

Stefan Reichert

Eclipse RCP im Unternehmenseinsatz

**Verteilte Anwendungen entwerfen, entwickeln,
testen und betreiben**



dpunkt.verlag

Stefan Reichert
stefan@wickedshell.net

Lektorat: René Schönfeldt
Copy-Editing: Annette Schwarz, Ditzingen
Satz: Science & More, www.science-and-more.de
Herstellung: Nadine Thiele
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: Koninklijke Wöhrmann B.V., Zutphen, Niederlande

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;
detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

ISBN 978-3-89864-573-7

1. Auflage 2009
Copyright © 2009 dpunkt.verlag GmbH
Ringstraße 19
69115 Heidelberg

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

5 4 3 2 1 0

Für Steffi

Inhaltsverzeichnis

Einführung	1
I Architektur	5
1 Aspekte der Architektur verteilter Anwendungen	7
1.1 Architekturvarianten	7
1.1.1 Client-Server-Architektur	8
1.1.2 3-Schichten-Architektur	9
1.1.3 Mehrschichten-Architektur	10
1.1.4 Architekturen für EERCP-Anwendungen.....	11
1.2 Entwurfsprinzipien	11
1.2.1 Modularisierung	11
1.2.2 Abhängigkeit und Hierarchie	14
1.2.3 Serviceorientierung	16
1.3 Funktionale Kernbestandteile	16
1.3.1 Datenerfassung	17
1.3.2 Datenvalidierung	19
1.3.3 Datenverarbeitung	21
1.3.4 Datenzugriff und -haltung.....	22
1.3.5 Visualisierung der Daten	23
1.4 Kommunikation	23
1.4.1 Kommunikation in verteilten Systemen	23
1.4.2 Kommunikationsprotokolle.....	25
1.4.3 Eignung von Protokollen für die Kommunikation.....	28
2 Komponenten	29
2.1 Komponenten in Eclipse	30
2.1.1 Technische Komponenten	30
2.1.2 Der Begriff Bundle	30
2.1.3 Bundletypen.....	31
2.1.4 Bundles und Features	32

2.2	Strukturierungsprinzipien	32
2.2.1	Fachliche Bundles: Der richtige Schnitt.....	33
2.2.2	Technische Bundles: Querschnittsfunktionalität	33
2.3	Komponentenbildung	34
2.3.1	Bundle-Struktur	34
2.3.2	Packagestruktur.....	36
2.3.3	Public und Private API.....	37
2.4	Versionierung von Bundles.....	38
2.4.1	Versionierungsschemata	38
2.4.2	Versionsnummern vergeben.....	39
2.4.3	Verschiedene Versionen des gleichen Bundles verwenden	40
2.5	Bundle Classloading	41
2.5.1	Standard-Classloading	41
2.5.2	ClassLoader Hell	41
2.5.3	Beispiel	43
3	UI-Architektur	51
3.1	Eclipse-Oberflächen	52
3.1.1	Hauptfenster	53
3.1.2	Menüzeile	53
3.1.3	Toolbar	53
3.1.4	Statuszeile	53
3.1.5	Editoren	53
3.1.6	Views.....	54
3.1.7	Perspektiven	54
3.2	Oberflächenmetaphern	55
3.2.1	Die Workbench-Metapher	55
3.2.2	Die Desktop-Metapher	57
3.2.3	Die richtige Metapher wählen	58
3.3	UI-Varianten	59
3.3.1	Workbench mit Perspektiven	59
3.3.2	Workbench mit einer Perspektive	61
3.3.3	Eclipse Riena.....	62
3.4	Oberflächenkonzept	62
3.4.1	Aufbau der Oberfläche	63
3.4.2	Inhalt der Oberfläche	64
3.4.3	Oberflächen und Rollen	65
4	Referenzanwendung	67
4.1	Dysis, eine Zeiterfassung.....	67
4.2	Fachliche Beschreibung der Anwendung	68

4.3	Anwendungsfälle	69
4.3.1	Am System anmelden	69
4.3.2	Vom System abmelden	69
4.3.3	Stammdaten erfassen	70
4.3.4	Stammdaten bearbeiten	70
4.3.5	Zeiten buchen	70
4.4	Datenmodell	70
4.5	Architektur	71
4.5.1	Fachlicher Schnitt	72
4.5.2	Umsetzung der Clientseite	73
4.5.3	Umsetzung der Serverseite	74
4.5.4	Kommunikation	75
4.6	Installation	75
4.6.1	Das Dysis-Projekt	76
4.6.2	Hostnamen konfigurieren	76
4.6.3	Die Datenbank einrichten	77
4.6.4	Den Server einrichten	77
4.6.5	Den Client installieren	78

