Contents

Foreword xxix
Preface xxxi

Origin of the Book xxxi
Goals xxxiii
Intended Audience and Prerequisites xxxiv
How the Book Is Organized xxxiv
Coding Conventions xxxvi
CD-ROM xxxvi
Where to Find Further Information xxxvii

Guide to Reading This Book xxxix

Acknowledgments xliii

About the Authors xlv

Chapter I Read Me First 1

Eclipse as a Host for Application DevelopmentTools 2
Beyond Tools: Eclipse as a Host for Client Applications 4
What Is Eclipse? 4
  Integration Platform for Tools and Applications 5
  Java Development Environment 7
  Open Source Community 7
  The Goals of the Eclipse Foundation 8
  Participating in the Community as a User and a Contributor 9
  The Eclipse License 10

Getting and Installing Eclipse 11
## The Eclipse Platform Runtime

### User Interface Frameworks

- Standard Widget Toolkit 227
- JFace User Interface Framework 228

### Workbench Frameworks

- Storing Persistent Values 232
- Creating Applications with the Rich Client Platform 232
- Remaining Responsive to the User Using the Concurrency Framework 234
- Managing the User Interface Through Capabilities 234
- Extending the Workbench Integrated Development Environment 234

### Chapter Summary

### References

---

### Chapter 9 Getting Started: Plug-in Development

#### Getting Started with Plug-ins

- Integration Scenarios 239
- Distributing Your Plug-in 242

#### Getting Started with Extensions and Extension Points

#### Basic Steps of Implementing a Plug-in

- Declaring Your Plug-in Manifest 249
- Defining Your Plug-in Class 252
- Installing Your Plug-in 254

#### Using the Plug-in Development Environment

- PDE Views and Editors 255
- Managing the Runtime and Development Environments 257
- Creating and Running a Plug-in 260

#### Exercise Summary

#### Chapter Summary

#### References

---

### Chapter 10 Creating Applications Using the Rich Client Platform

#### Reasons Applications Are Being Built on the RCP

- Advantages for the Enterprise 264
- Advantages for Software Developers 265
- Advantages for Users 265
Chapter 11 Creating Extension Points: How Others Can Extend Your Plug-ins 287

Relationship Between Extension Points and Extensions 288

Viewing the Official List of Enabled Plug-ins, Extensions, and Extension Points 289

How to Define Extension Points 290

Choosing the Existing Code You Want to Enable Others to Extend or Configure 291
Declaring the Existence of the Extension Point 293
Defining an Interface for Expected Behavior 294
Processing the Registry Entries for Your Extension Points 297
Invoking the Appropriate Methods Per the Previously Defined Interface Contract 298

Handling Dynamic Plug-ins 300

How to Enable the Schema-Based New Extension Wizard 301

Why Define a Schema? 302
Schema Editor 302

Example Summary 303
Chapter Summary 303
References 304
Chapter 15 JFace Viewers

**Viewer Basics**
- What Is a Viewer? 367
- What Do Viewers Look Like? 368
- What Types of Viewers Exist? 369
- How Do I Use a Viewer? 369
- Why Should Viewers Be Used? 370

**Viewer Framework**
- Class Structure 370
  - Content Provider-Content Viewer-Label Provider Relationships 372

**Creating the Basic Content Viewer Arrangement**
- Identifying the Input 374
- Creating the Viewer 374
- Creating the Content Provider 375
- Creating the Label Provider 376
- Connecting the Viewer with the Content and Label Providers 377
- Defining an Input to the Viewer 377
- Implementing Model-Viewer Synchronization 379

**Controlling the Content Displayed in a Structured Viewer**
- Filtering the Viewer Content 384
- Sorting the Viewer Content 385

**Interacting with Viewer Objects**
- Handling User Selection of Viewer Objects 388
- Options for Creating Viewer Pop-up Menu Actions 389
- Adding Viewer Actions 390
Chapter 16 Dialogs and Wizards

**Dialogs**—The User Interface Beyond Workbench Parts

- Point of Reference—The User Interface Framework
- Workbench Extension Points for Dialogs and Wizards
- Common Reusable Dialogs

**Contributing to the Preferences Dialog**

- Defining a Preference Page Extension
- Implementing a Preference Page
- Defining a Preference Page User Interface
- Adding Preference Value Logic to a Preference Page
- Building a Field Editor Preference Page

**Contributing to the Properties Dialog**

- Defining a Property Page Extension
- Implementing a Property Page
- Defining the Property Page User Interface
- Adding Resource Property Access Logic
- Saving Resource Type-Specific Properties
- Implementing Your Own Properties Dialog

