A JDEVELOPER PRIMER FOR ORACLE FORMS DEVELOPERS

Peter Koletzke, Quovera

Real programmers can write assembly code in any language.

—Larry Wall, author of the Perl language

You've been working in the world of Oracle Forms Developer and, are accustomed to interacting with that tool to quickly create applications. You now see Java on your horizon and decide to take a look at JDeveloper —Oracle's Java development tool. When you first open the tool, you see some similarities to Forms, but quickly find that you are in a very different world. You will probably come up with a number of questions such as the following:

- Do I have to give up on Developer and focus development work on JDeveloper?
- Which tool is better for web application development?
- How do I choose between the Java and traditional Oracle development environments?
- How does development work using JDeveloper differ from development work in Forms?

This paper answers these questions. The paper starts with a discussion of how you make the decision about which tool to use for a particular application. This discussion will help you answer the first two questions and give you an idea of the positioning of the tool in the Oracle development world. The paper then provides a brief orientation on how the Java programming environment works from the perspective of a Forms developer to answer the last two questions.

The objective of this paper is to provide a primer or introduction to the tool by relating the work in JDeveloper and the Java environment to the work you are familiar with as a user of Oracle Forms Developer. Although you need to have a working knowledge of the Java language to be effective in JDeveloper, this paper does not assume a background in Java. If you have used client/server development tools other than Oracle Developer, the following discussion will still be of some use because the tasks and considerations are similar regardless of the development product.

Note

The term, Oracle Developer, used in this paper is a combination of the products known (at this writing) as Oracle Forms Developer and Oracle Reports Developer. Oracle Developer has evolved from development tools that Oracle has offered for many years under names such as IAD, SQL*Forms, Cooperative Development Environment (CDE) Forms, Oracle Forms, Oracle Developer Forms, Developer/2000 Forms, and Form Builder (and variations on all of those names for the report development tool).

If you've watched JDeveloper progress through its various releases from its days as App Builder up to the current release 3.2, you will probably ask an additional question: Is the product ready for prime time? The answer is "definitely yes." As you start to explore the current version, you will quickly understand why the 3.2 release is a solid, productive development environment.

Which Development Tool to Use?

The answer to the question of which is better for you, Developer or JDeveloper, will always be “It depends.” The products are intended for different uses. However, if you are currently working in an Oracle Developer environment, you might wonder which tool to use for a specific application. If you decide that Java is the direction to follow, JDeveloper is the best choice. If you mainly have access to Oracle Developer talent and only need to deploy your system internally, your decision is similarly easy.

This author has used and promoted Oracle Developer for many years (and has even co-authored an Oracle Press book on the subject). Developer has many compelling strengths and is the tool of choice in many situations. JDeveloper also has many compelling strengths and is the tool of choice in many other situations. The two products are not in direct competition—they are complementary parts of a suite of development products that is marketed and sold at this writing as Internet Developer Suite (IDS) although it may soon be renamed as Oracle9*i* Developer Suite. Therefore, there is no purchasing decision to be made; when you purchase IDS, you have both tools at your disposal.

The first thing you do when creating a system is to examine the business requirements and issues. Then, you must select a particular technology (for example, Java or more traditional client/server) that will best satisfy these requirements. You then choose a tool that fits that technology and your environment. Java is now a technology to consider. Therefore, one of the first steps after determining the requirements of the business is to decide whether to launch into the Java environment.

Why Use Java?

Java is an emerging language that provides many advantages over other programming languages. Java offers a modern, fully object-oriented environment for development and web deployment. The object orientation provides you with benefits in analysis and design because you can more easily match business concepts with objects than with standard relational structures.

The biggest drawback of Java is its newness. There are fewer experienced developers in Java than in traditional development environments. The reason is that the Java environment is nowhere near as mature as a traditional client/server environment that accesses a relational database. It is the author's opinion that there are relatively few Java-based production systems built to run against an Oracle database.

Working in a Java environment is very different from working with Oracle Developer, PowerBuilder, or Visual Basic (VB). For those who are new to the language, it will take time to learn the nuances of this environment. Even if you are committed to creating an organization-wide Java environment, building a traditional client/server application still makes sense under certain circumstances. If your development team has skills in another language, Java will require some retraining and ramp-up time. However, building applications directly in Java provides improved flexibility and the ability to build sophisticated applications. It also makes the transition of your business to the Web easier because Java is a primary language of the Web.

Deciding to Move to a Java-Based Development Environment

Most developers have experience building systems. Many shops, whether they are using Oracle Forms Developer, C++, VB, or PowerBuilder, are building client/server applications successfully and efficiently. This paper discusses the JDeveloper environment, which can be used in place of all of the previous development efforts. Does this mean that you should abandon your old development environment? Probably not. It is possible to set up a largely Java-based development environment (JDeveloper coupled with HTML, JavaScript, and XML) that can be used to support an entire organization's development needs. However, in the author's opinion, this is still very much bleeding-edge technology.. Very soon, it is likely that the development community will shift in the direction of Java-based tools and applications. In the long term, many organizations will move into an entirely Java-centric development environment.

Keeping it Simple

It is very difficult to maintain and support disparate development environments, even for large organizations. Organizations with applications built using whatever was the fashionable tool or language of the moment usually ended up with systems in chaos. It is preferable to work in a development environment that allows you to minimize the number of different tools and languages required to build a system.

One reason that the Oracle Developer environment has such an advantage over VB and PowerBuilder is that you can use PL/SQL for both front- and back-end development. There is a huge startup cost to becoming efficient in any programming environment. Requiring developers to work efficiently in two programming languages for the long term is not advantageous. This is why the Oracle Forms Developer environment is so compelling—it uses only one language.

In the Java environment, knowledge of only the Java language is not sufficient. Coding complex applications for e-commerce still requires knowledge of HTML (and JavaScript) as well as JSP tags and cascading style sheets. In addition, a secondary product such as Macromedia Dreamweaver or Microsoft FrontPage must be used as an HTML editor. Ultimately, it will be possible to leave the existing tools behind and move to a strictly Java-based environment. However, rushing too quickly toward such a decision is not advisable and in the interim period, developers will have to have expertise in both old and new environments.

Which Is Best?

With all of the current development tools trying to improve their web-enabled capabilities, it becomes increasingly difficult to make a convincing argument to abandon any specific technology. For example, following a long and difficult transition, Oracle Developer running over the Web is now a stable and viable environment. Oracle has not stopped there. A new technology currently called "Cherokee" (part of the upcoming Oracle9*i* Developer Suite) promises to provide yet another alternative for RAD development of Forms-style applications over the Internet.

The ability to cleanly and seamlessly deploy sophisticated applications using Developer talent is already available. The best bet at this point is to leave the core application development in whatever legacy environment you are comfortable with and build a few limited-scope systems in the Java/JDeveloper environment. Once you have some experience in building and deploying production applications of that style, you can make an informed decision about whether your organization is ready to make the transition to an entirely Java-based environment. There may still be good reasons to stick with your legacy environment for core applications and only use a Java-based environment for e-commerce and other web-based applications.

