The Java™ Developer's Guide to Eclipse

Foreword

What makes Eclipse interesting isn't so much that it's a well-designed tool integration platform, or that it's an open source project with an active worldwide developer community, or even that there are some 30 or so (and growing) industry-leading companies behind eclipse.org. But all of these things together…they make Eclipse very interesting.

What brings all these people and organizations with diverse interests together? The answer is the Eclipse vision: a new way of looking at how we build tools. In the past, tool developers pretty much had control over how the tools worked and how they looked to the user. Developers thought of tools as stacks of technology—tools, built on frameworks, upon more frameworks, upon basic services, and so on. Worse, there was really no common stack. Providing meaningful, "deep" integration across different tool stacks was virtually impossible.

The Eclipse vision is different—it's a "platform-centric" rather than "tool-centric" way of thinking. The bare-bones Eclipse Platform is essentially a "Universal IDE"—an IDE for anything and nothing in particular, all at the same time. When you want to build a new tool for Eclipse, you think about how you "teach" the Eclipse Platform about your problem, rather than about how you can bolt a monolithic "tool" on top of it. And the way you teach Eclipse about your problem is by writing tool plug-ins that hook into well-defined plug-in points. The result is that rather than see a new "tool" added to Eclipse, users instead see new capabilities that the Eclipse Platform is now able to perform.

This idea didn't just emerge out of nowhere, and it didn't come from one person or one place. It is a product of the reality of the software industry and the influence of Internet technologies on the kinds of applications developers found themselves needing to build. In the late 1990s everyone was trying to bring their business functions to the Internet to connect with customers and business partners directly. These e-business applications required developers to work with many different types of resources: XML, JSPs, Java code, graphics files, HTML files, style sheets, and so on. Having separate tools for working

with each type of resource—or worse, multiple tools for one resource type—started to become burdensome. It was often difficult to use the different tools together—they didn't understand each other's data, clobbered each other's files, and provided different user interfaces that each had to be learned.

The Eclipse Project was started by IBM and OTI in response to this need to find a new way to enable people to independently build tools that when used together work as if they are part of a single integrated tool set. This is the trick of Eclipse. Different people, different teams, different organizations can build different tools at different times, and yet when they are assembled into the Eclipse Platform on the user's desktop they behave, if it's done right, as if they were designed to provide a single integrated experience. Of course, it's possible to get this wrong by building tools that don't fit well with Eclipse. But if your tool is a good Eclipse "citizen" and follows the paradigms established by the base platform, things tend to work out pretty well.

The best tools integrate so seamlessly you can't tell where one tool starts and another ends. In fact, to the user at least, the concept of distinct "tools" almost disappears. But to build good tools you need to understand how your tool should extend the environment to solve your particular problem—how your tool should "plug in" to the integration points defined by Eclipse. You also need to decide when to invent something new and when you should just extend or enhance an existing function.

This book does an excellent job of helping you with these questions. It was written by members of the IBM Jumpstart team who have considerable experience teaching courses about Eclipse and helping people get started with Eclipse. They have discovered what tends to trip people up, and they have honed their approach to explaining what Eclipse is, how to use it, and how to extend it. They show you how to use Eclipse proficiently, and then transition nicely into extending the environment yourself and building your own tools. The practical examples and exercises included have been proven in real-life course situations and are invaluable in helping you to get up and running quickly.

The idea of Eclipse has captured the imagination of developers from all corners of the globe—it has caught on beyond our wildest dreams. But the real value of Eclipse comes not from Eclipse itself, but from the tool plug-ins people build that teach it how to work with things—Java files, Web content, graphics, video…. It's limited only by your imagination!

—Dave Thomson
Eclipse Project Program Director
Object Technology International, Inc.

Preface

Origin of the Book

Starting in late 1999, the authors formed the core of a group within IBM called the Jumpstart team. Our team was created to share knowledge of the Eclipse technology throughout IBM and with its business partners, that is, to "jumpstart" the IBM and partner development community on Eclipse. Part of this effort included the creation of a set of presentations, lecture materials, and accompanying exercises. Over the ensuing months, as the Eclipse technology matured, the presentations and exercises matured as well. As the Eclipse community grew to include various companies and academic institutions, requests for this information grew as well. After every class we taught, we revised and improved the materials. When our schedules could not keep pace with the demand, we adapted the materials and made them available for use in a self-study mode. This was the genesis of the book. You can think of each chapter in the book as a classroom lesson. The exercises and solutions reinforce the concepts of the chapters and provide you with practice using or extending aspects of Eclipse.

Goals

Our goals in bringing you this book are to

- Provide information for those new to Eclipse
  A new user can leverage this book as a tutorial, starting with the first chapter and progressing sequentially through the book. We do not assume prior Eclipse knowledge.
- Explore the capabilities of Eclipse
  The book will cover both using Eclipse as your development environment and extending Eclipse. The chapters in Part I start with Eclipse as

a general development environment and then progress to developing and debugging Java, and more advanced usage topics, for example, using Eclipse in a team environment. In the chapters on extending Eclipse in Part II, we cover the most common classes for each Eclipse framework. References to design patterns, where applicable, illustrate the architectural relationships among the classes. The intent is not to replace the Javadoc that is included with Eclipse, but to complement the documentation by focusing on how to bring a set of classes together to complete a task.

