

The New Java EE

CDI - the re-invented component model for Java EE 6. Better than Spring?

Johan Eltes | johan.eltes@callistaenterprise.se | Cadec 2010-01-20



Are you a Java or XML developer?

```
spring-system-conf-j2ee-persistence.xml

</property>
<property name="objectType">
  <value>javax.sql.DataSource</value>
</property>
</bean>

<bean id="ldDataSource" class="com.foo.ldsupport.DynamicPropertyManagerDataSourceProxy">
  <property name="dataSource">
    <ref local="websphereDataSourceProxy" />
  </property>
</bean>

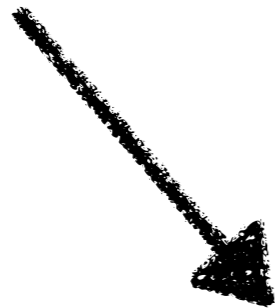
<!-- WebSphere XA-aware proxy for XQ DataSources which intercepts
close calls and issues a rollback if not within an XA transaction
-->
<bean id="websphereDataSourceProxy" class="com.foo.daosupport.WebsphereDataSourceProxy">
  <property name="userTransaction">
    <ref local="userTransactionBean" />
  </property>
  <property name="activeTxDataSource">
    <!-- DataSource implementation for Sharable datasource -->
    <bean class="org.springframework.jndi.JndiObjectFactoryBean">
      <property name="jndiName">
        <value>java:comp/env/jdbc/poc/sharable</value>
      </property>
    </bean>
  </property>
  <property name="inactiveTxDataSource">
    <!-- DataSource implementation for Unsharable datasource. -->
    <bean class="org.springframework.jndi.JndiObjectFactoryBean">
      <property name="jndiName">
        <value>java:comp/env/jdbc/poc/unsharable</value>
      </property>
    </bean>
  </property>
</bean>

<!-- The UserTransaction implementation for in-container -->
<bean id="userTransactionBean" class="org.springframework.jndi.JndiObjectFactoryBean" >
  <property name="jndiName">
    <value>java:comp/UserTransaction</value>
  </property>
</bean>

<!-- LobHelper for CLOB support. -->
<bean id="lobHandler" class="org.springframework.jdbc.support.lob.OracleLobHandler">
  <property name="nativeJdbcExtractor"><ref local="webSphereJdbcExtractor"/></property>
</bean>
<bean id="webSphereJdbcExtractor"
  class="org.springframework.jdbc.support.nativejdbc.WebSphereNativeJdbcExtractor" />
</beans>
```

