Überblick

- Motivation & Einführung
- Annotations
  - Deklaration
  - Anwendung
  - Auswertung
- Aktuelle Einsatzgebiete
- Diskussion

Programmieransätze

Imperativ

Verhalten wird programmatisch ausgedrückt (direkte Anweisungen)

Spezifiziert **WIE** etwas zu tun ist

Business-Logik potentiell ‘vermischt‘ mit Crosscutting-Concerns / Services

Beispiele:

- Security

if(User.inRole(Roles.BERATER) {...Business-Logik...}

- Transaktionsverhalten

userTransaction.start() {...Business-Logik...} userTransaction.commit()

- Persistenz

Customer.setBirthday(sqlResult.getDate(“DAY_OF_BIRTH“))

Programmieransätze

Deklarativ

Deklarative Zuweisung von Services mittels Metadaten

Zuweisung liegt ‚ausserhalb' des Codes (z.B. Deploym.Deskriptor)

Service-Anbieter (Container) wertet Metadaten zur Compile- / Laufzeit aus

Spezifiziert **WAS** zu tun ist (und überlässt das WIE der 'Umgebung')

Beispiel:

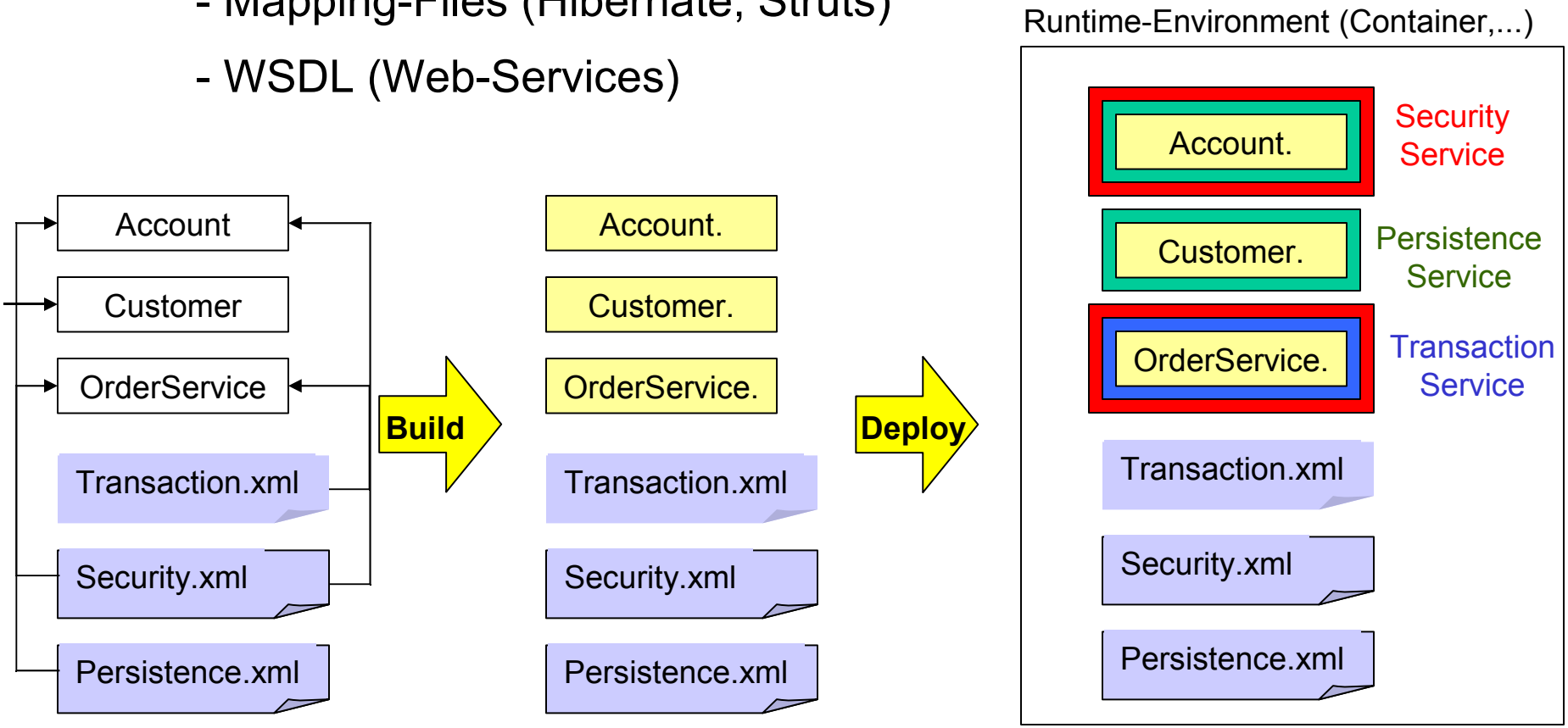
```
<enterprise-beans>
...
    <session id="OrderService">
        <ejb-name>OrderService</ejb-name>
        ...
        <remote>com.mgi.bank.OrderService</remote>
        ...
        <session-type>Stateless</session-type>
        <transaction-type>Container</transaction-type>
        ...
```

