Java EE Pattern

Simon Martinelli - simas GmbH - www.simas.ch

simas

Was sind Pattern?

- Pattern beschreiben allgemeine Lösungen zu häufig wiederkehrenden Problemen
- Ermögliche Wiederverwendung auf konzeptueller Ebene
- Stammen aus der Gebäudearchitektur

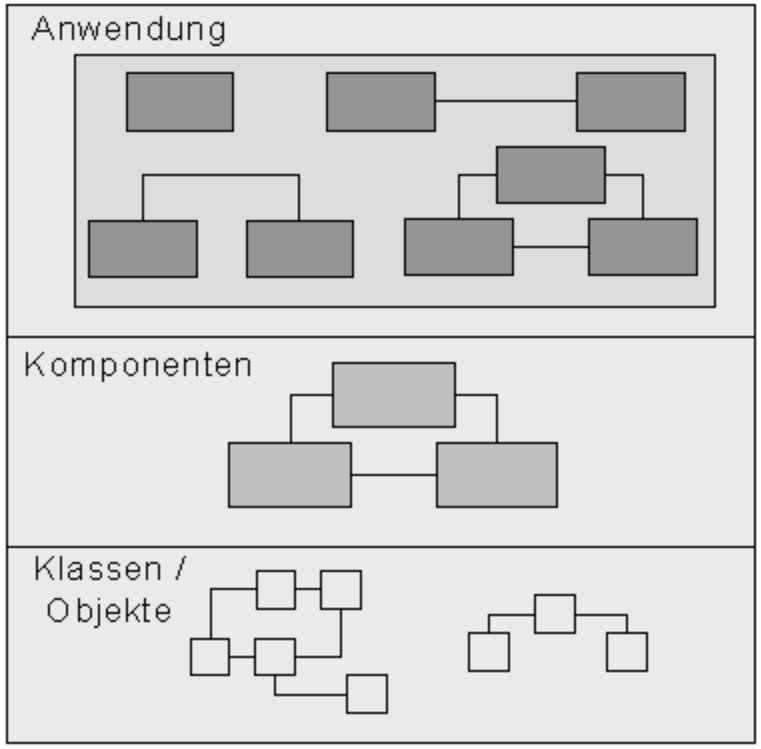
Geschichte der Pattern

- 1987: Ward Cunningham und Kent Beck schreiben die ersten fünf Software-Pattern und stellen sie auf der OOPSLA 87 vor
- 1990: Erich Gamma stösst auf Pattern im Rahmen seiner Dissertation
- 1991: Erich Gamma und Richard Helm treffen sich auf Bruce Andersons Workshop „Toward an Architecture Handbook“ auf der OOPSLA 91
- 1992: Ralph Johnson und John Vlissides stossen hinzu und bilden mit Erich Gamma und Richard Helm die „Viererbande“ (GoF, Gang of Four)
- 1994: Die erste Ausgabe von „Design Patterns“ erscheint und wird schnell zum Bestseller
- 1996: „Pattern-Oriented Software Architecture“ von Buschmann et. al. erscheint.
- 1999: Patterns werden zum „Hype“
  Neben Design- und Architektur-Pattern finden sich unter anderem Analyse-Pattern, Prozess-Pattern und viele andere

Charakteristiken von Pattern

- Werden durch Beobachtung identifiziert, nicht neu entwickelt
- Werden in einem vorgegebenen strukturierten Format aufgeschrieben
- Existieren auf verschiedenen Abstraktionsebenen
- Werden in der Regel in Kombination eingesetzt werden
- Sie stammen immer aus realen Projekten, werden abstrahiert und aufbereitet

Ebenen zum Einsatz von Pattern



Musterarchitekturen ← **Architektur Pattern**

Architekturmuster ← **J2EE-Patterns**

Entwurfsmuster ← **GoF-Patterns**

GoF Pattern

- Drei Musterkategorien
  - **Erzeugungsmuster** verstecken den Prozess der Instanziierung von Objekten vor der Anwendung. Beispiel: Abstract Factory
  - **Strukturmuster** beschreiben die Komposition von Klassen oder Objekten zu grösseren zusammenhängenden Gebilden. Beschrieben werden die beteiligten Klassen oder Objekte sowie die strukturellen Abhängigkeiten. Beispiel: Proxy
  - **Verhaltensmuster** beschreiben die Art und Weise der Zusammenarbeit zwischen Klassen und Objekten. Beispiel: Observer

