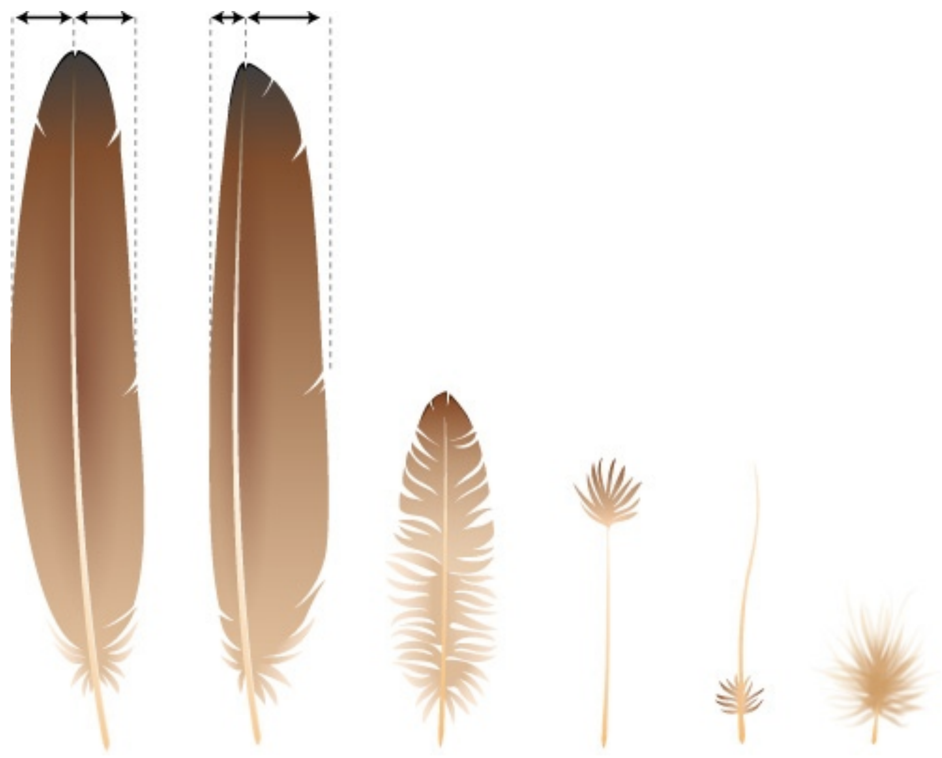
The New Java EE

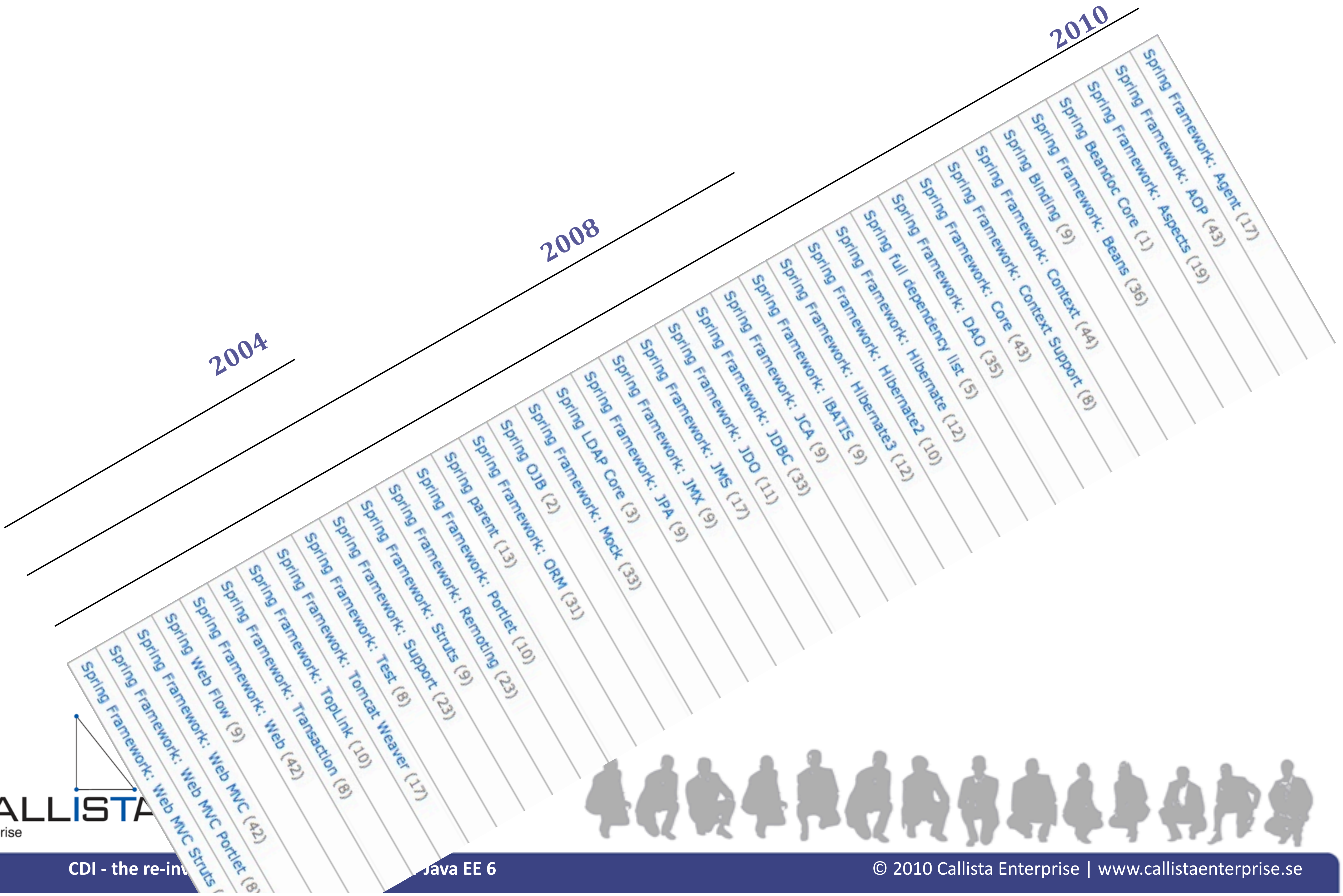
CDI - the re-invented component model
for Java EE 6. Better than Spring?

