Orientation in Objects

# CDI - Dependency Injection Standard für Java

Ein Überblick
OIO Orientierungspunkt April 2014

**Orientation in Objects GmbH**

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.0

## Ihr Sprecher

Orientation in Objects

Thomas Asel

*Trainer, Berater, Entwickler*

***Schwerpunkte***
*Frontend-Architektur,*
*Entwicklung von Web-Anwendungen,*
*Web-Performance-Optimierung*

*http://blog.oio.de*
*@Tom_Asel*
*thomas.asel@oio.de*

© 2014 Orientation in Objects GmbH

## CDI - JSR 299

- Entwickelt im Rahmen von JSR 299
  - Offener Standardisierungsprozess (JCP)
  - Final seit 12/2009
  - Erstmals in Java EE 6 enthalten
  - Aktuell: CDI 1.1 JSR 346 (Java EE7)
- Treibende Kraft: Gavin King, RedH

- Vorläufer: JBoss Seam, RedHat

- Referenzimplementierung: JBoss Weld

- Alternative Implementierung: Apache Open WebBeans

© 2014 Orientation in Objects GmbH


## CDI im Java EE Stack

- Kein "Layer" im Sinne einer Schichtenarchitektur
- Container für Managed Beans
- DI-Mechanismus für EE-Komponenten
- Events
- Interceptoren
- …

© 2014 Orientation in Objects GmbH

## Dependency Injection – Einfaches Beispiel

Orientation in Objects

```
public class MessageService {

   @Inject EntityManager entityManager;

   public void saveMessage(String message) {
        this.entityManager.persist(message);
   }

}
```



## Injection Points

Orientation in Objects

```
public class MessageBean {

   @Inject private LogService logService;
   private final String defaultMessage;
   private MessageService messageService;

   @Inject
   public MessageBean (String defaultMessage) {
        this.defaultMessage = defaultMessage;
   }

   public MessageService getMessageService() {
        return messageService;
   }

   @Inject
   public void setMessageService(MessageService messageService) {
        this.messageService = messageService;
   }
}
```

## Typesafe Resolution mit Qualifier

Orientation in Objects

- Realisierung über eigenen Annotationstyp:

```
@Qualifier
@Retention(RUNTIME)
@Target({TYPE, METHOD, PARAMETER, FIELD })
public @interface SimpleLogging {

}
```



## Typesafe Resolution mit Qualifier

Orientation in Objects

- Implementierung:

```
@SimpleLogging
public class SimpleLogger implements Logger {

    @Override
    public void log(String message) {
          System.out.println("Logger: " + message);
    }
}
```

- Konsument:

```
public class LogService {

    @Inject @SimpleLogging Logger simpleLogger;
    @Inject @DetailedLogging Logger detailedLogger;

    public void log(String logMessage) {
          detailedLogger.log(logMessage);
          simpleLogger.log(logMessage);
    }
}
```

## EL-Namen

Orientation in Objects

- `@Named("loginCredentials")`
- Referenzierung von Beans ausserhalb von Java-Code möglich
  - JSP
  - JSF Views
- Expression Language:
  - Data Binding von UI-Komponenten an Beans und deren Attribute
  - Referenziert wird der Bean-Name und der Name des Attributes
  - Getter / Setter werden implizit aufgerufen, falls notwendig.
- Beispiel:

```
<label>Username: </label>
<h:inputText id="username"
    value="#{loginCredentials.username}" />
```



## Scope, Context und Lifecylce

Orientation in Objects

- Bean Instanzen sind kontextuell
  - Beispiele für Kontext: Request, Http-Session
  - Beans können an einen kontextuellen Scope gebunden werden
- Container bestimmt anhand Scope den Lebenszyklus einer Bean
  - Erzeugung der Instanz bei erster Verwendung (Injektion)
  - Verwaltung der Instanzen
  - Zerstörung der Instanz am Ende des Kontexts