Metadaten & XML

Metadaten liegen in separaten XML-Dateien

Getrennt von den Klassen, auf welche diese sich beziehen (n:m)

- Deployment-Deskriptoren (EJB)
- Mapping-Files (Hibernate, Struts)
- WSDL (Web-Services)

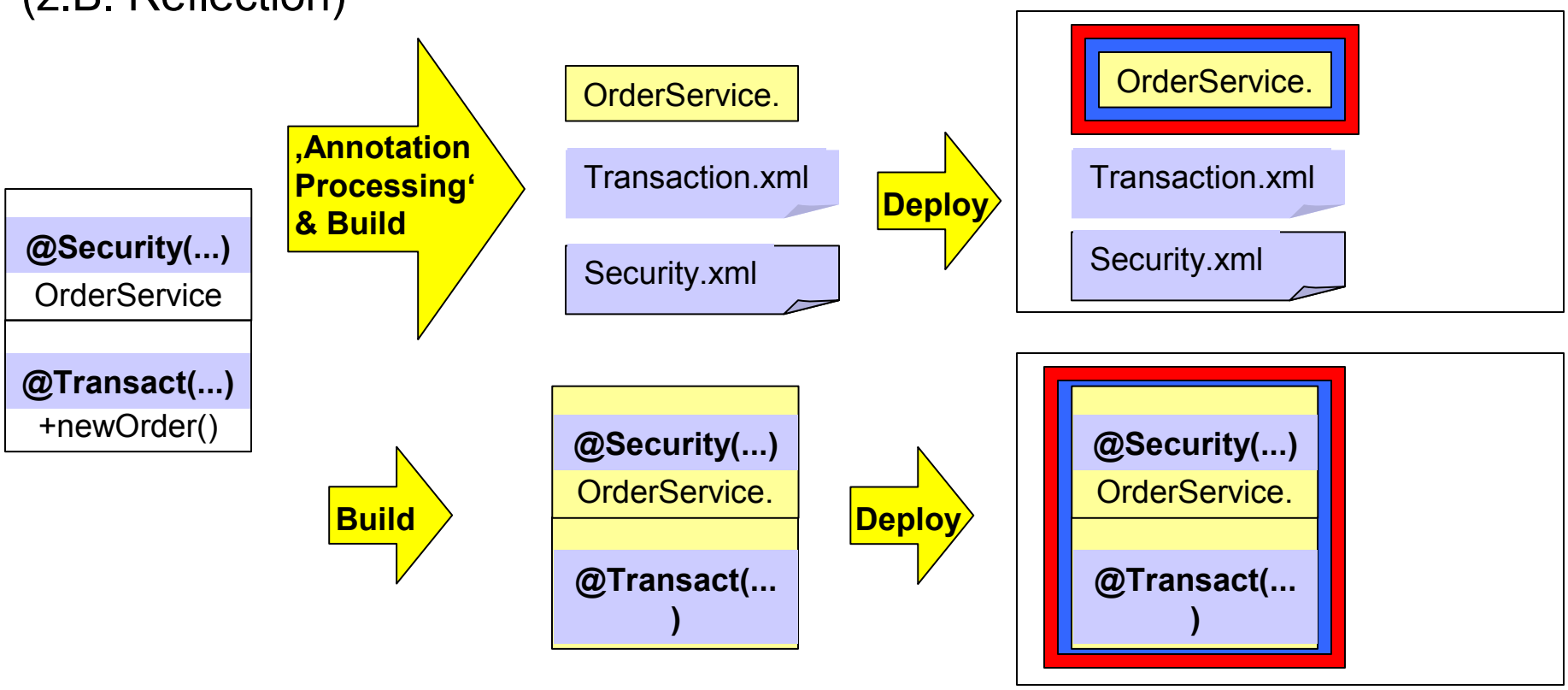


Annotationen

Direkte Zuweisung von Metadaten an diverse Elemente einer Klasse (Klasse, Methode, Feld, ... , vgl. XDoclet, JSR 175)

Metadaten werden direkt an den betroffenen Stellen im Code ‚annotiert'

(Laufzeit-)Umgebung kann Metadaten direkt aus Klassen ermitteln (z.B. Reflection)



Deklaration von Annotationen - Marker-Annotationen

Anwesenheit der Annotation trägt Bedeutung in sich

Annotation trägt keine weiteren Attribute (vgl. Deprecated)

Ähnlich Marker-Interfaces (Serializable), Doclets (@Deprecated)

Beispiel:

```
package com.mgi.annotations;

public @interface ToDo {}


package com.mgi.bank;   import com.mgi.annotations.ToDo;

public class Customer{
        ...
        @ToDo
        public int calcCreditRating(){
                return 0;
        }
}
```

Single-Value-Annotationen

Annotation kann Attribut(e) besitzen

Attribute können zusätzliche Semantik tragen

Attribute können von Umgebung ausgewertet werden

(Kurzschreibweise bei nur einem Attribut (value) möglich)

Beispiel:

```
public @interface ToDo {
        String value();
}


public class Customer{
        …
        @ToDo( "Formel zur Bonitätsberechnung implementieren" )
        public int calcCreditRating(){
                return 0;
        }
}
```

Multi-Value-Annotationen

Attribute werden mittel ‚abstrakter Methode' deklariert
(Methodenname entspricht nicht der JavaBean-Spezifikation)

Beispiel:

```
