Professional Java™ Native Interfaces with SWT/JFace

Jackwind Li Guojie

Wiley Publishing, Inc.

Professional Java™ Native Interfaces with SWT/JFace

Jackwind Li Guojie

Wiley Publishing, Inc.

Professional Java™ Native Interfaces with SWT/JFace

Published simultaneously in Canada

2004018715

ISBN: 0-470-09459-1

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

About the Author

Jack Li Guojie is an independent Java developer who has been building various types of Java applications since 1998. His areas of interest and experience include artificial intelligence, user interfacing, Web applications, and enterprise system architecture. He has contributed articles to many leading software journals.

I thank John Wiley & Sons, the publisher, for its trust in me and investment in this book. In particular, many thanks to Gaynor Redvers-Mutton, my acquisitions editor, who encouraged me to write this book. The book could never have been completed without my development editors, Emilie Herman and Kenyon Brown, who corrected my English and reshaped the manuscript into a readable form. I also thank Roy Miller, the technical reviewer, for his invaluable comments.

Credits

Acquisitions Editor
Gaynor Redvers-Mutton

Development Editor
Emilie Herman

Technical Editor
Roy Miller

Copy Editor
Nancy Rapoport

Editorial Manager
Kathryn A. Malm

Vice President & Executive Group Publisher
Richard Swadley

Vice President and Publisher
Joseph B. Wikert

Project Coordinator
Bill Ramsey

Graphics and Production Specialists
Beth Brooks
Kelly Emkow
Carrie Foster
Lauren Goddard
Jennifer Heleine

Quality Control Technicians
Susan Moritz
Brian H. Walls

Proofreader
Susan Sims

Indexer
Joan Griffitts

Contents

Introduction

Eclipse is an open source universal tool platform, dedicated to providing a robust, full-featured industry platform for the development of highly integrated tools. With millions of downloads, Eclipse becomes more and more popular. One of the most important common facilities provided by the Eclipse framework is the portable native widget user interface called the Standard Widget Toolkit (SWT), which provides a set of OS-independent APIs for widgets and graphics. SWT is analogous to AWT and Swing except SWT uses a rich set of native widgets. Built on SWT, JFace is a user interface toolkit handling many common UI programming tasks. JFace is designed to work with SWT without hiding it. Some of the advantages SWT/JFace offers over Swing include support for native features, fast execution speed, and flexible programming models.

This book teaches you how to build practical user interfaces with SWT/JFace. After introducing each widget, I present a great deal of Java source code to show you how to use the widget effectively. You can use the sample code as the basis to develop real-world applications quickly. Additionally, many techniques and tips are presented to help you save time. Finally, the last chapter shows you how to build an FTP client by combining everything covered in the book.

The comprehensive coverage of the SWT/JFace framework also makes this book an ideal reference.

Who Should Read This Book

This book is targeted primarily at Java user interface developers, Eclipse enthusiasts, and technical managers. The first few chapters help nontechnical people gain insight into the SWT/JFace framework. The later chapters contain a lot of technical details and practical examples that Java developers should find of great use.

In order to understand the code samples in this book, you need to have a good knowledge of the Java programming language. Background on user interface development is an advantage but not a necessity.

Those who have some experience with SWT/JFace programming can skip the first three chapters and jump right to Chapter 4. Others should read from start to finish.

What This Book Covers

This book covers the latest SWT/JFace version 3.0, which was released in June 2004.

How This Book Is Organized

This book is organized into 24 chapters. The first few chapters introduce the SWT/JFace framework and cover some SWT/JFace fundamentals such as event handling, layout, and the like. The next few chapters discuss each SWT widget individually and give practical advice on the usage of each one. After the introduction of all the SWT widgets, topics such as JFace dialogs and wizards are covered. Finally, the book covers special topics such as OLE support and Draw2D and concludes with the development of an FTP client application.

