Developing Swing-based Java Clients using Oracle JDeveloper 10*g* and Oracle ADF JClient

An Oracle Whitepaper
January 2004

ORACLE

Developing Swing-based Java Clients using Oracle JDeveloper 10*g* and Oracle ADF JClient

Developing Swing-based Java Clients using Oracle JDeveloper 10*g* and Oracle ADF JClient

INTRODUCTION

J2EE is a standardized toolbox and blueprint that provides application developers with a choice of technologies for building applications in Java and XML. Java Swing is the tool to use for business applications that require features like:

- Rich and highly interactive user interfaces
- Instant field validation
- Scrolling
- MDI Windows support
- Local deployment and client-server deployment
- Offline support

ADF JClient is the name of a set of Java classes and editors in Oracle JDeveloper 10*g* that simplify the creation of databound Java Swing applications.

ADF JClient leverages the new Oracle Application Development Framework (ADF) to declaratively bind Swing components to datasources like Oracle ADF Business Components, Enterprise Java Beans (EJB), Oracle Application Server TopLink, web services, and JavaBeans. Using ADF JClient for building business applications in Swing provides a boost in developer productivity, complying with the J2EE standard.

ORACLE APPLICATION DEVELOPMENT FRAMEWORK (ADF) OVERVIEW

Oracle JDeveloper 10*g* includes an integrated J2EE design time and runtime framework, the Oracle Application Development Framework (ADF). Using Oracle ADF, application developers follow a consistent approach for developing J2EE applications independent from the user interface technology and business model they use—thus, the user's experience in JDeveloper is one of "productivity with choice".

The Application Development Framework model is an abstraction layer that knows how to manage and present data from different business services in a consistent way. It consists of a databinding layer and ADF Data Controls, as

described in JSR 227[1]. The Data Controls represent the complete data model of an application, which can include one or many business services of different types.

The databinding layer uses XML-specified meta-data to define how UI components on a page or Java panel access model data. The same meta-data is used during runtime to instantiate the databinding objects.

The role of a business service, for example ADF Business Components, is to provide the persistence layer for an application including business rules and data security.

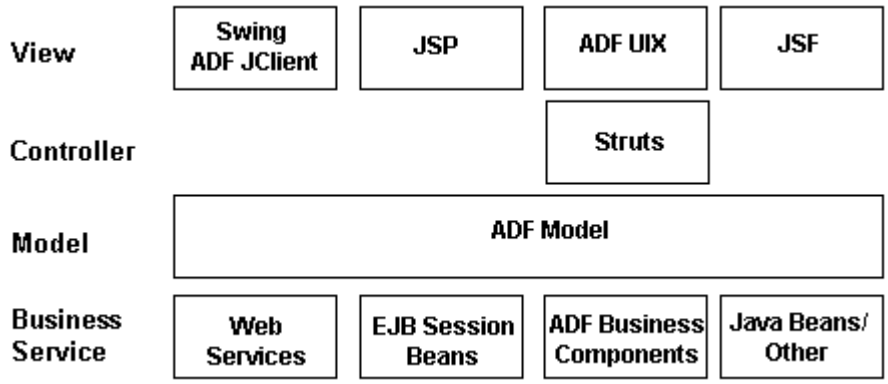

Figure 1: Oracle ADF Model View Controller architecture

In modern Model-View-Controller (MVC) architectures, the application user interface (the view) is decoupled from the business logic. Because Oracle ADF follows MVC, views can be built using Java Server Page (JSP), Oracle ADF UIX, Java Server Faces (JSF) and Java Swing. (JClient) for the same data model. The ADF model interacts with events from the controller and feeds data from the business service to the application view.

ADF JCLIENT OVERVIEW

Applications built with Oracle ADF JClient use Java Swing to render application interfaces. The Java Swing architecture incorporates a specialized version of the MVC architecture.

The View and the Controller in Swing exist as one component, the UI Delegate. The model of a Swing component defines the component's state, like a checkbox that can be checked or unchecked based on the state of the data it represents.

When the user selects a checkbox, data is updated in the model through events handled by the Swing controller of the UI Delegate. Another example of a model in Swing is the table model that can be based on an array of Strings or the result set of a JDBC connection.

