

# BEA and OSS

## Leading Java Innovation

Mark Heeren



# BEA Driving Java Innovation

## Looking Back

### Challenge

- **Too many moving parts and redundant services**
- **J2EE development is tedious and complex**
- **Hard to build a service-oriented architecture**

### WebLogic 8.1 Solution

**Integrated platform**

**Workshop IDE with runtime framework**

**Control based service model**



# BEA, Standards and OSS

- **Standards critical to technology adoption and market creation**
  - API standards for investment protection, protocol standards for interoperability
- **OSS complements BEA's standards strategy by promoting innovation**
  - *Innovate then Standardize vs Standardize then Innovate*
  - Successful projects are architected for *participation*
  - OSS is a standardizing market force through portability with no single vendor lock-in
- **BEA - an active OSS community member**
  - Patronage of OSS projects to provide BEA support and leadership for emerging technologies (CodeHaus, AspectWerkz, ControlHaus)
  - Open source BEA innovations (Beehive, XMLBeans)
  - Join existing efforts (AspectJ, Eclipse)



# BEA Doubledown on OSS: Beehive + Eclipse

- **BEA donates Beehive to Apache Software Foundation (May 2004)**
  - OSS successor of WebLogic Workshop framework
- **BEA joins Eclipse Foundation as Strategic Developer (February 2005)**
  - Board membership
  - Co-lead for Web Tools Platform project (WTP)
- **BEA has already contributed technology to Eclipse**
  - jRokit JVM plug-in: *Profiling and Memory Leak Diagnosis at production speed*
  - AspectJ(/AspectWerkz) plug-in: *AOP development supporting annotations*
  - Pollinate plug-in: *(visual) Beehive application development*
- **BEA proposes new technologies to Eclipse**
  - Language Development Tools project: *generic extensible language compiler toolkit*



# BEA's Eclipse Commitment



- **BEA rebuilds all central tooling for WebLogic products on top of Eclipse and Beehive (Fall 2005)**
  - WebLogic Workshop 9.0 'Daybreak' release
  - Provide patches and many new plug-ins to Eclipse
  - Contribute to exemplar plug-ins, lowering shared development cost
  - Offer additional commercial plug-ins
- **Strategic Developer Status**
  - Highest level of participation: board member, architecture, planning, requirements council members; project leadership
  - Other members (8) include Borland, CA, IBM, Intel, Sybase
  - Significant contribution in money and people: BEA architects, developers, testers, program managers
    - Initial participation: ~16 people @ 100% + ~7 people @ 50%
- **Focusing on sub-projects strategic to BEA value proposition**
  - Web Tools Platform Project (co-lead)
  - Language Development Tools Project (proposal)
  - Pollinate
  - AspectJ (merged with AspectWerkz)



# What does BEA get from Eclipse?



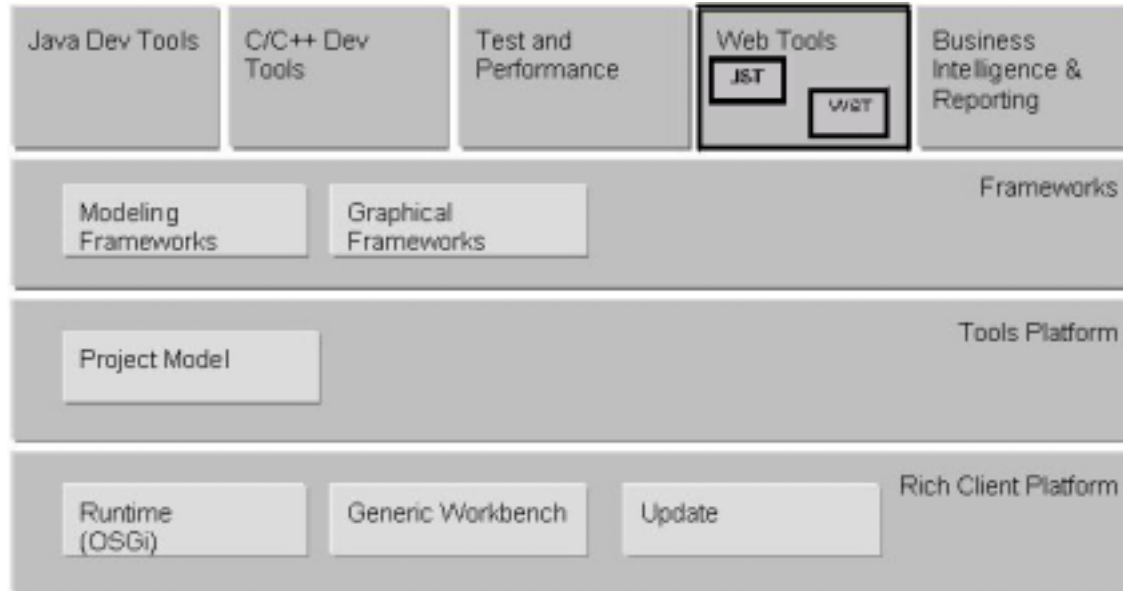
- **Faster and wider technology adoption**
  - De facto IDE standard
  - ISVs will be more motivated to extend an Eclipse platform
  - Customers will be more motivated to adopt an Eclipse platform
- **Solid base IDE implementation**
  - Inherit wealth of Eclipse plug-ins and Core IDE features
  - BEA can focus on value added features like Beehive / Pollinate programming model (Workshop experience on Eclipse)
- **Massive, growing developer community**
  - Marketing opportunity



<http://www.eclipse.org/community/index.html>



# Web Tools Platform Project



## ■ J2EE Standard Tools (JST):

- J2EE project natures, builders, module views
- JSP editor, wizards
- Servlet editor, wizards
- EJB editor, wizards
- J2EE server tools, adapters (servers view, run on server)
- Web Service wizards, engine adapters (e.g. Axis)

## ■ Web Standard Tools (WST):

- Web project natures, builders
- Html, css, javascript, xml, dtd, xsd, wsdl (graphical) editors
- Web browser, proxy config,
- Web Services UDDI explorer, WSI- validator
- Rdb connection wizard, server explorer, data browser, sql editor





# BEA and Eclipse WTP

- **Contribute to / lead the Web Tools Platform project**
  - WebLogic Workshop takes dependency on WTP 1.0
  - BEA starts by contributing patches, not many new plug-ins
    - It appears WTP can use more help in working with what it has.
  - BEA architects also help solidify public APIs
  - *Shipping WTP v1.0 this summer is a priority for everyone!*



# Language Development Tools Project

- **BEA new Technology subproject proposal**
- **Based on extensible Javelin compiler engine inside WebLogic Workshop**

