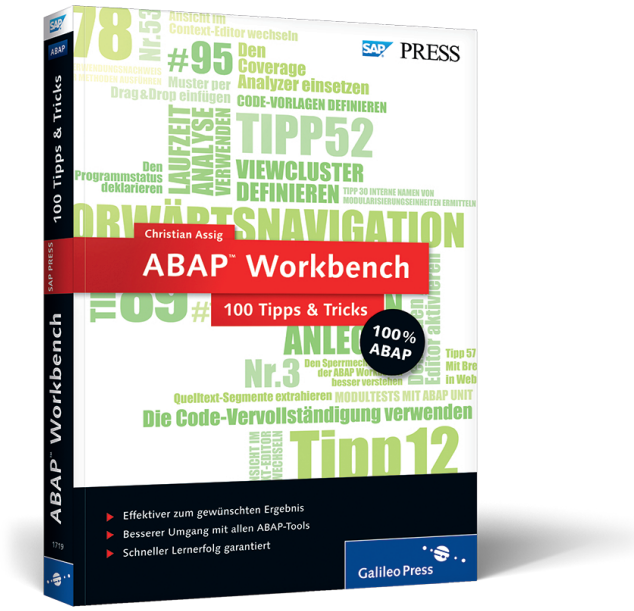
Christian Assig

ABAP™ Workbench – 100 Tipps & Tricks



Galileo Press

Bonn • Boston

Auf einen Blick

TEIL 1 **Quelltext bearbeiten** 17

TEIL 2 **Quelltext automatisch erzeugen** 73

TEIL 3 **Mit Zusatzinformationen zu Entwicklungsobjekten arbeiten** 97

TEIL 4 **Allgemeines zur ABAP Workbench** 147

TEIL 5 **ABAP-Dictionary-Objekte und Tabellenpflegewerkzeuge bearbeiten** 163

TEIL 6 **Web-Dynpro-Components bearbeiten** 195

TEIL 7 **Anwendungen testen** 211

TEIL 8 **Anwendungen debuggen** 239

TEIL 9 **In Entwicklungsobjekten suchen** 287

TEIL 10 **Werkzeuge zur Analyse von Anwendungen** 309

TEIL 11 **Mit Transportaufträgen arbeiten** 369

Inhalt

Einleitung 13

TEIL 1 Quelltext bearbeiten 17

Tipp 1 Den neuen ABAP Editor aktivieren 18
Tipp 2 Neue Ansichten ab Release 7.0 EHP2 23
Tipp 3 Den Sperrmechanismus besser verstehen 27
Tipp 4 Die mehrstufige Zwischenablage nutzen 30
Tipp 5 Quelltext spaltenweise markieren 33
Tipp 6 Zwischen Groß- und Kleinschreibung wechseln 37
Tipp 7 Kommentare mit dem Pretty Printer automatisch einrücken 40
Tipp 8 Automatische Formatierung beim Einfügen aktivieren 43
Tipp 9 Den Pretty Printer als Vorstufe der Syntaxprüfung nutzen 46
Tipp 10 Fehlermeldungen der Syntaxprüfung besser verstehen 49
Tipp 11 Die automatische Fehlerkorrektur verwenden 53
Tipp 12 Objektorientierte Ereignisse 57
Tipp 13 Variablendefinitionen bereinigen 62
Tipp 14 Methoden verschieben 65
Tipp 15 Quelltext-Segmente extrahieren 68

TEIL 2 Quelltext automatisch erzeugen 73

Tipp 16 Muster per Drag & Drop einfügen 74
Trick 17 Funktionale Schreibweise in Mustern aktivieren 76
Tipp 18 Eigene Muster definieren 78
Tipp 19 Remote Function Calls mit dem BAPI Browser erzeugen 81
Tipp 20 Die Code-Vervollständigung verwenden 84
Tipp 21 Funktionalitäten der Code-Vervollständigung 86
Tipp 22 Einstellungen zur Code-Vervollständigung 89
Tipp 23 Code-Vorlagen definieren und einfügen 91
Tipp 24 Textsymbol aus Literal anlegen 94

TEIL 3 Mit Zusatzinformationen zu Entwicklungsobjekten arbeiten 97

Tipp 25 Den Programmstatus deklarieren 98
Tipp 26 Die Standardnachrichtenklasse festlegen 101
Tipp 27 Informationen aus dem Objektkatalog auswerten 103
Tipp 28 Bevorzugte Parameter von Methoden deklarieren 107
Tipp 29 Die Editorsperre verwenden 110
Tipp 30 Interne Namen von Modularisierungseinheiten ermitteln 114
Tipp 31 Typen in Klassen und Interfaces definieren 119
Tipp 32 Das Workflow-Interface integrieren 122
Tipp 33 Versionen von Entwicklungsobjekten anlegen 124
Tipp 34 Versionsstände vergleichen 126
Tipp 35 Alte Versionen von Entwicklungsobjekten zurückholen 130
Tipp 36 Pakete definieren 133
Tipp 37 Paketschnittstellen definieren 136
Tipp 38 Verwendungserklärungen definieren 139
Tipp 39 Paketabhängigkeiten prüfen 142

TEIL 4 Allgemeines zur ABAP Workbench 147

Tipp 40 Die Vorwärtsnavigation nutzen 148
Tipp 41 Mit der Historie im SAP GUI arbeiten 151
Tipp 42 Formulare schneller ausfüllen 154
Tipp 43 Die Spaltenbreite anpassen 157
Tipp 44 Darstellungen von Entwicklungsobjekten aktualisieren 160

TEIL 5 ABAP-Dictionary-Objekte und Tabellenpflegewerkzeuge bearbeiten 163

Tipp 45 Elementare Suchhilfen definieren 164
Tipp 46 Fremdschlüssel definieren 168
Tipp 47 Schlüsselfelder und Initialwerte in Strukturen 173
Tipp 48 Die Reihenfolge der Felder in Datenbanktabellen korrigieren 176
Tipp 49 Pflegedialoge anlegen und aktualisieren 179
Tipp 50 Pflegedialoge um Anwendungslogik erweitern 183
Tipp 51 Pflege-Views definieren 186
Tipp 52 Viewcluster definieren 190

TEIL 6 Web-Dynpro-Components bearbeiten 195

Tipp 53 Ansicht im Context-Editor wechseln 196
Tipp 54 Die quelltextbasierte Ansicht in Web Dynpro nutzen 199
Tipp 55 Mit dem Code-Wizard Quelltext generieren 202
Tipp 56 Mit dem Code-Wizard Benutzeroberflächen generieren 205
Tipp 57 Mit relativen Breitenangaben in Web Dynpro arbeiten 208

TEIL 7 Anwendungen testen 211

Tipp 58 Die Testumgebung für Klassen und Funktionsbausteine einsetzen 212
Tipp 59 Sperren testen 218
Tipp 60 Die Performance-Beispiele nutzen 221
Tipp 61 Modultests mit ABAP Unit durchführen 224
Tipp 62 Tests von Benutzeroberflächen mit eCATT automatisieren 228
Tipp 63 Assertions, Breakpoints und Logpoints aktivieren 235

TEIL 8 Anwendungen debuggen 239

Tipp 64 Zwischen dem klassischen und dem neuen ABAP Debugger umschalten 240
Tipp 65 Debugging von Systemprogrammen 243
Tipp 66 Debugging aus einem Pop-up aktivieren 246
Tipp 67 Debugging von Hintergrundanwendungen aktivieren 249
Tipp 68 Debugging für laufende Web-Dynpro-Anwendungen aktivieren 253
Tipp 69 Besonderheiten beim Debugging von Web-Dynpro-Anwendungen 256
Tipp 70 Breakpoints bei bestimmten Anweisungen anlegen 260
Tipp 71 Breakpoints mit Bedingungen verwenden 264
Tipp 72 Watchpoints verwenden 267
Tipp 73 Layer-Aware Debugging 271
Tipp 74 Layouts im ABAP Debugger konfigurieren 274
Tipp 75 Klassenbasierte Ausnahmen im ABAP Debugger auswerten 278
Tipp 76 Informationen aus einem Kurzdump auswerten 282

TEIL 9 In Entwicklungsobjekten suchen 287

Tipp 77 Im Quelltext inkrementell suchen 288
Tipp 78 Systemweit Quelltext durchsuchen 291
Tipp 79 Klassen schneller finden 297
Tipp 80 Grenzen des Indexes für den Verwendungsnachweis kennen 300
Tipp 81 Den Verwendungsnachweis für Methoden ausführen 303
Tipp 82 Die Umfeldermittlung verwenden 306

TEIL 10 Werkzeuge zur Analyse von Anwendungen 309

Tipp 83 Quelltexte mit dem Splitscreen-Editor vergleichen 310
Tipp 84 Die erweiterte Prüfung verwenden 314
Tipp 85 Die Funktionalitäten des Code Inspectors kennen 318
Tipp 86 Inspektionen, Objektmengen und Prüfvarianten im Code Inspector 322
Tipp 87 Syntaxprüfung, erweiterte Prüfung und Modultests automatisieren 328
Tipp 88 Eigene Code-Inspector-Tests implementieren 333
Tipp 89 Die unterschiedlichen Möglichkeiten des Performance-Trace 339
Tipp 90 Datenbanktabellen zu einer laufenden Anwendung bestimmen 342
Tipp 91 Die Laufzeitanalyse verwenden 345
Tipp 92 Die Laufzeitanalyse mit Web-Dynpro-Applikationen verwenden 352
Tipp 93 Den Performancemonitor in Web-Dynpro-Anwendungen einsetzen 356
Tipp 94 Den Speicherbedarf von Anwendungen analysieren 360
Tipp 95 Den Coverage Analyzer einsetzen 364

TEIL 11 Mit Transportaufträgen arbeiten 369

Tipp 96 Standardaufträge definieren 370
Tipp 97 Transportaufträge organisieren 373
Tipp 98 Mit Sperren in Transportaufträgen arbeiten 378
Tipp 99 Einträge in Transportaufträgen verschieben 382
Tipp 100 Mit Entwicklungsobjekten in ein anderes System umziehen ... 386

Der Autor und die Ideengeber 393
Index 397

Einleitung

Die Entwicklung von Software in ABAP unterscheidet sich deutlich von der Entwicklung in anderen Programmiersprachen wie Java, C++ oder C#. Während es bei zahlreichen anderen Programmiersprachen eine Vielzahl von Entwicklungsumgebungen gibt, aus der ein Entwickler die für sich passende Entwicklungsumgebung aussuchen kann, besitzt die ABAP Workbench mehr oder weniger ein Monopol für die Softwareentwicklung in ABAP. Jeder ABAP-Entwickler kennt darum die ABAP Workbench und ihre grundlegenden Funktionalitäten.

