Quick Web Development Using JDeveloper 10*g*

By Bradley D. Brown

*Have you ever done something the long way and then learned about a new shortcut that saved you a lot of time and energy? There was a time when tools like Oracle Forms were easier to use for making major edits by editing the INP file rather than using the GUI development tool. Maybe you've performed some database administration tasks and you prefer the SQL*Plus prompt over Oracle Enterprise Manager (OEM).*

If you've ever encountered Web-based Java development-using Struts, for example-you know the "long way" of developing such an application. Oracle's new framework, bundled into a powerful GUI development environment, is called Application Developer Framework (ADF). ADF allows you to quickly and easily develop a GUI/HTML-based Web application using this point-and-click, drag-and-drop development tool.

This article shows you how to:

- Map a database object to an ADF business component.
- Build an ADF business component view object.
- Create an ADF business component application module.
- Compose the ADF data bindings and control components.
- Use the Struts controller.
- Generate the Java Server Page.
- Run the application.

In a short period of time, you'll learn how to quickly develop an application using JDeveloper 10*g*.

Quick Web Development

JDeveloper allows you to branch from a database table to an HTML-based Web application with full navigation control in no time at all. The most impressive thing about it is that ADF implements a full MVC (model/view/controller) or Struts model into this framework. Think of the model as your business components, view as your presentation layer (a JSP) and controller as Struts. The following diagram shows the ADF graphically.

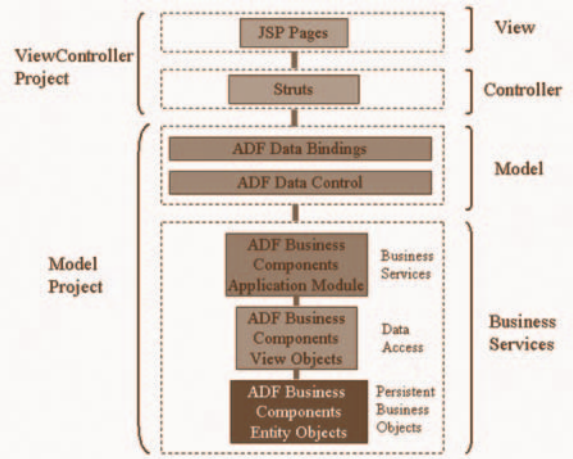


Figure 1. Oracle's Application Developer Framework

There are eight simple steps to creating your first success with ADF. These include:

Create a:

1. New workspace
2. Model (business components) diagram
3. Persistent business object
4. View object
5. Application module Controller
6. Create a page flow View
7. Create a JSP
8. Run the application

Step 1 – Create a New Workspace

For the first step to creating your ADF-based application, you'll need to create a new workspace. Right click on Applications in the navigator and then click on New Application Workspace, as shown in Figure 2.

Figure 2. Creating a new application workspace

This will display the Create Application Workspace dialog box where you will need to provide a name for the workspace (**NOTE:** *You can name it whatever you wish.*). Be sure to use the default application template (Web application) and then click OK. Your new application workspace is then created and you are ready for Step 2.

Step 2 – Create a Model (Business Components) Diagram

You will notice that two new projects are automatically created under the application:

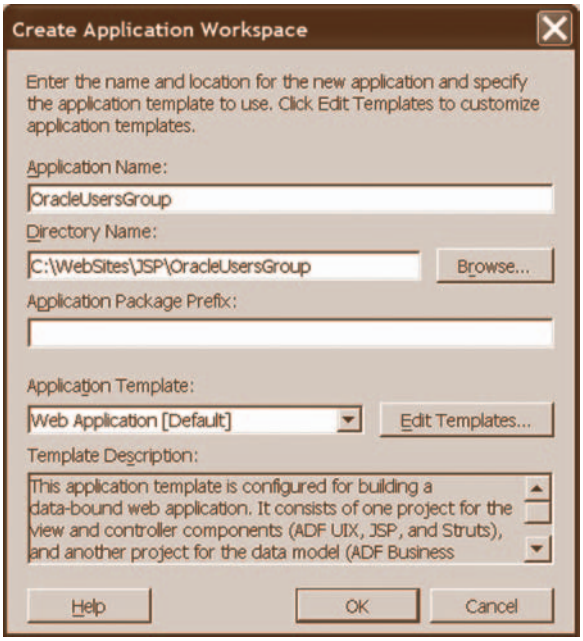
- Model
- 

Figure 3. Automatically created projects

Right click on the Model and then click New. This will display the categories of components available. Click on Business Components under the Business Tier and, from the gallery of components displayed, choose Business Components Diagram. Then click OK.

