

Preface

When we were first exposed to Eclipse back in late 1999, we were struck by the magnitude of the problem IBM was trying to solve. IBM wanted to unify all its development environments on a single code base. At the time, the company was using a mix of technology composed of a hodgepodge of C/C++, Java, and Smalltalk.

Many of IBM's most important tools, including the award-winning VisualAge for Java IDE, were actually written in Smalltalk—a wonderful language for building sophisticated tools, but one that was rapidly losing market share to languages like Java. While IBM had one of the world's largest collections of Smalltalk developers, there wasn't a great deal of industry support for it outside of IBM, and there were very few independent software vendors (ISVs) qualified to create Smalltalk-based add-ons.

Meanwhile, Java was winning the hearts and minds of developers worldwide with its promise of easy portability across a wide range of platforms, while providing the rich application programming interface (API) needed to build the latest generation of Web-based business applications. More important, Java was an object-oriented (OO) language, which meant that IBM could leverage the large body of highly skilled object-oriented developers it had built up over the years of creating Smalltalk-based tools. In fact, IBM took its premiere Object Technology International (OTI) group, which had been responsible for creating IBM's VisualAge Smalltalk and VisualAge Java environments (VisualAge Smalltalk was the first of the VisualAge brand family and VisualAge Java was built using it), and tasked the group with creating a highly extensible integrated development environment (IDE) construction set based in Java. Eclipse was the happy result.

OTI was able to apply its highly evolved OO skills to produce an IDE unmatched in power, flexibility, and extensibility. The group was able to replicate most of the features that had made Smalltalk-based IDEs so popular the decade before, while simultaneously pushing the state of the art in IDE development ahead by an order of magnitude.

The Java world had never seen anything as powerful or as compelling as Eclipse, and it now stands, with Microsoft's .NET, as one of the world's premier development environments. That alone makes Eclipse a perfect platform for developers wishing to get their tools out to as wide an audience as possible. The fact that Eclipse is completely free and open source is icing on the cake. An open, extensible IDE base that is available for free to anyone with a computer is a powerful motivator to the prospective tool developer.

It certainly was to us. At Instantiations and earlier at ObjectShare, we had spent the better part of a decade as entrepreneurs focused on building add-on tools for various IDEs. We had started with building add-ons for Digitalk's Smalltalk/V, migrated to developing tools for IBM's VisualAge Smalltalk, and eventually ended up creating tools for IBM's VisualAge Java (including our award-winning VA Assist product and our jFactor product, one of the world's first Java refactoring tools). Every one of these environments provided a means to extend the IDE, but they were generally not well-documented and certainly not standardized in any way. Small market shares (relative to tools such as VisualBasic) and an eclectic user base also afflicted these environments and, by extension, us.

As an Advanced IBM Business Partner, we were fortunate to have built a long and trusted relationship with the folks at IBM responsible for the creation of Eclipse. That relationship meant that we were in a unique position to be briefed on the technology and to start using it on a daily basis nearly a year and half before the rest of the world even heard about it. When IBM finally announced Eclipse to the world in mid-2001, our team at Instantiations had built some of the first demo applications IBM had to show. Later that year when IBM released its first Eclipse-based commercial tool, WebSphere Studio Application Developer v4.0 (v4.0 so that it synchronized with its then current VisualAge for Java v4.0), our CodePro product became the very first commercial add-on available for it (and for Eclipse in general) on the same day.

Currently, the CodePro product adds hundreds of enhancements to Eclipse and any Eclipse-based IDE. Developing CodePro over the last several years has provided us with an opportunity to learn the details of Eclipse development at a level matched by very few others (with the obvious exception of the IBM and OTI developers, who eat, sleep, and breathe this stuff on a daily basis). CodePro has also served as a testbed for many of the ideas and tech-

niques presented in this book, providing us with a unique perspective from which to write.

Goals of the Book

This book, originally titled *Eclipse: Building Commercial-Quality Plug-ins*, provides an in-depth description of the process involved in building commercial-quality extensions for the Eclipse and the IBM Software Development Platform (SDP)—IBM’s commercial version of Eclipse—development environments. To us, “commercial-quality” is synonymous with “commercial-grade” or “high-quality.” Producing a *commercial-quality* plug-in means going above and beyond the minimal requirements needed to integrate with Eclipse. It means attending to all those details that contribute to the “fit and polish” of a commercial offering.