- **Initial committers:**

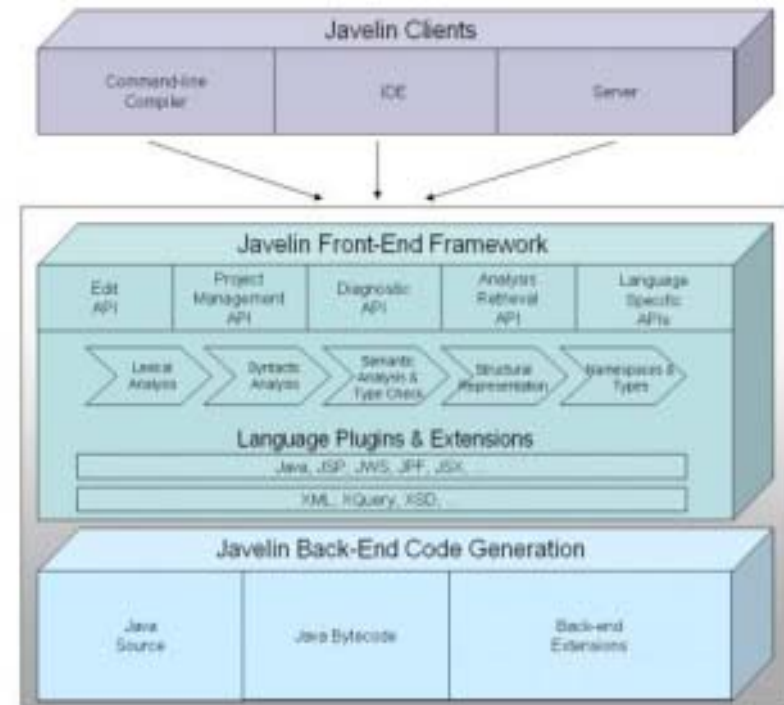
- BEA Javelin developers
- JDT committers
- others?

- **Long Term Goals:**

- develop cross/multi-language analysis and compiler services
- supporting language-aware services throughout Eclipse
- ...without the cost of duplication

- **Short-term Goal:**

- integrate annotation processing / awareness into JDT, ideally for JDT 3.1



# Java Trends

- **Power to the POJO**
  - Free components from heavyweight containers introducing complexity and container dependency
  - No lock-in to specific business component container (like EJB)
- **Metadata annotation**
  - Configuration / enrichment
  - Beehive, Spring, Hibernate, AspectJ, EJB3
  - Doclet -> JSR 175 (Java 5 annotations)
- **Flexible SOA programming model (service authoring & consumption)**
  - XMLBeans, Beehive WSM
- *OSS projects are driving the innovation for the next standard Java programming model*





# Apache Beehive Update

## What is Apache Beehive?

*Extensible Java application framework with an integrated metadata-driven programming model for web applications, web services and resource access*

## Beehive Technologies

- Java Page Flows
  - Web application MVC framework, Struts based, with easy-to-use, single-file programming model based on JSR-175 metadata annotations
- Java Web Services
  - Easily expose Web Services by annotating Java classes. Reference implementation for JSR-181. Leveraging Axis.
- Java Controls
  - Lightweight component framework based upon annotated JavaBeans, exposing simple and consistent client model for accessing variety of J2EE resource types

## Beehive Project Status

- Anticipating 1.0 release Summer 2005
- Beehive has 26 committers to date
- Sponsored and mentored by Craig McClanahan (Struts, JSF)



