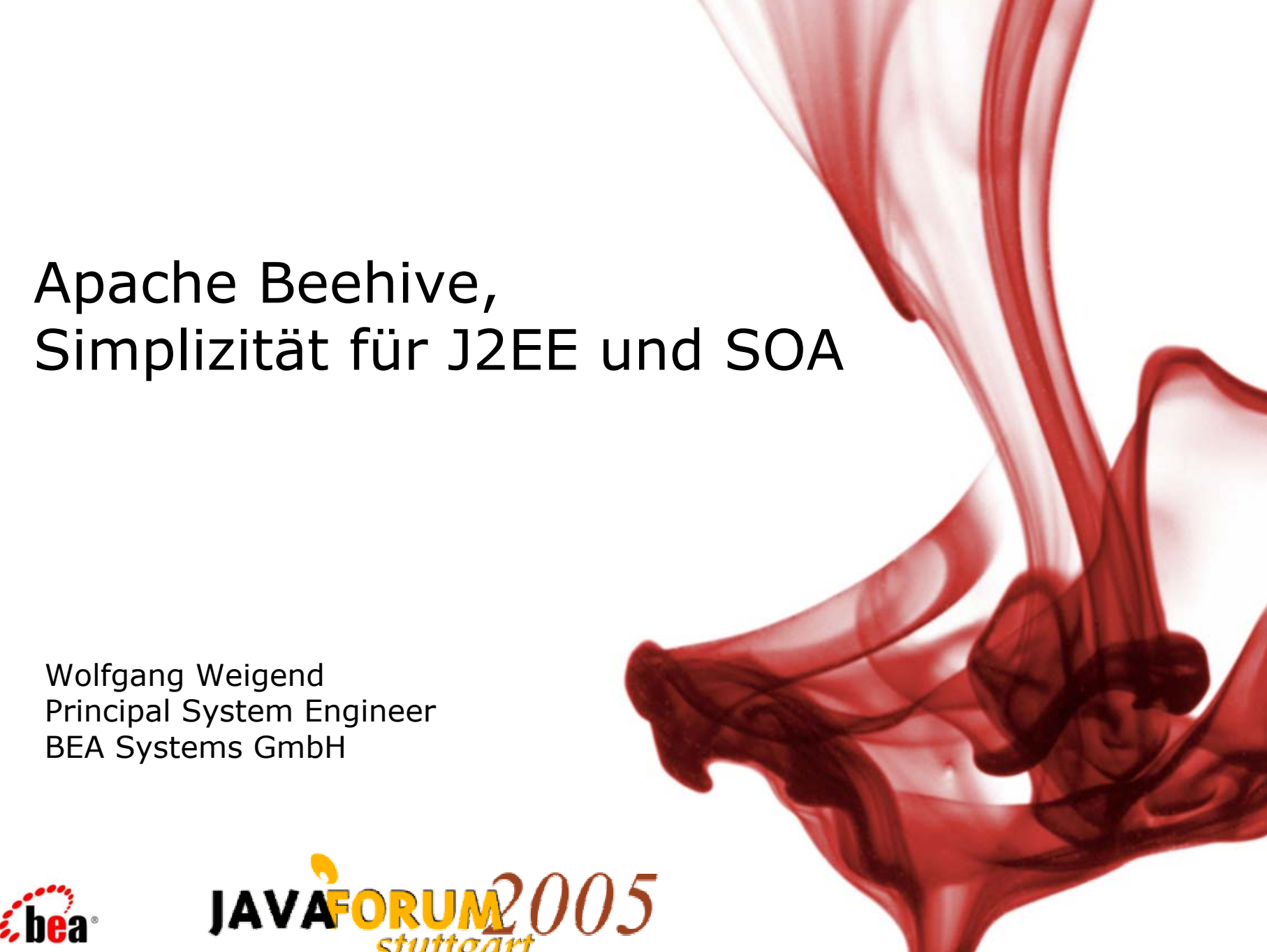
Apache Beehive, Simplizität für J2EE und SOA

Wolfgang Weigend
Principal System Engineer
BEA Systems GmbH



JAVAFORUM 2005 stuttgart

What is Project Beehive

Open-Source Framework for development

- Apache Beehive is an extensible Java application framework with an integrated metadata-driven programming model for web services, web applications, and resource access
- BEA donated Workshop Application Framework as project Beehive to the Apache Software Foundation (incubator)
- Comprehensive Open Source framework with jdk 5.0
- Beehive key components are XML Beans, Java Web Services, Java Controls and Java Page Flows
- Beehive supports JSR-175 and-JSR 181
  - JSR-175: Metadata Facility for Java Programming Language
  - JSR-181: Web Services Metadata for the Java™ Platform

bea®

Apache Beehive

- XMLBeans
  - Java and XML binding tool
  - Provides fully access to XML via Java
- Java Web Services
  - A component implementation of the JSR-181 specification that uses metadata annotations in Java methods and classes to easily build Web services.
- Java Controls
  - Lightweight component framework based upon annotated JavaBeans, exposing a simple and consistent client model for accessing a variety of J2EE resource types.
- Java Page Flows
  - A web application framework based on Apache Struts with an easy to use, single-file programming model based on JSR-175 metadata.
- Apache Beehive project status
  - Anticipating 1.0 release Summer 2005
  - Beehive has more than 25 committers to date.

bea

XMLBeans

- XMLBeans are in a separate Apache project
  - Bundled with the Beehive distribution
- XMLBeans provide a JavaBean based view of XML data
  - Without losing access to the original, native XML structure
  - XMLBeans are bound to the XML document
- XML schema is used to compile strongly-typed java interfaces and classes
  - XMLBeans supports all XML schema definitions
  - Full XML Infoset fidelity
  - XMLBean reflect into the XML schema via an XML schema object models
- XMLBeans access to XML
  - Strong-typed getter and setter
  - XQuery transformations
  - Loosly-typed via cursors

Two ways getting the data out

```
<person>
  <name>Andy Piper</name>
  <birthday>01-05-1970</birthday>
</person>
```

#1: With Java type translation:

```
String name = doc.getPerson().getName();
Date bd = doc.getPerson().getBirthday();
```

Easy!

.. But fragile

#2: Without Java type translation:

```
NodeList nodes = dom.getChildNodes();
for (int i = 0; i < nodes.length(); i++)
{
    if (nodes.item(i).getNodeName().equals("name"))
        name = nodes.item(i).getNodeValue();
    if (nodes.item(i).getNodeName().equals("birthday"))
        bd = nodes.item(i).getNodeValue();
```

Flexible!

.. But clumsy

The natural tension between XML and Java

Some examples

XML is order-sensitive, Java is not

- Java getBuyRecord(), getSellRecord() are not ordered.
- But in XML, you can know if <buy> or <sell> comes first.