In the world of Eclipse plug-ins, very few people take the time to really go the extra mile, and most plug-ins fall into the open source, amateur category. For folks interested in producing high-quality plug-ins (which would certainly be the case for any software company wanting to develop Eclipse-based products), there are many details to consider. Our book is meant to encompass the entire process of plug-in development, including all the extra things that need to be done to achieve high-quality results. This book has several complementary goals:

- Provide a quick introduction to using Eclipse for new users
- Provide a reference for experienced Eclipse users wishing to expand their knowledge and improve the quality of their Eclipse-based products
- Provide a detailed tutorial on creating sophisticated Eclipse plug-ins suitable for new and experienced users

The first three chapters introduce the Eclipse development environment and outline the process of building a simple plug-in. The intention of these chapters is to help developers new to Eclipse quickly pull together a plug-in they can use to experiment with.

The first chapter, in particular, introduces the reader to the minimum set of Eclipse tools that he or she will need to build plug-ins. It is a fairly quick overview of the Eclipse IDE and relevant tools (one could write an entire book on that topic alone), and we would expect expert Eclipse users to skip that chapter entirely.

The second chapter introduces the example that we will use throughout most of the book and provides a very quick introduction to building a working plug-in from start to finish. The third chapter presents a high-level overview of the Eclipse architecture and the structure of plug-ins and extension points.

The fourth and fifth chapters cover the Standard Widget Toolkit (SWT) and JFace, which are the building blocks for all Eclipse user interfaces (UIs). These chapters can act as a stand-alone reference; they are intended to provide just enough detail to get you going. Both of these topics are rich enough to warrant entire books and several are currently available.

The subsequent chapters, comprising the bulk of this book, focus on describing each of the various aspects of plug-in development and providing the reader with in-depth knowledge of how to solve the various challenges involved. Each chapter focuses on a different aspect of the problem, and includes an overview, a detailed description, a discussion of challenges and solutions, diagrams, screenshots, cookbook-style code examples, relevant API listings, and a summary.

We have structured the book so that the most important material required for every plug-in project appears in the first half of it. Some of the packaging- and building-oriented material is placed at the end (for example, features and product builds). This organizational scheme left several topics that, while not critical to every plug-in, were important to the creation of commercial-quality plug-ins. These topics have been placed in the second half of the book in an order based on the importance of each and how it related to earlier material. Internationalization, for example, is one of those topics. It isn't critical, and it isn't even all that complicated when you get right down to it. It is, however, important to the book's premise, so we felt it was a topic we needed to include. Since we aren't assuming that the reader is an Eclipse expert (or even a plug-in developer), we have tried to take the reader through each of the important steps in as much detail as possible. While it is true that this is somewhat introductory, it is also an area that most plug-in developers totally ignore and have little or no experience with.

Sometimes a developer needs a quick solution, while at other times that same developer needs to gain in-depth knowledge about a particular aspect of development. The intent is to provide several different ways for the reader to absorb and use the information so that both needs can be addressed. Relevant APIs are included in several of the chapters so that the book can be used as a stand-alone reference during development without requiring the reader to look up those APIs in the IDE. Most API descriptions are copied or paraphrased from the Eclipse platform Javadoc.

As the originators of Eclipse and a major consumer of Eclipse-based technology, IBM is justifiably concerned that new plug-ins meet the same high-quality standards that IBM adheres to. To that end, IBM has established a rigorous *Ready for Rational Software* (RFRS) certification program meant to

ensure the availability of high-quality add-ons to Eclipse and the IBM Software Development Platform. RFRS certification should be one of the ultimate goals for anyone wishing to build and market Eclipse plug-ins. Every chapter covers any relevant RFRS certification criteria and strategies.

The examples provided as part of the chapters describe building various aspects of a concrete Eclipse plug-in that you will see evolve over the course of the book. When used as a reference rather than read cover-to-cover, you will typically start to look in one chapter for issues that are covered in another. To facilitate this type of searching, every chapter contains numerous cross-references to related material that appears in other chapters.