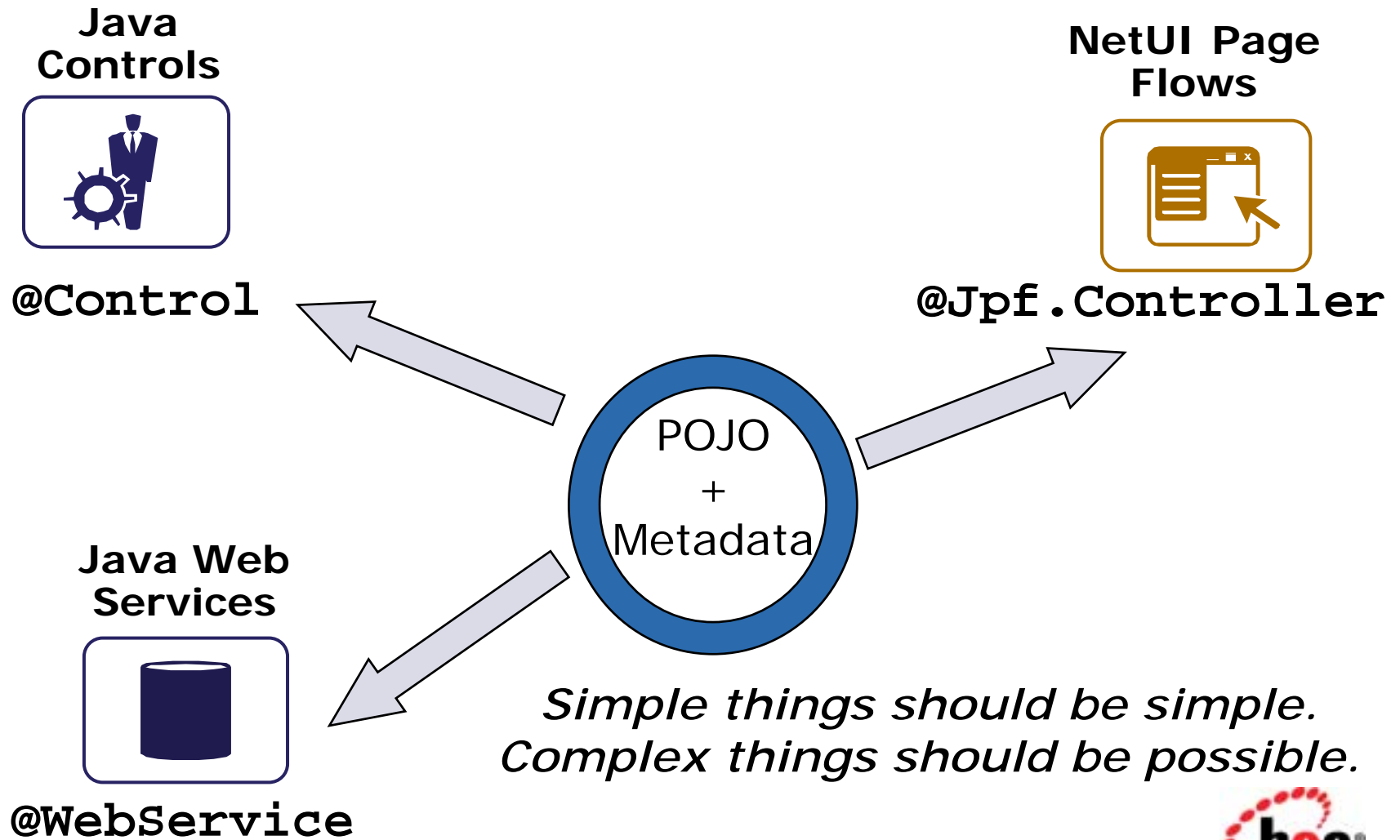


# Beehive's Ease of Use Mission

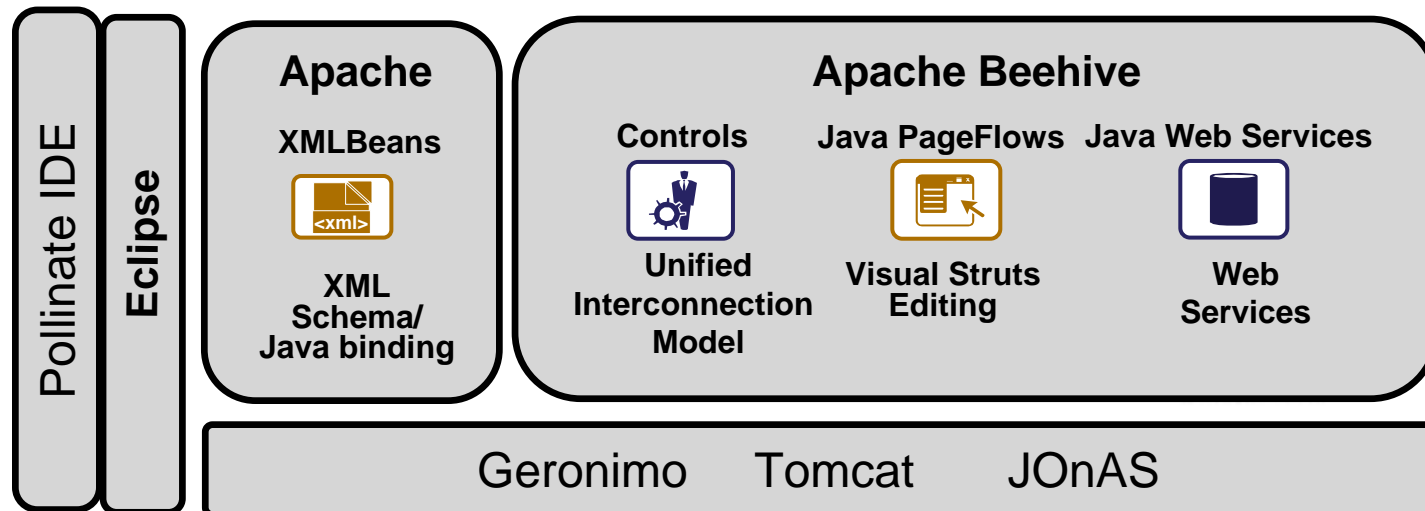
- **Deliver a comprehensive, easy-to-use framework for building dynamic J2EE and SOA applications**
  - Based on over 3 years of experience “making it easy” with WebLogic Workshop
  - Designed with tool-ability in mind from the outset
- **Strengthen Java market by providing world-class ease-of-use to all Java developers**
  - Target less experienced Java developers as well as J2EE experts
  - Productivity boost
- **Provide ultimate investment protection and transparency through open source and portability**



# Beehive simplifies web development

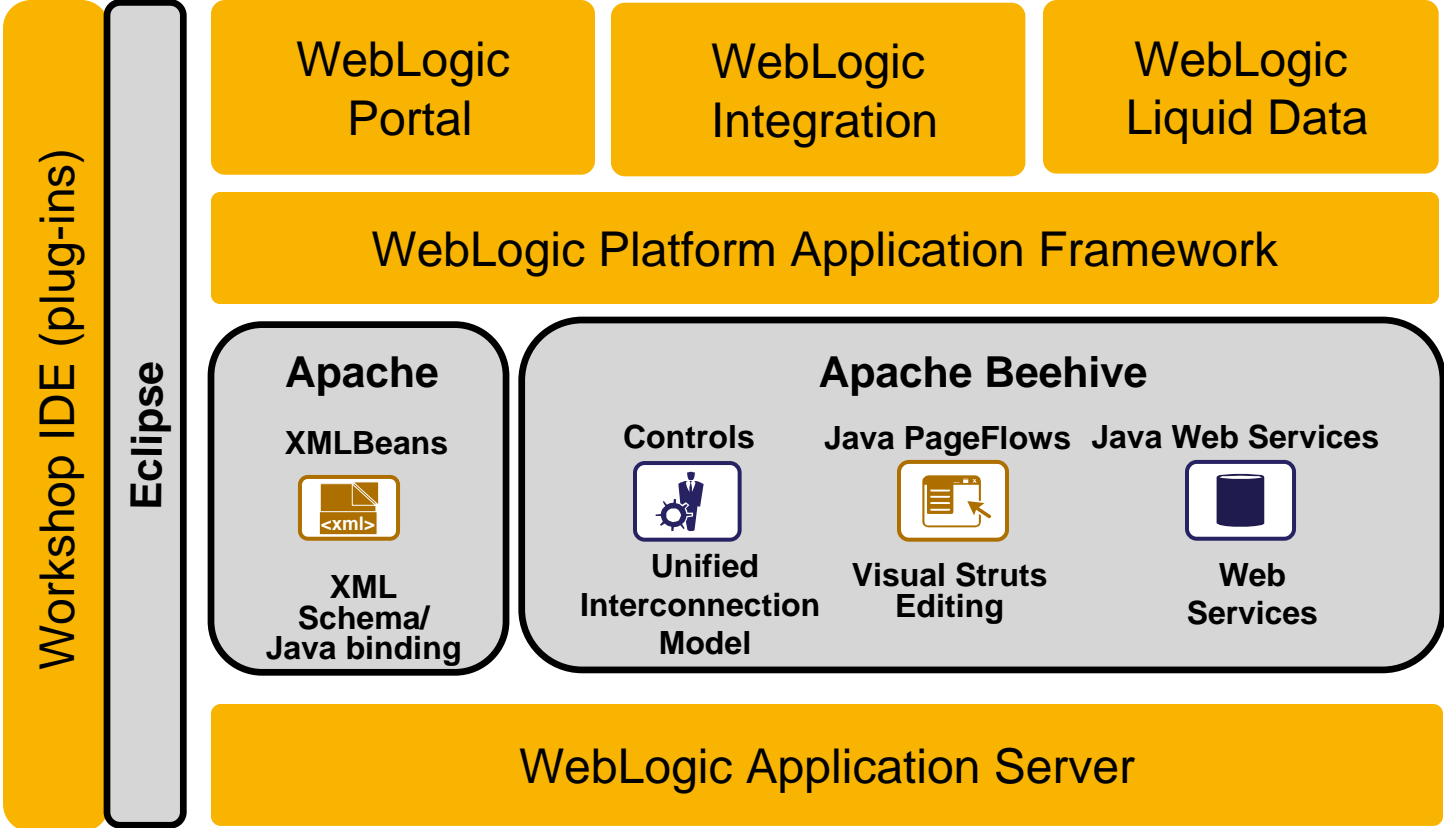


# Beehive in the OSS community



# Beehive in WebLogic

*Beehive within the BEA WebLogic Platform:*



 Open Source     BEA Value Add





# Apache XMLBeans

- **XML-Java binding tool allowing access to the full power of XML in a Java-friendly way**
- **XML Schema definitions (XSD) compiled into strongly-typed Java interfaces**
  - Access to XML instance data through familiar getter / setter accessors
  - Full XML Schema type support (compiles any valid XSD)
  - Full XML Infoset fidelity (e.g. order, comments, white space); holds in-memory store of XML doc
  - Reflect into the XML schema itself through an XML Schema Object model (e.g. list valid enum type values)
  - XQuery and XPath support
  - XML traversal through XMLCursor
- **XMLBeans is in a separate Apache project but will be packaged with the Beehive distribution**