- Unterscheidung zwischen klassen- und objektbasierten Mustern
  - **Klassenbasierte** Muster arbeiten mit Vererbung und Schnittstellenimplementierung
  - **Objektbasierte** Muster konzentrieren sich auf die Komposition von Objekten und ihre Beziehungen

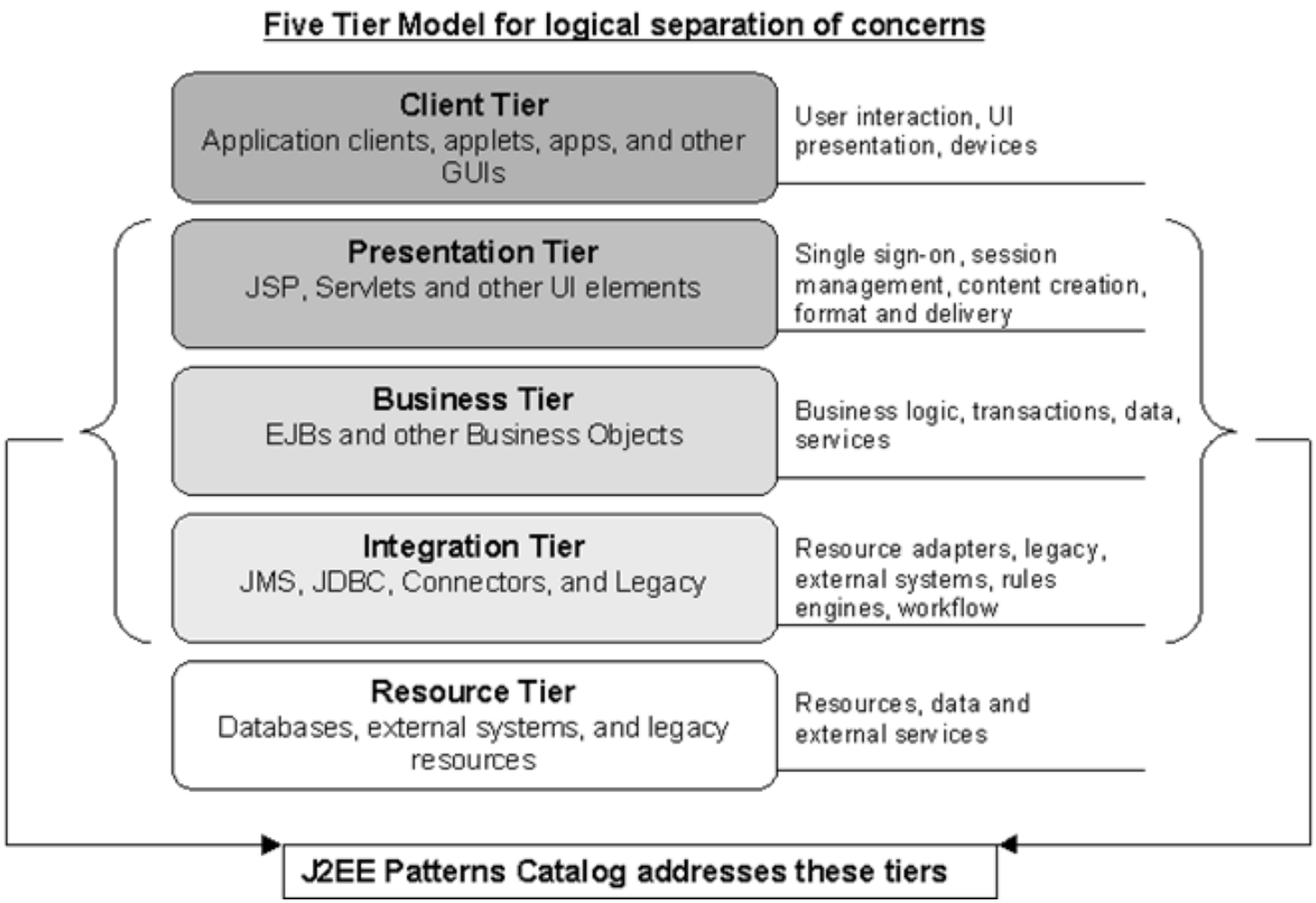
Beschreibung von Pattern

- Problem
  - Kurze Darstellung des Problems, das durch das Pattern gelöst werden kann
  - Ausführliche Beschreibung der Problematik
- Kräfte
  - Beschreibt Voraussetzungen, bzw. Fragestellungen wann das Pattern eingesetzt werden sollte
  - Beschreibt den Kontext in dem der Einsatz des Pattern Sinn macht
- Lösung
  - Liefert eine Zusammenfassung, sowie eine ausführliche Beschreibung der Lösung
  - Beschreibt die Elemente, ihre Rollen, ihre Beziehungen, die Verantwortlichkeiten und die Interaktionen
  - Beispielimplementierung
- Konsequenzen
  - Beschreibt positive und negative Konsequenzen bei Einsatz des Pattern
  - Was wird erreicht durch den Einsatz des Pattern?
  - Was kann nicht erreicht, gelöst werden?

J2EE Pattern

- Sind von der Granularität her zwischen Design- und Architektur Pattern anzusiedeln
- Wurden von Sun auf Basis der ersten Erfahrungen mit dem J2EE Standard identifiziert und ausgearbeitet
- Werden nach den Schicht klassifiziert
  - Presentation
  - Business
  - Integration
- Decken insbesondere das Design von Web Applikationen auf der Basis von Servlets und EJB ab
- ~20 J2EE Pattern
- **Viele wurden durch Java EE obsolet**

