Vorlesung
Software aus Komponenten

4. Moderne Komponentenkonzepte

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2013/14

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Allgemeine Konzepte

- Genauere Trennung und Spezifikation von
  - Entwicklungsmodell
  - Interaktionsmodell
  - Plattformmodell
  - Packaging and Deployment
- Fokus auf serviceorientierte Architekturen
  - stärkere Trennung von **Geschäftslogik** innerhalb der Komponenten und **Interaktionslogik** im Framework
- Fokus auf eng gekoppeltes Zusammenspiel der Interaktionslogik auf einer gemeinsamen BS-Basis (Applikationsserver, Client-PC)
  - Spring-Framework
  - OSGi – Open Service Gateway Initiative
  - CCM – Corba Component Model
- Im Vergleich zu J2EE leicht-gewichtigere Container und Entwicklungsstrukturen (Lean Software Movement)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Separation of Concerns

- **Concerns** = Spezifische Anforderung oder Gesichtspunkte, die in einem Software-System behandelt werden müssen, um die übergreifenden Systemziele zu erreichen (Laddad, 2003).
- Problem: Mehrdimensionalität dieser Anforderungen lässt sich in modularem Entwurfsparadigma nur bedingt abbilden
- Trennung bereits auf der Ebene der Anforderungserhebung in
  - **Fachanforderungen** (*core concerns*), die als Komponenten ausgebildet und früh in die finale Applikation integriert werden
  - **Querschnittsanforderungen** (Logging, Sicherheit, Transaktionskonzepte, *cross cutting concerns*), deren Integration möglichst lange hinausgezögert wird
- In modernen Komponentenmodellen werden Querschnittsanforderungen der Fachapplikation als Basisdienste aus der Plattform über einen klar ausgeprägten **Kontext** (Laufzeitumgebung) injiziert.
  - Ansätze: Dependency Injection, Inversion of Control
  - Verallgemeinerung: Aspektorientierte Programmierung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Aspekte

Dependency Injection

- Klassisch: Objekt ist selbst zuständig, seine Abhängigkeiten (Zuordnung von benötigten Objekten und Ressourcen) zu erzeugen und zu verwalten.
- EJB 2: Steuerung liegt ausschließlich beim Framework, Anforderungen werden über Deployment-Deskriptoren spezifiziert
- Neu: Objekte werden vom Framework (im Kontext) erzeugt und verwaltet und von dort auf der Ebene der Anwendung (POJO) verfügbar gemacht – Verallgemeinerung des Factory-Pattern
  - Dezentralisierung durch Annotationen – Verschiebung von der Werkzeug- auf die Sprachebene

Inversion of Control

- Anwendung (POJO) verhält sich gegenüber dem Kontext wie ein Client gegenüber dem Server

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Aspekte

Aspektorientierte Programmierung

- Kontext wird nicht nur als Datenhintergrund und Lebenszyklus-Verwaltung verstanden, sondern als Laufzeitumgebung, in der die Fachlogik des Anwendungscodes abläuft
- Definition von Joinpoints im Anwendungscode, an denen Aspekte eingeflochten werden können
  - Joinpoints (Compilezeit) und Joinpoint shadows (Laufzeit)
- Definition von Regeln, nach denen an solchen Joinpoints wirkliche Pointcuts ansetzen
- Advices – Art der möglichen Eingriffe: before, after, around
- Vorteil der Aspektorientierung ist die logische und physische Trennung der Semantik von Querschnittsanforderungen in den Komponenten von deren fachlicher Realisierung (Aspekt)
- Entwicklung eines Sprachstandards für derartige Funktionalität.
- Steckt noch in den Kinderschuhen – heute meist nur in Form expliziter Sprachkonstrukte wie in EJB 3 verfügbar.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Das Spring Framework

- Leichtgewichtiges Open Source Applikationsframework für die Java-Plattform, um Entwicklungen innerhalb der J2EE zu vereinfachen
  - Alternative zu den Sun J2EE Blueprints als Architekturempfehlung für J2EE-Anwendungen, dem ein strenges Schichtenkonzept (Web-Schicht, Business-Schicht auf EJB-Basis, Daten-Schicht) zu Grunde liegt.
- Verwirklicht beispielhaft das Zusammenspiel zwischen POJO's auf der Anwendungsebene und verschieden konditionierbaren Framework-Kontexten (Entwicklung, Testumgebung, Produktivumgebung) durch weitere Einbettung der Kontexte
- Auf der Basis existieren weitere Projekte, welche die Anbindung von Spring an verschiedene häufig anzutreffende Entwicklungsumgebungen und -anforderungen realisieren.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Ansatz und Community