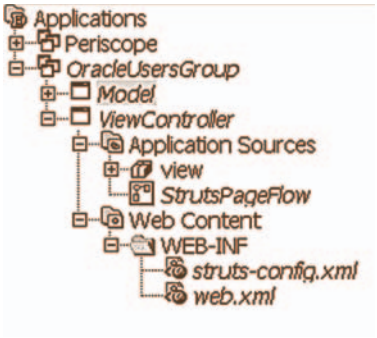


Figure 4. Creating a new business components diagram

You will need to provide a name for your Business Component Diagram in the dialog box provided. (**NOTE:** *You can name it as you wish*). Now that you have a new business components diagram, you're ready to begin building your components, as you'll see in Step 3.

Step 3 – Create a Persistent Business Object

Keep in mind that persistent business objects are really entity objects. These objects implement the Data Access Object (DAO) design pattern. They are responsible for persisting, caching and validating data. This is where JDeveloper starts to demonstrate its power. Previously, you were required to write such objects from scratch. Using JDeveloper, these objects are now created by dragging a table object and dropping it in the diagram.

Take a minute to look back at the diagram in Figure 1. Note that the entity object is at the bottom of this diagram in the business services section.

The connections tab in JDeveloper is used to select database (and other) objects. You must be able to connect to the database from your JDeveloper machine. In the navigator, select Connections at the bottom tab. Then, right click on Database and select New Database Connection, as shown in Figure 5.

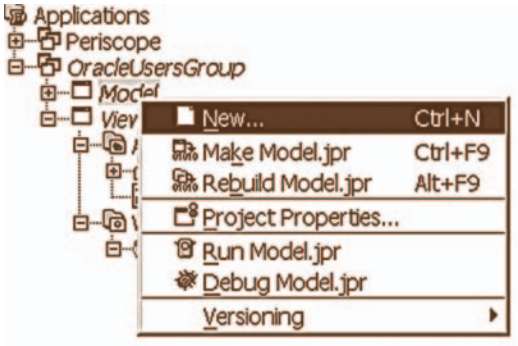


Figure 5. The Connections tab for creating a new database connection

The database connection wizard includes four steps to a successful connection. First, you'll need to name the connection. Next, you'll specify a username and password to connect to. Then, you'll specify the connection parameters (i.e., the database and machine to connect to). Finally, you'll test the connection.

continued on page 12

Once the database connection has been created, you can drag-and-drop a table from the Connections window into your Business Components diagram. In the navigator, you'll drill into the tables for your connection. Once you find the table of your choice, drag-and-drop it into the Business Components diagram, as shown in Figure 6.

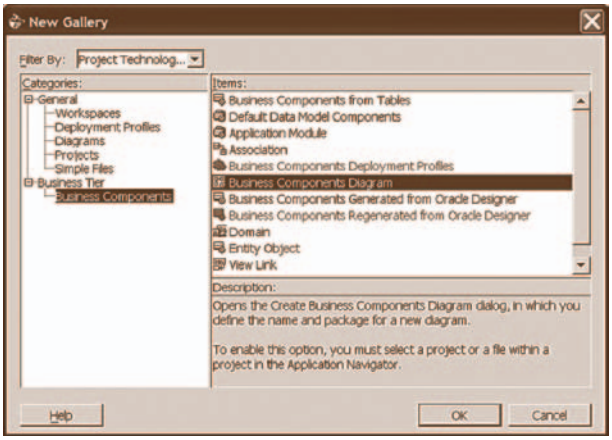


Figure 6. Dragging and dropping a table onto the Business Components diagram

JDeveloper allows you to edit the entity object as much as you like-through the GUI interface. To edit the persistent business object, simply double click the object. You can edit, add or delete attributes, turn on batch updates, establish validation and authentication rules, publish and subscribe events and more. The object itself is a BC4J or EJB object that encapsulates all of the business rules.