Spring wasn't built bottom up for noxml

Spring wasn't built bottom up for noXML



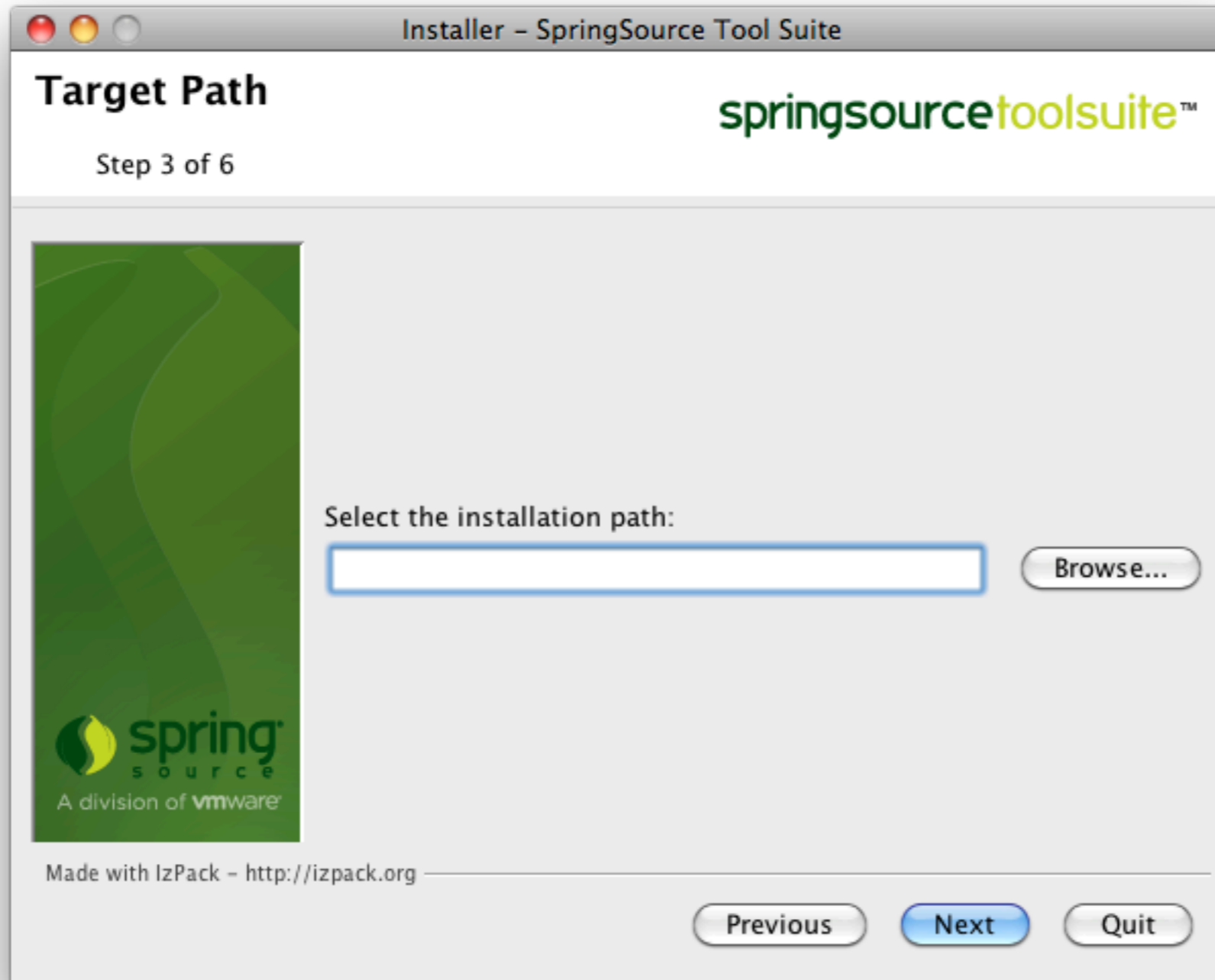
You need to understand what can be done using JavaConfig / annotations and when to step down to XML.



