

# K Ergänzungen zu EJB

---

## K.1

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AvID-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 1 Überblick

---



- Bewertung des EJB2-Entwicklungsprozesses
- XDoclet
- Bewertung der EJB2-Architektur
- Alternativer Ansatz
- Spring Framework
- EJB 3
- Beispiele und Vergleich von EJB3 und Spring

## K.2

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AvID-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 2 Bewertung des EJB2-Entwicklungsprozesses

---



- Viele Software/Deployment-Artefakte
  - ◆ Bean-Implementierung, Home/Bean-Interfaces
  - ◆ Bean-Descriptor, Application-Descriptor
    - Jeweils generischer und Server-spezifischer Teil
- Vorteile
  - ◆ Code getrennt von deklarativer Konfiguration
  - ◆ Flexible „Application Assembly“
    - Austauschbare Module mehrerer Zulieferer
- Nachteile
  - ◆ Komplexer Zugang zur Sicht auf Gesamtsystem
  - ◆ Konsistenz bei größeren Projekten schwer einzuhalten
  - ◆ Redundanz zwischen den Artefakten
- \* Ansatz: Artefakte aus Implementierung generierbar, z.B. mit XDoclet

**K.3**

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AvID-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 3 XDoclet

---



- Javadoc-ähnliche Tags in Kommentaren
  - ◆ Vor das betreffende Element gestellt
  - ◆ Innerhalb Javadoc-Kommentar `/** ... */`
  - ◆ Format: `@tag.name attribute="value" attribute...`
- Tags für diverse Technologien
  - ◆ Servlets, EJB, Persistenz-Frameworks, ...
  - ◆ Unterstützung diverser Container und Hersteller
- Generator-Skript (z.B. mit ANT)
  - ◆ Selbst zu erstellen oder generiert/anpassbar durch IDE
  - ◆ Durchsucht Auswahl an Dateien
  - ◆ Erzeugt Code, Deskriptoren
  - ◆ Abhängig von Tag-Gruppe (z.B. `@ejb.*`, `@jboss.*`)

**K.4**

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AvID-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

### 3 XDoclet (2)



#### ■ Beispiel: Entity-Bean für JBoss inkl. Finder

```
/**
 * @ejb.bean name="Entry"
 *           jndi-name="ejb/Entry"
 *           type="CMP"
 *           cmp-version="2.x"
 *           view-type="local"
 *           primkey-field="id"
 *
 * @jboss.unknown-pk class="java.lang.String"
 * @jboss.persistence alter-table="true"
 *
 * @ejb.finder
 *   view-type="local"
 *   signature="java.util.Collection findByName(...String name)"
 *   query="SELECT object(l) FROM Layer AS l WHERE l.name = ?1"
 */

public abstract class EntryBean implements EntityBean { ... }
```

**K.5**

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AviD-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

### 3 XDoclet (3)



#### ■ Beispiel: Create-Methode, CMP-Field mit DVO-Anbindung

```
/**
 * @ejb.create-method view-type = "local"
 */

public String ejbCreate(...) throws CreateException {...}

/**
 * CMP Field: selfmade Relation to Owner
 *
 * @ejb.value-object match="light"
 * @ejb.persistent-field
 * @ejb.interface-method view-type="local"
 */

public abstract String getFKOwnerId();
```

**K.6**

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AviD-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

### 3 XDoclet (4)



#### ■ Beispiel: 1:N Container-Managed Relationship mit DVO-Anbindung

```
/**
 * @ejb.interface-method   view-type="local"
 *
 * @ejb.relation           name = "Layer-Layer-Hierarchy"
 *                         role-name = "Layer-parents-Layers"
 *
 * @ejb.value-object
 *     relation="external"
 *     aggregate="torg.beans.entity.LayerLightValue"
 *     aggregate-name="SubLayer"
 *     members="torg.beans.entity.LayerLocal"
 *     members-name="SubLayer"
 */

public abstract java.util.Set getSubLayers();
```

K.7

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AvID-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

### 3 XDoclet (5)



#### ■ Vorteile

- ◆ Alles im Quellcode erfasst, keine extra Deskriptoren
- ◆ Hohe Lokalität von Code und zugehörigem Deployment
- ◆ Namen/Typen z.T. implizit ermittelbar
- ◆ Helper-Klassen automatisch generierbar
  - Für DVOs, Persistenz, JNDI-Caches, ...

