



ABAP Development Tools (ADT) in Eclipse

Leitfaden

**JETZT
NEU**

Leitfaden
auch auf
GitHub
verfügbar



<https://1dsag.github.io/ADT-Leitfaden/>

Stand: März 2023

Inhaltsverzeichnis

Inhaltsverzeichnis	2
Abbildungsverzeichnis	6
Tabellenverzeichnis	13
Einleitung	14
Warum ein DSAG-Leitfaden über ABAP Development Tools (ADT) in Eclipse?	14
Motivation und Aktualität des Leitfadens.....	14
Gender-Erklärung	15
1 Einführung Eclipse und ADT	16
1.1 Warum eigentlich Eclipse?	16
1.2 Ein kurzer Blick auf die Entstehungsgeschichte	16
1.3 Das Prinzip von Erweiterungen	16
1.4 Unterschied zwischen Perspektive und Sicht	17
1.5 Die Macht der Tastenkürzel.....	17
1.6 Ab wann ist welche Funktion mit Eclipse verfügbar?.....	18
2 Motivation für ADT	19
2.1 Sie möchten modernes ABAP anwenden und umsetzen.	19
2.2 Sie möchten eine Entwicklungsumgebung für alles nutzen.....	20
2.2.1 Technologischer Aspekt.....	20
2.2.2 Entwicklung auf mehreren Systemlinien	20
2.3 Sie möchten sich von technischen Einschränkungen des SAP GUI lösen.	20
2.3.1 Ihnen sind sechs Sessions in einem System zu wenig.....	20
2.3.2 Sie möchten durch Netzwerkunterbrechungen keinen Code verlieren.....	21
2.4 Sie verbessern Ihren ABAP-Code gerne durch Refactoring.....	21
2.5 Sie passen die Darstellung Ihrer Entwicklungsumgebung gerne Ihren Wünschen an.	22
2.6 Sie möchten noch mehr Tool-basierte Unterstützung?	23
2.7 Sie kennen Eclipse bereits von anderen Programmiersprachen. Prima!.	23
2.8 Sie haben Interesse an Neuem.....	23
2.9 Die Vorteile für die Organisation.....	24
2.10 Ihnen fehlt der formularbasierte Editor	25
2.11 Warum ABAP Development Tools	26
3 Arbeiten mit ADT	28

3.1	Einführung: Grundlagen der Arbeit mit ADT	28
3.1.1	Der Einstieg in das Arbeiten mit den ABAP Development Tools ..	28
3.1.2	Der Umstieg von der formularbasierten zur textorientierten Code-Erstellung	29
3.1.3	Kontext des hier gezeigten Übungsbeispiels:	30
3.1.4	Verbinden des Entwicklungssystems – Neues Projekt	30
3.1.5	Das Erstellen einer Klasse im Textmodus	33
3.1.6	Definition einer Methode in der Klasse	36
3.1.7	Automatische Ergänzung und Formatierung des Codes.....	38
3.1.8	Implementierung der Methode mittels Quick-Fix.....	40
3.1.9	Umbenennung von Parametern – Refactoring.....	42
3.2	Funktionen von ADT.....	47
3.2.1	Übergreifende Features.....	47
3.2.2	Suche und Navigation.....	55
3.2.3	ABAP Editor.....	59
3.2.4	Andere Objekttypen	65
3.2.5	ABAP Views.....	66
3.2.6	Refactoring von Code mit ADT	93
3.2.7	Versionsverwaltung und Vergleichen.....	95
3.2.8	Dokumentation mit den ABAP Doc	98
3.2.9	Ausführen von Source-Code.....	100
3.2.10	Data Preview	103
3.2.11	Core Data Services.....	105
4	Troubleshooting-Werkzeuge in Eclipse.....	112
4.1	Troubleshooting mit den ABAP Development Tools.....	112
4.2	Der Debugger in den ABAP Development Tools.....	113
4.2.1	Breakpoints und Soft-Breakpoints	113
4.2.2	Debugging-Perspektive.....	113
4.2.3	Besonderes Verhalten im Debugger	114
4.2.4	Weitere Informationen.....	114
4.3	Checkpoint IDs und dynamische Logpoints.....	115
4.4	Performance-Analyse	118
4.4.1	Hinweis zu HANA Studio und SAP HANA Tools.....	128
4.5	Feed Reader	129

4.6	Doku-Links	132
5	Installation, Verteilungs- und Update-Strategien	133
5.1	Abgrenzungen	133
5.1.1	Installation Guide von SAP	133
5.2	Vorbereitungen	133
5.2.1	Java Development Kit und Java Runtime Environment	134
5.2.2	Backend	134
5.2.3	SAP-GUI-Installation	135
5.2.4	Visual Studio Redistributable	135
5.3	Technischer Aufbau einer Eclipse-Installation	135
5.4	Plug-ins	136
5.4.1	Eclipse Marketplace	137
5.4.2	Update-Site	138
5.5	Installations- und Verteilungsstrategien	140
5.5.1	Übersicht und Vergleich	140
5.5.2	Manuelle Installation	142
5.5.3	Vorkonfigurierte Erstinstallation	142
5.5.4	Eclipse Installer	143
5.6	Fehlersituationen	160
5.6.1	Fehler beim Update oder Upgrade	160
5.6.2	Single-sign-on-Bibliotheken	161
5.6.3	“No repository found containing”	162
5.6.4	PKIX - Certificate Error	162
5.6.5	macOS aarch64 support & SAP GUI for Java	162
5.6.6	Offline-Installation – Download ADT-Abhängigkeiten	162
6	Best Practices Eclipse-Konfiguration	163
6.1	Einstellungen in Eclipse	163
6.1.1	Globale Einstellungen	163
6.1.2	Projektspezifische Einstellungen	169
6.2	Views und Perspektiven	171
6.2.1	Views	171
6.2.2	Perspektiven	174
6.3	Empfohlene zusätzliche Views	176
6.4	Vorschläge zur Verteilung	177

7	Plug-ins	178
7.1	Einführung.....	178
7.2	Nützliche Open-Source-Plug-ins	178
7.2.1	Open Editors.....	178
7.2.2	AnyEdit Tools.....	179
7.2.3	PDE Tools.....	181
7.3	Nützliche Open Source ADT Plugins.....	182
7.3.1	ABAP Favorites	182
7.3.2	ABAP Continuous Integration	184
7.3.3	ABAP ADT Extensions.....	185
7.3.4	ADT Classic Outline.....	188
7.3.5	ABAP Quick Fix	189
7.3.6	ABAPQuickFixS4Conversion.....	190
7.3.7	ABAP Tags	192
7.3.8	ABAP Search and Analysis Tools	194
7.3.9	ABAP Code Search	195
7.3.10	abapGit Eclipse Plug-in.....	197
7.4	Eigene ADT Plug-ins entwickeln	197
7.4.1	Voraussetzungen.....	197
7.4.2	Einrichtung der Entwicklungsumgebung.....	198
7.4.3	Wichtige Konzepte/Artefakte.....	200
7.4.4	Erstellung eines Plug-in-Projektes.....	200
7.4.5	Erstellung eines Feature-Projektes.....	204
7.4.6	Erstellung einer Update-Site.....	206
7.4.7	Erweiterung des ADT Backends mit ABAP Code	209
7.4.8	Java Code Snippets für wiederkehrende Aufgaben in ADT	211
7.4.9	Projekt-Set-up mit Maven	215
	Autoren	216
	Impressum	220

Abbildungsverzeichnis

Abbildung 1 Erstellung eines ABAP-Projekts in Eclipse	31
Abbildung 2 Der Project Explorer	31
Abbildung 3 Hinzufügen von Packages zu den Favoriten	32
Abbildung 4 Erstellen einer neuen ABAP-Klasse im Projekt Explorer	33
Abbildung 5 Eigenschaftsdialog: Erstellung ABAP-Klasse	34
Abbildung 6 Transportauftragsdialog.....	35
Abbildung 7 Anzeige der neuen Klasse in den ADT	36
Abbildung 8 Bearbeiten der Klasse	37
Abbildung 9 Beispiel Code Completion für den Import-Parameter	39
Abbildung 10 Nutzung des Quick Fix zur Methodenimplementierung	40
Abbildung 11 Auswahl der Komponente mittels Code Completion	42
Abbildung 12 Umbenennung von Methodenparametern	44
Abbildung 13 Abfragedialog zum Speichern des Code (File: 2023_01_Save_flight.png)	44
Abbildung 14 Eingabe neuer Parametername	45
Abbildung 15 Auswahl des Transports und Optionen.....	45
Abbildung 16 Vorschau der Umbenennung.....	46
Abbildung 17 Abfrage des Workspace-Verzeichnisses	47
Abbildung 18 Wechseln des Workspace	49
Abbildung 19 Der Workspace-Dialog	50
Abbildung 20 Hinzufügen von Packages zu den Favoriten	51
Abbildung 21 Detailbild Project Explorer mit Buttonleiste.....	52
Abbildung 22 Working-Sets-Einstellungen	53
Abbildung 23 Anlage und Bearbeitung der Working Sets.....	53
Abbildung 24 Zuordnung Projekt zu Working Set.....	54
Abbildung 25 Einstellung der Projekt-Explorer-Anzeige	54
Abbildung 26 Darstellung Projekte in Working Sets	55
Abbildung 27 Dialog zur Objektsuche	56
Abbildung 28 Ergebnis der Suche	57
Abbildung 29 Anzeige weiterer Suchoptionen	57
Abbildung 30 Ergebnis mit Objekt- und Typfilter	58

Abbildung 31 Navigations-Ikonen.....	59
Abbildung 32 Aktionen für den Project Explorer.....	59
Abbildung 33 ABAP Editor – Hauptfenster.....	60
Abbildung 34 Element Info für eine Methode.....	60
Abbildung 35 Element Info für ein Datenelement.....	61
Abbildung 36 Anzeige der Element Info nach Auswahl des Objekts.....	62
Abbildung 37 Kontextmenü für die Formatierung.....	63
Abbildung 38 Einstellungen für den ABAP Formatter.....	64
Abbildung 39 Anzeige der Refactoring Optionen.....	65
Abbildung 40 Vergleich Funktionsgruppen im Project Explorer der ADT und in der SE80.....	66
Abbildung 41 Anzeige der Eigenschaften in den Outlines.....	67
Abbildung 42 Anzeige der Meldungen im Problems View.....	67
Abbildung 43 Anzeige der Optionen des Views.....	68
Abbildung 44 Konfiguration der angezeigten Punkte.....	69
Abbildung 45 Properties View.....	70
Abbildung 46 Historie der Transporte.....	70
Abbildung 47 Transport View.....	71
Abbildung 48 Template View Browser.....	72
Abbildung 49 Auswahl der Templates im Content Assist.....	73
Abbildung 50 Ergebnis des Verwendungsnachweises/Where-Used-List.....	74
Abbildung 51 Filter für Where-Used-Search.....	75
Abbildung 52 Menü zur Erstellung des Bookmarks.....	76
Abbildung 53 Eingabe eines Namens (Bookmark).....	76
Abbildung 54 Darstellung eines Bookmarks im Quellcode.....	77
Abbildung 55 Bookmarks View.....	77
Abbildung 56 Teilen des Quellcodes als Link (Kontextmenü).....	78
Abbildung 57 Dialog zum Teilen des Links.....	79
Abbildung 58 Öffnen der ABAP Type Hierarchie.....	80
Abbildung 59 Anzeige der Type Hierarchy im View.....	80
Abbildung 60 Transport Organizer View.....	81
Abbildung 61 Darstellung eines Laufzeitfehlers.....	81
Abbildung 62 Beispiel für Systemmeldung.....	82

Abbildung 63 Ausführung des ABAP Unit Test über das Kontextmenü	82
Abbildung 64 Ergebnisanzeige des ABAP Unit Test	83
Abbildung 65 Neustart der Ausführung	83
Abbildung 66 Dialog zur Einstellung der Ausführung von ABAP Unit Tests	84
Abbildung 67 Durchführung der Abdeckungsmessung	85
Abbildung 68 Farbliche Hervorhebung von Quellcode nach Unit Test	85
Abbildung 69 Anzeige der Ergebnisse des ATC-Laufs.....	86
Abbildung 70 Beantragung von Ausnahmen über den ATC View	87
Abbildung 71 Dialog zur Klassifizierung der Ausnahme	87
Abbildung 72 Aufruf der ABAP-Sprachhilfe über das Kontextmenü	88
Abbildung 73 ABAP Language Help View.....	89
Abbildung 74 Button-Leiste des Views	89
Abbildung 75 Navigation zum Help Content.....	90
Abbildung 76 Übersicht der verfügbaren Hilfen und Dokumentationen	91
Abbildung 77 Suche in der Hilfe	92
Abbildung 78 Weiterführende Hilfen und Dokumentationen.....	92
Abbildung 79 Darstellung der Navigationspfade	93
Abbildung 80 Kontextmenü zum Vergleichen von Versionen.....	95
Abbildung 81 Comparison View – Vergleich von zwei Versionen	97
Abbildung 82 Kontextmenü zum kompletten Übernehmen einer Version aus der lokalen Versionsverwaltung	97
Abbildung 83 ABAP-Doc Dokumentation der Methode	98
Abbildung 84 Ausführung einer Klasse in SAP GUI	100
Abbildung 85 Ergebnis der Ausführung.....	101
Abbildung 86 Auswahl des Projekts	101
Abbildung 87 Ausgabe in die Console.....	102
Abbildung 88 Starten des Data Preview über die Tabelle	103
Abbildung 89 Anzeige des Data Preview	103
Abbildung 90 Navigation über Assoziationen	104
Abbildung 91 SQL Console	104
Abbildung 92 Objekttypen der CDS in der Navigation.....	106
Abbildung 93 Unterschiedliche Dateien legen die Eigenschaften einer CDS View Entity fest	107

Abbildung 94 Übersicht über ein CDS View Entity mit Hilfe von Element Info	108
Abbildung 95 Aufruf des Dependency Analyzers über das Kontextmenü	108
Abbildung 96 SQL Dependency Tree.....	109
Abbildung 97 SQL Dependency Graph	109
Abbildung 98 Complexity Metrics	110
Abbildung 99 Aktive Annotationen eines Views	111
Abbildung 100 Debugging Perspektive in Eclipse	113
Abbildung 101 Werte der Variablen in der Debugging Perspektive.....	114
Abbildung 102 Erstellung eines Log Points über das Kontextmenü	116
Abbildung 103 Attribute beim Erstellen eines Log Points.....	117
Abbildung 104 Log Points View in der Debugging Perspektive.....	118
Abbildung 105 ABAP Trace Requests in der Debugging Perspektive.....	119
Abbildung 106 Kontextmenü eines Traces.....	120
Abbildung 107 Übersicht über die Eigenschaften eines Traces	121
Abbildung 108 Aggregierte Übersicht eines Trace-Verlaufs.....	121
Abbildung 109 Absprung in den SQL Trace	122
Abbildung 110 Einstieg in die Verwaltung der Benutzerparameter	123
Abbildung 111 Setzen des Benutzerparameters HDB_OPEN_STUDIO.....	123
Abbildung 112 Auswahl der eclipse.exe als neue Möglichkeit zur Anzeige von *.plv Dateien	124
Abbildung 113 Auswahl einer konkreten Selektion	124
Abbildung 114 Download der *.plv Datei.....	125
Abbildung 115 Anzeige des Abfrage-Ausführungsplans	125
Abbildung 116 Erstellung von ABAP Cross Traces.....	127
Abbildung 117 Detaillierte Ansicht der Operationen.....	128
Abbildung 118 Abonnieren populärer RSS Feeds.....	130
Abbildung 119 Mehrere Laufzeitfehler innerhalb eines Feeds	130
Abbildung 120 Ansicht eines SAP Gateway Fehlers aus dem Error Log	131
Abbildung 121 Einstieg in den Eclipse Marketplace	137
Abbildung 122 Exemplarische Suche nach Plug-ins im Eclipse Marketplace	138
Abbildung 123 Installation neuer Software über das Kontextmenü	139
Abbildung 124 Eintragen der Update-Site.....	140
Abbildung 125 Wechsel zwischen Indizes.....	146

Abbildung 126 Hinzufügen und Festlegen von Eigenschaften	147
Abbildung 127 Komponenten einer "minimalen" ADT-Installation	149
Abbildung 128 Komponenten einer minimalen ADT-Installation	150
Abbildung 129 Bestandteile der SAP Update Site	152
Abbildung 130 Möglichkeit zur Anzeige der bereits gespeicherten Einstellungen..	153
Abbildung 131 Bereits gespeicherte Einstellungen	153
Abbildung 132 Wechsel in den "Advanced Mode"	155
Abbildung 133 Wechsel zwischen Indizes.....	156
Abbildung 134 Auswahl der Product-Version	157
Abbildung 135 Auswahl des Streams.....	158
Abbildung 136 Abfrage weiterer Variablen.....	159
Abbildung 137 Einstieg in die globalen Einstellungen	163
Abbildung 138 Einstellung für das Dark-Theme	164
Abbildung 139 Einstellung zur Einrückung des Quellcodes	165
Abbildung 140 Beispiel für die Einstellung	165
Abbildung 141 Ergebnisbild im Quellcode.....	165
Abbildung 142 Farbeinstellungen zur Hervorhebung der Schlüsselworte im Quellcode.....	167
Abbildung 143 Verwaltung der ABAP Templates in den Einstellungen.....	168
Abbildung 144 Einfügen des Templates in den Quellcode.....	168
Abbildung 145 Einstieg in die projektspezifischen Einstellungen	170
Abbildung 146 Mögliche Einstellungen für den Pretty Printer / ABAP Formatter....	171
Abbildung 147 Verschieben des Views über die Bezeichnung/Reiter	171
Abbildung 148 Die Markierungen deuten die Platzierbarkeit des Fensters an	172
Abbildung 149 Darstellung von gestapelten Views	172
Abbildung 150 Minimieren von View-Gruppen	172
Abbildung 151 Wiederherstellung der View-Gruppen.....	173
Abbildung 152 Schließen eines Views	173
Abbildung 153 Einblenden einer View.....	174
Abbildung 154 Wechseln zwischen verschiedenen Perspektiven	174
Abbildung 155 Zurücksetzen einer Perspektive	175
Abbildung 156 Speichern einer Perspektive.....	175
Abbildung 157 Benennung der neuen Perspektive	176

Abbildung 158 Neu Perspektive mit Name.....	176
Abbildung 159 Mögliche Einstellung der ABAP Perspektive	177
Abbildung 160 Mögliche Einstellung der Debugger Perspektive	177
Abbildung 161 Dialog zur Anpassung der Sortierreihenfolge von geöffneten Editoren	179
Abbildung 162 Beispiele für verfügbare Operationen im Kontextmenü	180
Abbildung 163 Historie für Zwischenablage (Tastenkürzel Strg+Shift+V)	181
Abbildung 164 ABAP Favorites View	183
Abbildung 165 Kontextmenü eines Ordners im ABAP Favorites View	184
Abbildung 166 Farbige Hervorhebung der Statusleiste pro Projekt + Teststatus ...	185
Abbildung 167 Verwaltung von Paketen, für die Unittests und/oder ATC-Prüfläufe eingepflegt sind.....	185
Abbildung 168 View zur Verwaltung der hinterlegten Zugangsdaten von ABAP Systemen.....	186
Abbildung 169 Kontextmenü auf Projekt im Project Explorer	187
Abbildung 170 Statusleiste im Eclipse-Fenster	187
Abbildung 171 Classic Outline View.....	188
Abbildung 172 ABAP Code vor Quick-Fix-Ausführung.....	189
Abbildung 173 ABAP Code nach Quick-Fix-Ausführung	189
Abbildung 174 Beispiel für Quick-Fix-Verfügbarkeit bei einer SELECT-Anweisung	191
Abbildung 175 SELECT-Anweisung nach Anwendung des Quick Fix.....	191
Abbildung 176 View "Tag Manager"	192
Abbildung 177 Search-Dialog mit Seite "ABAP Object Search"	193
Abbildung 178 Search-Dialog auf Seite "ABAP Object Search"	194
Abbildung 179 View "CDS Analyzer" - Top-Down-Analyse	195
Abbildung 180 Search-Dialog mit "ABAP Code Search"-Seite	196
Abbildung 181 Eclipse Bundle "Eclipse IDE for RCP and RAP Developers"	198
Abbildung 182 Plug-in Project Wizard – Einstieg	201
Abbildung 183 Plug-in Project Wizard – Inhalt	202
Abbildung 184 Plug-in-Projekt im Project Explorer View	203
Abbildung 185 Feature Project Wizard - Einstieg	204
Abbildung 186 Feature Project Wizard - Plug-in-Auswahl.....	205
Abbildung 187 Update Site Wizard	207

Abbildung 188 Compiler-Einstellungen im Eclipse-Einstellungsdialog	208
Abbildung 189 Dialog zum Hinzufügen einer Update-Site.....	208
Abbildung 190 GitHub-Repository-Einstellungen für GitHub Pages.....	209
Abbildung 191 Beispiel für eine Simple Transformation zur Transformation von ABAP <-> XML.....	210
Abbildung 192 Beispiel für EMF-Modell zur Serialisierung von XML- <-> Java-Objekt	211

Tabellenverzeichnis

Tabelle 1 Vergleich unterschiedlicher Installationsmöglichkeiten.....	141
Tabelle 2 Begrifflichkeiten in Oomph.....	144
Tabelle 3 Wichtigste Elemente.....	147
Tabelle 4 Wichtigste Elemente eines Projects	149
Tabelle 5 Eigenschaften Eclipse Ini Task.....	155
Tabelle 6 Autoren.....	216

Einleitung

Warum ein DSAG-Leitfaden über ABAP Development Tools (ADT) in Eclipse?

Vielleicht geht es Ihnen ja wie einigen von uns, und Sie sind gerade erst dabei, sich mit Eclipse vertraut zu machen? Oder Sie sind schon ein Profi und kennen sich mit den ADT bereits bestens aus, benötigen aber grundlegende Informationen und eine Handreichung, um diese Entwicklungsumgebung auch Ihren Kollegen schmackhaft zu machen. Vielleicht gehören Sie auch zur Gruppe der regelmäßigen Eclipse-Nutzern für eine andere Programmiersprache als ABAP und möchten wissen, was bei ABAP anders läuft als z. B. bei JAVA.

Aus vielen Rückmeldungen im Rahmen von Veranstaltungen des AK-Development wurde schnell klar, dass ein von DSAG-Mitgliedern erstellter Leitfaden in der "Tradition" früherer Leitfäden (z. B. Programmierrichtlinien oder ABAP-Test-Cockpit) auf großes Interesse stoßen würde. Erfreulicherweise haben sich auf den AK-Development-Aufruf im Februar 2022 von Sebastian Freilinger-Huber viele Freiwillige zum Erstellen des Leitfadens gemeldet, so dass die Arbeiten im Laufe des Frühjahrs begonnen werden konnten.

Ein Blick in die Kapitelliste zeigt, dass wir versucht haben, möglichst viele Aspekte des Entwickelns mit den ADT abzudecken und dass sich die einzelnen Kapitel auch an verschiedene Zielgruppen wenden. Picken Sie sich die Themen heraus, die für Sie am interessantesten sind! An entsprechenden Stellen verweisen wir auf weiterführende Quellen, um diesen Leitfaden nicht zu überfrachten, Ihnen aber trotzdem die Möglichkeit zu geben, einfach auf detaillierte und weiterführende Informationen zuzugreifen.

Wir hoffen, dass Sie im Leitfaden unabhängig von Ihren bisherigen Erfahrungen mit Eclipse und ADT die für Sie hilfreichen Informationen finden und wünschen Ihnen viel Vergnügen beim Lesen der verschiedenen Kapitel!

Motivation und Aktualität des Leitfadens

Wissensvorsprung, Einflussnahme und Netzwerk – dies sind die drei wichtigsten Aspekte aus Sicht der deutschsprachigen SAP Anwendergruppe e.V. (DSAG).

Dieser Leitfaden wurde von Mitgliedern des DSAG-Arbeitskreises Development initiiert und adressiert den ersten Aspekt: Wissensvorsprung für Anwender und Partner.

In den letzten Jahren wurde in zahlreichen DSAG-Mitgliedsunternehmen umfangreiches Wissen und Erfahrung im Rahmen der ADT mit Eclipse aufgebaut. Unser Ziel ist es, diese Erkenntnisse der Allgemeinheit in ansprechender Form zur Verfügung zu stellen.

Da die ADT kontinuierlich durch SAP (und auch über Plug-ins der Community) weiterentwickelt werden, kann dieser Leitfaden keinen abgeschlossenen Zustand repräsentieren. Vielmehr soll er als "lebendes Objekt" verstanden werden. Neue Erkenntnisse werden stetig integriert und bereits gewonnenes Wissen aktualisiert. Aus diesem Grund wird neben einer veröffentlichten Version 1.0 (PDF) ein kontinuierlich aktuell gehaltenes [Git Repository](#) zur Verwaltung der Inhalte bereitgestellt. Sie können die bereits etablierten Mechanismen von Git (Issues) nutzen, um Ihr Feedback zum Leitfaden zu platzieren. Die aktuelle Version des Leitfadens ist [per Webseite](#) kontinuierlich erreichbar, die PDF-Version wird in regelmäßigen Abständen aktualisiert.

Gender-Erklärung

Das Autorenteam hat darüber diskutiert, und wir haben uns wegen der besseren Lesbarkeit für das "generische Maskulinum" entschieden. Bitte verstehen Sie die ausschließliche Verwendung der männlichen Form geschlechtsunabhängig.

1 Einführung Eclipse und ADT

1.1 Warum eigentlich Eclipse?

Warum ist Eclipse die richtige Plattform für ADT-Tools und nicht VSCode, NetBeans oder irgendeine andere Entwicklungsumgebung? Eine vollumfassende Antwort wird hier nicht möglich sein. Fakt ist jedoch, dass SAP sich strategisch für den Einsatz von Eclipse als Grundlage für die zukünftige ABAP-Entwicklungsumgebung (als Nachfolger der klassischen ABAP Workbench) entschieden hat und plant, diesen Ansatz auch weiter zu verfolgen. Das ist so auch aus den Roadmaps der ABAP-Plattform ersichtlich und kann dort nachgelesen werden.

Es muss aber erwähnt werden, dass mit der Anbindung an Eclipse eine offene Schnittstelle (zwischen der Entwicklungsumgebung und dem SAP-Backend) entstanden ist, die von anderen Umgebungen genutzt werden kann. Für VSCode gibt es beispielsweise bereits Erweiterungen, die auf diesen Schnittstellen basieren. Über die Vollständigkeit und Nutzbarkeit wird jedoch im Leitfaden nicht eingegangen. Wenn Sie hierzu mehr Informationen benötigen, werden Sie über die üblichen Suchmaschinen schnell passende Ergebnisse finden.

1.2 Ein kurzer Blick auf die Entstehungsgeschichte

Der Ursprung von Eclipse ist bei IBM und wurde im Jahr 2004 mit der Eclipse-Foundation eigenständig. Eclipse wird in der Programmiersprache Java implementiert und steht unter einer Open-Source-Lizenz offen zur Verfügung.

Zunächst wurden die neuen Eclipse-Versionen nach Objekten aus dem Weltall benannt (u.a. Jupiter-Monde). Der Einfachheit halber hat man sich aber auch bei der Eclipse-Foundation 2018 für eine nachvollziehbare Namenskonvention für die einzelnen Versionen entschieden: < (Jahr)-(Monat) >. Zeitgleich wurde auf deutlich schnellere (dreimonatige) Release-Zyklen umgestellt.

1.3 Das Prinzip von Erweiterungen

Die Basisinstallation von Eclipse wird bereits mit einem Grundstock an Funktionen und Erweiterungen ausgeliefert. Der Umfang dieser Funktionen und Erweiterungen hängt maßgeblich davon ab, welche Version Sie auf der Webseite zum Download auswählen. Technisch betrachtet ist der Kern der Anwendung immer identisch. Alle zusätzlichen Funktionalitäten werden mittels Erweiterungen geliefert – das gilt auch für die ABAP Development Tools. Das heißt also, dass Sie jederzeit alle

Erweiterungen im Nachhinein Ihrer Installation hinzufügen bzw. wieder entfernen können.

Typischerweise beziehen Sie die Erweiterungen über den Eclipse-Marketplace, den Sie auch direkt aus dem Tool heraus erreichen.

1.4 Unterschied zwischen Perspektive und Sicht

Ein wichtiges Merkmal von Eclipse besteht in der Anpassbarkeit. Hierauf wird in den folgenden Kapiteln noch im Detail eingegangen. Essentiell ist hierbei, die grundlegenden Begrifflichkeiten zu verstehen.

Eine Sicht (oft auch View genannt) ist ein eigenständiger Programmteil, der entweder bereits in der Basisinstallation enthalten ist oder per Erweiterung Eclipse nachträglich hinzugefügt wurde. Diese Sicht kann vom Benutzer an verschiedenen Positionen innerhalb der Entwicklungsumgebung angeordnet werden.

Eine Perspektive beinhaltet ein spezifisches Layout von Eclipse, also alle angezeigten Sichten und deren Position. Perspektiven geben Ihnen damit den Fokus für eine Tätigkeit mit allen Sichten, die Sie hierfür benötigen. Details hierzu finden Sie im [Kapitel 6 - Best Practices Eclipse-Konfiguration](#).

Die Kenntnis der Begrifflichkeiten ist essentiell, um effizient mit Eclipse arbeiten zu können. Dokumentationen aller Art nutzen diese Terminologie, um Funktionen zu erklären. Das gilt auch für diesen ADT-Leitfaden.

1.5 Die Macht der Tastenkürzel

Ein prägnantes Merkmal von Eclipse liegt in der Verwendung von Tastenkürzeln. Sie werden am Anfang viel Zeit damit verbringen, die Funktionalitäten, die Sie von der klassischen ABAP-Workbench (Transaktion: SE80) gewohnt sind, wiederzufinden und sich einzuprägen. Im Gegensatz zur SE80 ist Eclipse jedoch dahingehend optimiert, mit Tastenkürzeln bedient zu werden. Behalten Sie das im Hinterkopf und nutzen Sie die Möglichkeiten, sich die Tastenkürzel zu verinnerlichen. Weitere Informationen hierzu finden Sie auch in [Kapitel 3 - Arbeiten mit ADT](#).

1.6 Ab wann ist welche Funktion mit Eclipse verfügbar?

Da der Funktionsumfang der ABAP Development Tools fortlaufend durch SAP erweitert wird, ist es wichtig zu prüfen, welche Funktionen in der von Ihnen verwendeten ADT-Version zur Verfügung stehen. Dabei ist zu beachten, dass einige der bereitgestellten Funktionen auch vom Release des verwendeten SAP-Systems abhängen. Um den aktuellen Stand über die letzten Updates der ADT zu erfahren, sind die offiziellen Quellen von SAP das Mittel der Wahl, da diese laufend gepflegt werden.

Für Cloud:

<https://help.sap.com/docs/BTP/5371047f1273405bb46725a417f95433/ab03dcd9072f4a2d85c945d05929d3fb.html>

Für On-Premise:

https://help.sap.com/doc/2e9cf4a457d84c7a81f33d8c3fdd9694/Cloud/en-US/inst_guide_abap_development_tools.pdf

2 Motivation für ADT

In diesem Kapitel stellen wir Ihnen mehrere Gründe vor, die für den Einsatz der ABAP Development Tools (ADT) als Entwicklungsumgebung sprechen. Am Ende gehen wir auch auf den mitunter größten “Knackpunkt” für langjährige ABAP-Entwickler ein und geben Hinweise, warum dieser manchmal als Nachteil betrachtete Punkt in Wirklichkeit ein Vorteil ist.

Wir freuen uns, wenn wir Sie mit diesem Kapitel vom Nutzen der ADT überzeugen können oder zumindest Ihre Neugier geweckt haben, diesen Leitfaden zu studieren.

2.1 **Sie möchten modernes ABAP anwenden und umsetzen.**

Der erste und naheliegendste Grund für eine umfassende Nutzung der ABAP Development Tools in Eclipse ist die strategische Ausrichtung seitens SAP. Die ABAP Development Tools wurden 2012 eingeführt und haben sich in diesen zehn Jahren zu einer stabilen Entwicklungsumgebung mit einem großen Funktionsumfang entwickelt.

Im Gegensatz dazu befinden sich die SAP-GUI-gebundenen Entwicklungswerkzeuge mit ihrem prominentesten Beispiel, der ABAP Workbench, im Wartungsmodus. Das bedeutet, dass hier zwar weiterhin Fehlerkorrekturen stattfinden, jedoch keine neuen Funktionen mehr ausgeliefert werden.

Dies ist vielleicht kurzfristig nicht der motivierendste Grund, allerdings wird dies jeden Entwickler im SAP-Umfeld eines Tages einholen. Daher empfehlen wir, den Einstieg und Umstieg in und auf die ABAP Development Tools lieber heute als morgen anzugehen.

Aus diesem Grund empfiehlt SAP den Einsatz der ADT als Standardumgebung für die ABAP-Entwicklung, um mit jedem Release von neuen Funktionen und Korrekturen zu profitieren. Der tatsächlich vorhandene Funktionsumfang hängt vom Release-Stand des ABAP-Stacks der verwendeten SAP-Systeme ab. Eine grobe Übersicht und weitere Informationen hierzu finden sich im Abschnitt: [Kapitel 1 - Einführung Eclipse und ADT](#).

2.2 Sie möchten eine Entwicklungsumgebung für alles nutzen.

2.2.1 Technologischer Aspekt

Mit den ABAP Development Tools (ADT) können Sie nicht nur für On-Premise Systeme wie beispielsweise SAP-ERP oder S/4HANA entwickeln. Die ADT sind auch die einzige Möglichkeit, Entwicklungen für Cloud-Systeme wie die Business Technology Platform (BTP) durchzuführen.

Dieser Umstand ist insbesondere für ABAP-Entwickler wichtig, die im Kontext ABAP Cloud (z.B. SAP BTP, ABAP Environment oder S/4HANA Cloud Public Edition, ABAP Environment) entwickeln. In diesem Kontext können klassische, SAP-GUI-orientierte Entwicklungswerkzeuge nicht verwendet werden und Entwicklungen können nur mit den ADT durchgeführt werden.

Eng damit verbunden ist auch die Arbeit mit einigen neuen Entwicklungsartefakten. Die Erstellung bzw. Pflege von [CDS Views](#) ist nur mit ADT möglich. Und auch das neue Programmiermodell der SAP, das "RESTful Application Programming Model" (RAP) kann nur mit ADT verwendet werden.

2.2.2 Entwicklung auf mehreren Systemlinien

Eine Entwicklungsumgebung für alles gilt bei den ADT aber auch für den Fall, dass Sie auf mehreren Entwicklungssystemen arbeiten dürfen oder müssen. In der Eclipse-Umgebung sind verschiedene Systeme als Projekte aufgeführt, und Sie können hierüber sehr komfortabel und übersichtlich auf diese zugreifen. Mittels der [Working Sets](#) (siehe Kapitel 3 - "Arbeiten mit ADT") können die Systeme in Überbegriffen gruppiert und sogar mittels zusätzlichen [Plug-ins](#) (siehe Kapitel 7) farblich gekennzeichnet werden. Das Öffnen der Systeme aus dem SAP Logon Pad, Anmelden am System und Öffnen der SE80, entfällt. Daraus ergeben sich zahlreiche weitere Synergien bei der Arbeit, die in diesem Leitfaden beschrieben werden.

2.3 Sie möchten sich von technischen Einschränkungen des SAP GUI lösen.

2.3.1 Ihnen sind sechs Sessions in einem System zu wenig

SAP-GUI-basierte Entwicklungswerkzeuge wie die ABAP Workbench unterliegen aufgrund ihrer Ausführung im SAP GUI verschiedenen Einschränkungen, die nicht relevant sind, wenn Sie mit den ADT arbeiten.

Hierzu zählt z. B. die Abhängigkeit von der maximalen Anzahl gleichzeitiger SAP GUI Sessions (Modi, vgl. Parameter "rdisp/max_alt_modes"). Dieser Wert wird von dem

per SAP GUI aufgerufenen System individuell vorgegeben. Im Standard sind es sechs gleichzeitige SAP GUI Sessions pro Benutzer im gleichen System. Für die ADT gilt diese Vorgabe nicht.

Wenn Sie als Entwickler mit unterschiedlichen Systemen gleichzeitig arbeiten, profitieren Sie von einem weiteren Vorteil: Jede Verbindung zu einem System wird als "ABAP Project" für On-Premise-Systeme bzw. "ABAP Cloud Project" für Cloud-Systeme innerhalb der ADT gepflegt. Die Verbindungen zu diesen Systemen können gleichzeitig genutzt werden. Das ermöglicht es Ihnen, Quelltext aus einem Quellsystem in die Zwischenablage zu übernehmen und in einem gleichzeitig geöffneten Zielsystem einzufügen – und das alles innerhalb desselben Anwendungsfensters von Eclipse.

Darüber hinaus ist auf diese Weise auch ein einfacher Code-Vergleich zwischen verschiedenen Systemen, auch zwischen Systemen ohne bestehende RFC-Verbindung, möglich. Die Anzeige mehrerer SAP-GUI-Fenster wie in früheren Zeiten entfällt.

2.3.2 Sie möchten durch Netzwerkunterbrechungen keinen Code verlieren.

Aufgrund technischer Gegebenheiten der SAP GUI ist es erforderlich, dass die Netzwerkverbindung stabil und unterbrechungsfrei läuft. Gibt es Netzwerkunterbrechungen, während Sie in einem GUI-Fenster Code erstellen, kann es vorkommen, dass die Arbeit der letzten Minuten vergebens war, da SAP GUI die Verbindung zum Server verloren hat und das Fenster schließt.

Mit den ADT ist eine Netzwerkunterbrechung kein Problem mehr. Das Eclipse-Fenster bleibt geöffnet, auch wenn die Verbindung unterbrochen ist. Sobald die Verbindung wiederhergestellt ist, kann der Code im SAP-System gespeichert werden.

Sollten größere Probleme im Netzwerk auftreten, kann der Code notfalls einfach komplett als Text per Copy-and-paste in einem alternativen Texteditor zwischengespeichert werden, bis das SAP-System wieder verfügbar ist.

2.4 Sie verbessern Ihren ABAP-Code gerne durch Refactoring.

Die Wartbarkeit von Entwicklungsartefakten ist eine zentrale Herausforderung der Software-Entwicklung. Um eine gute Wartbarkeit zu erreichen, orientiert sich das Vorgehen beim Entwickeln häufig an Clean-Code-Prinzipien (vgl. Clean ABAP). Zur Einhaltung der Clean-Code-Prinzipien ist ein wiederholtes Überarbeiten von Entwicklungsartefakten, auch Refactoring genannt, unabdingbar.

Die ADT unterstützen typische Refactoring-Aufgaben mit Hilfe der [Quick Assists](#)-Funktionen, zu denen auch die Quick Fixes gehören. Die Quick Assists sind kontextsensitiv. Sie können beispielsweise komplexe Quelltextabschnitte in kleinere, neue und vor allem eigenständige Methoden auslagern, was die Verständlichkeit und damit die Wartbarkeit deutlich erhöht. Ohne Unterstützung durch die Quick Assists ist ein solches iteratives Vorgehen deutlich arbeitsintensiver und auch fehleranfälliger.

Weitere und detailliertere Informationen zum Refactoring und Werkzeugen, die Ihnen in den ADT dafür zur Verfügung gestellt werden, finden Sie in [Kapitel 3 - Arbeiten mit ADT](#).

2.5 Sie passen die Darstellung Ihrer Entwicklungsumgebung gerne Ihren Wünschen an.

Die ADT basieren auf Eclipse, einer weit verbreiteten Open-Source-Entwicklungsumgebung. Stärken dieser Entwicklungsumgebung sind u. a. die vielfältigen Anpassungsmöglichkeiten wie bspw. an der Darstellung (vgl. Views und Perspektiven in [Kapitel 3 - Arbeiten mit ADT](#)) oder an den Shortcuts ([Tastaturkürzeln](#)) und der Unterstützung durch hilfreiche Funktionen wie bspw. einer umfassenden Suche.

Als Nutzer von Eclipse können Sie die Entwicklungsumgebung daher auf vielfältige Art individualisieren und verwenden, was jedem Anwender eine bessere Unterstützung seiner bevorzugten Arbeitsweise erlaubt.

Da die ADT den Zugriff auf verschiedene SAP-Systeme gleichzeitig ermöglichen, können Sie über alle Systeme hinweg einheitlich arbeiten. Dies steht im Kontrast zur Arbeit mit SAP-GUI-gebundenen Entwicklungswerkzeugen wie der ABAP Workbench, die in jedem System separat individualisiert werden müssen.

Mit den sogenannten Workspaces haben Sie die Möglichkeit, verschiedene Konfigurationen der Arbeitsumgebung abzuspeichern. Zum Beispiel sind hier verschiedene Projekte, verschiedene Favorite-Packages, welche Sichten und Objekte geöffnet sind und sogar die Stelle im Code, an der zuletzt gearbeitet wurde, im jeweiligen Working-Set gespeichert. Zusätzlich können verschiedene Workspaces in mehreren parallel laufenden Eclipse-Instanzen geöffnet werden. Damit haben Sie die Möglichkeit, abhängig von Projekt, Kunde oder Aufgabe, sich jeweils die effizienteste Umgebung zusammenzustellen. In [Kapitel 3 - Arbeiten mit ADT](#) und in [Kapitel 6 - Best Practices Eclipse-Konfiguration](#) finden Sie Informationen, wie Workspaces zu verwenden sind.

2.6 Sie möchten noch mehr Tool-basierte Unterstützung?

Falls Ihnen der Funktionsumfang der von SAP bereitgestellten Funktionen nicht ausreicht, können Sie den Funktionsumfang durch weitere Plug-ins, die es als Ergänzung zu den ADT gibt, erweitern.

Eine Auswahl dieser Plug-ins wird in [Kapitel 7 - Plug-ins](#) vorgestellt. Mit dem richtigen Know-how können Sie bei Bedarf auch eigene Plug-ins erstellen und der SAP Community zur Verfügung stellen.

In der Vergangenheit haben verschiedene ABAP-Entwickler von dieser Möglichkeit bereits Gebrauch gemacht und eigene Plug-ins entwickelt, die sie der SAP-Community kostenlos zur Verfügung stellen. Ein Beispiel hierfür sind die [ABAP Quick Fix](#). Dies ist eine Erweiterung der im Standard der ADT angebotenen Quick Fixes zur Unterstützung der automatischen Konvertierung der klassischen ABAP-Syntax in das jeweilige moderne ABAP-Syntax-Pendant.

2.7 Sie kennen Eclipse bereits von anderen Programmiersprachen. Prima!

Eclipse als Grundlage der ADT ist in verschiedenen Unternehmen bereits im Einsatz. Das liegt daran, dass es z. B. für JAVA eine der am weit verbreitetsten Entwicklungsumgebungen ist. JAVA wiederum ist eine häufig verwendete Programmiersprache (vgl. [TIOBE-Index](#)) und spielt daher in vielen Entwicklungsprojekten eine Rolle. Daneben ist Eclipse aber auch für andere Entwicklungszwecke einsetzbar, z. B. für die Entwicklung in der Programmiersprache Python oder der Arbeit mit Daten im Extensible-Markup-Language-Format (XML).

Es besteht also eine gewisse Wahrscheinlichkeit, dass Entwickler und Unternehmen bereits Vorkenntnisse im Umgang mit Eclipse als Entwicklungsumgebung besitzen. Hierdurch gelingt der erweiterte Einsatz auf Basis der ADT einfacher und schneller, als wenn auf ein vollständig neues, möglicherweise gänzlich unbekanntes Entwicklungswerkzeug gesetzt wird.

2.8 Sie haben Interesse an Neuem.

Zu guter Letzt möchten wir noch auf das Interesse vieler Menschen an Neuem und der damit verbundenen Attraktivität als Motivationsfaktor für den Umstieg hinweisen. Handelte es sich bei den Gründen zum Umstieg in den vorherigen Abschnitten um eher sachliche Aspekte, geht es hier um im Menschen tief verankerte Emotionen und

Werte: das Interesse an etwas Neuem und die Attraktivität, die etwas Neues hat. Oder kurz: um Fortschritt und Veränderung.

Für manche Menschen bedeutet der Stillstand mit gelegentlichen kleinen Veränderungen eine gewisse Sicherheit im Arbeitsalltag. Diese Sicherheit ist allerdings trügerisch, da sich die (IT-)Welt rasant verändert, und nicht immer bieten bestehende Technologien und Methoden die passenden Antworten auf aktuelle und zukünftige Herausforderungen.

Insbesondere in der ABAP-Welt hat sich in den letzten zehn Jahren sehr viel verändert. Nachdem zahlreiche neue Sprachkonstrukte ABAP bereichert haben, sind neue Artefakte wie CDS Views und neue Programmiermodelle wie das ABAP Restful Application Programming Model (RAP) dazugekommen. Und auch der technologische Wandel wie der Weg in die Cloud sind nicht von der Hand zu weisen. Konsequenterweise erfordert dies auch die Weiterentwicklung der Tools, die bei der Entwicklung verwendet werden.

Zudem kann es bei einem technologischen Stillstand auch zu einem Ausschluss nachfolgender Generationen an Entwicklern kommen, da diese Technologien mit einem oftmals gänzlich anderen Blick betrachten, ihre bereits gemachten Erfahrungen mit anderen Entwicklungsumgebungen und Programmiersprachen abgleichen und ihre Bewertung auf dieser Basis vornehmen.

2.9 Die Vorteile für die Organisation

Wir möchten mit diesem Leitfaden nicht nur den ABAP-Entwickler motivieren, ADT einzusetzen. Auch für die Organisation ergeben sich durch den umfassenden Einsatz von ADT im SAP-Entwicklungsbereich zahlreiche Vorteile.

Neben der höheren Effizienz der Entwickler, u. a. durch bessere Tool-Unterstützung zur Erstellung und Verbesserung des Codes, stellen hier auch wieder Zukunftsfähigkeit und einheitliches Tooling die wichtigsten Gründe für die Organisation dar, den Einsatz der ADT in den SAP-Entwicklungs-Teams zu fördern und einzufordern.

Um den Einsatz für den einzelnen Entwickler reibungsfrei zu ermöglichen und um Hemmnisse in Bezug auf einen Umstieg von Anfang an zu vermeiden, und damit einen breiten Einsatz der ADT zu erreichen, müssen folgende Themen zentral geklärt und in Form von allgemein verfügbaren Dokumentationen den Entwicklern bereitgestellt werden:

- Rahmenbedingungen und generelle Informationen zu ADT
- Installation von Eclipse
- Zugriff von Eclipse auf Ressourcen im Internet (Updates und Plug-in-Installationen)
- Berechtigungen der Entwickler für ADT-Objekte im Backend (S_RFC for ADT*)

Dies stellt auch hier wieder zuerst eine Investition dar, die sich aber auszahlt, sobald die Entwickler-Teams einheitlich mit ADT arbeiten und die Vorteile der Entwicklungsumgebung in der täglichen Arbeit voll ausgenutzt werden können.

Zahlreiche Informationen, Hilfestellungen und Best Practices hierzu finden Sie in [Kapitel 5 - Installation, Verteilungs- und Update-Strategien](#).

2.10 Ihnen fehlt der formularbasierte Editor

Die ADT nutzen an vielen Stellen eine textorientierte Darstellung von Entwicklungsartefakten, was Ihnen von anderen Entwicklungsumgebungen wie beispielsweise Microsoft Visual Studio Code (VSCode) bekannt ist. Damit wird auf die bisherige, formularbasierte Darstellung, die Sie von den SAP-GUI-orientierten Entwicklungswerkzeugen kennen, verzichtet.

Dies ist sicher eine der größten Hürden für eingefleischte ABAP-Entwickler, die von GUI-basierten Werkzeugen auf Eclipse umsteigen wollen oder sollen. Um diese Hürde zu überwinden, finden Sie einen ausführlichen Einstieg in die Arbeit mit den ADT in [Kapitel 3 - Arbeiten mit ADT](#).

Mit dieser Veränderung in der Darstellung geht auch eine gewisse Veränderung der Arbeitsweise einher. Der häufige Wechsel zwischen verschiedenen SAP-GUI-Oberflächen, die in einem bestimmten Kontext stehen, entfällt. Stattdessen gibt es oftmals "nur" Text – also Anweisungen, die eingegeben bzw. gelesen werden. Das führt zu einer starken Konzentration auf die eigentlichen Anweisungen, ihre Wirkungen und Zusammenhänge (Syntax und Semantik).

Ein Beispiel für diese veränderte Darstellung ist die Signatur eines Funktionsbausteins. Mit den SAP-GUI-basierten Entwicklungswerkzeugen wird die Signatur eines Funktionsbausteins, bestehend aus IMPORT, EXPORT, CHANGING, TABLES und EXCEPTIONS als fünf separate Register dargestellt. In den ADT wird die Signatur als Text dargestellt und gepflegt. Ein Wechsel zwischen unterschiedlichen kontextgebundenen Registern entfällt.

Nach der anfänglichen Umgewöhnung werden Sie sicherlich schnell die Vorteile der textorientierten Arbeitsweise erkennen, die sich durch den Wegfall der

Navigationsschritte durch die GUI und auch durch die zahlreichen Unterstützungen wie Code-Vervollständigung und Quick Fixes ergeben.

2.11 Warum ABAP Development Tools

Falls Sie die bisher beschriebenen Vorteile von ADT noch nicht überzeugt haben, möchten wir Sie mit Zitaten der Autoren des Leitfadens motivieren, sich einmal in die Sicht von ADT-Usern zu versetzen. Eventuell finden Sie ja hier den Anstoß, sich doch dem Thema anzunehmen.

Michael Keller: *“Clean ABAP ohne die ADT und damit die Unterstützung durch die Quick Fixes ist für mich undenkbar - schließlich sparen sie dem Entwickler viel Zeit und Arbeit.”*

Florian Henninger: *“Refactoring ohne ADT ist so ein bisschen wie zu versuchen, eine Suppe mit einer Gabel essen zu wollen - kann funktionieren, macht nur keiner.”*

Bärbel Winkler: *“Durch die Mitarbeit bei der Erstellung dieses Leitfadens habe ich viele gute Gründe kennengelernt, in Zukunft häufiger als bisher mit den ADT zu arbeiten.”*

Jens Zähringer: *“Obwohl die ABAP Development Tools bereits seit über 10 Jahren zur Verfügung stehen, habe ich erst kürzlich den Umstieg von ABAP Workbench zu ADT für mich vollzogen. Die Umstellung war nicht ohne Herausforderungen, aber am Ende hat es sich definitiv gelohnt!”*

Peter Luz: *“Mittels des Verwendungsnachweises schnell ermitteln, wo eine Methode verwendet und wie sie dort aufgerufen wird. Dann diese mehrfach verwendete Methode umbenennen. Anschließend ein Stück Code daraus in eine eigene Methode extrahieren. Dann noch zum Abschluss den Code auf dem zentralen Entwicklungssystem mit der Version auf der Q-Maschine einer Systemlinie vergleichen.*

In ADT nur ein paar Tastenkombinationen entfernt und in Sekundenschnelle durchgeführt. So macht das Erstellen und Überarbeiten von Software Spaß und hilft, die Qualität der Software maßgeblich durch Nutzung der hier im Leitfaden beschriebenen Werkzeuge zu verbessern. Für mich ist das Erstellen von ABAP-Software ohne ADT inzwischen undenkbar.”

Michael Biber: *“Ja, SE80 ist mittlerweile ganz gut. Jedoch sehe ich dies wie mit dem Umstieg auf Objektorientierung: Am Anfang fragt man sich ‚Warum?‘ und sieht vorhandene Stolpersteine. Sobald man jedoch mal die andere Seite (Objektorientierung, ADT ...) erlebt hat, möchte man nicht mehr zurück. Die Vorteile der besseren Übersichtlichkeit (Element Info), theoretisch unendlich parallel geöffneten Sourcen, Live-Syntaxprüfung und viele mehr überwiegen für mich, über*

alle kleinen Medienbrüche (→ ältere SAP-Releases) und andere Denkansätze hinweg.“

Björn Schulz: “Ohne die ADT würde ich viel langsamer an Informationen aus dem System kommen und wahrscheinlich mit nur sechs Modi nicht mehr klarkommen.”

Dr. Wolfgang Röcklein: “Gibt es ABAP Entwicklung ohne ADT?” “Kein Clean Code ohne Refactoring, kein Refactoring ohne ADT.”

Uwe Fetzer: “Schneller, komfortabler, sicherer kann man ABAP Clean Code nicht entwickeln”.

Sebastian Freilinger-Huber: “Lieber heute als morgen die ‘Komfortzone SE80’ verlassen - es lohnt sich. Sollten Sie noch zweifeln, finden Sie zahlreiche Argumente für den Umstieg in den folgenden Kapiteln”.

3 Arbeiten mit ADT

Die ABAP Development Tools bieten sehr viele Funktionen, die auch entsprechend zahlreiche Möglichkeiten der Nutzung bieten. Dies kann für Entwickler, die noch nicht mit den ADT vertraut sind, verwirrend sein.

Der erste Schritt ist immer der schwerste. Zu Beginn dieses Kapitels möchten wir dem Einsteiger den Start in die ADT-Welt am Beispiel der Erstellung einer Klasse erleichtern. Die hier gezeigte Vorgehensweise lässt sich dann auf andere Entwicklungsobjekte übertragen, und der Einstieg ist geschafft.

Die weiteren Funktionalitäten werden im darauf folgenden Abschnitt übersichtsartig beschrieben, und es werden Hinweise gegeben und Best Practices erläutert, wie die zahlreichen Werkzeuge und auch Hilfen in der täglichen Arbeit eingesetzt werden und einen Mehrwert bieten können.

Neben den ADT-Einsteigern werden aber auch in ADT erfahrene Entwickler viele nützliche Hinweise und vielleicht auch Neues für die tägliche Arbeit mit den ABAP Development Tools finden.

Mit dem [SAP ABAP Development User Guide](#), im Folgenden User-Guide genannt, stellt SAP die offizielle Dokumentation zu den ADT zur Verfügung. Für weitergehende Informationen zu einzelnen Funktionen sind die Links zu dem entsprechenden Abschnitt im User-Guide vermerkt.

3.1 Einführung: Grundlagen der Arbeit mit ADT

3.1.1 Der Einstieg in das Arbeiten mit den ABAP Development Tools

Dieser Abschnitt richtet sich an Entwickler, die bisher noch nicht mit den ADT gearbeitet haben und die ersten Schritte gehen möchten. Im User-Guide befindet sich im Abschnitt "[Getting Started](#)" ein Abschnitt zum Einstieg in ADT, in dem alle Funktionen ausführlich erläutert werden. In diesem Leitfaden möchten wir mittels einer Schritt-für-Schritt-Beschreibung zur Erstellung einer ABAP-Klasse den Einstieg so leicht wie möglich gestalten und die Vorteile der Nutzung der ABAP Development Tools aufzeigen.

Folgende Arbeitsschritte werden hierbei Schritt für Schritt beschrieben:

- Einrichtung des Projekts
- Einrichtung der Favoritenpakete
- Erstellung einer Klasse mit einer Methode
- Arbeiten am Code und Refactoring

Zum Zweck der generellen Nachvollziehbarkeit werden die hier gezeigten Beispiele in einer Instanz des BTP Trial Accounts dargestellt, diese können aber in gängigen On-Premise-Systemen problemlos angewendet werden.

Entwickler, die bereits den Einstieg geschafft haben und sich einen Überblick über die einzelnen Funktionen verschaffen möchten oder detaillierte Arbeitshinweise suchen, können diesen Abschnitt überspringen.

3.1.2 Der Umstieg von der formularbasierten zur textorientierten Code-Erstellung

Im Gegensatz zu den SAP-GUI-basierten Transaktionen wie SE80 oder SE24 findet sich in ADT kein formularbasierter Editor. Die Erstellung von Klassen (und auch Funktionsbausteinen etc.) erfolgt in ADT rein textbasiert. Für Entwicklungswerkzeuge, die abhängig von ADT und Backend-Version noch nicht in Eclipse zur Verfügung stehen, kann aus Eclipse auf diese Transaktionen integriert zugegriffen werden.

Gewöhnungsbedürftig ist auch der Tausch der Tasten **F2** und **F3**. Während im SAP-GUI die F3-Taste als Zurück-Taste verwendet wird, dient die F3-Taste in den ADT zur Vorwärtsnavigation, die F2-Taste zur kontextsensitiven Hilfe.

Dies stellt für den Einstieg mitunter auch die größte Hürde dar, da lang eingeübte Praxis und gewohnte Arbeitsprozesse sich mit dem Umstieg auf ADT ändern. Und wenn es schnell gehen soll, greift man gerne auf gewohnte und bestens vertraute Arbeitsweisen zurück.

Der Einstieg und die Umstellung erfordert also erstmal Zeit und Bedarf der Einübung. Doch der anfängliche Mehraufwand zahlt sich bereits nach kurzer Zeit wieder aus. Denn nach etwas Eingewöhnung und Übung stellen die beschriebenen Punkte kein Problem mehr dar. Die zahlreichen Funktionen, die die ABAP Development Tools bieten, erleichtern das Schreiben und Überarbeiten von ABAP Code und erhöhen damit die Effizienz beim Entwickeln. Daher sollte der Umstieg von SE80 & Co. auf ADT in Eclipse als ein persönliches Investment in eine effiziente und zukunftsfähige Arbeitsweise betrachtet werden.

3.1.3 Kontext des hier gezeigten Übungsbeispiels:

Das hier gezeigte Beispiel ist bewusst einfach gewählt, da vorrangig die grundlegenden Funktionen und Arbeitsweisen der ADT dargestellt werden sollen.

Wir möchten eine kleine Klasse anlegen, die folgende Funktionen anbietet:

1. Ermittlung von Flügen aus der Tabelle */DMO/FLIGHT* gemäß Eingabe.
2. Berechnung der verfügbaren Plätze des Fluges.
3. Berechnung des Flugpreises auf Basis einer zusätzlichen prozentualen Gebühr.

Die Klasse dient als interne Service-Klasse und bietet keine UI bzw. Ausgabe von Daten an.

Dabei werden die grundlegenden Arbeitsweisen und meistgenutzten Funktionen gezeigt, die zum Effizienzgewinn während der Code-Erstellung und Änderung führen.

3.1.4 Verbinden des Entwicklungssystems – Neues Projekt

Ein Entwicklungssystem wird in den ADT in Form eines Projekts dargestellt. Um ein Entwicklungssystem mit den ADT zu verknüpfen, müssen wir daher ein neues Projekt anlegen.

In einer neuen Installation von ADT wird ein neues Projekt über

File → New → ABAP Projekt

angelegt.

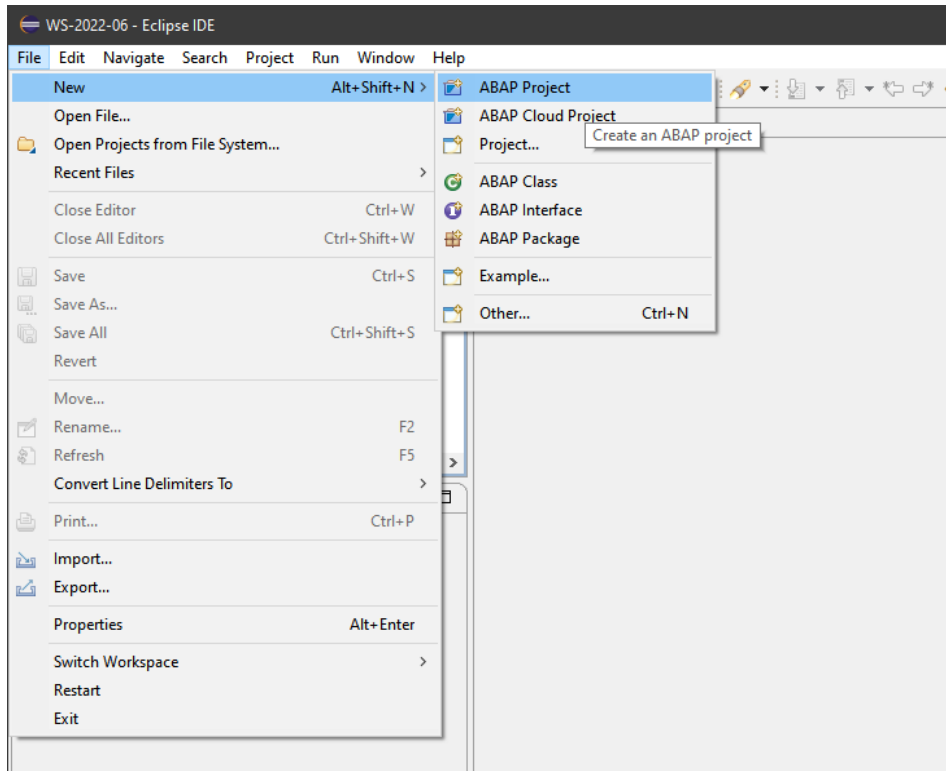


Abbildung 1 Erstellung eines ABAP-Projekts in Eclipse

Bei Anlage eines ABAP-Projekts für On-Premise-Systeme wird die Liste der im SAP-Logon verknüpften Systeme angezeigt. Die Login-Daten sind zu hinterlegen, sofern kein SSO verwendet wird und im letzten Schritt kann dem Projekt noch ein sprechender Name gegeben werden. Als Default ist die Sprache Englisch gewählt, diese muss ggf. angepasst werden.

Das neu erstellte Projekt und damit das verbundene Entwicklungssystem wird im sogenannten [Project Explorer](#) dargestellt.

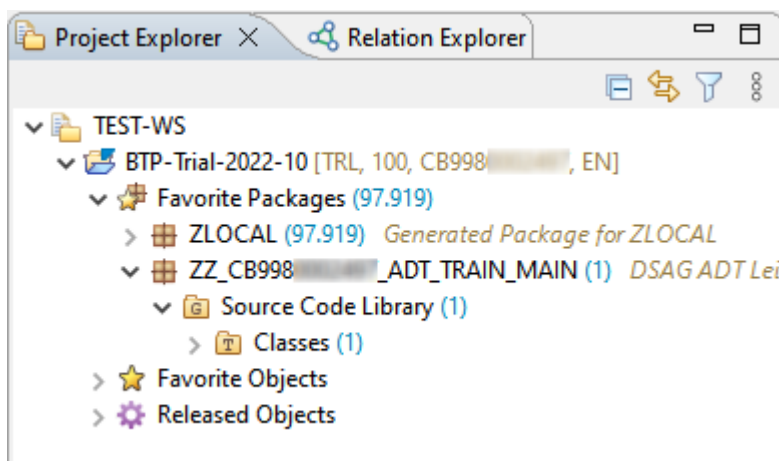


Abbildung 2 Der Project Explorer

Der Project Explorer ist der zentrale Einstiegspunkt und Objektkatalog, nachdem das entsprechende Entwicklungssystem geöffnet wurde. Die Objekte werden basierend auf den Paketen in hierarchischer Form dargestellt, wie man es bereits aus der SE80 kennt. In der täglichen Arbeit werden die zu bearbeitenden Objekte hieraus geöffnet.

Der Hauptarbeitsbereich ist der Knoten **Favorite Packages**. Um das Paket zu den Favorite Packages hinzuzufügen, in das die zu erstellende Klasse eingefügt werden soll, führen Sie den Befehl "Add Package" mittels des Kontextmenüs aus.

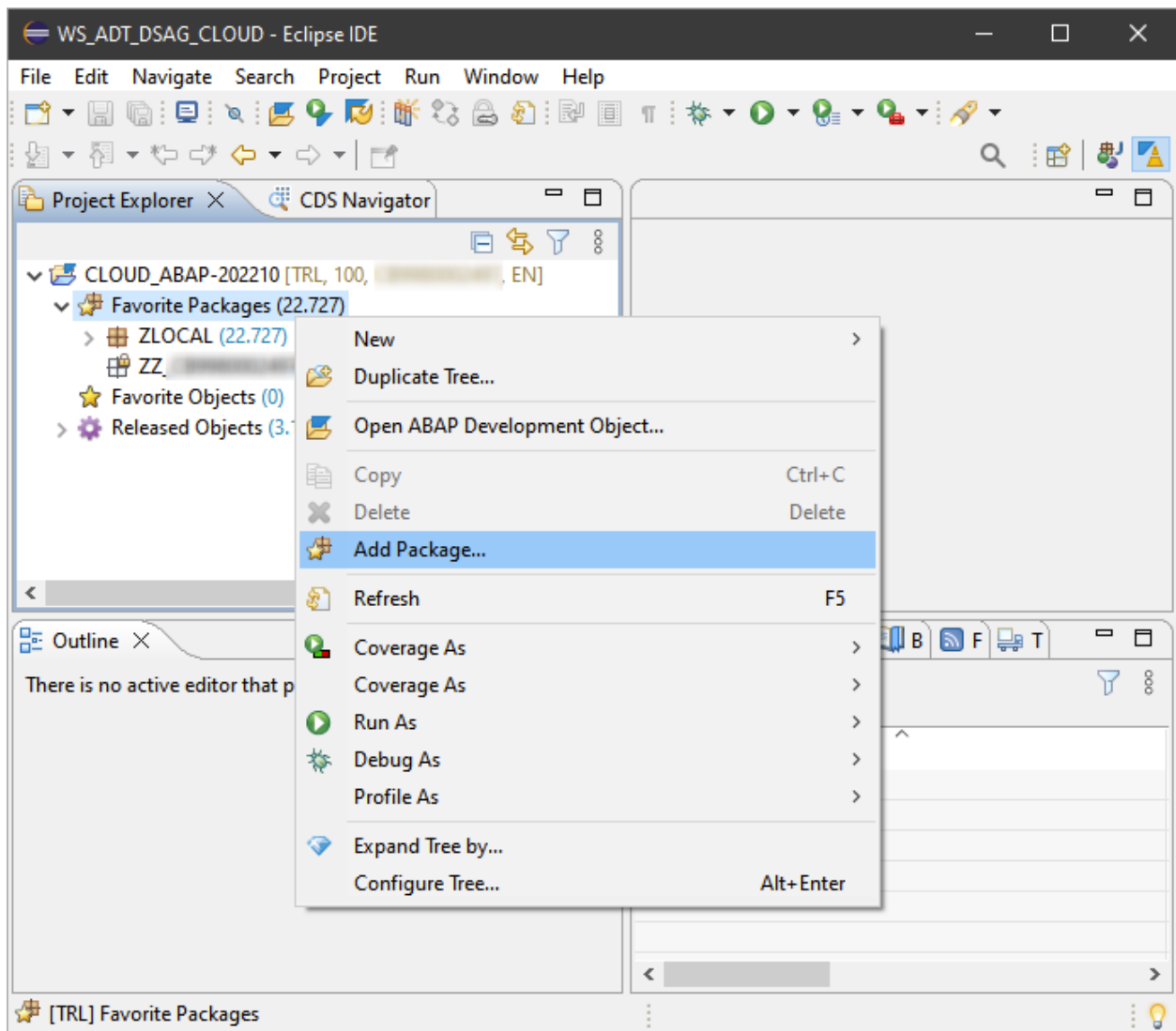


Abbildung 3 Hinzufügen von Packages zu den Favoriten

3.1.5 Das Erstellen einer Klasse im Textmodus

Zur **Erstellung einer neuen ABAP-Klasse** navigieren Sie im Project Explorer in das gewünschte Paket, erreichen das Kontextmenü mittels der rechten Maustaste und finden dort den Befehl

New → ABAP Class

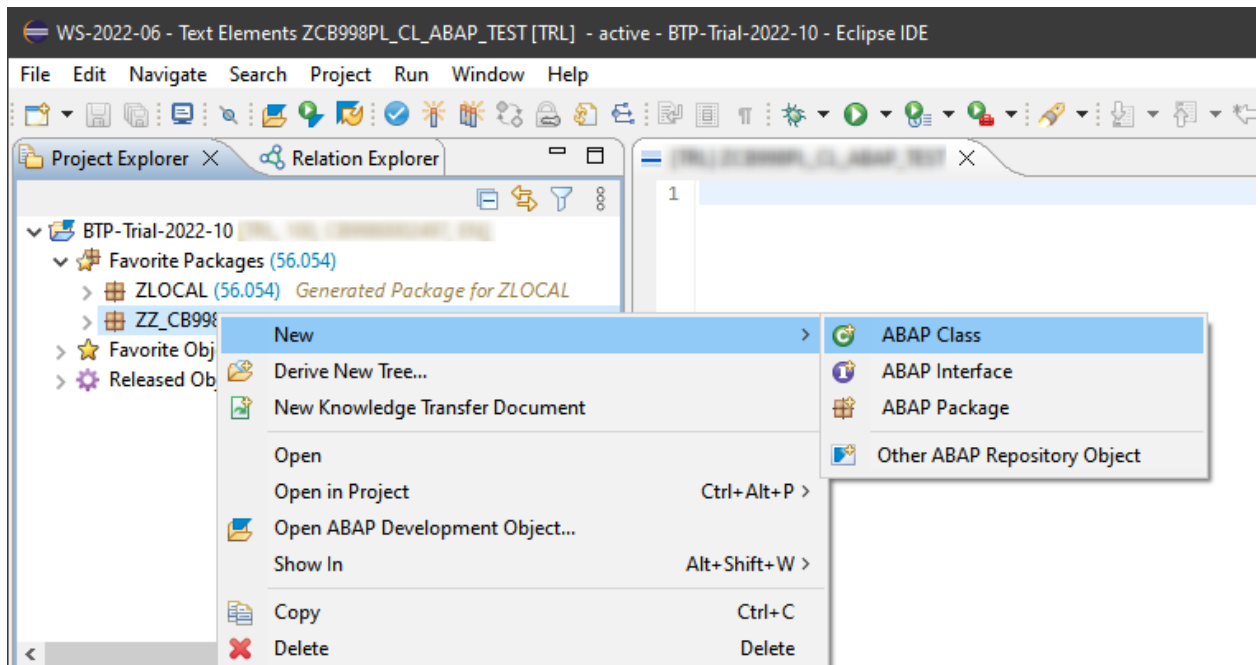


Abbildung 4 Erstellen einer neuen ABAP-Klasse im Projekt Explorer

Es öffnet sich ein Fenster, in dem die Daten der Klasse angegeben werden können/müssen.

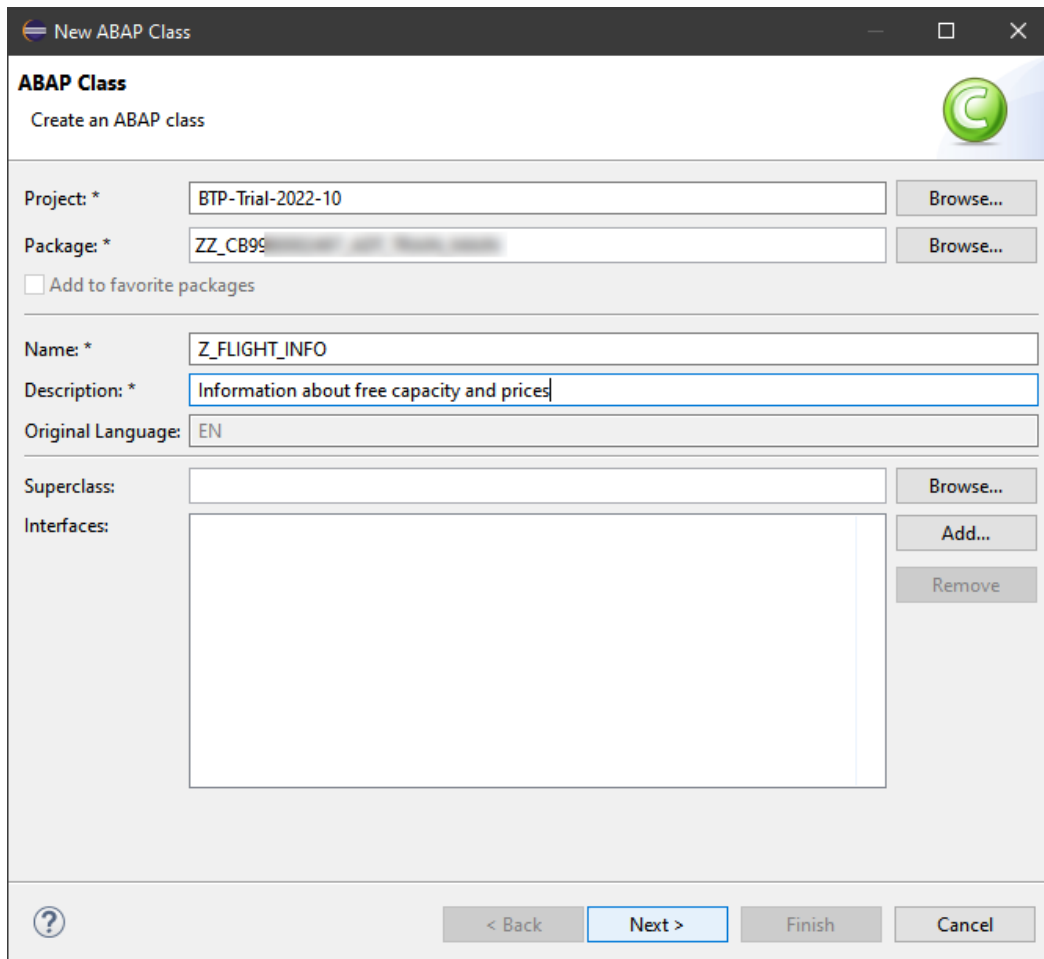


Abbildung 5 Eigenschaftsdialog: Erstellung ABAP-Klasse

Hier können bei Bedarf bereits die Super-Klasse und zu referenzierende Interfaces angegeben werden. Dies kann aber auch später textbasiert direkt im Quellcode erfolgen. Es öffnet sich nach Klick auf "Next" das Fenster zur Auswahl bzw. Anlage des Transportauftrages.

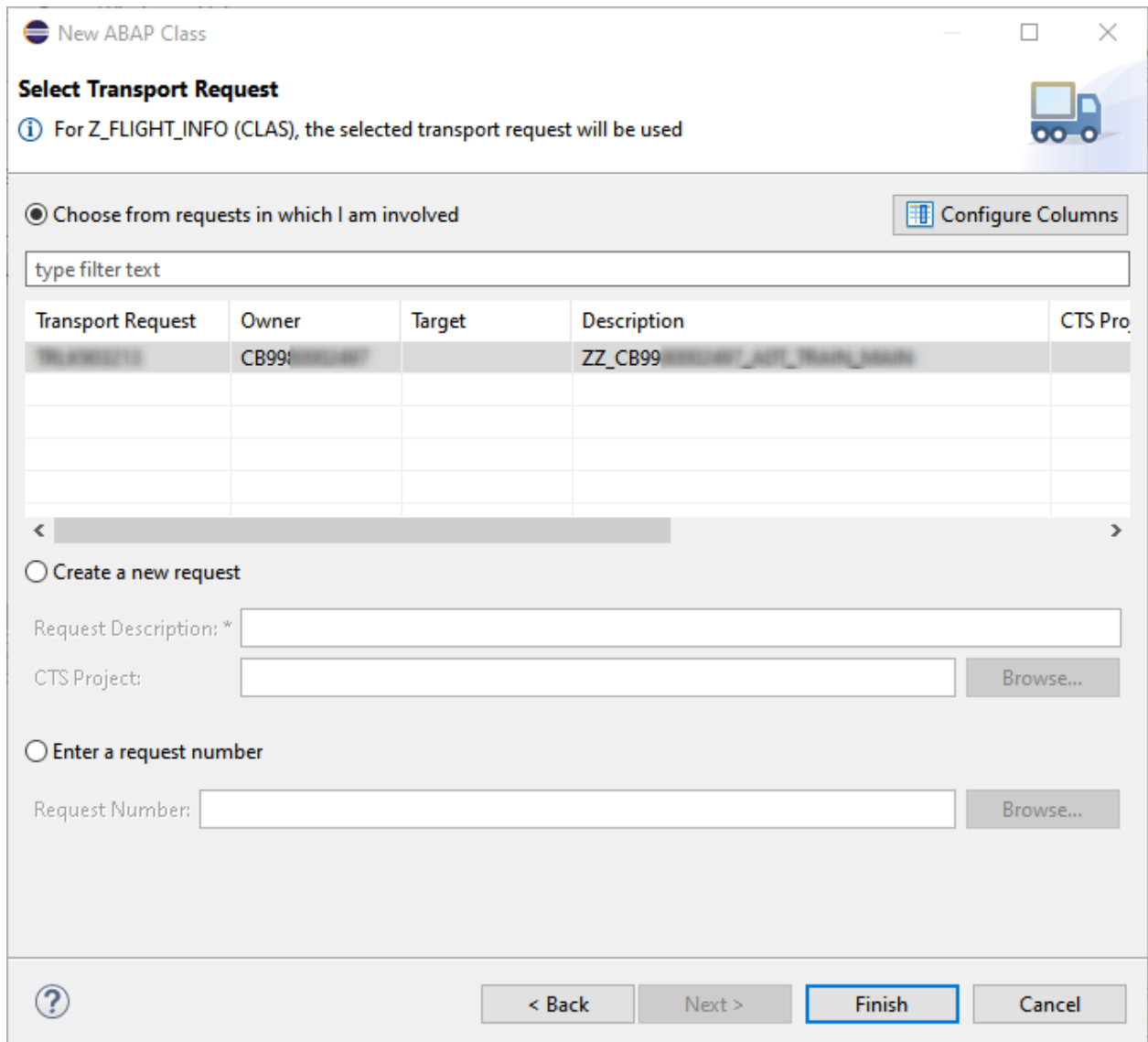


Abbildung 6 Transportauftragsdialog

Nach Klick auf “Finish” wird die Klasse angelegt, und diese findet sich sowohl im Project Explorer im Objektbaum als auch im Quellcode-Editor auf der rechten Seite der ADT.