- Provide exercises and working examples that are simple and focused on the chapter topic
  Our intent is not to provide a single, "real world" example or that the completion of all the exercises will result in a single, functioning Eclipse-based tool. Instead, the exercises augment the chapter topics and illustrate key points. The chapter text concentrates on the concepts and outlines the basic steps to accomplish a task while providing small sections of code or screen captures to illustrate the point. The exercises provide detailed coding instructions and screen captures to apply the concepts described in the chapter. The CD-ROM that comes with this book contains solutions to the step-by-step exercises as well as additional working examples to supplement chapters in the book.
- Provide reference material for those experienced in using Eclipse
  Since the material in this book is in a modular form, you can explore each chapter individually. You can use this book as a reference guide, and jump to chapters you are interested in exploring.
- Promote the Eclipse community
  This book provides you with the basic knowledge of Eclipse so that you can become an active participant and help grow the Eclipse open source community.

Though the term "Eclipse" conveys the image of a solar event causing darkness, the intent of this book is to shed light, add clarity, and focus on a powerful new platform. Whether you are new to Eclipse or one of the early adopters, we welcome you to the Eclipse community.

Intended Audience and Prerequisites

The audience for this book includes Java programmers who plan to use Eclipse as their integrated development environment (IDE), those who will use Eclipse-based offerings, advanced users who want to customize Eclipse

further, and tool providers who seek to develop tools that will integrate with Eclipse and other Eclipse-based offerings. Prior Eclipse experience is not needed; however, this book assumes that you are familiar with the Java programming language. While it describes how to use the Java development tools, it does not teach the syntax and semantics of the Java programming language.

How the Book Is Organized

Part I of the book applies to those using Eclipse as their development environment. The book begins by covering the basic navigation and terminology of Eclipse. You will learn about the Java development environment, including secrets to becoming a power user. You will learn how to use the flexibility of Eclipse to maximize your productivity and fit your own personal style. Students who are studying the Java programming language may find using Eclipse, instead of simply a command line environment, a much more productive and exciting way to learn the richness and power of Java. Instructors may discover how using Eclipse in the classroom will accelerate the student's mastery of the language and be a productive tool to use in research.

Part II focuses on extending Eclipse with additional capabilities and building an offering based on Eclipse. The chapters in this part begin with the fundamentals of building a plug-in and working with resources. Part II describes the various Java frameworks provided to make contributing additional function to Eclipse easier and more consistent. It covers extending the user interface by adding menu choices, toolbar buttons, views, editors, dialogs, wizards, and online documentation. The later chapters in Part II cover more advanced topics such as creating extensible plug-ins, and includes chapters for those with special interests such as integrating existing OLE or Swing code with Eclipse and extending the Eclipse Java development tools.

Learning in a programming environment without actually writing code is difficult. Part III contains a series of detailed exercises to reinforce the concepts presented in the book. Part III depends on the files included on the CD-ROM. The CD-ROM contains solutions to all of the exercises and has many code samples augmenting the material in the chapters. The samples do not depend on one another, so you can study them in any order.

There are many Eclipse-based tools available today and more under development, including ones for C++, Web services, J2EE programming, and UML modeling. This book focuses on the use of Eclipse by the Java developer. However, the fundamentals of Eclipse covered in the first few chapters can be used across all types of tools.

Coding Conventions

XML and Java code in this book is set in a `monospace` typeface. To highlight a section of the code under discussion, the code will be **`bold`**. Specific filenames are also set in `monospace`.

References in the book that describe the user interface, such as a series of menu selections, are set in **bold**. For example, **File > New > Project** indicates that the first menu choice selected is **File** followed by the menu choice **New** and then **Project**. Often menu choices have a corresponding toolbar icon. Icons used in Eclipse, such as those found on the toolbar or those used with editors and views, are in the left margin next to the appropriate text. For example, the following text shows the three icons representing a Java source file, a Java class, and a Java interface in the left margin in the order referenced in the text.

Double-click on a Java source file, class, or interface, or select one of these in a view and press **F3** or **Enter**.

CD-ROM

The CD-ROM included with this book contains the following.

- Eclipse SDK version 2.0
- Eclipse Examples version 2.0
- Exercise template files
- Exercise solutions and other samples
- A `readme.html` file with installation instructions

To complete the exercises, you must install Eclipse SDK version 2.0. The Eclipse SDK requires that you install a Java Development Kit (JDK) version 1.3 or higher. You can download a JDK from http://www.ibm.com/java/jdk or http://java.sun.com. The files on the CD-ROM are designed for Windows 2000 and Windows XP. Because the examples are written in the Java programming language, you can use them on other operating systems as well, as long as the code or instructions do not depend on Windows-specific function. See the file `readme.html` on the CD-ROM for more information.

Acknowledgments

The authors would like to express their gratitude to all of those who have graciously given us their support, encouragement, and shared their wisdom.

