Introduction

This is not a book on data binding.[1]

Rich Clients

We should describe what we mean by *Filthy Rich Clients*. But first, we need to describe what rich clients are. *Rich clients* is a phrase commonly associated with desktop applications. Rich client applications are usually contrasted with *thin client* or *Web client* applications, which are essentially software applications running on the server with a simple front end that runs in a browser on the user's desktop.

Rich client applications have more of the program logic and functionality local to the user's desktop machine. On one extreme, the application may be all local, such as a word-processing application or photo-editing software. Or the application may run in the client-server world, as do thin client applications. The data may still be provided by the server, and important functionality may still come from a server, a database, the network, or wherever. But the local application is

1. We figured we should be honest about this disclaimer. When we described the book and its outline in a blog entry, we received a comment that, in fact, the book should be about data binding. While the interaction of Java Desktop applications with data sources is an interesting and critical area to discuss, it's really not what this book is about. At all. If you opened *Filthy Rich Clients* assuming that it would talk about data binding, you might want to close the book and look on a nearby shelf for other books instead. Or change your mind now and realize that this book will be a lot more fun to read.

responsible for much more of the logic, user interface, and interactivity than is a typical Web client.

The distinction between Web and rich clients is an important one because there are trade-offs with each approach that application developers must be aware of in deciding which route to go with their products. The trade-offs vary between different application domains, systems, and technologies but basically boil down to the following:

Web Clients

These applications look like simple Web pages to the user. Their great advantage is their simplicity. They may start up faster than rich clients, taking just the time that it takes for the server to process information and send it over the network to the user's computer. These applications also tend to have a simple, browser-oriented graphical user interface (GUI). This simplicity comes at a cost, however. The application model tends to be very standard: Each page has content, fields for the user to fill in, and buttons to submit information back to the server. The interaction model tends to be batch-oriented: The user sends information, the server processes the information, and the resulting page is sent back to the user. Significant delays in interaction can occur with this complete-send-process-return-display application model.

Rich Clients

These applications have a very "rich" user experience, taking advantage of native facilities of the user's desktop computer, such as graphics hardware acceleration, to provide a more robust and full-featured application experience than is provided by Web clients. Rich client applications can sometimes take longer to start up than a simple Web page because there is more going on in the application, and the GUIs tend to be more involved than Web GUIs because there is more happening in the application than in simple Web-oriented applications. The interaction model is quite different because much of the logic of the application is local, even if the application is talking to a server on the back end.

Lately, a new model has emerged for Web clients, called Asynchronous JavaScript and XML (AJAX), where much of the client-server interaction can be handled in parallel with the user's interacting with each Web page. This transparent client-server interaction can allow for dynamically updated Web pages instead of the more tedious complete-send-process-return-display model of traditional Web client applications. However, this model is still limited by the

browser container in which the application lives and by many of the constraints that that browser model places on the application, including the extent to which JavaScript features are supported, the security model of the browser, and the physical GUI of the browser container around the application.

AJAX applications are starting to explore some of the Filthy Rich features described in this book, including some graphical effects in their GUIs. This is obviously great. We believe that these features can make much more useable applications. But given the browser constraints of AJAX, it is still a Web client technology, and we focus our discussion on the rich client model instead.

Filthy Rich Clients

Filthy Rich Clients is a term that we coined to refer to applications that are so graphically rich that they ooze cool. They suck users in from the outset and hang onto them with a death grip of excitement. They make users tell their friends about the applications. In short, they make users actually *enjoy* their application experience. When was the last time you enjoyed using a software application? Maybe you need more Filthy Rich Clients in your life.

The keys to Filthy Rich Clients are graphical and animated effects. These kinds of effects provide ways of enhancing the user experience of the application through more attractive GUIs, dynamic effects that give your application a pulse, and animated transitions that keep your user connected to the logical flow of the application.

We are not just talking about media players here. We are talking about enhancing *all* kinds of software, from typical enterprise form-based applications to the most gratuitously whizzy consumer application. All applications could benefit from thinking about the user experience and how to apply Filthy Rich effects in ways to improve that experience.

As an example of Filthy Rich Client effects and a shameless teaser for content you will see later in the book, let's see some screenshots (Figures I-1 through I-7).

"Effectives": Effects Enabling More Productive Applications

Graphical effects, especially animated ones, can be overdone. The last thing that a user wants is for everything in an application to be blinking, swooping, pulsing, and fading constantly. The techniques we outline in this book can, if misused,

Figure I-1 Chapter 10, "Layered Panes," shows how to support multiple layers of information in your user interface.

Figure I-2 Reflection, discussed in Chapter 11, "Repaint Manager," brings realism and richness to an application.

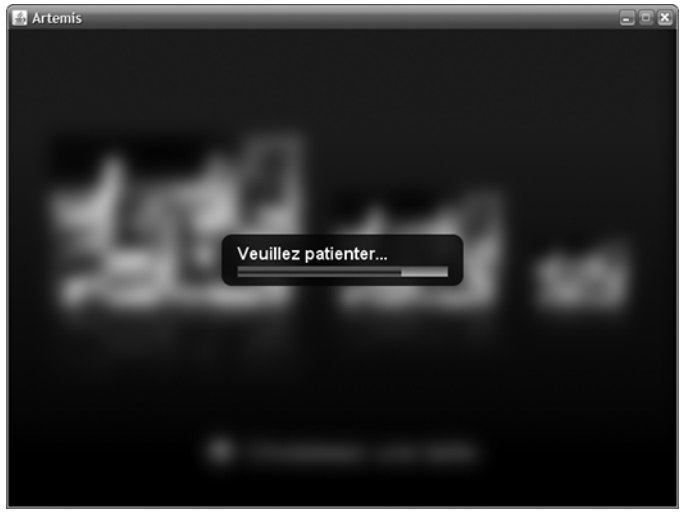

Figure I-3 The blur effect, discussed in Chapter 16, "Static Effects," can be useful for focusing the user's attention on nonblurred items.

Figure I-4 The Aerith application, available in source and binary form at http://aerith.dev.java.net, demonstrates many of the effects and techniques discussed in this book.

Figure I-5 Aerith's loading screen demonstrates the pulsating effect discussed in Chapter 17, "Dynamic Effects."

Figure I-6 The bloom effect is applied to Aerith's loading screen, as discussed in Chapter 17.

Figure I-7 Chapter 18, "Animated Transitions," discusses automating animations between different states of a GUI. Here, a change in thumbnail sizes causes the pictures to automatically and smoothly animate to their new locations and sizes in the window.

contribute to this horror show. We show how to enrich the graphics and animate anything you want. We also discuss how to do so *effectively*, making sure to enrich applications in sensible ways. It is important that you make the application more effective for the user, not just effect-ridden. Done right, the addition of graphical effects to an application can really draw users in and keep them there.

Note that this book does not attempt to cover the deep topic of interface design. We touch on this topic in the context of particular effects and techniques as we discuss them, but if you wish to know more about designing user interfaces, check out some of the great books out there on the subject. You might start with Chapter 19 of this book, however, which discusses the UI design process used in developing a particular application.

Why Java and Swing for Filthy Rich Clients?

The techniques that we discuss in this book apply to most graphically oriented development toolkits. Anything that allows you to change the appearance of the GUI elements of your application can take advantage of the approaches explored here.

However, we have developed the sample code, the utility frameworks, and most of the techniques around the programming environment of Java and Swing. Swing is the library of classes for developing GUIs for Desktop Java applications. Swing's great advantage over other GUI toolkits is its flexibility and customization. These capabilities are exploited to a great extent in this book as we explore how to use custom rendering and animation to create great-looking effects in applications.