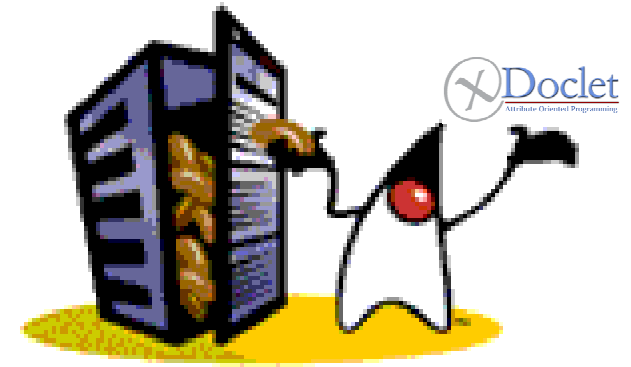# Metadata Attribute driven J2EE development

Magnus Larsson

Callista Enterprise AB

magnus.larsson@callista.se

http://www.callista.se/enterprise

Agenda

☐ Introduction

  ▫ The problem

  ▫ Current solutions

  ▫ An alternative - Metadata Attributes

  ▫ History, today and future

☐ XDoclet

  ▫ What is it?

  ▫ Example

  ▫ Findings

  ▫ Demo

☐ Summary

CALLISTA

The problem

☐ J2EE/EJB specific code and declarations

    ▫ Extra code and declarations is required outside of the EJB-bean class

        • Extra Java Code required
            – Remote/Local Interface
            – Home Interface

        • J2EE Deployment Descriptors
            – A lot of details…

        • Vendor specific Deployment Descriptors
            – Different formats for each J2EE vendor…

    ▫ **This is a pain!!!**

CALLISTA

Current solutions

☐ Wizard driven development tools

Enterprise Editions of Java IDE's with J2EE wizards

- ▫ Expensive and Complex

- ▫ Software Vendor lock in
  - Tool A support J2EE server B and C
    - Including version dependencies

- ▫ Wizards are great for beginners!
  - But a lot of manual settings for a experienced developer
    - Defining 20 EJB Beans in a wizard is not fun…
  - Expose every detail in J2EE

- ▫ No support for applying to architectural guidelines

CALLISTA

An alternative - Metadata Attributes

☐ Annotate the source code with attributes that indicates specific behavior of the code

☐ Such annotations are called *metadata*

☐ Source Code generation based on *metadata attributes*

☐ Development tools create *metadata attributes*
  - Reduced complexity for development tools
  - Standardized metadata enable competition
  - Opens up for [less complex] Open Source tools

CALLISTA

Metadata Attributes
in the history, today and in the future

☐ History

　☐ EJBDoclet – initial release in 2000

　　• Today replaced by XDoclet

☐ Today

　☐ .NET

　☐ XDoclet

☐ Future

　☐ Java 1.5 (near term)

　☐ J2EE 1.5 (long term)

CALLISTA

Metadata Attributes today

- [ ] Already in use in .NET

    - Declarative transaction attribute in C#

    ```
    [Transaction(TransactionOption.Supported)]
    public class Account : ServicedComponent {
        ...
    }
    ```

    - Test attribute in C# for NUnit (jUnit for .NET...)

    ```
    [Test]
    public void TwoPlusTwo() {
      AssertEquals(4, 2+2);
    }
    ```

CALLISTA

Metadata Attributes today

☐ Also already in use in Java and J2EE using XDoclet

  ▫ Annotating an EJB-bean class

```
/**
 * @ejb.bean       name            = "OrderService"
 *                 type            = "Stateless"
 *                 transaction-type = "Container"
 *                 jndi-name       = "${OrderService.jndiname}"
 */
public class OrderServiceBean implements SessionBean {...}
```

  ▫ Annotating an EJB-bean method

```
/**
 * @ejb.interface-method
 * @ejb.transaction     type = "Required"
 */
public Collection findByUsername(String username) {...}
```

  ▫ **Note:** More details on XDoclet later...