public @interface ToDo {
    public enum Prio { HOCH, MITTEL, NIEDRIG };
    String aufgabe();
    Prio prio();
    String verantwortlicher();
}


public class Customer{

    @ToDo(    aufgabe="Formel zur Bonitätsberechnung implementieren",
              prio=TODO.Prio.MITTEL,
              verantwortlicher="Mickey Mouse" )
    public int calcCreditRating(){
              return 0;
    }
}
```

Default-Werte

Attribute werden mit Standardwerten vorbelegt

Nicht gesetzte Attribute werden mit Standardwerten belegt

‘Convention over Configuration‘ (vgl. EJB 3, Ruby on Rails)

Beispiel:

```
public @interface ToDo {
    public enum Prio { HOCH, MITTEL, NIEDRIG };
    String aufgabe();
    Prio prio() default Prio.MITTEL;
    String verantwortlicher() default “Donald Duck”;
}

public class Customer{
    @ToDo(  aufgabe=“Formel zur Bonitätsberechnung implementieren”,
            verantwortlicher=“Mickey Mouse” )
    public int calcCreditRating(){
            return 0;
    }
}
```

Meta-Annotation *Target*

Anwendung von Annotationen auf Annotationen

Einschränkung des Anwendungsbereichs von Annotationen mittels *Target*

Beispiel:

```
import java.lang.annotation.ElementType;
import java.lang.annotation.Target;

@Target({
        ElementType.TYPE,                   // Klasse
        ElementType.METHOD,                 // Methode
        ElementType.CONSTRUCTOR,            // Konstruktor
        ElementType.ANNOTATION_TYPE}        // Annotation
)
public @interface ToDo {
        public enum Prio { HOCH, MITTEL, NIEDRIG };
        String aufgabe();
        Prio prio() default Prio.MITTEL;
        String verantwortlicher() default "Donald Duck";
}
```

