From Apache Beehive to Oracle's Application Development Framework (Oracle ADF)

An Oracle White Paper
August 2008

ORACLE

NOTE:

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

From Apache Beehive to Oracle's Application Development Framework (Oracle ADF)

Note: 2
Executive Overview 4
Beehive – a brief history 4
The ever changing world of The Java Enterprise 5
Is Vanilla Java EE 5.0 good enough to replace Beehive? 5
Oracle ADF Overview 6
  Beehive Controls = ADF Model Layer 6
  NetUI = ADF Controller 7
  NetUI JSP Tags = ADF Faces Components 7
ADF Additional Functionality 9
  Drag and Drop Data Binding 9
  Built in Ajax support 10
  More Data Sources 10
  4GL/SQL Oriented Persistence Layer 10
  Security 10
Oracle ADF a Safe Choice 11
  Support 11
  Training 11
  Source availability 11
  Community 11
  Vendor Commitment 11
Conclusion 12

From Apache Beehive to Oracle's Application Development Framework (Oracle ADF)

EXECUTIVE OVERVIEW

The BEA sponsored Apache Beehive framework has been around for a while but in recent years hasn't progressed much. In this paper we'll explain why Oracle's Application Development Framework is a natural step forward for organizations which are currently using Beehive and are looking for a new framework to update their solution. We'll show how Oracle ADF offers solutions to the same problems that the Beehive framework aimed to address, and highlight some of the areas where ADF excels beyond the functionality offered by Apache Beehive.

BEEHIVE – A BRIEF HISTORY

Back in 2003 as part of the Workshop for WebLogic 8.1 release BEA introduced a new Apache Struts based framework aimed at simplifying J2EE application development. Hoping to get broader adoption of this framework BEA donated the framework to the Apache organization in 2004 as the Beehive project.

The main goal behind Beehive was to try and simplify the development of Java EE based applications. The Beehive framework adopted the Model-View-Controller (MVC) design pattern and focused on three basic areas:

- Business services – exposed as Controls in Beehive
- Controller – NetUI classes that extend the Struts framework
- User interface – NetUI JSP tags

The first version of Beehive was released in 2005 with a couple of updates done in 2006. But the framework hasn't developed much further since then.

THE EVER CHANGING WORLD OF THE JAVA ENTERPRISE

Over the past couple of years the Java EE space has changed as well. Sun had realized that Java EE application development was overly complex and work started on several projects which finally surfaced in Java EE 5.0 and would drastically change the development experience of enterprise Java applications.

Oracle took a key part in shaping Java EE 5.0 playing a leading role in the EJB 3.0 and JPA JSRs on the business services side, and the JSF expert group on the user interface sides.

EJB 3.0 greatly simplifies the creation of business services by removing a lot of the mundane configuration and coding tasks involved in the definition of enterprise JavaBean components.

Another result of the EJB 3.0 effort was the creation of a standard API for persisting Java objects known as JPA (Java Persistence API). Borrowing ideas from existing Object Relational framework such as Oracle TopLink, JPA offers a simpler standardized way to map objects from the Java world to data in relational databases such as Oracle.

The world of Web user interface development didn't sit still either. Noticing the fragmentation of the user interfaces market (with over 30 Java based frameworks competing to simplify Web UI Development), Sun began to work on a framework that incorporated ideas from the leading WebUI frameworks including Struts and Oracle's UIX framework. The result was JavaService Faces (JSF) introduced in 2004 and further enhanced with version 1.2 two years later.

With a support from all the key players in the Java market JSF has been gaining increased popularity and has a thriving market of development tools and ready-to-use components.

IS VANILLA JAVA EE 5.0 GOOD ENOUGH TO REPLACE BEEHIVE?

Java EE 5.0 eliminated a lot of the mundane coding needed to access enterprise resources such as EJBs. It also introduced a standard controller layer for Web applications (JSF) that takes care of state management. But Java EE 5.0 still misses some of the additional functionality offered by Beehive, functionality that impacts both reusability and productivity when building applications.

Java EE 5.0 doesn't offer the abstraction provided by the control concepts in Beehive. Developers still need to interact with POJO, EJB or Web Services in different ways. This is especially problematic for UI developers who want to bind

their interfaces to various back-end components and need to learn various protocols to achieve this.

In addition the JSF controller is missing some of the capabilities offered by the Beehive controller layer such as reusable flows, and better stateful page flow support.

One other key aspect that is missing from the Java EE 5.0 development solution is a development approach that would be more appealing to enterprise developers coming from a background of 4GL RAD client/server tools such as Visual Basic and PowerBuilder. This involves having a development tool that will switch the experience of working with a framework from that of a code editor to a more visual and declarative approach employing visual editors, property inspectors and dialogs simplifing the overall experience, reducing coding errors and increaing productivity.

The result is that developers still need a framework on top of Java EE 5.0 to maximize their development potential. So what's the solution?

The solution is the Oracle Application Development Framework (Oracle ADF) – This is a complete meta-framework that builds on top of Java EE 5.0 and adds the missing parts.