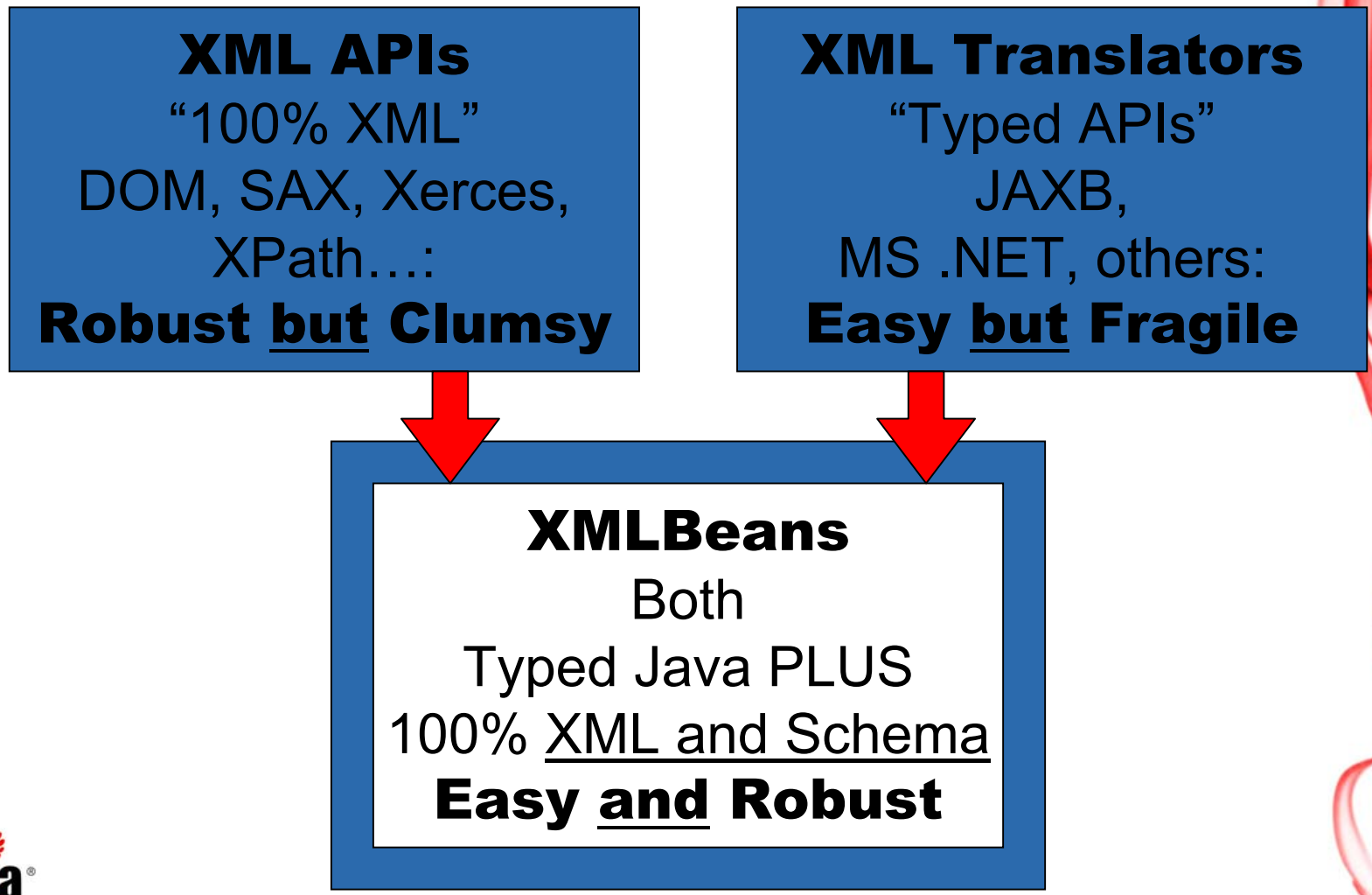
XML has late-bound features not present in Java

- Strongly typed Java compiles all methods and fields.
- But in XML, wildcards allow an instance to add "any extra attributes or elements here" – important for data evolution.

Type systems differ in fundamental ways

- Schema has simple and complex types, Java complex only.
- Schema restricts or extends, Java inherits only by extension.
- Schema has nineteen primitive types, Java has six.

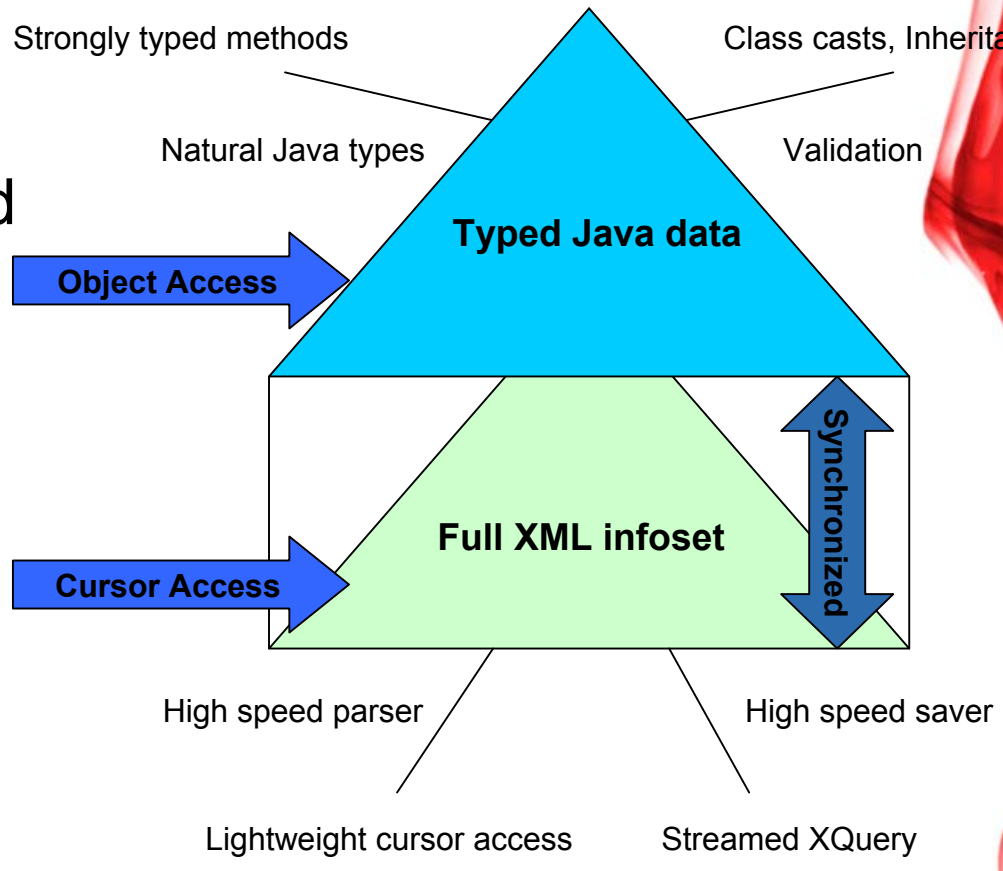
XMLBeans API Approach



XMLBeans Data Model

Convenient Typed JavaBean Tree

Complete XML Token Stream



Basic Approach for Developer

- Load XML using the XMLBean loader

```
XmlObject obj = XmlLoader.load(new File("purchase.xml"));
```

- You can use a cursor to see your document

```
XmlCursor c = obj.newCursor();
c.toFirstElementChild();
QName q = c.getName(); // etc.
```

- If you have a schema, compile it into XMLBeans

```
... -classpath="xmltypes.jar"
```

- Now you can coerce your XML to a compiled type

```
XmlObject obj = XmlLoader.load(new File("purchase.xml"));
PurchaseOrderDocument doc = (PurchaseOrderDocument)obj;
PurchaseOrder po = doc.getPurchaseOrder();
LineItem[] items = po.getLineItemArray();
for (int i = 0; i < items.length; i ++)
    System.out.println(items[i].getDescription());
```

XMLBeans API Overview

Four major parts of the API

XmlLoader is the front door

```
XmlLoader.load methods
```

XmlObject is the base class for all instances

```
xobj.schemaType(), xobj.newCursor(), xobj.validate()
methods

CustomerDocument cdoc = (CustomerDocument)xobj; coercion
```

XmlCursor is used to navigate XML trees

```
xcur.toNextElement(), xcur.toNextToken() navigation

xcur provides full XML Infoset visibility
```

SchemaType is the xsd type system reflection API

```
stype.isSimpleType(), stype.getContentModel() reflection

stype provides full XSD Schema Type inspection
```

Java Web Services (JSR-181)

- JSR-181 Web Service Metadata for Java
  - Defines annotations to enable easier web service programming

```
@WebService public class ZinsRechner {
@WebMethod public float berechneZins(String kreditVertrag) { }
}
```

- Abstract SOAP marshalling, Java-to-XML binding, WSDL file creation, underlying bean deployment, ...