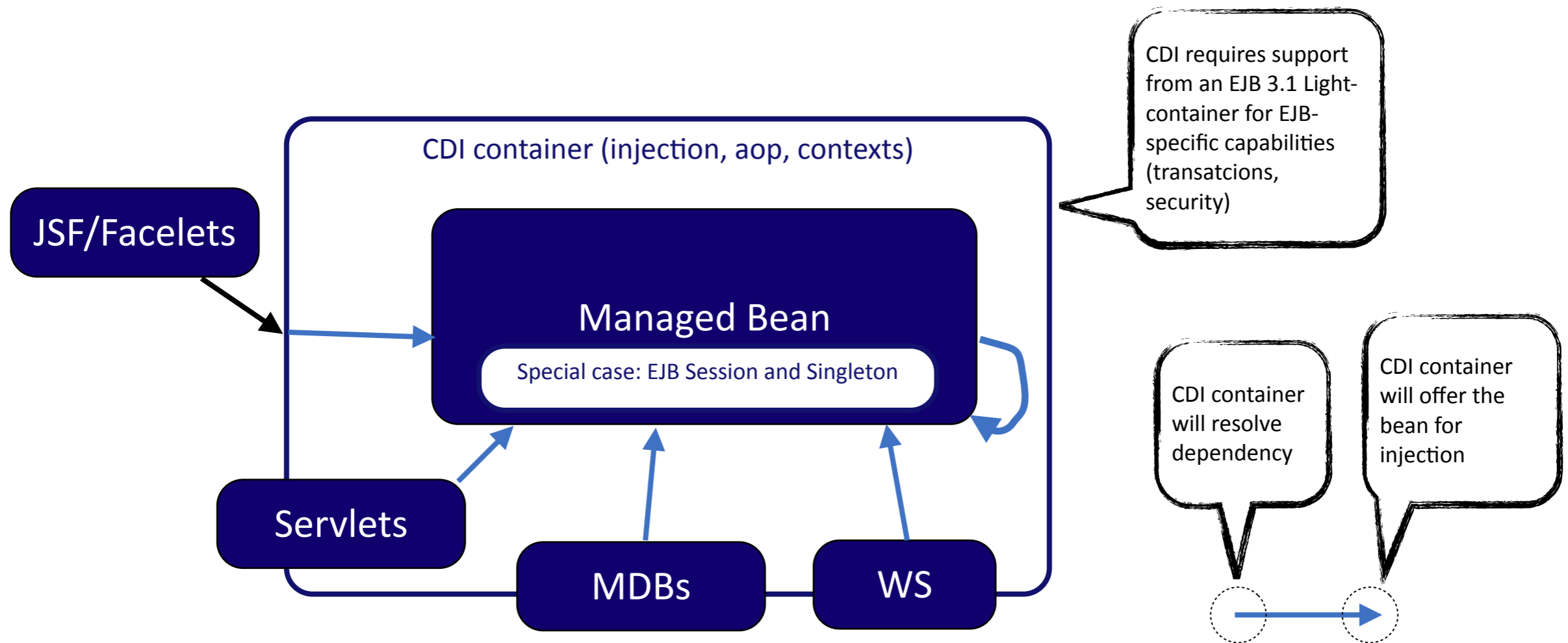
There are so many options in Spring that you need a 100 page company-best-practice guide so that developers can read each others code.