Der "Tier-Stack" von Sun



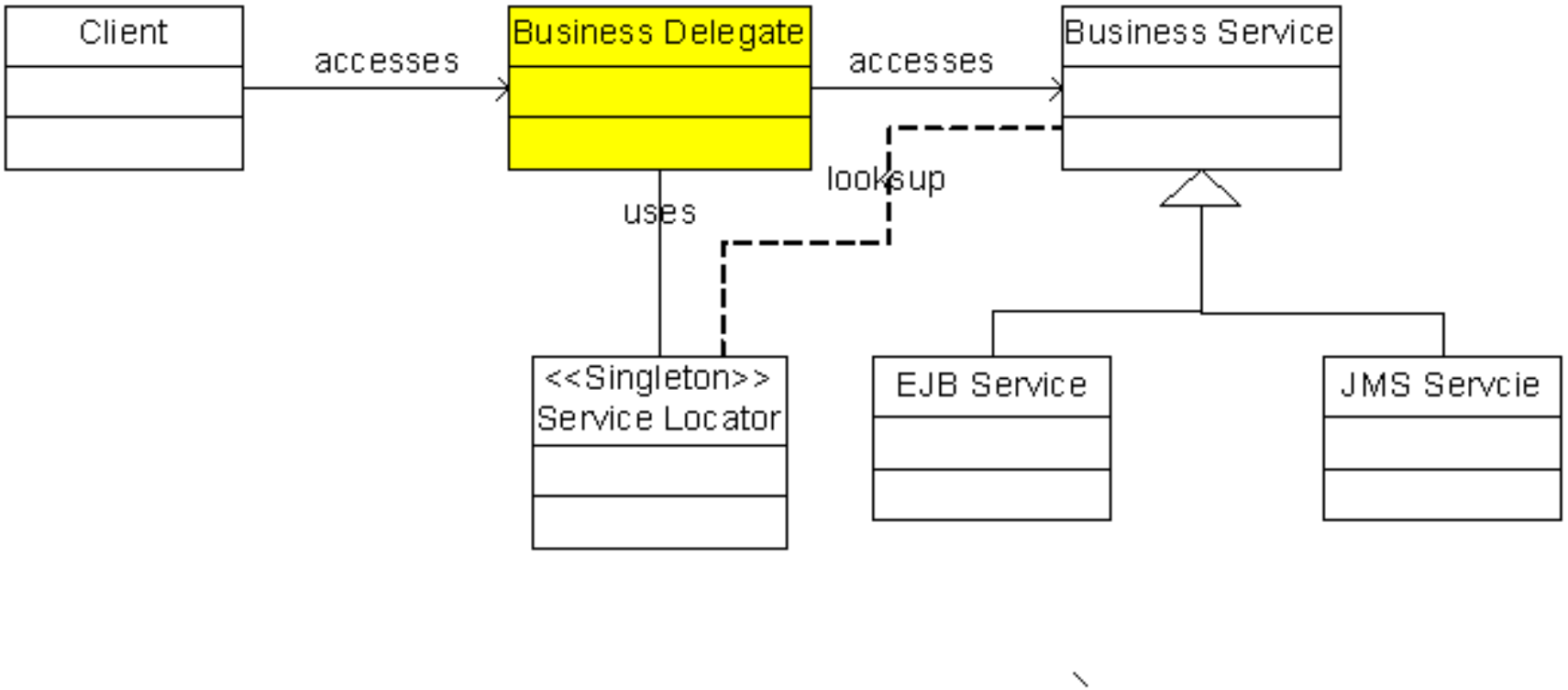
Presentation Tier

- Annehmen und Überwachen der Clientzugriffe
  - Intercepting Filter, Front Controller
- Aufbereiten und Speicherung der eingegebenen Daten
  - Context Object
- Lokalisierung von Business Komponenten und Dialogen
  - Application Controller, Dispatcher View
- Verwaltung komplexer Dialoge
  - Composite View
- Trennung von reiner Präsentation und Darstellungslogik
  - View Helper