Decision Points

The tool that you select for a particular application is a decision that requires careful deliberation because the tool, or at least the language or environment, that you select will become integral to the system. If you make a mistake, or change your mind after development is complete, a major effort and cost will be required to modify the language or environment of the system. You need to consider many aspects of your situation, such as the intended audience (for example, web or client/server) and the talent available for development, deployment, and maintenance. Other factors that can influence your decision are the following:

- **The stability of the company behind the tool** - to help you predict whether support, bug fixes, and enhancements will be available in the future
- **Market direction** - to lead to an assurance that there will be development talent available
- **The stability of the code** - so that you are comfortable that the development tool will not require you to spend time finding and reporting bugs in the tool.

An additional factor to consider in this decision is the hardware and software environment that your target audience uses.

User Environment Considerations

Your application must support the equipment available to your target audience. For example, if you are deploying an Oracle Developer application into a client/server environment, you need to ensure that the users' PCs have enough memory and disk space for the runtime executables and DLLs.

You also need to be certain that the screen resolution for which you develop the application is supported. For example, you might develop an Oracle Developer application for a screen resolution of 1024x768 and develop the window and canvas sizes accordingly. If users are not able to set that resolution because of hardware limitations, they will need to use the window scrollbars in all screens in your application. This is not a desired effect and will frustrate users quickly. A post-development fix for this condition would require significant effort.

This consideration is not as important for HTML-based code such as JSP applications because the browser size can determine how the elements are laid out inside of it. Java client/server-based applications (Java applications and applets) can resize and reposition objects automatically if they make use of layout managers. Browser support is more important than screen resolution in a Java-based application. If your users may access your system with either Internet Explorer or Netscape, you need to be sure that all of the code that you write works in both browsers.

Systems Development Methodologies

Regardless of the tool or environment, creating systems always requires good software engineering practices. Therefore, at the core, the system lifecycle phases for a project are the same whether you use JDeveloper or Oracle Developer. Based on the system requirements, you need to select a sound methodology. You may want to use a method that includes analysis, design, and development with some variation on the traditional waterfall or iterative cycle. Alternatively, you may decide to use a rapid-application development methodology. As you progress through the analysis phase, you need to decide which tool would be best for the system that you want to create. After that decision is made, you can then target the design efforts at a specific environment.

One strategy you can use to minimize your risk of wanting to change products at a later time is to build "thin" applications. You can place all major code routines in the database and base your applications on complex views (frequently using INSTEAD OF triggers). In this way, 90% or more of your program logic is not embedded in a specific product and it is relatively easy to rewrite your application if you change your deployment platform in the future. This strategy also improves the logical organization and efficiency of your applications. Code that is more closely associated with the database resides there. This strategy can be taken to its logical conclusion by placing all business rules in a repository and driving very simple applications from the repository.

In addition to the language and tool decision, in the design phase you need to decide what your intended deployment technology will be. For example, in a Java environment, you can place your application into production in one of a number of formats—Java application, applet, or JSP application. (This author's paper "Creating Java Applications, Applets, and JSPs in JDeveloper" in the ODTUG 2001 proceedings contains some details on these formats). Oracle Developer gives you the option of deploying in client/server or on the Web using a Java applet. Your choice of a technology depends upon the same considerations of the application's audience (intranet or Internet) and the type of application (e-commerce or internal). This factor can be used to validate your tool decision or to help you make your tool decision in the first place.

The differences in system development appear for the first time, therefore, before the design phase. You identify the target audience and business requirements before proceeding. You then try to match those parameters with the tool's targeted audience and use.

Target Uses for JDeveloper

JDeveloper is intended to be the tool of choice for creating scalable, web-based systems with a wide range of audiences—intranet (and extranet) and Internet—and with any business purpose, including departmental applications, internal corporate applications, and e-commerce.

JDeveloper is primarily a Java-based development tool. Java applications naturally support all types of media files that may be presented on the Web. They can be relatively undemanding of the client machine. For example, if you deploy a JSP application, the only requirement on the client side is an HTML-aware browser.

JDeveloper is primarily a 3GL code-generation tool that requires some coding by hand but offers many wizards that assist greatly with this task. It requires a different kind of development skill (low-level Java programmer), and this requirement must be a key factor in deciding whether to use Java technology, in general, and JDeveloper, as a development tool.

JDeveloper can also be used to create client/server applications using Java. This is particularly important for shops that have programmers who are skilled in Java development and need a client/server–style application.

Another strength of the tool is its ability to easily create BC4J code that you can use to separate data access and business logic from the user interface. BC4J provides a common layer that different kinds of applications with different target users can access.

Target Uses for Oracle Developer

Oracle Developer is intended to be the tool of choice for creating scalable, web-based and client/server systems with an intranet audience and with specific business purposes: departmental applications and internal corporate applications. The primary example of the use of Oracle Developer is its largest user—Oracle ERP (prepackaged applications such as Financials and Manufacturing), although future releases could use JDeveloper.

Oracle Developer has evolved from a long line of development products that have transitioned from mainframe to client/server to web architectures. The modern release of Oracle Developer runs on the Web in a Java applet that closely emulates the look and feel of standard client/server applications. Developers who are accustomed to creating client/server applications will find that, other than some web-specific limitations and a different architecture, there is virtually no learning curve to deploy those same applications on the Web. Users who are accustomed to client/server applications will have less of a learning curve with an Oracle Developer application than with a JDeveloper JSP application (although JDeveloper can also create Java applications and applets that emulate client/server).

True to its heritage, Oracle Developer is also extremely strong in implementing client/server applications. The primary scripting language is PL/SQL, but much of the programming work is declarative in nature. Instead of writing lines of code, you work in a 4GL programming environment that uses definitions and declarations of objects. Oracle Developer also supports multimedia files, such as those found in all web applications, through extensions to the tool. Oracle Forms Developer applications running

on the Web can even incorporate JavaBeans created in JDeveloper. Due to its nature as a 4GL high-level tool, Developer can create prototype applications faster than JDeveloper and this makes it well suited to Rapid Application Development (RAD) methodologies.

Other Comparisons

To help you in the decision of which tool to select, you need more detailed information about the similarities and differences between the tools. This type of information will also be helpful if you have a background in Oracle Developer and need to supplement your knowledge with information about JDeveloper (one of the objectives of this paper). There are two categories of comparisons that will help you better understand JDeveloper and the differences between it and Oracle Developer:

- Programming environment
- Work in the tool

Comparing Programming Environments

JDeveloper creates applications within a Java environment. This environment is different from the "traditional" Oracle Developer environment because of the way that application projects employ files, the languages, and the kind of code that the tools create.

The Files

JDeveloper creates a number of different types of files for a specific application. Each Java class file usually has a fairly granular purpose although it can contain a number of methods and other objects; a single application screen can consist of many class files. In Oracle Developer, a single file (for example, a Forms .fmb file) can handle a number of purposes. Therefore, there are inherently more files in a Java project than in an Oracle Developer project. You can combine these files into a single unit—the package—that is deployed as a single unit, but the package is made up of a number of distinct files.

