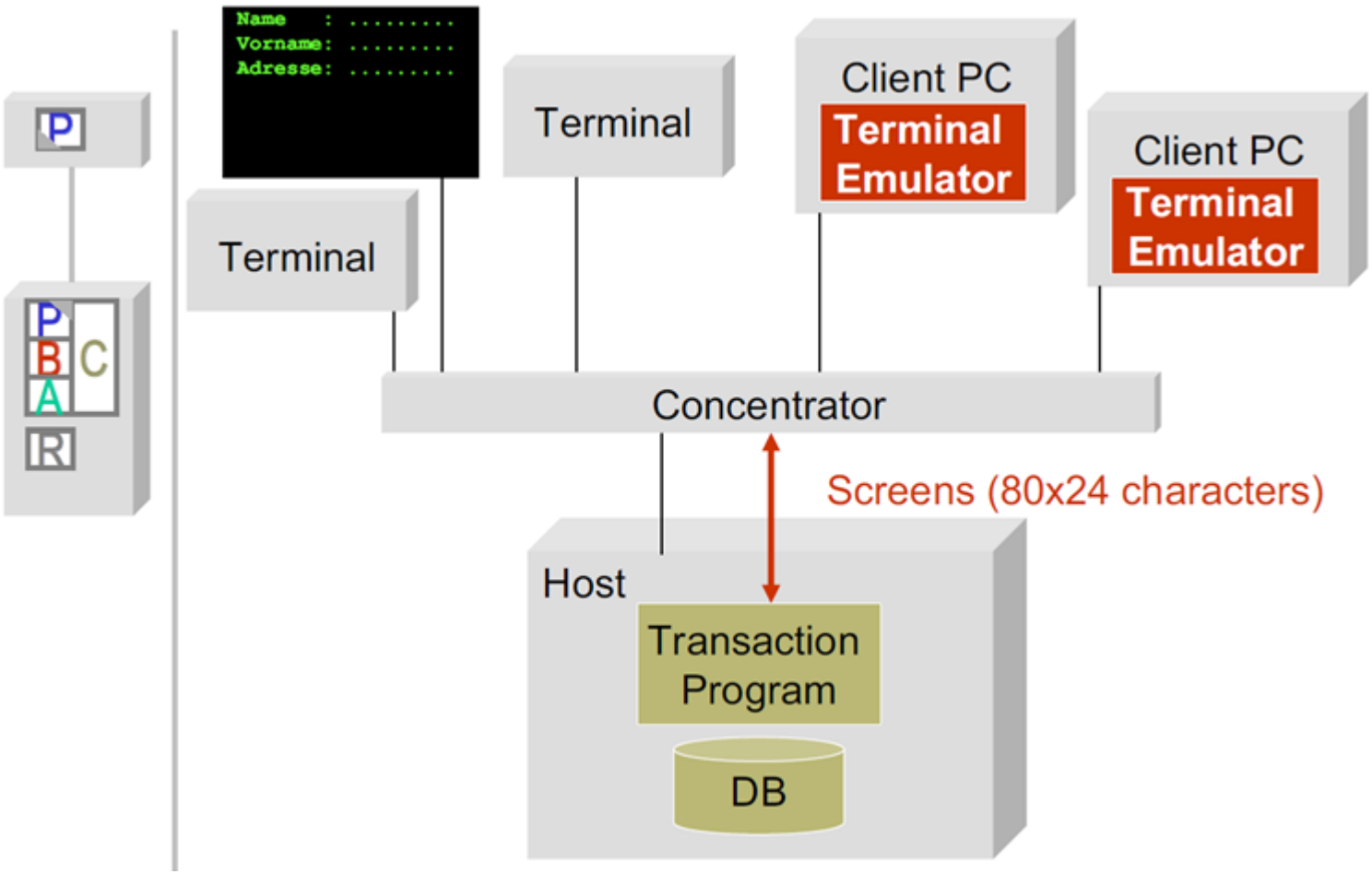
- Primäre Aufgabe
  - Dienste die für die dauerhafte Verwaltung der fachlichen Daten in einem Datenbanksystem sorgen
- Weiter Aufteilung in
  - Datenhaltung
    - Datenbanksystem (meist relational)
  - Datenzugriff
    - O/R Mapping (z.B. JPA)
    - JDBC
    - SQL