⇨ Der Scope bestimmt die Lebensdauer einer Bean-Instanz

## Verfügbare Scopes

Orientation in Objects

- Built-in Scopes (`javax.enterprise.context.*`)
  - @RequestScoped
  - @SessionScoped
  - @ApplicationScoped
  - @ConversationScoped
- Pseudo – Scopes
  - @Dependent
  - @Singleton (javax.inject.Singleton)

© 2014 Orientation in Objects GmbH


## Conversation Scope

Orientation in Objects

- Motivation:
  - Request Scope oftmals zu kurz
  - Session Scope zu langlebig
  - Beispiel: Assistent / Wizard
- Häufiges Antipattern: Session Scope für alles
  - Problem: Wann werden die Instanzen wieder aus der Session entfernt?
- Conversation Scope:
  - Beginn und Ende werden vom Entwickler festgelegt
  - Programmatische Demarkation über Conversation-Objekt

© 2014 Orientation in Objects GmbH

## Conversation Scope

Orientation in Objects

```
Conversation#begin()
Conversation#begin(String)
Conversation#end()
Conversation#getId()
Conversation#isTransient()
Conversation#getTimeout()
Conversation#setTimeout(long)
```



## Beispiel: Conversation Scoped Bean

Orientation in Objects

```
@Named
@ConversationScoped
public class RegisterBean implements Serializable {
   private static final long serialVersionUID = 1L;

   @Inject Conversation conversation;

   public String doRegister() {
        conversation.begin();
        ...
   }

   public String finishRegistration() {
        ...
        conversation.end();
   }
}
```

## Pseudo Scopes – Singleton

Orientation in Objects

- @Singleton
  - 1 Instanz pro Container
  - Expliziter Verzicht auf Proxy
    - ⇨ **Clients halten Referenz auf Instanz**
    - **Achtung bei Injektion in serialisierbare Beans (@SessionScoped,@ConversationScoped)**
    - **Lösungen:**
      - Transient – Schlüsselwort
      - writeReplace() und readResolve() implementieren
- @Singleton vs @ApplicationScopes:
  - ⇨ @ApplicationScoped in Web-Anwendungen
  - ⇨ @Singleton in Non-Web Anwendungen

© 2014 Orientation in Objects GmbH


## Pseude Scope - Dependent

Orientation in Objects

- Default-Scope
  - Wird verwendet, wenn Bean nicht explizit einen anderen Scope deklariert
- Lebenszyklus gebunden an Bean des Injection Point
  - Instantiierung mit Instantiierung der Bean
  - Zerstörung mit Zerstörung der Bean
- Achtung bei Verwendung von EL-Namen:
  - Jeder EL-Ausdruck erzeugt eine Instanz
- Kein Proxy
  - ⇨ Clients halten Referenz auf Instanz

© 2014 Orientation in Objects GmbH

## Producer Methoden

Orientation in Objects

```
@ApplicationScoped
public class Resources {

   @Prodcues @DailyQuote
   String getTodaysMessage() {
        if (isMonday()){
               return "Need more coffee today ...";
        }
        ...
   }

   @Produces
   Logger createLogger(InjectionPoint injectionPoint) {
        return Logger.getLogger(injectionPoint.getMember()
               .getDeclaringClass().getName());
   }
}
```



## Disposer Methoden

Orientation in Objects

- Aufruf am Ende des Lifecyles einer Bean
- Parameter und Qualifier müssen dem Typ des Producer entsprechen

Producer:

```
@Produces @UserDBConnection
public Connection openJDBCConnection() {
   Connection con = new ...
}
```

Disposer:

```
public void closeJDBCConnection(
   @Disposes @UserDBConnection Connection con) {

   con.close();
}
```

## Interceptor

Orientation in Objects

- Abfangen von Methodenaufrufen und Lifecycle Callbacks
- Einfacher AOP-Ansatz
  - Addressierung von Cross-Cutting-Concerns
- Typische Beispiele:
  - Transaktions-Demarkation
  - Authorisierung / Authentifizierung / Security
  - Logging / Auditing



