holisticon
MEMBER OF SIGMA GROUP

Freundliche Eindringlinge

Moderne Web-Applikationen mit CDI

Norman Erck

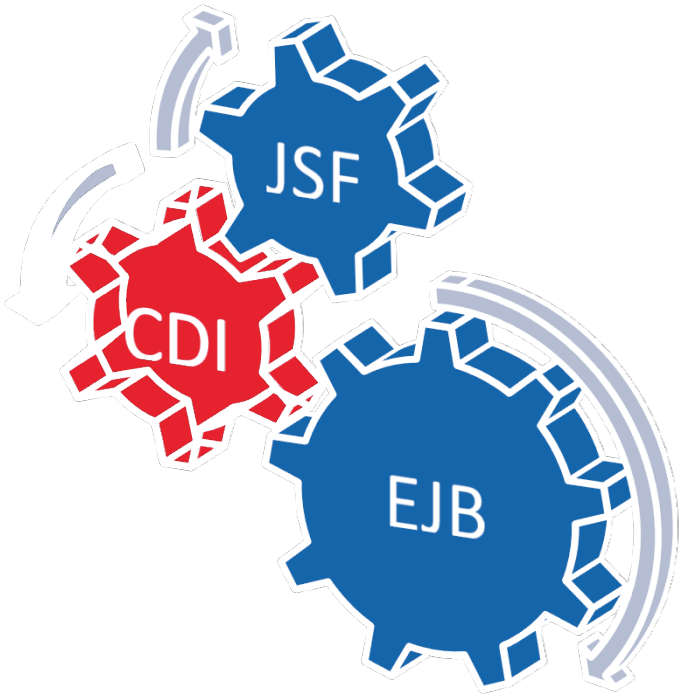
- Holisticon AG - Management- und IT-Beratung
  - Architektur
  - Agil/Projektmanagement
  - BPM/SOA
- Berater
  - Architektur
  - eBusiness
  - Web-Technologien
  - Portale

Inhalt

- Motivation
- Theorie
  - Neuerung
  - Bean-Verständnis
  - Konzepte
- Praxis
- Demo „Friendly Invaders“
- Ausblick Weld 2.0
- Zusammenfassung

Motivation - JSR 299

- Noch kein umfassendes DI Framework im Java EE Standard
- Standardisierung und Vereinfachung des Zusammenspiels von JSF und EJB
  - Präsentationsschicht
  - Geschäftslogik



Neuerungen – Was bietet CDI?

- DI-Mechanismus (Dependency Injection)
- Deklarative Verknüpfung von Beans
- Scopes
- Interzeptorenkonzept
- Dekoratoren
- Bean-Verwaltung zur Laufzeit

Bean-Verständnis - Beans

- JSF, EJB, Spring & CDI
  - Eine Bean ist ein Objekt, dessen Lebenszyklus vom Container verwaltet wird.
- Managed Beans
  - Parameterloser Konstruktor | mit @Inject annotiert
  - Lebenszyklus & Beziehungen verwaltet
- Session Beans
  - Managed Beans + Lebenszyklus, Statusmanagement- & Nebenläufigkeitsmodell
  - Stateless Session Bean und Singleton Session Bean „scopeless“
  - Stateful Session Beans beliebiger Scope