II Implementierung 79

5	Enterprise UI	81
5.1	Login	81
5.1.1	Authentifizierung	82
5.1.2	Automatische Authentifizierung	82
5.1.3	Sprachauswahl	85
5.2	Navigation	87
5.2.1	Strukturen für Funktionen	88
5.2.2	Funktionsgruppen darstellen	89
5.3	Tabellen und Bäume	91
5.3.1	Datenmengen darstellen	91
5.3.2	Datenmengen laden	92
5.4	Suche	95
5.4.1	Schema einer Suche	95
5.4.2	Generische Suche	96
5.4.3	Grafischer Aufbau	97
5.4.4	Funktionales Schema	100
5.5	Nebenläufigkeit	101
5.5.1	Serviceaufrufe auslagern	101
5.5.2	Das UI synchronisieren	102
5.5.3	Controller-Objekte einsetzen	103

5.6	Wizards einsetzen	111
5.6.1	Das New-Wizard-DropDown-Menü	111
5.6.2	Wizards an Perspektiven binden	112
5.6.3	Perspektiven an Wizards binden	113
5.7	UI-Elemente maskieren	114
5.7.1	Datumsfelder darstellen	115
5.7.2	Textfelder maskieren	115
6	Ressourcenschonende Implementierung	117
6.1	Betriebssystemressourcen und Speicher	117
6.1.1	SWT	117
6.1.2	Handles	118
6.1.3	UI-Elemente	118
6.1.4	Weitere Betriebssystemressourcen	120
6.1.5	Speicherbedarf	120
6.2	Profiling	124
6.2.1	Überwachung der Handles	124
6.2.2	Speicherverbrauch	125
6.3	Optimierungspotenziale	126
6.3.1	Systemressourcen wiederverwenden	127
6.3.2	Clientseitig cachen	127
6.3.3	UI-Designrichtlinien	127
7	Databinding	129
7.1	Idee und Grundgedanke	129
7.1.1	Komplexe Verbindungen	130
7.1.2	MVC – Model-View-Controller	130
7.1.3	Synchronisation	131
7.2	Konzept der Synchronisation	132
7.2.1	Transformation	132
7.2.2	Validierung	133
7.2.3	Phasen der Synchronisation	133
7.2.4	Synchronisationsarten	133
7.3	Validierung	134
7.3.1	Validierungstypen	134
7.3.2	Ergebnis der Validierung	135
7.3.3	Einbindung der Validierung	137
7.4	Eclipse Databinding	137
7.4.1	DataBindingContext	137
7.4.2	IObservableValue	137
7.4.3	Synchronisationsarten	138
7.4.4	Konvertierung	139
7.4.5	Validierung	139
7.4.6	Darstellung	141

8	Hilfe	149
8.1	Eclipse-Online-Hilfe	149
8.1.1	Abhängigkeiten	150
8.1.2	Aufbau	151
8.1.3	Online-Hilfen erstellen	152
8.1.4	Online-Hilfe einbinden	161
8.2	Kontextsensitive Hilfe	162
8.2.1	Dynamic Help Bundles	162
8.2.2	Dynamic Help erstellen	163
8.3	Cheat Sheets	165
8.3.1	Cheat Sheet Bundles	166
8.3.2	Cheat Sheets erstellen	167
8.3.3	Cheat Sheets integrieren	167
9	Remoting	171
9.1	Remoting-Mentalitäten	172
9.1.1	Transparentes Remoting	172
9.1.2	Service Remoting	172
9.1.3	Rahmenbedingungen	172
9.2	Einbindung in den Client	173
9.3	Spring Remoting	174
9.3.1	Konfiguration des Service	174
9.3.2	Konfiguration des Clients	177
9.3.3	Service Proxys verwenden	178
9.3.4	Das ServiceLocator-Entwurfsmuster	179
9.3.5	Service Proxys im OSGi-Kontext	182
9.3.6	Clientseitige Dependency Injection	184
9.4	OSGi Remoting	188
10	Nichtfunktionale Anforderungen	191
10.1	Verfügbarkeit	191
10.1.1	Clustering	191
10.1.2	Offline-Fähigkeit	192
10.2	Performance	193
10.2.1	Granularität der Kommunikation	194
10.2.2	Datentransport	194
10.2.3	Serverseitiges Caching	196
10.2.4	Clientseitiges Caching	196
10.3	Sicherheit	203
10.3.1	Authentifizierung	203
10.3.2	Autorisierung	209
10.3.3	Datensicherheit	212
10.3.4	Codesicherheit	215