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



CALLISTA
Enterprise

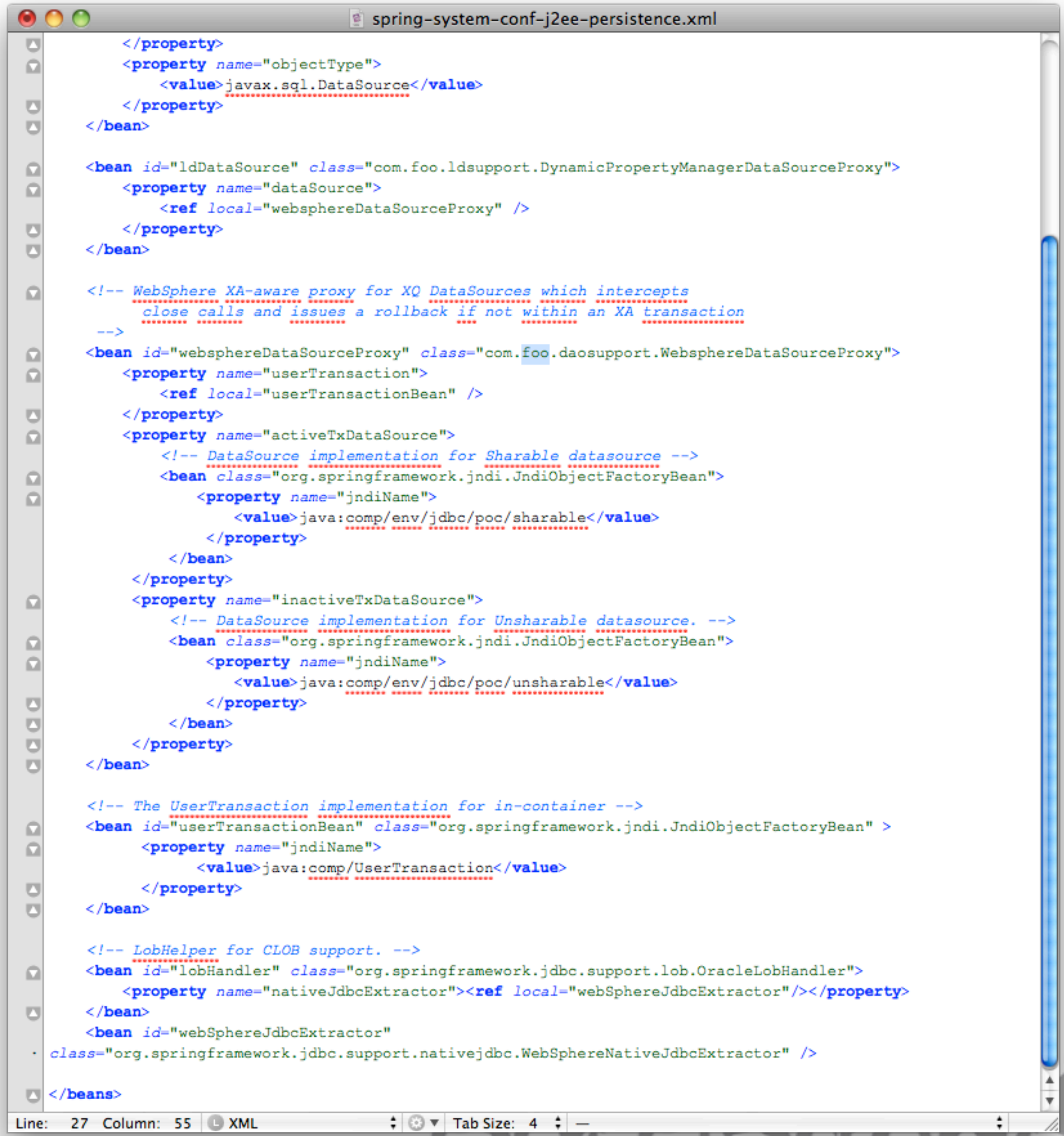
Once upon a time Spring was comprehensible…



Spring Framework: Agent (17)
Spring Framework: AOP (43)
Spring Framework: Aspects (19)
Spring Beandoc Core (1)
Spring Framework: Beans (36)
Spring Binding (9)
Spring Framework: Context (44)
Spring Framework: Context Support (8)
Spring Framework: Core (43)
Spring Framework: DAO (35)
Spring full dependency list (5)
Spring Framework: Hibernate (12)
Spring Framework: Hibernate2 (10)
Spring Framework: Hibernate3 (12)
Spring Framework: IBATIS (9)
Spring Framework: JCA (9)
Spring Framework: JDBC (33)
Spring Framework: JDO (11)
Spring Framework: JMS (17)
Spring Framework: JMX (9)
Spring Framework: JPA (9)
Spring LDAP Core (3)
Spring Framework: Mock (33)
Spring OJB (2)
Spring Framework: ORM (31)
Spring parent (13)
Spring Framework: Portlet (10)
Spring Framework: Remoting (23)
Spring Framework: Struts (9)
Spring Framework: Support (23)
Spring Framework: Test (8)
Spring Framework: Tomcat Weaver (17)
Spring Framework: TopLink (10)
Spring Framework: Transaction (8)
Spring Framework: Web (42)
Spring Web Flow (9)
Spring Framework: Web MVC (42)
Spring Framework: Web MVC Portlet (8
Spring Framework: Web MVC Struts (

CALLISTA
Enterprise

Are you a Java or XML developer?