[1] http:/ / www.jcp.org/ en/ jsr/ detail?id=227

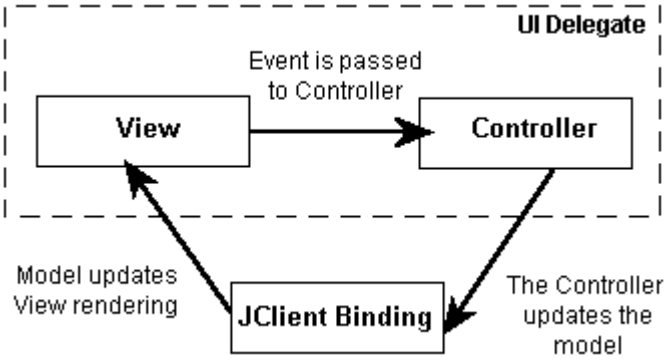

Figure 2: Swing UI Delegate architecture with JClient model binding

Oracle ADF JClient is a thin binding layer that acts as the model in Swing. It provides declarative binding of Swing UI components to business services like Enterprise JavaBeans. The ADF JClient binding is set on a Swing component using its setDocument() or setModel() method, while the properties of the JClient binding are specified as meta-data.

With Oracle ADF JClient, creating databound user interfaces in Swing is a completely declarative task that greatly improves developer productivity.

ADF JCLIENT ARCHITECTURE IN JDEVELOPER 10*g*

ADF JClient uses the Oracle Application Development Framework to declaratively bind Swing components to business services (Figure 3).

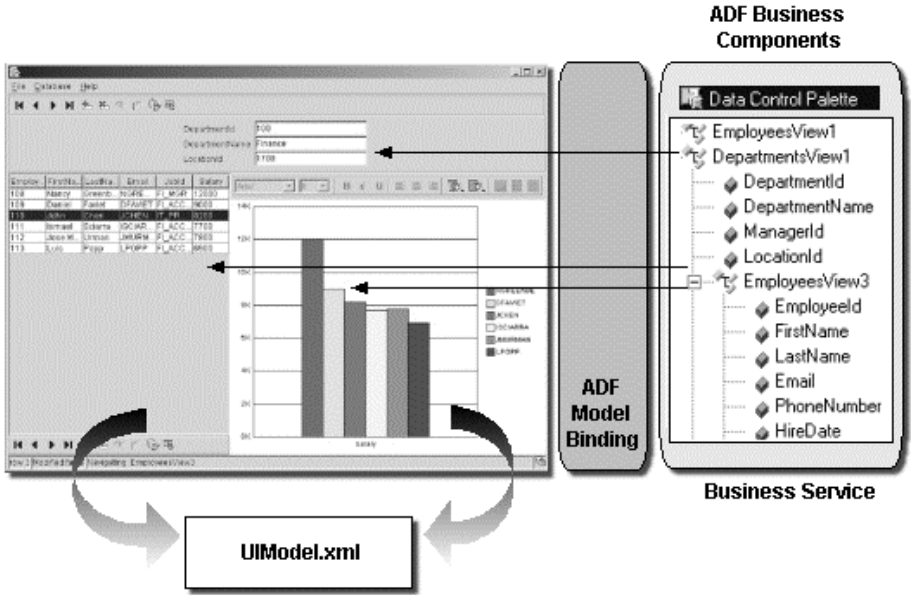

Figure 3: JClient architecture in Oracle ADF

A Swing component that is dropped into a JClient panel or form is bound to the ADF model by the databinding container. The databinding container maps UI components on a JClient panel or form through meta-data created at design time.

The meta-data file, <panel name>UIModel.xml, is automatically created for each JClient panel and form generated by the JClient panel and form wizards[2]. For example, A JClient panel "EmployeesView.java" will have its UI databinding defined in a meta-data file "EmployeesViewUIModel.xml".