CALLISTA

Metadata Attributes in the future

❑ Metadata Attributes on its way into Java 1.5 and J2EE 1.5

   ❑ Strongly influenced by XDoclet

   ❑ Java 1.5 (Tiger)
- JSR 175 - A Metadata Facility for Java

     `@Remote public Collection findByUsername(String username) {...}`

   ❑ J2EE 1.5
- Major theme is "ease of development"

- JSR 220 - Enterprise JavaBeans™ 3.0
  - Deployment Descriptors, Component and Home Interfaces replaced by standardized metadata attributes

- JSR 181 - Web Services Metadata for the Java™ Platform

CALLISTA

Agenda – Where are we?

☐ Introduction

- ❑ The problem
- ❑ Current solutions
- ❑ An alternative - Metadata Attributes
- ❑ History, today and future

☐ XDoclet

- ❑ **What is it?**
- ❑ Example
- ❑ Findings
- ❑ Demo

☐ Summary

CALLISTA

XDoclet

☐ Open Source project

  ▫ http://xdoclet.sourceforge.net/

☐ Adds metadata attributes as JavaDoc tags

☐ Uses Ant to start source code generation

☐ Easy to extend

  ▫ Highly modularized design
  ▫ Template based

CALLISTA

XDoclet

☐ Strong support for J2EE and EJB

- ◻ Its how it all started with EJBDoclet...

- ◻ Provides good default values for most metadata attributes
  - Compact code

- ◻ All details in the deployment descriptors can be configured
  - Full control when required

- ◻ Supports 10+ J2EE servers out of the box
  - WebLogic, WebSphere, OC4J, JBoss, Orion, JOnAS...
  - Enabling J2EE portability "in reality"

- ◻ XDoclet tags
  - `@ejb, @web` for standard features
  - `@jboss, @weblogic, @orion`… for vendor specific features

CALLISTA

XDoclet

☐ Some vendor specific tags has moved into standard tags

    ▫ E.g. `jndi-name`, `table-name` and `column-name`

    ▫ Example:

```
/**
 * @ejb:bean       name           = "OrderEntity"
 *                 local-jndi-name = "${OrderEntity.jndiname}"
 *
 * @jboss:bean     local-jndi-name = "${OrderEntity.jndiname}"
 * @orion:bean     local-jndi-name = "${OrderEntity.jndiname}"
 * @weblogic:bean  local-jndi-name = "${OrderEntity.jndiname}"
 *
 */
public abstract class OrderEntityBean implements EntityBean {
```

Not required!

CALLISTA

XDoclet

☐ Based on these annotations XDoclet generates

　　▫ Remote, Local and Home Interfaces

　　▫ Utility classes
　　　　• Factory-class, Value Objects, Primary Key classes

　　▫ Deployment Descriptors
　　　　• Both J2EE and vendor specific

CALLISTA

XDoclet

☐ Generating source code

    ▫ XDoclet is invoked by Ant-tasks

    ▫ Creating files for an EJB Module
        • Use Ant-task `<ejbdoclet>`

    ▫ Creating files for an Web Module
        • Use Ant-task `<webdoclet>`

CALLISTA

XDoclet

☐ Not only used with J2EE and EJB

    ▫ Out of the box support for

- JDO
- Hibernate
- Struts
- WebWork
- Web Services
- Portlets
- Mock Objects
- JMX

CALLISTA

XDoclet

☐ Development tools with support for XDoclet

    ◻ Eclipse plugins
- MyEclipse (http://www.myeclipseide.com)
- JBoss IDE (http://www.jboss.org)
- EMF - Eclipse Modeling Framework (http://www.eclipse.org/emf)

    ◻ Model/UML driven tools
- AndroMDA   (http://sourceforge.net/projects/andromda)
- JAG   (http://sourceforge.net/projects/jag)
- Middlegen   (http://boss.bekk.no/boss/middlegen)
- EclipseUML Enterprise Edition (http://www.omondo.com/index.jsp)

CALLISTA

Agenda – Where are we?

☐ Introduction

    ▫ The problem

    ▫ Current solutions

    ▫ An alternative - Metadata Attributes

    ▫ History, today and future

☐ XDoclet

    ▫ What is it?