spring-system-conf-j2ee-persistence.xml

```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

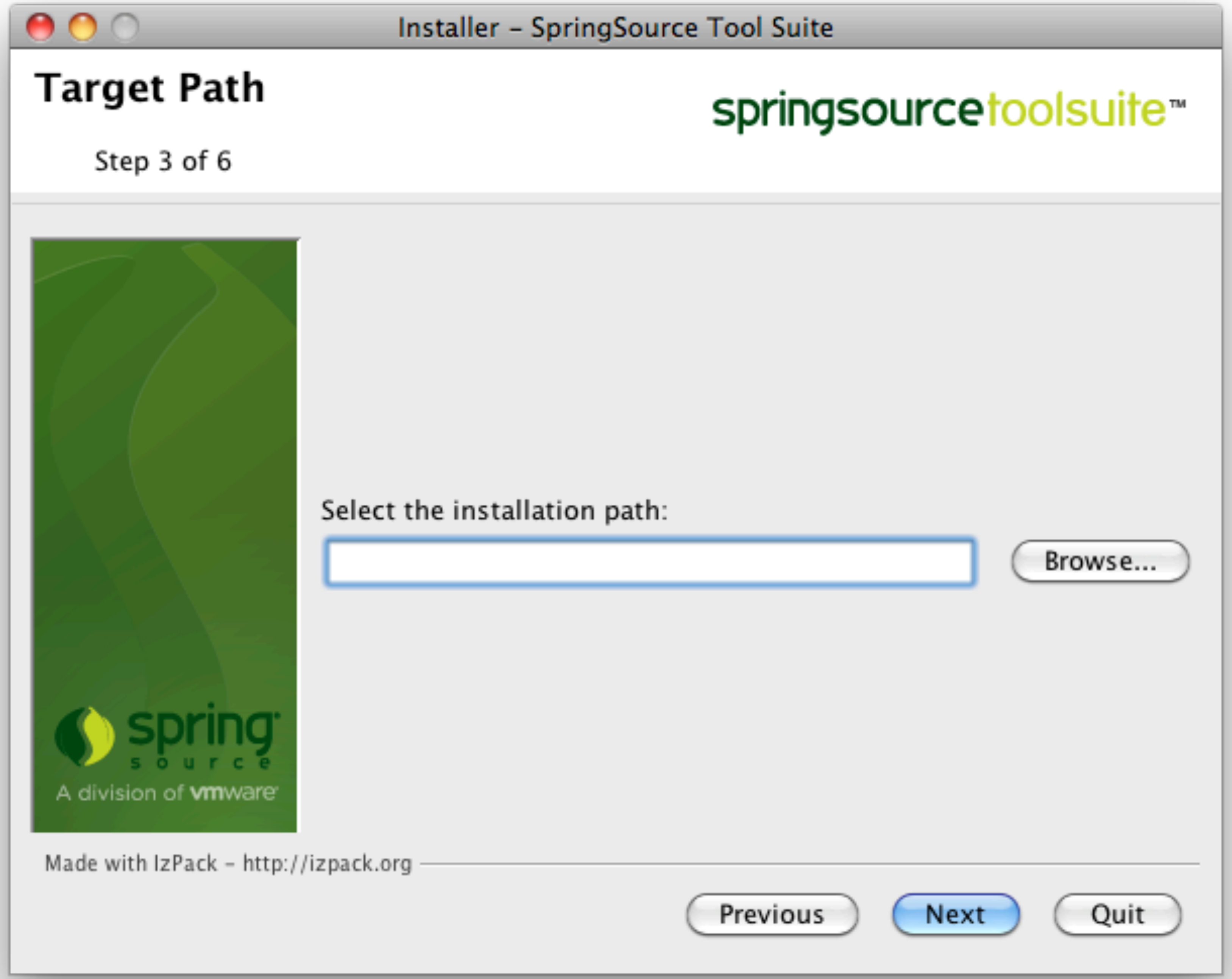
Spring wasn’t built bottom up for noXML

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.

CALLISTA
Enterprise

Complex tools require tool suites



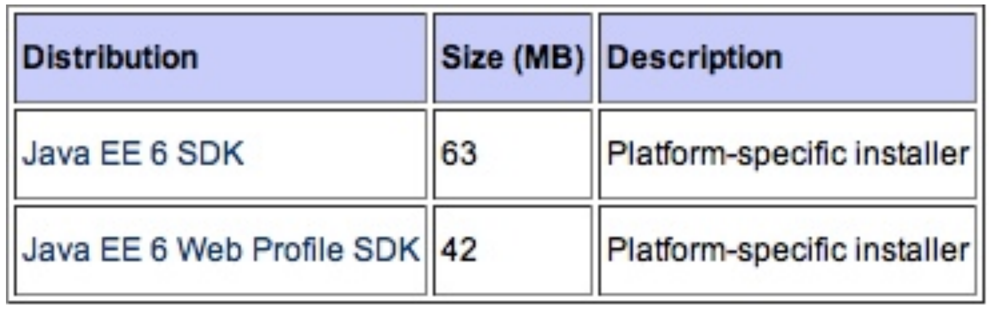
Target Path

Step 3 of 6

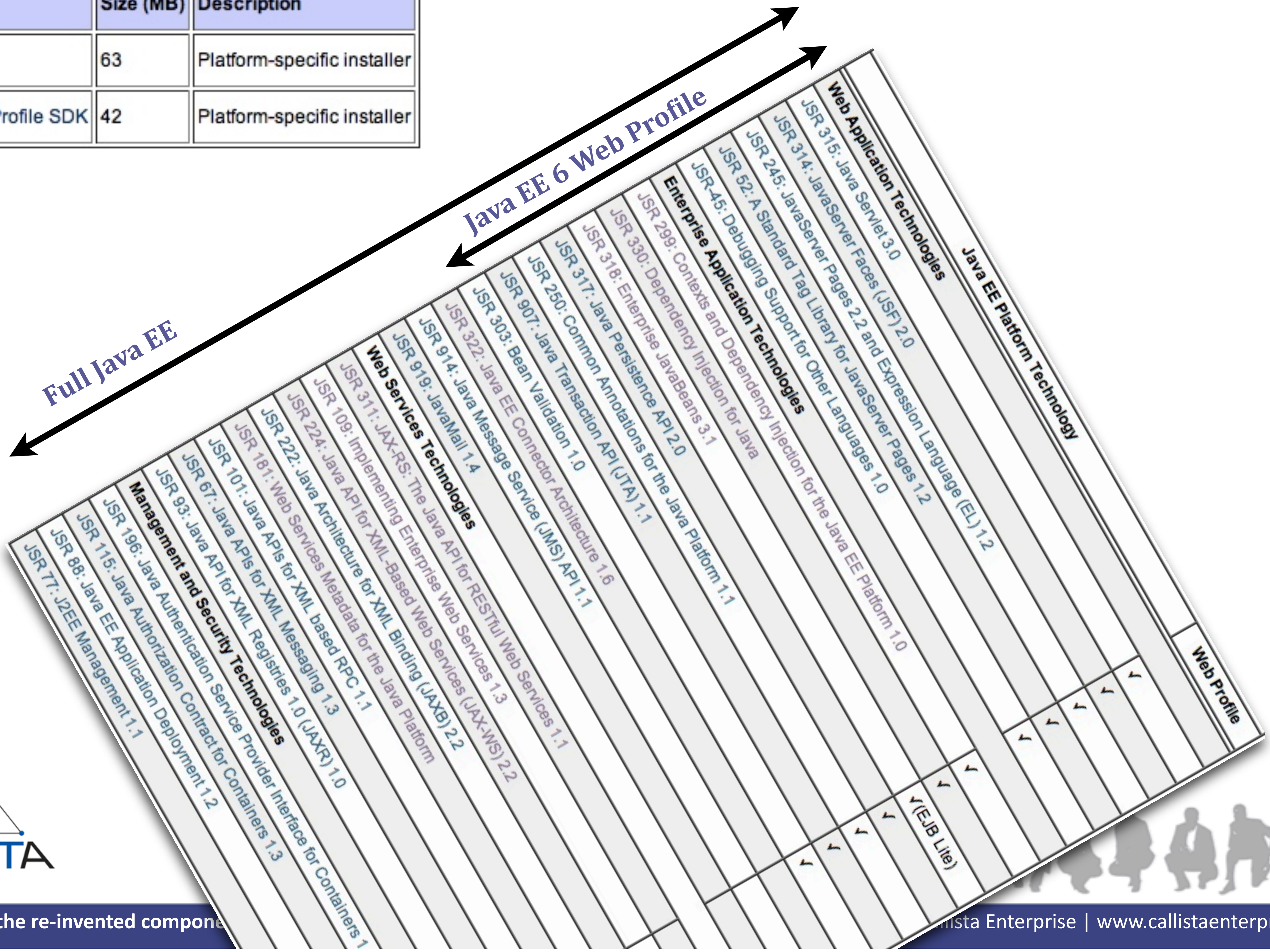
Select the installation path:

Made with IzPack – http://izpack.org

Meanwhile, Java EE is shrinking...



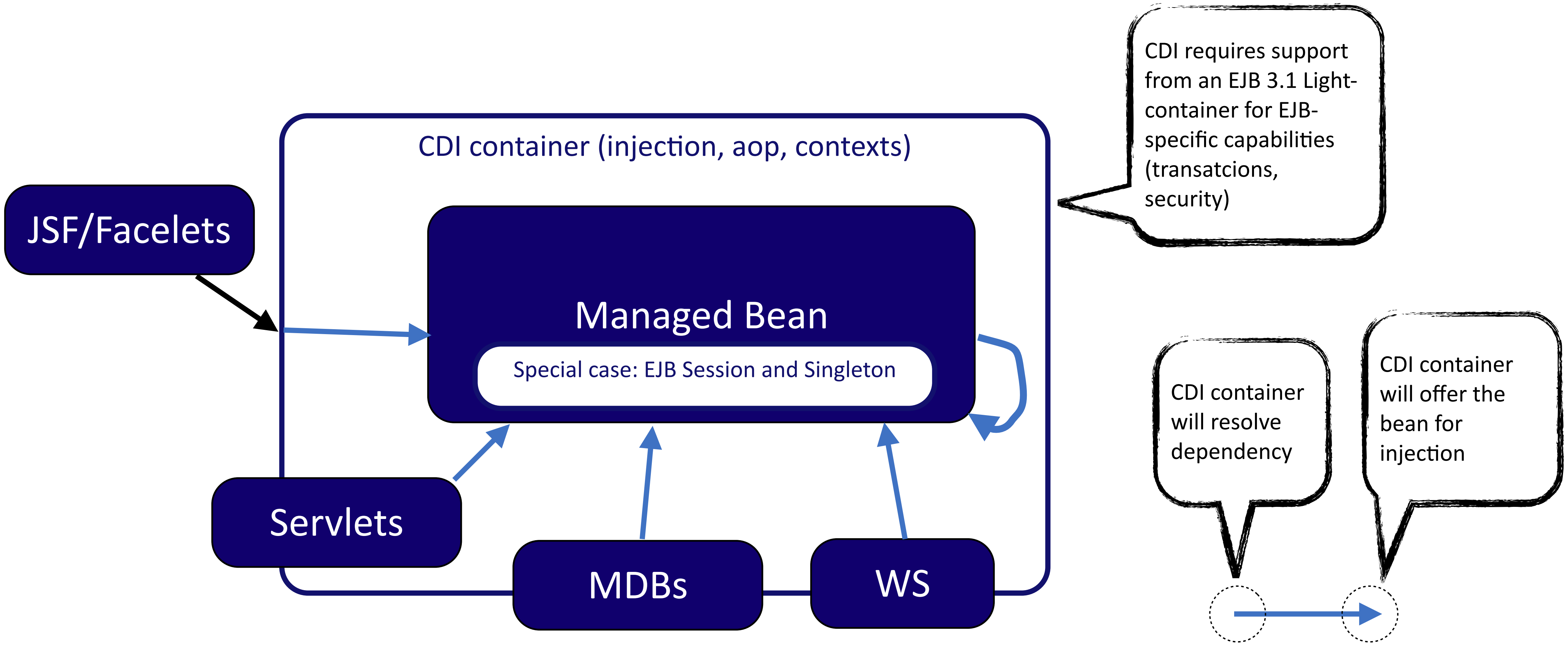
Distribution	Size (MB)	Description
Java EE 6 SDK	63	Platform-specific installer
Java EE 6 Web Profile SDK	42	Platform-specific installer


Java EE 6 Web Profile
Full Java EE

Java EE Platform Technology	Web Profile
Web Application Technologies	
JSR 315: Java Servlet 3.0	✓
JSR 314: JavaServer Faces (JSF) 2.0	✓
JSR 245: JavaServer Pages 2.2 and Expression Language (EL) 1.2	✓
JSR 52: A Standard Tag Library for JavaServer Pages 1.2	✓
JSR-45: Debugging Support for Other Languages 1.0	✓
Enterprise Application Technologies	
JSR 299: Contexts and Dependency Injection for the Java EE Platform 1.0	✓