- J2EE 1.4 Web service infrastructure is used (JAX-RPC 1.1) and JSR-175

Java Web Services (JSR-181)

- JSR 181 provides a defined set of JSR-175 tags
- Generates Web Service infrastructure for annotated code
- WSDL 1.1 definitions linked with existing code
- Service implementations should be tight to the JavaBean standard

Java Controls

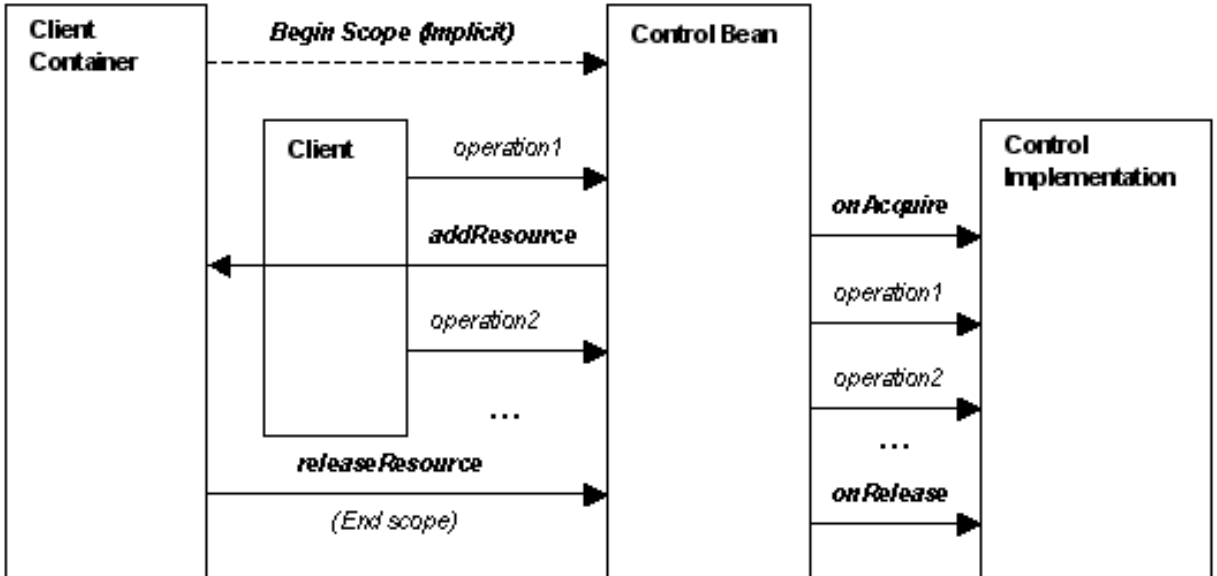
- Access business logic in a consistent and straight forward way as a POJO
- Can be used from different clients
- J2EE experts can encapsulate their knowledge into controls, and could be reused by other developers
- Can be customized with annotations (JSR-175)

Java Controls

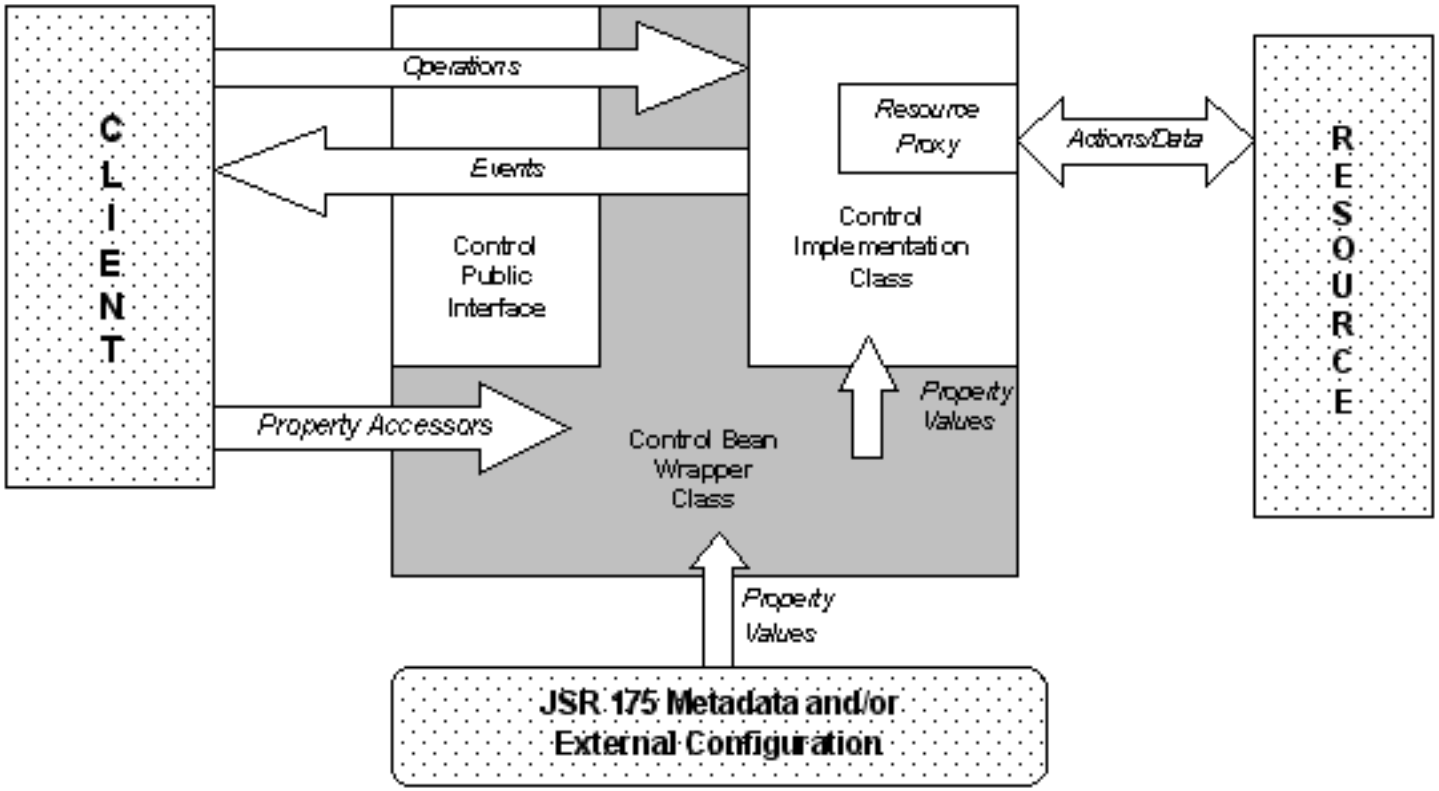
- Provide a consistent model for discovery of
  - Resource operations (actors)
  - Resource events (notifications)
  - Configuration options (properties)
- Client programming model based on JavaBean type
  - JavaBean wrapper around the control implementation
  - Dynamic properties stored per instance
  - Property resolution services
  - Event listener routing
  - Management for context services and nested controls