## Interceptor - Binding

Orientation in Objects

```
@InterceptorBinding
@Retention(RUNTIME)
@Target({TYPE, METHOD, PARAMETER, FIELD })
public @interface Transactional {}
```

## Interceptor - Verwendung

Orientation in Objects

```
@ApplicationScoped
public class UserManager implements Serializable {

   ...

   @Transactional // Interceptor-Binding
   public User updateUser(User user) {
        ...
   }
}
```



## Interceptor - Implementierung

Orientation in Objects

```
@Transactional // Interceptor-Binding
@Interceptor
public class TransactionInterceptor {

   @Resource UserTransaction transaction; // not CDI-specific

   @AroundInvoke
   public Object manageTransaction(InvocationContext ctx)
        throws Exception{

        transaction.begin();   // not CDI-specific
        Object result = ctx.proceed();
        transaction.commit();  // not CDI-specific
        return result;
   }
}
```

## Interceptor - Aktivierung

Orientation in Objects

```
<beans xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/beans_1_0.xsd">

<interceptors>
  <class>
      de.oio.messageboard.messaging.TransactionInterceptor
  </class>
</interceptors>

</beans>
```



## javax.interceptor.InvocationContext

Orientation in Objects

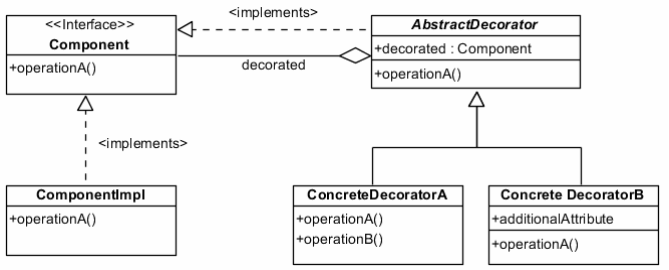
```
public interface InvocationContext {
 public Object getTarget();
 public Object getTimer();
 public Method getMethod();
 public Object[] getParameters();
 public void setParameters(Object[] params);
 public java.util.Map<String, Object> getContextData();
 public Object proceed() throws Exception;
}
```

- Ursprung: JSR-318 (EJB 3.1)
- Gemeinsamer Zugriff aller Interceptoren auf Instanz
  - pro Interception

## Decorators - Motivation

Orientation in Objects

- Interceptoren eignen sich für technische Querschnittsaspekte
  - Aber: Semantik der abgefangenen Methode ist Interceptor nicht zugänglich
- Decoratoren implementieren die Schnittstelle des dekorierten Objekts
  - Zugriff auf dekoriertes Objekt
- Implementierung des GoF Decorator Pattern






## Decorator - Implementierung

Orientation in Objects

```
@Decorator
public abstract class LoggingStore implements
   MessageStore {

   @Inject @Delegate @Any MessageStore delegate;
   @Inject Logger log;

   @Override
   public Message addMessage(Message message) {
       log.info("Message" + message));
       return delegate.addMessage(message);
   }
}
```

## Decorator - Aktivierung

Orientation in Objects

```
<beans xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="
      http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/beans_1_0.xsd">

   <decorators>
       <class>
              de.oio.messageboard.messaging.AuditedMessageStore
       </class>
   </decorators>

</beans>
```



## Events

Orientation in Objects

- Event-Objekt befördert "Nutzlast" von Producer zu Observer
- Beide Seiten der Übermittlung sind entkoppelt
- Spezialisierung der Events durch Qualifier
- Annotation Observer-Methode: @Observes