11	Continuous Integration	217
11.1	Automatisiertes Bauen	217
11.1.1	Produktexport aus dem UI	218
11.1.2	PDE-Build-Skript	218
11.1.3	Produktexport mit Ant	220
11.1.4	Automatisierter Produktexport	224
11.2	Automatisiertes Testen	228
11.2.1	JUnit-Plugin-Tests	228
11.2.2	JUnit-Plugin-Tests aus dem UI ausführen	230
11.2.3	Anwendungen für Tests starten	232
11.2.4	Automatisierte JUnit-Plugin-Tests	235
11.2.5	Oberflächen testen	241
11.3	Continuous Build Server	242
III Betrieb		245
12	Deployment	247
12.1	Eclipse RCP als Client	247
12.2	Softwareverteilung	248
12.2.1	Rahmenbedingungen auf dem Zielsystem	248
12.2.2	Softwareverteilungssysteme	250
12.2.3	Installation via Java Web Start	250
12.2.4	Installation via NSIS	256
12.3	Softwareaktualisierung	256
12.3.1	Die Update-Problematik	256
12.3.2	Externe Softwareaktualisierung	258
12.3.3	Den Eclipse Update Manager verwenden	258
13	Monitoring	263
13.1	Logging und Tracing	263
13.1.1	Eclipse Log	264
13.1.2	Tracing	267
13.1.3	Logging-Frameworks	270
13.1.4	Einheitliches Logging	270
13.1.5	End-to-End Logging	273
13.2	Keep-alive-Meldungen	274
13.2.1	Session Handling	274
13.2.2	Time-out	274
13.2.3	Logout	275
13.2.4	Keep-alive	275
Literaturverzeichnis		277
Index		279

Einführung

Individualsoftware stellt nach wie vor ein wesentliches Element in Enterprise-Architekturen von Unternehmen dar. Die Vielzahl der Produkte und Branchenlösungen, wie beispielsweise ERP-Systeme oder Software für Finanzunternehmen, sind zwar in einem gewissen Rahmen anpassbar oder können als Basis für eine Neuentwicklung verwendet werden. In vielen Fällen bedingt jedoch ein Katalog aus spezifischen fachlichen Anforderungen die Konzeption und Umsetzung einer gänzlich individuellen Softwarelösung.

Individualsoftware

Neben den fachlichen Anforderungen spielen auch technische Anforderungen an die Software eine ganz wesentliche Rolle. Dabei existieren sowohl interne als auch externe technische Anforderungen. Interne Anforderungen ergeben sich aus der Software selbst heraus, stehen also in direkter Verbindung mit den funktionalen und nichtfunktionalen Anforderungen. Daneben gibt es auch externe Anforderungen, die ihren Ursprung in der Enterprise-Architektur oder in unternehmensspezifischen Konventionen haben. Die Software bzw. deren Architektur muss in jedem Fall beide Arten von Anforderungen hinreichend erfüllen.

Aus unternehmerischer Sicht ist es verständlicherweise sehr reizvoll, trotz hochgradig unterschiedlicher fachlicher Anforderungen trotzdem auf technischer Ebene nach wiederverwendbaren Lösungen zu suchen. Die Entwicklungsdauer von Software kann so deutlich reduziert werden. Die Ansätze für eine Beschleunigung der Softwareentwicklung lassen sich grob in die Kategorien Entwicklungsprozess und Standardarchitekturen unterteilen. In Bezug auf den Entwicklungsprozess hat beispielsweise die modellgetriebene Entwicklung (engl. Model Driven Development) einen starken Impuls zur Veränderung gegeben. In Bezug auf Standardarchitektur wurden in den letzten Jahren häufig Webanwendungen als eine Art technische Patentlösung gehandelt.

Wiederverwendung

Mit Ajax und Java ServerFaces (JSF) ist es zweifellos möglich, solide Anwendungen zu entwerfen und zu bauen. Allerdings stoßen diese Techniken häufig bei komplexen Anforderungen an ihre Grenzen. In manchen Fällen wird diese Grenze bereits in der Entwicklungsphase sichtbar, manchmal taucht sie erst später im Betrieb der Software auf. Die Gründe sind dabei vielschichtig und keineswegs zu verallge-

Webtechniken

meinern. Sie reichen von mangelnder Toolunterstützung über fehlende Browser-Standards bis hin zu einer schwer umzusetzenden ergonomischen Benutzerführung.

