Pollinate and Beehive

Project Update

Tuesday, March 1, 2005
Topics

- What is Pollinate?
- What is Beehive?
- NetUI PageFlows Demo
- Controls
- WebService
- History
- Challenges
- Milestones
- Future
What is Pollinate?

- an Eclipse technology project
- built on top of Eclipse and WebTools
- an IDE for building web applications based upon Apache Beehive

Focus:
“All things related to Apache Beehive”

Motto:
“Reuse what we can, build what we must”
What does Pollinate do?

Pollinate based tools generate Beehive based web applications
What is Apache Beehive?

**Goals:**

http://incubator.apache.org/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.”

**History:**

- Originally evolved as part of BEA WebLogic Workshop
- BEA contributed Beehive to the Apache in May 2004
- BEA continues to support ongoing Beehive development
What is Apache Beehive?

Apache Beehive project has three main parts:

- **NetUI PageFlows** – “A web application framework built on top of Struts allowing easier tooling as well as automatic updating of Struts configuration files with the use of metadata.”

- **Controls** – “Lightweight component framework that helps programmers build components that incorporate metadata into their programming model.”

- **Web Services** – “An Implementation of JSR-181, an annotation driven programming model for web services.”
NetUI PageFlows

NetUI PageFlows = Struts plus:
- Automatic state management
- Integration with Controls
- Java Server Faces
- XML Beans
@Jpf.Action(
    forwards = {
        @Jpf.Forward(
            name = "demoResults",
            path = "DemoResults.jsp"),
        @Jpf.Forward(
            name="success",
            path="newPage.jsp")
    }
)

public Forward DemoSubmit( DemoForm demoForm )
{
    _name = demoForm.getName();
    if (demoForm.getName().equals("foo")
        return new Forward( "success" );
    
    _results = none;
    if (demoForm.getName().equals("animals")
        _results = animals;
    else if (demoForm.getName().equals("frumpels")
        _results = frumpels;
    
    return new Forward( "demoResults" );
}
Controls

- Lightweight framework
- Wrap services, databases, POJOs (Plain Old Java Objects), etc
- Automatically manages property state and resource allocation/release

- Pollinate compiles Controls, but no UI yet
- Simple Pollinate Controls demo that counts entries
Using Controls

```java
public class Controller extends PageFlowController {
    protected SequenceControl _sequenceControl;
    // ... <snip> ...
    @Opf.Action(
        @Opf.Forward(
            name = "demoResults",
            path = "DemoResults.jsp"
        )
    )
    public Forward DemoSubmit( DemoForm demoForm )
    {
        _name = demoForm.getName();
        _results = none;
        if (demoForm.getName().equals("animals"))
            _results = animals;
        else if (demoForm.getName().equals("frumpels"))
            _results = frumpels;
        else {
            try {
                int id = _sequenceControl.getNextSequenceId(_name);
                Collection<String> strings = new ArrayList<String>();
                strings.add("Control value: " + id);
                strings.addAll(Arrays.asList(controlMessage));
                _results = new String[0].toArray(new String[strings.size()]);
            } catch (InvalidIdentifierException ex) {
                // Nothing
            }
        }
```
Web Services

“an implementation of the JSR-181 specification and is a key piece of the Beehive framework. JSR 181 uses JSR-175 metadata annotations in Java methods and classes to easily build Web services”

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

In a nutshell

JWS Annotations ⇔ WSDL
(Java Web Services) (Web Service Definition Language)

- Annotations in Java files define which services (classes and methods) can be accessed remotely

- WSDL files encode the same information in an XML format
Web Services

WSDL

```xml
<portType name="PetstoreInventoryManager">
  <operation name="listProducts">
    <input message="categoryId"/>
    <output message="ProductList"/>
  </operation>
</portType>

<message name="ProductList">
  ...
</message>

<message name="categoryId">
  ...
</message>
```

JWS

```java
@WebService(name = "PetstoreInventoryManager",
            serviceName = "PetstoreInventoryManagementService",
            targetNamespace = "http://beehive.apache.org/petstore")
//@SOAPBinding(style = SOAPBinding.Style.RFC, use = SOAPBinding.Use.ENCODED)
public class PetstoreInventoryManager implements ServiceLifeCycle {  

  @WebMethod(action = "listProducts")
  @WebResult(name = "ProductList")
  public org.apache.beehive.samples.petstore.model.ws.Product[] listProducts
  {
    @WebParam(name = "categoryId") String categoryId;
    
    CatalogControlBean ccb = getCatalogBean();
    Product[] prods = ccb.getProductListByCategory(categoryId);
    org.apache.beehive.samples.petstore.model.ws.Product[] cutProds =
    new org.apache.beehive.samples.petstore.model.ws.Product[prods.length];
```
Pollinate History

Jun ‘04 – research Beehive and make high level proposal to Eclipse
  * layout plan for Milestone 1
  * start development

Aug ‘04 – Pollinate becomes a formal Technology project
  * investigate WebTools… not stable yet
  * develop our own simple server tools

Sep ‘04 – deliver Milestone 1
  * move web & CVS infrastructure to Eclipse.org
  * Eclipse annotation support still in early phases

Nov ‘04 – deliver Milestone 2 corresponding with ApacheCon
  * Eclipse annotation support stable

Dec ‘04 – switch server tooling framework
  * WebTools stabilizes
  * discard our simple server tooling in favor of WebTools

soon – deliver Milestone 3
Pollinate Challenges

**Beehive**
- work in progress
- alpha V1 release currently available

**Eclipse annotation parsing**
- early support in Sept ’04
- not completed until Nov ’04

**WebTools Project**
- available but not stable when we started
- stable release became available in Dec ’04

**APT - Annotation Processing Tool**
- nothing in Eclipse that provides similar functionality
- Pollinate depends on Sun’s APT in “tools.jar”
- Hopeful for future Eclipse enhancements to provide needed functionality

**Support for non “.java” file extensions**
- JDT compiler and editors hard coded for “.java”
- current workaround copies non “.java” files to “.java” before compilation
Pollinate Milestones

✅ **Milestone 1** – September ‘04
  - Compile page flow with minimal UI
  - Use JDT compiler, wrapper Beehive APT with new Eclipse builder
  - Simple Pollinate project creation wizard

✅ **Milestone 2** – November ‘04
  - Basic PageFlow editor
  - Simple web server tooling for local testing
  - Compile Controls, but no UI yet
  - Enhanced project creation wizard with page flow and controls examples

✅ **Milestone 3** – soon
  - WebTools stable… move Pollinate on top of WebTools
  - Enhanced PageFlow editor

🔍 **Milestone 4** – planned for late April or early May ‘05
  - Controls UI
  - WebServices compilation and UI
  - Investigate emerging Eclipse APIs to remove dependence on Sun’s APT in “tools.jar”
  - Package Pollinate as a full IDE / RCP
  - Deeper integration into WebTools
Pollinate Future

- Discover and manage Controls
- Drag and drop Controls in PageFlow Editor
- Present Control properties in a view
- Wizard for creating a new Controls project

- Compile JSR-181 WebServices
- Generate WSDL from JSR-181 defined WebService
- Drag and drop WebServices in PageFlow Editor
- Present WebService properties in a view
- Wizard for creating a new WebService project

- Remove dependency on Sun’s APT in “tools.jar”
- Package Pollinate as an RCP IDE
- Deeper integration into WebTools
Pollinate Questions?

For more information, see
- http://www.eclipse.org/pollinate
- http://incubator.apache.org/beehive

Questions?