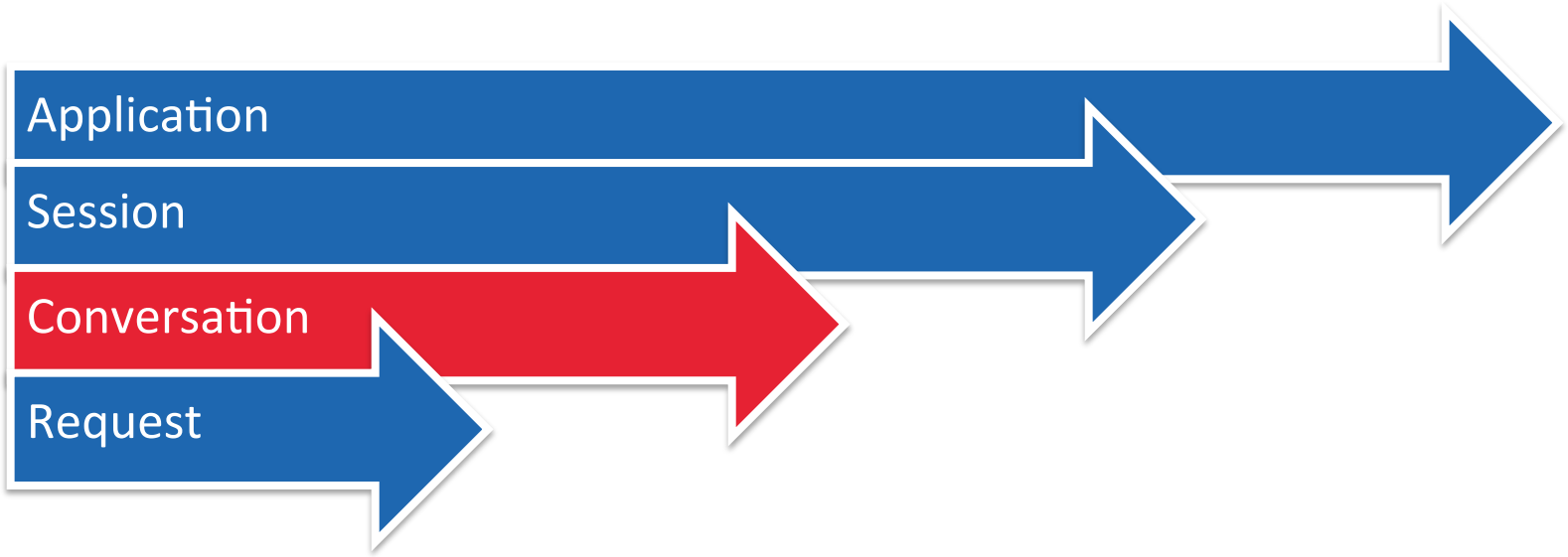
Bean-Verständnis - Nicht Beans

- Message-Driven Beans
- Interzeptoren
- Servlets
- JAX WS Services
- JSP Tag Handler
- Tag Library Event Listener

- Können Beans injiziert bekommen

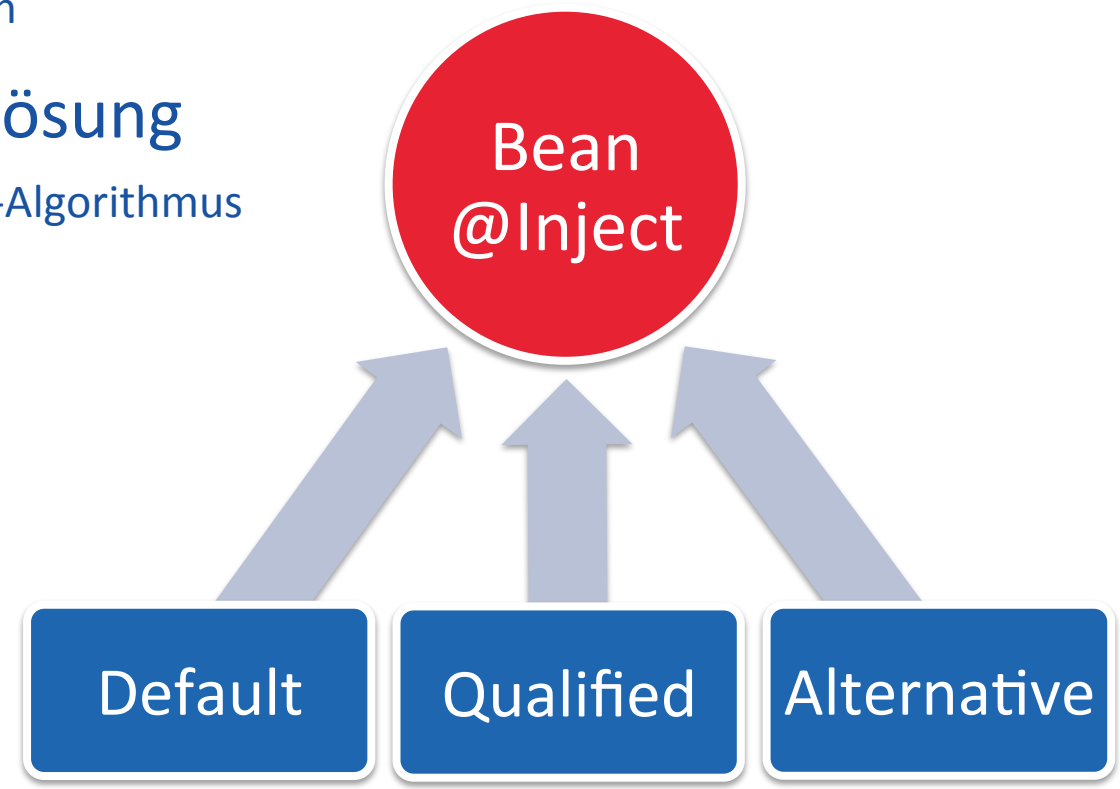
Konzepte

- Container, Kontext & Scope
  - Verwaltung Lebenszyklus
  - Beziehungen zwischen verwalteten Objekten
  - Sichtbarkeit