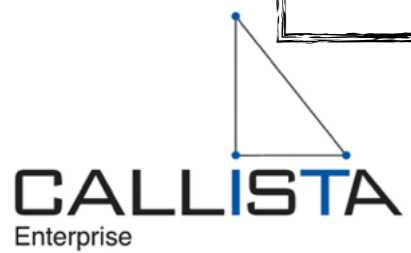
Complex tools require tool suites



...and moving to a very slick DI-based component model: CDI



CDI consolidates JSR 250 (common annotations), JSR-330 (javax.inject) and "EJB 3.1 light" into a unified "POJO" component model.



So, let's have a look at CDI!

- Use-cases for CDI

- UC 1: Basic DI for Unit testing
- UC 2: Factories and qualifiers
- UC 3: Integration testing in JUnit
- UC 4: AOP with interceptors
- UC 5: Loose coupling with events
- UC 6: Handling of scopes
- UC 7: Extensions

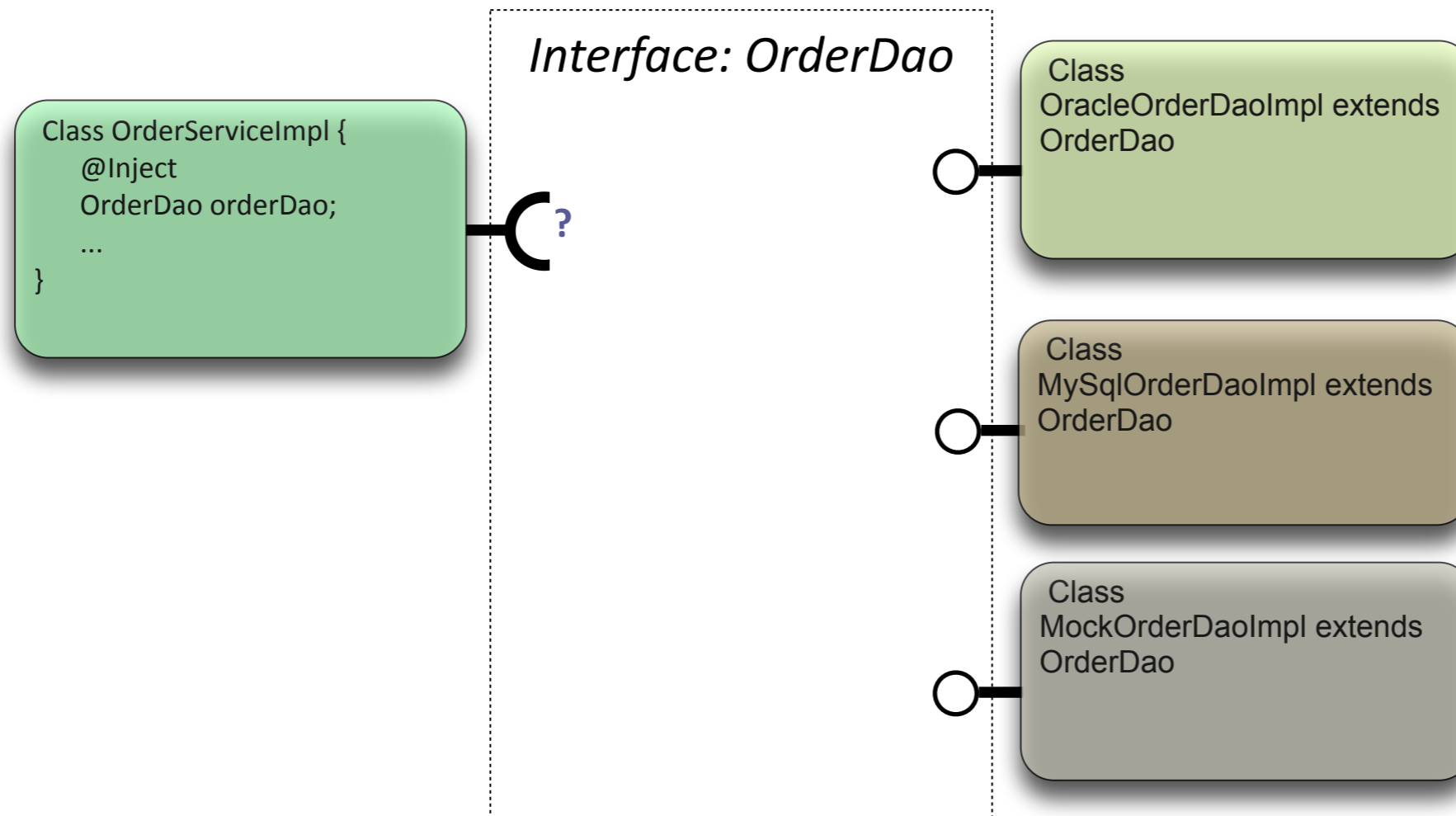
All demo code for the use-cases are available for public checkout at this svn repo:

<https://svn.callistaenterprise.se/public/Cadec/Cadec2010/cdi/trunk>

Annotations, annotations, annotations... @Inject, @Produces, @Qualifier, @Alternative, @Stateless, @Conversation