# Apache XMLBeans: Better DTO's

```
<xs:complexType name="MoneyType">
  <xs:sequence>
    <xs:element name="Value">
      <xs:simpleType>
        <xs:restriction base="xs:decimal">
          <xs:pattern
            value="(\d){1,15}(\.(\d){2})?" />
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="Currency">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="USD" />
          <xs:enumeration value="EUR" />
          <xs:enumeration value="GBP" />
          <xs:enumeration value="AUS" />
          <xs:enumeration value="YEN" />
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

XML Schema

XMLBean Interface

```
public interface MoneyType {
  public void setCurrency(
    MoneyType.Currency.Enum enum);
  public void setValue(BigDecimal num);
}
```

JSP with NetUI

```
<netui:label value="${pageFlow.price.value}"/>
<netui:label value="${pageFlow.price.currency}"/>
```

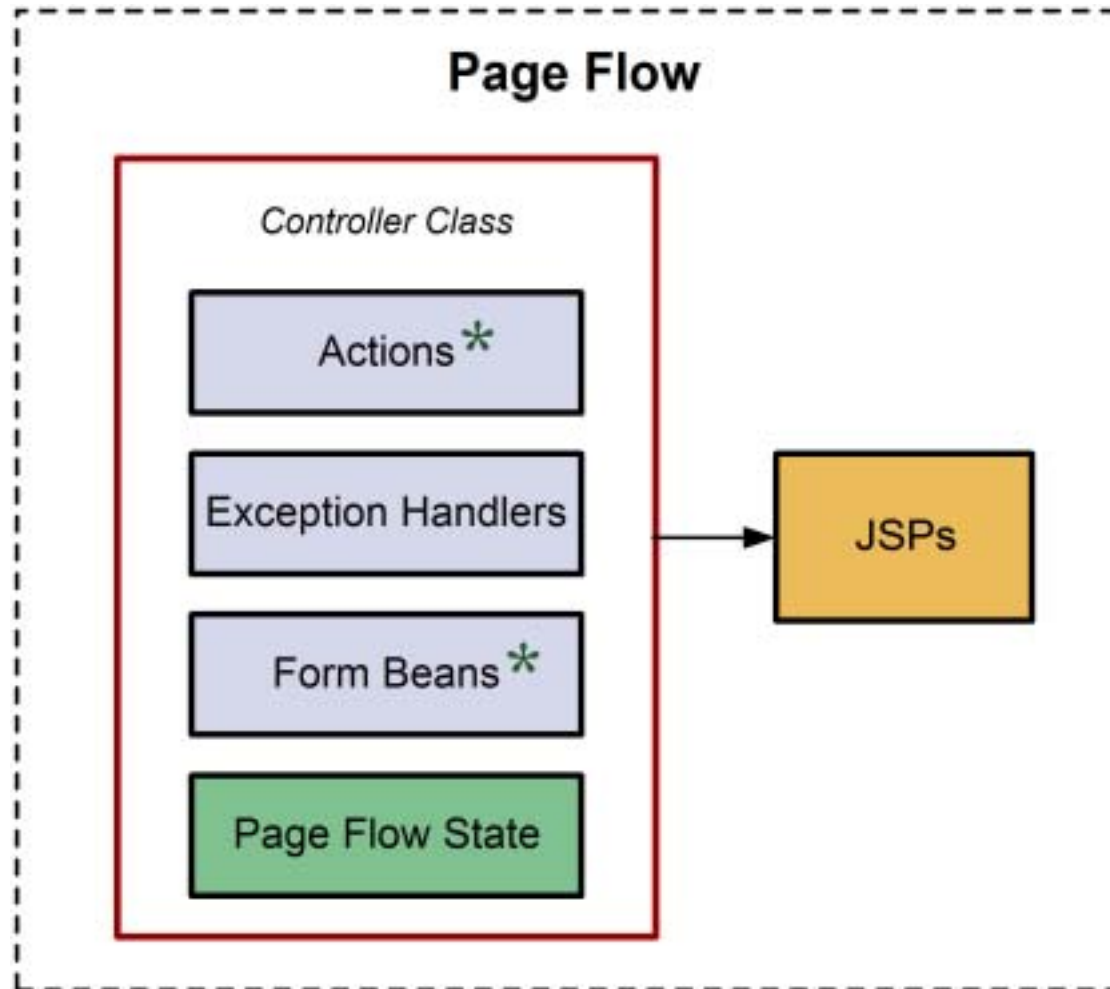


# Beehive Page Flows

- **Web app development hard and time-consuming**
  - Many MVC frameworks (Struts, Spring MVC, WebWork, ..) and UI presentation technologies (JSP, JSF, Velocity, ..)
  - Mostly non-trivial introducing steep learning curve
- **Need for more intuitive 'easy-to-use' programming model**
  - Get productive fast, even with little experience
  - Toolable model, allowing for visual development experience
- **Leverage powerful feature sets of proven technologies**
  - Struts
  - JSP



# Beehive Page Flows



\* validation rules

# Beehive Page Flow Controller

- **Built on Struts**
  - Compiles into Struts 1.2 module
  - Can easily coexist / integrate with native Struts modules
- **Simple single-file authoring based on declarative annotations**
  - Everything defined in one place: actions, forward rules, form beans, validations, exception handling, flow state, etc
  - No separate configuration files
- **Modular**
  - Decompose web app into multiple self-contained, reusable flows