- Oktober 2002: Grundlagen werden von Rod Johnson im Rahmen seiner Buchreihe „Expert One-On-One J2EE Design and Development“ entwickelt, der auch den Sourcecode des Framework-Gerüsts als freien Download zur Verfügung stellt.
- Juni 2003: erstes Release, unter der Apache 2.0 Lizenz
- März 2004: Spring 1.0 – seitdem fand das Framework sehr schnelle Verbreitung. Entwicklung wird von SpringSource koordiniert
  - http://www.springsource.org
- Gemeinsame Entwicklungsbemühungen um einen leichtgewichtigen Nachfolger der J2EE
  - Bibliotheken stehen unter der Apache 2.0 Lizenz frei zur Verfügung
  - *Spring Community Forums* als themenspezifische Kommunikationsplattformen
  - *SpringSource Enterprise Bundle Repository* als eine Sammlung von Basiskomponenten

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Mission Statement

We believe that:

- J2EE should be easier to use
- It is best to program to interfaces, rather than classes. Spring reduces the complexity cost of using interfaces to zero.
- JavaBeans offer a great way of configuring applications.
- OO design is more important than any implementation technology, such as J2EE.
- Checked exceptions are overused in Java. A platform shouldn't force you to catch exceptions you're unlikely to be able to recover from.
- Testability is essential, and a platform such as Spring should help make your code easier to test.

We aim that:

- Spring should be a pleasure to use
- Your application code should not depend on Spring APIs
- Spring should not compete with good existing solutions, but should foster integration. (For example, JDO, Toplink, and Hibernate are great O/R mapping solutions. We don't need to develop another one.)

(Quelle www.springsource.org, 2011)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Geschichte

- Oktober 2006: Version 2.0
- November 2007: Version 2.5
- September 2009: SpringSource wird von VMWare übernommen
- Dezember 2009: Version 3.0
  - Neue *Spring Expression Language*, mit der Ausdrücke zur Laufzeit ausgewertet werden können. So lassen sich dynamische Elemente des Systems implementieren, ohne komplexe Techniken wie eine Programmiersprache nutzen zu müssen.
  - Konfiguration mit Annotationen und Meta-Annotationen
  - REST-Unterstützung (Representational State Transfer) im MVC-Framework. Erlaubt den standardisierten Austausch von Zustandsinformationen zwischen verschiedenen Komponenten
  - *Abhängigkeitsmanagement* aus der Plattform ausgelagert, da neue Konzepte und Werkzeuge wie Maven, ivy oder OSGi existieren, die nicht Spring spezifisch sind

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Geschichte

- Seit 2012: Fokus auf Einsatzszenarien und Integration der Konzepte als **Spring IO Platform**
  - http://spring.io als Weiterleitung der bisherigen Adresse springsource.org
  - *Let's build a better Enterprise:* Spring helps development teams everywhere build simple, portable, fast and flexible JVM-based systems and applications.
  - *Build Anything:* Write clean, testable code against the infrastructure components of your choice and accomplish any task – without re-inventing the wheel.
  - *Run Anywhere:* Keep it portable – Spring-based apps run anywhere the JVM does. Deploy standalone, in an app server, on a PaaS or all of the above.
  - *Rest Assured:* Code with confidence – Spring provides an open programming model that is comprehensive, cohesive, widely understood and well-supported.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Geschichte

- 2014: Spring IO brings together the Spring family of projects into a cohesive and versioned foundational platform for modern applications. On top of this foundation it also provides domain-specific execution environments for a variety of enterprise workloads.
- Spring IO brings together the core Spring APIs into a cohesive and versioned foundational platform for modern applications. On top of this foundation it also provides domain-specific runtime environments (DSRs) optimized for selected application types. Spring IO is comprised of the Spring IO Foundation and Spring IO Execution layers.
- Dez 2013: First public milestone of Spring Integration 4.0

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Komponenteneinsatz auch auf der Ebene der Applikationsentwicklung
  - Applikationsdienst wird im Zusammenwirken mehrerer Komponenten (diese werden **Applikationsobjekte** genannt) erbracht, die **innerhalb** der Applikation als Kontext konfiguriert werden.
  - dabei Rückkehr zu plain old Java Objekten (POJO)
- Schichtenmodell mit Standardisierung auf verschiedenen Abstraktionsebenen in eigenen Teilprojekten
  - Konfiguration und Basisarchitektur: Spring Core
  - Kommunikation und Datenaustausch: Spring Integration
  - Prozessmodellierung: Spring Web Flow
  - Benutzerschnittstellen (UI)-Entwicklung: Spring Faces, Spring Web
  - Verteilung und Performance: Spring JEE
  - Datenbankschicht: Spring ORM, Spring DAO
  - Querschnittsfunktionen: Spring AOP