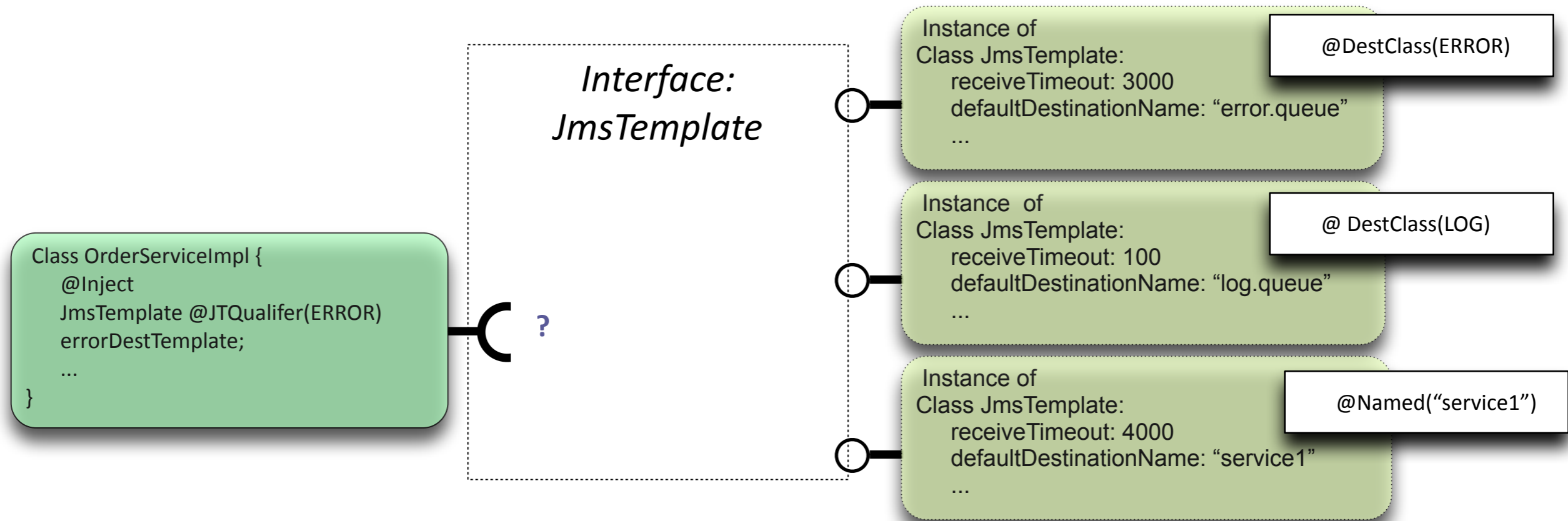
UC 1: Basic DI for Unit testing



@Inject is by default auto-injection by type



UC 2: Factories and qualifiers



```
Class  
JmsTemplateFactory {  
    @Producer @DestClass(ERROR) JmsTemplate getErrorDestTemplate() {}  
    @Producer @DestClass(LOG) JmsTemplate getLogDestTemplate() {}  
    @Producer @Named("invoice") JmsTemplate getInvoiceDestTemplate() {}  
}
```

With `@Produces` we can create multiple beans of the same type
With `@Qualifier`-annotations, we select which bean to inject



UC 3: Integration testing with JUnit



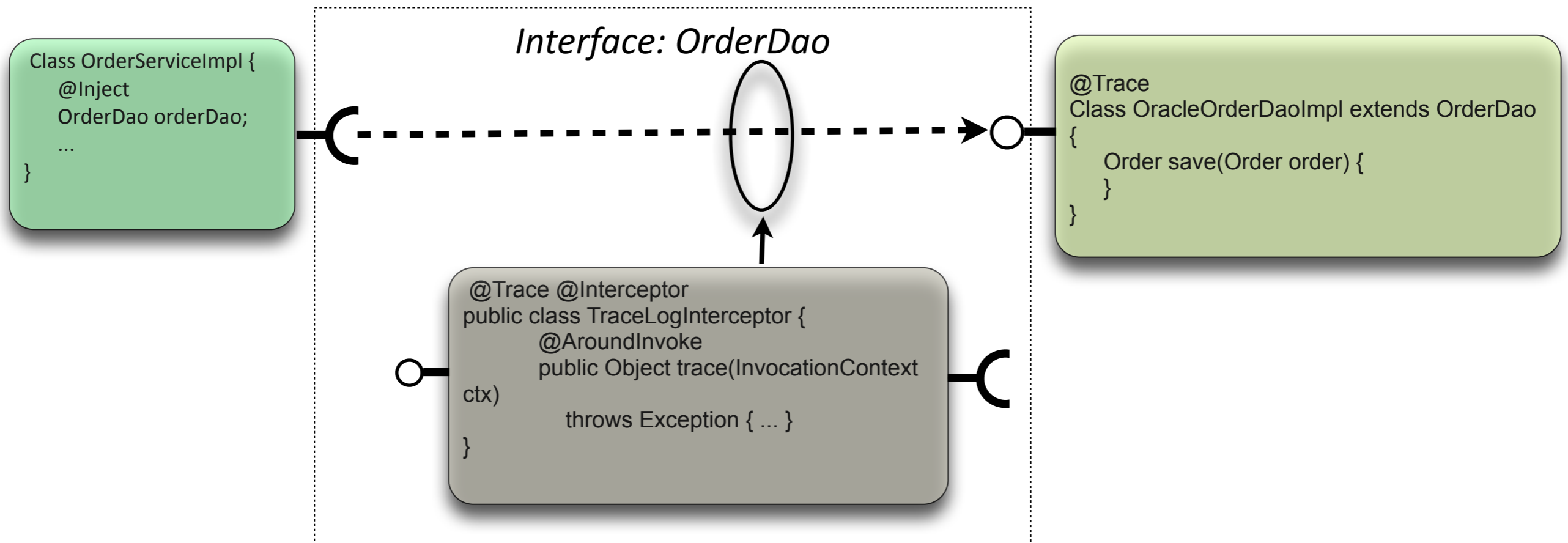
How much weight has EJB lost??