ORACLE ADF OVERVIEW

Starting in 1999, Oracle has provided the Oracle Application Development Framework to simplify the task of Java EE development for our customers. Interestingly enough there are quite a lot of similarities between Beehive and ADF both in their goals and the approaches taken for the solution.

The basic concepts offered by both frameworks are those of abstraction of business service implementation, controller layer, UI components and declarative data binding. Let's look at some of the key components of Oracle ADF and how they map to their Beehive equivalents.

Beehive Controls = ADF Model Layer

Similar to the Beehive controls concept, Oracle ADF provides an abstraction layer on top of business services known as the ADF Model layer – implemented through data controls. These are the components containing your business logic and the connection to your data. You can expose Java classes, Web Service, EJB, Java Content Repositories, files and other sources of data and operations as a data control. The data control layer describes the attributes and operations offered by the business service using meta-data. Developers can then use declarative features to add validation, and even UI properties to business entity attributes.

Once exposed as a data control the components, methods, and attributes are exposed to the UI developer in a standard way abstracting the underlying technology from view.

Oracle ADF defines a data control by creating an XML file that describes the business object in terms of its properties and methods. This use of a meta-data layer maintains a clean separation between the service and its consumption. Oracle JDeveloper manages the creation and maintenance of the meta-data, saving the user from the need to code it by hand. Further editing of the files can be done through dialog and property inspectors.

NetUI = ADF Controller

Unlike Beehive ADF offers the choice of multiple controllers rather than being restricted to a Struts based implementation. This includes the controller provided by JavaServer Faces..

Oracle was one of the first companies frameworks to embrace the JSF standard and incorporate it into the ADF framework as both the controller and view layer of choice since early 2006.

While JSF is a step in the right direction extending the functionality of both the controller layer and the UI layer – it still misses some of the functionality offered by the Beehive UINet controller – specifically in the areas of reusable flows and advance state management.

Oracle has addressed exactly these issues in Oracle ADF 11g by extending the JSF controller with additional capabilities and creating the ADF Controller layer with its reusable Task Flow concept. Going beyond the reusability offered for flows usage inside other flows – ADF controller also offers the possibility of reusing complete flows inside JSF pages. Task Flows can also manage their own memory and transaction scope in a declarative way.

NetUI JSP Tags = ADF Faces Components

For the user interface design Oracle ADF can use the JSF standard. A key concept promoted by JSF is the creation of user interfaces using components that construct the pages. Not satisfied with just the built in components offered by the JSF reference implementation, Oracle has created a set of JSF components that offer a much richer user experience with richer functionality. These ADF Faces components, which were introduced in early 2006, offer built in functionality that includes partial page rendering, dialog, framework, menu framework, skinning , accessibility, and internalization support and more. In 2006 Oracle contributed the set of ADF Faces components to the Apache organization and they have been

further developed by Oracle developers under the Apache MyFaces Trinidad project. In Oracle ADF 11g Oracle introduces the ADF Faces Rich Client components that extend the ADF Faces set of components to over a 130 components all with built in Ajax functionality and additional framework capabilities that include support for drag and drop, smart data streaming, push to the client technology, and more. In addition ADF includes a set of data visualization JSF components that can render over 50 types of graphs, maps, gauges, hierarchy browsers, and gantt charts. These JSF components offer even richer data display with interactivity and animation capabilities.

Here is a table that lists the various NetUI capabilities and how they map to ADF capabilities and components.

Beehive Functionality	Description	Oracle ADF
Stateful Page Flows	Stores the flow-related state	ADF Controller
Modular Page Flows	A single web application can have multiple page flows within it	ADF Controller
Inheritance and Shared Flow	Share actions, exception handlers, configuration, etc. among controller classes	Task Flow templates and ADF libraries
Nested Page Flows	An entire page flow can be inserted, or "nested", inside of another page flow	Task Flows can be embedded into other task flows and also into JSF pages as regions
Declarative Exception Handling	The desired exception handling is declared in the form of metadata annotations	Task Flows can declare their own exception handlers
Powerful JSP Tags	JSP tags to render HTML components and data components	ADF Faces Rich Client Components – over 130 JSF components with built in Ajax support
Page Templates	Enable templating for Web pages	ADF Faces Rich Client Components contain a JSF

		templating solution
JSP Data binding	Enable binding through expression language	ADF Faces Rich Client Components use expression language to bind to data