#### ■ Nachteile

- ◆ Enge Kopplung von eigener Logik und sehr spezifischen Metadaten
- ◆ Fehleranfällige Metainformation als „Plain-Text“
- ◆ Keine Typisierung oder Compiler-Prüfung
- ◆ Teils noch schwache IDE-Unterstützung

#### \* Lösung: Java Annotations?

K.8

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AvID-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 4 Exkurs: Java Annotations

---

- Sprachelemente mit Metadaten verknüpfbar
  - ◆ Typisiert, vom Compiler prüfbar, gute IDE-Unterstützung

- Deklaration als spezielle „@...“ Schnittstelle

```
import java.lang.annotation.*;

@Retention(RetentionPolicy.RUNTIME)
@Target( {ElementType.METHOD, ElementType.TYPE} )

public @interface MyCMS {
    String [] authors() default {""};
}

@MyCMS({"Me", "She"}) class Test { ... }
```

- Anwendung auf verschiedene Ziele bzw. „Targets“
  - ◆ Typen, Methoden, Parameter, ...
- Sichtbarkeit je nach „Retention“
  - ◆ Nur im Quellcode, im Class-File, auch zur Laufzeit per Reflection

**K.9**

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AvID-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 5 Bewertung der EJB2-Architektur

---

- Nicht-funktionale Eigenschaften unterstützt
  - ◆ Authorisierung, Transaktionsmanagement, Persistenz, Logging, Assembly, ...
  - ◆ In der Regel Querschnittsaspekte, sog. „Crosscutting Concerns“
- Vor EJB
  - ◆ Enge Kopplung von eigener Logik und Bibliotheksfunktionen
  - ◆ Negative Effekte: Code-Tangling und Scattering/Crosscutting
- Mit EJB2
  - ◆ Zentrale Unterstützung im Container
  - ◆ Deklarative Konfiguration, ggf. mit vorbereitetem Code (abstract ...)
- \* Teilweise Umsetzung der „Separation of Concerns“ (SoC)
  - ◆ Nur eine Illusion?

(QA=Querschnittsaspekt)

**K.10**

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AvID-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 5 Bewertung der EJB2-Architektur (2)

---

- ▲ Hohe Anforderungen an Bean-Code
  - ◆ Signaturen, Interfaces, System-Aufrufe, ...
  - ◆ Enge Kopplung der eigenen Logik an EJB-Prinzipien
- ▲ Keine echte „Separation of Concerns“
  - ◆ Entwicklung nie ganz losgelöst von QA realisierbar
  - ◆ Konzentrierte Entwicklung eines einzelnen QA schwierig
- ▲ „Inversion of Control“ (IoC) unvollständig
  - ◆ Lose Kopplung mit externer Konfiguration
  - ◆ „Dependency Lookup“ an JNDI-Service im eigenen Code
- ▲ Enge Kopplung an EJB-Container
  - ◆ Portabilitätsprobleme zwischen EJB-Versionen
  - ◆ Kein eigenständiger Test ohne Container/Datenbank/QA möglich
  - ◆ Aufwendige Entwicklungszyklen: langwieriges Deployment

**K.11**

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AvID-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 6 Alternativer Ansatz

---

- \* Alternative: POJOs + nicht-invasive, leichtgewichtige Frameworks
- POJO = „Plain Old Java Object“
  - ◆ Reguläres Java-Objekt ohne Bindung an spez. Framework/Interfaces
  - ◆ Test-driven Development ohne Container/Datenbank/QA möglich
  - ◆ Schnelle Entwicklungszyklen
  - ◆ Höhere Portabilität da keine Technologie-Bindung
- Nicht-invasive Frameworks
  - ◆ Aufrufe an POJO-Methoden werden abgefangen („Interception“)
  - ◆ Für „umhüllende“ QA nutzbar: Transaktionen, Authorisierung, ...
  - ◆ Oft präzisere, benutzerfreundliche Metadaten (subjektiv?)
  - ◆ Evolution/Austausch mit geringeren Änderungen in Code und Metadaten
- Aktuelle Repräsentanten: Spring (Java/.Net), Hibernate, JDO, EJB3

**K.12**

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AvID-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 7 Spring Framework

---



- \* Leichtgewichtiges POJO-Framework
- Flexibel anpassbar
  - ◆ Core für Container-Infrastruktur und Komponentenmodell
  - ◆ Module für AOP, ORM/DAO, TXN, Web, Services, MVC auswählbar
- Inversion of Control konsequent umgesetzt
  - ◆ Dependency Injection (Setter/Constructor Injection)
    - Hollywood Principle „Don't call us, we'll call you!“
  - ◆ Deklarative Konfiguration der Daten-Umgebung
  - ◆ Container sorgt für Verknüpfung mit benötigten Objekten
- Paradigma „aspektororientiertes Programmieren“ (AOP) genutzt
  - ◆ Generische Schnittstelle für Interception und QA
  - ◆ Container erweitert Methodenaufrufe um QA-„Advices“
  - ◆ Bsp.: TXN\_BEGIN vor Methodenaufruf, TXN\_COMMIT danach