Java Controls

- Resource management and contract
  - Acquisition mechanism and resource release guarantee
  - Cooperates with outer container to manage resources
- Extensibility model
  - Define operations for specific resource use cases
  - Customized resource facades using metadata
- Developer focus on „what", not on „how" to do

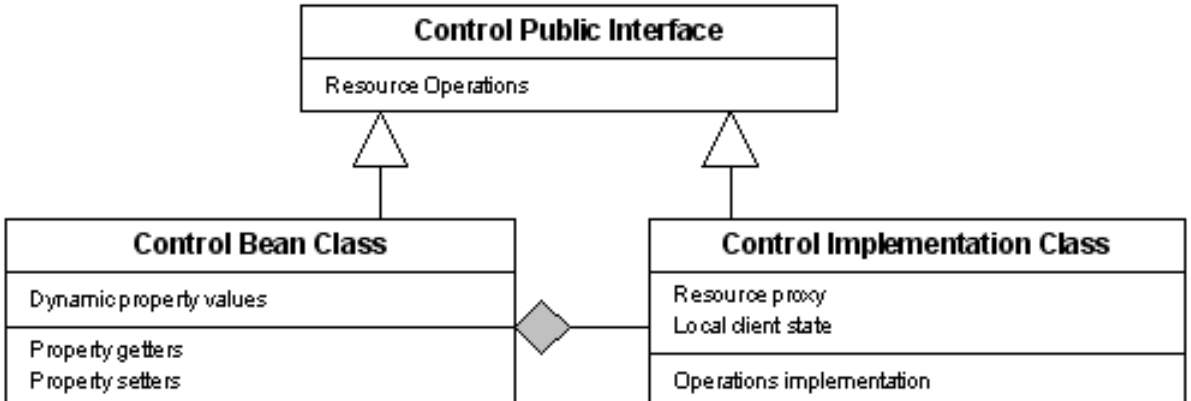


Java Control Architecture



Java Controls

- Public interface
  - Source file defines the set of operations, events, extensibility model, and properties associated with the control type
- Implementation class
  - Source file provides the implementation of the operations and extensibility model described by the public interface
- ControlBean generated class
  - Code generated JavaBean class is derived from the public interface, and the implementation class from the control compiler



Java Controls

- The programmatic client model follows the basic pattern of JavaBeans construction and event handling

```
TimerControlBean myTimerBean =
   (TimerControlBean)Controls.instantiate(classloader, "com.myco.TimerControlBean");
myTimerBean.setTimeout("3 seconds");
myTimerBeans.addTimerControlEventListener(
     new TimerControlEventListener()    // anonymous event handler class
     {
         public void onTimeout(long time)
         {
             // timer event handling code
         }
     }
);
```

- The following example is equivalent to the preceding example, but uses declarative style and construction and event routing

```
@Timer(timeout="3 seconds") TimerControlBean myTimerBean;

...

public void myTimerBean_onTimeout(long time)
{
    // timer event handling code
}
```

bea

JSR-175 Based Annotations

Annotations in Java J2SE 5.0

- Special form of Interface definition
- Defines meta-annotations for location, scope, inheritance, etc.

Benefits

- Lends credibility to "annotated" Java programming model

Limitations

- Annotations are types instead of comments
  - Results in compile time dependencies
- Limited number of supported datatypes
  - No support for Date, BigDecimal, User Defined Classes, etc
- No Inheritance of annotation *definitions*
  - Reuse is limited to a composition model
- No model for external override

Addressing Limitations

Type Limitations

- We've defined meta annotations to constrain "base" types
  - Date
  - Bounded Range
  - Decimal
  - and others (comparable to tag-grammar file in WLW 8.1)

External Overrides

- We've defined a meta annotation to indicate that an annotation can be externally overridden.
  - Applies to the entire annotation
  - Some member types don't make sense (e.g. Class)

Java Web Service Annotation Example

WLW 8.1

```
/** @common:target-namespace namespace="tns" */
public class Counter implements com.bea.jws.WebService
{
  static final long serialVersionUID = 1L;
  /** @common:operation @jws:conversation phase="start" */
  public void doStart() {}
}
```

WLW 9.0

```
@javax.jws.WebService(targetNamespace="tns")
public class Counter implements java.io.Serializable
{
  static final long serialVersionUID = 1L;
  @weblogic.jws.Conversation(value = weblogic.jws.Conversation.Phase.START)
  @javax.jws.WebMethod()
  public void doStart() {}
}
```

bea

Control Annotation Example - Producer

```
@ControlDisplay( resourceBundle="com.bea.control.Resources" )
@ControlInterface (defaultBinding="com.bea.wlw.runtime.core.control.DatabaseControlImpl")
public interface DatabaseControl extends Control
{
  public enum CommandType
  {
    NONE, GRID, DETAIL, UPDATE // and more ...
  }

  @PropertySet
  @Inherited
  @AnnotationConstraints.AllowExternalOverride
  @Retention(RetentionPolicy.RUNTIME)
  @Target({ElementType.TYPE, ElementType.FIELD})
  public @interface ConnectionDataSource
  {
    @AnnotationMemberTypes.JndiName
    (
      resourceType = AnnotationMemberTypes.JndiName.ResourceType.DATASOURCE
    )
    String jndiName();   // no default ... value is required
  }
...
}
```

Annotation Definition

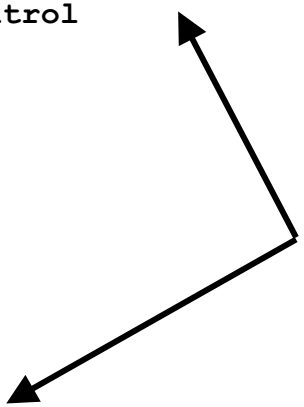
Member Constraint

Control Annotation Example - Consumer

```
@DatabaseControl.ConnectionDataSource(jndiName = "cgSampleDataSource")
@org.apache.beehive.controls.api.bean.ControlExtension()
public interface CustomerDb extends DatabaseControl
{
  static final long serialVersionUID = 1L;

  static public class Customer
  {
    public int id;
    public String name;
  }

  @DatabaseControl.SQL(statement = "SELECT …")
  Customer findCustomer(int id);
…
}
```