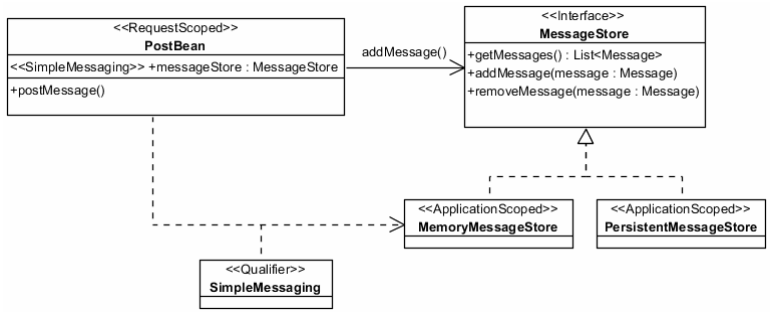
```
addMessage(@Observes @MessageEvent Message message) {…}
```

- Event-Producer lässt sich Event-Instanz injezieren:

```
@Inject @MessageEvent
private Event<Message> messageEvent;
```

## Bisher: Abhängigkeit von Sender und Empfänger

Orientation in Objects

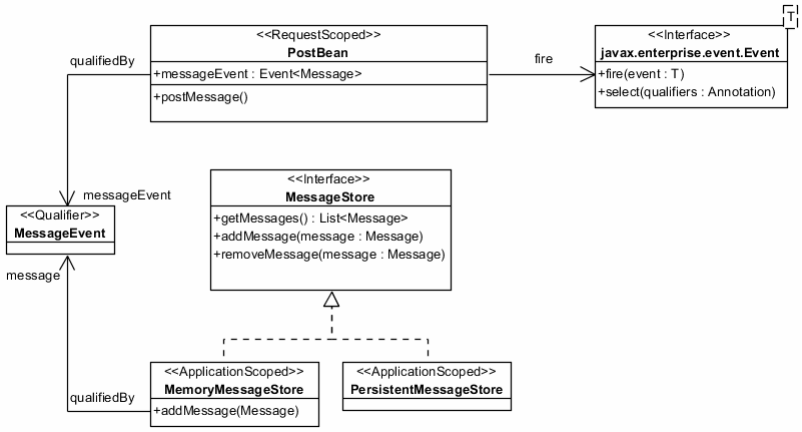



- Probleme:
  - Enge Kopplung
  - Sender-Implementierung muss zumindest Interface des Empfängers kennen
  - Mehrere Empfänger = Kopplung von Sender an mehrere Empfänger



## Lösung: Entkopplung durch Events

Orientation in Objects



## Event – Producer

Orientation in Objects

```
@RequestScoped
public class PostBean {

   @Inject @MessageEvent
   private Event<Message> messageEvent;

   public void postEventMessage() {
       Message message = new Message("Hello World!");
       messageEvent.fire(message);
   }
}
```



## Event - Observer

Orientation in Objects

```
@ApplicationScoped
public class EventBasedMessageStore implements
   MessageStore {

   public void addMessage(
       @Observes @MessageEvent Message message) {
       messages.add(message);
   }

}
```

## Eigene Stereotypen - Motivation

Orientation in Objects

- Problem: Annotation-Hell

```
@ApplicationScoped              // Scope
@Audited @Transactional         // Interceptor-Bindings
@DefaultStore @EventAware       // Qualifier
@Named                          // EL-Name
public class EventBasedMessageStore {...}
```

- Lösung: Eigener Sterotype

```
@Service   // Stereotype
public class EventBasedMessageStore {...}
```

© 2014 Orientation in Objects GmbH


## Eigene Stereotypen - Implementierung

Orientation in Objects

```
@Stereotype                 // Stereotyp Deklaration
@ApplicationScoped
@Audited
@Transactional
@DefaultStore
@EventAware
@Named
@Retention(RUNTIME)
@Target({TYPE})
public @interface Service {}
```

© 2014 Orientation in Objects GmbH

## @Model - Stereotype

Orientation in Objects

- Built-In Stereotype
- kann für Standard-JSF-Beans verwendet werden

```
@Named
@RequestScoped
@Stereotype
@Target({TYPE, METHOD})
@Retention(RUNTIME)
public @interface Model {}
```

