

# An Overview of Databinding Features in Oracle ADF

*An Oracle White Paper  
April 2004*

# An Overview of Databinding Features in Oracle ADF

Introduction .....	3
ADF Architecture .....	3
Key Benefits of the ADF Model Layer .....	4
A Layer of Abstraction .....	4
A Consistent Design-Time Experience .....	5
Databinding for Any Type of Business Service .....	5
Support for Standards .....	5
Fully Extensible Framework .....	5
Model Layer Components .....	5
ADF Data control .....	6
ADF Bindings .....	7
Further Reading .....	8

# An Overview of Data Binding Features in Oracle ADF

## INTRODUCTION

The Oracle Application Development Framework (ADF) in JDeveloper 10g contains a powerful set of new features for creating data-bound J2EE applications. Databinding functionality is abstracted into a set of *data controls* and *bindings*, objects that support binding any J2EE client to any business service. New design-time tools simplify the task of creating a data-bound client.

## ADF Architecture

Oracle ADF applications are based on the *Model-View-Controller (MVC)* architecture. This widely used approach divides applications into layers, each with its own specific responsibilities. This separation of responsibilities avoids code duplication and makes applications easier to maintain and expand.

The ADF application architecture has four layers, illustrated in Figure 1 below:

- **Business Services Layer**  
Implement the bulk of your application's functionality. This layer is where data access, persistence, business rules and transaction handling are implemented.
- **Model Layer**  
Provides a client-side wrapper and abstraction of your business service's data model and operations. The view accesses business services through the model layer.
- **View Layer**  
Renders the contents of the model. The view defines how your data should be presented.
- **Controller Layer**  
Defines the application's flow and behavior. It takes user input and performs the appropriate action, such as routing the user to a different page, or calling an operation in the model.

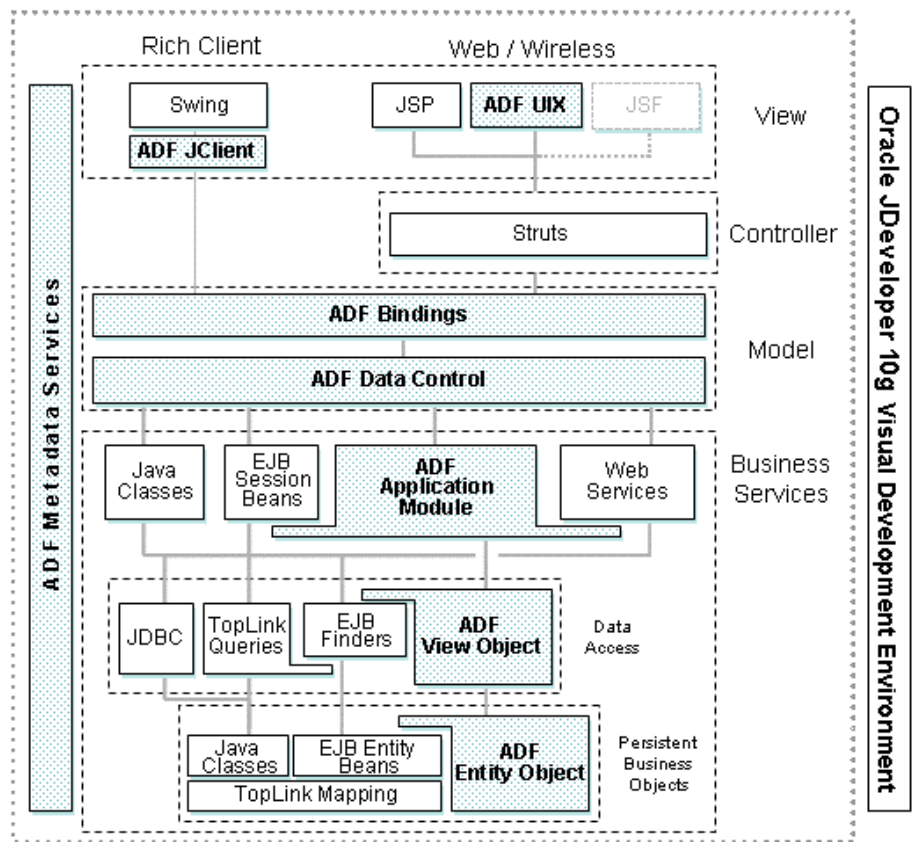


Figure 1: Oracle ADF Architecture

### Databinding and the ADF Architecture

Figure 1 shows where databinding fits into the ADF architecture: the bindings and data controls that implement databinding functionality make up the model layer of an ADF application, with the application's view and controller accessing data and operations through the model layer.