To the entire OTI development team who gave us Eclipse, without which there would be no book.

To our technical reviewers, who helped improve the quality of the book: Kyle Brown, Kevin Haaland, Ed Hintz, Veronika Irvine, Lee Nackman, W. David Pitt, and Scott Rich.

To the management team at IBM for their understanding and support, especially Paul Buck, Lee Nackman, and Liz Hines.

To all those who helped us produce the book: Tyrrell Albaugh, Amy Fleischer, Rebecca Greenberg, Mary O'Brien, and Ann Sellers.

To our IBM publishing mentor, Dave Peterson, for his advice on the book-writing process.

Sherry thanks her parents, Marvin and Marilyn Weisbrod, who taught her always to do her best and gave her the "teaching" genes. Sherry also thanks Jon, the man who captured her heart, and her sister, Robin, who provided endless encouragement.

Jim thanks his loving wife, Maya, and children, Christopher and Lisa, for their patience and support during his all-too-frequent preoccupation with the preparation of this book.

Scott would like to thank his wife, Melanie, and children, Ashley and Kelly, for putting up with a real pain in the neck.

Dan would like to thank his wife, Laura, for her love, patience, and support while working on this book, and his three sons, Nico, Matthew, and Nathan, for both driving him crazy and keeping him sane during those all-too-brief writing breaks.

John would like to thank his wife, Kristin, and his sons, Ben and Peter, for their patience and support.

Pat would like to thank his wife, Chris, for supporting the time spent working on this book, his children Brendan and Caitlin, for the smiles and the odd extra keystroke while cubbyholed upstairs, and his parents, Pat Sr. and Sandy, for supporting the vacation that kept the effort balanced with reality.

About the Authors

Sherry Shavor is a Senior Software Engineer at IBM in Research Triangle Park, North Carolina. Sherry most recently managed the Eclipse Jumpstart team, which provides Eclipse education to the IBM development community and business partners. Sherry has taught Java technology to customers in the United States, Canada, and China. She has also taught at North Carolina State University. Since joining IBM, Sherry has held a variety of positions in product development, presented at numerous conferences, and published articles in the *Java Developer's Journal* and *IBM Systems Journal*. She received a B.S. in Computer Science from the State University of New York at Stony Brook.

Jim D'Anjou is a Senior Software Engineer located at the IBM Silicon Valley Lab in San Jose, California. He has a degree in Computer Science from the University of California at Berkeley. Jim has over 25 years of industry experience at IBM and elsewhere. He has held a variety of technical and management positions developing products for relational databases, database tools, application repositories, and application development tools. He holds two U.S. patents for work in software process automation. In March 2001 he joined the Eclipse Jumpstart team, where his focus is on Team support. Jim helps repository ISVs enable the Eclipse Platform.

Scott Fairbrother is an Advisory Software Engineer at IBM in Research Triangle Park, North Carolina. Scott is a software developer with over 20 years of experience. He has developed object-oriented application frameworks for business process management. He has written specifications for IBM middleware on Windows 2000 and has also written about Microsoft Visual Studio .NET. Most recently, Scott has worked on the Eclipse Jumpstart team, helping IBM and partners create commercial offerings based on Eclipse. He received a B.S. in Marine Biology from the University of North Carolina at Wilmington.

Dan Kehn is a Senior Software Engineer at IBM in Research Triangle Park, North Carolina. His interest in object-oriented programming goes back to 1985, long before it enjoyed the acceptance it has today. He has a broad range of software experience, having worked on development tools like VisualAge for Smalltalk, operating system performance and memory analysis, and user interface design. Dan worked as a consultant for object-oriented development projects throughout the U.S. as well as for four years in Europe. His recent interests include object-oriented analysis/design, application development tools, and Web programming with the WebSphere Application Server. In May 2001 he joined the Eclipse Jumpstart team, which helps ISVs create commercial offerings based on the Eclipse Platform.

John Kellerman joined IBM in 1984 with a Computer Science degree from Purdue University. He has since completed graduate degrees in Computer Engineering at North Caroline State and Business Administration at the University of North Carolina at Chapel Hill. He has spent the majority of his 19 years at IBM in the development and management of application development tool products, including ISPF/PDF, VisualAge Smalltalk, VisualAge Generator, and Eclipse. John was a founding member of the Eclipse Project, which got under way in late 1999. He is currently IBM Product Manager of Eclipse. His responsibilities include working closely on behalf of IBM with eclipse.org and the member companies to help grow the Eclipse community of contributors and commercial offerings.

Pat McCarthy, a Senior Software Engineer at IBM, is a specialist in the use and management of development technologies on a variety of run-time platforms. Pat's IBM career has included hands-on development of business application systems in Poughkeepsie, New York, and 12 years of project management for the development of IBM Redbooks and education offerings in San Jose, California. He has spent the last several years in Raleigh, North Carolina, focused on supporting the use of Eclipse technology in IBM application development products. Pat has a B.S. from Indiana University of Pennsylvania and an M.S. from Marist College, and is the co-author of more than 20 IBM Redbooks.