Note that while introducing some SWT widgets, I bring in some related JFace model-view-controller (MVC)–based components. For example, when discussing the SWT table widget, I cover the JFace table viewer. In this way, you learn two different approaches to achieve the same result—you can either use the traditional approach by manipulating the table widget directly, or you can take the MVC approach with the table viewer. You can compare the two approaches in order to choose the best one for you.

Part I: Fundamentals

Part I introduces you to the fundamentals of SWT/JFace.

Chapter 1 offers you a tour of Java GUI frameworks. Toolkits such as AWT, Swing, and SWT/JFace are discussed and compared. The chapter covers the features of SWT/JFace and compares them with those from other toolkits.

Chapter 2 explains some of the mechanisms used by SWT/JFace. First, the chapter introduces the implementation of SWT. The rest of the chapter is about resource management in SWT; here you can find useful resource management techniques and practical tips. The last part explains how the model-view-controller (MVC) design fits in JFace.

Chapter 3 covers setting up your IDEs to develop applications with SWT/JFace, writing your first SWT programs, using JFace to simplify UI programming, and deploying your applications to multiple platforms using Java Web Start.

Chapter 4 introduces SWT event handling and the threading mechanism. The `Display` class, which plays the most important role in SWT event handling, is introduced, too.

Chapter 5 first provides a tour of the hierarchy of the SWT widgets. Several typical widget types are discussed. After that, you learn about three kinds of basic SWT widgets—Shells, Buttons, and Labels. Additionally, this chapter covers focus traversal.

Chapter 6 shows you how to use layouts to manage the position and size of children in composites. The chapter introduces you to four standard layouts and one custom layout provided in SWT: `FillLayout`, `RowLayout`, `GridLayout`, `FormLayout`, and `StackLayout`. At the end of this chapter, you learn how to create your own layouts.

Part II: Design Basics

Part II introduces you to the basics of designing layouts in SWT/JFace.

Chapter 7 introduces two kinds of SWT controls: `Combos` and `Lists`. Both combos and lists allow the user to choose items from a list of items. Additionally, `ListViewer`, an MVC viewer based on the `List` control, is introduced.

Chapter 8 introduces two kinds of SWT text controls: `Text` and `StyledText`.

Chapter 9 teaches you how to use menus, toolbars, and cool bars in SWT. Additionally, you learn how to use the JFace action framework to simplify the task of creating menus and toolbars.

Chapter 10 shows you how to use the SWT `Table` control to display, navigate, and edit data. Additionally, the JFace `TableViewer` is introduced to help you simplify these tasks by taking advantage of MVC programming.

Chapter 11 shows you how to use the SWT `Tree` control to display and edit a hierarchy of items. Event handling of trees is also introduced. Additionally, you see how to use `TreeViewers` and the MVC approach to program with trees.

Chapter 12 covers UI objects that can be used to acquire particular types of data input from the user. In this chapter, you learn how to use each of the SWT dialogs: `ColorDialog`, `DirectoryDialog`, `FileDialog`, `FontDialog`, and `MessageBox`. Additionally, this chapter guides you to create your own dialogs.

Part III: Dynamic Controls

Part III introduces you to adding dynamic controls in SWT/JFace.

Chapter 13 teaches you how to use controls that can be used to present numerical values. The controls include `Scale`, `Slider`, and `ProgressBar`.

Chapter 14 introduces several miscellaneous SWT components: `Group`, `Sash`, `SashForm`, `TabFolder`, and `Browser`.

Chapter 15 discusses topics concerning graphics and image handling. The first part of this chapter teaches you how to perform various drawing operations with graphics context—drawing lines, arcs, shapes, images, and text and filling shapes, and so forth. SWT image handling is introduced in the second part. You learn how an image is represented in SWT. Additionally, the chapter introduces practical image manipulation techniques.

