An Oracle White Paper
June 2011

Oracle Application Development Framework Overview

Introduction

Java EE is a standard, robust, scalable, and secure platform that forms the basis for many of today's enterprise applications. Java EE provides a set of specifications for building multi-tier applications using the Java™ language. In the past, there was a direct correlation between the robust nature of an application to the complexity required to achieve it. However, with the advent of the Oracle ADF framework, you are able to provide the implementation of extremely rich Java EE applications, adhering to standard patterns and practices with greatly reduced effort.

Additionally, the increased need for organizations to build composite applications that utilize Service Oriented Architecture (SOA) principles has forced developers to create applications that are extremely agile. Implementing these best practices in agile applications usually involves writing a significant amount of infrastructure code, adding another obstacle for developers building their first Java EE application.

In addition to providing robust, performant, and maintainable applications; Oracle Application Development Framework also provides the best of breed infrastructure code to implement agile SOA based applications thereby removing the effort involved in an organization "rolling their own" and allowing a team to jump right in to adding value versus building an infrastructure.

Oracle ADF – Making Java EE Development Simpler

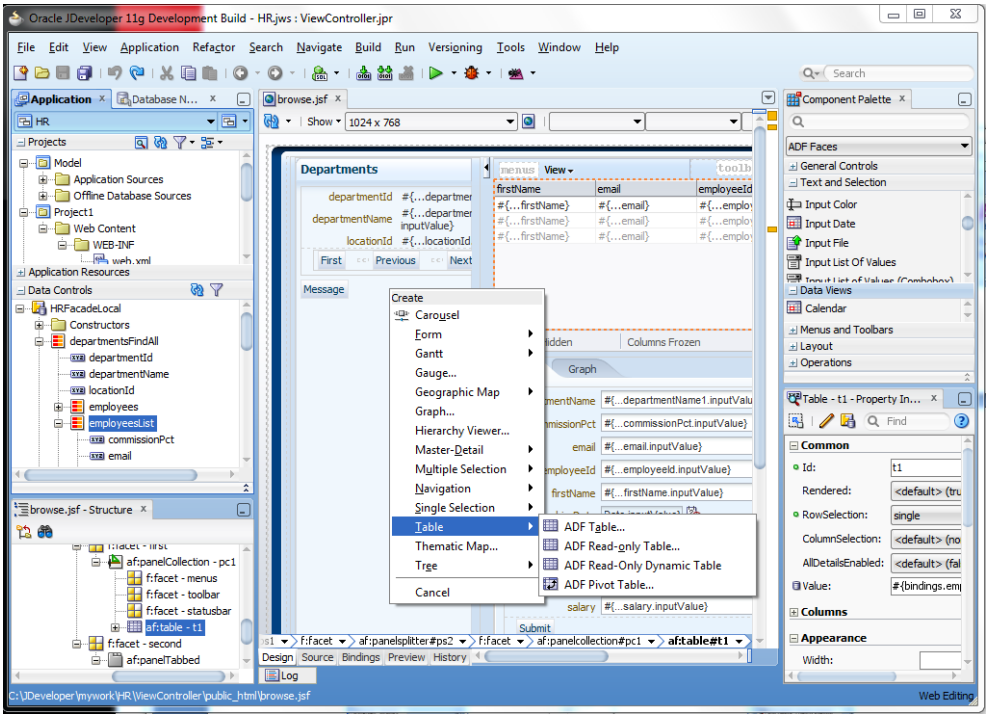
Oracle Application Development Framework (Oracle ADF) is an innovative, yet mature Java EE development framework available from Oracle and directly supported and enabled by the award winning development environment, Oracle JDeveloper 11g.

Oracle ADF simplifies Java EE development by minimizing the need to write code that implements the application's infrastructure allowing the developers to focus on the features of the actual application. Oracle ADF provides these infrastructure implementations as part of the framework. To recognize a set of runtime services is not enough, Oracle ADF is also focused on the development experience to provide a visual and declarative approach to Java EE development through the Oracle JDeveloper 11g development tool.

Oracle ADF implements the Model-View-Controller design pattern and offers an integrated solution that covers all the layers of this architecture with solution to such areas as: Object/Relational mapping, data persistence, reusable controller layer, rich Web user interface framework, data binding to UI, security and customization. Extending beyond the core Web based MVC approach, ADF also integrates with the Oracle SOA and WebCenter Portal frameworks simplifying the creation of complete composite applications.