The Languages

The primary language that you use in JDeveloper, Java, is an object-oriented language. Oracle Developer provides some object-oriented concepts such as subclassing, but is more accurately termed an "object-based" or "object-friendly" tool. Object orientation requires a different way of thinking. Thinking in Java means that you need to think about objects. An object in Java is an instantiation of a class. (An *instantiation* is a programmatic declaration that creates something new. A *class* is a user-defined data structure that has attributes and methods and acts as a pattern from which the object is built.) Objects are created using programmatic declarations. In PL/SQL a user-defined datatype is implemented as a class in Java. PL/SQL variables are implemented as objects in Java. For example, the following code creates a text field object called "empnoTextField" from a class called JTextField.

```
JTextField  empnoTextField  = new JTextField();
```

Once the object is created in this way, you can assign values to its attributes (properties) using `set()` methods and call its other methods programmatically. If you want to see the object on the screen, you call an `add()` method on the container that will visually contain the object. The code to create and manipulate objects is always exposed. This gives you the ultimate in flexibility because you can code almost anything you want. However, as with any other tool, this flexibility comes with the price of responsibility. You are responsible for more of the low-level details in Java because they are available to you. JDeveloper eases this responsibility by supplying wizards that generate much of the basic code for you.

Oracle Developer and the PL/SQL language do not use this concept of wizards as heavily because they work at a higher level. You do not create most objects programmatically; instead, you create them in the Object Navigator or Layout Editor. Some objects such as timers, parameter lists, and record groups can be created, maintained, and deleted programmatically without being represented in the Object Navigator. The non-programmatic objects are visible in the Object Navigator, and you can assign property values to them at design time. You can also call built-in procedures to modify their property values programmatically, but you cannot create or destroy non-programmatic objects at runtime. The built-in procedures that modify property values (such as `set_item_property`) are like the `set()` methods for Java classes. Since you cannot break into the code that creates objects (it is in the runtime file but hidden from the designer interface), you do not have the responsibility for ensuring that all requirements of the object are met. That responsibility is handled under the covers by the tool.

JDeveloper generates the “add” and “set” code for you when you interact with its UI Designer. The code is available for you to modify and supplement. In fact, although JDeveloper generates completely working applications, it is likely that you will have to modify almost every file it creates to fit your application's requirements. Oracle Developer does not provide access to the code to create each object in the designer interface. When you create an object, there is nothing to modify other than its properties.

Both Oracle Developer and Java contain rich event models. When the user acts upon the screen objects, an event (called a *trigger* in Oracle Developer) occurs that can be handled by code that you write.

> **Note**
>
> Another effect that you will experience when starting to work in a Java environment is the shock of new terminology. There are lots of new terms, many of which are from the world of object orientation. To assist in your learning process, do not be afraid to ask questions when a new term is used.

The Code

JDeveloper creates Java-based code. The source code (.java) files are normal Java files that you can open and modify in any text editor. JDeveloper also creates files in a number of other languages and technologies such as XML, JSP, and HTML. All files are plain text files. Oracle Developer creates source code files (.fmb for Forms and .rdf for Reports) in tokenized binary formats that are proprietary to the tool. In both tools, you compile the files into runtime files used to deploy the application. In Oracle Developer, the runtime file (.fmx for Forms and .rep for Reports) is run by the proprietary Developer runtime engine. In Java, the runtime bytecode is a .class file that runs in a standard (nonproprietary) Java Virtual Machine (JVM) or other engine. XML, JSP, and HTML files are all run from their source code files since they are purely interpreted tag language files.

There are no standalone executables created by either tool. Both tools use some kind of runtime program to interpret the compiled runtime files. Oracle Developer uses a runtime program that corresponds to the JVM (or server-side JSP container) in Java.

Comparing Work in the Tools

One way to understand the differences and similarities between JDeveloper and Oracle Developer is to take a quick tour through their development work areas and describe how the development process is different in each. Both are full-featured IDEs and, therefore, support the basic operations of code development, compiling, debugging, and packaging. So that the following discussion can be reasonably brief—it will focus on the development of a Java application in JDeveloper and a form in the Oracle Forms Developer. Developing reports in Oracle Developer is a bit different, but the Oracle Reports Developer IDE has many similarities to the Forms IDE.

The first comparison to discuss is the IDE window. In JDeveloper, you can anchor windows to the outer frame so that they do not float within the frame. Oracle Developer presents all internal windows as separate windows without the ability to anchor the window to a particular side of the outer frame although you can maximize the window to fill the outer frame. There are additional comparisons possible in different areas inside the IDE window.

> **Note**
>
> Another difference between JDeveloper and Oracle Developer is their help systems. Since you have to work with lower-level code in JDeveloper, you need to know how to use the Javadoc documentation for class libraries that you are using in JDeveloper (such as Swing and AWT components). There is also a wealth of information about these non-Oracle class libraries on many Java developer web sites. As a Java programmer, you need to have this information at your fingertips. In Oracle Developer, the help system is relatively complete. All reference material for the built-in procedures is available directly in the Oracle Developer help system.

The Navigators

The JDeveloper and Oracle Developer navigators are similar in appearance, as Figure 1 shows. The objects displayed in each are arranged in a hierarchy. Oracle Developer organizes the objects so that the file type is the top-level node and the file is the next node. Objects can have child objects represented under them. JDeveloper uses the workspace as the top-level node. This workspace corresponds in Oracle Developer to a set of open files. It allows you easy access to a number of files by opening one file. There is no parallel to the single-file workspace concept in Developer. The Navigation Pane (top part of the window) shows project files and their child files. The workspace is also stored in a file, which means that the main objects represented in the Navigation Pane are files.