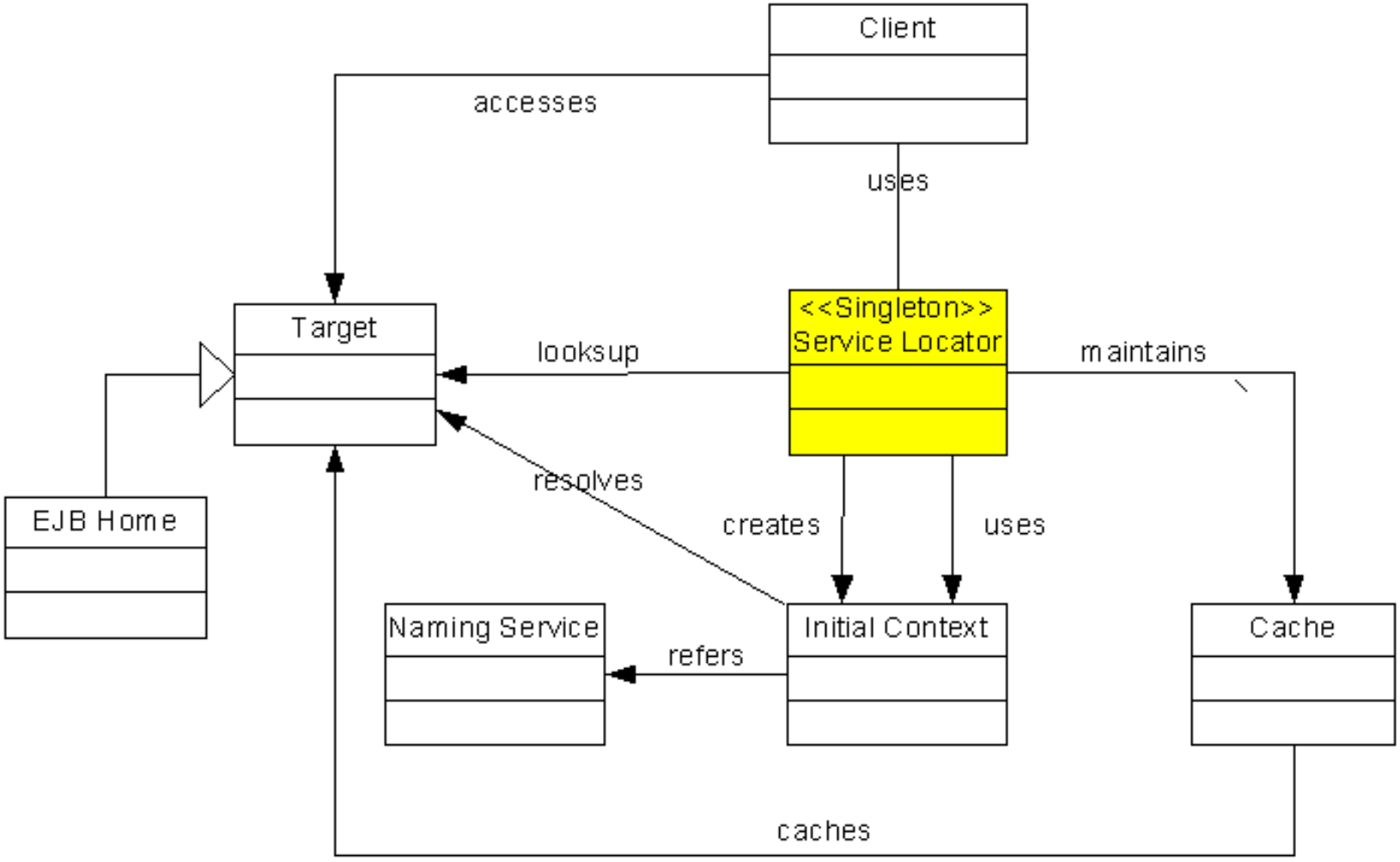
Business Tier

- Transparenter Zugriff vom Client auf den Server
  - Business Delegate
- Auffinden von Services am Server
  - Service Locator
- Serviceorientierte Schnittstelle, Verbergen der Datenschicht
  - Session Facade
- Verwaltung der Datenschicht
  - Business Object
- Transport von Daten, Datenverwaltung, Minimierung von Zugriffen
  - Transfer Object