# Beehive Page Flow Controller

- **Stateful**

- Conversational state kept in controller member variables
- Thread-safe
- Automatic state management
  - Instances stored in user session
  - Hitting a page or action = “entering” the page flow; create instance
  - Navigating to another page flow = “leaving” the current one; cleanup instance
- Page Flow acts as data binding scope
  - request <= pageFlow <= session

- **Nestable**

- Temporarily move to other flow, to return later
- Save current flow and its state for later (stack)
- Similar to a method invocation: call and return

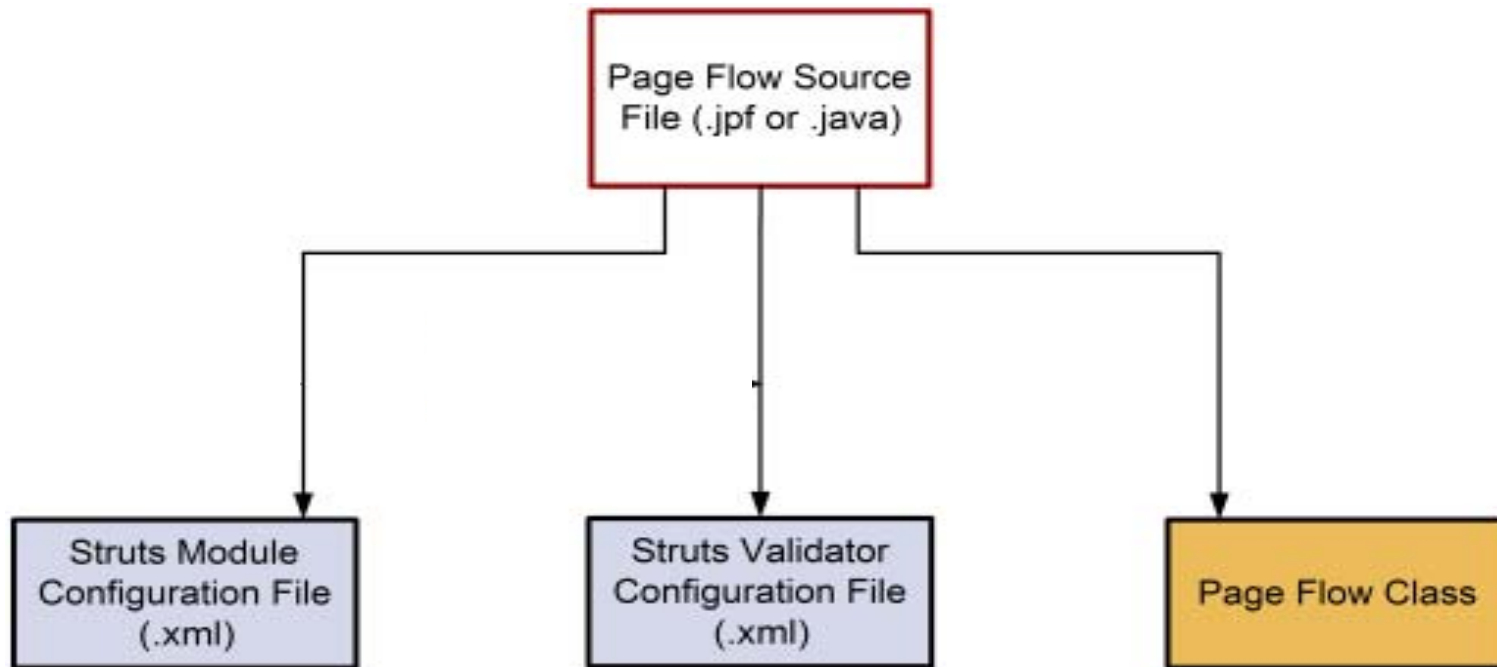
# Beehive Page Flow Controller

- **Shared Flow**
  - Fallback location for shared / unhandled actions
    - E.g. login, home, search
  - Contains state shared among flows
- **Form Beans**
  - Any JavaBean can be used as a form bean
  - Typically (but not necessarily) implemented as inner class
- **Declarative validation through annotations**
  - Controller class
  - Action method
  - Form Bean getter method



# Beehive Page Flow Controller Build

Based on Sun's Annotation Processing Tool - APT



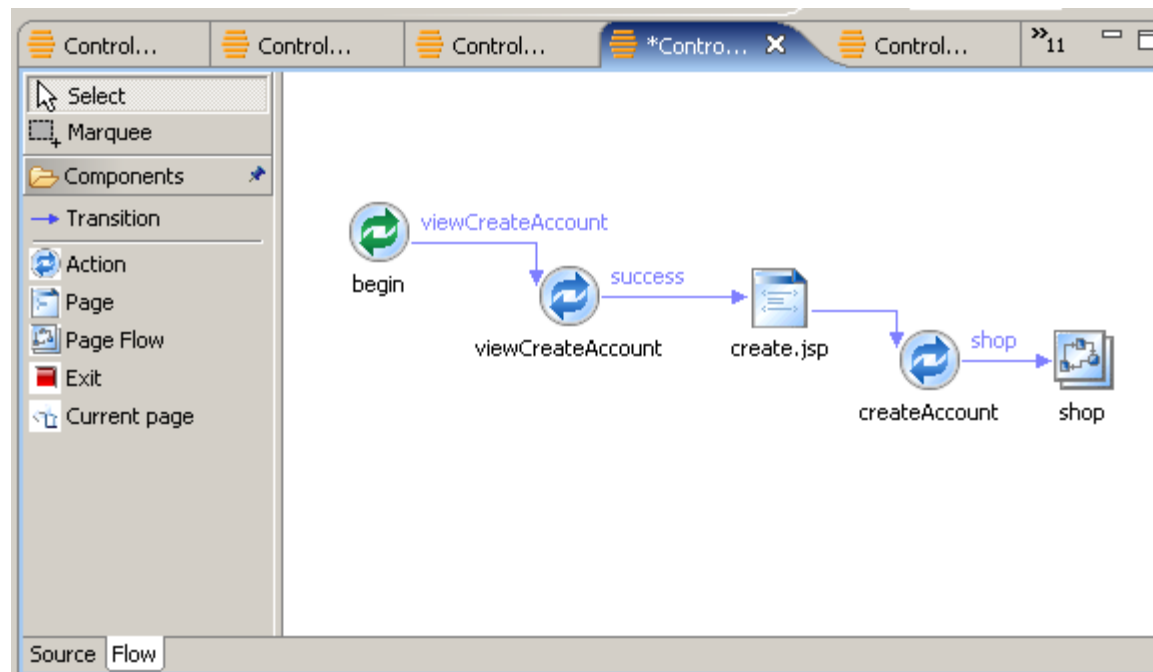
# Beehive Page Flow Source View

```
@Jpf.Controller
public class AccountPageFlow extends PageFlowController {
    @Control
    private AccountWebService accountWS;

    @Jpf.Action(forwards=
        {@Jpf.Forward(name="success", path="index.jsp")}
    )
    public Forward viewAccounts() {
        Account[] accounts = accountWS.getAccounts();
        Forward f = new Forward("success");
        f.addActionOutput("accounts", accounts);
        return f;
    }
}
```



# Beehive Page Flow Design View



- Pollinate Graphical Page Flow Editor in Eclipse

# Beehive Page Flow JSP

- **Leveraging JSP 2.0**

- Supporting EL / .tag files / JSTL / SimpleTags

- **NetUI tag libraries**

- `<netui:xxx>`: basic HTML rendering