Chapter 16 introduces various ways to transfer data within an application and between applications easily. I show you how to enable your applications to supply data and to accept data in the drag-and-drop process. Finally, you learn how to use the clipboard to exchange data within an application or between different applications.

Chapter 17 shows you how to add the printing functionality to your existing programs. This chapter first introduces you to the basic printing mechanism. A real-world example is then used to guide you step by step to code for printing and print preview. Finally, you learn about multiple page printing and pagination.

Chapter 18 shows you how to use the JFace windows framework (`org.eclipse.jface.window`) to simplify windows creation and management tasks. Additionally, this chapter covers JFace dialogs.

Part IV: Application Development

Part IV takes you through the steps to create a sample application in SWT/JFace.

Chapter 19 introduces you to the JFace wizard framework with a sample application.

Chapter 20 gives you a brief overview of the JFace text framework. Then it shows you how to create a basic custom text editor with JFace text. The custom text editor is then improved by adding the following add-ons: content assist and syntax highlighting.

Chapter 21 provides a framework for creating flat, web-like user interfaces. This chapter shows you how to use the Eclipse Forms frame. You learn how to use a toolkit to create basic forms or scrollable forms. Eclipse Form custom widgets are then introduced, such as hyperlinks, form texts, sections, and so on.

Chapter 22 teaches you how to embed OLE documents and ActiveX controls in SWT widgets on Windows platforms. As an example, a Microsoft Word document is embedded in an SWT application. The chapter walks you through the steps to embed the OLE document: creating the OLE container, creating an OLE site for the OLE document, activating the OLE object, and deactivating the OLE object.

Chapter 23 introduces you to a lightweight rendering framework—Draw2D. With Draw2D, you can create complex figures easily. This chapter shows you how to create simple UML diagrams with Draw2D. The sample application displays the selected class in a UML diagram. By combining small figures, you can create manageable complex figures without tedious code.

Chapter 24 guides you through the development of a simple FTP client application using SWT/JFace. By combining knowledge acquired in previous chapters, you can create complex practical applications. With the FTP client sample application, you learn how to use application windows, actions, menu bars, and toolbars. Furthermore, you learn how to make main UI components resizable by using sash forms properly. You can use drag and drop to improve the user interface and make it more accessible to the user.

What You Need to Use This Book

In order to run the sample code, you need to download and install Eclipse version 3.0, which is available online at `www.eclipse.org/`.

Conventions

To help you get the most from the book and keep track of what's happening, I've used a number of conventions throughout.

> **Boxes like this one hold important, not-to-be forgotten information that is directly relevant to the surrounding text.**

Tips, hints, tricks, and asides to the current discussion are offset and placed in italics like this.

As for styles in the text:

- I *highlight* important words when I introduce them.
- I show keyboard strokes like this: *Ctrl+A*.
- I show file names, URLs, and code within the text like so: `persistence.properties`.
- I present code in two different ways:

```
In code examples I highlight new and important code with a gray background.
```

```
The gray highlighting is not used for code that's less important in the present
context, or has been shown before.
```

Source Code

As you work through the examples in this book, you may choose either to type in all the code manually or to use the source code files that accompany the book. All of the source code used in this book is available for download at `www.wrox.com`. Once at the site, simply locate the book's title (either by using the Search box or by using one of the title lists) and click the Download Code link on the book's detail page to obtain all the source code for the book.

When you download the code, just decompress it with your favorite compression tool. Alternately, you can go to the main Wrox code download page at `www.wrox.com/dynamic/books/download.aspx` to see the code available for this book and all other Wrox books.

Updates (Errata)

I've made every effort to ensure that there are no errors in the text or in the code. However, no one is perfect, and mistakes do occur. If you find an error, such as a spelling mistake or faulty piece of code, I would be very grateful for your feedback. By sending in errata you may save another reader hours of frustration and at the same time you will be helping to provide even higher quality information.