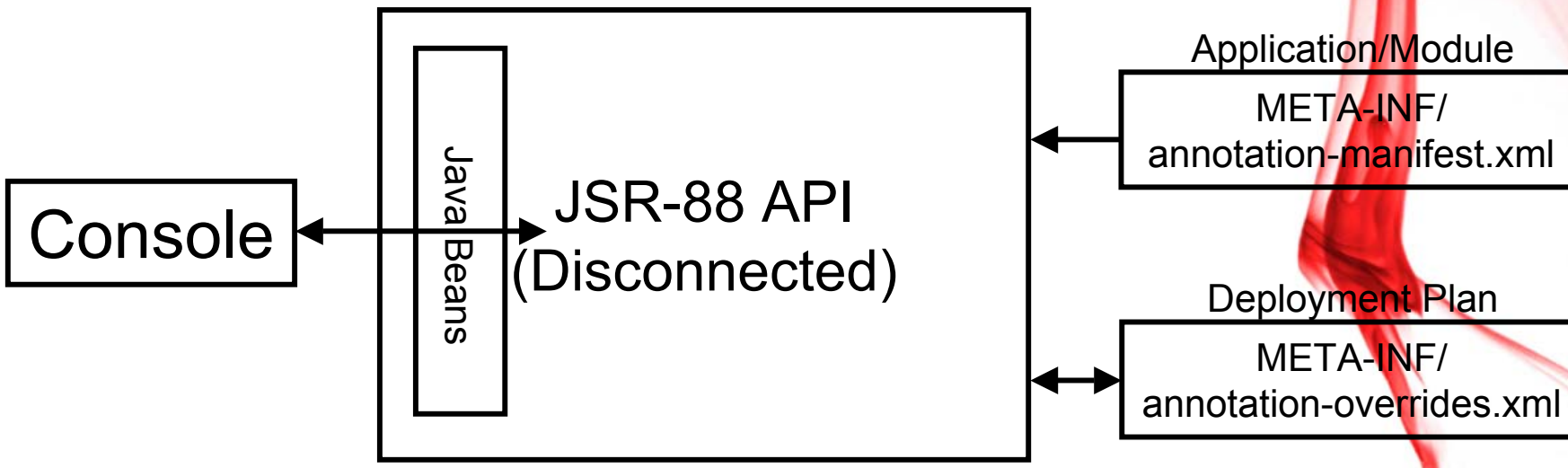


Annotation Overrides

Approach is to integrate overrides with JSR-88

- Build process generates an annotation-manifest.xml
  - Contains annotations (and metadata) that can be overridden via the console.
- weblogic-extension.xml is used to
  - Registers the annotation-manifest.xml with the WLS JSR-88 implementation
    - Makes annotations and overrides visible in the JSR-88 DeploymentConfiguration API
  - Declare a custom module as part of the application
    - Provides access to the deployment plan and a listener for plan changes
- Console interacts with JSR-88 to read/write the plan

JSR-88 J2EE Application Deployment



JSR-88 Configuration Overview:

(1) Clients "Load" a DeploymentConfiguration

(2) The Configuration is exposed as JavaBeans

(3) Clients edit the Configuration

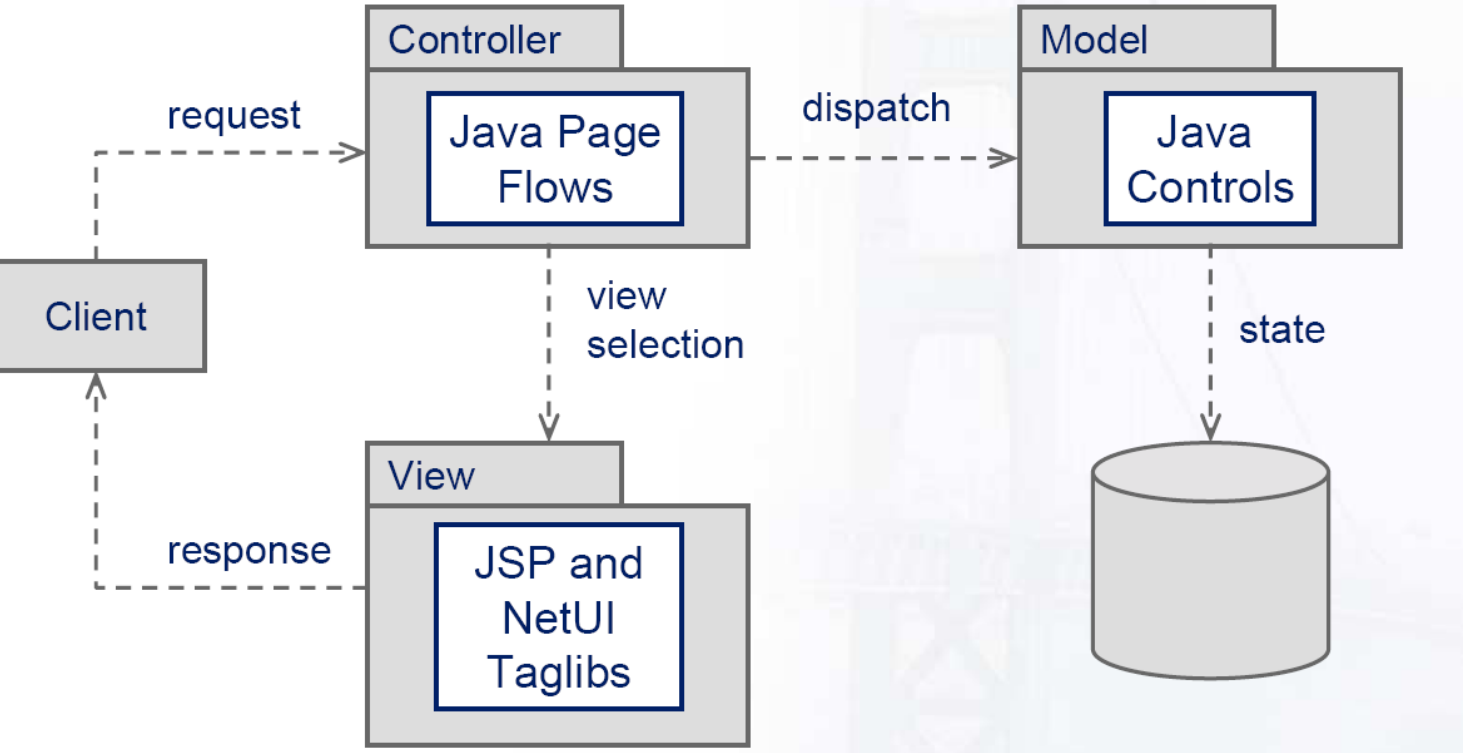
(4) Clients request that the Configuration be "Saved"

(5) The "Serialized" Configuration is a Deployment Plan

Java Page Flows

- MVC web application framework built on Apache Struts
- Manages the presentation and interaction flow for the web application
- Focused on the controller
- Unit of code and metadata (JSR-175)
- Supports Java Server Faces for a view
- Basic elements
  - Controller: page flow class
  - Actions: annotated methods
  - Exception Handlers: annotated methods
  - Page Flow State: member fields

Java Page Flows



http://incubator.apache.org/beehive/

Welcome to Beehive! Our goal is to make J2EE programming easier by building a simple object model on J2EE and Struts. Using the new JSR-175 and JSR-181 metadata annotations, Beehive reduces the coding necessary for J2EE. The initial Beehive project has three pieces.

• NetUI Page Flow – An annotation-driven web application framework built on Struts that centralizes navigation logic/metadata/state in *reusable, encapsulated* "page flow" controller classes. It provides an integrated set of JSP tags, as well as first-class integration with JavaServer Faces and with raw Struts.

• Controls – Lightweight component framework that helps programmers build components that incorporate metadata into their programming model. This projec comes with a few pre-made controls as well, for example, see the Database Control Sample.

• Web Services – An implementation of JSR-181, an annotation-driven programming model for web services.

bea

Beehive Controls

•Assembly and Binding: Definition of "assembly", a build-time stage where controls can do code-gen and side-effect their J2EE container. Discusses an approach to allowing users to specify implementation bindings for control types.

•Controls Packaging: Describes the base JAR packaging model for controls, how it relates to JavaBeans packaging and introspection, and the annotation syntax for setting manifest and descriptor information.

•Controls Threading Model: Describes the threading model for control users and developers.

•Controls Versioning: Describes the annotations and semantics around versioning o controls source artifacts.

•Implementing Annotation-based Programming Model Features: Describes a genera mechanism for adding annotation-based features to the controls programming mode via JavaBean contextual services.