The modeling component within JDeveloper includes full knowledge of the database's referential integrity rules. If you add objects that have referential integrity established, JDeveloper automatically picks up on these relationships.

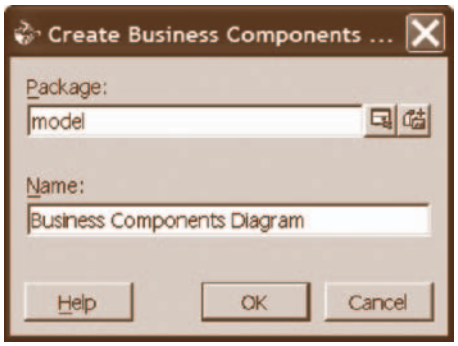


Figure 7. Referential integrity knowledge

Once you create all of your entity objects (an object for each database table object), you're ready to create your view objects. All database operations should go through these entity objects since they are responsible for all persisting, caching and validating data in and out of the table.

Step 4 – Create a View Object

It's now time to create your view objects. View objects are simply SQL statements that represent a collection of entity objects. In the SQL world, you can think of view objects as a SQL statement or database query. The view object can be found in Figure 8.

In the Business Components diagram, you'll first click View object. Click on the diagram to place the view object. You can name the view as you wish (i.e., EmpView). At this point you have an empty view (no entities to pull).

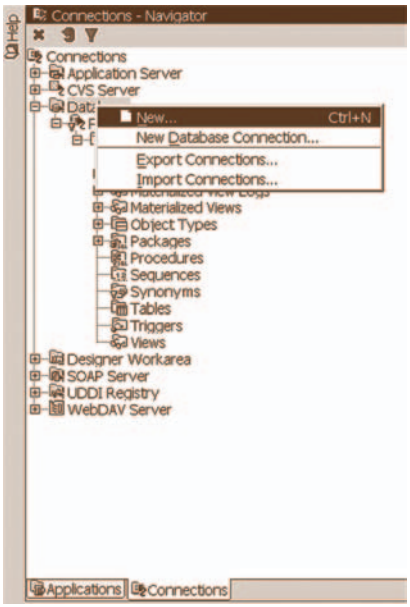


Figure 8. View object that was created

It's now time to place the Entity objects into your new View object. To do so, simply drag and drop the entity object(s) (i.e., Emp and Dept) into the View object (i.e., EmpView). This creates a 1:1 mapping from the Entity object by default (all columns, all rows). Note that if you add related tables, JDeveloper will automatically create the join statements.

Figure 9. Entity objects placed in the View object

Just like you can edit Entity objects in JDeveloper, you can also edit View objects. JDeveloper allows you to add additional Entity objects into the view (or use the drag and drop), add or remove attributes from the entities selected, edit the query (manually), add a where and/or order by clause, set fetch sizes, query hints, etc.

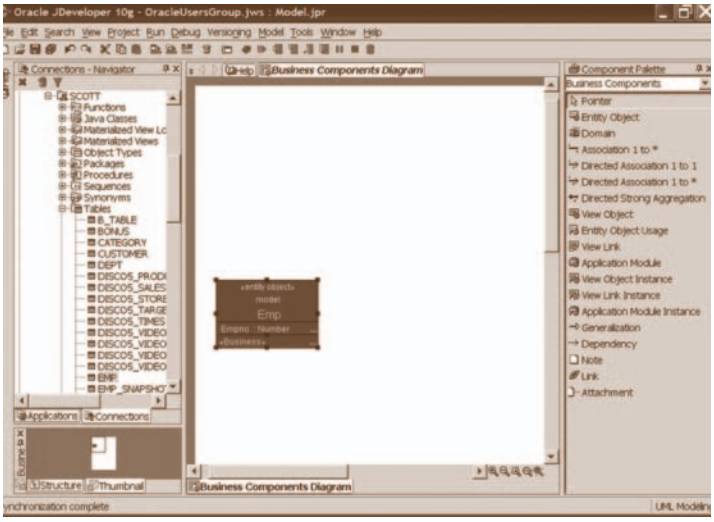


Figure 10. The View object editor

Once you're done creating your view objects, you're ready to create your application modules in Step 5.

