Oracle JDeveloper 10*g*: Empowering J2EE Development

HARSHAD OAK

Apress™

ISBN (pbk): 1-59059-142-9

Printed and bound in the United States of America 12345678910

Technical Reviewer: Kenneth Cooper Jr.

Editorial Board: Steve Anglin, Dan Appleman, Gary Cornell, James Cox, Tony Davis, John Franklin, Chris Mills, Steven Rycroft, Dominic Shakeshaft, Julian Skinner, Martin Streicher, Jim Sumser, Karen Watterson, Gavin Wray, John Zukowski

Assistant Publisher: Grace Wong

Project Manager: Tracy Brown Collins

Copy Editor: Kim Wimpsett, Brian MacDonald

Production Manager: Kari Brooks

Production Editor: Janet Vail

Proofreaders: Elizabeth Barry and Patrick Vincent

Compositor: Kinetic Publishing Services, LLC

Indexer: Valerie Perry

Cover Designer: Kurt Krames

Manufacturing Manager: Tom Debolski

Distributed to the book trade in the United States by Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY, 10010 and outside the United States by Springer-Verlag GmbH & Co. KG, Tiergartenstr. 17, 69112 Heidelberg, Germany.

In the United States: phone 1-800-SPRINGER, email orders@springer-ny.com, or visit http://www.springer-ny.com. Outside the United States: fax +49 6221 345229, email orders@springer.de, or visit http://www.springer.de.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, email info@apress.com, or visit http://www.apress.com.

The source code for this book is available to readers at http://www.apress.com in the Downloads section. You will need to answer questions pertaining to this book in order to successfully download the code.

Contents at a Glance

CHAPTER 11

Tips and Tricks

In this chapter, you will look at tips and tricks that can be useful while using JDeveloper. These are solutions to some problems you might encounter as well as tricks to make you even more efficient at using JDeveloper. This chapter also presents some keyboard shortcuts.

The Tips

The following are tips to help you use JDeveloper more effectively.

Configuring the Web Browser and Proxy Server

The Web Browser/Proxy section in the Preferences dialog box (Tools ➤ Preferences) is where you can define the browser and proxy server that JDeveloper will use while executing Web applications. If you have access to the Internet only through a proxy server, you will need to configure the proxy server to use features such as Check for Updates or other Web Services features.

Speeding Up JDeveloper Startup: Extension Profiles

The minimum hardware requirements for running JDeveloper are quite high, and not everyone has the luxury of running high-end machines with loads of random access memory (RAM). So is there a way to speed up JDeveloper on startup?

The easiest way to speed up the JDeveloper startup on a sluggish machine involves making good use of the Extension Manager section in the Preferences dialog box (Tools ➤ Preferences). If you are developing a Java User Interface (UI) using Abstract Window Toolkit (AWT) and Swing, uncheck all extension categories and then select the Client Tier Development category. JDeveloper will automatically select the bare-minimum tools from the General category. Click the Save As button to save this profile for later use. Click OK, and restart JDeveloper. This will significantly speed up the startup time.

Furthermore, in the Applications Navigator, only load applications that you are actually using. The fewer the applications that need to load, the quicker the startup time.

Using Technology Scopes

Technology scope is a way of telling JDeveloper, "I am working with X, Y, and Z technologies. Only show me tools and features related to these technologies, and do not bother me with anything else." You can configure the technology scope for a project through the Common ➤Technology Scope section in Project properties. You can set the technology scope at the time of creating an application workspace by clicking the Edit Template button provided in the New Application Workspace window.

Refer to Chapter 3 for a discussion about technology scopes and application templates.

Refactoring Your Code

Refactoring is when you make changes to the code and improve its internal structure without changing how it works. However, even simple things such as renaming or moving a class can get quite tedious if done manually, because of the implications it might have for many other classes in the application. JDeveloper provides excellent refactoring capabilities so you can easily rename a class, move a class, and extract new methods by selecting that bit of code. These features are provided when you select the Tools ➤ Refactor command.

Installing ADF Runtimes

In Chapter 7, you developed applications using the Oracle Application Development Framework (ADF) and executed them on the embedded Oracle Application Server Containers for Java (OC4J) server. ADF applications can very well be deployed on any other server, such as Tomcat and WebLogic; however, for this you first need to install the ADF runtime on these servers. After selecting the Tools ➤ ADF Runtime Installer command, specify the location where Tomcat, WebLogic, or JBoss is installed. The wizard will then install the ADF Runtime Installer onto those servers.

Changing the JDeveloper Look and Feel

The JDeveloper look and feel is highly configurable. You can specify the look and feel for JDeveloper as a whole, and you can even specify the color to be used for something as specific as an arrow in a diagram.