**Contributing Wizards**

- Defining a Wizard Extension
- Implementing a Wizard
- Implementing a Wizard Page
- Customizing a Wizard
- Controlling Wizard Page Progression
- Delaying Creation of a Wizard Page User Interface
- Reusable Creation of a Wizard Page User Interface

Directly Opening a Wizard Dialog for Your Wizard

**Example Summary**
Chapter 17  Dialog Settings, Preferences, 
and User Settings  441

Overview of the Options  441
  Dialog Settings  442
  Preference Values  442
  User Settings  442

Storing Values in Dialog Settings  443
  Dialog Settings Management  443
  Usage Considerations  444

Storing Preference Values for a Plug-in  445
  Preference Value Management  445
  Establishing Default Preference Values  446
  Reacting to Preference Value Changes  447

Scoped Value Management with User Settings  448
  Scopes, Nodes, and Stored Values  448
  Scope Hierarchy  449
  Accessing User Settings  450
  Change Notification  452

Example Summary  452

Chapter Summary  452
References  453

Chapter 18  Views  455

Views: The General-Purpose Workbench Part  456
  Behavior  456
  Architecture  457
  Function  458

View Implementation  459
  Designing the View  459
  Declaring the View Extension  460
  Creating the View Class  461
  Defining the User Interface  462
  Adding View Actions  463
  Saving and Restoring View State  465
  Opening a Second Instance of a View  466
  Supporting the Show In Request  467
  Implementation Review  468
<table>
<thead>
<tr>
<th>Chapter 19 Editors</th>
<th>483</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Editor Behavior and Architecture</strong></td>
<td>484</td>
</tr>
<tr>
<td><strong>Basic Implementation Steps</strong></td>
<td>485</td>
</tr>
<tr>
<td>Designing the Look of the Editor</td>
<td>485</td>
</tr>
<tr>
<td>Creating a New Wizard for the Editor Input (Optional)</td>
<td>486</td>
</tr>
<tr>
<td>Declaring the Editor Extension</td>
<td>487</td>
</tr>
<tr>
<td>Creating the Editor Class</td>
<td>488</td>
</tr>
<tr>
<td>Instantiating the Model Using the Editor Input</td>
<td>488</td>
</tr>
<tr>
<td>Defining the User Interface</td>
<td>490</td>
</tr>
<tr>
<td>Handling User Modifications of the Editor Input</td>
<td>490</td>
</tr>
<tr>
<td>Handling the Saving of the Editor Input</td>
<td>491</td>
</tr>
<tr>
<td>Associating the Viewer and the Model</td>
<td>492</td>
</tr>
<tr>
<td>Synchronizing the Model and the Editor</td>
<td>492</td>
</tr>
<tr>
<td>Handling User Selections</td>
<td>492</td>
</tr>
<tr>
<td>Defining Editor Actions</td>
<td>493</td>
</tr>
<tr>
<td><strong>Beyond the Basic Implementation Steps</strong></td>
<td>496</td>
</tr>
<tr>
<td>Reusing Specialized Editors of the Rich Client Platform and IDE Workbench</td>
<td>496</td>
</tr>
<tr>
<td>Linking an Editor and the Outline View</td>
<td>498</td>
</tr>
<tr>
<td>Navigating Between Editors and Views</td>
<td>499</td>
</tr>
</tbody>
</table>

| Example Summary | 501 |
| Chapter Summary | 501 |
| References | 502 |

| Chapter 20 Perspectives | 503 |
| **Creating a Perspective** | 503 |
| **Using the perspectiveExtensions Extension Point** | 506 |

| Example Summary | 506 |
| Chapter Summary | 506 |
| References | 507 |
Chapter 21  Action Contributions:
The Integration Fast Track

Action Contribution Extension Points

The Fundamentals of Action Contributions

The Action Contribution API
Specifying Contributions in the Plug-in Manifest File
Specifying Actions in Menus and Toolbars
Using Filters to Refine Where Your Actions Appear
The Class Selection Filter
Creating Elaborate Filters Using Action Expressions

Contributing to the Workbench Menu Bar and Toolbar

Defining Menu and Toolbar Actions Independent of Perspectives

Contributing to the View's Menu and Toolbar

Contributing to an Editor's Menu and Toolbar

Contributing Context Alenues to Views and Editors

Contributing a Context Menu to a View
Contributing a Context Menu to an Editor
Contributing a Context Menu to Selected Objects in a View or Editor