```
<netui:span value="${actionForm.cart.subTotal}">
```

- `<netui-data:xxx>`: complex data set rendering

- Support for rich tree and rich data grid rendering

```
<netui-data:declarePageInput name="accounts" type="org.openUri.Account[]"/>
```

```
<netui-data:repeater dataSource="pageInput.accounts">
```

```
  <netui:span value="${container.item.name}">
```

```
  <netui:span value="${container.item.zipCode}">
```

```
</netui-data:repeater>
```

- `<netui-template:xxx>`: layout templating

```
<netui-template:template templatePage="/site/template.jsp">
```

```
  <netui-template:section name="leftnav"> .. </netui-template:section>
```

```
</netui-template:template>
```



# Beehive Page Flow JSP

- **Data Binding**

- EL support
- Additional scopes:
  - actionForm: current form bean
  - pageFlow: current page flow controller
  - pageInput: attribute map attached to current Forward (addOutput)
  - bundle: declared bundle property set (`<netui-data:declareBundle name="myBundle"/>`)
  - container: complex data set within `<netui-data>` repeater or grid
- Action output / page input
  - Ensure that JSPs get the data they need
- Any JavaBean can be used as a form bean



# Beehive Page Flow JSF

- **Page flows also support Java Server Faces view components**

- JSF page = JSP + backing bean
- Integrate JSF tags in JSP; raise actions from view tag or backing bean (can attach form beans)

```
<f:view>
```

```
  <h:form id="go2form">
```

```
    <h:inputText id="foo" value="#{backing.foo}"/>
```

```
    <h:commandLink id="go2button" action="go2" value="go to page3" />
```

```
    <h:commandLink id="go3button" action="#{backing.commandHandler3}" value="go to page3 (pass a form)" />
```

```
  </h:form>
```

```
</f:view>
```

```
@Jpf.FacesBacking public class page1 extends FacesBackingBean {...
```

```
  @Jpf.CommandHandler(raiseActions = {@Jpf.RaiseAction( action="go3", outputFormBean="bar" ) })
```

```
  public String commandHandler3() {
```

```
    bar.setBar( _foo );
```

```
    return "go3"; }
```

- JSF support enabled by plugging page flow application factory in WEB-INF/faces-config.xml

```
...<application-factory> org.apache.beehive.netui.pageflow.faces.PageFlowApplicationFactory </application-factory>...
```



# Beehive Controls

- **Diversity of resource access mechanisms and APIs**
- **Enterprise access has too many!**
  - JDBC, EJB, JMS, JCA, JAX-RPC, ...
- **Developer learning curve**
- **Resource-specific custom facades / tooling**

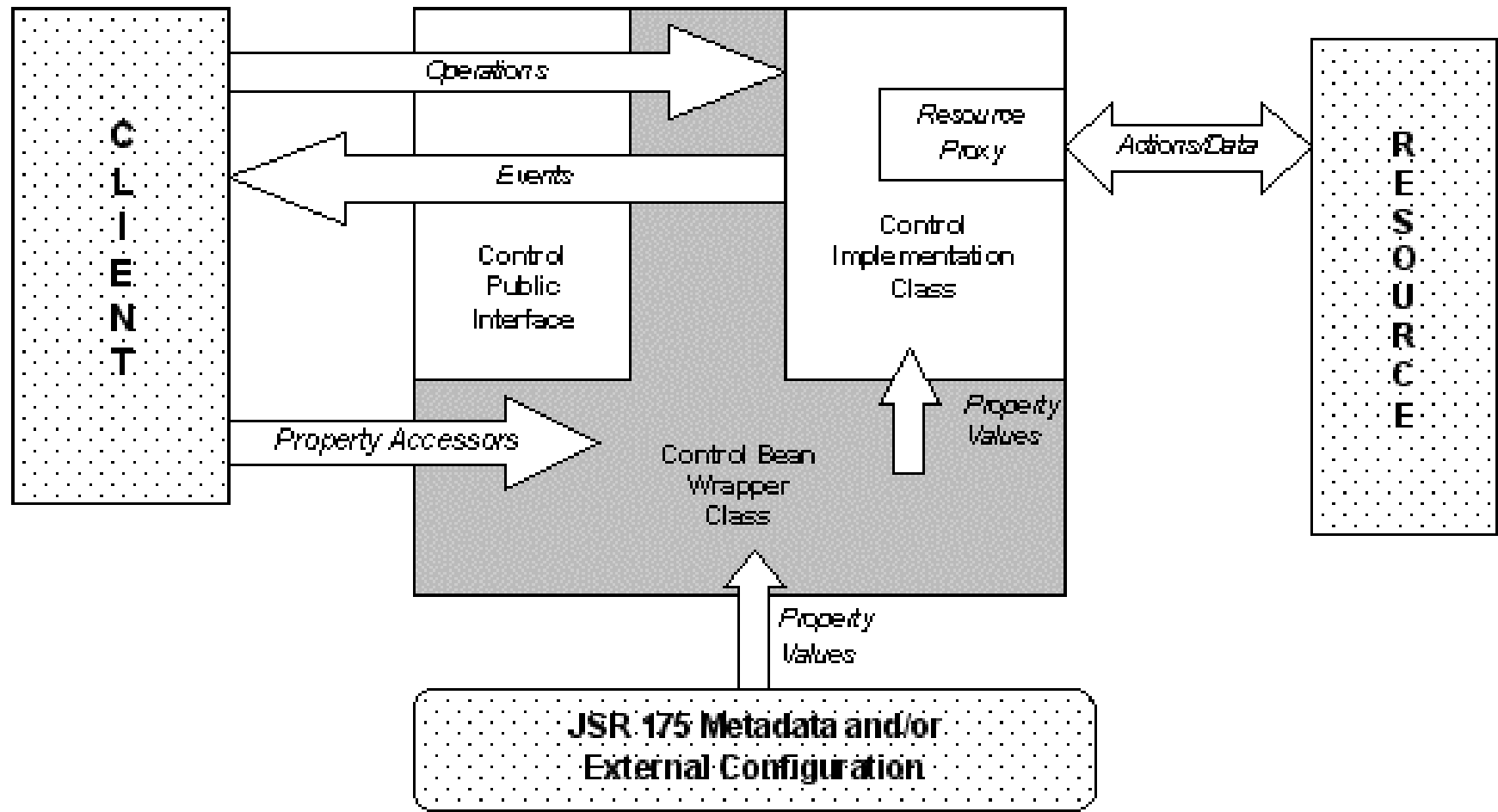


# Beehive Controls

- **Unified simple client programming model for resource access**
- **Façade**
  - All resources appear as JavaBeans
- **Unified**
  - One client model (JavaBeans)
  - One configuration model (Properties)
  - One tooling model (Introspection)
- **Simple**
  - Declarative programming using annotations
  - Extensibility model



# Beehive Controls Architecture



# Beehive Controls Sample

## Interface:

```
@ControlInterface public interface JmsMessageControl {  
    public void sendTextMessage(String text);  
    @EventSet public interface Callback {  
        public void onMessage(Message m);  
    }  
}
```

## Implementation:

```
@ControlImplementation public class JMSMessageControlImpl implements  
    JMSMessageControl {  
    @Client Callback client;  
    public void sendTextMessage(String text) { ...  
        client.onMessage(m);  
    }  
}
```

## Client:

```
@Control public JmsMessageControlBean myJmsControl;  
@EventHandler ( field="myJmsControl", evenSet= JmsMessageControl.Callback.class,  
    eventName="onMessage")  
public void myJmsBeanMessageEventHandler(Message m) { ...  
}
```



# Beehive Controls Property Configuration

- **Control interfaces declare supported properties using annotations**

```
public enum DestinationType { QUEUE, TOPIC }  
  
@PropertySet(prefix="Destination")  
  
@Target({FIELD, TYPE})  
  
@Retention(RetentionPolicy.RUNTIME)  
  
public @interface Destination {  
    public DestinationType type() default QUEUE;  
    public String name();  
}
```

- **Clients can access property values using:**

- JavaBean property accessors

```
myJMSControl = (JMSMessageControlBean) java.beans.Beans.instantiate(cl, "org.openuri.JMSMessageControlBean");  
myJMSControl.setDestinationName("myJMSQueue");
```

- External XML configuration file

```
...<destination><name>myJMSQueue</name></destination>...
```

- Annotated control instances

```
@Control @Destination(name="myJMSQueue") myJMSControl;
```



# Beehive Controls-Spring Integration

- **Beehive ControlFactory SPI allows pluggable Control instantiation and Configuration**
- **SpringControlFactory implementation based on Spring BeanFactory provides for Spring integration**
- **Sample Spring applicationContext.xml**

```
<beans>
  <bean name="Kate" parent="MinorBean">
    <constructor-arg index="1"><value>Kate</value></constructor-arg>
    <property name="firstName"><value>Kate</value></property>
    <property name="lastName"><value>Marvin</value></property>
  </bean>
  <bean id="MinorBean" class="org.apache.beehive.samples.spring.control.PersonBean" abstract="true" singleton="false" >
    <constructor-arg index="0"><null/></constructor-arg>
    <constructor-arg index="1"><null/></constructor-arg>
    <constructor-arg index="2"><null/></constructor-arg>
    <property name="controlImplementation">
      <value>org.apache.beehive.samples.spring.control.MinorImpl</value>
    </property>
  </bean>
</beans>
```



# Beehive Controls Extensibility

- **Extend new Controls (resource usage views) from Extensible Controls by extending Control Interface with annotated operations**
- **Extensible Control has common generic implementation; customized / constrained behavior conform annotated properties**
- **Sample: Extensible DatabaseControl executing SQL Statements**

```
@ControlExtension
```

```
@DatabaseControl(jndiDataSourceName="derbyDB")
```

```
public interface CustomerDB extends DatabaseControl {
```

```
    @sql (statement="INSERT INTO CUSTOMERDB (ID, NAME) VALUES ({id}, {name})")
```

```
    int newCustomer(int id, String name) throws SQLException;
```

```
    @sql (statement="SELECT * FROM CUSTOMERDB WHERE ID = {id}")
```

```
    Customer findCustomer(int id);
```

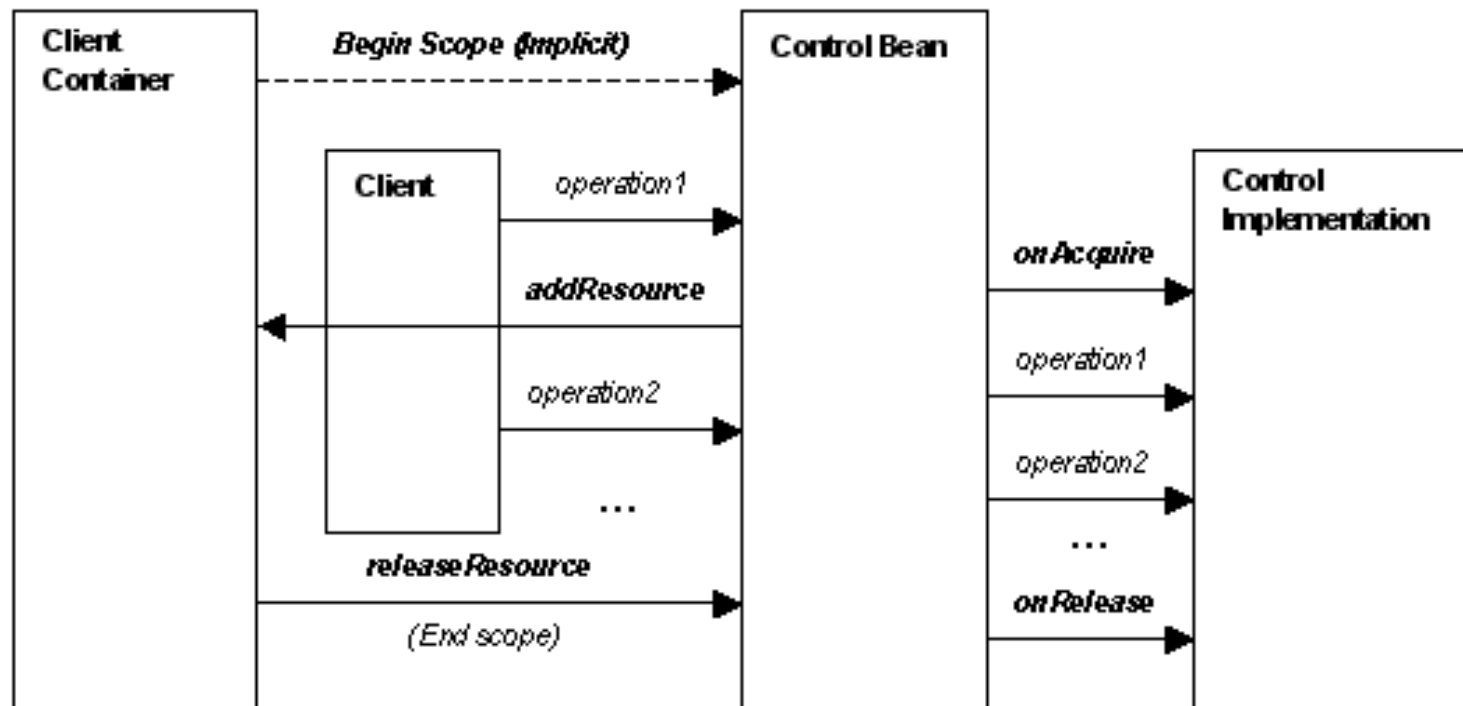
```
}
```

- **Highly toolable using wizards and property editors**
- **Extensible controls available from ControlHaus**
  - JMS, JDBC, Web Service, EJB



# Beehive Controls Resource Management

- Acquire/release resources (connections, sessions, EJB stubs, WS client proxies)
- Transparent to Control Client
- Implemented by ControlBean according to resource scoping offered by Client Container (Servlet, EJB, Application)



# Beehive Controls Packaging

- Leveraging JavaBeans packaging in simple JAR
- Allows tools to discover Controls by introspection and build palettes of available controls





# Beehive Web Services

- **Simplify WS development with metadata annotations**
- **Write service implementation bean**
- **Expose it as a WS using annotations; generate remaining artifacts**
- **No expertise on WS deployment descriptors and APIs required**
- **Implementing JSR 181 - WS metadata annotations**
- **Leveraging Axis JWS container (can run on various)**
  
- **Can be container for Beehive Controls**
- **Can be accessed via Beehive extensible Web Service Control**



# Beehive Web Service Sample

- Service.jws