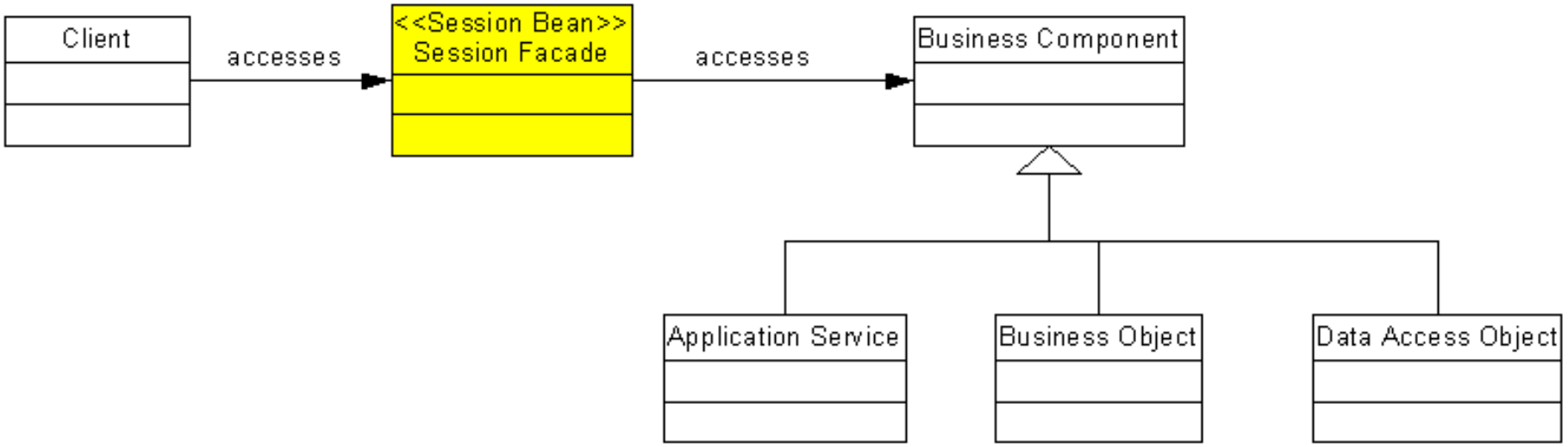
Business Delegate



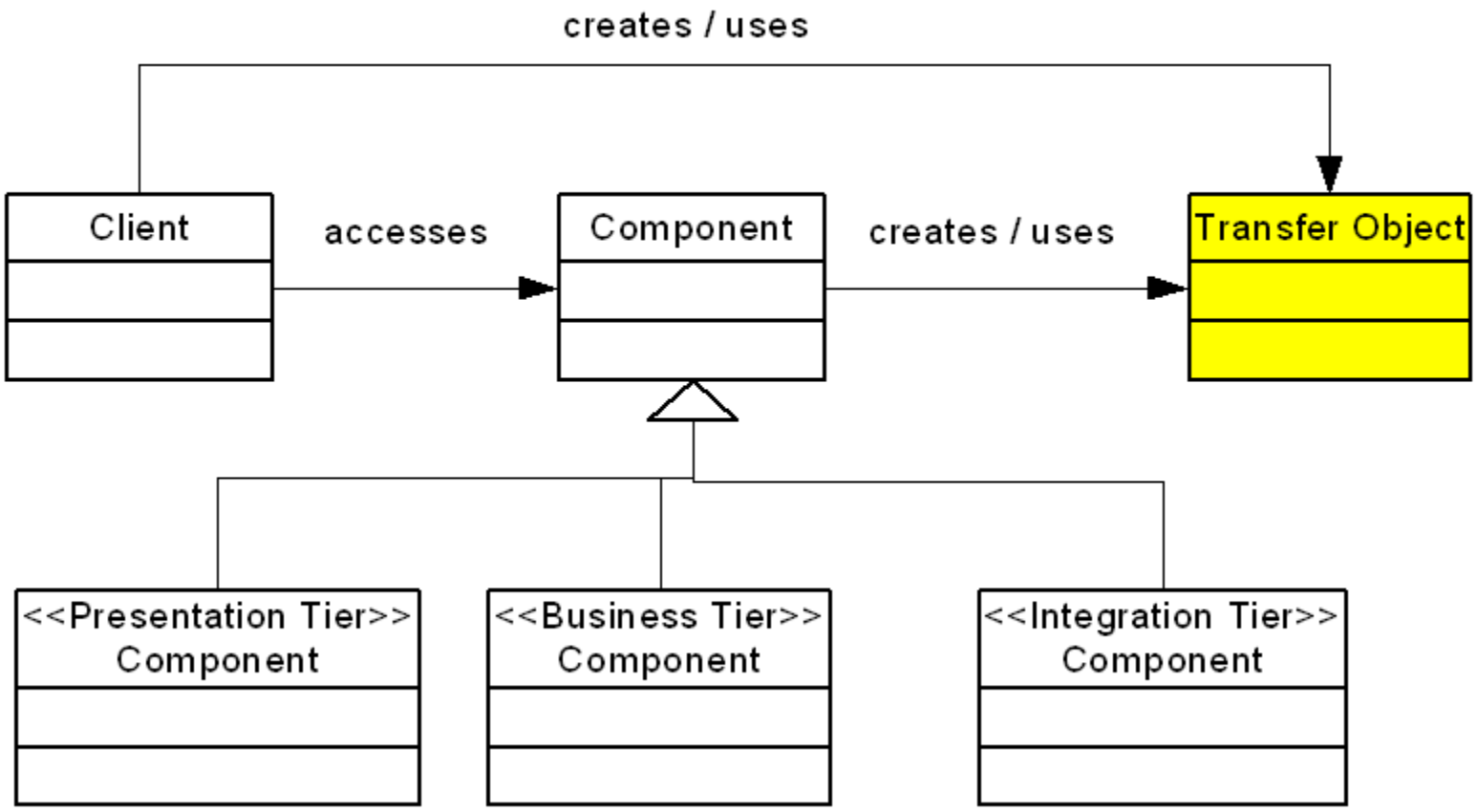
Service Locator