•System Controls: A set of starter controls to make the runtime more generally usefu out of the box. Beehive includes four "system" controls -- JDBC, JMS, EJB, and web services. And Testing Controls

bea

Beehive Distribution Structure

```
ant/
    beehive-tools.xml
    beehive-runtime.xml
  docs/
    ...
  lib/
    common/
      *.jar
    controls/
      *.jar
    netui/
      resources/
      *.jar
    wsm/
      *.jar
  samples/
    ...
  beehive-imports.xml
  README.txt
  LICENSE.txt
  INSTALL.txt
  NOTICE.txt
```

Distribution Content Description

- the ant/ directory contains the Ant support for obtaining various configurations of the Beehive runtime and for building Beehive-enabled projects / applications.
- the lib/ directory contains the runtime bits divided by Beehive sub-project.
- the lib/common/ directory contains the runtime bits that are shared between two or more sub-projects.
- the lib/controls/ directory contains the controls runtime, Velocity JARs, and Beehive provided system controls
- the lib/netui/ directory contains the NetUI runtime, compiler, tag libraries, Struts runtime, and webapp files
- the lib/wsm/ directory contains the WSM runtime and compiler JARs
- the samples/ directory contains Beehive samples for Controls, NetUI, and WSM. In addition, "blank" projects are included for these three.

What's NOT in a Distribution?

Beehive users need to install following software packages (not included in distributions):

- JDK 5.0
- Tomcat
- Ant

bea

Beehive Developer Information

Useful to committers and contributors to Beehive

Contents

Beehive Releases

- V1Alpha
- V1Beta
- V1ReleaseCriteria
- 1.0m1 Known Issues

Processes

- *Feature Planning*
- Release Process
- *How to Cut a Release*

Apache Beehive - Downloads

- Beehive 1.0m1 Binary Distribution
- Daily Binary Distributions
- Older Releases
  - Beehive 1.0 Beta Binary Distribution
- Verifying Downloaded Files
  - Verifying with PGP
  - Verifying with GnuPG
  - Verifying with MD5
- Making Your Own Distribution

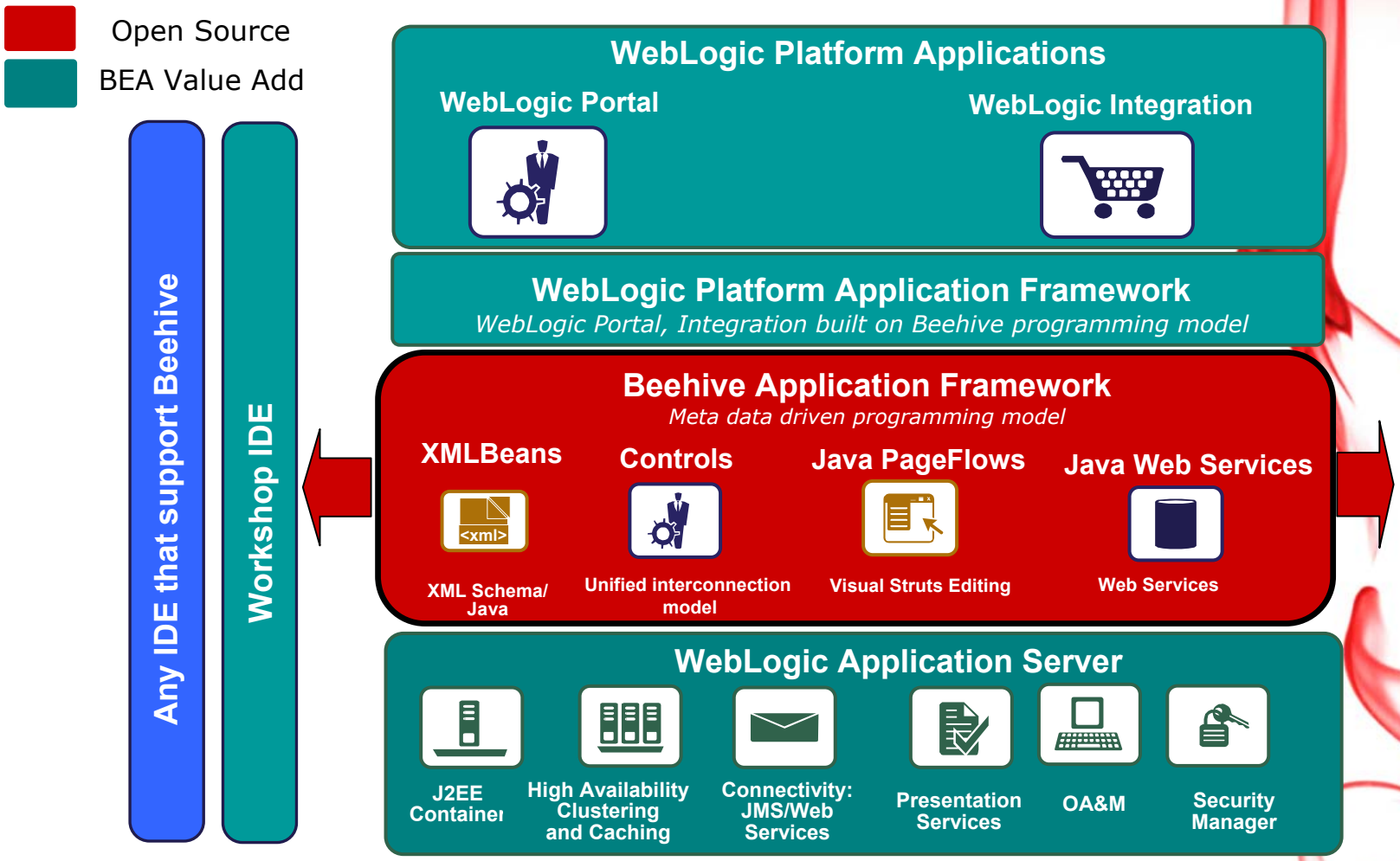
Beehive News

- Nightly builds are now available at http://cvs.apache.org/dist/incubator/beehive/nightlies/
- The Beehive Beta distribution is now available
- Beehive has a wiki
- Bugs or feature requests can be filed in our issue tracking system: JIRA
- Pollinate is an Eclipse technology project building an IDE for the Beehive Framework