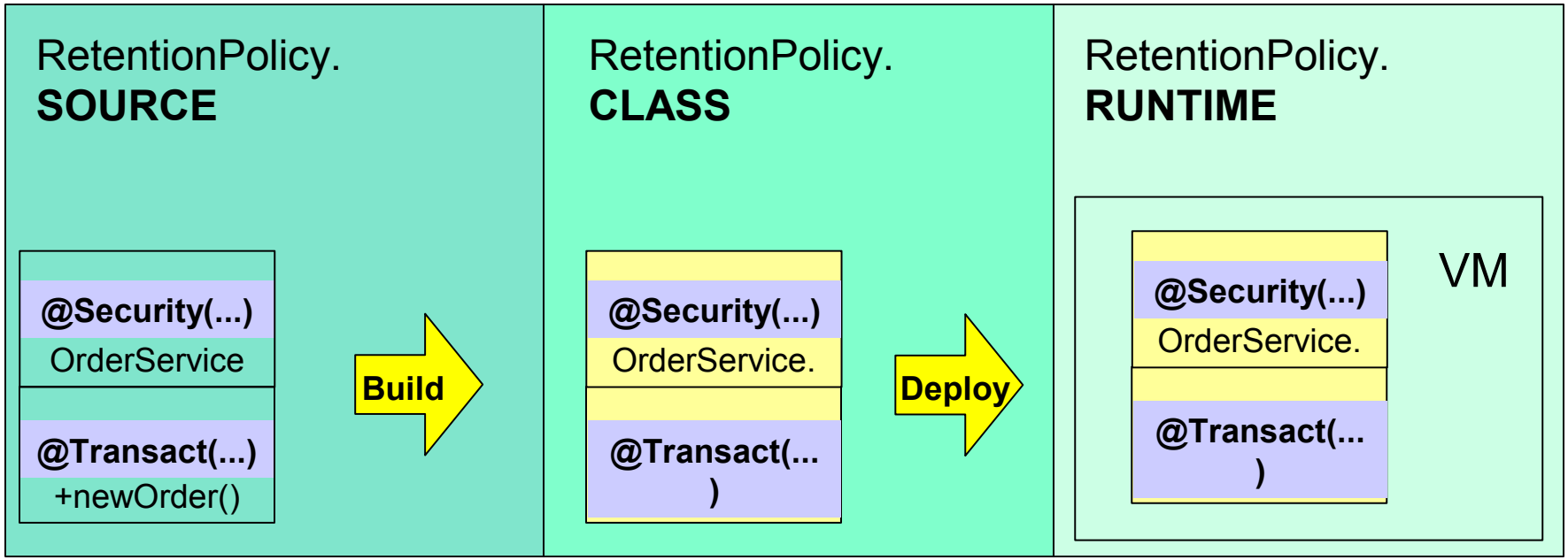
Meta-Annotation *Retention*

Definition der ‘Lebensdauer‘ einer Annotation

Abhängig vom zugedachten Auswertungszeitpunkt der Annotation

Beispiel:

```
@Retention(RetentionPolicy.RUNTIME);
@Target(...)
public @interface ToDo {...}
```



Auswertung von Annotationen

Annotation u.a. zur Laufzeit mittels Reflection auswertbar

Beliebige ‘Aktionen‘ aufgrund von Annotationen definierbar

(Bsp. ‘Report aller Klassen, welche ToDos mit Prio 1 oder Prio 2 besitzen‘)

Beispiel:

```
public class ToDoReportGenerator{
    public static reportPrio1andPrio2ToDos
      ( Class clazz, OutputStream out ){
      if( clazz.isAnnotationPresent( ToDo.class ) ){
        ToDo toDo = (ToDo) clazz.getAnnotation( ToDo.class );
        if( ToDo.Prio.MITTEL == toDo.prio() ||
            ToDo.Prio.HOCH == toDo.prio() ){
          out.println( clazz.getName() + ":" +
                       toDo.aufgabe() + ":" +
                       toDo.verantwortlicher() );
        }
...
```