To find the errata page for this book, go to `www.wrox.com` and locate the title using the Search box or one of the title lists. Then, on the book details page, click the Book Errata link. On this page you can view all errata that has been submitted for this book and posted by Wrox editors. A complete book list including links to each book's errata is also available at `www.wrox.com/misc-pages/booklist.shtml`.

If you don't spot "your" error on the Book Errata page, go to `www.wrox.com/contact/techsupport.shtml` and complete the form there to send the error you have found. I'll check the information and, if appropriate, post a message to the book's errata page and fix the problem in subsequent editions of the book.

p2p.wrox.com

For author and peer discussion, join the P2P forums at `p2p.wrox.com`. The forums are a Web-based system for you to post messages relating to Wrox books and related technologies and interact with other readers and technology users. The forums offer a subscription feature to e-mail you topics of your choosing when new posts are made to the forums. Wrox authors, editors, other industry experts, and your fellow readers are present on these forums.

At `http://p2p.wrox.com` you will find a number of different forums that will help you not only as you read this book, but also as you develop your own applications. To join the forums, just follow these steps:

1. Go to `p2p.wrox.com` and click the Register link.
2. Read the terms of use and click Agree.
3. Complete the required information to join as well as any optional information you wish to provide and click Submit.
4. You will receive an e-mail with information describing how to verify your account and complete the joining process.

You can read messages in the forums without joining P2P but in order to post your own messages, you must join.

When you join, you can post new messages and respond to messages other users post. You can read messages at any time on the Web. If you would like to have new messages from a particular forum e-mailed to you, click the Subscribe to this Forum icon by the forum name in the forum listing.

For more information about how to use the Wrox P2P forum, be sure to read the P2P FAQs for answers to questions about how the forum software works as well as many common questions specific to P2P and Wrox books. To read the FAQs, click the FAQ link on any P2P page.

Part I: Fundamentals

Chapter 1: Overview of Java UI Toolkits and SWT/JFace

Chapter 2: SWT/JFace Mechanisms

Chapter 3: Jump Start with SWT/JFace

Chapter 4: SWT Event Handling, Threading, and Displays

Chapter 5: Basic SWT Widgets

Chapter 6: Layouts

1

Overview of Java UI Toolkits and SWT/JFace

This chapter outlines the three main Java user interface (UI) toolkits: AWT, Swing, and JFace. First I provide a brief introduction to all three, and then I compare them, highlighting some of the advantages SWT/JFace offers. SWT/JFace allows you to access native features easily, and programs based on SWT/JFace are considerably faster than those based on Swing in terms of execution speed. SWT/JFace is designed to be very flexible, so you can program using either the traditional approach or the model-view-controller approach. After reading this chapter, you should have a general overview of SWT/JFace. The chapters that follow introduce various aspects of SWT/JFace in detail.

Evolution of Java GUI Frameworks

This section covers the following Java graphical user interface (GUI) frameworks:

- **Abstract Window Toolkit (AWT):** The first and the simplest windowing framework.
- **Swing:** Built on AWT, Swing offers peerless components.
- **Standard Widget Toolkit (SWT) and JFace:** SWT is a native widget UI toolkit that provides a set of OS-independent APIs for widgets and graphics. JFace is a UI toolkit implementation using SWT to handle many common UI programming tasks.

This section outlines the evolution of the Java GUI framework and highlights the key features we'll compare and contrast in the next section.

Abstract Window Toolkit

The first version of Java, released by Sun Microsystems in 1995, enabled you to create programs on one platform and deliver the products to other Java-supported systems without worrying about the local environment — "Write Once, Run Anywhere." Most early Java programs were fancy animation applets running in Web browsers. The underlying windowing system supporting those applets was the Abstract Window Toolkit (AWT).

AWT has a very simple architecture. Components, graphics primitives, and events are simply perched on top of similar elements from the underlying native toolkit. A layer of *impedance matching* sits between the AWT and various underlying native toolkits (such as X11, Macintosh, and Microsoft Windows) to ensure the portability of AWT.