For example, Oracle ADF makes it easy to develop agile applications that expose data as services by coupling a service interface to the built-in business services in ADF. This separation of business service implementation details is performed in Oracle ADF via metadata. Use of this metadata-driven architecture enables application developers to focus on the business logic and user experience, rather than the details of how services are accessed.

Creating the user experience is as simple as dragging-and-dropping the desired business services onto a visual page designer and indicating what type of component should represent that data. In the example illustrated, we are able to take a database table exposed as a business service, and request JDeveloper to render the data as a table simply by dragging-and-dropping the control on the page and responding to the automatic popup by indicating a table as the desired rendering component. Oracle ADF takes care of the rest.



Oracle ADF stores the implementation details of these services in metadata in the ADF Model layer. This enables developers to exchange services without modifying the user interface, making the application extremely agile. Additionally, the developer creating the user interface does not need to bother with business service access details. Instead they can focus on developing the application interface and interaction logic.

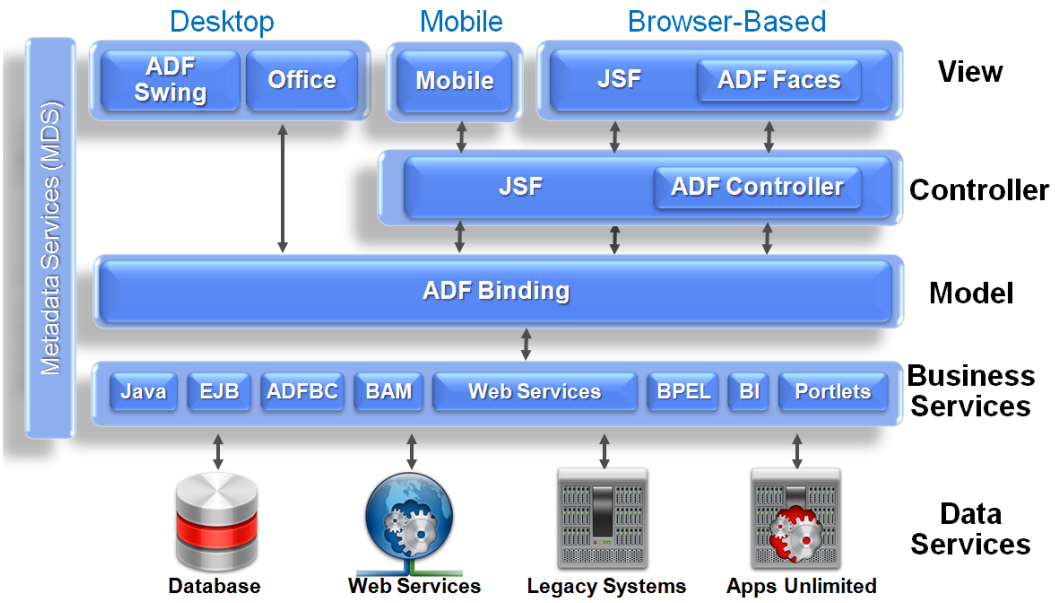
THE ORACLE ADF ARCHITECTURE

Oracle ADF is based on the Model-View-Controller (MVC) design pattern. An MVC application is separated into: 1) a model layer that handles interaction with data-sources and runs the business logic, 2) a view layer that handles the application user interface, and 3) a controller that manages the application flow and acts as the interface between the Model and the View layers.

Separating applications into these three layers simplifies maintenance and reuse of components across applications. The independence of each layer from the others results in a loosely coupled, Service Oriented Architecture (SOA).

Oracle ADF implements MVC and further separates the model layer from the business services to enable service-oriented development of applications. The Oracle ADF architecture is based on four layers:

- The Business Services layer - provides access to data from various sources and handles business logic.
- The Model layer - provides an abstraction layer on top of the Business Services layer, enabling the View and Controller layers to work with different implementations of Business Services in a consistent way.
- The Controller layer - provides a mechanism to control the flow of the Web application.
- The View layer - provides the user interface of the application.