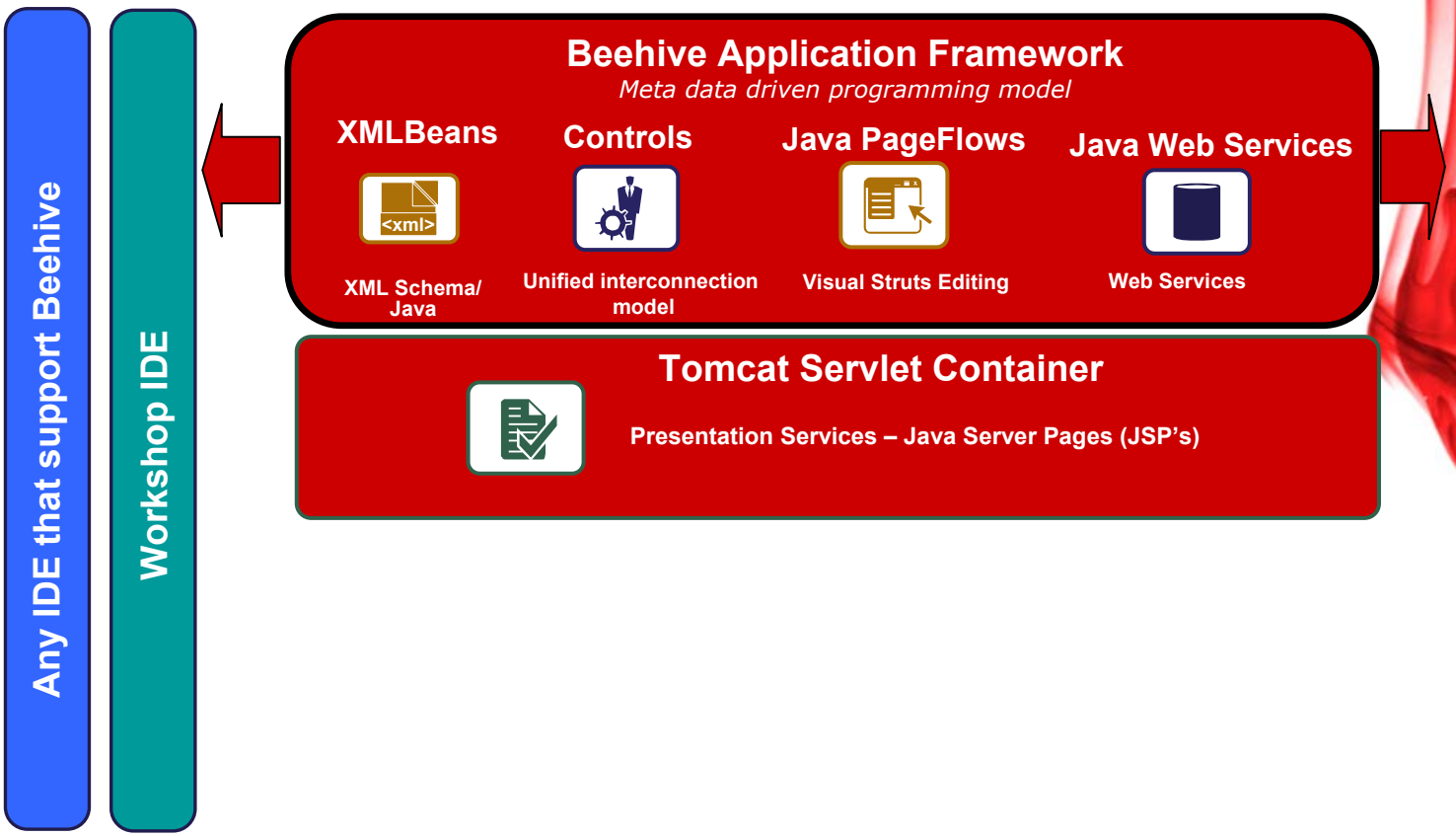
References

- [Apt] Sun Microsystems. Annotation Processing Tool (apt). http://java.sun.com/j2se/1.5.0/docs/guide/apt/index.html.
- [Axis] Apache Software Foundation. Web Services – Axis. http://ws.apache.org/axis/.
- [JSR-101] R. Chinnici. Java APIs for XML based RPC. http://www.jcp.org/en/jsr/detail?id=101.
- [JSR-109] J. Knutson. Implementing Enterprise Web Services. http://www.jcp.org/en/jsr/detail?id=109.
- [JSR-175] G. Bracha. A Metadata Facility for the Java Programming Language. http://www.jcp.org/en/jsr/detail?id=175.
- [JSR-181] M. Mihic, J. Trezzo. Web Services Metadata for the Java Platform. http://www.jcp.org/en/jsr/detail?id=181.
- [Beehive] A. Bien. Presentation „SOA with Beehive“.