### KEY BENEFITS OF THE ADF MODEL LAYER

The separation of databinding objects into the application's model layer, and JDeveloper's design-time tools for creating data bound applications, provide several benefits to application developers.

#### A Layer of Abstraction

The ADF model introduces a layer of abstraction: the client binds to the model layer instead of binding directly to the business service. The client contains no direct references to the business service; instead the client contains references to the ADF model layer; and the ADF model layer contains reference to the business service. This clean separation of layers makes applications easier to maintain. For example, a change to the business service requires only one change to the model layer; it doesn't require a change to every JSP page in your application.

## **A Consistent Design-Time Experience**

JDeveloper 10g provides a drag and drop interface for binding data and operations to a user interface. This drag and drop interface is the same for all types of clients and business services, resulting in a more consistent and intuitive design-time experience.

## **Databinding for Any Type of Business Service**

JDeveloper 10g provides the ability to create data bound clients for a number of different business services technologies, including ADF Business Components, EJB session beans, web services, TopLink mapped Java objects, and JavaBeans. This provides a productivity gain for developers who prefer to use non-framework technologies to develop their business services, or who want to create clients for existing web services or EJB session beans.

## **Support for Standards**

Until now there has been no standard for databinding in Java, so developers typically hard-coded databinding into their client tier. Oracle is working as the specification lead for JSR 227

(<http://otn.oracle.com/products/jdev/htdocs/techinfo/jsr227.html>), which proposes a standard databinding and data access facility for J2EE. The ADF databinding in JDeveloper 10g is an early implementation of JSR 227, and the features will evolve in future releases to support the final JSR.

## **Fully Extensible Framework**

The ADF framework provides ready-made data controls for several types of business services. You can also use the ADF model API to implement your own data controls, allowing you to use the ADF databinding design time to build clients for your own custom data source.

## **MODEL LAYER COMPONENTS**

The ADF model layer consists of two types of components:

- **ADF data controls**  
Implement a client-side abstraction of the business service layer's data model.
- **ADF bindings**  
Define how the client works with data control.

You can also think of the data controls as the middle-tier-facing components, and the bindings as the client-facing components. Figure 1 above shows these two types of components.

## ADF Data control

The data control defines the data model returned by the business service, generally made up of collections, attributes, and operations. You create a data control by dragging a business service into JDeveloper's data control palette.

Figure 2 shows how an example data control is displayed at design time in the data control palette; this data control has a data model consisting of a collection of customers and a collection of orders, linked in a master-detail relationship. In this case the underlying business service happens to have been created with ADF Business Components. However, since the data control does not expose implementation details of the underlying business service, the example could also be showing the data control for another type of business service, such as a JavaBean.

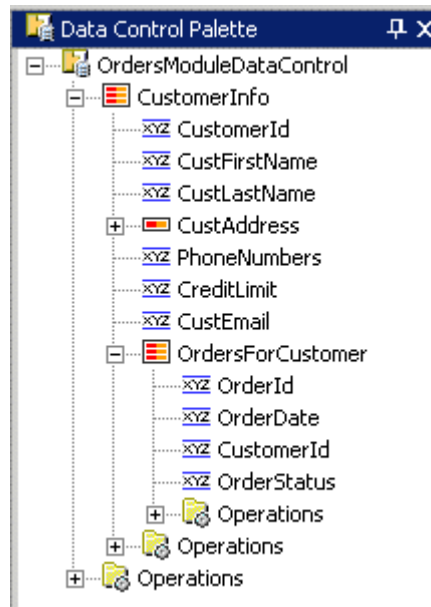


Figure 2: Data control with customers and orders collections

Figure 3 expands the customer Operations folder in figure 2, showing the list of operations available for the customer collection.