- Verwendung:

```
@Model
public class AdminBean {
    ...
}
```



## Built-In Qualifier: @New

Orientation in Objects

- Erlaubt Injection von Dependent-Scope Objekten
- Es wird eine eigene Instanz für die Bean erzeugt

```
@ApplicationScoped
public class MessageStore { }

public class PostBean {
   @Inject private MessageStore simpleMessageStore;
   @Inject @New private MessageStore otherMessageStore;
}
```

## Built-In Qualifier: @Default und @Any

Orientation in Objects

- @Default:
  - Standard-Qualifier
  - Wird implizit angenommen, wenn Bean keinen anderer Qualifier definiert
- @Any:
  - Schliesst alle Qualifier ein
  - Ermöglicht "Sammel"-Injection-Point

```
@Inject
void resetStores(@Any Instance<MessageStore> stores) {
     for (MessageStore store: stores) {
            store.reset();
     }
}
```



## Qualifier mit Membern

Orientation in Objects

- Alternative zur Definition neuer aber ähnlicher Qualifier
- Beispiel:

```
@Qualifier
@Retention(RUNTIME)
@Target({TYPE, METHOD, PARAMETER, FIELD })
public @interface Restricted {
     UserRole value();
}

public class PostBean {
    @Inject @Restricted(UserRole.ADMIN)
    private MessageStore simpleMessageStore;
}
```

## Interceptors – Weiterführende Themen

Orientation in Objects

- Interceptor Bindings mit Membern
  - Analog zu Qualifiern
  - Spezialisierung von Interceptoren
- Mehrere Interceptor Bindings pro Typ möglich

```
@Transactional @Audited      public class MessageStore {...}
```

- Interceptor Inheritance

```
@InterceptorBinding
@Audited                     // Interceptor Binding
@Retention(RUNTIME)
@Target({TYPE, METHOD, PARAMETER, FIELD })
public @interface Transactional {...}
```



## Alternatives

Orientation in Objects

- Ermöglichen Austausch von Implementierungen
  - Deployment spezifisch
  - Typischer Anwendungsfall: Mock-Objekte
- Aktivierung in beans.xml (analog Decorator):
- Alternatives für Stereotypes möglich
  - z.B: "Schalter" um ganze Schicht / Modul gegen Mock zu ersetzen

## Alternatives - Beispiel

Orientation in Objects

- Implementierungen:

```
public interface MessageStore {...}

public class MemoryMessageStore implements MessageStore {...}

@Alternative
public class MessageStoreMock implements MessageStore {...}
```

- beans.xml

```
<alternatives>
    <class>
    de.oio.messageboard.messaging.MessageStoreMock
    </class>
</alternatives>
```



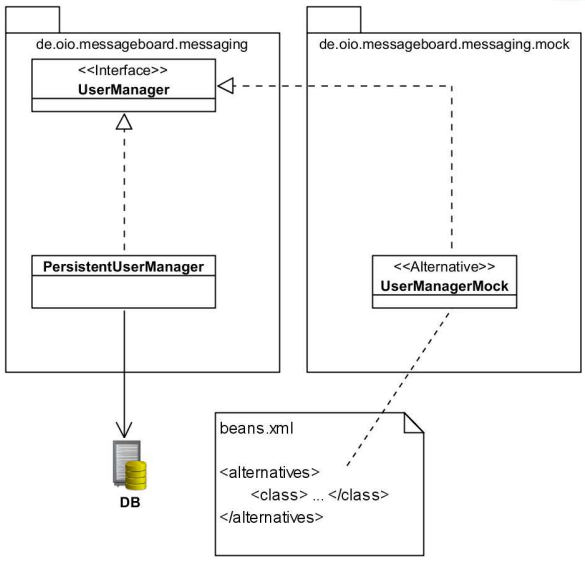
## CDI-Pojo oder EJB?