JSR 330: Dependency Injection for Java	✓
JSR 318: Enterprise JavaBeans 3.1	✓(EJB Lite)
JSR 317: Java Persistence API 2.0	✓
JSR 250: Common Annotations for the Java Platform 1.1	✓
JSR 907: Java Transaction API (JTA) 1.1	✓
JSR 303: Bean Validation 1.0	✓
JSR 322: Java EE Connector Architecture 1.6	
JSR 914: Java Message Service (JMS) API 1.1	
JSR 919: JavaMail 1.4	
Web Services Technologies	
JSR 311: JAX-RS: The Java API for RESTful Web Services 1.1	
JSR 109: Implementing Enterprise Web Services 1.3	
JSR 224: Java API for XML-Based Web Services (JAX-WS) 2.2	
JSR 222: Java Architecture for XML Binding (JAXB) 2.2	
JSR 181: Web Services Metadata for the Java Platform	
JSR 101: Java APIs for XML based RPC 1.1	
JSR 67: Java APIs for XML Messaging 1.3	
JSR 93: Java API for XML Registries 1.0 (JAXR) 1.0	
Management and Security Technologies	
JSR 196: Java Authentication Service Provider Interface for Containers 1	
JSR 115: Java Authorization Contract for Containers 1.3	
JSR 88: Java EE Application Deployment 1.2	
JSR 77: J2EE Management 1.1	

CALLISTA
Enterprise