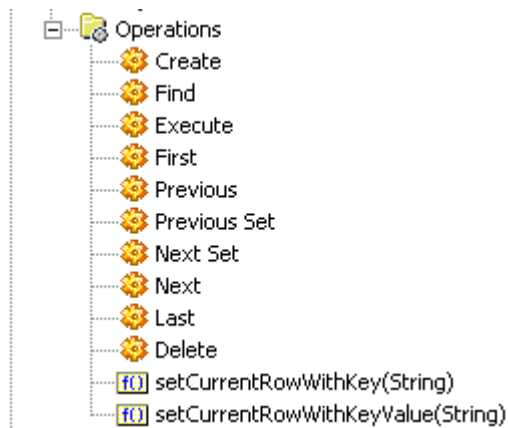


Figure 3: Customer Operations

### ADF Bindings

ADF Bindings define how client components work with the data control. For example, Figure 4 shows a very simple JSP page that displays a customer id and last name, using buttons to navigate to the next or previous customer record.



Figure 4: Simple JSP showing customer information

Figure 5 shows the structure pane, containing the bindings used by this simple page:

- **CustomerInfoIterator:** an *iterator binding* to maintain the currency and state of the customer row set;
- **CustomerId:** a *text binding* to render the CustomerId attribute as a text string;
- **CustLastName:** another *text binding* to render the LastName attribute as a text string;
- **Next:** an *action binding* to execute the Previous action;
- **Previous:** an *action binding* to execute the Next action.

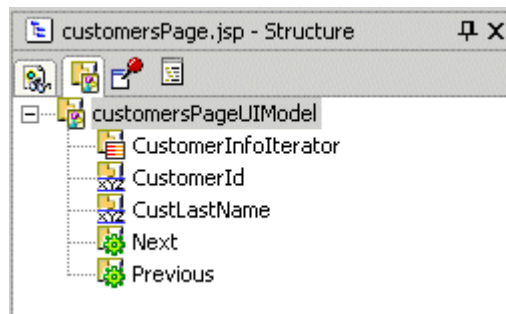


Figure 5: Bindings used by the customer page

The set of bindings described above make up the page's *UI model* or *binding container*. The UI model defines how this particular page renders data from our OrdersModule data control. A different page might render the same data in a different way; for example, CustLastName could be rendered as a dropdown list, in which case the page's UI model would contain a list binding for the CustLastName attribute.

Bindings are created automatically when you drop an attribute from the data control palette onto a page. The JSP page uses JavaServer Pages Standard Tag Library (JSTL) tags with standard expression language (EL) to display the data. For example, the tag to display the customer id is:

```
<c:out value="\${bindings.CustomerId}"/>
```

which displays the value of the CustomerId binding (remember that the CustomerId binding renders the value of the CustomerId attribute as a text string).

This example describes only a few of the different types of bindings; see the Further Reading section for links to more detailed information.

## FURTHER READING

This document has given a high-level overview of the ADF model layer. Below are links to more detailed documentation, as well as some samples and tutorials to help you work with the ADF model layer.

- Oracle ADF Databinding Primer: More detailed overview of the ADF model.  
<http://otn.oracle.com/products/jdev/htdocs/adfprimer/index.html>
- OBE tutorials: Detailed tutorials that walk you through creating ADF applications.  
<http://otn.oracle.com/obe/obe9051jdev/index.htm>
- Online demonstrations of ADF databinding features.  
[http://otn.oracle.com/products/jdev/viewlets/new\\_viewlet.html](http://otn.oracle.com/products/jdev/viewlets/new_viewlet.html)
- JDeveloper documentation: The full set of JDeveloper help topics. In JDeveloper, select Help > Topics from the main menu.



An Overview of Databinding Features in Oracle ADF

April 2004

Author: Blaise Ribet

Version 1.0

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](http://www.oracle.com)

Copyright © 2004, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.