    ▫ **Example**

    ▫ Findings

    ▫ Demo

☐ Summary

CALLISTA

XDoclet - example

☐ Creating files for an EJB Module

   ☐ Input to `<ejbdoclet>` from ProductEntityBean.java

```
/**
 * @ejb:bean name        = "ProductEntity"
 *           type        = "CMP"
 *           cmp-version = "2.x"
 *           view-type   = "local"
 *
 * @ejb.persistence table-name = "product"
 *
 * @ejb:value-object
 *
 * @ejb:transaction type = "Required"
 */
public abstract class ProductEntityBean implements EntityBean {
```

CALLISTA

XDoclet - example

☐ Creating files for an EJB Module

```xml
<ejbdoclet   destdir = "${srcGen.dir}">

   <fileset   dir      = "${src.dir}">
     <include name     = "**/*Bean.java"/>
     <include name     = "**/*MDB.java"/>
   </fileset>

   <packageSubstitution
       packages        = "beans"
       substituteWith = "interfaces"/>

   <valueobject/>

   <utilobject/>
```
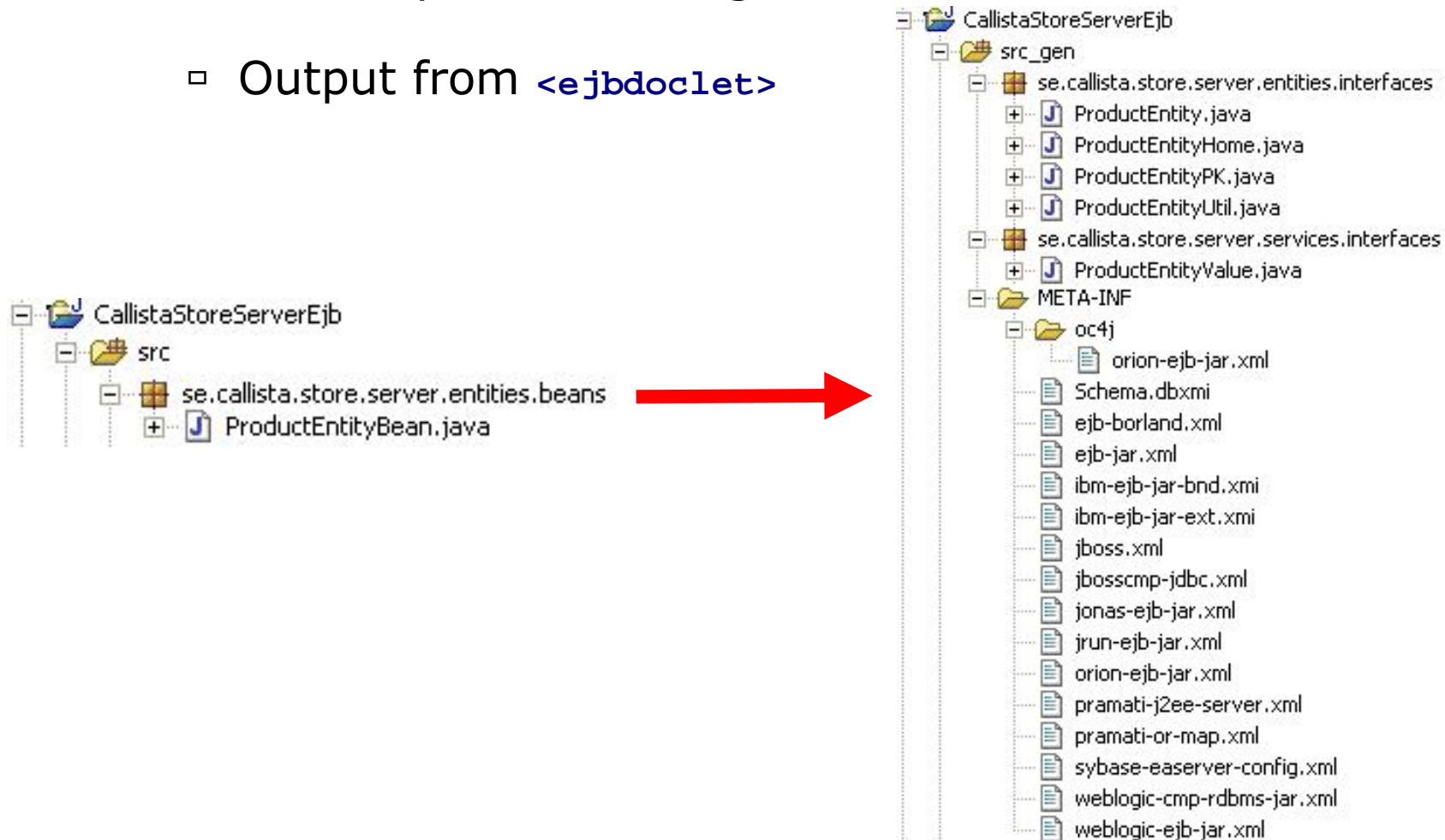
CALLISTA