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


CDI container (injection, aop, contexts)
JSF/Facelets
Managed Bean
Special case: EJB Session and Singleton
Servlets
MDBs
WS
CDI requires support from an EJB 3.1 Light-container for EJB-specific capabilities (transatcions, security)
CDI container will resolve dependency
CDI container will offer the bean for injection

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!

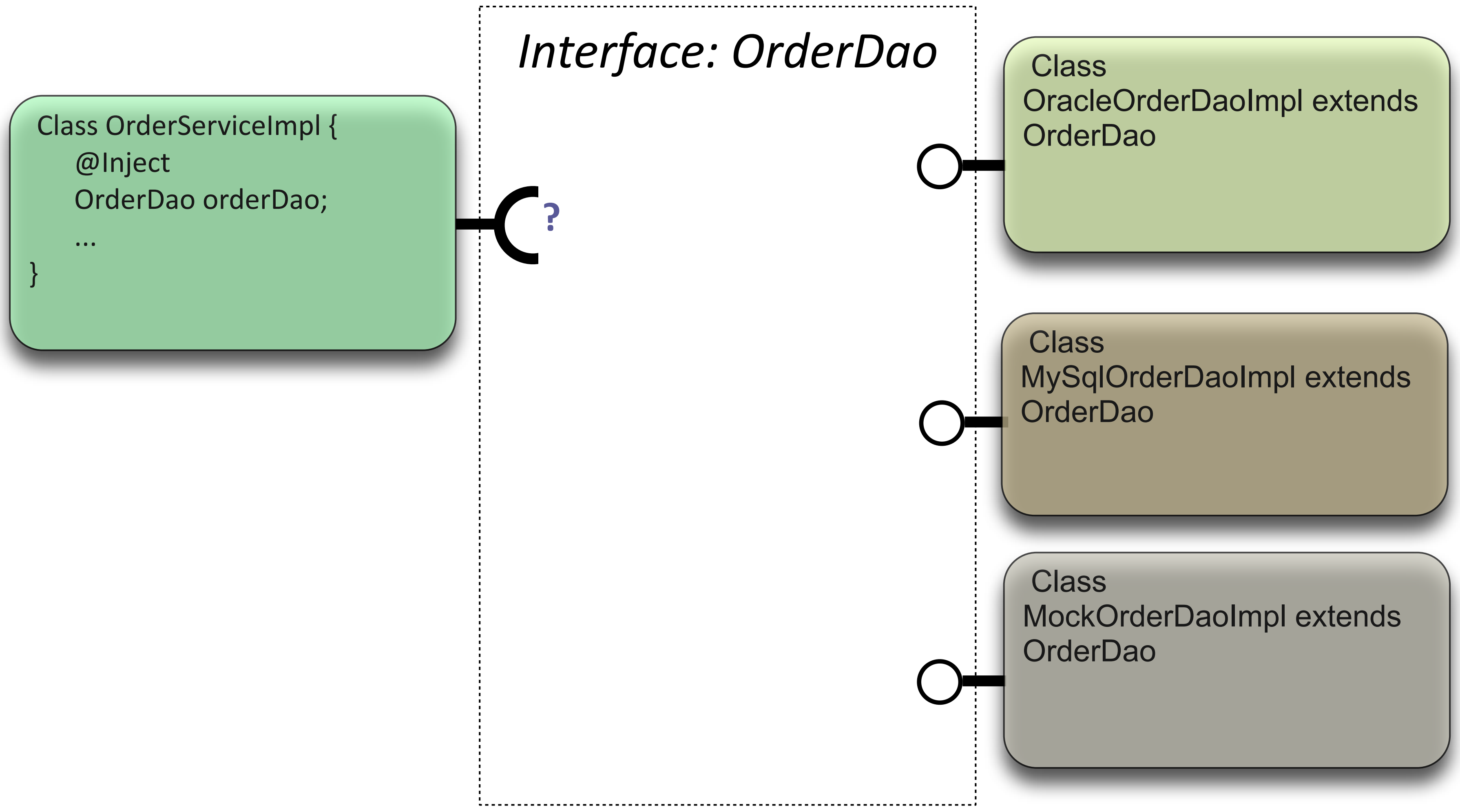
All demo code for the use-cases are available for public checkout at this svn repo:

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

- 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

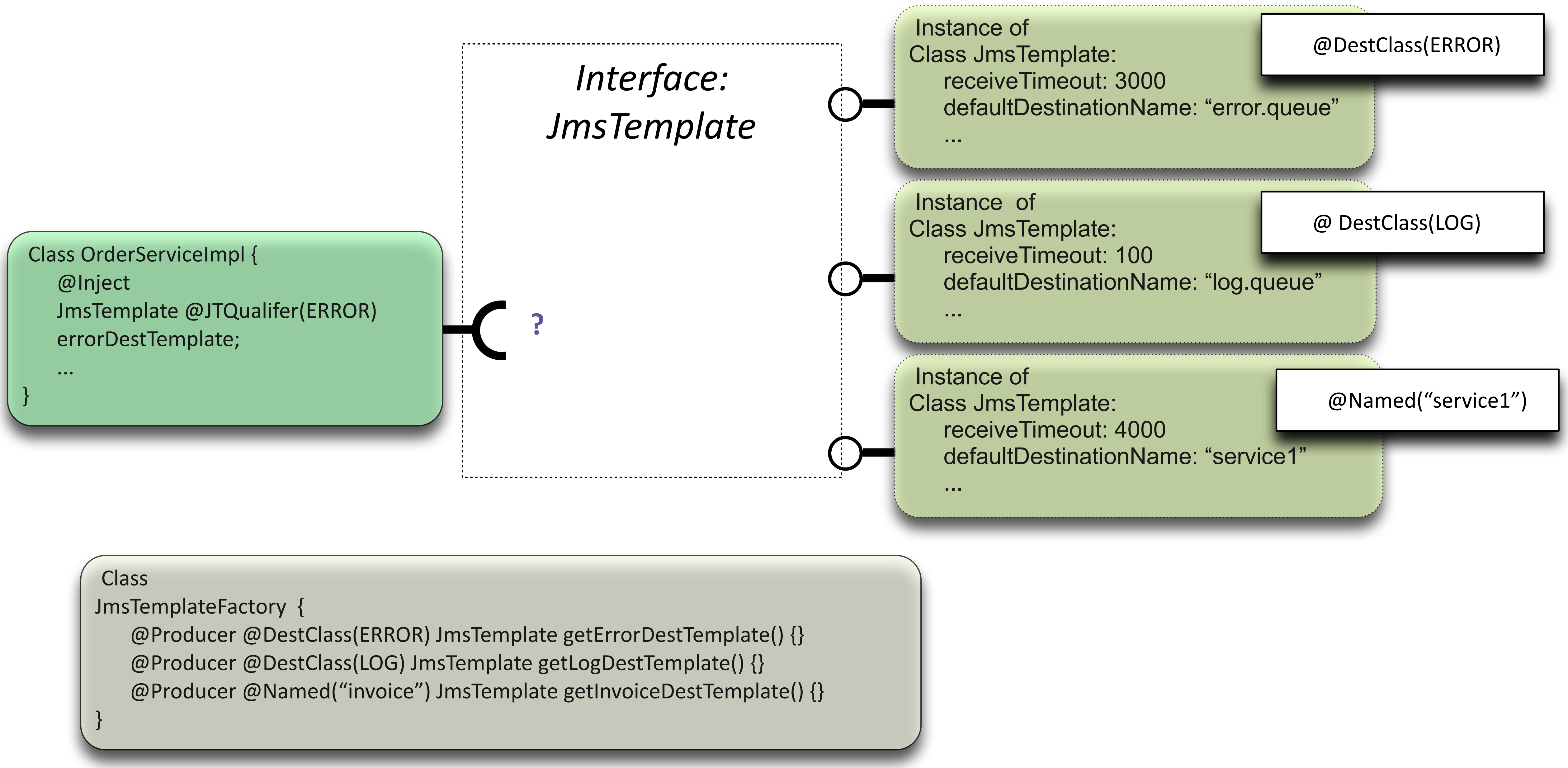
Annotations, annotations, annotations... @Inject, @Produces, @Qualifer, @Alternative, @Stateless, @Conversation

UC 1: Basic DI for Unit testing


Interface: OrderDao
Class OrderServiceImpl {
@Inject
OrderDao orderDao;
...
}
?
Class OracleOrderDaoImpl extends OrderDao
Class MySqlOrderDaoImpl extends OrderDao
Class MockOrderDaoImpl extends OrderDao

@Inject is by default auto-injection by type

UC 2: Factories and qualifiers