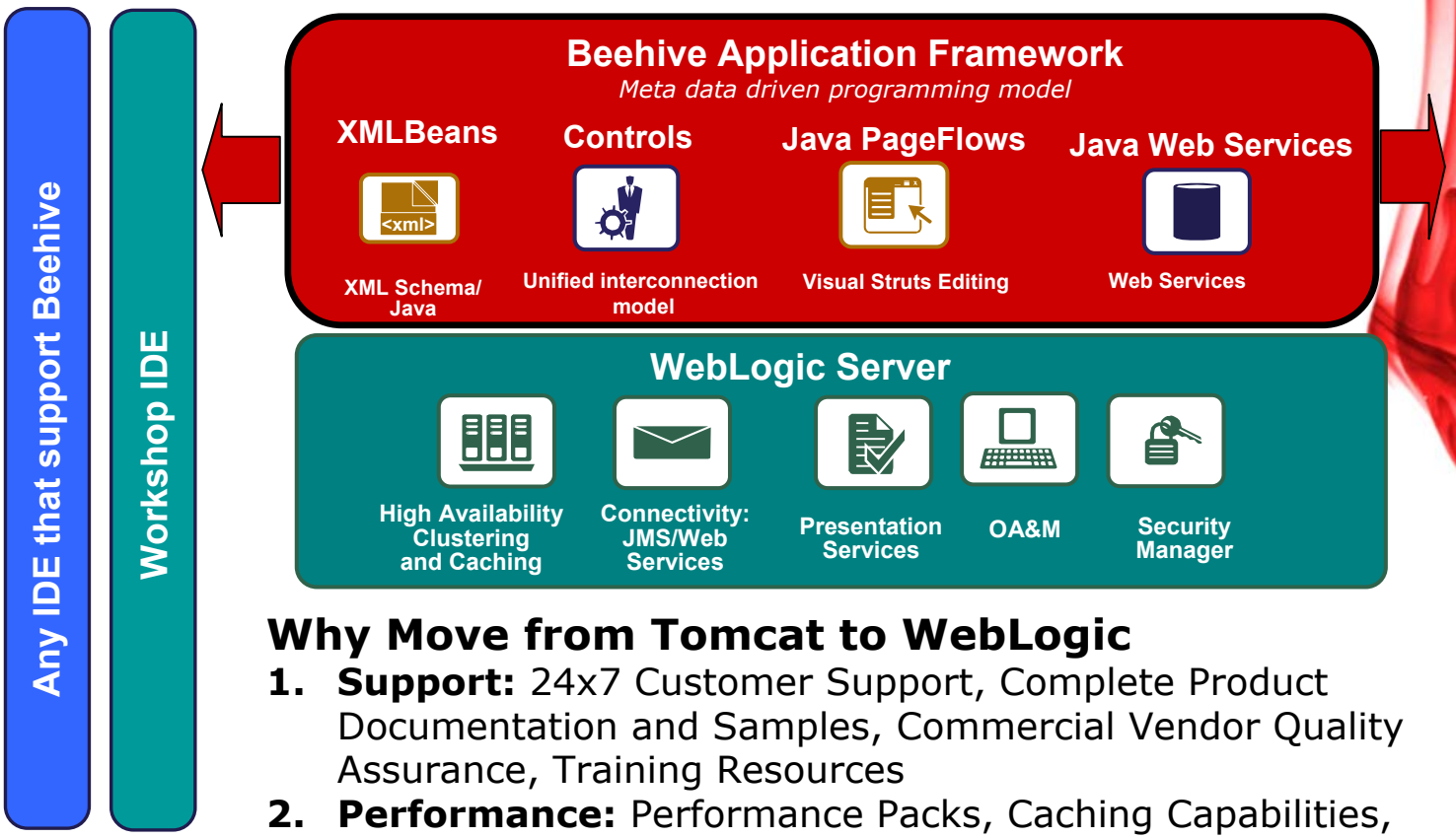
Project Beehive Components



Project Beehive on Tomcat



Project Beehive on WebLogic

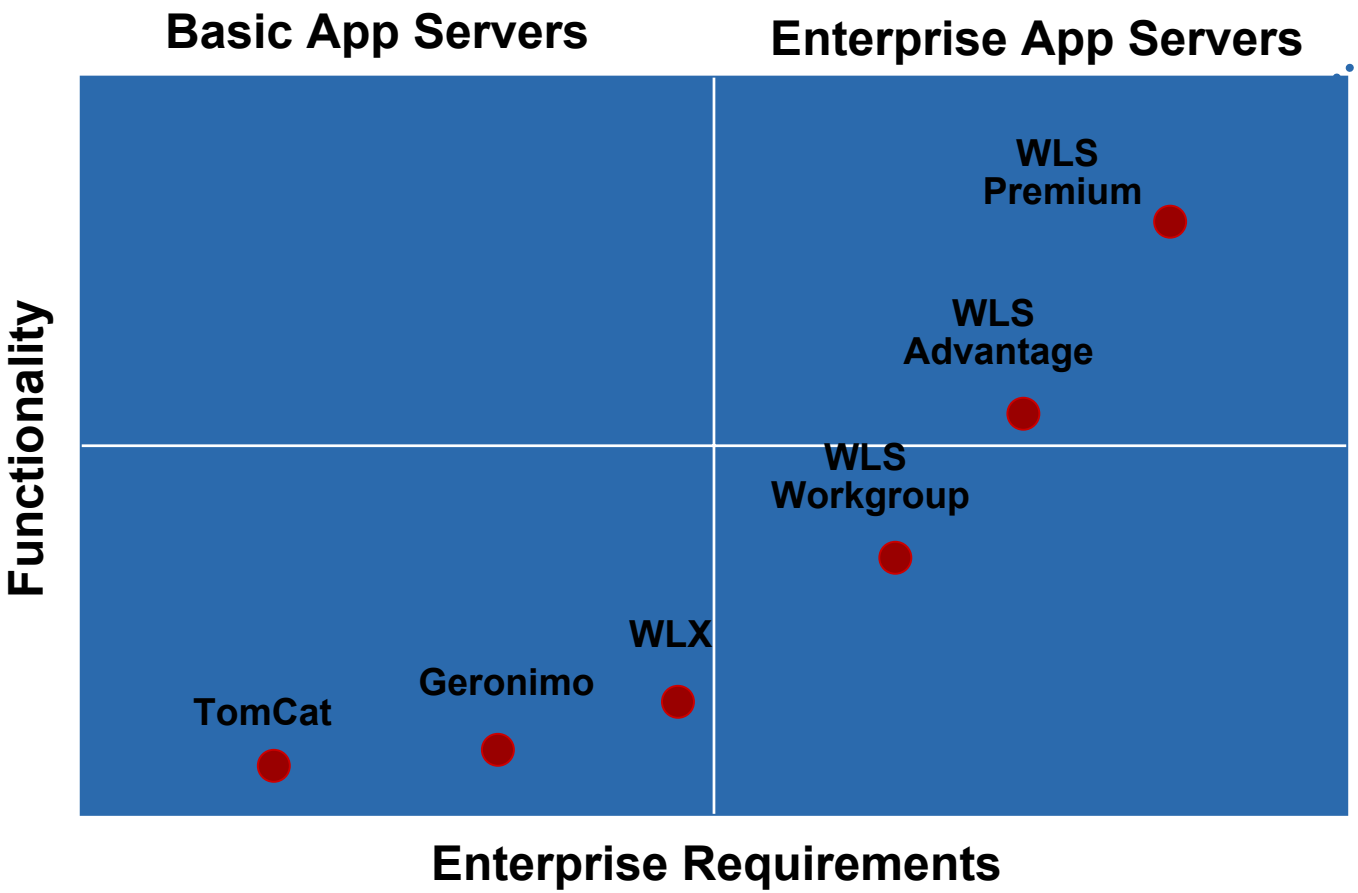


Why Move from Tomcat to WebLogic

1. **Support:** 24x7 Customer Support, Complete Product Documentation and Samples, Commercial Vendor Quality Assurance, Training Resources
2. **Performance:** Performance Packs, Caching Capabilities, Connection Pooling
3. **Reliability:** Clustering Capabilities, Load Balancing and Automatic Fail over, Transaction Manager, Transaction Recovery Service
4. **Manageability:** System Administration and Monitoring, Migration Tools, Built-in Security Framework

bea

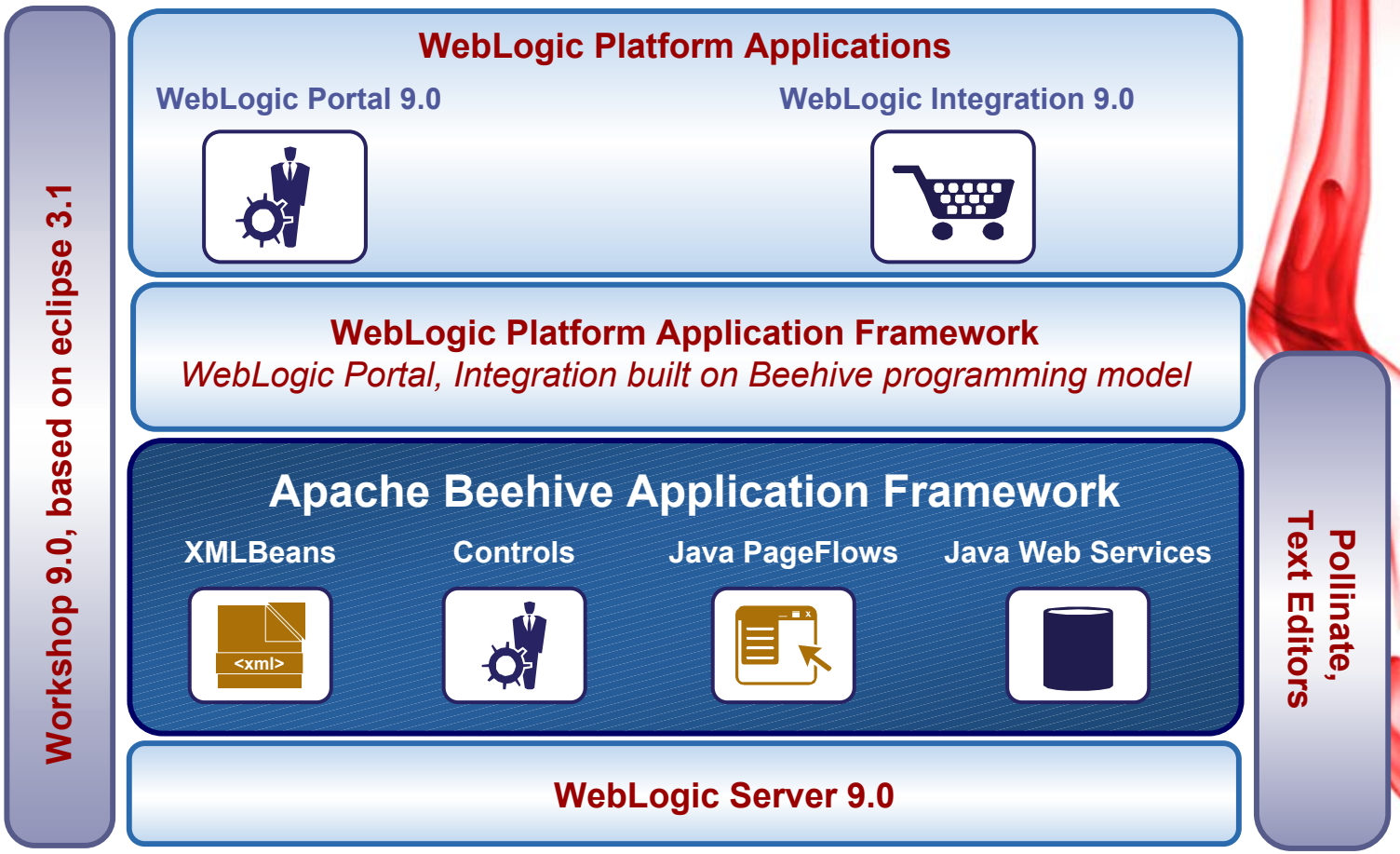
Upgrade Path from OSS



WebLogic Platform

bea

Development Tools with Platform 9.0



Baseline Beehive Applications also run on JOnAS, Geronimo, Tomcat


bea®
Think liquid.™