Assigning Accelerator Keys to Your Actions

Example Summary

Chapter Summary

References

Chapter 22  Providing Help

Integrating Your Online Documentation

Creating Help Content as HTML Content
Declaring the Help Extension
Creating Table of Contents Files
Eclipse Online Documentation

Creating Context-Sensitive Help

Declaring the Help Context Extension
Defining the Contents of the Infopop
Associating the Infopop with the UI Context

Running the Standalone Help Infocenter

Customizing Your Help System

Example Summary
Chapter 26 Building a Custom Text Editor with JFaceText

Standard Text Editor Functions 636
- Text Editing and Viewing 636
- Standard Menus and Toolbar Items 637
- Standard Marker Representation 637

Editor Configuration: Customization Points 638
- What Is Content Assist? 639
- What Is Syntax Highlighting? 640
- What Is Content Formatting? 640
- Miscellaneous Customizations 641

Under the Covers of the Text Editor 642
- Choosing a Starting Point for Your Editor 642
- TextViewer Class 643
- AbstractDecoratedTextEditor Class Relationships 644
- Where Do Documents Come From? 644
- Model-View-Controller Relationship 646
- Partitioning a Document 647

Example Summary 649
Chapter Summary 649
References 650

Chapter 27 Extending the Java Development Tools 651
What Is the JDT and What Does It Do? 652
JAVA Model 653
Accessing Reusable JDT Functionality 655
- JavaCore Class 655
- JavaUI Class 660
- Tool Factory Class 661
- SearchEngine Class 662

Compiling Java Source Code 662
Analyzing Java Source Code 662
- JDT Abstract Syntax Tree 663
- Finer-Grain Parsing of Java Source Code 667

Manipulating Java Source Code 671
- Simple Source Code Manipulations using ISourceManipulation 672
- Modest Source Code Manipulations Using IBuffer 672
- Complex Source Code Manipulations Using AST Rewriting 675
Internationalization Steps
- Move Translatable Strings into *.properties Files
- Separate Presentation-Dependent Parameters
- Use Proper Locale-Sensitive Data Formatting and Substitutions APIs
- Test in Your Domestic Language
- Create Initial Translated Plug-in Fragments
- Eclipse Language Pack JAR

Accessibility

Example Summary

Chapter Summary

References

Chapter 32 Performance Tuning

Why Eclipse Should Start Quickly

Diagnosing Startup Problems Using the Runtime Spy
- Installing the Core Tools
- Spying on the Workbench Development
- Spying on the Runtime Workbench
- Where to Go from Here

Case Study: Improving IBM WebSphere Studio’s Startup
- Helping Eclipse Start Quickly
- Reintroducing the Runtime Spy
- Spying on WebSphere Studio

Avoiding Performance Regressions Using the Performance Monitor
- Installation
- Example Use

Common Solutions to Eclipse-Specific Performance Problems

Chapter Summary

References

Chapter 33 Swing Interoperability

Embedding AWT and Swing in Views and Editors
- Embedding Swing/AWT in aView

Launch-and-Edit Integration
- Modes of Invocation
- Improving the End User Experience
- General Integration Issues
Exercise 6 Developing Your First Plug-in

Exercise Setup

Section 1: "Hello, World" in Five Minutes or Less

Section 2: "Hello, World" with Detailed Step-by-Step Instructions

Section 3: Testing with the Runtime Workbench

Section 4: Debugging with the Runtime Workbench

Section 5: Defining a Feature to Support the Plug-in

Section 6: Deploying a Feature and Its Associated Plug-ins

Section 7: Exploring (and Sometimes Correcting) the Eclipse Platform Code

Section 8: Correcting Common Problems

Exercise Activity Review

Exercise 7 Developing Your First Rich Client Application

Exercise Setup

Getting Started

Section 1: Creating an Application Extension

Section 2: Creating a Perspective Extension

Section 3: Creating a Workbench Advisor

Section 4: Implementing the Application's Main Program

Section 5: Running and Testing Your RCP Application

Section 6: Configuring Workbench Window User Interface Elements

Section 7: Adding Menu and Toolbar Actions

Section 8: Deploying Your Application

Section 9: Adding "Optional" Components

Exercise Activity Review

Exercise 8 Developing a Rich Client Application with Dynamically Added Plug-ins

Exercise Setup

Section 1: Create Your RCP Application Launch Configuration

Section 2: Implement Your Plug-in Life Cycle startQ Method

Section 3: Implement Your Configurator

Section 4: Develop a Simple Plug-in Discovery Mechanism

Section 5: Installing the Bonus Pack Dynamically

Exercise Activity Review