Auch am Buchmarkt spiegelt sich der Unterschied in den Entwicklungslandschaften wider: Zu Entwicklungsumgebungen für andere Programmiersprachen wie Eclipse oder Microsoft Visual Studio ist eine Vielzahl von Büchern verfügbar. In diesen Büchern steht die Entwicklungsumgebung im Vordergrund. Die Programmiersprache spielt, wenn überhaupt, nur eine untergeordnete Rolle. In der ABAP-Welt gibt es dagegen viele Bücher, die sich mit der Programmiersprache ABAP oder mit bestimmten Technologien in der ABAP-Welt beschäftigen. Die ABAP Workbench als Entwicklungsumgebung thematisieren diese Bücher dabei jedoch nur am Rand.

Natürlich wäre es auch wenig sinnvoll, ein Buch zu schreiben, das alle Grundlagen der ABAP Workbench, vom Anmelden am System über den Entwicklerschlüssel bis hin zum Aktivieren von Entwicklungsobjekten, im Detail erläutert. All das kennt jeder ABAP-Entwickler ohnehin, da es wahrscheinlich keinen ABAP-Entwickler gibt, der niemals mit der ABAP Workbench gearbeitet hat.

Aber über die Grundlagen hinaus bietet die ABAP Workbench noch eine Vielzahl von Möglichkeiten, die nicht jedem Entwickler bekannt sind. Manche erlauben es Ihnen, Dinge zu tun, von denen Sie bisher dachten, dass sie in der ABAP-Welt nicht möglich seien. Andere helfen Ihnen dabei, Ihr Ziel deutlich schneller und mit viel weniger Aufwand zu erreichen als auf die Ihnen bereits bekannte Art.

Aber wo kann ein ABAP-Entwickler nachschlagen, wenn er mehr über das Werkzeug erfahren möchte, mit dem er tagtäglich arbeitet? Beim Versuch diese Frage zu beantworten, entstand die Idee zu diesem Buch.

Zielgruppe

Dieses Buch richtet sich an alle, die in Ihrem beruflichen Alltag mit der Software-Entwicklung in ABAP zu tun haben. Egal, ob Sie vornehmlich kleinere Erweiterungen zu bestehenden SAP-Produkten schreiben oder größere Komponenten selbst entwickeln, ob Sie gerade erst in die ABAP-Entwicklung eingestiegen sind oder sich in der ABAP-Welt bereits seit Jahren zu Hause fühlen: Sie werden in diesem Buch neue Ideen finden, die Sie bei der Arbeit mit der ABAP Workbench weiterbringen.

Zielsetzung und Inhalt

Ich verzichte in diesem Buch bewusst darauf, Ihnen Grundlagen zu erläutern, die Sie seit Ihren ersten Gehversuchen in ABAP ohnehin schon kennen. Stattdessen ist es das Ziel, Ihnen in Form der Tipps und Tricks möglichst viele neue Themen zu vermitteln, die Ihnen den Umgang mit der ABAP Workbench erleichtern, und damit Ihre Arbeit produktiver machen. Je nach Ihrem Kenntnisstand werden Sie sicherlich einige oder vielleicht sogar viele Themen bereits kennen, die im Buch erwähnt werden. Das Ziel des Buches ist es nicht, nur Geheimnisse zu offenbaren, die vorher noch keinem Entwickler bekannt waren. Vielmehr möchte ich Ihnen einen Überblick über die wichtigsten Themen vermitteln, die für jeden ABAP-Entwickler interessant sein können. Und ich bin mir sicher, dass auch Sie in diesem Buch Tipps finden werden, deren Inhalt Sie bisher noch nicht im Alltag anwenden.

Zum Aufbau

Das Buch enthält elf Teile, die jeweils mehrere Tipps zu einem Themengebiet zusammenfassen. Die einzelnen Tipps setzen nicht voraus, dass Sie auch die vorhergehenden Tipps gelesen haben. Sie können also an beliebiger Stelle damit beginnen, einzelne Tipps oder einzelne Teile in beliebiger Reihenfolge zu lesen. Wo inhaltliche Zusammenhänge zwischen mehreren Tipps bestehen, enthalten die Tipps jeweils Verweise auf die anderen Tipps, die notwendiges Vorwissen oder weitere Informationen zum aktuellen Tipp beinhalten.

Systemvoraussetzungen

Die meisten Inhalte dieses Buches können Sie in jedem SAP-System anwenden, das auf einem SAP NetWeaver Application Server ABAP basiert. Dazu gehören neben SAP ERP auch die meisten anderen betriebswirtschaftlichen Systeme von SAP. Einige Tipps beziehen sich auf Funktionalitäten, die erst in den letzten Jahren neu zur ABAP Workbench hinzugekommen sind.

Diese Tipps enthalten darum Hinweise dazu, ab welchem Release die beschriebenen Funktionalitäten verfügbar sind. Diese Release-Angaben beziehen sich jeweils auf das Basis-Release, das heißt darauf, welche Version der ABAP-Laufzeit- und Entwicklungsumgebung in Ihrem System enthalten ist. Um herauszufinden, mit welchem Basis-Release Sie arbeiten, rufen Sie im SAP GUI den Menüeintrag **System ▸ Status...** auf. Klicken Sie dann im Bereich **SAP-Systemdaten** auf das Detail-Symbol ( ). In der Zeile mit der Softwarekomponente `SAP_BASIS` sehen Sie dann in der Spalte **Release**, mit welchem Basis-Release Sie arbeiten.

Mit dem Basis-Release 7.0 können Sie den Großteil der Tipps aus diesem Buch nutzen. Die Funktionalitäten aus allen Tipps können Sie nutzen, sobald Sie mindestens mit dem Basis-Release 7.0 EHP2 arbeiten.

Danksagung

Die Ideen zu den einzelnen Tipps sind im beruflichen Alltag mit der Softwareentwicklung in ABAP entstanden. Natürlich habe ich mir nicht alle Tipps selbst ausgedacht. Eine Vielzahl von Vorschlägen für Tipps haben Experten für verschiedene Themenbereiche der Softwareentwicklung in ABAP beigetragen. Diese Ideengeber sind am Ende des Buches im Abschnitt »Der Autor und die Ideengeber« genannt. Ihnen gilt mein besonderer Dank. Ohne sie wäre es nicht möglich gewesen, ein Buch wie dieses zu schreiben.

Christian Assig
SAP-Entwickler, Thalia Holding

TEIL 1

Quelltext bearbeiten

Die Zeiten, in denen Entwickler jedes Zeichen im Quelltext manuell eintippen mussten, sind zum Glück lange vorbei. Die ABAP Workbench unterstützt Sie mit einer Vielzahl von Werkzeugen dabei, Quelltext schnell und einfach zu erstellen oder zu verändern. Zu diesen Werkzeugen gehören beispielsweise die Zwischenablage, die nicht nur den letzten, sondern die letzten zwölf kopierten Werte für Sie bereithält, oder Tastaturkürzel, mit denen Sie zwischen Groß- und Kleinschreibung wechseln können. Nicht mehr benötigte Variablen und zahlreiche Syntaxfehler kann die ABAP Workbench sogar automatisch entfernen.

Tipps in diesem Teil

Tipp 1	Den neuen ABAP Editor aktivieren	18
Tipp 2	Neue Ansichten ab Release 7.0 EHP2	23
Tipp 3	Den Sperrmechanismus besser verstehen	27
Tipp 4	Die mehrstufige Zwischenablage nutzen	30
Tipp 5	Quelltext spaltenweise markieren	33
Tipp 6	Zwischen Groß- und Kleinschreibung wechseln	37
Tipp 7	Kommentare mit dem Pretty Printer automatisch einrücken	40
Tipp 8	Automatische Formatierung beim Einfügen aktivieren	43
Tipp 9	Den Pretty Printer als Vorstufe der Syntaxprüfung nutzen	46
Tipp 10	Fehlermeldungen der Syntaxprüfung besser verstehen	49
Tipp 11	Die automatische Fehlerkorrektur verwenden	53
Tipp 12	Objektorientierte Ereignisse	57
Tipp 13	Variablendefinitionen bereinigen	62
Tipp 14	Methoden verschieben	65
Tipp 15	Quelltext-Segmente extrahieren	68

Tipp 1

Den neuen ABAP Editor aktivieren

Aktivieren Sie ihn am besten sofort: Der neue ABAP Editor ist die Basis für viele der in diesem Buch genannten Tipps und Tricks und bietet zahlreiche Vorteile!

In den Benutzerprofilen zahlreicher Entwickler ist noch der alte ABAP Editor aktiviert. Der neue ABAP Editor ist zwar in aktuellen SAP-Systemen mittlerweile als Standard voreingestellt. Wer aber in einem SAP-System bereits ein Benutzerkonto hatte, als es den neuen ABAP Editor noch nicht gab, arbeitet in der Regel weiter mit dem alten ABAP Editor, bis er sich explizit für einen Wechsel entscheidet.

Entwickler, die den neuen ABAP Editor nie verwendet haben, vermissen ihn auch nicht unbedingt. Der neue ABAP Editor bringt jedoch viele Vorteile im Alltag, ein Umstieg lohnt sich, versprochen!