To bind a Swing component to a model definition in the UIModel.xml file, the following syntax is used in a call to the Swing components setModel() or setDocument() method:

```
jTable1.setModel((TableModel)panelBinding.bindUIControl(
"EmployeesView1", jTable1));
```

Example 1: JTable binding to a View Object based on the Employees table

ADF JCLIENT DESIGN TIME

The ADF JClient design time consists of the following components

- Visual editor, to declaratively build Swing forms and panels
- Java Code Editor, to add application logic synchronized with the Visual editor
- UI Debugger, an innovative way to debug Swing applications and track UI events
- Property Inspector, to declaratively modify the Swing UI Beans
- Component Palette, to drag and drop unbound Swing components. The Component Palette for Swing also contains a 'Browse' button that enables developers to reuse existing JClient panels in their current project.
- Constraint window to declaratively define layout constraints for Swing components in the GridBagLayout
- Data Control Palette to drag and drop databound visual Swing elements to a Java layout. Application developers select the data attribute or view in the Data Control Palette and choose a UI component in the "Drag And Drop As" selection box to render it. The component is added to the JClient panel or form using drag and drop.
- Structure window to show the hierarchical structure of a Swing panel and to access container components. The Structure window also allows access

[2] The next version of ADF JClient converts existing Swing panels and frames into JClient panels and forms when adding the first databound UI component from the Data Binding Palette.

to binding editors for the declarative customization of attributes displayed in the JClient form or panel

- Application Navigator to filter the number of files displayed in a project to only those that need editing. The Application Navigator also presents a hierarchical view of an application based on its package structure
- System Navigator to show all files contained in a JClient project

Additional development support is provided through UML modelers for building the business service's data model.

Design Time source files

The following source files are created automatically when building ADF JClient applications. Only two files need editing when developing Swing applications with ADF JClient.

***.java** – The JClient application source files. This file can be edited through the visual Swing editor or the code editor.

***.dcx** – Specify the factory classes for registered ADF data controls that are not of type ADF Business Components. The dcx file is generated automatically and does not require editing.

***.cpx** – The DataBinding.cpx file is automatically created and contains the data controls for the business service used within an application as well as a reference to the UI client model definition in the UIModel.xml files. This file does not need editing.

***.xml** – The <panel name> UIModel.xml file is automatically created when dragging databound Swing components from the Data Control Palette to a JClient panel or frame. The UIModel.xml file defines the model reference for the components used within a particular panel. The UIModel file can be visually edited in JDeveloper.

The meta-data files used at design time are deployed with the application and also used during application runtime

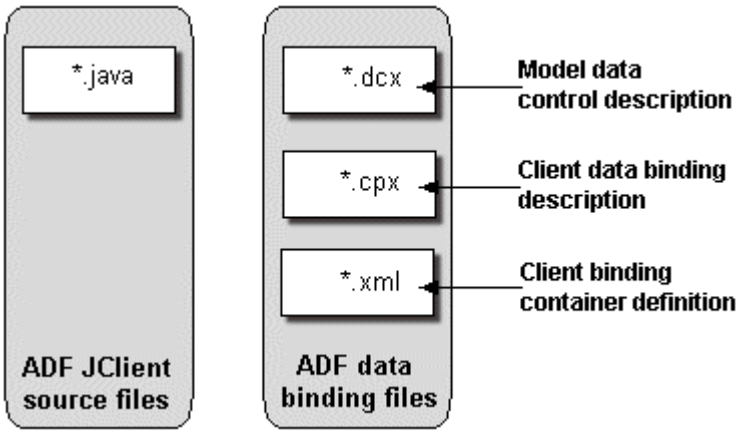

Figure 4: ADF JClient source files

Oracle ADF Data Control Palette

The Oracle ADF Data Control Palette is new in Oracle JDeveloper 10*g* and is used to declaratively bind Swing UI components to collections, structured objects, attributes, or methods in the data model.