Step 5 – Create an Application Module

The application module is shown in Figure 11. It is a service object that coordinates view objects for a specific task (i.e., a form to display the employee data).

To build the application module, once again you'll use the Business Components diagram. Simply click Application Module and then click in the Business Components diagram to place the Application module. Next, you'll name the module (i.e., EmpModule). You'll notice that this process is similar to the one creating a view object. In fact, you now have an empty module (no views to pull). You'll need to drag and drop the view (i.e., EmpView) objects into the module.

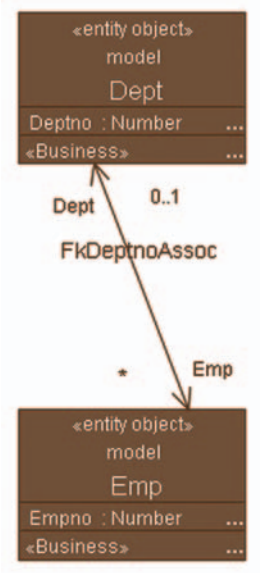


Figure 11. Application module with a view

Just like entity and view objects, you can edit application module objects. You can edit the attributes of the application module by double clicking the object. These objects can be deployed as EJB, CORBA or Web service.

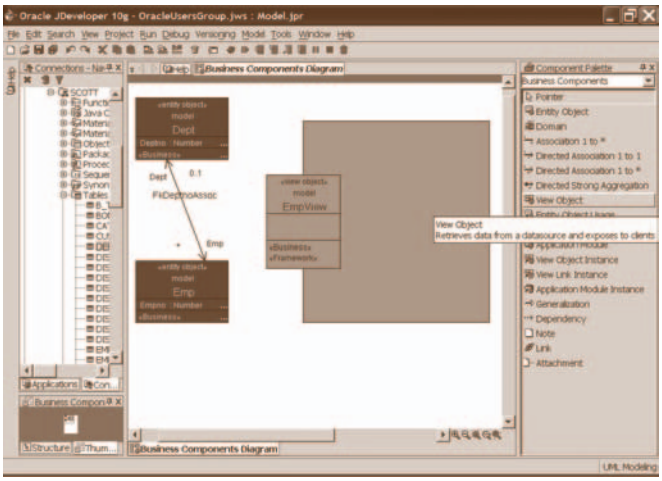


Figure 12. The Application module editor

Once you've created your application modules, the business services are complete. All of the business components have now been created-specifically, our entity objects, view objects and application modules. It's now time to create the GUI application. In other words, we're getting closer!

Step 6 – Create the Controller (Page Flow)

Referring back to Figure 1, we're in the controller, or Struts, section of the diagram. In JDeveloper projects, this is our ViewController project. The controller (Struts) separates the visual representation of Web pages (view) from their flow and actions.

In the JDeveloper navigator, you'll want to double click the StrutsPageFlow. Next, you'll want to click on page forward in the component palette. Then click in the StrutsPageFlow window to place the new page. Change the name from "Page1" (don't remove the forward slash) to the name of your choice (i.e., /emp).

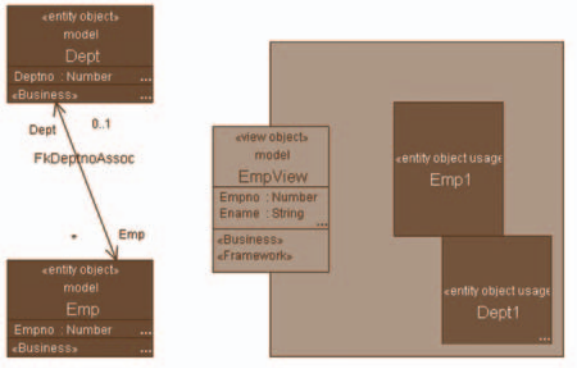


Figure 13. Struts page flows

That's an easy task now, isn't it? It's now time to generate the JSP.

continued on page 14

Step 7 – Create a JSP

We're now at the top of Figure 1-in the view section. It's time to create a JSP that automatically binds to a Struts action. To accomplish this, simply double click the Struts/forward action icon on the page flow diagram and then name the Java Server Page. Make sure edit this page now is clicked and click OK. You'll start off with a blank JSP. You'll need to drop your business components onto the page at this time.