AWT 1.0 uses a callback delegation event model. Events are propagated or delegated from an event "source" to an event "listener." The interested objects may deal with the event, and the super-event handler is not required. The event model in AWT 1.1 was reimplemented from the callback delegation event model to an event subscription model. In AWT 1.1, the interested objects must register themselves with the components to receive notification on certain events. When the events are fired, event object are passed to registered event listeners.

AWT was slightly enhanced in later releases of Java. However, even the latest version of AWT fails to delivery a rich set of GUI components. Following is a list of components provided by AWT:

- `Button`
- `Canvas`
- `Checkbox`
- `Choice`
- `Container`
  - `Panel`
  - `ScrollPane`
  - `Window`
- `Label`
- `List`
- `Scrollbar`
- `TextComponent`
  - `TextArea`
  - `TextField`

To give you a more complete overview of the AWT user interface, I've created a simple GUI program. This tiny program allows the user to upload a photo to a server, or anywhere else. Figure 1-1 shows the user interface of the photo uploader implemented using Abstract Window Toolkit.

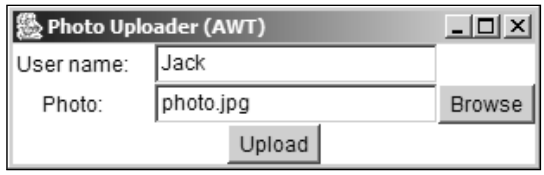

Figure 1-1

Click the Browse button to bring up the file selection dialog (see Figure 1-2). The name of the selected file is inserted into the text after the Photo label. The upload process starts when the user clicks the Upload button. The program exits when uploading is complete.

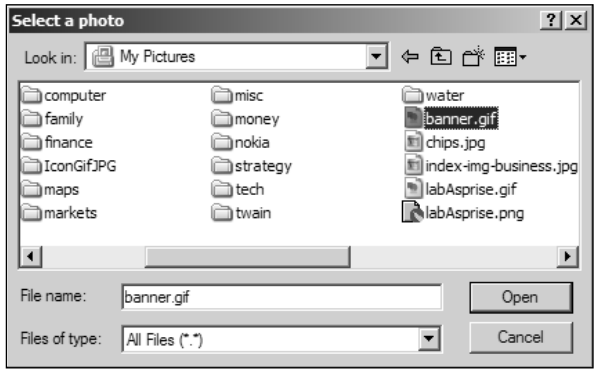

Figure 1-2

If you are familiar with Microsoft Windows systems, you may notice that the file selection dialog in Figure 1-2 is exactly the same as those used by native Windows programs. The Abstract Window Toolkit passes the call for file selection to the underlying native toolkit, i.e., Windows toolkit, and as a result, a native Windows file selection dialog pops up.

The Abstract Window Toolkit is sufficient for developing small user interfaces and decorations for Java applets, but it's not suitable for creating full-fledged user interfaces. Sun Microsystems recognized this as well and in 1997, JavaSoft announced Java Foundation Classes (JFC). JFCs consist of five major parts: AWT, Swing, Accessibility, Java 2D, and Drag and Drop. Swing helps developers to create full-scale Java user interfaces.

Swing

Swing is a pure Java UI toolkit built on top of the core Abstract Window Toolkit (AWT) libraries. However, the components available in Swing are significantly different from those in AWT in terms of underlying implementation. The high-level components in Swing are lightweight and peerless, i.e. they do not depend on native peers to render themselves. Most AWT components have their counterparts in Swing with the prefix "J." Swing has twice the number of components of AWT. Advanced components such as trees and tables are included. The event-handling mechanism of Swing is almost the same as that of AWT 1.1, although Swing defines many more events. Swing has been included in every version of Java since Java 1.2.