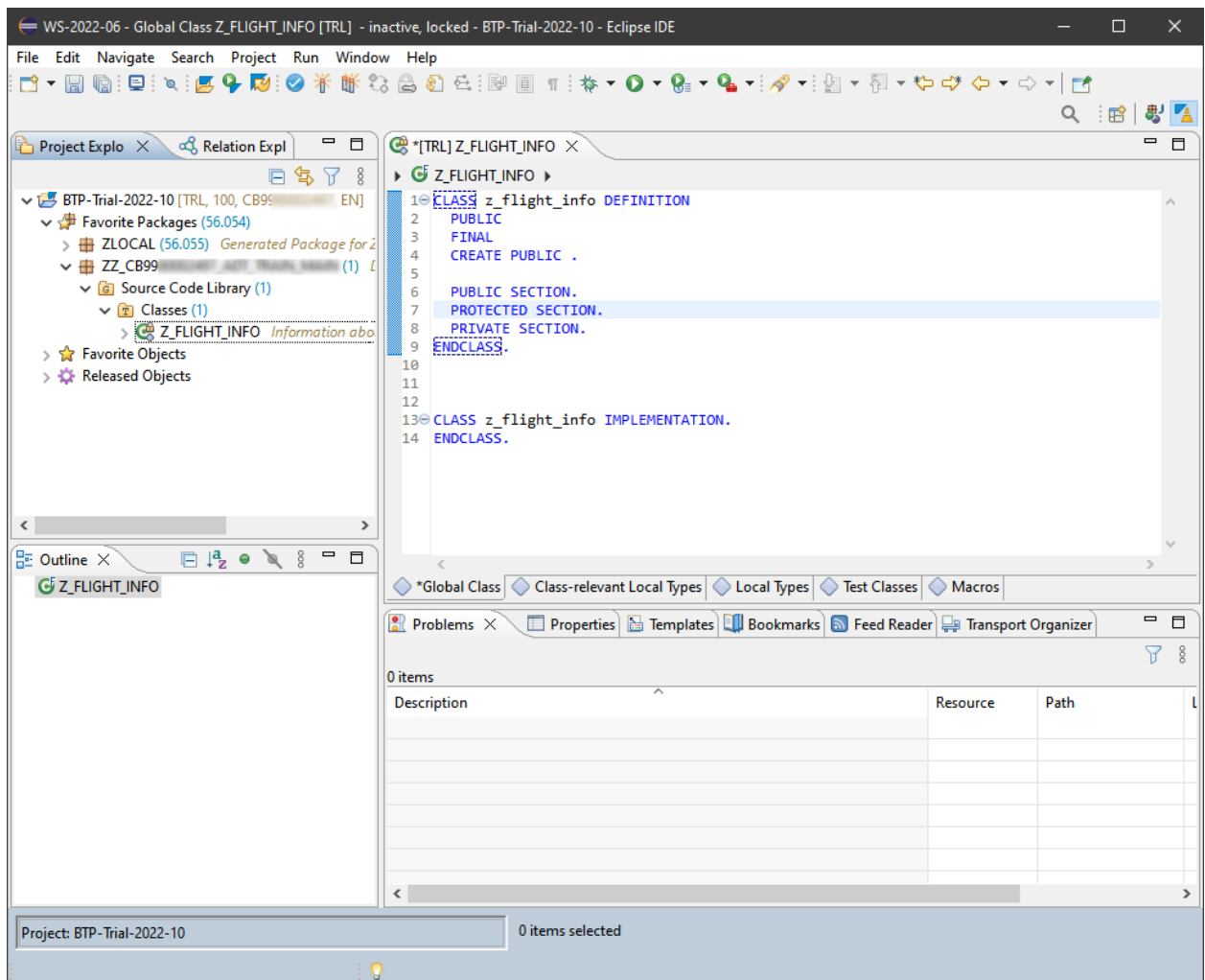


Abbildung 7 Anzeige der neuen Klasse in den ADT

Alle weiteren Operationen an der Klasse werden nun im Quellcode durchgeführt, d. h. alle Typen, Datendefinitionen und Methoden werden textbasiert als Quellcode in der Klasse geschrieben. Dies erscheint anfangs kompliziert und ungewohnt, die ADT bieten aber einige Funktionen an, die die Ausarbeitung der Klasse sehr effizient gestalten.

3.1.6 Definition einer Methode in der Klasse

Nun möchten wir die erste Methode der Klasse erstellen, die Daten aus der Tabelle /DMO/FLIGHTS liest und die Anzahl der freien Plätze zu einem definierten Flug ausgibt.

Wir beschränken uns hier nur auf die Kernfunktionalitäten und werden keine zusätzlichen Funktionen für Output etc. bereitstellen.

Eine ABAP-Klasse ist in ADT in die Hauptbereiche "Definition" und "Implementierung" aufgeteilt. Dementsprechend werden wir für unsere erste Methode zuerst im Bereich "Definition" die Methode mit ihren Parametern definieren und anschließend die Implementierung mit dem Quellcode durchführen. Die sogenannte "Quick Fix"-Funktion wird uns dabei Tipparbeit ersparen.

Um eine Methode zu erstellen, navigiert man im Reiter/Tab "Global Class" in den Bereich

```
CLASS <classname> DEFINITION
```

und platziert den Cursor in den Sichtbarkeitsbereich der Klasse, in der die Methode zur Verfügung steht. In unserem Fall soll die Methode für andere Verwender sichtbar sein und daher im Bereich PUBLIC definiert werden. Die Definition der Methode wird mit dem Schlüsselwort METHODS eingeleitet.

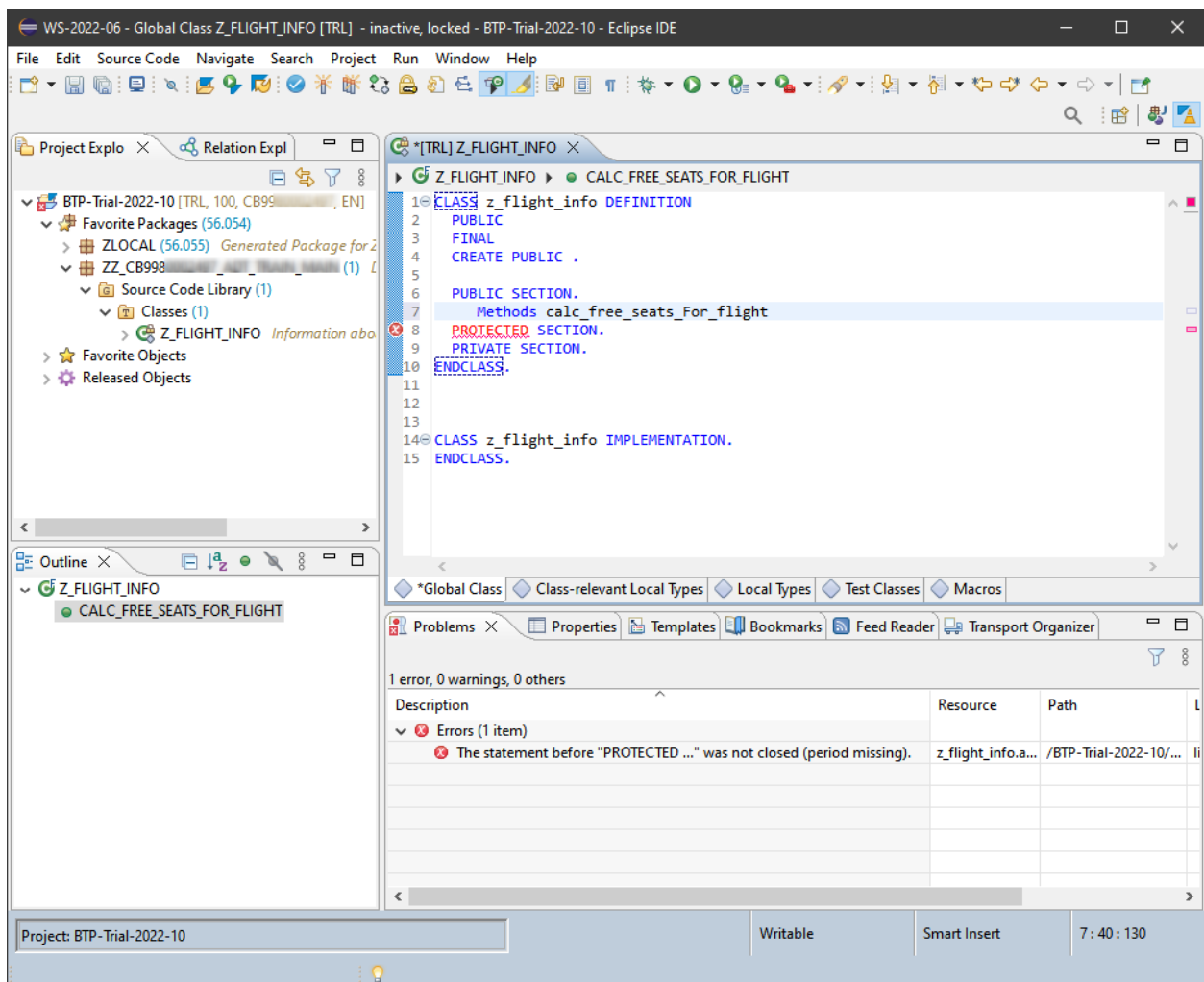


Abbildung 8 Bearbeiten der Klasse

Bereits zu diesem Zeitpunkt zeigt sich einer der Hauptvorteile von ADT gegenüber der SE80. Sobald Code eingegeben wurde, läuft automatisch der Syntax-Check und zeigt an, ob der Code syntaktisch korrekt ist.

Da der Abschlusspunkt nicht vorhanden ist, zeigt ADT sowohl im linken Balken einen Hinweis (als Hover-Message) als auch die View-Problems den Syntaxfehler an. Ein extra auszuführender Syntax-Check ist nicht erforderlich. Nur eine Kleinigkeit, die in der täglichen Arbeit eine signifikante Effizienzsteigerung bedeutet. Spätestens wenn man aus diversen Gründen eine Änderung in den GUI-basierten Tools vornimmt, wird man diese Funktion vermissen.

3.1.7 Automatische Ergänzung und Formatierung des Codes

Wir schließen die Methodendefinition mit Erstellung der Parameter und des Abschlusspunkts ab. Durch die Nutzung der Code Completion wird uns die Arbeit dabei sehr erleichtert. Dazu geben wir jeweils nur die ersten zwei bis drei Buchstaben des gewünschten Schlüsselwortes an. Die Tastenkombination **STRG+LEERTASTE** zeigt uns die passenden Schlüsselworte an. Diese können daraufhin mittels **TAB+Pfeiltasten** aus der Vorschlagsliste ausgewählt werden. Zur weiteren Automatisierung der Code-Erstellung bietet sich die Nutzung von [Templates](#) an.

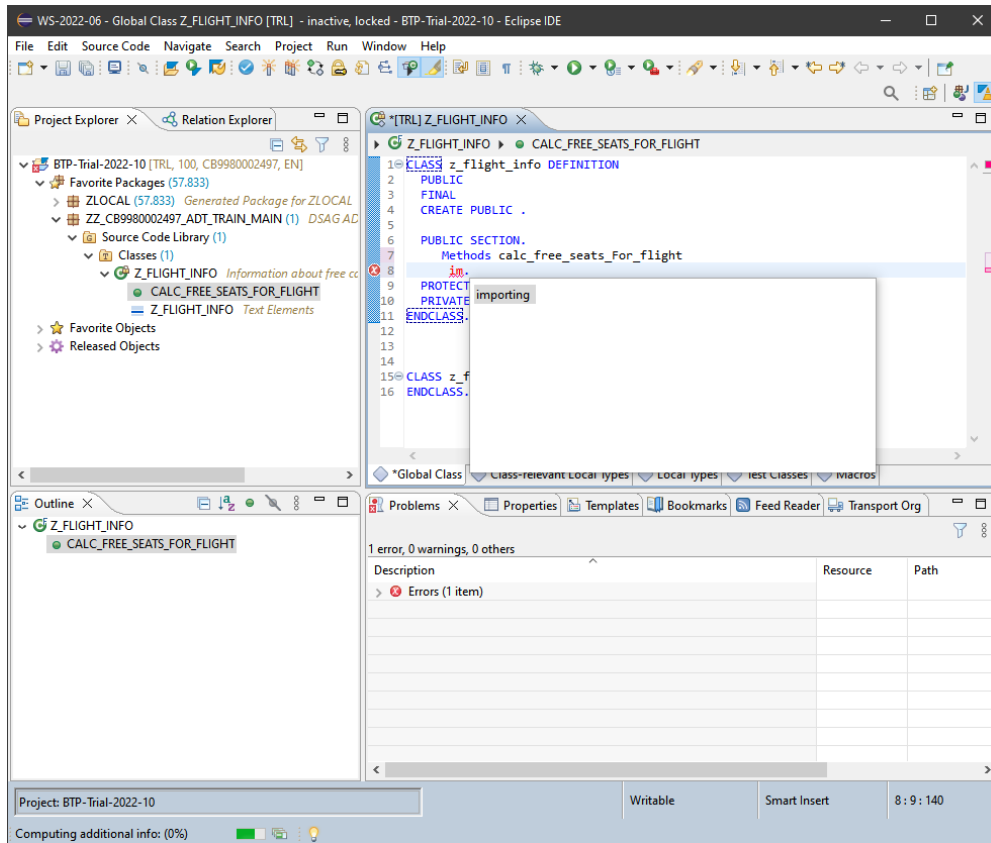


Abbildung 9 Beispiel Code Completion für den Import-Parameter

Nachdem die Import-Parameter und der Return-Parameter definiert sind, wird der Cursor im Textbereich des Codes positioniert und die Formatierung des Codes mittels des Kontextmenüs

Rechte Maustaste → Source Code → Format bzw. der Tastenkombination **Shift+F1**

ausgeführt. Dies entspricht dem Pretty Printer in den GUI-Transaktionen. Anschließend wird der Code mittels

STRG+S

gespeichert.

Sollte ein Syntaxfehler vorliegen, zeigt das Problem-View und die farbliche Markierung im Code den Fehler an. Ist der Code syntaktisch korrekt, kann das Artefakt anschließend mittels **STRG+F3** aktiviert werden.

Nach Einübung der neuen, vor allem Tastatur-basierten, Arbeitsweise entsteht nach kurzer Zeit ein Automatismus, der nach dem Schreiben einiger Codezeilen und anschließender Abfolge der o.g. Tastenkombinationen einen formatierten, geprüften und gesicherten Code als Ergebnis hat. Zeitraubende Überraschungen in Form von Syntaxfehlern beim Aktivieren, wie es in den GUI-Tools vorkommen kann, bleiben dadurch erspart.

3.1.8 Implementierung der Methode mittels Quick-Fix

Die Methode ist nun definiert, aufgrund der fehlenden Implementierung zeigt ADT im Problems-View folgenden Fehler an:
 "Implementation missing for method
 "CALC_FREE_SEATS_FOR_FLIGHT".

Dieses "Problem" lässt sich sehr effizient mit Hilfe der Quick Fixes beheben.

Zur Methodenimplementierung nutzen Sie die Quick-Fix-Funktion, die sich über das Kontextmenü oder die Tastenkombination **STRG+1** aufrufen lässt.

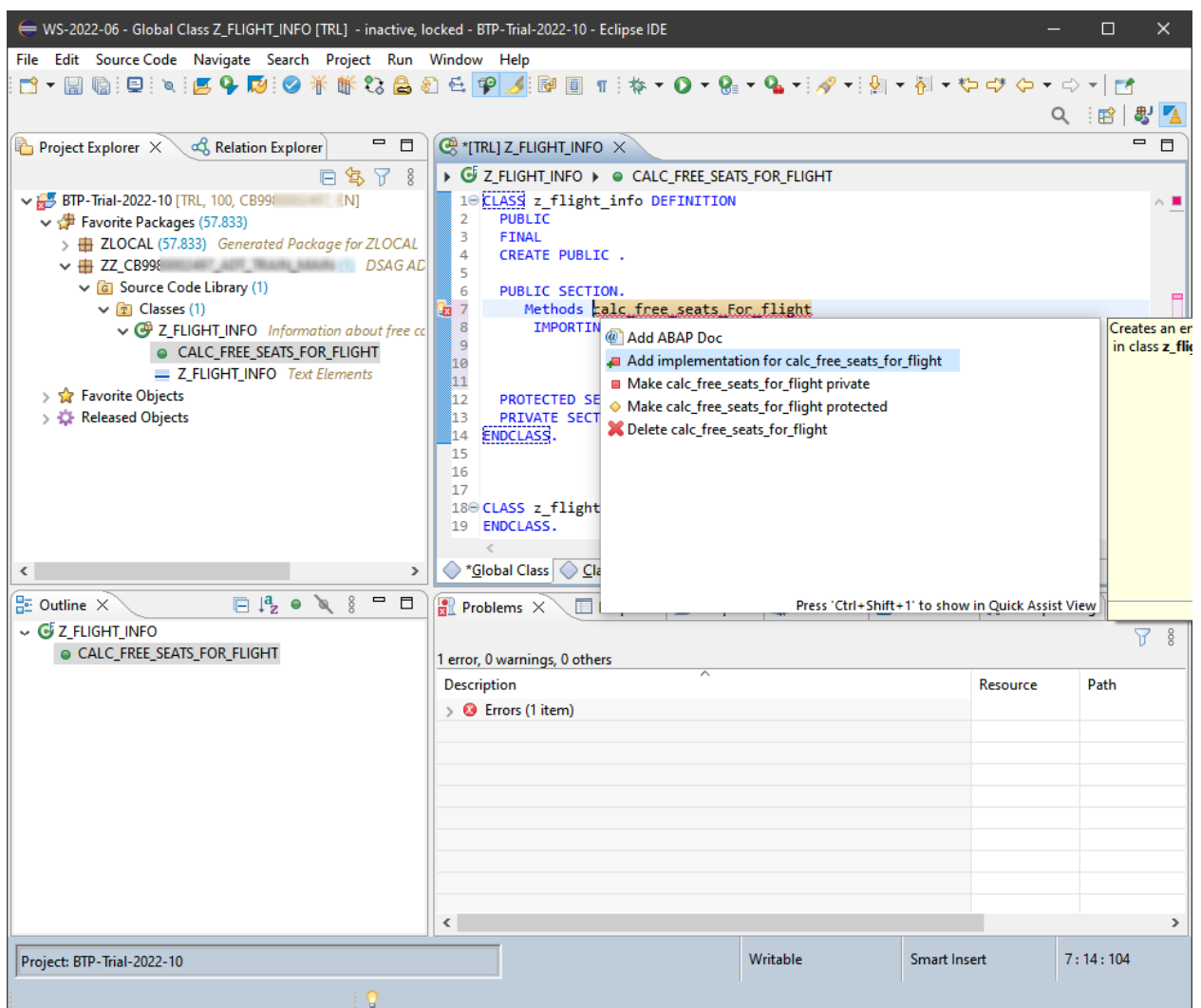


Abbildung 10 Nutzung des Quick Fix zur Methodenimplementierung

Sie wählen "Add Implementation for ..." aus und bestätigen mit `ENTER`. Zum Ausführen der Quick-Fix-Befehle sollte der Code gespeichert und fehlerfrei sein, da manche Quick Fixes (automatische Code-Korrekturen/Refactorings) dies als Voraussetzungen haben.

Die Methodenimplementierung bedeutet hier, dass automatisiert die Sektion

METHOD

ENDMETHOD

im Bereich der "Class Implementation" durch ADT erstellt wird und somit eine leere Methodenimplementierung vorhanden ist. Die Ausprägung der Methodenlogik innerhalb dieses Bereiches ist dann Ihre Aufgabe als Entwickler.

Zwischen Definition und Implementierung kann einfach durch Drücken der `F3`-Taste hin und her gesprungen werden.

Mit Platzierung des Cursors auf den Methodennamen und `F2` kann man sich die Parameter der Methoden anzeigen lassen. Dies erleichtert das Schreiben des Codes und erspart umständliches Navigieren in den Definitionsbereich. Alternativ steht hierfür die View [ABAP Element Info](#) zur Verfügung.

In unserem Übungsbeispiel verwenden wir die Import-Parameter, um mittels eines `SELECT`-Befehls den gewünschten Datensatz zu lesen. Anschließend werden die Anzahl der freien Sitze kalkuliert und als Returning-Parameter an den Verwender zurückgegeben.

Auch hier hilft die Verwendung der Code Completion, die mit der Tastenkombination `STRG+LEERTASTE` aufgerufen wird, um den Code effizient und frei von Tippfehlern zu erstellen.

Um den Returning-Parameter "r_f_free_seats" nicht ausschreiben zu müssen und Typinformationen zu bekommen, schreiben wir die Struktur und den Komponentenseparator "-" und bekommen mittels der Tastenkombination `STRG+LEERTASTE` die Komponenten angezeigt, die dann ausgewählt und in den Code eingefügt werden können.

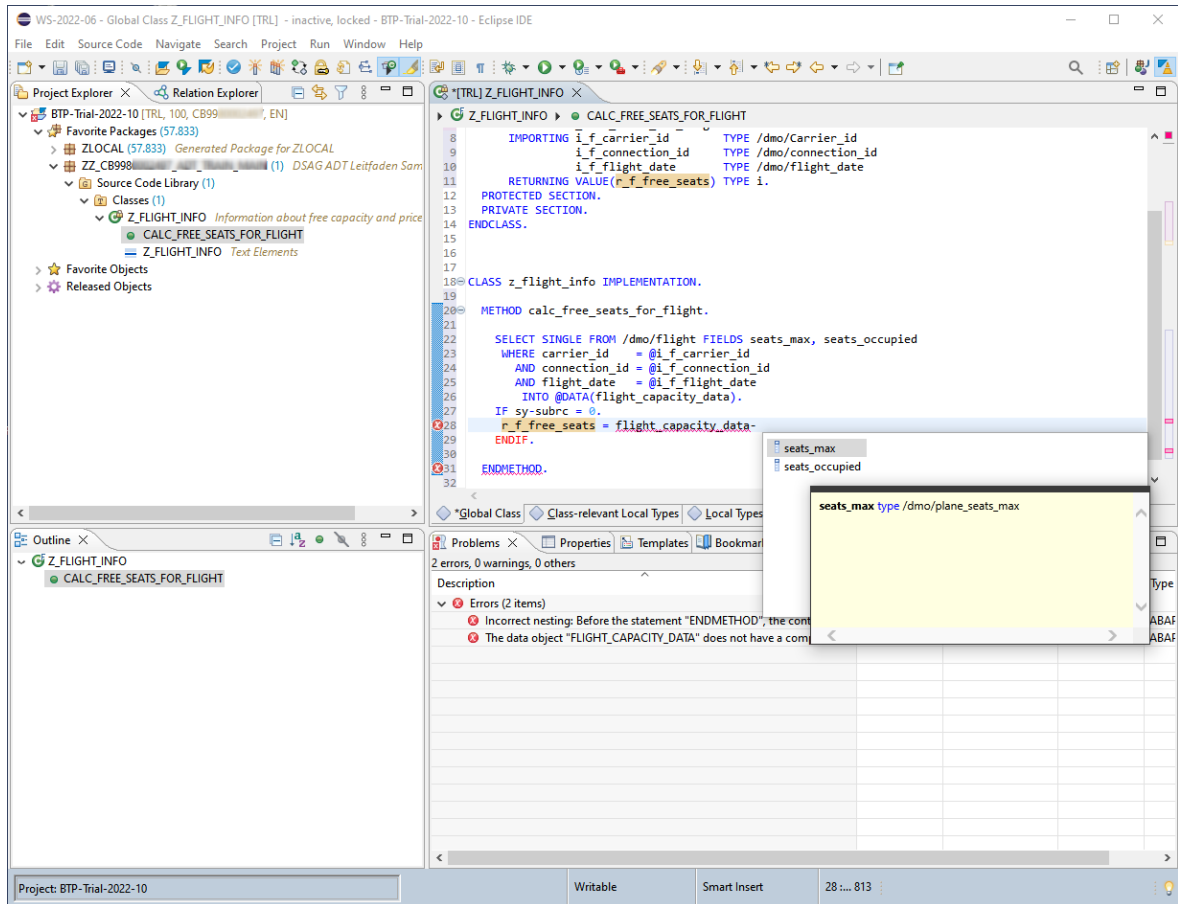


Abbildung 11 Auswahl der Komponente mittels Code Completion

Diese Möglichkeiten der Vorwärtsnavigation und Inline-Anzeige weitergehender Elementinformationen sind in dieser Form in SE80 nicht vorhanden und ein weiterer Baustein zur Steigerung der Entwicklereffizienz in ADT.

Wir haben damit die Klasse mit der ersten Methode in ADT definiert und implementiert.

3.1.9 Umbenennung von Parametern – Refactoring

Die laufende Optimierung von bestehenden Code ist eine wichtige Aufgabe jedes Entwicklers. Dies wird von ADT bestens unterstützt. Die verschiedenen Möglichkeiten des Refactoring werden im Abschnitt: [Refactoring von Code mit ADT](#) ausführlich erläutert. Weitere Informationen finden sich auch im [User-Guide](#).

Wir möchten die generelle Vorgehensweise des Refactoring anhand einer Umbenennung im Detail darstellen.

Gewohnheitsgemäß haben wir die ungarische Notation verwendet und möchten nun die Parameter der Methode umbenennen, um die Präfixe zu entfernen (vgl. ABAP Clean Code - <https://github.com/SAP/styleguides/blob/main/clean-abap/CleanABAP.md>).

Während eine solche Aktion sich im SAP-GUI-Umfeld u. U. sehr aufwendig und fehleranfällig gestalten kann, bietet hier die Rename-Funktion in ADT eine sehr komfortable Möglichkeit, die Umbenennungen von Variablen, Parametern und Methodennamen über alle Verwendungen durchzuführen. Das bedeutet, dass in allen Entwicklungsobjekten, welche die Methode aufrufen, die Parameter automatisiert umbenannt werden. Eine aufwendige Suche nach Verwendern über den Verwendungsnachweis entfällt. Dies funktioniert aber natürlich nur, wenn keine dynamischen Methodenaufrufe verwendet werden. Solche Fälle kann ADT nicht erkennen.

Somit sind Code Cleaning und Refactoring mit ADT sehr effizient durchzuführen, und das Risiko von dadurch entstehenden Fehlern ist gegenüber der manuellen Methode deutlich reduziert.

Zur Umbenennung der Parameter wird die Rename-Funktion aus dem Kontextmenü über rechte Maustaste → Source-Code → Rename oder die Tastenkombination

ALT+SHIFT+R oder per Quickfix-Auswahl über

STRG+1

ausgeführt.

Arbeiten mit ADT

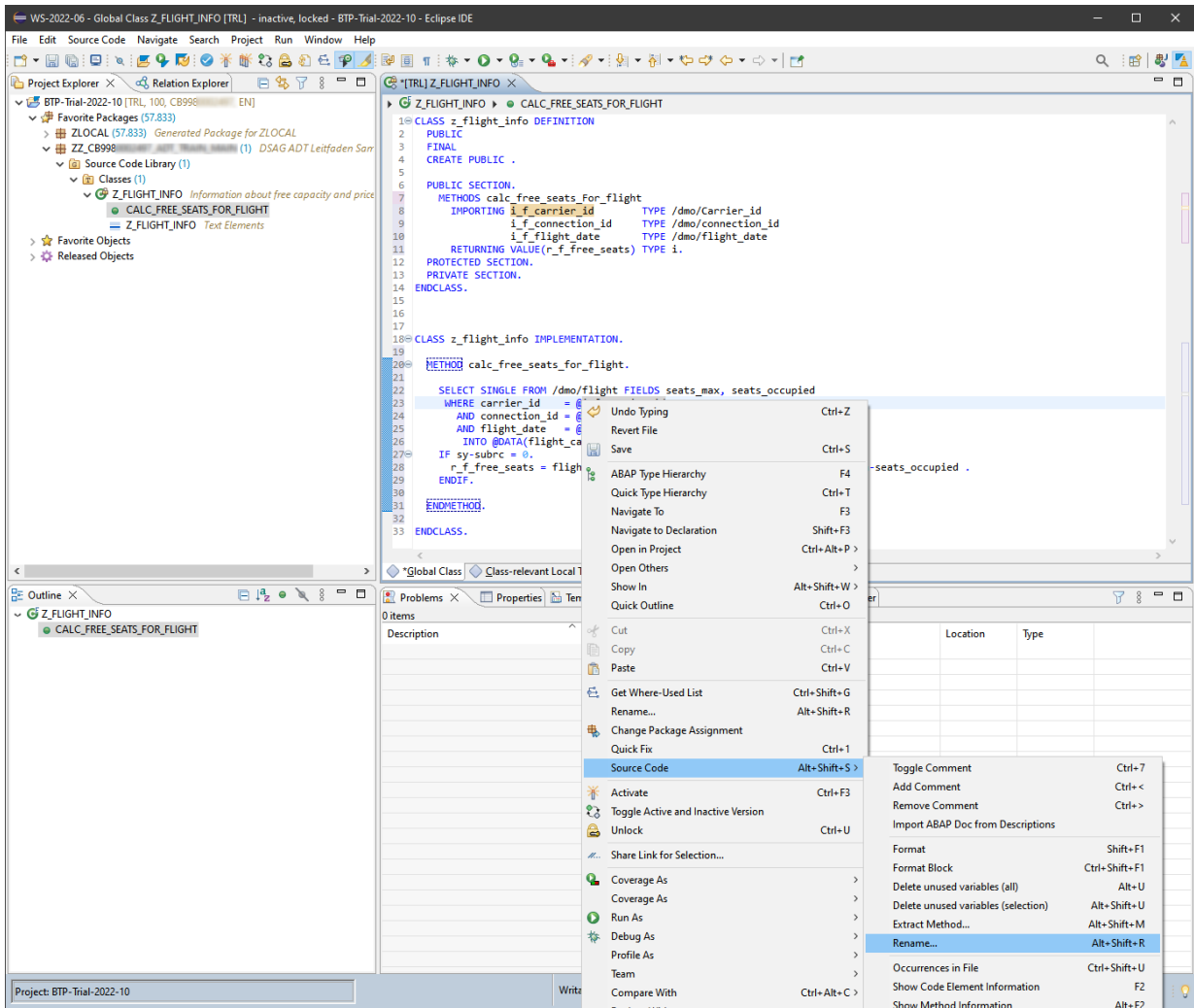


Abbildung 12 Umbenennung von Methodenparametern

Zuerst muss der Code gespeichert werden. Falls dies nicht erfolgt ist, erscheint eine Abfrage zum Speichern, die bestätigt werden muss.

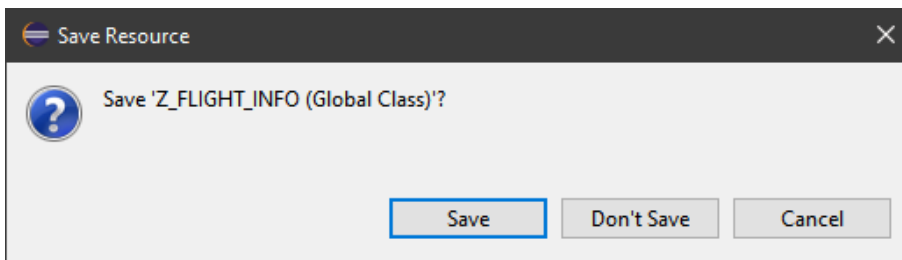


Abbildung 13 Abfragedialog zum Speichern des Codes

Es erscheint eine Dialog-Box zur Eingabe des Parameternamens.

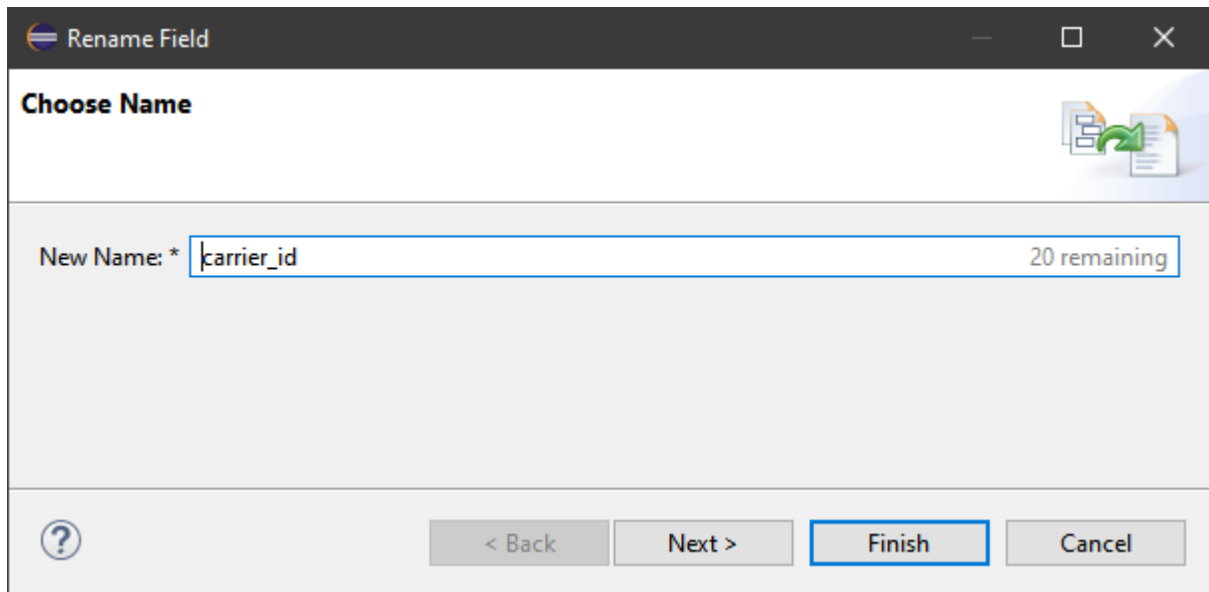


Abbildung 14 Eingabe neuer Parametername

Sofern das Objekt bereits einem Transport zugeordnet ist, kann mit "Finish" die Umbenennung direkt durchgeführt werden.

Mit "Next" können weitere optionale Einstellungen, wie der zu verwendende Transport und die Aktivierungsoption, vorgenommen werden.

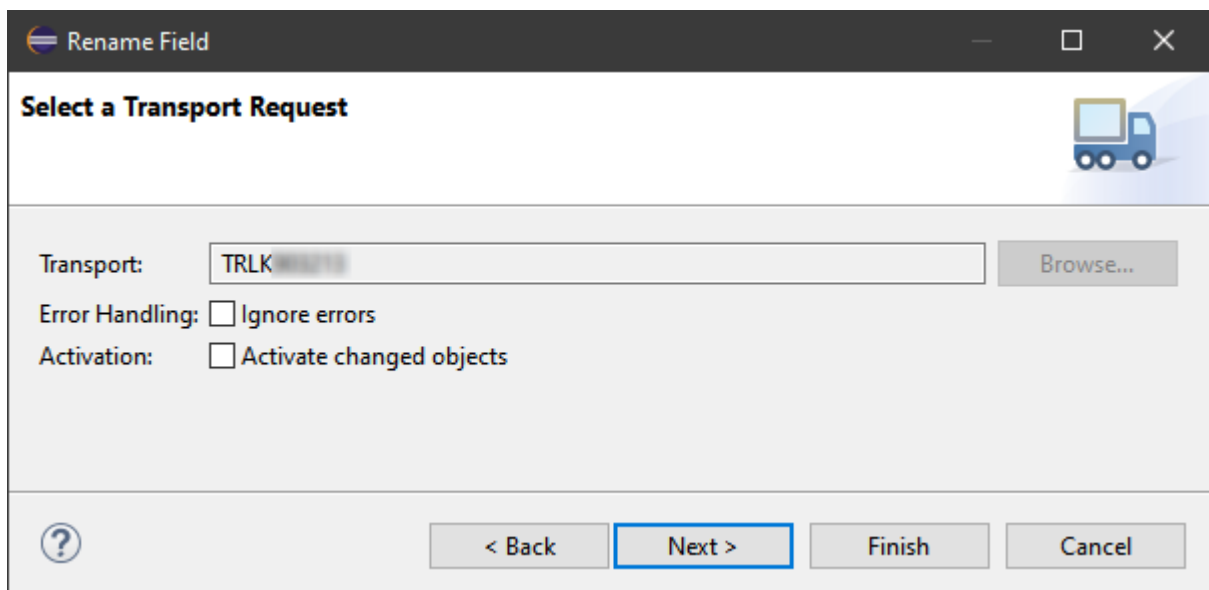


Abbildung 15 Auswahl des Transports und Optionen

Vor der finalen Durchführung kann eine Vorschau auf die Änderung angezeigt werden.

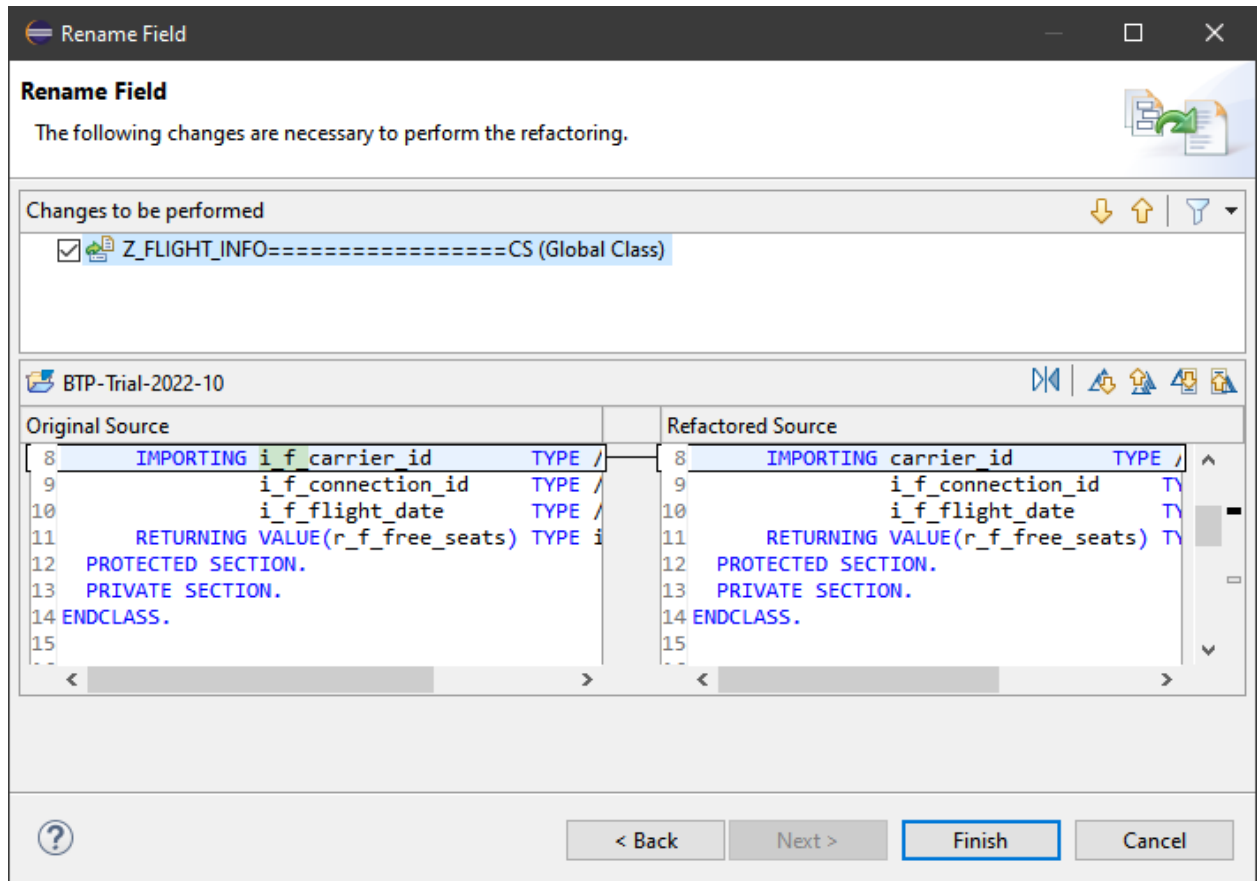


Abbildung 16 Vorschau der Umbenennung

Nach Klick auf "Finish" sind sowohl die Parameter in der Definition, die Verwendung in der Methode als auch die Parameternamen an den Stellen, an denen die Methode verwendet wird, passend abgeändert. Die hier gezeigte Umbenennung funktioniert analog in gleicher Weise für Variablen, Methoden und sogar Klassennamen.

Somit zeigt sich hier ein sehr starker Vorteil gegenüber den SAP-GUI-basierten Entwicklungswerkzeugen, in denen eine objektübergreifende Umbenennung nicht automatisiert möglich ist.

Damit sind die ersten Schritte gemacht, die grundlegenden Funktionen und Arbeitsweisen in ADT sind bekannt und können nun angewendet werden. Die erste Hürde ist genommen und die Basis für die Anwendung der zahlreichen Funktionen der ABAP Development Tools, die in den folgenden Abschnitten detailliert erläutert werden, geschaffen.

3.2 Funktionen von ADT

Der vorige Abschnitt dient vor allem dem Einstieg in die Arbeitsweise und dem Entwickeln mit den ABAP Development Tools in Eclipse. Der Abschnitt Funktionen widmet sich vorrangig der Beschreibung der zahlreichen Features und der Vermittlung von Best Practices im Umgang mit ADT im täglichen Arbeiten.

Nachdem ADT erfolgreich eingerichtet und ein Projekt, d. h. eine Verbindung zu einem Netweaver On-Premise oder eine ABAP Cloud Environment in der SAP BTP hergestellt wurde, kann mit der Entwicklung begonnen werden. In den nachfolgenden Kapiteln bieten wir einen Überblick über die Möglichkeiten, welche die ADT bei der Entwicklung von neuen Objekten und auch bei der Erweiterung/Überarbeitung ("Refactoring") von bestehenden Objekten bietet.

3.2.1 Übergreifende Features

3.2.1.1 Workspaces

Als Hauptebene der Arbeitsstrukturierung und Ablage der Eclipse- und ADT-Konfiguration dienen die sogenannten Workspaces. Beim ersten Start von Eclipse erscheint die Abfrage, in welchem Verzeichnis der Workspaces abgelegt werden soll.

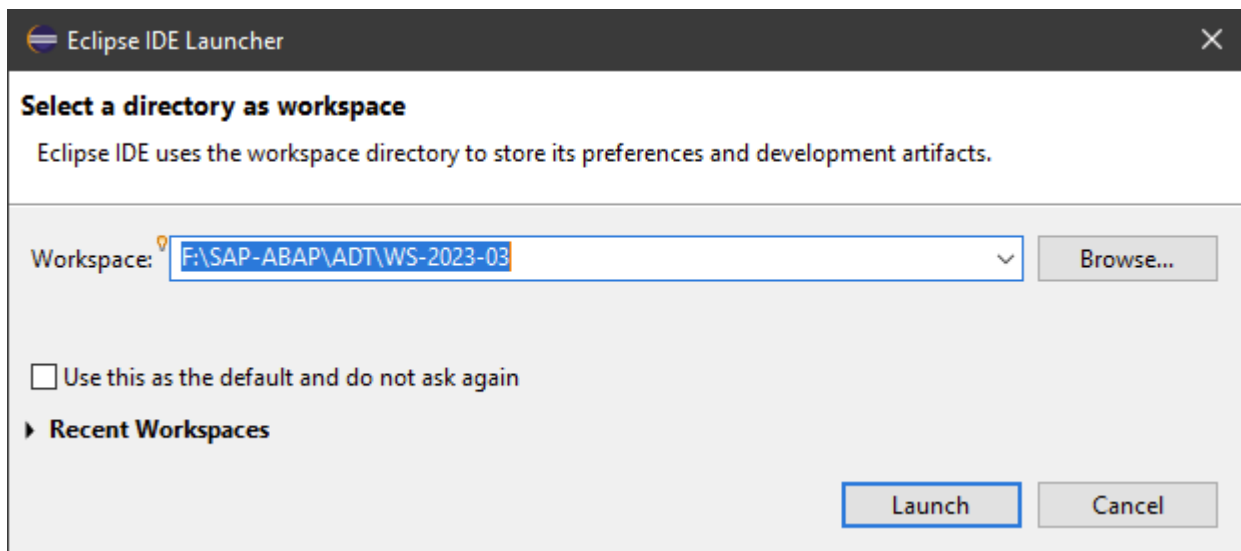


Abbildung 17 Abfrage des Workspace-Verzeichnisses

Eine Möglichkeit, wie die Verzeichnisstruktur gestaltet sein kann, findet sich in [Kapitel 6 - Best Practices Eclipse Konfiguration](#).

In diesem Verzeichnis werden zahlreiche Konfigurationseinstellungen abgelegt. Dies sind u.a.

- die Projekte und damit zugreifbaren SAP-Systeme,
- welche Favorite-Packages in den Projekten verwendet werden,
- welche Perspektiven verwendet werden,
- welche Sichten und Objekte geöffnet sind.

Wer mit einem Workspace auskommt, kann den Haken bei “Use this as default ...” anklicken, damit zukünftig dieser Workspace ohne Nachfrage verwendet wird. Diese Einstellung ist in den Einstellungen jederzeit änderbar.

Für die meisten Fälle ist ein Workspace ausreichend. Arbeiten Sie in mehreren Projekten mit unterschiedlichen Systemlinien oder mit unterschiedlichen Kunden, können die Workspaces helfen, die jeweils genutzte Systemumgebung übersichtlich zu halten und für jede Situation die effizienteste Konfiguration zur Verfügung zu haben.

Falls Bedarf besteht, einen neuen Workspace zu erstellen oder zu wechseln, wird mittels

File → Switch Workspace

entweder ein vormals geöffneter Workspace aus der Liste ausgewählt oder mittels

File → Switch Workspace → Other

der Workspace-Dialog aufgerufen.

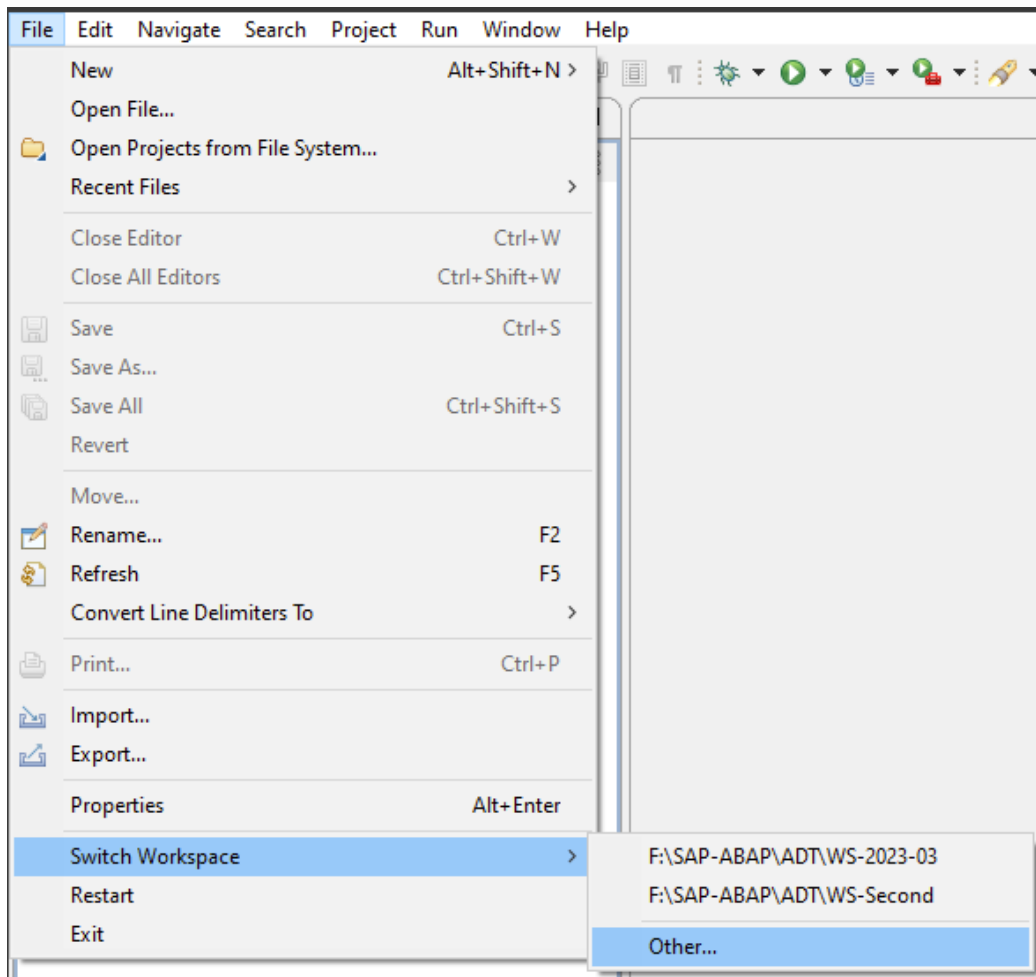


Abbildung 18 Wechseln des Workspace

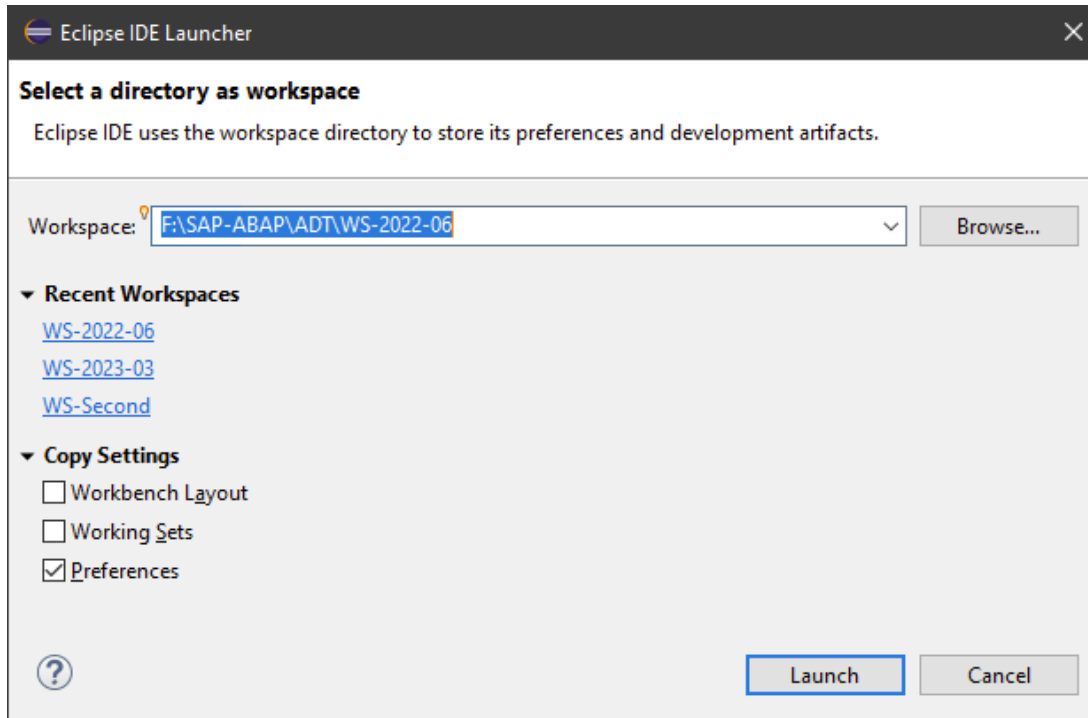


Abbildung 19 Der Workspace-Dialog

Wenn Sie einen neuen Workspace anlegen möchten, geben Sie im Feld Workspace einen neuen Namen ein. Dadurch wird nach Klick auf Launch ein neuer Workspace mit den aktuellen Einstellungen angelegt. Welche Settings aus dem Quell-Workspace übernommen werden sollen, legen Sie mittels der Copy Settings fest. Alternativ können Sie hier einen der angezeigten Workspaces unter Recent Workspaces direkt über das Anklicken der blau hinterlegten Links aufrufen.

3.2.1.2 Project Explorer

Der **Project Explorer** ist ein zentraler Bestandteil zur Navigation in den eingebundenen Systemen. Sobald Sie ein neues System als "ABAP Project" einbinden, taucht es in der Liste auf. Wenn Sie sich an einem System anmelden und aufklappen, erhalten Sie je nach System weitere Informationen zu freigegebenen Objekten, Favoriten, inaktiven Objekten etc. Diese **Repository Trees** können frei definiert und angepasst werden. Auf Ebene des Pakets verhält sich der View wie die SE80 und bildet Objekt-Hierarchien ab, durch die Sie navigieren können.

3.2.1.3 Favorite Packages

Für die tägliche Arbeit empfiehlt es sich, Pakete als Favoriten hinzuzufügen, in denen man regelmäßig tätig ist oder die in die persönliche Zuständigkeit fallen. Damit hat man eine gute Übersicht und findet schnell “seine” Objekte.

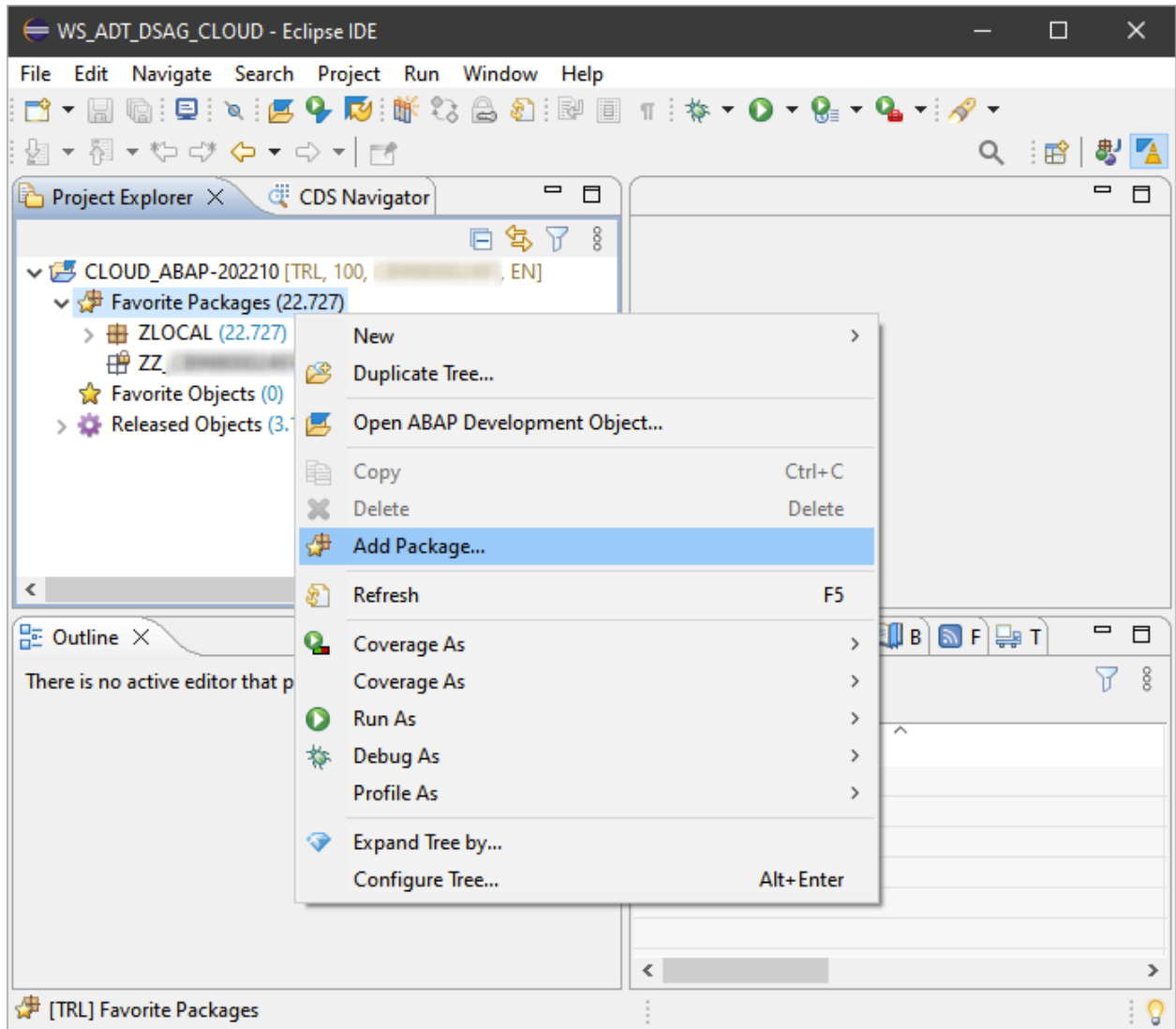


Abbildung 20 Hinzufügen von Packages zu den Favoriten

Für eine Ablage und Organisation von Favoriten auf Objektebene kann das Plug-in “ABAP Favorites” empfohlen werden, das über den Plug-in-Installationsmechanismus in Eclipse installiert werden kann, siehe [Kapitel 7 - Plug-ins](#).

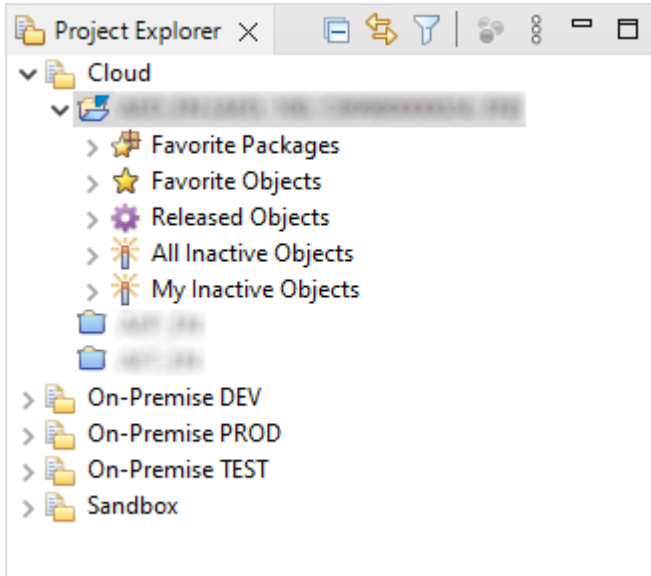


Abbildung 21 Detailbild Project Explorer mit Buttonleiste

In der Buttonleiste gibt es noch einige nützliche Funktionen, mit denen Sie Einstellungen an der View vornehmen können:

- Doppelpfeil (Link with Editor) - Objekte, die im Editor im Fokus sind, werden im Project Explorer angezeigt, es wird dazu die Hierarchie geladen.
- Drei Punkte (View Menü) - Weitere Einstellungen der Views, um zum Beispiel Working Sets anzulegen. Hierbei handelt es sich um Ordner, mit denen man Systeme gruppieren kann (siehe Screenshot oben).

3.2.1.4 Working Sets

Wenn Sie als Entwickler mit mehreren Systemlinien arbeiten, empfehlen wir die Verwendung der Working Sets. Diese ermöglichen es, Projekte in Eclipse zu gruppieren und somit mehrere Systeme übersichtlich darzustellen.

Über das Drei-Punkte-Icon in der rechten oberen Ecke des Project Explorer finden sich diverse Einstellungsmöglichkeiten. Unter anderem können hier die Working Sets erstellt und konfiguriert werden.

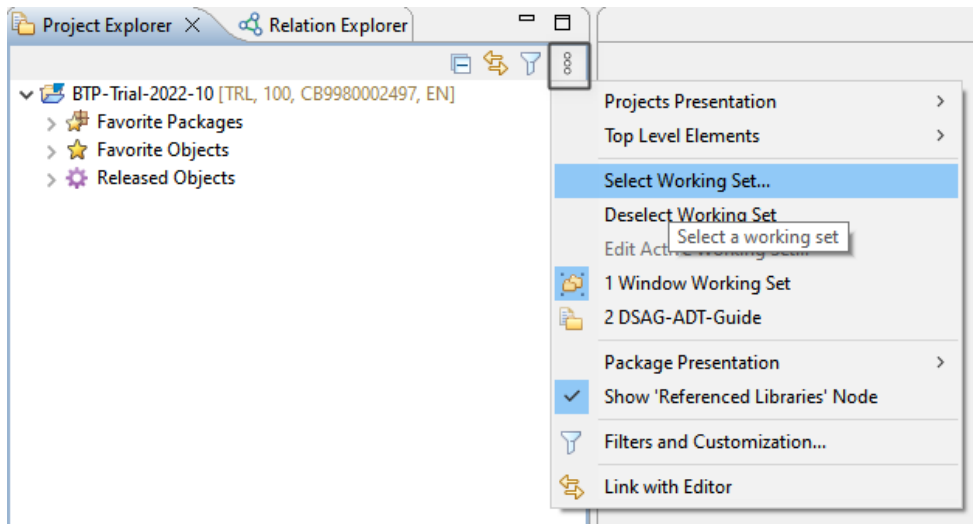


Abbildung 22 Working-Sets-Einstellungen

Über den Dialog können nun Working Sets angelegt werden (New) und die Zuordnungen erfolgen (Edit).

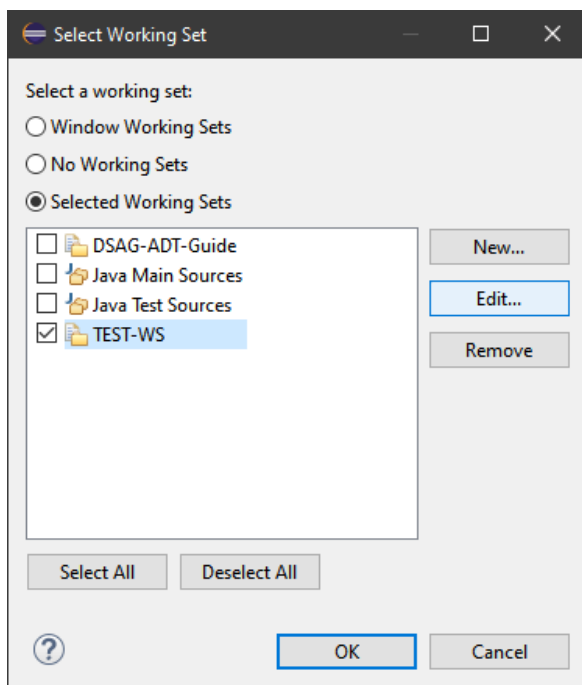


Abbildung 23 Anlage und Bearbeitung der Working Sets

Über den Edit-Button können im Folgedialog die gewünschten Projekte dem Working Set zugeordnet werden.

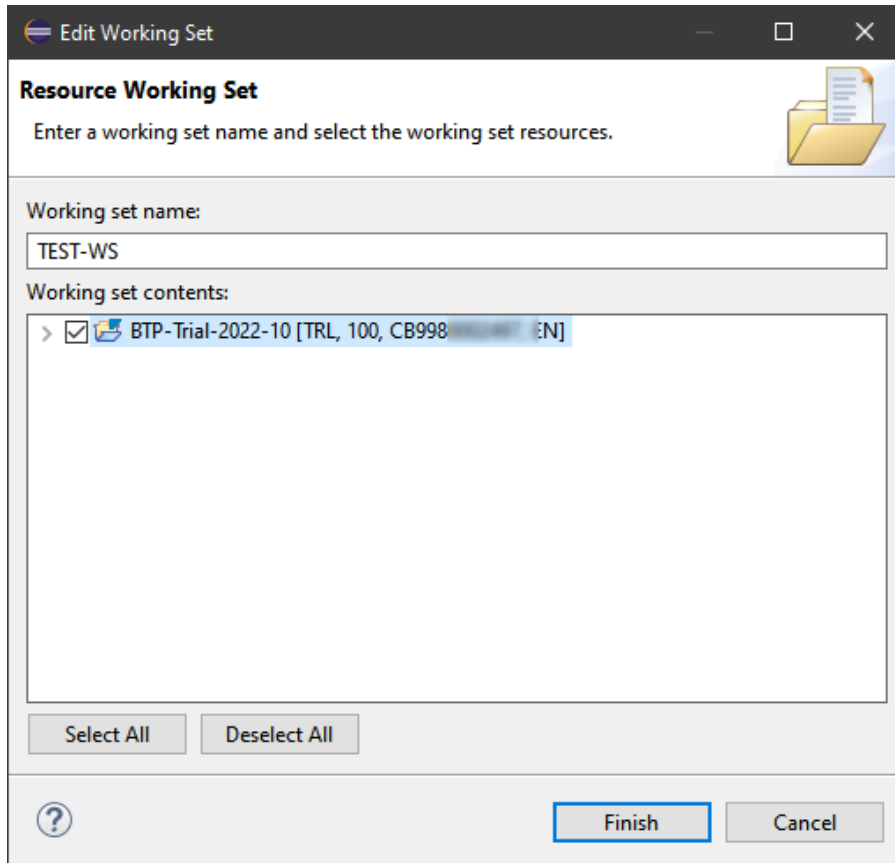


Abbildung 24 Zuordnung Projekt zu Working Set

Diese Funktion ermöglicht eine übersichtliche Strukturierung der Systeme nach Systemlandschaft oder ggf. nach Projekt bzw. Kunde. Abschließend muss noch die Anzeige der Top Level Elements auf Working Sets eingestellt werden.

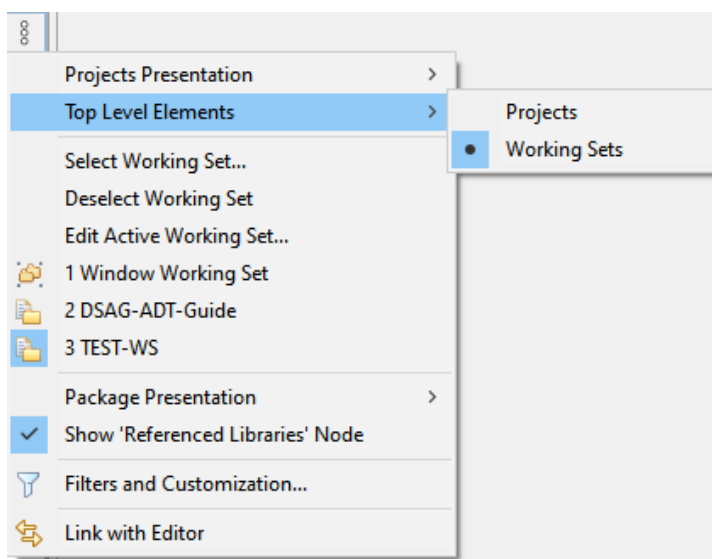


Abbildung 25 Einstellung der Projekt-Explorer-Anzeige

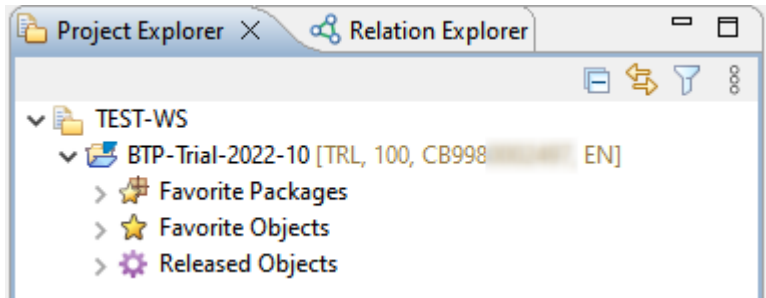


Abbildung 26 Darstellung Projekte in Working Sets

3.2.2 Suche und Navigation

Die Suche nach Objekten in Eclipse ist zentraler Bestandteil der täglichen Arbeit, ebenso wie die Navigation zwischen den Objekten oder auch die Vorwärtsnavigation. In diesem Abschnitt erfahren Sie mehr über die Suche und Navigation zwischen ABAP-Objekten.

3.2.2.1 Objekte suchen

Um ein Objekt im System zu suchen bzw. zu öffnen, können Sie den Dialog “Open ABAP Development Object” nutzen (erreichbar über die Tastenkombination **STRG+SHIFT+A**).

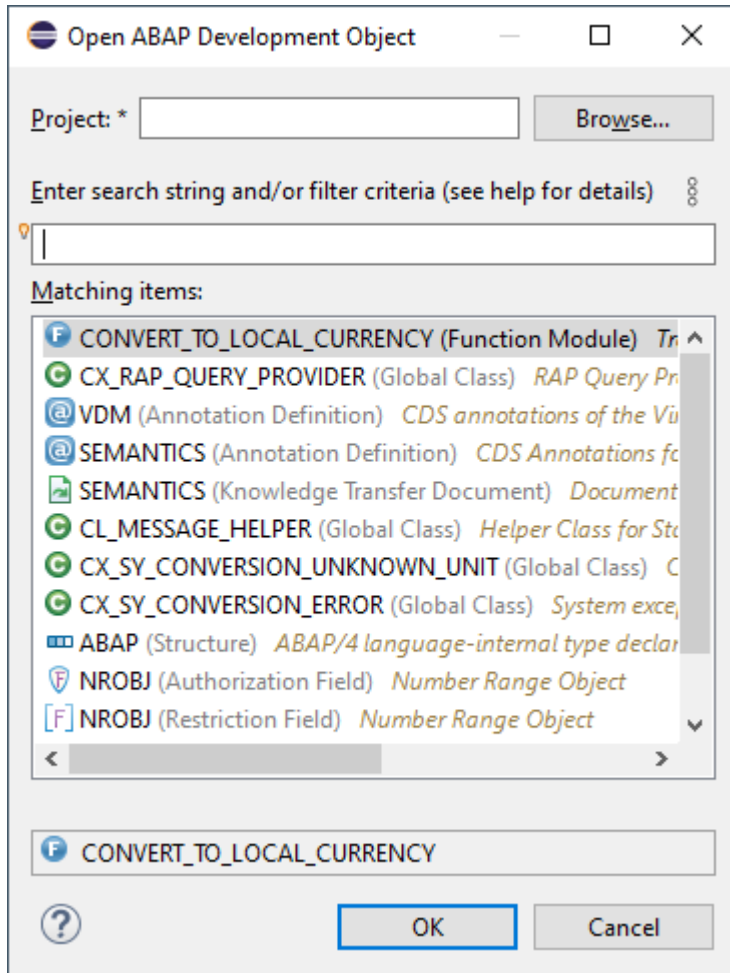


Abbildung 27 Dialog zur Objektsuche

Im Dialog haben Sie die Möglichkeit, im oberen Teil das ABAP-Projekt zu ändern und damit zu entscheiden, auf welchem System Sie nach dem Objekt suchen möchten. Wurde im Suchfeld nichts eingegeben, erhalten Sie eine Historie der zuletzt geöffneten Objekte. Über das *Fragezeichen* im unteren Bereich erhalten Sie weitere Informationen, Tipps und Tricks zur Nutzung der Suche. Detaillierte Informationen zur Objektsuche finden Sie im [User-Guide](#).

3.2.2.2 Objekte filtern

In der Objektsuche haben Sie nun die Möglichkeit, mit Such-Strings und Pattern zu arbeiten, um die Ergebnismenge weiter einzuschränken. Das Feld unterstützt den “Content Assist” (**STRG+LEERTASTE**), um weitere Einschränkungen und Filter zu verwenden. Eine einfache Suche könnte wie folgt aussehen:

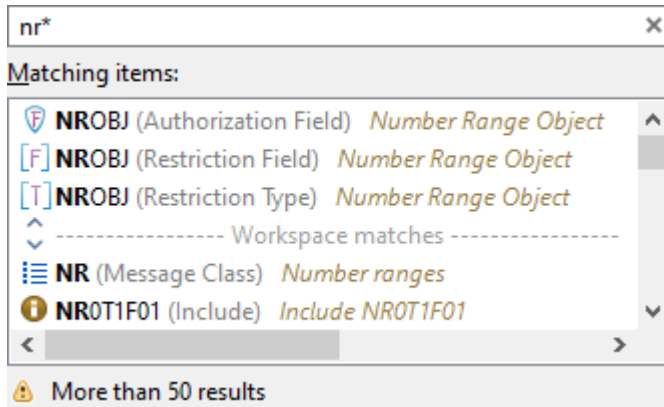


Abbildung 28 Ergebnis der Suche

Allerdings werden mehr als 50 Ergebnisse (Standardeinstellung) angezeigt und wahrscheinlich ist das gewünschte Ergebnis nicht mit in der Ergebnismenge erhalten. In diesem Fall können Sie den “Content Assist” aufrufen, um weitere Optionen zum Filtern zu erhalten.

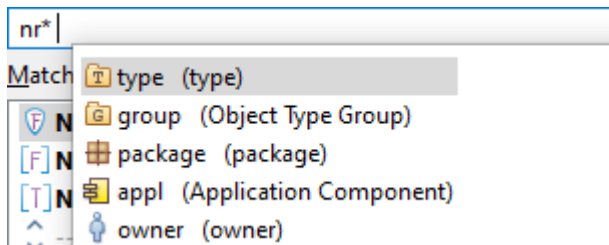


Abbildung 29 Anzeige weiterer Suchoptionen

Wenn Sie zum Beispiel nun auf Tabellentypen einschränken möchten, dann würden Sie weiter nach dem Typ (TYPE) einschränken. Der “Content Assist” schlägt weiterhin auch die verschiedenen Typen von Objekten vor, sodass Sie auch den Tabellentyp (TTYP) finden.

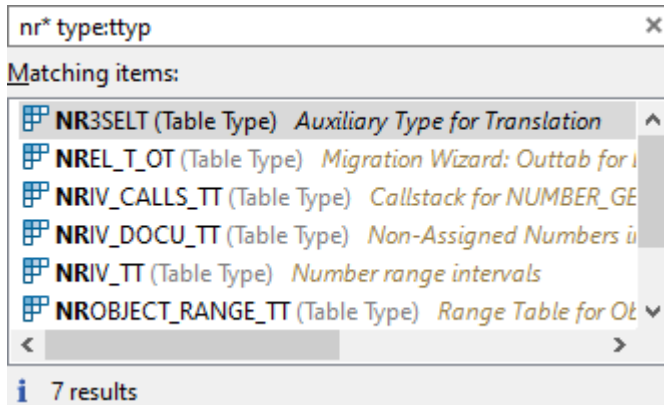


Abbildung 30 Ergebnis mit Objekt- und Typfilter

Die weiteren Filter und Typen können frei miteinander kombiniert werden, um die gewünschte Zielmenge oder das Zielobjekt zu finden. Mit einem Doppelklick auf den gewünschten Eintrag wird das Objekt dann im Editor angezeigt.

Um die Suche zu beschleunigen, empfiehlt es sich, den Typ des gewünschten Objektes vorzugeben, da sonst die Suche – verglichen mit der gewohnten Geschwindigkeit in der SE80/SE11/etc. – sehr lange dauert.

3.2.2.3 Navigation

In der ABAP Workbench funktioniert die [Navigation](#) zum nächsten Objekt über einen Doppelklick auf den entsprechenden Ausdruck im Quellcode. In Eclipse wird hier lediglich der Quelltext markiert. Um die Vorwärtsnavigation auszulösen, gibt es drei Möglichkeiten:

- Cursor auf dem Objekt platzieren, F3 drücken
- Mit gedrückter **STRG**-Taste das Objekt anklicken
- In der Oberfläche wird ein klickbarer Link angeboten (z. B. Datenelement → Domäne)

Das Objekt wird in einem neuen Tab innerhalb des Editors geöffnet, das Quellobjekt bleibt weiterhin geöffnet und Sie können zwischen den zuletzt geänderten Objekten sehr einfach über die Tastenkombinationen navigieren:

- **ALT+Pfeil rechts**: vorwärts
- **ALT+Pfeil links**: rückwärts

Dies kann analog auch mit den Pfeiltasten im Bereich der Drucktastenleiste durchgeführt werden. Im Bereich der Drucktastenleiste gibt es dafür auch verschiedene Optionen, um zum letzten verwendeten Tab (**ALT+Pfeil links**) zu gelangen.