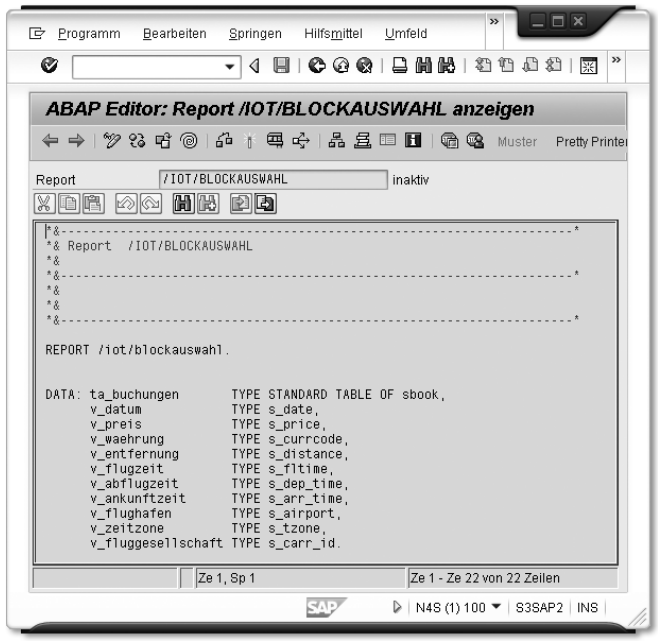
› Und so geht's

Ist der neue ABAP Editor in Ihrem Benutzerprofil noch nicht aktiviert, rufen Sie aus der ABAP Workbench heraus den folgenden Menüeintrag auf:

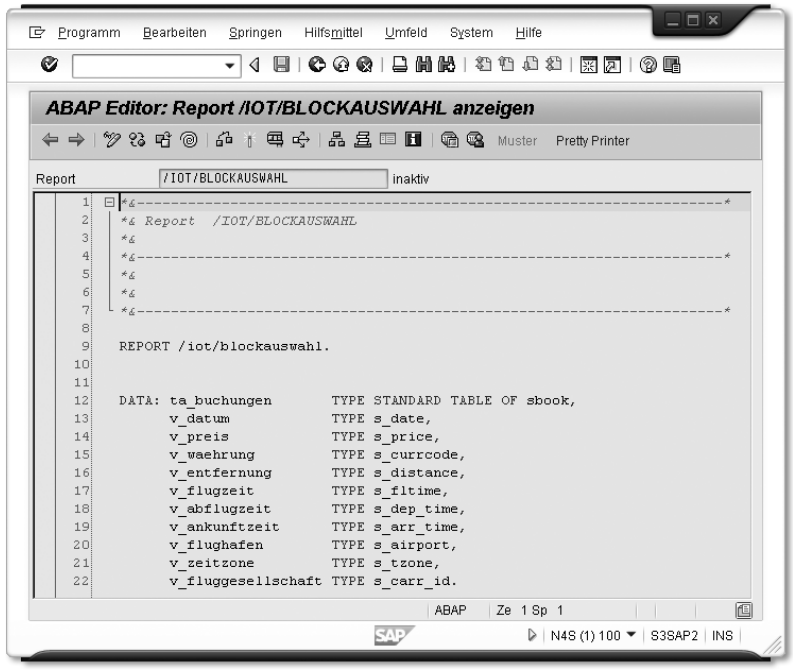
Hilfsmittel ▸ Einstellungen

Wechseln Sie anschließend auf der Registerkarte **ABAP Editor** auf die Unterregisterkarte **Editor**. Hier können Sie über die Einstellung **Frontend Editor (neu)** den neuen ABAP Editor aktivieren.

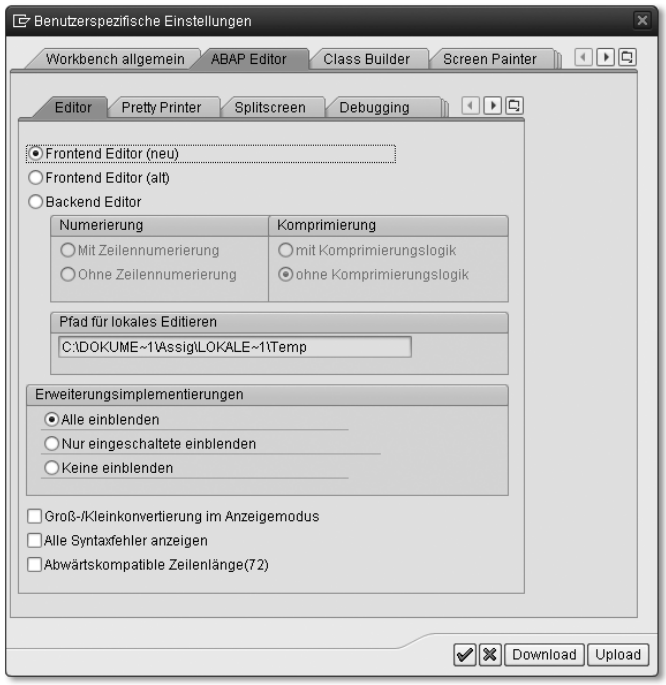
Nach der Aktivierung des neuen ABAP Editors wird Ihnen der ABAP-Quelltext angezeigt – erste Unterschiede fallen schon rein optisch auf.



ABAP-Quelltext im alten ABAP Editor



ABAP-Quelltext im neuen ABAP Editor



Einstellung zum Aktivieren des neuen ABAP Editors

Zu den zusätzlichen Funktionalitäten, die der neue ABAP Editor bereitstellt, gehören unter anderem:

- **Automatische Code-Vervollständigung**: Während Sie die ersten Zeichen einer ABAP-Anweisung eingeben, bietet Ihnen der neue ABAP Editor verschiedene Möglichkeiten an, wie die weiteren Bestandteile der Anweisung aussehen könnten.
- **Code-Vorlagen**: Häufig benötigte Blöcke aus Kommentaren und ABAP-Quelltext werden Ihnen automatisch vorgeschlagen, wenn Sie die ersten Zeichen eines solchen Blocks eingeben. Haben Sie beispielsweise das Wort `CASE` eingegeben, bietet Ihnen der neue ABAP Editor an, einen vollständigen `CASE`-Block inklusive `WHEN`-Anweisungen und `ENDCASE`-Anweisung einzufügen. Sie können auch eigene Code-Vorlagen definieren, die Ihnen auf demselben Weg angeboten werden.
- **Mehrstufige Zwischenablage**: Der neue ABAP Editor merkt sich nicht nur den letzten Wert, den Sie in die Zwischenablage kopiert haben, sondern die letzten zwölf Werte. So können Sie zunächst mehrere Werte kopieren und diese dann einzeln an anderer Stelle wieder einfügen.

- **Farbliche Hervorhebung im Quelltext**: Die verschiedenen Bestandteile des Quelltextes wie Schlüsselwörter, Kommentare oder Operatoren werden in verschiedenen Farben und Schriftstilen dargestellt. Welche Farben und Schriftstile verwendet werden, können Sie auch individuell einstellen.
- **Einfache Syntaxprüfung während der Eingabe**: Schon bevor Sie den Quelltext sichern, aktivieren oder prüfen, führt der neue ABAP Editor bestimmte Syntaxprüfungen durch. So erkennen Sie beispielsweise bereits während der Eingabe an der farblichen Hervorhebung, wenn Sie sich bei einem Schlüsselwort vertippt haben.
- **Hervorhebung von Blöcken**: Ebenfalls optisch hervorgehoben stellt der neue ABAP Editor zusammengehörige Anweisungen dar, beispielsweise welche `ELSE`- und welche `ENDIF`-Anweisung zu einer gerade markierten `IF`-Anweisung gehört.
- **Zusammenklappen von Blöcken**: Sie können zusammenhängende Quelltext-Blöcke zusammenklappen, zum Beispiel Kommentarblöcke, Schleifen oder andere Kontrollstrukturen. So können Sie aktuell für Sie nicht relevante Quelltext-Bestandteile ausblenden und den Blick auf das Wesentliche fokussieren.
- **Zeilennummerierung**: Am linken Rand jeder Quelltext-Zeile wird die Zeilennummer angezeigt. Diese Funktionalität ist hilfreich, um bestimmte Zeilen zu referenzieren, zum Beispiel wenn Sie sich am Telefon mit einem anderen Entwickler über den Quelltext eines Programms unterhalten.
- **Markierung veränderter Zeilen**: Der neue ABAP Editor markiert Zeilen, die Sie seit dem Öffnen des Quelltextes und seit dem letzten Speichern bearbeitet haben. So haben Sie die Möglichkeit, schnell festzustellen, ob Sie versehentlich auch Zeilen bearbeitet haben, die Sie eigentlich nicht bearbeiten wollten.
- **Inkrementelle Suche**: Mit der inkrementellen Suche springt der ABAP Editor bereits zu dem ersten passenden Treffer, während Sie noch eintippen, wonach Sie suchen möchten. So müssen Sie in der Regel nicht einmal ein vollständiges Wort eingeben, nach dem Sie suchen möchten, da Ihnen schon vorher die gesuchte Stelle angezeigt wird.
- **Automatisches Setzen von schließenden Klammern**: Bei der Eingabe einer öffnenden Klammer setzt der neue ABAP Editor automatisch eine entsprechende schließende Klammer. Einige Entwickler stört diese Funktionalität mehr als sie ihnen nützt, da sie daran gewöhnt sind, die schließende Klam-

mer selbst einzugeben. Darum können Sie diese Funktionalität – wie die meisten anderen auch – auf Wunsch abschalten.

- **Frei konfigurierbare Tastaturkürzel**: Die meisten Funktionalitäten des ABAP Editors können auch über Tastaturkürzel aufgerufen werden. Die standardmäßig vorgegebenen Tastaturkürzel können Sie durch Ihre eigenen ersetzen.
- **Lesezeichen**: Sie können interessante Stellen im Quelltext mit Lesezeichen markieren. Mit wenigen Mausklicks oder einem Tastaturkürzel können Sie dann schnell wieder zu einer solchen Stelle springen.
- **Fensterteilungslinie**: Das aktuelle Fenster des ABAP Editors können Sie mit der Fensterteilungslinie horizontal aufteilen. So können Sie sich in einem Modus gleichzeitig mehrere Stellen im Quelltext eines Programms ansehen bzw. diese Stellen bearbeiten.

Bei Bedarf können Sie natürlich jederzeit zum alten ABAP Editor zurückwechseln. Aufgrund von Stabilitätsproblemen, die bei der Einführung des neuen ABAP Editors noch auftraten, ist ein Wechsel in den alten ABAP Editor bei aktuellen Releases aber nicht mehr notwendig; Probleme dieser Art gehören inzwischen der Vergangenheit an.