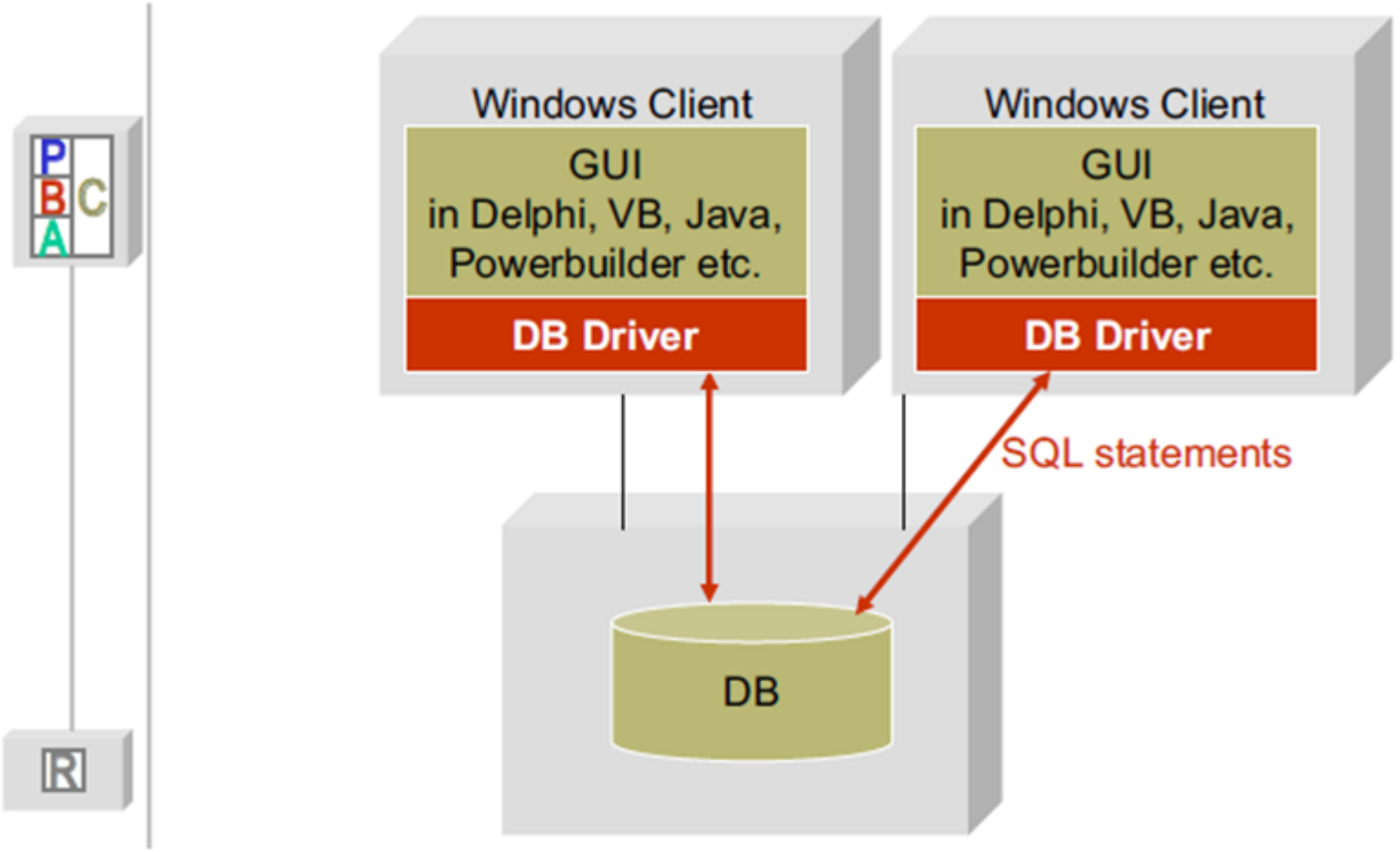
Schichten und Verteilung



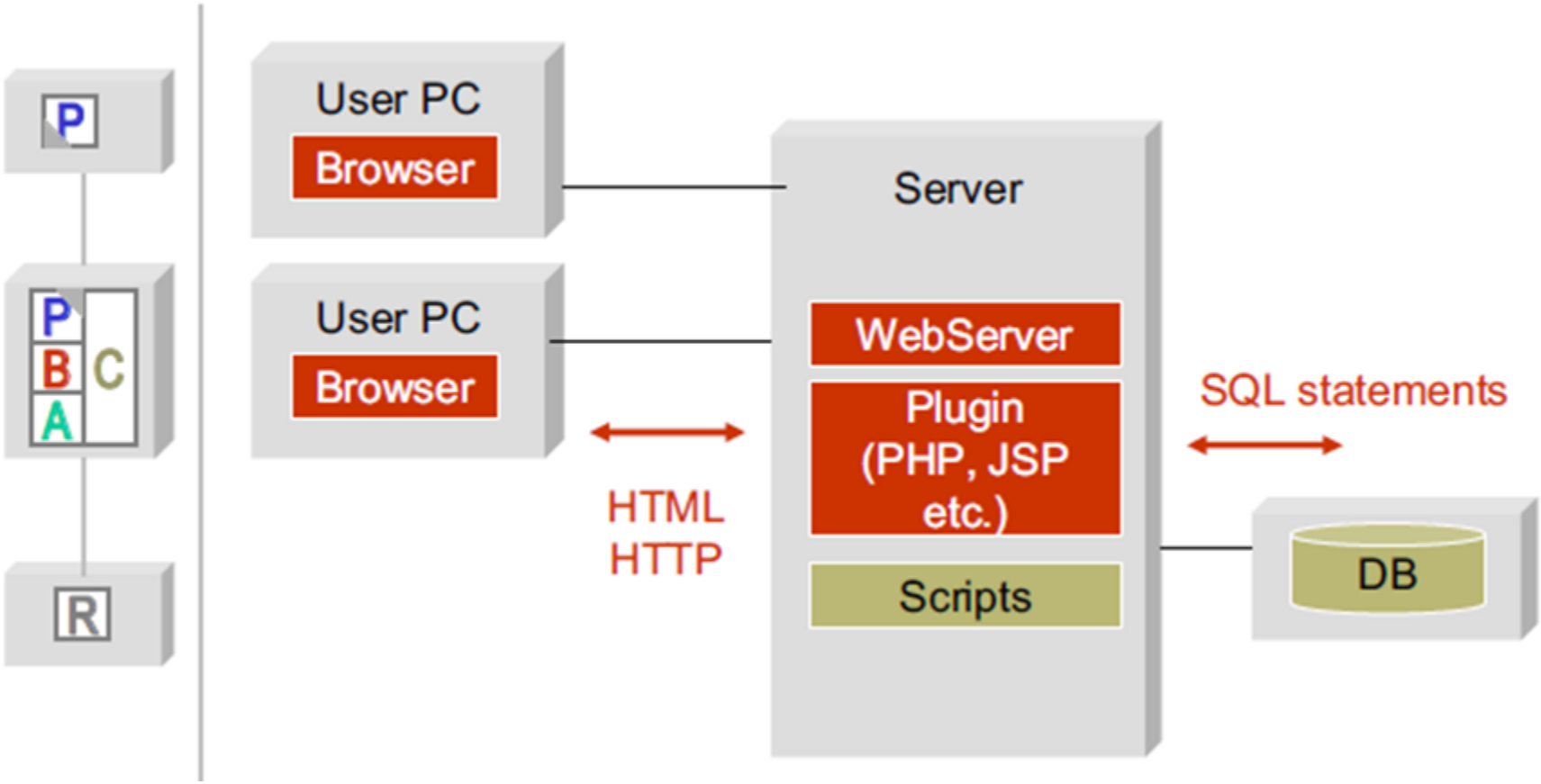
HOST


Name : .....
Vorname: .....
Adresse: .....
Terminal
Terminal
Client PC
Terminal Emulator
Client PC
Terminal Emulator
P
P
B
A
C
R
Concentrator
Screens (80x24 characters)
Host
Transaction Program
DB