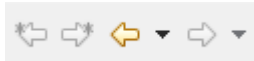


Abbildung 31 Navigations-Ikonen

3.2.2.4 ABAP Repository Tree anzeigen

Nachdem Sie ein Objekt gefunden haben, möchten Sie in vielen Fällen in diesem Paket weiter arbeiten oder recherchieren. Dazu können Sie sich den Objektbaum laden lassen, indem Sie im Project Explorer den doppelten Pfeil (“Link with Editor”) aktivieren.

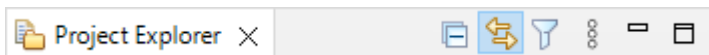


Abbildung 32 Aktionen für den Project Explorer

In diesem Fall wird die Pakethierarchie zu dem im Editor fokussierten Objekt geladen. Im Anschluss können Sie im Project Explorer über die weiteren Objekte und Strukturen navigieren.

Die Arbeit mit dem Repository Tree ist detailliert im [User-Guide](#) beschrieben.

3.2.3 ABAP Editor

Der [ABAP Editor](#) ist ein einfacher Texteditor, der die rein textuelle Erstellung von ABAP-Artefakten ermöglicht. Über die Kontext-Funktion können dort die wichtigsten Funktionen wie Quick Fixes, Refactoring-Funktionen und Formatierungsfunktionen aufgerufen werden. Der Einstieg in den ABAP Editor ist im Abschnitt [Das Erstellen einer Klasse im Textmodus](#) näher beschrieben.

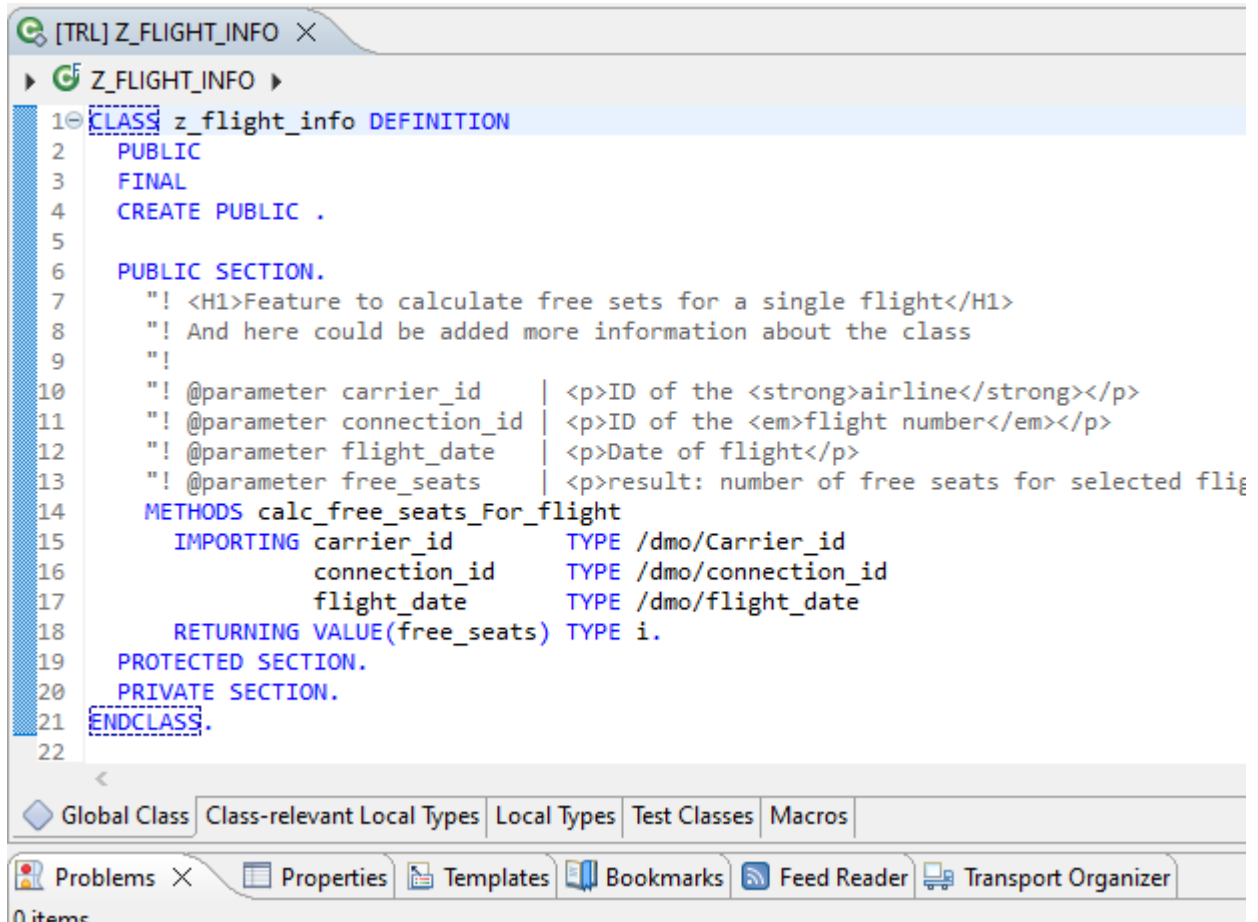


Abbildung 33 ABAP Editor – Hauptfenster

3.2.3.1 Element Info

Über die Positionierung des Cursors auf einem Objekt und dem Shortcut **F2** erscheint ein Pop-up mit Zusatzinformationen. Hier am Beispiel einer Methode und eines Datenelements:

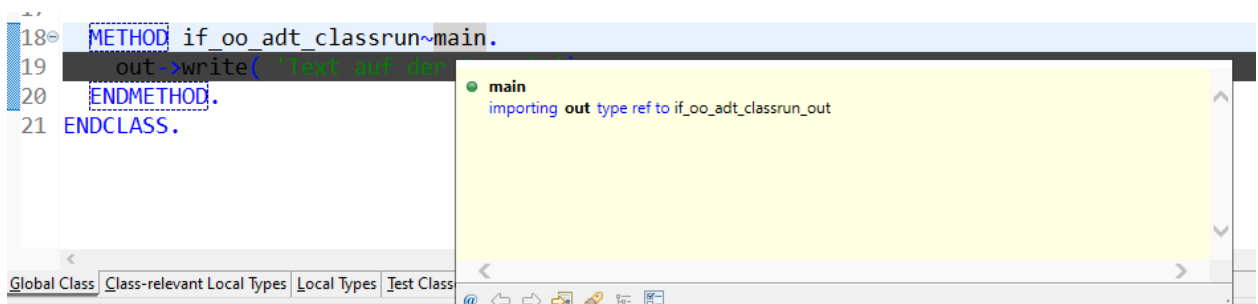


Abbildung 34 Element Info für eine Methode

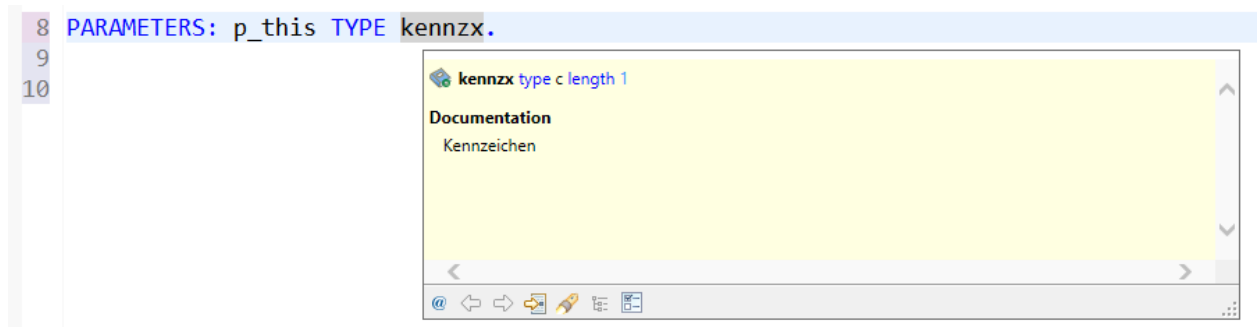



Abbildung 35 Element Info für ein Datenelement

In der **Element Info** sind somit direkt Details ersichtlich, für die man in der alten Welt noch im System navigieren musste. Zusätzlich lässt sich im Element Info weiter navigieren, um Details tieferer Ebenen zu sichten, z. B. die hinterlegte Domäne etc.

Die ABAP Element Info existiert auch als eigenständige View. Diese kann über das Menü `Windows → Show View → ABAP Element Info` aktiviert werden. Danach wird über **“Link with Selection”**  bei jedem Klick auf ein relevantes Entwicklungsobjekt automatisch dessen Element Info sowie die Dokumentation aus ABAP Doc angezeigt:

```

7 PUBLIC SECTION.
8   METHODS: constructor.
9 PROTECTED SECTION.
10 PRIVATE SECTION.
11 DATA: class_runner TYPE REF TO if_oo_adt_classrun.
12 ENDCLASS.
13
14
15

```

Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

Properties | Search | Templates | Bookmarks | Feed Reader | Transport Organizer | ABAP Unit

if_oo_adt_classrun in **seo_adt**

- main

Documentation

Implement this interface to execute an ABAP class (Classrun)

Provides a light-weight solution for executing an ABAP program without launching the integrated SAP GUI. Furthermore, this feature enables you to display any kind of text and/or content of data into the Console View.

Example:

```


METHOD if_oo_adt_classrun~main.
  SELECT FROM ... FIELDS ... INTO TABLE ... .
  out->write( EXPORTING data = lt_data name = 'Meaningful description ...' ).
ENDMETHOD.

```

Output

BP_ID	COMPANY_NAME	EMAIL_ADDRESS	PHONE_NUMBER	FAX_NUMBER
0100000083	Consumer demo_an_uia@sap.com	+49 408311-357370	+49 491629-6077	
0100000084	Consumer demo_hk_uia@sap.com	+49 681933-49944	+49 720349-874614	
0100000085	Consumer demo_a2etest_uia@sap.com	+49 80642-81594	+49 246107-953045	

Abbildung 36 Anzeige der Element Info nach Auswahl des Objekts

Über “Pin this view”  wird die Information dauerhaft angezeigt, auch wenn auf ein anderes Element geklickt oder die Element Info per **F2** für ein anderes Entwicklungsobjekt geöffnet wird.

3.2.3.2 Quelltextformatierung mit dem ABAP Formatter

In der SAP GUI heißt das Werkzeug zum Formatieren des Quellcodes Pretty Printer. Das Pendant in ADT ist der **ABAP Formatter**. Er wird entweder über die Tastenkombination

SHIFT+F1

oder über das Kontextmenü im Quelltext aufgerufen.

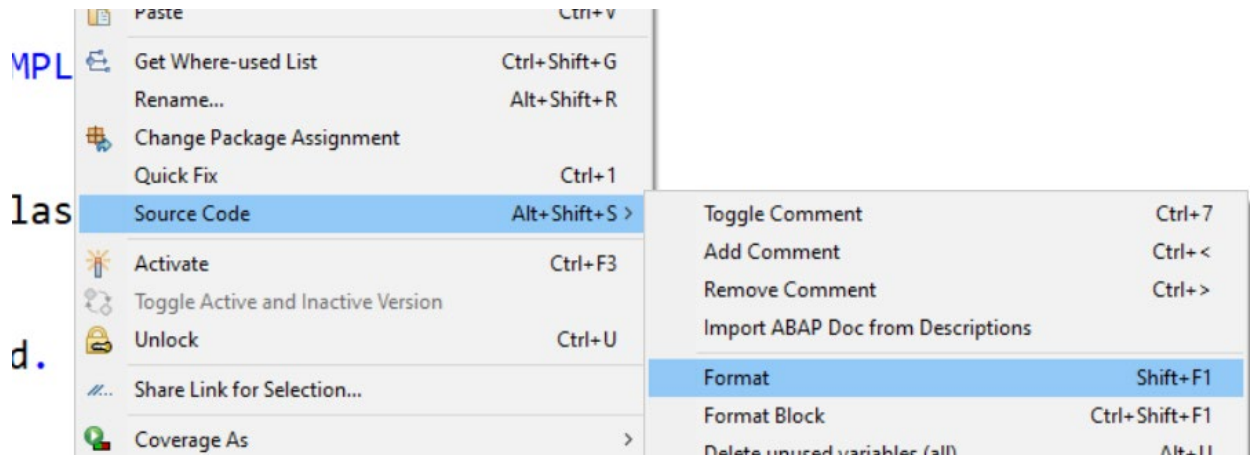


Abbildung 37 Kontextmenü für die Formatierung

Damit der ABAP Formatter seinen Dienst leisten kann, muss er vorab konfiguriert werden. Dabei legt man analog zu den Pretty-Printer-Einstellungen in der SAP GUI fest, ob Einrückungen gemacht werden sollen und wie die Groß-/Kleinschreibung formatiert wird. Dies geschieht für jedes ABAP-Projekt separat.

3.2.3.2.1 Einstellung des ABAP Formatters

Falls die Einstellungen noch nicht vorgenommen wurden, erscheint ein Pop-up-Fenster mit der Meldung, dass dies zuvor erledigt werden muss. In diesem Pop-up-Fenster ist auch ein Link zu den Einstellungen enthalten. Alternativ kann man diese auch direkt über den Kontextmenü-Eintrag *Properties* des entsprechenden Projektes aufrufen. In der folgenden Abbildung sehen Sie die Position in den Einstellungen.

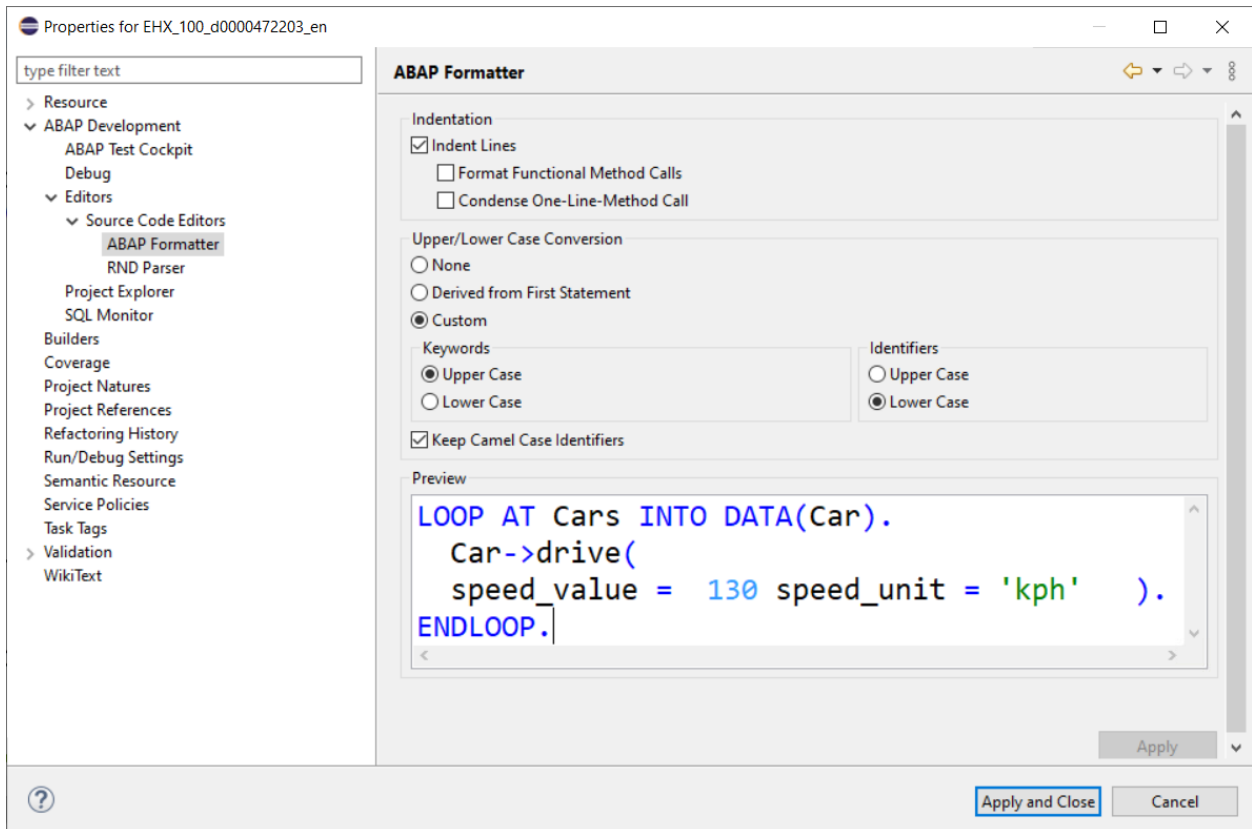


Abbildung 38 Einstellungen für den ABAP Formatter

Die Einstellungen entsprechen denen in der SAP GUI. Wenn man die einzelnen Optionen durchprobiert, sieht man im Vorschaufenster das jeweilige Ergebnis. Neu ist die Möglichkeit, dass Camel-Case-Bezeichner beibehalten werden. Das ist gerade im Zusammenhang mit den CDS Views sehr praktisch, da diese im virtuellen Datenmodell von SAP (VDM) konsequent verwendet werden.

3.2.3.3 Quick Fixes

Quick Fixes sind automatisierte Lösungen für gängige Probleme im Rahmen der Software-Entwicklung mit ABAP. Quick Fixes werden dabei nativ durch die ADT angeboten, können aber auch durch verschiedene Plug-ins erweitert werden. Die Verwendung durch die Vielzahl der verfügbaren Quick Fixes macht das Arbeiten mit

den ADT wesentlich effizienter als mit der SE80. Zusätzlich wird das Risiko von Fehlern durch manuelle Anpassungen reduziert. Die ADT erledigen die Anpassungen automatisiert und immer identisch.

Quick Fixes liefern Funktionen für zwei Bereiche:

- Automatisches Anlegen von nicht vorhandenen Objekten (z. B. Methodenimplementierungen)
- Automatisches Verändern von bestehenden Objekten ohne Veränderung der Funktionalität (sog. Refactoring, z. B. "Methode extrahieren")

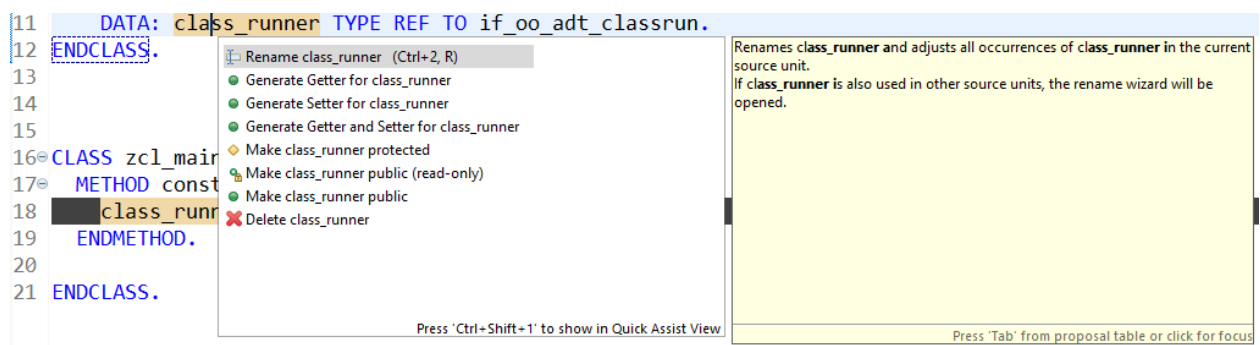


Abbildung 39 Anzeige der Refactoring Optionen

Aufgrund der Vielzahl an Quick Fixes und der ständigen Änderungen in diesem Bereich werden die einzelnen Quick Fixes hier nicht beschrieben. Ein Überblick ist in der [Dokumentation](#) zu finden.

3.2.4 Andere Objekttypen

3.2.4.1 Programme und Funktionsgruppen

Programme und Funktionsgruppen werden in der Navigation des Project Explorers im Ordner *Source Code Library* angezeigt. Unter den entsprechenden Objekttypen werden alle Komponenten entsprechend der SAP GUI angezeigt.

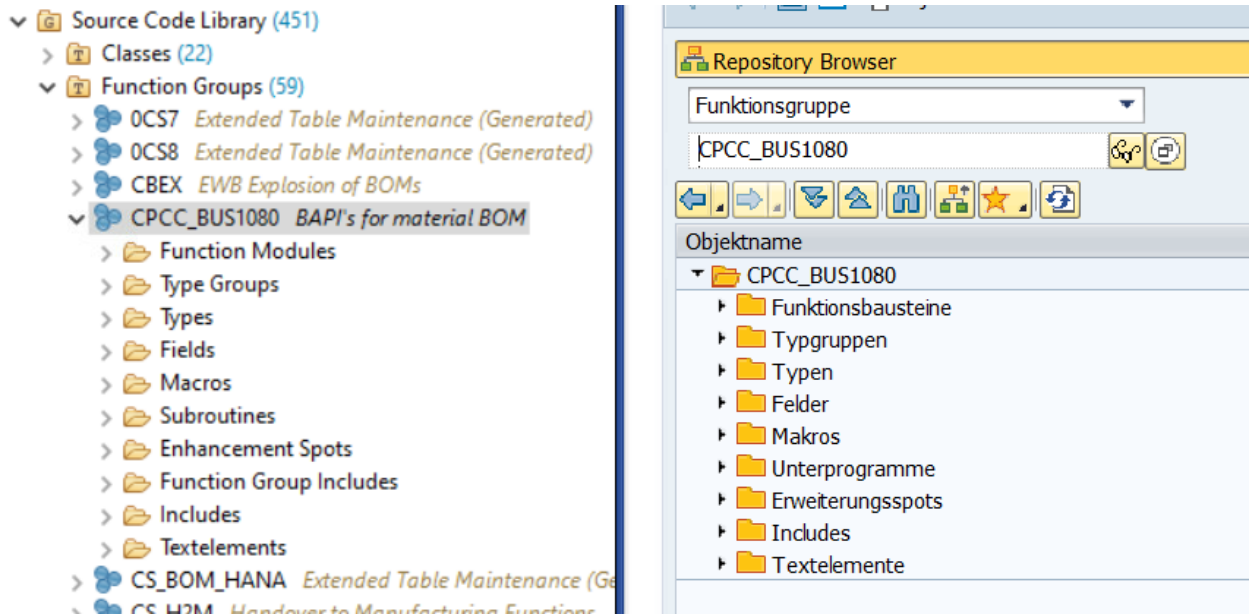


Abbildung 40 Vergleich Funktionsgruppen im Project Explorer der ADT und in der SE80

Es wird der gleiche Quelltexteditor wie für ABAP-Klassen verwendet. Damit sind alle Features vom ABAP Formatter bis zur Language-Help weitgehend identisch.

3.2.5 ABAP Views

3.2.5.1 Outline

Die **Outline** View liefert Informationen zum aktuell fokussierten Entwicklungsobjekt und löst Variablen, lokale Klassen, Typen etc. auf. Die View kann mit der SE80 verglichen werden, zeigt Ihnen aber immer nur den aktuellen Kontext des Objekts an. Im Screenshot sehen Sie eine Klasse, darunter die entsprechenden Methoden und zwei private Attribute der Klasse. Mit einem Klick auf einen Eintrag navigieren Sie an die entsprechende Stelle im Quellcode.

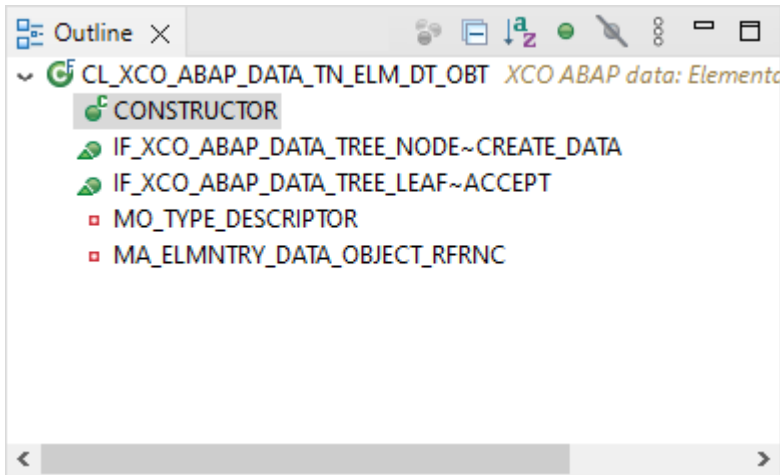


Abbildung 41 Anzeige der Eigenschaften in den Outlines

In der Button-Leiste gibt es weitere verschiedene Funktionen für den View:

- “Sort” – Sortierung der Einträge nach Alphabet oder nach Typ und Alphabet
- “Hide Non-Public Members” – Es werden nur Attribute und Methoden angezeigt, die auch von außen verwendbar sind (Thema Sichtbarkeit)

Hinweis: Im [Kapitel 7 - Plug-ins](#) finden Sie weitere Informationen zum Plug-in “Classic Outlines”, welches die Outlines erweitert.

3.2.5.2 Problems

Die **Problems View** ist wahrscheinlich eine der wichtigsten Views. Sie zeigt Informationen zu Fehlern innerhalb von Entwicklungsobjekten in Form einer Liste. Die View aktualisiert sich automatisch, wenn neue Fehler hinzukommen oder bestehende korrigiert werden. Ein andauerndes “Prüfen” des Source-Codes ist nicht notwendig.

Dargestellt werden in der Standardkonfiguration dabei systemübergreifend alle Fehler in den eigenen geöffneten Objekten, nicht nur des aktuell in Bearbeitung befindlichen Objekts.

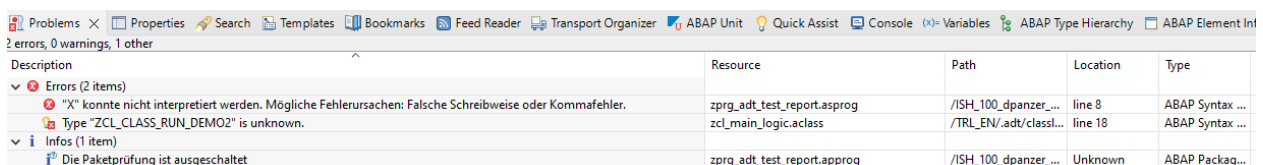


Abbildung 42 Anzeige der Meldungen im Problems View

Durch Doppelklick kann an die entsprechende Stelle im Source-Code navigiert werden.

Über den Button  kann der View weiter konfiguriert werden:

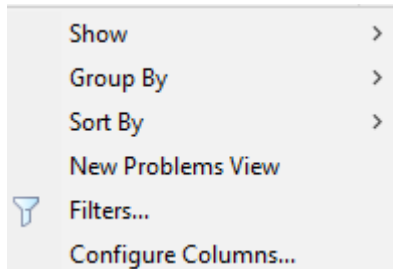


Abbildung 43 Anzeige der Optionen des Views

- Unter “Show” kann konfiguriert werden, welche Fehler/Warnungen in der Problems View angezeigt werden, z. B. nur die des aktuell in Bearbeitung befindlichen Entwicklungsobjekts oder alle.
- “Group by” ermöglicht eine Gruppierung nach verschiedenen Kriterien, üblicherweise wird hier nach “Severity”, d. h. Fehler/Warnung/Info gruppiert.
- Über “Sort By” kann die Reihenfolge der Darstellung verändert werden.
- “New Problems View” dupliziert die View.
- “Filters” ermöglicht, die Ergebnisliste noch einmal im Detail bis hinunter auf den Entwicklungsobjekttyp zu filtern.
- “Configure Columns” ermöglicht es, Spalten ein- und auszublenden sowie die Reihenfolge der Spalten zu ändern.

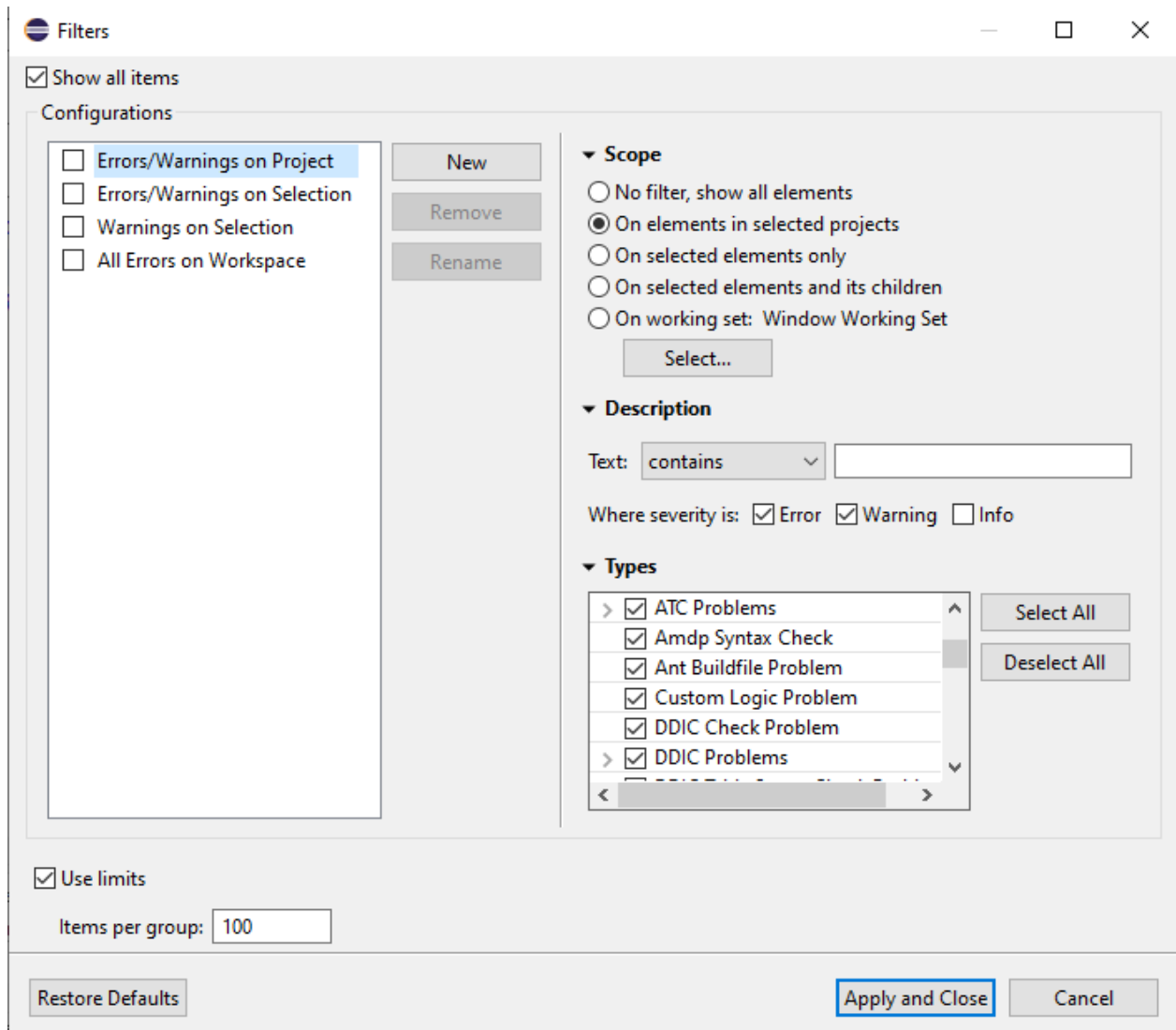
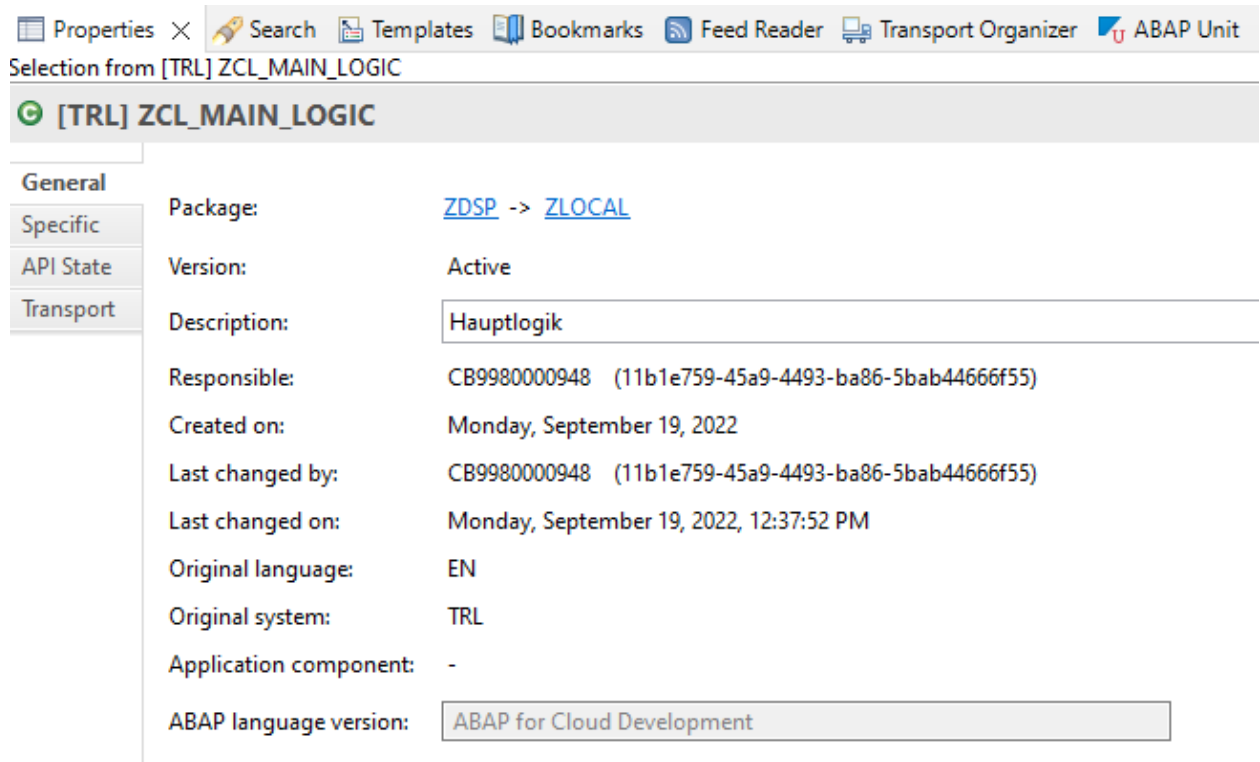


Abbildung 44 Konfiguration der angezeigten Punkte

3.2.5.3 Properties

Die **Properties View** sammelt Informationen, die in der klassischen SAP GUI unter "Eigenschaften" dargestellt werden. Dazu zählen beispielsweise:

- Zustand Entwicklungsobjekt (aktiv/inaktiv)
- Erstellungs- und Änderungsinformationen
- Paketuordnung



Properties View for [TRL] ZCL_MAIN_LOGIC

General

Specific

API State

Transport

Package: [ZDSP](#) -> [ZLOCAL](#)

Version: Active

Description: Hauptlogik

Responsible: CB9980000948 (11b1e759-45a9-4493-ba86-5bab44666f55)

Created on: Monday, September 19, 2022

Last changed by: CB9980000948 (11b1e759-45a9-4493-ba86-5bab44666f55)

Last changed on: Monday, September 19, 2022, 12:37:52 PM

Original language: EN

Original system: TRL

Application component: -

ABAP language version: ABAP for Cloud Development

Abbildung 45 Properties View

Im Bereich "Specific" werden objektspezifische Eigenschaften angezeigt und können dort teilweise geändert werden. Für Klassen ist hier beispielsweise die Festpunktarithmetik einstellbar.

Im Bereich Transport ist eine Historie der Transportaufträge gelistet, in denen das Objekt enthalten ist.



[TRL] ZCL_MAIN_LOGIC

Unreleased Transport Requests (1)

Request	Description	Owner	Created	Changed	# Objects	# Entries	Status
TRLK901142	Dummy	CB9980000948 (11b1e759-45a9-449...	1 day ago	1 day ago	3	3	Modifiable

Abbildung 46 Historie der Transporte

Über das Kontextmenü kann wiederum in den Transportauftrag zur weiteren Analyse verzweigt werden.

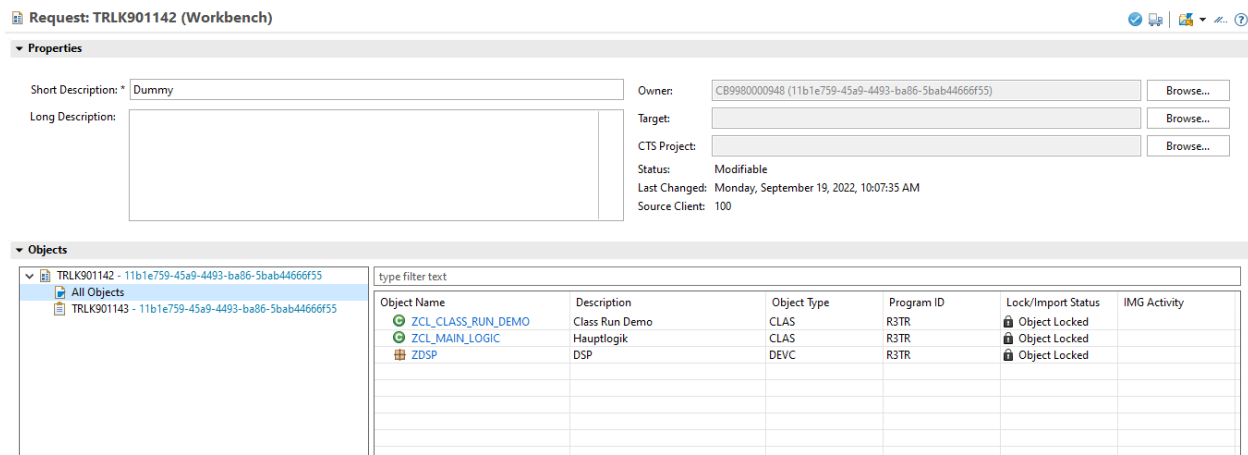


Abbildung 47 Transport View

Abhängig vom Typ des geöffneten Entwicklungsobjektes existieren weitere Bereiche, die spezifische Informationen zum jeweiligen Objekt enthalten.

Um mehrere Objekte parallel zu sichten, lassen sich über den Button  auch mehrere Properties Views erzeugen.

Der Properties View hat im Vergleich zur klassischen SAP-GUI-Entwicklung mehrere Vorteile:

- Er ist (in der Regel) ständig eingeblendet und verfügbar, so dass nicht aufwendig navigiert werden muss.
- Er aggregiert Informationen, die zuvor nur über mehrere Transaktionen/Tabs/Reiter identifizierbar waren.

3.2.5.4 Templates

Code Templates sind vorgefertigte Muster an Quellcode, die beliebig in eine Anwendung implementiert werden können. Diese Muster geben statischen Quellcode wieder und besitzen dynamische Elemente in Form von Variablen. In der Standardauslieferung der ADT werden einige Templates ausgeliefert.

TEMPLATE VIEW

Templates werden über einen eigenen View zur Verfügung gestellt (Window → Show View → Templates) und können auch über die Einstellungen angepasst werden (General → ABAP Development → Editors → Source Code Editors → ABAP Templates).

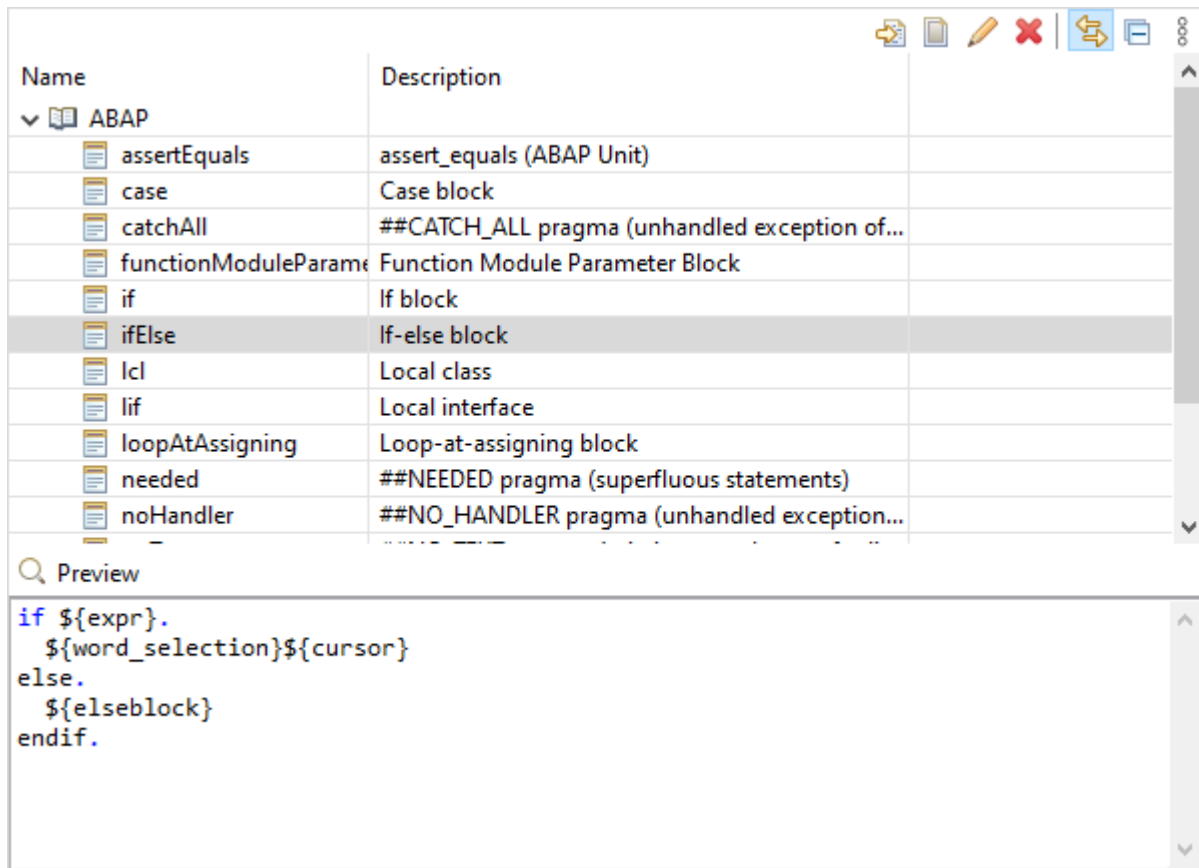


Abbildung 48 Template View Browser

Der View besteht im oberen Bereich aus einer Button-Leiste, einer Liste der Code Templates und im unteren Teil aus einem Preview des Templates.

TEMPLATE VERWENDEN

Das **Template** kann direkt im Quellcode verwendet werden. Beginnen Sie, den Namen zu tippen, und wählen Sie mit Hilfe des "Content Assist" das passende Template aus (hier die ersten beiden Einträge).

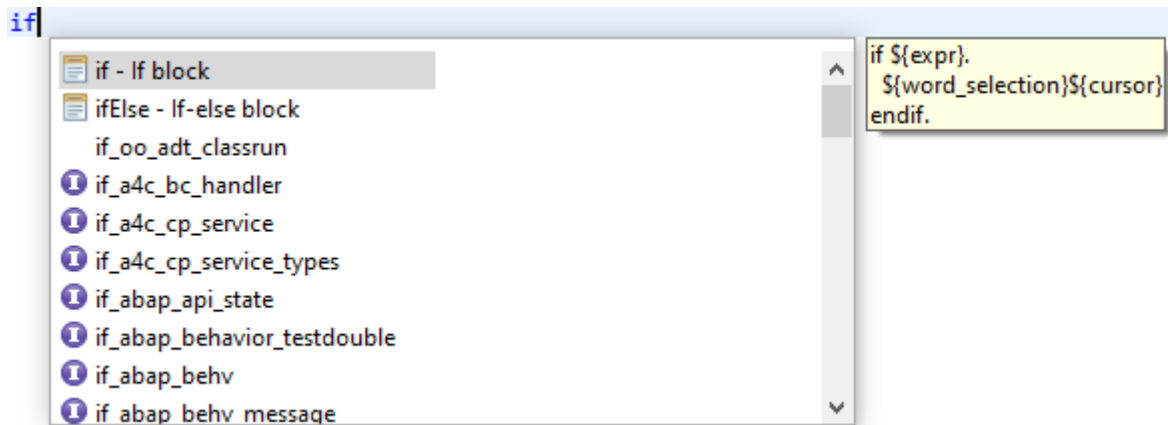


Abbildung 49 Auswahl der Templates im Content Assist

Das gesamte Template wird eingefügt, und Sie können damit beginnen, die Platzhalter (Variablen) zu befüllen. Mit dem Tabulator springen Sie zwischen den einzelnen Platzhaltern hin und her.

Häufig verwendete Templates zur Nutzung sind beispielsweise:

- **lcl** - Anlage einer lokalen Klasse
- **testClass** - Anlage einer Testklasse
- **functionModuleParameter** - Beispielschnittstelle für Funktionsbausteine

Die Anlage von eigenen Templates eignet sich sehr gut zur Einsparung von Entwicklungsaufwand bei wiederholenden Aufgaben oder ähnlichen Code-Abschnitten. Weiterhin können sie in Schulungen hilfreich sein, wenn Sie größere Code-Abschnitte einfügen möchten, ohne per Copy-and-paste zu arbeiten.

TEMPLATE ANLEGEN

Ein Template kann grundsätzlich mit allen in ABAP verwendeten Sprachbefehlen definiert werden (Quellcode, Kommentare). Variable Teile des Templates können Sie mit Platzhaltern versehen (“\${placeholder}”). Zur Ableitung von kontextspezifischen Informationen stehen auch Variablen zur Verfügung. Diese gibt es für die folgenden Szenarien:

- Name des Objekts
- Name des Pakets
- ID des Systems
- User, Datum und Uhrzeit
- Aktuelles Jahr
- Cursor-Position nach dem Einfügen

Gleichnamige Platzhalter werden nach dem Einfügen immer einheitlich angepasst (z. B. der Name der Klasse).

VERFÜGBARKEIT VON TEMPLATES

Templates stehen innerhalb eines Eclipse-Workspace zur Verfügung, sind im Gegensatz zu eigenen Mustern aber über Systeme hinweg verfügbar. Templates können über die Einstellungen importiert und exportiert werden, um sie unter Kollegen/Mitarbeitern auszutauschen. Eine zentrale An- und Ablage von Templates für alle Entwickler ist nicht möglich.

3.2.5.5 Verwendungsnachweis/Where-Used-List

Der **Verwendungsnachweis** findet alle statischen Verwendungen eines Entwicklungsobjektes im Source-Code des aktuellen Projektes. Der Verwendungsnachweis ist über die Tastenkombination

STRG+SHIFT+G (Get-Where-Used-List)

erreichbar. Das Ergebnis wird im Reiter "Search" dargestellt:

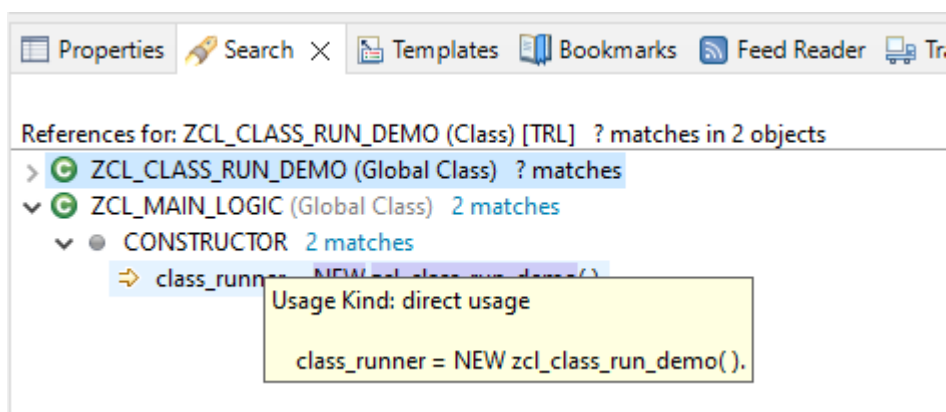


Abbildung 50 Ergebnis des Verwendungsnachweises/Where-Used-List

Durch die Verwendung der Filterfunktion mit Hilfe des Filter-Icons kann hierbei auf Pakete, Objekttypen und Benutzer eingeschränkt werden. Auch hier kann per **STRG+SPACE** die automatische Vervollständigung genutzt werden, um Objekte schneller zu finden.

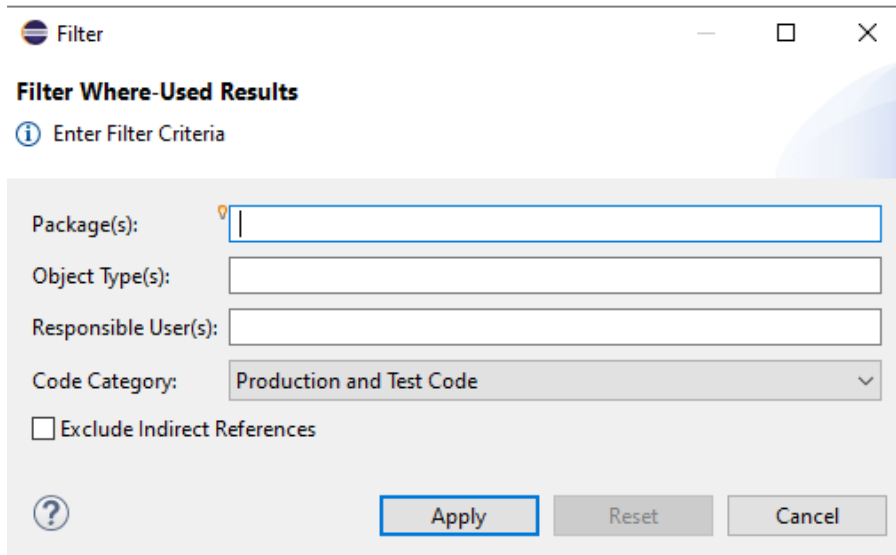


Abbildung 51 Filter für Where-Used-Search

Über **STRG+.** und **STRG+,** lassen sich die Fundstellen komfortabel browsen, d. h. die jeweils nächste oder vorherige Fundstelle anzeigen.

3.2.5.6 Bookmarks

Bookmarks sind "Lesezeichen" zu im System definierten Entwicklungsobjekten. Oft ergeben sich Schlüsselstellen, an denen des Öfteren Anpassungen notwendig sind. Das kann verschiedene Gründe haben, beispielsweise:

- Ein großer historisch gewachsener Include, an dem immer wieder Erweiterungen stattfinden.
- Eine Klasse mit komplexer Logik, die sich als fehleranfällig herausstellt.
- Objekte, an denen regelmäßig gearbeitet wird.

Bookmarks können durch Rechtsklick auf die Liste neben dem Source-Code erstellt werden:

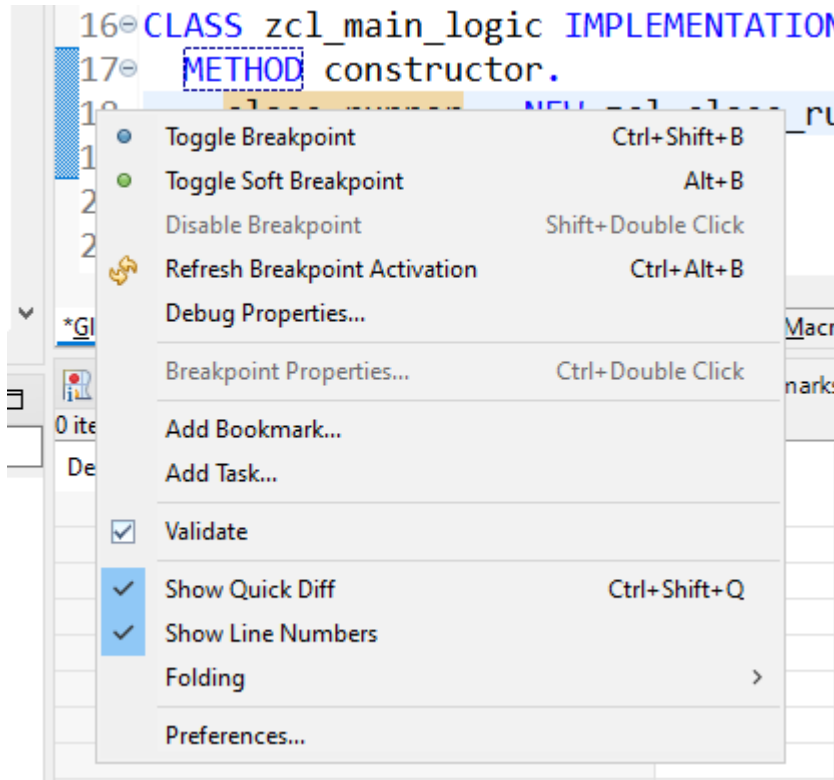


Abbildung 52 Menü zur Erstellung des Bookmarks

Als Name wird per Standard das Coding der selektierten Zeile angegeben. Es bietet sich an, einen sprechenden und fachlich sinnvollen Namen zu vergeben, über den das Bookmark dann auch einfach wiedergefunden wird.

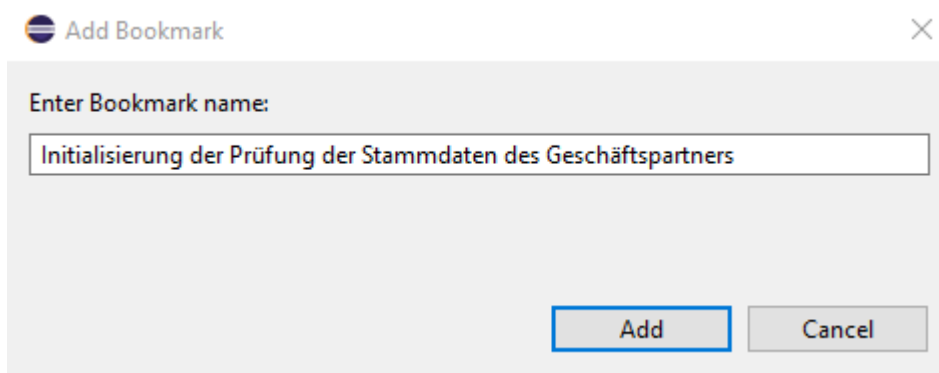


Abbildung 53 Eingabe eines Namens (Bookmark)


Innerhalb des Source-Code-Editors erscheint nun eine kleine blaue Fahne neben der selektierten Zeilennummer:

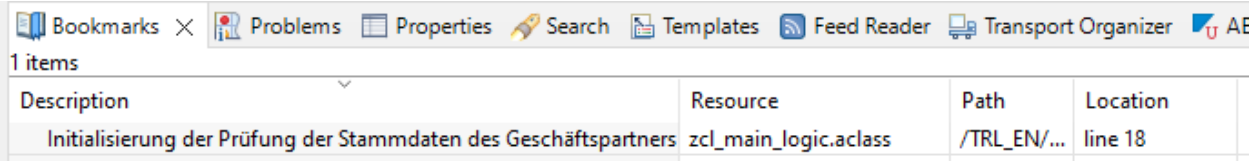
```

16 CLASS zcl_main_logic IMPLEMENTATION.
17 METHOD constructor.
18     class_runner = NEW zcl_class_run_demo( ).
19 ENDMETHOD.
20
21 ENDCLASS.

```

Abbildung 54 Darstellung eines Bookmarks im Quellcode

Das Bookmark ist dann in der Liste verfügbar und kann durch Doppelklick aufgerufen werden. Über  lassen sich zudem die Ansicht anpassen und Filterungen durchführen. Über das Kontextmenü ist ein Löschen des Bookmarks sowie das Editieren der Beschreibung möglich.



Description	Resource	Path	Location
Initialisierung der Prüfung der Stammdaten des Geschäftspartners	zcl_main_logic.aclass	/TRL_EN/...	line 18

Abbildung 55 Bookmarks View

3.2.5.7 Teilen von ADT-Links

Im Entwickler-Alltag kommt es oft vor, dass gemeinsam über Code gesprochen werden muss (z. B. bei Reviews) oder ein Problem in einem Stück Coding gefunden wird, welches in der Verantwortung einer anderen entwickelnden Person liegt (kein Shared-Code-Ownership). Oft heißt es dann “Kannst du mal bitte in der Klasse XYZ Methode ABC Zeile 1203 schauen... Ich glaube, da ist ein Bug?”. Der andere Entwickler muss aufwändig durch die IDE navigieren, bis er die erwähnte Stelle findet.

ADT bietet die Möglichkeit, einen Link zu versenden, der den Empfänger direkt an die passende Code-Stelle führt, wenn er auf diesen klickt. Hierzu muss ein Bereich im Source-Code markiert und dann im Kontextmenü “Share Link” ausgewählt werden.

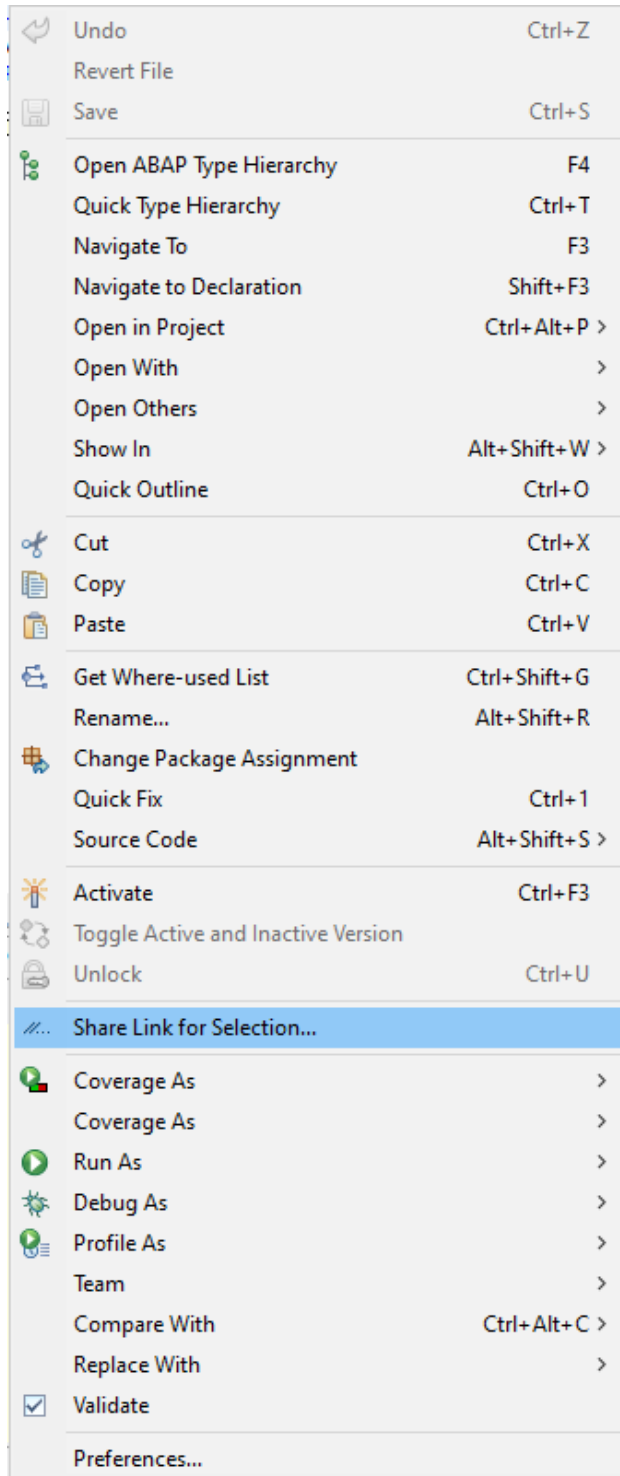


Abbildung 56 Teilen des Quellcodes als Link (Kontextmenü)

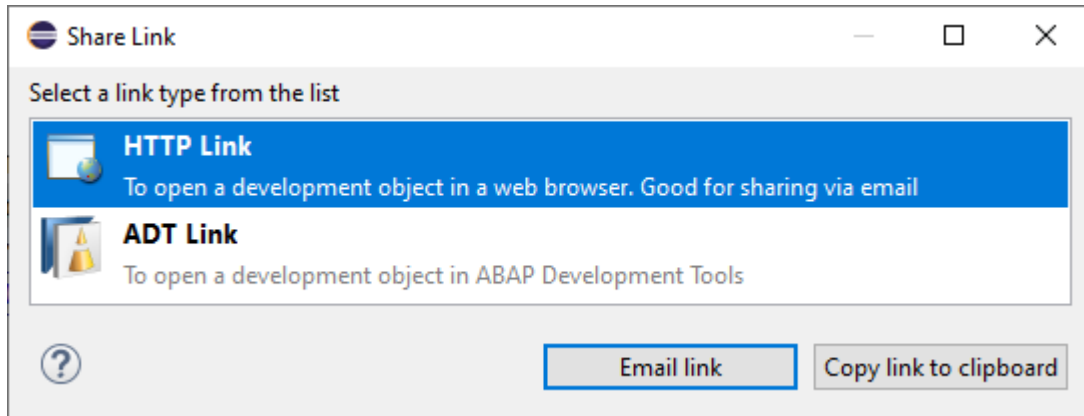


Abbildung 57 Dialog zum Teilen des Links

Der entsprechend generierte Link kann dann an den Kollegen per Mail versendet oder in die Zwischenablage kopiert und beispielsweise über ein Chat-Programm versendet werden. Man hat die Auswahl zwischen HTTP-Link und ADT-Link. HTTP-Links werden direkt im Browser geöffnet, ADT-Links verzweigen in die ADT (Eclipse).

Aufbau eines ADT-Links (URI):

```
adt://<System>/sap/bc/adt/oo/classes/<Klasse>/source/<Methode>#start=18,0
```

Mehr Details sind im [User-Guide](#) zu finden.

3.2.5.8 ABAP Type Hierarchy

Die View [Type Hierarchy](#) dient dazu, die Vererbungshierarchie von Klassen und Interfaces darzustellen. Um die View zu nutzen, braucht man nur den Cursor auf die Klasse oder das Interface zu setzen und den Shortcut **F4** zu drücken. Alternativ kann man über das Kontextmenü die ABAP Type Hierarchy öffnen.

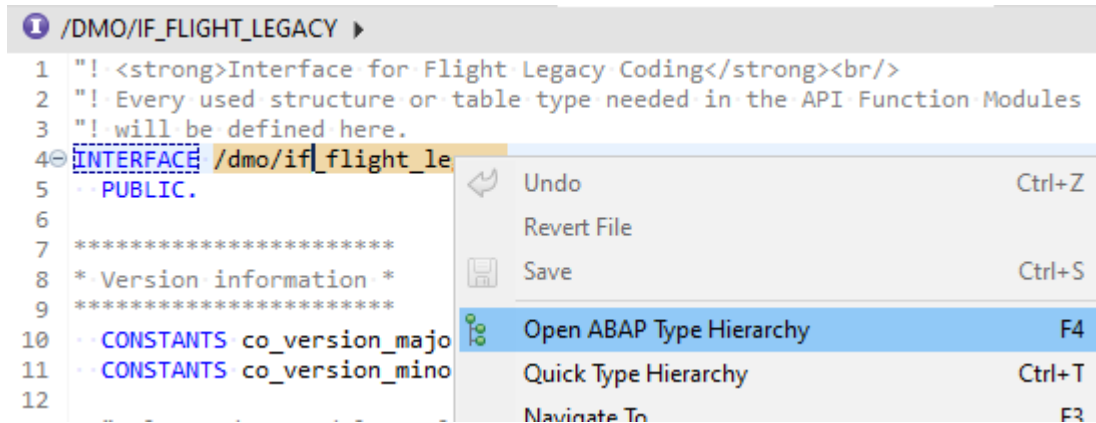


Abbildung 58 Öffnen der ABAP Type Hierarchie

Die View zeigt die Hierarchie in einer Baumstruktur an.

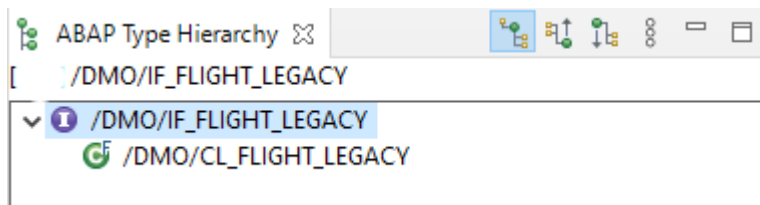


Abbildung 59 Anzeige der Type Hierarchie im View

Per Doppelklick kann man das markierte Objekt im ABAP Editor öffnen. Über **STRG+T** lässt sich auch inline im Code die [Quick Type Hierarchy](#) öffnen.

3.2.5.9 Transport Organizer

Der **Transport Organizer** zeigt eine Übersicht der geöffneten ABAP-Projekte. Unter den Systemen finden Sie die aktuellen Transporte im System. In der Standardeinstellung sehen Sie alle eigenen Transporte. Über einen Rechtsklick auf ein System und den Punkt "Configure Tree ..." können diese Einstellungen überschrieben und auch Transporte von anderen Entwicklern eingesehen werden.

Transport Request	Owner	Type	Description
Workbench			
Local Change Requests			No Target # Release
Modifiable			
K900073			Local RAP
K900074		Development/Correction	Local RAP
YBS_		Package	Test für RAP Modell
YBS_		Package	Einfache Demo
YBS_		Table	Simple data
YBS_		Technical Attributes of a Table	
YBS_		Data Definition Language Source	Einfache View
YBS_		Data Definition Language Source	Simple View C
YBS_		Behavior Definition	Einfache View
ZBP_		Class (ABAP Objects)	Behavior Implemen

Abbildung 60 Transport Organizer View

Es stehen alle Funktionen des Transport Organizer (SE09/SE10) aus der SAP GUI zur Verfügung:

- Doppelklick - Details zu Auftrag/Aufgabe in eigenem View anzeigen
- Rechtsklick - Verschiedene Funktionen wie zum Beispiel: Benutzer ändern, Konsistenzprüfung, Freigeben

3.2.5.10 Feed Reader

Der **Feed Reader** kann im Zusammenhang mit ADT genutzt werden, um bestimmte Benachrichtigungen vom SAP-System zu erhalten. Standardmäßig werden bei einem ABAP-Projekt folgende Benachrichtigungen konsumiert:

- Laufzeitfehler (Dump) verursacht durch den eigenen User
- Laufzeitfehler für Objekte, für die der eigene User verantwortlich ist
- Systemmeldungen

Project / Feed Query / Feed Entry	Date	Time	Author	Total	Unread
Runtime Errors caused by me	04.10.2022	16:02:30		2	0
The current application has intentionally triggered a termination with a short dump.	04.10.2022	16:01:23		1	0
Runtime Errors for objects I am responsible for	04.10.2022	16:02:30		1	0
System Messages	04.10.2022	16:02:30		0	0
Native Feeds				0	0

The current application has intentionally triggered a termination with a short dump.
 Runtime Errors caused by me () | [Show in Runtime Error Viewer](#)

Contents
[Header Information](#)
[What happened?](#)
[Error analysis](#)
[Information on where terminated](#)
[Source Code Extract](#)
[Active Calls/Events](#)

Header Information
 Short Text: The current application has intentionally triggered a termination with a short dump.
 Runtime Error: MESSAGE_TYPE_X_TEXT

Abbildung 61 Darstellung eines Laufzeitfehlers

Innerhalb der View kann man die Liste nach Status filtern, Testfälle erneut ausführen und sich Details zu fehlerhaften Läufen anzeigen lassen. Letzteres erscheint durch Anklicken der betroffenen Testmethode.

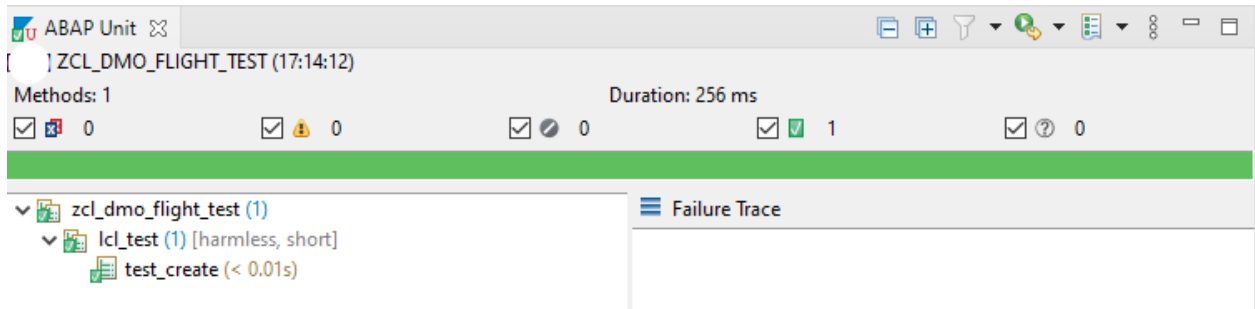


Abbildung 64 Ergebnisanzeige des ABAP Unit Test

Den Umfang der auszuführenden Testmethoden kann man über verschiedene Möglichkeiten bestimmen. Zum einen hängt dies vom Kontext ab. Hat man zum Beispiel den Fokus auf einer einzelnen Testmethode, so wird auch nur diese Methode ausgeführt. Sitzt der Fokus auf der zu testenden Klasse, dann werden alle Testklassen (und Testmethoden) dazu ausgeführt. Man kann das Ganze sogar auf ein komplettes Paket ausweiten, indem man das Paket im Projekt Explorer markiert und die Unit Tests ausführt. Außerdem kann man in der View per Kontextmenü einzelne oder alle Tests erneut ausführen - je nachdem, welche Ebene man wählt. Zum Beispiel könnte man alle Testmethoden nur einer Testklasse ausführen. Diese Möglichkeit ist besonders hilfreich, wenn ein Testfall nicht erfolgreich ist und man das Verhalten analysieren muss.

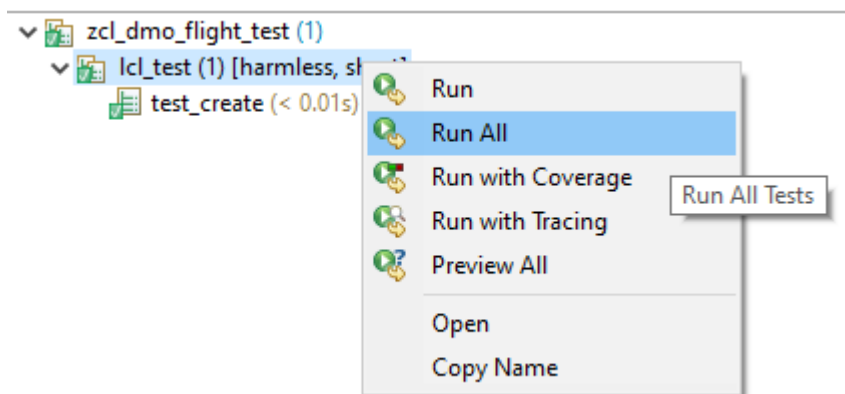


Abbildung 65 Neustart der Ausführung

Zum anderen kann man über “ABAP Unit Test With...” festlegen, welche Art von Tests durchgeführt werden sollen.

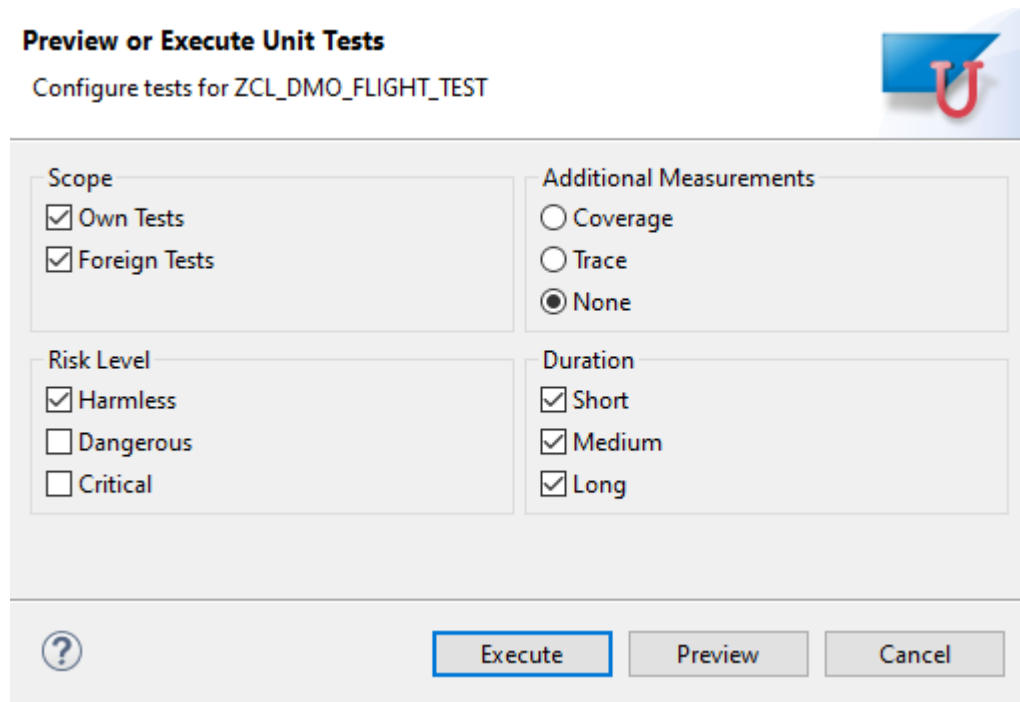


Abbildung 66 Dialog zur Einstellung der Ausführung von ABAP Unit Tests

Somit könnte man zum Beispiel nur die Testmethoden in einem Paket ausführen, die das Risk-Level “Dangerous” und die Duration “Medium” haben.

3.2.5.12 ABAP Coverage

Die View **ABAP Coverage** erscheint, wenn man ABAP Unit Tests mit Coverage (Testabdeckung) ausführt. Die Test-Coverage bietet einen Hinweis darauf, welcher Code nicht durch automatisierte Tests abgedeckt ist. Die dortige Testabdeckung kann eine bewusste Entscheidung sein, da eine Testabdeckung von hundert Prozent auf Dauer sehr viel Aufwand in der Entwicklung macht. Coverage kann auch Hinweise darauf liefern, wo mehr Testabdeckung nötig sein könnte. Eine pauschale Empfehlung für eine Testabdeckung kann nicht gegeben werden und ist gegebenenfalls auch abhängig von der Kritikalität der Anwendung.

Diese Art der Ausführung kann per Shortcut **STRG+Shift+F11** oder über das Kontextmenü mit dem Menüpunkt “Coverage As” gestartet werden.

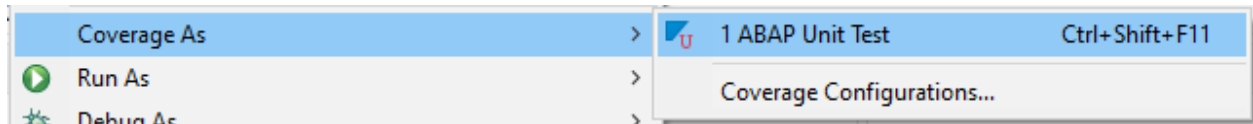


Abbildung 67 Durchführung der Abdeckungsmessung

Die View stellt den durchlaufenen Code in einer Baumstruktur dar und gibt Auskunft darüber, wie viele Statements absolut und relativ durch die ausgeführten Testmethoden ausgeführt wurden. Zusätzlich wird im ABAP Editor farblich markiert, welche Statements genau ausgeführt wurden (mit grün) und welche nicht (mit rot).

```

22 CLASS zcl_dmo_flight_test IMPLEMENTATION.
23
24 METHOD create_travel.
25     /dmo/cl_flight_legacy=>get_instance( )->create_travel(
26         EXPORTING is_travel = VALUE #( agency_id = '070001' customer_id = '000001'
27         begin_date = '20220101' end_date = '20220105' )
28         IMPORTING es_travel = DATA(ls_travel)
29     ).
30     travel_id = ls_travel-travel_id.
31 ENDMETHOD.
32
33 METHOD remove_travel.
34     /dmo/cl_flight_legacy=>get_instance( )->delete_travel( travel_id ).
35 ENDMETHOD.
36
37 ENDCLASS.