Tipp 2

Neue Ansichten ab Release 7.0 EHP2

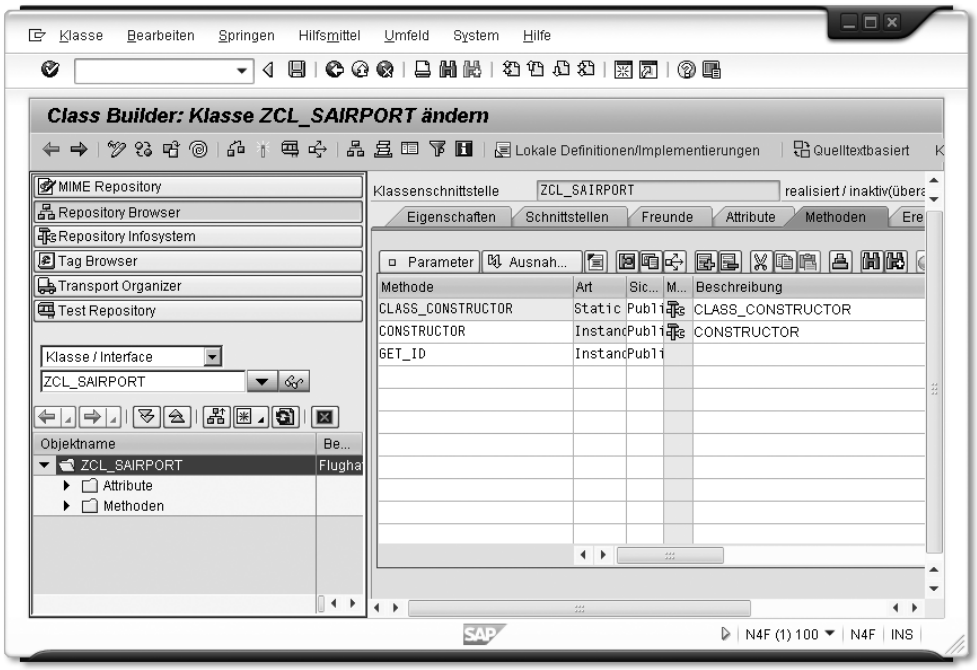
Seit Release 7.0 EHP2 können Sie ABAP-Klassen auch so darstellen, wie Klassen in vielen anderen Entwicklungsumgebungen abgebildet werden: in der quelltextbasierten Form mit dem Quelltext aller Methoden untereinander.

Erinnern Sie sich noch an Ihren ersten Kontakt mit einer Klasse im Class Builder? Wenn Sie Klassen bereits aus Entwicklungsumgebungen für andere Programmiersprachen wie Java oder C++ kannten, waren Sie sicher überrascht über die formularbasierte Darstellung einer Klasse.

Viele andere Entwicklungsumgebungen legen alle Methoden einer Klasse in einer Datei ab und stellen die Methoden im Editor untereinander dar. Die ABAP Workbench dagegen legt die Quelltexte jeweils einer Methode einer globalen Klasse separat in der Datenbank ab, und stellte in der Vergangenheit auf dem Bildschirm immer nur den Quelltext einer Methode dar. Seit Release 7.0 EHP2 können Sie entscheiden, ob Sie die ABAP-typische formularbasierte Darstellung für übersichtlicher halten oder eine Klasse lieber in der alternativen quelltextbasierten Form bearbeiten.

› Und so geht's

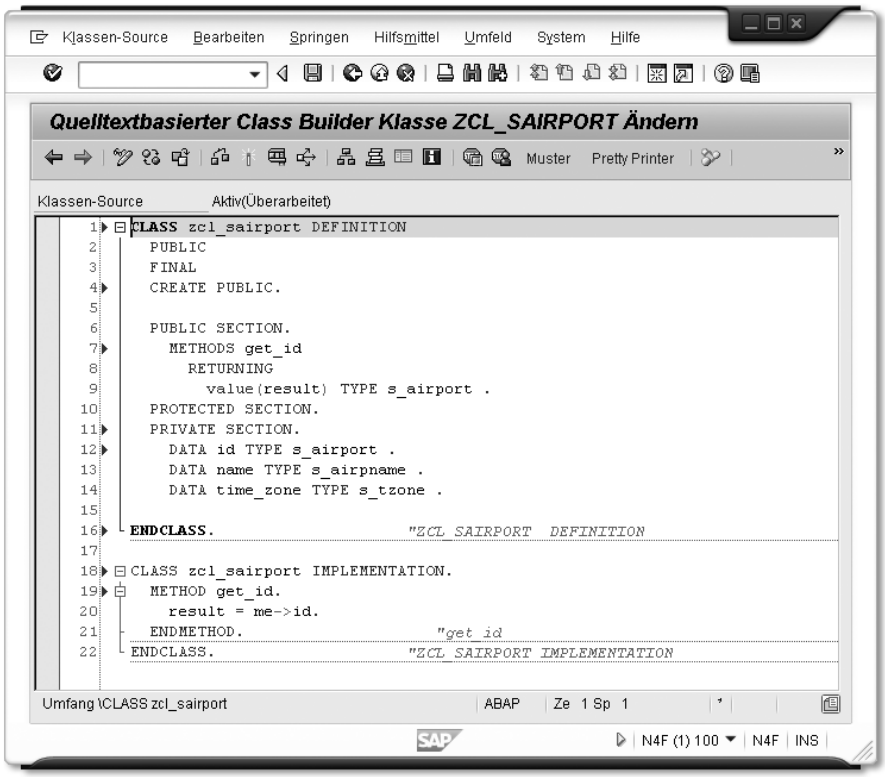
Wenn Sie die Möglichkeit haben, zwischen der formularbasierten und der quelltextbasierten Darstellung zu wechseln, erscheint die Schaltfläche **Quelltextbasiert** mit dem Symbol zum Wechseln der Ansicht ( ) in der Symbolleiste des Class Builders. Klicken Sie auf diese Schaltfläche, um in die quelltextbasierte Ansicht zu wechseln.



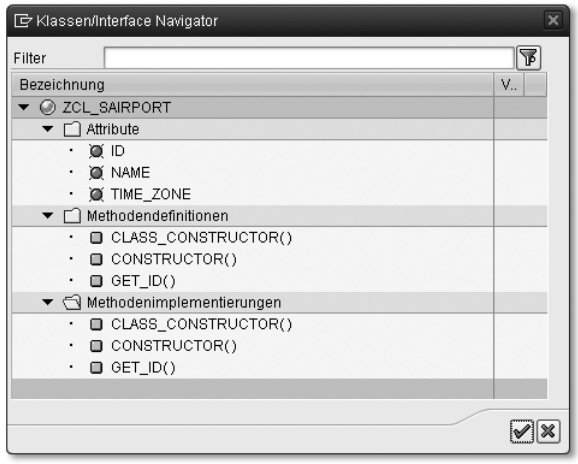
Formularbasierte Ansicht mit der Möglichkeit zum Wechsel in die quelltextbasierte Ansicht

In der quelltextbasierten Ansicht zeigt Ihnen der Class Builder den gesamten Quelltext einer globalen Klasse in einem Editorfenster an. Dazu zählen die Klassendefinition mit der Public Section, der Protected Section und der Private Section sowie alle Methodenimplementierungen. Diese Darstellung entspricht der Darstellung von lokalen Klassen in ABAP, und sie ähnelt der Darstellung von Klassen, wie sie in Entwicklungsumgebungen für andere Programmiersprachen typisch ist.

Im Rahmen der quelltextbasierten Ansicht steht Ihnen außerdem ein neuer Dialog zur Navigation innerhalb der aktuell geöffneten Klasse zur Verfügung: der Klassen-/Interface-Navigator. Sie können den Klassen-/Interface-Navigator in der quelltextbasierten Ansicht über das Navigations-Symbol ( ) in der Symbolleiste des Class Builders aufrufen. Über ihn erhalten Sie schnell einen Überblick über die Attribute und Methoden der geöffneten Klasse. Mit einem Doppelklick auf einen Bestandteil der Klasse im Klassen-/Interface-Navigator navigieren Sie direkt zur entsprechenden Stelle im Quelltext der Klasse. Neben dem Klassen-/Interface-Navigator steht Ihnen auch in der quelltextbasierten Ansicht weiterhin die Navigation über die bekannte Objektliste am linken Bildschirmrand zur Verfügung.



Quelltextbasierte Ansicht einer globalen Klasse



Klassen-/Interface-Navigator

Mit der quelltextbasierten Ansicht versucht SAP nicht, die für globale Klassen bisher ausschließlich verwendete formularbasierte Darstellungsform abzulösen. Die quelltextbasierte Ansicht ist vielmehr als Alternative für Entwickler gedacht, die diese Darstellungsform aus anderen Programmiersprachen bzw. Entwicklungsumgebungen gewohnt sind. Außerdem ist die Verwendung der quelltextbasierten Ansicht sinnvoll, wenn Sie kleine Änderungen an vielen Methoden einer Klasse durchführen möchten. Auch das Ausdrucken oder das Kopieren einer ganzen Klasse in der quelltextbasierten Ansicht ist deutlich einfacher. Die formularbasierte Ansicht, mit der Navigation über Registerkarten und der Darstellung von jeweils einer Methode im Editor, ist dagegen übersichtlicher. Die Definition von Attributen und Methoden erfordert mit den Formularen deutlich weniger Wissen über die Details der ABAP-Syntax als die direkte Definition im Quelltext.

Tipp 3

Den Sperrmechanismus besser verstehen

Gleiches Recht für alle: Denselben Sperrmechanismus, den Sie in Ihren Programmen verwenden, verwendet auch die ABAP Workbench, um zu verhindern, dass mehrere Entwickler gleichzeitig an demselben Entwicklungsobjekt arbeiten.

Die Entwicklung in ABAP ist durch ein pessimistisches Sperrverfahren geschützt. Möchten Sie eine Methode ändern, die gerade von einem Ihrer Kollegen bearbeitet wird, erhalten Sie eine Fehlermeldung mit dem Hinweis auf den Namen des Benutzers, der Ihnen zuvorgekommen ist.

Technisch gelöst ist die Implementierung durch das SAP-Sperrkonzept, mit dem auch Sie in Ihren Programmen konkurrierende Zugriffe verhindern müssen. Das hat zur Folge, dass Sie die Werkzeuge des SAP-Sperrkonzeptes zur Auswertung der gesetzten Sperren auch anwenden können, um herauszufinden, welche Sperren die ABAP Workbench gerade gesetzt hat. Wenn Sie in einem großen Team mit vielen Entwicklern arbeiten, haben Sie so die Möglichkeit, sich schnell einen Überblick darüber zu verschaffen, an welchen Stellen aktuell gearbeitet wird.

› Und so geht's

Möchten Sie sich einen Überblick über die Entwicklungsobjekte verschaffen, die gerade in einem SAP-System bearbeitet werden, schauen Sie in die Liste der Sperreinträge (Transaktion SM12). Für jedes Entwicklungsobjekt, das gerade bearbeitet wird, zeigt Ihnen die Liste der Sperreinträge mindestens einen entsprechenden Sperreintrag an.