```
@WebService(targetNamespace="http://beehive.apache.org/EmployeeDB")
public class Service {
    @Control public EmployeeDBControl employeeDB;
    @WebMethod public void insertEmployee(WSEmployee e) throws RemoteException, SQLException {
        employeeDB.insertEmployee(e);
    }
    @WebMethod public WSEmployee selectEmployee(
        @WebParam (name="employeeNumber") int p_id
    ) throws RemoteException, SQLException {
        return employeeDB.selectEmployee(p_id);
    }
}
```



# Beehive Web Service annotations

- Control WSDL mapping

`@WebService(name | serviceName | operationName | targetNamespace | action)`

`@WebMethod(operationName | action)`

`@WebParam(name | targetNamespace | mode | header)`

`@WebResult(name | targetNamespace)`

- Control Bindings

`@SoapBinding(type | use)`

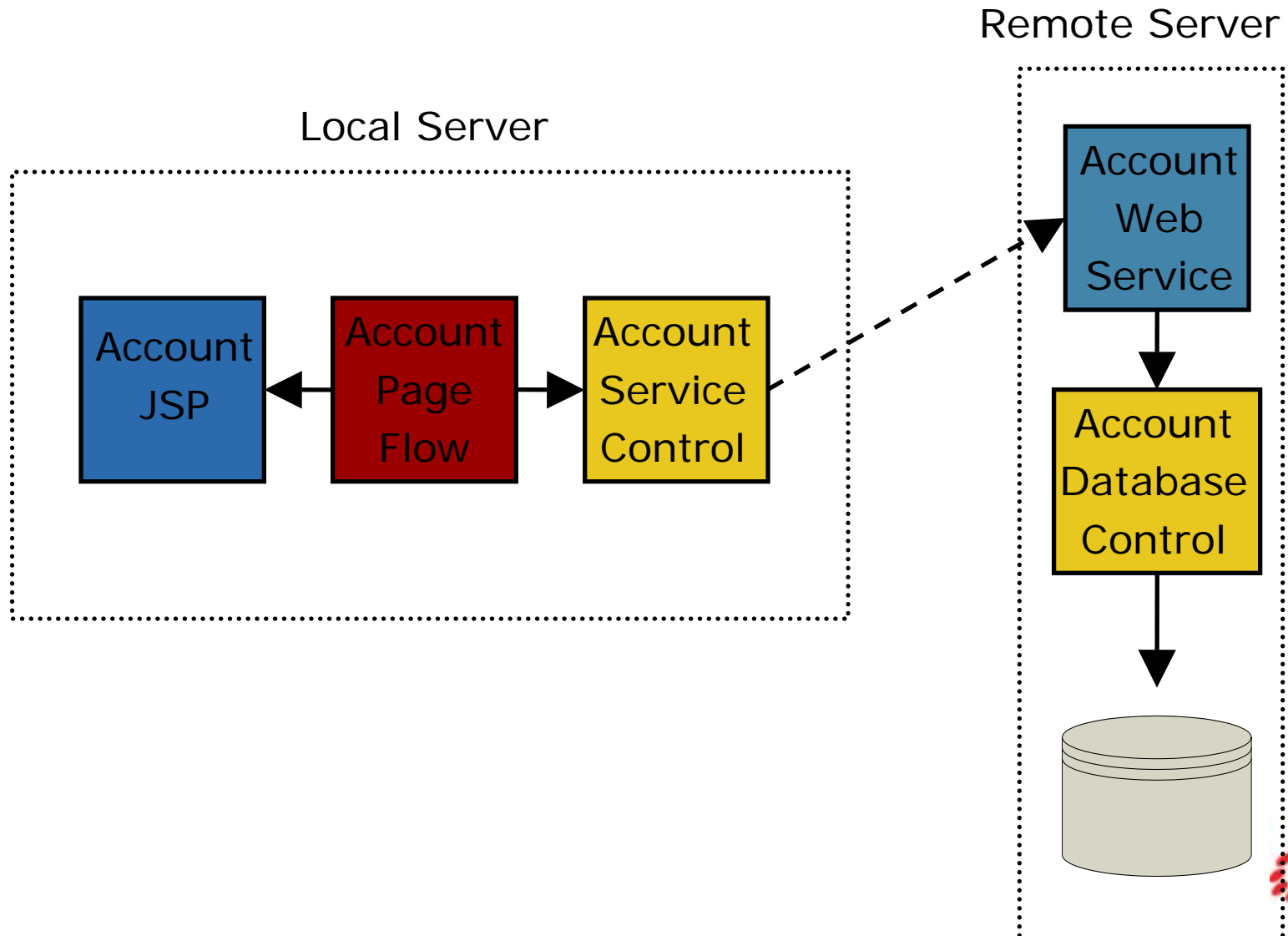
- Control Message Handlers

`@HandlerChain`

`@SOAPMessageHandlers`



# All Together Now



# Summary

- **BEA DoubleDown on OSS: Eclipse + Beehive**
- **WebLogic Workshop IDE is rebuilt on Eclipse and WTP**
- **Apache Beehive framework at foundation of WebLogic Platform**
  - Ease-of-use J2EE / SOA programming model
  - Power to the annotated POJO:
    - page flows: easy web app development
    - controls: easy resource access
    - annotated web services: easy service authoring
  - Open and fully portable, no vendor lock-in
- **Go and take it for a spin!**





# Check it out!

## Apache Beehive Project

- <http://incubator.apache.org/beehive>

## Apache XMLBeans Project

- <http://xmlbeans.apache.org>

## Eclipse Web Tools Platform Project

- <http://www.eclipse.org/webtools>

## Eclipse Pollinate Project

- <http://www.eclipse.org/pollinate>

## Eclipse AspectJ Project

- <http://www.eclipse.org/aspectj>

## CodeHaus OSS Community

- <http://www.codehaus.org>

## ControlHaus OSS Community

- <http://www.codehaus.org>

