

TRIONA – INFORMATION UND TECHNOLOGIE GMBH

Spring - ein Überblick

Holger Klatt, 19. Oktober 2007

SPRING 2.0

INHALT

- Überblick
- Grundprinzipien (IoC, DI)
- Patterns (Service, DAO, DTO)
- Spring Module
- Anwendungsfall



J2EE „GESTERN“

- J2EE 1.x bis 1.4
 - „Klassische“ J2EE Architektur (remote EJB und Entity Beans)
 - Lokale EJB Architektur (local EJBs)
 - Ad hoc J2EE Architektur (ohne EJB, „lightweight“)





JEE „HEUTE“

- Java EE 5 (inkl. EJB 3, JSF 1.2)
 - neue, vereinfachte EJB API, die eine einfachere Entwicklung fördert
 - neue API für *persistence management* und objekt-relacionales mapping
 - „Standard“ framework
- Spring 2.0.x
 - Leichtgewichtiger Container
 - Inversion of Control (*Umkehr der Abhängigkeiten*)
 - Dependency Injection



SPRING 2.0 LEICHTGEWICHTIGER CONTAINER



- Keine bis minimale Abhängigkeit zur Container API
- Basiert auf Plain Old Java Objects
- Schnelle Entwicklungszyklen
 - Schnelles Starten/Herunterfahren
 - Keine zusätzlichen Deployment-Schritte
 - Integration in andere Umgebungen

SPRING 2.0

INVERSION OF CONTROL (IOC)

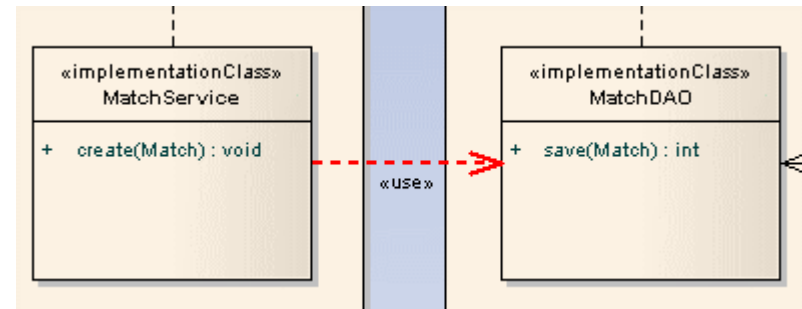


- „Don't call us, we'll call you“ oder das Hollywood-Prinzip (Keine **faulen Abhängigkeiten**: Highlevel-Komponenten, die von Lowlevel-Komponenten abhängig sind, die von Highlevel-Komponenten abhängig sind, die von Geschwister-Komponenten abhängig sind)
 - Entkopplung unterschiedlicher Abstraktionsebenen
 - Lowlevel-Komponenten werden eingeklinkt, ohne dass Highlevel-Komponenten von ihnen abhängig sind
- Basiert auf Plain Old Java Objects
- Schnelle Entwicklungszyklen möglich



SPRING 2.0 DEPENDENCY INJECTION

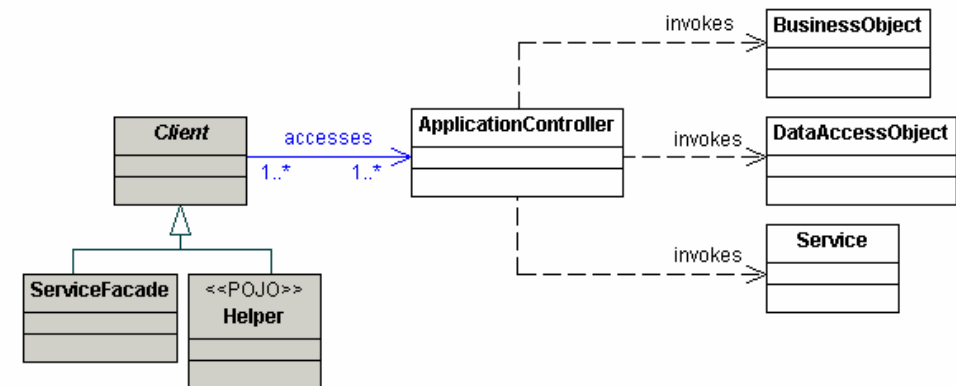
- Angewandte Version der IoC
 - Schwerpunkt: Erzeugung und Initialisierung von Objekten
- Abhängige Objekte werden zur Laufzeit injiziert
 - Besonders flexibel: Verwendung von abhängigen Interfaces
 - Mocks oder andere Implementierungen verwendbar