XDoclet - example

☐ Creating files for an EJB Module

```
<remoteinterface/>
<homeinterface/>
<localinterface     pattern="{0}"/>
<localhomeinterface pattern="{0}Home"/>
<entitypk/>
<deploymentdescriptor destdir="${srcGen.dir}/META-INF" useIds="true"/>

<jboss     destdir = "${srcGen.dir}/META-INF"/>
<websphere destdir = "${srcGen.dir}/META-INF" useIds="true"/>
<weblogic  destdir = "${srcGen.dir}/META-INF"/>
<orion     destdir = "${srcGen.dir}/META-INF"/>
<oc4j      destdir = "${srcGen.dir}/META-INF/oc4j"/>
<jonas     destdir = "${srcGen.dir}/META-INF"/>
<jrun      destdir = "${srcGen.dir}/META-INF"/>
<easerver  destdir = "${srcGen.dir}/META-INF"/>
<borland   destdir = "${srcGen.dir}/META-INF"/>
<pramati   destdir = "${srcGen.dir}/META-INF"/>
<sunone    destdir = "${srcGen.dir}/META-INF"/>
</ejbdoclet>
```

CALLISTA

XDoclet - example

☐ J2EE examples – creating files for an EJB Module

    ❑ Output from `<ejbdoclet>`

```
□ CallistaStoreServerEjb
  □ src
    □ se.callista.store.server.entities.beans
      ⊞ J ProductEntityBean.java
```

➡️

```
□ CallistaStoreServerEjb
  □ src_gen
    □ se.callista.store.server.entities.interfaces
      ⊞ J ProductEntity.java
      ⊞ J ProductEntityHome.java
      ⊞ J ProductEntityPK.java
      ⊞ J ProductEntityUtil.java
    □ se.callista.store.server.services.interfaces
      ⊞ J ProductEntityValue.java
    □ META-INF
      □ oc4j
          orion-ejb-jar.xml
        Schema.dbxmi
        ejb-borland.xml
        ejb-jar.xml
        ibm-ejb-jar-bnd.xmi
        ibm-ejb-jar-ext.xmi
        jboss.xml
        jbosscmp-jdbc.xml
        jonas-ejb-jar.xml
        jrun-ejb-jar.xml
        orion-ejb-jar.xml
        pramati-j2ee-server.xml
        pramati-or-map.xml
        sybase-easerver-config.xml
        weblogic-cmp-rdbms-jar.xml
        weblogic-ejb-jar.xml
```

CALLISTA

Agenda – Where are we?

☐ Introduction

    ▫ The problem

    ▫ Current solutions

    ▫ An alternative - Metadata Attributes

    ▫ History, today and future

☐ XDoclet

    ▫ What is it?

    ▫ Example

    ▫ **Findings**

    ▫ Demo

☐ Summary

CALLISTA

XDoclet - findings

□ An example was built

    ▫ To learn some more than just "Hello World"