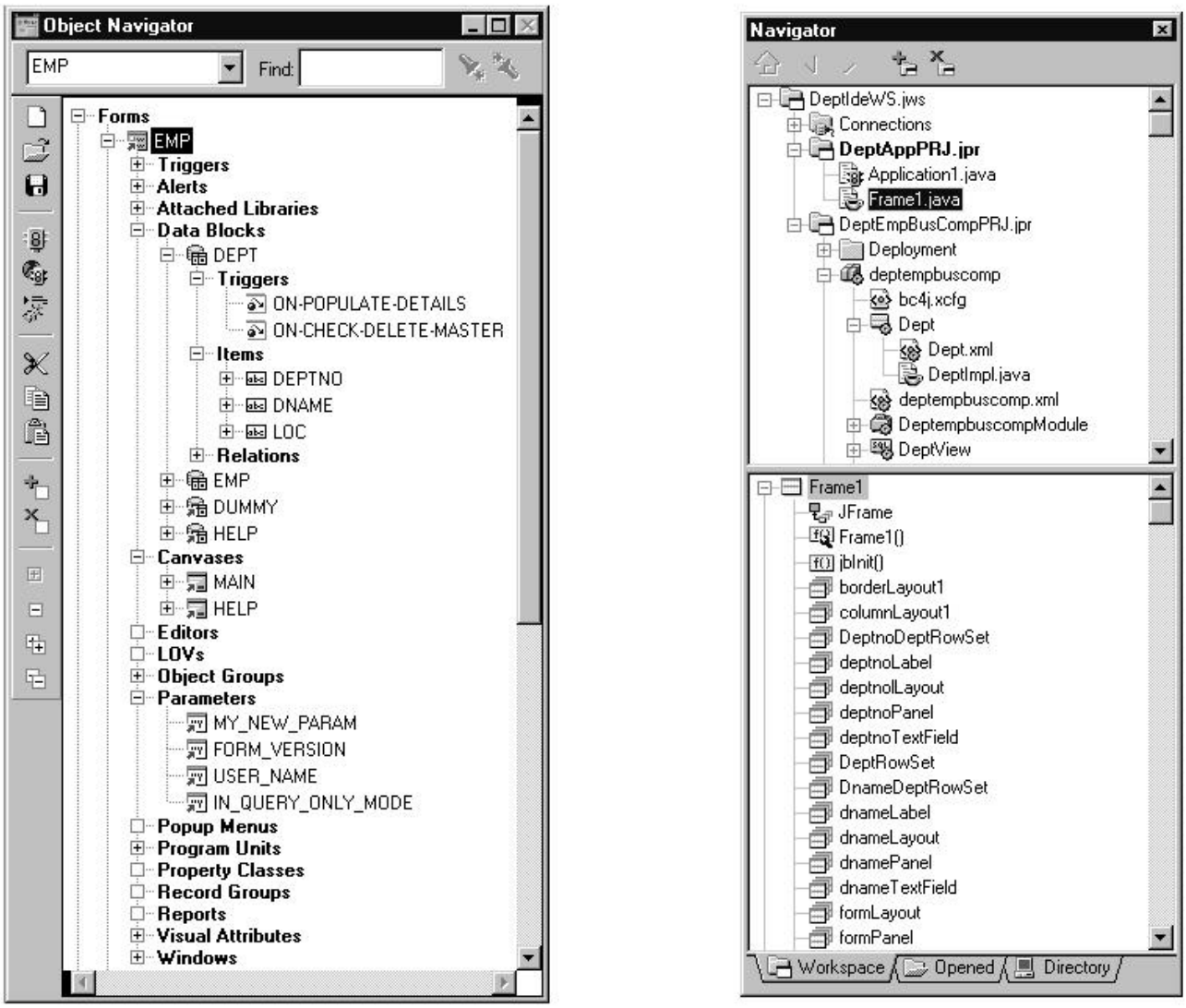
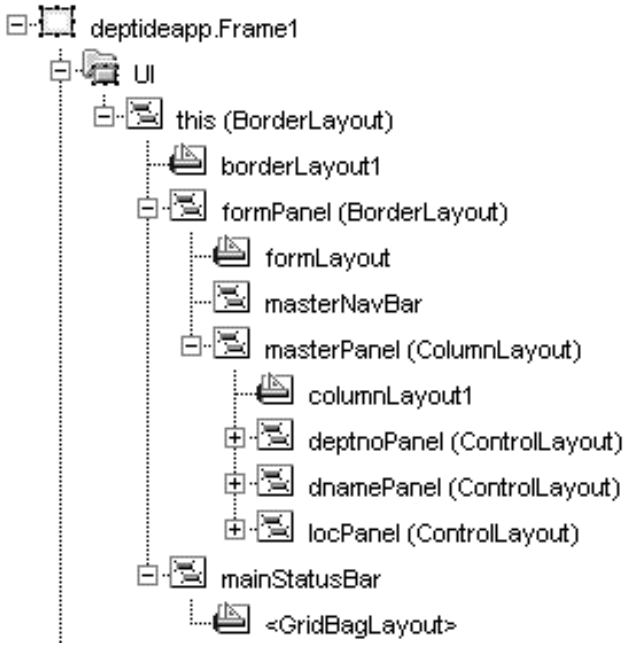


Figure 1. The navigators

The bottom part of the window, the Structure Pane, displays the objects inside each file. In summary, both navigators show files and objects in the files using a hierarchical tree structure. You can add objects to the application by dropping them into the Structure Pane. Both Oracle Developer and JDeveloper allow you to cut and paste objects in the navigators. JDeveloper does not allow the same drag-and-drop reordering of objects that Oracle Forms Developer allows, but you can use Cut and Paste to accomplish the same task in JDeveloper.

The Structure Pane changes to something like the following display when the Design tab of the Viewer window is active. This view represents the objects in a hierarchical structure so that you can see the relationship between objects.



In both tools, selecting an object in the navigator will select the object in the UI Designer and load the property inspector with the object's properties.

The JDeveloper Navigator also contains Opened and Directory tabs that have no equivalent in Oracle Developer. Both tools offer toolbar buttons within the navigator window to act upon objects in the window. The menu in both tools contains the same actions as the toolbar buttons. In addition, both tools offer toolbars under the menu bar for common main menu selections.

UI Designers

When you create a Java application or applet in JDeveloper, you can view the frame as it will be displayed at runtime using the UI Designer. You can also use this tool to add or modify objects on the frame. The changes that you make to the objects in this window will be represented by code in the Source Editor. The UI Designer is very similar to the Oracle Forms Developer Layout Editor. These tools are shown in Figures 2 and 3. There is als o a Menu Designer that corresponds to the Oracle Forms Developer Menu Editor.