Intended Audience

The audience for this book includes Java tool developers wishing to build products that integrate with Eclipse and other Eclipse-based products, relatively advanced Eclipse users wishing to customize their environments, or anyone who is curious about what makes Eclipse tick. You do not need to be an expert Eclipse user to make use of this book because we introduce most of what you need to know to use Eclipse in Chapter 1, *Using Eclipse Tools*. While we don't assume any preexisting Eclipse knowledge, we do anticipate that the reader is a fairly seasoned developer with a good grasp of Java and at least a cursory knowledge of extensible markup language (XML).

Conventions Used in This Book

The following formatting conventions are used throughout the book.

Bold—the names of UI elements such as menus, buttons, field labels, tabs, and window titles

Italic—emphasize new terms and Web site addresses

`Courier`—code examples, references to class and method names, and filenames

Courier Bold—emphasize code fragments

“Quoted text”—quotation marks surrounding text indicates words to be entered by the user

What's New in the Third Edition

In this edition, we use the same **Favorites** view example as in the first and second editions, but have reworked much of the content and recreated the code from scratch. Some Eclipse concepts, such as views and editors, are similar but with additional functionality and capabilities; other areas, such as commands, GEF and PDE Build have been added. The following are some of the major changes in this third edition:

Eclipse Command Framework

The Eclipse command framework replaces the older Action framework. Throughout the book, use of the older Action framework has been replaced with new content describing how to accomplish the same thing with the new command framework. This is most obvious in Chapter 6 where the first half of the chapter is entirely devoted to the command framework. Chapter 7 and Chapter 8 also have lots of new material describing use of commands with views and editors.

Eclipse 3.4 and Java 5

All of the screen shots, text and code examples throughout the book have been updated to use the latest Eclipse 3.4 API and Java 5 syntax. Chapter 1 has been overhauled to include descriptions of new capabilities in Eclipse 3.4 including a new overview of using Mylyn and discussion of new preferences. Both Chapter 1 and Chapter 2 cover cool PDE and SWT tools available in Eclipse 3.4.

New GEF Chapter

GEF, the Graphical Editing Framework from Eclipse.org, provides a toolkit for building dynamic interactive graphical user interface elements. Chapter 20 takes you step by step through the process of building a GEF-based view for graphically presenting the relationships between the favorites items and their underlying resources. We then go further, building a GEF-based editor with the ability to add, move, resize, and delete the graphical elements representing those favorites items.

Put PDE Build through its paces

Over the past several years, the PDE build process has been steadily maturing. The proprietary Ant scripts in Chapter 19 of earlier versions of our book have been completely replaced with step-by-step instructions for getting an Eclipse PDE Build process up and running to automate assembly of your product for distribution.

New “p2” update site creation description

“p2” debuts in Eclipse 3.4, replacing the older Update Manager. We take you through the process of using update sites and then building your own in Section 18.3, Update Sites, on page 679.

Acknowledgments

The authors would like to thank all those who have had a hand in putting this book together or who gave us their support and encouragement throughout the many months it took to create.

To our comrades at Instantiations, who gave us the time and encouragement to work on this book: Rick Abbott, Brent Caldwell, Devon Carew, Jim Christensen, Taylor Corey, Dianne Engles, Marta George, Nick Gilman, Seth Hollyman, Mark Johnson, Ed Klimas, Tina Kvalle, Alexander Lobas, Warren Martin, David Masulis, Nancy McClure, Steve Messick, Alexander Mitin, Gina Nebling, John O’Keefe, Keerti Parthasarathy, Phil Quitslund, Mark Russell, Rob Ryan, Andrey Sablin, Balaji Saireddy, Konstantin Schegloy, Chuck Shawan, Bryan Shepherd, Julie Taylor, Mike Taylor, Solveig Viste, Brian Wilkerson, and Jaime Wren.

A special thanks to Jaime Wren, who provided much of the basic research and initial content for Chapter 20.

To our agent, Laura Lewin, and the staff at Studio B, who encouraged us from day one and worked tirelessly on our behalf.