ADF ADDITIONAL FUNCTIONALITY

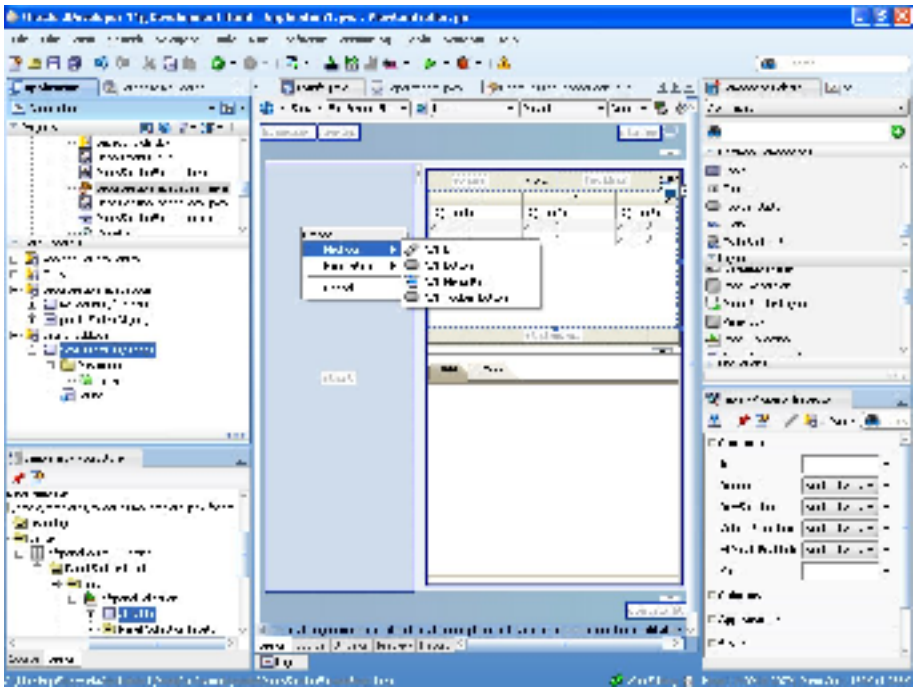
Oracle ADF doesn't simply match the functionality offered by Apache Beehive but also adds more capabilities to provide a richer development experience.

Here are some of the key additional features

Drag and Drop Data Binding

The integration of Oracle ADF with the Oracle JDeveloper IDE allows developer to drag and drop data controls into their user interface to bind the two together.

Data Controls exposed through the ADF Model layer appear in a data control palette and can be bounded to JSF and JSP pages as well as to Swing user interfaces. Developers can drag over complete collections, specific fields as well as methods, return values and parameters.



Visual JSF editing with drag and drop data binding to EJB

Built in Ajax support

The ADF Faces Rich Client Components offer built in functionality to enable RIA development. With over 150 components and with built-in Ajax functionality developers are able to create much richer and more dynamic user interfaces. The ADF Faces framework also include capabilities that enable drag and drop, pop-up, partial page rendering, menu and other functionality all in a declarative way.

In addition the rich set of data visualization components that include over 50 types of graphs, geographical maps, gauges, hierarchy browser, pivot table and Gantt charts add another layer of visualization and interactivity to enhance your JSF applications.

More Data Sources

Oracle ADF is able to expose more resources as data control. In addition to exposing Java classes, Web Services and EJBs, Oracle ADF has built in features to create data controls from CSV and XML files, BI queries from the Oracle DB, multi-dimensional databases such as Essbase, Java Content Repository, and JMX. This increases the set of sources for data and operations that the developer can work with without an increased leaning cost.

4GL/SQL Oriented Persistence Layer

ADF includes an additional sub framework called ADF Business Components (ADF BC) – this framework offers a highly declarative and proven solution for mapping between Java objects and any relational database. ADF BC follows the principles of providing a highly productive visual design time environment for defining components, validation and behavior. In addition it offers event driven approach for adding business logic to these components by offering a wide range of events that developers can hook to in order to augment or override the default behaviors offered by the framework. These include transaction processing events, database interaction events and lifecycle events. ADF BC offers a huge amount of functionality out of the box without any additional coding including query by example, list of values, application scalability and failover and more.

Security

Oracle ADF integrates a complete security solution for your Java EE application. The security framework allows you to use both authentication and authorization to limit the functionality and data different users can get from the system.

ORACLE ADF A SAFE CHOICE

Beyond the technical merits of a framework, organizations should consider a few other aspects when committing to a development framework for their future projects development.

Support

Oracle ADF is an official Oracle product and as such is serviced by the Oracle Support organization. This provides around the clock support from an established organization.

Training

Oracle University offers regular instructor lead courses on Oracle ADF and JDeveloper.

Source availability

Oracle provides the source code for the ADF framework to customers with a support license. Having the source available can help developers understand the underlying mechanisms of the framework and debug problems in their applications

Community

Oracle ADF has a very active community of developers. The OTN discussion forum for JDeveloper and ADF is averaging around 80 new threads each day. Both developers and product managers from Oracle monitor the forum and provide solutions to questions. In addition there are multiple blogs by both Oracle employees and external developers documenting techniques and tips for working with ADF. For more information visit http://oracle.com/technology/jdev

Vendor Commitment

Oracle is using the Oracle ADF framework for developing its Fusion generation of enterprise applications, other Oracle products, and various internal Oracle projects. This shows Oracle's commitment to the quality and productivity of Oracle ADF.

CONCLUSION

The goal and ideas behind the Beehive framework are good, however the framework failed to achieve wide adoption in the Java community and didn't evolve fast enough to keep with the changing world of Enterprise Java. When looking for frameworks that incorporate the ideas of Beehive combined with the latest innovation in the world of Java EE 5.0 and extensive tooling support the Oracle Application Development Framework stands out as an obvious choice for the future.

From Apache Beehive to Oracle's Application Development Framework (Oracle ADF) August 2008

Author: Shay Shmeltzer
Contributing Author: Duncan Mills

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com