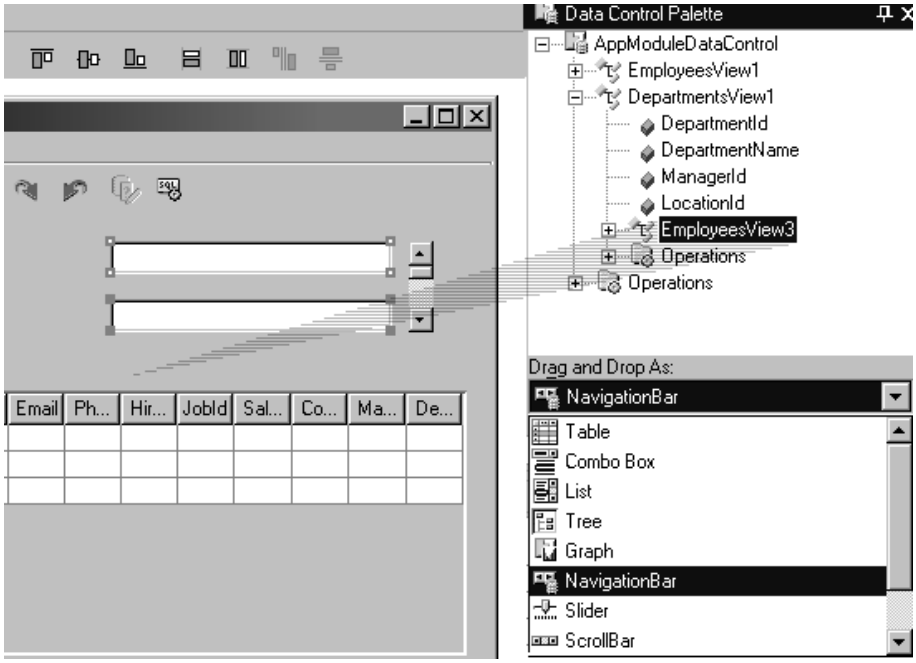

Figure 5: Data Control Palette for drag-and-drop development

Selecting an entry in the model tree (shown in Figure 5 for ADF Business Components) automatically preselects a list of possible UI components suitable to render this data. The application developer uses drag-and-drop to add the databound Swing component to a JClient panel or form. The UIModel.xml binding file is updated for the new component.

The databinding information for a Swing component can be customized by using the Structure window or the Property Inspector.

Structure Window

The Structure window shows the structure of a selected file dependent on its file type. For Java files the Structure window shows a list of methods and variables together with their access privileges. Swing panels are shown in their component hierarchy, allowing direct access to UI containers nested within the panel.

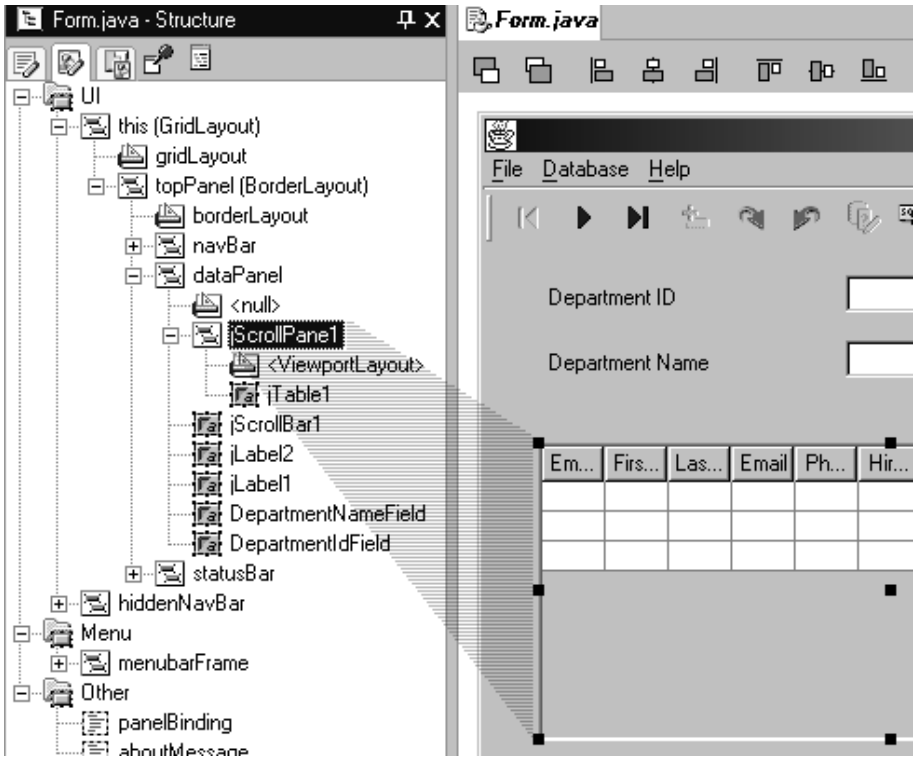

Figure 6: Accessing Swing containers using the Structure Window

Selecting the UIModel.xml file (or just clicking the UI Model tab in the Structrue window for the open JClient panel) shows the contained RowSetIterators and the UI component data bindings.