Utilizing the business service you created in the model, you'll create the JSP components. In the data control palette, select the view object (i.e., EmpView). Select how you wish to drop the object into the page (i.e., HTML table, dynamic table, input form, read-only form, graph, etc.). Finally, drag and drop it onto the page.

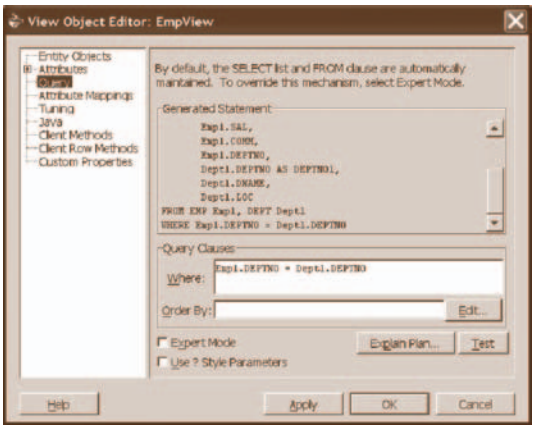


Figure 14. Creating an input form

To create an input form for a view object, drag and drop the EmpView object as an input form. Each column from the table(s) is displayed by default. Note that no navigation buttons exist (only submit) by default.

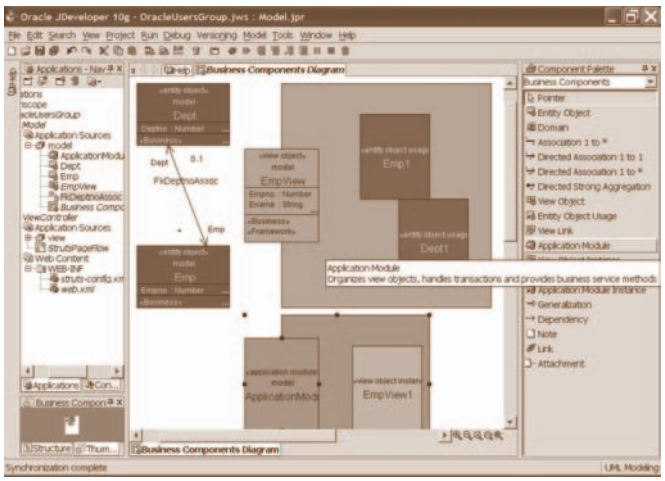


Figure 15. Default input form

At this point, you can add navigation to your form. In the data control palette under the view and under operations, you will find navigation operations.

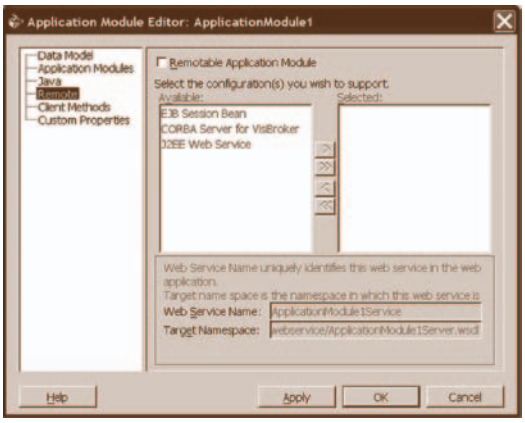


Figure 16. Data control palette's operations

These can be dragged and dropped onto the JSP.