```
Instance of
Class JmsTemplate:
    receiveTimeout: 3000
    defaultDestinationName: “error.queue”
    …
```
@DestClass(ERROR)

```
Instance  of
Class JmsTemplate:
    receiveTimeout: 100
    defaultDestinationName: “log.queue”
    …
```
@ DestClass(LOG)

```
Instance of
Class JmsTemplate:
    receiveTimeout: 4000
    defaultDestinationName: “service1”
    …
```
@Named(“service1”)

```
Class OrderServiceImpl {
    @Inject
    JmsTemplate @JTQualifer(ERROR)
    errorDestTemplate;
    …
}
```

```
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

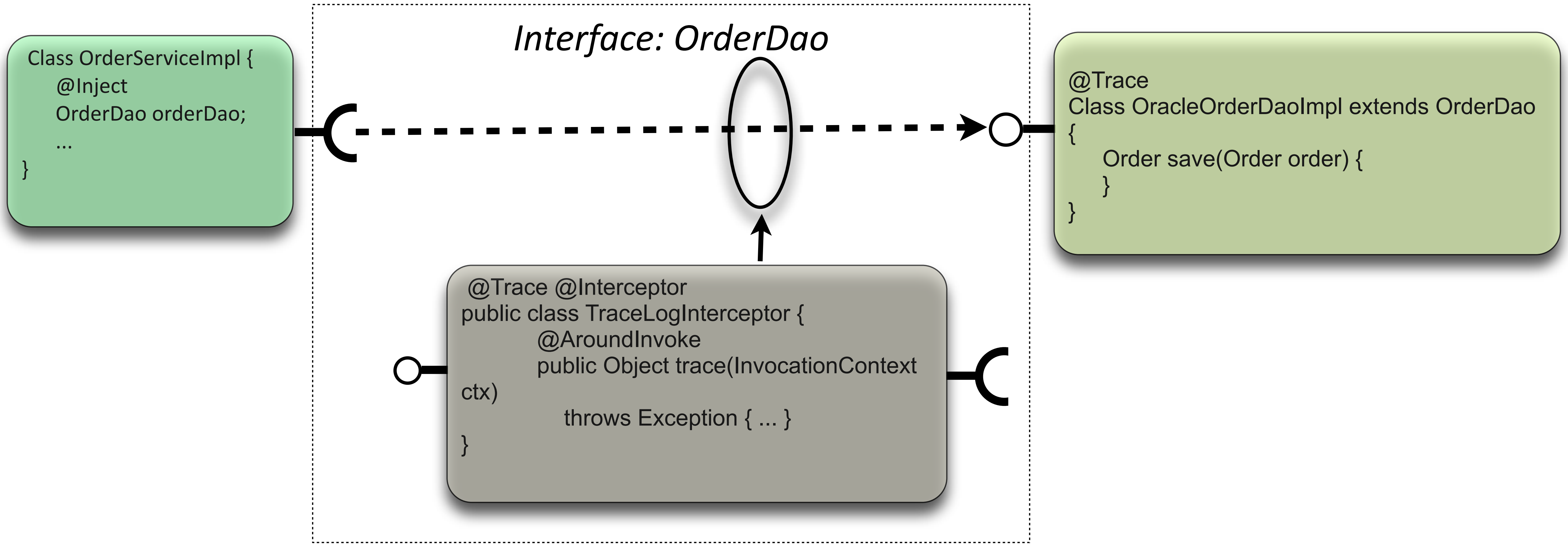
UC 3: Integration testing with JUnit

How much weight has EJB lost??