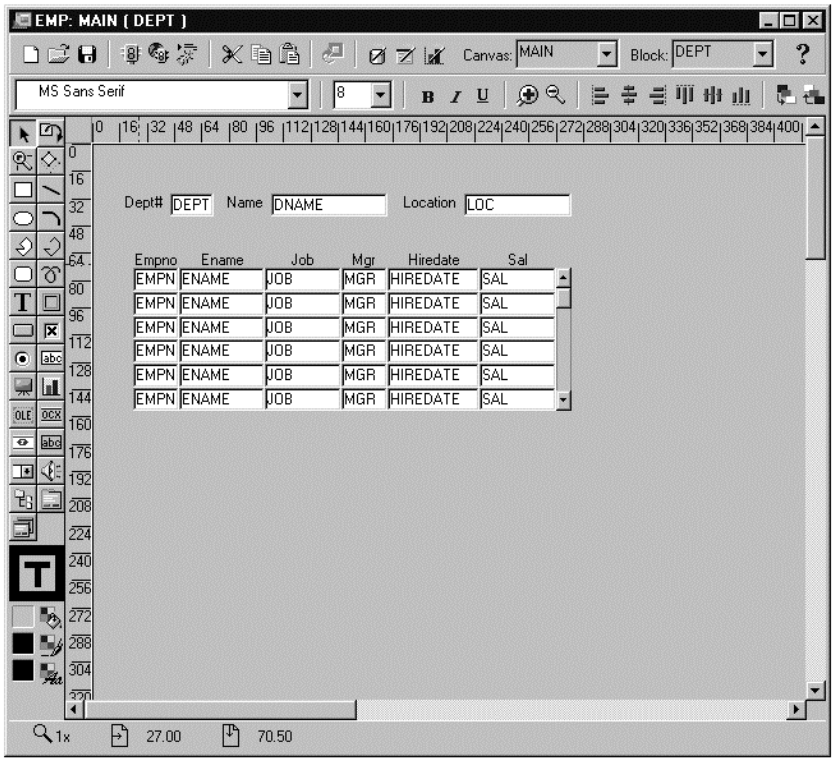


Figure 2. Oracle Forms Developer Layout Editor

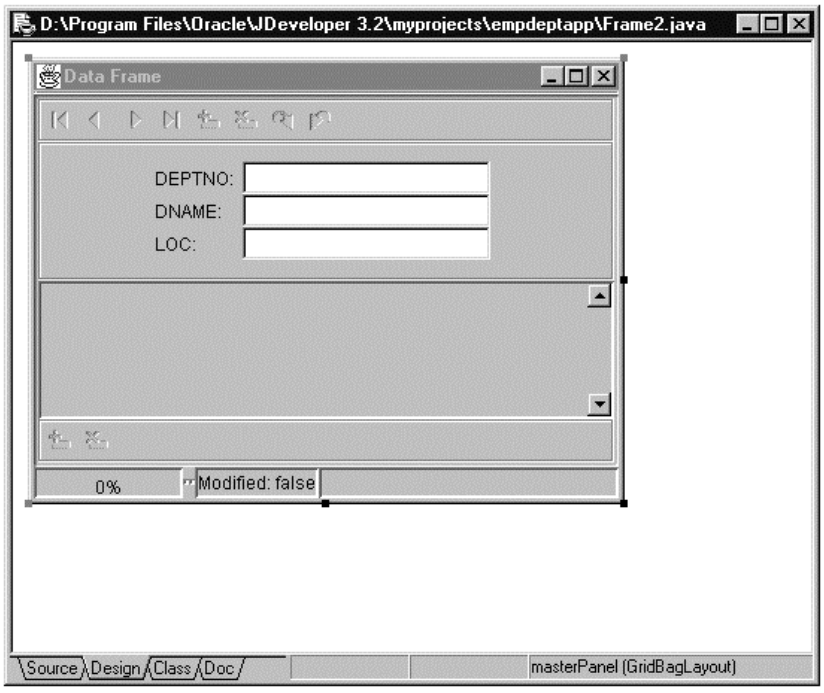


Figure 3. JDeveloper UI Designer Figure 3. JDeveloper UI Designer

One difference between the tools is in the dependency between the source code and layout editors. JDeveloper will only display either the UI Designer or the source code editor for a particular file. Oracle Developer can display both PL/SQL Editor and Layout Editor at the same time. Since changing an object in the JDeveloper UI Designer will change the code, you are essentially rewriting and editing the source code when you make a change to the layout. Similarly, when you make a change to the code, the UI Designer will reflect that change if it affects a visual object.

> **Note**
>
> JSP code does not use the UI Designer for emulating the runtime display. Therefore, most of your JDeveloper work on JSP applications is done using the Source Editor.

Source Code Editors

Both tools contain source code editors that represent keywords and comments in different colors. Figures 4 and 5 show the source code editors in both tools. Both tools provide assistance in filling in syntax (the Syntax Palette in Oracle Developer and Code Insight in JDeveloper). Both tools also allow you to open more than one source code window at the same time.

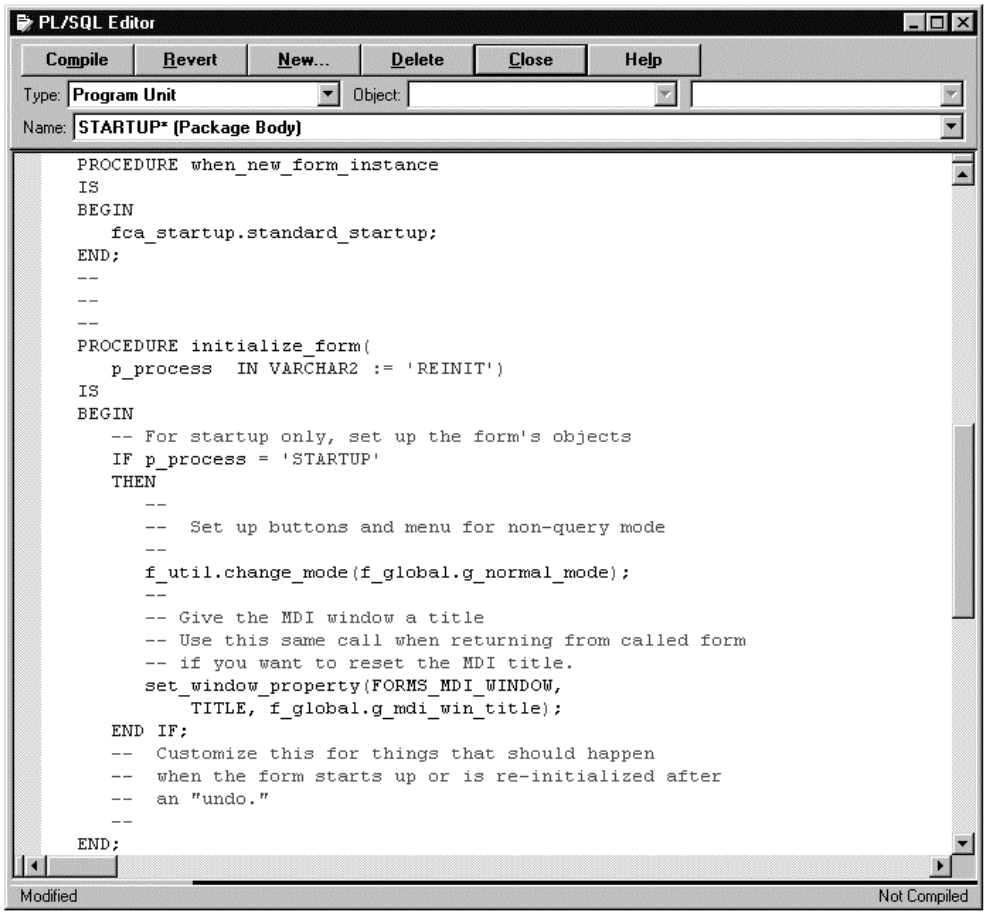


Figure 4. Oracle Forms Developer PL/SQL Editor

Figure 5. JDeveloper Source Code Editor

Property Inspectors

JDeveloper provides an Inspector window that allows you to set property values easily. The property values consist of characters that you type in, selections from a pulldown list, or additional dialogs that help you fill in the proper values. This is similar to the Property Palette in Oracle Developer. Figure 6 shows the property windows from both tools.

The JDeveloper Inspector provides an Events tab for adding code for specific events. This facility adds listener code and a code stub for the event that you click in the Inspector window. It also switches cursor focus to the code stub so that you can easily find it and fill in the required functionality.

There are some cosmetic differences in the windows of both tools. For example, the JDeveloper palette lists properties in alphabetical order, whereas Oracle Developer organizes the properties into groups with headings.