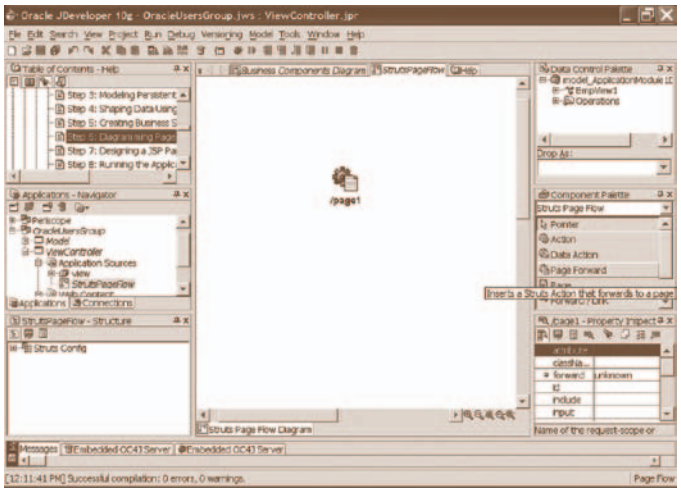


Figure 17. Operations on the JSP

When you're done visually editing the page, it's time to run your application-how exciting!

Step 8 – Run the Application

It's now time to run the application. Navigate back to the StrutsPageFlow diagram. You'll notice that the DataAction element was added for the JSP. Right click on the DataAction element and choose run.

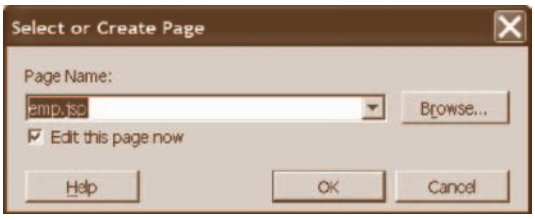


Figure 18. Run the application

The form is complete and fully functional! It can perform insert, update, delete and query operations. If you go back and change any validation rules on an entity object, redeploy the application. This is implemented into the application. Everything gets inherited. That's why we build all of these components.

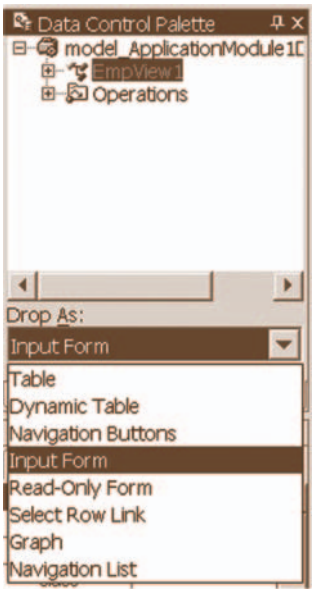


Figure 19. The fully functional application

The application that is created is a HTML-based application. In other words, there is no Java running on the browser (unlike Oracle Forms). The application is easy to deploy using the one-click deployment functionality of JDeveloper.

So, how easy is this process? If you've done it the "long way," you'll be amazed at how simple and powerful this is. If you haven't, you're probably thinking it's tough to do. Imagine taking calculus without first having algebra.

Tips and Tricks

JDeveloper is a powerful tool. A few tips that have helped me when developing applications include:

- Create a workspace for each application
- Create a project for each piece of application
- Set up an independent Application Server for testing

When it comes to developing JSP, putting too much Java code directly in JSPs is dangerous/bad-that's why Struts is so good. It follows best practices. When it comes to logging warning and error messages, Log4J (Log for Java) by the Jakarta Project is excellent. You'll probably find that JDeveloper's error messages aren't always clear, but they are getting better with each release.

Conclusion

JDeveloper is a powerful product! There are many great new features in the 10*g* release. My favorite is the Application Developer Framework. Enhancements to the visual creating and editing of code have been well done. Overall, I believe JDeveloper decreases development time and reduces the Java learning curve, which is good!

About the Author

Bradley D. Brown is chairman of the board, chief architect and co-founder of TUSC. Brad is recognized globally as an expert in Web technology. His current drive, motivation, and passion are behind Oracle9*i*AS, Oracle Portal, XML, wireless and Java technologies. In 2002, he added the development and release of "Periscope" to his impressive list of accomplishments. Brad is the author of "Oracle9*i* Web Development" which is part of "The Ultimate Oracle Library" series of books TUSC has written for Oracle Press and Osborne/McGraw-Hill. Along with co-founders **Richard J. Niemiec** and **Joseph C. Trezzo**, Brad won the Ernst & Young Entrepreneur of the Year Award in 2001 in the category of E-developer and was inducted into the Chicago-Area Entrepreneurship Hall of Fame.