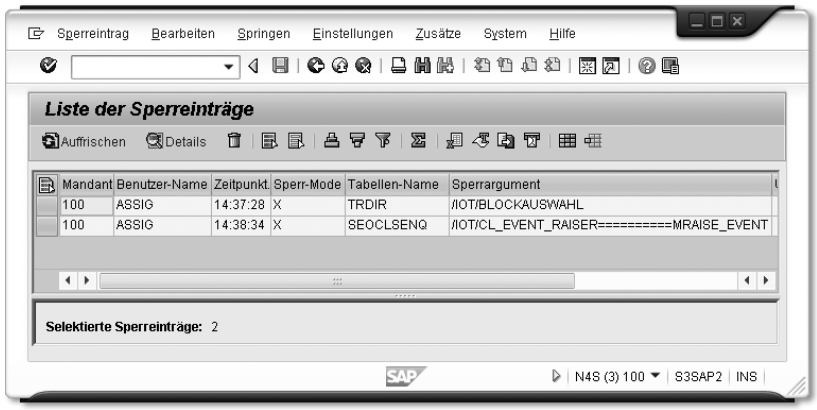
Im Selektionsbild zur Liste der Sperreinträge können Sie auswählen, welche Sperren Sie sehen möchten. Beispielsweise können Sie sich alle Sperren eines Benutzers oder alle Sperren in einem Mandanten ansehen. Sie können

in dem Selektionsbild auch mit den Platzhaltern `?` (ein beliebiges Zeichen) und `*` (beliebig viele beliebige Zeichen) arbeiten.

Selektionsbild zur Liste der Sperreinträge

Da die ABAP Workbench eine Vielzahl verschiedener Tabellennamen für die Sperren verwendet, ist es eher schwierig, über den Tabellennamen die Selektion auf Entwicklungsobjekte eines bestimmten Typs einzuschränken. Schon eher geeignet ist das Sperrargument. Der Name des Entwicklungsobjekts kommt in der Regel im Sperrargument vor. So können Sie sich beispielsweise über den Ausdruck `*/IOT/CL*` alle aktuell bearbeiteten Klassen aus dem Namensraum `/IOT/` anzeigen lassen.

Die folgende Abbildung zeigt ein Beispiel für die Liste der aktuell gesetzten Sperren. In diesem Fall bearbeitet der Benutzer ASSIG das Programm `/IOT/BLOCKAUSWAHL` und die Methode `RAISE_EVENT` der Klasse `/IOT/CL_EVENT_RAISER`.



Liste der aktuell gesetzten Sperren

Je nachdem, was Sie gerade bearbeiten, setzt die ABAP Workbench Sperren mit unterschiedlichen Granularitäten. Bearbeiten Sie beispielsweise im Class Builder die Definition einer Klasse, ist die gesamte Klasse gesperrt. Ein anderer Entwickler kann daher zur gleichen Zeit keine Methode der Klasse bearbeiten. Wenn Sie dagegen die Implementierung einer Methode bearbeiten, ist nur diese eine Methode gesperrt. Andere Methoden derselben Klasse können parallel auch von anderen Entwicklern bearbeitet werden.

Ähnlich verhält sich auch der Dialog zur Bearbeitung von Nachrichtenklassen (Transaktion SE91). Bearbeiten Sie die gesamte Nachrichtenklasse, verhindern Sie, dass ein anderer Entwickler parallel eine andere Nachricht derselben Nachrichtenklasse bearbeiten kann. In der Regel werden Sie jedoch nur eine oder wenige Nachrichten auf einmal bearbeiten. Wechseln Sie dann auch nur für die einzelnen Nachrichten, die Sie ändern möchten, in den Änderungsmodus. So ermöglichen Sie es Ihren Kollegen, parallel Änderungen an anderen Nachrichten derselben Nachrichtenklasse vorzunehmen.

Tipp 4

Die mehrstufige Zwischenablage nutzen

Mit mehr als nur einem Wert in der Zwischenablage sparen Sie sich unnötiges Hin- und Herkopieren.

Möchten Sie mehrere Zeilen Quelltext aus einem Entwicklungsobjekt in ein anderes kopieren, verwenden Sie vermutlich eine der folgenden Vorgehensweisen: Entweder wechseln Sie mehrfach zwischen Fenstern hin und her und kopieren dabei jeweils die Blöcke, die Sie übernehmen möchten. Oder Sie kopieren einen zusammenhängenden Block, der auch Zeilen enthält, die Sie eigentlich nicht benötigen, und löschen dann nachträglich wieder die Zeilen, die Sie eigentlich gar nicht benötigen.

Die elegante Alternative, die die ABAP Workbench für Aufgaben dieser Art anbietet, ist nur wenigen Entwicklern bekannt.

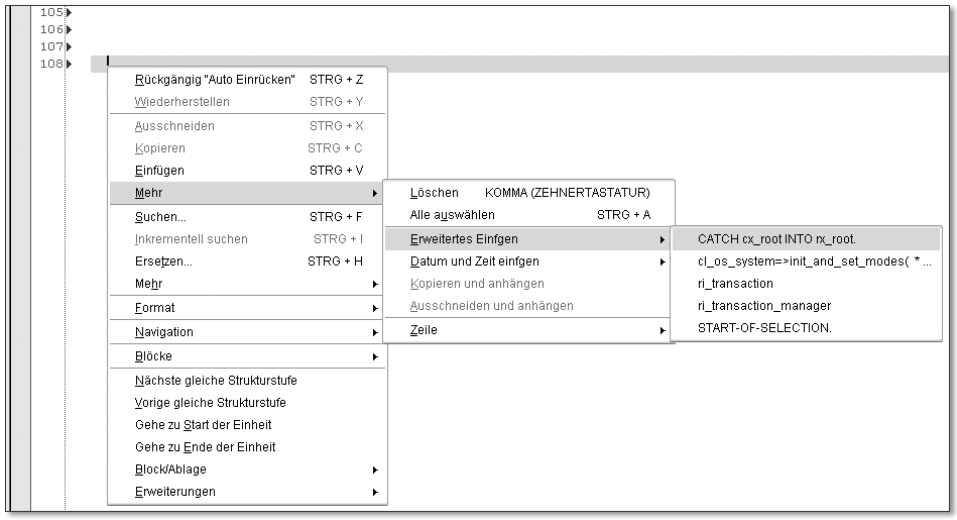
› Und so geht's

Der neue ABAP Editor der ABAP Workbench merkt sich automatisch nicht nur den letzten Wert, sondern die letzten zwölf Werte, die Sie aus dem Editor in die Zwischenablage kopiert haben.

Um diese mehrstufige Zwischenablage zu nutzen, kopieren Sie den markierten Text einfach auf dem gewohnten Weg in die Zwischenablage, das heißt entweder über das Tastaturkürzel Strg + C oder über den Eintrag **Kopieren** aus dem Kontextmenü. Um einen der letzten zwölf Werte, den Sie in die Zwischenablage kopiert haben, wieder einzufügen, haben Sie dann ebenfalls zwei Möglichkeiten:

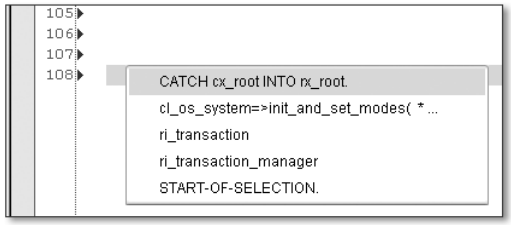
- Sie können sich über das Kontextmenü unter dem Eintrag **Mehr ▸ Erweitertes Einfügen** die letzten Werte anzeigen lassen, die Sie in die Zwischen-

ablage kopiert haben, und den gewünschten Wert auswählen, um ihn einzufügen.



Mehrstufige Zwischenablage, aufgerufen über das Kontextmenü

- Über die Tastatur erreichen Sie dieselbe Liste auch direkt mit dem Kürzel Strg + ⇧ + V. Mit den Cursor-Tasten (↑ bzw. ↓) können Sie den gewünschten Eintrag markieren und den Wert mit der ↵-Taste einfügen.



Mehrstufige Zwischenablage, aufgerufen über das Tastaturkürzel

Unabhängig davon, für welche der beiden Möglichkeiten Sie sich entscheiden, müssen Sie zuvor mindestens zwei Werte aus dem ABAP Editor in die Zwischenablage kopiert haben, damit Ihnen die Liste der letzten Werte angezeigt wird.

Um die mehrstufige Zwischenablage zu nutzen, ist keine besondere Aktion beim Kopieren notwendig – sie ist automatisch aktiv. Daher ist es auch mög-

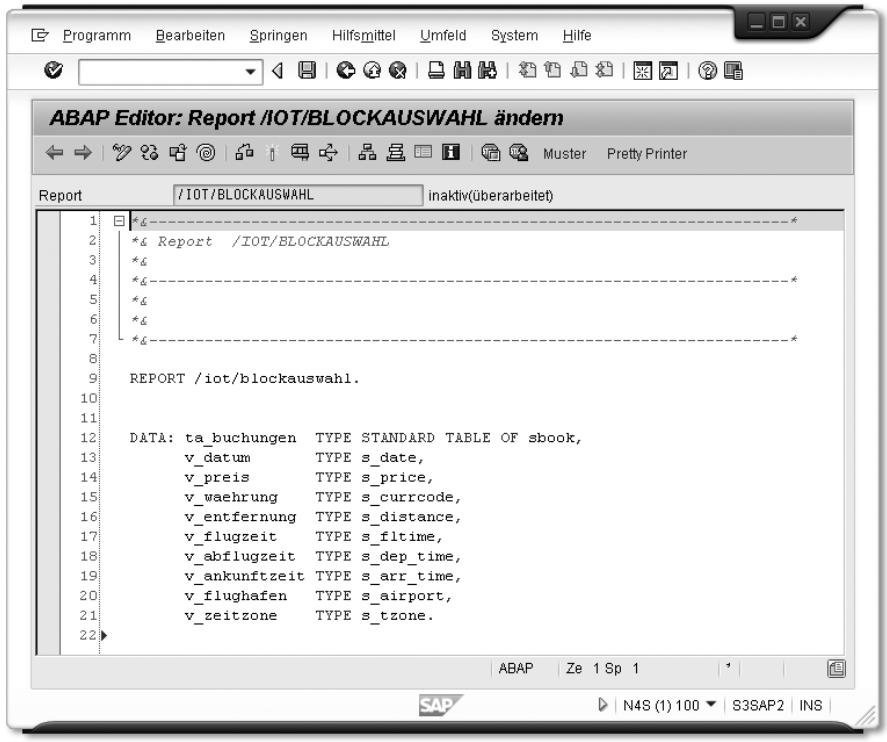
lich, Werte einzufügen, die Sie zuvor in die Zwischenablage kopiert haben, ohne dass Sie sich beim Kopieren bereits bewusst waren, dass Sie den Wert später noch einmal benötigen. Haben Sie beispielsweise den Namen einer Variablen vor kurzem in die Zwischenablage kopiert, die Zwischenablage aber zwischenzeitlich für andere Dinge verwendet, können Sie den Variablennamen später erneut einfügen, ohne ihn zuvor wieder in die Zwischenablage kopieren zu müssen.

