Redpill Linpro

# CDI .

## *Standardized Dependency Injection in JEE6*


*jens.augustsson@redpill-linpro.com*

# CDI.

1. *What it is*
2. *Features*
3. *Advices*

Redpill Linpro

# Jens X Augustsson.

- jBPM
- Hibernate
- JBoss AS 6
- Seam
- Spring Core

*jens.augustsson@redpill-linpro.com*

# next...

## 1. What it is

Redpill Linpro

# Example.

```
public class TextTranslator {

   private final SentenceParser sentenceParser;
   private final Translator sentenceTranslator;

   @Inject
   public TextTranslator(SentenceParser sentenceParser,
                         Translator sentenceTranslator) {

      this.sentenceParser = sentenceParser;
      this.sentenceTranslator = sentenceTranslator;
   }

   public String translate(String text) {
      StringBuilder sb = new StringBuilder();
      for (String sentence: sentenceParser.parse(text)) {
          sb.append(sentenceTranslator.translate(sentence));
      }
      return sb.toString();
   }
}
```

Redpill Linpro

# Example - 2.

```
@Stateless

public class SentenceTranslator implements Translator {

   public String translate(String sentence) { ... }

}
```

```
public class SentenceParser {

   public List<String> parse(String text) { ... }

}
```

- **Injection of:** Managed Bean, EJB session beans
- **Injection to:** MDB, interceptor, Servlet, JAX-WS SE, JSP (tag h / ev lis)

Redpill Linpro

# new Instance(), anyone?.

- **Dependencies**
- **Container managed**
- **Configuration**
- **Life cyle**

**...**

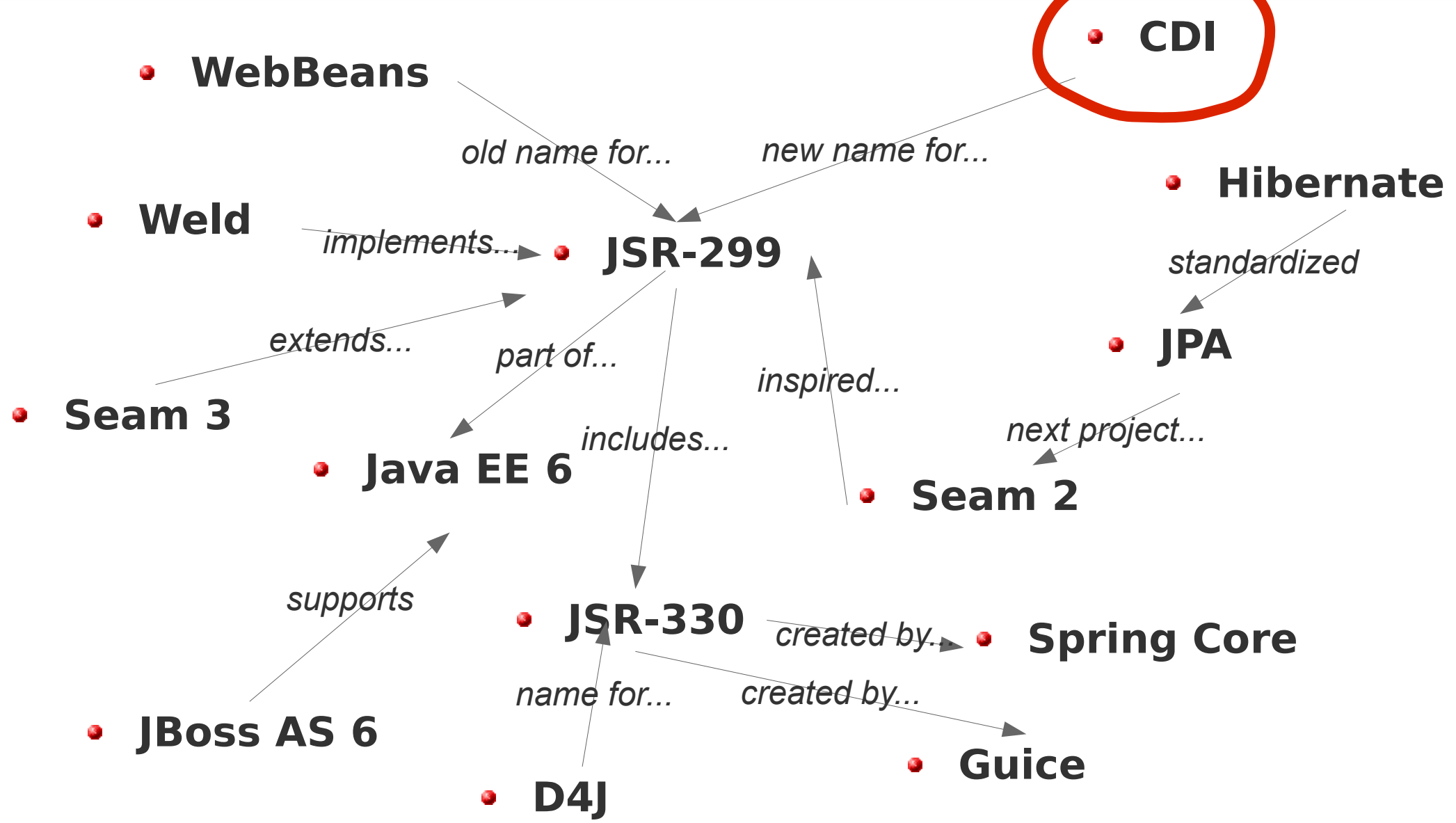
«The fundamental choice is between Service Locator and Dependency Injection»