```


Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

ABAP Unit | ABAP Coverage

ZCL_DMO_FLIGHT_TEST (19.09.2022, 17:19:23)

Element	Statement Coverage	Covered Statements
<ul style="list-style-type: none"> ZCL_DMO_FLIGHT_TEST=====CP 	60.00%	3
<ul style="list-style-type: none"> ZCL_DMO_FLIGHT_TEST <ul style="list-style-type: none"> CREATE_TRAVEL REMOVE_TRAVEL 	60.00%	3
	100.00%	3
	0.00%	0

Abbildung 68 Farbliche Hervorhebung von Quellcode nach Unit Test

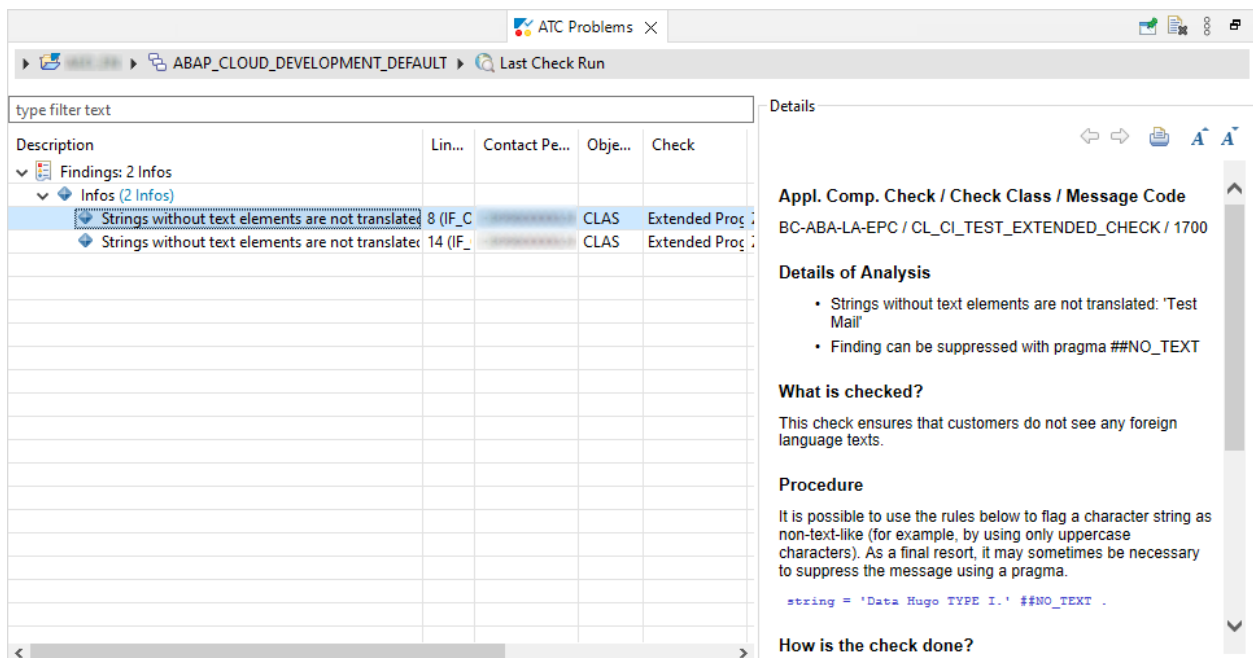
Ist die farbliche Ansicht im Source-Code nicht sichtbar, so kann diese über das Icon  aktiviert werden.

3.2.5.13 ATC und Exemption

Das **ABAP Test Cockpit** kann in ADT ebenso wie in der SAP GUI ausgeführt werden. Dabei haben Sie verschiedene Möglichkeiten, wie Sie die Prüfung starten können:

- Über die Tastenkombination **STRG+SHIFT+F2**
- Rechtsklick im Projekt Explorer unter dem Punkt “Run As”
- Im Menüband oben, unter dem Button zum Starten des Objekts

Nach Ausführung der Prüfungen erhalten Sie die View für die “ATC Problems”, also die Rückmeldungen über die gefundenen Meldungen durch die eingestellten Prüfungen.



The screenshot shows the Eclipse IDE interface for the ATC Problems view. The left pane displays a table of findings, and the right pane shows the details for a selected finding.

Description	Lin...	Contact Pe...	Obj...	Check
Findings: 2 Infos				
Infos (2 Infos)				
Strings without text elements are not translated	8 (IF_C		CLAS	Extended Proc
Strings without text elements are not translated	14 (IF_		CLAS	Extended Proc

Details

Appl. Comp. Check / Check Class / Message Code
BC-ABA-LA-EPC / CL_CI_TEST_EXTENDED_CHECK / 1700

Details of Analysis

- Strings without text elements are not translated: 'Test Mail'
- Finding can be suppressed with pragma ##NO_TEXT

What is checked?

This check ensures that customers do not see any foreign language texts.

Procedure

It is possible to use the rules below to flag a character string as non-text-like (for example, by using only uppercase characters). As a final resort, it may sometimes be necessary to suppress the message using a pragma.

```
string = 'Data Hugo TYPE I.' ##NO_TEXT .
```

How is the check done?

Abbildung 69 Anzeige der Ergebnisse des ATC-Laufs

Auf der linken Seite werden die Meldungen sortiert nach der Schwere des Fehlers angezeigt. Rechts erhalten Sie Informationen zum gewählten Eintrag. Hier wird für Sie noch einmal erklärt, was geprüft wurde und wie eine Korrektur aussehen kann. In der Button-Leiste im oberen Bereich der View kann das Ergebnis auch wieder gelöscht werden. Auf diese Weise verschwinden die Markierungen im Quellcode.

Mit einem Rechtsklick auf die Meldung kann über den Menü-Eintrag “Request Exemption” auch eine Ausnahme beantragt werden.

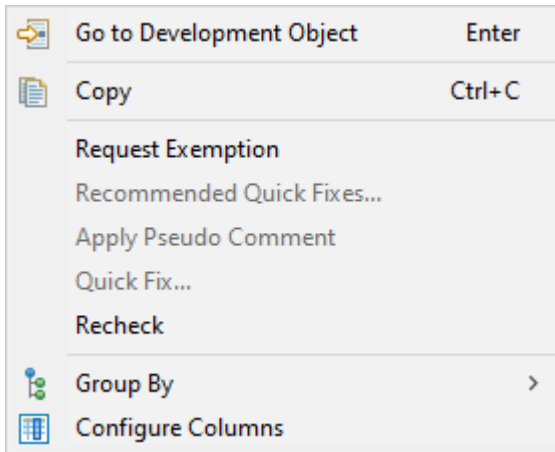


Abbildung 70 Beantragung von Ausnahmen über den ATC View

Das Formular entspricht von den Informationen her der SAP GUI und leitet Sie durch den Freigabeprozess. Am Ende kann die Anfrage wie gewohnt über das ATC Cockpit bearbeitet werden.

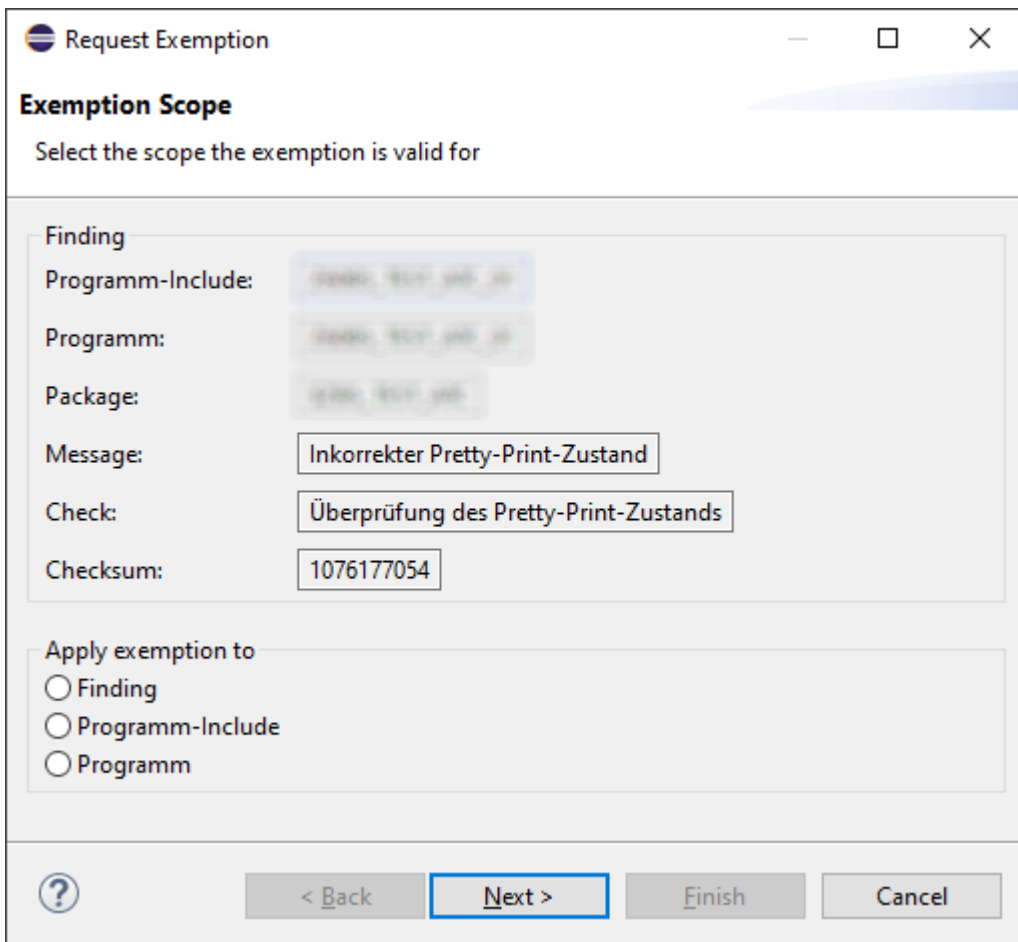


Abbildung 71 Dialog zur Klassifizierung der Ausnahme

3.2.5.14 ABAP Language Help

In jedem Quelltexteditor, beispielsweise für ABAP, CDS oder BDL, kann für das Schlüsselwort, auf dem der Cursor ist, mit der Taste F1 direkt die jeweilige Sprachhilfe (nicht nur ABAP!) aufgerufen werden. Alternativ kann man diese auch über das Kontextmenü per Rechtsklick auf die entsprechende Anweisung bekommen:

Source Code → *Show ABAP Language Help*

Damit kann man jederzeit Unterstützung bekommen, falls man sich der exakten Syntax einer Anweisung nicht sicher ist.

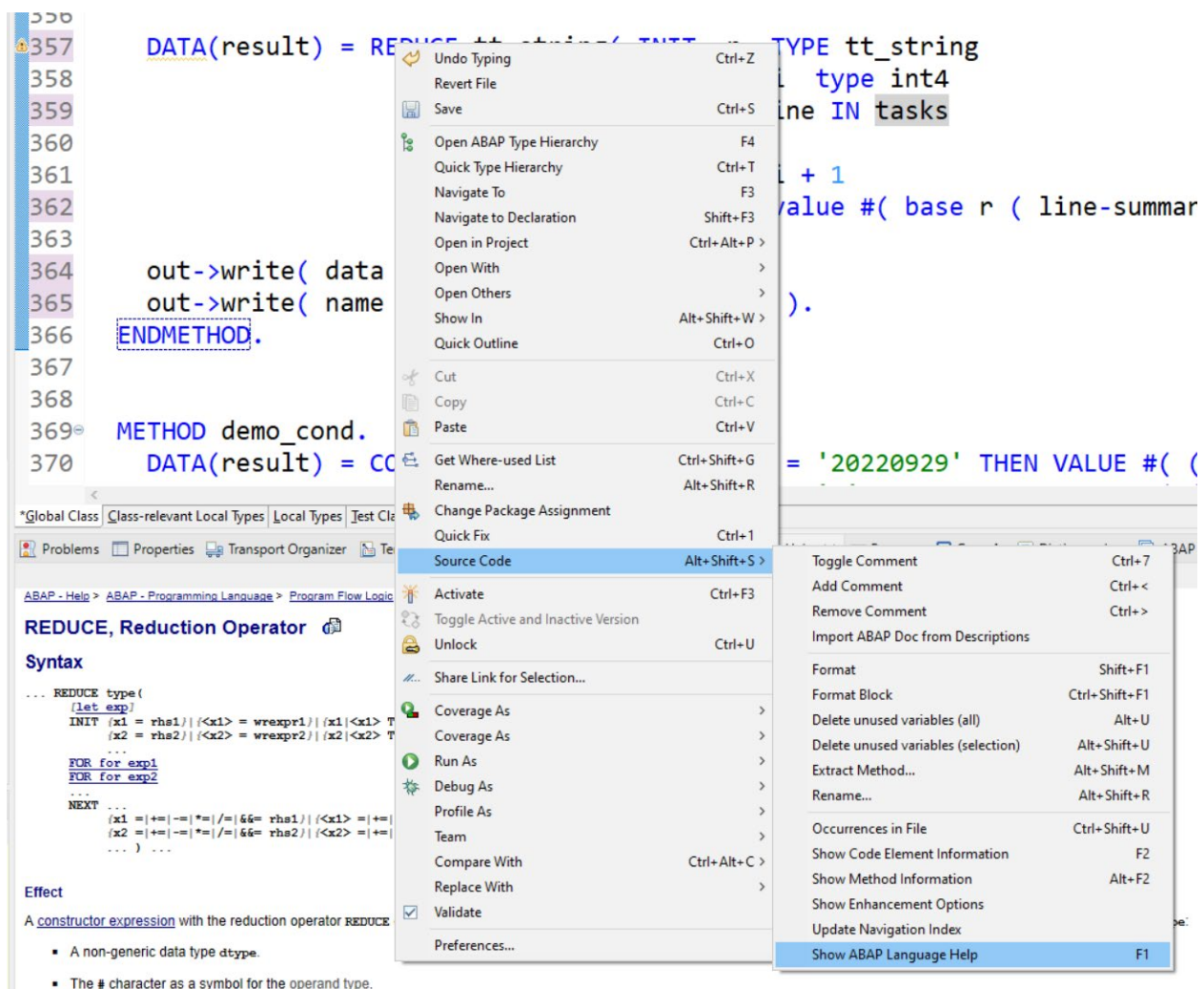


Abbildung 72 Aufruf der ABAP-Sprachhilfe über das Kontextmenü

3.2.5.15 Der ABAP Language Help View

Die entsprechende Dokumentation wird im ABAP Language Help View als HTML-Dokument angezeigt. Damit ist eine Vorwärtsnavigation über Hyperlinks möglich.

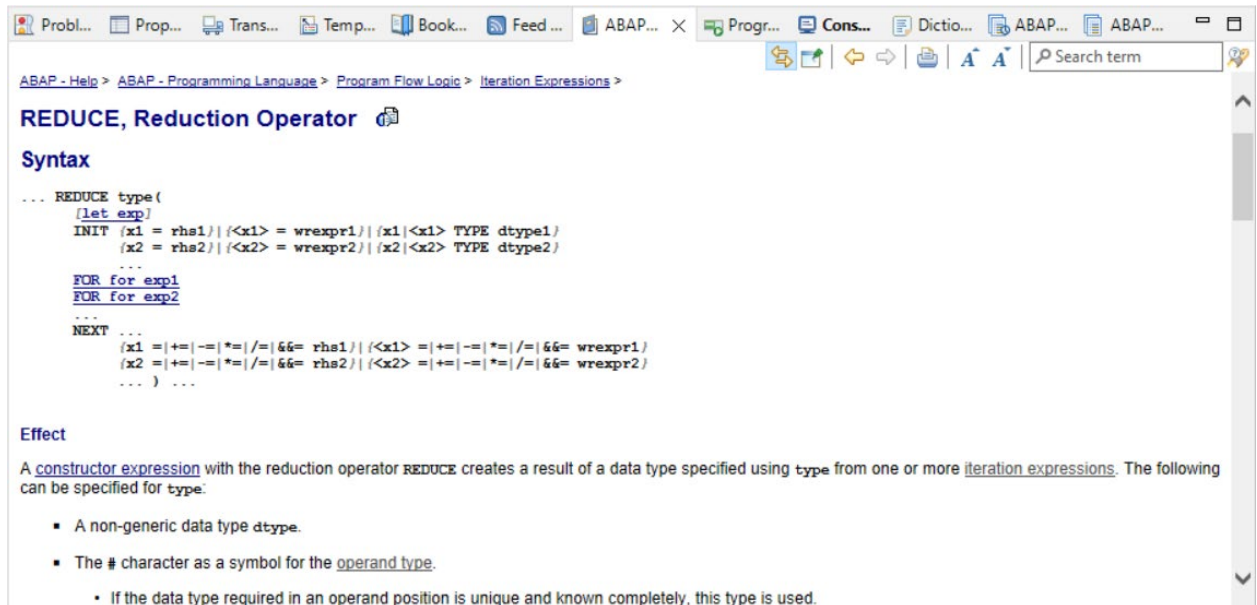


Abbildung 73 ABAP Language Help View

Wie in vielen Views in Eclipse sind hier einige nützliche Standard-Buttons vorhanden.



Abbildung 74 Button-Leiste des Views

- Der gelbe Doppelpfeil koppelt die View mit dem aktiven Editor. Damit zeigt die View stets die passende Hilfe für eine Anweisung an, auf der der Cursor momentan steht.
- Der grüne Pin hält den Inhalt des Views fest. Wenn ein weiteres Mal mit F1 eine Hilfe angefordert wird, öffnet sich eine neue View für die ABAP Language Help.
- Die gelben Pfeile nach rechts und links dienen der Navigation (analog eines Web-Browsers).
- Mit Hilfe des Drucker-Symbols kann man mit entsprechender Hardware eine papierhafte Kopie der ABAP-Hilfe anfertigen.
- Die beiden Symbole A mit den Pfeilen nach oben und unten sind für die Änderung der Schriftgröße zuständig.
- Mit dem Suchfeld kann die komplette ABAP-Hilfe, einschließlich der anderen Sprachen wie CDS oder BDL, durchsucht werden.

3.2.5.16 Application Help

Neben der ABAP Keyword Documentation (bzw. ABAP Language Help) stellt SAP für jedes Entwicklungsszenario sogenannte Eclipse Help Plug-ins zur Verfügung. Klicken Sie hierfür

Help → Help Contents

in der Menüleiste, um den Hilfe-Browser zu öffnen.

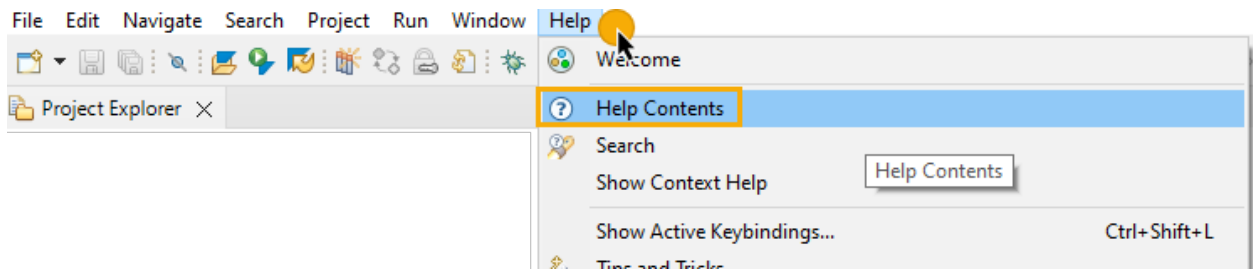


Abbildung 75 Navigation zum Help Content

Sie erkennen die Help Plug-ins von SAP an dem gelben Buch-Icon. Momentan gibt es die folgenden Help Plug-ins:

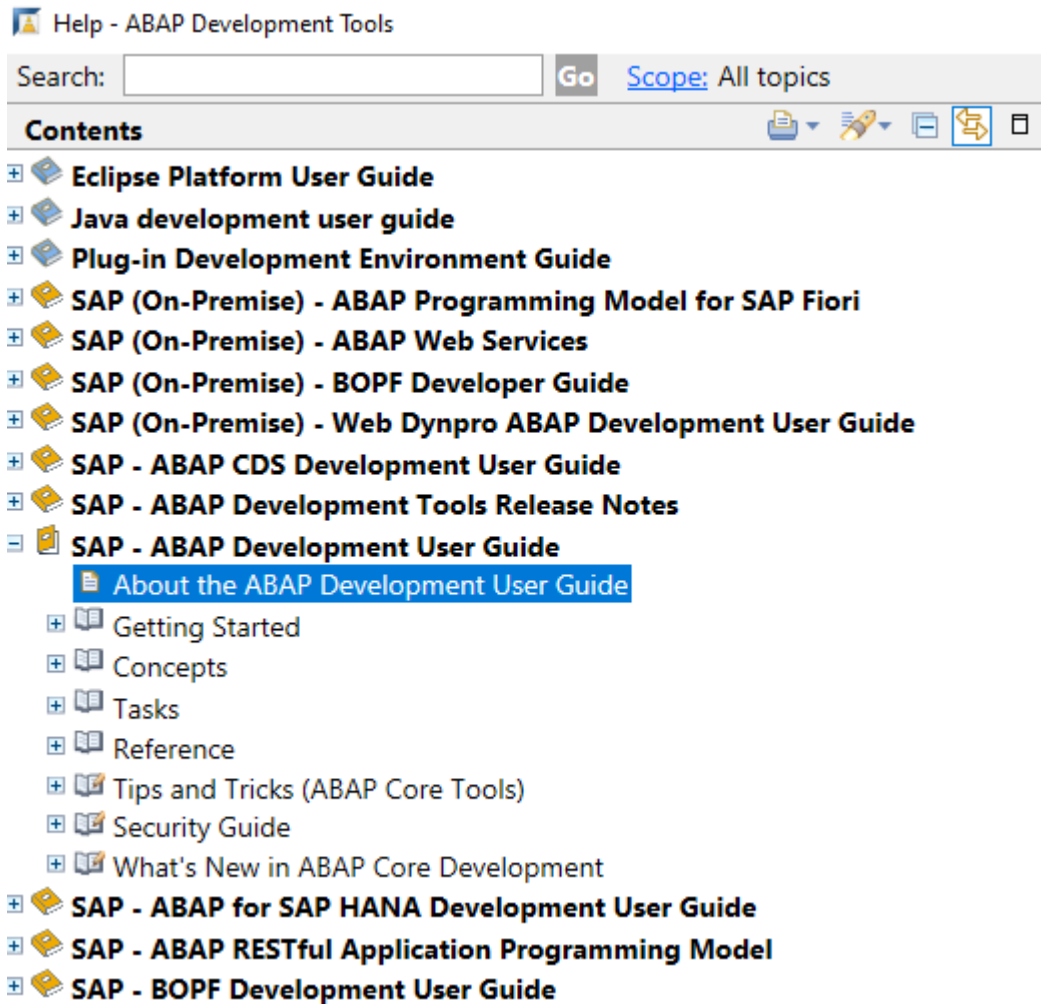


Abbildung 76 Übersicht der verfügbaren Hilfen und Dokumentationen

Mit Hilfe der Suche (Search) können Sie nach Stichworten suchen. Mittels Scope können Sie die Suche auf ein oder mehrere Help Plug-ins eingrenzen.

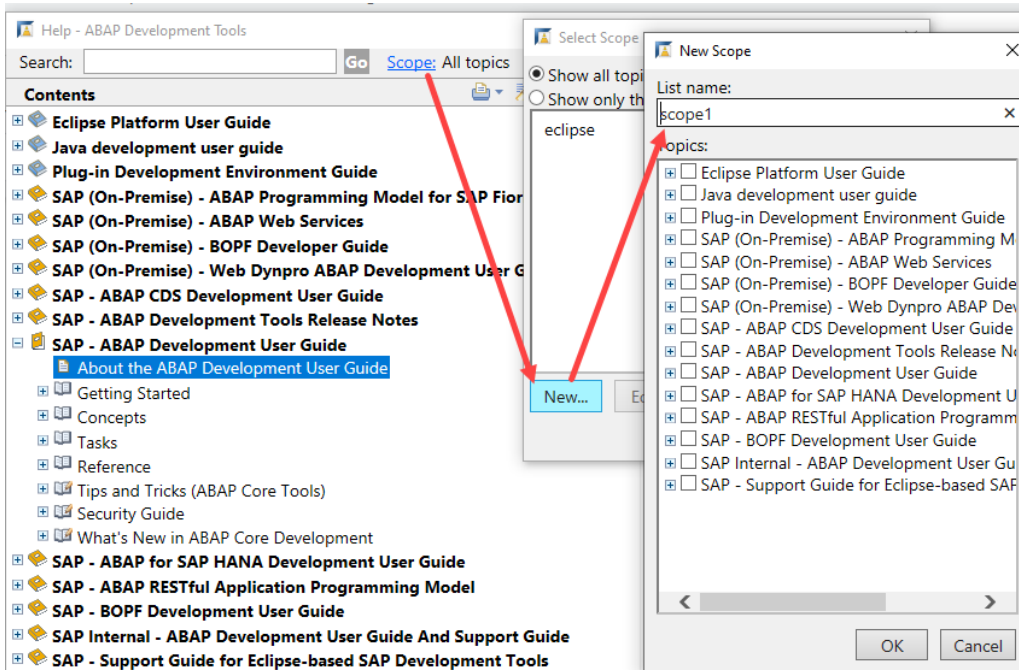


Abbildung 77 Suche in der Hilfe

In Wizards, die die ?-Ikone anbieten, können sie die kontextsensitive Hilfe öffnen. Diese führt Sie direkt zu dem jeweiligen Hilfeinhalt, den es für den Wizard gibt.

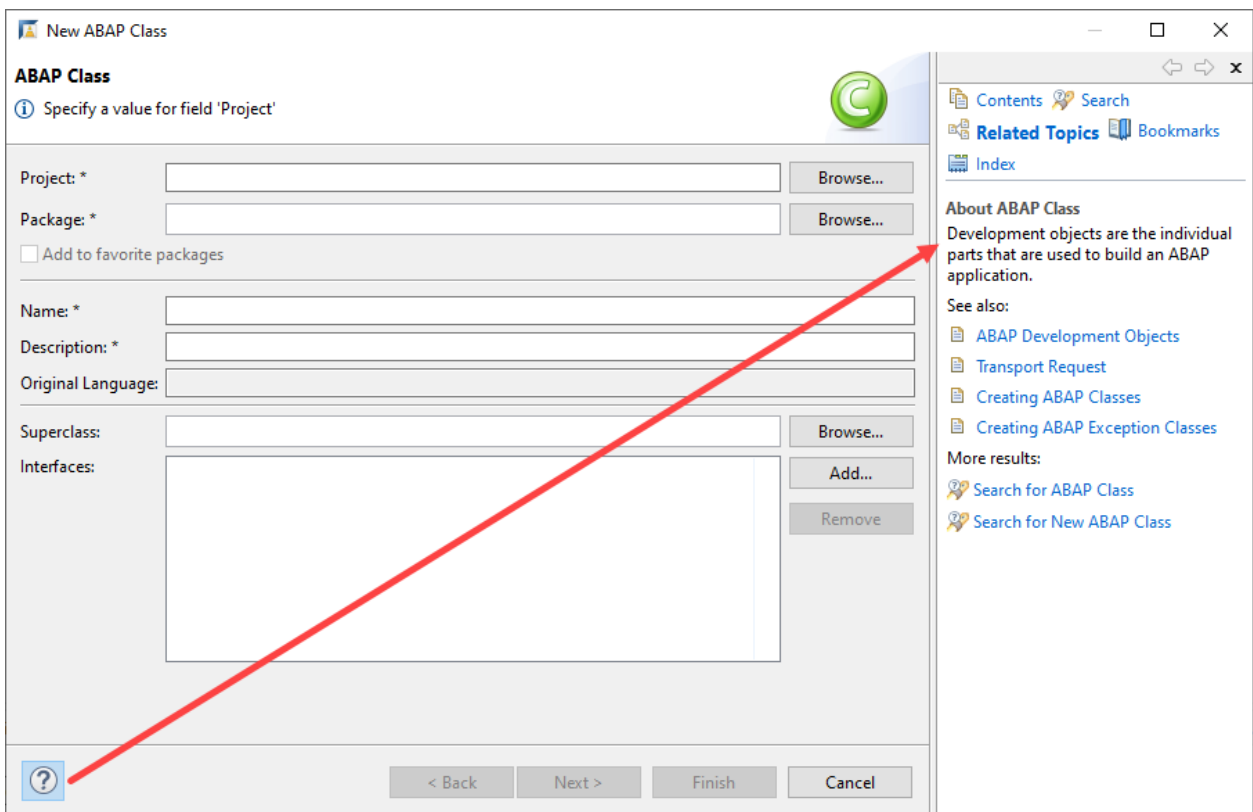


Abbildung 78 Weiterführende Hilfen und Dokumentationen

Sie können so genannte Active Links in einer Hilfeseite klicken, um aus der Hilfe heraus ein Eclipse-UI zu öffnen. Sie erkennen Active Links an der grünen Ikone mit dem weißen Pfeil.


[SAP - ABAP Development User Guide](#) > [Tasks](#) > [Using Troubleshooting Tools](#) > [Debugging ABAP Code](#)

Setting ABAP Debugging Preferences

Context

You can change how the ABAP debugger behaves with the *General Settings*.

Procedure

1. Open the  *debugger preference* page. (Choose [Window](#) > [Preferences](#) > [ABAP Development](#) > [Debug](#).)
2. Change debugger settings as required. The settings apply to all of your ABAP projects.

Enable debugging of system programs	Mark this checkbox to make ABAP system programs debuggable. If this option is not activated, then the debugger will not debug system programs without displaying the code. The d
-------------------------------------	--

Abbildung 79 Darstellung der Navigationspfade

Der identische Hilfeinhalt, wie er Ihnen im ADT Client zur Verfügung steht, ist auch online auf dem [SAP Help Portal](#) verfügbar.

3.2.6 Refactoring von Code mit ADT

Wie bereits im [Kapitel 2 - Motivation für ADT](#) kurz erwähnt, ermöglichen die zahlreichen Funktionen in ADT auch das Refactoring von Code. Doch was ist [Refactoring](#), welche Vorteile bietet es und welche Funktionen werden hierfür von ADT bereitgestellt? Diese Fragen sollen im folgenden Abschnitt detailliert beantwortet werden.

Refactoring bezeichnet die Veränderung von Source-Code, um dessen Struktur und Lesbarkeit zu verbessern, ohne dabei die Funktionen zu ändern. Dies bedeutet, es kommen keine neuen Funktionalitäten hinzu, es fallen keine Funktionalitäten weg und es bleibt die Korrektheit der Lösung erhalten, d. h. sie liefert weiterhin das korrekte Ergebnis. Es werden keine neuen Bugs eingeführt.

Die Erhaltung der Korrektheit ist hierbei sicherlich der wichtigste und auch in der SAP-Welt am schwersten zu erreichende Aspekt. Korrektheit kann am einfachsten über automatisierte Tests bewiesen werden. Leider sind diese im SAP-Kosmos kaum verbreitet, wurden in der Vergangenheit nur wenig unterstützt und sind oft schwierig

zu implementieren, da die Strukturen von historischem ABAP-Code dafür schwer geeignet sind. Somit stellt allerdings das Herstellen einer automatisierten Testbarkeit ein primäres Ziel von Refactorings dar.

Darüber hinaus gibt es noch weitere Gründe für ein Refactoring:

- Erhöhung der Verständlichkeit des Codes (“Clean Code”)
- Verbesserung der Anpassbarkeit des Codes für Erweiterungen
- Abbau von technischen Schulden
- Aktualisierungen von veralteten Befehlen/Modulen

Refactoring ist ein integraler Bestandteil der Software-Entwicklung und sollte beim täglichen Entwickeln durchgeführt werden, um einen gewissen Qualitätsstandard zu halten. Es ist davon abzuraten, spezielle “Refactoring Sprints” o. ä. durchzuführen, da diese oft von Geldgebern skeptisch betrachtet oder gar nicht genehmigt werden. Die Autoren empfehlen daher die Boy-Scout-Rule zu beachten: “Always leave the code better than you found it.”

In der Vergangenheit war dies mit der SE80 meist mit hohem Aufwand verbunden. Durch die mangelnde Unterstützung der IDE mussten die Refactorings zu großen Teilen händisch durchgeführt werden. Dieser hohe manuelle Aufwand und deren Fehleranfälligkeit führten zu einer geringen Akzeptanz dieses Prozesses und von Clean Code im Allgemeinen.

Mit den ADT hat sich diese Situation nun verändert. Existieren keine automatisierten Tests als doppelter Boden, so ist es immer noch möglich, sogenannte Save Refactorings durchzuführen, die wir an dieser Stelle beschreiben möchten. Ein Save Refactoring kennzeichnet sich dadurch, dass es Tool-gestützt, d. h. durch Funktionen der IDE oder auch mit zusätzlichen Plug-ins, durchgeführt wird. Dadurch entfällt das Risiko, durch manuelle Änderungen neue Fehler in den Code einzubauen. Durch die Automatisierung können Refactorings leicht durchgeführt werden und damit zum Bestandteil der täglichen Arbeit werden.

Die ADT bieten über die Quick Assists (**STRG+1**) folgende Refactorings an:

1. Rename Identifier – Umbenennung innerhalb eines Codeblocks oder global
2. Extract Method – Extrahieren einer Methode aus dem Source-Code oder aus einem Ausdruck
3. Extract Constants – Textliterals in Konstanten umwandeln
4. Extract Variables – Variablen extrahieren und konvertieren
5. Move Member – Attribute von Klassen verändern und bewegen
6. Exception Handling – Automatisiertes Anlegen/Transformieren von Exception-Blöcken

Insbesondere die Rename- und die Extract-Method-Funktion unterstützen den Entwickler dabei, den Code sauber zu halten und Code-Redundanzen zu vermeiden bzw. zu reduzieren.

Da z. B. die Rename-Funktion Identifier nicht nur innerhalb der Einheit, sondern über alle Verwender hinweg behandelt, ist es nun ein Leichtes, einen unpassend gewählten Namen in einen besser zum Gesamtkontext passenden Namen zu ändern. Dabei wird nicht das Risiko eingegangen, dass Verwender vergessen und so Fehler im Code eingebaut werden.

Die Extract-Funktion analysiert den markierten Code, bietet Hilfestellung bei der Parametervergabe und ersetzt die Stelle des Codes mit dem Aufruf der neu erstellten Methode. Wenn die zu extrahierende Methode mit einem Kommentar versehen ist, wird dieser als Vorschlag für die Benennung der Methode herangezogen.

Weiterhin empfehlen die Autoren die Verwendung des Plug-ins “ABAP Quick Fix” (<https://marketplace.eclipse.org/content/abap-quick-fix>) von Lukasz Pegiel, welches im [Kapitel 7 - Plug-ins](#) beschrieben wird. Generell bieten die mittels Quick Fixes bereitgestellten Refactoring Tools in ADT sowohl bei der Erstellung als auch bei der Überarbeitung von bestehendem Code eine enorme Hilfe. Die Nutzung dieses hilfreichen Plug-ins verbessert auf diese Weise einerseits den aktuell bearbeiteten Code, hilft aber auch bei der Erstellung von neuem Code, die neueren Sprachkonstrukte selbst anzuwenden, falls man darin noch wenig geübt ist.

3.2.7 Versionsverwaltung und Vergleichen

Hinter dem Kontextmenüpunkt “[Compare with](#)” verstecken sich einige der wichtigsten Features für die tägliche Arbeit. Diese funktionieren bei allen Quelltexteditoren in den ADT, nicht nur bei der Entwicklung von ABAP-Code.

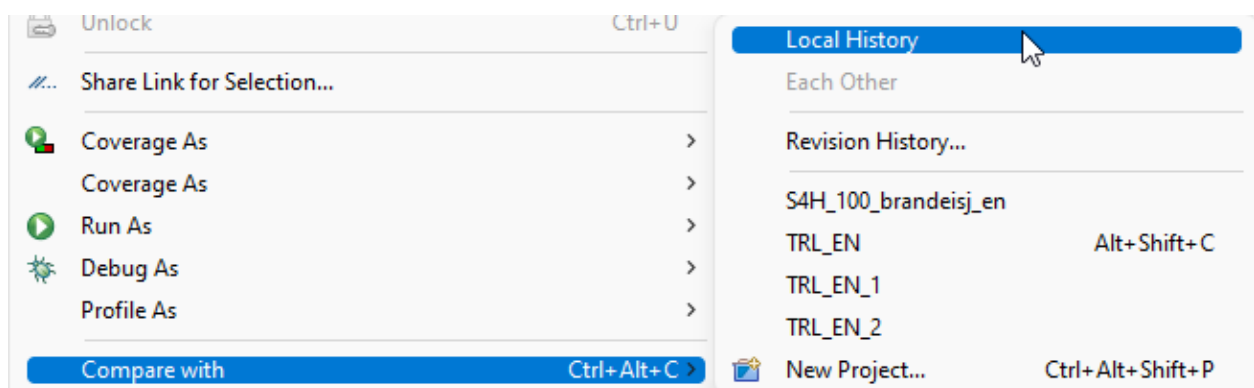


Abbildung 80 Kontextmenü zum Vergleichen von Versionen

3.2.7.1 Local History – Die lokale Versionsverwaltung

Die Local History ermöglicht den Zugriff auf ältere Versionen des aktuellen Quelltextdokuments aus dem Eclipse Workspace, mit dem der Benutzer gerade arbeitet. Mit jedem Speichern des Objektes wird eine Version gezogen. Das bedeutet, dass man sehr komfortabel seine eigene Arbeit im Laufe der Stunden und Tage nachverfolgen und ohne Weiteres auch wieder auf ältere Versionen zurückwechseln kann.

Da sich die lokale Versionshistorie nur auf den eigenen Eclipse Workspace bezieht, kann es passieren, dass man Änderungen auch auf einem anderen Gerät hat oder ein Kollege zuletzt Änderungen durchgeführt hat.

3.2.7.2 Revision History – Die Versionsverwaltung des ABAP

Unter dem Menüpunkt Revision History erreicht man die “normale” Quelltextverwaltung des ABAP-Servers, die auch schon in der SAP GUI zur Verfügung stand. Hier werden standardmäßig Versionen gezogen, wenn ein Transportauftrag freigegeben wird.

Die Versionen der Revision History sind entsprechend global für alle User zugänglich, unabhängig vom Workspace der Eclipse-Installation.


3.2.7.3 Anderer Projektname – Systemübergreifend vergleichen

Wenn man sich ein Projekt aus der Liste rauspickt, kann man systemübergreifend vergleichen. Das kann ein System aus der gleichen Systemlandschaft sein, z. B. das Produktivsystem oder auch ein ganz anderes System. Somit kann aus ADT heraus Code von unterschiedlichen Systemen miteinander verglichen werden, die keine RFC-Verbindung zueinander haben. Dies bietet in komplexen Systemlandschaften einen großen Vorteil gegenüber den GUI-basierten Vergleichsmöglichkeiten.

3.2.7.4 Comparison View

In der Comparison View kann man links den aktuellen Stand des Quelltextes sehen und rechts die zum Vergleich ausgewählte Version. Die Abweichungen werden hervorgehoben:

- **Grün** sind die Dinge, die in der aktuellen Version neu sind
- **Rot** sind die Dinge, die gelöscht wurden und in
- **Grau** sind die Änderungen hinterlegt

Mit den Buttons  kann der alte Zustand durch Kopieren von rechts nach links wiederhergestellt werden. Es ist aber auch möglich, dass man direkt in dieser Vergleichs-View auf der linken Seite Änderungen durchführt. Nach dem Speichern wird der Vergleich wiederholt.

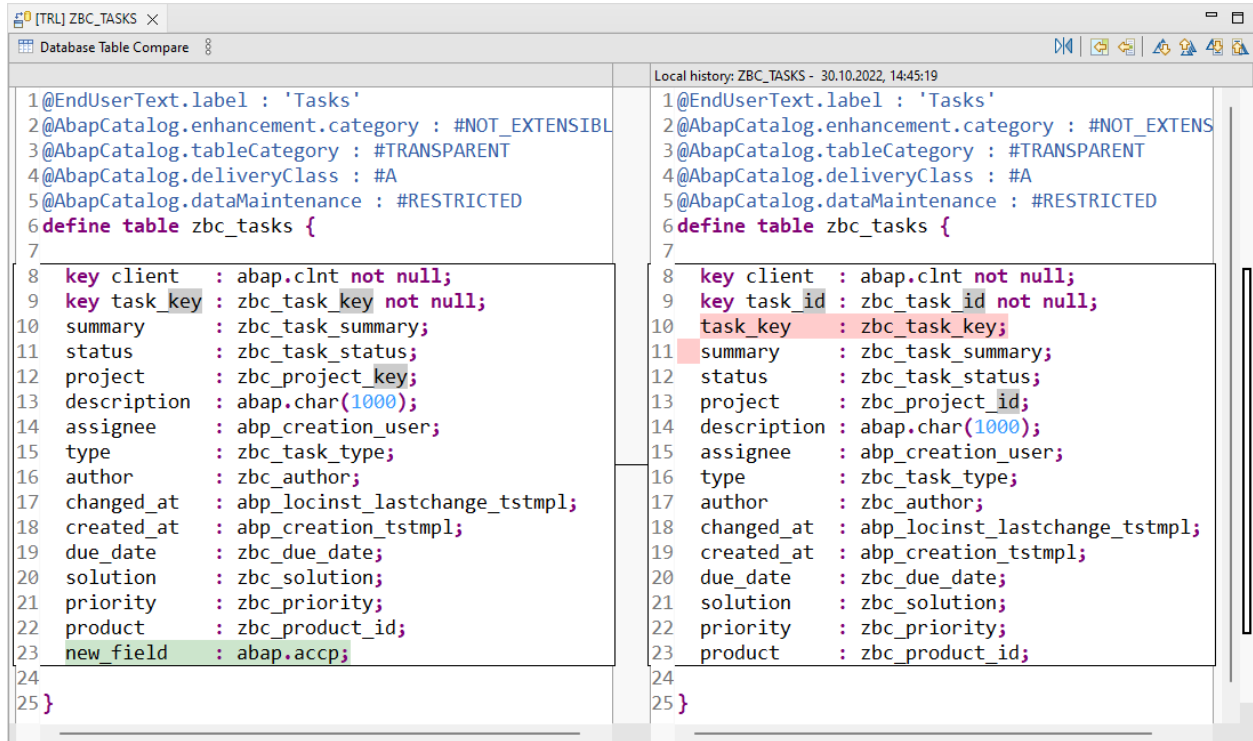


Abbildung 81 Comparison View – Vergleich von zwei Versionen

Wenn man eine alte Version vollständig übernehmen will, kann man aus dem Kontextmenü mittels *Replace With* → *Local History* direkt die passende Version auswählen.

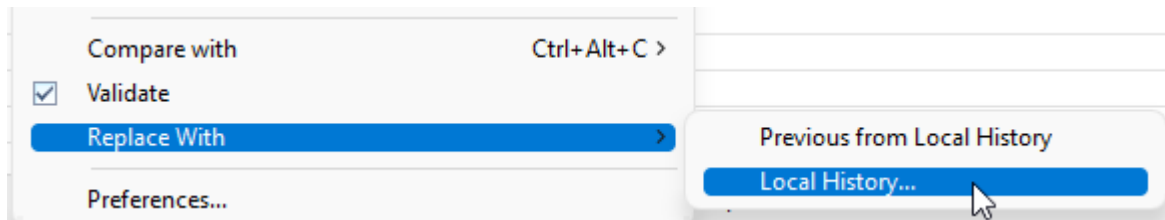


Abbildung 82 Kontextmenü zum kompletten Übernehmen einer Version aus der lokalen Versionsverwaltung

3.2.8 Dokumentation mit den ABAP Doc

3.2.8.1 Was sind ABAP Doc?

ABAP Doc ermöglicht die Code-basierte Dokumentation wie z. B. von Methoden und deren Parametern.

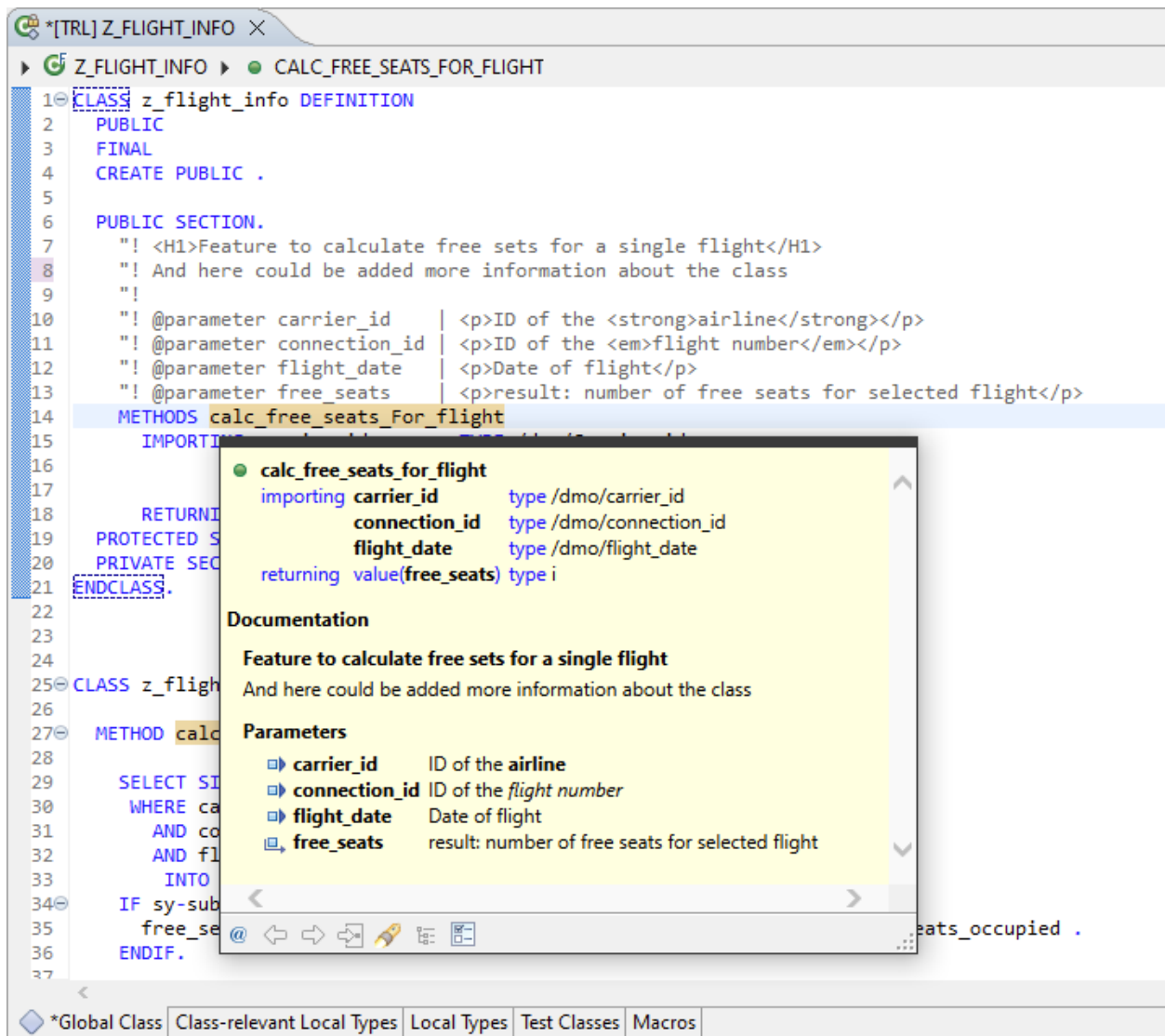


Abbildung 83 ABAP-Doc Dokumentation der Methode

ABAP Doc ist ein Feature, das nur in den ADT unterstützt wird. Anstatt des formularbasierten Editors mit der Möglichkeit der Kurzbeschreibung von Methoden, der in den ADT nicht mehr verfügbar ist, wurde mit den ABAP Docs ein deutlich mächtigerer Ersatz etabliert, der in ähnlicher Form auch in anderen Programmiersprachen verfügbar ist (z. B. Javadoc).

Im Folgenden wird für die bessere Lesbarkeit des Texts detailliert auf den Einsatz der ABAP Docs im Kontext von Klassen/Methoden eingegangen. Die ABAP Docs sind aber auch auf andere Entwicklungsartefakte wie z. B. Funktionsbausteine anwendbar (s. SAP-Hilfe).

Mit Hilfe der ABAP-Doc-Funktion können für Klassen und deren Methoden textuelle Beschreibungen erfasst werden. Des Weiteren können auch für die einzelnen Parameter und Ausnahmen Beschreibungen in ABAP Doc hinterlegt werden.

Die mit ABAP Doc zu erstellenden Hinweise werden im Bereich der Definition angelegt. Der Mehrwert entsteht aber vor allem durch den einfachen Aufruf dieser Dokumentation durch den Verwender. Dies ist sowohl an der Aufrufstelle als auch im Bereich der Implementierung von Entwicklungsartefakten mittels der Taste F2 möglich. Zusätzlich können die in ABAP Docs erstellten Texte sogar mittels HTML-Tags formatiert werden. So können die Dokumentationen mit Überschriften oder Textformatierungen angereichert und somit noch ansprechender und strukturierter dargestellt werden.

Um in ABAP Doc erstellte Beschreibungen in die SAP-GUI-angezeigten Kurztexte zu übernehmen, wird das **“Synchronized”** Tag verwendet, so dass auch bei Betrachtung mittels SE24/SE80 die Überschriften sichtbar sind.

Dies kann sinnvoll sein, falls Objekte Enhancements enthalten, die nicht direkt in ADT editiert

werden können und darum die Modifikation noch in den GUI-basierten Tools erfolgen muss.

Eine Mischung von ABAP Doc und GUI-Kurztexten empfehlen wir nicht. Die Nutzung von ABAP Doc ist das Mittel der Wahl, um den funktionalen Code von Kommentaren zu entlasten und dem Verwender hilfreiche Hinweise zu den Entwicklungsartefakten zu geben.

3.2.8.2 Nutzung der Quick Fixes zur Erstellung von ABAP Doc

Die Erstellung der ABAP Doc ist über den Aufruf der Quick Fixes einfach. Dazu markiert man die Methodendefinition, ruft die Quick Fixes auf und wählt **“Add ABAP Doc”** aus. Wurde eine Methodendefinition geändert, z. B. indem ein Parameter ergänzt wurde, und es ist eine Aktualisierung der Dokumentation erforderlich, kann die ABAP Doc aktualisiert werden, indem hier nicht die Methodensignatur, sondern der ABAP-Doc-Bereich direkt markiert wird und damit die Quick Fixes aufgerufen werden.

3.2.8.3 Weitergehende Informationen zu ABAP Doc

Weitergehende Informationen zu ABAP Doc finden sich in der offiziellen SAP-Hilfe (z. B.)

unter [ABAP Doc - ABAP-Schlüsselwortdokumentation \(sap.com\)](#) (7.50), in der Beispielsklasse CL_DEMO_ABAP_DOC und im [User-Guide](#).

3.2.9 Ausführen von Source-Code

Auch in den ADT kann geöffneter Source-Code weiterhin komfortabel ausgeführt werden. Über F8 wird eine SAP-GUI-Instanz des jeweiligen Systems initialisiert und das geöffnete Entwicklungsobjekt ausgeführt. Bei Klassen entspricht dies beispielsweise der Funktion "Testen Klasse X", bei Reports wird normal der Report ausgeführt.

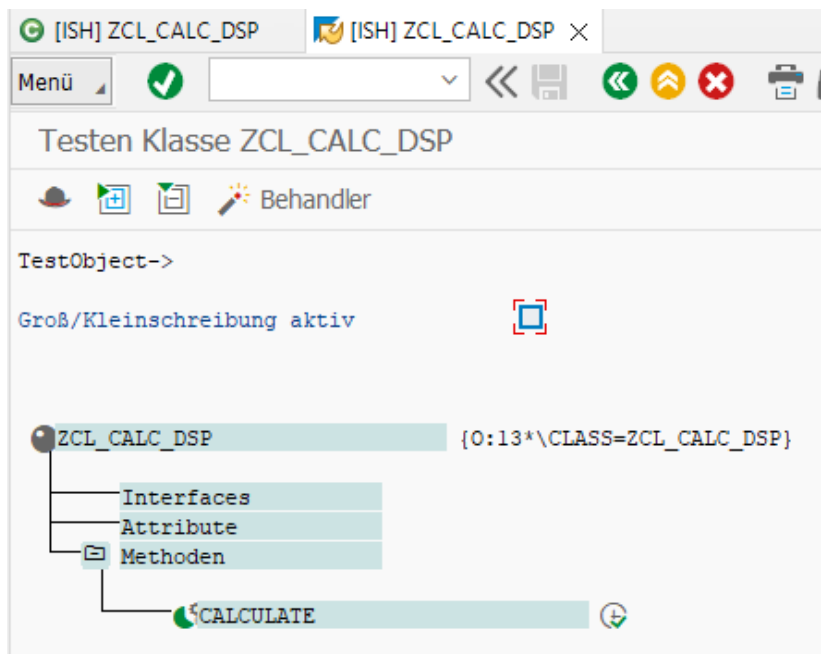


Abbildung 84 Ausführung einer Klasse in SAP GUI

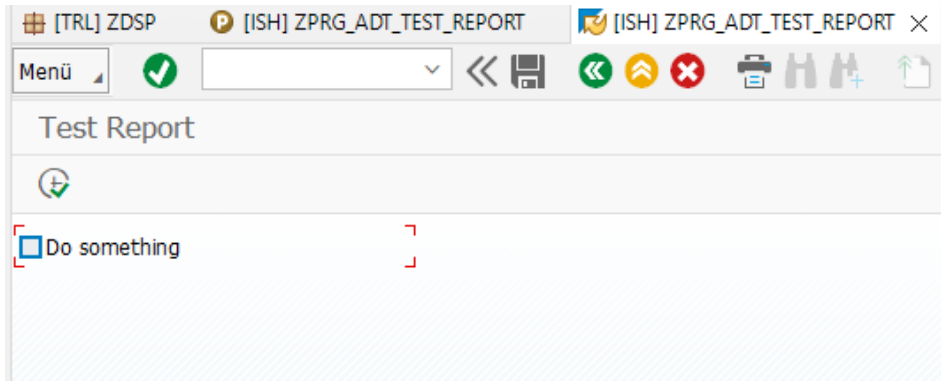


Abbildung 85 Ergebnis der Ausführung

Soll ein beliebiges Entwicklungsobjekt ausgeführt werden, so kann mit **ALT+F8** ein Objekt über den Object Finder gesucht werden. Dabei kann ein beliebiges Projekt ausgewählt werden, d. h. ein an ADT angebundenes SAP-System. Dies muss nicht das System sein, in dem gerade entwickelt wird – es kann auch ein Qualitätssicherungssystem sein. Wichtig ist hierbei, dass die ADT-Funktionen für dieses System freigeschaltet sind bzw. die entsprechenden Berechtigungen existieren.

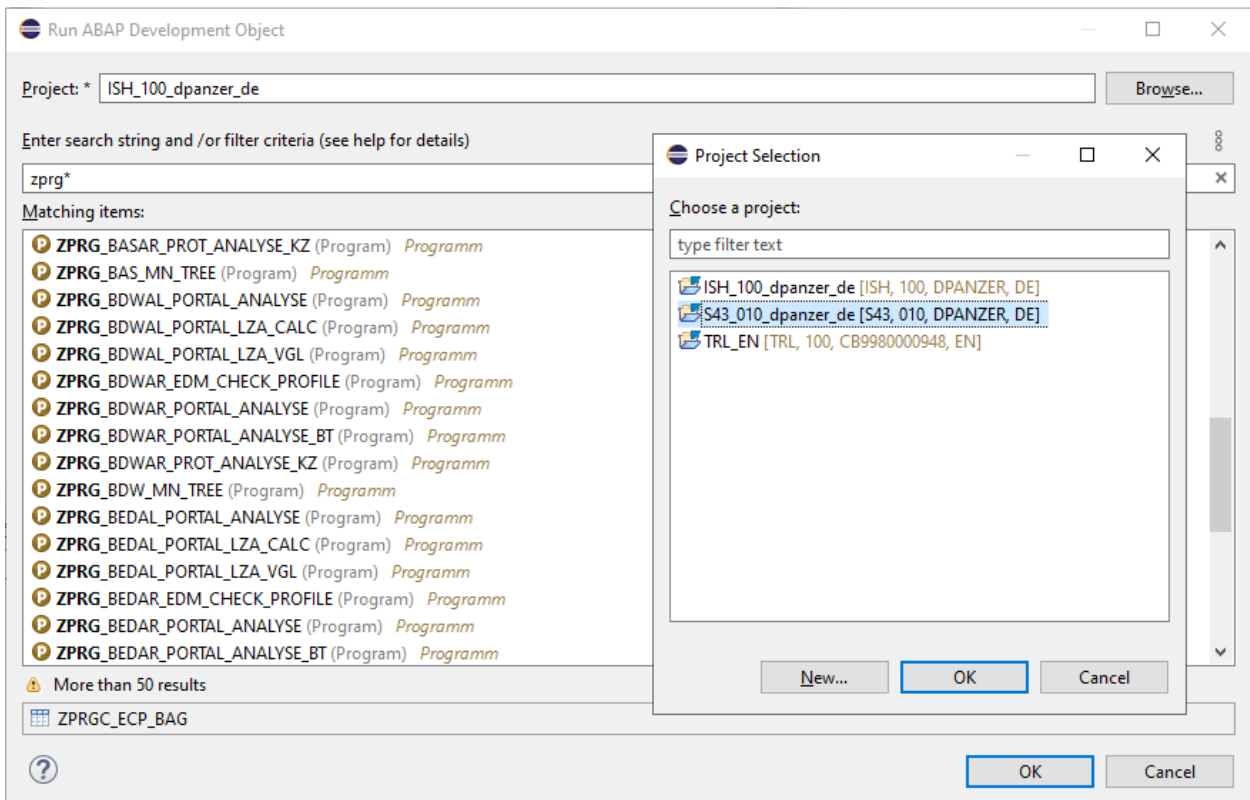
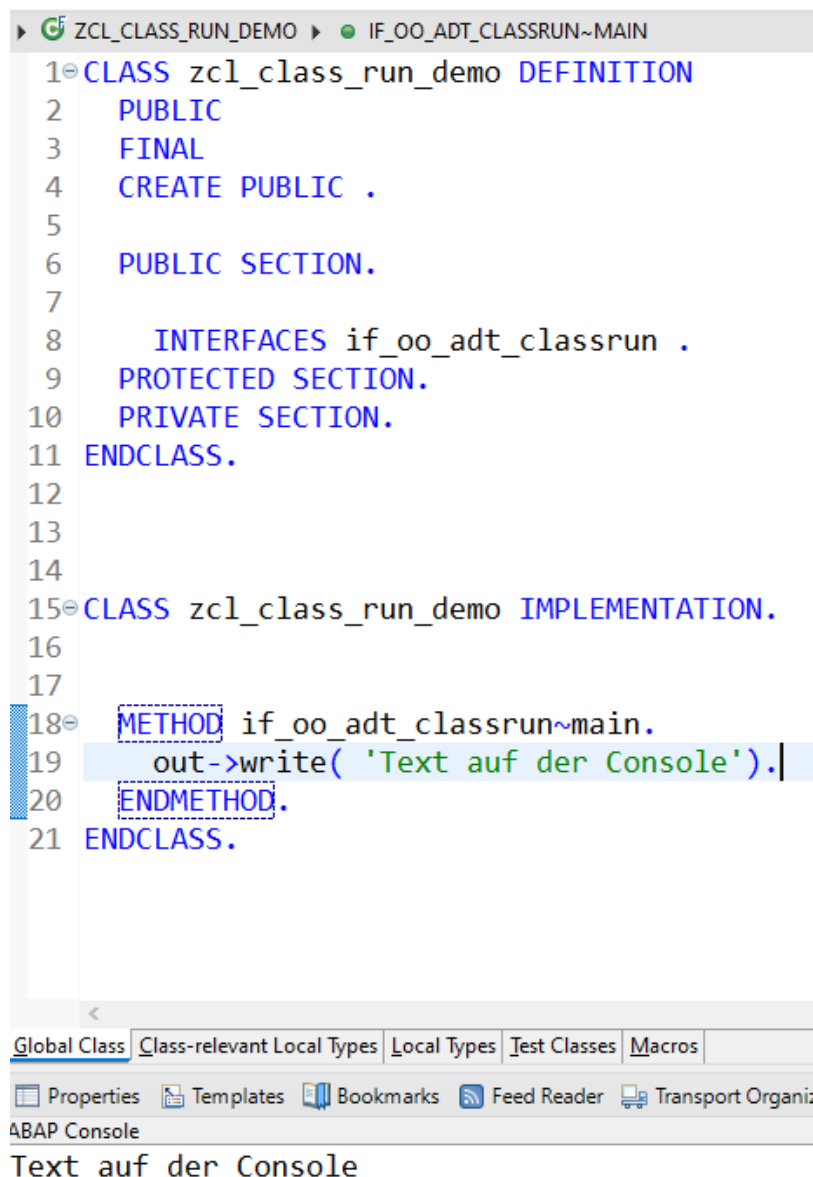


Abbildung 86 Auswahl des Projekts

Über den Menüpunkt “Run → Run History” stehen zusätzlich Informationen über bereits ausgeführte Objekte zur Verfügung, so dass deren Ausführung komfortabel wiederholt werden kann.

Klassen, die das Interface `if_oo_adt_classrun` implementieren, können ebenfalls direkt über F9 als **Konsolen-Applikation** ausgeführt werden und erzeugen somit Output in der Konsole.

Reports, die eine Write-Ausgabe erzeugen, können über F9 ausgeführt werden. Die WRITE-Ausgabe wird dann ebenfalls in die Konsole umgeleitet.



```
1 CLASS zcl_class_run_demo DEFINITION
2   PUBLIC
3   FINAL
4   CREATE PUBLIC .
5
6   PUBLIC SECTION.
7
8   INTERFACES if_oo_adt_classrun .
9   PROTECTED SECTION.
10  PRIVATE SECTION.
11 ENDClass.
12
13
14
15 CLASS zcl_class_run_demo IMPLEMENTATION.
16
17
18 METHOD if_oo_adt_classrun~main.
19   out->write( 'Text auf der Console').|
20 ENDMETHOD.
21 ENDClass.
```

Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

Properties | Templates | Bookmarks | Feed Reader | Transport Organ...

ABAP Console

Text auf der Console

Abbildung 87 Ausgabe in die Console

3.2.10 Data Preview

Die View **Data Preview** kann man nutzen, um sich Daten von Datenbanktabellen und (CDS-)Views anzeigen zu lassen. Die View öffnet sich, indem man entweder im Project Explorer ein entsprechendes Objekt markiert und den Shortcut **F8** drückt, oder das Kontextmenü nutzt.

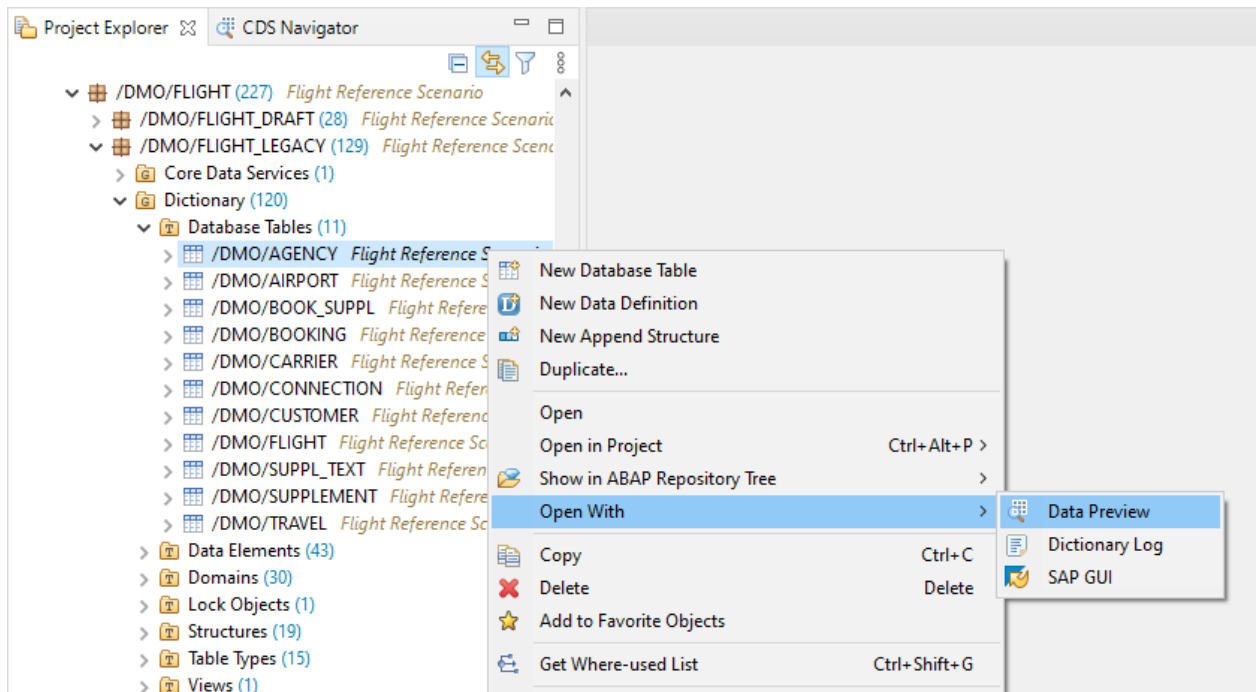


Abbildung 88 Starten des Data Preview über die Tabelle

Der Shortcut **F8** funktioniert auch, wenn man ein entsprechendes Objekt geöffnet hat und der Fokus auf dem Objekt liegt.

Die View führt beim Öffnen unverzüglich die Datenselektion aus und listet dann die selektierten Daten in Tabellenform auf. Zusätzlich zeigt sie die Anzahl der selektierten Zeilen und die benötigte Zeit dafür an.

The screenshot shows the Data Preview view in Eclipse IDE. The view displays a table of data with the following columns: CLIENT, TRAVEL_ID, AGENCY_ID, CUSTOMER_ID, BEGIN_DATE, END_DATE, BOOKING_FEE, TOTAL_PRICE, and CURRENCY_CODE. The table contains 100 rows of data, with the first few rows visible. The status bar indicates '100 rows retrieved - 9 ms (partial result)'. The view also shows a 'Filter pattern' field and a 'Show Log' button.

CLIENT	TRAVEL_ID	AGENCY_ID	CUSTOMER_ID	BEGIN_DATE	END_DATE	BOOKING_FEE	TOTAL_PRICE	CURRENCY_CODE
200	00004141	070002	000001	2021-06-24	2021-07-01	500.00	500.00	EUR
200	00004142	070001	000001	2021-06-24	2021-07-01	0.00	0.00	
200	00000003	070046	000093	2021-02-27	2021-12-26	80.00	4164.00	USD
200	00000004	070042	000665	2021-02-27	2021-12-26	40.00	1871.00	USD

Abbildung 89 Anzeige des Data Preview

Man hat in der View diverse Möglichkeiten, die Selektion anzupassen. Dazu zählen:

- Anzahl der selektierten Zeilen
- Selektierte Spalten
- Filterkriterien
- Sortierung (Anklicken der Spaltenüberschrift)

Außerdem kann man nach einem Muster in den angezeigten Daten suchen (inkl. ? und * als Joker-Zeichen). Passende Daten werden dann farbig und fett hervorgehoben. Darüber hinaus können die Gesamtanzahl der betroffenen Einträge und ein Log über ausgeführte Aktionen eingesehen werden. Über den Speichern-Button kann man die angezeigten Werte in unterschiedlichen Formaten innerhalb einer Datei speichern. Es ist sogar möglich, ein ABAP Value Statement dadurch zu generieren, was sehr nützlich zur Erstellung von Testdaten sein kann.

Bei **CDS-Views mit Associations** ist es möglich, den Associations zu folgen und somit die verknüpften Daten anzuzeigen. Dazu markiert man einen der Datensätze und wählt oben über den Pfeil die gewünschte Association aus.

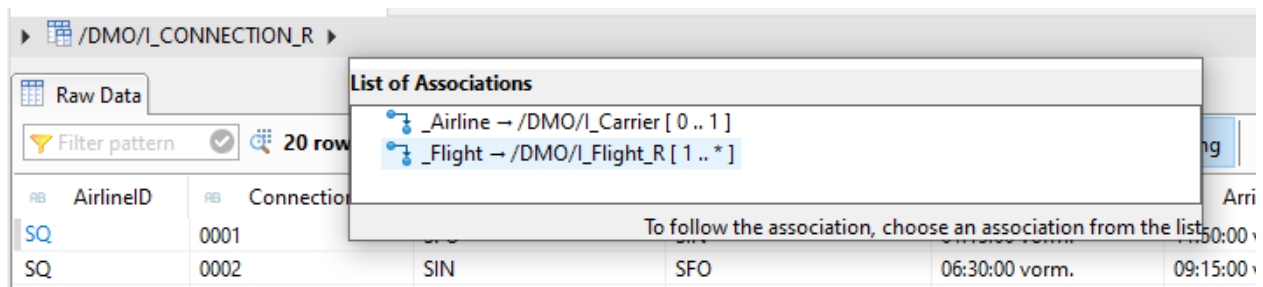


Abbildung 90 Navigation über Assoziationen

Ein Highlight des Data Preview Views ist die **SQL Console**. Anhand der selektierten Spalten, angegebenen Filtern und der Sortierung wird ein SQL Select Statement generiert, welches dann zur Selektion der Daten verwendet wird.

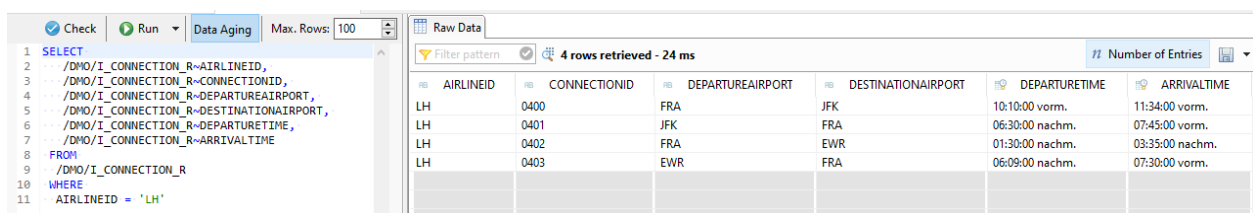


Abbildung 91 SQL Console

Dieses SQL Statement kann man individuell anpassen, prüfen und ausführen. Dabei gelten folgende Regeln und Einschränkungen:

- Es sind nur SELECT Statements gemäß ABAP Open SQL Syntax erlaubt.
- Es ist möglich, Aggregationen und komplexe Selektionen, wie z. B. mit JOIN und UNION, zu erstellen.
- Es sind nur lesende Zugriffe möglich (keine SQL Statements mit Datenänderung)
- Schlüsselwörter mit Bezug zu internen Tabellen können nicht verwendet werden

Man kann die SQL Console auch direkt aufrufen, indem man im Project Explorer das Kontextmenü für das ABAP Projekt öffnet und den Menüpunkt SQL Console auswählt. Der View zeigt das zuletzt verwendete SQL Statement an und führt es unverzüglich aus.

Alles in allem ist die SQL Console ein mächtiges Tool, mit dem man unkompliziert Selektionen ausführen, Daten auswerten oder Anpassungen an Selects testen kann.

Details zum Data Preview sind in der Eclipse-Hilfe für ADT zu finden.

3.2.11 Core Data Services

Im Bereich der Core Data Services (CDS) gibt es unterschiedliche **Dateitypen**, die in den ADT angelegt werden können:

- Data Definitions (DDLs) - Quelltextdateien für Datenmodelle in den folgenden Varianten:
 - DDIC Based CDS Views
 - CDS View Entities
 - Abstrakte CDS-Entitäten
 - Hierarchien
 - Erweiterungen der Views
 - CDS Table Functions
- Access-Control-Dateien (DCLS) - Zugriffs-Definitionen
- Metadata Extensions (DDLX) - Auslagerung von Annotationen aus der CDS-Definition
- Behavior Definitions (BDEF) - Verhaltens-Definitionen für RAP-Business-Objekte

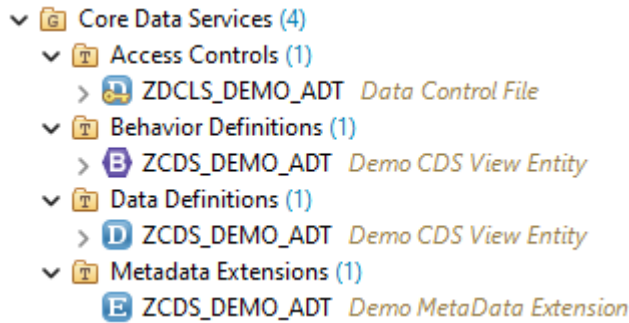


Abbildung 92 Objekttypen der CDS in der Navigation

Die Quelltext Editoren für die unterschiedlichen Dateitypen der Core Data Services verhalten sich weitgehend wie der Quelltexteditor für ABAP-Code. Unter anderem sind folgende Features enthalten:

- Code Completion (**STRG+SPACE**) - Vorschlagswerte, die im Kontext passen.
- Element Info (**F2**) - Informationen über das Element, auf dem der Cursor steht.
- Pretty Printer (**SHIFT+F1**)

Ein Unterschied sind die Farben, die im Editor verwendet werden.

Ein Grundproblem bei den Core Data Services ist, dass die Eigenschaften eines Objektes (z. B. einer CDS View Entity) aus mehreren Dateien und den Eigenschaften der Datenquellen zusammengesetzt sind. Diese Dateien erklären jeweils ihre Zugehörigkeit, und die Datenquellen propagieren ihre Feldeigenschaften (Annotationen). Das ist für das Erweiterungskonzept sehr praktisch. Aber die Transparenz leidet darunter, denn die Dateien liegen nicht notwendigerweise im gleichen Entwicklungspaket.

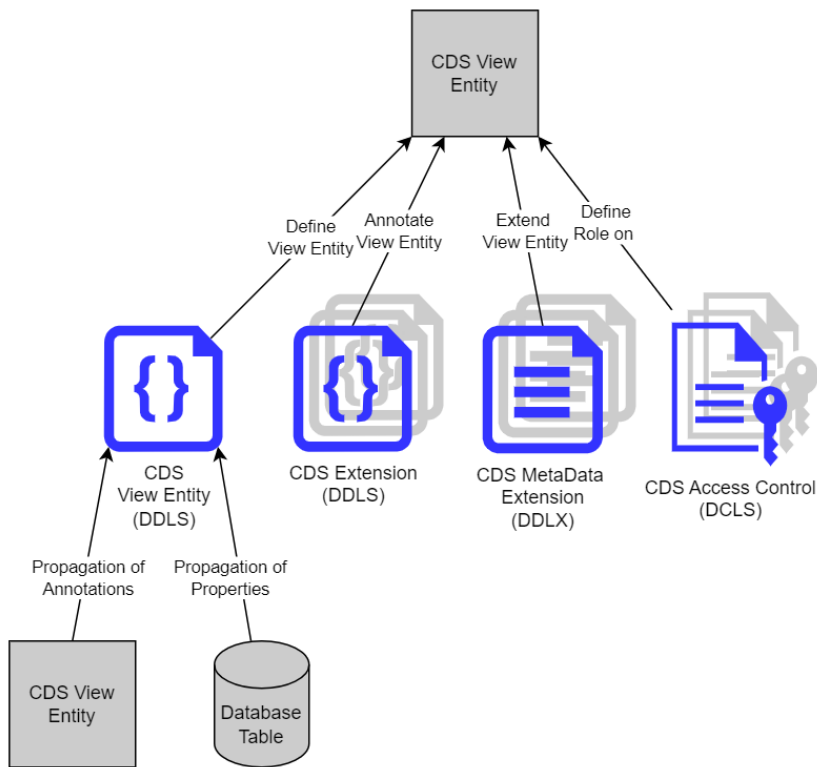


Abbildung 93 Unterschiedliche Dateien legen die Eigenschaften einer CDS View Entity fest

Um ein vollständiges Bild eines CDS-Objektes unter Berücksichtigung aller Dateien und Propagationen zu bekommen, sind darum Hilfsmittel notwendig. Dazu gehören:

- Element Info
- Dependency Analyzer
- Active Annotations

3.2.11.1 Element Info für CDS

Mit [F2](#) oder der separaten [Element Info View](#) bekommen Sie für einen CDS View eine gute Übersicht über die Datenstruktur und Assoziationen, unabhängig davon, wo diese definiert wurden. Außerdem werden alle relevanten Erweiterungsdateien angezeigt.