The main Swing packages are as follows:

- **javax.swing:** Contains core Swing components.
- **javax.swing.border:** Provides a set of class and interfaces for drawing various borders for Swing components.
- **javax.swing.event:** Contains event classes and corresponding event listeners for events fired by Swing components, in addition to those events in the java.awt.event package.
- **javax.swing.plaf:** Provides Swing's pluggable look-and-feel support.
- **javax.swing.table:** Provides classes and interfaces for dealing with JTable, which is Swing's tabular view for constructing user interfaces for tabular data structures.
- **javax.swing.text:** Provides classes and interfaces that deal with editable and noneditable text components, such as text fields and text areas. Some of the features provided by this package include selection, highlighting, editing, style, and key mapping.
- **javax.swing.tree:** Provides classes for dealing with JTree.
- **javax.swing.undo:** Provides support for undo and redo features.

In addition to the lightweight high-level components, Swing introduced many other features over AWT. Pluggable look-and-feel is one of the most exciting of the bunch. Swing can emulate several look-and-feels, and you can switch the look-and-feels at runtime. If you do not like any of them , you can even create your own. Other features include tooltip support, keyboard event binding, and additional debugging support.

The photo uploader program can be rewritten using Swing. Figure 1-3 shows the user interface of the Swing photo uploader with Windows look-and-feel; Figure 1-4 shows the user interface with Java metal look-and-feel.

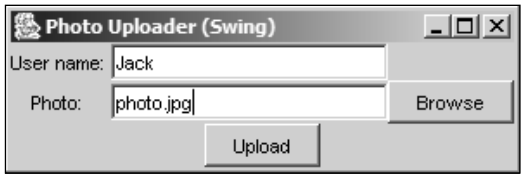

Figure 1-3

Figure 1-4

The Swing file selection dialog user interfaces for Windows look-and-feel and Metal look-and-feel are shown in Figures 1-5 and 1-6, respectively. The Swing file selection dialog with Windows look-and-feel looks similar to the AWT (i.e. the native dialog); however, they are quite different. Swing simply emulates

the Windows native file dialog. If you look carefully, you'll find that some features of Windows native file dialogs are missing in the Swing file dialog. In Windows native file dialogs, you can view the files using different modes: list, details, thumbnails, and so on. Additionally, more operations are available in the popup menu when you right-click. Both of these features are not available to Swing file selection dialogs.

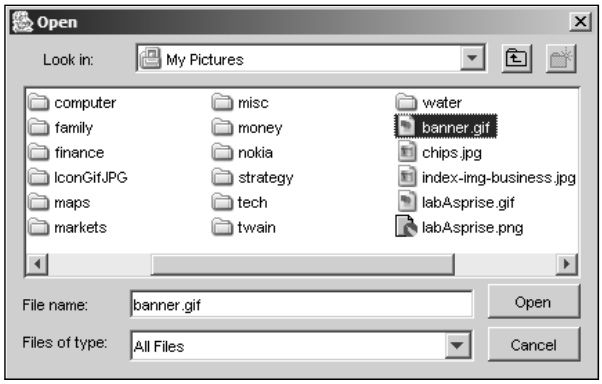

Figure 1-5

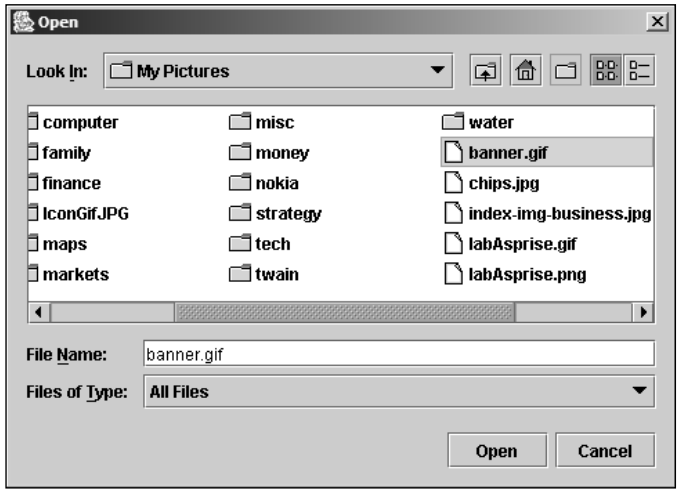

Figure 1-6

Swing fails to support native features of the underlying system. Another obstacle to widespread usage is that programming with Swing is very complex.