(Martin Fowler)

- «Inversion of Control» ... «The Hollywood principle»

Redpill Linpro

# «but I've heard...»



# 2. CDI Features

Redpill Linpro

# Injection points.

- **Class constructor**

```
public class Checkout {
   private final ShoppingCart cart;

   @Inject
   public Checkout(ShoppingCart cart) {
      this.cart = cart;
   }
}
```

- **Initializer method**

```
public class Checkout {
   private ShoppingCart cart;

   @Inject
   void setShoppingCart(ShoppingCart cart) {
      this.cart = cart;
   }
}
```

- **Direct field**

```
public class Checkout {
   private @Inject ShoppingCart cart;
}
```

Redpill Linpro

# Injectable bean types.

- **In a JEE module with a /META-INF/beans.xml**
- **A user-defined class or interface**

declare

```
public class CreditCardPaymentService
      implements PaymentService {
   ...
}
```

use

```
...
@Inject
CreditCardPaymentService ps;
...
```

**or?**

```
...
@Inject
PaymentService ps;
...
```

**...«there can be only one»...**

Redpill Linpro

# Non-default qualifiers.

- **Your custom annotations...**

CDI anno →
JavaSE annos →

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

- **...links a declared injection point...**

Your anno →

```
...
@Inject @Preferred PaymentService ps;
...
```

- **...to a qualified bean**

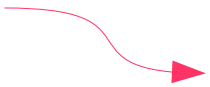
```
@Preferred
public class CreditCardPaymentService implements PaymentService {
   public void process(Payment payment) { ... }
}
```

Redpill Linpro

# Qualifiers with members.

- **Your custom annotation and enumeration...**

Your enum



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

- **...links a declared injection point...**

```
...
private @Inject @PayBy(CREDIT) PaymentProcessor checkPayment;
...
```

- **...to a qualified bean**

```
@PayBy(CREDIT)
public class CreditCardProcessor implements PaymentProcessor {
   public void process(Payment payment) { ... }
}
```

# Producer methods.

- **Run time qualifier**

```
public PaymentAction {
    @Inject @Preferred PaymentService userPaymentService;
    ...
}
```

```
public AnyClass {

    @Inject
    User user;

    @Produces @Preferred
    public PaymentService getUserPaymentService() {
       return user.getPaymentServices().get(0);
   }
}
```

Redpill Linpro

# JEE comp env resources.

CDI anno

JEE anno

```
@Produces @Resource(lookup="java:global/env/jdbc/CustomerDatasource")
@CustomerDatabase Datasource customerDatabase;
```

Your anno

```
...
@Inject @CustomerDatabase Datasource ds;
...
```

```
@Produces @PersistenceUnit(unitName="CustomerDatabase")
@CustomerDatabase EntityManagerFactory customerDatabasePersistenceUnit;
```

```
@Produces @WebServiceRef(lookup="java:app/service/Catalog")
Catalog catalog;
```

```
@Produces @EJB(ejbLink="../their.jar#PaymentService")
PaymentService paymentService;
```

# JEE integration.

- **Built-in beans**
  - the current JTA *UserTransaction*
  - a *Principal* representing the current caller identity
  - the default Bean *ValidationFactory*
  - a *Validator* for the default ValidationFactory

Redpill Linpro

# Scopes and Contexts.

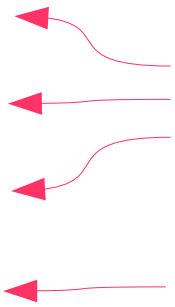
- **First: The ”dependent” pseudo-scope**

```
public class Calculator { ... }
                =
@Dependent public class Calculator { ... }
```