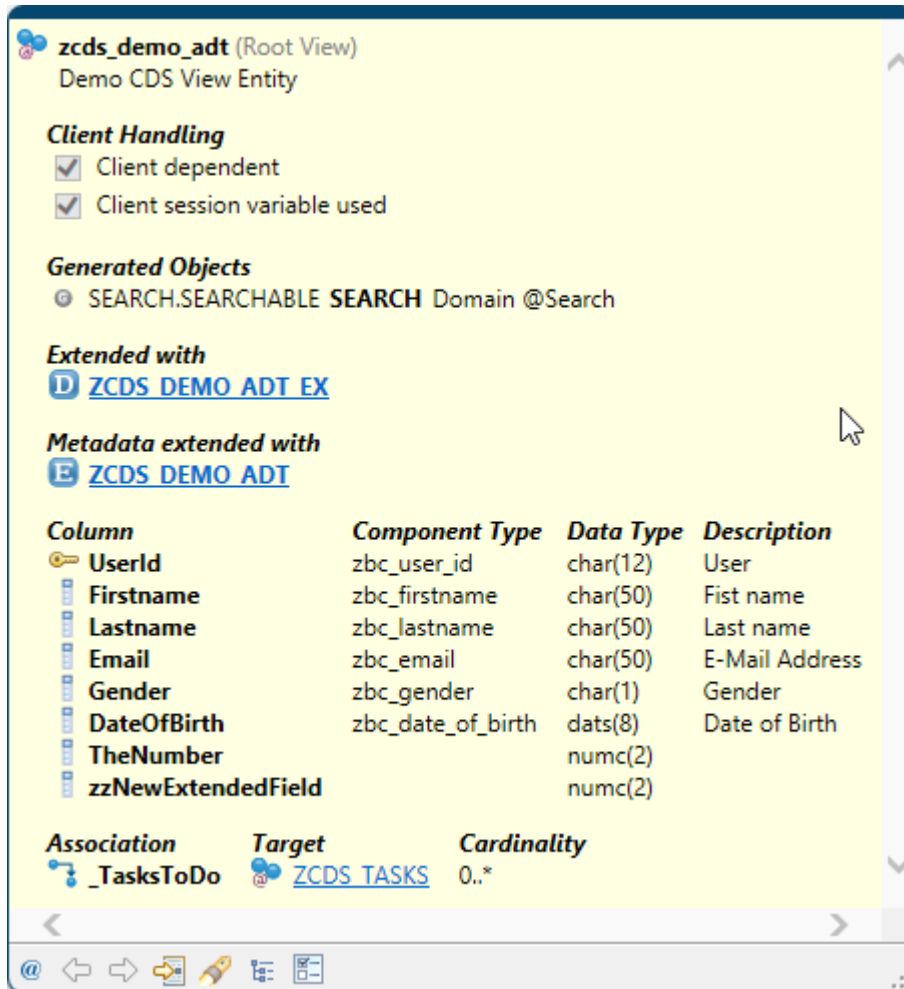


Abbildung 94 Übersicht über ein CDS View Entity mit Hilfe von Element Info

3.2.11.2 Dependency Analyzer

Der [Dependency Analyzer](#) bietet eine gute Übersicht über die Herkunft der Daten. Er wird über das Kontextmenü aufgerufen..

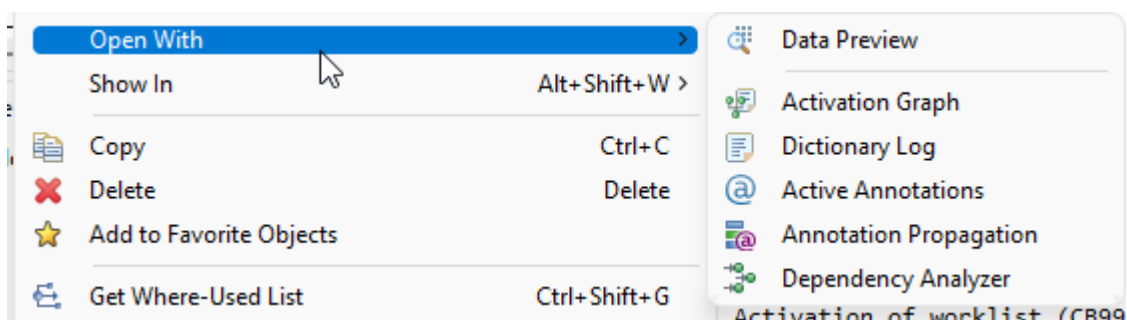


Abbildung 95 Aufruf des Dependency Analyzers über das Kontextmenü

Der Dependency Analyzer hat drei Tab-Reiter, die Informationen über eine View anzeigen:

- Der **SQL Dependency Tree** zeigt die hierarchische Struktur in Tabellenform an

SQL Name	SQL Relation	Object Type	Entity Name	Database Object	Access Control
▼ ZC_STATUS		CDS Projection View (STOB)	zc_status	True	None
▼ ZI_STATUS	From	CDS View Entity (STOB)	zi_status	True	None
ZBC_STATUST	From	Database Table (TABL)		True	
▼ ZI_STATUS_TEXT	Left Outer Join	CDS View Entity (STOB)	zi_status_text	True	None
ZBC_STATUS_TEXT	From	Database Table (TABL)		True	

Find

SQL Dependency Tree | SQL Dependency Graph | Complexity Metrics

Abbildung 96 SQL Dependency Tree

- Der **SQL Dependency Graph** zeigt die gleichen Informationen grafisch an.

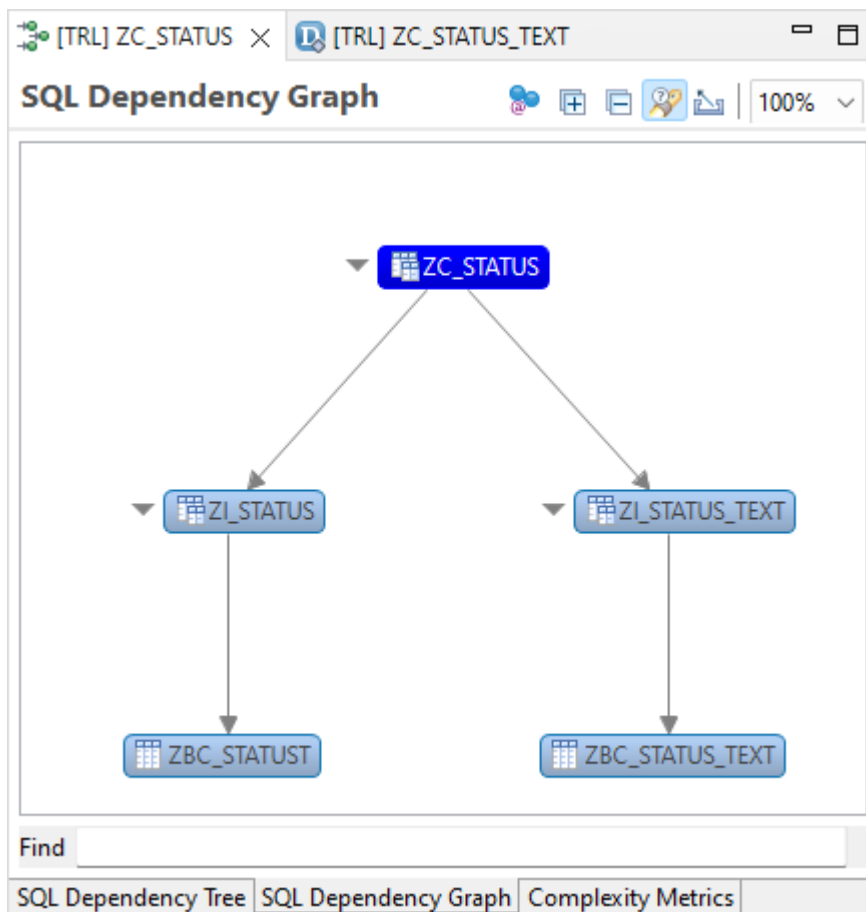


Abbildung 97 SQL Dependency Graph

- Der Tab-Reiter **Complexity Metrics** zeigt zusätzliche Informationen zur Gesamtkomplexität des CDS Views an, inklusive aller Quell-Views. Gerade bei Performance-Problemen lohnt hier ein Blick.

Used Data Sources	
Database tables:	2
Database views:	2
CDS views with parameter:	0
CDS table functions:	0
SQL Operations	
JOIN operations:	1
SET operations:	0
GROUP BY clauses:	0
Performance Related Function Calls and Operations	
Function calls:	0
Explicit type CAST operations:	0
CASE expressions:	0

SQL Dependency Tree | SQL Dependency Graph | **Complexity Metrics**

Abbildung 98 Complexity Metrics

3.2.11.3 Active Annotations

Die [View Active Annotations](#) wird ebenfalls über das Kontextmenü in der Navigation aufgerufen. In ihr werden die Werte sämtlicher aktiven Annotationen der Views angezeigt:

- Auf **View-Ebene** sind nur genau die Annotationen sichtbar, die in dem View definiert wurden.
- Auf **Feld-Ebene** sind alle gültigen Annotationen sichtbar. Ihre Herkunft, wie beispielsweise Datenelement, Metadata-Extension oder Datenquelle, wird ebenfalls mit angezeigt.
- Auf **Parameter-Ebene**

Manche Eigenschaften, z. B. Feldtexte, können schon durch die Datenelemente festgelegt und diese dann in die View propagiert werden. Dies ist in folgender Abbildung beispielhaft dargestellt:

The screenshot shows the Eclipse IDE interface with the 'Active Annotations for Entity zi_tasks' window open. The window title is 'Active Annotations for Entity zi_tasks' and it has a search bar labeled 'type filter text'. The main content is a table with the following columns: 'Annotated Elements', 'Annotation Value', 'Translated Text', 'Origin Data Source', and 'Origin Data Type'. The table is expanded to show annotations for the 'zi_tasks' entity, including '@AccessControl', '@EndUserText', 'TaskKey', '@ObjectModel', and 'Summary'.

Annotated Elements	Annotation Value	Translated Text	Origin Data Source	Origin Data Type
zi_tasks				
Entity annotations				
@AccessControl				
authorizationCheck	#NOT_REQUIRED			
@EndUserText				
label	'Tasks'			
TaskKey				
@EndUserText				
heading			ZBC_TASKS (Database Table)	ZBC_TASK_KEY (Data Element)
label		Key	ZBC_TASKS (Database Table)	ZBC_TASK_KEY (Data Element)
quickInfo		Key	ZBC_TASKS (Database Table)	ZBC_TASK_KEY (Data Element)
@ObjectModel				
upperCase	true		ZBC_TASKS (Database Table)	ZBC_TASK_KEY (Data Element)
Summary				
@EndUserText				
heading			ZBC_TASKS (Database Table)	ZBC_TASK_SUMMARY (Data Elem...
label			ZBC_TASKS (Database Table)	ZBC_TASK_SUMMARY (Data Elem...
quickInfo		Task summary	ZBC_TASKS (Database Table)	ZBC_TASK_SUMMARY (Data Elem...

Abbildung 99 Aktive Annotationen eines Views

4 Troubleshooting-Werkzeuge in Eclipse

Dieses Kapitel gibt einen Überblick über die Werkzeuge, die in den ABAP Development Tools zum **Troubleshooting** zur Verfügung stehen. Dazu zählt u. a. Debuggen von Entwicklungsartefakten, Performance-Analyse und weitere Analyse- und Fehlerfindungsmethoden.

Neben der Beschreibung der Tools geben wir Hinweise für den sinnvollen Einsatz. Für detaillierte Funktionsbeschreibungen empfehlen wir die offizielle Dokumentation und geben, sofern vorhanden, hilfreiche Links an.

4.1 Troubleshooting mit den ABAP Development Tools

Wer den Einstieg in die ABAP Development Tools geschafft hat und die Vorzüge von ADT bei der Entwicklung zu schätzen weiß, möchte ungern weiter in SAP-GUI-basierter Umgebung entwickeln. Doch die ABAP Development Tools sind nicht nur eine Umgebung, um Code zu erstellen und zu ändern. In ADT finden sich zahlreiche Tools zur generellen Analyse von Code, dem Finden von Fehlern und der detaillierten Analyse der Performance der Funktionalitäten. Zusammenfassend können diese Aktivitäten als Troubleshooting bezeichnet werden.

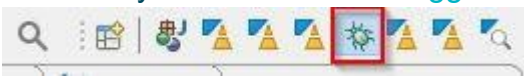
Aber wie bereits beim Einstieg in die Entwicklung mit ADT, gibt es bei den Troubleshooting-Tools eine steile Lernkurve zu überwinden, deren Bezwingung mit zahlreichen Vorteilen belohnt wird. Beispielhaft sei hier erwähnt, dass es möglich ist, im Debugger den fehlerhaften Code direkt ändern zu können, statt umständlich parallel einen neuen Modus zu öffnen, die betreffende Stelle zu suchen und dann die Änderung durchzuführen. So gibt es in den Troubleshooting-Tools viele Vorteile zu entdecken, die hier in den einzelnen Abschnitten beschrieben werden.

Neben den direkten Vorteilen ist der Aspekt "Eclipse-ADT als gleiches Tool für alles" auch im Bereich des Troubleshootings nicht zu vernachlässigen.

Wenn man z. B. im SE80-Debugger sehr versiert ist, fällt einem der Umstieg schwerer, da die Vorteile des ADT-Debugger erst erarbeitet werden müssen. Für den Entwicklungsprozess in Summe ist es aber sinnvoll, ein Tool für alle Anwendungsfälle zu nutzen und das Hin- und Herwechseln zwischen den Umgebungen zu reduzieren bzw. gänzlich zu vermeiden.

4.2 Der Debugger in den ABAP Development Tools

Im Eclipse steht für Analysezwecke ein **Debugger** zur Verfügung. Dieser wird über die

Schaltfläche  gestartet. Mit dem Starten wechselt die Perspektive in Eclipse automatisch auf die Perspektive des Debugger.

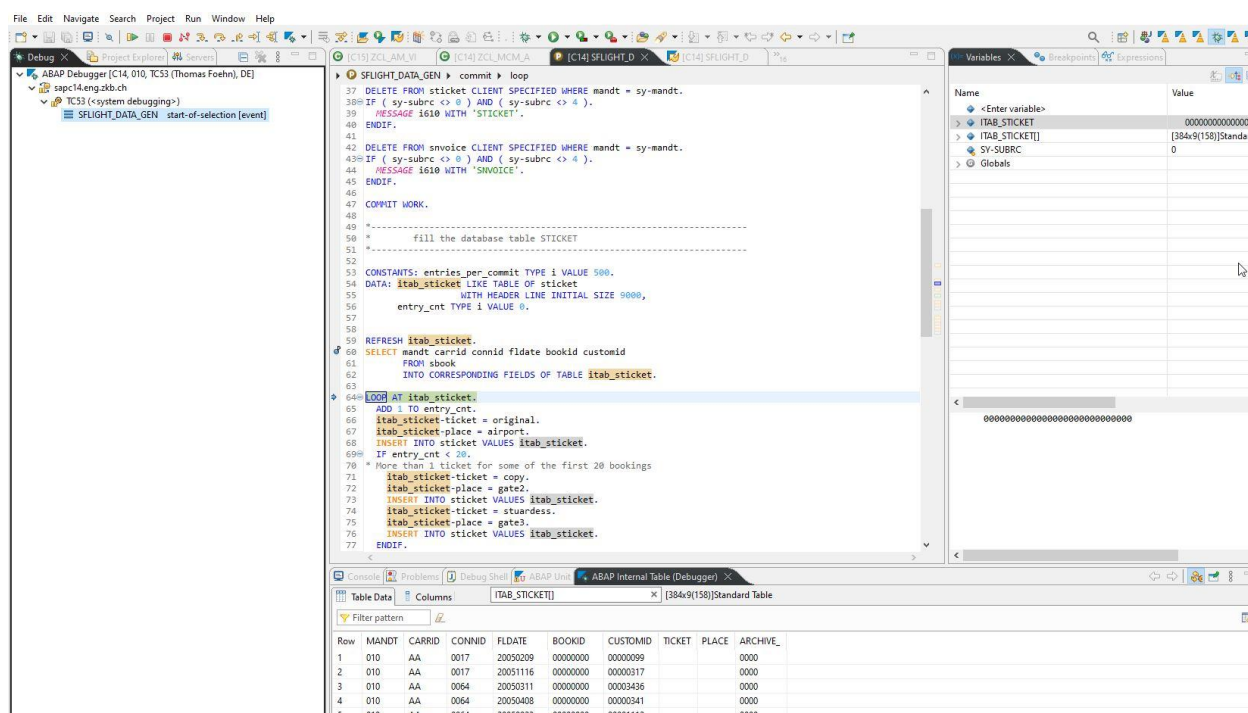
4.2.1 Breakpoints und Soft-Breakpoints

Breakpoints in Eclipse sind ausschließlich externe Breakpoints. Die Breakpoints werden bei jedem Durchlauf der Software gezogen. Der Ablauf wird an der entsprechenden Stelle unterbrochen.

Breakpoints können in allen Perspektiven im Editor links neben der Zeilennummer gesetzt werden. Alternativ können Breakpoints über das Kontextmenü gesetzt werden. Gesetzte Breakpoints werden durch einen blauen Punkt neben der Codezeile gekennzeichnet.

Die ABAP Development Tools bieten neben den normalen Breakpoints die Möglichkeit von Soft Breakpoints. Diese werden über das Kontextmenü gesetzt und mit einem grünen Punkt gekennzeichnet. Im Gegensatz zu den Standard-Breakpoints wird der Programmablauf an der Stelle nur gestoppt, wenn die Software im Debugging-Kontext läuft. Ansonsten werden Soft-Breakpoints übersprungen.

4.2.2 Debugging-Perspektive



Row	MANDT	CARRID	CONNID	FLDATE	BOOKID	CUSTOMID	TICKET	PLACE	ARCHIVE_
1	010	AA	0017	20050209	00000000	00000099			0000
2	010	AA	0017	20051116	00000000	00000317			0000
3	010	AA	0064	20050311	00000000	00003436			0000
4	010	AA	0064	20050408	00000000	00000341			0000
5	010	AA	0064	20050923	00000000	00001112			0000

Abbildung 100 Debugging Perspektive in Eclipse

Die Debugging-Perspektive in Eclipse bietet einen schnellen Überblick über den Programm-Code, Call-Stack, Variableninhalte und Inhalte von internen Tabellen. Die Variablen und interne Tabellen können mit Doppelklick im Programm-Code ausgewählt werden. Sie werden auf der rechten Seite angezeigt.

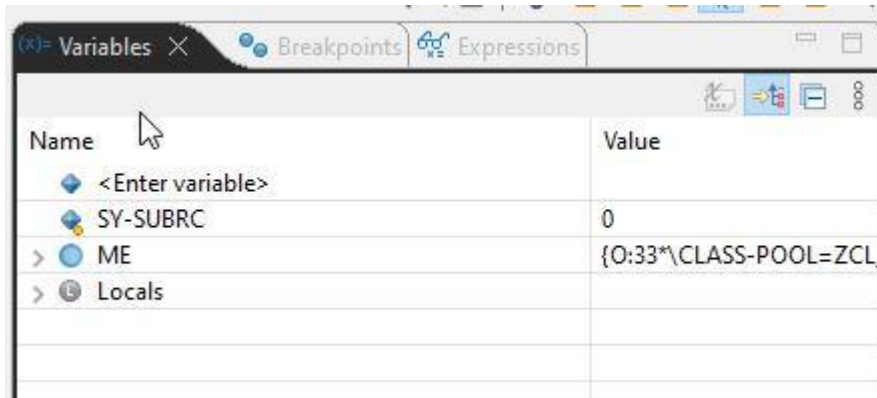


Abbildung 101 Werte der Variablen in der Debugging Perspektive

4.2.3 Besonderes Verhalten im Debugger

Im Debugger der ABAP Development Tools ist es möglich, den Code direkt zu modifizieren und zu aktivieren. Im aktuellen Debugging-Kontext ist das Coding jedoch noch nicht aktiv. Der Programmlauf muss neu gestartet werden.

4.2.4 Weitere Informationen

Weitere Besonderheiten über Debugging mit den ABAP Development Tools in Eclipse finden sich in den SAP-Blogs. Hervorzuheben sind dabei die folgenden beiden. Sie beschreiben Fälle und Lösungen, die beim Arbeiten mit dem Debugger auftreten können.

<https://blogs.sap.com/2020/04/21/adt-abap-debugger-what-to-do-if-your-program-does-not-stop-at-breakpoints/>

<https://blogs.sap.com/2015/11/02/breakpoint-validity-scope-and-activation-conflicts-in-abap-development-tools-adt/>

4.3 Checkpoint IDs und dynamische Logpoints

Ein sehr hilfreiches Tool im Bereich der Fehleranalyse und Debugging sind die sogenannten Checkpoint IDs. Diese können über die Transaktion SAAB oder in ADT unter "others" angelegt werden. Diese IDs werden mittels der Befehle:

- BREAK POINT ID [GRUPPENNAME]
- LOG POINT ID [GRUPPENNAME]
- ASSERT ID [GRUPPENNAME]

im Code verankert. Für den detaillierte Syntax und die Optionen der Befehle verweisen wir auf die [SAP-Hilfe](#).

Wie bereits im Debugger des SAP GUI können diese dynamischen Breakpoints für das Debuggen aktiviert bzw. zur Protokollierung genutzt werden. Der Hauptvorteil liegt hier darin, dass der Entwickler im Voraus wichtige Stellen im Code mit Breakpoints versehen kann. Falls der Code analysiert werden soll, muss die Checkpoint ID nur einmalig entsprechend aktiviert werden. Beim Aufruf der Einheit wird dann bei einem aktiven Breakpoint der Debugger an der betreffenden Stelle aufgerufen.

Um die Checkpoints effektiv einzusetzen, empfiehlt es sich, entsprechende Templates anzulegen, die dann einfach mittels Quick Fixes aufgerufen werden können ([siehe Kapitel 3 - Arbeiten mit ADT im Abschnitt zu Templates](#)).

Während Checkpoint IDs auch im GUI-basierten Debugger genutzt werden können, bietet ADT für On-Premise-Systeme zusätzlich die Möglichkeit, dynamische Log Points im Debugger zu setzen, die zum Auslesen programm-interner Werte verwendet werden können. Diese Option ist dann hilfreich, wenn es nicht möglich ist, den Produktivcode zu ändern oder auch eine Analyse sehr zeitnah auf dem Produktivsystem durchgeführt werden muss.

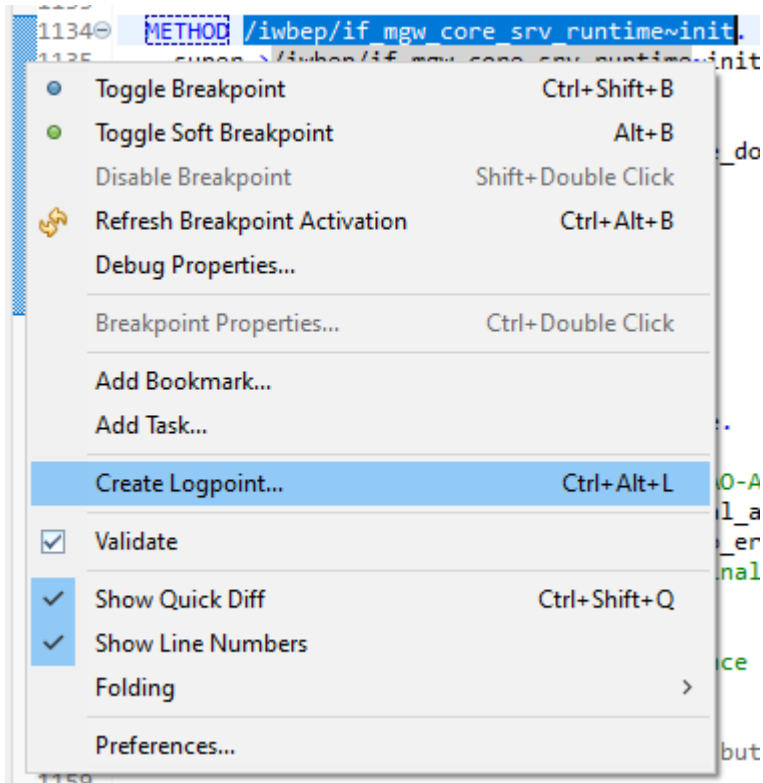


Abbildung 102 Erstellung eines Log Points über das Kontextmenü

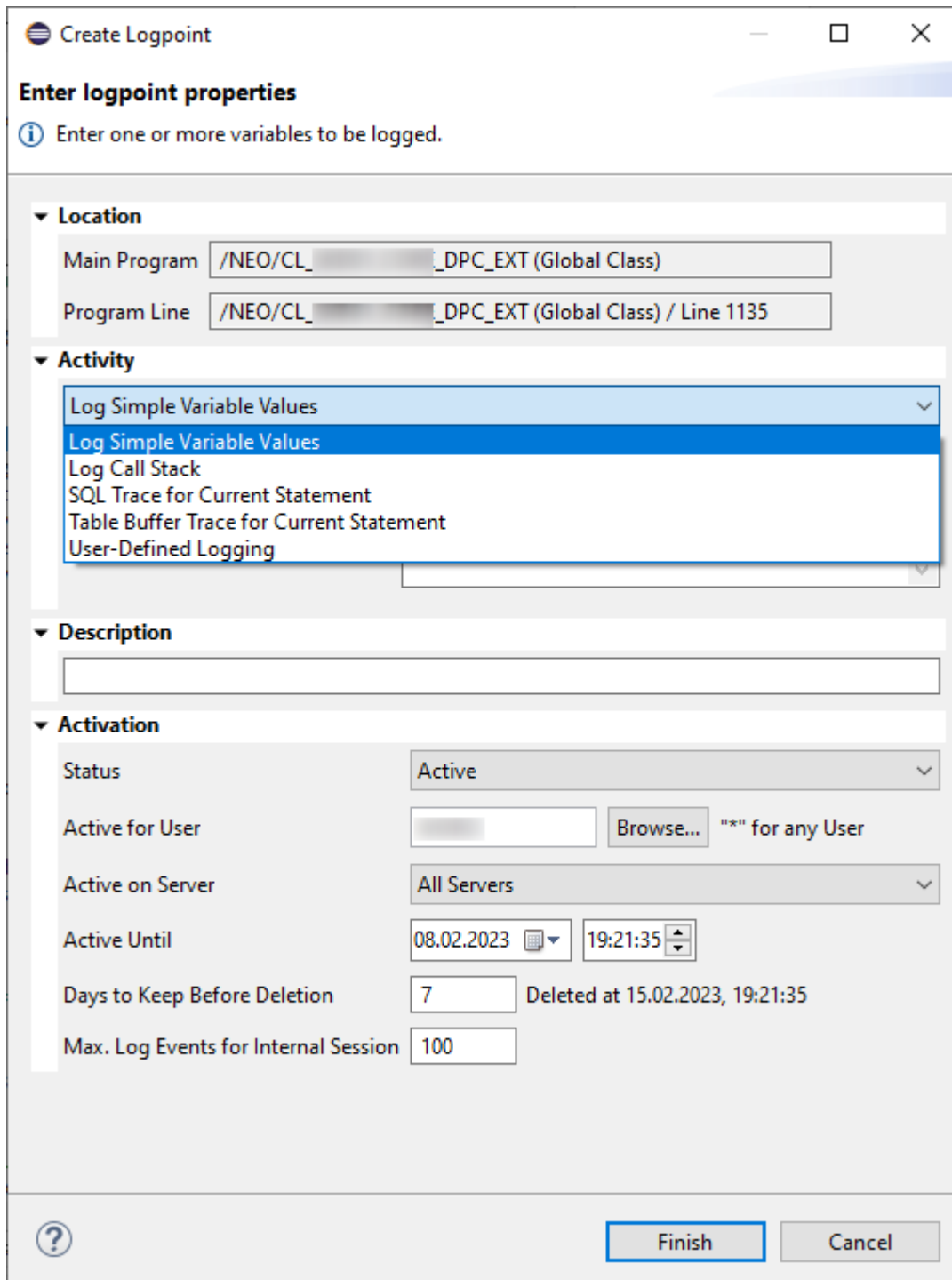


Abbildung 103 Attribute beim Erstellen eines Log Points

Sie können in dem Dialog entscheiden, was im Log aufgezeichnet werden soll, Sie können dem dynamischen Logpoint eine Beschreibung mitgeben, die dann in der Log-Ausgabe verwendet wird, und Sie können verschiedene Kriterien mitgeben, ob (anhand einer Bedingung – in der Bildschirmkopie verdeckt – und/oder Benutzer/Server) und wie lange die Log-Ausgabe erfolgen soll. Erstellte Logpoints

werden im Editor am linken Rand und am rechten Rand neben der vertikalen Scroll-Leiste angezeigt sowie im View “Logpoints” aufgelistet:

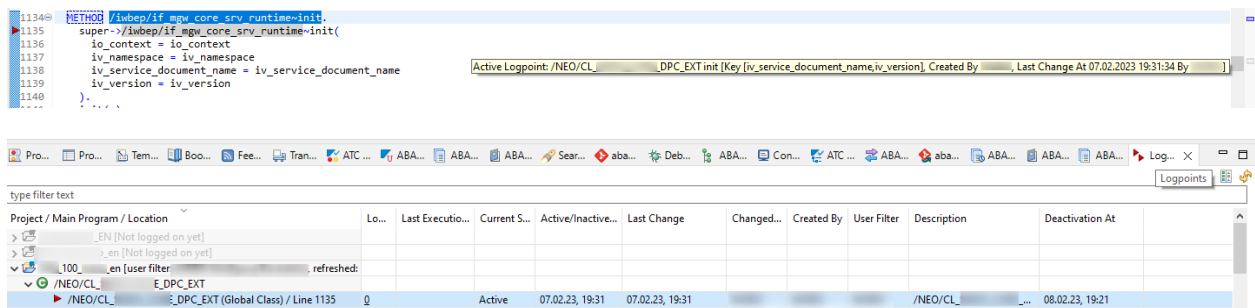


Abbildung 104 Log Points View in der Debugging Perspektive

Wir empfehlen, hierzu die SAP-Dokumentation ([On-Premise](#)) zu studieren, um Details über die Anwendung zu erfahren. Einen guten Einstieg bietet hierzu auch dieser [Blogbeitrag: Dynamic Logpoints in ABAP | SAP-Blogs](#)

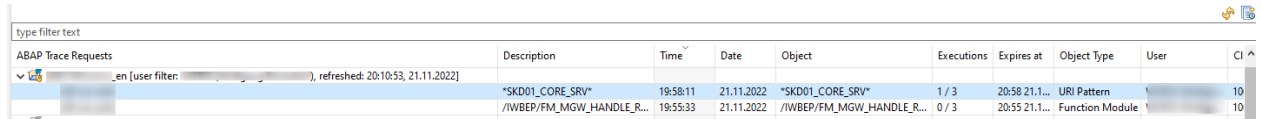
4.4 Performance-Analyse

Für eine integrierte und graphische Performance-Analyse bieten Ihnen die ADT einen komfortablen Zugang zum ABAP Profiler ([On-Premise/Cloud](#)) als Nachfolger u.a. der Transaktion SAT. Zum Starten des Profilers gibt es verschiedene Möglichkeiten:

Wenn Sie ein ausführbares Programm (nur On-Premise), eine Konsolenanwendung (nur ABAP Cloud) oder ein Artefakt mit zugeordneten Unit-Tests geöffnet haben, können Sie den Profiler direkt über das Kontextmenü “Profile as” starten. Alternativ dazu kann auch der Wizard verwendet werden, der über das Run-Menü erreichbar ist.

Für den Fall, dass man einen anderen Startpunkt benötigt, kann man mit Trace-Requests Anforderungen für den Start des Profilers erstellen. Dazu zuerst die View “ABAP Trace Requests” (diese finden Sie in der View-Liste unterhalb von ABAP, vgl. [Views und Perspektiven](#)) einblenden. In dieser View bekommen Sie nach Auswahl eines Systems mit dem Create-Trace-Request-Icon einen Wizard. Mit diesem können Sie verschiedene Trigger für den Start des Profilers auswählen. Wird mit HTTP(S)-Zugriffen direkt auf das System zugegriffen (Achtung: Das ist in einer Hub/FES-Konfiguration im Backend normalerweise nicht der Fall), kann ein Muster für die URL verwendet werden, z. B. der Name des OData-Services mit vor- und nachgestelltem Stern. Des Weiteren können ein RFC-Aufruf eines Funktionsbausteins oder der Start eines Hintergrund-Jobs und diverse weitere Trigger für den Start des Trace verwendet werden. In der SAP-Fiori-Entwicklung kann in einer Hub/FES-Konfiguration für OData-Zugriffe im Backend-System der Funktionsbaustein /IWBP/FM_MGW_HANDLE_REQUEST als Trigger verwendet werden. Die Anzahl

der Trigger-Aktivierungen kann begrenzt werden, ebenso gibt es eine Möglichkeit zur zeitlichen Limitierung. Über verschiedene Einstellungen können Sie den Umfang der Datenermittlung steuern. Über das Kontextmenü in der Liste können Trace-Requests gelöscht werden.



Description	Time	Date	Object	Executions	Expires at	Object Type	User	CI
SKD01_CORE_SRV	19:58:11	21.11.2022	*SKD01_CORE_SRV*	1 / 3	20:58 21.1...	URI Pattern		10
/IWBEP/FM_MGW_HANDLE_R...	19:55:33	21.11.2022	/IWBEP/FM_MGW_HANDLE_R...	0 / 3	20:55 21.1...	Function Module		10

Abbildung 105 ABAP Trace Requests in der Debugging Perspektive

Darüber hinaus gibt es noch die Möglichkeit, aus dem ADT-Debugger heraus den Trace zu starten ([On-Premise/Cloud](#)).

Im Falle der Trace-Requests kann über das Aktualisierungs-Icon im Trace-Request-View die Anzahl der pro Anforderung bereits erzeugten Traces aktualisiert werden. Über das Kontextmenü bzw. per Doppelklick kann in die View mit der Liste der Traces gesprungen werden. Von dort kann die Anzeige eines Trace geöffnet werden. Entweder kann über das Kontextmenü gezielt in die verschiedenen Tabs der Analyse oder per Doppelklick in die Übersichtsseite gesprungen werden.

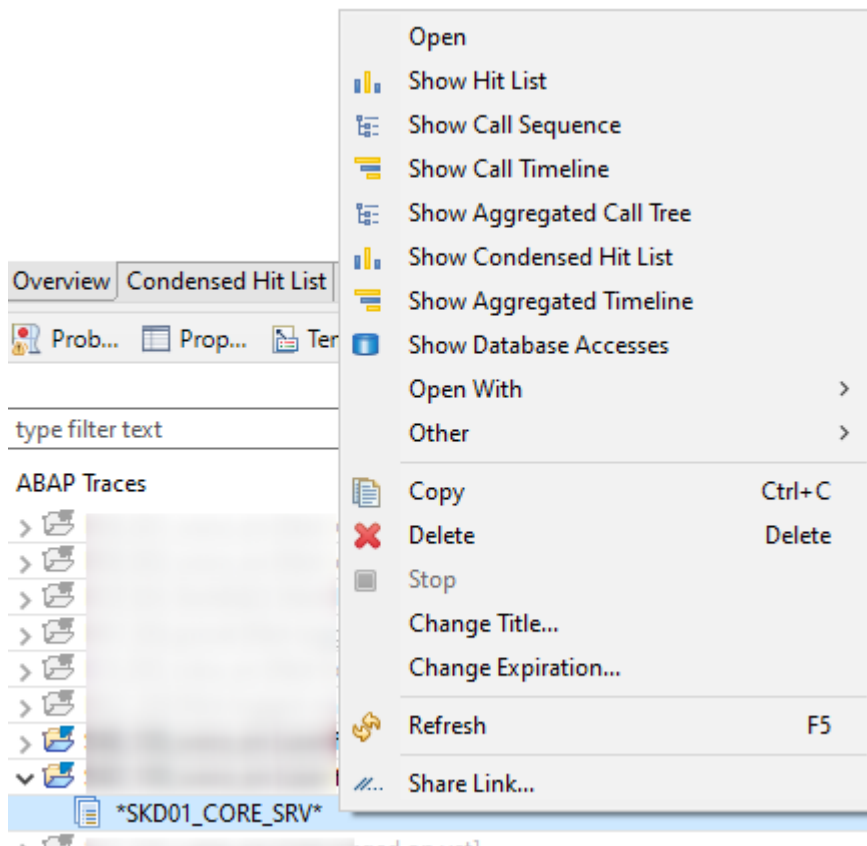


Abbildung 106 Kontextmenü eines Traces

Die Übersichtsseite bietet neben einem knappen Überblick über die Laufzeit ebenfalls direkte Absprungpunkte in die verschiedenen Tabs.

Overview

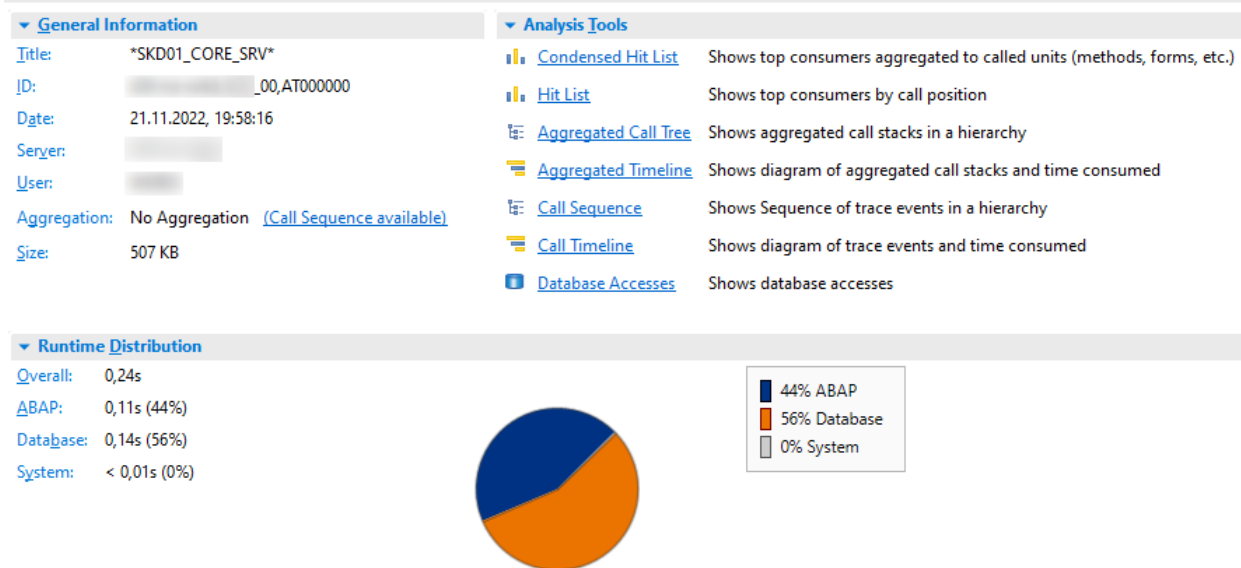


Abbildung 107 Übersicht über die Eigenschaften eines Traces

Insbesondere die graphische Analyse der Timeline ist nur in den ADT verfügbar und erleichtert eine Analyse. Bewegt man den Mauszeiger über die Blöcke, bekommt man direkt Details angezeigt und kann über das Kontextmenü auch direkt in den Quellcode navigieren.

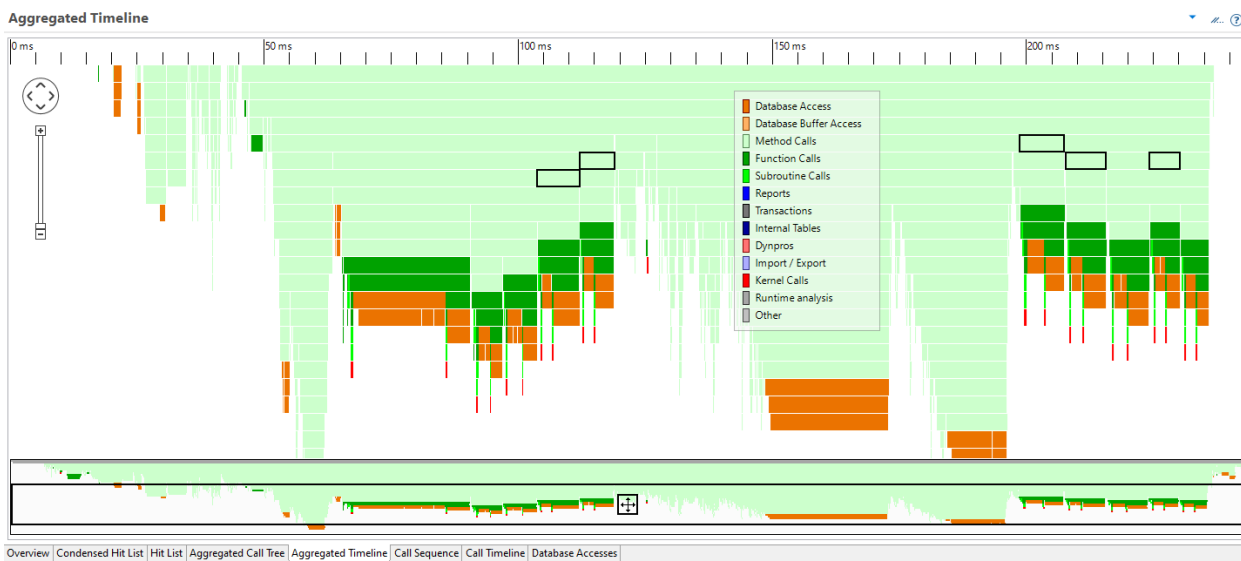


Abbildung 108 Aggregierte Übersicht eines Trace-Verlaufs

Der SQL-Trace, der insbesondere bei HANA als Datenbank die PLV-Dateien für eine visuelle Analyse der Query-Pläne liefert, kann im Kontextmenü eines Systems im

Project Explorer gestartet werden (funktioniert auch im ABAP Environment). Die Ergebnisdarstellung erfolgt dann aber in einer Webanwendung außerhalb der ADT (oder in der Transaktion ST05).

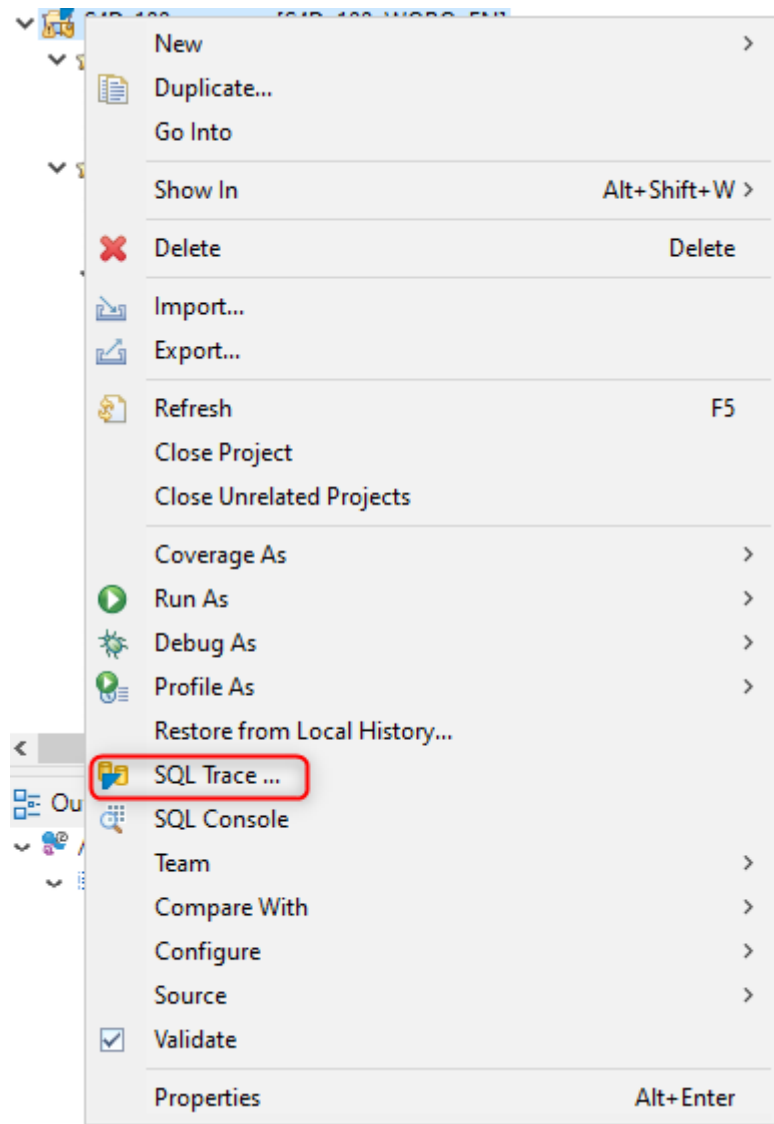


Abbildung 109 Absprung in den SQL Trace

Sie können für die visuelle Analyse aktuell noch die SAP HANA Administration Tools aus dem HANA Studio zusätzlich in der ADT-Eclipse (oder das komplette SAP HANA Studio parallel) installieren und somit den automatischen Start der visuellen Analyse aus der Transaktion heraus konfigurieren. Dazu setzen Sie den Benutzerparameter HDB_OPEN_STUDIO auf X

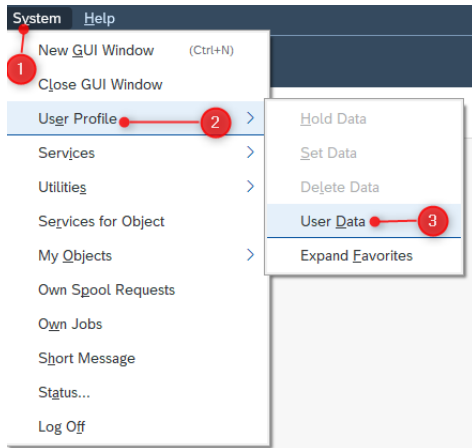


Abbildung 110 Einstieg in die Verwaltung der Benutzerparameter

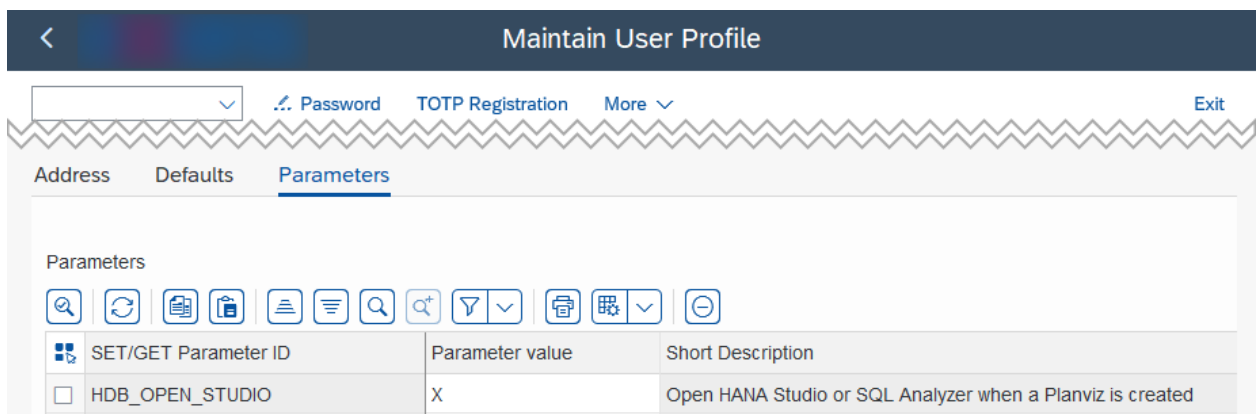


Abbildung 111 Setzen des Benutzerparameters HDB_OPEN_STUDIO

und verknüpfen im Betriebssystem das Öffnen von *.plv-Dateien mit der richtigen eclipse.exe. Beispielsweise gibt es unter Windows dafür im Datei-Explorer im Kontextmenü der Datei die Option "Öffnen mit..." und dort findet sich dann ganz unten die Option "Andere App auf diesem PC suchen", nach Klick müssen Sie die "eclipse.exe" ihrer Eclipse/ADT-Installation suchen und auswählen sowie die Option "Immer diese App ... verwenden" auswählen.



Abbildung 112 Auswahl der eclipse.exe als neue Möglichkeit zur Anzeige von *.plv Dateien

Nach Auswählen einer Zeile in den SQL-Trace-Liste der ST05 oder in Auswahl eines Trace-Record und eines SQL-Statement in der SQL-Trace-Analyse des Technical-Monitoring-Cockpit kann man die HANA-PlanViz-Query-Plan-Visualisierung anfordern:

	Start Time	Duration	Records	Program Name	Object Name	Statement
	■ 6.612.223		■ 1.338			
<input type="checkbox"/>	14:37:18.362	2.752	1	/NEO/CL_SKD_FLAT_CONFIG=====CP	/NEO/SKD_FCONF_S	SELECT <FDA READ> WHERE "MANDT
<input type="checkbox"/>	14:37:18.383	2.566	1	/NEO/CL_SKD_FLAT_CONFIG=====CP	/NEO/SKD_FCONF_S	SELECT <FDA READ> WHERE "MANDT
<input type="checkbox"/>	14:37:18.388	2.241	1	/NEO/CL_SKD_FLAT_CONFIG=====CP	/NEO/SKD_FCONF_S	SELECT <FDA READ> WHERE "MANDT
<input type="checkbox"/>	14:37:18.433	32.451	44	CL_SADL_SQL_EXECUTOR=====CP	/NEO/CSKDNDODRESP	SELECT WHERE "MANDT" = '205' AND
<input type="checkbox"/>	14:37:18.764	2.664	1	/NEO/CL_SKD_FLAT_CONFIG=====CP	/NEO/SKD_FCONF_S	SELECT <FDA READ> WHERE "MANDT
<input type="checkbox"/>	14:37:18.788	1.919	1	/NEO/CL_SKD_FLAT_CONFIG=====CP	/NEO/SKD_FCONF_S	SELECT <FDA READ> WHERE "MANDT
<input type="checkbox"/>	14:37:18.942	1.766	1	/NEO/CL_SKDEAM_S_PMCSOR_SAMPLECP	/NEO/SKD_INT	SELECT WHERE "MANDT" = '205' AND
<input type="checkbox"/>	14:37:19.083	2.821	110	/NEO/CL_SKDEAM_S_PMCSOR_SAMPLECP	/NEO/SKD_LEVEL, /NEO/SKD_SKL	SELECT <FDA READ> <JOIN> WHERE '
<input checked="" type="checkbox"/>	14:37:19.326	1.540.622	915	CL_SADL_SQL_EXECUTOR=====CP	ZI_NEOS_RESOURCETIMESLOTSASS	SELECT WHERE "MANDT" = '205' AND
<input type="checkbox"/>	14:37:20.877	7.465	39	ZCL_NEOS_RES_TIME_SLOTS=====CP	/NEO/SKD_RES_ADR	SELECT DISTINCT WHERE "MANDT" =
<input type="checkbox"/>	14:37:20.887	8.449	1	ZCL_NEOS_RES_FOR_SUGGESTION===CP	ZI_NEOS_SKILLMATCH	SELECT <FDA READ> WHERE "MANDT
<input type="checkbox"/>	14:37:20.893	2.373	1	ZCL_NEOS_RES_FOR_SUGGESTION===CP	ZI_NEOS_SKILLMATCH	SELECT <FDA READ> WHERE "MANDT

Abbildung 113 Auswahl einer konkreten Selektion

Troubleshooting-Werkzeuge in Eclipse

Technical Monitoring Cockpit

SQL Trace Analysis

Trace Directory Trace Records SQL Statement Prepared Plan Executed Plan

> 2023-01-30 15:42 +01:00 (Local Time) 1D As Collected 2023-01-31 15:42

Download PLV File

Operator	Subtree Cost	Individual ...	Exclusive ...	Rows	Object Sch...	Object Na...	Table Type	Execution ...	User CPU ...	Kernel CP...
HEX Search	100.000 %	0.000 %	408 µs	0					0.000 ms	0.000 ms
HEX Search	100.000 %	100.000 %	408 µs	1				HEX	0.244 ms	0.000 ms
Unique Index Lookup "SAPAB...	0.000 %	0.000 %	0 µs	0	SAPABAP	REPOLOAD	COLUMN ...		0.132 ms	0.000 ms
REPOLOAD.\$trexexternalke...	0.000 %	0.000 %	0 µs	0					0.132 ms	0.000 ms
UniqueIndexLookupOp	0.000 %	0.000 %	0 µs	0					0.132 ms	0.000 ms
Project	0.000 %	0.000 %	0 µs	0					0.112 ms	0.000 ms
ProjectBufferOp	0.000 %	0.000 %	0 µs	0					0.106 ms	0.000 ms
ProjectFetchOp	0.000 %	0.000 %	0 µs	0					0.006 ms	0.000 ms

> Operator Details
> Table Details
> Index Details

Abbildung 114 Download der *.plv Datei

Eclipse startet dann automatisch die richtige View, und im Tab-Executed-Plan ist die visuelle Analyse des Query-Plans zu sehen.

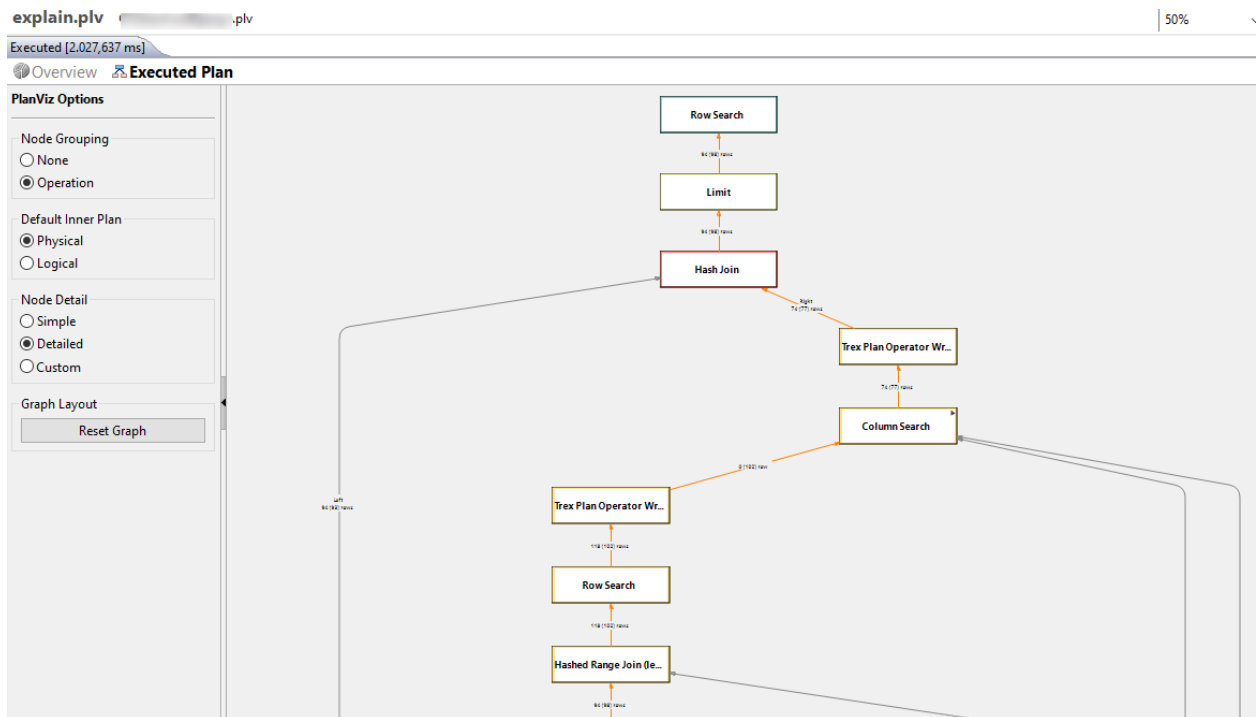


Abbildung 115 Anzeige des Abfrage-Ausführungsplans

Für die Entwicklung mit dem RAP (ABAP RESTful Application Programming Model) gibt es ein spezielles Trace-Werkzeug namens Cross Trace (Doku [On-Premise/Cloud](#)), mit dem Sie Anfragen von Fiori-Apps über den RAP Softwarestack (SAP Gateway, BO Behavior, SADL, ABAP Core) hinweg analysieren können. Zur Benutzung ist eine entsprechende gesonderte Berechtigung nötig.

Zum Starten lassen Sie sich die View "ABAP Cross Trace" anzeigen. Diese View hat zwei Tabs. Im ersten Tab können Sie im Kontextmenü eines Systems eine neue Cross-Trace-Konfiguration erstellen. Eine Cross-Trace-Konfiguration kann mit einer Beschreibung zur Unterscheidung versehen werden, kann aktiv oder inaktiv sein, eine automatische Deaktivierung nach einer bestimmten Anzahl von Trace-Requests ist möglich, ebenso eine automatisierte Löschung zu einem bestimmten Zeitpunkt. Sie können entscheiden, ob nicht-sensitive oder sensitive Daten aufgezeichnet werden sollen. Sie können optional nach Benutzer, Zugriffsart und -ziel filtern (z. B. nur ein bestimmter OData-Service; hier ist auch * für eine Wildcard-Filterung möglich), ebenso können Sie angeben, ob und mit welchem Trace-Level zu den jeweiligen Cross-Trace-Komponenten eine Aufzeichnung stattfinden soll.

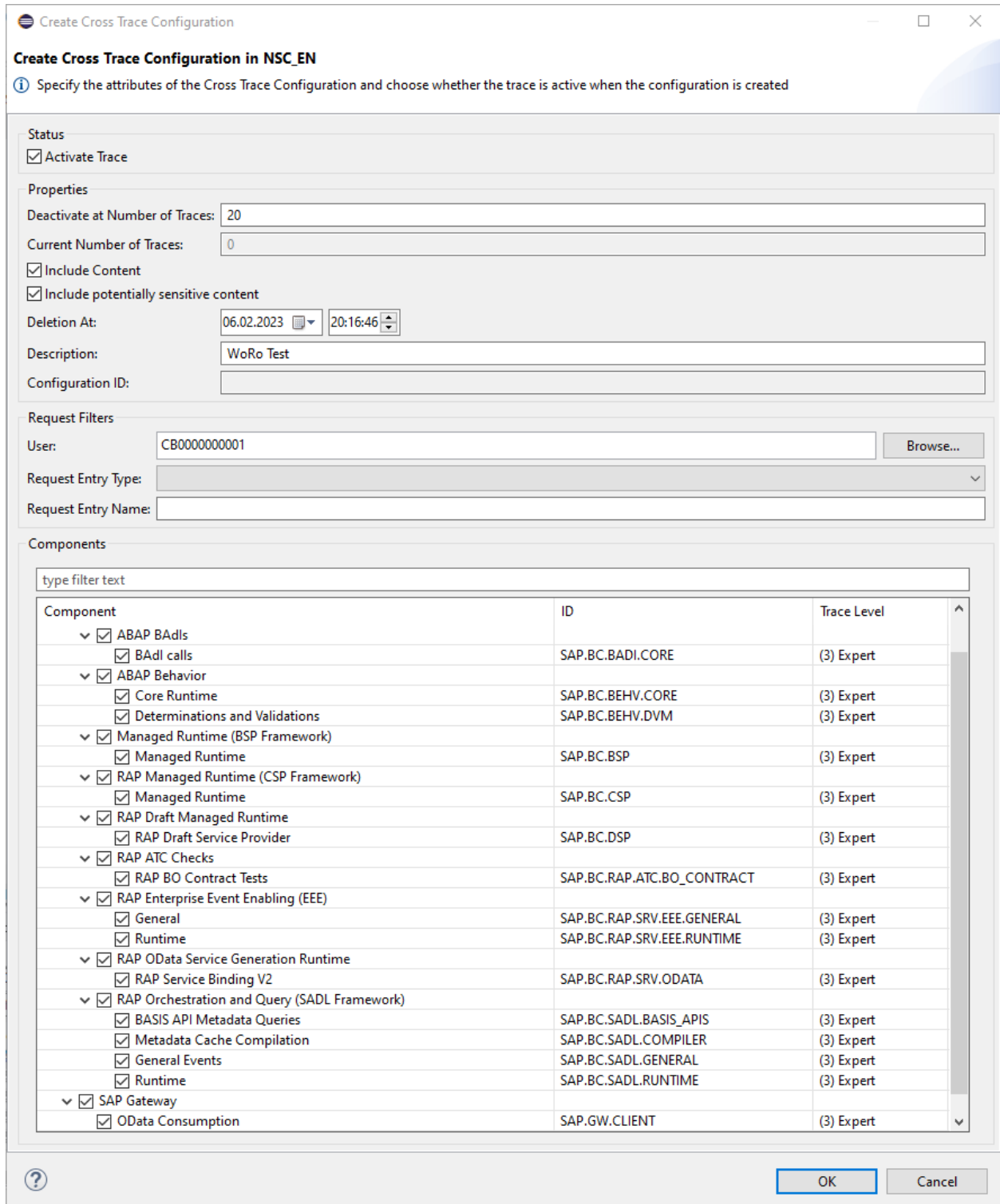


Abbildung 116 Erstellung von ABAP Cross Traces

Nach Bestätigung mit OK wird die Konfiguration im View angezeigt, hier kann auch der aktuelle Zustand (aktiv/inaktiv), die Beschreibung und die Anzahl der verbliebenen aufzuzeichnenden Zugriffe eingesehen werden. Im Kontextmenü kann eine Konfiguration editiert sowie aktiviert/deaktiviert/gelöscht/aktualisiert werden. Eine

globale Aktualisierung ist rechts oben im View möglich. Zur Ergebnisanzeige wechseln Sie auf den zweiten Tab der View. Hier sehen Sie je einzelner Zugriff einen Trace. Im Kontextmenü können Sie einen Trace öffnen (und löschen usw.). Der Trace öffnet sich dann in einer weiteren View. Dort sehen Sie den Weg des Zugriffs durch die einzelnen Komponenten hindurch und können die angezeigten Zeilen filtern bzw. eine Textsuche starten. Anhand der Offset-Spalte können Sie den zeitlichen Verlauf des Zugriffs nachvollziehen. Zu jeder Zeile werden in der Properties View Details angezeigt.

[NCS] /SAP/OPU/ODATA4/NEO/SKD01_CONFIG_O4_SRV/SRVD | 06.02.23, 18:59:01 | CB0000000001 | 02595F0682DE1EDDA9C8FDF090DBAE5C

⚙ No extended filters active. Maximum trace level: (3) Expert

Procedure	Processed Objects	Message	Record Properties	Content Size (...	Component
OData Provisioning request		POST Request received	URI:/sap/opu/odata4/neo/skd01	9943	SAP.GW.SERVER
IF_HTTP_LOCATION_EXC_PLUGIN--GET_LOCA	CL_HTTP_LOCATION_EXC_DEFAULT	BAdI call		157	SAP.BC.BADI.CORE
IF_HTTP_LOCATION_EXC_PLUGIN--GET_LOCA				74	SAP.BC.BADI.CORE
Get MDP For Exposure		ID = 'SRVD#/NEO/SKD_CONFIG_RESTYPEGRP'		46	SAP.BC.SADL.RUNTIME
Get MDP For Exposure					SAP.BC.SADL.RUNTIME
Initialize Application	/NEO/SKD01_CONFIG_O4_SRV (Servi	Service Group /NEO/SKD01_CONFIG_O4_SRV			SAP.GW.SERVER
Initialize W4 DPC		Initialize		18	SAP.BC.SADL.RUNTIME
Initialize W4 DPC					SAP.BC.SADL.RUNTIME
OData Entity Set Query		Querying set from '/NEO/C_SKD_RESOURCEYPEGRPAL'		1168	SAP.BC.SADL.RUNTIME
Read Context Flags		IS_STICKY_SESSION: false		24	SAP.BC.SADL.RUNTIME
ABQJ Construction		ENTITY: 'SRVD#/NEO/SKD_CONFIG_RESTYPEGRP~/NEO/C_SKD_RESOURCEYPEGR'		75	SAP.BC.SADL.RUNTIME
Query Entity Collection		(see content for query details)		3312	SAP.BC.SADL.RUNTIME
SADL Query		Querying '/NEO/C_SKD_RESOURCEYPEGRPAL', 8 elements requested, with paging		769	SAP.BC.SADL.RUNTIME
SADL Query		Query retrieved 1 row(s)		702	SAP.BC.SADL.RUNTIME
GET_LOAD		ENTITY: '/NEO/C_SKD_RESOURCEYPEGRPAL'		42	SAP.BC.SADL.RUNTIME
GET_LOAD		Result code: OK		986	SAP.BC.SADL.RUNTIME
ABQJ Construction		ENTITY: 'EXP#CDS#/NEO/C_SKD_RESOURCEYPEGRPAL~/NEO/C_SKD_RESOURCE'		81	SAP.BC.SADL.RUNTIME
GET_PROPERTIES		ENTITY: '/NEO/C_SKD_RESOURCEYPEGRPAL, KEYS: 1		972	SAP.BC.SADL.RUNTIME
GET_LOAD		ENTITY: '/NEO/C_SKD_RESOURCEYPEGRPAL'		42	SAP.BC.SADL.RUNTIME
GET_LOAD		FEATURES: [TABLE<3x(7)>]		582	SAP.BC.SADL.RUNTIME
GET_LOAD		ENTITY: '/NEO/C_SKD_RESOURCEYPEGRPAL'		42	SAP.BC.SADL.RUNTIME
GET_LOAD		ASSOCIATIONS: [TABLE<1x(12)>]		339	SAP.BC.SADL.RUNTIME
GET_STATIC_FEATURES		Request for: '/NEO/C_SKD_RESOURCEYPEGRPAL (Root: '/NEO/C_SKD_RESOURCE'	ENTITY: '/NEO/C_SKD_RESOURCE'	92	SAP.BC.BEHV.CORE
GET_STATIC_FEATURES		Global Result for: '/NEO/C_SKD_RESOURCEYPEGRPAL (Root: '/NEO/C_SKD_RESOURCE'	ENTITY: '/NEO/C_SKD_RESOURCE'	92	SAP.BC.BEHV.CORE
GET_STATIC_FEATURES		Instance Result for: '/NEO/C_SKD_RESOURCEYPEGRPAL (Root: '/NEO/C_SKD_RESOURCE'	ENTITY: '/NEO/C_SKD_RESOURCE'	92	SAP.BC.BEHV.CORE
CALL_GLOBAL_AUTHORIZATION		Job Triggers: (1)	JOB:(o:2228) GAJOB.LOG:(o:2273)	76	SAP.BC.BEHV.CORE
CALL_HANDLER(Global Authorization)	/NEO/C_SKD_RESOURCEYPEGRPAL	Call (Global Authorization) (o:2218) HANDLER_PROJ	ROOT:/NEO/C_SKD_RESOURCE'	150	SAP.BC.BEHV.CORE
CALL_HANDLER(Global Authorization)					SAP.BC.BEHV.CORE
CALL_INSTANCE_CONTROLLERS		Job Triggers: (1)	JOB:(o:2182) FJOB.CONTENT:COF	76	SAP.BC.BEHV.CORE
CALL_HANDLER(Instance Features)	/NEO/C_SKD_RESOURCEYPEGRPAL	Call (Instance Features) (o:2145) HANDLER_PROJ	ROOT:/NEO/C_SKD_RESOURCE'	277	SAP.BC.BEHV.CORE
CALL_HANDLER(Instance Features)					SAP.BC.BEHV.CORE
GET_PROPERTIES				3008	SAP.BC.SADL.RUNTIME

Record 1

General	Content	Call Stack	Record Properties	Trace Header
61 ~path_translated_expanded	/sap/opu/odata4/neo/skd01_config_o4_srv/srvt/neo/skd_config_restypegrp/0001/\$batch			
62 ~gui_header_handler_field	/sap/opu/odata4			
63 HTTP BODY	--batch_id-1675706340487-950 content-type:application/http content-transfer-encoding:binary			
	GET ResourceGrpAllSet?sap-client=100&filter=ResourceTypeGroupAllId%20eq%201%20and%20(IsActiveEntity%20eq%20false%20or%20SiblingEntity/I			Accept:application/json;odata.metadata=minimal;IEEE754Compatible=true

Abbildung 117 Detaillierte Ansicht der Operationen

Sie können direkt zur auslösenden Quellcodezeile springen, sich die Aufrufhierarchie anzeigen lassen usw.

4.4.1 Hinweis zu HANA Studio und SAP HANA Tools

HANA Studio wird von SAP nur begrenzt weiterentwickelt. Eine zukünftige Lösung für die visuelle Analyse der Query-Pläne innerhalb der ADT oder für die ABAP Cloud steht noch aus. Als eine Lösung außerhalb von Eclipse gibt es ein Visual Studio Code Plug-

in, das ebenfalls *.plv-Dateien öffnen und grafisch anzeigen kann, vgl. [SQL Analyzer Extension](#).

Die SAP HANA Tools folgen einem anderen Release-Zyklus als die ADT und sind daher häufig nicht in der aktuellen bzw. der "latest" Update-Site enthalten. Beispielsweise waren die SAP HANA Tools im Januar 2023 in der aktuellsten Version nur auf der <https://tools.hana.ondemand.com/2022-09> Site verfügbar.

4.5 Feed Reader

Feeds ermöglichen ereignisbezogene Benachrichtigungen in den ADT inklusive Zugriff auf eine Liste der bisherigen Ereignisse. Die Feeds werden in einer eigenen View (vgl. [Views und Perspektiven](#)) dargestellt. Diese finden Sie in der Liste der Views unterhalb von ABAP mit dem Titel "Feed Reader". Die einzelnen Quellen werden als Feeds bezeichnet und für jeden Feed kann eingestellt werden, wie oft dieser von der Quelle aktualisiert werden soll und ob Benachrichtigungsmeldungen in Eclipse angezeigt werden sollen.

Neben den in der Doku unter "Getting Feeds" ([On-Premise/Cloud](#)) beschriebenen ABAP-Runtime-Errors/-Dumps und Systemnachrichten können insbesondere auch (abhängig vom Release-Stand des Quellsystems) folgende Informationen im Feed Reader angezeigt werden:

- Gateway/OData-Fehler (vielfältig filterbar, einschl. Benutzer)
- ATC-Ergebnisse (vielfältig filterbar, einschl. Benutzer)
- Enterprise-Event-Fehler (filterbar nach Kanal und Benutzer)
- BW Job Repository (vielfältig filterbar, einschl. Benutzer)
- URI Creation Error (nur für die ADT-Entwickler relevant)

Dazu werden die Ereignisse im Pull-Verfahren im Hintergrund abgefragt. Damit die Hintergrundabfrage aus SAP-Systemen funktioniert, müssen Sie allerdings nach einem Eclipse-Start mindestens einmal in irgendeiner Form (das können Sie auch durch Klick auf einen Feed auslösen) auf die gewünschten Systeme zugegriffen und dabei die Anmeldeprozedur durchlaufen haben.

Daneben können auch beliebige Atom-/RSS-Feeds abonniert werden, das kann beispielsweise der RSS-Feed für die letzten Blog-Posts zu einem Tag auf blogs.sap.com sein:

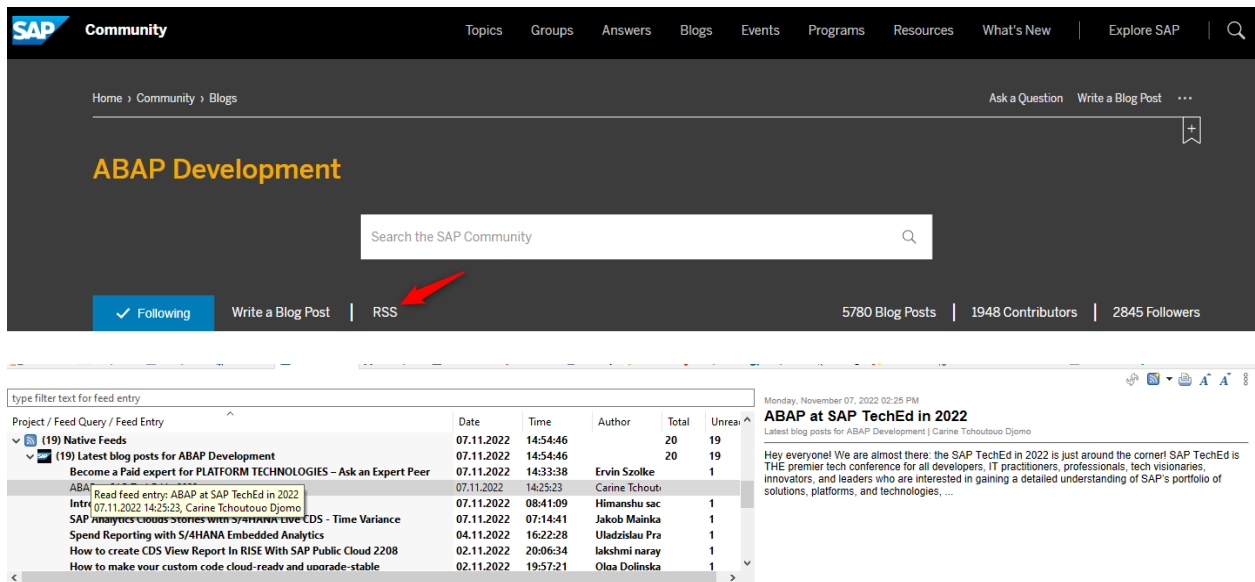


Abbildung 118 Abonnieren populärer RSS Feeds

Mit einem Links-Klick auf den Titel in der Detailanzeige bzw. Rechts-Klick in der Liste und den Kontextmenüpunkt Open können Sie den Blog-Artikel öffnen (dafür ist es sinnvoll, in Eclipse einen externen Browser einzustellen, weil der Eclipse-interne Browser sich als IE11 für die Webseite ausgibt).

Der Feed zu den Systemnachrichten zeigt die aktuell vorliegenden Nachrichten der Systemadministratoren an.

Sie können Feeds zu Laufzeitfehlern (ST22-Dumps) mit verschiedenen Filtermöglichkeiten ergänzen, u.a. kann gemäß auslösendem Benutzer, verantwortlichem Benutzer, Objekt- und Paket-Benutzer oder Paket gefiltert werden. Die Filter können auch mit und/oder-

(all/any-) Verknüpfungen zu einem hierarchischen Filter-Baum angeordnet werden. Sie können einen Laufzeitfehler in einer eigenen Editor-Ansicht öffnen (Kontextmenü in der Liste, Link in den Details; dort ist auch das bekannte Langtext Format bzw. eine unformatierte Anzeige verfügbar). Sie können direkt in den ADT-Quelltexteditor zu den Quellcodezeilen der auslösenden Stelle und der Aufrufhierarchie navigieren.

Ein Beispiel mit mehreren Laufzeitfehlern in einem Feed:

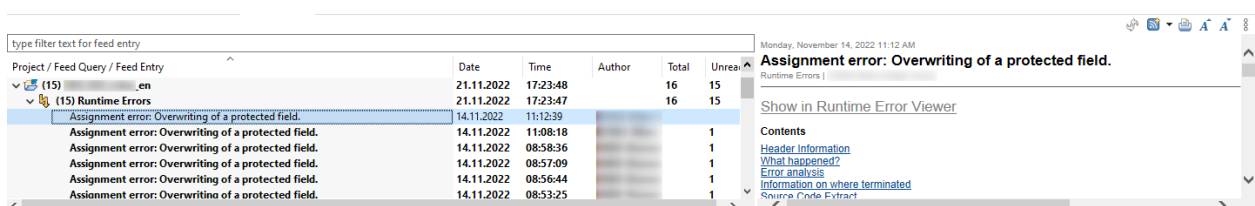


Abbildung 119 Mehrere Laufzeitfehler innerhalb eines Feeds

Dokumentation zu den SAP Gateway Error Log Feeds findet sich im PDF-Dokument aus dem Hinweis [1797736 - SAP Gateway Troubleshooting Guide](#) und im Blog [How to use the SAP Gateway Error Log in ADT](#).