Orientation in Objects

- Funktionalität von CDI und EJB überschneiden sich teilweise
- EJB: Praktisch nur in Container ausführbar
  - Schwierige zu testen
  - ⇨ In-Container-Testing: Arquilian
- Dependency Injection und Context – Verwaltung
- Faustformel: *"EJB = CDI + Concurrency + Transactions"*

⇨ Wenn Enterprise – Features von EJB nicht benötigt werden, sollte CDI bevorzugt werden

## Alternatives - Struktur






## Programmatischer Zugriff: Injection Point

```
@Produces
public Logger createLogger(InjectionPoint injectionPoint)
{
    String name =
injectionPoint.getMember().getDeclaringClass().
           getSimpleName();
    return Logger.getLogger(name);
}

Type InjectionPoint#getType()
Set<Annotation> InjectionPoint#getQualifiers()
Bean<?> InjectionPoint#getBean()
Member InjectionPoint#getMember()
Annotated InjectionPoint#getAnnotated()
boolean InjectionPoint#isDelegate()
boolean InjectionPoint#isTransient()
```

## Portable Extensions

Orientation in Objects

- Extension:
  - Interagiert mit Container
  - Kann eigene Beans, Interceptor und Decorator bereitstellen
  - Eigene Scopes möglich
    - **Beispiel: Business-Scope Extension**
- CDI stellt SPI zur Schaffung von Extensions bereit
- Zugriff auf Lifecycle-Events:
  - BeforeBeanDiscovery
  - ProcessAnnotatedType
  - AfterBeanDiscovery
  - AfterDeploymentValidation
  - …
- BeanManager – Objekt
  - Injezierbar
  - Bietet programmatischen Zugriff auf Kernfunktionen des Containers
  - z.B: BeanManager.getBeans(…)

© 2014 Orientation in Objects GmbH


## Apache Delta Spike

Orientation in Objects

- Modul-Sammlung von Portable Extensions
- Module
  - Core
    - **Exception-Handling**
    - **Typesafe Config**
    - **Project-Stages**
  - Security
    - **Bindings zu 3rd-Party Security frameworks**
    - **Bietet Einheitliche Schnittstelle**
  - JPA
    - **Transactions ohne EJB**
  - JSF
    - **Web-Security Features**
    - **Typesafe Navigation**
  - Container-Control
    - **Boot/Shutdown des Containers in SE-Anwendunger**
    - **Kontrolle über Lifecycle der eingebauten Contexts**


© 2014 Orientation in Objects GmbH

## Testen von CDI-Anwendungen (1)

Orientation in Objects

- Unit-Tests
  - CDI-Container isoliert starten
  - Implementierungsabhängiges Feature:
  - Kein Zugriff auf EJB- oder Container Managed Services (z.B. JTA-Transaktionen)

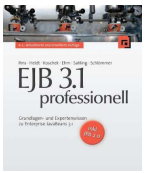
```
public class CdiBasedTest {

   static MessageBean mb;

   @BeforeClass
   public static void setUp() {
        WeldContainer weld = new Weld().initialize();
        mb = weld.instance().select(MessageBean.class).get();
   }

   @Test
   public void someTest() {…}
}
```



## Literaturhinweise

Orientation in Objects


- Workshop Java EE 7: Ein praktischer Einstieg in die Java Enterprise Edition mit dem Web Profile
- **ISBN-13:** 978-3864900471




- EJB 3.1 professionell: Grundlagen- und Expertenwissen zu Enterprise JavaBeans 3.1
- **ISBN-13:** 978-3898646123

## Links

- Spezifikation (JSR 299)
  - http://jcp.org/en/jsr/detail?id=299
- JBoss Weld (Referenzimplementierung)
  - http://seamframework.org/Weld
  - http://docs.jboss.org/weld/reference/latest/en-US/html/
- Apache OpenWebBeans
  - http://openwebbeans.apache.org/
- Apache DeltaSpike
  - http://incubator.apache.org/deltaspike/index.html
- JBoss Arquilian
  - http://arquillian.org/

© 2014 Orientation in Objects GmbH


Fragen ?

**Orientation in Objects GmbH**

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Orientation in Objects

# Vielen Dank für ihre Aufmerksamkeit !

**Orientation in Objects GmbH**

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de