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

...with embeddable EJB container

UC 4: Cross-cutting concerns



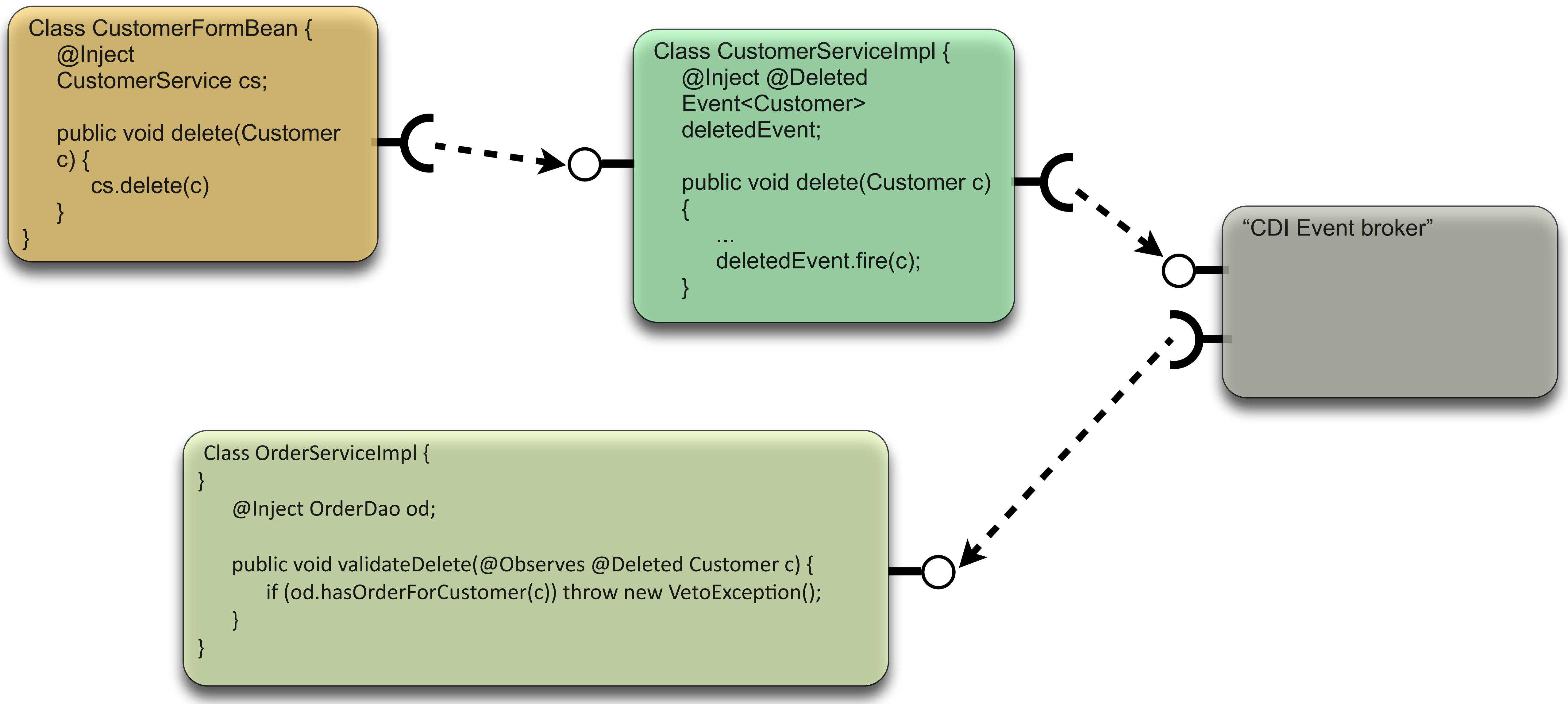
```
Class OrderServiceImpl {
    @Inject
    OrderDao orderDao;
    ...
}
```

```
@Trace
Class OracleOrderDaoImpl extends OrderDao
{
    Order save(Order order) {
    }
}
```

```
 @Trace @Interceptor
public class TraceLogInterceptor {
          @AroundInvoke
          public Object trace(InvocationContext
ctx)
             throws Exception { ... }
}
```

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

UC 5: Events



```
Class CustomerFormBean {
    @Inject
    CustomerService cs;

    public void delete(Customer
    c) {
        cs.delete(c)
    }
}
```

```
Class CustomerServiceImpl {
    @Inject @Deleted
    Event<Customer>
    deletedEvent;

    public void delete(Customer c)
    {
        ...
        deletedEvent.fire(c);
    }
```

```
Class OrderServiceImpl {
}
    @Inject OrderDao od;

    public void validateDelete(@Observes @Deleted Customer c) {
        if (od.hasOrderForCustomer(c)) throw new VetoException();
    }
}
```

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

UC 6: Declarative scope management

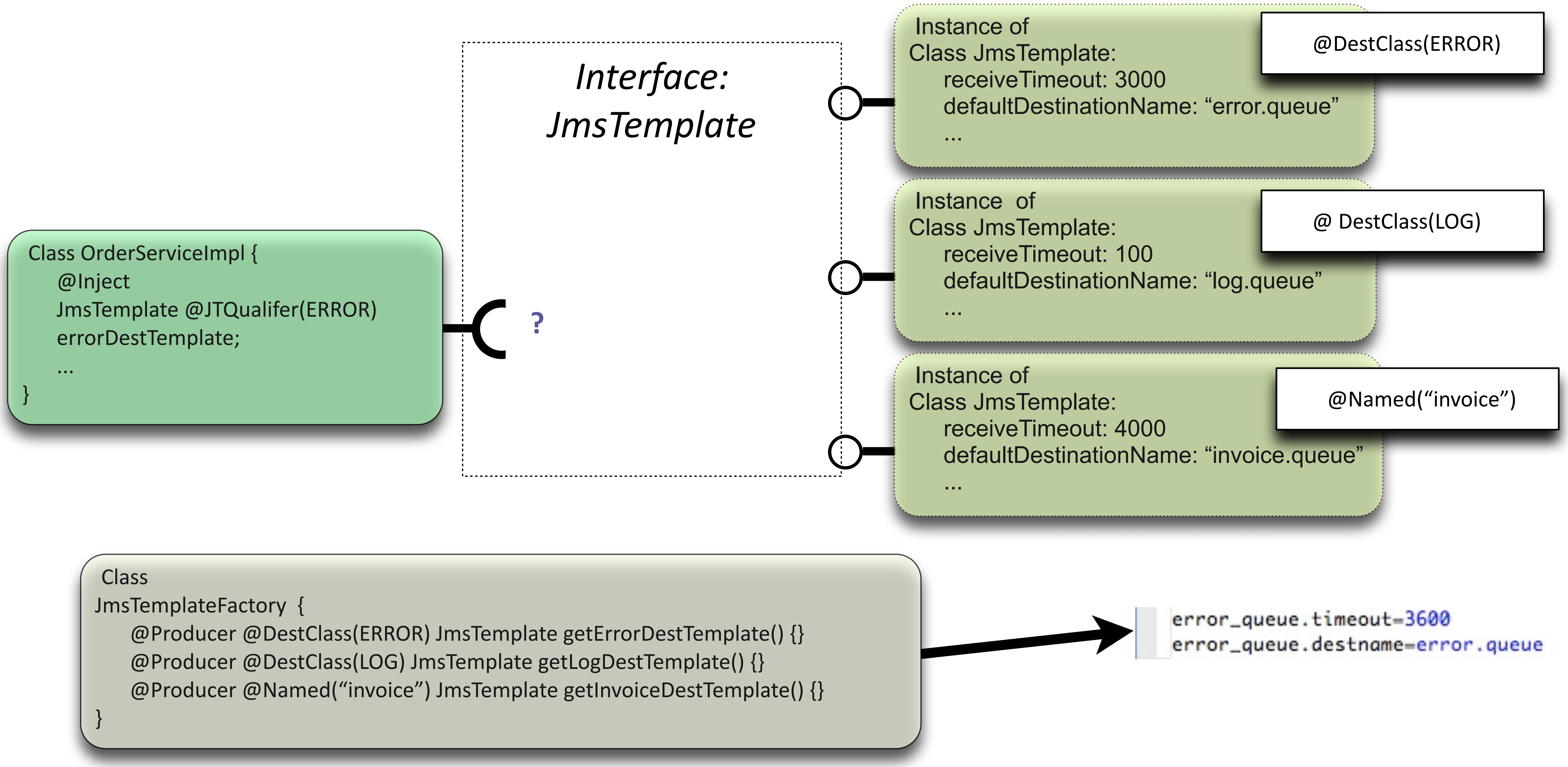
- ejb Session
- ejb Singleton
- ApplicationScoped
- RequestScoped
- SessionScoped
- ConversationScoped