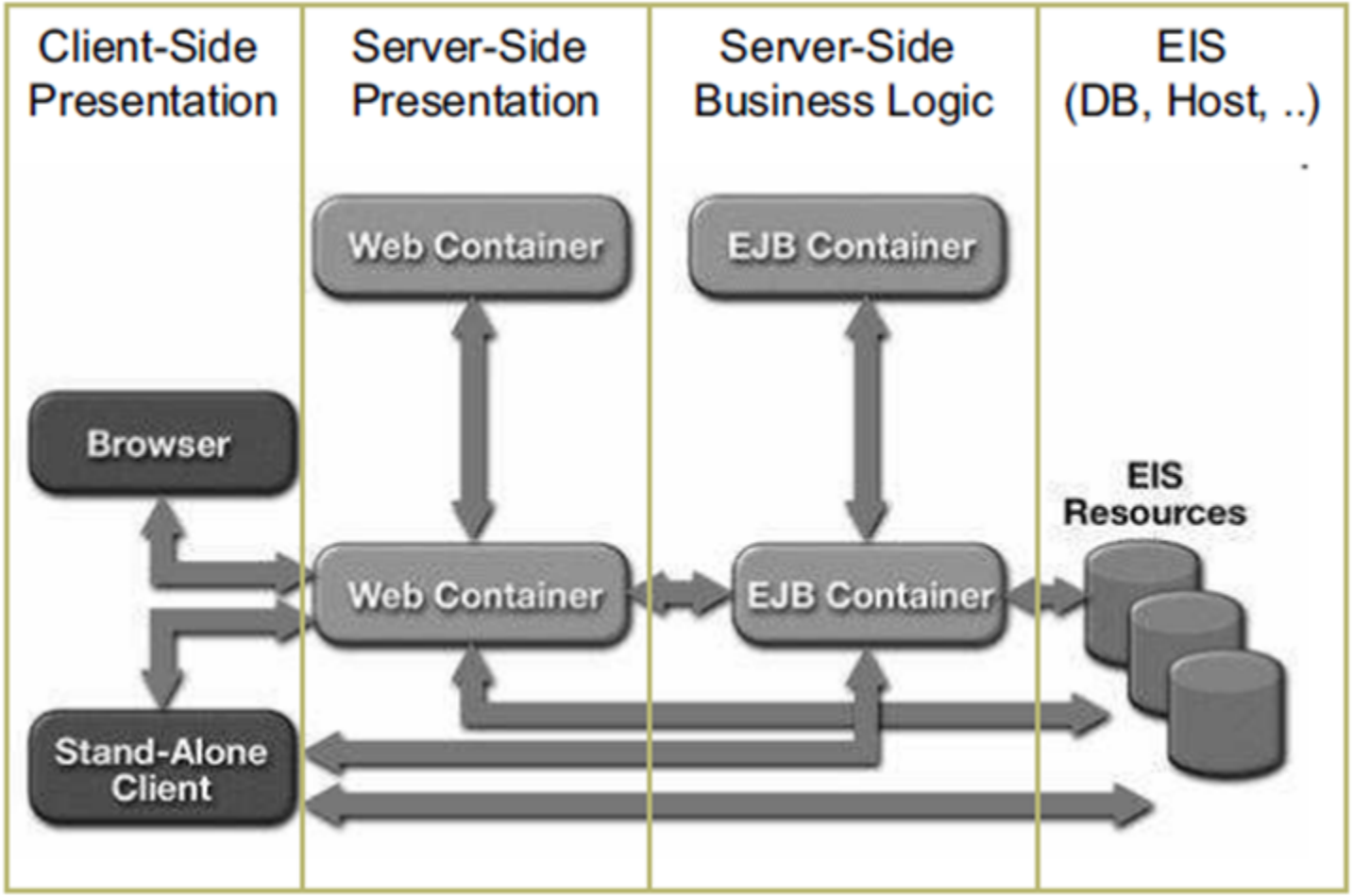
Fat Client


P
B
A
C
R
Windows Client
GUI
in Delphi, VB, Java,
Powerbuilder etc.
DB Driver
Windows Client
GUI
in Delphi, VB, Java,
Powerbuilder etc.
DB Driver
SQL statements
DB