To change the look and feel of JDeveloper, select one of the four choices provided in the Look and Feel drop-down list in the Environment section of the Preferences dialog box. You will need to restart JDeveloper for the new look and feel to be visible. All the figures in this book use the Windows look and feel.

Generating Project Javadoc with Diagrams

You can change the Javadoc configuration for a project from the Project Properties ➤ Development ➤ Javadoc screen. Apart from the normal Javadoc configuration, you can also specify if you want to document diagrams used in the project. So if you have a Web Service diagram, that diagram can be converted into PNG, JPG, and other formats and saved right into the Javadocs for the project.

This is a really useful feature because now all your project documentation can be maintained in the Javadocs, and you do not need to maintain separate project diagrams.

Editing Default Project Properties

Every time a new project is created, it inherits project properties from the default project properties. So if there is some property that you keep changing in every project created, you might as well change that in the default project properties. Once a change is made in the default project properties, every new project created afterward will have those properties. You can view and change the default project properties by choosing Project ➤ Default Project Properties from the menu bar.

Using Local/Hosted Documentation

If you rarely use the JDeveloper documentation or cannot spare the more than 70 megabytes that the documentation takes up, you have the option to not maintain local documentation but instead to use the documentation hosted on the Oracle Technology Network (OTN). The Documentation section in the Preferences dialog box provides this configuration.

Highlighting All Occurrences in Find

While using the Find tool to search for words in code or text, I more often than not use the Highlight All Occurrences option. So instead of checking that box every time I use Find, I just changed the Initial Find options in the Code Editor ➤ Find Options section in the Preferences dialog box to make the Highlight All Occurrences option checked by default.

Using Regular Expressions Find

The Find tool has a nice capability to use *regular expressions*. So if you want to search for the words *java* and *html* in a piece of text, just specify *java|html* as the text to search for and check the Regular Expressions check box. JDeveloper will now find all occurrences of both words.

Configuring External Tools

Although JDeveloper provides almost everything you require for Java development, you might have situations where you need to launch an external tool to get a task done. JDeveloper provides the capability to configure external tools that you can execute right from within JDeveloper. For example, if you want to edit all *.php files using your favorite PHP editor, you can configure that editor as an external tool using Tools ➤ External Tools. Not only can you define when and where the tool will be displayed, but you can even pass some arguments to the external tool.

Splitting the Code Editor

If you have two code files being displayed in two separate tabs in the Code Editor, you might have to constantly switch between these tabs. You could instead split the Code Editor in such a way that both files are displayed simultaneously. Simply click the tab for a file, and drag it to the left or right to split the editor window vertically. If you drag it to the bottom of the screen, the window will split horizontally.

Checking for Updates

You can easily install updates and extensions to JDeveloper from within JDeveloper. Select Help ➤ Check for Updates to start the Check for Updates Wizard. You will need an OTN account to use this feature. You can register at the OTN at http://otn.oracle.com/membership.

Changing Properties and the Default Package

You can use the Property Inspector to change some of the properties for a project. Select the project listed in the Applications Navigator, and you will notice that the Property Inspector now lists project properties—such as `defaultPackage`, `userAuthor`, `userCompany`, and so on—that can be modified easily.

Generating Accessors (Getter/Setter)

A common requirement is to generate getter and setter methods for any new fields you introduce. Instead of having to manually write those methods, you can select Code ➤ Generate Accessors, or you can right-click anywhere in the code and select Generate Accessors. In the window that pops up, select the fields for which you want the getter and setter methods to be generated.

Overriding Methods

You may often have a requirement where you need to change the method implementation provided in a superclass. To override a method in the super class it is important to get the method signature exactly the same as that stated in the superclass. Use the Tools ➤ Override Methods option, and you will get a window listing all the methods inherited from the superclass. Just select the method to be overridden, and the method code will be generated for you.

Organizing Imports

During development if you often tend to use the asterisk (*) character to import entire packages instead of importing specific classes, you can use the JDeveloper Organize Imports tool to automatically edit the code and import only classes that you actually use in the code. You can access the Organize Imports tool by using the Code menu or by right-clicking in the Code Editor. You have the following options: Sort Imports, Widen Imports, Narrow Imports, and Remove Unused Imports.

Managing Libraries

JDeveloper uses libraries to easily manage related Java archive (JAR) files and classes. So if you have a set of JAR files that you use often, it makes sense to create a new library containing these JAR files and then associate that library with various projects. For example, if you are developing an application using a framework, there might be three or four JAR files provided by the framework, all of which need to be used at the same time. In such a case you could create a new library holding all of the framework-related JARs and then associate this library with various projects.

Use Tools ➤ Manage Libraries to manage libraries, Java 2 Standard Edition (J2SE) definitions, and Java Server Page (JSP) tag libraries.