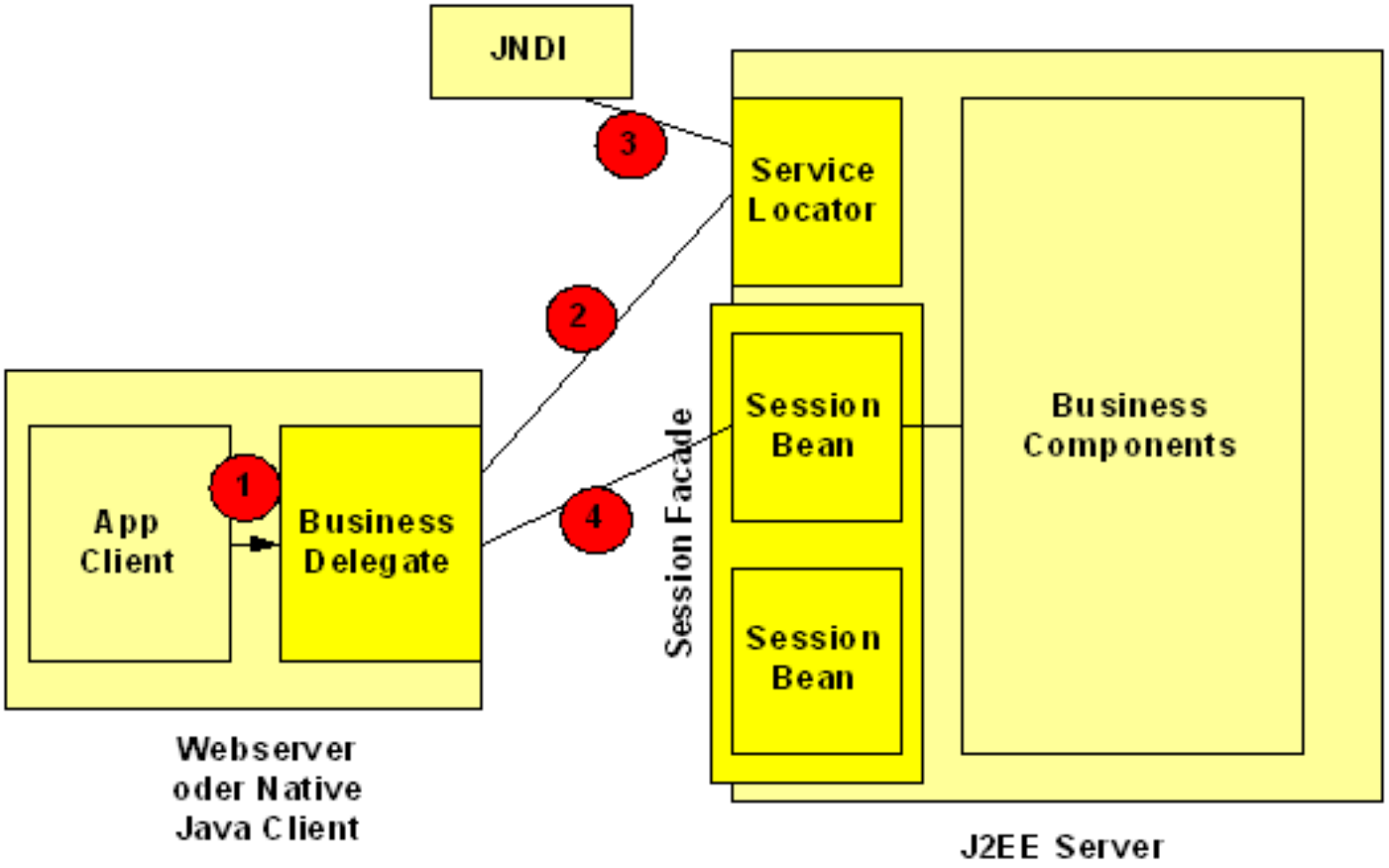
Session Facade



Transfer Object



Kombination



Integration Tier

- Datenbankzugriffe
  - Data Access Object
- Asynchrone Kommunikation
  - Service Activator
- Zusammenfassung von Business Services zu Web Services
  - Web Service Broker

Data Access Object