Demo:
JUnit TestCase->TestEjb->Ejb->Dao->JPA

...with embeddable EJB container



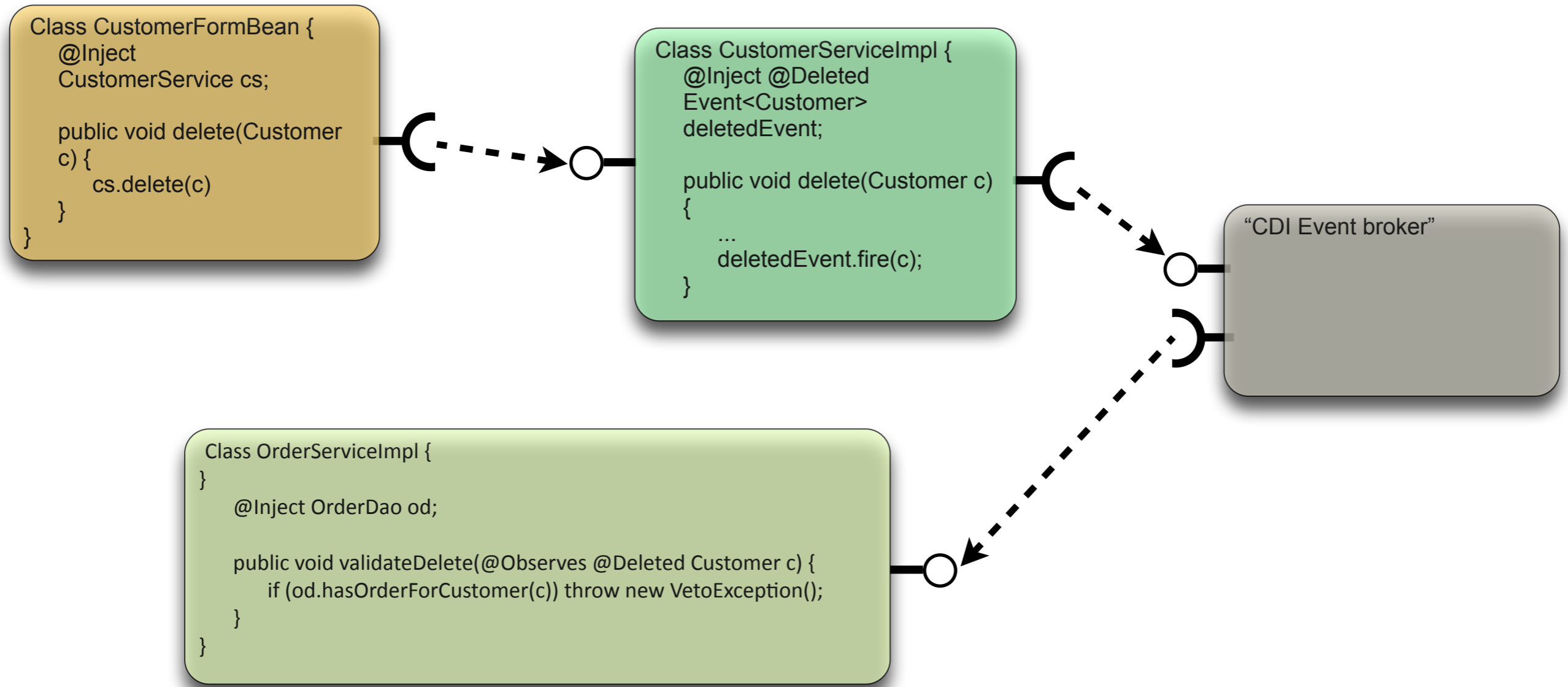
UC 4: Cross-cutting concerns



Sample: Trace entry and exit of method invocations in all classes annotated with @Trace



UC 5: Events



Order service listens to customer deletion events to prevent deletion of customers with orders.