Toggling Line Numbers

Displaying line numbers for your code can make working with the code a little easier. To enable line numbers, right-click in the line gutter on the left of the Code Editor and select the option Toggle Line Numbers.

Surrounding With

JDeveloper provides a *Surround With* tool with which you can just select a block of code and insert a `for` loop, `while` loop, `try-catch-finally`, and so on around it. Select a piece of code, and then either select Code ➤ Surround With or select the Surround With option from the right-click menu.

Editing Component Palettes

The Component Palette is useful for easily inserting components into JSP pages, UIX pages, Swing UI, and so on. It is just as easy to insert new components into the palette as it is to edit existing components. Either select Tools ➤ Configure Palette or right-click in the Component Palette and select Properties to get the Configure Component Palette. Here you can easily add new components or edit existing ones.

Scanning Source Path to Determine Project Contents

The Project Properties dialog box provides a Scan Source Path to Determine Project Contents option. This feature is not enabled by default; however, I recommend enabling the feature because all files for a project are generally maintained in a separate directory, and if you have many people working on the project, these files might also keep changing constantly. It is easier to have JDeveloper scan and pick up new files rather than doing it manually.

Comparing Files for Differences

JDeveloper provides a Compare Files tool that can be useful if you are not using a source-control tool and yet have more than one person working on the same file. Select Tools ➤ Compare Files, and provide the two files you want to compare. JDeveloper will compare the two files and highlight all differences between them.

Using Code Templates

You looked at code templates in Chapter 3. However, I cannot emphasize using and creating code templates enough. Because a lot of Java coding is repetitive in nature, code templates can save you a lot of time. You can check out the code templates that come predefined as well as define new templates in the Code Editor ➤ Code Templates section in the Preferences window. Also use the Imports tab in this dialog box to import the classes that you are using as part of the template.

JDeveloper Help

JDeveloper's help has four sections:

- **Table of Contents**: This displays a list of all the help books that are provided. You can expand each book listing to view the subsections of the book.
- **Index Search**: The Index Search section is where only the index is searched and not the contents of all the books.
- **Full Text Search**: The entire text is searched.
- **Glossary Search**: This search is useful if you are looking for definitions of words such as *instantiate*, *relational database*, and so on.

Using Run Manager

You can view the Run Manager by selecting View ➤ Run Manager. This window helps you know which processes are running and terminate processes if they hang or misbehave.

Navigating Code Using Bookmarks

Bookmarks mark locations in the code to which you can easily navigate. If you have 1,000 lines of code in one file, it makes sense to have bookmarks in place that will help you quickly move to specific code with which you need to work. To set a bookmark, either right-click in the line gutter and select the Toggle Bookmark option or move the cursor to the line to be bookmarked and press Ctrl+K. Once

you have set the bookmarks, you can use the Search ➤ Bookmarks menu command to easily move through the bookmarks.

You will now look at keyboard shortcuts that will make working with JDeveloper easier.

Keyboard Shortcuts

How comfortable developers are with a tool certainly depends on how easily the developer can get things done using just the keyboard and not having to touch the mouse. As interactive and useful as a mouse can get, it certainly cannot match the speed that keyboard shortcuts can give you.

For keyboard usage, JDeveloper gives you the option of using seven different key mappings. In the Accelerators section in the Preferences dialog box, click the Load Preset button to see a list of the key mapping alternatives.

Table 11-1 lists useful shortcuts for the default key mapping.

Table 11-1. Default Keyboard Shortcuts

Description	Keys
Go to matching brace	Alt+] and Alt+[
Completion insight	Ctrl+spacebar
Parameter insight	Ctrl+Shift+spacebar
Delete next word start	Ctrl+T
Delete previous	Shift+Backspace
Delete until end of line	Ctrl+Shift+Y
Run project	F11
Debug project	Shift+F9
Make project	Ctrl+F9
Rebuild project	Alt+F9
Make selected class	Ctrl+M
Find in Navigator	Alt+Home
File list	Alt+0
Tab size 2	Ctrl+2
Tab size 4	Ctrl+4
Tab size 8	Ctrl+8

Table 11-1. Default Keyboard Shortcuts (continued)

Description	Keys
Close window	Ctrl+W and Ctrl+F4
Find	Ctrl+F
Replace	Ctrl+R
Go to line	Ctrl+G
Find in files	Ctrl+Shift+F
Go to Java class	Ctrl+Minus
Toggle bookmark	Ctrl+K
Go to bookmark	Ctrl+Shift+K
Go to next bookmark	Ctrl+Q
Go to previous bookmark	Ctrl+Shift+Q
Expand template	Ctrl+Enter
Toggle line comments	Ctrl+/
Next editor frame	Ctrl+Tab
Frame on right	Alt+Right
Frame on left	Alt+Left
Step over	F8
Step into	F7
Step out	Shift+F7
Resume	F9