SPRING 2.0 ARCHITEKTUR PATTERN: SERVICE

Service Layer /Schicht

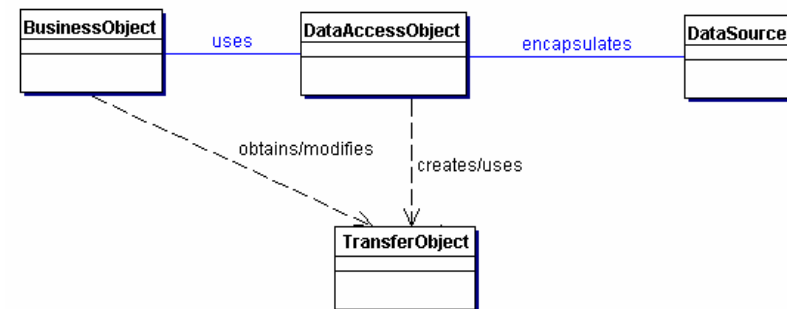
Fowler: „...provides an API that's easier to use because it's typically oriented around use cases. It also makes a convenient point for adding transactional wrappers and security checks.“



- Liegt unterhalb der Präsentationsschicht und oberhalb der Domainschicht
- Abbildung von Applikationslogik (Nachrichten, Geschäftstransaktionen, Anbindung Persistenz, Security Aspekte), Dekorierung von Facade-Objekten
- Auch SessionBeans sind Anwendung des Service Layers