Sie können Feeds mit verschiedenen Filtermöglichkeiten ergänzen, u.a. kann auch hier gemäß Benutzer, Service, Namensraum oder Paket gefiltert werden. Um auch mit vielen Einträgen zurechtzukommen, kann ein Blättern (Paging) aktiviert werden. In der Detailanzeige kann direkt in die Transaktion /IWFND/GW_CLIENT zum Replay gesprungen werden. Außerdem kann man direkt in den ADT-Quelltexteditor analog zu den Laufzeitfehlern navigieren, vgl. folgendes Beispiel:

Project / Feed Query / Feed Entry	Date	Time	Author	Total	Unrea
(4) ..._en	21.11.2022	17:51:20		5	4
Gateway error log entries for my user (...)	21.11.2022	17:50:59		0	0
Runtime Errors caused by me (...)	21.11.2022	17:50:58		0	0
Runtime Errors for objects I am responsible for (...)	21.11.2022	17:50:59		0	0
(4) SAP Gateway Error Log (all)	21.11.2022	17:51:20		5	4
Frontend Error: VokabAnnotatDatei 'UI_PREPAREFORBILLG_VAN', Vers. '0001',	21.11.2022	17:34:34		1	
Frontend Error: Kein Service für Namensraum '', Name 'UI_PREPAREFORBILLG', Versi	21.11.2022	17:34:33		1	
Frontend Error: Kein Service für Namensraum '', Name 'UI_PREPAREFORBILLG'	21.11.2022	17:34:33		1	
Frontend Error: Kein Service für Namensraum '', Name 'UI_PERFORMREPAIRS',	21.11.2022	17:34:31		1	
Frontend Error: Kein Service für Namensraum '', Name 'UI_PROCREPAIRQTANS	21.11.2022	17:34:29		1	
System Messages	21.11.2022	17:50:58		0	0

Frontend Error: Kein Service für Namensraum '', Name 'UI_PREPAREFORBILLG', Version '0001' gefunden

SAP Gateway Error Log (all)

Content

Header Information
 Service Information
 Error Content
 Source Code Extract
 Active Calls/Events

Header Information

Short Text Frontend Error: Kein Service für Namensraum '', Name 'UI_PREPAREFORBILLG', Version '0001' gefunden

Transaction ID AA7E06A76199442C852DEC58F069688B2 (Replay in GW_Client)

Abbildung 120 Ansicht eines SAP Gateway Fehlers aus dem Error Log

Im Kontext von Enterprise Events können Sie Feeds zu Fehlern aus der Event Verarbeitung hinzufügen, z. B. um Fehler beim Weiterreichen der Events an den Event Mesh zu sehen. Dazu können Sie Kanal und Benutzer filtern.

Sie können Feeds zu den ATC-Ergebnissen aus Prüfungen in einem zentralen ATC-Check-System anlegen, vgl. Doku ([On-Premise/Cloud](#)). Auch hier sind diverse Filtermöglichkeiten verfügbar. In der Feed-Liste können Sie zu den Details und zur Quellcodezeile eines Ergebnisses navigieren.

Sofern Sie BW/S4HANA-Systeme haben, können Sie sich mit dem BW Job Repository Feed über den Status verschiedener Job-Typen (z. B. DATAFLOWCOPY oder DTP_LOAD) informieren lassen, vgl. [Doku](#). Von den Feed-Einträgen können Sie dann zur Anzeige der Job-Details verzweigen.

Da der Feed Reader das ideale Tool für ein proaktives Monitoring von Anwendungen ist (z. B. bei und nach einem Go-live einer neuen Anwendung), ist es sinnvoll, die Berechtigung für die ADT nicht nur für die Entwicklungssysteme zu betrachten, sondern den Entwicklern mittels Berechtigungen zu ermöglichen, die Feeds auch von Test- und Produktivsystemen einzusammeln.

4.6 Doku-Links

ABAP-Debugger-Konzept:

<https://help.sap.com/docs/BTP/5371047f1273405bb46725a417f95433/4ec365a66e391014adc9ffe4e204223.html>

Troubleshooting-Tools:

<https://help.sap.com/docs/BTP/5371047f1273405bb46725a417f95433/4ecc7d3a6e391014adc9ffe4e204223.html>

Syntax für Breakpoint-Bedingungen:

<https://help.sap.com/docs/BTP/5371047f1273405bb46725a417f95433/d878e676fe904eba9f4bb79193154092.html>

5 Installation, Verteilungs- und Update-Strategien

Die ADT basieren auf dem quelloffenen und vor allem in anderen Programmiersprachen sehr bekannten Eclipse Framework. Für die Installation und Verteilung hat das den Vorteil, dass das Rad nicht neu erfunden werden muss. Je nach Unternehmensgröße und Heterogenität der IT-Landschaft eines Unternehmens ist es sogar möglich, dass Eclipse-basierte Entwicklungsumgebungen schon im Unternehmen eingesetzt und verteilt werden. In diesem Fall empfiehlt es sich, die bestehende Infrastruktur weiter zu verwenden. Auch kann dies etwaige Aufwände der ein oder anderen Lösung reduzieren.

5.1 Abgrenzungen

5.1.1 Installation Guide von SAP

SAP veröffentlicht einen eigenen Installationsleitfaden ([Link](#)). Dieser führt neben dem hier ebenfalls aufgeführten manuellen Installationsweg auch weitere Varianten für abgeriegelte Umgebungen mit sehr begrenztem Internet-Zugriff auf. Da es sich hierbei um Lösungen für spezielle Situationen handelt, wird auf diese in den folgenden Abschnitten nicht mehr eingegangen.

5.1.1.1 Andere Werkzeuge mit identischem Installationsweg

SAP veröffentlicht auf Basis von Eclipse noch weitere Werkzeuge, wie beispielsweise die BW Tools (verpflichtend ab BW/4HANA 1.0) und HANA Development Tools. Da sich dieser Leitfaden aber auf die ABAP-Entwicklung mit ADT konzentriert, wird nicht weiter auf die genannten anderen Werkzeuge eingegangen. Die Erkenntnisse sind jedoch größtenteils übertragbar.

5.2 Vorbereitungen

Für jede ADT-Installation (teilweise auch Eclipse ohne ADT) gibt es gewisse Voraussetzungen. Diese sind unabhängig von der Installations-/Verteilungsstrategie und werden detailliert im o. g. ADT Installation Guide beschrieben.

5.2.1 Java Development Kit und Java Runtime Environment

Wird mit der Installation lediglich ABAP entwickelt (und insbesondere kein Java, d. h. auch keine Plug-in-Entwicklung zur Ergänzung der ADT, siehe [Kapitel 7 Plug-Ins](#)), so ist auch kein Java Development Kit (JDK) notwendig. Ein Java Runtime Environment (JRE) reicht aus. Wird der offizielle Installer von Eclipse als Basis verwendet (siehe [Eclipse Installer](#)), kommt dies automatisch mit.

Zu beachten ist dabei, dass Eclipse mittlerweile standardmäßig als 64-Bit-Anwendung installiert wird (oft auch als *x64* oder *x86_64* bezeichnet). Da das JRE dieselbe Architektur verwenden muss wie die Eclipse-Installation, wird auch eine 64-Bit-JRE installiert.

5.2.2 Backend

Neben der lokalen ADT-Installation muss im Falle der Entwicklung auf einem On-Premise-System auch selbiges für die Verbindung mit ADT vorbereitet werden. Diese Schritte werden ebenfalls im offiziellen [SAP ADT Configuration Guide](#) beschrieben. Zwei essentielle Schritte werden darin jedoch gerne übersehen, weshalb hier noch einmal explizit darauf hingewiesen werden soll.

5.2.2.1 Web-Services

Für die Kommunikation nutzen die ADT spezielle Web-Services auf dem Backend, die mit den unterstützten Basis-Releases ausgeliefert werden. Die Definition der Web-Services und die darin eingetragenen Handler-Klassen sind ein Grund, warum der Funktionsumfang der ADT zwischen den Basis-Releases unterschiedlich ist. Diese Web-Services sind standardmäßig inaktiv und müssen zunächst aktiviert werden. Die aktuelle Liste kann dem oben verlinkten Configuration Guide von SAP entnommen werden.

Zum Zeitpunkt dieses Dokuments sind dies die folgenden Services:

- ABAP Docu (notwendig)
 - o *default_host* → *sap* → *public* → *bc* → *abap* → *docu*
 - o *default_host* → *sap* → *bc* → *abap* → *docu*
- Fehlertexte und Element Info (notwendig)
 - o *default_host* → *sap* → *public* → *bc* → *abap* → *toolsdocu*
 - o *default_host* → *sap* → *bc* → *abap* → *toolsdocu*
- Teilen von HTTP-Links (optional)
 - o *default_host* → *sap* → *bc* → *adt*
- Web Dynpro (nur notwendig für WD-Entwickler)

- `default_host` → `sap` → `bc` → `webdynpro` → `sap` → `wdy_aie_vd_preview`

5.2.2.2 Berechtigungen

Um die genannten Web-Services nutzen zu dürfen, müssen die User berechtigt werden. Zusätzlich werden noch RFC-Bausteine und Transaktions-Codes benötigt.

SAP liefert hierzu zwei Rollen als Vorlage aus:

1. `SAP_BC_DWB_ABAPDEVELOPER` → Entwicklerrolle mit allen Features
2. `SAP_BC_DWB_WBDISPLAY` → Anzeigeberechtigungen für alle Features

Details bezüglich der darin verbauten Berechtigungen und deren Zweck können dem [SAP ADT Configuration Guide](#) entnommen werden.

5.2.3 SAP-GUI-Installation

Soll auf einem Backend-System ein SAP-GUI-Transaktionsstart möglich sein (nicht verfügbar in SAP Public Cloud und SAP BTP ABAP Environment/Steampunk), so wird hierfür eine lokale SAP-GUI-Installation benötigt. Die ADT liefern diese nicht mit.

5.2.4 Visual Studio Redistributable

Unter Windows wird zudem das Visual-Studio-2013 (VC++ 12.0)-Redistributable-Paket in exakt dieser Version benötigt. Oftmals ist diese Abhängigkeit schon aufgrund anderer bereits auf dem Zielgerät installierter Software vorhanden.

5.3 Technischer Aufbau einer Eclipse-Installation

Wie bereits vorhergegangenen Kapiteln erwähnt, besteht eine Eclipse-Installation aus den folgenden Komponenten:

- Installationsordner (reine Software-Pakete und `eclipse.exe`)
- User-Settings oder Configuration-Area (Plug-ins, Teile der Konfiguration)
- Workspace (benutzerspezifischer Teil: Einstellungen, Ansichten, Systemverbindungen etc.)

Die Installation von Eclipse legt den Installationsordner und User-Settings an. Mittels fortgeschrittener Techniken können auch schon Teile des Workspace vorbelegt werden.

Aufgrund diverser Probleme empfiehlt es sich, alle Komponenten in einen mit normalen Benutzerrechten beschreibbaren Ordner abzulegen (also **nicht** C:\Program Files\).

Beispielsweise könnte eine Verzeichnisstruktur für die Eclipse-Installation wie folgt aussehen:

C:\ADT\	Gesamtverzeichnis für ADT
C:\ADT\IDE	Verzeichnis für Ablage der verschiedenen Eclipse-Versionen
C:\ADT\IDE\2022-12	entpackte Dateien der Eclipse-Version 2022-12
C:\ADT\IDE\2023-03	entpackte Dateien der Eclipse-Version 2023-03
C:\ADT\WS	Verzeichnis zur Ablage der verschiedenen Workspaces
C:\ADT\WS\2022-12	Verzeichnis für Workspaces der Version 2022-12
...	

Diesem Beispiel liegt die Annahme zu Grunde, dass die Eclipse-Installation über das Entpacken der zip-Files erfolgt und die Versionen parallel verwendet werden sollen.

Da die Workspaces mit aufsteigenden Versionen in die jeweils neue Version konvertiert werden, sollten die Workspaces pro Version kopiert werden, um bei Bedarf die älteren Versionen weiterhin verwenden zu können.

Inwiefern Workspaces sinnvoll eingesetzt werden, wird in [Kapitel 3 "Arbeiten mit Eclipse"](#) detaillierter erläutert.

5.4 Plug-ins

Plug-ins wie die ADT können über die Angabe der Update-Site im Dialog Help → Install New Software zu einer bestehenden Eclipse-Plattform hinzugefügt werden. Einfacher ist jedoch die Verwendung des Eclipse Marketplace, sofern das Plug-in dort gelistet ist. Die ADT sind hier Stand 2022 leider nicht gelistet, so dass für die ADT der Weg über die Update-Site notwendig ist.

5.4.1 Eclipse Marketplace

Der Eclipse Marketplace ist im Help-Menü versteckt.

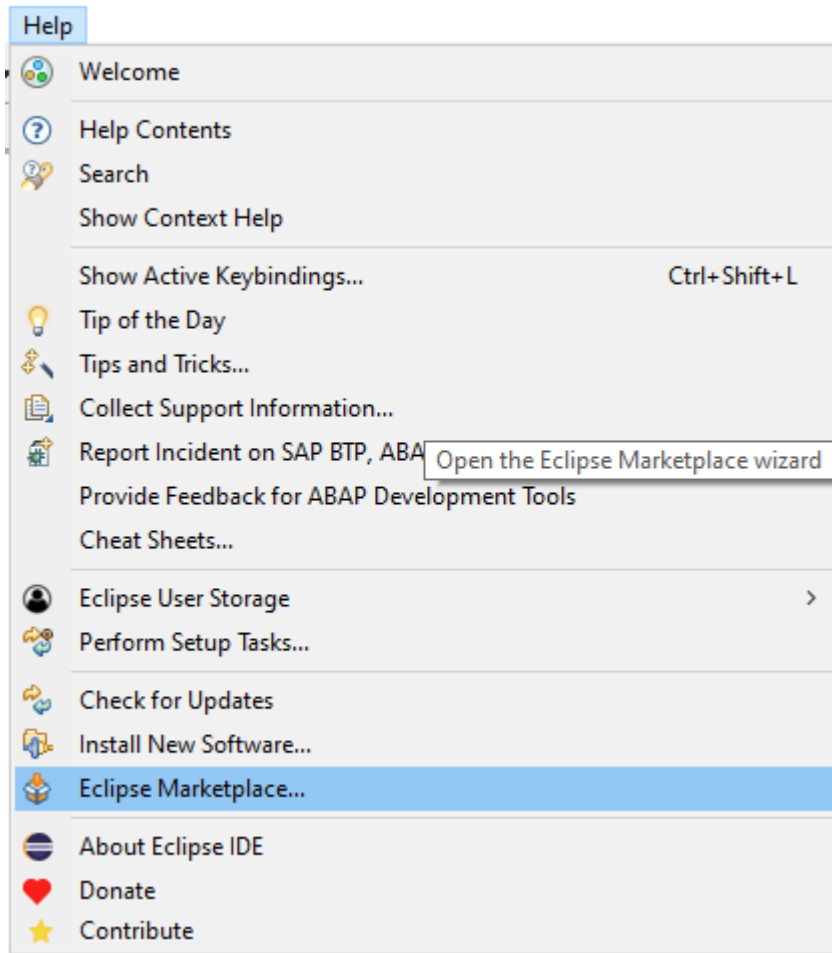


Abbildung 121 Einstieg in den Eclipse Marketplace

Hier kann dann nach Plug-ins gesucht werden. Das Suchwort *ABAP* lieferte Stand 2022 in diesem Beispiel 11 Treffer.

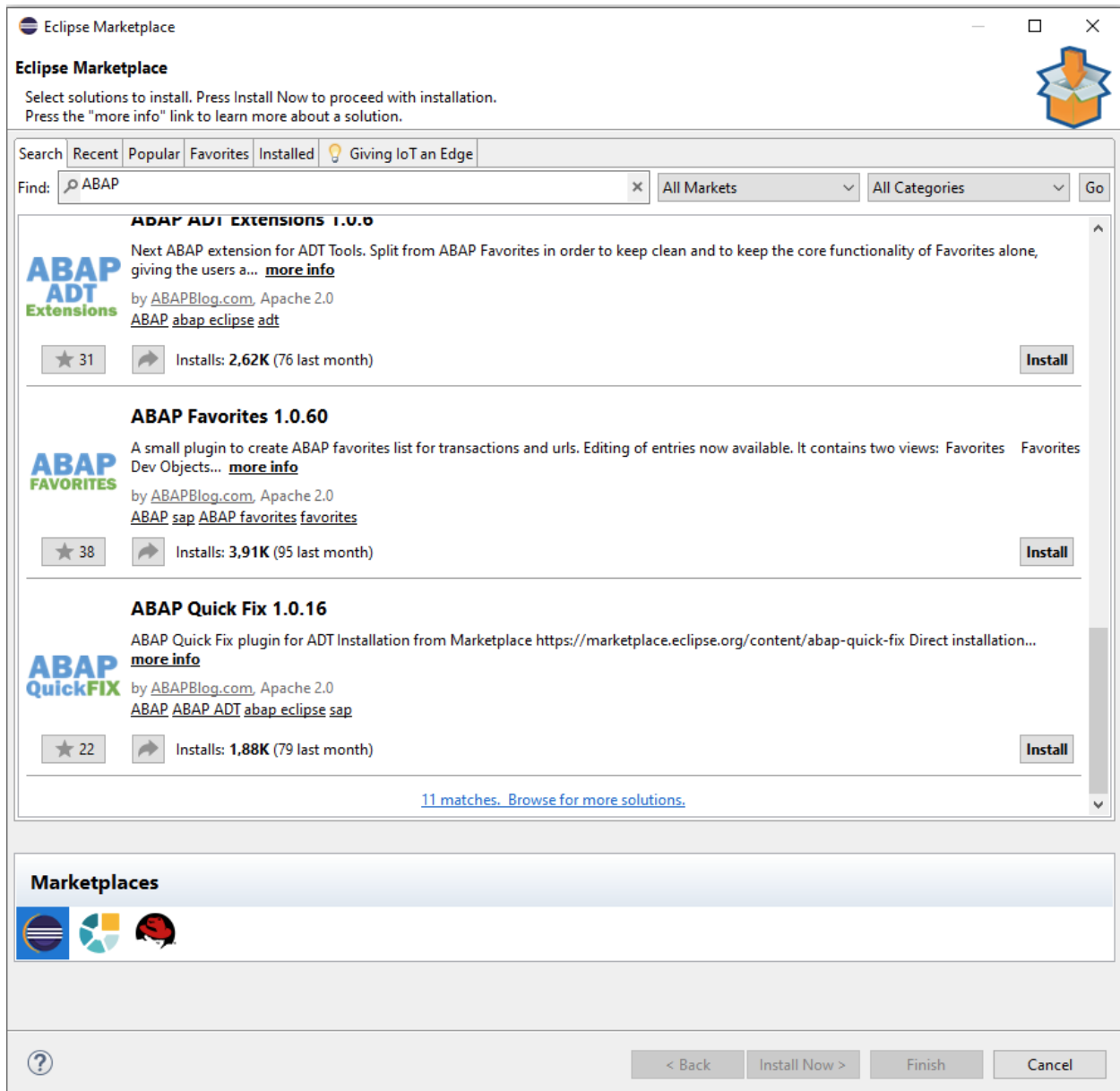


Abbildung 122 Exemplarische Suche nach Plug-ins im Eclipse Marketplace

Jedes Plug-in besitzt auf der rechten Seite einen eigenen *Install*-Button. Gegebenenfalls müssen noch eine Lizenz bestätigt und Zertifikaten vertraut werden. Abschließend ist ein Neustart von Eclipse notwendig.

5.4.2 Update-Site

Kennt man die Update-Site eines Plug-ins oder ist es nicht auf dem Eclipse Marketplace gelistet, kann auch der klassische Installationsweg verwendet werden.

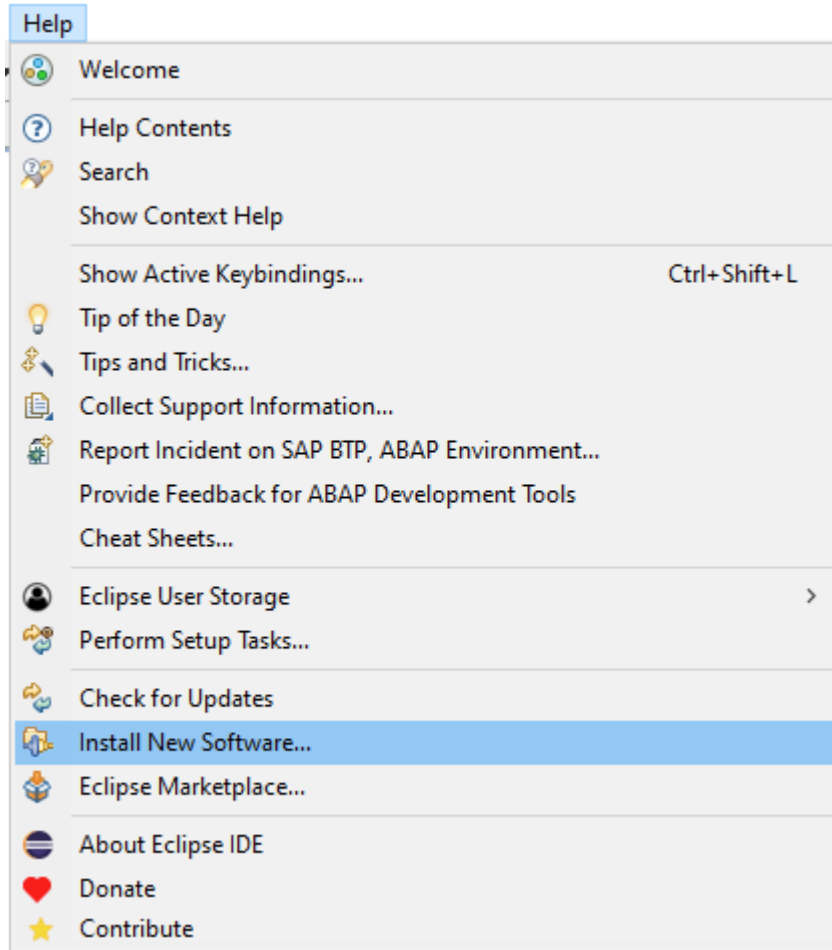


Abbildung 123 Installation neuer Software über das Kontextmenü

Im Feld *Work with* wird die Update-Site eingetragen. Ist diese gültig, werden unten die dort verfügbaren Plug-ins angezeigt. Neben Webseiten kann auch eine heruntergeladene Version des Plug-ins in einer zip-Datei eine Update-Site sein. In letzterem Fall müssen Updates jedoch händisch mit einem weiteren Download durchgeführt werden.

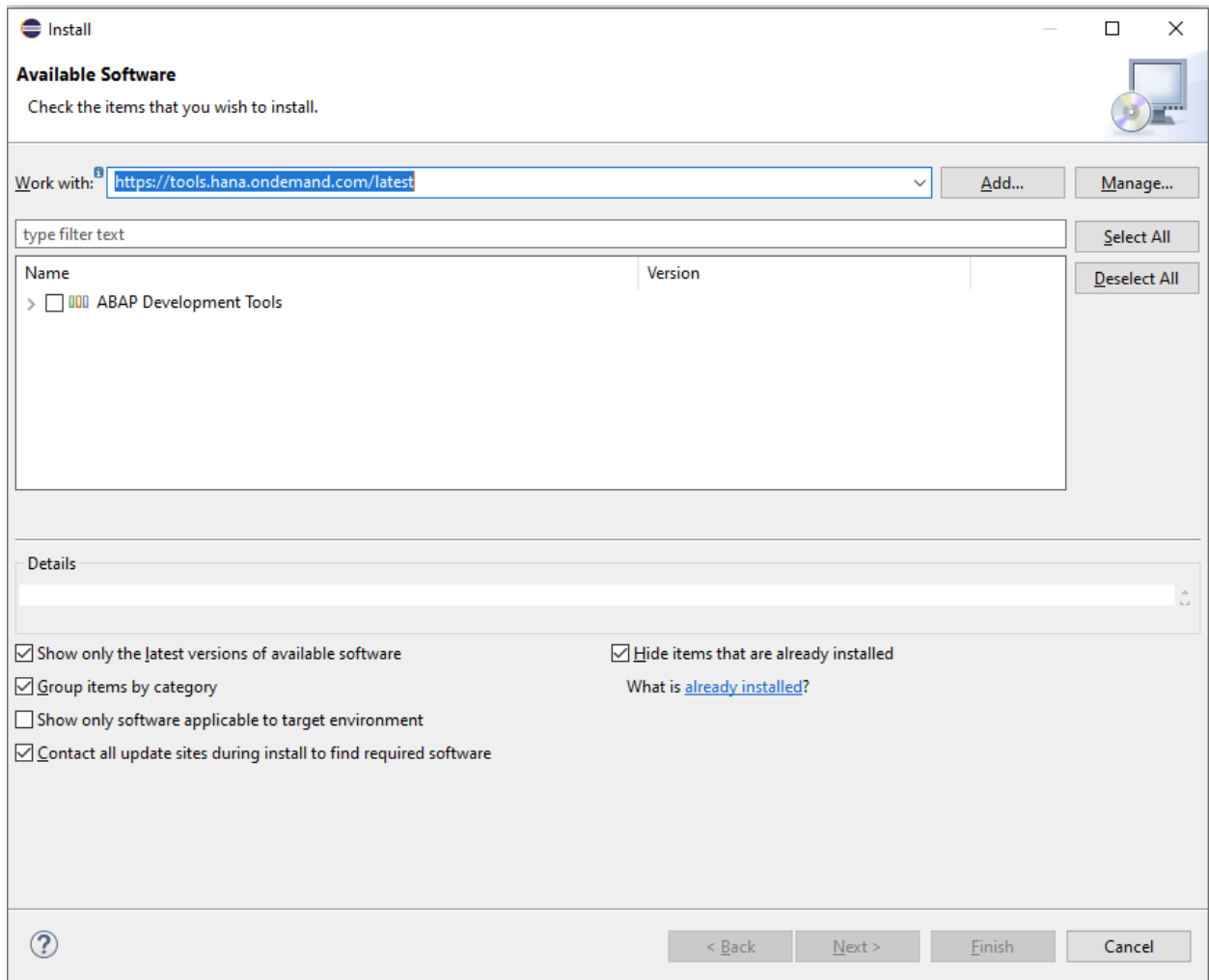


Abbildung 124 Eintragen der Update-Site

Auch hier müssen gegebenenfalls Lizenzen akzeptiert und Zertifikaten vertraut werden. Nach einem Neustart steht das Plug-in zur Verfügung.

5.5 Installations- und Verteilungsstrategien

5.5.1 Übersicht und Vergleich

Da die ADT erst vergleichsweise spät dem Eclipse-Ökosystem hinzugefügt wurden, haben sich außerhalb des SAP-Umfelds bereits diverse Installations-Strategien für Eclipse-basierte Entwicklungsumgebungen etabliert. Die verbreitetsten sind:

1. Komplette manuelle Einrichtung
2. Vorkonfigurierte Erstinstallation
3. Eclipse Installer (Oomph)

In den folgenden Kapiteln wird auf diese Mechanismen im Detail eingegangen. Neben diesen existieren noch viele weitere Möglichkeiten, die entweder für eher ungewöhnliche Situationen (z. B. fehlender Internetzugriff) konzipiert wurden oder mittlerweile durch komfortable Varianten verdrängt wurden.

Zunächst sollen die näher betrachteten Varianten jedoch einmal in einer Übersicht verglichen werden. Die dabei verwendeten Symbole + (*gut*), ◦ (*mittel*) und - (*schlecht*) sind als Rangfolge zu verstehen und nicht als absolute Werte am Anfang, Mitte und Ende des jeweiligen Spektrums. Grund hierfür ist, dass beispielsweise ein Aufwand sehr individuell wahrgenommen wird. So kann ein mit - (*schlecht*) gekennzeichnete hoher Aufwand für eine Person trotzdem akzeptabel sein. Wohl jedoch stellen diese Symbole eine Reihenfolge dar, da messbare Unterschiede existieren.

Tabelle 1 Vergleich unterschiedlicher Installationsmöglichkeiten

Kriterium	Manuelle Installation	Vorkonfigurierte Erstinstallation	Eclipse Installer
Aufwand Anwender	-	◦	+
Notwendiges Wissen Anwender	-	◦	+
Aufwand zentrale Verwaltung	+ (keiner)	◦	-
Benötigte zentrale Infrastruktur	+ (keine)	◦	-
Aufwand Anwender bei Update/Upgrade	◦	◦	◦
Aufwand zentrale Verwaltung bei Update/Upgrade	+ (keiner)	-	◦
Automatische Verteilung Add-ons, Einstellungen	- (unmöglich)	◦ (einmalig)	+ (kontinuierlich)

Kriterium	Manuelle Installation	Vorkonfigurierte Erstinstallation	Eclipse Installer
Empfohlen für Unternehmensgröße	<i>Einzelpersonen/ kleine Unternehmen</i>	<i>Mittlere und große Unternehmen</i>	<i>Große Unternehmen</i>

5.5.2 Manuelle Installation

In dieser Variante wird eine Standard-Eclipse-Installation neu heruntergeladen und mittels weniger Klicks installiert. Da es keinen vorgefertigten Installer für ABAP gibt, werden die ADT einzeln nachinstalliert. Ohne Plug-ins, Einstellungen und Perspective-Anpassungen ist hier nach ungefähr 15 Minuten eine benutzbare Installation vorhanden.

SAP hat auf seiner Lernplattform eine bebilderte Anleitung zur Verfügung gestellt: <https://developers.sap.com/tutorials/abap-install-adt.html>.

Ist diese Installation abgeschlossen, müssen die gewünschten Einstellungen vorgenommen und die Systemverbindungen hinzugefügt werden. Sollten weitere Plug-ins zum Einsatz kommen (siehe Kapitel 7 "Plug-ins"), müssen auch diese nachinstalliert werden.

5.5.3 Vorkonfigurierte Erstinstallation

Eine vorkonfigurierte Erstinstallation ist im Prinzip eine manuelle Installation, deren Zustand direkt nach der Einrichtung gesichert wird (in der Regel gepackt als zip-Datei). Dieser Zustand kann dann über diverse Wege an andere Personen verteilt werden.

Bei macOS als Zielsystem gilt es zudem zu beachten, dass hier eine sogenannte App-Zip-Translocation greifen könnte. Diese erstellt von einer frisch *im selben Verzeichnis* entpackten Software wie Eclipse bei Ausführung eine Art "Schattenkopie". Das Ergebnis verhält sich wie fehlende Schreibrechte im Installationsordner. Sollten Updates eingespielt werden, kommt es zu Fehlern.

Je nach Einsatzzweck können mehr oder weniger der drei Komponenten der Installation (Installationsordner, User-Settings, Workspace) mit hineingepackt werden.

Nimmt man die User-Settings mit, so lassen sich beispielsweise die ADT und deren Update-Site für spätere Updates gleich mit verteilen. In Citrix-Umgebungen wäre dies hingegen hinderlich, da die User-Settings Schreibrechte benötigen. Nimmt man hierfür zwei getrennte Pakete, könnte auch der Installationsordner in einen schreibgeschützten, zentral provisionierten Teil untergebracht werden.

Den Workspace könnte man ebenfalls als getrenntes Template packen und verteilen. Somit können initiale Einstellungen einmalig verteilt werden. Da sich dieser Bereich allerdings sehr häufig ändert und bei einem Update ohnehin nicht ohne Verlust der aktuellen Einstellungen, Layouts und geöffneten Objekte verwendet werden kann, sollte dies nur mit Bedacht geschehen.

5.5.4 Eclipse Installer

Auf der Eclipse-Webseite kann ein komfortabler Installer mit sehr geringer Download-Größe heruntergeladen werden (auch *Oomph Installer* genannt). Bei dessen Start kann man dann eine Basiskonfiguration und ein paar Detailsinstellungen auswählen und daraufhin genau diese Installation erstellen lassen.

Hier finden sich allerdings keinerlei Konfigurationen inklusive der ADT wieder. Jedoch handelt es sich hierbei nicht um Magie, sondern die verfügbaren Einstellungen sind lediglich in einem bestimmten Format auf einem Server der Eclipse Foundation abgelegt. Der Pfad, wo der Installer nach Konfigurationen suchen soll, ist anpassbar. Somit ist es möglich, unternehmensspezifische Konfigurationen zu erstellen, die jeder nach Belieben installieren kann. Dieses Verfahren könnte in einigen Unternehmen bereits für andere Entwicklungssprachen im Einsatz sein, was den Wartungsaufwand deutlich reduziert.

Lediglich der Eclipse Installer muss dann noch beispielsweise per Software-Verteilungsmechanismus zusammen mit der Einstellung, wo die Konfigurationen zu finden sind, verteilt werden. Zusätzlich bietet diese Variante die Möglichkeit, einzelne Einstellungen im Workspace vorzubelegen und aktuell zu halten.

Der Nachteil dieser Variante ist der vergleichsweise hohe zentrale Aufwand. Sie ist somit nicht für einzelne/wenige Standardinstallationen geeignet.

Das Oomph-Projekt hat eine umfangreiche englischsprachige Dokumentation zur Verfügung gestellt, welche sich gut als Nachschlagewerk eignet: [Link](#).

Administratorinformationen

In den folgenden Abschnitten wird die Anlage und Anpassung der Oomph-Konfigurationen erklärt. Diese wird in der Regel von wenigen Administratoren durchgeführt.

Begriffsdefinitionen

Liest man sich die Dokumentation von Oomph durch, so wird man erst einmal mit vielen neuen Begriffen erschlagen. Daher hier vorab die wichtigsten Begriffe, die in

den folgenden Abschnitten verwendet werden. Es werden hier bewusst die englischen Originalbegriffe verwendet, um Wiedererkennungswert zu schaffen.

Tabelle 2 Begrifflichkeiten in Oomph

Begriff	Beschreibung
Setup-Model	Die Oomph-Konfigurationen sind, wie in den meisten Programmiersprachen, dateibasiert. Ein Satz dieser Dateien mit spezifischem Format heißt Setup-Model.
Product	Eine Konfiguration auf Installationsebene (Plattform mit bestimmter Version + Plug-ins)
Project	Projektspezifische Einstellungen. In der git-basierten Welt kann dies beispielsweise die Vorgabe von Standard-Repositories sein. Konfiguration auf Workspace-Ebene.
Index	Die Bibliothek an verfügbaren Konfigurationen, die im Eclipse Installer ausgewählt werden können.

Eine Installation mittels Eclipse/Oomph Installer installiert somit immer eine Plattform, die Plug-ins, Oomph Updater, Oomph Recorder und zuletzt die Projekteinstellungen im Workspace.

Benötigte Software für Administratoren

Wer eigene Konfigurationen entwerfen und verwalten will, braucht das Oomph SDK. Das ist nichts anderes als ein Set von Plug-ins auf einer Standard-Eclipse-Plattform. Hiermit werden die benötigten Views und vorgefertigte Perspektiven ausgeliefert, um die Konfigurationsdateien grafisch aufbereitet bearbeiten zu können (Baumansichten, Formulare etc.).

Schritt für Schritt zu einer Basiskonfiguration

Bei der Anlage von Konfigurationen geht man in der Regel von allgemein nach spezifisch vor. Für alle im Folgenden genannten benötigten Dateien bietet das Oomph SDK Wizards an (File → New → Other → Oomph → ...). Zu beachten ist, dass Oomph

sehr viele Optionen an vielen Stellen zulässt. Somit gibt es nicht “die eine, richtige” Implementierung. Was funktioniert und nachvollziehbar ist, ist richtig.

Index

Das bedeutet, man legt zuerst einen *Index* an. Ein Index verweist auf verfügbare *Product-Catalogs* und *Project-Catalogs*. Der Index trägt standardmäßig den Dateinamen *org.eclipse.setup*, kann aber beispielsweise auch *myFirst.setup* heißen.

Folgend der Aufbau eines Index:

```
<?xml version="1.0" encoding="UTF-8"?>
<setup:Index
  xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI"
  xmlns:setup="http://www.eclipse.org/oomph/setup/1.0"
  name="myCompany Eclipse Setups"
  label="index">
  <productCatalog
    href="myCompany.products.setup#"/>
  <projectCatalog
    href="myCompany.projects.setup#"/>
</setup:Index>
```

In der erweiterten Ansicht des Eclipse Installer hat man dann auch die Möglichkeit, zwischen mehreren verfügbaren Indizes zu wechseln.

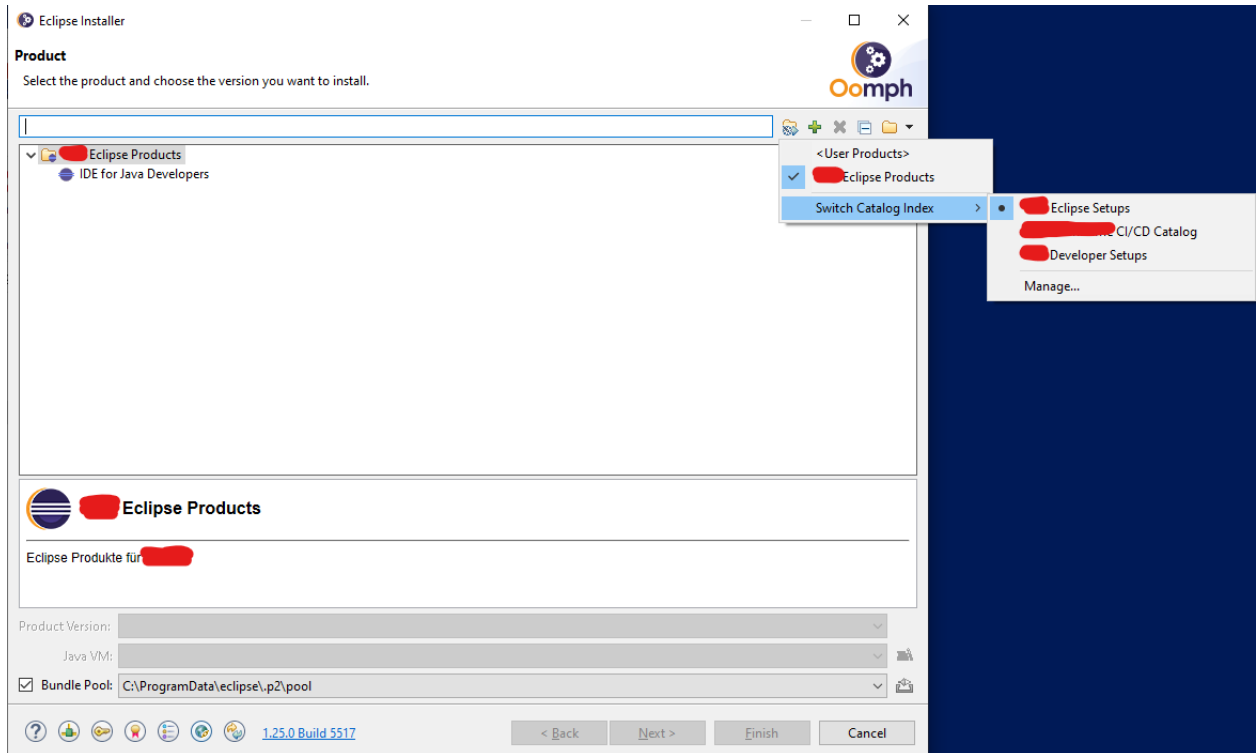


Abbildung 125 Wechsel zwischen Indizes

Product-Catalog und Product

Ein Product-Catalog listet verschiedene Products. Er enthält darüber hinaus auch übergreifende Einstellungen, wie beispielsweise:

- interne Umleitungen von Update-Sites auf lokale Caches
- Definition von Variablen, die später pro Product (Version) oder Project anders gefüllt werden können (beispielsweise für versionsspezifische Update-Sites)
- die Installation des Oomph Client für die spätere Verteilung von Aktualisierungen der in den Products und Projects festgelegten Einstellungen, Oomph Recorder (siehe Abschnitt [Anwenderinformationen](#)).

Auf allen Ebenen (Product-Catalog, Product und Product-Version) können folgende Eigenschaften hinterlegt werden:

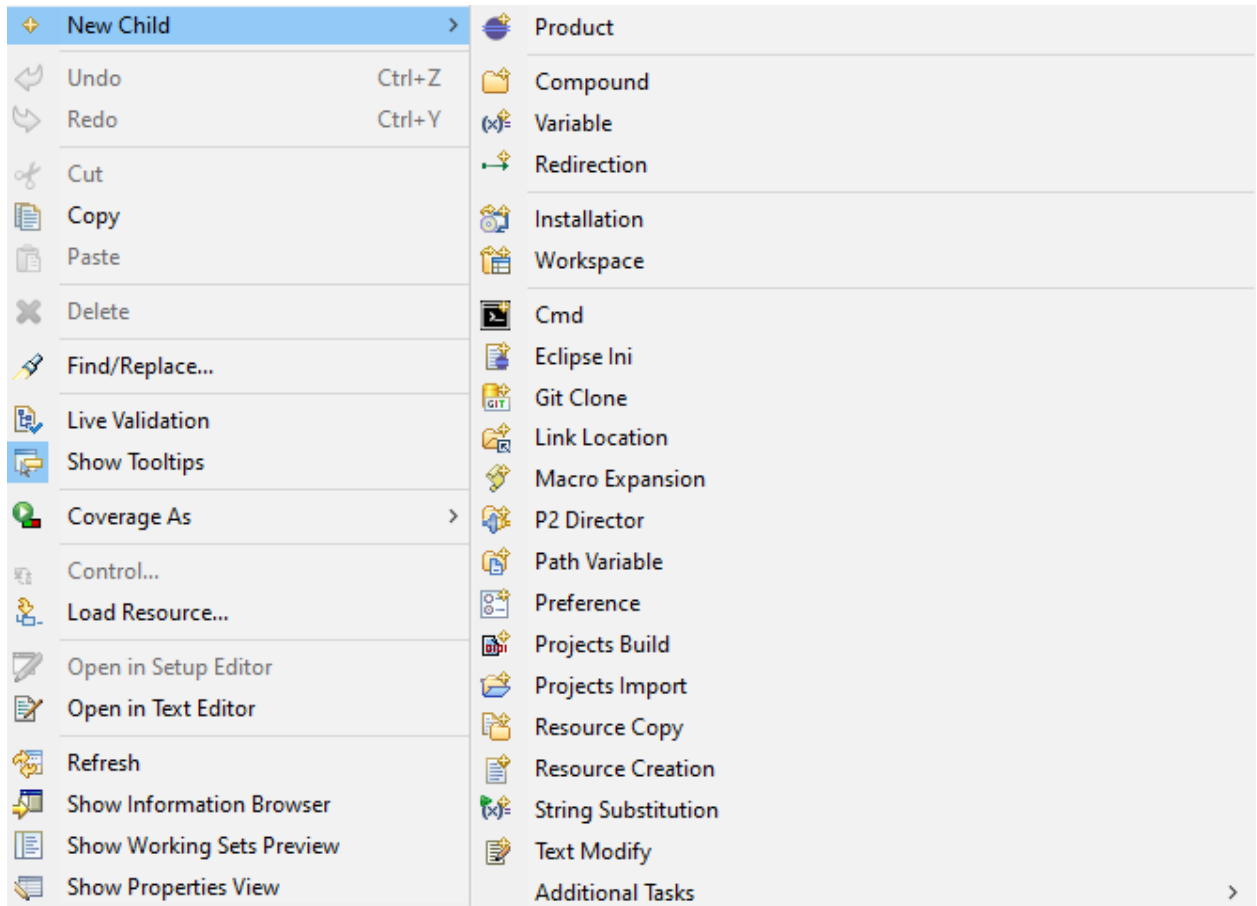


Abbildung 126 Hinzufügen und Festlegen von Eigenschaften

Die meist benötigten Elemente sind:

Tabelle 3 Wichtigste Elemente

Product	Fügt ein Product dem Product-Catalog hinzu (z. B. IDE for JAVA Developers). Kann nur auf Ebene Product-Catalog hinzugefügt werden.
Product-Version	Fügt einem Product eine Version hinzu, in der dann versionsspezifische Einstellungen vorgenommen werden können. Kann nur auf Ebene Product hinzugefügt werden.
Compound	Dies ist eine Art Ordner, in dem später Einstellungen gruppiert werden können.
Eclipse ini	Fügt der Eclipse.ini eine Option hinzu.

Variable	Variable mit Wertzuweisung. Später kann darauf referenziert werden.
Redirection	URL-Weiterleitung. Somit können die Installationen die Originalquellen einbinden, tatsächlich wird jedoch beispielsweise auf ein Artifactory zugegriffen.
P2 Director	Liste der zu installierenden Feature Groups.
Repository	Stellt eine Installationsquelle für Feature Groups

Für eine minimale ADT-Installation wird Folgendes benötigt:

- Ein Product, z. B. "SAP"
- Eine Product-Version, z. B. "2022-03 (4.23)"
- Ein P2 Director Task mit:
 - o Eclipse Platform Packages
 - `epp.package.java` (Value-Range beginnt bei gewünschtem Release → 4.23)
 - `org.eclipse.platform` (Value-Range beginnt bei gewünschtem Release → 4.23)
 - `org.eclipse.rcp` (Value-Range beginnt bei gewünschtem Release → 4.23)
 - `org.eclipse.buildship`
 - `org.eclipse.tips.feature`
 - `org.eclipse.epp.mpc`
 - o Repository-URLs für die Eclipse Platform Packages
 - o <https://download.eclipse.org/releases/2202-03/202203161000> (Link Release-abhängig!)
 - o <https://download.eclipse.org/technology/epp/packages/2022-03/202203101200> (Link Release-abhängig!)

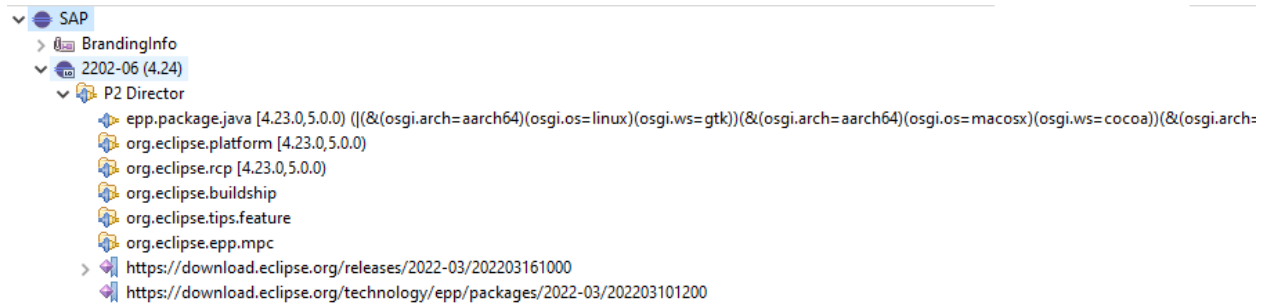


Abbildung 127 Komponenten einer “minimalen” ADT-Installation

Project-Catalog und Project

Der Project-Catalog listet verfügbare Projects. Letztere enthalten Anweisungen für die Veränderung des Workspace. Hier werden beispielsweise auch die zu installierenden Plug-ins hinterlegt.

Die wichtigsten Elemente in einem Project sind:

Tabelle 4 Wichtigste Elemente eines Projects

P2 Director	Gruppier Requirements und Repositories
Requirement	Zu installierende Feature Group
Repository	Update-Site-URL
Stream	Verpflichtendes Objekt. Unterschiedliche Konfigurationen pro Stream möglich. Kann mit Namen, aber ohne Inhalt vorhanden sein.
Variable	Variabler String mit Referenzierbarkeit z. B. in Repositories
Eclipse ini	Veränderungen der Eclipse.ini
Preference	Voreingestellte Veränderung der Einstellungen (Window → Preferences).

Für eine minimale ADT-Installation wird benötigt:

- P2-Director-Knoten

- ADT Feature Groups
 - com.sap.adt.tools.hana.devedition
 - com.sap.core.devedition
- Repository für ADT
 - <https://tools.hana.ondemand.com/latest>
 - Alternativ: `${Variable}` → z. B. `${sap.repository.url}`
- Einen leeren Stream, standardmäßig "Master"

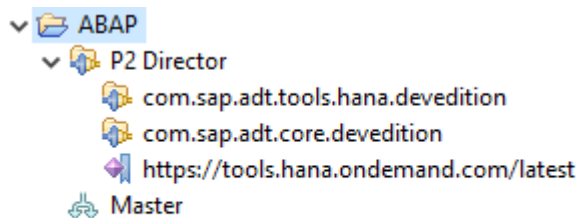


Abbildung 128 Komponenten einer minimalen ADT-Installation

Zu beachten ist, dass ein Plug-in für Eclipse aus mehreren Feature Groups bestehen kann. Wie man diese herausfinden kann, wird im Abschnitt [Zusätzliches Plug-in installieren](#) erklärt.

Verteilung des Installer und Konfiguration

Der Eclipse Installer ist dieselbe exe-Datei, die man von der Eclipse-Startseite herunterladen kann. Diese muss auf die Endgeräte verteilt werden.

Die Verteilung der Konfiguration erfolgt über die Anlage/Anpassung einer Datei mit festgelegtem Namen und Pfad:

```
C:\Users\<currentUser>\.eclipse\org.eclipse.oomph.setup\setups\indices.xmi
```

Beispiel einer Indexliste:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<base:Annotation
```

```
  xmi:version="2.0"
```

```
  xmlns:xmi="http://www.omg.org/XMI"
```

```
xmlns:base="http://www.eclipse.org/oomph/base/1.0"  
source="IndexLocations">  
<detail key="https://pages.github.com/path/subpath/myFirst.setup">  
  <value>Description of my first Oomph Catalog</value>  
</detail>  
</base:Annotation>
```

Hierbei ist der Pfad zu einem oder mehreren Indizes anzugeben. Der Pfad kann auch ein Weblink oder git-Repository-Pfad sein. Die verlinkten Dateien werden dann bei jedem Start des Eclipse Installer heruntergeladen/aktualisiert.

Nun kann jeder Anwender, der im Besitz des Installer ist, diese Konfigurationen auswählen und installieren. Alternativ kann auch der Installer noch per Software-Verteilung im Unternehmen verteilt werden. Somit könnte ein Gleichlauf der Versionen zwischen Oomph-Komponenten in den Installationen und dem verwendeten Eclipse Installer erreicht werden.

Anpassungsbeispiele

Bisher wurden minimale Konfigurationen zur Installation von Eclipse und SAP ADT beschrieben. Die Stärken der Verwendung des Oomph Installer ergeben sich jedoch erst in der weiteren Vorkonfiguration bei der Installation. Auf die häufigsten Erweiterungswünsche wird nun eingegangen.

Zusätzliches Plug-in installieren

Einer der Vorteile von Eclipse als Entwicklungsplattform ist die Offenheit gegenüber Erweiterungen. Somit können von Drittherstellern oder engagierten Community-Mitgliedern Plug-ins geschrieben werden, die den Funktionsumfang der ADT noch erweitern.

Diese können in einem Oomph Project automatisch bei allen Installationen vorinstalliert werden. Leider ist es jedoch nicht möglich, ein Repository als Update-Site hinzuzufügen, dann aber kein Plug-in darauf zu installieren. Diese Update-Sites werden bei der Installation von Oomph verworfen (Stand: 2022).

Nun besteht ein Plug-in aus mindestens einer Feature Group. Im Oomph Project muss letztere hinterlegt werden. Der Name ist jedoch in der Regel nicht bekannt. Hierfür existiert im Oomph SDK eine View "Repository Explorer" (Window → Show View → Other → Oomph → Repository Explorer). Unter (manueller) Angabe einer Update-Site wird eine Abfrage der bereitgestellten Feature Groups durchgeführt. Diese können dann per Drag-and-drop oder **STRG+C** und **STRG+V** in den P2-Director-Knoten des Oomph Project eingefügt werden. Zusätzlich zu den Feature Groups muss auch ein Repository-Knoten existieren, der diese Update-Site zur Verfügung stellt (kann auch allgemeiner, d. h. im Project-Catalog oder Product (-Catalog) erfolgen).

Stand Oktober 2022 sieht die Abfrage der SAP-Update-Site wie folgt aus:

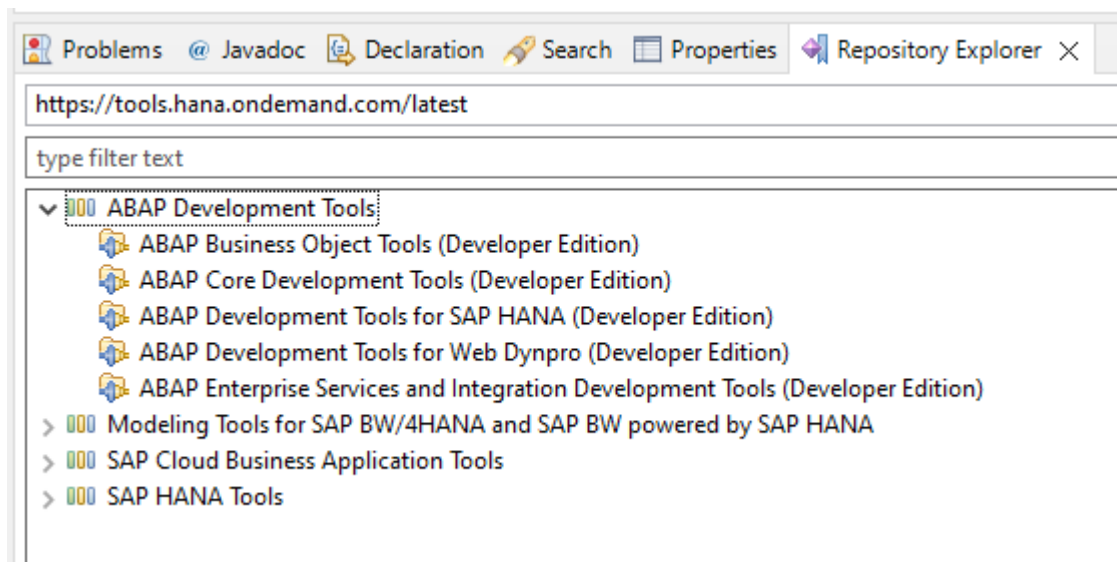


Abbildung 129 Bestandteile der SAP Update Site

Vorgeben von Einstellungen

Neben der Installation von ADT ist die Voreinstellung diverser Einstellungen für alle Installationen einer der größten Vorteile.

Alle zu vergebenden Einstellungen kommen in den Compound/Ordner *User-Preferences* des Oomph Projects. Darunter gibt es dann einzelne Unterordner pro Feature (als Teil einer Feature Group) und darin dann die Einstellungen. Hierbei ist es jedoch jedem Feature selbst überlassen, in welcher Darstellung es seine Einstellungen speichert. Einige Einstellungen sind als direkte Werte abgelegt (klassische Checkboxes), andere beispielsweise als ein großes XML pro Einstellungsseite. Gerade bei letzterem können dann nur alle Einstellungen der Seite oder keine vorgegeben werden. Ein Beispiel für die XML-Darstellung sind die Code Templates der SAP ADT.

Da diese nicht einheitliche Darstellung unpraktisch zu administrieren ist und die Namen der Features in der Regel auch nicht bekannt sind, existieren auch hier Hilfswerkzeuge: In den Einstellungen wird ein Oomph Recorder installiert. Näheres zur Verwendung als Anwender kann dem Abschnitt [Oomph Recorder](#) entnommen werden. Der Oomph Recorder zeichnet den letzten Stand aller geänderten Einstellungen auf Benutzerebene auf (also übergreifend über Eclipse-Installationen) und fragt nach dem Schließen der Einstellungen, ob diese immer/einmal/nie gespeichert werden sollen. Ebenso wird damit ein Button in der Button-Leiste von Eclipse verfügbar, um sich die bereits gespeicherten Einstellungen anzuschauen.

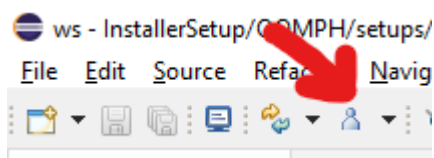


Abbildung 130 Möglichkeit zur Anzeige der bereits gespeicherten Einstellungen

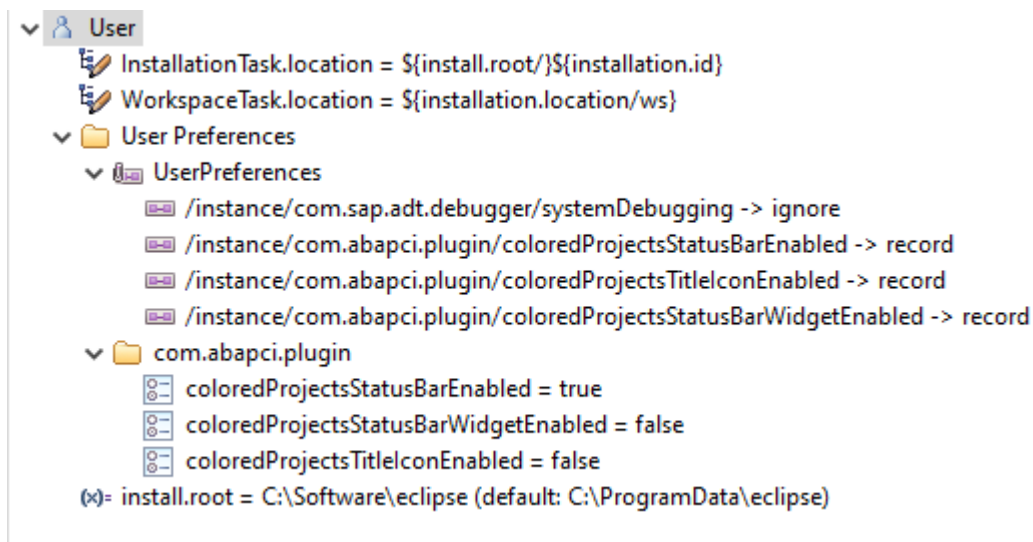


Abbildung 131 Bereits gespeicherte Einstellungen

Hierbei handelt es sich wieder um setup-Dateien. Somit können auch hier wieder die aufgezeichneten Einstellungen mit Drag-and-drop oder **STRG+C** und **STRG+V** in das Oomph Project übernommen werden.

Folgend ein paar Beispiele für allgemein verteilte Einstellungen. Hier finden sich einige der ADT-exklusiven Features, die SAP leider nicht alle standardmäßig aktiviert.

- Code Templates

- Aktivierung der zusätzlichen Code-Highlights in ABAP (verschiedene Farben)
- Kontinuierliche Syntaxprüfung aktivieren
- Aktivieren des Occurrence Marker
- Einheitliche Darstellung von Einrückungen (Tabs oder Leerzeichen, Breite der Einrückung)

Grenzen Arbeitsspeicher anheben

Mit SAP ADT existieren keine Grenzen mehr in der Anzahl der geöffneten Fenster/Objekte. Das hat aber auch zur Folge, dass bei vielen gleichzeitigen Tabs der benötigte Arbeitsspeicher in die Höhe schnellt. Da Eclipse selbst auch eine Java-Anwendung ist, müssen VM-Parameter festgelegt werden. Ist der verfügbare Speicher einer Java-VM überschritten, findet analog der RAM-Verwaltung des Betriebssystems ein Swapping in Festplattendateien statt. Das macht Eclipse langsam und der Spaß an der Benutzung leidet. Es können für den VM-Speicher Mindest- und Maximalgrenzen, initial reservierte Speichermenge und vieles mehr eingestellt werden. Da hiermit aber auch eine Menge kaputt gemacht werden kann und die benötigten Grenzen doch sehr individuell sind, wird bewusst auf konkrete Empfehlungen verzichtet.

Sollten doch solche Parameter für alle Nutzer vorgegeben werden, so müssen diese in der Eclipse.ini angegeben werden. Somit müssen im Oomph Project entsprechende Eclipse Ini Tasks angelegt werden.

Erster Start mit ABAP Perspective

Werden die SAP ADT installiert, so sind die Standard-Perspectives zunächst nur vorhanden, aber nicht geöffnet. Eclipse startet beispielsweise mit der Java Perspective, welche die meisten ABAP-Entwickler eher selten benötigen werden. Daher ist es wünschenswert, die Installation direkt mit der ausgelieferten ABAP Perspective zu starten. Die Hinterlegung weiterer Perspectives (z. B. Debug, ABAP Profiling etc.) in den zuletzt verwendeten Perspectives oben rechts ist leider nicht möglich.

Das Erzwingen einer Start-Perspective ist eine Startoption in der Eclipse.ini. Es muss somit ein Eclipse Ini Task mit den folgenden Eigenschaften angelegt werden:

Tabelle 5 Eigenschaften Eclipse Ini Task

Option	-perspective
Value	com.sap.adt.ui.AbapPerspective
VM	false

5.5.4.1 Anwenderinformationen

Installation mittels Eclipse Installer

Beim ersten Start des Eclipse Installer kann es sein, dass dieser im einfachen Modus startet. Für die Verwendung eigener Konfigurationen ist jedoch der erweiterte Modus nötig. Über das Menü oben rechts kann in diesen gewechselt werden (“Advanced Mode”).

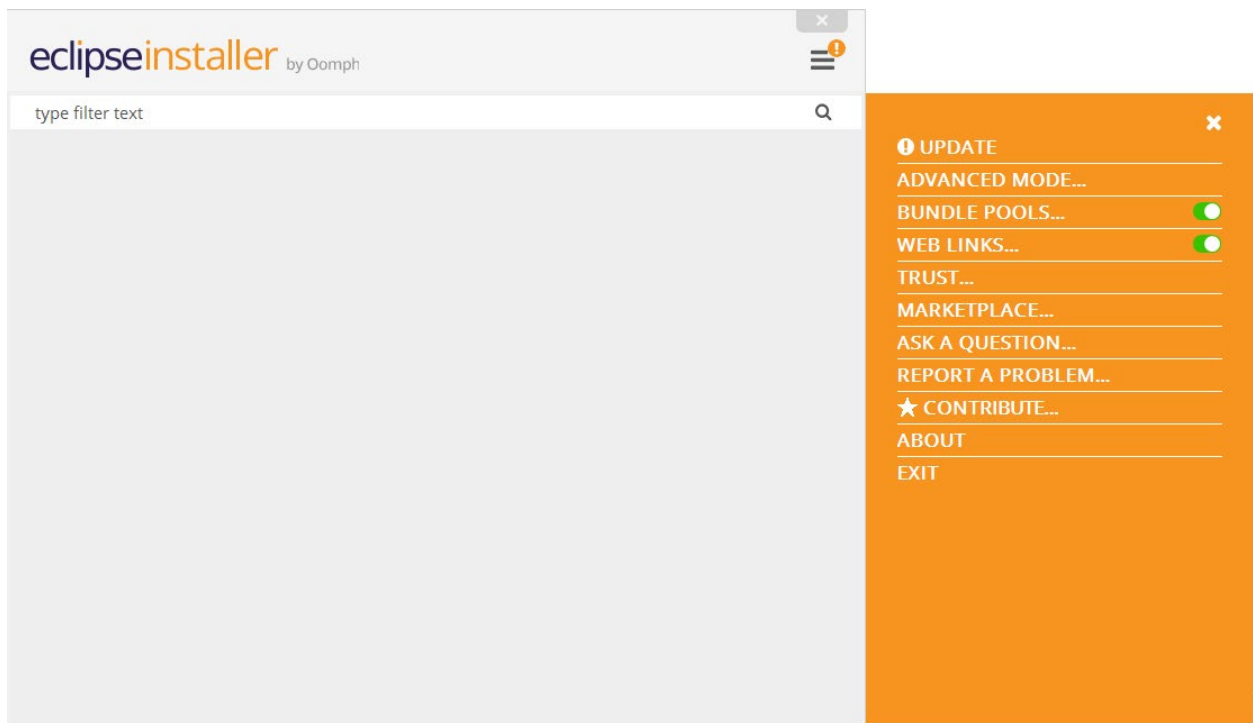


Abbildung 132 Wechsel in den “Advanced Mode”

Nun sieht man eine Liste an verfügbaren Products. Dies ist der Inhalt des ersten referenzierten Index in der Indices.xml. Oben rechts kann zwischen allen gelisteten Indizes gewechselt werden.

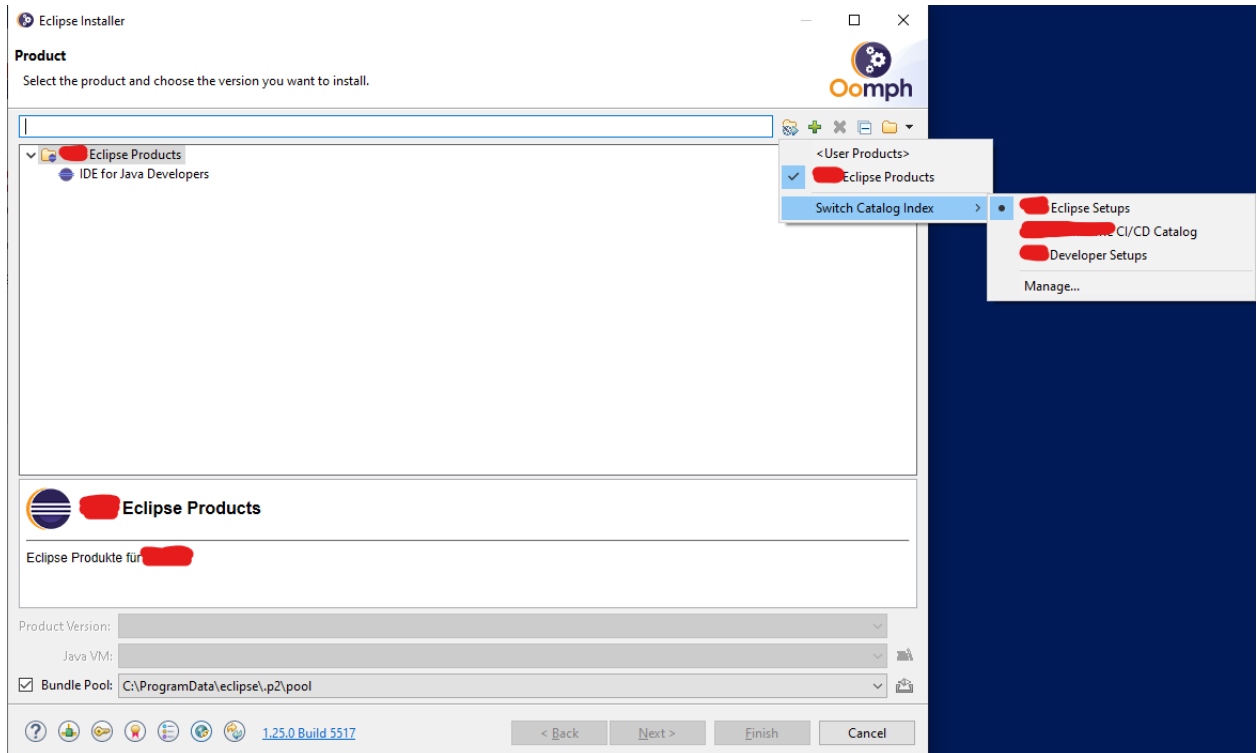


Abbildung 133 Wechsel zwischen Indizes

Hier wählt man einen passenden Eintrag aus. Sobald dies geschehen ist, wird unten die verfügbare Produktversion auswählbar.

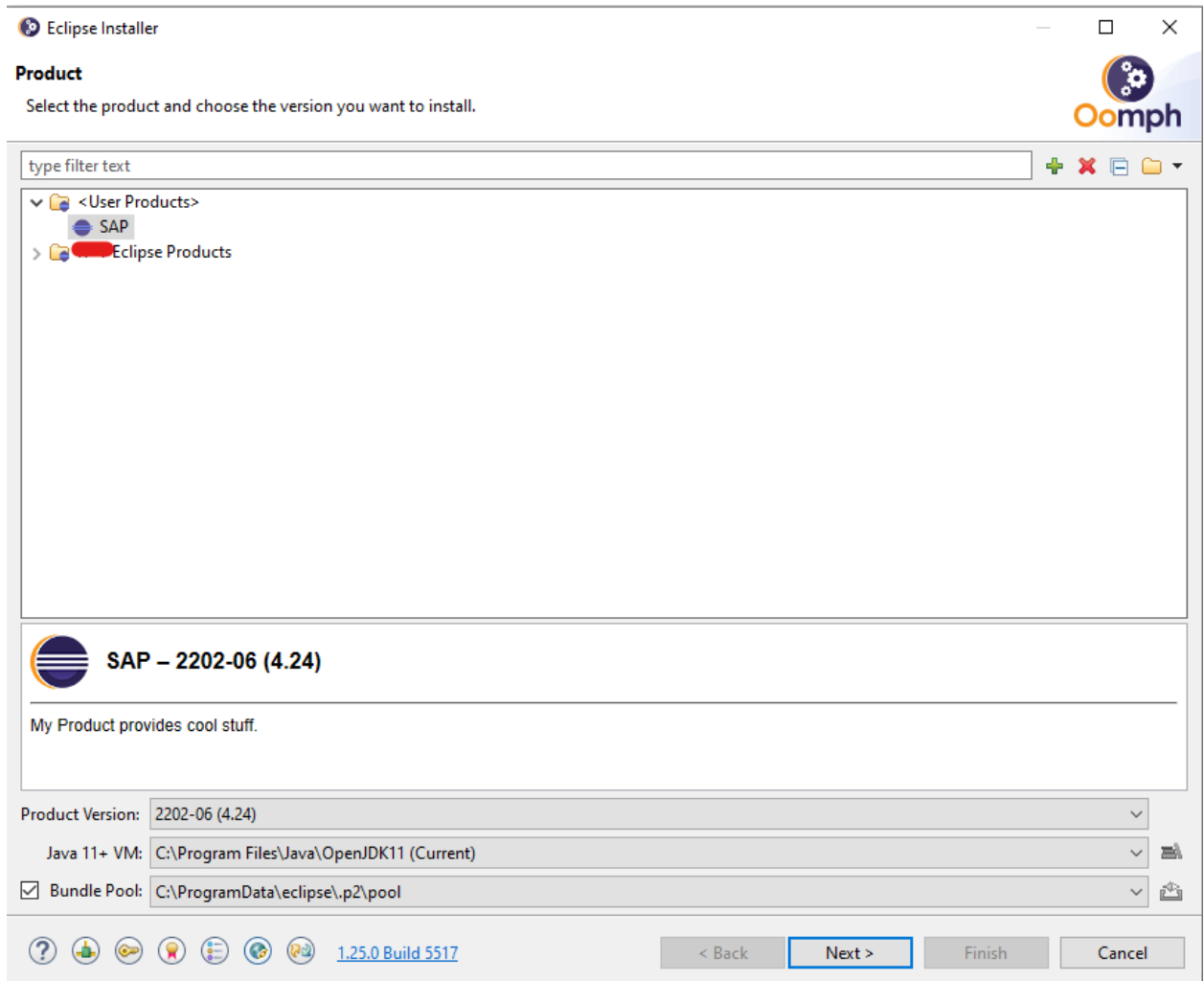


Abbildung 134 Auswahl der Product-Version

Im nächsten Bild sind die Projects auswählbar. Hier können theoretisch mehrere Projects für eine Installation gewählt werden. Jedoch kann es passieren, dass diese Projects dann konkurrierende Einstellungen vornehmen, und es kommt zu Problemen.

Zu einem Project muss dann auch zwingend ein Stream gewählt werden. Wurde nur einer definiert, ist dieser bereits vorausgewählt.

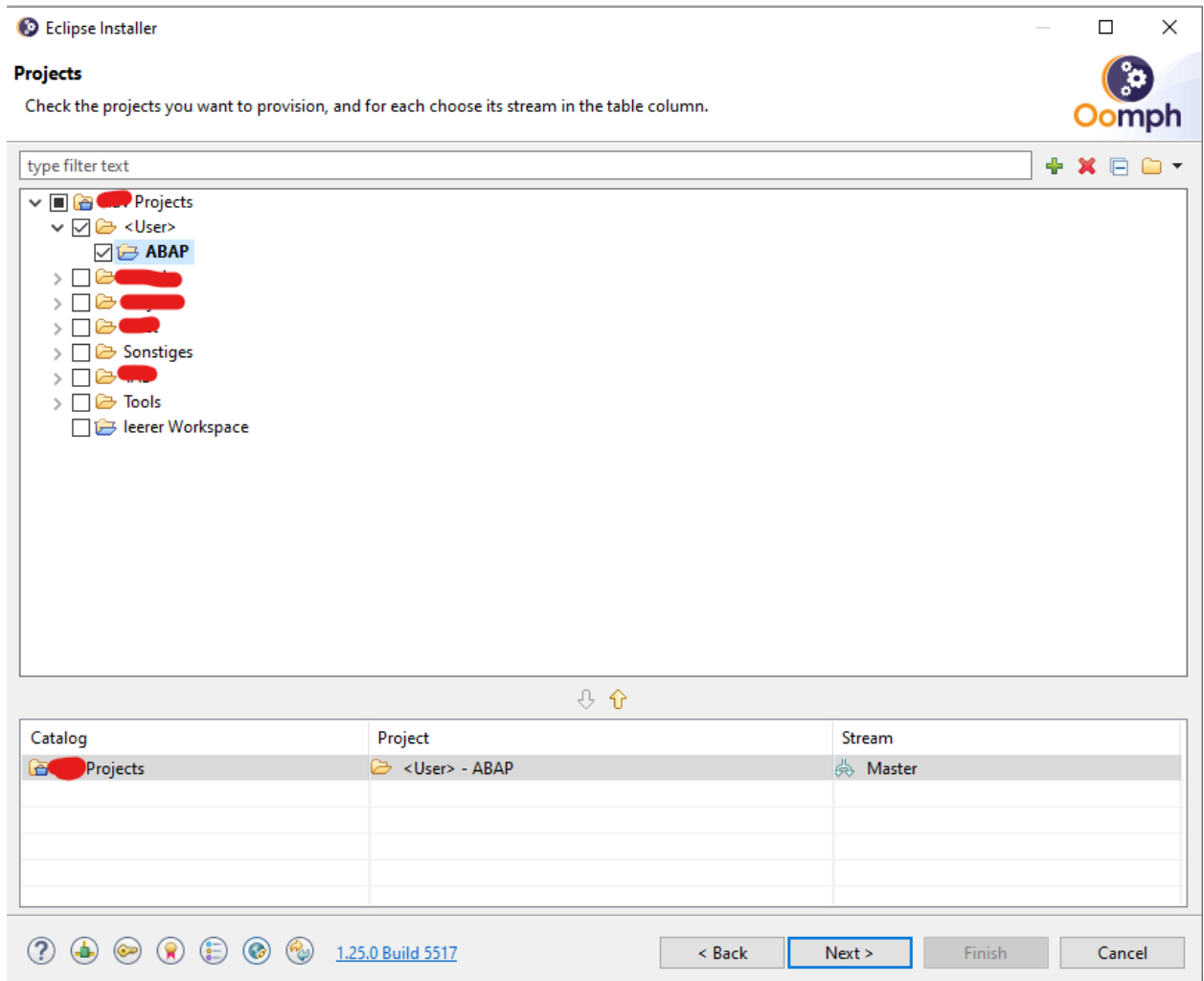


Abbildung 135 Auswahl des Streams

Nun werden noch definierte und verwendete, aber nicht gefüllte Variablen abgefragt. Dies können beispielsweise die Pfade zu Installation und Workspace sein.

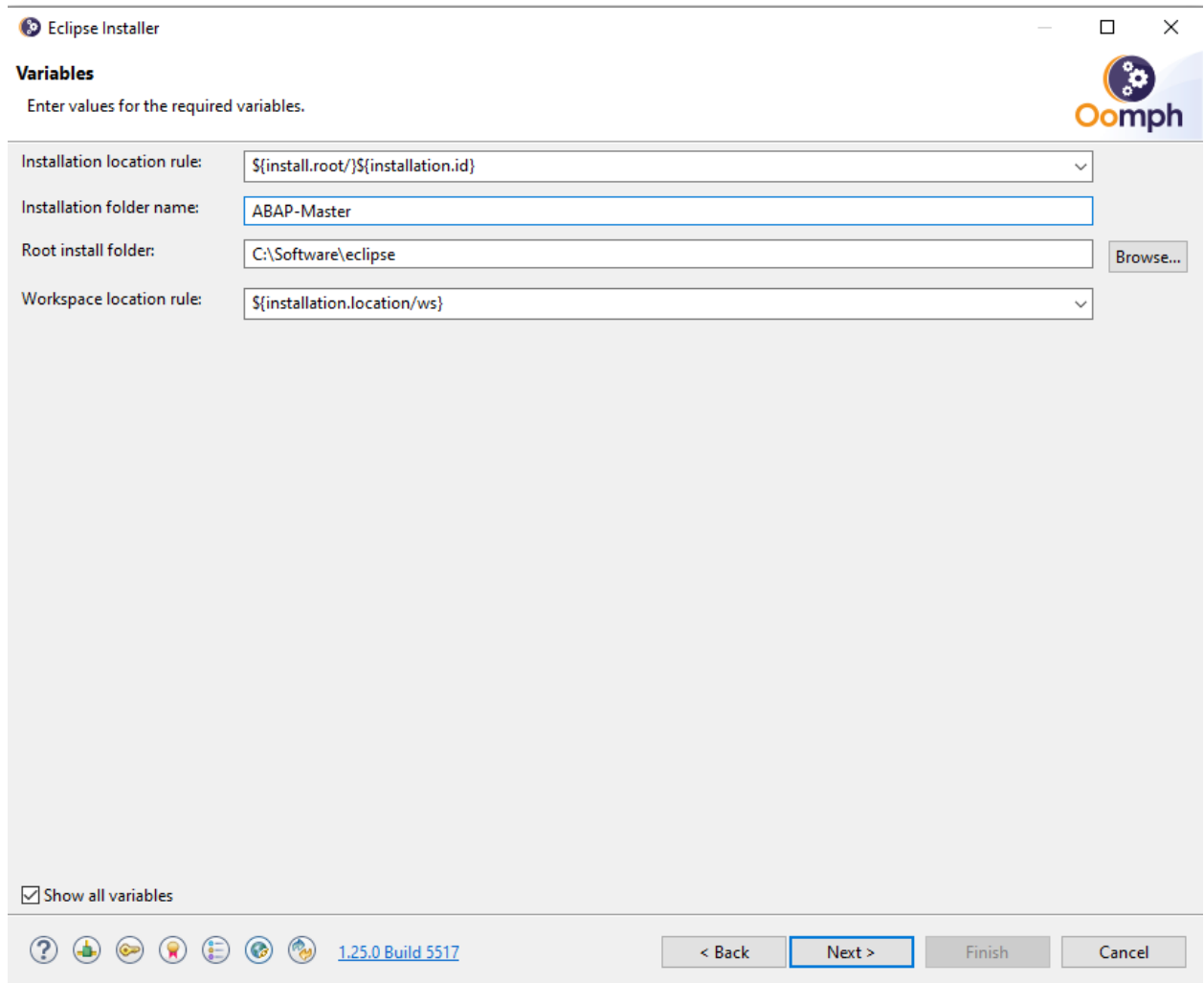


Abbildung 136 Abfrage weiterer Variablen

Zuletzt wird noch eine Bestätigungsseite angezeigt, die aber in der Regel keine neuen Informationen enthält.

Während der Installation kann es passieren, dass Pop-ups angezeigt werden, in welchen Lizenzbedingungen akzeptiert und/oder Zertifikaten vertraut werden sollen.

Ist der Haken im Installer gesetzt, wird die eben installierte Instanz nach der Installation gestartet. Dabei werden dann noch einmal die Einstellungen des Workspace vorkonfiguriert, da der Workspace erst jetzt existiert. Nun ist die Installation abgeschlossen.

Oomph Recorder

Der Oomph Recorder ist ein Hilfswerkzeug, um Einstellungen über Eclipse-Installationen hinweg zu vereinheitlichen und zu sichern. Optional können diese Einstellungen auch an den Eclipse User Account geladen werden, so dass hier auch noch eine geräteübergreifende Synchronisation möglich ist. In den meisten Firmen ist die Verwendung eines Eclipse Accounts aber vermutlich nicht möglich oder nicht gern gesehen.

Der Oomph Recorder wird unter Window → Preferences mit einem neuen Aufnahmesymbol aktiviert. Ab dann wird bei jedem Schließen der Einstellungen abgefragt, ob die gerade veränderten Einstellungen immer/einmal/nie im letzten Zustand gespeichert werden sollen.

Diese Einstellungen auf User-Ebene werden bei einer neuen Eclipse-Installation erst nach dem ersten Start und nach den Einstellungen des verwendeten Oomph Projects angewendet. Somit können hier auch Standardeinstellungen benutzerindividuell übersteuert werden oder eigene Standardeinstellungen definiert werden.

5.6 Fehlersituationen

Während der Installation oder eines Updates kann es zu verschiedenen Fehlern kommen. Auf die häufigsten Fehler soll hier einmal eingegangen werden.

5.6.1 Fehler beim Update oder Upgrade

Bei der Durchführung eines Updates oder Upgrades kann es vorkommen, dass die Installation fehlschlägt.

Eine Installation läuft in der Regel in den folgenden Phasen ab:

1. Zielversionen aller Komponenten und deren Pakete berechnen, inklusive Beachtung der Abhängigkeiten
2. Pakete herunterladen
3. Pakete installieren

Die erfahrungsgemäß häufigste Fehlerursache bei der Berechnung der Zielversionen sind nicht erfüllbare Abhängigkeiten.

So fordert in einem erfundenen Beispiel ein ADT-Upgrade auch eine neuere Version von Eclipse selbst. In den verfügbaren Update-Sites ist jedoch nur die aktuelle Version zu finden. Somit ist die Fehlerursache hier eine falsche oder veraltete Update-Site.

In schwierigeren Fällen kann es sein, dass beispielsweise in neueren Releases von Eclipse eine Komponente entfernt oder ersetzt wird. Ein Plug-in wie die ADT hat jedoch noch eine Abhängigkeit dazu definiert. Hier kann nur der Hersteller des Plug-in Abhilfe schaffen.

Eine solche Situation kann bei Verwendung von HANA Studio als Eclipse-Plattform entstehen. Hier werden bei der Installation keine öffentlichen Update-Sites für die Eclipse-Plattform eingestellt, weshalb sie spätestens nach sechs Monaten (= zwei vierteljährliche Releases) nicht mehr mit aktuellen ADT-Versionen zusammenpasst.

Deutlich seltener treten Fehler beim Download auf. Im ersten Schritt wurden bereits die Inventarlisten der Update-Sites (artifacts.xml und contents.xml) untersucht, und es wurde eine passende Paketversion gefunden. Befindet sich diese dann allerdings nicht im entsprechenden Unterordner, so kommt es zu einem Download-Fehler. Hier ist dann die Update-Site inkonsistent. Dies kann ebenfalls nur der Anbieter des Pakets beheben.

5.6.2 Single-sign-on-Bibliotheken

Je nach Single-sign-on-Strategie können Umgebungsvariablen auf Bibliotheken notwendig sein. Wenn SAP GUI lediglich als 32-Bit-Anwendung installiert wird (bis SAP GUI 7.70 einzige Option), werden immer die Bibliotheken verwendet, die in den Umgebungsvariablen SNC_LIB und SNC_LIB_2 referenziert werden.

Wird Eclipse jedoch als 64-Bit-Anwendung installiert (Standardfall), so wird die in der Umgebungsvariable SNC_LIB_64 referenzierte SSO-Bibliothek verwendet.

In älteren Blog-Einträgen werden noch die Kerberos-Bibliotheken von Windows empfohlen (32 Bit: gsskrb5.dll, 64 Bit: gx64krb5.dll). Diese sind jedoch in neueren Windows-Installationen nicht mehr vorhanden und sollten auch nicht mehr verwendet werden. SAP liefert mit dem SAP GUI eine SAP CryptoLib aus. Diese liegt in 32 Bit und 64 Bit vor.

5.6.3 “No repository found containing”

<https://launchpad.support.sap.com/#/notes/2186770>

Ab und zu scheint es Probleme zu geben, die ADT zu aktualisieren. Im Protokoll erscheinen mehrere Fehler “No repository found containing: ...”. Der Hinweis empfiehlt, die Update-Site zu entfernen, Eclipse neu zu starten und dann die Update-Site neu hinzuzufügen.

5.6.4 PKIX - Certificate Error

<https://launchpad.support.sap.com/#/notes/3131747>

Hier handelt es sich um einen Zertifikatsfehler. Zur Update-Site wird eine verschlüsselte Verbindung aufgebaut (HTTPS). Wenn im Unternehmensnetzwerk SSL-Verbindungen aufgebrochen werden oder es keinen gemeinsamen Keystore für interne und externe Update-Sites gibt, dann kann es zu diesem Fehler kommen. Der Hinweis gibt eine mögliche Lösung hierfür. Eine andere Möglichkeit ist die Verteilung eigener angepasster JDK.

5.6.5 macOS aarch64 support & SAP GUI for Java

<https://launchpad.support.sap.com/#/notes/3251738>

Die Architektur von SAP GUI und Eclipse-Installation sollte grundsätzlich mit derselben Prozessorarchitektur installiert werden. Gerade bei Apple M1 und folgend könnte es hier zu Abweichungen kommen.

5.6.6 Offline-Installation – Download ADT-Abhängigkeiten

<https://launchpad.support.sap.com/#/notes/2369308>

Soll ADT offline installiert werden, müssen diverse Abhängigkeiten beachtet werden. Dieser Hinweis gibt einige Möglichkeiten für Abhilfe in diesem Fall.

6 Best Practices Eclipse-Konfiguration

6.1 Einstellungen in Eclipse

In Eclipse gibt es zahlreiche Einstellungsmöglichkeiten, die das Leben als Entwickler leichter, aber manchmal auch schwerer machen können. In diesem Abschnitt erfahren Sie mehr über die verschiedenen Möglichkeiten und die wichtigsten Einstellungen. Wichtig zu wissen: Es gibt zwei Ebenen, auf denen Sie Einstellungen vornehmen können. Die *globale Ebene* für Eclipse und die *projektspezifische Ebene* für ein SAP-System.

6.1.1 Globale Einstellungen

Nach dem Öffnen der Einstellungen über das Menü (Window → Preferences) sehen Sie alle Einstellungen für Eclipse. Im Fenster auf der linken Seite befindet sich die Struktur mit Unterknoten für die Navigation, darüber ein Suchfeld, um nach Knoten oder Einstellungen zu suchen. Auf der rechten Seite befinden sich die Einstellungen zum gewählten Punkt.

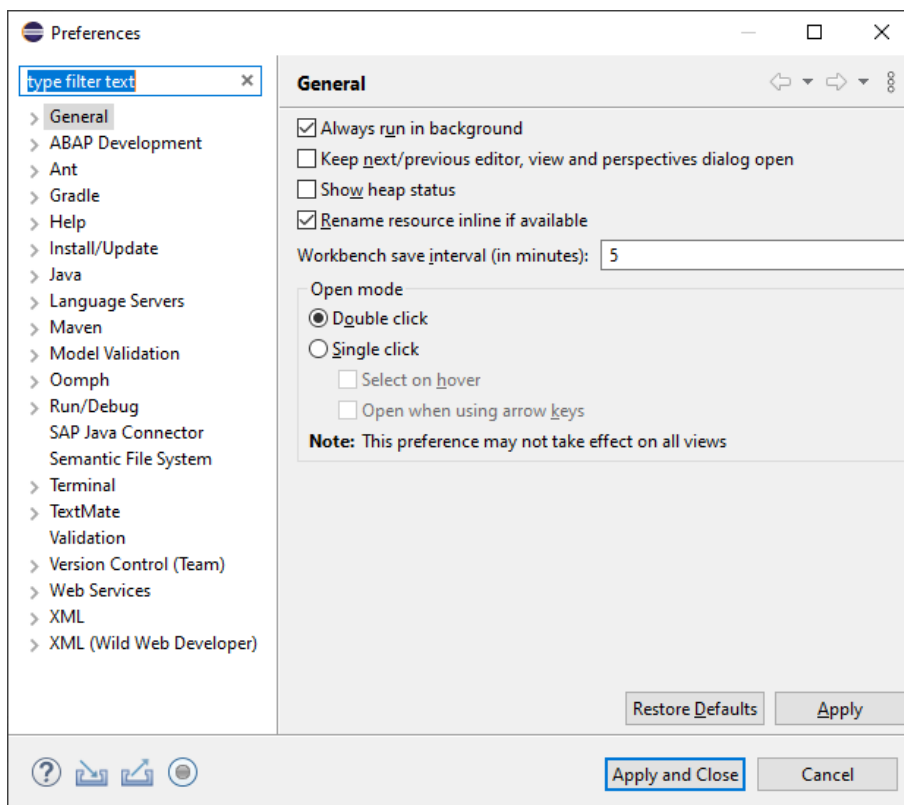


Abbildung 137 Einstieg in die globalen Einstellungen

In den folgenden Abschnitten zeigen wir unseren Vorschlag für diese Einstellungen, den Pfad innerhalb der Einstellungen, um diese zu finden und eine kurze Erklärung der Auswirkungen.

6.1.1.1 Dark Theme

(General → Appearance)

Viele Entwicklungsumgebungen bieten mittlerweile die Möglichkeit, mit einem hellen oder dunklen Theme zu arbeiten, um die Augen zu schonen oder einfach nur dem persönlichen Geschmack zu entsprechen.

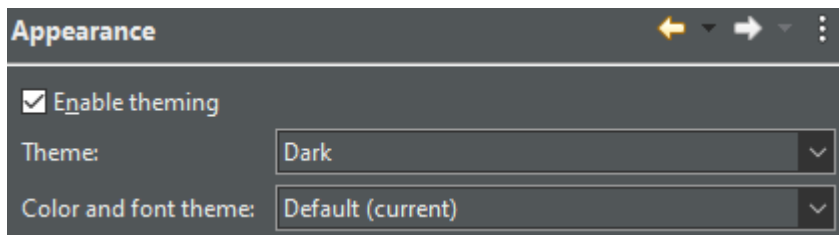


Abbildung 138 Einstellung für das Dark-Theme

6.1.1.2 Einrückung

(General → Editors → Text Editors)

Beim Schreiben von ABAP Quellcode wird oft mit einer Tabulatorweite von 2 Leerzeichen gearbeitet, standardmäßig ist sie in Eclipse aber auf 4 eingestellt ("Displayed tab width"). Außerdem können Sie einstellen, ob statt einem Tabulator Leerzeichen eingefügt werden ("Insert spaces for tabs") und ob beim Löschen gleich ein ganzer Tabulator entfernt werden soll ("Remove multiple spaces on backspace/delete").

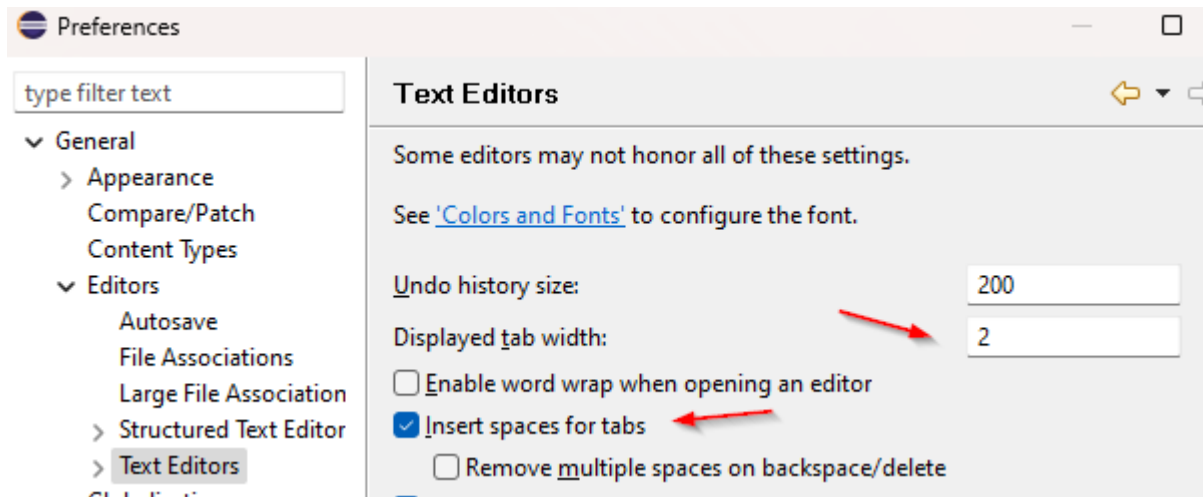


Abbildung 139 Einstellung zur Einrückung des Quellcodes

6.1.1.3 Fehlermeldung im Code

(General → Editors → Text Editors)

Fehlermeldungen tauchen als Ikonen links neben dem Quellcode auf. Um die Information zum Fehler zu erhalten, müssen Sie mit der Maus über die Ikone gehen. Über die Option „Show code minings for problem annotations“ können Sie sich die gesamte Fehlermeldung auch direkt im Code anzeigen lassen. Wählen Sie dazu die Art der Meldung aus.

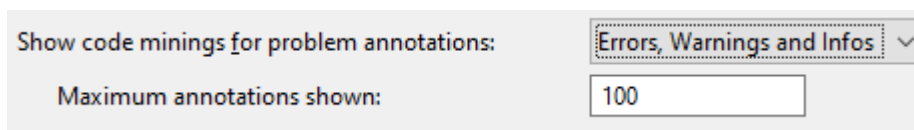


Abbildung 140 Beispiel für die Einstellung