Oracle ADF Architecture

Oracle ADF lets developers choose the technology they prefer to use when implementing each of the layers. The diagram above shows the various options available for developers when building Oracle ADF applications. The glue that integrates the various components of Java EE applications and makes development so flexible is the Oracle ADF model layer. EJB, Web Services, JavaBeans, JPA/EclipseLink/TopLink objects and many others can all be used as Business Services for the Oracle ADF Model. View layers can include Web based interfaces implemented with JSF, Desktop Swing applications and MS Office front ends, as well as interfaces for mobile devices.

The Business Services Layer

The Business Services layer manages interaction with a data persistence layer. It provides such services as data persistence, object/relational mapping, transaction management, and business logic execution.

The Business Services layer in Oracle ADF can be implemented in any of the following options: As simple Java classes, EJB, Web services, JPA objects, and Oracle ADF Business Components. In addition, data can be consumed directly from files (XML or CSV) as well as REST.

The Controller Layer

The controller layer manages the applications flow and handles user input. For example, when you click a Search button on a page, the controller determines what action to perform (do a search) and where to navigate to (the results page).

There are two controller options for web-based applications in JDeveloper: the standard JSF controller or the ADF Controller which extends the JSF controller functionality. Whichever controller you use, you will typically design your application flow by laying out pages and navigation rules on a diagram.

With the ADF controller you can break your application's flow into smaller, reusable task flows; include non-visual components such as method calls and decision points in your flow; and create "page fragment" flows that run inside a region of a single containing page. This approach encourages maximum reusability for user interface fragments and simplified integration into portals and mashup applications.

The View Layer

The View layer represents the user interface of the application.

Oracle ADF support multi-channel access to your business services allowing you to reuse your business services and access them from a Web client, a client-server swing desktop based application, Microsoft Excel spreadsheets, or a mobile devices such as a smart-phone.

For Web based interface Oracle ADF offers a rich set of over a 150 Ajax enabled JSF components that simplified the creation of dynamic and appealing user interfaces.

The Model Layer

The model layer connects the business services to the objects that use them in the other layers. Oracle ADF provides a model layer implementation that sits on top of business services, providing a single interface that can be used to access any type of business service. The model layer consists of two components, data controls and data bindings, which utilize metadata files to define the interface. Data controls abstract the business service implementation details from

clients. Data bindings expose data control methods and attributes to UI components, providing a clean separation of the view and model. Due to the metadata architecture of the model layer, developers get the same development experience when binding any type of Business Service layer implementation to the View and Controller layers.

Productivity with Choice

Developers can choose different technologies to implement each of Oracle ADF's layers, and still get the same productive development experience. For example, the same gestures and methods would be used to create an ADF Swing application based on ADF Business Components as would be used to create a ADF Faces application based on Enterprise JavaBeans. In addition to the choice of technologies implemented, developers can choose their development style (declarative, visual, or by coding), allowing developers with different skills and ways of working to cooperate on the same project.

Declarative Customization and Personalization

A unique aspect of Oracle ADF is the ability to tailor the resulting application to specific user requirements. A layer called MDS – MetaData Services – allows developers to customize XML definitions that Oracle ADF relies on and use different versions for different users.

The first level of customization is enabled for end users at runtime – users can personalize the interface of their application - for example re-order fields in a table, expand specific accordions etc – and MDS will persist those changes for them.

The second level of customization, called seeded customization, allows developers to implement modifications in business rules, page flows, page layouts and other items that ADF persist in XML format. These modifications will be applied at runtime based on the user that logs into the application. This layered approach to customization allows an organization to use one base application to serve different customer needs.

Integrated Security

Oracle ADF comes with built in security implementation across the layers. Developers can define users and roles and assign various authorizations to them. Privileges can be assigned at the business service layer, the controller layer or the UI layer. This integrated security framework across the layers eliminate the need to replicate security mechanisms at each layer.

For example, if a field in a business service is defined as non-updateable to some users, the UI layer will automatically switch the display of this field to be in read only mode in all screens without the developer needing to code this check at the UI layer.

ORACLE ADF BENEFITS

Visual and Declarative Java EE Development

A critical aspect of making a development framework useful is having a development tool that simplifies the creation of applications using this framework.

Oracle offers visual and declarative tools for each layer of Oracle ADF. These tools, which are integrated into the JDeveloper IDE, benefit Java developers even if they don't use the runtime features of Oracle ADF.