*Renaissance des
Rich Clients*

So vollzieht sich in der letzten Zeit eine Art Renaissance des Rich Clients, mehr noch, schon warnen die ersten Stimmen vor einem erneuten Hype. Nüchtern betrachtet stellen Rich-Client-Plattformen als Basis für die Entwicklung von Anwendungen in keinem Fall einen vollständigen Ersatz für Webtechnologien dar. Sie bilden aber bei entsprechenden Anforderungen eine sinnvolle Alternative, die im Auswahlprozess der geeigneten Techniken vorbehaltlos berücksichtigt werden sollte.

Enterprise Eclipse RCP

Die Eclipse-Plattform hat in den vergangenen Jahren eine bemerkenswerte Entwicklung erlebt. Als reine Entwicklungsumgebung bekannt und anerkannt geworden, etablierte sie sich mit Eclipse RCP zu einer echten Rich-Client-Plattform für allerlei Arten von unterschiedlichen Anwendungen. Ein wesentlicher Faktor für die Akzeptanz war und ist der modulare Aufbau und die damit einhergehende Erweiterbarkeit. Die Möglichkeit, die Anwendung mit eigenen Plug-ins spezifisch zu erweitern, macht einen großen Teil der Attraktivität der Eclipse-Plattform aus. Darüber hinaus überzeugt die Reichhaltigkeit und Reife der plattforminternen Funktionen wie das Jobs API, JFace Viewer, JFace Data-binding oder das Workbench UI.

EERCP-Anwendungen

Der Umstand, dass diese bewährten Techniken der Eclipse-Plattform auch für eigene individuelle Softwarelösungen verwendet werden können, rückt Eclipse RCP auch in den Fokus für Enterprise-Anwendungen, also mehrschichtige verteilte Anwendungen. Enterprise-Anwendungen, die Eclipse RCP für die Präsentationsschicht einsetzen, nenne ich im Folgenden Enterprise-Eclipse-RCP-Anwendungen (EERCP-Anwendungen). Typischerweise tauchen hier aufgrund der Verteilung Entwicklungseigenheiten auf, die sich von der reinen Plug-in-Entwicklung für die Eclipse IDE unterscheiden. Relevant sind dabei unter anderem die Unterschiede des Benutzertyps einer Enterprise-Eclipse-RCP-Anwendung und einer IDE. Die Anforderungen eines Benutzers an individuelle Softwarelösungen unterscheiden sich in puncto Bedienbarkeit, Erweiterbarkeit, Ergonomie und Flexibilität aufgrund der fachlichen Komplexität meist deutlich von einer IDE. Eine solche Enterprise-Eclipse-RCP-Anwendung implementiert zudem üblicherweise eine Kommunikation mit einem entfernten Server, sodass auch Latenz, Kontrolle über die ausgetauschte Datenmenge und Sicherheitsaspekte ein großes Gewicht bei der Implementierung haben.

Warum dieses Buch?

Ich arbeite nun schon länger mit Eclipse RCP als Frontend für Enterprise-Anwendungen. Erfreulicherweise steht über die reine Entwicklung von Eclipse-RCP-Anwendungen eine große Menge an guter Literatur zur Auswahl. Viele dieser Bücher konzentrieren sich allerdings fast ausschließlich auf die Entwicklung eines Eclipse-RCP-Clients und die vielen umfangreichen Features der Eclipse-Plattform. Der Aspekt der Verteilung in einem Enterprise-Kontext wird wenig oder gar nicht berücksichtigt.

*Verteilung im
Enterprise-Kontext*

Dieses Buch hat den Anspruch, diese Lücke zu füllen und für diese speziellen Anforderungen einer Individualsoftware Lösungsansätze und Beispiele zu geben. Die Herausforderung besteht darin, ein Buch über die Entwicklung von Individualsoftware zu schreiben, was per Definition schon schwierig ist. Der Anspruch ist also, mehr oder weniger allgemeingültige Anforderungen und Systematiken so zu erörtern, dass die dafür passenden Lösungen auch in der von Ihnen eingesetzten Umgebung mit den von Ihnen verwendeten Techniken adaptierbar sind. Die Lösungsansätze in diesem Buch sind selbstverständlich mit einem von mir ausgewählten Technologiestack entwickelt. Dennoch sind sie so allgemein beschrieben, dass es problemlos möglich ist, sie auch mit anderen Techniken zu übernehmen.