Swing is so powerful that you can use it to create full-scale enterprise Java user interface programs. So why do we see so few Swing-based GUI programs? James Gosling, creator of the Java language, said during a keynote presentation at a Mac OS X conference that there is a "perception that Java is dead on the desktop." Complexity of building Swing GUIs, lack of native features, and slow running speed are some of obstacles keeping Swing from succeeding on desktops.

Is any other Java GUI toolkit available that can create full-featured user interface programs? The answer is yes. Standard Widget Toolkit (SWT), along with JFace, provides a complete toolkit for developing portable native user interfaces easily.

SWT and JFace

Eclipse is an open source universal tool platform dedicated to providing a robust, full-featured, industry platform for the development of highly integrated tools. IBM, Object Technology International (OTI), and several other companies launched the Eclipse project in 2001. Today, the Eclipse Board of Stewards includes companies such as Borland, Fujitsu, HP, Hitachi, IBM, Oracle, Red Hat, SAP, and Sybase. With more than 3 million downloads, Eclipse has attracted a huge number of developers in over 100 countries.

The Eclipse platform defines a set of frameworks and common services that are required by most tool builders as common facilities. One of the most important common facilities is the portable native widget user interface. The Standard Widget Toolkit (SWT) provides portable native user interface support, as well as a set of OS-independent APIs for widgets and graphics.

Built on SWT, JFace is a pure Java UI framework handling many common UI programming tasks. The following subsections introduce SWT and JFace in detail.

Figure 1-7 shows the Eclipse platform's native user interface — in this case, Windows. SWT is integrated tightly with the underlying native window system.

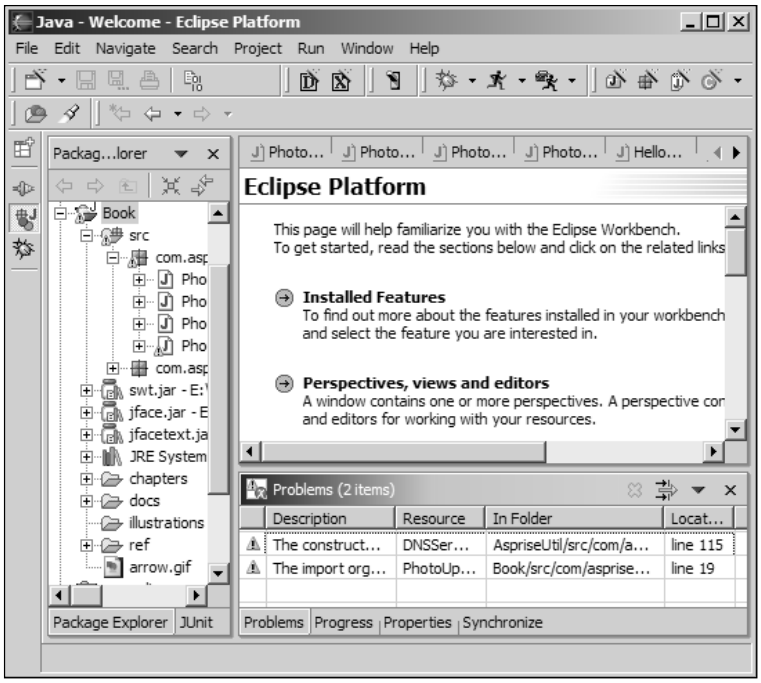

Figure 1-7