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Komponenten **und** Kontext (als leichtgewichtiger Container) sind Gegenstand der Entwicklung
  - Lösung der Applikation von der inhärenten Bindung an einen J2EE-Server
  - Komponente trifft keine unnötigen Annahmen über den Kontext
    - Applikationsobjekte werden zur Laufzeit von außen konfiguriert.
    - Konfiguration der Komponente transparent für den Anwender, er hat Zugriff auf die Dienstschnittstelle, nicht aber auf die Zuordnung der Ressourcen. Entsprechende Konfigurationsmethoden sind nur auf der Implementierungsklasse bekannt.
    - Explizites Deployment wie bei EJB wird damit von vornherein vermieden.
  - Nutzung von Dependency Injection (Inversion of Control) und Annotationen zur Kommunikation zwischen Komponente und Kontext

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Spring bietet speziellen Support für typische Laufzeitumgebungen an
  - Spring MVC – Framework für Webanwendungen
  - Spring Web Flow – Framework für Abläufe auf einer Web-Site
  - Spring Security (vormals Acegi) – Framework für Sicherheit
  - Spring Rich Client – Framework zur Verteilung der Dienst-erbringung auf Server und Client
- Unterstützung aspektorientierte Ansätze
  - Idee: Basisdienste (cross cutting concerns) werden vom Kontext angeboten und über entsprechende deklarative Schnittstellen (Interzeptoren) in die Komponente eingebunden.
  - Spring bietet eigene Annotationen u.a. im Bereich Transaktionen und Bindung an die Persistenzschicht.
- Fazit: Ein und dasselbe Komponentenmodell kann sowohl für grob granulare Fassadenkomponenten (etwa EJB) wie auch für fein granulare Applikationsobjekte verwendet werden.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Das CORBA Komponentenmodell (CCM)

- mit CORBA 3.0 endgültig spezifizierte ambitionierte (logische) Erweiterung des EJB-Ansatzes
  - Aktuell CCM 4.0 (April 2006)
- CCM-**Anwendung** besteht aus CCM-**Komponenten**
  - EJB erfüllen die CCM-Komponenten-Spezifikation
- CCM-Komponenten sind in **Komponentenpaketen** zusammengefasst
- CCM-**Sammlungen** (CCM assemblies) enthalten Komponentenpakete zusammen mit einer **Beschreibung** der Abhängigkeiten und der Montage-Beschreibung im XML-Format
- Eine CCM-Komponente kann aus mehreren **Segmenten** bestehen
  - CCM-**Laufzeitumgebungen** laden Anwendungen segmentweise

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Das CORBA Komponenten-Entwicklungsmodell

- Mit Komponentenmodell wird auch ein Entwicklungsmodell mit vier Ebenen spezifiziert
  - **Abstract Component Model**: Äußere Eigenschaften von CCM Komponenten und Definition von Komponententypen in IDL
  - **Component Implementation Framework**: Programmiermodell zur Erstellung von Komponentenimplementierungen sowie zur Beschreibung der Relationen zu Implementierungen anderer Komponenten
  - **Container Programming Model**: Beschreibung der Container-Architektur und der Schnittstellen zum ORB und zu den Komponenten.
  - **Packaging and Deployment**: Verpacken von Komponenten und Assemblies, Spezifikation von Deskriptoren sowie des Deployment-Vorgangs

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

CCM in der Praxis

- Im Gegensatz zu Spring ist CCM auf grob granulare Komponentenstrukturen der Anwendungsschicht ausgerichtet und nicht auf Zusammenarbeit von leichtgewichtigen Komponenten innerhalb eines Dienstes
- CCM-Anwendungen laufen nur mit CORBA-3-konformen ORBs
  - wird auch auf der Client-Seite benötigt, wenn die ganze CCM-Funktionalität (etwa Navigation) ausgenutzt werden soll
  - CCM-Standard unterstützt aber abgerüstete Klienten auf pre-CORBA-3-Plattformen (component-unaware clients)
- Es gibt im Gegensatz zu J2EE/EJB und .NET bisher kaum Plattformen, ORBs oder Applikationsserver, die CCM vollständig unterstützen.