```

▲Use functional writing style: 007 - Functional writing style for CALL METHOD
CALL METHOD main.

◆EXPORTING can be omitted: 030 - EXPORTING can be omitted | ▲Ungelesene Felder: Erweiterte Programmprüfung
DATA(test) = one_parameter( EXPORTING i_test = abap_true ).

◆Irreführender Feldname: Erweiterte Programmprüfung (SLIN) | ▲Unbenutzte Felder: Erweiterte Programmprüfung
data x TYPE i.

■Found a suspicious comment: ...: 070 - Find comment markers
"Fixme
    
```

Abbildung 141 Ergebnisbild im Quellcode

6.1.1.4 Rechtschreibprüfung: Kommentare

(General → Editors → Text Editors → Spelling)

Wenn Sie Kommentare in Eclipse nicht auf Englisch schreiben, erhalten Sie von der Rechtschreibprüfung viele rote Kommentare. Diese Prüfung können Sie über die Einstellungen (“Enable spell checking”) deaktivieren oder Sie laden das Wörterbuch für Deutsch nach.

6.1.1.5 Tastenkombinationen

(General → Keys)

Konfiguration der Tastenkombinationen in Eclipse, mit denen Sie Ihre Wunscheinstellungen definieren können. Außerdem können Sie sich Shortcuts einblenden lassen, wenn diese ausgelöst wurden (“Through keyboard”) oder es zur ausgeführten Aktion eine Tastenkombination gibt (“Through mouse click”). Diese Option ist immer dann sinnvoll, wenn Sie Schulungen halten, Kollegen bei der Einarbeitung in Eclipse unterstützen oder sich selbst in die Tastenkombinationen einarbeiten möchten.

Siehe auch:

Blog Post [Useful Keyboard Shortcuts for ABAP in Eclipse](#)

SAP Help [Keyboard Shortcuts for ABAP Development](#)

6.1.1.6 Debugging

(ABAP Development → Debug)

Möglichkeit zur Festlegung der allgemeinen Debugger-Einstellungen, aber auch Aktivierung (“Enable debugging of system programs”) des System-Debuggings.

6.1.1.7 Farbformatierung

(ABAP Development → Editors → Source Code Editors → ABAP Keyword Colors)

Um wichtige Schlüsselwörter in Eclipse hervorzuheben, können Sie diese mit zusätzlichen Farbkombinationen hervorheben. Dazu können Sie einzelne Schlüsselwörter hervorheben oder alle (“Select all”). Damit lassen sich im Quellcode wichtige Passagen leichter identifizieren.

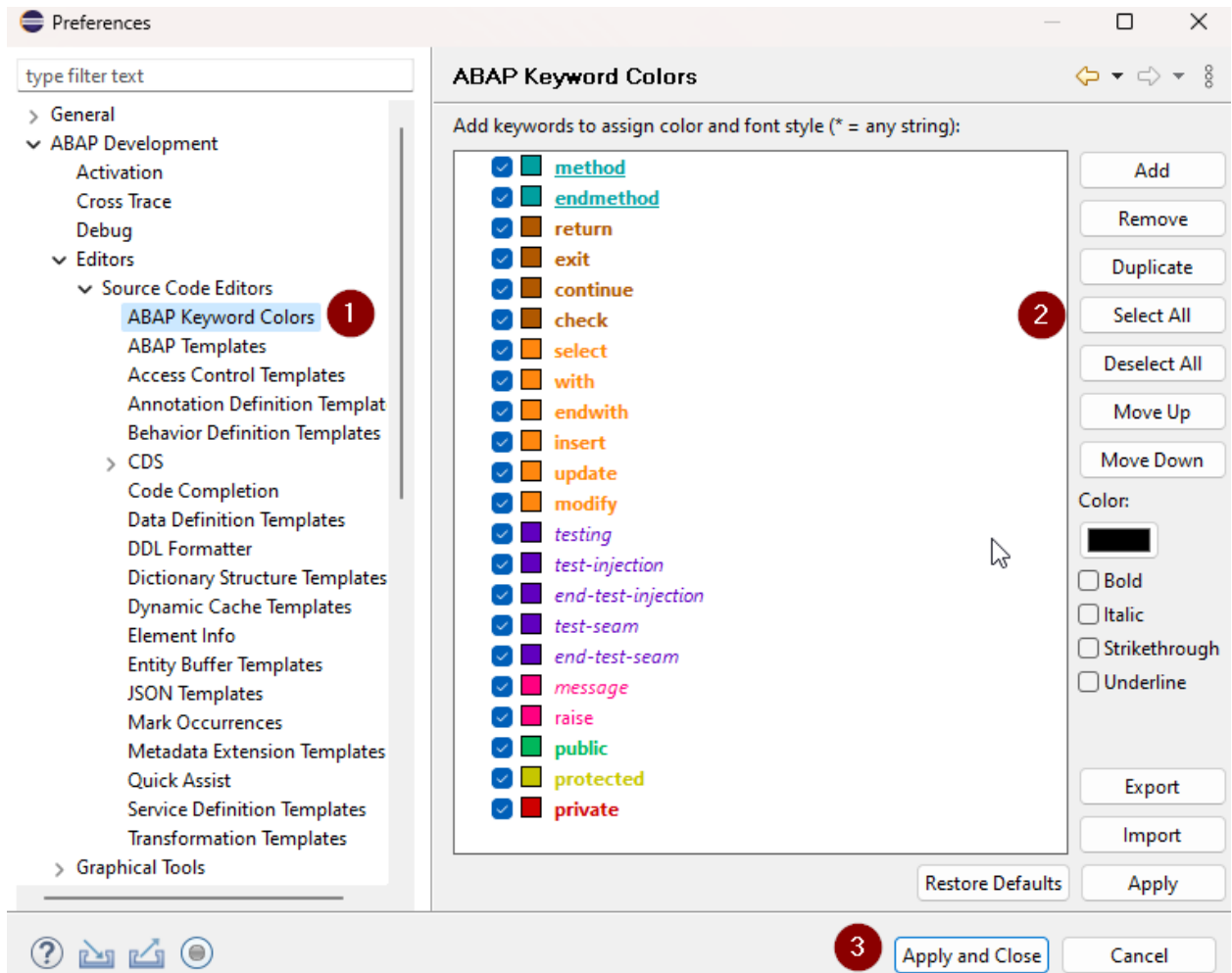


Abbildung 142 Farbeinstellungen zur Hervorhebung der Schlüsselwörter im Quellcode

6.1.1.8 Code-Vorlagen

(ABAP Development → Editors → Source Code Editors → ABAP Templates)

Für häufig verwendete Code-Fragmente liefert SAP Vorlagen aus, die man nach Belieben an eigene Bedürfnisse anpassen kann. Auch neue Vorlagen sind möglich. Die Vorlagen werden im Coding durch Eingabe des Vorlagennamens und Autocomplete (**STRG+SPACE**) eingefügt.

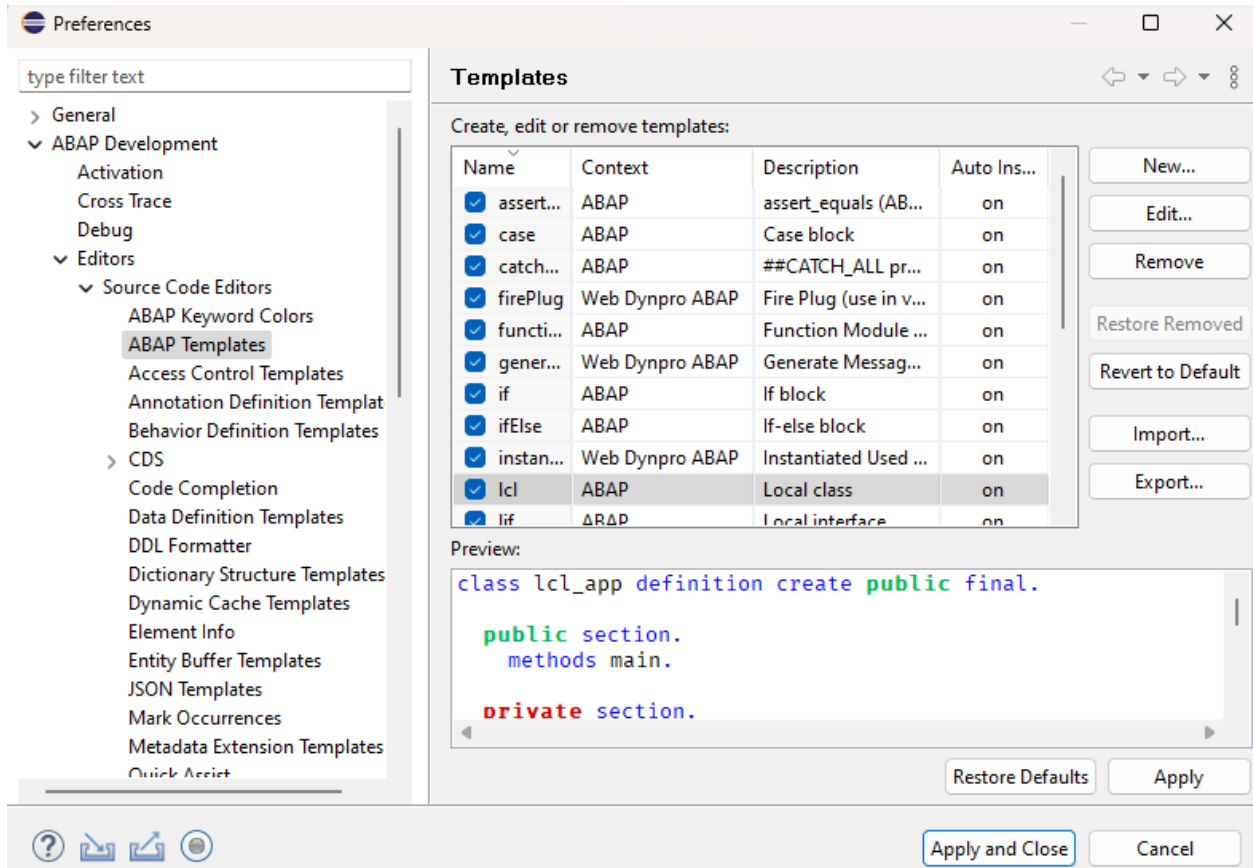


Abbildung 143 Verwaltung der ABAP Templates in den Einstellungen

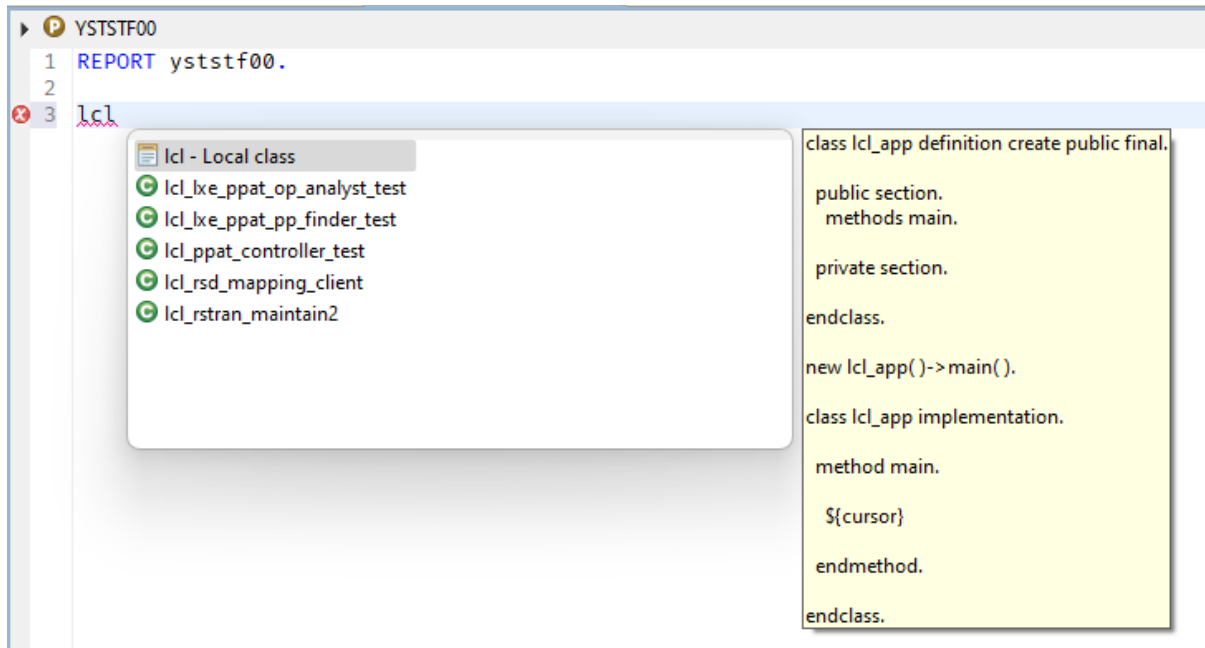


Abbildung 144 Einfügen des Templates in den Quellcode

6.1.1.9 Alias für CDS-Felder

(ABAP Development → Editors → Source Code Editors → CDS → Code Completion)

Bei der Erstellung eines Core Data Service (CDS) werden bei der Einbindung einer Tabelle die Feldnamen ohne Unterstrich und in Camel Case mit einem Alias zur Verfügung gestellt. Mit der Option (“Add aliases for table fields ...”) wird dies standardmäßig beim Einfügen über “Insert all elements” durchgeführt (Default-Einstellung).

6.1.1.10 Auto-Vervollständigung

(ABAP Development → Editors → Source Code Editors → Code Completion)

Standardmäßig ergänzt Eclipse Klammern und Anführungsstriche am Ende eines Ausdrucks (“Automatically close brackets and literals”) und fügt Leerzeichen innerhalb von Klammern ein (“Add additional whitespace inside ...”). Wenn Sie diese Optionen stören, können sie hier deaktiviert werden. Weiterhin können Sie sich auch Nicht-Schlüsselwörter von Eclipse vorschlagen lassen (“Also suggest non-keywords”), wodurch Ihnen dann z. B. auch Variablennamen vorgeschlagen werden.

6.1.1.11 Suche

(ABAP Development → Search)

Hier können Sie Einstellungen am Suchdialog (**STRG+SHIFT+A**) vornehmen, z. B. ob das alte Such-Pattern weiterverwendet wird (“Use pattern from previous search”) oder auch die Anzahl der angezeigten Treffer (“Maximum number of results”). Wichtig ist aber auch der Typ des Objektes (“Display object types”) und in welchem Paket (“Display packages”) es sich befindet.

6.1.2 Projektspezifische Einstellungen

Sie finden die systemspezifischen Einstellungen mit einem Rechts-Klick auf das ABAP-Projekt unter “Properties”. Der Aufbau des Fensters ist dem der globalen Einstellungen ähnlich und lässt sich gleich bedienen.

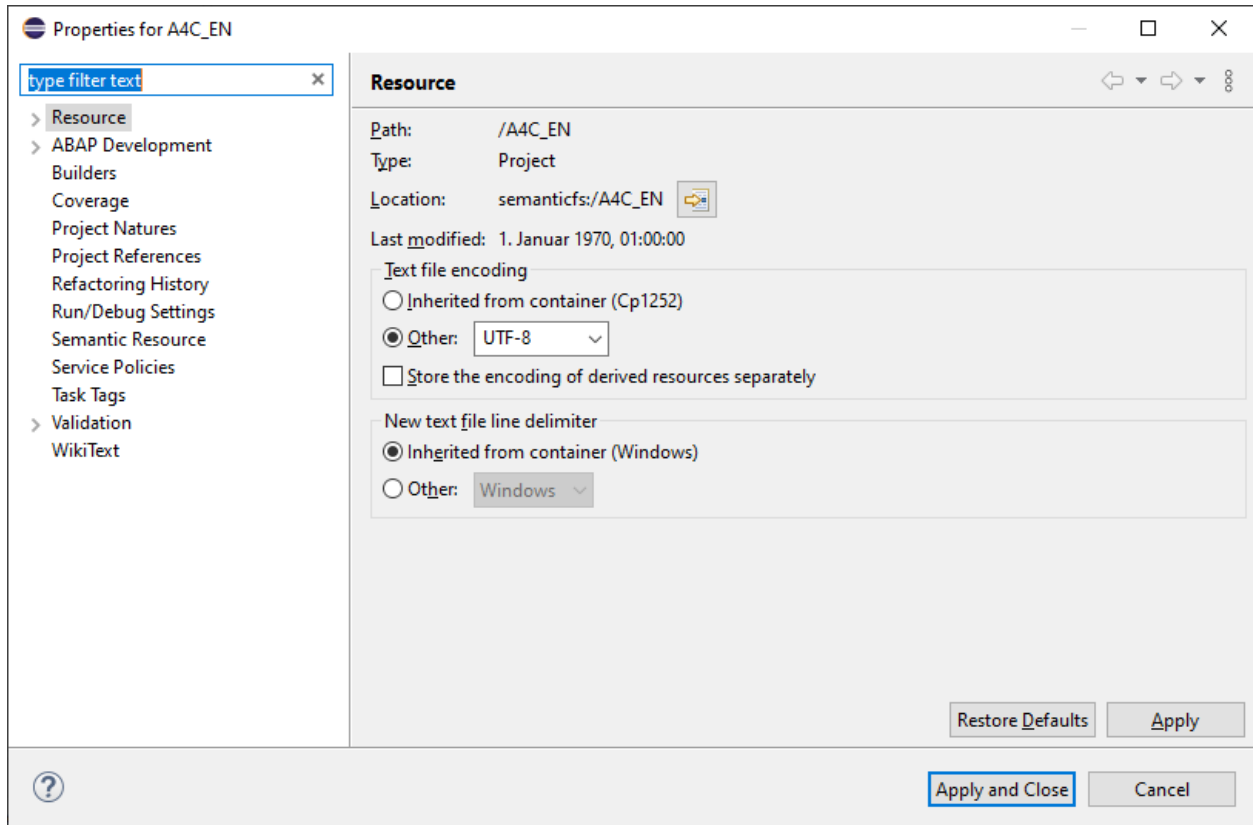


Abbildung 145 Einstieg in die projektspezifischen Einstellungen

6.1.2.1 Externes Debugging

(ABAP Development → Debug)

Mit dieser Option können Sie einstellen, für welchen User das Debugging aktiv ist. Das kann der aktuelle Anwender ("Logon User") oder ein anderer Anwender ("User") sein, wenn Sie ein externes Debugging durchführen möchten.

Breakpoints in Eclipse sind automatisch für alle Zugriffsarten (SAP GUI, ABAP Unit, HTTP, RFC) aktiv. Es gibt kein explizites "externes Debugging" mehr.

6.1.2.2 Pretty Printer

(ABAP Development → Editors → Source Code Editors → ABAP Formatter)

Wie bei Pretty Printer nehmen Sie hier die Einstellungen der Formatierung vor, wenn der Code Formatter (**SHIFT+F1**) ausgeführt wird. Ein Standard hierfür wäre zum Beispiel:

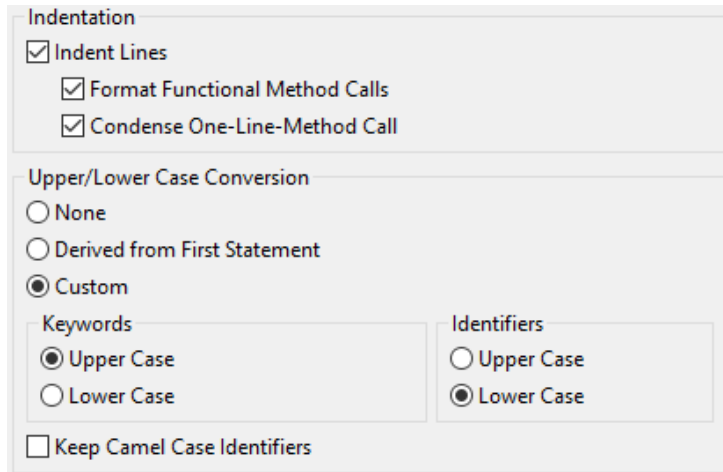


Abbildung 146 Mögliche Einstellungen für den Pretty Printer / ABAP Formatter

(Einstellungen abhängig von der Backend-System-Version, z. B. ist “Keep Camel Case Identifiers” erst in S/4HANA vorhanden)

6.2 Views und Perspektiven

Begriffsdefinitionen: siehe [Kapitel 1](#)

Arbeiten mit den unterschiedlichen Views: siehe [Kapitel 3](#)

6.2.1 Views

Alle Informationen, die Sie sehen und mit denen Sie arbeiten, werden in Views (“Unterbilder” des Bildschirms) dargestellt, z. B. der Project Explorer oder der Editor. Views können dabei beliebig auf den Bildschirm verschoben werden, indem Sie den Tabellenreiter der View anfassen (Maustaste halten, “Drag”) und ziehen.

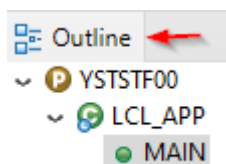


Abbildung 147 Verschieben des Views über die Bezeichnung/Reiter

Während des Verschiebens wird eine Vorschau des neuen Layouts angezeigt.



Abbildung 148 Die Markierungen deuten die Platzierbarkeit des Fensters an

Nach dem Loslassen (“Drop”) wird die View an diese Stelle verschoben.

Views können auch außerhalb des Eclipse-Bildschirms platziert werden und dort existieren. Das ist gerade beim Arbeiten mit mehreren Monitoren sinnvoll.

Wenn Sie die View neben einen anderen Tabellenreiter ziehen, werden die Views gestapelt, das heißt in einer View-Gruppe zusammengefasst.

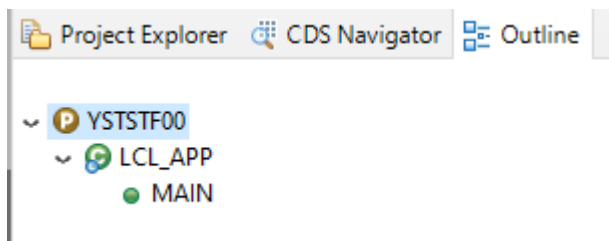


Abbildung 149 Darstellung von gestapelten Views

View-Gruppen können gemeinsam minimiert und wiederhergestellt werden.

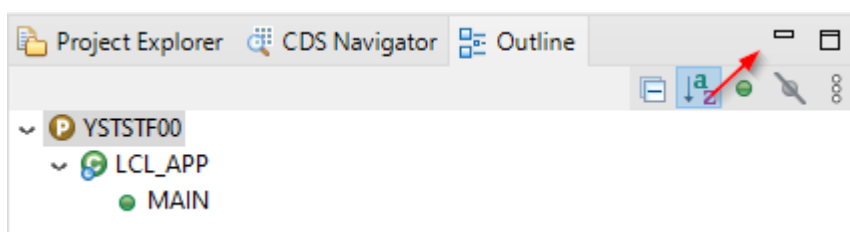


Abbildung 150 Minimieren von View-Gruppen

Ergebnis ist, dass die View-Gruppe minimiert am Rand des Bildschirms angezeigt wird. Über den Druckknopf “Restore” können Sie die View-Gruppe wiederherstellen.

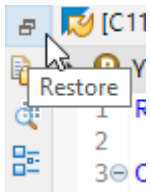


Abbildung 151 Wiederherstellung der View-Gruppen

Nach einem Doppelklick auf einen Tabellenreiter wird die View im Vollbild angezeigt. Dies ist vor allem bei Editor-Views oder großen Dynpros, die im SAP GUI View angezeigt werden, sehr nützlich. Ein erneuter Doppelklick auf den Tabellenreiter verkleinert den View wieder.

Nicht mehr benötigte Views können über das Schließen-Symbol geschlossen werden – so zum Beispiel auch der Feature-Explorer, nachdem Sie das Tutorial durchgearbeitet haben.

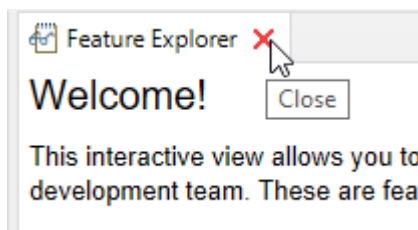


Abbildung 152 Schließen eines Views

Neue oder versehentlich geschlossene Views können Sie nachträglich zu einer Perspektive hinzufügen.

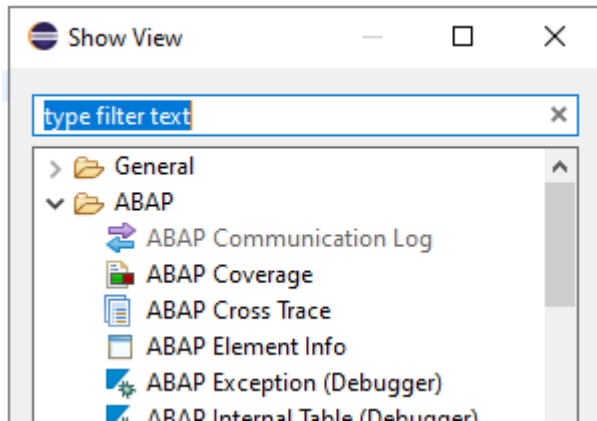


Abbildung 153 Einblenden einer View

So können auch Nicht-ABAP-Views (s. “Empfohlene zusätzliche Views”) der ABAP-Perspektive zugeordnet werden.

6.2.2 Perspektiven

Die Anordnung aller Views auf dem Bildschirm wird in einer Perspektive gespeichert. Für unterschiedliche Aktivitäten werden eigene Perspektiven ausgeliefert, die beliebig angepasst werden können.

In den ABAP Development Tools werden hauptsächlich die Perspektiven ABAP und Debugging verwendet, zwischen den man beliebig wechseln kann.



Abbildung 154 Wechseln zwischen verschiedenen Perspektiven

Tip: Gerade in den ersten Wochen der ADT-Nutzung wird nach einer Debugging-Session gerne vergessen, zur ABAP-Perspektive zurückzukehren.

Wenn Sie Ihre Perspektive “zu sehr” angepasst haben, können Sie über das Menü den Auslieferungszustand der Perspektive wiederherstellen.

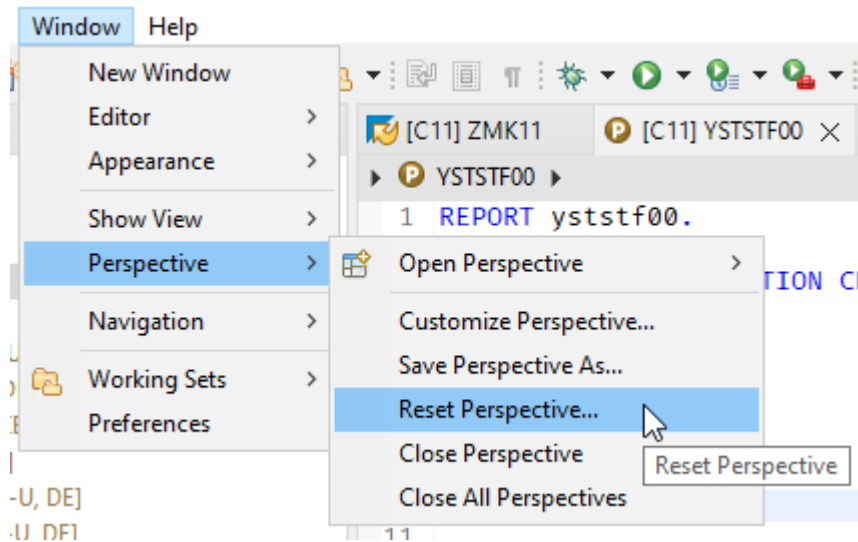


Abbildung 155 Zurücksetzen einer Perspektive

6.2.2.1 Eigene Perspektiven

Sie können auch eigene Perspektiven definieren. Dies ist vor allem dann sinnvoll, wenn Sie mit verschiedenen Monitor-Konfigurationen arbeiten (z. B. zwei Monitoren). Dadurch kann die Größe und Anordnung der Views angepasst werden. Eine eigene Perspektive kann über "Save Perspektive As..." gespeichert werden.

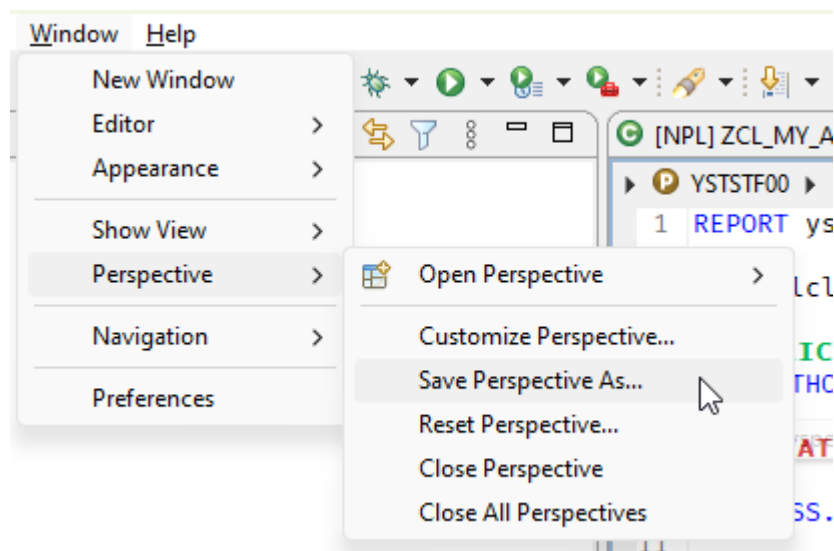


Abbildung 156 Speichern einer Perspektive

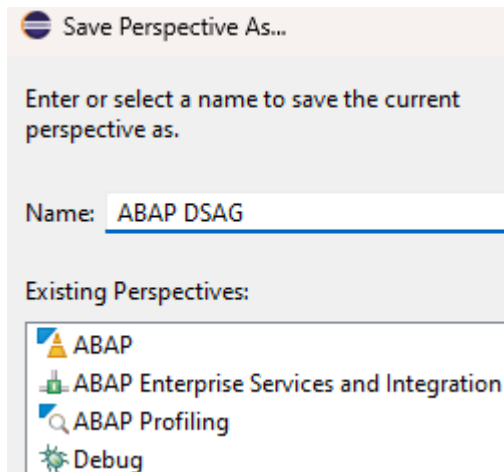


Abbildung 157 Benennung der neuen Perspektive

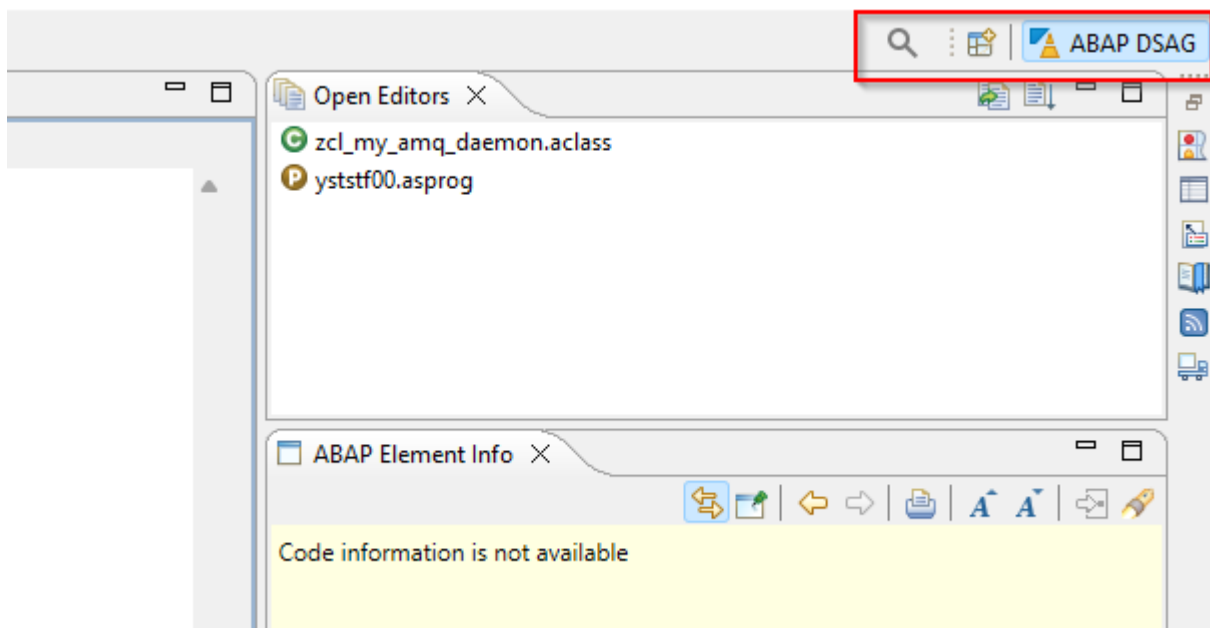


Abbildung 158 Neu Perspektive mit Name

6.3 Empfohlene zusätzliche Views

Über Eclipse bzw. ABAP Development Tools bereits installierte Views:

- ABAP Element Info
- Minimap

Über Eclipse Marketplace installierbare Views

- [Open Editors](#)

6.4 Vorschläge zur Verteilung

Die Einrichtung eines virtuellen Arbeitsplatzes ist so individuell wie die eines realen. Neben den persönlichen Vorlieben kommt es z. B. auch auf die Größe des Monitors an, wie viele Views gleichzeitig sinnvoll angezeigt werden können. Es können deshalb hier nur sehr subjektive Vorschläge gemacht werden.


Project Explorer	Editor	Open Editors	
		ABAP Element Info	
Outline		Minimap	

Abbildung 159 Mögliche Einstellung der ABAP Perspektive

Project Explorer	Editor	Variablen
		Callstack ("Debug")
	Interne Tabellen	

Abbildung 160 Mögliche Einstellung der Debugger Perspektive

7 Plug-ins

7.1 Einführung

Eclipse ist eine integrierte Entwicklungsumgebung (IDE), die im Kern aus vielen kleinen Einheiten – sog. Plug-ins – besteht. Wenn man beispielhaft die Eclipse-Varianten Eclipse IDE for Java Developers oder Eclipse IDE for C/C++ Developers betrachtet, dann sind diese vorkonfigurierten Pakete nur Sammlungen von Plug-ins, die für einen bestimmten Zweck entwickelt worden sind.

Die ABAP Development Tools (ADT) sind kategorisch genau das Gleiche, also eine Sammlung von Plug-ins, und stellen somit ABAP-Entwicklern ein modernes Entwicklungswerkzeug zur Verfügung. Eben genau dieser modulare Aufbau ermöglicht es jetzt jedem Entwickler, eigene Plug-ins zu erstellen, um Eclipse und/oder ADT weiter anzupassen und dadurch z. B. wiederkehrende Aufgaben zu vereinfachen oder Funktionen bereitzustellen, die durch die ADT nicht angeboten werden und nur in der ABAP Workbench zu finden sind.

Solche Plug-ins können aus reinem UI-Code bestehen, wie z. B. das ABAP Favorites Plug-in, welches Funktionen des Easy-Access-Menü (SAP GUI) nach Eclipse bringt. Sie können jedoch auch umfangreicher sein und extra ABAP-Code auf dem SAP-System benötigen. Ein Beispiel dafür ist das ABAP Code Search Plug-in, das vergleichbar ist mit der SAP-GUI-Transaktion CODE_SCANNER.

7.2 Nützliche Open-Source-Plug-ins

7.2.1 Open Editors

Bietet eine neue View, die alle geöffneten Editoren in Eclipse anzeigt. Diese View bietet auch die Möglichkeit, die Sortierreihenfolge der Editoren anzupassen.

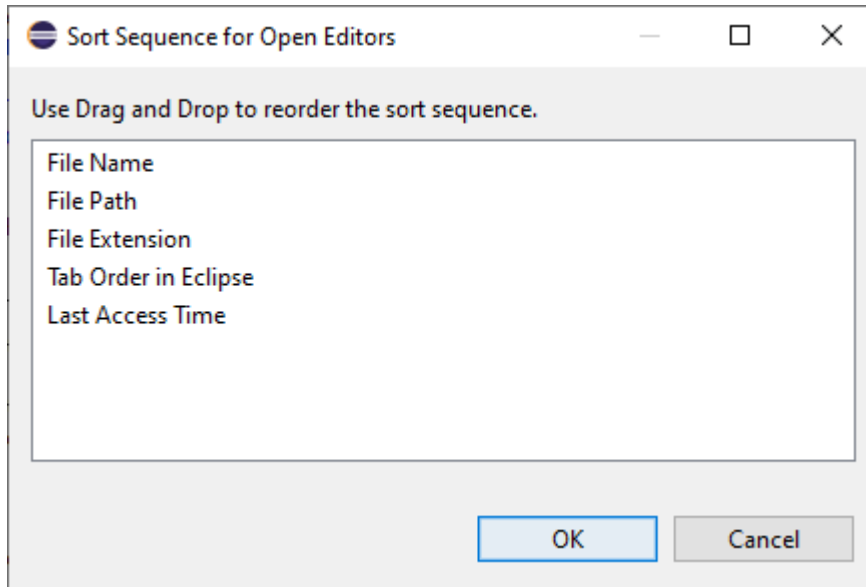


Abbildung 161 Dialog zur Anpassung der Sortierreihenfolge von geöffneten Editoren

Voraussetzungen Eclipse:

- Eclipse IDE for Java Developers

Links:

- Source-Code auf [GitHub](#)
- [Eclipse Marketplace](#)

7.2.2 AnyEdit Tools

Bietet viele Möglichkeiten zum Editieren von Text/Quellcode:

- Umwandlung von Text in Klein-/Großbuchstaben
- Umwandlung von Text von Pascal- zu Camel-Schreibweise
- Sortierung der selektierten Zeilen (alphabetisch, numerisch, nach Zeilenlänge)
- ...

Zusätzlich bietet das Plug-in auch viele Möglichkeiten zum Vergleichen von Text.

- Vergleich eines Editors mit Text in der Zwischenablage
- Vergleich eines Editors mit einem beliebigen anderen Editor
- Vergleich eines Editors mit einer externen Datei

Alle möglichen Operationen stehen über das Kontextmenü eines Editors zur Verfügung.

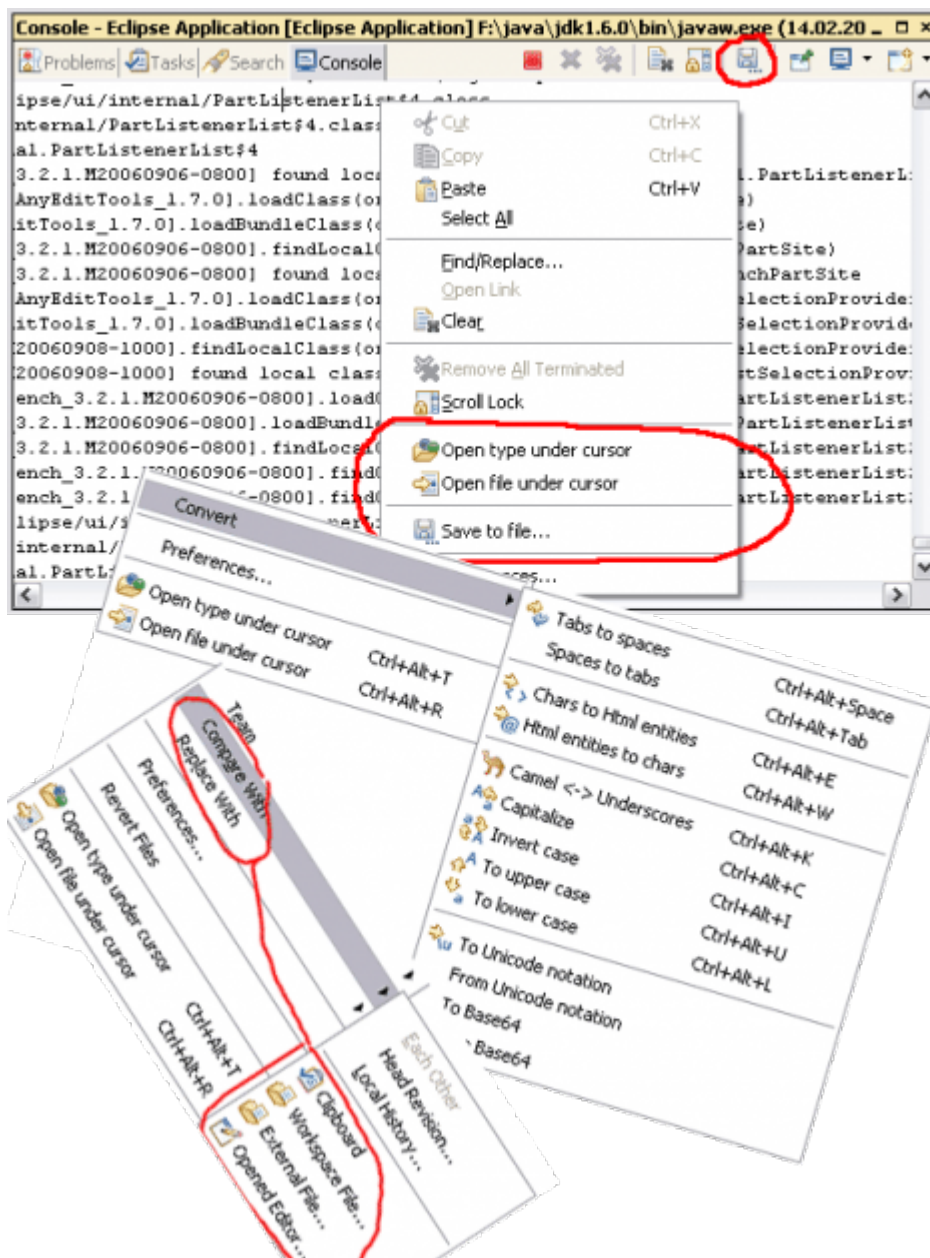


Abbildung 162 Beispiele für verfügbare Operationen im Kontextmenü

Voraussetzungen Eclipse:

- Eclipse IDE for Java Developers

Links

- Source-Code auf [GitHub](#)
- [Eclipse Marketplace](#)

7.2.3 PDE Tools

Dieses Plug-in erweitert Eclipse um nützliche Tools für Plug-in-Entwickler:

- Vorschau von Icon-Dateien direkt im Project Explorer
- Generierung von Java-Konstanten für Icon-Ordner
- Screenshot-Tool für UI-Elemente im Eclipse Workbench

Es gibt jedoch Features, die auch außerhalb der Plug-in-Entwicklung hilfreich sind:

- Erweiterte Historie der Zwischenablage
- Direkter Start eines neuen Eclipse-Fensters mit einem bestimmten Workspace

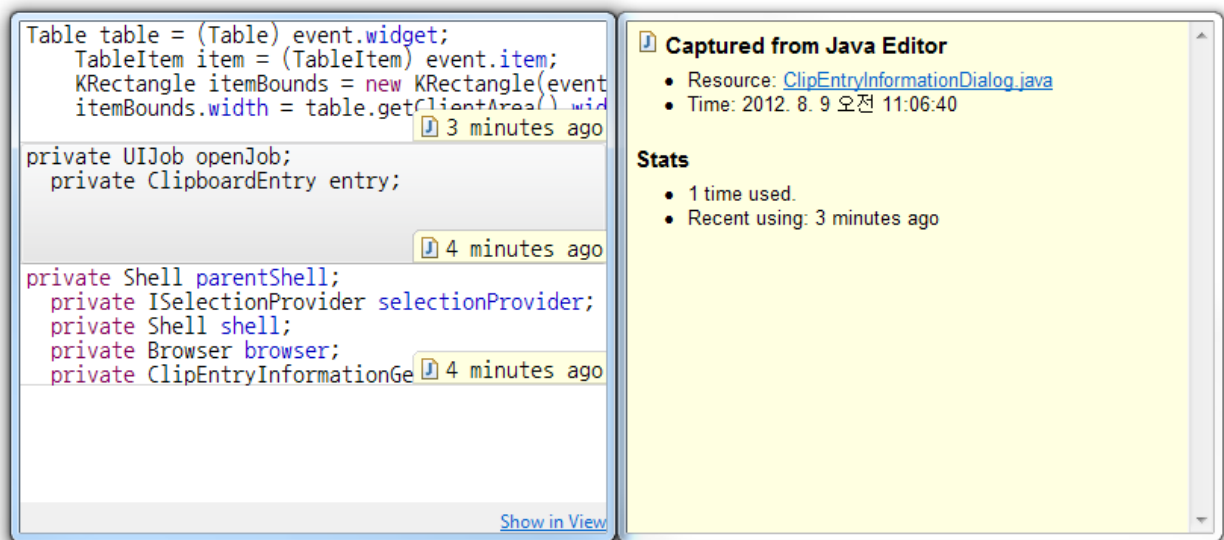


Abbildung 163 Historie für Zwischenablage (Tastenkürzel Strg+Shift+V)

Links:

- Source-Code auf [GitHub](#)
- [Eclipse Marketplace](#)

7.3 Nützliche Open Source ADT Plugins

7.3.1 ABAP Favorites

Das ABAP Favorites Plug-in wurde entwickelt, um die Funktionalität des SAP GUI User Menu abzubilden. In diesem Menü kann jeder Benutzer Transaktionen, Reports oder URLs seinen Favoriten hinzufügen und diese nach seinen persönlichen Vorlieben strukturieren.

Die Plug-in-Installation bringt mit Favorites und Favorite DevObjects zwei neue Views, verfügbar über Windows → Show View → Others. Beide Views bieten eine gefilterte Baumansicht, in der die favorisierten Objekte verwaltet werden können.

Der Unterschied der beiden Views liegt in den Möglichkeiten zur Erstellung der Ordner (Container). Die Favorites View ermöglicht zwei Arten dieser Ordner: "Standard" für Transaktionen, Reports und URL und "DevObject" zur Verwaltung von Entwicklungsobjekten wie Klassen, Funktionsbausteinen, CDS Views usw. Im Falle der Favorite DevObjects können ausschließlich "DevObject"-Ordner erstellt werden. Die Aufteilung dieser Views ermöglicht dem Verwender zu wählen, ob er alle Ordner vermischt oder gemäß der beschriebenen Trennung verwalten möchte.

Vergleicht man den "Standard"- mit dem "DevObject"-Ordner liegt neben den auswählbaren Objekttypen der größte Unterschied darin, dass bei "Standard" durch einen Doppelklick die Objekte ausgeführt werden. Ein Doppelklick in einem "DevObject"-Ordner bewirkt das Öffnen des ausgewählten Objekts.

Plug-ins

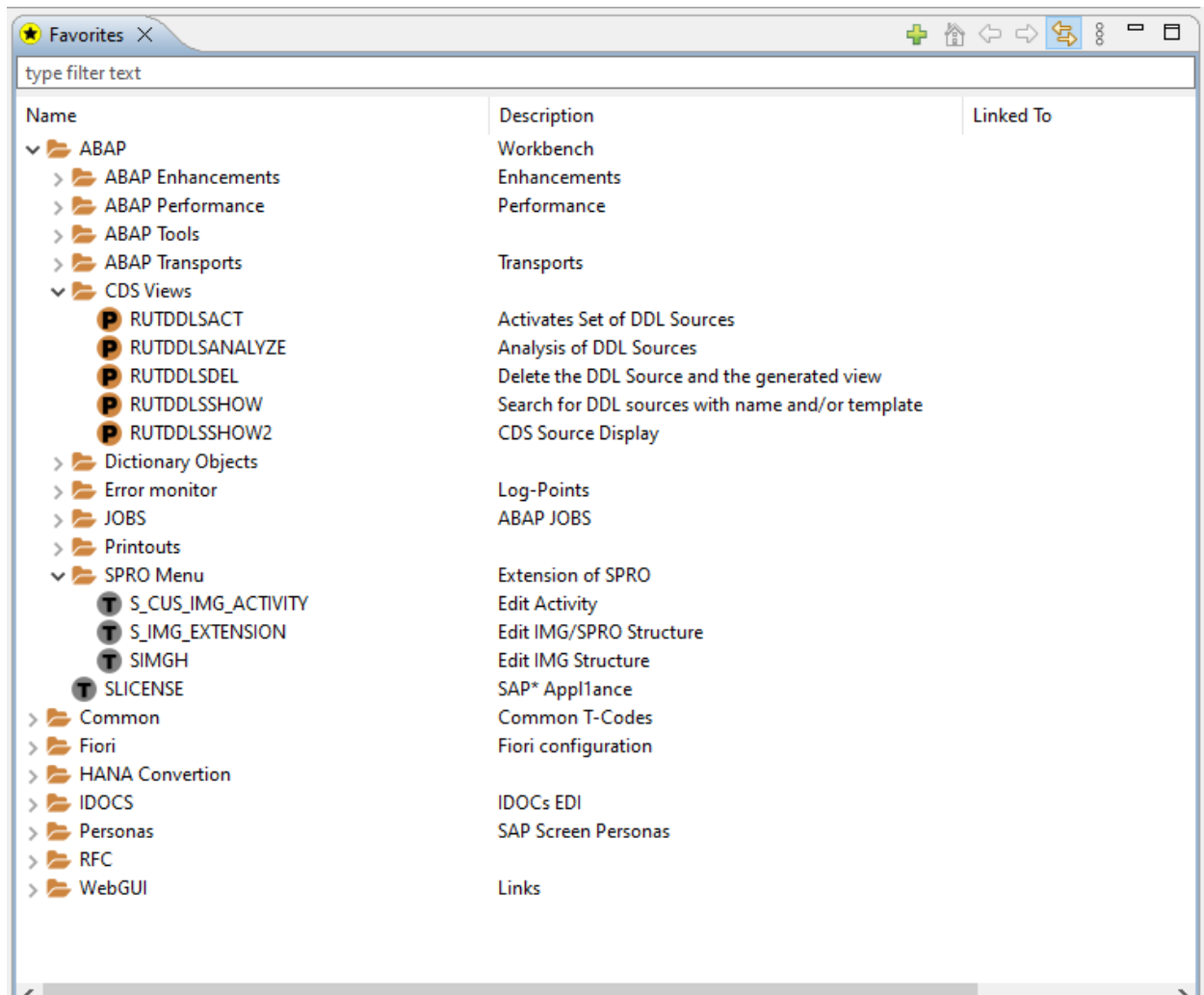


Abbildung 164 ABAP Favorites View

Um neue Objekte den Favoriten hinzuzufügen, können die Kontextmenüs der Favorites View, des ABAP Editors oder des Project Explorers genutzt werden.

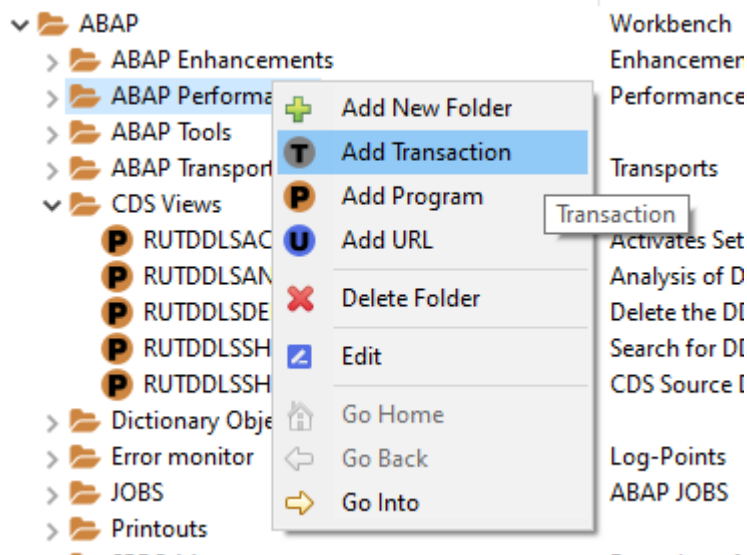


Abbildung 165 Kontextmenü eines Ordners im ABAP Favorites View

Voraussetzungen:

- Eclipse IDE for Java Developers
- ADT

Links:

- Source-Code auf [GitHub](#)
- [Eclipse Marketplace](#)

7.3.2 ABAP Continuous Integration

AbapCI ist ein Open Source Eclipse Plug-in, das verschiedene Continuous Integration (CI) Tools für die ABAP-Entwicklung mit Eclipse bereitstellt. Das Plug-in basiert auf den CI-Funktionen von ADT.

Das Plug-in stellt folgende Funktionen bereit:

- Automatische Unit-Testläufe
- Automatische ATC-Läufe
- Visualisierung des Quellcode-Status auf der Benutzeroberfläche
- Unterschiedliche Farbgebung für jedes ABAP-Projekt
- Automatische Quellcode-Formatierung
- Shortcut für abapGit
- Auslösen von Jenkins aus Eclipse (experimentell)

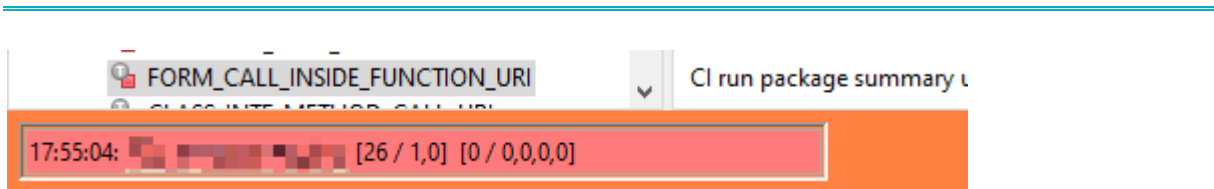
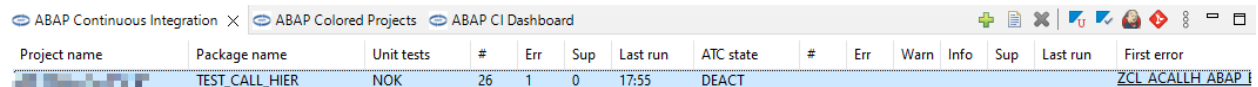


Abbildung 166 Farbige Hervorhebung der Statusleiste pro Projekt + Teststatus



Project name	Package name	Unit tests	#	Err	Sup	Last run	ATC state	#	Err	Warn	Info	Sup	Last run	First error
	TEST_CALL_HIER	NOK	26	1	0	17:55	DEACT							ZCL_ACALLH ABAP

Abbildung 167 Verwaltung von Paketen, für die Unittests und/oder ATC-Prüfläufe eingeplant sind

Weitere Informationen können im GitHub Repository nachgelesen werden.

Voraussetzungen:

- Eclipse IDE for Java Developers (<= 2022-06, Installation mit neueren Versionen aktuell nur mit Workaround möglich; siehe [Issue](#) auf GitHub)
- ADT

Links:

- [Source-Code auf GitHub](#)
- [Eclipse Marketplace](#)

7.3.3 ABAP ADT Extensions

Dieses Plug-in erweitert die ADT um mehrere zusätzliche Funktionalitäten.

7.3.3.1 Automatisches Einloggen in SAP-Systeme

Die Funktionalität “Automatisches Einloggen” ermöglicht dem Entwickler, seine Benutzer/Passwort-Kombinationen innerhalb des Secure Storage von Java zu verwalten. Sicherheitstechnisch sollten diese verschlüsselt werden.

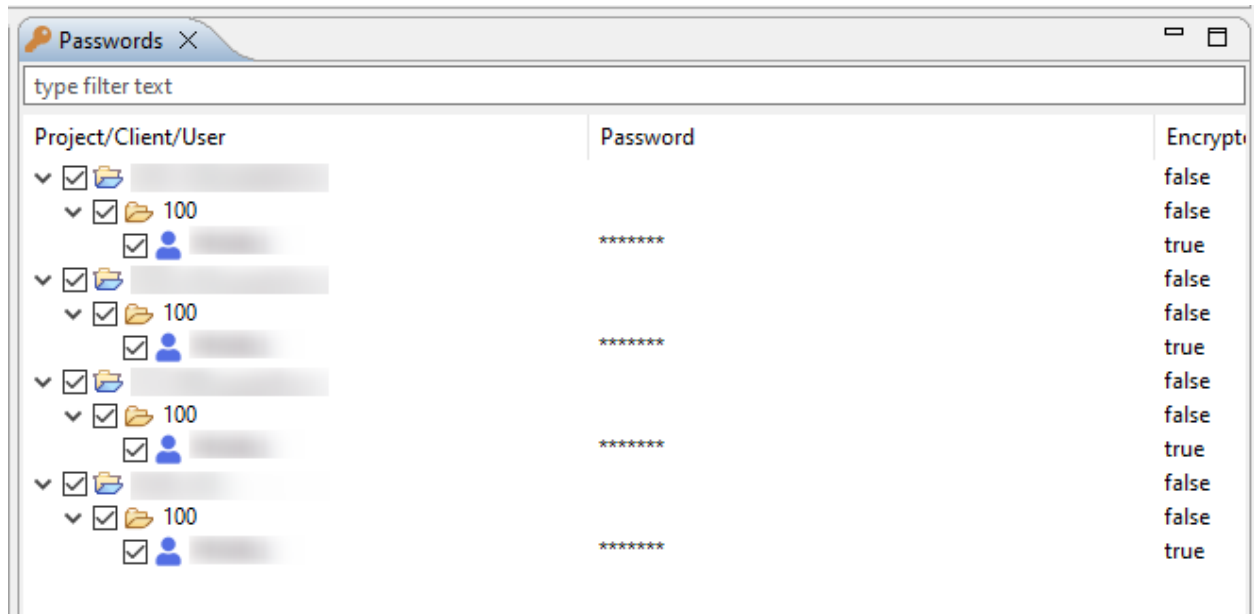


Abbildung 168 View zur Verwaltung der hinterlegten Zugangsdaten von ABAP Systemen

Abhängig von den Einstellungen des Plug-in kann man sich in jedes der ausgewählten On-Premise-SAP-Systeme, die mittels ABAP/BW-Projekten abgebildet sind, automatisch einloggen lassen. Die Passwörter können bei der Erstellung des Projekts oder erst später über die Passwort-View gepflegt werden.

7.3.3.2 Verändern von ABAP-Projekt-Attributen

Durch das Kontextmenü des Project Explorers können für ABAP/BW-Projekte die Projektattribute (Client, User, Sprache) geändert werden. Zudem können die Breakpoint-User gesetzt werden.

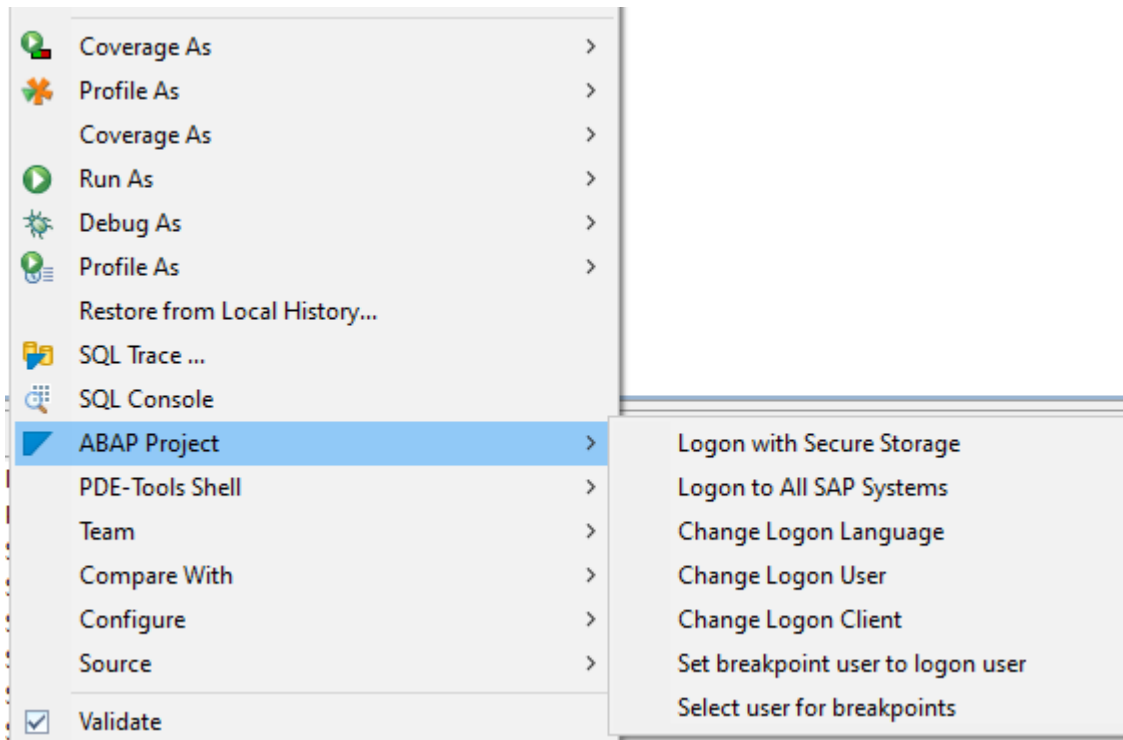


Abbildung 169 Kontextmenü auf Projekt im Project Explorer

7.3.3.3 Eingabefeld zur Ausführung von Transaktions-Codes

Nach der Installation erscheint in Eclipse im unteren rechten Bereich eine Toolbar mit einem Eingabefeld für Transaktions-Codes. Das Feld kann per Maus oder über den Shortcut **SHIFT+F8** bedient werden. Nach der Eingabe des Codes und dem Bestätigen per Enter wird die Transaktion im momentan aktiven Projekt ausgeführt.



Abbildung 170 Statusleiste im Eclipse-Fenster

Voraussetzungen:

- Eclipse IDE for Java Developers
- ADT

Links:

- [Source-Code auf GitHub](#)
- [Eclipse Marketplace](#)

7.3.4 ADT Classic Outline

Dieses Plug-in fügt Ihrer Oberfläche eine neue View namens "Classic Outline" hinzu, die gewissermaßen die SE80-Objektliste abbildet. In den meisten Fällen kann diese Ansicht die eingebaute ADT Outline ersetzen. Die angezeigte Objektliste kann gefiltert werden und erlaubt die Ansicht der Objektattribute, ähnlich der eingebauten ADT Outline. Durch einen Doppelklick (bzw. einfachen Klick abhängig von den Einstellungen) kann in das selektierte Objekt navigiert werden.

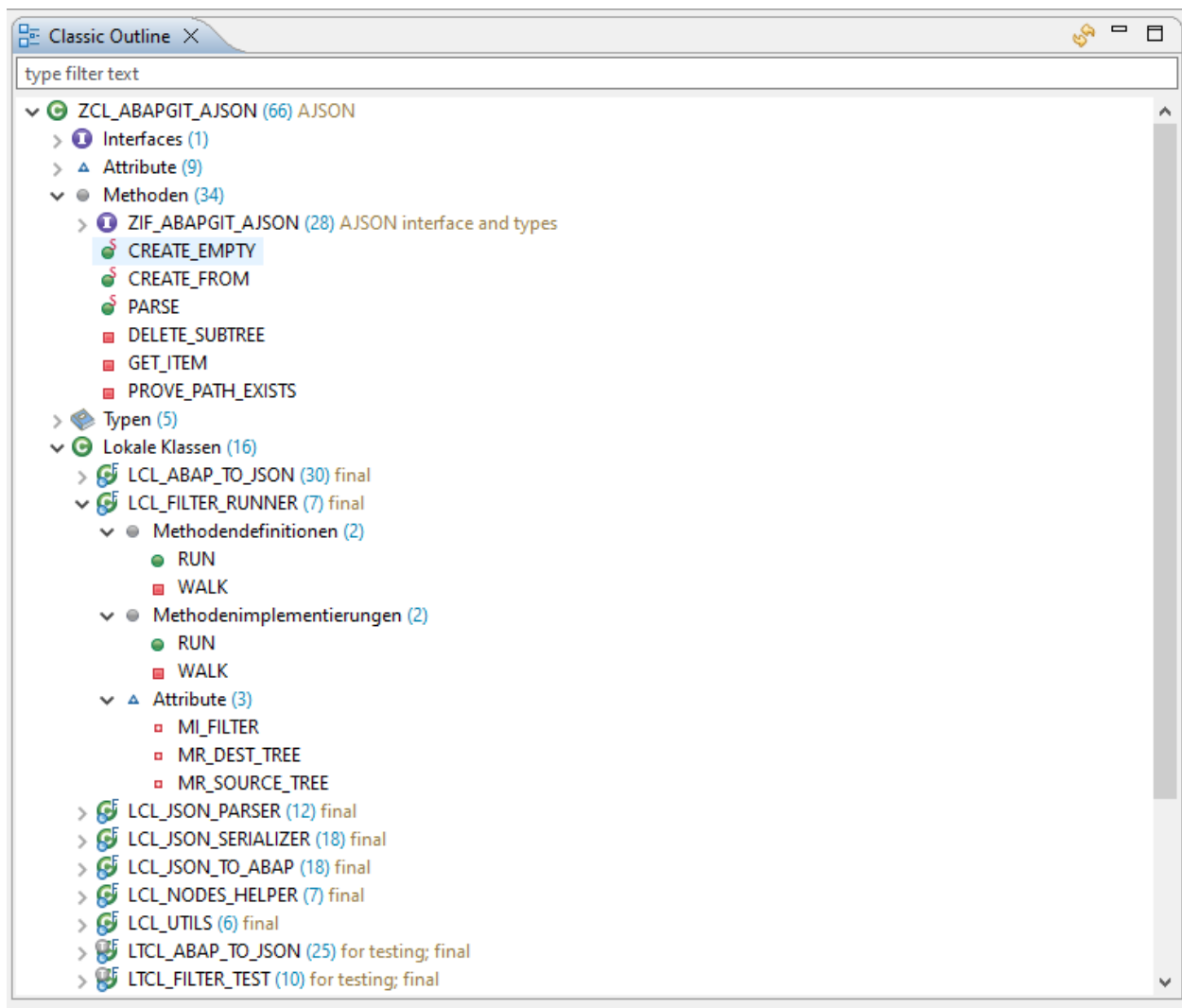


Abbildung 171 Classic Outline View

Voraussetzungen:

- Eclipse IDE for Java Developers
- ADT

Voraussetzungen ABAP:

- SAP NetWeaver 7.40 SP08 oder neuer
- abapGit repository [ADT Classic Outline Backend](#) muss installiert sein

Links:

- Source-Code auf GitHub: [Frontend](#) und [Backend](#)
- [Eclipse Marketplace](#)

7.3.5 ABAP Quick Fix

Quick Fixes sind Teil der Eclipse IDE. Im ADT-Standard werden sie im Backend-System verarbeitet und können vom Nutzer bei Bedarf mittels Shortcut **CTRL+1** verwendet werden. Das ABAP Quick Fix Plug-in bietet zusätzliche Quick Fixes, die direkt von der Eclipse-Umgebung prozessiert werden.

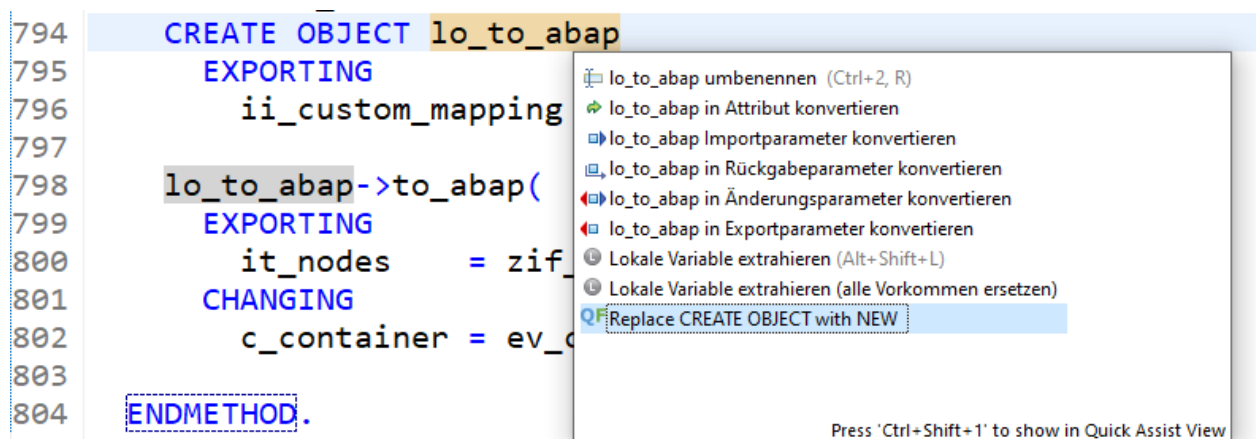


Abbildung 172 ABAP Code vor Quick-Fix-Ausführung

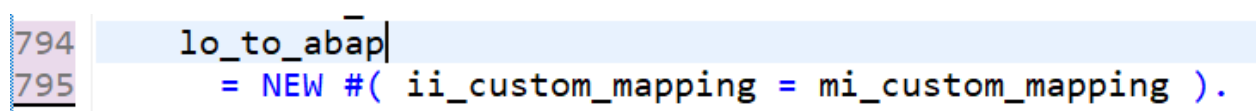


Abbildung 173 ABAP Code nach Quick-Fix-Ausführung

Einen Auszug der verfügbaren Features finden Sie in der folgenden Liste:

- Ersetze READ TABLE durch ASSIGN, REF#, Table Expression oder line_exists.
- Ersetze CALL METHOD durch den direkten Aufruf.
- Ersetze MOVE durch die direkte Zuweisung.
- Ändere APPEND TO in APPEND VALUE#() TO.
- Ersetze CREATE OBJECT durch NEW.
- Entferne "full line comments" vom Statement.
- Unterlasse die Selbstreferenz ME->.
- Ersetze die Operatoren EQ, NE, GT, GE, LT, LE mit =, <>, >, >=, <, <=
- Rücke Operatoren im markierten Bereich entsprechend ein.
- Rücke TYPE und LIKE im Deklarationsblock der Variablen entsprechen ein.

Voraussetzungen:

- Eclipse IDE for Java Developers
- ADT

Links:

- Source-Code auf [GitHub](#)
- [Eclipse Marketplace](#)

7.3.6 ABAPQuickFixS4Conversion

Dieses Plug-in ist ein sehr schönes Beispiel für die Zusammenarbeit von Entwicklern innerhalb der Community. ABAPQuickFixS4Conversion ist eine Erweiterung des ABAP Quick Fix Plug-in von SAP und ergänzt dieses um folgende Funktionalitäten:

- Konvertiere SELECT SINGLE nach SELECT ... UP TO 1 ROWS ... ORDER BY
- Passe für jede beliebige Tabelle die custom ORDER BY-Liste an
- Ändere SELECT SINGLE auf den modernen SQL-Stil
- Konvertiere SELECT/ENDSELECT in den modernen SQL-Stil
- Transformiere MOVE_CORRESPONDING nach CORRESPONDING #()