Tipp 5

Quelltext spaltenweise markieren

Wenn Sie die Alt-Taste beim Ausrichten von Quelltext für sich entdeckt haben, wird sie Sie nicht mehr loslassen – nicht umgekehrt.

Viele Entwickler legen Wert auf eine optisch ansprechende Formatierung des Quelltextes. Dazu kann beispielsweise gehören, dass bei mehreren aufeinanderfolgenden Variablendefinitionen das Schlüsselwort `TYPE` immer in derselben Spalte steht.



Ausgangssituation mit zehn Variablen

Was geschieht aber, wenn bereits zehn Variablen definiert und ausgerichtet sind, und nun eine elfte Variable hinzugefügt werden soll, deren Name länger ist als die Namen der bereits vorhandenen Variablen? Es ist lästig, alle zehn Zeilen einzeln zu bearbeiten, um dort jeweils die benötigten weiteren Leerzeichen einzufügen.

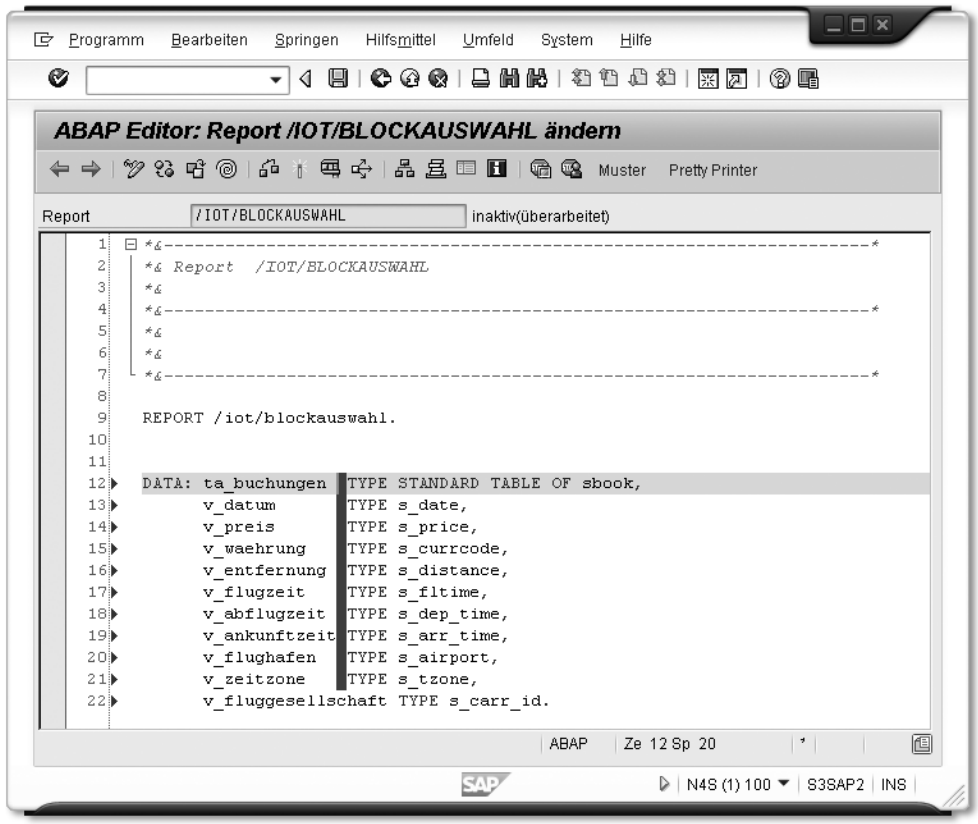
› Und so geht's

Mit gedrückter Alt-Taste können Sie im ABAP Editor mit der sogenannten *Blockauswahl* den Quelltext spaltenweise markieren. So ist es zum Beispiel möglich, nur die Variablennamen aus mehreren Zeilen mit Variablendefinitionen zu kopieren, in mehreren Zeilen gleichzeitig etwas einzufügen oder aus mehreren Zeilen gleichzeitig bestimmte Teile zu löschen.

Um die Blockauswahl zu verwenden, halten Sie die Alt-Taste gedrückt und markieren einen Block aus mehreren Zeilen und Spalten mit der Maus. Auch ohne die Maus können Sie die Blockauswahl verwenden, indem Sie die Alt-Taste und die ⇧-Taste gedrückt halten und den Block mit den Cursor-Tasten markieren. Den markierten Block können Sie wie gewohnt über Strg + C kopieren. Dabei werden nur die markierten Zeichen in die Zwischenablage kopiert, nicht aber der Teil der Zeilen, den Sie mit der Blockauswahl nicht markiert haben.

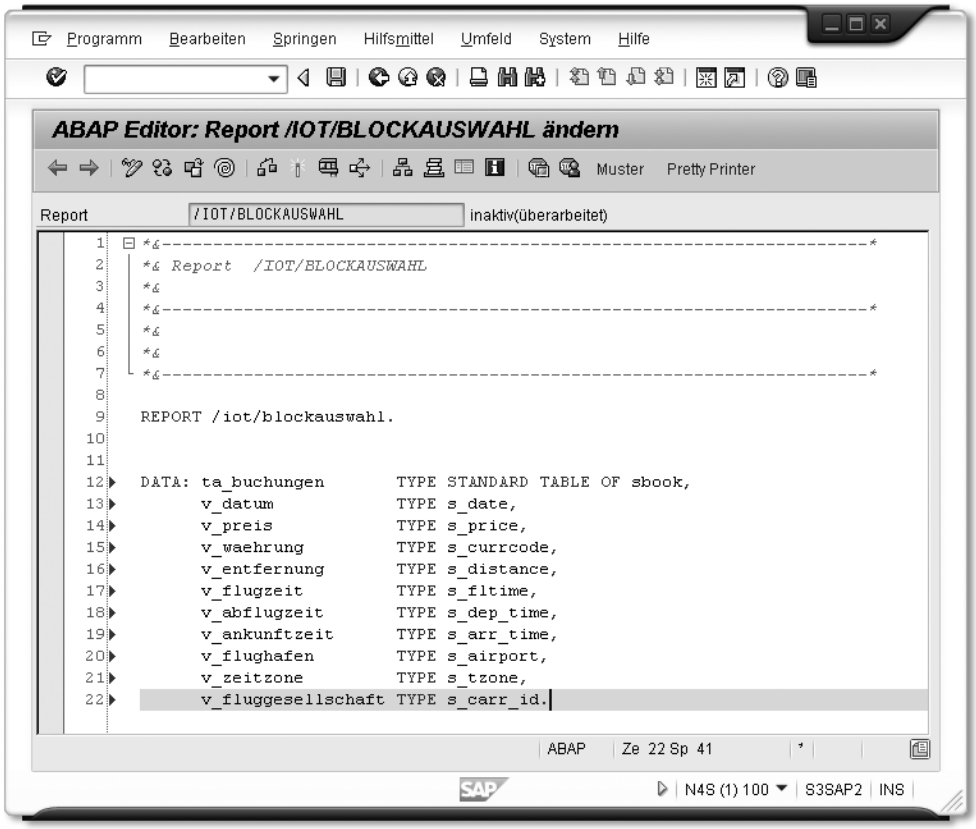
Wenn Sie einen Block auswählen und dann den Inhalt der Zwischenablage einfügen, ersetzt der ABAP Editor die markierten Zeichen durch den Inhalt der Zwischenablage. Mit der Entf-Taste oder mit der ← -Taste können Sie die per Blockauswahl markierten Zeichen aus mehreren Zeilen löschen. Ebenfalls in Kombination mit der Blockauswahl nutzbar ist die Funktionalität, die Einrückung von Quelltext über die ⇆-Taste zu vergrößern oder zu verkleinern. In Kombination mit der Blockauswahl ändern Sie jedoch nicht die Einrückung der gesamten Zeile, sondern nur die Einrückung des markierten Blocks inklusive aller Zeichen, die sich rechts vom markierten Block befinden.

Folgende Abbildung zeigt die eingangs skizzierte Problemstellung. Das Schlüsselwort `TYPE` soll wieder in allen Zeilen untereinander stehen. Um dies zu erreichen, wurde bereits ein Block markiert. Der markierte Block und alles, was rechts davon steht, sollen weiter nach rechts verschoben werden.



Variablendefinitionen vor der Anpassung der Formatierung

Mit der ⇥-Taste können Sie die rechte Hälfte der Variablendefinitionen nun jeweils um zwei Leerzeichen nach rechts verschieben, bis sie an der gewünschten Position angekommen sind. Mit der ⇧-Taste und der ⇥-Taste können Sie jeweils zwei Leerzeichen entfernen und damit den rechten Teil der Variablendefinitionen wieder nach links verschieben. Haben Sie genau eine Spalte markiert, bevor Sie mit dem Verändern der Einrückung beginnen, können Sie zum Abschluss noch eine Feinkorrektur vornehmen, indem Sie den markierten Block und damit aus jeder Zeile ein einzelnes Leerzeichen mit der Entf-Taste löschen. Bei der Formatierung von Quelltext auf die beschriebene Art ist es darum sinnvoll, mit der Blockauswahl immer genau eine Spalte zu markieren, die nur Leerzeichen enthält, und dann die Einrückung anzupassen.