    ▫ Cover most essential parts of J2EE

    ▫ Deploy on different J2EE servers and databases

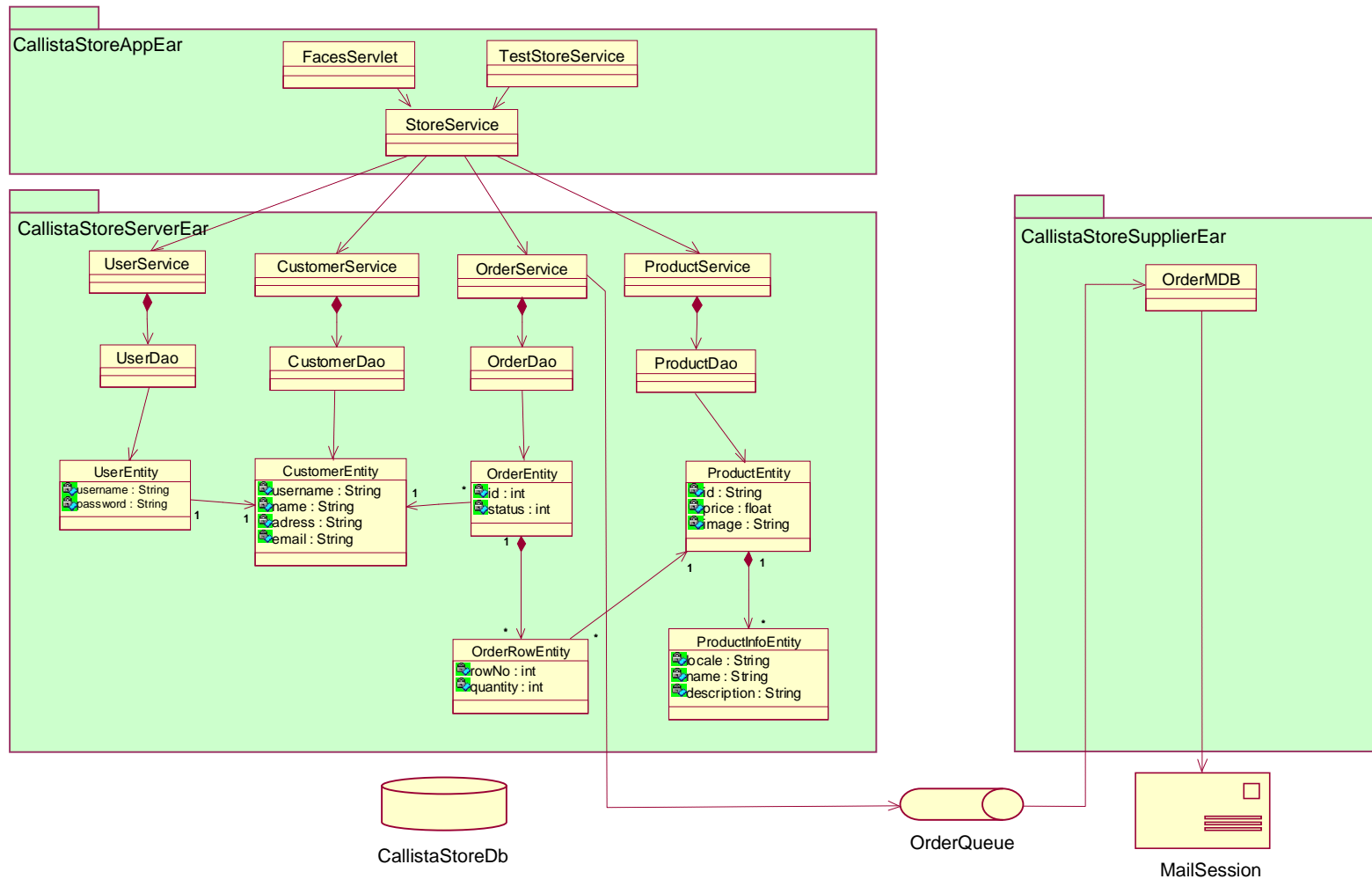   ▫ More than one J2EE Application (subsystem)