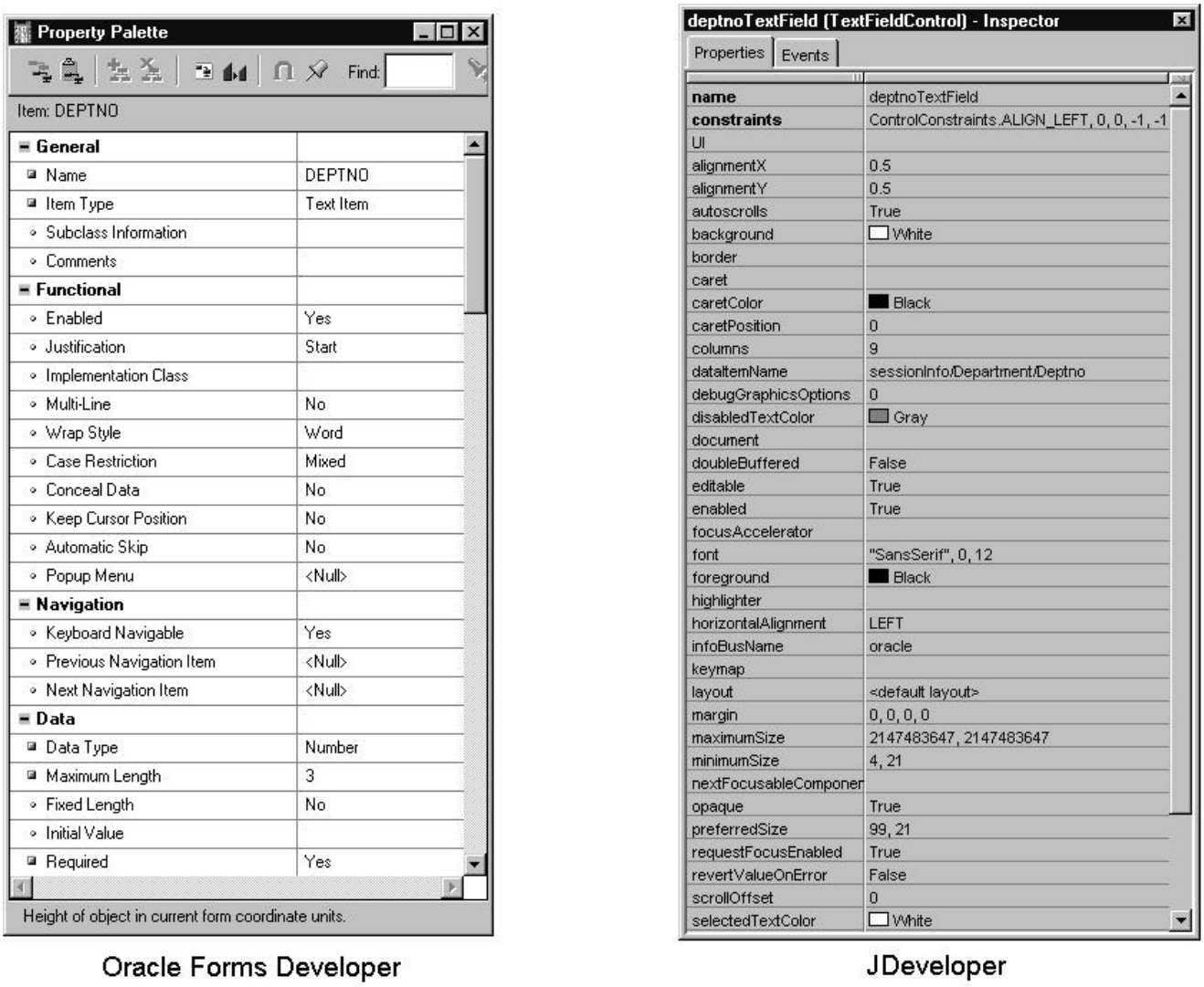
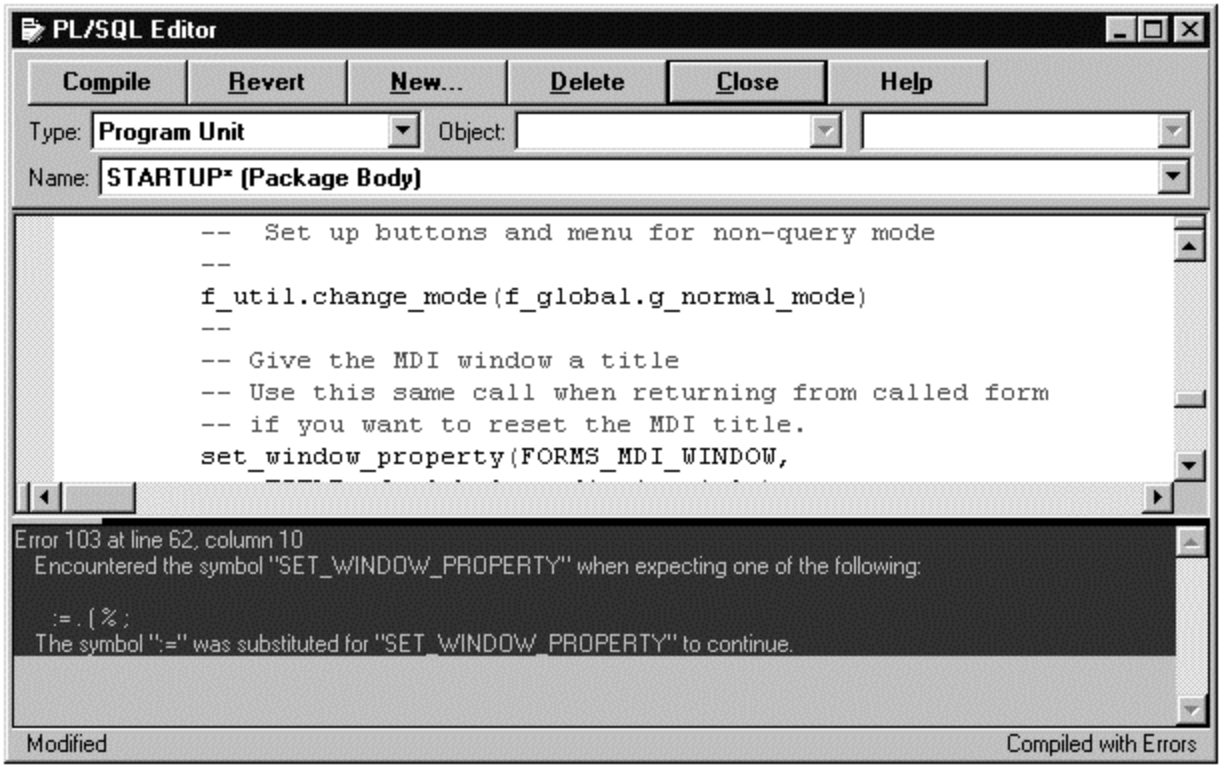