To our editors, Greg Doench and John Neidhart, our production editors, Elizabeth Ryan and Kathleen Caren, our copy editors, Marilyn Rash and Camie Goffi, our editorial assistants, Michelle Housley and Mary Kate Murray, our art director, Sandra Schroeder, our marketing managers, Brandon Prebynski and Beth Wickenhiser, and the staff at Pearson, for their encouragement and tremendous efforts in preparing this book for production.

To Simon Archer and Robert Konigsberg who contributed an unparalleled number of changes and suggestions to various editions of the book, and helped us improve them in almost every dimension.

To Linda Barney who helped us polish and edit the second edition.

To our technical reviewers who helped us enhance the book in many ways: Matt Lavin, Kevin Hammond, Mark Russell, Keerti Parthasarathy, Jaime Wren, Joe Bowbeer, Brian Wilkerson, Joe Winchester, David Whiteman, Boris Pruesmann, Balaji Saireddy, and Raphael Enns.

To the many readers of the first and second editions who contributed errata that have gone into this third edition: Bruce Gruenbaum, Tony Saveski, James Carroll, Tony Weddle, Karen Ploski, Lori Kissell, Brian Vosburgh, Peter Nye, Chris Lott, David Watkins, Simon Archer, Mike Wilkins, Brian Penn, Bernd Essann, Eric Hein, Dave Hewitson, Frank Parrott, Catherine Suess,

William Beebe, Janine Kovack, David Masulis, Jim Norris, Jim Wingard, Roy Johnston, David Karr, Chris Gage, Paul Tamminga, Asim Ullah, and Ebru Aktuna.

To the series editors, Erich Gamma, Lee Nackman, and John Weigand, for their thoughtful comments and for their ongoing efforts to make Eclipse the best development environment in the world.

We would also like to thank our wives, Karen and Kathy, for their endless patience, and our children, Beth, Lauren, Lee, and David, for their endless inspiration.

About the Authors



Eric Clayberg is Senior Vice President for Product Development for Instantiations, Inc. Eric is a seasoned software technologist, product developer, entrepreneur, and manager with more than seventeen years of commercial software development experience, including twelve years of experience with Java and nine years with Eclipse. He is the primary author and architect of more than a dozen commercial Java and Smalltalk add-on products, including the popular WindowBuilder Pro, CodePro, and the award-winning VA Assist product lines. He has a B.S. from MIT, an MBA from Harvard, and has cofounded two successful software companies—ObjectShare and Instantiations.



Dan Rubel is Chief Technology Officer for Instantiations, Inc. He is an entrepreneur and an expert in the design and application of OO technologies with more than fifteen years of commercial software development experience, including thirteen years of experience with Java and nine years with Eclipse. He is the architect and product manager for several successful commercial products, including RCP Developer, WindowTester, jFactor and jKit, and has played key design and leadership roles in other commercial products such as VA Assist, and CodePro. He has a B.S. from Bucknell and is a cofounder of Instantiations.

Instantiations is an Advanced IBM Business Partner and developer of many commercial add-ons for Eclipse and IBM's VisualAge, WebSphere, and Rational product lines. Instantiations is a member of the Eclipse Foundation and a contributor to the Eclipse open source effort with responsibility for the Eclipse Collaboration Tools project known as Koi and for the Eclipse Pollinate project (Beehive).

How to Contact Us

While we have made every effort to make sure that the material in this book is timely and accurate, Eclipse is a rapidly moving target and it is quite possible that you may encounter differences between what we present here and what you experience using Eclipse. The Eclipse UI has evolved considerably over the years, and the latest 3.4 release and upcoming 3.5 release are no exceptions. While we have targeted it at Eclipse 3.4 and used it for all of our examples, this book was completed after Eclipse 3.4 was finished and during the initial phases of development of Eclipse 3.5. If you are using an older or newer version of Eclipse, this means that you may encounter various views, dialogs, and wizards that are subtly different from the screenshots herein.

- Questions about the book's technical content should be addressed to:
info@qualityeclipse.com
- Sales questions should be addressed to Addison-Wesley at:
www.informit.com/store/sales.aspx
- Source code for the projects presented can be found at:
www.qualityeclipse.com/projects
- Errata can be found at:
www.qualityeclipse.com/errata
- Tools used and described can be found at:
www.qualityeclipse.com/tools