Konzepte

- Dependency Injection
  - Nicht neu
  - Java-EE-Applikationen
- Typsichere Auflösung
  - Typesave-Resolution-Algorithmus
  - Qualifier
  - Alternativen



Konzepte

- Lose Kopplung
  - Client & Server
  - Lebenszyklus
  - Produzenten & Konsumenten
- Bean-Namen
  - EL-Namen
  - *Keine* JSF Managed Beans
- Interceptor-Binding & Dekoratoren
- SPI
  - Portable Integration von Erweiterungen

Praxis

- Weld
  - Referenzimplementierung JBoss
- Deployment-Deskriptor
  - WAR: WEB-INF/beans.xml
  - JAR: META-INF/beans.xml
  - Kann leer sein

Praxis

- „Friendly Invaders“
- Werkzeugkiste
  - DI
  - Qualifier
  - Alternativen
  - EL-Namen
  - Scopes
  - Stereotypen
  - Events
  - Interceptor-Binding
  - Dekoratoren
  - Produzenten

Dependency Injection

```
@Stateless
public class BlogServiceBean implements BlogService {
  ...
}


public class InvasionController {
   @Inject
   private BlogService blogService;
   ...
}
```

Qualifier

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

```
@Holisticon
@Stateless
public class HolisticonBlogServiceBean
    implements BlogService {
  ...
}
```

Qualifier

```
public class InvasionController {

  @Inject @Holisticon
  private BlogService blogService;

  ...
}
```

- Mehrere Implementierungen?
  - @Default
  - Sonst: Start verweigert

```
WELD-001409 Ambiguous dependencies for type [BlogService] with
qualifiers [@Default] at injection point ...
```

Alternativen

```
@Alternative
public class BlogServiceMock implements BlogService,
    Serializable{

  ...
}
```

- Aktivierung in beans.xml

```
<alternatives>
  <class>
    de.holisticon.blog.fiendlyInvaders.service.
      mock.BlogServiceMock
  </class>
</alternatives>
```

EL-Namen

- Bean CDI Kontext hinzufügen

```
@Named
public class InvasionController{
  @Inject @Holisticon
  private BlogService blogService;
  ...
}
```

```
<ui:repeat var="blogArticle“
  value="#{invasionController.blogArticles}“>
  #{blogArticle.title}
</ui:repeat>
```

Scopes

```
@Named
@SessionScoped
public class InvasionController implements Serializable {
   private static final long serialVersionUID = 1L;
   @Inject @Holisticon
   private BlogService blogService;
   ...
}
```

- Serialisierbar
  - SessionSoped & ConversationScoped
  - Sonst: Start verweigert

```
WELD-000072 Managed bean declaring a passivating scope must be
passivation capable.
```

Scopes

```
@Stateful
@ConversationScoped
public class CheckOutServiceBean implements ... {

  @Inject
  private Conversation conversation;
  private KudosCart kudosCart;

  public KudosCart createKudosCart() {
    kudosCart = new KudosCart();
    // if no conversation is "open“
    if (conversation.isTransient())
      conversation.begin();
    return kudosCart;
  }

  public void checkout() {
    kudosCart.getCart().clear();
    conversation.end();
  }
}
```

Stereotypen

- Annotationen gebündelt

```
@SessionScoped
@Named
@Stereotype
@Retention(RUNTIME)
@Target(TYPE)
public @interface NSS {}


@NSS
public class InvasionController implements Serializable{
   ...
}
```

Events

```
@Qualifier
@Target({FIELD, PARAMETER})
@Retention(RUNTIME)
public @interface Invaded {}
```

- Producer klassifizieren
  - Filtern ermöglichen

```
@Inject @Invaded
private Event<BlogArticle> blogArticleEvent;

public String giveKudos(BlogArticle blogArticle, int amount{
  blogArticleEvent.fire(blogArticle);
  ...
}
```

Events

- Observer lauschen
  - InvasionTweeter
  - InvasionMailer
  - …

```
@RequestScoped
public class InvasionTweeter{

  public void afterBlogIntrude(
    @Observes @Invaded BlogArticle blogArticle) {
    ...
  }

}
```

Events

- Observers + Producers = Observer Pattern?
  - Jein
  - Producer von Observer & Observer von Producern
  - Verzögertes Aufrufen
  - Filtern/Auswählen von Events (z.B. @Invaded vs. @Any)