Themen und Aspekte

Dieses Buch greift wesentliche Punkte der Entwicklung einer Enterprise-Eclipse-RCP-Anwendung auf. Dabei geht es zum einen um Eclipse-spezifische Lösungen und Lösungsansätze für typische Anforderungen der Individualsoftwareentwicklung. Die Plattform bietet hier bereits einige Built-in-Features, in anderen Fällen kann man sich Eclipse-RCP-Basistechnologien zunutze machen. Des Weiteren erläutere ich wichtige architektur-spezifische Aspekte einer Enterprise-Eclipse-RCP-Anwendung. Solche Aspekte sind beispielsweise die Kommunikationsstruktur, die Komponentenbildung oder die UI-Struktur. Die vorgestellten Themen und Ansätze haben sich in der Praxis bewährt und können Ihnen helfen, Ihre Enterprise-Eclipse-RCP-Anwendung flexibel und robust zu entwerfen, zu entwickeln, zu testen und zu betreiben.

*Eclipse- und
architekturspezifische
Lösungen*

Ich habe absichtlich darauf verzichtet, Eclipse-RCP-Grundwissen in dieses Buch mit aufzunehmen. Ich setze daher entsprechende Grundkenntnisse voraus. Es ist also unerlässlich, dass Sie entweder bereits eingehende Kenntnisse und Erfahrungen in der Entwicklung von Plug-ins bzw. Eclipse-RCP-Anwendungen haben oder alternativ entsprechende

Literatur zu Rate ziehen. Für die Eclipse-RCP-Grundlagen existieren einige exzellente Bücher sowohl in englischer als auch in deutscher Sprache, die das nötige Wissen recht schnell vermitteln.

Aufbau und Gliederung

*Architektur,
Implementierung,
Betrieb*

Das Buch ist in drei Teile gegliedert, die sich am Softwareentwicklungsprozess orientieren. Der erste Teil setzt sich mit architektonischen Aspekten der Anwendung auseinander. Hier wird beschrieben, welche Dinge in der Konzeptphase beim Entwurf des Clients berücksichtigt werden müssen. Der zweite Teil geht auf implementierungsrelevante Themen ein. Dabei geht es um konkrete Punkte bei der Umsetzung der funktionalen und nicht-funktionalen Anforderungen der Anwendung. Mit dem Betrieb von Eclipse-RCP-Anwendungen im Enterprise-Umfeld beschäftigt sich der dritte Teil. Hier gehe ich insbesondere auf Deployment und Monitoring des Eclipse-RCP-Clients ein.

*Eine
Beispielanwendung*

Dieses Buch beinhaltet eine Menge an Codeauszügen, um Ihnen die einzelnen Themen an einem praktischen Beispiel zu erläutern. Diese Auszüge stammen aus einer Referenzanwendung namens Dysis, die ich im Rahmen der Arbeit zu diesem Buch entwickelt habe. Dysis dient zur Veranschaulichung der im Buch aufgeführten Lösungen im Gesamtkontext einer Enterprise-Eclipse-RCP-Anwendung und ist als Open-Source-Projekt bei Sourceforge.net verfügbar [27]. Das Kapitel 4 liefert hierzu eine detaillierte Beschreibung und eine Installationsanleitung.

Dankeschön

Mein erster Dank geht an Peter Friese, ohne den dieses Buch zweifellos nicht entstanden wäre. Ich danke Dir für Deine vielen Ideen und Texte, die dieses Buch grundlegend mitgestaltet haben. Als wir vor gut drei Jahren das erste Mal über eine Gliederung für dieses Buch geflachat haben, hätte ich das, was sich jetzt daraus entwickelt hat, niemals für möglich gehalten.

Bedanken möchte ich mich auch bei René Schönfeldt vom dpunkt.verlag, der mit sehr viel Geduld, Unterstützung und wertvollen Ratschlägen zum Gelingen dieses Buchs beigetragen hat.

Mein nächster Dank geht an meine Frau Steffi. Ich weiß, dass ich in der letzten Zeit lediglich physisch anwesend war. Danke für Deine Ruhe und Nachsicht. Ich bin überglücklich, nun wieder komplett an unserem wunderschönen Leben teilnehmen zu dürfen.

Zu guter Letzt danke ich dem Mädels und den Jungs: Josephine, Napoleon, Cuno und Dante. Ihr wart mir in der Zeit des Schreibens und Tüftelns eine wirklich traumhafte Gesellschaft.