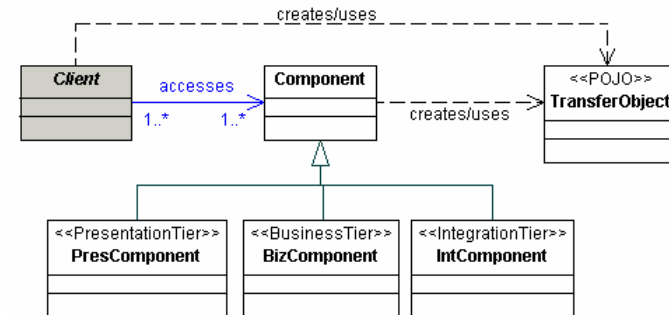
SPRING 2.0

DESIGN PATTERN: DAO



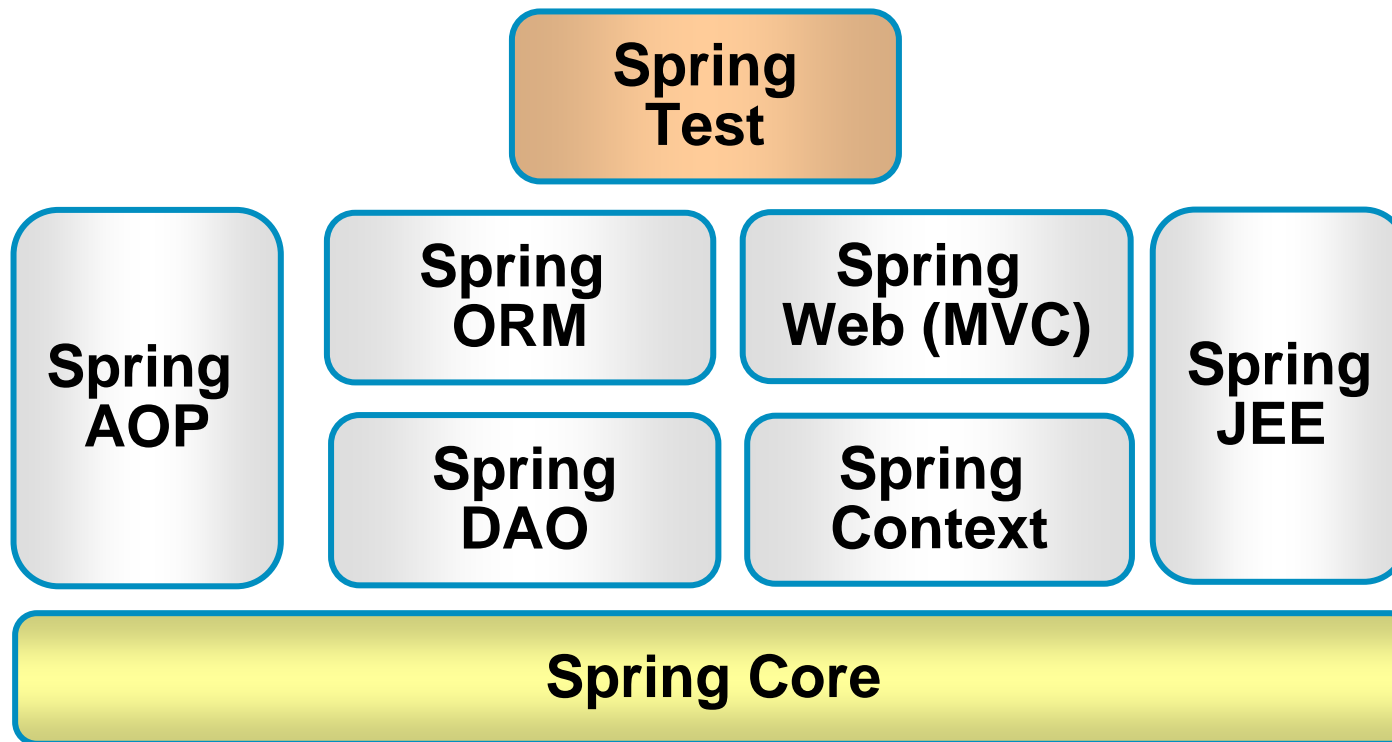
- DAO = **D**ata **A**ccess **O**bject
- Ziel: Persistenzlogik (Datenbankzugriff über JDBC, Hibernate, JDO, eine Datei usw.) von der eigentlichen Programmlogik trennen
- Datenquelle kann unabhängig von der Logik geändert werden (Webservice, Datenbank, Datei, ...)
- Die Businesslogik benutzt das DAO zum Anfordern und Abspeichern von Daten
- Ein Transferobjekt (DTO) kann zur Datenkapselung verwendet werden

SPRING 2.0 DESIGN PATTERN: DTO



- DTO = **D**ata **T**ransfer **O**bject (Business Tier)
- Ziel: Mannigfaltige Datenelemente durch unterschiedliche Applikationsschichten führen („carry multiple data elements across a tier“)
- Führt zur Reduzierung von Aufrufen entfernter Objekte in anderen Schichten
- Datenobjekte werden entsprechend der Anforderung einer Clientschicht mit vom Domain Object abweichender Struktur neu zusammengebaut bzw. generiert
- Aber: Erhöht Komplexität der Anwendung wegen ggf. notwendiger Synchronisierung und Versionskontrolle

SPRING 2.0 MODULE



SPRING 2.0

SPRING CORE & CONTEXT

- Der Kern von Spring stellt in Form eines Containers die grundlegende Funktionalitäten bereit, die maßgeblich die Konfiguration einer Anwendung vereinfachen:
 - Inversion of Control
 - Dependency Injection
 - Application Context („Verdrahten“ der Komponenten)
- Durch Nutzung dieser Elemente könnten Programmbausteine in Enterprise-Anwendungen ohne explizite Abhängigkeiten miteinander gekoppelt werden. Insbesondere erlaubt dies eine konsequente Verwendung von Schnittstellen.