CALLISTA

XDoclet - findings

□ An example - CallistaStore

    ◻ Modeled after the [in]famous Java Petstore

        • But with a much much smaller code base

        • Cover the the same level of J2EE-functionality

    ◻ Three J2EE Applications

        • CallistaStoreServerEar   - Business data and rules

        • CallistaStoreAppEar   - Customer Web interface

        • CallistaStoreSupplierEar - Deliver created orders

CALLISTA
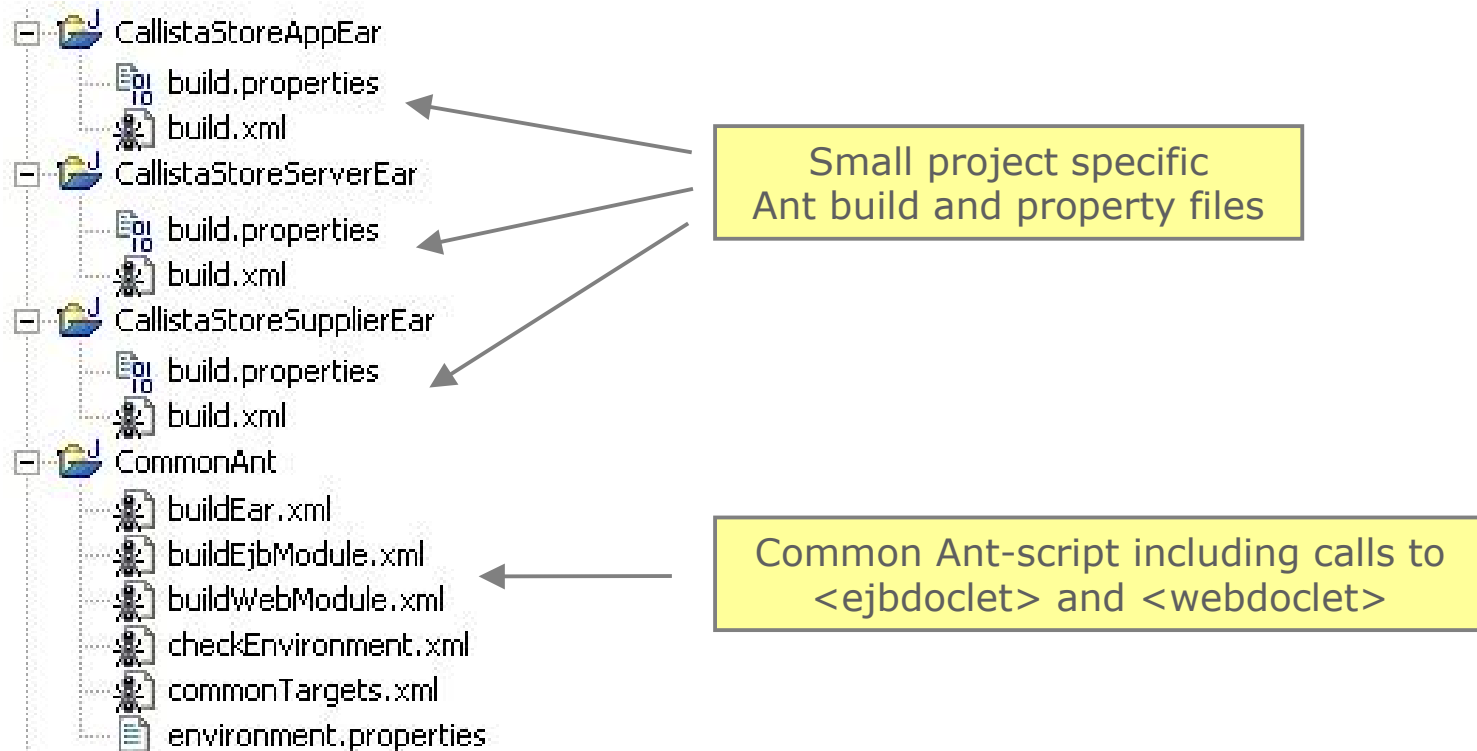
XDoclet - findings
An example – Callista Store

XDoclet - findings

+ XDoclet works well for the example

- Slightly inconsistent tags
  - □ E.g. @ejb.ejb-ref and @web.ejb-ref has different syntax
  - □ Standard tags for vendor information
    not yet used by all vendor modules

- Complex XDoclet Ant-scripts
  - □ Common set of reusable Ant-scripts (see next slide)

+ Time and quality
  - □ Automation (see next slide)

+ Fast round-trip
  - □ Efficient development (see demo)

CALLISTA

XDoclet - findings

- ☐ Complex XDoclet Ant-scripts
    - ▫ A problem with a large number of developers
    - ▫ Structure Ant-tasks so that common XDoclet-scripts can be shared by projects

```
CallistaStoreAppEar
    build.properties
    build.xml
CallistaStoreServerEar
    build.properties
    build.xml
CallistaStoreSupplierEar
    build.properties
    build.xml
CommonAnt
    buildEar.xml
    buildEjbModule.xml
    buildWebModule.xml
    checkEnvironment.xml
    commonTargets.xml
    environment.properties
```

Small project specific
Ant build and property files

Common Ant-script including calls to
<ejbdoclet> and <webdoclet>

CALLISTA

XDoclet - findings

□ Time and Quality

    □ Automation to avoid human mistakes and delays

        • Ant-script that "does it all"
- CVS checkout
- Generate source code with XDoclet
- Compile source code
- Package J2EE-modules and EAR-files
- Deploy EAR-files
- Run tests

CALLISTA

Agenda – Where are we?