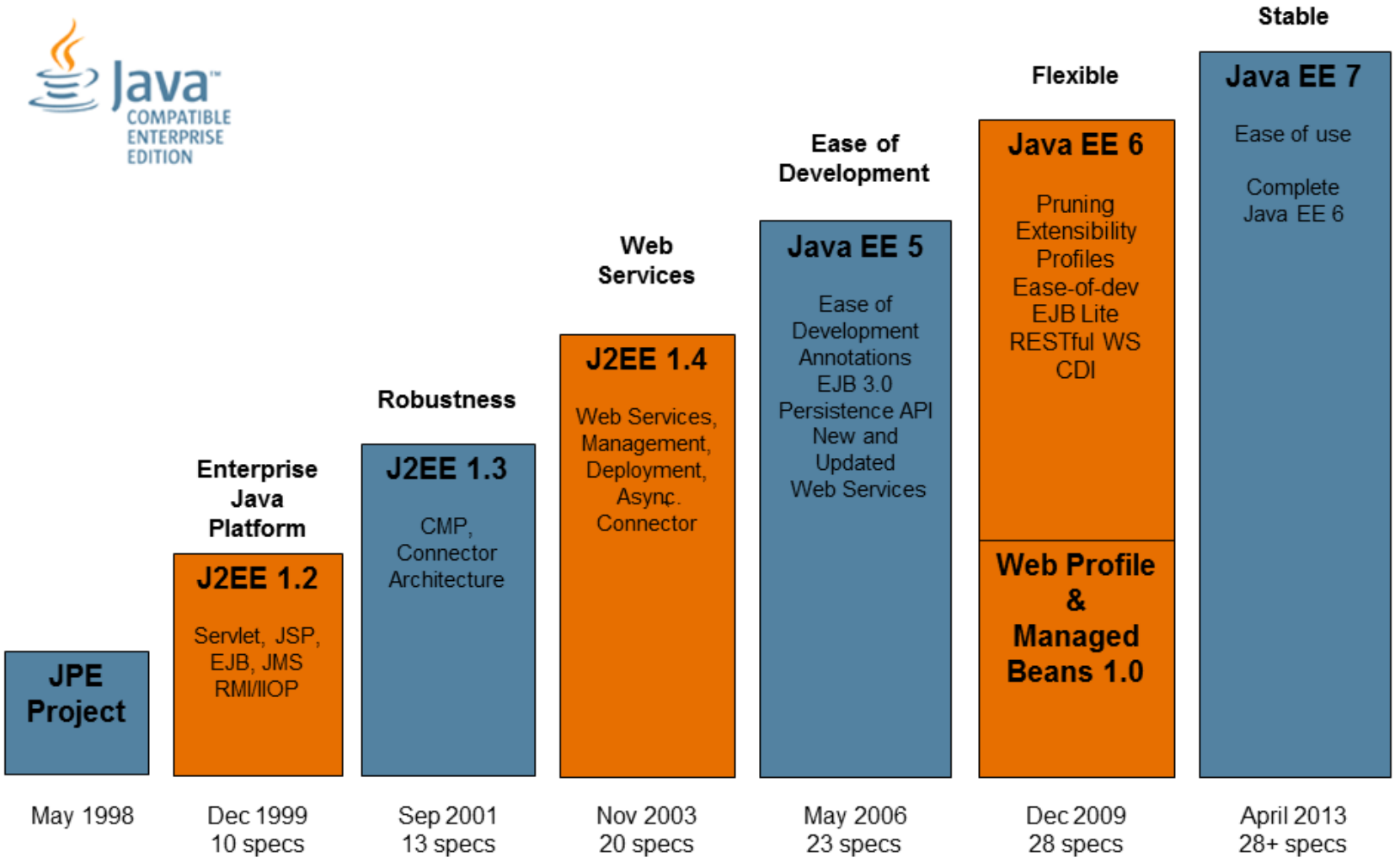
Web


P
P
B
A
C
R
User PC
Browser
User PC
Browser
HTML
HTTP
Server
WebServer
Plugin
(PHP, JSP
etc.)
Scripts
SQL statements
DB