Variablendefinitionen nach der Anpassung der Formatierung

Index

%SurroundedText% 93
*$&$MUSTER 79
/$sync 161
_EDITOR_EXIT 80

A

ABAP Debugger 100, 216, 218, 240, 241, 243, 246, 249, 255, 256, 260, 264, 267, 271, 274, 278, 282, 361
ABAP Dictionary 119, 165, 168, 173, 176, 179, 183, 187
ABAP Editor 18, 30, 34, 37, 43, 48, 75, 76, 78, 82, 84, 89, 91, 116, 221, 288
ABAP Unit 224, 328
  lokale Testklasse 299
  Testklasse 299
ABAP-Paketkonzept 133, 139
ABAP-Stack 244
Abgleichkennzeichen 181
Abhängigkeit zwischen Paketen 139, 142
Ablaufzeit 153
Aggregation 346
aktive Version, Rückkehr 131
aktivierbarer Checkpoint 235
Aktivieren und Datenbank anpassen 178
alle Syntaxfehler anzeigen 55
alte Version wiederherstellen 130
Analyse
  lexikalische 49
  semantische 50
  syntaktische 49
Änderer, letzter 297
Änderung, untersagte 110
Änderungsgrund 182
Anlegen eines Transportauftrags mit Vorlage 374
Ansicht, formularbasierte 199
Anweisung
  problematische 317
  überflüssige 317
  veraltete 317
Anweisungsabdeckung 367
Anwendungskomponente 134
Anwendungslogik 183
Anzeigeformat 214
ASSERT 235
Assertion 235
Aufgabe umhängen 377
Aufgabentyp ändern 383
Aufruf
  Konstruktor 75
  remotefähiger Funktionsbaustein 81, 339
Aufrufhierarchie 350
Auftrag
  Auftragsstatus 371
  Auftragstyp 371, 387
  schützen 377
  verschmelzen 377
Aufzeichnung eines SAP-GUI-Kommandos 231
Aufzeichnungsroutine 181
Ausgabelänge 119, 157
Auslösen eines Ereignisses 61
Auslösestelle zeigen 280
Ausnahme, klassenbasierte 278, 283
Ausnahmeklasse 299
Ausnahmeobjekt immer erzeugen 278
Ausnahmeobjekt, letztes 279
automatische Code-Vervollständigung 20
automatische Fehlerkorrektur 53
automatische Formatierung nach jedem Einfügen 43

B

BAPI Browser 81
Batch-Workprozess 251
Bedingungseingabe, freie 268
Befehlsfeld 161, 243, 246, 250, 253, 261, 361, 390
Benennung des Benutzer-Desktops 277
Benutzer-Desktop benennen 277
Benutzeroberfläche generieren 205
Benutzerprofil 18
Berechtigungsgruppe 180
Beschreibung der Ausnahme 283
beteiligte Datenbanktabelle 342
bevorzugter Parameter 107
Bezugstyp 120

BI_PERSISTENT 122
Block hervorheben 21
Block zusammenklappen 21
Blockauswahl 34
Breakpoint 235, 249, 251, 257, 260, 261, 263, 264, 267, 270, 277, 283
Bedingung ändern 266
Bedingung anlegen 265
Bedingung löschen 266
externer 249, 256
Breitenangabe, relative 208
Bruttozeit 349
BUFFER 80
Business Configuration Set 181
Business Server Pages 249

C

CALL FUNCTION 315
CALL METHOD 76
Checkpoint, aktivierbarer 235
Checkpoint-Gruppe 235
CHEGUI-Kommando 232
CL_AUNIT_ASSERT 225
CL_CI_TEST_ROOT 334
CL_CI_TEST_SCAN 334
CL_OO_INCLUDE_NAMING 114
Class Builder 23, 99, 115, 117, 122, 213, 225
Classic Debugger 242
Code-Hinweis 84, 89, 91
Code-Vervollständigung 84, 86, 89
automatische 20
Code-Vorlage 20, 88, 91
Code-Vorschlag 45, 89
Compiler 49
Container-Formular erzeugen 207
Context 196, 206
Context-Editor 196
Coverage Analyzer 326, 364
Customizing 276
Customizing-Auftrag 370, 376
an Workbench-Auftrag hängen 376
Customizing-Tabelle 179

D

Darstellung
formularbasierte 23
quelltextbasierte 23
Data Browser 105
Datenbank
anpassen 178
Eigenschaft 173
Utility 177
View 165
Datenbankobjekt 177
Datenbanktabelle 119, 165, 168, 173, 176, 179, 183, 186, 190
beteiligte 342
Datendeklaration löschen 63
Datenschutz 152
Datentyp 119
Debugger-Layer konfigurieren 271
Debugger-Profil 272
Einstellungen ändern 244, 276, 278
Debugger-Sitzung 276, 277
Debugging 241, 243, 248, 249, 253, 256, 267, 271, 274
aktivieren 254
deaktivieren 255
einschalten 217
Definition eines Ereignisses 57
Desktop 274
Dialogstruktur 191
Dialog-Workprozess 251
direkte Typeingabe 121
Dokumentation 39
Domäne mit Festwert 164
Drag & Drop 66, 74, 77, 248
dynamisch Quelltext generieren 78
dynamischer Test 321
dynamisches Muster 79
Dynpro 205
Konsistenz 315
Stack 244

E

eCATT 228
Editiersperre 110, 112
Editor, methodenbasierter 199
Eigenschaft, semantische 170
Einfügen, erweitertes 30
Eingabehilfe 168
Eingabeprüfung 168
Eingabevorschlag 151

Einrückung 40, 43
Quelltext 34
Einstellung 18, 43, 76, 89, 91, 129, 199, 242, 244
einstufiger Pflegedialog 181
Eintrag
aus der Historie löschen 152
Maximalanzahl 153
Element, sichtbares 137
elementare Suchhilfe 164
End2End 357
Endlosschleife 249
Enqueue-Trace 341
Entwicklungsdatenbank 127
Entwicklungselement, nach
außen sichtbares 138
Entwicklungsklasse 106
Entwicklungsobjekt
gelöschtes 103
wiederherstellen 130
Entwicklungspaket 133
Entwicklungssystem, neues 386
Ereignis 57
auslösen 61
definieren 57
objektorientiertes 57
statisches 58
Ereignisbehandler 57
registrieren 61
Ereignisbehandlermethode 60
Ereignisparameter 58
Ergebnis der Inspektion 332
Ersetzen generischer Parameter 115
erweiterte Prüfung 314, 328
erweitertes Einfügen 30
EU_INIT 301
EU_PUT 301
EU_REORG 301
Exception-Kette 279
exklusive Sperre 220
EXP 166
Export-Parameter 166
Extended Computer-Aided Test Tool 228
externer Breakpoint 249, 256

F

F4-Hilfe 164
farbliche Hervorhebung 21
Fehleranalyse 284
Fehlerkorrektur, automatische 53
Fehlermeldung 49
Fehlerschwere 140
Fehlersuche 278, 282
Feldabhängigkeit 192
Feldeigenschaft 316
Fenster, modales 246
Fensterteilungslinie 22
FOR TESTING 225
Format, internes 215
Formatierung 43
automatische 43
von Quelltext 33
formularbasierte Ansicht 199
formularbasierte Darstellung 23
Formularfeld 154
freie Bedingungseingabe 268
Fremdschlüssel 164, 168
Fremdschlüsseltabelle 168
Frontend-Editor (neu) 18
Function Builder 217, 219
funktionale Kurzschreibweise 107
funktionale Schreibweise 75, 76
Funktionsbaustein, remotefähiger 81, 339

G

gelöschtes Entwicklungsobjekt 103
wiederherstellen 130
Generierung eines Tabellenpflege-Dialogs 190
generischen Parameter ersetzen 115
Granularität von Sperren 29
GridLayout 209
Groß- und Kleinschreibung 37
Großschreibung 37

H

Hauptpaket 133
kein Hauptpaket 135
Hervorhebung
eines Blocks 21
farbliche 21
Hintergrundanwendung 219, 249
Hintergrundjob 361

Hintergrundverarbeitung 237
Historie 151
Beschreibungstext 153
Eintrag löschen 152
löschen 152
Hitliste 349

I

ICF 253, 256
IF_WORKFLOW 122
IMP 166
Implementierung 199
Import-Parameter 166
Include 173
Include-Name 115
Index des Verwendungsnachweises 301
individuelle Aufzeichnungsroutine 181
Initialisierungsflag 173
Initialwert 173
inkrementelle Suche 21, 288, 289
Inspektion 319, 322, 328
als Job einplanen 326
zyklisch einplanen 327
Inspektionsergebnis 332
Instanzereignis 58
Integrität, referenzielle 168
Interface 303
Internationalisierung 317, 334
interner Name 114
internes Format 215
Internet Communication Framework 253, 256
Inversion of Control 57

J

Job debuggen 250
Jobübersicht 249

K

Kardinalität 170
kein Hauptpaket 135
keine Aufzeichnungsroutine 181
Klasse suchen 297
Klasse, persistente 299
Klassen-/Interface-Navigator 24
klassenbasierte Ausnahme 278, 283
Klassenbibliothek 307
klassischer ABAP Debugger 240
Kleinschreibung 38
Kommandofeld → Befehlsfeld
Kommentar 40, 78, 84, 149
Kommentarzeile 40
Konsistenz 168
Konvertierungsroutine 119
Kopfkommentar 41
Kopie transportieren 380, 389
Kopieren eines Transportauftrags 374
Korrekturvorschlag 53
Kundennamensbereich 79, 191
Kundenprogramm, produktives 98
Kurzdump 282
Kurzschreibweise 76, 101
funktionale 107

L

Langtext des Laufzeitfehlers 283
Laufzeit 356, 357
messen 222
Laufzeitanalyse 326, 345, 352
aktivieren 353
deaktivieren 353
Laufzeitfehler 282
Laufzeitverhalten 213, 221, 339, 345
Layer 272
Layer-Aware Debugging 271
Layer-Step 272
Layout 205, 208, 258, 274
Layoutdaten 209
Lesezeichen 22
letzter Änderer 297
letztes Ausnahmeobjekt 279
lexikalische Analyse 49
Liste
gesetzte Sperren 219
Laufzeitfehler 282
Sperreinträge 27
Literal 94
LOG-POINT 235
Logpoint 235
lokale ABAP-Unit-Testklasse 299
lokale Testklasse 226
Löschen

Eintrag aus der Historie 152
ungenutzte Datendeklaration 63
Löschungskennzeichen 106
Lower case 38
LPos 166

M

Markieren
spaltenweises 34
veränderte Zeilen 21
Massenänderung von Objektkatalogeinträgen 390
MatrixLayout 209
maximale Anzahl von Einträgen 153
mehrstufige Zwischenablage 20, 30
Memory Inspector 358, 360
MESSAGE 101
MESSAGE-ID 102
Messgenauigkeit 355
Methode verschieben 65
methodenbasierter Editor 199
Metrik 321
Microsoft Word 39
modales Fenster 246
Modifikation 317
Modularisierung 68
Modultest 224, 328
Muster 75, 76, 93
bearbeiten 79
dynamisches 79
selbst definiertes 78
Musterfunktion 74, 76, 78, 81

N

nach außen sichtbares Entwicklungselement 138
Nachricht 101
Nachrichtenklasse 29, 101
Nachrichtennummer 101
Nachrichtentyp 101
nächste Objektmenge 272
Name, interner 114
Namensraum 28
generierter 191
reservierter 79
Navigationsfenster 149
Navigationsstapel 149
Nettozeit 349
neuer ABAP Debugger 240
neues Entwicklungssystem 386