Business Services Development

Oracle JDeveloper includes a variety of ways to construct business services including: EJB/JPA, web services, simple Java objects, and ADF BC, among others. "Productivity with Choice" is a cornerstone to this approach. When generating these, it is possible to make use of a wizard-driven approach to generate Business Services that provide Java interfaces to these tables. With simply a right-click these interfaces can then be exposed as web services, including SDO based web services. Additionally, keeping with the theme of being visual and declarative, it is also possible to accomplish the same thing via visual modeling to generate these interfaces.

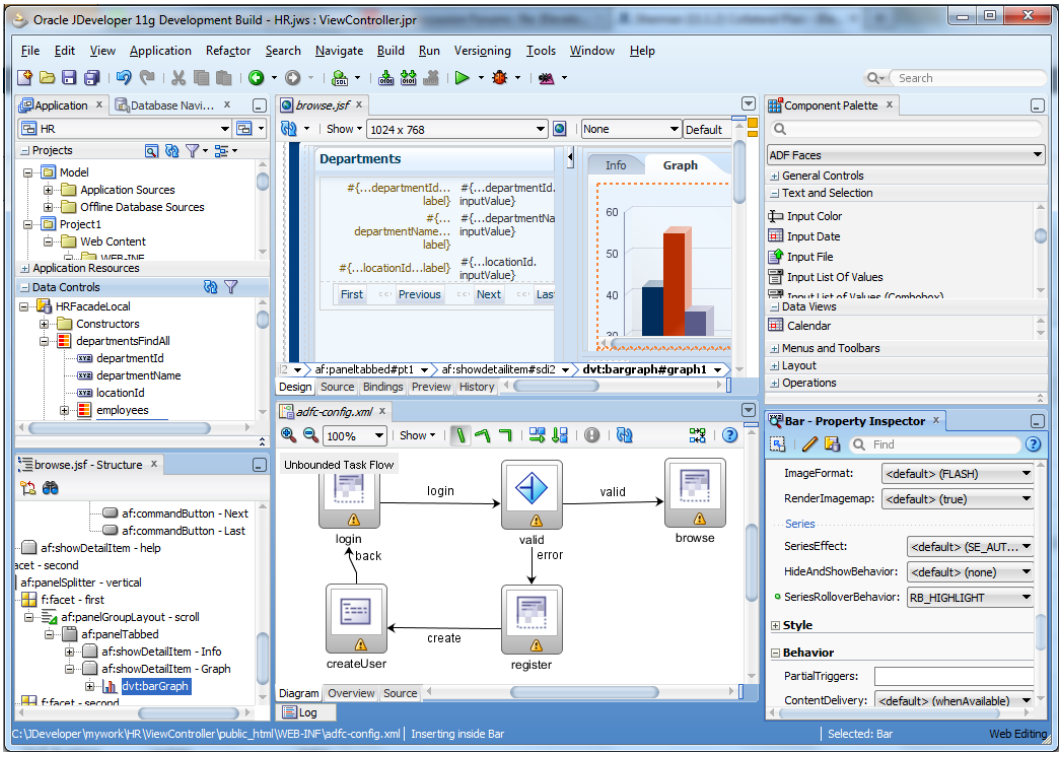
Oracle ADF Business Components is a framework focused on creating objects, which implement the Business Services layer on top of a data source, in a more declarative way. It provides out-of-the-box services such as transaction management, resource pooling, locking, declarative validation rules, translation, and object-relational mapping. Oracle ADF BC should feel familiar to developers with a background in 4gl declarative database driven development, offering such features as defining java objects based on SQL, declarative definition of validation rules, and pre-defined events where code can be injected into the business service life cycle. ADF BC development is done through declarative dialogs and property inspectors. With built-in implementation of common JAVA EE design patterns in the framework, the performance and scalability of the application is assured.

User Interface Development

Visual and declarative development features of the View and Controller layers of an application are plentiful in Oracle JDeveloper:

- ADF Faces - a large set of over a 150 UI components built on top of the standard JSF APIs that leverage the latest technologies — including partial page rendering and Ajax — to provide a rich, interactive user interface.
- A page flow modeler for the ADF controller and the standard JSF framework page flow, providing visual page flow modeling using simple drag and drop of components onto a diagram.