Java EE


Client-Side Presentation
Server-Side Presentation
Server-Side Business Logic
EIS (DB, Host, ..)
Web Container
EJB Container
Browser
Web Container
EJB Container
EIS Resources
Stand-Alone Client

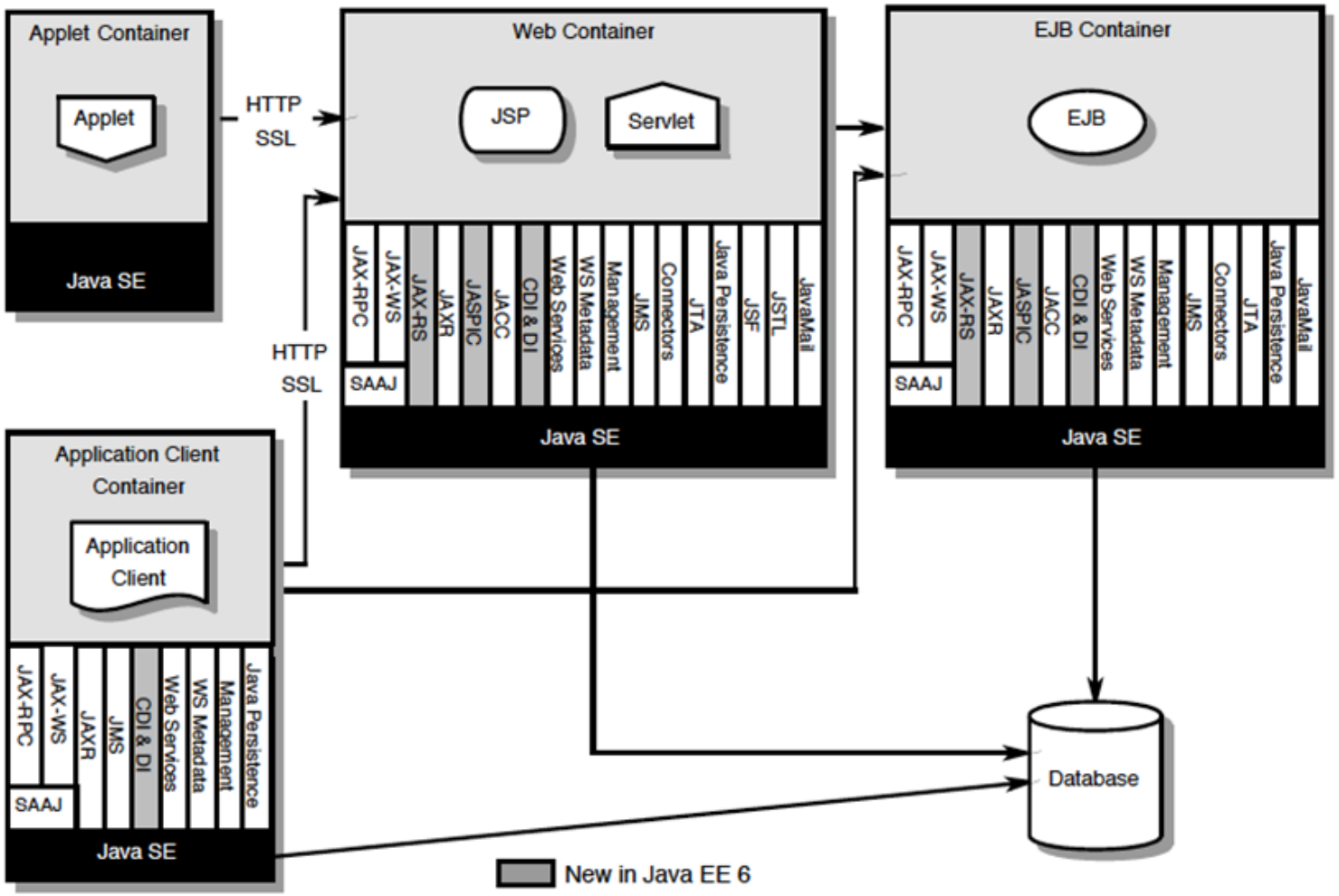
Java EE Geschichte