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

- Additional scopes may be added via extention api ("SPI")

Use-case 5: Contextual services

UC 7: Extentions - Sample: config file



```
Class OrderServiceImpl {
    @Inject
    JmsTemplate @JTQualifer(ERROR)
    errorDestTemplate;
    ...
}
```

```
Instance of
Class JmsTemplate:
    receiveTimeout: 3000
    defaultDestinationName: “error.queue”
    ...
```

```
Instance  of
Class JmsTemplate:
    receiveTimeout: 100
    defaultDestinationName: “log.queue”
    ...
```

```
Instance of
Class JmsTemplate:
    receiveTimeout: 4000
    defaultDestinationName: “invoice.queue”
    ...
```

```
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
```

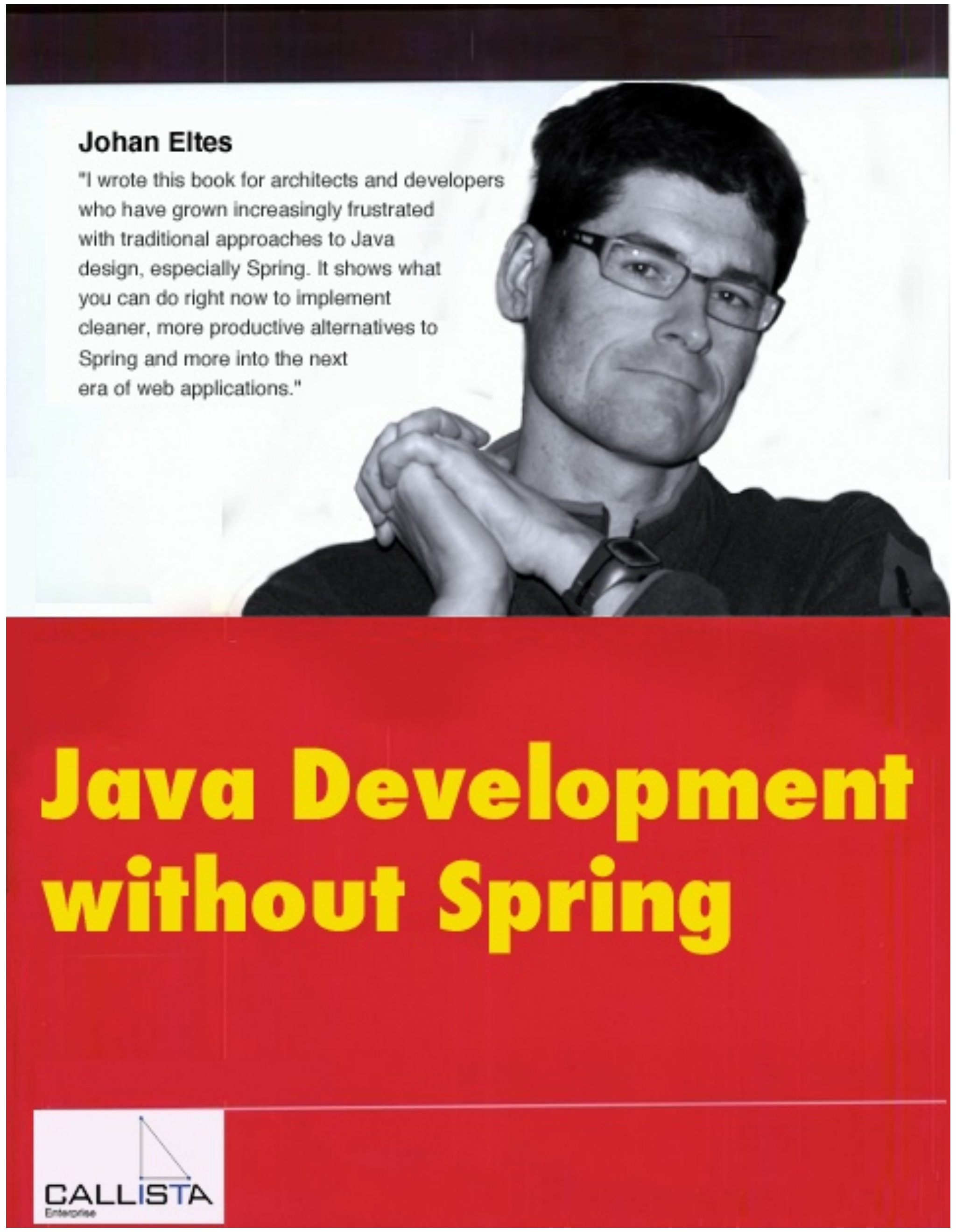
Sample: Extension for file-based configuration of @Producers

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

CALLISTA Enterprise

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://askabiologist.asu.edu