☐ Introduction

   ▫ The problem

   ▫ Current solutions

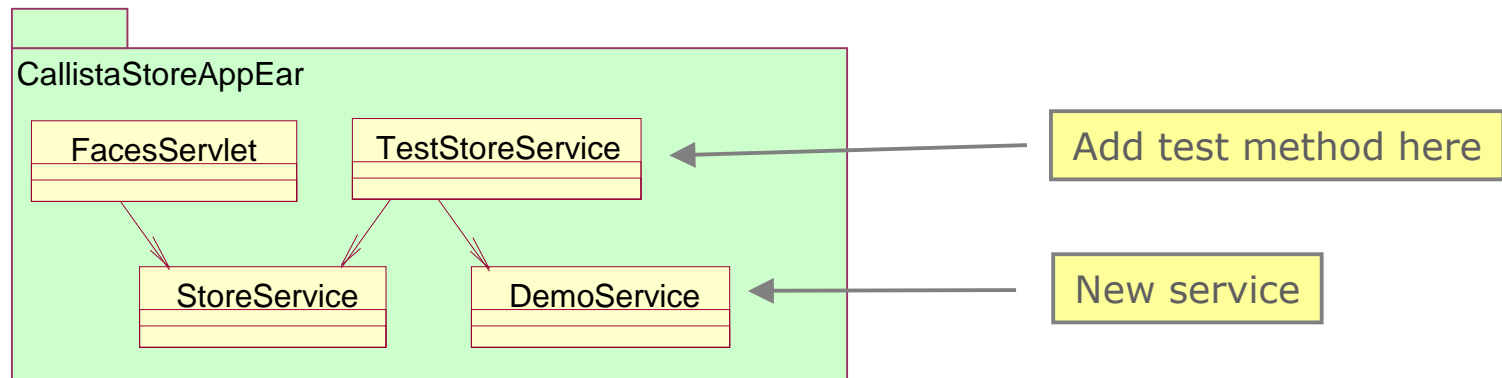   ▫ An alternative - Metadata Attributes

   ▫ History, today and future

☐ XDoclet

   ▫ What is it?

   ▫ Example

   ▫ Findings

   ▫ **Demo**

☐ Summary

CALLISTA

XDoclet - demo

☐ Add a demo-service to the CallistaStoreAppEar (EJB Session Bean)

CallistaStoreAppEar

| FacesServlet | TestStoreService |

Add test method here

| StoreService | DemoService |

New service

☐ Do it with "Test Driven Design"
1. Write a test method
2. Smallest possible implementation
   - Makes the test method to compile
3. Ensure that the test fail
4. Implement until the test succeed

CALLISTA

XDoclet - demo

☐ Demo – step by step

1. Verify that test suite succeed

2. Write a test case for a new service
   - **testDemoMethod()** in **TestStoreService.java**

3. Smallest possible implementation of the new service
   - Create a new EJB Session Bean
   - Add "empty" **demo()** - method
   - Use XDoclet-tags to make it an EJB-method
   - Run build-script

4. Ensure that the test fail
   - Run test-suite

5. Implement **demo()** - method until the test succeed

CALLISTA

XDoclet - demo

☐ Demo environment (100% Open Source)

- JDK
  - J2SE 1.4.2_03

- IDE
  - Eclipse 2.1.2
  - EMF 1.1.1
  - JBoss IDE 1.2.2

- Build tool
  - Ant 1.5.3

- Code Management
  - CVS

- Metadata Attribute driven code generator
  - XDoclet 1.2

- Test tool
  - JUnitEE 1.8

- J2EE server
  - JBoss 3.2.3

- Database
  - Hypersonic (bundled with JBoss)

CALLISTA

Summary

☐ Metadata Attributes relieve the J2EE developer from current level of details in J2EE

☐ Metadata Attribute driven development (XDoclet) offers
  ▫ Efficiency
  ▫ Control
  ▫ Quality
  ▫ Low Cost
  ▫ J2EE Portability in reality

☐ Metadata Attributes on its way into Java 1.5 and J2EE 1.5

☐ The question is not **if** using Metadata Attributes, the question is **when**!

CALLISTA