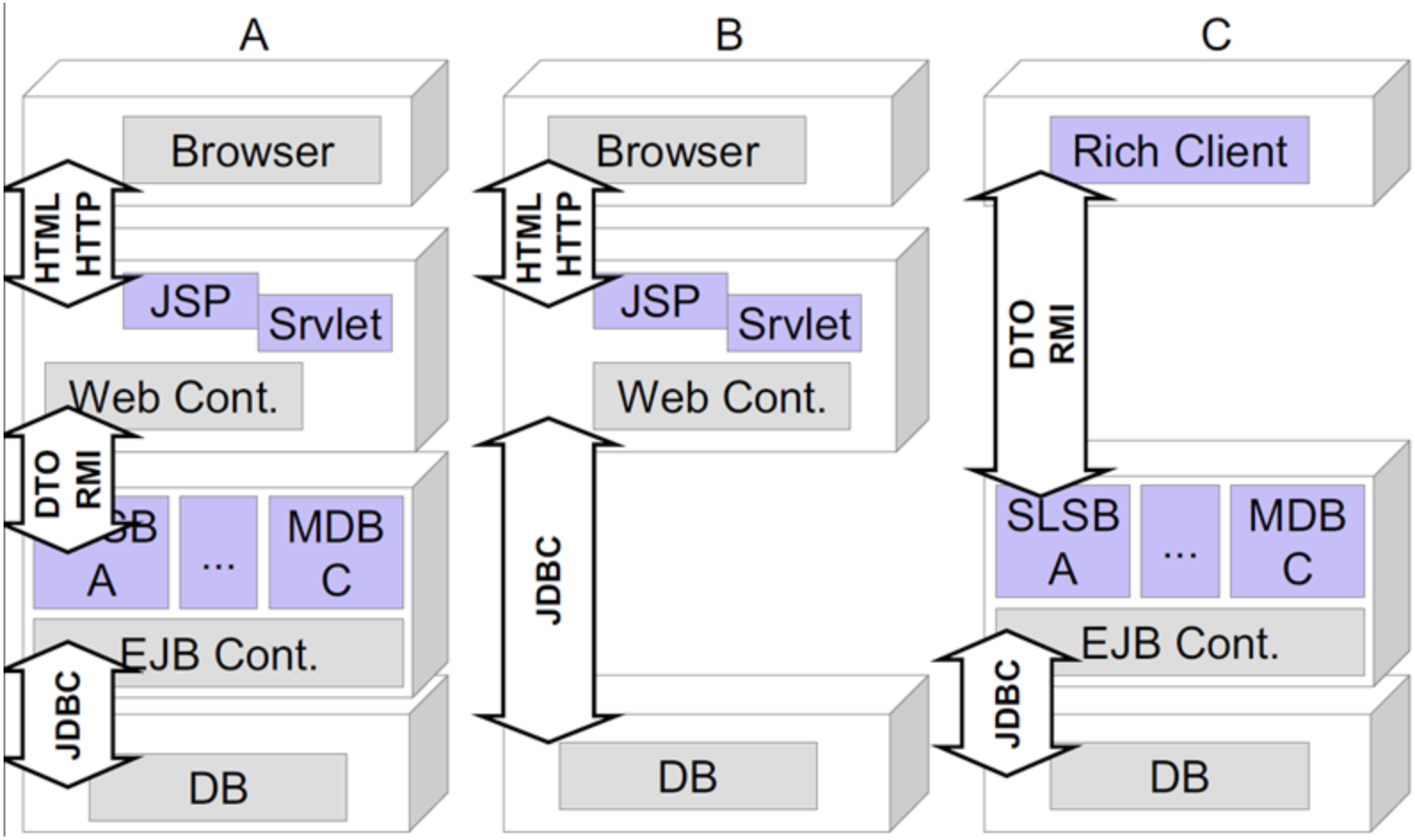
Neue Spezifikationen in Java EE 6

- Bean Validation 1.0 JSR 303
- Java API for RESTful Web Services JSR 311
- Dependency Injection for Java JSR 330
- Context and Dependency Injection for Java EE JSR 299