Interceptor-Binding

```
@InterceptorBinding
@Retention (RUNTIME)
@Target ({METHOD, TYPE})
public @interface LoggerBinding {}


@LoggerBinding @Interceptor
public class LoggingInterceptor implements Serializable {

   private static final long serialVersionUID = 1L;
   @Inject
   private Logger log;

   @AroundInvoke
   public Object onMethodCall(InvocationContext context)
       throws Exception {
     ...
   }
}
```

Interceptor-Binding

```
@Stateful
@ConversationScoped
public class CheckOutServiceBean implements ... {
  ...
  @LoggerBinding
  public void checkout() {
    kudosCart.getCart().clear();
    conversation.end();
  }
}
```

- Aktivierung in beans.xml

```
<interceptors>
  <class>
    de.holisticon.blog.friendlyInvaders.
      interceptor.LoggingInterceptor
  </class>
</interceptors>
```

Dekoratoren

```
@Decorator
public abstract class CheckoutServiceDecorator
  implements CheckoutService, Serializable {
  ...
  @Inject @Delegate
  private CheckoutService checkoutService;

  public void giveKudos(
      BlogArticle blogArticle, float amount) {
    if (blogArticle.getAuthor().getName().
        endsWith("Erck")){
      logger.info("No Kudos in the form of money.");
    }else{
      checkoutService.giveKudos(blogArticle, amount);
    }
  }
}
```

Dekoratoren

- Aktivierung in beans.xml

```
<decorators>
  <class>
    de.holisticon.blog.friendlyInvaders.
      decorator.CheckoutServiceDecorator
  </class>
</decorators>
```

- Interzeptoren vs. Dekoratoren
  - Jeder Java-Typ
  - Technische Aspekte
  - Konkreter Java-Typ
  - Kennen Semantik
  - Geschäftslogik

Produzenten

```
public class HcoLoggerFactory {

  @Produces @HcoLogger
  public Logger produceLogger(
      InjectionPoint injectionPoint){
    return LoggerFactory.getLogger(
      injectionPoint.getMember().getDeclaringClass().
        getName());
  }
}
```

- Default Scope: @Dependent
  - Scope der Zielklasse → Mehrere Instanzen möglich

Produzenten

```
@Decorator
public abstract class CheckoutServiceDecorator
  implements CheckoutService, Serializable {
  ...
  @Inject @HcoLogger
  private Logger logger;
  ...
}
```

- Einsatzgebiete
  - Variierender Typ zur Laufzeit
  - Individuelle Initialisierung

Beispielapplikation

Ausblick Weld 2.0

- Verbesserung des Speichermanagements
  - Speicherbedarf um 50% bis 80% reduziert
- Gleichzeitige Bean Validierung und Bean Deployment
  - Multi-Core CPUs werden genutzt
  - Ladezeit verringert
- Bug Fixes

Zusammenfassung

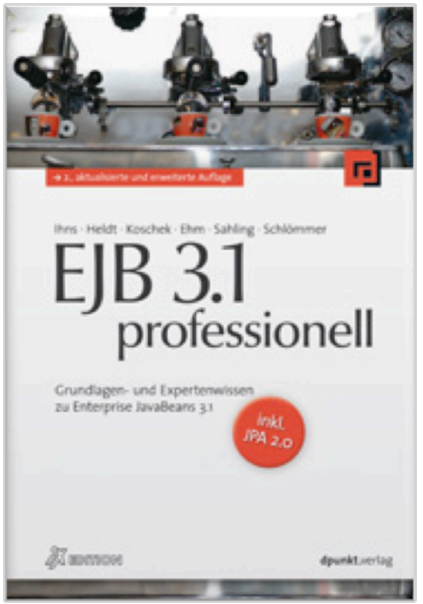
- DI Mechanismus (Dependency Injection)
  - @Inject
- Deklarative Verknüpfung von Beans
  - @Qualifier, @Alternative, @Stereotype
- Neue Scopes
  - @ConversationScope & @Dependent
- Interzeptorenkonzept
  - @InterceptorBinding
- Dekoratoren
  - @Decorator & @Delegate
- Beanverwaltung zur Laufzeit
  - @Producer

„Friendly Invaders“ & Tonspur

http://blog.holisticon.de

Darüber hinaus...

„Grundlagen- und Expertenwissen zu Enterprise Java Beans 3.1“



Vielen Dank!


holisticon
MEMBER OF SIGMA GROUP
Holisticon AG
Friedensallee 120
22763 Hamburg
www.holisticon.de
Mobile: +49 176 1616 9011
Office: +49 40 5074 2722
Fax: +49 40 5074 2724
E-Mail: norman.erck@holisticon.de
M. Sc. (eBusiness)
Norman Erck
Consultant