UC 6: Declarative scope management

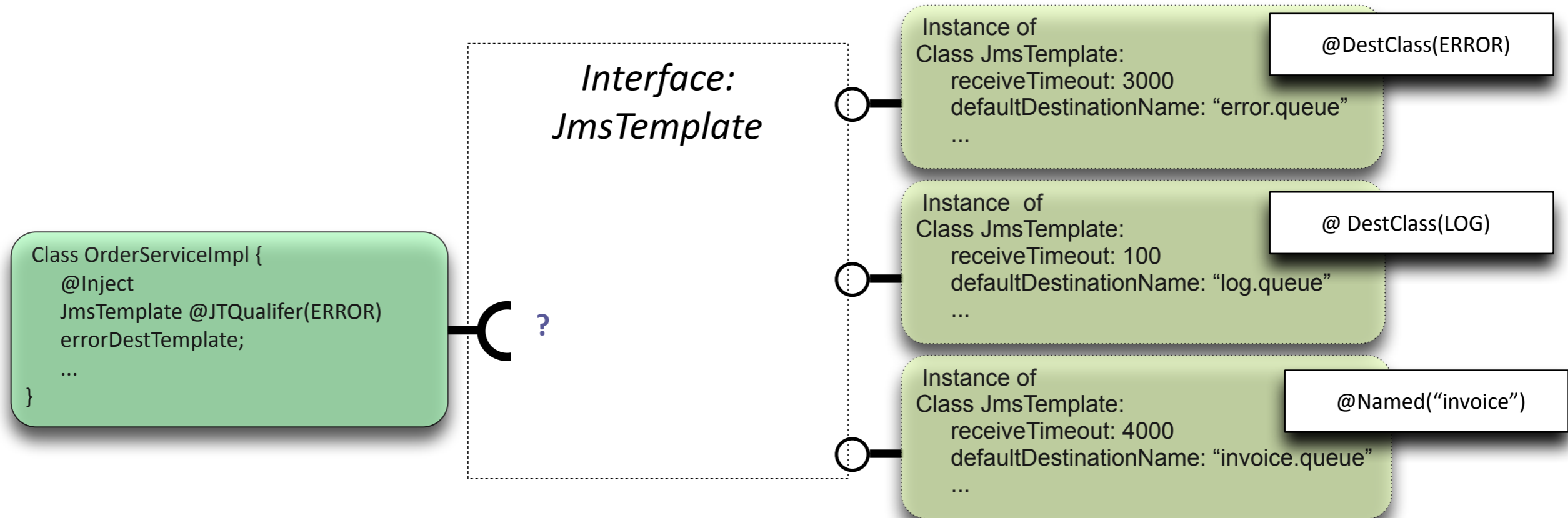
- `ejb Session`
 - `ejb Singleton`
 - `ApplicationScoped`
 - `RequestScoped`
 - `SessionScoped`
 - `ConversationScoped`
- Additional scopes may be added via extension api (“SPI”)

CDI handles the scoping / context aspects of EJBs as well (without an EJB container)

Use-case 5: Contextual services



UC 7: Extensions - Sample: config file



```

Class
JmsTemplateFactory {
    @Producer @DestClass(ERROR) JmsTemplate getErrorDestTemplate() {}
    @Producer @DestClass(LOG) JmsTemplate getLogDestTemplate() {}
    @Producer @Named("invoice") JmsTemplate getInvoiceDestTemplate() {}
}
    
```

```

error_queue.timeout=3600
error_queue.destname=error.queue
    
```