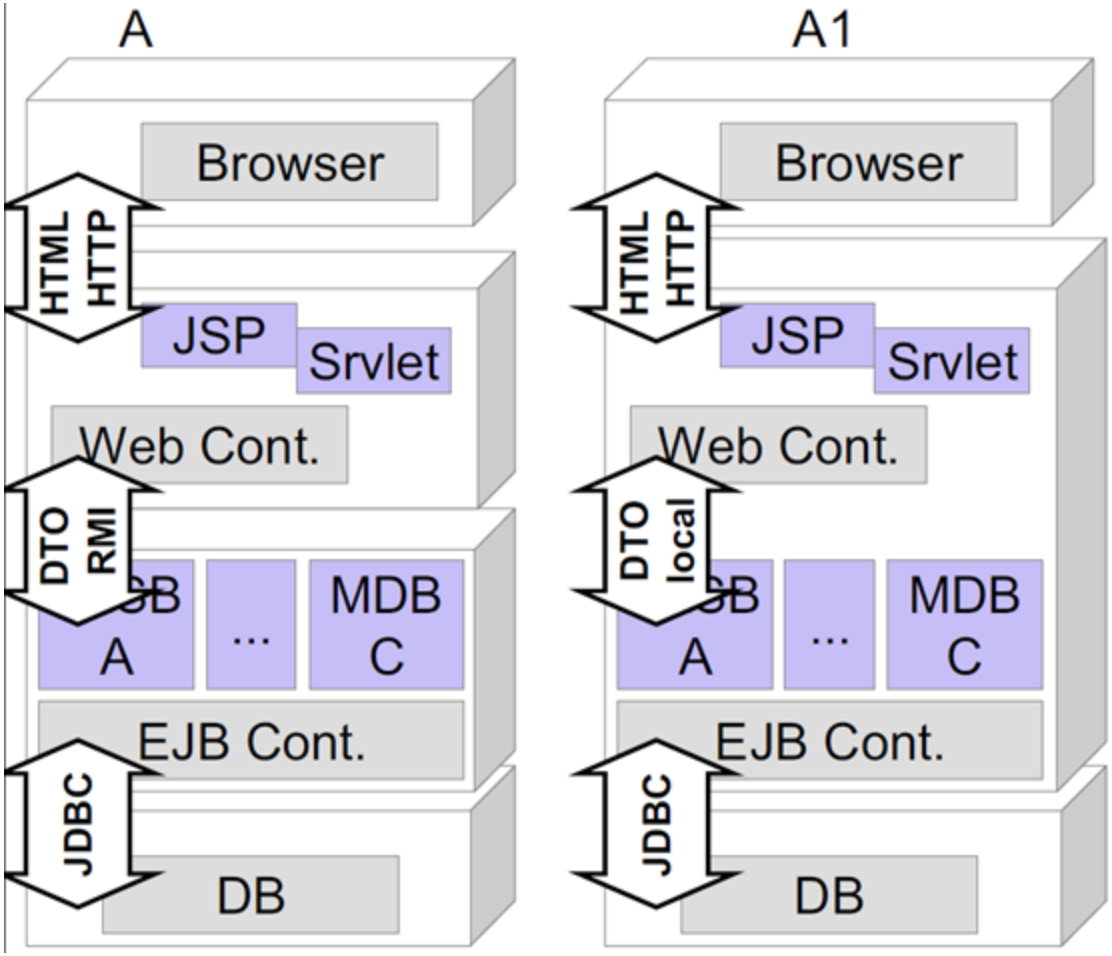
Die Containers



Laufzeit Szenarios



Variante zu A



Varianten zu C