- **Scope determines...**
  - When an instance is created
  - When an instance is destroyed
  - Which injected references refer to an instance
  - CDI features an extensible context model

# Scopes misc.

- **Also: Built-in scopes**
  - @RequestScoped
  - @SessionScoped
  - @ApplicationScoped
  - @ConversationScoped



*JEE defined*

*Defined by you*

```
@SessionScoped public class Calculator { ... }
```

- **The @New qualifier**

```
public class PaymentCalc {

   @Inject Calculator calculator;
   @Inject @New Calculator newCalculator;

}
```

Redpill Linpro

# Conversation Scope.

```
@ConversationScoped @Stateful
public class OrderBuilder {
   private Order order;
   private @Inject Conversation conversation;
   private @PersistenceContext EntityManager em;

   public Order createOrder() {
      order = new Order();
      conversation.begin();
      return order;
   }

   public void addLineItem(Product product, int quantity) {
      order.add(new LineItem(product, quantity));
   }

   public void saveOrder() {
      em.persist(order);
      conversation.end();
   }

   @Remove
   public void destroy() {}
}
```

# Interceptors.

- **business method interception**
- **lifecycle callback interception**
- **timeout method interception (ejb3)**

# Interceptors - 2.

- **Binding**

CDI anno

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

Your anno

```
public class ShoppingCart {
   @MySecurity public void checkout() { ... }
}
```

# Interceptors - 3.

- **Implementation - business method:**

```
@MySecurity @Interceptor
public class MySecurityInterceptor {

   @AroundInvoke
   public Object manageSecurity(InvocationContext ctx) throws Exception { ... }

}
```

- **Implementation - lifecycle:** @PostConstruct, @PreDestroy...

- **Implementation - timeout:** @AroundTimeout

Redpill Linpro

# Decorators .

- **Interceptors capture orthogonal application concerns**
- **The reverse is true of decorators**

```
@Decorator
public abstract class LargeTransactionDecorator implements Account {

   @Inject @Delegate @Any Account account;
   @PersistenceContext EntityManager em;

   public void withdraw(BigDecimal amount) {
      account.withdraw(amount);
      if ( amount.compareTo(LARGE_AMOUNT)>0 ) {
         em.persist( new LoggedWithdrawl(amount) );
      }
   }

   public void deposit(BigDecimal amount);
      account.deposit(amount);
      if ( amount.compareTo(LARGE_AMOUNT)>0 ) {
         em.persist( new LoggedDeposit(amount) );
      }
   }

}
```

# Events.

- **Become observable....**

```
@Inject @Updated Event<Document> documentEvent;
...
document.setLastModified(new Date());
documentEvent.fire(document);
```

Optional qualifier

- **Become observer....**

```
public void handleDocs(@Observes @Updated Document document) { ... }
```

- **Conditional observations...**

```
public void handleDocs(@Observes(during = AFTER_SUCCESS) @Updated Document doc) {
 ..
 }
```

# Stereotypes.

- **Declare**

EJB →
your →
CDI →
JSE →

```
@Stateless
@Transactional(requiresNew=true)
@Secure
@Stereotype
@Target(TYPE)
@Retention(RUNTIME)
public @interface BusinessLogic {}
```

- **Use**

```
@BusinessLogic
public class UserService { ... }
```

- **Predefined by CDI: @Model**

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

# 3. Advices

Redpill Linpro

# Personal experiences.

- **Good stuff**
  - Seam improvement – no outjection, method-time injection etc.
  - Great for use with other frameworks – like jBPM
  - XML-hell is /actually/ gone
- **But be careful**
  - Start off with managed beans – switch when needed
  - Annotations are adjectives (@Preferred), not nouns (@CreditCardPayment)
  - Avoid injection from ”thinner” context – use @Dependent
  - Weld documentation not finished
  - Avoid ”upgrade” JBoss AS 5.x
  - XML Configuration in Seam 3 Module
  - Annotation Frustration…

Redpill Linpro

# Get started!.

- **In JBoss 6.0.0.Final (Weld 1.1.0.Beta2)**
- **In GlassFish Server 3.1 (Weld 1.1.0.Final)**
- **Embed Weld in Tomcat, Jetty... Android almost :-)**
- **Generate CDI project using Seam Forge or M2Eclipse**
- **And read more!**
  - Dan Allens slideshare: *Google "Dan Allen slideshare cdi"*
  - Gavin King and Bob Lee flamewar*: Google "Gavin King Bob Lee jsr*"

Redpill Linpro

# End.

- **jens.augustsson@redpill-linpro.com**