Sample: Extension for file-based configuration of @Producers

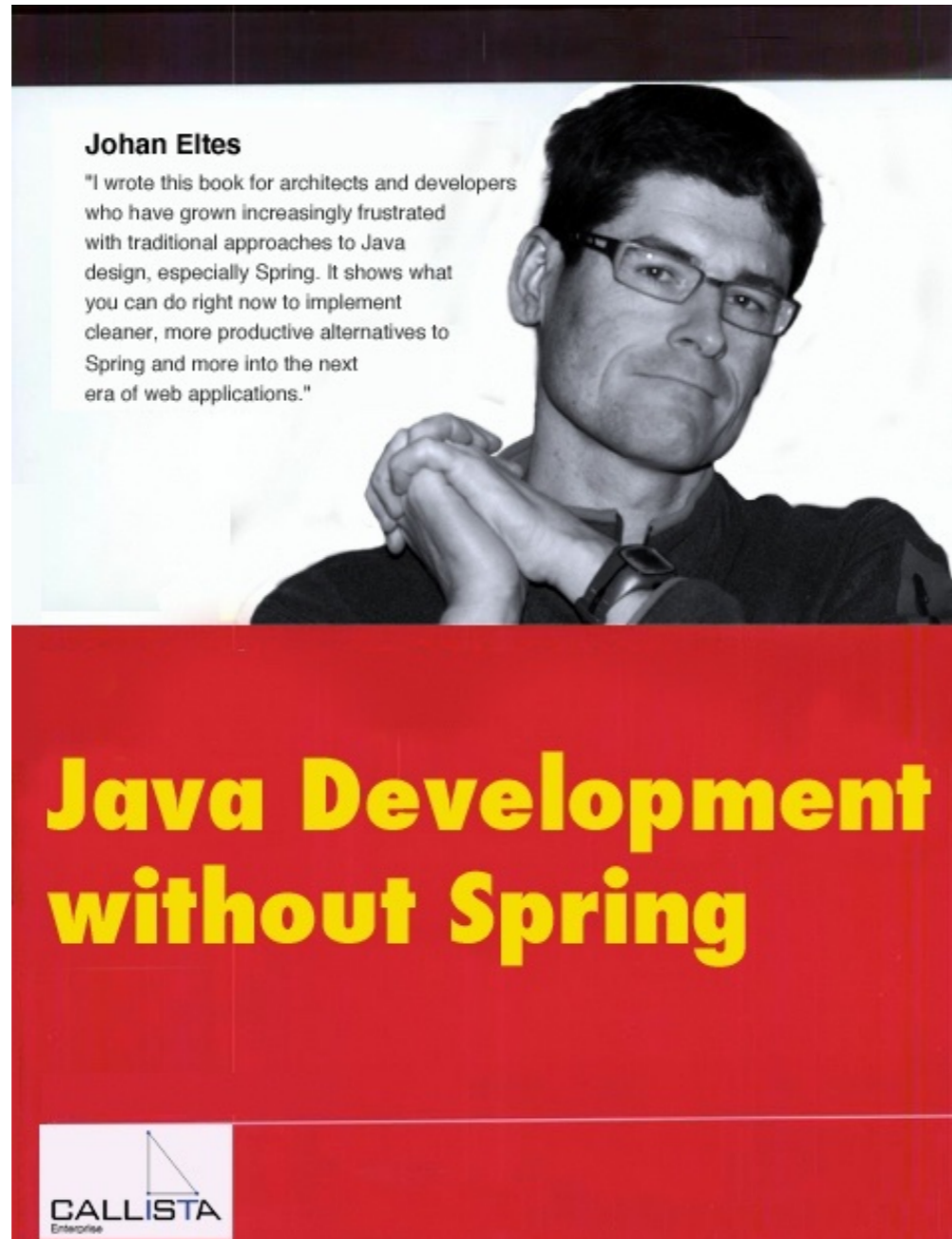


Is Java EE 6 and CDI for me?

- Why?
 - Java EE development is FUN (productive) again!
 - It is a neat, simple model
 - Zero config / overhead / layers if you want to start simple
 - Do we still need “help” from Spring?
- When?
 - JBoss, Oracle and Sun seems to be there “soon”
 - Likely to get OS-extensions that bring it to Tomcat very soon
- Who?
 - Very good news for no-floss-shops :)
 - Option for building up Spring competence - Java EE 6 Web Profile is likely an easier start




Thank's for listening!



Johan Eltes
"I wrote this book for architects and developers who have grown increasingly frustrated with traditional approaches to Java design, especially Spring. It shows what you can do right now to implement cleaner, more productive alternatives to Spring and more into the next era of web applications."

Java Development without Spring



Credits

- Image of feathers:
 - <http://askbiologist.asu.edu>