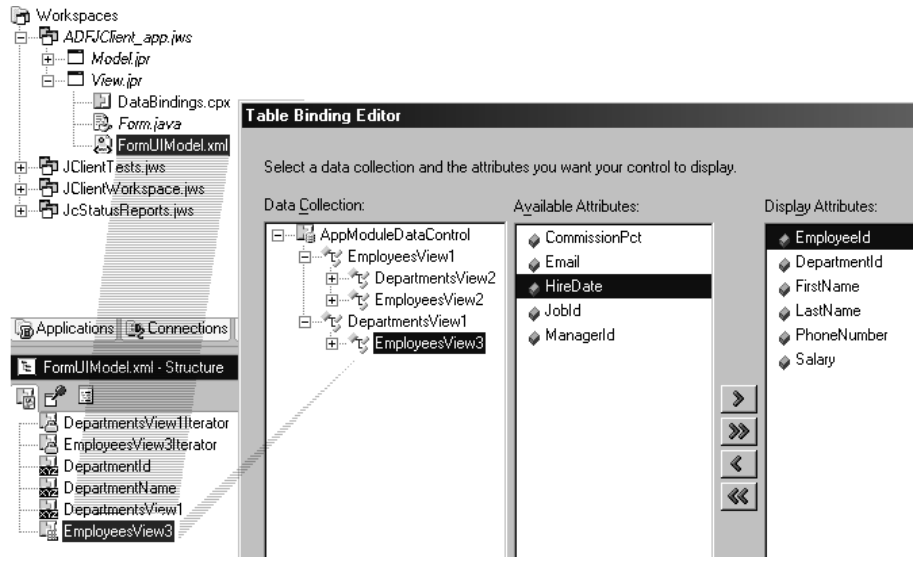

Figure 7: Customizing the binding container meta-data UIModel.xml file

Selecting a UI component binding in the Structure Window and choosing "Edit" from the right mouse button context menu, displays the binding's editor to declaratively customize the data shown by this UI component.

ADF JCLIENT DEPLOYMENT OPTIONS

ADF JClient applications can be deployed locally as client-server applications and remotely in a three-tier architecture. Using the Java Web Start support in JDeveloper, the ADF JClient application can be maintained on the middle tier server and automatically deploy to a local client the first time a user requests the application.

Using ADF Business Components as a business service in Oracle ADF, the business logic used with JClient can either be deployed on the local client or as remote EJB session beans.

The deployment option to use can be determined at the end of the application development process and need not be considered when building it.

BACKWARD COMPATIBILITY

Binding Swing components to data provided by business services has changed in ADF JClient in Oracle JDeveloper 10*g* compared to Oracle9*i* JDeveloper. The benefit of this change is that ADF JClient fully supports the Oracle ADF declarative databinding architecture, which means that JClient applications now can easily be bound to any datasource, for example Oracle ADF Business Components and OracleAS TopLink.

JClient applications built in Oracle9*i* JDeveloper continue working in Oracle JDeveloper 10*g* and can be further developed using the code editor.

The visual Swing editor of Oracle JDeveloper 10*g* will always generate the new ADF binding style and should not be used to further develop existing JClient panels. Developers can still use the visual editor of Oracle9*i* JDeveloper to maintain these applications.

A JClient application must not mix the new binding style with Oracle9*i* databinding style in one JClient panel. It is possible to build a JClient application that consists of panels using the "old style" binding syntax and other panels using the ADF style binding. This way, applications don't need to be upgraded, to be used and further developed in Oracle JDeveloper 10*g*.

CONCLUSION

Oracle ADF JClient in Oracle JDeveloper 10*g* uses the new Oracle Application Development Framework (ADF) to declaratively bind Swing components to data sources like ADF Business Components, Enterprise Java Beans, web services, and many more. ADF JClient increases the productivity of Swing application developers by offering a declarative development environment that fully supports the complete development lifecycle of Swing applications.

Developing Swing-based Java Clients using Oracle JDeveloper 10*g* and Oracle ADF JClient
January 2004
Author: Frank Nimphius
Contributing Authors:

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
www.oracle.com

Oracle Corporation provides the software
that powers the internet.