SPRING 2.0

SPRING JEE

- Sammelt die verschiedenen Dienste zur Integration und Kommunikation, die in JEE definiert werden, u.a.
 - JMX (Management Extensions), JMS, JCA (Connector Architecture)
 - Remoting (Web Services, RMI, Hessian, Burlap)
 - Enterprise JavaBeans (EJBs)
 - EMail (JavaMail)
- Hauptanliegen ist, die Konfiguration dieser Dienste Container-unabhängig und in möglichst einfacher Struktur anzubieten

SPRING 2.0

SPRING WEB

- Spring bietet mit MVC eine Eigenimplementierung des MVC2-Konzeptes
- das Ausgabeformat ist komplett frei wählbar - egal ob Excel, WML, PDF oder HTML
- Anbindung an alle Standardpräsentationsschichten mit allen Vorteilen von Spring
- weitere Webframeworks werden unterstützt: JSF, Struts, JSP, Velocity, Tapestry, Jasper Reports

SPRING 2.0

SPRING ORM

- Schnittstellen zu vielen etablierten Standard-ORM-Frameworks
 - Hibernate
 - JDO
 - Oracle TopLink
 - iBATIS SQL Maps
 - Java Persistence API (JPA)
- Einfache Konfiguration des Datenbankzugriffs
- Geringer Migrationsaufwand bei Wechsel zu einer anderen Datenbank bzw. einem anderen ORM-System

SPRING 2.0

SPRING DAO

- Verteilte Transaktionen über Datenbankzugriffsmechanismen (JDBC, ORM, EJB, usw.) hinweg
- Deklarative Transaktionsdemarkation (Transaktionsgrenzen werden nicht in den Business-Code hineinprogrammiert sondern konfiguriert)
- Abstraktion von Transaktionsdetails unterschiedlichster Persistenz-Frameworks durch eine einheitliche Unterstützung für JTA, JDBC, Hibernate, JPA und JDO
- Hierarchie von Datenzugriffs-Exceptions (nicht an JDBC gebunden)

SPRING 2.0

SPRING AOP - 1

Begriffsbestimmung AOP:

- Global benötigtes Systemverhalten (*crosscutting concerns*) kann in eigenen Klassen modularisiert und für alle fachlichen Klassen verwendet werden
- Querschnittsfunktionen wie Logging, Tracing, Transaktionen und Sicherheit werden über sogenannte Aspekte aus dem allg. Programmcode in separate Bausteine ausgelagert und über Konfigurationen mit der Logik verbunden
- Programmierparadigma, das die Modularisierung von Programmen fördert

SPRING 2.0

SPRING AOP - 2

Umsetzung in Spring:

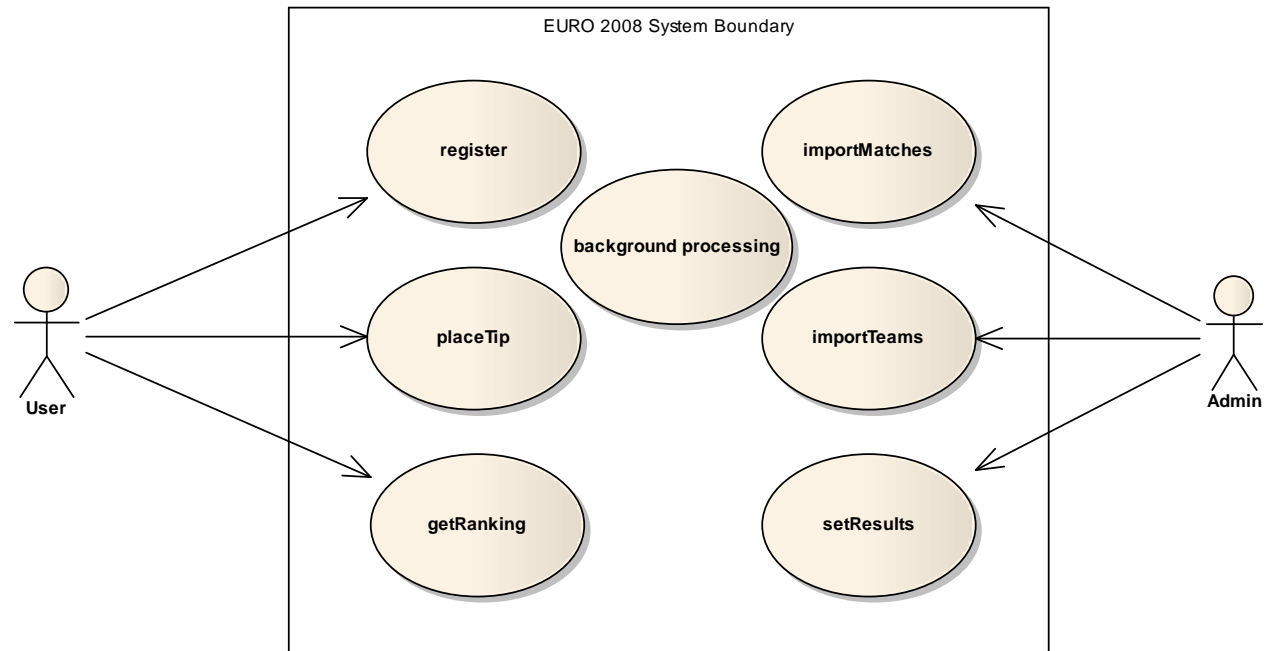
- Stellt leistungsfähige Mechanismen bereit, um Aspekte zur Laufzeit in Programme zu integrieren
 - Aspektorientierte Programmierung ohne Abhängigkeiten von einer Entwicklungsumgebung
 - **AspectJ** als verbreitetste AOP Lösung kann integriert werden
 - Logische und physische Trennung der Semantik (Komponenten) von dem technischen Details (Aspekten)
 - funktionale Erweiterungen ohne Anpassung der Basiskomponenten werden ermöglicht