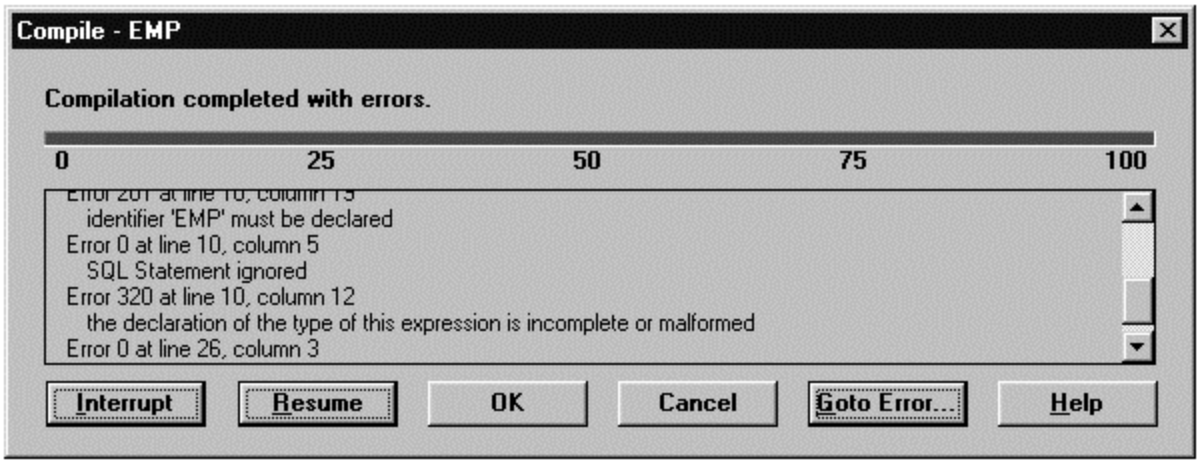
Figure 6. Property windows

Message Windows

Both tools offer message windows that display compilation and runtime status messages. Oracle Developer uses two areas—the bottom part of the PL/SQL Editor window that displays the results of a compilation problem in the program unit and a Compile window that shows the results of compiling all code in the form. These windows are shown in Figure 7.



PL/SQL Editor message pane



Compile window

Figure 7. Oracle Developer message areas

JDeveloper contains a similar message window that allows you to view compilation errors, as shown in Figure 8. This window also allows you to display messages from the code on a separate tab page. You can use this feature for simple debugging messages by inserting calls to `System.out.println()` at various points in the code, for example:

```
System.out.println("*** After pack.");
```

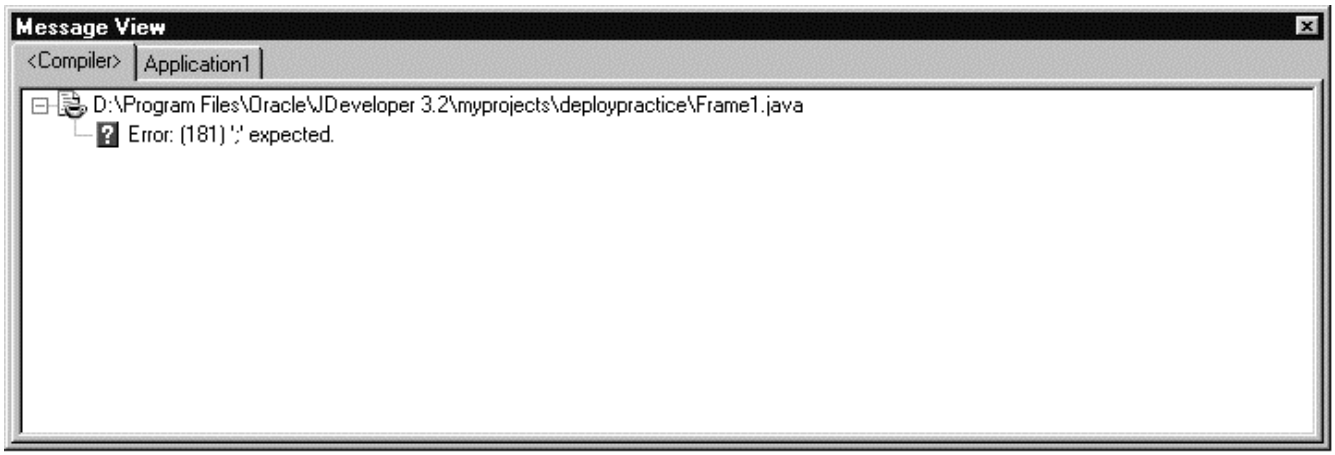
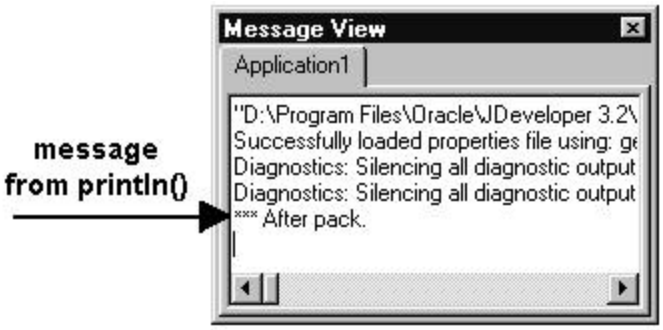


Figure 8. JDeveloper Message View

When you run the code from JDeveloper, the Message View will display the print statement when it reaches that line of code, as in the following illustration:



In addition to these message areas, both tools provide a status bar at the bottom of the main window that displays the file name or status of a compilation or save operation.

> **Note**
>
> In general, the debugger in JDeveloper can accomplish the same tasks as the debugger in Oracle Developer but contains features that are closer to other 3GL environment debuggers. A more detailed comparison of the two debuggers would not be useful because the development environments (3GL for JDeveloper and 4GL for Oracle Developer) are so different.

Wizards

JDeveloper contains a large number of wizards. There are wizards for creating a default BC4J project, a Java application, an applet, and a JSP file, as well as for lower-level objects such as frames, applications, web beans, and so on. The wizards are key to starting a particular task and are a true benefit of using JDeveloper because they create bug-free code that is as complete as possible. You will need to add application-specific logic and objects that cannot be handled by defaults. However, the wizards create a lot of code for you and allow you to concentrate on customizing the application for the business needs.

Since Oracle Developer is a higher-level development environment, it does not supply as many wizards. There is no need for a separate database access layer, so there is nothing like the Business Components Project Wizard. There are wizards in Oracle Developer for defining a data block and its source database objects (table, view, procedure, REF CURSOR, or SELECT statement). There are also wizards for block layout, LOV definition, reports, and charts. There are no wizards for lower-level objects because the work that you are doing is declarative in nature; it does not create lines of code for everything that is required in the application.

Figure 9 shows a corresponding wizard page for defining the data source of a single-table application in Oracle Developer and in JDeveloper.