```

38
39 select single tkonn tposn from wbit into (tkonn, tposn)
40 where t
41 and t
42
43
44
45
46
47

```

Quick-Fixes: QF Remove all ABAP Comments, SC Replace select single with select up to one rows

Abbildung 174 Beispiel für Quick-Fix-Verfügbarkeit bei einer SELECT-Anweisung

```

38
39 select tkonn, tposn
40 from wbit
41 into (@tkonn, @tposn)
42 up to 1 rows
43 where tkonn = @tkonn
44 and tposn = @tposn
45 order by doc_type, doc_nr, doc_year, item, sub_item.
46 endselect.
47

```

Abbildung 175 SELECT-Anweisung nach Anwendung des Quick Fix

Voraussetzungen:

- Eclipse IDE for Java Developers
- ADT
- ABAP Quick Fixes plugin

Links:

- Source-Code auf [GitHub](#)
- [Eclipse Marketplace](#)

7.3.7 ABAP Tags

Das ABAP Tags Plug-in ermöglicht es, Tags zu erstellen, welche dann zu beliebigen Entwicklungsobjekten hinzugefügt werden können. Die Tags und deren zugeordneten Objekte werden dabei auf dem jeweiligen ABAP-System persistiert. Dies erleichtert den Zugriff auf getaggte Objekte durch andere Benutzer. Generell erlaubt das Plug-in, Tags entweder im globalen oder im benutzerspezifischen Geltungsbereich anzulegen. Benutzerspezifische Tags können dabei auch mit anderen geteilt werden und dadurch die Zusammenarbeit erleichtern.

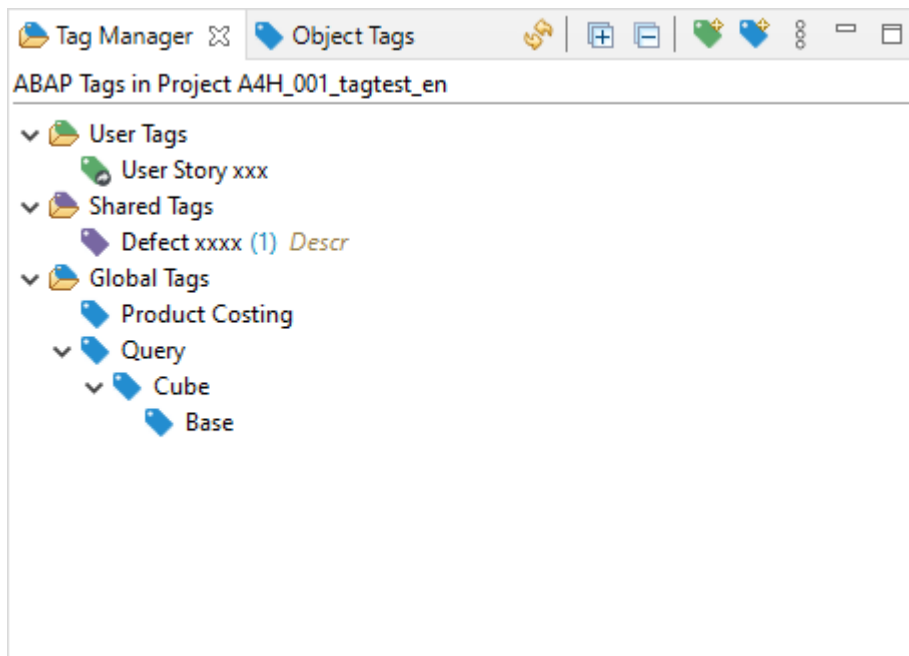


Abbildung 176 View “Tag Manager”

Das Taggen von Objekten ist intuitiv über das Kontextmenü aus dem Editor oder dem Project Explorer heraus möglich. Die getaggten Objekte können dann entweder aus dem View “Tag Manager” mittels Kontextmenü-Aktion, oder über die in den “Search”-Dialog integrierten “ABAP Tagged Object Search” gesucht und angezeigt werden.

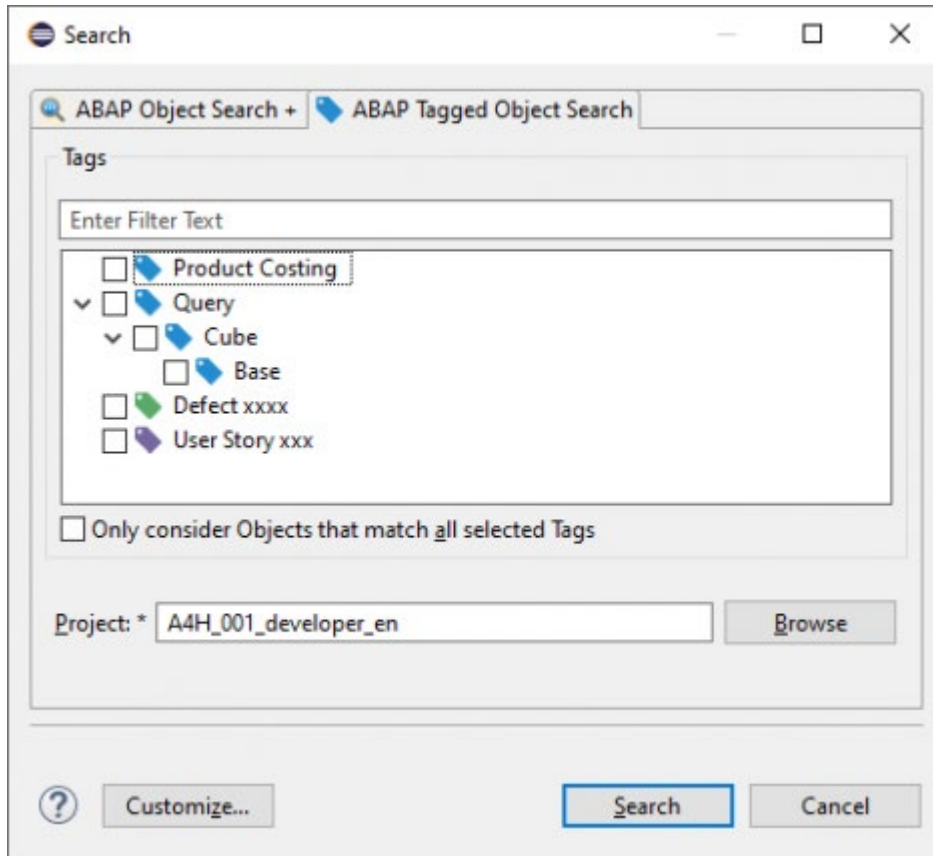


Abbildung 177 Search-Dialog mit Seite “ABAP Object Search”

Voraussetzungen Eclipse:

- Eclipse Platform Runtime oder Eclipse IDE for Java Developers
- ADT

Voraussetzungen ABAP:

- SAP NetWeaver 7.40 SP08 oder neuer
- abapGit repository [abap-tags-backend](#) muss installiert sein

Links:

- Source-Code auf [GitHub](#)
- [Eclipse Marketplace](#) (Marketplace Client muss installiert sein)

7.3.8 ABAP Search and Analysis Tools

Dieses Plug-in erweitert die ADT um weitere Such- und Analysefunktionen für die folgenden Objekttypen:

- Klasse/Interface
- Datenbanktabelle/-view
- CDS View

Die Suchfunktionen sind in den Eclipse-Such-Dialog integriert (**Strg+H**). Die Bedienung ist dabei ähnlich wie beim Dialog “Open ABAP Development Object” (**Strg+Shift+A**). Den Objekttyp kann man mittels Dropdown ändern. Dieser steuert unter anderem die verfügbaren Filter im Feld “Search Filters”.

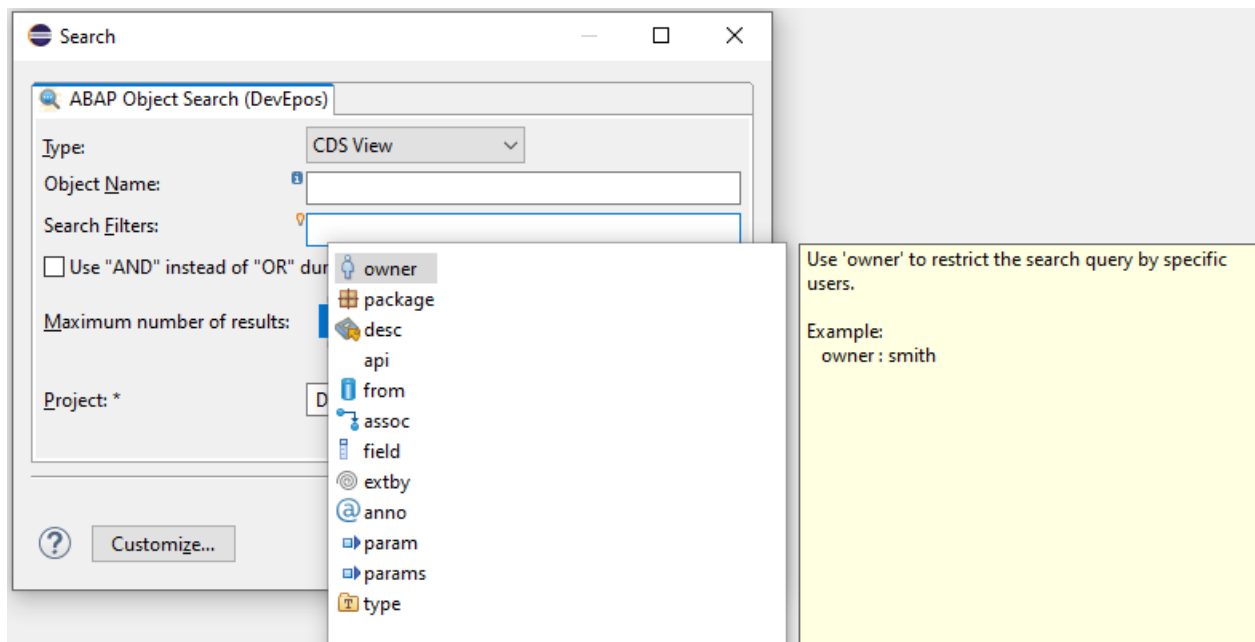


Abbildung 178 Search-Dialog auf Seite “ABAP Object Search”

Zusätzlich zu den Suchfunktionen stellt das Plug-in noch den View “CDS Analyzer” zur Verfügung, der die folgenden Analysen auf CDS Views erlaubt:

- Top-Down
- Auswertung auf alle verwendeten Entitäten eines CDS View
- Verwendungsnachweis von Datenbankentitäten als Datenquelle (“*select from*” oder “*association*”)
- Analyse auf Feldebene

- Top-Down (Herkunftsermittlung)
- Bottom-up (Verwendung eines Feldes in Feldern anderer CDS Views)

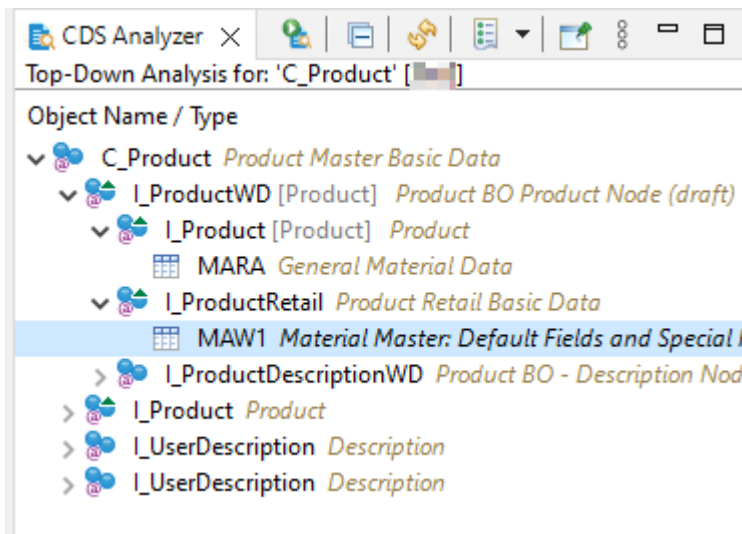


Abbildung 179 View “CDS Analyzer” - Top-Down-Analyse

Voraussetzungen Eclipse:

- Eclipse Platform Runtime oder Eclipse IDE for Java Developers
- ADT

Voraussetzungen ABAP:

- SAP NetWeaver 7.40 SP08 oder neuer
- abapGit repository [abap-search-tools](#) muss installiert sein

Links:

- Source-Code auf [GitHub](#)
- [Eclipse Marketplace](#) (Marketplace Client muss installiert sein)

7.3.9 ABAP Code Search

Dieses Plug-in bringt die bekannte SAP-GUI-Transaktion CODE_SCANNER nach Eclipse. Wie der Name schon vermuten lässt, ist die “ABAP Code Search” auch im Eclipse-Search-Dialog integriert. Neben der Verwendung von regulären Ausdrücken gibt es auch spezielle Suchmodi wie z. B. *Single Pattern mode* oder *Sequential Matching*. Weitere Merkmale der Suche sind:

- Parallele Ausführung (optional pro Benutzer steuerbar)
- Suche kann jederzeit gestoppt werden
- Komplette Systemsuche möglich, da auf dem Anwendungs-Server immer nur kleine Pakete verarbeitet werden
- Tags können zur Objektauswahl verwendet werden → erfordert Installation des ABAP Tags Plug-in

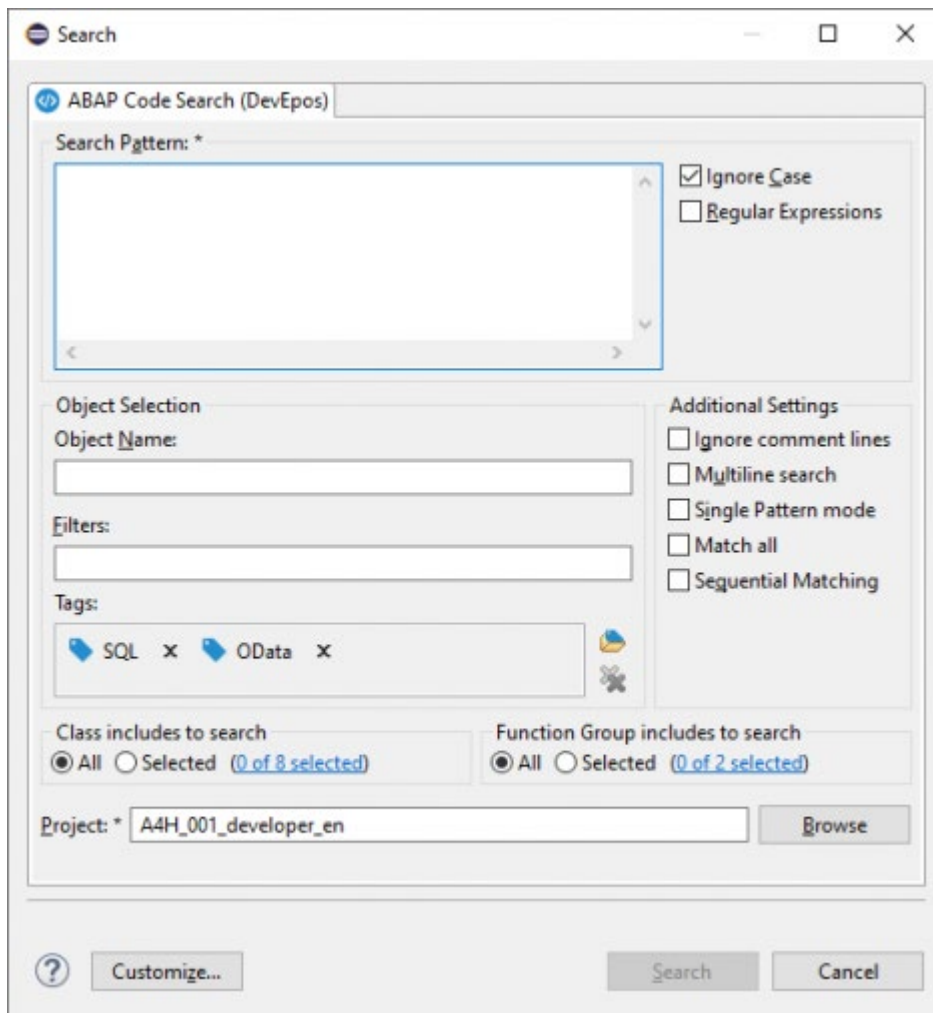


Abbildung 180 Search-Dialog mit “ABAP Code Search”-Seite

Voraussetzungen Eclipse:

- Eclipse Platform Runtime oder Eclipse IDE for Java Developers
- ADT

Voraussetzungen ABAP:

- SAP NetWeaver 7.40 SP08 oder neuer
- abapGit repository [abap-code-search-tools](#) muss installiert sein

Links:

- Source-Code auf [GitHub](#)
- [Eclipse Marketplace](#) (Marketplace Client muss installiert sein)

7.3.10 abapGit Eclipse Plug-in

Stellt die Funktionen der abapGit-SAP-GUI-Transaktion als Eclipse Plug-in bereit. Der volle Funktionsumfang ist aktuell (2022) jedoch nur über die SAP-GUI-Transaktion gewährleistet.

Voraussetzungen Eclipse:

- Eclipse IDE for Java Developers
- ADT

Voraussetzungen ABAP:

- SAP NetWeaver 7.50 oder neuer
- Vollständige [abapGit](#)_Installation
- abapGit repository [ADT_Backend](#) muss installiert sein

Links:

- Source-Code auf [GitHub](#)
- Installation über Update-Site <https://eclipse.abapgit.org/updatesite/>

7.4 Eigene ADT Plug-ins entwickeln

7.4.1 Voraussetzungen

Da Plug-ins für Eclipse in Java zu entwickeln sind, empfiehlt es sich, dort schon einige Kenntnisse zu haben. Der notwendige Kenntnisgrad richtet sich dabei nach dem Plug-in, das man entwickeln will.

7.4.2 Einrichtung der Entwicklungsumgebung

7.4.2.1 Installation Eclipse for RCP/RAP Development

Um Plug-ins für Eclipse zu entwickeln, benötigt man eine bestimmte Variante der Eclipse-Plattform: *Eclipse IDE for RCP and RAP Developers* (RAP = Remote Application Platform). Diese Variante bietet ein vollständiges Toolset, um sowohl Plug-ins für Eclipse als auch Rich-Client-Anwendungen (RCP) auf Basis von Eclipse zu entwickeln. Sie kann direkt von eclipse.org bezogen werden.

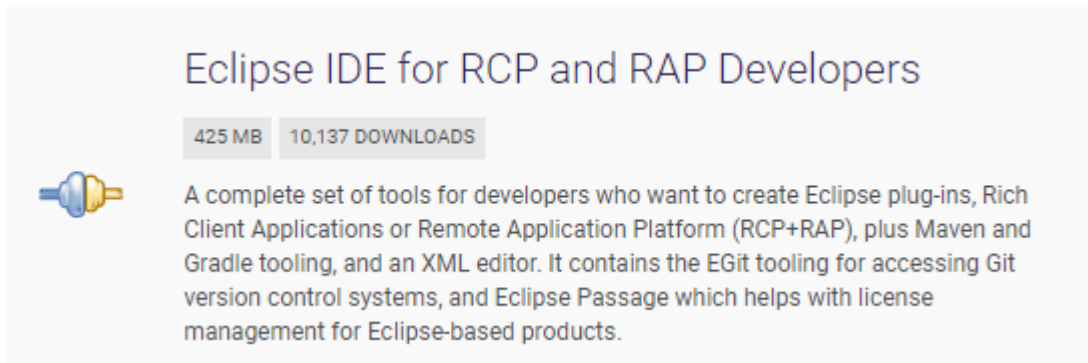


Abbildung 181 Eclipse Bundle “Eclipse IDE for RCP and RAP Developers”

Des Weiteren wird eine Installation des Java Development Kits (JDK) benötigt. Dieses kann z. B. von den folgenden Quellen bezogen werden:

- <https://adoptium.net/de/>
- <https://openjdk.org/>
- <https://sap.github.io/SapMachine/>

Hinweis: In den aktuelleren Eclipse-Versionen wird dieses bereits mitgeliefert.

Welche Java-Version?

Seit Eclipse v2020-09 bzw. ADT v3.16 ist Java 11 die Mindestvoraussetzung, und somit sollte auch das JDK in mindestens dieser Version installiert werden.

7.4.2.2 Installation von ADT

Als nächsten Schritt muss ADT in Eclipse installiert werden, da sonst nicht gegen das ADT SDK entwickelt werden kann.

Um die bestmögliche Kompatibilität zu haben, empfiehlt es sich, ADT immer in der gleichen Version wie Eclipse zu installieren. Die neueste ADT-Version kann hierbei über <https://tools.hana.ondemand.com/latest> bezogen werden. Für ältere Versionen von ADT muss einfach das *latest* im Pfad gegen die gewünschte Eclipse-Version ausgetauscht werden (für Eclipse 2020-09 wäre es z. B.: <https://tools.hana.ondemand.com/2020-09>).

7.4.2.3 Installation nützlicher Plug-ins (optional)

Neben der Eclipse-Installation empfiehlt es sich, noch folgende Plug-ins zu installieren:

ENHANCED CLASS DECOMPILER

Marketplace Link: <https://marketplace.eclipse.org/content/enhanced-class-decompiler>.

Dieses Plug-in erlaubt es, kompilierten Source-Code lesbar anzuzeigen. Es ist sogar möglich, in solchen de-kompilierten Klassen Haltepunkte zu setzen und den Code zur Laufzeit zu analysieren.

WEB DEVELOPER TOOLS

Marketplace Link: <https://marketplace.eclipse.org/content/eclipse-web-developer-tools-0/promo>.

Wenn man für sein Plug-in auch eine Hilfe anbieten möchte, erweitert dieses Plug-in Eclipse um Editoren mit Syntax-Highlighting für die typischen Web-Dateiendungen (css, html etc.).

WINDOWBUILDER

Marketplace Link: <https://www.eclipse.org/windowbuilder/>

Das Erstellen von GUI-Elementen, wie z. B. Dialoge oder eigene Sichten, kann zeitweise sehr aufwendig sein. Der WindowBuilder kann dabei Abhilfe schaffen und erlaubt es, GUI-Elemente mit Hilfe eines grafischen Editors zu erstellen.

7.4.3 Wichtige Konzepte/Artefakte

7.4.3.1 *Plug-in*

Ein Plug-in wird verwendet, um Code zu einer modularen, erweiterbaren und gemeinsam nutzbaren Einheit zusammenzufassen. Die gesamte Eclipse-Anwendung besteht aus vielen solcher Plug-ins.

7.4.3.2 *Feature*

Ein Feature dient zur Gruppierung von einem oder mehreren Plug-ins zu einer einzigen installierbaren und aktualisierbaren Einheit.

7.4.3.3 *Update-Site*

Update-Sites werden verwendet, um Features zu organisieren und zu exportieren, damit sie in Eclipse-Anwendungen installiert werden können.

7.4.4 Erstellung eines Plug-in-Projektes

Ein neues Plug-in-Projekt kann über File → New → Plug-in Project erstellt werden. Daraufhin öffnet sich der Plug-in Project Wizard:

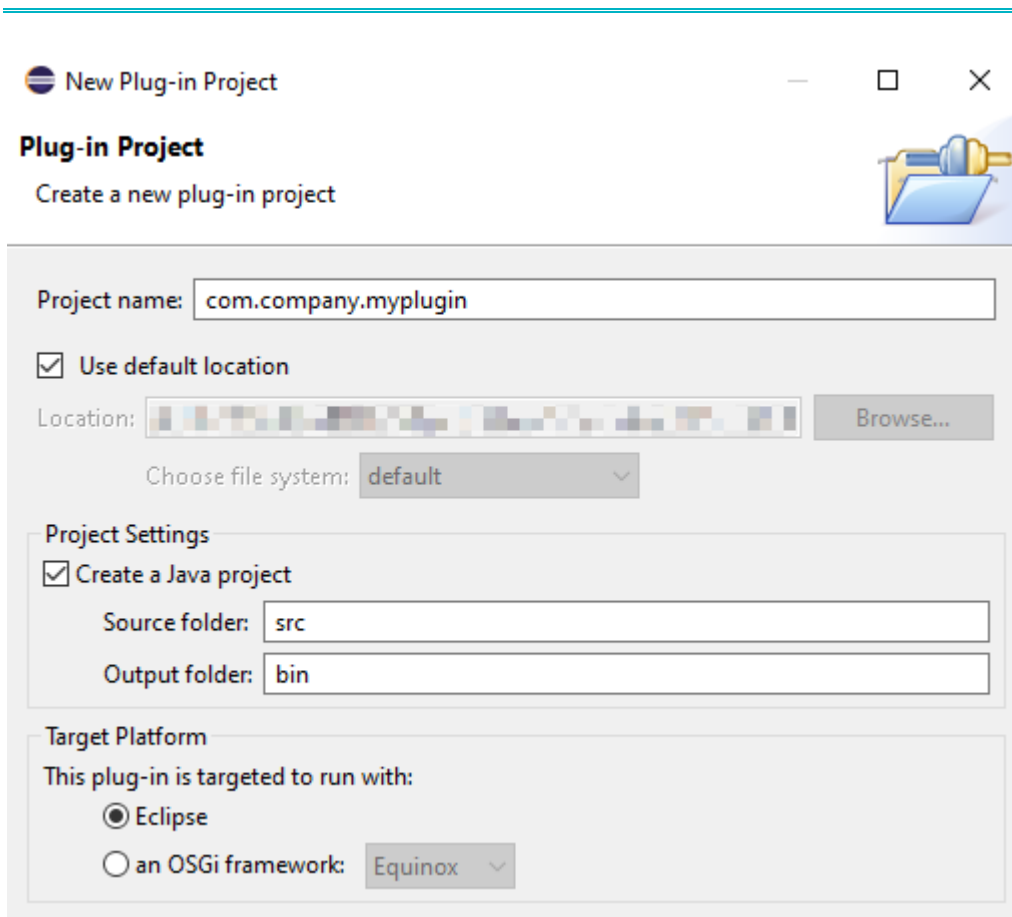


Abbildung 182 Plug-in Project Wizard – Einstieg

Hier muss zunächst ein Name für das Projekt vergeben werden. Beim Namen wird die sog. *Reverse-Naming-Domain-Notation* empfohlen (z. B. `com.company.myplugin`), jedoch kann hier auch jede beliebige andere Namenskonvention gewählt werden.

Standardmäßig wird ein Plug-in Projekt immer als Java-Projekt angelegt weil die meisten Plug-ins Code beisteuern. Diese Option kann jedoch auch abgewählt werden, z. B. für Plug-ins, welche nur Dokumentation bereitstellen.

Da in diesem Guide explizit auf die Plug-in-Entwicklung für Eclipse eingegangen wird, ist die *Zielplattform* immer Eclipse.

Durch den Klick auf *Next* geht es auf die nächste Seite des Wizards.

New Plug-in Project

Content
Enter the data required to generate the plug-in.

Properties

ID:

Version:

Name:

Vendor:

Execution environment:

Options

Generate an activator
Activator:

This plug-in will make contributions to the UI

Enable API analysis

Rich Client Application

Create a rich client application? Yes No

Abbildung 183 Plug-in Project Wizard – Inhalt

Hier werden die Plug-in-spezifischen Eigenschaften erfasst. Für die ID wird empfohlen, den Projektnamen zu verwenden, verpflichtend ist dies jedoch nicht. Die Version muss dem Muster *major.minor.micro.qualifier* folgen. Der *.qualifier*-Teil ist dabei optional. Er wird beim Build durch einen Zeitstempel ersetzt (z. B. 1.3.0.202205011550).

Die Felder *Name* und *Vendor* sind übersetzbar und repräsentieren den Plug-in-Namen und dessen Anbieter.

Beim *Execution Environment* ist die minimal erforderliche Java-Version einzutragen. Für die ADT Plug-ins ist aktuell Java 11 die Mindestvoraussetzung und sollte demnach auch für eigene Plug-ins gesetzt werden.

Wenn die Option *Generate an activator* gesetzt ist, wird eine *Activator*-Klasse generiert. Solch eine Klasse kann pro Plug-in genau einmal existieren und ist nur notwendig, wenn Aktivitäten beim Starten bzw. beim Stoppen des Plug-in notwendig sind. Die Option *This plug-in will make contributions to the UI* regelt die verfügbaren Templates, die auf der nächsten Seite des Wizard ausgewählt werden können.

Als letzten optionalen Schritt kann auf der nächsten Seite noch ein Template ausgewählt werden. Templates existieren zum Beispiel für das Erstellen von eigenen Views oder Editoren.

Nach Abschluss des Wizard mittels *Finish* wird das Plug-in-Projekt an der ausgewählten Stelle im Dateisystem generiert und anschließend im Eclipse Workspace angezeigt.

7.4.4.1 Struktur eines Plug-In-Projektes

Ein Plug-in-Projekt hat immer den folgenden Aufbau. Die Datei *plugin.xml* und der Ordner *OSGI-INF* sind dabei optional und existieren nur, wenn die Notwendigkeit besteht.

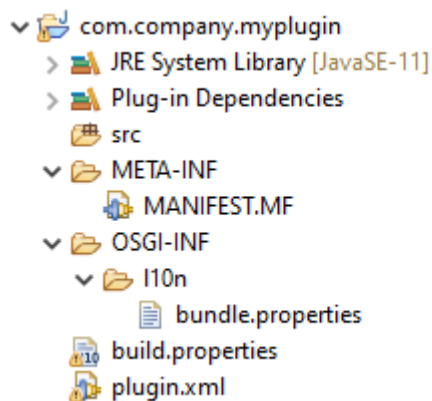


Abbildung 184 Plug-in-Projekt im Project Explorer View

Die wichtigsten Dateien sind *manifest.mf*, *build.properties* und *plugin.xml*. Öffnet man eine dieser drei Dateien, wird standardmäßig der Plug-in-Manifest-Editor geöffnet. Dieser Editor ermöglicht die Pflege aller Metadaten eines Plug-in, welcher der Editor in folgende Bereiche untergliedert:

- **Overview**
Diese Sicht dient als genereller Einstiegspunkt. Sie erlaubt die Pflege der grundlegenden Plug-in-Daten wie Name, Version etc. und den Absprung zu den anderen Sichten.
- **Dependencies**
Hier müssen alle Plug-ins aufgelistet werden, die in diesem Plug-in benötigt werden.

- **Runtime**
Dient zur Spezifikation der Java-Pakete, die für andere Plug-ins sichtbar sein sollen.
Für jedes Paket kann außerdem noch ein API-Status gesetzt werden.
- **Extensions**
Hier erfolgt die eigentliche Erweiterung von Eclipse um z. B. neue Menüs, Commands, Views etc.
- **Extension Points**
Definition der Erweiterungspunkte, die dieses Plug-in für andere bereitstellt.
- **Build**
Konfiguration, welche Dateien im Build-Ergebnis enthalten sein sollen.

7.4.5 Erstellung eines Feature-Projektes

Ein neues Feature-Projekt kann über File → New → Feature Project erstellt werden. Daraufhin öffnet sich der Feature Project Wizard:

The screenshot shows the 'New Feature' wizard in Eclipse. The title bar says 'New Feature'. The main heading is 'Feature Properties' with a sub-heading 'Define properties that will be placed in the feature.xml file'. The 'Project name' field contains 'com.company.myfeature'. The 'Use default location' checkbox is checked. The 'Location' field is empty, with a 'Browse...' button to its right. Below this is a 'Choose file system' dropdown menu set to 'default'. The 'Feature properties' section contains several fields: 'Feature ID' (com.company.myfeature), 'Feature Name' (Myfeature), 'Feature Version' (1.0.0.qualifier), 'Feature Vendor' (COMPANY), and 'Install Handler Library' (empty).

Abbildung 185 Feature Project Wizard - Einstieg

Die Eigenschaften bei einem Feature-Projekt sind ähnlich wie beim Plug-in-Projekt, und somit gelten auch die gleichen Regeln für z. B. *ID*, *Name* oder *Version*.

Nun kann der Wizard beendet werden oder man navigiert zur nächsten Seite, auf der man gleich die Plug-ins selektieren kann, die in diesem Feature enthalten sein sollen:

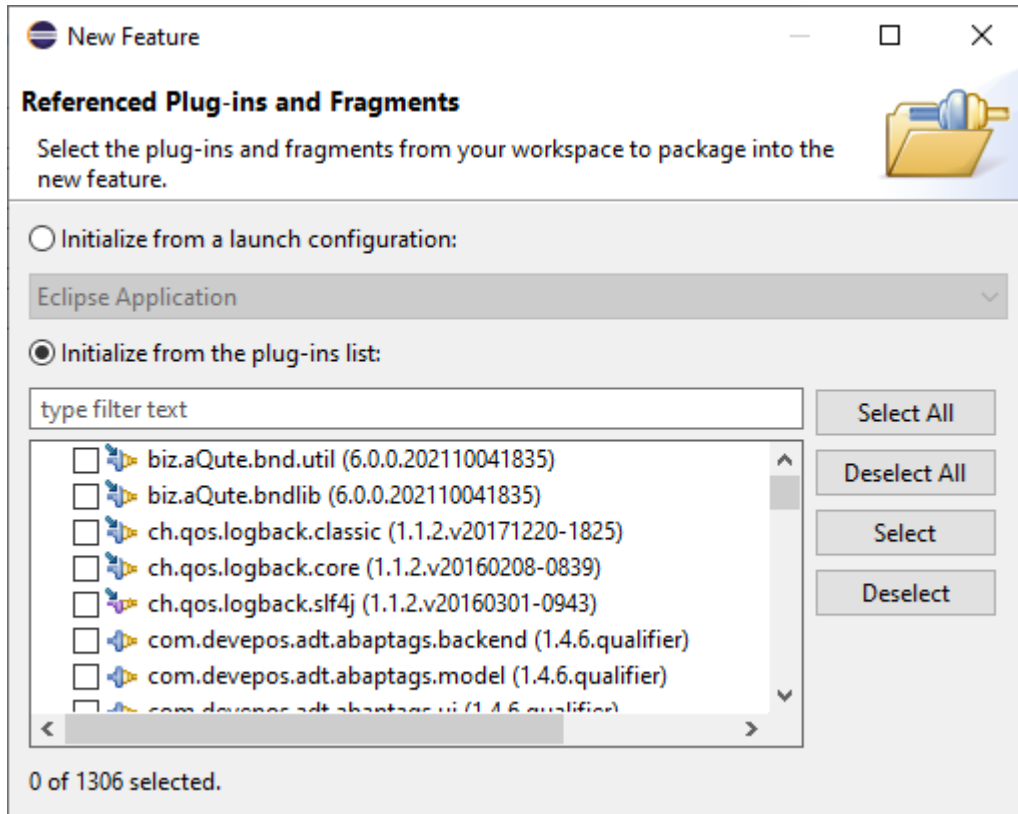


Abbildung 186 Feature Project Wizard - Plug-in-Auswahl

Nach Abschluss wird das Feature-Projekt generiert und im Workbench angezeigt.

7.4.5.1 Struktur eines Feature-Projekts

Ein Feature-Projekt hat eine sehr einfache Struktur. Es enthält nur die zwei Dateien *feature.xml* und *build.properties*. Wie beim Plug-in-Projekt gibt es zur Pflege der Feature-Metadaten einen eigenen Manifest-Editor, welcher sich automatisch beim Öffnen einer der beiden Dateien öffnet.

Dieser ist in die folgenden Sektionen unterteilt:

- **Overview**
Diese Sicht dient als genereller Einstiegspunkt. Sie erlaubt die Pflege der grundlegenden Plug-in-Daten wie Name, Version etc. und den Absprung zu den anderen Sichten.
- **Information**
Hier können Beschreibung, Copyright-Notice und Lizenzvereinbarung gepflegt werden.
- **Included Plug-ins**
Auswahl der Plug-ins, die in diesem Feature enthalten sind.
- **Included Features**
Features können zur Gruppierung anderer Features verwendet werden und somit können hier die enthaltenen Features gelistet werden.
- **Dependencies**
Normalerweise werden alle Abhängigkeiten beim Build berechnet. Dies passiert durch Analyse der Abhängigkeiten aller enthaltenen Plug-ins. Es besteht jedoch auch die Möglichkeit, die Abhängigkeiten hier manuell zu pflegen.
- **Build**
siehe Plug-in-Manifest

7.4.6 Erstellung einer Update-Site

Ein neues Update-Site-Projekt kann über File → New → Project... → Plug-in Development →> Update Site Project erstellt werden. Daraufhin öffnet sich der Update Site Project Wizard:

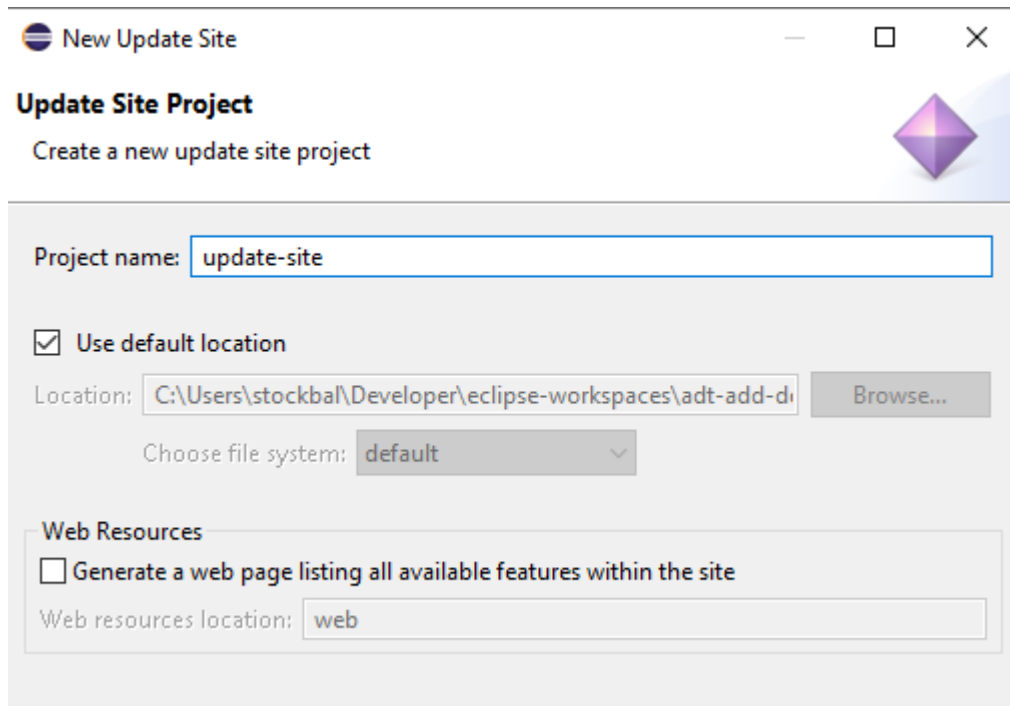


Abbildung 187 Update Site Wizard

Der Wizard enthält nur eine Seite, auf der man dem Projekt einen Namen gibt und den Ablageort auswählen kann. Nach Abschluss des Wizard sollte sich im Workspace ein Ordner mit dem gewählten Projektnamen befinden. In diesem Ordner befindet sich die Manifestdatei der Update-Site, genannt *site.xml*.

Bei Doppelklick auf diese Datei öffnet sich der Manifest-Editor für die Update-Site. In diesem Editor können nun die Features hinzugefügt werden, die auf der Site veröffentlicht werden sollen. Zur besseren Übersicht sollten die Features in Kategorien unterteilt werden.

Nachdem der Inhalt der Update-Site fertig konfiguriert ist, kann diese über den Button *Build all* im Editor erstellt werden. Es ist auch möglich, nur einzelne bzw. ausgewählte Features zu erstellen.

WICHTIG: Bevor man nun die Update-Site erstellt, sollte man nochmal die Java-Compiler-Einstellungen über Window → Preferences → Java → Compiler prüfen. Diese sollten auf die gleiche Java-Version eingestellt sein, die bei den Plug-ins als minimale Voraussetzung definiert wurde:

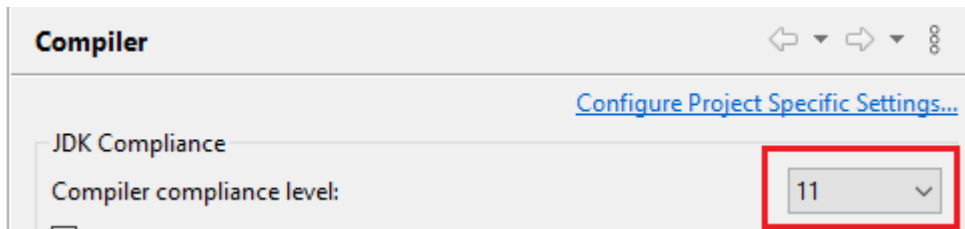


Abbildung 188 Compiler-Einstellungen im Eclipse-Einstellungsdialog

Wenn die Erstellung erfolgreich war, befinden sich die folgenden Dateien/Ordner im Anschluss im Projektordner der Update-Site:

- features (enthält jar-Dateien der Features)
- plugins (enthält jar-Dateien der Plug-ins)
- artifacts.jar
- content.jar

Zusätzlich kann auch noch ein Archiv mit dem Namen *logs.zip* erstellt worden sein. Dort befinden sich alle Meldungen, die während der Kompilierung aufgetreten sind.

7.4.6.1 Testen der Update-Site

Bevor die erstellte Site nun auf einem Webserver hochgeladen wird, möchte man diese vorher eventuell noch testen. Dazu sollte man sich eine neue Eclipse-Installation besorgen. Hierfür ist die Variante *Eclipse IDE for Java Developers* völlig ausreichend. In dieser Installation werden nun erst die ADT installiert, nach einem Neustart fügt man jetzt die neue – bis jetzt noch lokale – Update-Site über Help → Install New Software... → Add... hinzu:

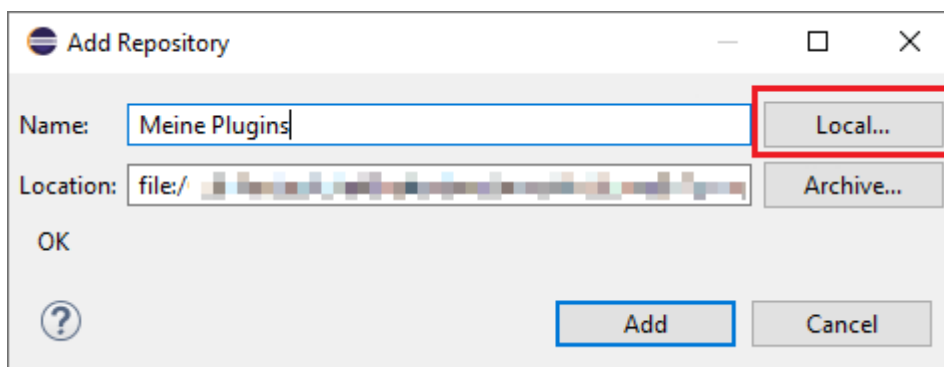


Abbildung 189 Dialog zum Hinzufügen einer Update-Site

Über den Button “Local...” wählt man dann das Verzeichnis des Update-Site-Projekts aus. Nach Klick auf den Button “Add” sollten die Kategorien und die zugewiesenen Features der Update-Site aufgelistet und installierbar sein.

7.4.6.2 Deployment

Wenn der Test der Update-Site erfolgreich war, kann diese nun auf einem Webserver hochgeladen werden, um die Artefakte für andere bereitzustellen. Sollte man die Kosten für einen eigenen Webserver scheuen, gibt es zum Beispiel über GitHub Pages eine kostenlose Möglichkeit, seine Update-Site bereitzustellen. Dazu initialisiert man im Projektverzeichnis der Update-Site ein neues Git Repository und veröffentlicht dieses in einem öffentlichen Repository auf GitHub. Im Anschluss kann in den Repository-Einstellungen auf GitHub die Option “GitHub Pages” aktiviert werden:

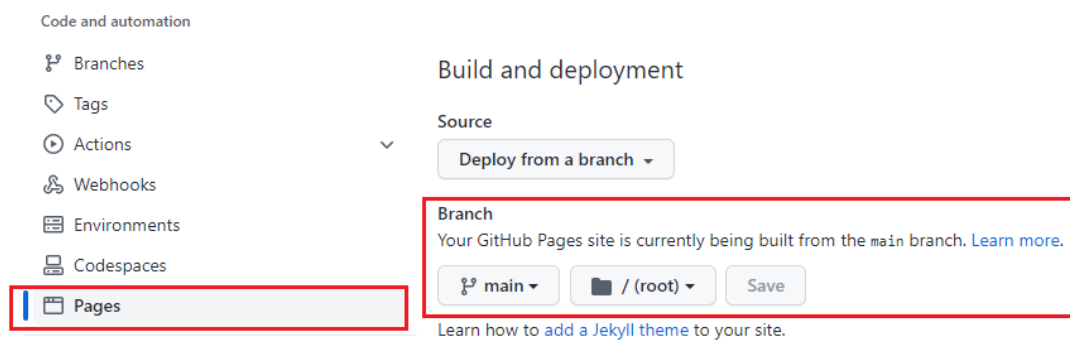


Abbildung 190 GitHub-Repository-Einstellungen für GitHub Pages

Nachdem das Erstellen der GitHub-Pages-Seite durch GitHub abgeschlossen ist, kann diese mit der URL `https://<username>.github.io/<repository-name>` in Eclipse als Update-Site eingetragen werden.

7.4.7 Erweiterung des ADT Backends mit ABAP Code

Für die Kommunikation von Eclipse zum ABAP Server verwenden die ADT RESTful APIs. Wie man solche APIs selbst entwickelt, kann in dem SAP Guide [How To... Create RESTful APIs and consume them in ADT](#) nachgelesen werden.

Obwohl hier die Rede von RESTful APIs ist, findet von ADT aus keine Kommunikation mittels HTTP statt. Sie erfolgt mit dem RFC-Protokoll. In der untersten Ebene der Kommunikationsschicht von ADT findet somit ein RFC-Aufruf statt, der einen bestimmten RFC-Funktionsbaustein auf dem ABAP Server aufruft.

Somit steht es jedem offen, entweder, wie im Guide beschrieben, eigene RESTful APIs mit dem BAdI-Erweiterungskonzept zu entwickeln oder alternativ einen RFC-fähigen Funktionsbaustein zu entwickeln und diesen mit dem RFC Java Connector aufzurufen. Die Java Connector API kann über das Plug-in “com.sap.conn.jco” eingebunden und verwendet werden.

Vor allem bei kleineren Plug-ins mag die BAdI-Methode als ziemlicher Overhead wirken, sowohl auf ABAP- als auch auf Java-Seite. Jedoch hat der BAdI-Ansatz auch seine Vorteile. Vor allem durch den Einsatz von EMF (Eclipse Modelling Framework) besteht die Möglichkeit, einen von ABAP nach XML serialisierten String ganz einfach in Objekte in Java umzuwandeln. Einfach beschrieben benötigt man dafür auf der ABAP-Seite eine “Simple Transformation” (Objektyp XSLT), um ABAP-Daten nach XML zu transformieren.

```
1 | k?sap.transform simple?>
2 | <tt:transform xmlns:tt="http://www.sap.com/transformation-templates"
3 |             xmlns:cst="http://www.devepos.com/adt/cst"
4 |             xmlns:adtbase="http://www.devepos.com/adt/base"
5 |             xmlns:adtcore="http://www.sap.com/adt/core"
6 |             xmlns:cl="http://www.sap.com/abapxml/types/class-pool/ZIF_ADCOSET_TY_ADT_TYPES">
7 |
8 |   <tt:root name="root" type="cl:ty_code_search_result"/>
9 |
10 |   <tt:include name="sadt_object_reference" template="objectReferenceAttributes"/>
11 |   <tt:include name="sadt_main_object" template="main_object"/>
12 |
13 |   <tt:template>
14 |     <tt:apply name="codeSearchResult">
15 |       <tt:with-root name="code_search_result" ref="root"/>
16 |     </tt:apply>
17 |   </tt:template>
18 |
19 |
20 |   <!-- Template for the result -->
21 |   <tt:template name="codeSearchResult">
22 |     <tt:context>
23 |       <tt:root name="code_search_result"/>
24 |     </tt:context>
25 |
26 |     <cst:result tt:extensible="deep">
27 |       <tt:attribute name="cst:numberOfResults" value-ref="code_search_result.number_of_results"/>
28 |       <tt:attribute name="cst:numberOfSearchedObjects" value-ref="code_search_result.number_of_searched_objects"/>
29 |       <tt:attribute name="cst:numberOfSearchedSources" value-ref="code_search_result.number_of_searched_sources"/>
30 |       <tt:attribute name="cst:linesOfSearchedCode" value-ref="code_search_result.loc"/>
31 |       <tt:attribute name="cst:queryTimeInMs" value-ref="code_search_result.query_time_in_ms"/>
32 |
33 |       <tt:apply name="codeSearchObjects">
34 |         <tt:with-root name="code_search_objects" ref="code_search_result.code_search_objects"/>
35 |       </tt:apply>
36 |     </cst:result>
37 |   </tt:template>
38 |
39 | </tt:transform>
```

Abbildung 191 Beispiel für eine Simple Transformation zur Transformation von ABAP <-> XML

Und seitens Java ist ein EMF-Modell notwendig.

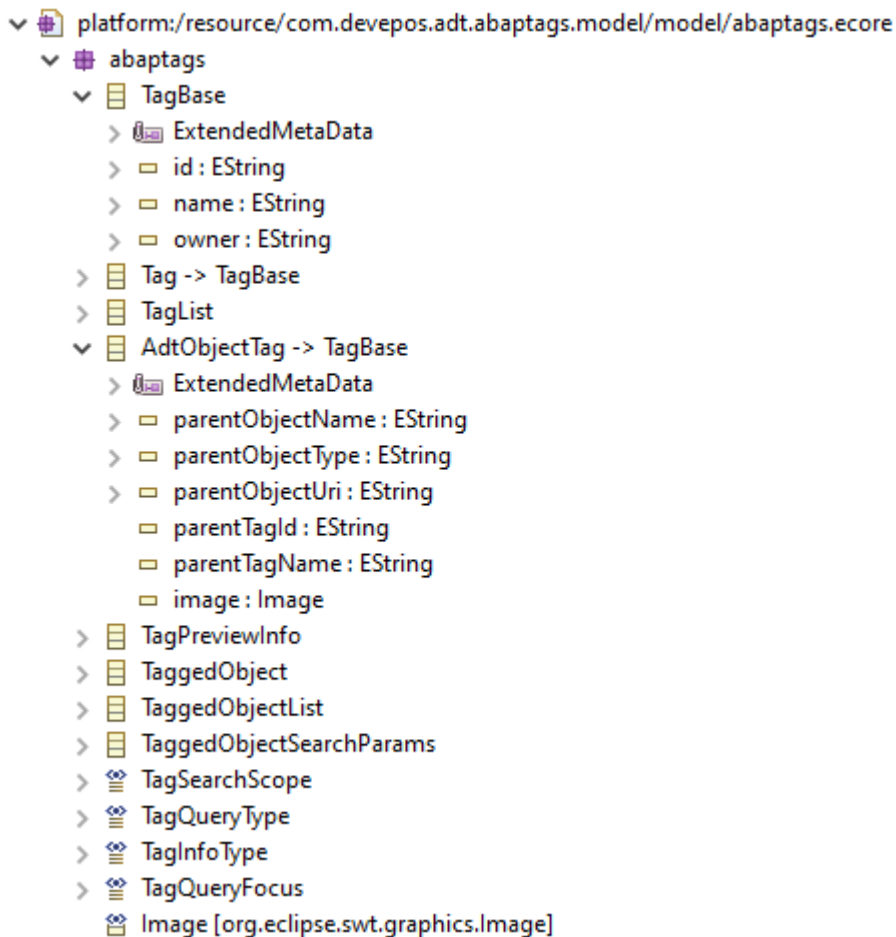


Abbildung 192 Beispiel für EMF-Modell zur Serialisierung von XML- <-> Java-Objekt

Durch diesen Ansatz lässt sich die Datentransformation sehr generisch bauen, und man spart auch wieder Entwicklungszeit ein.

7.4.8 Java Code Snippets für wiederkehrende Aufgaben in ADT

7.4.8.1 Aufruf eines RFC-Funktionsbausteins mit dem Java Connector

```
// 1) Lesen der Destination Id für eine ABAP Project Instanz
String destinationId =
AdtCoreProjectServiceFactory.createCoreProjectService().getDestinationId(project);

// 2) Lesen der JCo destination zur Destination Id
JCoDestination destination = JCoDestinationManager.getDestination(destinationId);

// 3) Lesen des RFC Bausteins
```

```
JCoFunction                function                =
destination.getRepository().getFunction("name_of_rfc_function");

// 4) Setzen eines Importing-Parameters

function.getImportParameterList().getField("I_PARAM1").setValue("PARAM_VALUE");

// 5) Ausführen der Funktion

function.execute(destination);

// 6) Lesen eines Exporting Tabellenparameters

JCoTable objectTree = function.getExportParameterList().getTable("E_PARAM1");
```

7.4.8.2 Lesen des ABAP-Projekts abhängig von der aktuellen Selektion im Workbench

```
// 1) Ermittlung der aktiven Page im Workbench

IWorkbenchPage                page                =
PlatformUI.getWorkbench().getActiveWorkbenchWindow().getActivePage();

// 2) Lesen der Workbench Window Instanz

IWorkbenchWindow window = page.getWorkbenchWindow();

// 3) Ermittlung der aktuellen Selektion im Window

ISelection adtSelection = window.getSelectionService().getSelection();

// 4) Ermittlung des aktiven ABAP Projekts

IProject project = ProjectUtil.getActiveAdtCoreProject(adtSelection, null, null,
IAbapProject.ABAP_PROJECT_NATURE);
```

7.4.8.3 Ermittle den Quellcode des aktiven Editors

```
// 1) Ermittle den aktiven Editor

IAdtFormEditor editor = (IAdtFormEditor)PlatformUI.getWorkbench()

                .getActiveWorkbenchWindow().getActivePage()

                .getActiveEditor();

// 2) Ermittle das Document des Editors

IDocument document = editor.getAdapter(AbapSourcePage.class).getDocument();
```

```
// 3) Ermittle den Quellcode  
String code = document.get();
```

7.4.8.4 Führe den Transaction-Code aus

```
// 1) Ermittle die Benutzereinstellung der Navigation zum Eclipse Editor für  
unterstützte Entwicklungsobjekte
```

```
boolean navigateToEclipse =  
    com.sap.adt.sapgui.ui.internal.Activator.getDefault()  
        .getPreferenceStore()  
        .getBoolean(com.sap.adt.sapgui.ui.internal.PreferenceInitializ  
            er  
                .PREF_KEY_USE_ECLIPSE_NAVIGATION);
```

```
// 2) Ausführung des Transaction Code
```

```
AdtSapGuiEditorUtilityFactory  
    .createSapGuiEditorUtility()  
    .openEditorAndStartTransaction(project, TRANSACTION_NAME,  
navigateToEclipse);
```

7.4.8.5 ABAP Scan Services – Prüfe, ob der Token ein Keyword ist

```
// 1) Ermittle die Instanz von AbapSourceUI  
IAbapSourceUi sourceUi = AbapSourceUi.getInstance();
```

```
// 2) Ermittle die Instanz von SourceScannerServices
```

```
IAbapSourceScannerServices = sourceUi.getSourceScannerServices();
```

```
// 3) Ermittle den aktiven ADT Editor
```

```
editor = (IAdtFormEditor)PlatformUI.getWorkbench()  
    .getActiveWorkbenchWindow().getActivePage().getActiveEditor();
```

```
// 4) Ermittle das Dokument vom Editor
```

```
IDocument document = editor.getAdapter(AbapSourcePage.class).getDocument();
```

```
// 5) Prüfe ob der Token an der Offset-Position ein Keyword ist (oder nicht)
Boolean isKeyword = scannerServices.isKeyword(document,OFFSET);

// 6) Ermittle den nächsten Token basierend auf der Offset-Position
Token nextToken = scannerServices.getNextToken(document,OFFSET);

// 7) Prüfe ob der nächste Token ein Keyword ist (oder nicht)
isKeyword = scannerServices.isKeyword(document,nextToken.offset);
```

7.4.8.6 Ermittle das Projekt und zeige den Selektionsdialog

```
// 1) Ermittle die Shell
Shell shell = PlatformUI.getWorkbench().getActiveWorkbenchWindow().getShell();

// 2) Zeige den Selektions-Dialog und ermittle das gewählte Projekt
IProject chosenProject = AbapProjectSelectionDialog.open(shell, null);
```

7.4.8.7 Ermittle die User und rufe den Selektionsdialog

```
// 1) Ermittle den User Service
IAdtUserServiceUI          adtUserService          =
AdtUserServiceUIFactory.createAdtUserServiceUI();

// 2) Rufe den User Selektions-Dialog und ermittle die selektierten User
String[] users = adtUserService.openUserNameSelectionDialog(null, false,
project,"");
```

7.4.8.8 Anmelden auf dem ABAP-System

```
// 1) Adaptieren eines IProject Objekts zu einem IAbapProject Objekt
final IAbapProject abapProject = project.getAdapter(IAbapProject.class);

// 2) Prüfen des Anmeldestatus mit automatischem Anmeldeialog falls noch keine
// Anmeldung vorliegt
IStatus logonStatus = AdtLogonServiceUIFactory.createLogonServiceUI()
    .ensureLoggedOn(abapProject.getDestinationData(), PlatformUI.getWorkbench())
```

```
.getProgressService()  
.isOK();
```

7.4.9 Projekt-Set-up mit Maven

Neben den Standardmöglichkeiten zur Entwicklung von Plug-ins, Features und Update-Sites, welche mit der *Eclipse IDE for RCP and RAP Developers* geliefert werden, gibt es auch noch die Option, Eclipse Tycho zu verwenden. Eclipse Tycho ist eine Sammlung von Plug-ins für Apache Maven. Weitere Informationen können der [Projektseite](#) von Tycho entnommen werden. Ein Tutorial zur Plug-in-Entwicklung mit Tycho gibt es [hier](#).

Autoren

Abschließend wollen wir uns als Autorenteam und kontinuierliche Ansprechpartner bzgl. jeglicher Fragen im Kontext der ABAP Development Tools gerne noch vorstellen.

Doch zunächst gilt unser spezieller Dank an dieser Stelle **Marc Zimmek** (Claas KGaA), **Jens Zähringer** (Nagarro) und der DSAG Geschäftsstelle, die uns in den finalen Phasen der Erstellung durch ihr tatkräftiges Korrekturlesen unterstützten.

Darüber hinaus wollen wir es nicht verpassen, auch die enge, konstruktive und unkomplizierte Zusammenarbeit mit Kollegen der SAP herauszuheben. Herzlichen Dank an **Thomas Fiedler** (Product Owner ADT), **Wolfgang Wöhrle** (ADT Dokumentation inkl. Konfiguration und Installation), **Yannic Soethoff** und **Sebastian Ratz** (beide ADT unter Mac OS).

Tabelle 6 Autoren

Name	Firma und Jobbezeichnung	Kurzbeschreibung eurer Tätigkeiten und Erfahrungen mit ADT
Michael Biber	R+V Allgemeine Versicherungen AG Systementwickler und Anwendungsmanager	Erste Anfänge 2016 im Selbststudium. Nach und nach verstärkter Einsatz. Heute exklusiver Einsatz. Seit ca. 2018 Betreuung der oomph-Konfigurationen für den SAP-Bereich.
Jörg Brandeis	Brandeis Consulting GmbH Geschäftsführer, Trainer, Entwickler und Berater	Jörg Brandeis ist Geschäftsführer der Brandeis Consulting GmbH in Mannheim, die Schulungen und Beratung im Bereich SAP BW/4HANA, SAP HANA, Modernes ABAP und SQLScript anbietet. Er ist Autor des Buches SQLScript für SAP HANA (SAP Press)

Autoren

<p>Uwe Fetzner</p>	<p>Silesia Gerhard Hanke GmbH & Co. KG</p> <p>Team Lead Global SAP Development & Analytics</p>	<p>Seit Anfang an ADT privat im Einsatz (Co-Autor SAPLink-Erweiterung für Eclipse), seit fünf Jahren exklusiver ADT-Einsatz bei Arbeitgebern</p>
<p>Thomas Foehn</p>	<p>Zürcher Kantonalbank</p>	<p>Verantwortlich für die ABAP-Entwicklungsrichtlinien und Best Practices. Noch kein flächendeckender Einsatz von ADT. Einsatz je nach Mitarbeitervorliebe, selbst mittlerweile intensiver Nutzer</p>
<p>Sebastian Freilinger-Huber</p>	<p>msg systems ag</p> <p>Principal IT Consultant</p> <p>DSAG Sprecher AK Development</p>	<p>Programmieren, Konzipieren, Architektur; Einsatz von Eclipse (ADT) seit Einführung; Forcieren des ADT-Einsatzes unternehmens- und bereichsintern</p> <p>Organisatorische Leitung des DSAG-ADT-Leitfadens</p>
<p>Florian Henninger</p>	<p>FIS Informationssysteme und Consulting GmbH</p> <p>Senior Consultant Development</p>	<p>Programmieren, Konzipieren, Architektur; Einsatz von Eclipse (ADT) seit Beginn an;</p> <p>Hauptgebiet Odata/fiori elements</p>
<p>Armin Junge</p>	<p>msg systems ag</p> <p>Lead IT Consultant</p>	<p>Architektur, Design, Implementierung</p> <p>Intensive Verwendung von ADT seit vielen Jahren. Forcierung der Verwendung im Bereich und beim Kunden.</p>

<p>Michael Keller</p>	<p>educatedBytes GmbH</p> <p>CEO, Berater, Entwickler, Trainer</p>	<p>Michael Keller arbeitet seit vielen Jahren als Berater, Entwickler und Trainer im Logistikumfeld von SAP-ERP-Systemen. Durch die Entwicklung mit der Programmiersprache ABAP hat er 2017 die ABAP Development Tools for Eclipse kennengelernt und arbeitet seitdem mit dieser Entwicklungsumgebung.</p>
<p>Peter Luz</p>	<p>Robert Bosch GmbH</p> <p>Development Engineer and - architect</p>	<p>ABAP-Entwickler und Koordinator für zentral entwickelte Komponenten und Lösungen im Bereich SAP Logistics Execution.</p> <p>Fan von ABAP-OO und ABAP Unit Tests.</p> <p>Einstieg in die ADT 2015. Mit den ADT ist die Erstellung von qualitativ hochwertigem und modernem Code deutlich effizienter und schneller geworden.</p>
<p>Dominik Panzer</p>	<p>Intense AG</p> <p>Senior Manager/ Entwicklungsleiter/ Agile Technical Coach</p>	<p>Langjähriger ADT-Nutzer und oft tief in Legacy-Codebases vergraben, um diese zu transformieren und in die neue Cloud-Welt zu bringen. Daher schätzt er die Refactoring-Features von ADT sehr. Neben technischen Themen, Clean Code und Software-Architektur liegen ihm auch Team-orientierte Themen wie Agile Games, Coding Dojos, Pair- und Mobprogramming am Herzen und vermittelt diese als Coach weiter.</p>
<p>Lukasz Pegiel</p>	<p>Hager Group</p> <p>Digital & Information Manager Poland</p>	<p>I'm responsible among others for:</p> <ul style="list-style-type: none"> - local teams management, which includes ABAP Development Team, Net Development Team, IT support and BIM (Building Information Modeling Team) - Collaboration with External Resources Providers - S/4HANA custom code adaptation.

Autoren

<p>Dr. Wolfgang Röckelein</p>	<p>Axians NEO Solutions & Technology</p> <p>Senior Architekt Produktentwicklung</p>	<p>Die ADT werden von mir und meinen Kollegen in der Produktentwicklung und in den Kundenprojekten zu unseren Produkten flächendeckend eingesetzt.</p>
<p>Björn Schulz</p>	<p>REWE digital GmbH</p> <p>Product Owner (SAP Development & Technology)</p>	<p>Product Owner und Entwickler für das Thema ABAP, Steampunk und BTP. Nur noch ADT seit knapp vier Jahren und stets bemüht, die Kollegen in Richtung ADT zu motivieren und die vielen Vorteile zu "verkaufen".</p>
<p>Ludwig Stockbauer-Muhr</p>	<p>msg systems ag</p> <p>Senior IT Consultant</p>	<p>Architektur, Design, Implementierung</p> <p>ADT seit vielen Jahren im Einsatz</p> <p>Leidenschaftlicher Add-on-Entwickler für ADT</p>
<p>Bärbel Winkler</p>	<p>Alfred Kärcher SE & Co. KG</p> <p>Consultant SAP Development</p>	<p>ABAP Development & Guidelines. Noch kein flächendeckender Einsatz der ADT und selbst nur "rudimentäre" Erfahrungen, nutze Eclipse aber immer häufiger und möchte auch andere dazu motivieren.</p>

Impressum

Wir weisen ausdrücklich darauf hin, dass das vorliegende Dokument nicht jeglichen Regelungsbedarf sämtlicher DSAG-Mitglieder in allen Geschäftsszenarien antizipieren und abdecken kann. Insofern müssen die angesprochenen Themen und Anregungen naturgemäß unvollständig bleiben. Die DSAG und die beteiligten Autoren können bezüglich der Vollständigkeit und Erfolgsgeeignetheit der Anregungen keine Verantwortung übernehmen.

Die vorliegende Publikation ist urheberrechtlich geschützt (Copyright).

Alle Rechte liegen, soweit nicht ausdrücklich anders gekennzeichnet, bei:

Deutschsprachige SAP® Anwendergruppe e.V.

Altrottstraße 34 a

69190 Walldorf | Deutschland

Telefon +49 6227 35809-58

Telefax +49 6227 35809-59

E-Mail info@dsag.de

dsag.de

Jedwede unerlaubte Verwendung ist nicht gestattet. Dies gilt insbesondere für die Vervielfältigung, Bearbeitung, Verbreitung, Übersetzung oder die Verwendung in elektronischen Systemen/digitalen Medien.

© Copyright 2023 DSAG e.V.