- A visual editor for JSP, JSF, HTML, Swing, and Wireless based user interfaces, allowing WYSIWYG development for all types of components.
- Declarative development tools for adding components to the user interface, including the creation of declarative components, a property inspector, extensible component palette, and data control palette.
- Reusability features – several features for maximizing reusability, including the creation of task flows, ADF Libraries, and declarative components.



Visual JSF development

The visual and declarative development tools are synchronized in the JDeveloper IDE so that the visual editor, property inspector, and modelers are synchronized with the source code at all times. Thus, developers can choose their development style – drag and drop, declaratively define properties, or edit source code directly.

Binding Business Services Components to the User Interface

Oracle JDeveloper provides a very easy way to bind components from the Business Services layer to your Controller and View layers using an innovative binding layer approach. The Data Control Palette provides a view into the Business Services layer. Developers can simply drag-and-

drop data objects and bind them to their user interface implementation. The same mechanism enables an easy binding of controller actions to methods defined in the Business Services layer. All using purely visual and declarative gestures.

Benefits of Oracle ADF Over Other Frameworks

The key characteristics of Oracle ADF that makes it unique among other Java EE frameworks are:

End-to-End Solution – Oracle ADF doesn't focus on just one layer of the Java EE architecture. ADF provides an integrated and complete solution for every Java EE layer from the view layer and data-bindings, through the business services and data access; as well as support for every development life-cycle phase from inception through support.

Development Environment – Many of the other Java EE frameworks lack strong integrated support by development tools. Oracle JDeveloper provides visual aids and a declarative approach to minimize the need to write framework code, making it a perfect tool for building Oracle ADF-based applications. This declarative development approach also reduces the learning curve for developers familiar with 4GL-style tools. Developers who wish to use another IDE, such as Eclipse, are able to do so with built-in features provided in Oracle Enterprise Pack for Eclipse packaging and relying on ADF's support for the Java EE standards.

Platform Independence – Other frameworks lock developers into a specific software vendor. The Oracle ADF runtime, however, can be installed on various Java EE compliant application servers and business services can connect to any SQL-92 compliant database.

Technology Choice – Developers have preferences for the way they implement different layers of an application. Oracle ADF supports multiple technologies for each of the layers of the application and doesn't enforce a specific technology or a specific development style on the developer.

Technology Commitment - It is important to note that Oracle ADF is the technology choice for the Oracle next generation set of enterprise applications – Oracle Fusion Applications - and is in continuous use for internal development purposes. The product is used to develop Portal applications, wireless applications, and web applications, and therefore provides a committed, supported, and consistent technology stack.

Metadata-Driven – All layers of the Oracle ADF framework offer declarative options for development, configured from XML metadata, while accommodating custom coding wherever necessary. You can choose to use all or part of the framework in the applications you build, making the application components much more reusable and flexible. The use of metadata also enables rules for data bound fields to be specified at the model layer. Labels, validation, and tooltip properties can be specified in the metadata for ADF data bindings - those properties are utilized independent of the user interface implementation.

Declarative Customization – Oracle ADF provides a unique solution that allows an organization to use a single base application and customize it to fit the requirements of different users. Oracle ADF works in conjunction with a MetaData Services (MDS) layer that provides for application customization via two different implementation layers: The first, would be "seeded customization" which refers to an application wide customization that would be in effect for anyone accessing the application for a particular group. The second is "user customization" often referred to as "personalization" in which the end user designates customizations to their personal experience that are then persisted via the MDS repository.

Enhanced Reusability – JDeveloper + ADF provides support for superior reusability features including: JSF templating, reusable task flows, task flow templating, reusable business services, ADF libraries, JSF fragment based regions, and much, much more.

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.

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.

CONCLUSION

Oracle ADF makes Java EE development simpler by providing out of the box implementation of design patterns and infrastructure code. ADF provides a choice of development approach, technologies used, and deployment platform. Combining the advanced architecture of Oracle ADF with the visual development environment of Oracle JDeveloper 11g provides a perfect solution for both novice and experienced developers looking to be more productive when developing Java applications.

More resources about Oracle ADF and Oracle JDeveloper can be found on the Oracle Technology Network (OTN), at http://www.oracle.com/technetwork/developer-tools/jdev/overview/index.html.

Oracle Application Development Framework Overview
June 2011
Author: Shaun O'Brien and Shay Shmeltzer

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

Oracle is committed to developing practices and products that help protect the environment

0109