Note

As you work with JDeveloper, you will find differences in the behavior of some controls. For example, when you create a multi-record block in Oracle Developer, you can modify any item type.You can change a text item to a poplist or checkbox control with one property setting. In JDeveloper, the grid control that emulates the multi-record block in Oracle Developer requires some nondefault code to modify the item types. (There is an example of this on the author's web site.) However, the grid control allows users to resize, hide, and display items at runtime—a feature that would require a significant amount of code in Oracle Developer.

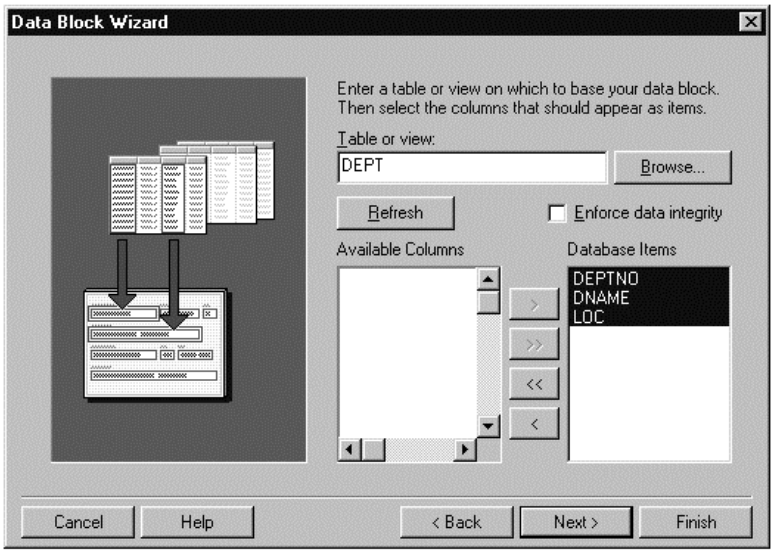
Development Method

As you would expect, the steps you need to follow in creating an application are different for each tool. The method you use in JDeveloper requires a specific order, as with any tool. Table 1 shows the major steps you would follow to create a Java application in JDeveloper. Java applications contain many of the same elements as applications that you create in Oracle Developer, and the table lists the corresponding steps that you would take in Oracle Developer. Other variations on Java code (such as applets or JSPs) would use some different steps.

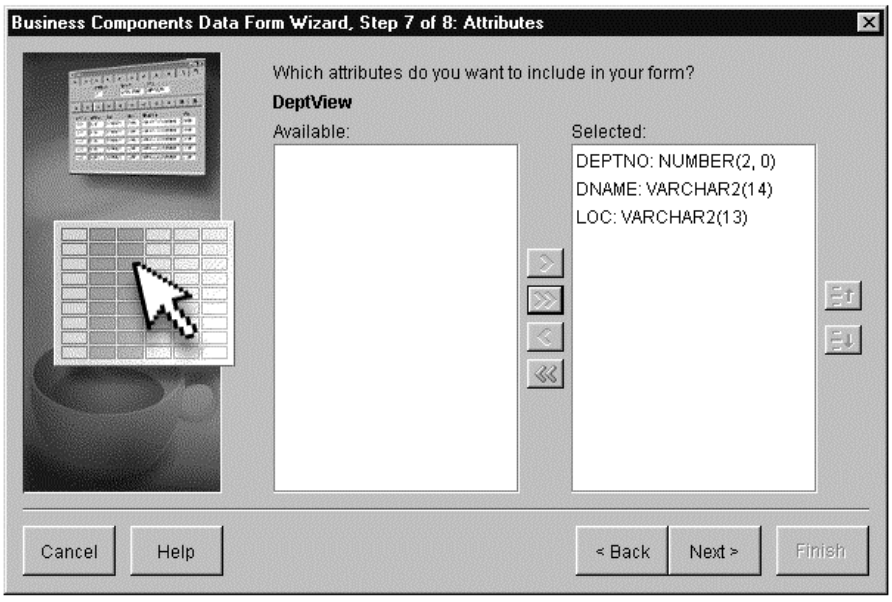
JDeveloper Steps	Corresponding Step in Oracle Developer
1. Define a connection object to access the database schema through JDBC.	Normal database grants and a Net8 connection.
2. Create a workspace to contain project files.	Create a file system directory to organize the source code files.
3. Create a data access project (BC4J) using the Business Components Project Wizard to set up the data layer.	(none)
4. Create a Java application using the Project Wizard and Business Components Data Form Wizard (or just the Application Wizard and Frame Wizard).	Create a form using a template and use the Data Block Wizard and Layout Wizard to define and lay out a block that connects to a database table or other data source.
5. Add and modify objects in the UI Designer and Menu Designer.	Add and modify objects in the Layout Editor and Menu Editor.
6. Modify object properties using the Inspector.	Modify object properties using the Property Palette.
7. Set other properties using code in the Source Editor.	(none)
8. Add event-handling code using the Source Editor.	Add trigger code using the PL/SQL Editor.

9. Add code using the Source Editor to enforce business rules and set object properties at runtime.	Add code using the PL/SQL Editor to enforce business rules and set object properties at runtime.
10. Compile, test, and debug the code using the IDE tools.	Compile, test, and debug the code using the IDE tools.
11. Create a deployment set with files to be copied to a production location.	Create a deployment set with files to be copied to a production location.

Table 1. Steps for creating a JDeveloper application compared to steps in Oracle Forms Developer



Oracle Forms Developer Data Block Wizard



JDeveloper Business Components Data Form Wizard

Figure 9. Defining a data source using a wizard

Note

You have many choices for the style of application you are creating in JDeveloper (for example, Java application, applet, or JSP). When learning JDeveloper, it is best to stick with the basics and create a sample Java application first. The Java application has many similarities with the applications that you create in Oracle Developer, so you will not have to make the transition step of learning a new architecture for deployment. After you feel comfortable with this style of application and with the way that JDeveloper works, you can branch out and explore the other styles.

About the Author

Peter Koletzke is a Technical Director and Principal Instructor for the Enterprise e.Commerce practice at Quovera (formerly Millennia Vision), in San Jose, California. Peter is Executive Vice President and Web Director for the IOUG-A and columnist for the *ODTUG Technical Journal*. He is a frequent speaker at various Oracle users group conferences where he has won awards such as Pinnacle Publishing's Technical Achievement, ODTUG Editor's Choice, and the ECO/SEOUC Oracle Designer Award. He is the co-author, with Dr. Paul Dorsey of the Oracle Press (Osborne McGraw-Hill) books: *Oracle JDeveloper 3 Handbook* (which supplied material for this paper), *Oracle Developer Advanced Forms and Reports*,*Oracle Designer Handbook, 2nd Edition*, and *Oracle Designer/2000 Handbook*. http://ourworld.compuserve.com/homepages/Peter_Koletzke

Quovera provides strategy, systems integration, and outsourced application management to Fortune 500, high-growth middle market and emerging market companies. The firm specializes in delivering intelligent solutions for complex enterprises, which improve productivity within the customer's business and optimize the customer's value chain, through integration of its customers and suppliers. The company also outsources the management of "best of breed" business applications on a recurring revenue basis. Quovera refers to its business model as "Intelligent – Application Integration and Management. http://www.quovera.com