**K.13**

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AvID-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 8 EJB 3

---



- Java Enterprise Edition 5
- Ziel: Vereinfachung — „Ease of Development“
  - ◆ Überarbeitetes EJB API
  - ◆ Überwiegender Verzicht auf obligatorische Deskriptoren
  - ◆ Eliminierung von Antipatterns: Dependency Injection statt JNDI-Abfragen
  - ◆ Neues Persistence API integriert CMP- /JDO-Konzepte
  - ◆ Bessere Unterstützung für Web-Services und XML
  - ◆ Stärkere Integration von JavaServer Faces
- Angepasstes Programmiermodell
  - ◆ Nutzt Java 2 Standard Edition 5 mit **Annotations**
  - ◆ Gestützt durch „Convention over Configuration“
    - Schnittstellen und Konfiguration aus Defaults, Namen, Typen abgeleitet
    - Optionale Deskriptoren ergänzen/verändern Standardverhalten

**K.14**

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AvID-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 9 Beispiele EJB3 und Spring

### ■ Application-Assembly durch „Dependency Injection“

### ■ Spring Framework

#### ◆ Deployment

```
<bean id="db" class="my.own.DatabaseConnector" />
<bean id="accountBean" class="my.own.Account">
  <property name="datasource">
    <ref bean="db" />
  </property>
</bean>
```

#### ◆ Optional Autowiring

- Ableitung der Verknüpfung aus Attribut-Typen/Namen

### ■ EJB3

#### ◆ Annotations und Ableitung aus Attribut-Typen/Namen

```
class SomeUser {
  @EJB private DatabaseConnector defaultDatabase;
}
```

K.15

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AvID-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 9 Beispiele EJB3 und Spring (2)

### ■ Transaktions-Management (vgl: ACM Queue (4)5, S. 41)

#### EJB3-Annotations

```
@Local
public interface BankXfer {
  ...
}
```

#### @Stateless

```
public class BankXferImpl
  implements BankXfer {
  ...
}
```

#### Spring Framework Descriptor

```
<aop:config>
  <aop:advisor
    pointcut=
      "execution(* *..BankXfer.*(..))"
  >
```

```
    advice-ref="txAdvice"
  >
</aop:config>
```

```
<tx:advice id="txAdvice">
  <tx:attributes>
    <tx:method name="*" />
  </tx:attributes>
</tx:advice>
```

K.16

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AvID-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 9 Beispiele EJB3 und Spring (3)



- Persistenz (vgl: ACM Queue (4)5, S. 43)

### EJB3-Annotations

```
@Entity
@Table(name="ACCOUNT")
public class Account {
    ...

    @Id @Column(name="A_ID")
    private int id;

    @Column(name="A_BALANCE")
    private double balance;

    ...
}
```

### Hibernate Descriptor (integriert in Spring)

```
<class name="Account" table="ACCOUNT">
    <id name="id" column="A_ID">
        <generator class="native"/>
    </id>

    <property name="balance"
        column="A_BALANCE" />

    ...
</class>
```

**K.17**

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AvID-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 10 Vergleich EJB3 und Spring



- Beide nutzen Dependency Injection
- Spring nutzt Deskriptoren intensiv
  - XDoclet-Unterstützung möglich
- EJB legt Schwerpunkt auf Defaults und Annotations
  - Overrides durch Deskriptoren möglich
- ▲ Grundsätzlicher Diskussionsbedarf bleibt
  - ◆ Unerwünschte Technologie-Bindung durch XDoclet-Tags oder Annotations im Quellcode?
- ▲ Marktbeobachter positiv/negativ
  - ◆ „EJB3 kam zu spät“
  - ◆ „(Web-Services mit) .NET einfacher zu entwickeln“
  - ◆ „J2EE im Server-Bereich stärker/flexibler/ausbaufähiger als .NET“
- \* Bilden Sie sich Ihre eigene Meinung!

**K.18**

© 2002-2006, Andreas I. Schmied, Verteilte Systeme, Univ. Ulm, [2006s-AvID-K-Ergaenzung.fm, 2006-07-24 16.58] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 11 Literatur

---



- Richardson: Untangling Enterprise Java  
ACM Queue, Vol.4 No.5, Juni 2006, <http://www.acmqueue.com>

**K.19**