EJB 3 SessionBean

Annotationen legen Typ und Zugriffsart der Bean fest

Lifecycle-Methoden nur bei Bedarf annotieren (beliebige Custom-Methoden)

Kein Implementieren von EJB-spezifischen Interfaces notwendig

Bean selbst ist ein POJO und dessen Business-Logik als solches ausserhalb des Containers testbar

Beispiel:

```
@javax.ejb.Stateless
@javax.ejb.Remote
public class OrderServiceBean implements OrderService {

  public Order createOrder(){...}

  @PreDestroy
  cleanUpOrders(){...}
...
}
```

Transaktions-Demarkation (EJB 3)

Annotation legt fest, auf welcher Methode Transaktionen liegen

Attribute legen Transaktions-Eigenschaften fest
(z.B. Rollback-Exceptions, Transaktionsattribut, Timeout, ...)

Convention over Configuration
(z.B. Standard-Timeout nach 120 Sekunden)

Beispiel:

```
@javax.ejb.Stateless
@javax.ejb.Remote
public class OrderServiceBean implements OrderService {

  @javax.ejb.TransactionAttribute(SUPPORTS)
  public Order createOrder(){...}

  @PreDestroy cleanUpOrders(){...}
}
```

Transaktions-Demarkation (Spring)

Ähnliche Annotation für Spring-‘Umgebung‘

Bean wäre nun sowohl in einem EJB-Container als auch einer Spring-Umgebung deploybar

Übersichtlich? (Standard-Annotations für Transaktions-Demarkation?)

Beispiel:

```
@javax.ejb.Stateless
@javax.ejb.Remote
@Transactional(readOnly=true)
public class OrderServiceBean implements OrderService {

  @javax.ejb.TransactionAttribute(SUPPORTS)
  @Transactional(readOnly=false,
                 rollbackFor=DuplicateOrderIdException.class)
  public Order createOrder(){...}
...
}
```

Security (EJB 3)

Zugrundeliegendes Rollen-Modell ‘separat‘ konfiguriert

Rechtefreigabe auf Rollenebene per Annotation

Beispiel:

```
import javax.ejb.MethodPermissions;
import javax.ejb.Unchecked;

@MethodPermissions("admin")
public class LoginBean{
    public void updateProfile(){...}

    @MethodPermissions(“guest")
    public void login(){...}

    @Unchecked
    public void homePage(){...}
}
```

Security (Acegi)

Zugrundeliegendes Rollen-Modell ‘separat‘ konfiguriert

Rechtefreigabe auf Rollenebene per Annotation

Beispiel:

```
import net.sf.acegisecurity.annotation.Secured;

@Secured(“ADMIN")
public interface BankManager{

   @Secured(“BERATER“,“KREDIT_ENTSCHEIDER“)
   public float getBalance( int id ){...}
...
}
```

O/R-Mapping (Hibernate 3)

Mapping von Beans / Bean-Properties auf Tabellen / Tabellenfelder

Deklaration von Beziehungen zwischen Beans

Feintunig (Cache-Strategien, Fetch-Strategien, ...)

Beispiel:

```
@Entity(access = AccessType.FIELD)
@Table(name="CUSTOMER")
public class Customer implements Serializable {

    @Id;
    Long id;

    String firstName;
    String lastName;
    @IndexColumn(name="DAY_OF_BIRTH", nullable=false)
    Date birthday;

    @Transient
    Integer age;

    @OneToMany(cascade=CascadeType.ALL)
    @JoinColumn(name="CUSTOMER_ID")
    Set<Order> orders;
...
}
```

O/R-Mapping (Hibernate 3)

Wo ist hier der Code ?