SPRING 2.0

ANWENDUNGSFALL – EURO2008

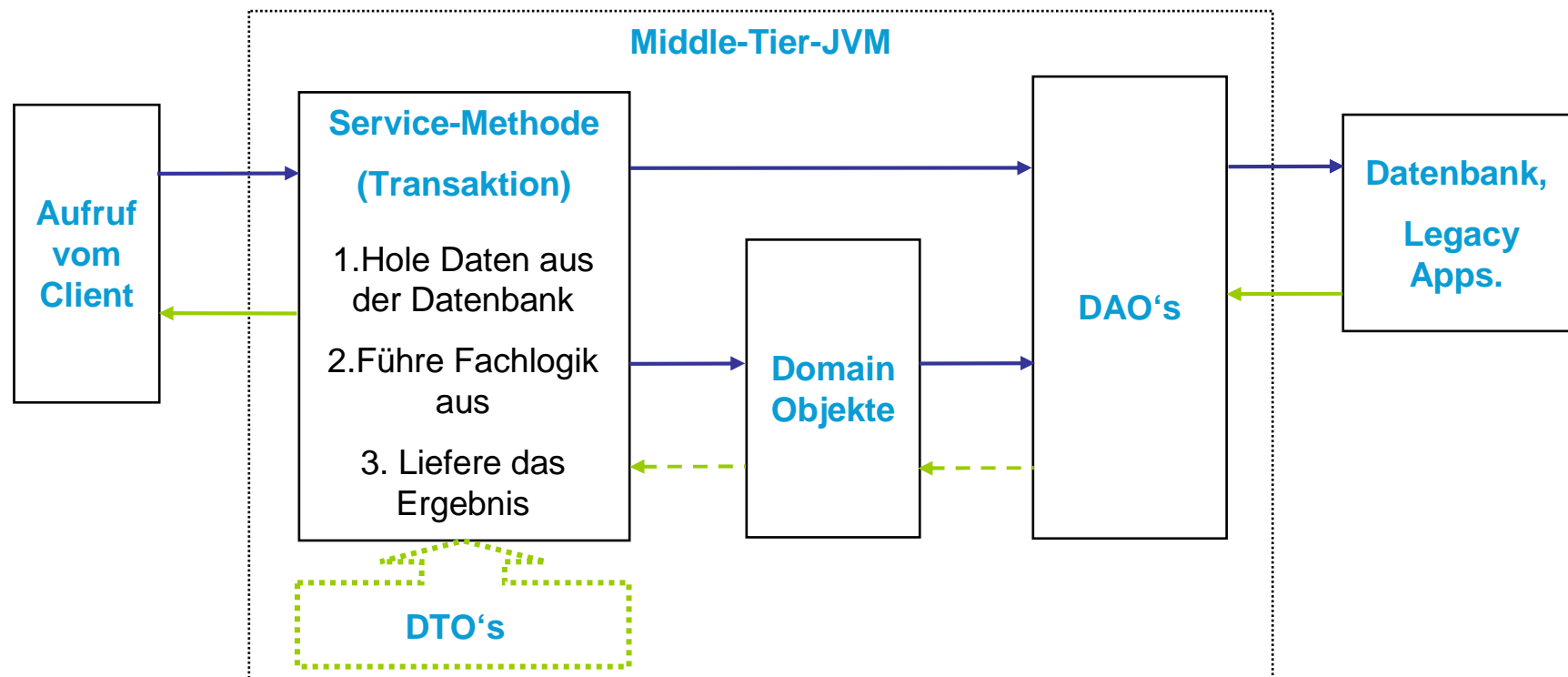


uc Primary Use Cases



SPRING 2.0

ANWENDUNGSFALL – ARCHITEKTUR



SPRING 2.0

ANWENDUNGSFALL – THEMEN

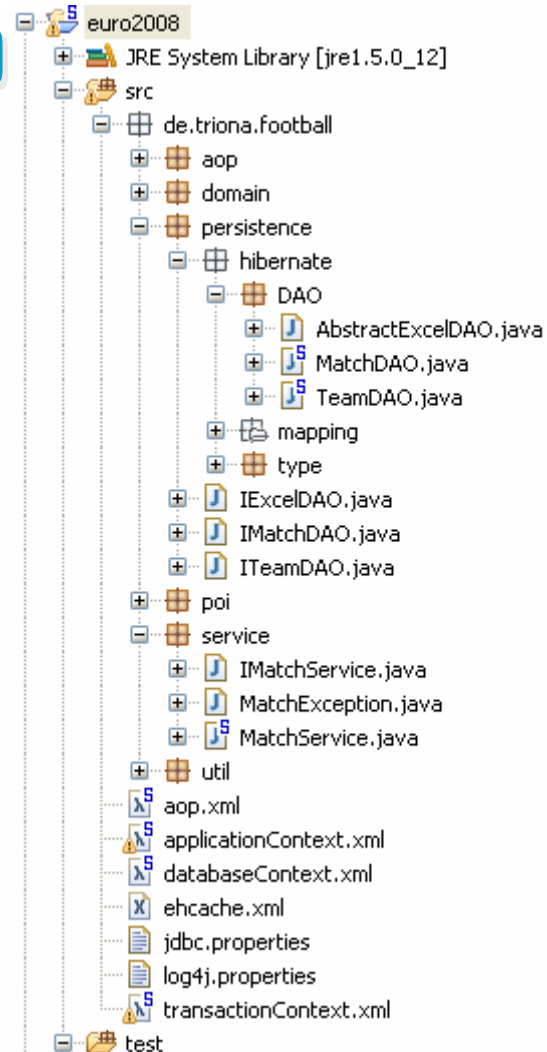


- Konfiguration: Application-Context
- ORM in Spring: Hibernate Integration
- Spring AOP: Einbindung AspectJ
- Tests und Spring Web

SPRING 2.0 EURO2008

- STRUKTUR

- Spring 2.0.6
- Java 1.5.0_12
- AspectJ 1.5.3
- Apache Commons 2.2
- HSQL 1.8
 - DB im Server mode
- Hibernate 3.2.5



SPRING 2.0

NOCH FRAGEN?