O

Object Navigator 82, 134, 139, 142
Object Services 123
Objekt
Selektion 325
sperren 380
Zuordnung 325
Objektkatalog 103, 112, 132, 389
Eintrag 103, 379
Eintrag ändern 389
Objektkollektor 326
Objektliste 24, 74, 77, 160, 297, 380
eines Transportauftrags 375
neu aufbauen 161
sortieren und verdichten 377
Objektmenge 272, 319, 322, 330
nächste 272
Objektname 105
objektorientiertes Ereignis 57
Objektstruktur 191
Objekttyp 105
offener Transportauftrag 371
optimistische Sperre 220
optionaler Parameter 107
Organisation von Transportaufträgen 373
Originalsprache 103
Originalsystem 103, 375, 386, 390

P

Paket 103, 133, 136
kein Hauptpaket 135
Pakethierarchie 133, 142
Paketprüfung 136, 142
als Client 143
als Server 143
Fehler 142
Paketschnittstelle 136, 139, 142
Paketstruktur 133

PAKPARAM 142
Parameter
bevorzugter 107
Export-Parameter 166
Import-Parameter 166
präferierter 108
Parser 49
Performance-Analyse 339
Performance-Beispiel 221
Performance-Monitor 356
Performance-Prüfung 320
Performance-Trace 339, 342
persistente Klasse 299
Persistenzabbildung 123
pessimistisches Sperrverfahren 27
Pflege eines Service 253, 352
Pflegedialog 179, 181, 183, 186, 190
Pflegemerkmal 188
Pflege-View 179, 183, 186
Platzhalter 28, 92, 101
POPUP_GET_VALUES 80
Pop-up-Fenster 246
Post-Mortem-Analyse 286
präferierter Parameter 108
Pretty Printer 40, 43, 46
Primärschlüssel 173
Primärtabelle 187
problematische Anweisung 317
produktives Kundenprogramm 98
produktives SAP-Standard-programm 98
Programm-ID 105
Programmierkonvention 321
Programmierrichtlinie 316
Programmierung, robuste 321
Programmobjekt 366
Programmstatus 98, 243
Programmtext anlegen 95
Proxy-Prüfung 321
Prozedurabdeckung 367
Prozessübersicht 251
Prüftabelle 168
Prüfung, erweiterte 314, 328
Prüfvariante 319, 322, 328

Q

Qualitätsprüfung 224
Quelltext
analysieren 50
durchsuchen 288
dynamisch generieren 78
extrahieren 69
Segmente extrahieren 68
strukturieren 41
vergleichen 310
Wildwuchs 78
quelltextbasierte Darstellung 23

R

RAISE EVENT 61
Refactoring 65, 69, 300
Refactoring-Assistent 65
referenzielle Integrität 168
Registrieren eines Ereignis-behandlers 61
Reihenfolge der Felder einer Datenbanktabelle 176
relative Breitenangabe 208
Release 6.40 119
Release 7.0 EHP1 145, 196, 356
Release 7.0 EHP2 23, 62, 87, 228, 238, 261, 264, 269, 271, 279, 351
Remote Function Call 81, 249, 339
remotefähiger Funktionsbaustein 81
Aufruf 81, 339
Rendering 359
Repository Browser 133
Repository-Infosystem 297, 306
reservierter Namensraum 79, 191
RFC 81, 249, 339
RFC-Trace 341
robuste Programmierung 321
ROOTUIELEMENTCONTAINER 209
RSWSOURCET 80
Run Time Type Services 335

S

SAP Business Workflow 122
SAP Code Inspector 291, 318, 322, 328, 333
eigenerTest 333
Prüfung 114
Test 333
SAP GUI 85, 86, 151

Kommando aufzeichnen 231
Verknüpfung 246
SAP NetWeaver Business Client 228
SAP-Hinweis 85, 142, 301
SAP-Sperrkonzept 27, 110, 217, 339, 378
SAP-Standardprogramm, produktives 98
Scanner 49
Schlüssel 173
Schlüsselattribut 122
Schlüsselfeld 170, 173
einer Texttabelle 170
Schreibweise, funktionale 75, 76, 107
Schutz aufheben 377
selbst definiertes Muster 78
Selektionsmethode 165
semantische Analyse 50
semantische Eigenschaft 170
SENDER 58
Service pflegen 253, 352
Session-Breakpoint 249, 257
SET HANDLER 61
SETUP 225
Sicherheitsprüfung 320
sichtbares Element 137
SLAD 271
SLAD-Cockpit 272
Software Layer-Aware Debugging 271
Softwarekomponente 134
Spaltenbreite 157
Spaltenüberschrift 119
spaltenweises Markieren 34
Speicherabzug 358, 360
erzeugen 361
vergleichen 361
Speicheranalyse 361
Speicherbedarf 360
Speicherverbrauch 356, 360
Sperr-/Importstatus 380
Sperranforderung 339
Sperrargument 28, 219
Sperrbaustein 217, 218
Sperre
exklusive 220
in Transportauftrag 104, 378, 385
optimistische 220
testen 218
Sperreintrag 27
Sperrkonflikt 218
simulieren 217
Sperrmechanismus 27
Sperrmodus 219
Sperrübersicht 379
Sperrverfahren, pessimistisches 27
Splitscreen-Editor 310
Splitter Control 276
SPos 166
SQL-Trace 340, 342
Stabilitätsproblem 22
Stack-Trace 341
Standardauftrag 370
rücksetzen 372
setzen 372
Standardaufzeichnungsroutine 181
Standardnachrichtenklasse 101
START_UP 277
statisches Ereignis 58
Statistik 321
String-Literal 94
Strukturerweiterung 317
Strukturierung des Quelltextes 41
Strukturpaket 133
stufenweises Suchen 289
Suche
ABAP-Anweisungsmuster 294
ABAP-Token 294
inkrementelle 21, 288, 289
Klasse 297
stufenweise 289
Suchfernglas 289
Suchfunktion 321
Suchfunktionalität 288
Suchhilfe 119, 164, 168
elementare 164
Suchhilfe-Exit 167
Suchhilfeparameter 165
sy-batch 251
syntaktische Analyse 49
Syntaxfehler, alle anzeigen 55
Syntaxprüfung 21, 46, 49, 53, 314, 320, 328
Systembefehl 247
System-Debugging 243, 254
Systeminformation 359
Systemprogramm 99, 243, 254

T

Tabellenpflegedialog generieren 190
Tabellenpflegegenerator 180, 183, 188

Tabellenpuffer 339
Tabellenpuffer-Trace 341, 342
Tabellensichtpflege 181
Tabellenumsetzung 177
TADIR 105
Tastaturkürzel 22, 30, 37, 40
Tastenkombination 85, 356
TEARDOWN 225
Template-Galerie 205
temporäre Version 125
Test 329
  dynamischer 321
Testklasse 225
  lokale 226
Testprogramm 99, 221
Testskript 228
Testumgebung 115, 212, 219
Testverwaltung 338
TEXT_ID 280
Textfeldliteral 94
Textsymbol 94
Texttabelle 170
Tippfehler 53, 152, 289
Token 49
Trace
  anzeigen 343
  ausschalten 343
  einschalten 343
  Liste 344
TransparentContainer 209
Transport
  von Kopien 380, 389
  Ziel 387
Transport Management System 125
Transport Organizer 371, 373, 380, 383, 387, 389
Transportauftrag 124, 179, 326, 370, 373, 378, 382
  anlegen 374
  Eintrag verschieben 382
  kopieren 374
  offener 371
  organisieren 373
  schützen 377
  Sperre 104, 378, 385
  verschmelzen 377
Transportschicht 135, 388
Typ
  Interface 119
  Klasse 119
Typeingabe, direkte 121
TYPES 121
Typgruppe 119
Typisierungsart 120

U

überflüssige Anweisung 317
Umbenennen von Variablen 62
Umfeld 307
Umfeldermittlung 306
Umsetzung erzwingen 178
Umzug 387
  Entwicklungsobjekt 386
  komplettes Paket 387
  Objekt mit Paketwechsel 387
  Objekt ohne Paketwechsel 387
Umzugstransport 386
untersagte Änderung 110
Upper case 37

V

Variable umbenennen 62
Variablendefinition 35
  bereinigen 62
veraltete Anweisung 317
veränderte Zeile markieren 21
Verantwortlicher 104, 375, 390
Vererbungshierarchie 303
Vergleich über Systeme 311
VERS_AT_IMP 125
Verschieben
  Eintrag in Transportauftrag 382
  Methode 65
Version 124
  erzeugen 125
  temporäre 125
  vergleichen 126
  wiederherstellen 130
  ziehen 125
Versionsdatenbank 127, 130
Versionshistorie 124
Versionsvergleich 128
Versionsverwaltung 124, 126, 130
Vervollständigung aktivieren 90
Verwaltung von Tests 338
Verwendungserklärung 136, 139, 142

Verwendungsnachweis 149, 300, 303, 306
Index 301
View-Cluster 190
Pflege 193
View-Feld 187
View-Typ 187
Volltextsuche 291
Vorlage 93
Vorschlag 84
Eingabe 151
Vorwärtsnavigation 148
VSEOCLASS 299

W

Watchpoint 263, 267, 270
anlegen 267
auf Objektattribut 269
WDY_BAPI_BROWSER 82
WDY_WB_GET_COMPONENT_
CLASS 117
WDY_WB_GET_CONTROLLER_
INCLUDE 117
Web Application Builder 82
Web Dynpro 199, 208
Anwendung 249, 253, 256, 352, 356, 361
Code-Wizard 202, 205
Component 117, 256
Controller 196, 199, 202
Web Dynpro Explorer 196, 199, 202, 205
Webservice 249, 253
Wechsel
zum klassischen ABAP Debugger 241
zum neuen ABAP Debugger 241
Werkzeug zum Ändern eines Objektkatalogeintrags 389
Wertehilfe 154, 171
Wiederverwendbarkeit 68
Workbench-Auftrag 370, 376
Workflow 122

Z

Zeilenendekommentar 40
Zeilennummerierung 21
Zeitpunkt, Pflegedialog 183
Ziel des Transports 387
Zielpaket 388
Zieltransportschicht 388
Zurück zur aktiven Version 131
Zurückholen 130
Zusammenklappen von Blöcken 21
zweistufiger Pflegedialog 181
Zwischenablage 30, 34, 43
mehrstufige 20, 30