‘Code-Pollution‘

Beispiel:

```
@Entity(    selectBeforeUpdate = true,
            dynamicInsert = true, dynamicUpdate = true,
            optimisticLock = OptimisticLockType.ALL,
            polymorphism = PolymorphismType.EXPLICIT)
@Where(clause="1=1")
@BatchSize(size=5)
@Cache(usage = CacheConcurrencyStrategy.NONSTRICT_READ_WRITE)
@FilterDef( name="minLength",
            parameters={ @ParamDef( name="minLength", type="integer" ) } )
@Filters( {  @Filter( name="betweenLength",
                      condition=":minLength <= length and :maxLength >= length"),
            @Filter( name="minLength",
                      condition=":minLength <= length") } )
public class Forest {...}
```

Design by Contract (MG Informatik)

Annotationen deklarieren - Invarianten auf Klassen
- Vor- bzw. Nachbedingungen auf Methoden

Fachliche Aussagekraft einer Klasse / Methode erhöhen

Überprüfung auf Einhaltung des Vertrags aktivieren (Entwicklung, Test) bzw. deaktivieren (Produktion)

Beispiel:

```
@invariant( conditions={ “balance >= 0“ } )
public class DispoAccount{

    @precondition(  condition=“amount >= 0 && toAccount != null && this.balance >= amount“ )
    @postcondition(condition=“toAccount.balance >= amount“ )
    public int transfer( int amount, DispoAccount toAccount ){...}

    public int getBalance(){...}
...
}
```

JUnit 4 (Preview)

Beispiel:

```
public class OrderDaoTest{
        @BeforeClass
        protected void createLocalDataSource{...}

        @Before
        protected void initOrderDaoHelpers {...}

        @Before
        protected void resetOrderTestTable(){...}

        @Test
        public void createOrder(){...}

        @Test(expected=DuplicateOrderException)
        public void createDuplicateOrder(){...}

        @Ignore
        @Test(timeout=1000)
        public void retrieveAllOrders(){...}

        @After
        protected void disposeOrderDao(){...}

        @AfterClass
        protected void disposeLocalDataSource(){...}
}
```

Weitere Einsatzgebiete

Web-Services (C#)

```
[WebService(Namespace=www.mg-informatik.de/CalculatorService)]
public class CalculatorService : System.Web.Services.WebService{

  [WebMethod]
  public int add( int a, int b){
    ...
```

Dependency-Injection (EJB 3)

```
@Inject(jndi-name="java:comp/env/jdbc/testDB")
void setDataSource(javax.sql.DataSource pDataSource){
  dataSource = pDataSource;
}
```

Validierung von Benutzereingaben (Java Server Faces)

```
public class Address {
    @MinLength(10)
    @MaxLength(20)
    public void setStreet(String street) {
        this.street = street;
    }
}
```

Diskussion - Imperativ vs. Deklarativ

Zielstellung: Vereinfachung des Entwicklungsprozesses

Testbarkeit

- Explizite Abhängigkeit von Services der Umgebung (Container) bei programmatischem Service-Einsatz

Nachvollziehbarkeit des Verhaltens einer Klasse

- Code-Tangling (BusinessLogik + Crosscutting-Concerns) vs. Code-Hiding

Lesbarkeit des Codes

- Code-Pollution (Code + Configuration) vs. Code-Tangling

Diskussion - separate oder annotierte Metadaten

Lesbarkeit / Notationsaufwand

- Explizite Kontextbeschreibung vs. Code-Pollution (Code + Config.)
- Explizite Deklaration vs. Vererbung

Wartbarkeit / Änderbarkeit / Erweiterbarkeit

- Redeploy vs. Rebuild (Konfiguration von class-Files)
- Zentrale vs. Dezentrale Konfiguration (One-to-Many vs. One-to-One)
- Überblick / Nachvollziehbarkeit über deklarierte Services

Typsicherheit

- Text vs. typsichere Attribute
- Defaultwerte

Quellen

JSR 175: A Metadata Facility for the Java Programming Language
http://www.jcp.org/en/jsr/detail?id=175

Narayanan Jayaratchagan (2004): Declarative Programming in Java
http://www.onjava.com/pub/a/onjava/2004/04/21/declarative.html

Mike Keith (2005): To annotate or not?
http://www.oracle.com/technology/pub/columns/annotations_opinion.html

Enterprise JavaBeans 3.0 Public Review Draft
http://www.jcp.org/en/jsr/detail?id=220

Hibernate Annotations
http://annotations.hibernate.org

Spring
http://www.springframework.org